

LibreOffice

Community

LibreOffice Documentation Team

Příručka aplikace Base



7.3

LibreOffice je registrovaná ochranná známka The Document Foundation
Další informace jsou dostupné na stránce cs.libreoffice.org

Autorská práva

Tento dokument je chráněn autorskými právy © 2022 týmem pro dokumentaci LibreOffice. Příspěvatelé jsou uvedeni níže. Dokument lze šířit nebo upravovat za podmínek licence GNU General Public License (<https://www.gnu.org/licenses/gpl.html>), verze 3 nebo novější, nebo the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0/>), verze 4.0 nebo novější.

Všechny ochranné známky uvedené v této příručce patří jejich vlastníkům.

Příspěvatelé

Pro toto vydání

Steve Fanning

Vasudev Narayanan

Robert Großkopf

Pro předchozí vydání

Martin Fox

Robert Großkopf

Pulkit Krishna

Jost Lange

Dan Lewis

Hazel Russman

Jochen Schiffers

Jean Hollis Weber

Zpětná vazba

Jakékoli připomínky nebo návrhy k tomuto dokumentu prosím směřujte do fóra dokumentačního týmu na adrese <https://community.documentfoundation.org/c/documentation/loguides/> (registrace je nutná) nebo pošlete e-mail na adresu: loguides@community.documentfoundation.org.



Poznámka

Vše, co napíšete do fóra, včetně vaší e-mailové adresy a dalších osobních údajů, které jsou ve zprávě napsány, je veřejně archivováno a nemůže být smazáno. E-maily zaslané do fóra jsou moderovány.

Datum vydání a verze programu

Vydáno Červen 2022. Založeno na LibreOffice 7.3 Community.

Jiné verze LibreOffice se mohou lišit vzhledem a funkčností.

Český překlad

Český překlad byl vydán v lednu 2023. Aktualizaci překladu z verze příručky 7.3 provedli Petr Kuběj, Radomír Strnad a Zdeněk Crhonek. Snímky obrazovky: Roman Toman. Technická výpomoc: Miloš Šrámek

Obsah

Autorská práva.....	2
Předmluva.....	6
Pro koho je tato kniha určena?.....	7
O čem je tato kniha?.....	7
Ukázka databází.....	7
Kde získat pomoc.....	7
Možné vzhledové odchylky.....	10
Používání LibreOffice na systému macOS.....	10
Jak se jednotlivé prvky nazývají?.....	11
Často kladené otázky.....	12
Co je nového v LibreOffice Base 7.3 Community?.....	13
Co je v této knize nového.....	14
Zpětná vazba.....	14
Kapitola 1 Představení Base.....	15
Úvod.....	16
Užitečné základní informace.....	17
Části hlavního okna Base.....	19
Base – kontejner pro databázový obsah.....	24
Bezpečná manipulace se souborem programu Base.....	31
Jednoduchá databáze – podrobný příklad.....	32
Kapitola 2 Vytvoření databáze.....	76
Úvod.....	77
Vytvoření databáze pomocí vestavěného databázového stroje.....	77
Přístup k externím databázím.....	80
Následná úprava vlastností připojení.....	123
Kapitola 3, Tabulky.....	126
Obecné informace o tabulkách.....	127
Relace mezi tabulkami.....	127
Tvorba tabulek.....	133
Propojení tabulek.....	151
Zadávání dat do tabulek.....	154
Kapitola 4 Formuláře.....	170
Formuláře usnadňují zadávání dat.....	171
Vytváření formulářů.....	171
Vlastnosti formuláře.....	175
Hlavní formuláře a podformuláře.....	221
Jeden pohled – více formulářů.....	234
Chybové zprávy při zadávání do formulářů.....	240
Vyhledávání a filtrování ve formulářích pomocí navigační lišty.....	241
Zadávání záznamů a navigace.....	247
Tisk z formulářů.....	248
Kapitola 5 Dotazy.....	250

Obecné informace o dotazech.....	251
Vytváření dotazů pomocí dialogového okna Návrh dotazu.....	251
Tvorba dotazů v režimu SQL.....	270
Použití aliasu v dotazu.....	278
Dotazy na vytvoření polí seznamu.....	279
Dotazy jako základ pro další informace ve formulářích.....	281
Možnosti zadávání dat v rámci dotazů.....	282
Použití parametrů v dotazech.....	287
Poddotazy.....	288
Související poddotazy.....	289
Dotazy jako zdrojové tabulky pro dotazy.....	289
Shrnutí dat pomocí dotazů.....	293
Rychlejší přístup k dotazům pomocí zobrazení tabulek.....	294
Chyby ve výpočtech v dotazech.....	294
Kapitola 6 Sestavy.....	298
Vytváření sestav pomocí nástroje Návrhář sestav.....	299
Uživatelské rozhraní nástroje Návrhář sestav.....	299
Funkce v nástroji Návrhář sestav.....	314
Příklady sestav vytvořených pomocí nástroje Návrhář sestav.....	323
Zdroje chyb ve zprávách.....	338
Kapitola 7 Propojení s databázemi.....	339
Obecné poznámky k propojení databází.....	340
Registrace databází.....	340
Prohlížeč zdrojů dat.....	340
Hromadná korespondence.....	348
Vytváření dokumentů hromadné korespondence.....	349
Tisk štítků.....	355
Přímé vytváření dokumentů hromadná korespondence a štítků.....	357
Externí formuláře.....	359
Použití databáze v aplikaci Calc.....	361
Převod dat z jedné databáze do druhé.....	365
Import záznamů do tabulky pomocí schránky.....	365
Importování PDF záznamů.....	365
Kapitola 8 Úlohy databáze.....	373
Obecné poznámky k databázovým úlohám.....	374
Filtrování dat.....	374
Vyhledávání dat.....	376
Zpracování obrázků a dokumentů v aplikaci Base.....	382
Čtení a zobrazování obrázků a dokumentů.....	391
Úryvky kódu.....	392
Kapitola 9 Makra.....	402
Obecné poznámky k makrům.....	403
Makra v Base.....	404
Zlepšení použitelnosti.....	430

Úlohy databáze rozšířené pomocí maker.....	471
Přístup k databázi MySQL pomocí maker.....	491
Dialogová okna.....	492
Psaní maker pomocí Access2Base.....	511
Knihovna ScriptForge.....	515
Kapitola 10 Údržba databáze.....	523
Obecné poznámky k údržbě databází.....	524
Komprimace databáze.....	524
Obnovení automatických hodnot.....	524
Dotazování na vlastnosti databáze.....	524
Export dat.....	525
Testování tabulek na nepotřebné položky.....	526
Rychlost vyhledávání v databázi.....	529
Příloha A Běžné úlohy databáze.....	531
Úvod.....	532
Čárové kódy.....	532
Datové typy pro editor tabulek.....	532
Datové typy v LibreOffice Basic.....	534
Vestavěné funkce a uložené procedury.....	535
Řídící znaky pro použití v dotazech.....	541
Některé příkazy UNO pro použití s tlačítkem.....	541
Informační tabulky pro HSQLDB.....	542
Oprava databáze pro soubory *.odb.....	543
Správa interní databáze Firebird.....	553
Dodatek B Porovnání HSQLDB a Firebird.....	554
Datové typy a funkce v HSQLDB a Firebird.....	555
Vestavěné funkce a uložené procedury.....	555
Datové typy pro editor tabulek.....	574



LibreOffice
Community



Příručka aplikace Base 7.3

Předmluva

Pro koho je tato kniha určena?

Tuto knihu jistě ocení každý, kdo chce s LibreOffice Base pracovat svižně, efektivně a s přehledem. Ať už jsme s databázemi nikdy předtím nepracovali nebo jsme s nimi pracovali v systému DBMS (Database Management System) nebo jste zvyklí na jiný databázový systém z kancelářského balíku nebo samostatného databázového systému, jako je MySQL, tato kniha je určena pro všechny. Možná bychom si mohli nejprve přečíst kapitolu 8, Začínám s programem Base, v příručce *Začínáme s LibreOffice*.

O čem je tato kniha?

Tato kniha představuje program Base, databázovou komponentu LibreOffice. Program Base používá pro tvorbu databázových dokumentů databázový stroj HSQLDB (HyperSQL DataBase). Umožňuje přistupovat k databázím vytvořeným v mnoha databázových programech, včetně Microsoft Access, MySQL, Oracle, Firebird a PostgreSQL. Program Base obsahuje přídatné funkce, které umožňují vytvářet úplné aplikace založené na datech.



Poznámka

Kromě stroje HSQLDB je k dispozici experimentální možnost vytvářet databázové dokumenty pomocí vestavěného databázového stroje Firebird. Pokročilí uživatelé mohou experimentální funkce povolit zaškrtnutím políčka **Povolit experimentální funkce** v dialogovém okně **Nástroje > Možnosti > LibreOffice > Pokročilé**.

Tato kniha představuje vlastnosti a funkce programu Base pomocí ukázkových databází.

- Vytvoření databáze
- Přístup k externím databázím
- Vytváření a používání tabulek v relačních databázích
- Vytváření a používání formulářů pro zadávání dat
- Použití dotazů ke spojení dat z různých tabulek, k výpočtu výsledků (v případě potřeby) a rychlému filtrování konkrétního záznamu z velkého množství dat
- Vytváření sestav pomocí nástroje Návrhář sestav
- Propojení databází s jiným dokumentu a externími formuláři včetně použití v hromadné korespondenci
- Filtrování a vyhledávání dat
- Používání maker k prevenci chyb při zadávání, zjednodušení úkolů a zlepšení použitelnosti formulářů
- Údržba databází

Ukázka databází

K této knize byla vytvořena sada ukázkových databází. Najdete je zde: <https://wiki.documentfoundation.org/images/5/52/Sample-databases.zip>.

Kde získat pomoc

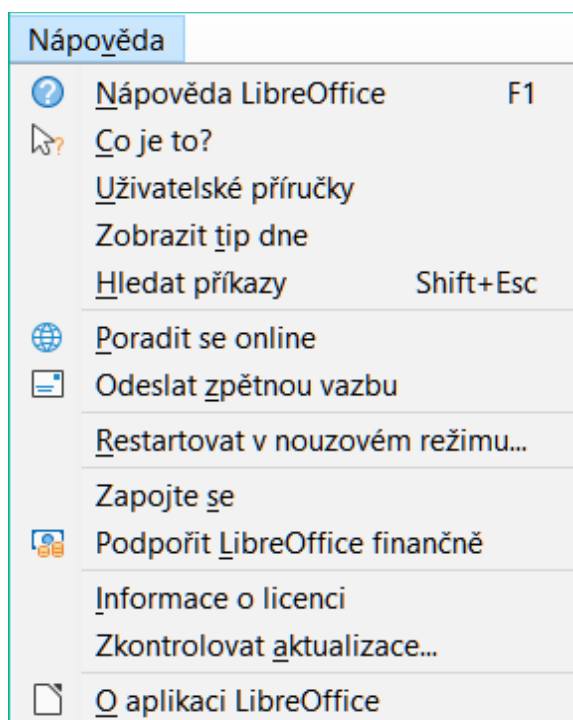
Tato kniha, ostatní příručky pro LibreOffice, vestavěná nápověda a podpora uživatelů předpokládají, že uživatel má základní znalosti s prací na počítači, tj. že dokáže spouštět programy či otevírat a ukládat soubory.

System nápovědy

LibreOffice přichází s rozsáhlým systémem nápovědy. Je prvním místem, kde získat pro LibreOffice podporu. Uživatelé Windows a Linux si mohou zvolit stažení a instalaci offline nápovědy pro použití ve chvíli, kdy nejsou připojeni k internetu; offline nápověda je nainstalována s programem na systému MacOS.

Pokud chceme zobrazit systém Nápovědy, stiskneme klávesu *F1* nebo v hlavní nabídce vybereme možnost **Nápověda > Nápověda LibreOffice**. Pokud nemáme v počítači nainstalovanou offline nápovědu a jsme připojeni k internetu, náš výchozí prohlížeč otevře stránky online nápovědy na webu LibreOffice.

Nabídka Nápověda (obrázek 1) obsahuje odkazy na další informace a podporu LibreOffice.



Obrázek 1: Nabídka nápověda

✓ Poznámka

Možnosti označené v níže uvedeném seznamu symbolem † jsou přístupné pouze v případě, že je počítač připojen k internetu.

- **Co je to?** – Rychlé tipy získáme, když umístíme ukazatel myši nad ikonu. Zobrazí se tip nápovědy s krátkým popisem funkce ikony. Podrobnější vysvětlení získáme, pokud v hlavní nabídce zvolíme **Nápověda > Co je to?** a kurzor myši podržíme nad ikonou. Kromě toho je možné aktivovat rozšířené tipy, a to pomocí **Nástroje > Možnosti > LibreOffice > Obecné** z hlavní nabídky.
- **Uživatelské příručky**† – Otevře výchozí prohlížeč na stránce Dokumentace na webu LibreOffice <https://documentation.libreoffice.org/cs/czech-documentation>. Najdeme zde kopie uživatelských příruček a další užitečné informace.
- **Zobrazit tip dne** – Otevře dialogové okno s užitečnými informacemi, které nám pomohou rozšířit znalosti o LibreOffice.
- **Poradit se online** † – Otevře výchozí prohlížeč na stránce Ask LibreOffice, stránce dotazů a odpovědí české a slovenské komunity uživatelů kancelářského balíku LibreOffice <https://ask.libreoffice.org/cs-cz/questions>.

- **Odeslat zpětnou vazbu** [‡] – Otevře výchozí prohlížeč na webové stránce zpětné vazby LibreOffice <https://www.libreoffice.org/get-help/feedback>. Odtud můžeme hlásit chyby, navrhopvat nové funkce a komunikovat s ostatními v komunitě LibreOffice.
- **Restartovat v nouzovém režimu** - Otevře dialogové okno s dotazem, zda chceme restartovat LibreOffice a přejít do nouzového režimu.
- **Zapojte se** [‡] – Otevře výchozí prohlížeč na webové stránce LibreOffice Připojte se k nám <https://cs.libreoffice.org/community/pipojte-se-k-nam>. Na ní si můžeme vybrat oblast zájmu, v níž pomůžeme program vylepšit.
- **Podpořit LibreOffice finančně** [‡] – Otevře výchozí prohlížeč na stránce Podpořte LibreOffice <https://cs.libreoffice.org/donate>.
- **Informace o licenci** – Otevře dialogové okno Licenční a právní informace s informací o licenci, pod kterou je LibreOffice přístupná.
- **Zkontrolovat aktualizace** [‡] – Otevře dialogové okno a zkontroluje web LibreOffice, zda neobsahuje aktualizace naší verze softwaru.
- **O aplikaci LibreOffice** – Otevře dialogové okno a zobrazí informace o verzi LibreOffice a používaném operačním systému. Tyto informace budou často vyžadovány, pokud požádáte komunitu o pomoc nebo asistenci se softwarem. K dispozici je tlačítko, které nám umožní zkopírovat tyto informace do schránky, abychom je mohli následně vložit do příspěvku na fóru, do e-mailu nebo do zprávy o chybě. (Na počítačích s operačním systémem macOS je tato položka pod hlavním menu **LibreOffice**.)

Další volně dostupná podpora na internetu

Komunita LibreOffice kromě vývoje softwaru poskytuje bezplatnou podporu od dobrovolníků. Kromě výše uvedených odkazů v nabídce Nápověda jsou k dispozici další možnosti online podpory komunity, viz tabulka 1.

Tabulka 1: Bezplatná podpora LibreOffice

Služba	Popis
Časté otázky	Odpovědi na často kladené otázky https://wiki.documentfoundation.org/Faq
Fórum	Podpora od komunity poskytovaná sítí zkušených uživatelů https://community.documentfoundation.org/
Otázky a odpovědi a Databáze znalostí	Pomocí webové služby Ask je poskytována komunitní podpora zdarma. Vyhledat podobná témata nebo otevřít nová můžeme na stránce https://ask.libreoffice.org/c/czech-and-slovak/13 . Služba je k dispozici v několika dalších jazycích; stačí nahradit /en/ za cs-cz, de, es, fr, ja, ko, nl, pt, tr, a mnoho dalších na výše uvedené webové adrese.
Podpora v různých jazycích	Webové stránky LibreOffice v různých jazycích https://cs.libreoffice.org/community/nlc/ E-mailové konference pro různé jazyky https://wiki.documentfoundation.org/Local_Mailing_Lists Informace o sociálních sítích https://wiki.documentfoundation.org/Website/Web_Sites_services
Možnosti zpřístupnění	Informace o dostupných možnostech pro zlepšení přístupnosti https://cs.libreoffice.org/get-help/accessibility/

Placená podpora a školení

Je také možno zakoupit podporu ve formě smlouvy o poskytování služeb, a to od prodejce nebo poradenské firmy specializované na LibreOffice. Informace o certifikované profesionální podpoře se nachází na webových stránkách The Document Foundation: <https://cs.libreoffice.org/get-help/professional-support/>

Pro školy, vzdělávací a výzkumné instituce a velké organizace viz <https://www.libreoffice.org/download/libreoffice-in-business/>.

Možné vzhledové odchylky

Ilustrace

LibreOffice lze instalovat a spouštět v operačních systémech Windows, Linux a macOS, přičemž každý z nich má několik verzí a uživatelé si je mohou přizpůsobit (písma, barvy, témata vzhledu, správce oken). Ilustrace v tomto návodu byly vytvořeny v různých operačních systémech a na různých počítačích. Proto je možné, že některé prvky v ilustracích nebudou přesně takové, jak je vidíme na svém počítači.

Některá dialogová okna se mohou lišit i kvůli různým nastavením samotného LibreOffice. Uživatel si totiž může zvolit, zda chce používat dialogová okna operačního systému, nebo okna LibreOffice. Tyto rozdíly se týkají především dialogových oken Otevřít, Uložit a Tisk. Zobrazování dialogových oken LibreOffice lze povolit a zakázat v dialogovém okně **Nástroje > Možnosti > LibreOffice > Obecné**, kde zaškrtneme či zrušíme zaškrtnutí možnosti **Použít dialogy LibreOffice**.

Ikony

Komunita LibreOffice vytvořila ikony pro několik sad ikon: Colibre, Breeze, Breeze Dark, Elementary, Karasa Jaga, Sifr, Sifr Dark a Sukapura. Každý uživatel si může vybrat preferovanou sadu. Snímky obrazovky byly pro tuto příručku vytvářeny v různých instalacích LibreOffice, které používaly různé sady ikon. Ikony, které opravdu uvidíme ve své verzi LibreOffice, se mohou lišit od těch, které jsou zobrazeny v této příručce.

Chceme-li změnit použitou sadu ikon, přejdeme na **Nástroje > Možnosti > LibreOffice > Zobrazení**. Vybereme z rozevíracího seznamu *Styl ikon*.

Poznámka

Sady ikon Galaxy a Oxygen již nejsou součástí standardního instalačního balíčku. Lze je přidat zpět stažením a instalací následujících rozšíření:
<https://extensions.libreoffice.org/extensions/galaxy-icon-theme>
<https://extensions.libreoffice.org/extensions/oxygen-icon-theme>
<https://extensions.libreoffice.org/en/extensions/show/tango-icon-theme-for-libreoffice>
Některá z dříve zahrnutých pozadí galerie jsou nyní k dispozici pouze jako rozšíření od:
<https://extensions.libreoffice.org/extensions/legacy-gallery-backgrounds>

Používání LibreOffice na systému macOS

Některé klávesové zkratky a položky nabídek jsou v systému macOS jiné než v systémech Windows a Linux. Tabulka 2 uvádí některé běžné náhrady instrukcí v tomto dokumentu. Podrobnější seznam se nachází v nápovědě aplikace.

Tabulka 2: Příklad klávesových zkratk pro různá prostředí

<i>Windows nebo Linux</i>	<i>Ekvivalent pro macOS</i>	<i>Akce</i>
Výběr v nabídce Nástroje > Možnosti	LibreOffice > Předvolby	Otevřou se možnosti nastavení.
Klepnutí pravým tlačítkem	<i>Control</i> + klepnutí a/nebo klepnutí pravým tlačítkem myši v závislosti na nastavení počítače	Otevře se místní nabídka.
<i>Ctrl</i> (Control)	⌘ (Command)	Používá se také s dalšími klávesami.
<i>F11</i>	⌘ + T	Otevře se postranní lišta Styly.

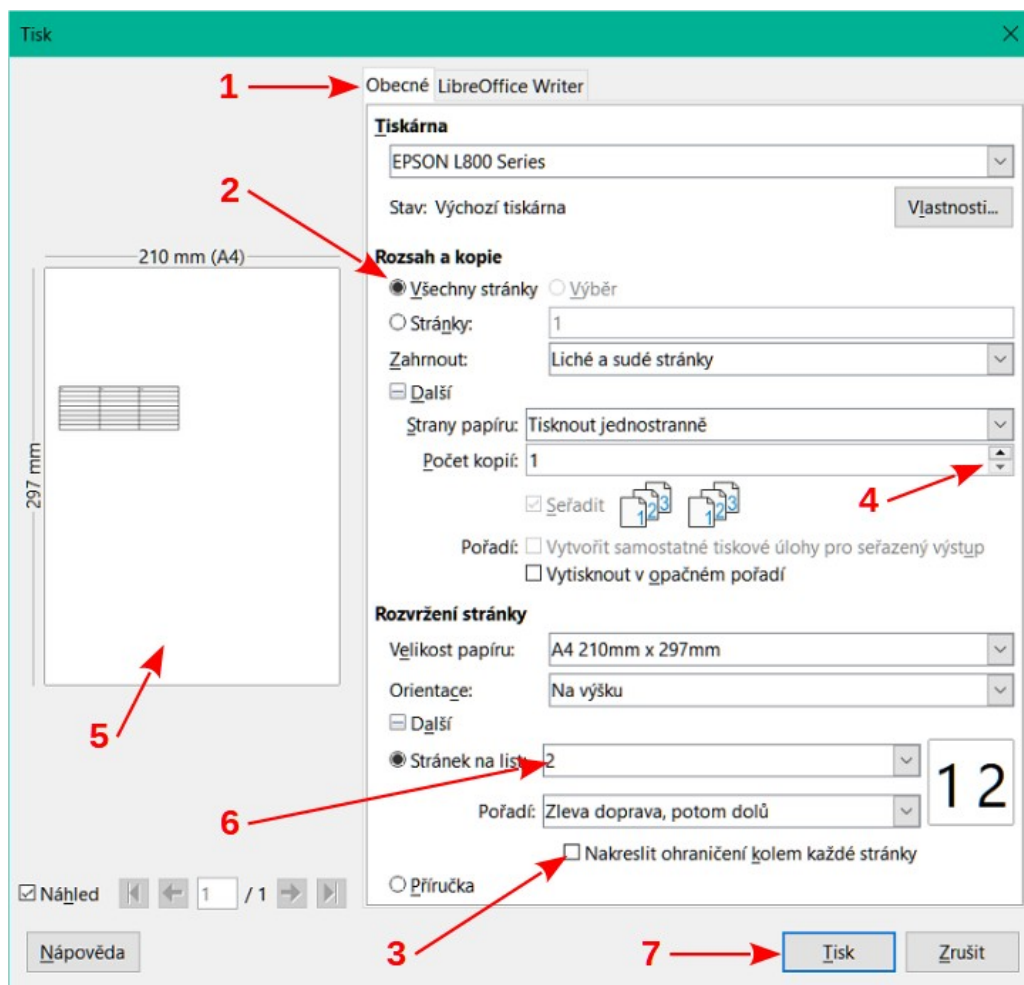
Jak se jednotlivé prvky nazývají?

Pojmy používané v LibreOffice pro označování prvků uživatelského rozhraní jsou stejné jako ve většině jiných programů.

Dialogové okno je speciálním typem okna. Jeho účelem je o něčem informovat, vyžádat si vložení údajů nebo obojí najednou. Okno obsahuje ovládací prvky, které je možné použít k provedení nějaké akce. Na obrázku 2 jsou zvýrazněny některé běžné ovládací prvky a v klíči za obrázkem je uveden jejich stručný technický popis. V této knize nebudeme ve většině případů technické pojmy používat, ale je užitečné tyto pojmy znát, protože nápověda a další zdroje informací s nimi často pracují.

Ve většině případů můžeme během doby, kdy je dialogové okno otevřeno, pracovat pouze s tímto oknem, nikoliv se samotným dokumentem. Když dialogové okno po použití zavřeme (obvykle klepnutím na **OK** nebo jiné tlačítko uložíme změny a dialogové okno se zavře), můžeme s dokumentem opět pracovat.

Některá dialogová okna je možné ponechat otevřená i během práce a přepínat se mezi nimi a dokumentem.



Obrázek 2: Dialogové okno, ve kterém jsou použity běžné ovládací prvky

- 1) Karty stránky (angl. Tabbed page, tab), které slouží k přepínání karet.
- 2) Přepínače (angl. Radio buttons), zvolená může být vždy jen jedna položka.
- 3) Zaškrtnutí pole (angl. checkbox), může být zvoleno několik položek najednou.
- 4) Číselník (angl. spin box). Klepnutím na šipky můžeme číslo zmenšit nebo zvětšit, nebo jej můžeme přímo zadat v textové části.
- 5) Náhled.
- 6) Z rozbalovacího seznamu zvolíme námi požadovanou položku.
- 7) Tlačítka (angl. Push buttons).

Často kladené otázky

Jakou licenci LibreOffice používá?

LibreOffice je šířen pod licenci Mozilla Public License (MPL), schválenou organizací the Open Source Initiative (OSI). Viz <https://www.libreoffice.org/about-us/licenses/>

Je založen na kódu z Apache OpenOffice, který je k dispozici pod licenci Apache License 2.0, ale zahrnuje také software, který se liší od verze k verzi v rámci řady dalších open source licencí. Nový kód je dostupný pod licenci GNU Lesser General Public License (LGPL) 3.0 and MPL 2.0.

Mohu LibreOffice dále distribuovat, rozdávat a šířit? Mohu LibreOffice prodávat? Mohu LibreOffice používat pro komerční účely a ve své firmě?

Ano.

Na kolik počítačů mohu LibreOffice nainstalovat?

Na libovolný počet.

Je LibreOffice dostupný v mém jazyce?

LibreOffice byl přeložen (lokalizován) do více než 100 jazyků, takže váš jazyk je pravděpodobně podporován. Navíc je k dispozici více než 100 slovníků pro kontrolu pravopisu, dělení a tezaury pro jazyky a jejich dialekty, do nichž není lokalizované uživatelské rozhraní. Slovníky jsou dostupné výběrem filtru tagu Dictionary v Centru rozšíření LibreOffice <https://extensions.libreoffice.org/>.

Jak je možné, že program LibreOffice je dostupný zdarma?

LibreOffice vyvíjejí a udržují dobrovolníci s podporou více firem.

Vyvíjím softwarovou aplikaci. Mohu použít zdrojový kód LibreOffice v mém programu?

V rámci parametrů stanovených v licencích můžete: <https://www.mozilla.org/MPL/2.0/>
<https://www.libreoffice.org/about-us/licenses/>.

Proč ke spuštění LibreOffice potřebuji Javu? Je napsán v Javě?

LibreOffice není psán v Javě; většinou je psán v jazyce C++. Java je jedním z jazyků, které lze použít k jeho rozšiřování. Běžné prostředí Java (JDK / JRE) je třeba jen kvůli některým funkcím. Jednou z nich je například databázový stroj HSQLDB, používaný komponentou Base.

Java je dostupná bezplatně. Další informace a odkazy ke stažení na příslušné vydání pro náš operační systém najdeme na adrese: <https://java.com/en/download/manual.jsp>



Poznámka

Pokud chceme používat funkce LibreOffice, které vyžadují Javu, je důležité, aby se správná 32 bitová nebo 64 bitová edice shodovala s nainstalovanou verzí LibreOffice.

Jak mohu k LibreOffice přispět?

Při vývoji a uživatelské podpoře LibreOffice lze pomoci různými způsoby. Není k tomu třeba být programátor. Chceme-li začít, podíváme se na tuto webovou stránku: <https://cs.libreoffice.org/community/get-involved/>

Mohu PDF soubor této knihy dále distribuovat, případně jej i tisknout a prodávat?

Ano, pokud jsou splněny podmínky alespoň jedné z licencí, které jsou uvedeny na začátku této knihy. Není k tomu třeba žádné speciální povolení. Jako přispěvatel k této knize vás však žádáme, abyste vzali v úvahu množství práce, které bylo na její psaní a překlad vynaloženo a abyste se podělili o část svého zisku.

Přispějte LibreOffice: <https://cs.libreoffice.org/donate/>

Co je nového v LibreOffice Base 7.3 Community?

LibreOffice 7.3 Community obsahuje mnoho změn, které nejsou viditelné v uživatelském rozhraní. Tyto změny zahrnují lepší interoperabilitu s proprietárními formáty souborů společnosti Microsoft a zlepšení výkonu při zpracování velkých souborů, otevírání některých souborů .docx a .xlsx, správně ukládání písem do mezipaměti a otevírání prezentací a kreseb obsahujících velké obrázky.

Více informací najdete v oznámení:

<https://blog.documentfoundation.org/blog/2022/02/02/libreoffice-73-community/>

Poznámky k verzi nalezneme zde: <https://wiki.documentfoundation.org/ReleaseNotes/7.3/cs>

Tato uživatelská příručka je založena na *Příručce programu Base 7.2*. Informace v kapitolách a přílohách, které následují po této předmluvě, se od *Příručky k programu Base 7.2* nezměnily –

hlavní práce prováděné na Base od 7.2 se týkaly průběžného vývoje experimentálních funkcí pro vytváření databázových dokumentů pomocí Firebirdu.

Co je v této knize nového

Od vydání Příručky k modulu Base 7.2 byla tato kniha podrobena revizi z hlediska přesnosti a úplnosti v rámci neustálé snahy přinést vám nejlepší příručku k modulu LibreOffice Base.

Kapitola 2

Přidáno nastavení SQLite.

Recenze kapitoly.

Kapitola Dodatek A

Přidán úvodní text.

Tato kniha se neustále aktualizuje a přepracovává pro vydání.

Zpětná vazba

Jakékoli připomínky nebo návrhy k tomuto dokumentu prosím směřujte do fóra dokumentačního týmu na adrese <https://community.documentfoundation.org/c/documentation/loguides/> (registrace je nutná) nebo pošlete e-mail na adresu: loguides@community.documentfoundation.org.



Poznámka

Vše, co napíšete do fóra, včetně vaší e-mailové adresy a dalších osobních údajů, které jsou ve zprávě napsány, je veřejně archivováno a nemůže být smazáno. E-maily zaslané do fóra jsou moderovány.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 1 *Představení Base*

Úvod

Co je Base?

Base je databázová front-end komponenta LibreOffice, volně dostupného, plně vybaveného kancelářského balíku s otevřeným zdrojovým kódem. Base poskytuje tyto důležité funkce:

- **Vytvoření databáze.** Nové vložené databáze lze vytvářet pomocí vloženého databázového stroje HSQLDB.
- **Připojení k dalším zdrojům dat.** Připojit se lze k mnoha běžně používaným databázovým strojům a dalším zdrojům dat, včetně tabulek, textových dokumentů a adresářů. Před vytvořením takového připojení musí být vytvořeny a správně nakonfigurovány externí zdroje dat.
- **Tabulky.** V mnoha databázích jsou data uspořádána do tabulek s řádky a sloupci. Všeobecně řečeno, sloupce představují kategorie dat, zatímco řádky představují jednotlivé případy. Tabulky v relační databázi jsou propojeny relacemi, které umožňují sdílení dat mezi tabulkami a eliminují redundanci dat.
- **Formuláře.** Formulář je uživatelsky přívětivé okno, které slouží k prohlížení a úpravám dat v tabulkách databáze. Formulář pomáhá uživateli rychle a přesně zadávat data a zajišťuje, že aktualizace dat jsou prováděny konzistentně ve všech dotčených tabulkách databáze.
- **Dotazy.** Dotaz načte data z databáze, v případě potřeby vypočítá výsledky a vyfiltruje záznamy, které nás zajímají, z velkého objemu dat. Dotazy mohou spojovat data z jedné nebo více tabulek, někdy prostřednictvím jiného dotazu nebo zobrazení.
- **Zobrazení.** Pro některé připojené typy databází je možné vytvářet databázové pohledy. Zobrazení je virtuální tabulka určená pouze pro čtení, která obsahuje řádky a sloupce. Je vytvořena pomocí dotazu, který shromažďuje data fyzicky uložená v jedné nebo více „skutečných“ tabulkách. Většina dotazů vytvořených pomocí Base je uložena v dokumentu Base, ale základní dotaz pohledu je uložen v databázovém stroji.
- **Sestavy.** Sestava se používá k extrakci klíčových údajů z databáze a jejich prezentaci ve snadno čitelném formátu, často pro potřeby lidí (např. rozhodovacích orgánů), kteří jinak nejsou přímo zapojeni do práce s databází.
- **Propojení.** Databáze lze propojit s dalšími dokumenty a externími formuláři, včetně použití při hromadné korespondenci.
- **Filtrování a vyhledávání.** Filtry se používají k omezení objemu zobrazených dat a zaměřují pozornost na záznamy, které splňují určitá kritéria filtru. Vyhledávání slouží k nalezení záznamů, které nás zajímají a odpovídají konkrétním vyhledávacím kritériím.
- **Makra.** Makra LibreOffice Basic a Python lze použít ke zjednodušení opakujících se úloh, aby se zabránilo chybám při zadávání, zvýšila se funkčnost a zlepšila použitelnost. LibreOffice obsahuje knihovny maker ScriptForge a Access2Base, které poskytují rozsáhlou sbírku funkcí, jež lze vyvolat ze skriptů jazyka Basic nebo Python.



Poznámka

V tomto vydání je většina maker v tomto dokumentu nakódována v jazyce LibreOffice Basic. Další informace o používání skriptů Python najdeme v nápovědě.

- **Údržba databáze.** To zahrnuje komprimaci databáze, resetování automatických hodnot, export dat a vyhledávání nepotřebných záznamů.

Použití správného nástroje pro danou práci

V každodenním kancelářském provozu se tabulky pravidelně používají k agregaci souborů dat a k jejich analýze. Vzhledem k tomu, že data v tabulkovém procesoru jsou zobrazena v tabulce, jsou přehledná a snadno se upravují nebo rozšiřují, mnoho uživatelů se ptá, proč by měli používat databázi místo tabulkového procesoru. Tato kniha vysvětluje rozdíly mezi těmito nástroji a poskytuje čtenářům informace, které jim pomohou vybrat nejlepší řešení pro jejich konkrétní požadavky.

Databáze poskytne výhody mnoha uživatelům a dobře navržená databáze minimalizuje duplicitu, chyby a nekonzistence, které lze v tabulkovém procesoru obtížně identifikovat a odstranit. Některé potenciální uživatele databází však může odradit neznalost databázových pojmů a terminologie. Uživatelé, kteří se obávají složitosti, by měli být ujištěni, že:

- Databázové koncepty a terminologie používané v systému Base jsou do značné míry shodné s koncepty a terminologií používanou v mnoha jiných běžných databázových systémech. To znamená, že mnoho informací o databázových technologiích, které jsou volně dostupné na internetu, může být užitečné pro pochopení aspektů databáze Base.
- Jelikož se jedná o klíčovou součást integrovaného balíku LibreOffice, uživatelé, kteří znají ostatní součásti, by neměli mít problémy s interakcí s aplikací Base, která využívá nabídky a nástrojové lišty klasického uživatelského rozhraní LibreOffice. Vzhledem k povaze aplikace Base neposkytuje (ani nepotřebuje) možnosti výběru uživatelského rozhraní (například varianty s kartami), které jsou k dispozici v některých jiných součástech LibreOffice.
- Base poskytuje průvodce, kteří pomáhají uživatelům, kteří s tvorbou databází teprve začínají (nebo jen začínají pracovat s tímto softwarem), vytvářet databáze, tabulky, dotazy, formuláře a sestavy, a dodává se se sadou předdefinovaných definic tabulek pro sledování majetku, zákazníků, prodejních objednávek, faktur a mnoha dalších běžných a užitečných položek.
- Kromě toho existuje mnoho dalších zdrojů pomoci, jak je popsáno v části „Kde získat další pomoc“ v předmluvě k této knize.

Ukázka databází

V této kapitole je představena jedna ze vzorových databází, která se používá v této knize – *Media_without_Macros.odt*. Tato databáze je určena pro pomoc s provozem knihovny médií a vším, co s ní souvisí. K vybavení patří:

- Zachycení podrobností o médiích.
- Zachycení informací o autorech a vypůjčitelích.
- Funkce správy výpůjček.

Kromě toho je v této kapitole použita ukázková databáze *Example_Sport.odt*, která je určena pro pomoc při organizaci sportovní soutěže.

Užitečné základní informace

Base můžeme použít dvěma různými způsoby:

- 1) Base může fungovat jako front end pro širokou škálu databázových strojů a poskytovat uživatelské rozhraní, které podporuje vytváření a používání prvků, jako jsou dotazy, formuláře, sestavy a makra. V této roli databázového front-endu vykonává hlavní (back-endové) databázové funkce, jako je ukládání a organizace dat, databázový stroj, nikoli databáze Base. Uživatel Base tak může pracovat s existujícími místními nebo síťovými databázemi, které mohou být víceuživatelskými prostředky sdílenými s ostatními členy organizace.

- 2) Base může pracovat také v samostatném režimu s vestavěným relačním databázovým strojem HSQLDB (HyperSQL DataBase, verze 1.8.0) v jazyce Java ve standardní distribuci LibreOffice. Jedná se o ideální řešení pro každodenní práci a pro lidi, kteří vyžadují snadno pochopitelný, jednoduše použitelný databázový systém pro jednoho uživatele namísto komunikace s externím databázovým strojem. Více informací o HSQLDB najdeme na stránce <http://hsqldb.org/>.

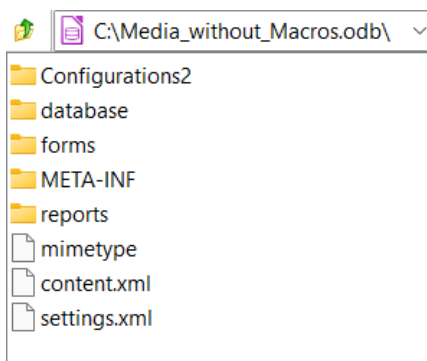
Pokračují práce na vylepšení integrace relační databáze Firebird (verze 3.0.7), která není v jazyce Java, jako alternativního vestavěného stroje v rámci databáze Base. Dokud nebude tato práce dokončena, je možnost vytvořit databázi pomocí Firebirdu k dispozici pouze jako experimentální funkce. Base se však může připojit k databázovému stroji Firebird prostřednictvím externího serveru bez povolení experimentálních funkcí. Více informací o databázi Firebird najdeme na stránce <https://firebirdsql.org/>.

Base se může připojit k některým z nejrozšířenějších víceuživatelských databázových strojů. Seznam dostupných typů databází se liší v závislosti na operačním systému, ale vždy by měl obsahovat následující typy:

- MariaDB – <https://mariadb.org/>
- MySQL – <https://www.mysql.com/>
- PostgreSQL – <https://www.postgresql.org/>

Kromě toho integrovaná podpora standardních ovladačů JDBC (Java Database Connectivity) a ODBC (Open Database Connectivity) umožňuje aplikaci Base připojit se k mnoha dalším existujícím databázovým strojům. Base se může připojit i k dalším zdrojům dat, včetně tabulek, textových dokumentů a adresářů.

Base ukládá své databázové informace do souborů s příponou ".odb", které jsou formátovány v souladu se standardem *Open Document Format for Office Applications (OpenDocument)* verze 1.2. Část 3: *Balíčky*. Soubory ODB vytvořené nástrojem Base obsahují kolekci souborů a složek, které jsou komprimovány pomocí bezeztrátové komprese dat ZIP (obrázek 3). Soubor ODB může obsahovat mnoho souborů, například data týkající se vloženého databázového stroje, formulářů, dotazů, sestav a maker. Pokud se soubor vztahuje k externí databázi, nikoli k vloženému stroji, bude obsahovat spíše informace o připojení k databázi než o datech uložených v tabulkách databáze.



Obrázek 3: Struktura nejvyšší úrovně složek a souborů v souboru ODB

Správné fungování některých funkcí Base závisí na prostředí Java Runtime Environment (JRE). Pokud není pro LibreOffice k dispozici JRE, nebudou k dispozici následující možnosti:

- Vestavěná databáze HSQLDB.
- Průvodci tabulkou, dotazem a formulářem.
- Návrhář sestav.

- Přístup k databázi prostřednictvím Java Database Connectivity (JDBC).

Base zobrazí příslušná chybová dialogová okna s upozorněním na nedostupnost JRE. Pokud je JRE nainstalováno, ale není povoleno v LibreOffice, lze jej povolit výběrem **Nástroje > Možnosti > LibreOffice > Pokročilé** v hlavní nabídce a kliknutím na možnost **Použít běhové prostředí jazyka Java**.

Oracle Java si můžeme stáhnout z adresy <https://www.java.com/en/download/manual.jsp>. Pokud používáme 32bitovou verzi LibreOffice, měli bychom nainstalovat 32bitovou verzi JRE. Pokud používáme 64bitovou verzi LibreOffice, měli bychom nainstalovat 64bitovou verzi JRE.

Upozornění

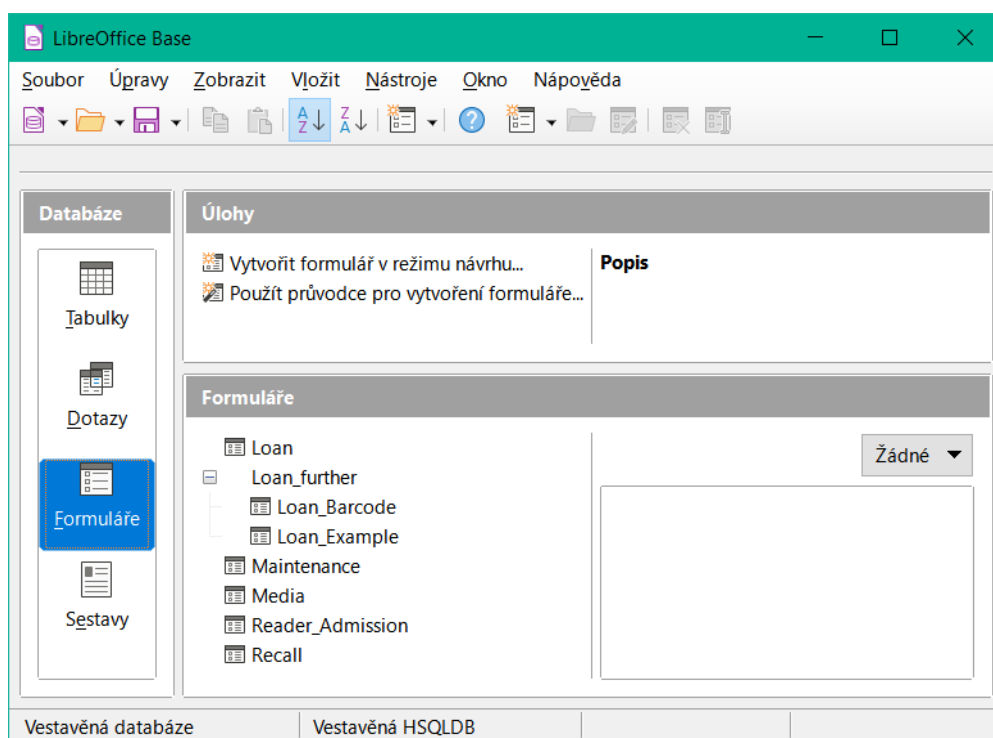
Na webu společnosti Oracle pro stahování je ve výchozím nastavení k dispozici 32bitový soubor JRE – pokud potřebujeme získat přístup k 64bitovému souboru JRE, měl by v názvu souboru obsahovat řetězec „x64“. 64bitová verze LibreOffice nerozpozná 32bitové JRE (a naopak), což je častým zdrojem problémů.

V systému macOS je třeba nainstalovat vývojovou sadu Java Development Kit (JDK) společnosti Oracle, nikoli pouze JRE. Odkazy ke stažení najdeme výběrem možnosti macOS na adrese: <https://www.oracle.com/java/technologies/downloads/>.

Pokud jsou naše data uložena v relační databázi, budeme nevyhnutelně chtít využít možnosti přímého zadávání příkazů SQL. SQL je zkratka pro Structured Query Language a jedná se o standardní jazyk (ISO/IEC 9075) pro přístup k datům v relačních databázích a manipulaci s nimi. Tato kniha obsahuje mnoho příkladů příkazů SQL. Pokud zpočátku neznáme syntaxi a sémantiku jazyka SQL, nemusíme se bát, na internetu jsou k dispozici kvalitní a bezplatné výukové materiály. Dobrým výchozím bodem je například výukový kurz SQL, který poskytuje W3Schools na adrese <https://www.w3schools.com/sql/default.asp>.

Části hlavního okna Base

Hlavní okno Base je zobrazeno na obrázku 4 a jeho funkce jsou popsány v následujících podkapitolách.



Obrázek 4: Hlavní okno Base

Záhlaví okna

V horní části okna Base se nachází titulkový pruh. Zobrazuje název souboru aktuálního databázového dokumentu.

Hlavní nabídka

Nabídka je přímo pod titulním pruhem ve Windows i Linuxu, v systému macOS se nachází na horním okraji displeje. Vybereme-li položku nabídky, otevře se podnabídka v některé z následujících variant:

- Příkazy, které přímo provedou nějakou akci, např. **Zavřít** nebo **Uložit** v nabídce **Soubor**.
- Příkazy, které otevírají dialogy. Jsou označeny třemi tečkami (...) za příkazem, například **Otevřít...** v nabídce **Soubor**.
- Příkazy, které otevírají následnou podnabídku. Poznáme je podle doprava směřující šipky za příkazem, například **Nástrojové lišty** a **Náhled** v nabídce **Zobrazit**. Přesunutím kurzoru na jednu z těchto možností se otevře její podnabídka.

Výchozí konfigurace nabídek v hlavní nabídce bude pravděpodobně vyhovovat většině uživatelů aplikace Base. Hlavní nabídku však můžeme přizpůsobit pomocí karty *Nabídky* dialogového okna **Nástroje > Přizpůsobit** – podrobnosti o této funkci nalezneme v systémové nápovědě.

Nástrojové lišty

Kromě hlavní nabídky nabízí aplikace Base několik nástrojových lišt, které umožňují rychlý přístup k často používaným možnostem. Umístěním kurzoru myši na ikonu nástrojové lišty se zobrazí malý rámeček, tzv. tip, který pojmenovává funkci tlačítka. Chceme-li tip uzavřít, přemístíme se z podkladového objektu nebo stiskneme klávesu *Esc*.

Výchozí nastavení při otevření databázového dokumentu Base je takové, že se nástrojová lišta Standardní zobrazí v horní části pracovní plochy (pod hlavní nabídkou) spolu s nástrojovou lištou Formulář. Nástrojová lišta Formulář se při různých výběrech v oblasti *Databáze* hlavního okna nahradí nástrojovou lištou Tabulka, Dotaz nebo Sestava.

Výchozí nástrojová lišta Standardní je zobrazena na obrázku 5 a nabízí následující možnosti (zleva doprava): (nahore): Nový; Otevřít; Uložit; Kopírovat; Vložit; Seřadit vzestupně; Seřadit sestupně; Formulář; nápověda LibreOffice.



Obrázek 5: Standardní nástrojová lišta

Výchozí nástrojová lišta Tabulka je zobrazena na obrázku 6 a nabízí následující možnosti (zleva doprava): Vpravo (zprava): Nový návrh tabulky; Otevřít objekt databáze; Upravit; Odstranit; Přejmenovat.



Obrázek 6: Nástrojová lišta Tabulka

Výchozí nástrojová lišta Dotaz je zobrazena na obrázku 7 a nabízí následující možnosti (zleva doprava): (zprava): Nový dotaz (režim návrhu); Nový dotaz (režim SQL); Otevřít objekt databáze; Upravit; Odstranit; Přejmenovat.



Obrázek 7: Nástrojová lišta Dotaz

Výchozí nástrojová lišta Formulář je zobrazena na obrázku 8 a nabízí následující možnosti (zleva doprava): Formulář; Otevřít databázový objekt; Upravit; Odstranit; Přejmenovat.



Obrázek 8: Nástrojová lišta Formulář

Výchozí nástrojová lišta Sestava je zobrazena na obrázku 9 a nabízí následující možnosti (zleva doprava): Sestava; Průvodce sestavou; Otevřít databázový objekt; Upravit; Smazat; Přejmenovat.



Obrázek 9: Nástrojová lišta Sestava

V nabídkách hlavního okna Base jsou k dispozici možnosti ukotvení/uvolnění a uzamčení/odemčení nástrojových lišt. Ty jsou však zahrnuty pouze z důvodu konzistence s nástrojovými lištami v jiných součástech LibreOffice a pro Base nejsou obvykle potřeba.

Výchozí konfigurace nástrojových lišt bude pravděpodobně vyhovovat většině uživatelů aplikace Base. Nástrojové lišty však můžeme přizpůsobit pomocí karty *Nástrojové lišty* dialogového okna **Nástroje > Přizpůsobit** – podrobnosti o této funkci najdeme v systémové nápovědě.

Stavový řádek

Stavový řádek v dolní části pracovní plochy poskytuje informace o otevřené databázi, přičemž zobrazené informace závisí na typu dané databáze. Například obrázek 10 zobrazuje stavový řádek, když byla otevřená databáze vytvořena pomocí vestavěného databázového stroje HSQLDB.

Vestavěná databáze	Vestavěná HSQLDB		
--------------------	------------------	--	--

Obrázek 10: Stavový řádek databáze vytvořené pomocí vestavěné HSQLDB

Obrázek 11 zobrazuje stavový řádek, pokud je otevřenou databází databáze MySQL. Program Base se připojil k databázi s názvem *sakila* (vzorová databáze dodávaná jako součást distribuce MySQL), která je dostupná prostřednictvím místního serveru MySQL (název připojení *localhost*, přihlášení pod uživatelským jménem *root*).

Konektor MySQL	Libretest	Roman	localhost
----------------	-----------	-------	-----------

Obrázek 11: Stavový řádek pro databázi MySQL

Oblast databáze

Oblast *Databáze* obsahuje čtyři tlačítka pojmenované **Tabulky**, **Dotazy**, **Formuláře** a **Sestavy** (stejně možnosti jsou dostupné pomocí nabídky **Zobrazit** v hlavní nabídce). Pomocí těchto tlačítek můžeme vybrat pracovní oblast, která se týká aktuálního úkolu. Alternativní klávesovou zkratkou pro akci některého z tlačítek v oblasti *Databáze* je stisknutí klávesy *Alt* spolu s podtrženým písmenem v legendě tlačítka (například stisknutí kláves *Alt + D* na klávesnici odpovídá kliknutí myší na tlačítko **Dotazy**).

V závislosti na zvolené pracovní oblasti lze provádět různé úlohy – vytvářet nový obsah nebo vyvolávat existující prvky. Při různých výběrech v oblasti *Databáze* aktualizuje Base obsah obou oblastí vpravo tak, aby odrážel novou pracovní oblast.

Ačkoli jsou tabulky jádrem každé databáze, při otevření existující databáze je výchozím nastavením databáze Base pracovní oblast formulářů, protože formuláře jsou prvky, které se při práci s databázemi používají nejčastěji. Pomocí formulářů můžeme provádět záznamy do tabulek a analyzovat obsah tabulek.

Oblast úkolů

Oblast *Úlohy* nabízí několik možností pro výběr běžných akcí ve vybrané pracovní oblasti. Možnosti pro jednotlivé pracovní oblasti jsou následující:

Tabulky

- Vytvořit tabulku v režimu návrhu...
- Použití průvodce pro vytvoření tabulky...
- Vytvořit pohled...

Dotazy

- Vytvořit dotaz v režimu návrhu...
- Použití průvodce pro vytvoření dotazu...
- Vytvořit dotaz v režimu SQL...

Formuláře

- Vytvořit formulář v režimu návrhu...
- Použití průvodce pro vytvoření formuláře...

Sestavy

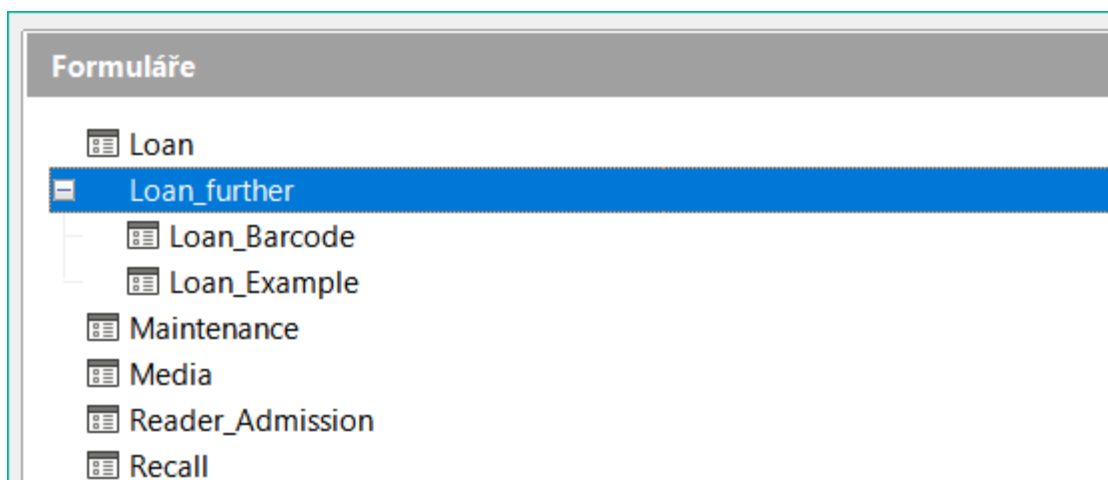
- Vytvořit sestavu v režimu návrhu...
- Použití průvodce pro vytvoření sestavy...

Napravo od těchto odkazů se nachází oblast s názvem *Popis*. Při přejetí myší nad některou z dostupných možností se tato oblast aktualizuje a zobrazí se stručný textový popis označené možnosti.

Oblast Tabulky / Dotazy / Formuláře / Sestavy

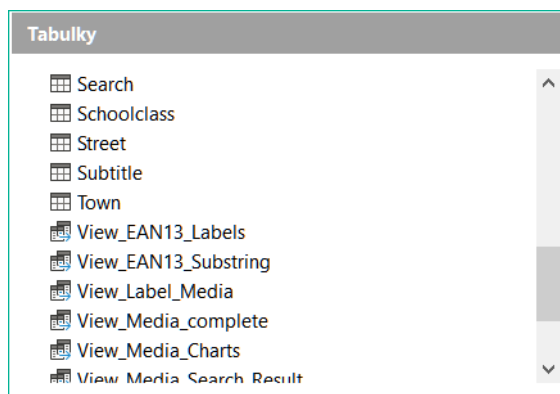
Název v horní části oblasti umístěné pod oblastí *Úlohy* se změní tak, aby odrážel zvolenou pracovní oblast, a je tedy jedním z *Tabulky*, *Dotazy*, *Formuláře* nebo *Sestavy*.

V levé části oblasti jsou uvedeny definované prvky příslušného typu. Pro pracovní oblasti *Tabulky* a *Dotazy* je to abecední seznam dostupných prvků. V pracovních oblastech formulářů a sestav jsou příslušné prvky uspořádány abecedně ve struktuře složek (obrázek 12). Toho lze dosáhnout buď přímo při ukládání prvků pomocí dialogového okna *Uložit*, nebo vytvořením nových složek pomocí **Vložit > Složka** v hlavní nabídce.



Obrázek 12: Struktura složek v oblasti Formuláře

Pokud databáze obsahuje pohledy, jsou v pracovní oblasti *Tabulky* uvedeny společně s tabulkami. Oba typy se liší použitím různých ikon. Obrázek 13 zobrazuje tabulky i pohledy v databázi *Media_without_Macros.odt*.



Obrázek 13: Oblast tabulek včetně pohledů

V pravé části oblasti jsou uvedeny další informace, které se ovládají pomocí rozbalovací nabídky se dvěma nebo třemi možnostmi v závislosti na vybrané pracovní oblasti.

Pro všechny pracovní oblasti lze vybrat možnost **Žádný**, pokud nechceme obrazovku zahlcovat dalšími informacemi.

Pro všechny pracovní oblasti lze zvolit možnost **Dokument**, která zobrazí náhled vybraného prvku, což může pomoci při identifikaci prvku, který nás zajímá. Obrázek 14 například zobrazuje náhled tabulky *Loan* databáze *Media_without_Macros.odt*.

ID	Media_ID	Reader_ID	Loan_Date	Return_Date	Extension
0	8	4	01.12.2021	20.12.2021	
1	10	4	05.12.2021		3
2	8	4	25.11.2021		
3	2	8	06.12.2021	21.12.2021	0
4	6	2	07.11.2021		
5	4	9	07.12.2021	20.12.2021	
6	7	9	08.12.2021	20.12.2021	
7	3	5	09.12.2021	21.12.2021	
8	5	1	10.12.2021		
9	11	7	11.12.2021		
10	1	3	13.12.2021		

Obrázek 14: Náhled dokumentu v oblasti Tabulky

Pro pracovní oblasti formulářů a sestav je k dispozici další možnost **Informace o dokumentu**. Obrázek 15 zobrazuje příklad datum a název informačního poskytovatele pro formulář *Maintenance* databáze *Media_without_Macros.odt*.

Formuláře

Informace o dokumentu

Autor:
Robert Großkopf

Datum:
09.04.2012, 08:52:29

Změnil:
Robert Großkopf

Změněno:
14.10.2012, 14:49:38

Obrázek 15: Informace o dokumentu v oblasti Formuláře

Base – kontejner pro databázový obsah

Zadávání dat pomocí formulářů

Formulář je prvek, který slouží k zadávání, zobrazování nebo úpravám dat v databázi. Přestože je možné zadávat data přímo do tabulek, u rozsáhlých tabulek může být někdy obtížné zajistit, aby byla data umístěna na správné místo. Efektivní formulář může urychlit a zjednodušit práci uživatele a snížit možnost zadání chybných údajů.

Chceme-li otevřít formulář, klikneme na **Formuláře** v oblasti *Databáze* v hlavním okně Base, klikneme pravým tlačítkem myši na název formuláře v oblasti *Formuláře* a v místní nabídce vybereme **Otevřít**. Ve všech komponentách LibreOffice je běžné, že uživatelské rozhraní poskytuje více metod k dosažení stejného cíle. Například po kliknutí na **Formuláře** v oblasti *Databáze* hlavního okna Base nabízí Base následující alternativní metody pro otevření konkrétního formuláře:

- 1) Dvakrát klikneme na název formuláře v oblasti *Formuláře*.
- 2) Klikneme na název formuláře v oblasti *Formuláře* a poté klikneme na ikonu **Otevřít databázový objekt** na nástrojové liště *Formulář*.
- 3) Klikneme na název formuláře v oblasti *Formuláře* a poté v hlavní nabídce vybereme možnost **Úpravy > Otevřít databázový objekt**.

✓ Poznámka

Abychom se vyhnuli zbytečnému opakování, bude ve zbytku této knihy obvykle uvedena pouze jedna z metod uživatelského rozhraní, která slouží k dosažení určitého cíle. Popsaná metoda není zamýšlena jako doporučení „nejlepšího postupu“ – to je věcí osobních preferencí.

Jednoduchý formulář může zobrazovat informace pouze z jedné tabulky, složitější formuláře jich mohou obsahovat mnohem více. Obrázek 16 ukazuje formulář *Loan* v ukázkové databázi *Media_without_Macros.odt*, který byl rozšířen, aby zobrazoval informaci z více tabulek a obsahoval dodatečné kontroly.

First Name	Last Name	ID
Jacob	Bartlett	8
Derek	Byers	2
William	Costa	9
Daisy	Gilmore	4

Medium	Loan Date
7 - The Road - by McCarthy, Cormac	14.11.22

Medium	Loan Date	Return Date	Extension	Loan Days	Balance Time
8 - Harry Potter and the Philosopher's Stone - by Rowling, JK	10.11.22			4 Days	10 Days
10 - Northern Lights - by Pulman, Philip	05.11.22		3	9 Days	26 Days

Obrázek 16: Formulář *Loan*

Následující poznámky upozorňují na některé vlastnosti formuláře *Loan*:

- Posuvný seznam registrovaných výpůjček v horní části formuláře odráží informace v tabulce *Reader*, seřazené nejprve podle příjmení a poté podle jména.
- Počet záznamů v seznamu vypůjčitelů lze v případě potřeby snížit, aby byl přehlednější. Pokud uživatel zadá písmeno „M“ do pole *Filter (Last Name)* v pravé části seznamu dlužníků a poté klikne na tlačítko **OK**, budou do seznamu zahrnuti pouze osoby, jejichž příjmení začíná na „M“. Podobně pokud uživatel zadá písmena „Mc“ a poté klikne na tlačítko **OK**, zobrazí se pouze osoby, jejichž příjmení začíná na „Mc“.
- Nové informace o dlužníkovi lze zadat přímo do polí seznamu vypůjčitelů. Všimneme si však, že takto registrovaný nový uživatel knihovny si nemůže vypůjčit mediální položky – správce knihovny by musel provést další akci, aby zrušil příznak uzamčení v tabulce *Reader*.
- Podrobnosti o předmětech, které mají být vypůjčeny, lze uvést v prostřední části formuláře. Po výběru příslušné položky v seznamu výpůjček se formulář aktualizuje a zobrazí se řetězec „Loan for Reader (Vypůjčka pro čtenáře)“ následovaný jménem vybraného vypůjčitele, přičemž obě položky jsou napsány zvýrazněným textem. Pokud je dříve vypůjčená položka po termínu a musí být vrácena před povolením dalších

vypůjček, je prostřední oblast uzamčena (není možný žádný vstup) a za jméno vypůjčitele je připojen řetězec „Loan temporarily locked! (Vypůjčka dočasně uzamčena!)“. Položky, které jsou již zapůjčeny, jsou uvedeny ve spodní části formuláře.

- *Loan Date (Datum vypůjčky)* se automaticky nastaví na aktuální datum. Položky, které je možné si zapůjčit jsou zobrazeny v rozbalovacím seznamu vlevo od tlačítka **Refresh**, seřazeno dle pole *ID* (*ID* je pole v tabulce *Media*). Položky, které jsou již vypůjčeny některému čtenáři, nejsou v této nabídce k dispozici.
- Položky médií vybrané k vypůjčce se přidají do oblasti *Current Loan (Aktuální vypůjčka)* kliknutím na tlačítko **Refresh**.
- Ve spodní části formuláře (označené *Return*) není možné odstranit datový řádek. Upravovat lze pouze pole *Return Date (Datum vrácení)* a *Extension*. Pokud byl vypůjčitel dříve zablokovan a následně vrátil opožděnou položku(y), lze prostřední oblast odemknout kliknutím na tlačítko **Refresh**.

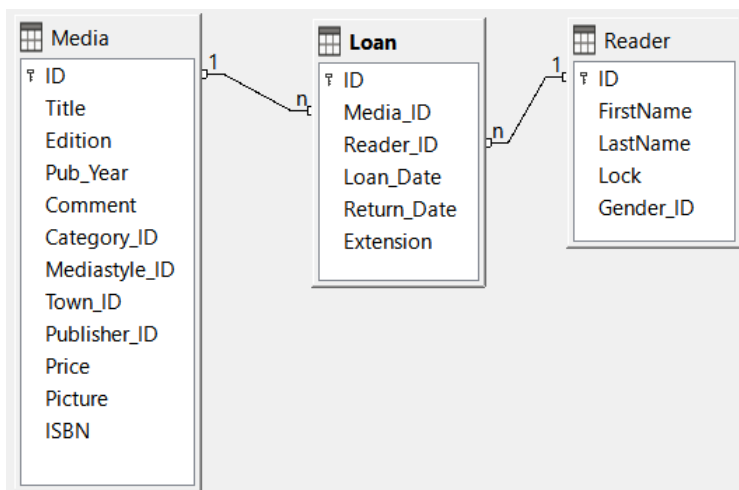
Všechny funkce formuláře *Loan* v této ukázkové databázi se provádějí bez použití maker.

Zadávání dat přímo do tabulky – základy pro zadávání dat

Tabulky jsou databázové prvky, které obsahují data v databázi uspořádaná do řádků a sloupců tvořících mřížku. Důležitou součástí návrhu databáze je definování vazeb mezi jednotlivými tabulkami v databázi. Vztah mezi dvěma tabulkami je vytvořen pomocí polí, která jsou společná pro obě tabulky.

Chceme-li zobrazit vztahy definované pro ukázkovou databázi *Media_without_Macros.odt*, vybereme na panelu nabídek možnost **Nástroje > Relace**. Base zobrazí okno Návrh relace, které graficky znázorňuje relace pomocí čar mezi tabulkami spojujícími pole z každé z nich. Na první pohled může být toto schéma skličující, a proto je vhodné zaměřit se jen na jeho část.

Tabulka *Loan* přímo souvisí s tabulkami *Media* a *Reader*, jak je znázorněno na obrázku 17. V obou případech se jedná o relaci jeden k mnoha (nebo mnoho k jednomu), jak je uvedeno v písmenech „1“ a „n“ na obou koncích spojovací čáry. To odráží skutečnost, že jeden čtenář si může vypůjčit více mediálních položek najednou a že jedna mediální položka může být vypůjčena vícekrát.



Obrázek 17: Relace mezi tabulkami *Loan* a *Media* a *Reader*

✓ Poznámka

Tabulky *Loan*, *Media*, a *Reader* mají každá relace s dalšími tabulkami, ale pro přehlednost jsou na obrázku 17 záměrně vynechány.

Při výpůjčce knihy se do tabulky *Loan* místo jejího názvu uloží pouze jedno číslo do pole *Media_ID*. V poli *ID* tabulky *Media* je uložen jedinečný identifikátor každého záznamu této tabulky. Toto pole je klíčovým polem tabulky *Media*: primární klíč.

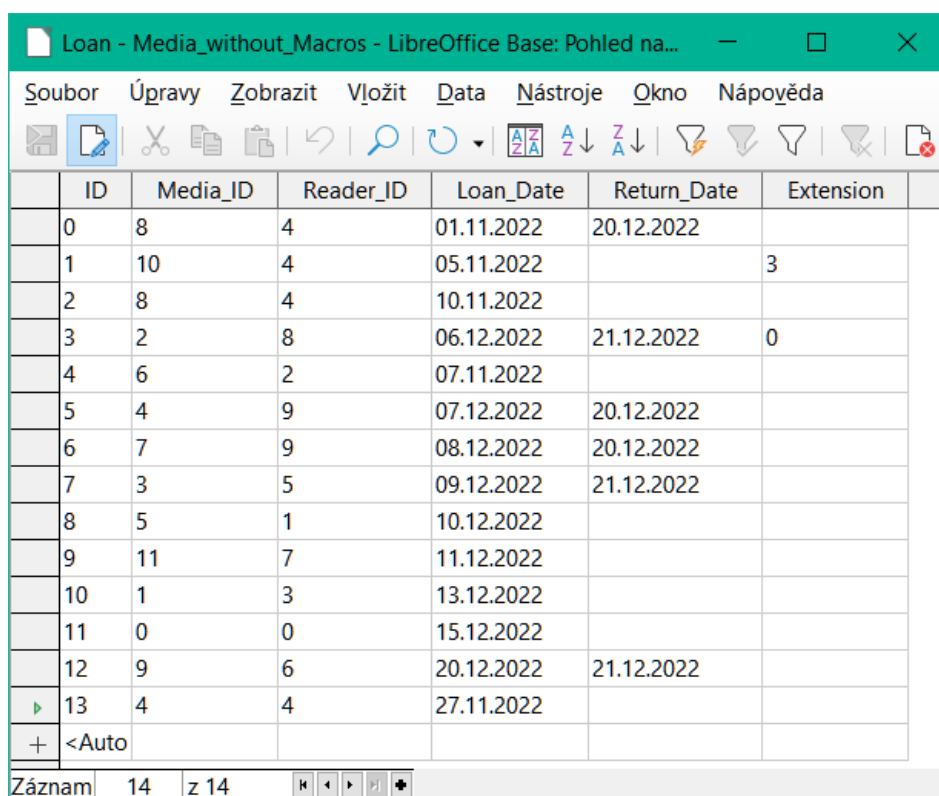
✓ Poznámka

Primární klíč jednoznačně určuje hodnoty jednotlivých polí v každém záznamu tabulky. Při výpůjčce položky se tedy číslo zadané do pole *Media_ID* v tabulce *Loan* shoduje s číslem v poli *ID* tabulky *Media*, které identifikuje záznam obsahující informace o vypůjčené položce.

Jméno vypůjčitele se do tabulky *Loan* nezadává pokaždé – tato informace se ukládá do tabulky *Reader*. Tabulka má také pole primárního klíče, které identifikuje každou osobu, která si předmět vypůjčila. Hodnotu tohoto pole pak lze zadat do tabulky *Loan*, přičemž pole *Reader_ID* identifikuje konkrétní osobu.

Vztahy mezi tabulkami mají tu výhodu, že se značně omezí práce s příslušným formulářem. Namísto bezchybného zadávání názvu média a jména a příjmení je lze zadat výběrem správných čísel pro pole *Media_ID* a *Reader_ID*, což umožní výběr správných položek média a jmen a příjmení. Nakonec je možné si stejnou mediální položku později vypůjčit znovu a tentýž čtenář si může při každé výpůjčce akci vypůjčit několik dalších mediálních jednotek.

Obrázek 18 zobrazuje tabulku *Loan* v databázi *Media_without_Macros.odt*. Chceme-li otevřít tabulku, klikneme v hlavním okně Base na **Tabulky** v oblasti *Databáze*, klikneme pravým tlačítkem myši na název tabulky v oblasti *Tabulky* a v místní nabídce vybereme **Otevřít**.



	ID	Media_ID	Reader_ID	Loan_Date	Return_Date	Extension
	0	8	4	01.11.2022	20.12.2022	
	1	10	4	05.11.2022		3
	2	8	4	10.11.2022		
	3	2	8	06.12.2022	21.12.2022	0
	4	6	2	07.11.2022		
	5	4	9	07.12.2022	20.12.2022	
	6	7	9	08.12.2022	20.12.2022	
	7	3	5	09.12.2022	21.12.2022	
	8	5	1	10.12.2022		
	9	11	7	11.12.2022		
	10	1	3	13.12.2022		
	11	0	0	15.12.2022		
	12	9	6	20.12.2022	21.12.2022	
	▶ 13	4	4	27.11.2022		
	+ <Auto					

Obrázek 18: Datová struktura tabulky *Loan*

Do řádků a sloupců tabulky *Loan* lze přímo zadávat stejné údaje, jaké byly dříve zadány pomocí formuláře. Vztahy mezi touto tabulkou a ostatními tabulkami databáze se používají ve formuláři.

- Nejdůležitějším polem je primární klíč (*ID*). Ten se generuje automaticky a zobrazuje nepostradatelný, jedinečný obsah většiny databází. Více informací o tomto tématu nalezneme v části "Vztahy mezi tabulkami" v kapitole 3, Tabulky.

- Druhé pole, *Media_ID*, ukládá hodnoty primárního klíče tabulky *Media*. Odkazuje na číslo v odpovídajícím poli *ID* v tabulce *Media*. Takový odkaz na primární klíč se nazývá cizí klíč. Ve formuláři se místo cizího klíče v seznamu zobrazí název a autor. Pole seznamu přenáší hodnotu na pozadí do cizího klíče tabulky.
- Třetí pole, *Reader_ID*, uchovává hodnoty primárního klíče tabulky *Reader*. V tomto příkladu je tento klíč pouze číslem, které odkazuje na čtenáře, který si vypůjčuje mediální položky. Ve formuláři je uvedeno příjmení a jméno čtenáře. Jak je vidět v tabulce, čtenář s primárním klíčem číslo „4“ má vypůjčeno více položek médií. Tabulka může uložit jedinečný primární klíč tabulky *Reader* jako cizí klíč *Reader_ID* mnohokrát. V žádném případě však nesmí být čtenář, který je uveden v cizím klíči tabulky *Loan*, smazán v tabulce *Reader*. Jinak by již nebylo srozumitelné, kdo si nyní média půjčuje. Databáze provádí výchozí nastavení tak, aby nebylo možné provést smazání. Odborně se tomu říká požadavek *referenční integrity*.
- Datum výpůjčky je uloženo ve čtvrtém poli (*Loan_Date*). Pokud je toto datum přítomno a neodpovídá aktuálnímu datu, zobrazí se odpovídající soubor dat pro čtenáře ve spodní tabulce formuláře pod tlačítkem **Return**.
- Čtvrté pole, *Return_Date*, ukládá datum, kdy byla mediální položka vrácena.
- Pole *Extension* obsahuje informace o prodloužení výpůjčky položky. Význam hodnot 1, 2... je vysvětlen později. Databáze obsahuje tabulku *Preferences*, která definuje počet dní na prodloužení a další související parametry.

Zadání těchto údajů umožňuje správu jednoduché knihovny médií.

Dotazy – získávání informací z dat v tabulkách

Dotaz je požadavek na načtení nebo manipulaci s daty z databáze, přičemž tato data mohou pocházet z tabulek, pohledů nebo jiných dotazů. Dotazy sdružují obsah z více tabulek v přehledném formátu. Databáze Base ukládá dotazy pomocí jazyka SQL, což je standardní jazyk pro přístup k databázím a manipulaci s nimi.

✓ Poznámka

Dotazy nejsou totéž co tabulky nebo pohledy, i když se někdy zdají být podobné.

Chceme-li otevřít dotaz, klikneme v hlavním okně Base na **Dotazy** v oblasti *Databáze*, klikneme pravým tlačítkem myši na název dotazu v oblasti *Dotazy* a v místní nabídce vybereme **Otevřít**.

	ID	Media_ID	Medium	Reader_ID	Loan_Date	Return_Da...	Extension	prolonged_for	LoanDays	BalanceTime
▶	55	3	3 - Robinson (8	8	09.01.2022			0	309	-295
	4	6	6 - Wolf Hall - 2	2	07.11.2022			0	7	7
	2	8	8 - Harry Pottt	4	10.11.2022			0	4	10
	56	7	7 - The Road - 4	4	14.11.2022			0	0	14
	1	10	10 - Northern	4	05.11.2022		3	21	9	26
	13	4	4 - Gulliver's T	4	27.11.2022			0	-13	27
	9	11	11 - No Time	7	11.12.2022			0	-27	34
	11	0	0 - The Dark S	0	15.12.2022			0	-31	38
	8	5	5 - To Kill a M	1	10.12.2022			0	-26	40
	10	1	1 - Lord of the	3	13.12.2022			0	-29	43
	54	2	2 - Animal Far	3	21.12.2022			0	-37	51
+	<Auto									

Obrázek 19: Dotaz *Media_Loan_Balance_Time*

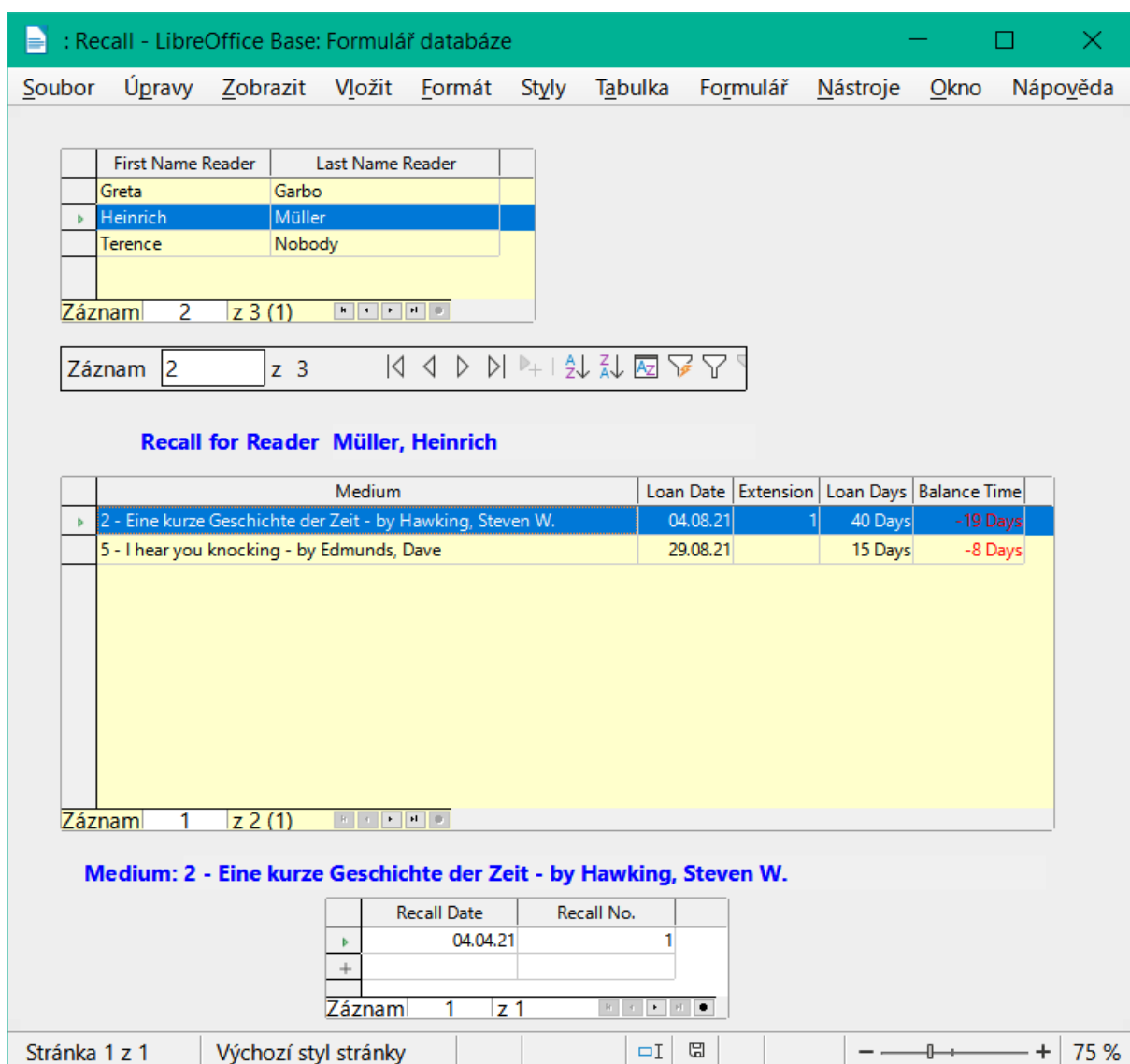
V dotazu *Media_Loan_Balance_Time* zobrazeném na obrázku 19 jsou uvedeny všechny položky médií, které jsou aktuálně vypůjčené. Vypočítá, jak dlouho je každá položka vypůjčena a jaký je zůstatek výpůjční lhůty. Na základě pole cizího klíče *Media_ID* dotaz sloučí primární klíč, název a autora položky média do jediného textového řetězce zobrazeného v poli *Medium*. Toto pole se používá ve formuláři *Loan*. Kombinovaná pole v dotazu slouží také jako spojovací pole z formuláře *Loan* do tabulky *Loan*, a to prostřednictvím polí *Media_ID* a *Reader_ID*.

- V tabulce *Loan* jsou uvedeny všechny položky médií, u kterých není zadáno datum vrácení. Dotazovací pole *Media_ID* a *Medium* poskytují dodatečné informace pro zvýšení použitelnosti.
- Odkaz na vypůjčitele se vytvoří tak, že se uvede primární klíč tabulky *Reader* (*Reader_ID*).
- Časový rozdíl ve dnech mezi datem půjčky (*Loan_Date*) a aktuálním datem je specifikován jako *LoanDays* (obrázek 19 ukazuje dotaz, který byl spuštěn 21.12.2021).
- Počet *LoanDays* se odečte od doby výpůjčky, čímž se získá zbývající počet dní výpůjční doby. Výpůjční doba se může u různých typů médií lišit a je řízena pomocí polí v tabulce *Mediastyle*.
- Hodnota 1 pro pole *Extension* odpovídá prodloužení výpůjční lhůty o 7 dní. V souboru dat na obrázku 19 mají řádky s *ID* nastavenými na 1 a 2 každý jedno prodloužení o 7 dní. Správce knihovny může přidělit více prodloužení, takže hodnoty v tomto sloupci nejsou omezeny na 1.

Sestavy – prezentace dat

Reporty umožňují načítat data z databáze a vypisovat výsledky v užitečném, uživatelsky přívětivém formátu. Databáze adresáře může být například plná údajů, jako jsou jména, adresy, telefonní čísla, data narození atd., ale můžeme chtít spustit sestavu, která bude zobrazovat pouze adresy naformátované pro tisk na štítky.

Formuláře obvykle není vhodné používat místo sestav. Vezměme si například formulář *Recall* zobrazený na obrázku 20, který zjevně není vhodný pro generování čistě naformátovaného dopisu vyzývajícího vypůjčovatele k vrácení opožděných multimediálních položek.



Obrázek 20: Formulář pro upomínku

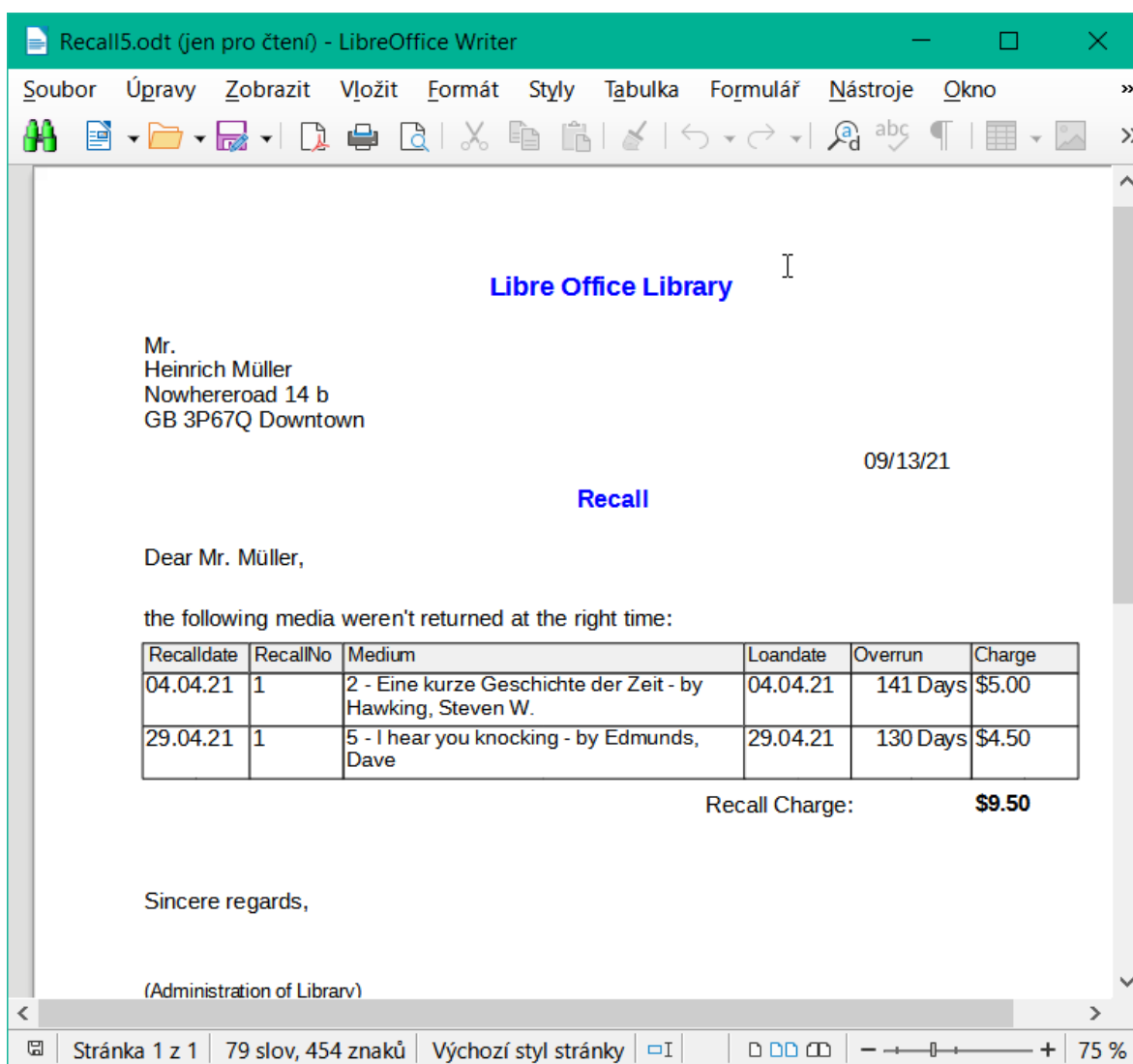
Před vytištěním sestavy formátované jako oznámení o vrácení je nutné zadat informace o upomínce do formuláře *Recall*. V tabulce ve formuláři jsou uvedeny všechny osoby, které mají vypůjčené předměty se zápornou zbývající dobou výpůjčky.

U každého média, které má být vráceno, se zadá datum vrácení a číslo oznámení upomínky. Datum upomínky je ve výchozím nastavení aktuální datum. Číslo upomínky je celé číslo, které se s každou další výzvou k vrácení pro určitého vypůjčitele/médium zvyšuje o 1.

Tento formulář v příkladu databáze *Media_without_Macros.odt* vyžaduje zadání uživatelem pro vytvoření oznámení o upomínce. Ve verzi makra (*Media_with_Macros.odt*) se automaticky zadá datum a vytiskne oznámení o upomínce.

Oznámení o upomínce (obrázek 21) je generováno pomocí dotazu z dříve zadaných dat. Uživatel databáze musí pouze vybrat sestavu *Recall* a může vytisknout dopis s upomínkou a odeslat jej všem osobám, které mají ve formuláři na předchozí straně záznam o upomínce.

Chceme-li otevřít sestavu, klikneme v hlavním okně Base na **Sestavy** v oblasti *Databáze*, klikneme pravým tlačítkem myši na název sestavy v oblasti *Sestavy* a v místní nabídce vybereme **Otevřít**.



Obrázek 21: Ukázka upomínky

V takové sestavě může být pro určitou osobu více záznamů (položek po době vrácení). Pokud tabulka obsahující položky pro tuto osobu přesahuje prostor na jedné stránce, je rozšířena o další stránku.

Taková sestava je podrobnější než dopis vytvořený pomocí LibreOffice Writer. Automaticky shromažďuje datové sady pro tisk a podle toho uspořádává potřebný doprovodný text.

Podobný dopis jako na výše uvedeném obrázku lze jinak realizovat pouze pomocí maker, jak je popsáno v kapitole „Vytvoření sestavy“ na straně 65.

Bezpečná manipulace se souborem programu Base

Tabulky, dotazy, formuláře a sestavy interní databáze HSQLDB jsou uloženy v souboru programu Base. Vzhledem k tomu, že databázový soubor je zapsán do paměti, je třeba s ním zacházet opatrně, protože obsahuje více objektů. Z hlášení chyb je zřejmé, že databázový soubor vyžaduje jen o něco opatrnější zacházení než například textový soubor napsaný v programu Writer.

Při práci se souborem programu Base je proto potřeba vzít v úvahu následující pokyny:

- Otevřený databázový soubor by neměl být uložen s jiným názvem pomocí **Soubor > Uložit jako** nebo ekvivalentní interakce. Pokud není jiná možnost, je třeba tabulky,

dotazy, formuláře a sestavy nejprve uzavřít. Databázový soubor je lepší zavřít a vytvořit jeho kopii.

- Nástroj Návrhář sestav v Base byl původně doplňkem, a přestože se již nezdá být samostatným rozšířením, funguje do značné míry nezávisle na databázovém souboru. Přejmenování databázového souboru odstraní nástroj Návrhář sestav z jeho základu.
- Uložení tabulky, dotazu, formuláře nebo sestavy neznámá, že byl uložen celý databázový soubor. Toto uložení je třeba provést samostatně. Při ukládání objektu (tabulky, dotazu, formuláře, sestavy) se informace zapíše do databázového souboru v paměti; při ukládání databázového souboru se vše, co je obsaženo v databázovém souboru v paměti, zapíše do souboru *.odb.
- Toto chování paměti platí zejména pro práci s nástrojem Report Builder. Příprava sestavy je stále nejnestabilnější složkou uvnitř souboru programu Base. Proto by se po každém kroku měla uložit jak sestava, tak soubor *.odb. Jakmile je sestava vytvořena, funguje sama o sobě bez zvláštních problémů.
- Po dokončení souboru *.odb se data přidaná do databáze zapíše pouze do databázového souboru v paměti, ale ne do souboru *.odb. Teprve když soubor *.odb zavřeme, uložíme do něj data. Obsah databáze HSQLDB bude zapsán zpět do souboru. Pád v tomto bodě může vést ke ztrátě dat. Proto by měla být vypracována strategie, aby byly záložní kopie vytvářeny včas. Kapitola 9, Makra, obsahuje makra pro automatizaci některých z těchto činností.
- Výrazně vyšší úroveň zabezpečení lze dosáhnout pomocí externích serverových databází, jako je MySQL, MariaDB nebo PostgreSQL. K nim může Base sloužit jako front end, který implementuje dotazy, formuláře a sestavy pro databázi.

Jednoduchá databáze – podrobný příklad

Přehled

Ve zbývajících částech této kapitoly jsou uvedeny pokyny k vytvoření databáze, která má pomoci při organizaci sportovních soutěží v různých disciplínách. Funkčnost této databáze je záměrně omezená, protože jediným cílem je seznámit čtenáře s aspekty uživatelského rozhraní a funkcností databáze Base. Databáze bude uložena jako *Example_Sport.odb*.

Vytvoření databáze

Vytvoření databáze a obsluha Průvodce databází jsou podrobně popsány v kapitole 2, Vytvoření databáze. Pro vytvoření databáze *Example_Sport.odb* se prozatím postupuje podle následujících kroků:

- 1) Spustíme Průvodce databází jedním z následujících způsobů:
 - a) Na úvodní obrazovce LibreOffice klikneme na možnost **Databáze** v oblasti *Vytvořit*.
 - b) V libovolné komponentě LibreOffice vybereme v hlavní nabídce možnost **Soubor > Nový > Databáze** nebo klikneme na šipku vpravo od ikony **Nový** na nástrojové liště Standardní a z rozevírací nabídky vybereme možnost **Databáze**.
 - c) V okně Base klikneme na ikonu **Nový** na nástrojové liště Standardní nebo zmáčkneme *Ctrl + N*.

Tip

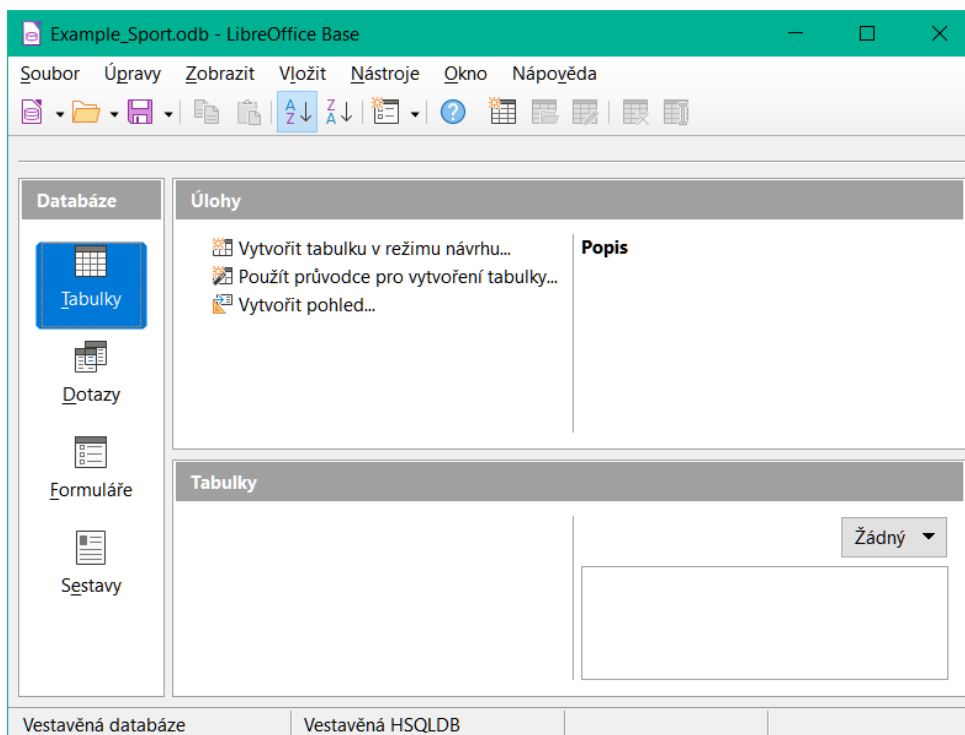
Stejně jako mnoho jiných softwarových aplikací i LibreOffice poskytuje pro určité činnosti klávesové zkratky, v tomto případě například *Ctrl + N*. Pro některé uživatele mohou zlepšit efektivitu, přesnost a přístupnost. Většina klávesových zkratk dostupných v aplikaci Base je společná pro více součástí LibreOffice. Nejlepší způsob, jak se seznámit s dostupnými klávesovými zkratkami, je používat nabídky a panely nástrojů, kdy můžeme sledovat ekvivalentní sekvenci kláves, která je uvedena v legendě napsané u možnosti nabídky nebo v nápovědě, která se zobrazí, když je ukazatel nad ikonou panelu nástrojů. Seznam klávesových zkratk můžeme zobrazit také na kartě *Klávesnice* dialogového okna **Nástroje > Přizpůsobit** (vybereme přepínač **LibreOffice**).

Abychom se vyhnuli zbytečnému opakování, nebudeme ve zbytku této knihy uvádět sekvence kláves spojené s možnostmi nabídek a panelů nástrojů.

- 2) V kroku 1 Průvodce databází vybereme možnost **Vytvořit novou databázi** a v nabídce **Vložená databáze** vybereme možnost *Vestavěná HSQLDB*.
- 3) Klikneme na tlačítko **Další >**.
- 4) V kroku 2 Průvodce databází vybereme možnost **Ne, neregistrovat databázi**, vybereme možnost **Otevřít databázi pro úpravy** a zkontrolujeme, zda není vybrána možnost **Vytvořit tabulky pomocí průvodce tabulkou**.
- 5) Klepneme na tlačítko **Dokončit**. Base zobrazí dialogové okno Uložit jako.
- 6) Přejdeme do složky, kde má být nová databáze uložena.
- 7) Do pole *Název souboru* zadáme text „Example_Sport.odb“.
- 8) Kliknutím na tlačítko **Uložit** uložíme novou databázi a zavřeme dialogové okno Uložit jako.

Tvorba tabulek

Po uložení databáze se zobrazí hlavní okno Base s vybranými *Tabulkami* v oblasti *Databáze* na levé straně okna (obrázek 22). Tabulky jsou centrálním úložištěm dat; bez tabulek by databáze neexistovala.

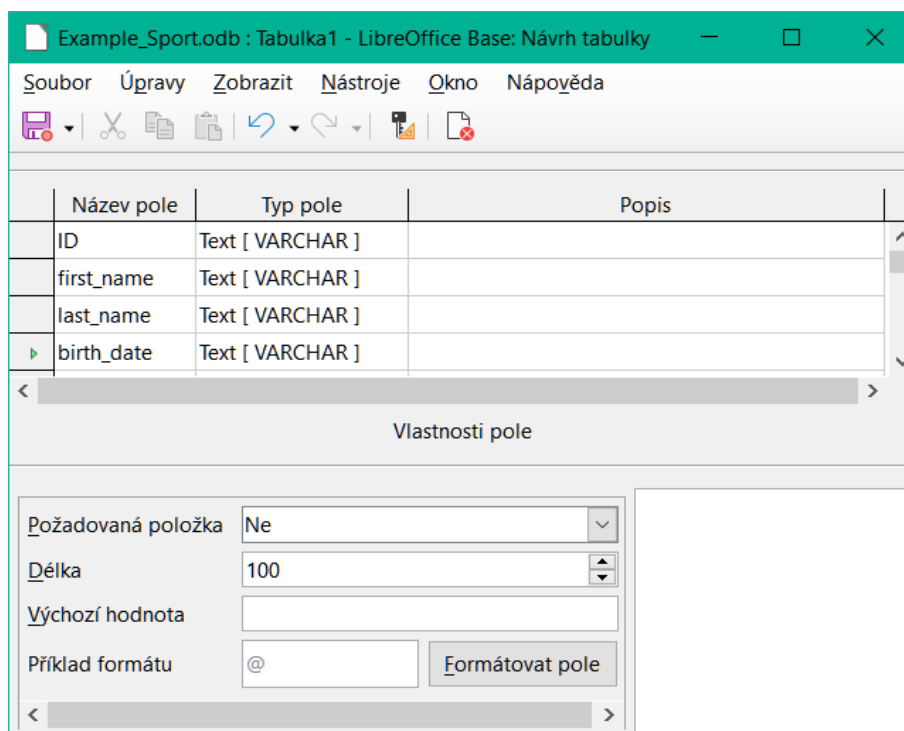


Obrázek 22: Hlavní okno základny s vybranou pracovní oblastí Tabulky

Databáze *Example_Sport.odt* bude obsahovat tři tabulky: jednu s údaji o účastnících, druhou s údaji o dostupných sportech a třetí, která určí, kterých sportů se chce každý účastník zúčastnit.

Vytvoření tabulky pro ukládání údajů o účastnících soutěže

Kliknutím na tlačítko **Vytvořit tabulku v režimu návrhu** otevřeme okno Návrh tabulky zobrazené na obrázku 23.



Obrázek 23: Okno návrhu tabulky

- 1) Prvním krokem je zadání názvů polí pro tabulku. Do tabulky budou nakonec zařazeni jak muži, tak ženy.
 - a) Název prvního pole zadáme jako „ID“. Odpovídající typ pole je ve výchozím nastavení nastaven na *Text[VARCHAR]*. Toto pole později získá hodnotu, která je pro každý záznam jedinečná. Pro vloženou databázi je nutné pole jedinečného klíče, jinak nelze do tabulky zadat žádné záznamy. Toto klíčové pole se v databázích nazývá *primární klíč*.

✓ Poznámka

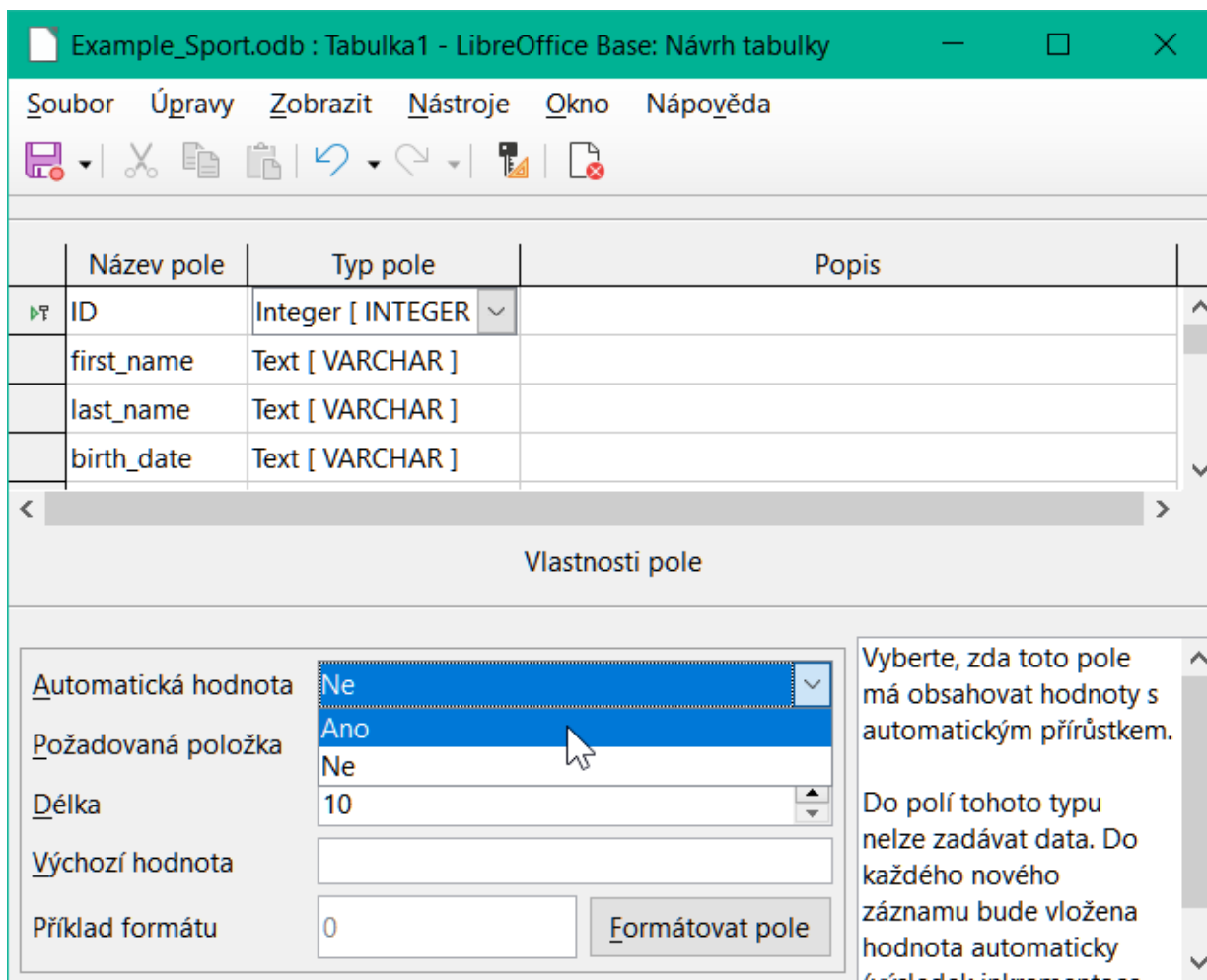
Pro tuto vlastnost lze použít jiné pole s vynecháním pole *ID*. Pokud by však bylo použito například *last_name* (příjmení), nebylo by možné uložit dvě osoby se stejným příjmením. Je možné deklarovat dvě pole společně jako jeden složený klíč, ale v této tabulce není zaručeno, že kombinace polí bude jedinečná. Proto se v tomto případě dává přednost navrženému jednoduchému provedení.

- b) Název druhého pole zadáme jako „first_name“ (jméno).
 - c) Název třetího pole zadáme jako „last_name“.
 - d) Název čtvrtého pole zadáme jako „birth_date“ (datum narození).
- 2) Dalším krokem je výběr požadovaných typů polí pro čtyři pojmenovaná pole.
 - a) Nastavíme pole *ID* na typ *Integer[INTEGER]* (obrázek 24). Tento typ pole má tu výhodu, že jej vestavěný engine HSQLDB může automaticky nastavit na nejbližší nejvyšší celé číslo.

Název pole	Typ pole
ID	Text (fix) [CHAR]
first_name	Tiny Integer [TINYINT]
last_name	BigInt [BIGINT]
birth_date	Image [LONGVARBINARY]
	Binary [VARBINARY]
	Binary (fix) [BINARY]
	Memo [LONGVARCHAR]
	Text (fix) [CHAR]
	Number [NUMERIC]
	Decimal [DECIMAL]
Požadovaná položka	Integer [INTEGER]
Délka	Small Integer [SMALLINT]
Výchozí hodnota	Float [FLOAT]
Příklad formátu	Real [REAL]
	Double [DOUBLE]
	Text [VARCHAR]
	Text [VARCHAR_IGNORECASE]
	Yes/No [BOOLEAN]
	Date [DATE]
	Time [TIME]
	Date/Time [TIMESTAMP]
	OTHER [OTHER]

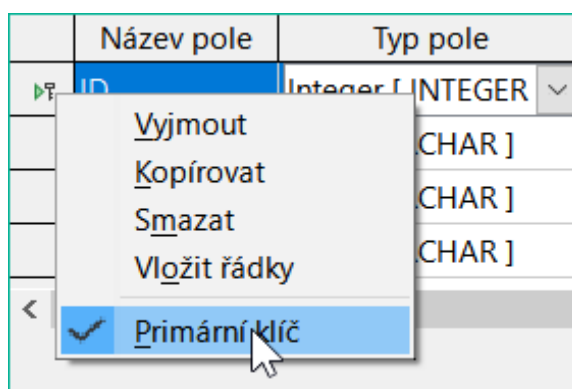
Obrázek 24: Nastavení typu pole *ID* na *Integer[INTEGER]*

- b) Pro pole *ID* nastavíme vlastnost **Automatická hodnota** na *Ano*, čímž nastavíme automatické nastavení zvyšujících se číselných hodnot (obrázek 25).



Obrázek 25: Nastavení vlastnosti Automatická hodnota pro pole ID

- c) Po nastavení *Automatické hodnoty* by se při opuštění výběru *Typu pole* měla v záhlaví řádku objevit ikona klíče. To znamená, že toto pole je primárním klíčem tabulky. Pokud není nastavena *Automatická hodnota*, lze primární klíč vybrat také v místní nabídce (obrázek 26).



Obrázek 26 Nastavení primárního klíče pomocí místní nabídky

- d) Nastavíme typ pole pro *birth_date* na *Date[DATE]* (obrázek 27). Tím je zajištěno, že budou zadávány pouze platné záznamy o datumu. Používá se také k třídění dat nebo například k výpočtu věku.

	Název pole	Typ pole
†	ID	Integer [INTEGER]
	first_name	Text [VARCHAR]
	last_name	Text [VARCHAR]
▶	birth_date	Date [DATE]

Obrázek 27: Nastavení pole *birth_date* na typ *Date[DATE]*

- e) V okně Návrh tabulky klikneme na ikonu **Uložit** na nástrojové liště Standardní. V dialogovém okně Uložit jako zadáme název „tblEntrants“ a klikneme na tlačítko **OK**. Tabulka se nyní uloží pod názvem *tblEntrants*, aby bylo možné následně zadávat data. Do pole *ID* není nutné zadávat hodnoty, protože se tak stane automaticky při uložení záznamu.

Tip

Jakékoli netriviální použití Base, zejména s makry, může vést k záměně názvů tabulek, dotazů, formulářů a sestav. Abychom zajistili jasné rozlišení, může být užitečné před každým názvem uvést krátkou zkratku, například „tbl“ (pro tabulky), „qry“ (pro dotazy), „frm“ (pro formuláře) a „rpt“ (pro sestavy).

Většina vzorových databází použitých v této knize se touto konvencí neřídí, protože byly vytvořeny před navržením pokynů. Je však vhodné přijmout tuto nebo jinou konvenci pojmenování předtím, než se pustíme do návrhu jakékoli významné databáze.

Výše uvedená konvence pojmenování je dodržena v databázi *Example_Sport.odt* popsané v této kapitole.

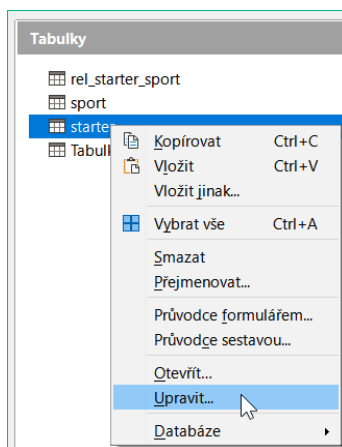
Poznámka

Databázový soubor je zazipovaná složka jednotlivých souborů. Uložení jednotlivého objektu, jako je tabulka, se tedy nezapisuje přímo do samotného databázového souboru. Proto je třeba na tlačítko **Uložit** klepnout samostatně pro uložení samotného souboru databáze i po vytvoření tabulek, dotazů, formulářů a sestav.

Pouze při opuštění datového řádku se zadaná data automaticky uloží.

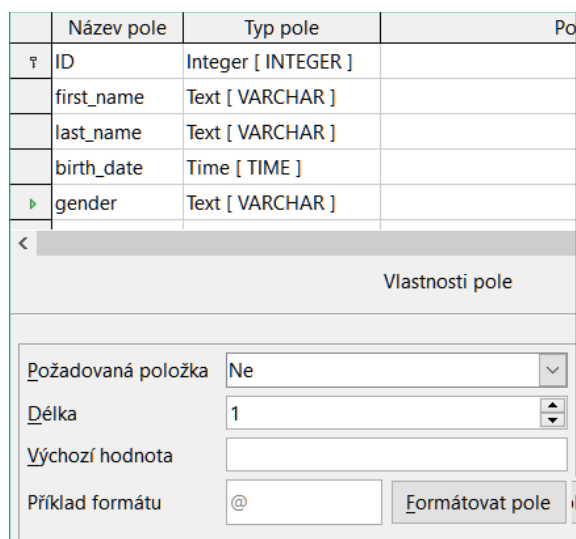
Do tabulky *tblEntrants* je nyní možné zadávat údaje o sportovcích. Nicméně jsme (záměrně!) zanedbali požadavek, aby bylo možné rozlišovat mezi závodníky a závodnicemi, takže je třeba do tabulky přidat další pole, a to následujícím způsobem:

- 1) Tabulku *tblEntrants* otevřeme pro editaci kliknutím na **Tabulky** v oblasti *Databáze* v hlavním okně Base, pravým kliknutím na název tabulky v oblasti *Tabulky* a výběrem **Upravit** v místní nabídce.



Obrázek 28: Výběr možnosti Upravit v místní nabídce

- 2) Přidání nového pole *gender* (pohlaví) do tabulky – nová pole lze přidávat pouze na konec tabulky při použití okna Návrh tabulky. Všimneme si však, že pomocí jazyka SQL je také možné přidat nová pole na určité pozice.
- 3) Necháme *Typ pole* jako *Text[VARCHAR]*, ale omezíme délku textu v tomto poli na jeden znak změnou vlastnosti pole *gender* *Délka* na 1 (Figure 29). Pro znaky 'm' (male - muž) a 'f' (female - žena) jako vstup stačí jeden znak.



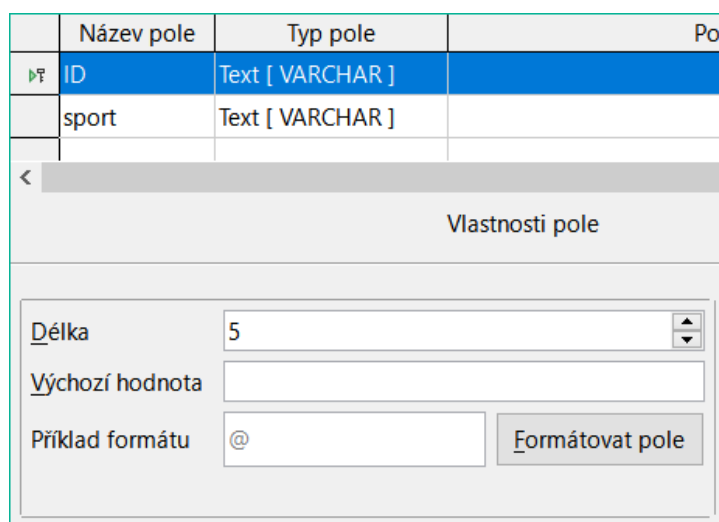
Obrázek 29: Přidání pole pohlaví do tabulky *tblEntrants*

- 4) Uložíme aktualizovanou tabulku *tblEntrants*.

Vytvoření tabulky pro uložení podrobností o sportech, které se mají hrát

Chceme-li definovat sporty, které mají být zahrnuty do soutěže, vytvoříme tabulku *tblSports* takto:

- 1) Kliknutím na **Vytvořit tabulku v režimu návrhu** v oblasti *Úlohy* hlavního okna Base znovu otevřeme okno Návrh tabulky.
- 2) Vytvoříme dvě pole s názvy *ID* a *sport_name* (název sportu). Pomocí místní nabídky, která je přístupná po kliknutí pravým tlačítkem myši na záhlaví řádku pole nastavíme *ID* jako primární klíč. Protože není mnoho různých sportů, není pro primární klíč vybrána možnost *Automatická hodnota*. Místo toho je typ pole ponechán jako *Text[VARCHAR]*, ale je omezen na pět znaků. Pro nalezení vhodné, ale jedinečné zkratky pro každý sport stačí pět znaků (obrázek 30).



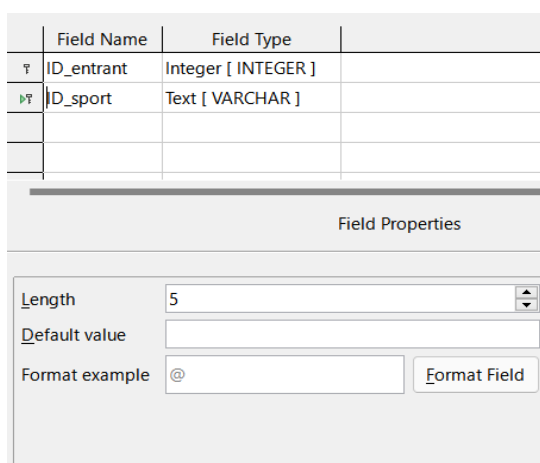
Obrázek 30: Vytvoření tabulky *tblSports*

- 3) Uložíme tabulku s názvem *tblSports*.

Vytvoření tabulky pro propojení účastníků s jejich sporty

Je zapotřebí další tabulka, která propojí tabulky *tblEntrants* a *tblSports* tak, aby každý účastník mohl být registrován v několika sportech a aby v každém sportu mohlo být registrováno více účastníků. K tomu slouží třetí tabulka, do které se ukládají hodnoty dvou primárních klíčů tabulek *tblEntrants* a *tblSports*. Protože bude uložena pouze kombinace těchto polí dohromady, jsou tato pole spojena do složeného klíče pro tuto tabulku. Provedeme následující akce:

- 1) Kliknutím na **Vytvořit tabulku v režimu návrhu** v oblasti *Úlohy* hlavního okna Base znovu otevřeme okno Návrh tabulky.
- 2) Vytvoříme dvě pole s názvy *ID_entrant* a *ID_sport*.
- 3) Nastavíme typ pole *ID_entrant* na *Integer[INTEGER]*, aby odpovídalo poli *ID* v tabulce *tblEntrants*. Neměníme typ pole *ID_sport*, protože již odpovídá poli *ID* v tabulce *tblSports*. Vlastnost *Délka* by však měla být změněna na 5 pro pole *ID_sport*.
- 4) Klikneme na záhlaví řádku pro pole *ID_entrant* a poté klikneme se zmáčknutým shiftem na záhlaví řádku pro pole *ID_sport*; tím vybereme obě pole. Klikneme pravým tlačítkem myši na záhlaví kteréhokoli řádku a poté v místní nabídce vybereme možnost **Primární klíč** pro zadání složeného klíče (obrázek 31).



Obrázek 31: Vytvoření tabulky *tblRel_entrant_sport*

- 5) Uložíme tabulku s názvem „tblRel_entrant_sport“.

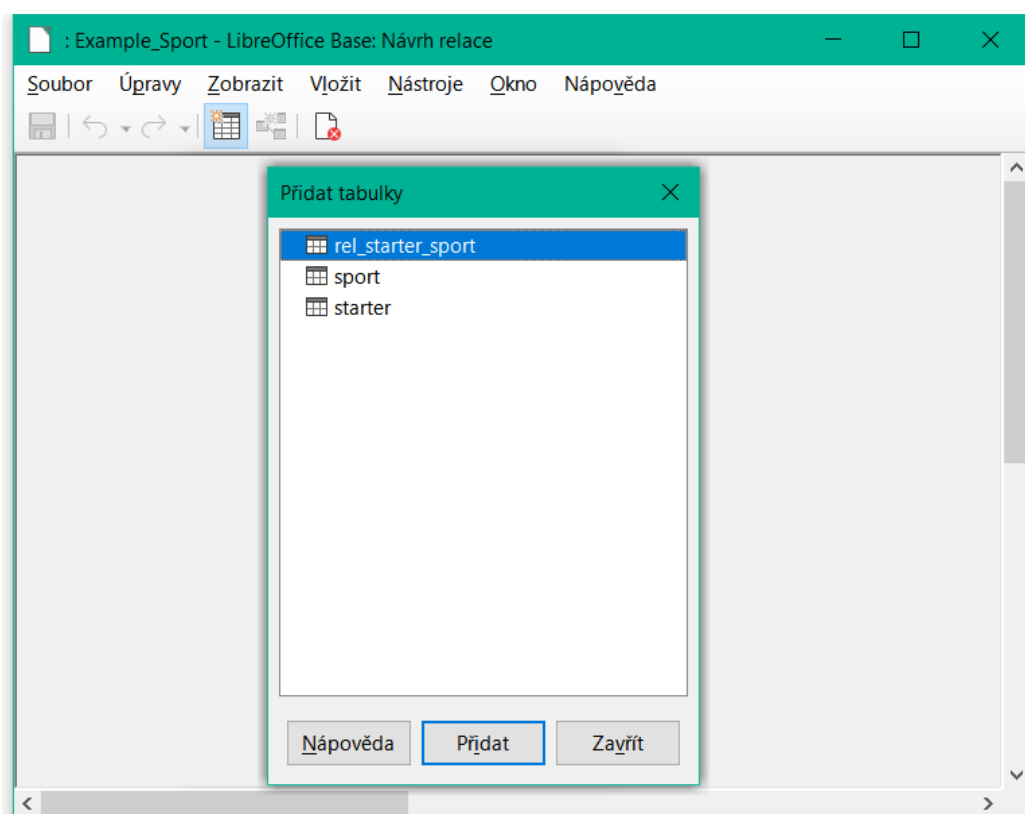
Tip

Do této tabulky lze zahrnout i výsledky soutěže. Pokud se však koná více závodů, musí být ke společnému primárnímu klíči připojeno datum závodu.

Definování relací mezi tabulkami

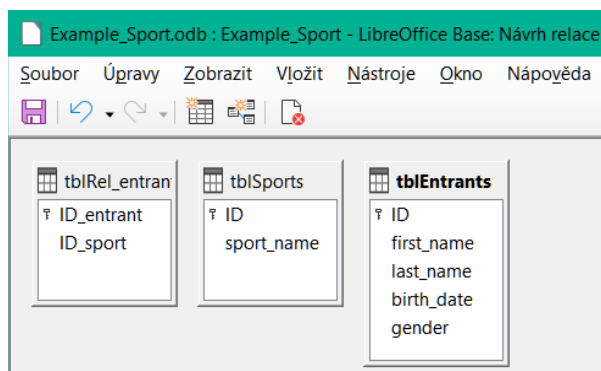
Po definování tabulek pro databázi *Example_Sport.odt* je třeba definovat relace mezi těmito tabulkami, aby bylo možné využít skutečnou sílu relační databáze. Pro definici relace jsou nutné všechny dosud vytvořené tabulky. Vztahy umožňují databázi zajistit, aby data byla platná a neopakovala se, bez možnosti neočekávané duplicity. To může zabránit tomu, aby se v tabulce *tblRel_entrant_sport* objevilo číslo účastníka, který není uveden například v tabulce *tblEntrants*. Provedeme následující kroky:

- 1) V hlavní nabídce vybereme možnost **Nástroje > Relace**.
- 2) Base otevře okno Návrh relace a dialogové okno Přidat tabulky, abychom mohli začít definovat relace (obrázek 32).



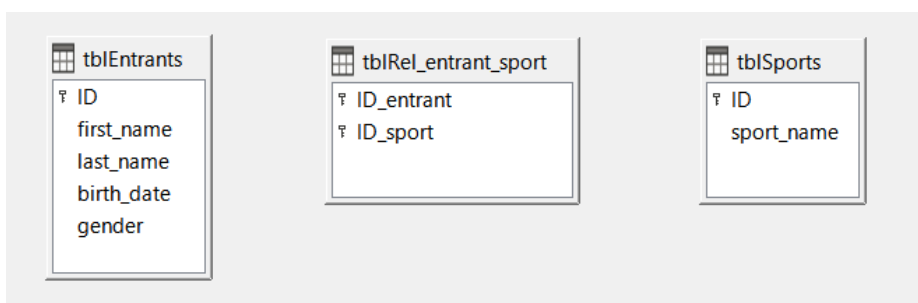
Obrázek 32: Počáteční konfigurace okna Návrh relace

- 3) Klikneme na název tabulky *tblRel_entrant_sport* a poté klikneme na tlačítko **Přidat**. Všimneme si, že se v hlavní oblasti okna objevil obdélník s názvem *tblRel_entrant_sport*.
- 4) Klikneme na název tabulky *tblSports* a poté klikneme na tlačítko **Přidat**.
- 5) Klikneme na název tabulky *tblEntrants* a poté klikneme na tlačítko **Přidat**.
- 6) Klikneme na tlačítko **Zavřít**.
- 7) Base zavře dialogové okno Přidat tabulky a ponechá tři tabulky reprezentované obdélníky v okně Návrh relace (obrázek 33). Všechna pole jsou uvedena v každé z přidaných tabulek. Pole primárního klíče jsou označena symbolem klíče.



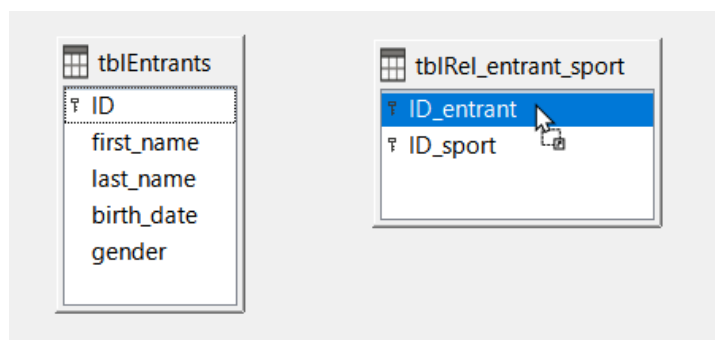
Obrázek 33: Tři tabulky přidané do okna Návrh relace

- 8) Podle potřeby přesuneme a změním velikost tabulek, abychom zlepšili rozložení okna (obrázek 34).



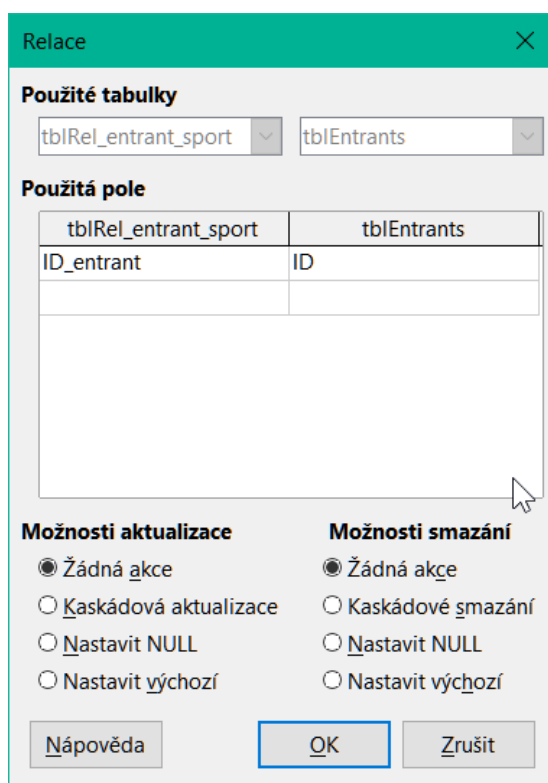
Obrázek 34: Přesun a změna velikosti tabulek

- 9) Klikneme levým tlačítkem myši na pole *ID* v tabulce *tblEntrants*. Podržíme stisknuté tlačítko myši a přesuneme ukazatel na pole *ID_entrant* v tabulce *tblRel_entrant_sport*. Kurzor změní vzhled, aby bylo viditelné, že se vytváří relace (obrázek 35). Uvolníme levé tlačítko myši.



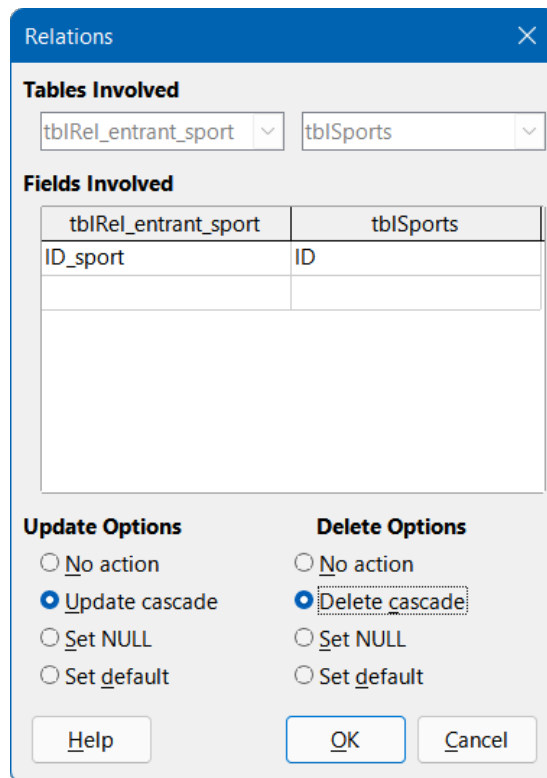
Obrázek 35: Vytvoření relace

- 10) Base zobrazí dialogové okno Relace, v němž lze podrobněji definovat povahu relace (obrázek 36).



Obrázek 36: Dialogové okno Relace –
tblRel_entrant_sport a tblEntrants

- 11) Zkontrolujte, zda je v oblasti **Možnosti aktualizace** vybrána možnost *Žádná akce* (jedná se o výchozí nastavení). Pole *ID* pro záznam v tabulce *tblEntrants* je automaticky generovaná hodnota.
- 12) V oblasti **Možnosti smazání** vyberte možnost *Kaskádové smazání*. Pokud je tato volba zapnuta, při odstranění záznamu z tabulky *tblEntrants* databázový stroj odstraní všechny záznamy pro daného účastníka také z tabulky *tblRel_entrant_sport*, aby byla zachována referenční integrita databáze bez generování chyby. Při vypnutí možnosti by odstranění účastníka mohlo vést k osiřelým záznamům v tabulce *tblRel_entrant_sport*, což by znamenalo chybový stav.
- 13) Kliknutím na tlačítko **OK** zavřeme dialogové okno Relace. Base nakreslí čáru mezi polem *ID* v tabulce *tblEntrants* (označené "1") a polem *ID_entrant* v tabulce *tblRel_entrant_sport* (označené "n").
- 14) Klikneme levým tlačítkem myši na pole *ID* v tabulce *tblSports*. Podržíme stisknuté tlačítko myši a přesuneme ukazatel na pole *ID_sport* v tabulce *tblRel_entrant_sport*. Kurzor změní vzhled na odkaz. Uvolníme levé tlačítko myši.
- 15) Base zobrazí dialogové okno Relace, ve kterém lze podrobněji definovat povahu relace.

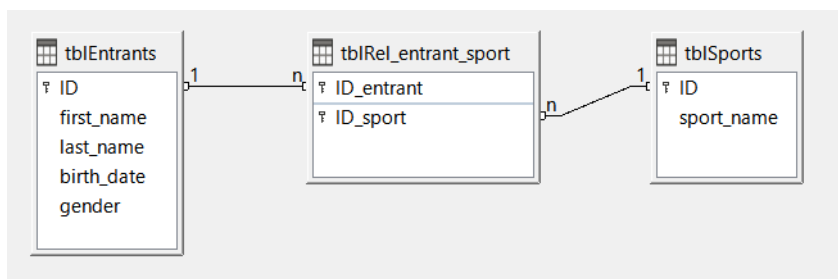


Obrázek 37: Dialogové okno Relace – *tblRel_entrant_sport* a *tblSports*

- 16) V oblasti **Možnosti aktualizace** vybereme možnost *Kaskádová aktualizace*. Pokud je tato volba zapnuta a změníme pole *ID* v tabulce *tblSports* na jiný pěti znakový řetězec, záznam v tabulce *tblRel_entrant_sport*, který má *ID_sport* nastaveno na *ID*, jež bylo změněno, se také aktualizuje, takže se neztratí vazba na správný sport. Při vypnuté volbě by změna pole *ID* sportu mohla vést k osiřelým záznamům v tabulce *tblRel_entrant_sport*, což by byl chybový stav.
- 17) V oblasti **Možnosti smazání** vyberte možnost *Kaskádové smazání*. Pokud je tato volba zapnuta, odstraní databázový stroj při mazání sportu z tabulky *tblSports* také všechny záznamy pro tento sport z tabulky *tblRel_entrant_sport*, aby byla zachována referenční integrita databáze bez generování chyby. Při vypnuté možnosti by odstranění sportu mohlo vést k osiřelým záznamům v tabulce *tblRel_entrant_sport*, což by znamenalo chybový stav.
- 18) Kliknutím na tlačítko **OK** zavřeme dialogové okno Relace. Base nakreslí čáru mezi polem *ID* v tabulce *tblSports* (označeno "1") a polem *ID_sport* v tabulce *tblRel_entrant_sport* (označeno "n").
- 19) Uložíme aktualizovaný návrh relace.

Požadované tabulky a pole jsou nyní zcela propojeny (obrázek 38). Relace naznačují, že účastník se může v tabulce *tblRel_entrant_sport* objevit opakovaně. Sport se může ve stejné tabulce objevit i opakovaně. Daná kombinace účastníka a sportu se může v tabulce objevit pouze jednou. Ze dvou relací *1:n* nyní existuje relace *n:m* prostřednictvím zasahující tabulky *tblRel_entrant_sport*.

Takový návrh tabulek může být únavný, pokud je třeba databázi naplnit psaním obsahu do tabulek. Vyžaduje otevření všech tří tabulek, když má být účastník zařazen do sportu. *ID* účastníka musí být vyhledáno v tabulce *tblEntrants* a zadáno do pole *ID_entrant* v tabulce *tblRel_entrant_sport*. Obdobně musí být vyhledáno *ID* sportu v tabulce *tblSports* a zadáno do tabulky *tblRel_entrant_sport*. Je to příliš složité a formulář by tento problém vyřešil elegantněji.



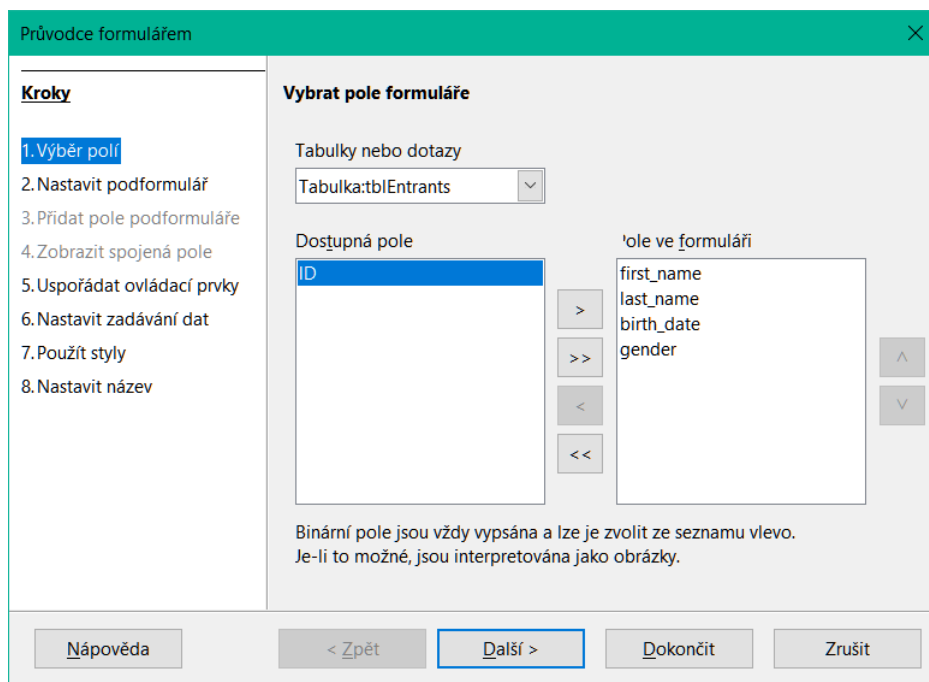
Obrázek 38: Relace mezi třemi tabulkami

Tvorba formuláře pro zadávání dat

Formuláře lze vytvářet přímo v zobrazení návrhu nebo pomocí průvodce. Dokonce i zkušený uživatelé zjistí, že mohou rychle použít průvodce a poté přizpůsobit jeho výstup tak, aby vytvořil formulář, který vyhovuje jejich potřebám. Tento přístup může často ušetřit čas.

Chceme-li vytvořit formulář pomocí Průvodce formulářem, provedeme následující kroky.

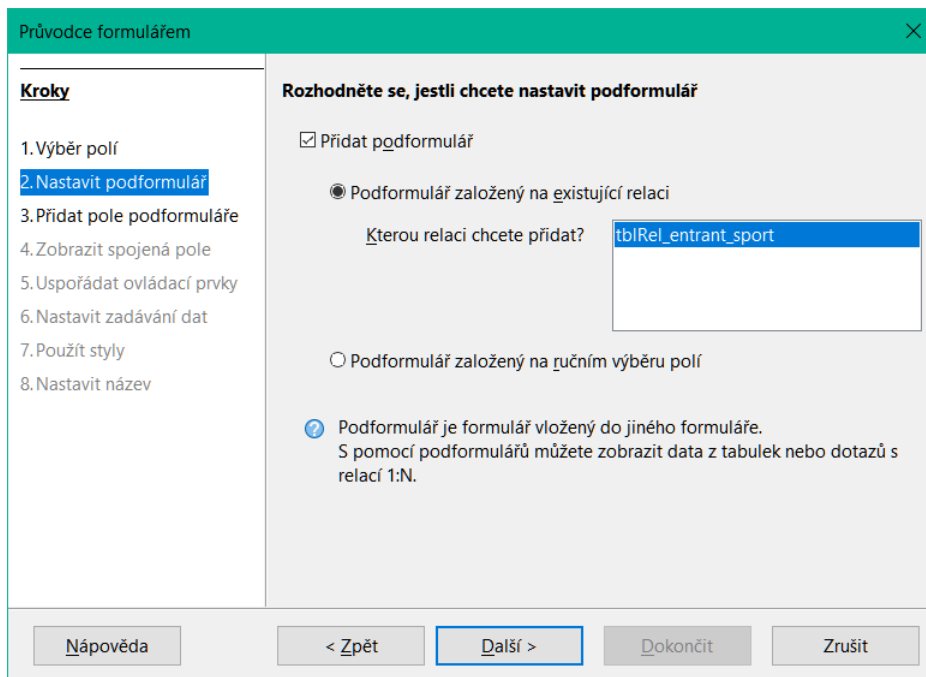
- 1) V hlavním okně Base klikneme na **Formuláře** v oblasti *Databáze*.
- 2) V oblasti *Úlohy* klikneme na **Použít průvodce pro vytvoření formuláře**.
- 3) Base otevře okno Databázový formulář a dialogové okno Průvodce formulářem, abychom mohli začít formulář vytvářet. Průvodce formulářem se otevře v kroku 1 z 8. Při provádění kroků Průvodce formulářem můžeme v základním okně Formulář databáze zobrazit součásti formuláře.
- 4) V rozevírací nabídce *Tabulky nebo dotazy* vybereme položku pro tabulku *tblEntrants*. Data tabulky *tblEntrants* by měla být zapsána v hlavním formuláři. Data z tabulky *tblSports* se načítají přímo s několika málo potřebnými sporty a aktualizují se jen zřídka.
- 5) Pomocí tlačítek se šipkou doprava přesuneme všechny položky kromě *ID* v oblasti *Dostupná pole* do oblasti *Pole ve formuláři* (obrázek 39). Všechna pole kromě pole primárního klíče (*ID*) jsou z tabulky *tblEntrants* potřebná. Pole primárního klíče je automaticky vyplněno jedinečnou hodnotou.



Obrázek 39: Průvodce formulářem krok 1

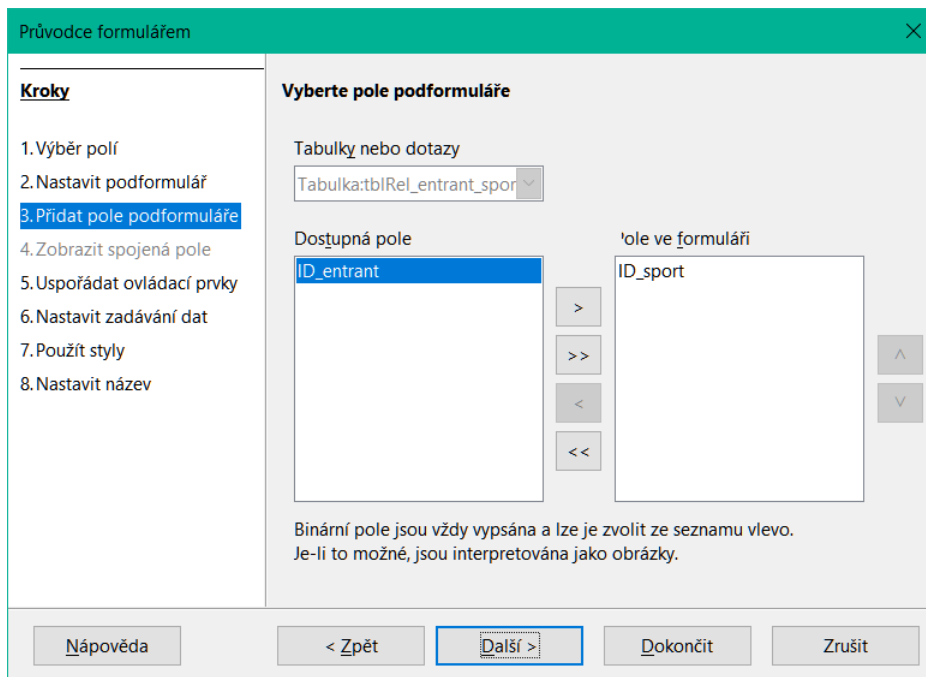
- 6) Klikneme na tlačítko **Další >**. Base zobrazí krok 2 Průvodce formulářem.

- 7) Vybereme **Přidat podformulář** a **Podformulář založený na existující relaci**. Měl by být vytvořen podformulář, kde je možné přiřadit sport k účastníkovi.
- 8) Vybereme položku *tblRel_entrant_sport* vedle popisku *Kterou relaci chcete přidat?* (obrázek 40). To potvrzuje dříve definovanou relaci.



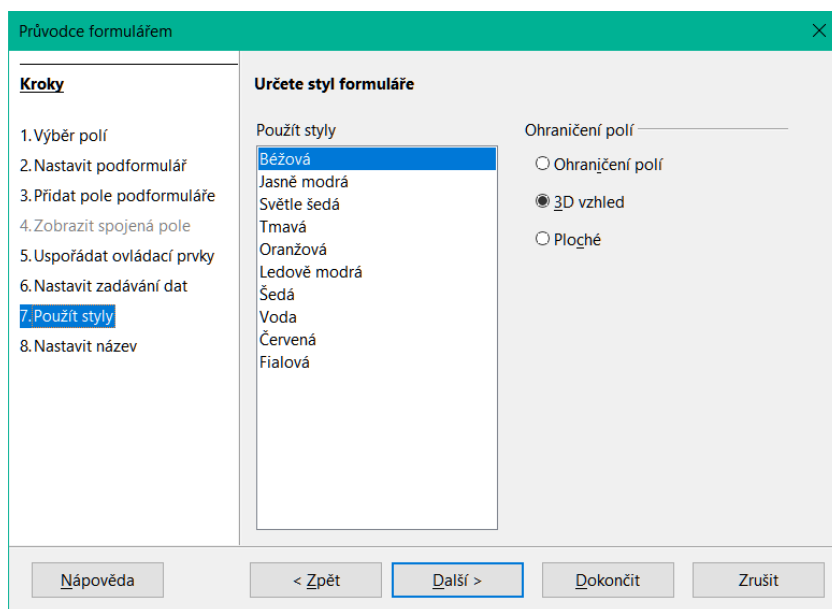
Obrázek 40: Krok 2 Průvodce formulářem

- 9) Klikneme na tlačítko **Další >**. Base zobrazí krok 3 Průvodce formulářem.
- 10) Pomocí tlačítek se šipkou doprava přesuneme pole *ID_sport* z oblasti *Dostupná pole* do oblasti *Pole ve formuláři* (obrázek 41). Primární klíč v tabulce *tblEntrants* poskytuje hodnotu pro pole *ID_entrant* pro aktuální záznam prostřednictvím připojení hlavního formuláře k podformuláři.



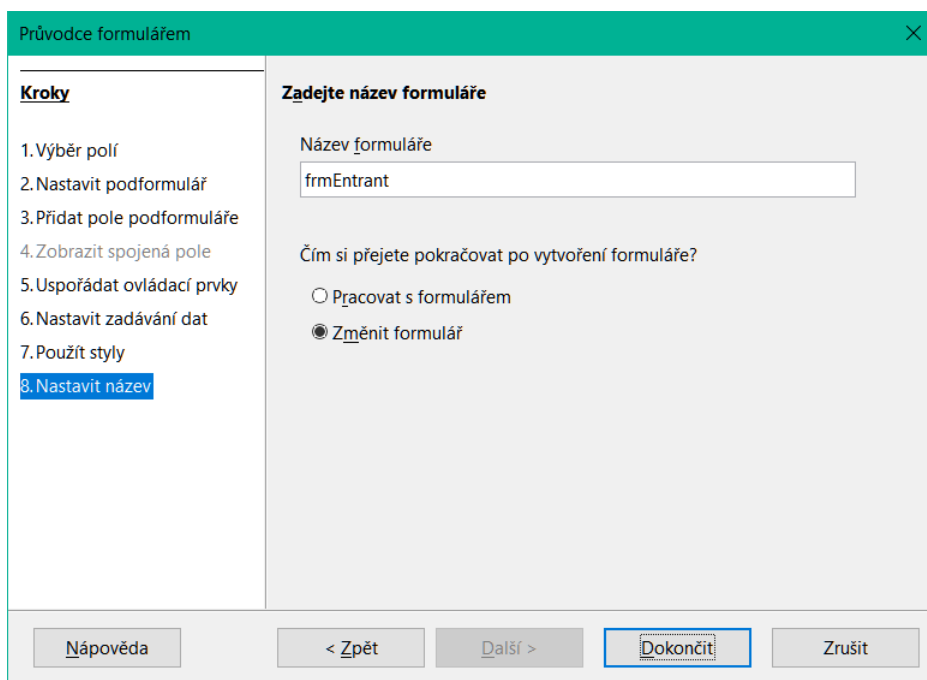
Obrázek 41: Krok 3 průvodce formulářem

- 11) Klikneme na tlačítko **Další >**. Base zobrazí krok 5 Průvodce formulářem. Krok 4 (*Zobrazit spojená pole*) je neaktivní, protože vazba již existuje.
- 12) Klikneme na druhou ze čtyř ikon pod označením *Uspořádání hlavního formuláře*. Toto uspořádání se nazývá *Sloupce - popisky v záhlaví*. Pro tento příklad není příliš důležité, jak jsou položky v hlavním formuláři a podformuláři uspořádány. Rozložení by však mělo být vždy přehledné a uživatelsky přívětivé, aby pomohlo nezkušeným uživatelům.
- 13) Klikneme na třetí ikonu ze čtyř pod označením *Uspořádání podformuláře* (obrázek 42). Toto uspořádání se nazývá *Jako tabulka*. V polích podformuláře se později zobrazí všechny sporty účastníků, proto je nejlepší ponechat výchozí nastavení podformuláře.



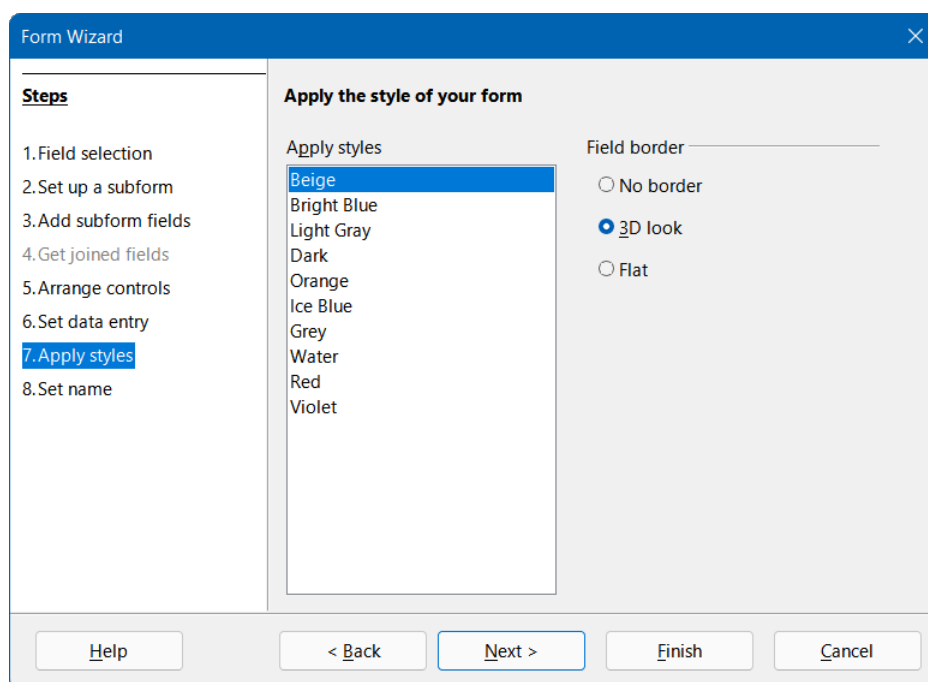
Obrázek 42: Krok 5 Průvodce formulářem

- 14) Klikneme na tlačítko **Další >**. Base zobrazí krok 6 Průvodce formulářem. Zkontrolujeme, zda je vybrána výchozí možnost **Formulář bude zobrazovat všechna data** (obrázek 43). To znamená, že lze vytvářet nové záznamy a upravovat stávající záznamy.



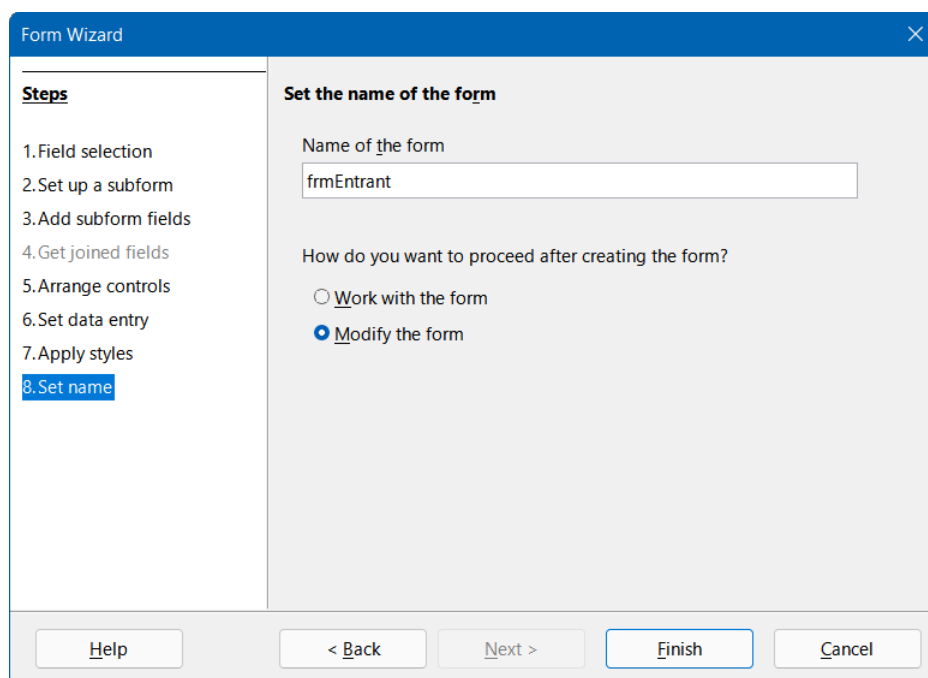
Obrázek 43: Krok 6 Průvodce formulářem

- 15) Klikneme na tlačítko **Další >**. Base zobrazí krok 7 Průvodce formulářem (obrázek 44). Použití stylů je otázkou vkusu. Jen pozor: některé styly obsahují nečekaně málo kontrastní obrázky, zejména v ovládacích polích tabulky. V takových případech může být nutné upravit barvu písma v polích datového listu.



Obrázek 44: Krok 7 Průvodce formulářem

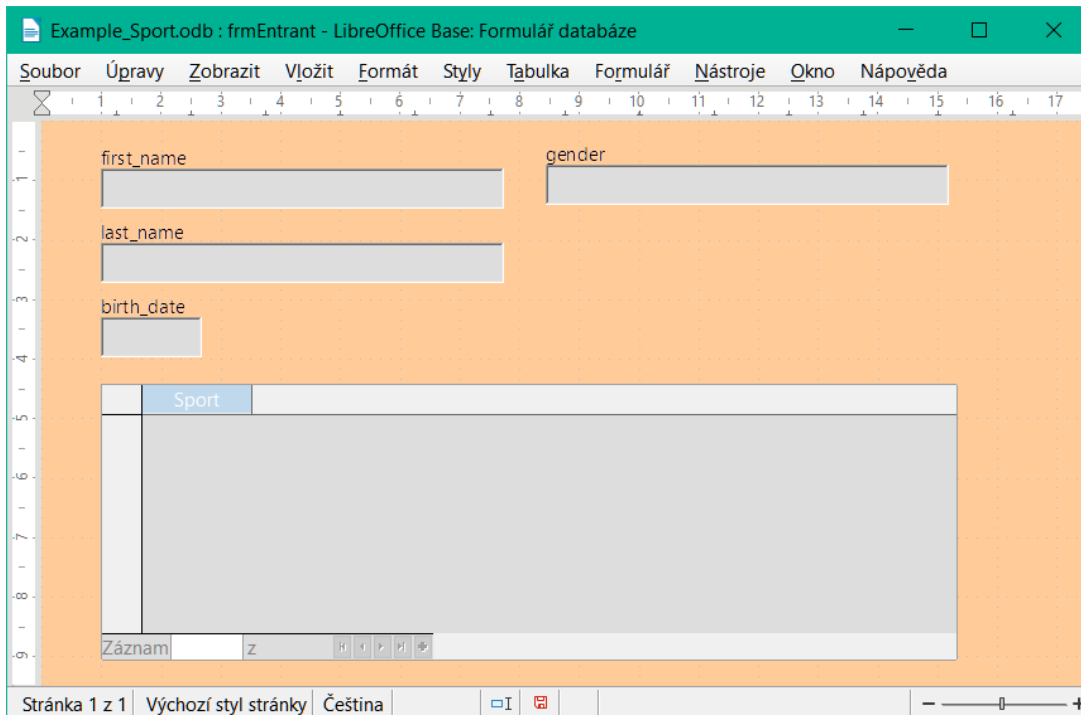
- 16) Klikneme na tlačítko **Další >**. Base zobrazí krok 8 Průvodce formulářem.
17) Do pole *Název formuláře* zadáme text „frmEntrant“.
18) Vybereme možnost **Změnit formulář** (obrázek 45).



Obrázek 45: Krok 8 Průvodce formulářem

- 19) Kliknutím na tlačítko **Dokončit** zavřeme Průvodce formulářem.

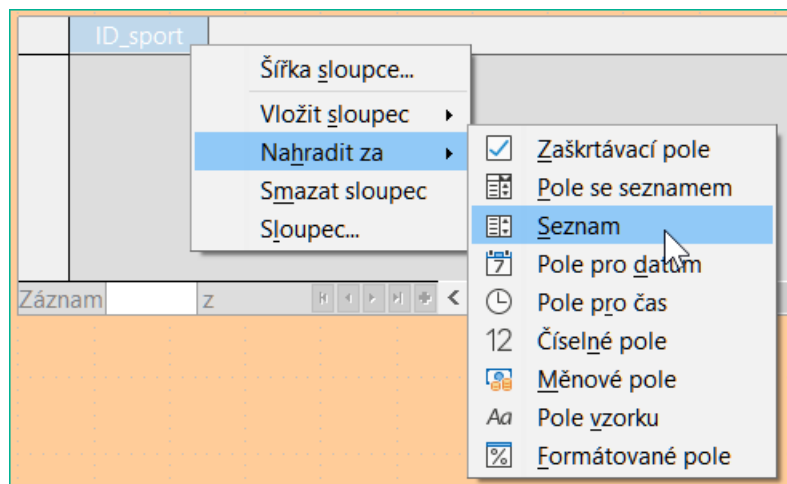
Po ukončení Průvodce formulářem se v okně Průvodce databází zobrazí počáteční verze formuláře. Měla by se podobat verzi na obrázku 46.



Obrázek 46: Počáteční verze formuláře frmEntrant

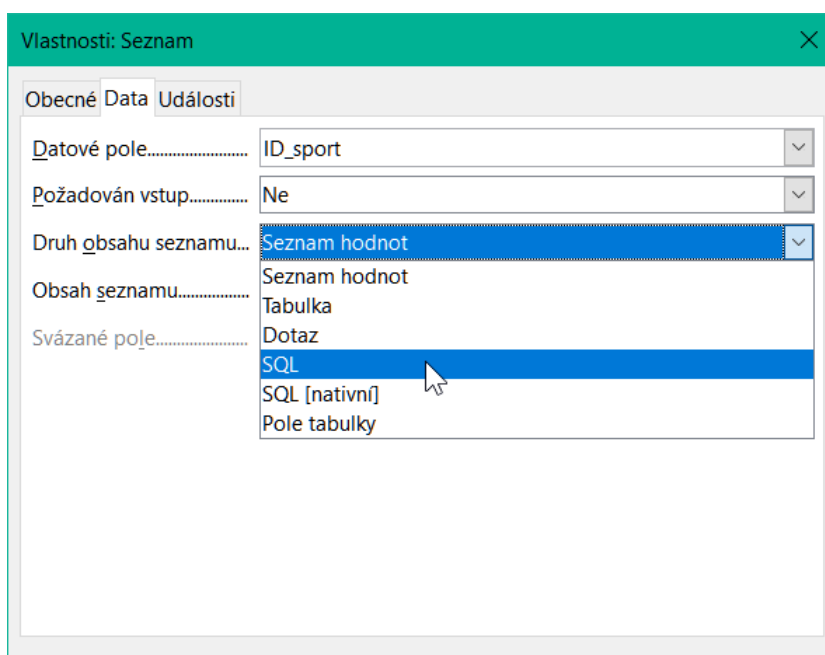
V této původní verzi podformuláře musí uživatel zadat zkratku každého sportu, ve kterém chce účastník soutěžit. Významným zlepšením by bylo poskytnout nabídku umožňující výběr sportů s použitím jejich celých názvů. Toho je dosaženo prostřednictvím následujících opatření.

- 1) V oblasti podformuláře v dolní části formuláře klikneme pravým tlačítkem myši na záhlaví tabulky *ID_sport* a v místní nabídce vybereme možnost **Nahradit za > Seznam**.



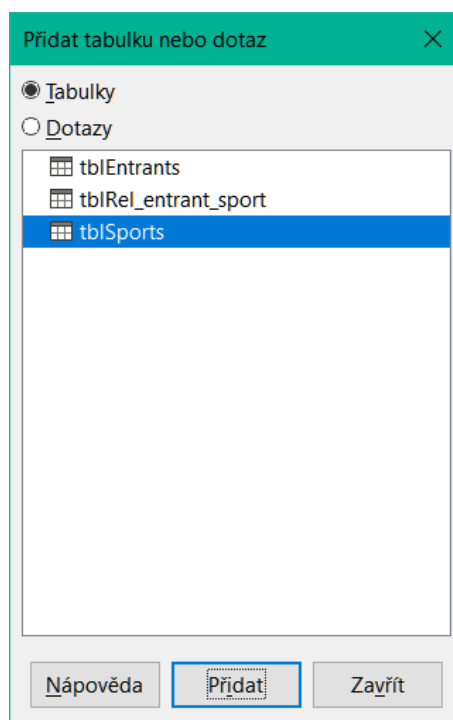
Obrázek 47: Převedení ID_sport do pole seznamu

- 2) Dále je třeba seznam upravit tak, aby mohl zobrazovat zamýšlené sportovní údaje. Znovu klikneme pravým tlačítkem myši na záhlaví tabulky *ID_sport*, ale tentokrát v místní nabídce vybereme možnost **Sloupec**. Base zobrazí Vlastnosti: dialogové okno Seznam.
- 3) Na kartě *Data* změníme *Druh obsahu seznamu* na **Sql** (obrázek 48). Pomocí jazyka SQL získá seznam svůj obsah z tabulky *tblSports*.



Obrázek 48: Nastavení pole Seznam pro použití dotazu SQL

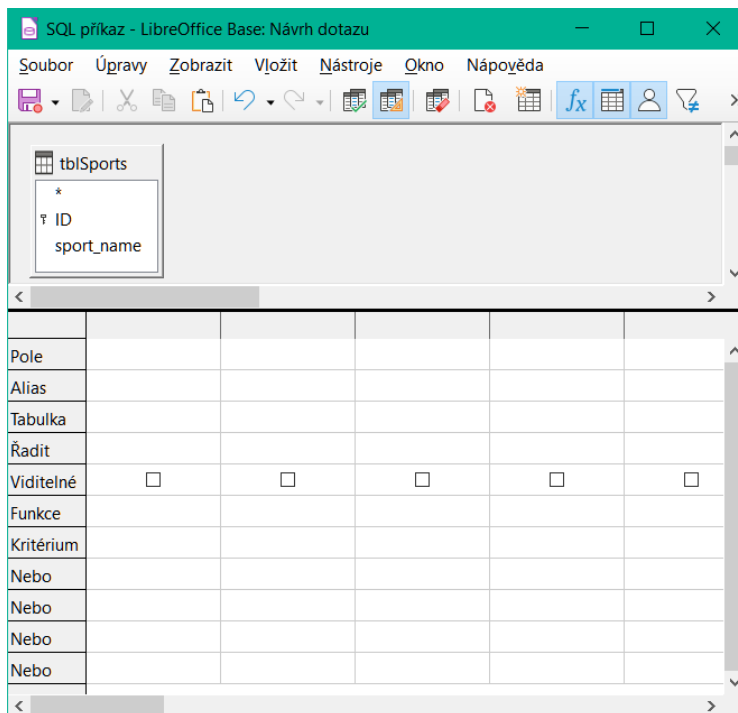
- 4) Kliknutím na tlačítko s třemi tečkami (...) napravo od pole *Obsah seznamu* otevřeme editor pro vytváření dotazů. Vytvoří se vhodný dotaz, který se nakonec uloží do samotného pole seznamu.
- 5) Base otevře okno *Návrh dotazu* a dialogové okno *Přidat tabulku nebo dotaz* (obrázek 49), abychom mohli začít vytvářet dotaz.



Obrázek 49: Dialogové okno *Přidat tabulku nebo dotaz*

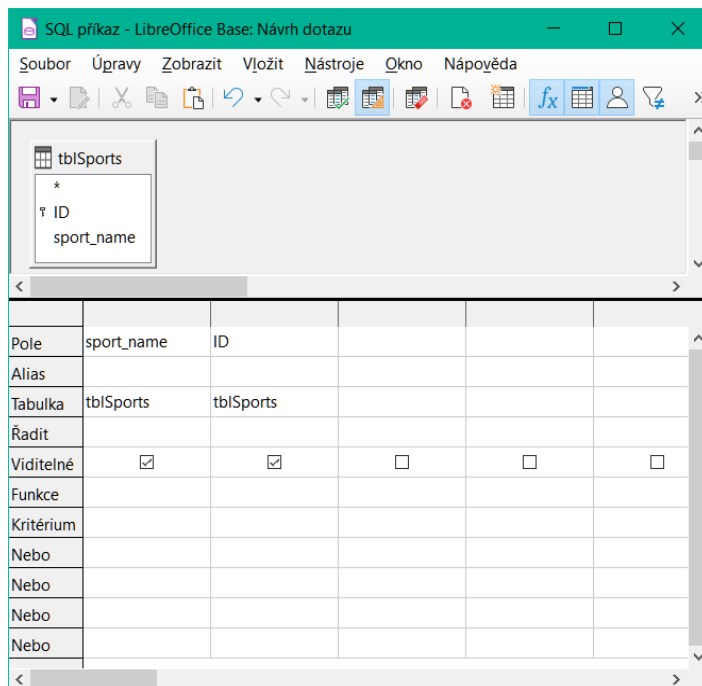
- 6) V dialogovém okně *Přidat tabulku nebo dotaz* vybereme přepínač **Tabulky** a poté vybereme položku **tblSports**.
- 7) Klikneme na tlačítko **Přidat** a poté na tlačítko **Zavřít**.

- 8) Base odstraní dialogové okno Přidat tabulku nebo dotaz a v horní části okna Návrh dotazu zobrazí pole představující tabulku *tblSports* (obrázek 50).



Obrázek 50: Okno návrhu dotazu zobrazující tabulku *tblSports*

- 9) V prvním sloupci dolní části okna klikneme na políčko vedle *Pole* a z rozevřací nabídky vybereme *tblSports.sport_name*.
- 10) Ve druhém sloupci vybereme pole *tblSports.ID* (obrázek 51). Toto pole poté předá svou hodnotu tabulce *tblSports*, která je zdrojem dat podformuláře. To umožňuje zobrazit definovaná slova a uložit příslušné zkratky.



Obrázek 51: Návrh dotazu pro vyplnění pole seznamu *ID_sport* na formuláři *frmEntrant*

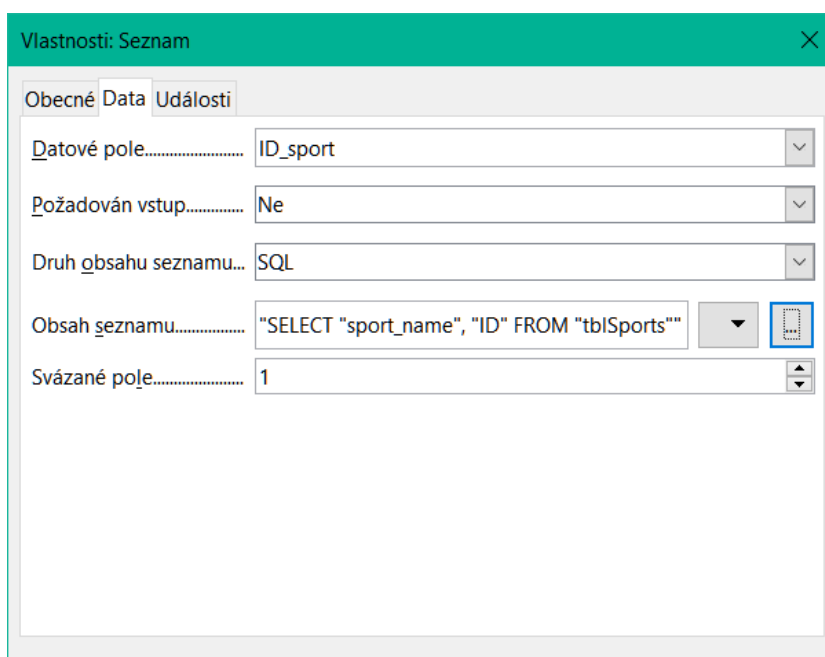
- 11) Uložíme dotaz, který se přenese do vlastností pole seznamu. Završíme editor dotazů.
- 12) Nyní se v poli *Obsah seznamu* ve vlastnostech v dialogovém okně Seznam zobrazí kód SQL, který byl vytvořen v editoru dotazů (obrázek 52):

```
SELECT "sport_name", "ID" FROM "tblSports"
```

Tento kód říká: V tabulce *tblSports* vybereme pole *sport_name* a přidruženou hodnotu klíče *ID*.

Tip

Často je užitečné zkontrolovat automaticky generovaný kód SQL, abychom se lépe seznámili s jazykem dotazů.

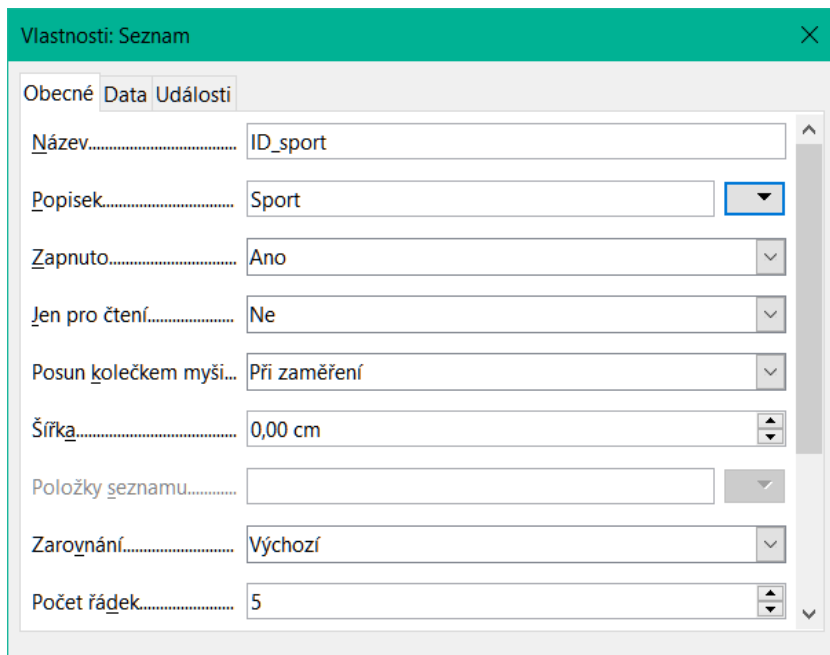


Obrázek 52: Kód SQL v poli *Obsah seznamu*

Tento dotaz ilustruje minimum, které by mělo být vybráno. Samozřejmě by mohlo být začleněno třídění. Uložením pečlivě vybraných zkratk získáme užitečný seznam sportů uložený v *ID*. Pokud nejsou záznamy seřazeny zadaným způsobem, provádí se třídění vždy podle pole primárního klíče. Aby se později v seznamu zobrazily sporty, musí být tento obsah zadán v tabulce *tblSport*.

Chceme-li změnit popis pole *ID_sport* tak, aby se zobrazoval jako *Sport*, provedeme následující kroky:

- 1) V dialogovém okně *Vlastnosti seznamu* klikneme na kartu *Obecné*.
- 2) Nahradíme obsah pole *Popisek* hodnotou „Sport“ (obrázek 53).

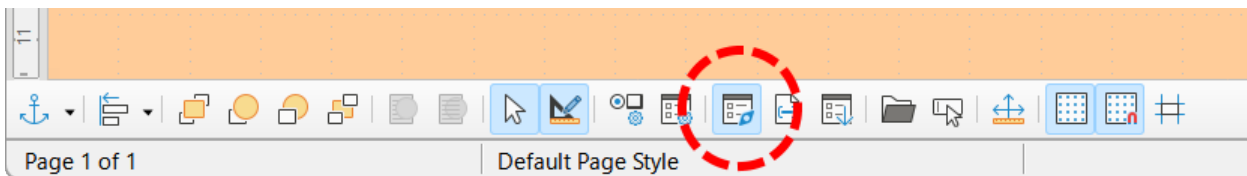


Obrázek 53: Změna zobrazovaného štítku pro ID_sport

3) Zavřeme dialogové okno Vlastnosti pole seznamu.

Změnu názvů ostatních polí ve formuláři nejlépe provedeme prostřednictvím Navigátoru formuláře. Pokud klikneme na pole, vyberou se nejen pole, ale i související popisky. Průvodce je seskupil dohromady. To pak vyžaduje další akci z místní nabídky.

Navigátor formuláře je přístupný po kliknutí na ikonu na nástrojové liště Návrh formuláře, která se při zobrazení obvykle nachází v dolní části okna Formuláře databáze. Obrázek 54 zobrazuje umístění ikony **Navigátor formulářem** na nástrojové liště Návrh formuláře, zvýrazněné červeným kroužkem.

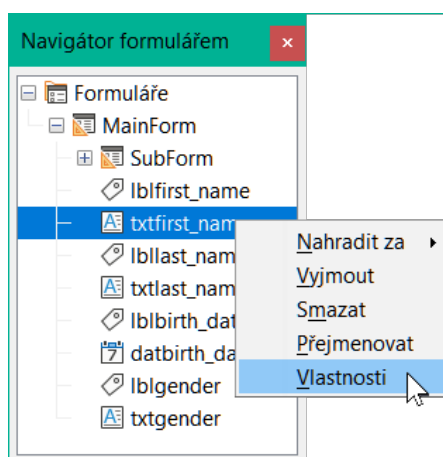


Obrázek 54: Umístění ikony Navigátor formuláře na nástrojové liště Návrh formuláře

i Tip

Pokud nástrojová lišta Návrh formuláře není viditelná, můžeme se k ní dostat výběrem možnosti **Zobrazení > Nástrojové lišty > Návrh formuláře** v hlavní nabídce.

Každé pole lze prozkoumat jednotlivě pomocí nástroje Navigátor formuláře a vlastnosti pole jsou pak přístupné v místní nabídce. Po přechodu na jinou vlastnost se položka vlastnosti automaticky uloží. I při otevřeném dialogovém okně Vlastnosti je možné přecházet z jednoho pole do druhého. Zde je také uložena příslušná střední úroveň.



Obrázek 55: Přístup k dialogovému oknu Vlastnosti z Navigátoru formuláře

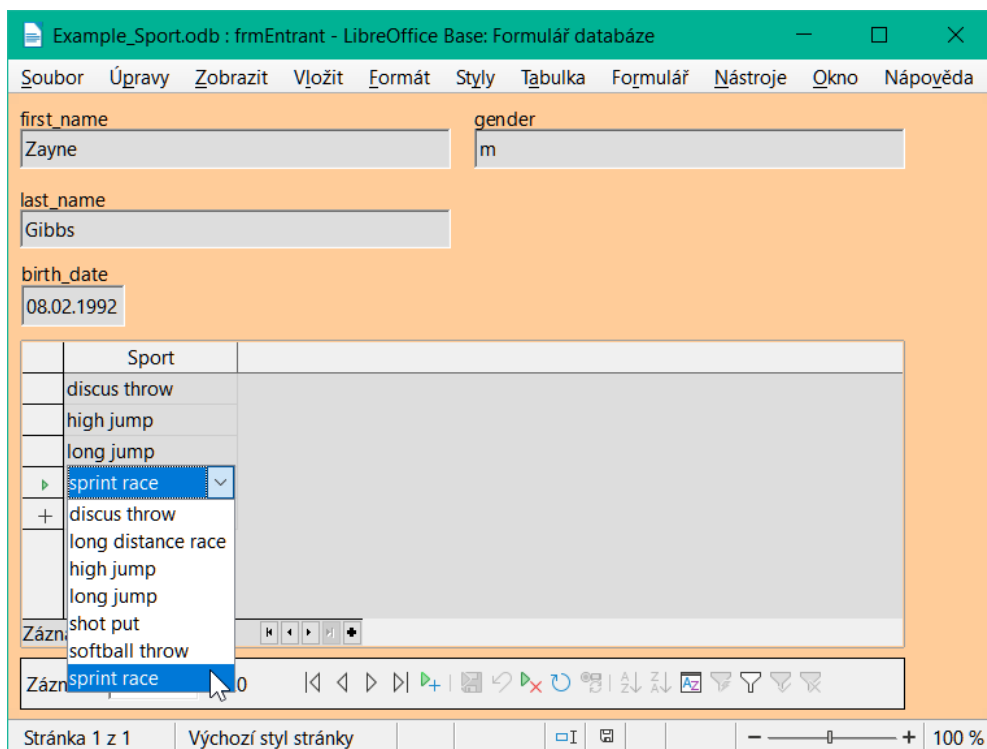
Pokud byl návrh dokončen, uložíme jej a zavřeme formulář. Poté soubor programu Base znovu uložíme.

Pro účely testování formuláře předpokládáme, že tabulka *tblSports* byla naplněna daty uvedenými v tabulce 3.

Tabulka 3: Data zadaná v tabulce *tblSports*

ID	sportovní_název
disc	hod diskem
dist	závod na dlouhé tratě
vysoká	skok do výšky
long	skok do dálky
shot	vrh koulí
soft	softbalový hod
sprt	závod ve sprintu

Prvním účastníkem soutěže je Zayne Gibbs, který by se rád přihlásil do čtyř disciplín – hod diskem, běh do dálky, skok do výšky a sprint. Po zadání dat do formuláře může vypadat podobně jako na obrázku 56. Po zadání údajů nezapomeneme záznam uložit.



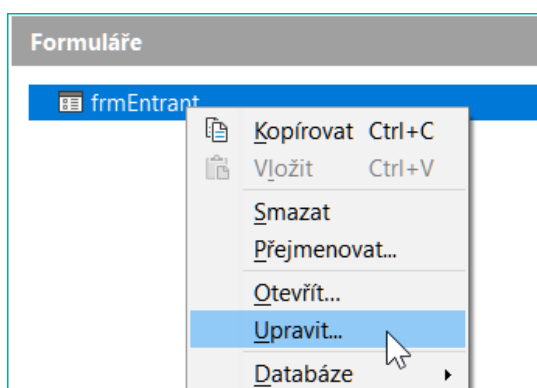
Obrázek 56: Ukázka dat ve formuláři frmEntrant

Do takového formuláře by samozřejmě mohly být začleněny mechanismy třídění.

Při používání formuláře si můžeme všimnout některých nepříjemností:

- Klávesa *Tab* slouží k rychlému procházení polí formuláře při zadávání dat. Ačkoliv, když je kurzor v poli *gender* ve formuláři, další zmáčknutí klávesy *Tab* skočí přímo na další záznam v tabulce *tblEntrants*. Výpis podformuláře sportů není přímo přístupný pomocí klávesy *Tab*.
- Pokud se kurzor nachází v některém ze čtyř polí v horní části formuláře, navigační panel v rámci nástrojové lišty Navigace na formuláři v dolní části okna zobrazí číslo záznamu v tabulce *tblEntrants*. Pokud však pracujeme v podformuláři, pak se navigační panel přepne na zobrazení čísla záznamu *tblSports* v rámci podformuláře. Tyto informace jsou již přítomny ve vyhrazeném navigačním panelu v oblasti podformuláře, a proto by bylo vhodnější, kdyby navigační panel v rámci panelu nástrojů Navigace formuláře vždy zobrazoval čísla záznamů účastníka v tabulce *tblEntrants*.
- Při zadávání pohlaví účastníka si uživatel musí zapamatovat sadu možných variant. V současné době je jediným omezením definovaným v tabulce *tblEntrants* omezení na jeden znak a měl by být stanoven bezpečnější způsob zadávání.

Chceme-li postupně vyřešit každý z těchto problémů, otevřeme nejprve formulář pro úpravy, nikoli pro zadávání dat (obrázek 57).



Obrázek 57: Otevření formuláře *frmEntrant* pro úpravy

Přepínání na podformulář

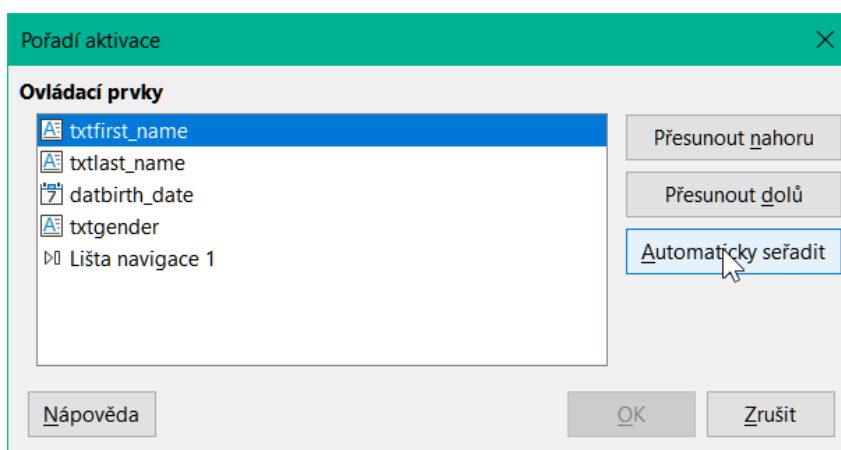
Aby se po zadání pohlaví účastníka nepřecházelo rovnou na další záznam pomocí klávesy *Tab*, je nutné upravit aktivační sekvenci.

- 1) Klikneme na tlačítko **Pořadí aktivity** na nástrojové liště Návrh formuláře, který se při zobrazení obvykle nachází v dolní části okna Formulář databáze. Obrázek 58 zobrazuje umístění ikony **Pořadí aktivity** na nástrojové liště Návrh formuláře, zvýrazněné červeným kroužkem.



Obrázek 58: Umístění ikony **Pořadí aktivity** na nástrojové liště Návrh formuláře

- 2) Base zobrazí dialogové okno **Pořadí aktivity** (obrázek 59).



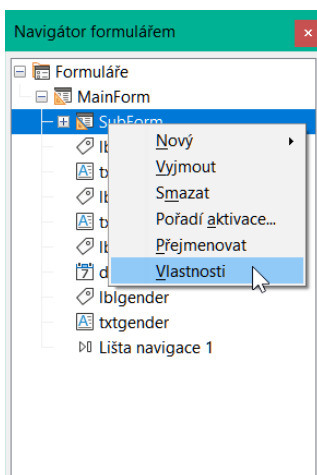
Obrázek 59: Dialog **Pořadí aktivity**

- 3) Klikneme na možnost **Automaticky seřadit**, která ovlivňuje nejen třídění zobrazených ovládacích prvků, ale také automatické přeměrování do podformuláře. Ačkoli to z dialogu není zřejmé, funguje to tak i na pozadí.
- 4) Kliknutím na tlačítko **OK** zavřeme dialogové okno **Pořadí aktivity**.

Později ukončíme režim úprav, otevřeme formulář pro zadávání dat, ujistíme se, že pořadí aktivity pro formulář *frmEntrant* je nyní podle očekávání a že můžeme podformuláře vybrat pomocí klávesy *Tab*.

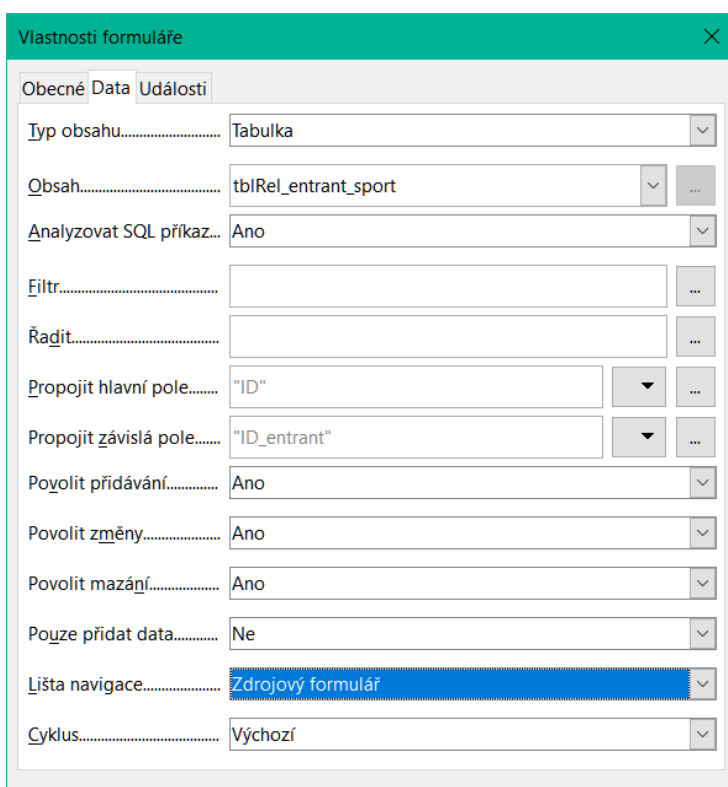
Správa navigačních panelů hlavního formuláře a podformuláře

- 1) Kliknutím na ikonu **Navigátor formulářem** na nástrojové liště Návrh formuláře (obrázek 54) otevřeme Navigátor formulářem.
- 2) Klepneme pravým tlačítkem myši na pole *Podformulář* v Navigátoru formulářem a vybereme položku **Vlastnosti** v místní nabídce (obrázek 60).



Obrázek 60: Dialogové okno pro přístup ke změně vlastností podformuláře

- 3) Base zobrazí dialogové okno Vlastnosti formuláře.
- 4) Na kartě *Data* změním nastavení pole *Lišta navigace* na **Zdrojový formulář** (obrázek 61).



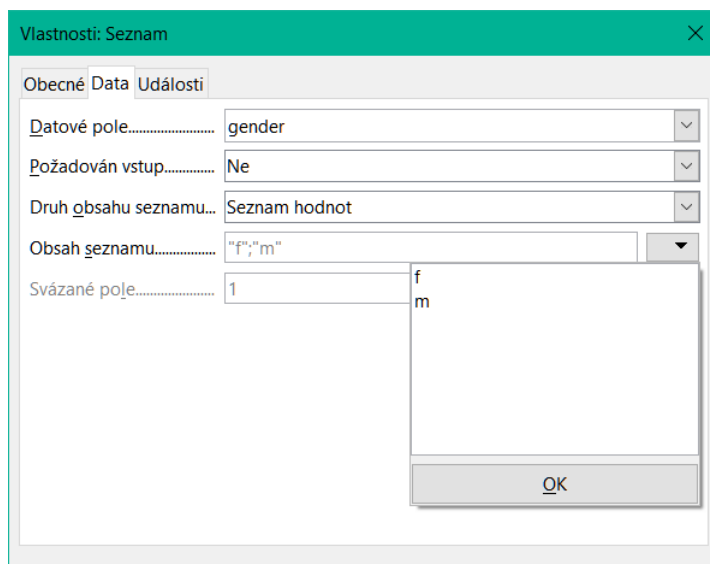
Obrázek 61: Změna nastavení podformuláře *Lišta navigace*

Později ukončíme režim úprav, otevřeme formulář pro zadávání dat a ujistíme se, že navigační panel na nástrojové liště Navigátor formulářem vždy zobrazuje pozici aktuálního záznamu v tabulce *tblEntrants*, a to i při práci v podformuláři.

Omezení vstupu do ovládacího prvku

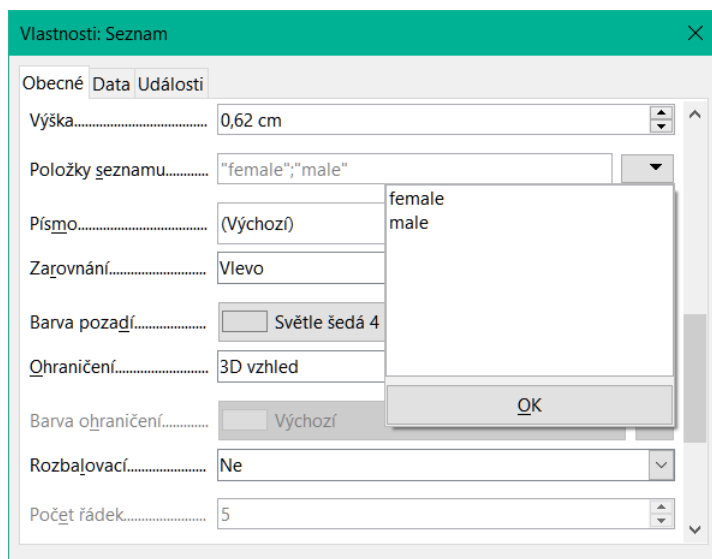
Aby bylo možné omezit vstup na zadané hodnoty, nemůže být ovládací prvek pouhým textovým polem. Jedním z řešení by bylo použití skupinového rámečku, jak je popsáno v kapitole 4, Formuláře. Dalším řešením je zobrazit volby v Seznamu, a to následujícím způsobem:

- 1) Kliknutím na ikonu **Navigátor formulářem** na nástrojové liště Návrh formuláře (obrázek 54) otevřeme Navigátor formulářem.
- 2) Klikneme pravým tlačítkem myši na pole Textové pole pro zadání pohlaví účastníka a v místní nabídce vybereme možnost **Nahradit > Seznam**.
- 3) Znovu klikneme pravým tlačítkem myši na stejné pole, ale tentokrát vybereme v místní nabídce možnost **Vlastnosti**.
- 4) Base zobrazí Vlastnosti: dialogové okno Seznam.
- 5) Na kartě *Data* klikneme na šipku dolů vpravo od pole *Obsah seznamu*. Do prvního řádku zobrazeného pole pro zadávání textu zadáme písmeno „f“, do dalšího řádku zadáme „m“ a klikneme na tlačítko **OK** (obrázek 62). Tyto zkratky jsou hodnoty, které budou předány do tabulky *tblEntrants*.



Obrázek 62: Nastavení povolených hodnot v poli pohlaví (karta *Data*)

- 6) Klikneme na kartu *Obecné*.
- 7) Posouváme se po této kartě dolů, dokud nedojdeme k poli *Položky seznamu*. Klikneme na šipku dolů vpravo od pole *Položky seznamu*. Do prvního řádku zobrazeného pole pro zadávání textu zadáme slovo „female“ (žena), do dalšího řádku zadejte „male“ (muž) a klikneme na tlačítko **OK** (obrázek 63). Jedná se o hodnoty, ze kterých bude uživatel vybírat, a musí být ve stejném pořadí jako hodnoty zadané v poli *Obsah seznamu* na kartě *Data*.



Obrázek 63: Nastavení hodnot povolených v poli pohlaví (karta Obecné)

- 8) V poli *Rozbalovací*, které se nachází o něco níže na kartě, vybereme možnost **Ano**.
- 9) Ujistíme se, že jsme uložili všechny změny ve formuláři a poté uložili databázi.

Později ukončíme režim úprav, otevřeme formulář pro zadávání dat a ujistíme se, že do pole *gender* lze zadávat pouze platné vstupy.

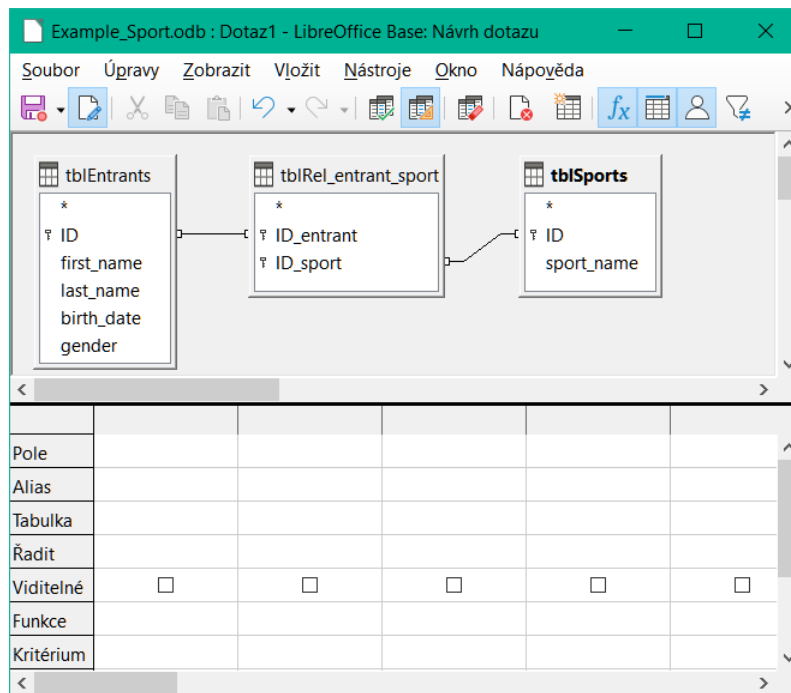
Nejvýraznější nepříjemnosti tak byly odstraněny. Může začít zadávání vstupů pro muže a ženy, stejně jako jejich přiřazení ke sportům.

Následující krok je užitečný: záznamy by se měly zadávat pouze jednou. Dbejme na to, aby účastníci mohli soutěžit i mezi sebou podle věku a sportu. Jinak by následné dotazy a sestavy nemusely dávat smysl.

Vytvoření dotazu

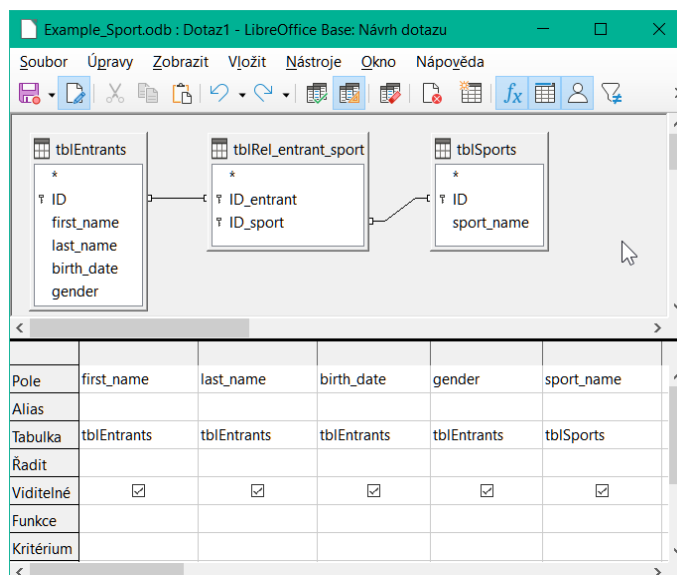
V dotazu lze seskupit obsah různých tabulek. V této části je vytvořen jednoduchý dotaz, který zobrazí každého z účastníků spolu se seznamem sportů na které se přihlásili.

- 1) Klikneme na **Dotazy** v oblasti *Databáze* hlavního okna Base.
- 2) Klikneme na tlačítko **Vytvořit dotaz v režimu návrhu** v oblasti *Úlohy* hlavního okna Base.
- 3) Base zobrazí okno *Návrh dotazu* a otevře dialogové okno *Přidat tabulku nebo dotaz* (obrázek 49).
- 4) Vybereme tabulku *tblEntrants* a klikneme na tlačítko **Přidat**.
- 5) Vybereme tabulku *tblRel_entrant_sport* a klikneme na tlačítko **Přidat**.
- 6) Vybereme tabulku *tblSports* a klikneme na tlačítko **Přidat**.
- 7) Kliknutím na tlačítko **Zavřít** odstraníme ze zobrazení dialogové okno *Přidat tabulku nebo dotaz*.
- 8) Všimneme si, že tyto tři tabulky a vztahy mezi nimi jsou znázorněny v horní části okna *Návrh dotazu* (obrázek 64).



Obrázek 64: Okno návrhu dotazu s přidánými třemi tabulkami

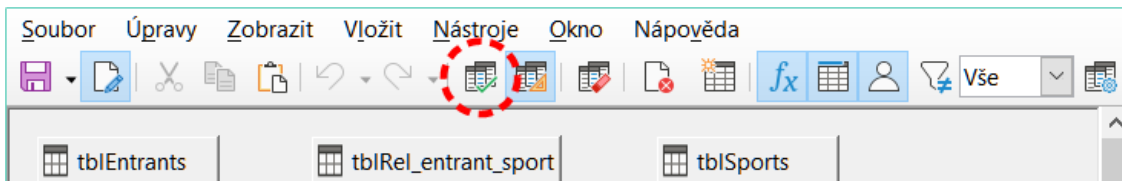
- 9) Dvakrát klikneme na pole *first_name*, *last_name*, *birth_date* a *gender* v tabulce *tblEntrants* a ověříme, že tato pole se přidají do sloupců dotazu ve spodní části okna Návrh dotazu. Dvojklik je asi nejjednodušší způsob přidání pole do dotazu.
- 10) Ve spodní části okna Návrh dotazu klikneme na prázdnou položku *Pole* umístěnou napravo od položky *gender* a poté kliknutím na šipku dolů otevřeme rozevírací seznam polí, která lze do dotazu zahrnout. Seznam se skládá z názvů polí a jim odpovídajících názvů tabulek indexovaných podle názvu tabulky. Aby bylo možné správně přiřadit pole tabulek k jejich tabulkám, jsou v dotazech označena "název tabulky". "název pole". Pokud je místo názvu pole použita "*", znamená to, že se zobrazí všechna pole příslušné tabulky. V seznamu vybereme položku *tblSports.sport_name* – to je alternativa k metodě dvojkliku popsané v předchozím kroku (obrázek 65).



Obrázek 65: Okno návrhu dotazu zobrazující pět polí přidanych do dotazu

Dotaz je kompletní a je vhodné jej před uložením spustit, abychom zjistili, zda přinese požadované výsledky.

Chceme-li spustit dotaz z okna Návrh dotazu, klikneme na ikonu **Spustit dotaz** na nástrojové liště Návrh dotazu, která je standardně umístěna v horní části okna (obrázek 66).



Obrázek 66: Umístění ikony Spustit dotaz

Výstupy z tohoto jednoduchého dotazu se zobrazí v seznamu umístěném v horní části okna Návrh dotazu (obrázek 67). Dotaz zobrazí všechny kombinace účastníků a sportů. Pokud se účastníci přihlásili do více sportů, mají tolik záznamů. Účastníci se neobjevují bez sportovních disciplín.

	first_name	last_name	birth_date	gender	sport_name
▶	Jessica	McCoy	12.02.1986	f	discus throw
	Jessica	McCoy	12.02.1986	f	long distanc
	Jessica	McCoy	12.02.1986	f	high jump
	Jessica	McCoy	12.02.1986	f	sprint race
	Patricia	Pike	02.12.1985	f	discus throw
	Patricia	Pike	02.12.1985	f	long jump
	Patricia	Pike	02.12.1985	f	shot put
	Patricia	Pike	02.12.1985	f	sprint race
	Stephen	Putnam	14.06.2002	m	discus throw
	Stephen	Putnam	14.06.2002	m	long distanc
	Stephen	Putnam	14.06.2002	m	long jump
	Stephen	Putnam	14.06.2002	m	sprint race
	Jerry	Brown	13.05.2002	m	discus throw
	Jerry	Brown	13.05.2002	m	high jump
Záznam	1	z 44			

Obrázek 67: Výstup dotazu

Abychom dotaz ještě trochu rozšířili, předpokládejme, že na některé události se mají vztahovat věková omezení. V takovém případě je důležité znát věk každého účastníka, aby mohl být zařazen do správné věkové kategorie pro každou z akcí. Jednoduchým krokem, který by s tím pomohl, by bylo určení věku účastníka na základě jeho data narození (uloženého v tabulce *tblEntrants*) a aktuálního data a zahrnutí výsledku tohoto výpočtu do dotazu.

Jednoduchým způsobem, jak toho dosáhnout, je použití dvou vestavěných funkcí HSQLDB:

- **NOW()**. Funkce NOW vrací hodnotu data a času, která představuje okamžik volání funkce, tedy aktuální datum a čas.
- **ROK(dt)**. Funkce YEAR extrahuje složku roku svého argumentu *dt*, což je hodnota data a času.

Napíšeme vzorec `YEAR(NOW()) - YEAR("birth_date")` ve vedlejší volné buňce *Pole* dotazu, jak je znázorněno na obrázku 68.

	first_name	last_name	birth_date	gender	sport_name	YEAR(NOW()) - YEAR("birth_date")
Pole						
Alias						
Tabulka	tblEntrants	tblEntrants	tblEntrants	tblEntrants	tblSports	
Řadit						
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Obrázek 68: Výpočet věku zahrnutý v dotazu

První část tohoto vzorce, `YEAR(NOW())`, vybere aktuální rok. Druhá část, `YEAR("birth_date")`, vybere rok narození účastníka. Mezi nimi se vytvoří rozdíl, který udává (přibližně) aktuální věk osoby. Výsledky jsou uvedeny na obrázku 69.

Tyto a mnoho dalších funkcí, které pracují s vestavěným databázovým strojem HSQLDB, jsou popsány ve zbývajících částech této knihy a jejich úplný seznam nalezneme v kapitole 9 Uživatelské příručky HyperSQL na adrese <http://hsqldb.org/doc/1.8/guide/index.html> (viz část s názvem Vestavěné funkce a uložené procedury).

	first_name	last_name	birth_date	gender	sport_name	YEAR(NOW()) - YEAR("birth_date")
▶	Zayne	Gibbs	08.02.1992	m	sprint race	30
	Zayne	Gibbs	08.02.1992	m	high jump	30
	Zayne	Gibbs	08.02.1992	m	long jump	30
	Zayne	Gibbs	08.02.1992	m	discus throw	30
	Stephen	Putnam	14.06.2002	m	long jump	20
	Stephen	Putnam	14.06.2002	m	sprint race	20
	Stephen	Putnam	14.06.2002	m	discus throw	20
	Stephen	Putnam	14.06.2002	m	long distance race	20
	Roger	McClure	04.09.1981	m	long jump	41
	Roger	McClure	04.09.1981	m	long distance race	41
	Patricia	Pike	02.12.1985	f	shot put	37
	Patricia	Pike	02.12.1985	f	sprint race	37
	Patricia	Pike	02.12.1985	f	discus throw	37
	Patricia	Pike	02.12.1985	f	long jump	37

Záznam 1 z 41 *

Obrázek 69: Výstup dotazu zobrazující vypočtený věk

Ve výchozím nastavení se kód zadaný do pole zobrazuje v záhlaví sloupce výstupu dotazu, což není uživatelsky příliš příjemné. To lze vylepšit zadáním aliasu, který se použije jako záhlaví sloupce místo kódu. Do řádku *Alias* pod pole obsahující kód zadáme výraz `sport_age`.

	first_name	last_name	birth_date	gender	sport_name	YEAR(NOW()) - YEAR("birth_date")
Pole						YEAR(NOW()) - YEAR("birth_date")
Alias						sport_age
Tabulka	tblEntrants	tblEntrants	tblEntrants	tblEntrants	tblSports	
Řadit						
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Obrázek 70: Použití aliasu k nahrazení vzorce

Alias `sport_age` se používá k odlišení vypočtené hodnoty od skutečného věku osoby. Obecně platí, že hodnota `sport_age` bude odpovídat skutečnému věku osoby pouze tehdy, pokud měla narozeniny v aktuálním roce.

Pokud je dotaz spuštěn znovu, záhlaví sloupce již neobsahuje kód, ale řetězec `sport_age`.

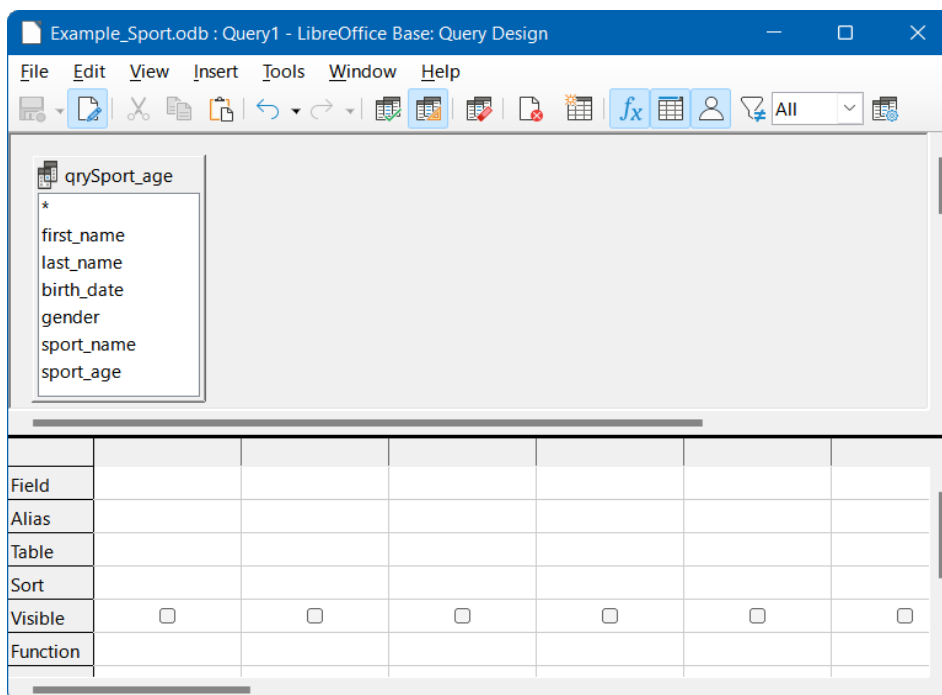
	first_name	last_name	birth_date	gender	sport_name	sport_age
▶	Zayne	Gibbs	08.02.1992	m	sprint race	30
	Zayne	Gibbs	08.02.1992	m	high jump	30
	Zayne	Gibbs	08.02.1992	m	long jump	30
	Zayne	Gibbs	08.02.1992	m	discus throw	30
	Stephen	Putnam	14.06.2002	m	long jump	20
	Stephen	Putnam	14.06.2002	m	sprint race	20
	Stephen	Putnam	14.06.2002	m	discus throw	20
	Stephen	Putnam	14.06.2002	m	long distance ra	20
	Roger	McClure	04.09.1981	m	long jump	41
	Roger	McClure	04.09.1981	m	long distance ra	41
	Patricia	Pike	02.12.1985	f	shot put	37
	Patricia	Pike	02.12.1985	f	sprint race	37
	Patricia	Pike	02.12.1985	f	discus throw	37
	Patricia	Pike	02.12.1985	f	long jump	37

Záznam 1 z 41 *

Obrázek 71: Výstup dotazu s použitím aliasu pro záhlaví posledního sloupce

Tento dotaz uložíme pod názvem *qrySport_age*. Na jeho základě se pak provede další dotaz, při němž se k zařazení osob do věkových skupin použijí poněkud složitější vzorce.

- 1) Klikneme na **Dotazy** v oblasti *Databáze* hlavního okna Base.
- 2) Klikneme na tlačítko **Vytvořit dotaz v režimu návrhu** v oblasti *Úlohy* hlavního okna Base.
- 3) Base zobrazí okno *Návrh dotazu* a otevře dialogové okno *Přidat tabulku nebo dotaz* (obrázek 49).
- 4) V horní části dialogového okna *Přidat tabulku nebo dotaz* vybereme možnost **Dotazy**.
- 5) Vybereme dotaz *qrySport_age*.
- 6) Klikneme na tlačítko **Přidat** a poté klikneme na tlačítko **Zavřít**, čímž ukončíme dialogové okno *Přidat tabulku nebo dotaz* (obrázek 72). Všimneme si, že ikona, která se objevuje před názvem *qrySport_age*, není stejná jako ikona používaná pro tabulky v tomto okně (viz například obrázek 64). Tento odlišný symbol je vizuální indikací, že základem tohoto dotazu je jiný dotaz, nikoli tabulka.



Obrázek 72: Okno návrhu dotazu s přidáním dotazem *qrySport_age*

- 7) V oblasti dotazu *qrySport_age* dvakrát klikneme na položku obsahující pouze hvězdičku (*). To má za následek výběr všech polí v dotazu *qrySport_age*. Alternativním způsobem, jak dosáhnout stejného výsledku, by bylo kliknout na první položku *Pole* a v rozevírací nabídce vybrat *qrySport_age.**.

✓ Poznámka

Pokud v tomto okamžiku spustíme nový dotaz, zjistíme, že poskytuje stejný výstup jako dotaz *qrySport_age*, ale bez použití vzorce.

Pole *sport_age* v dotazu *qrySport_věk* se nyní použije k určení věkové skupiny, které se jednotlivé osoby účastní. Aby výpočet nebyl příliš složitý, jsou sportovci mladší 20 let rozděleni do věkových skupin, přičemž každá skupina obsahuje dva věkové stupně počínaje číslem 0. Od 20 let výše se vytvářejí skupiny po 10 letech, například 20–29 let.

- 8) Do první prázdné buňky *Pole* zadáme následující vzorec: `CASEWHEN("sport_age" > 19, CEILING("sport_age" / 10) * 10, "sport_age" - MOD("sport_age", 2))`.
- 9) Napíšeme řetězec "*age_group*" do odpovídající buňky *Alias* (obrázek 73).

Pole	qrySport_age.*	CASEWHEN([sport_age] > 19, CEILING([sport_age] / 10) * 10, [sport_age] - MOD([sport_age], 2))
Alias		age_group
Tabulka	qrySport_age	
Řadit		
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Obrázek 73: Vzorec pro rozdělení startujících do věkových skupin

- 10) Klikneme na ikonu **Spustit dotaz** na nástrojové liště Návrh dotazu. Výstup dotazu je zobrazen na obrázku 74.

	first_name	last_name	birth_date	gender	sport_name	sport_age	age_group
▶	Jessica	McCoy	12.02.1986	f	discus throw	36	30
	Jessica	McCoy	12.02.1986	f	long distance ra	36	30
	Jessica	McCoy	12.02.1986	f	high jump	36	30
	Jessica	McCoy	12.02.1986	f	sprint race	36	30
	Patricia	Pike	02.12.1985	f	discus throw	37	30
	Patricia	Pike	02.12.1985	f	long jump	37	30
	Patricia	Pike	02.12.1985	f	shot put	37	30
	Patricia	Pike	02.12.1985	f	sprint race	37	30
	Stephen	Putnam	14.06.2002	m	discus throw	20	20
	Stephen	Putnam	14.06.2002	m	long distance ra	20	20
	Stephen	Putnam	14.06.2002	m	long jump	20	20
	Stephen	Putnam	14.06.2002	m	sprint race	20	20
	Jerry	Brown	13.05.2002	m	discus throw	20	20
	Jerry	Brown	13.05.2002	m	high jump	20	20

Záznam 1 z 44

Obrázek 74: Výstup dotazu zobrazující věkovou skupinu pro každého startujícího

11) Dotaz uložte pod názvem *qryRegistration*.

Takové vzorce by opravdu neměly být zahrnuty do úvodu do Base a bylo by možné udělat jednodušší rozdělení dle věku. Přestože je vzorec složitý, ukazuje sílu dostupných funkcí.

Podívejme se na jeho části trochu podrobněji:

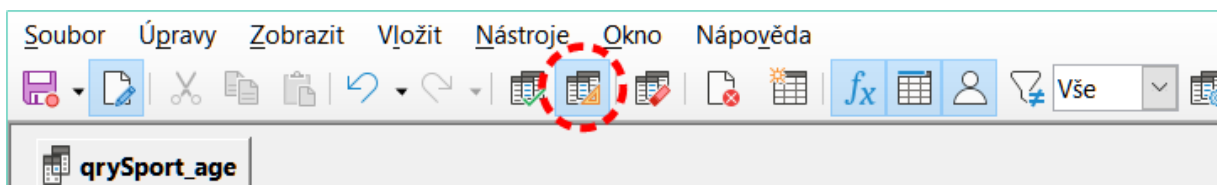
```
CASEWHEN( "sport_age" > 19, CEILING( "sport_age" / 10 ) * 10,
"sport_age" - MOD( "sport_age", 2 ) )
```

- CASEWHEN je další vestavěná funkce HSQLDB, která má tři argumenty. V abstraktnější podobě lze volání CASEWHEN zapsat jako CASEWHEN(výraz, hodnota1, hodnota2). Funkce vyhodnocuje výraz a pokud je pravdivý, je vrácena hodnota1; jinak je vrácena hodnota2.
- Funkce CEILING přijímá jeden argument. Pokud je volání funkce zapsáno jako CEILING(d), pak funkce vrátí nejmenší celé číslo, které není menší než d.
- Funkce MOD přijímá dva argumenty. Pokud je volání funkce zapsáno jako MOD(a, b) pak funkce vrátí a modulo b (což je zbytek po dělení a číslem b).

Vypnutí zobrazení návrhu v okně Návrh dotazu není ve skutečnosti nutné, protože všechny položky lze bez větších problémů zadat v zobrazení návrhu. Z výukového hlediska však může být poučné podívat se blíže na kód, který je základem dotazu.

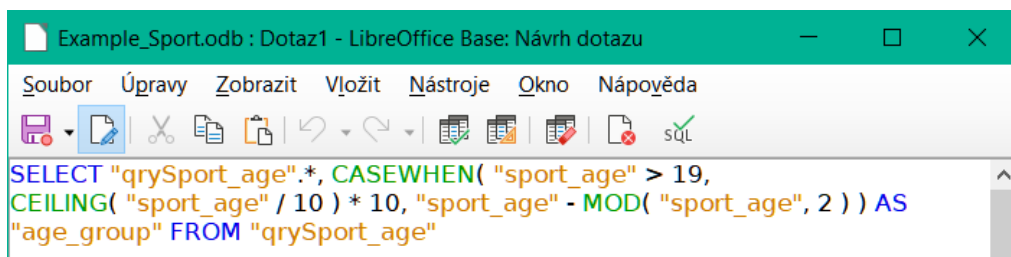
Je třeba si také uvědomit, že existují výrazy SQL, které se do návrhového zobrazení špatně hodí, nebo tam dokonce nejsou možné. V tomto případě můžeme v budoucnu chtít zadávat kód SQL přímo (s vypnutým zobrazením návrhu).

Klikneme na ikonu **Režim návrhu zap/vyp** na nástrojové liště Návrh dotazu (obrázek 75).



Obrázek 75: Umístění přepínače Zobrazení návrhu zapnuto/vypnuto na nástrojové liště Návrh dotazu

Při vypnutí zobrazení návrhu se v okně Návrh dotazu zobrazí kód SQL podle obrázku 76.



Obrázek 76: Kód SQL pro dotaz qryRegistration

Pro zobrazení kódu se používají následující konvence:

- Názvy polí a tabulek jsou uzavřeny v dvojitých uvozovkách a zobrazeny okrovou barvou.
- Výrazy SQL jsou zobrazeny modře.
- Názvy funkcí databáze jsou zobrazeny zeleně.

Poté, co jsme již probrali složky vzorce, můžeme se zaměřit na strukturu SQL:

```
SELECT "qrySport_age".*, ... AS "age_group" FROM "qrySport_age"
```

Z dotazu *qrySport_age* vybere všechny záznamy z a kromě záznamů určených vzorcem. Ty, které jsou určeny vzorcem, označujeme jako *age_group*.

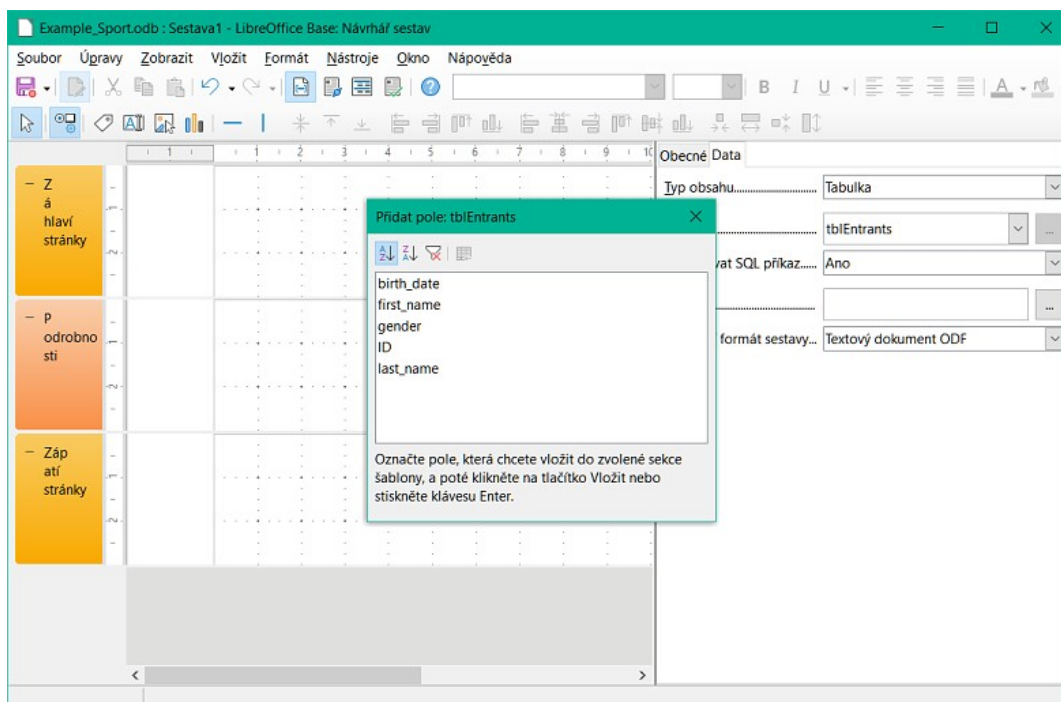
Kód nerozlišuje mezi tabulkami a dotazy jako základem dat. Funguje tedy pouze v grafickém uživatelském rozhraní programu Base. Dotaz nemůže mít stejný název jako tabulka; tabulka nemůže mít stejný název jako dotaz.

Při kliknutí na ikonu **SQL** na nástrojové liště SQL (pro přímé spuštění příkazu SQL) v zobrazení SQL databáze reaguje chybovou zprávou. Vestavěný engine HSQLDB nezná dotaz *qrySport_age*, na který odkazuje dotaz *qryRegistration*.

Vytvoření sestavy

Vytvoří se sestava se seznamem účastníků seřazených podle sportu, pohlaví a věkové skupiny.

- 1) V hlavním okně Base klikneme v oblasti **Databáze** na **Sestavy**.
- 2) V oblasti **Úlohy** klikneme na tlačítko **Vytvořit sestavu v režimu návrhu**.
- 3) Base zobrazí okno **Návrhář sestav** s dialogovým oknem **Přidat pole** v popředí.
- 4) V pravé části okna nástroje **Návrhář sestav** by se měla nacházet oblast pro zobrazení a úpravu vlastností aktuálně aktivního objektu. Pokud tato oblast není viditelná, vybereme v hlavní nabídce možnost **Zobrazit > Vlastnosti**. (Obrázek 77). Ve výchozí konfiguraci oblasti *Vlastností* pro databázi *Example_Sport.odt* je na kartě *Data* viditelné; pole *Typ obsahu* nastavené na **Tabulka**, a pole *Obsah* nastavené na **tblEntrants** (název první tabulky dle abecedy). Tato výchozí nastavení definují počáteční obsah dialogového okna **Přidat pole**.



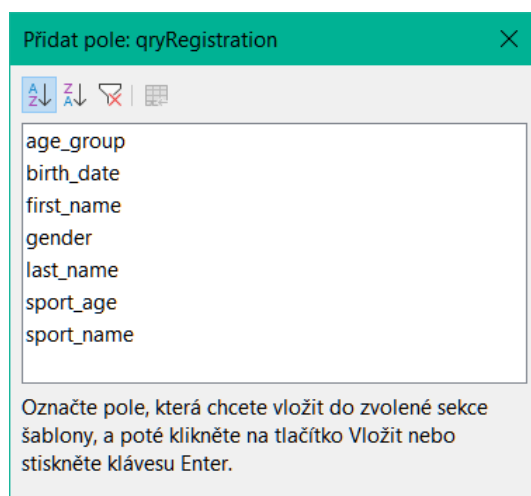
Obrázek 77: Počáteční zobrazení okna nástroje Návrhář sestav

- 5) Na kartě *Data* v oblasti *Vlastnosti* pro pole *Typ obsahu* vybereme **Dotaz**. Jako základ nové sestavy se použije dotaz *qryRegistration*.

✓ Poznámka

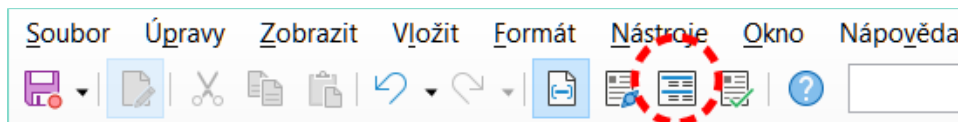
Výběr v rozevíracím seznamu *Typ obsahu* někdy způsobí, že velká část okna nástroje Návrhář sestav zčerná. Pokud k této chybě dojde, lze ji jednoduše obejít změnou velikosti okna pomocí ovládacího prvku v záhlaví okna a následným vrácením okna do původní velikosti. Tato akce by měla způsobit, že Base znovu zobrazí okno v normálním barevném schématu.

- 6) Na kartě *Data* v oblasti *Vlastnosti* vybereme v poli *Obsah* **qryRegistration**.
- 7) Base aktualizuje obsah dialogového okna Přidat pole tak, aby obsahoval seznam polí vybraného dotazu v alfanumerickém pořadí (obrázek 78).



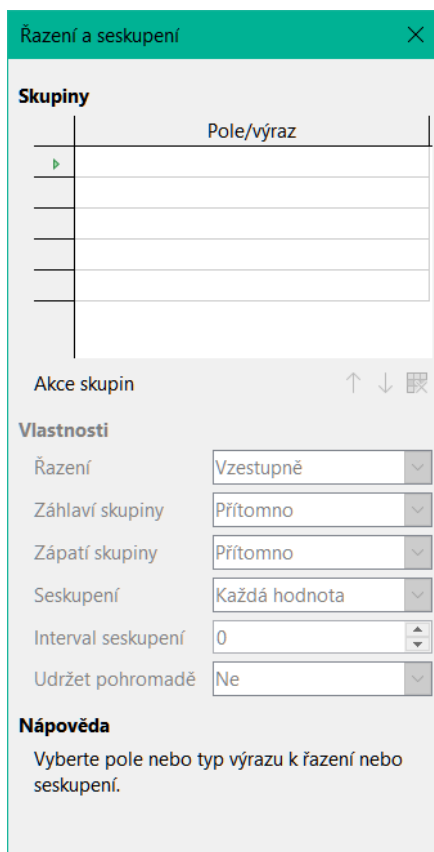
Obrázek 78: Dialogové okno Přidat pole zobrazující pole dotazu *qryRegistration*

- 8) Klikneme na ikonu **Řazení a seskupení** na nástrojové liště Standardní v okně nástroje Návrhář sestav (obrázek 79).



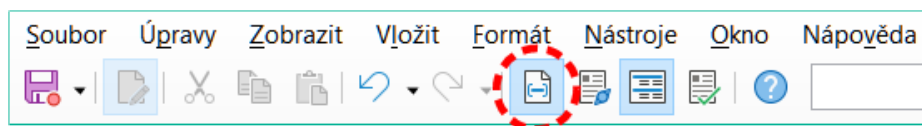
Obrázek 79: Umístění ikony Řazení a seskupení

- 9) Base zobrazí dialogové okno Řazení a seskupení (obrázek 80).



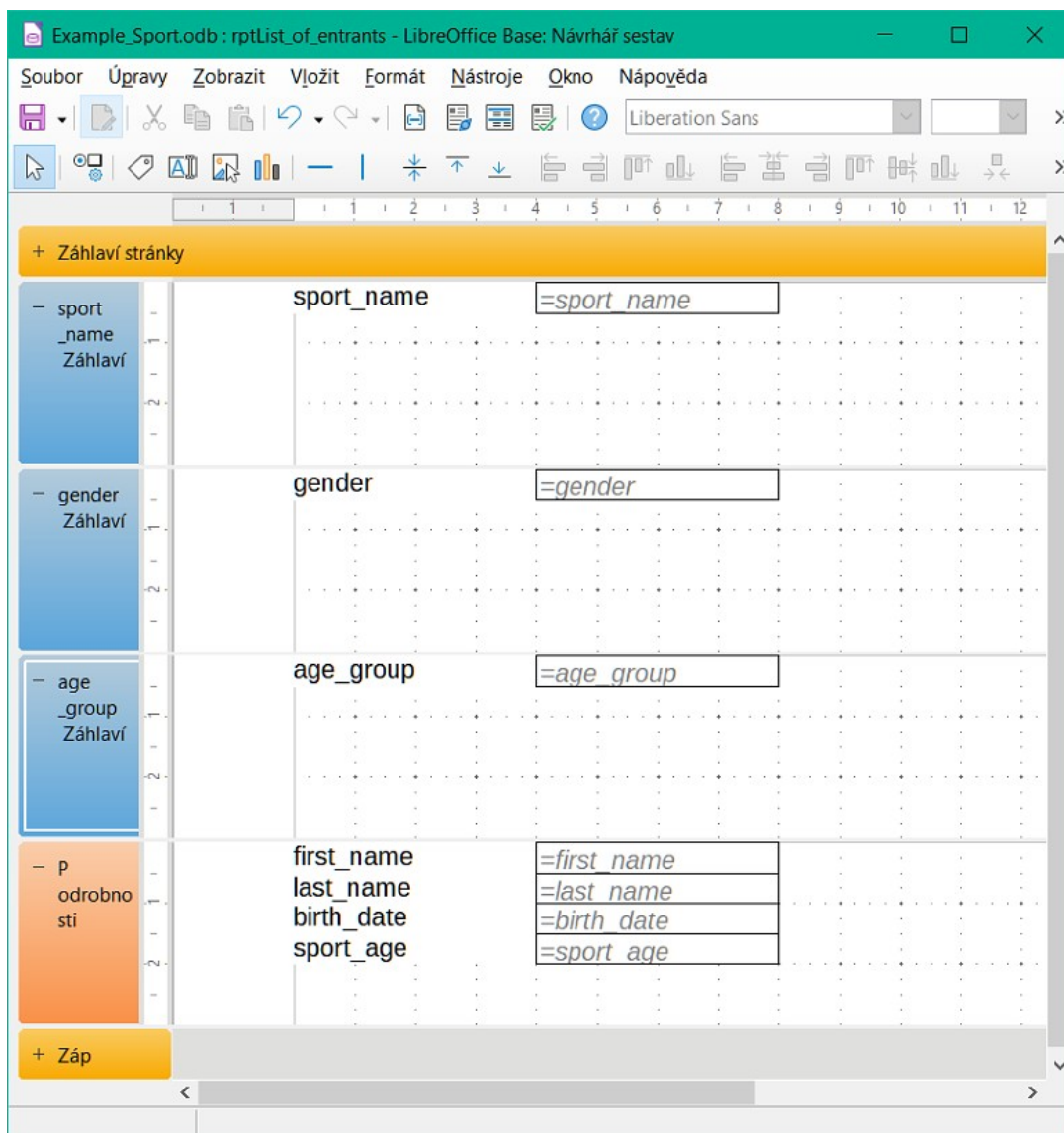
Obrázek 80: Dialogové okno Řazení a seskupení

- 10) V dialogovém okně Řazení a seskupení klikneme do první oblasti pod popiskem *Pole/výraz* a z rozevírací nabídky vybereme **sport_name**. Všimneme si, že do základního okna se přidá oblast *název_sportu Záhlaví*. Použijeme výchozí nastavení v oblasti *Vlastnosti* dialogového okna Řazení a seskupení.
- 11) Opakujeme krok 10) ale z rozbalovací nabídky vedle volného místa pod popiskem *Pole/výraz* vybereme *gender*.
- 12) Opakujeme krok 10) ale z rozbalovací nabídky vedle volného místa pod popiskem *Pole/výraz* vybereme *age_group*.
- 13) Zavřeme dialogové okno Řazení a seskupení.
- 14) V okně Návrhář sestav vybereme sekci skupiny *sport_name Záhlaví*. Base zvýrazní vybranou oblast bílým okrajem.
- 15) Pokud se dialogové okno Přidat pole nezobrazí, klikneme na ikonu **Přidat pole** na nástrojové liště Standardní v okně Návrhář sestav (obrázek 81).



Obrázek 81: Umístění ikony Přidat pole

- 16) Základna znovu zobrazí dialogové okno Přidat pole (obrázek 78).
- 17) V dialogovém okně Přidat pole vybereme položku *sport_name* a poté klikneme na ikonu **Vložit**.
- 18) Base vloží pole se štítkem a textové pole pro zobrazení obsahu pole *sport_name* dotazu *qryRegistrace* do zvýrazněné oblasti sestavy.
- 19) V okně Návrhář sestav vybereme část skupiny *gender Záhloví*. Base zvýrazní vybranou oblast bílým okrajem.
- 20) V dialogovém okně Přidat pole vybereme položku *gender* a klikneme na ikonu **Vložit**.
- 21) Base vloží pole se štítkem a textové pole pro zobrazení obsahu pole *gender* dotazu *qryRegistration* do zvýrazněné oblasti sestavy.
- 22) V okně sestavení sestavy vybereme oblast skupiny *age_group Záhloví*. Base zvýrazní vybranou oblast bílým okrajem.
- 23) V dialogovém okně Přidat pole vybereme položku *age_group* a klikneme na ikonu **Vložit**.
- 24) Base vloží pole se štítkem a textové pole pro zobrazení obsahu pole *age_group* dotazu *qryRegistration* do zvýrazněné oblasti sestavy.
- 25) V okně Sestavení sestavy vybereme oblast skupiny *Ddetail*. Base zvýrazní vybranou oblast bílým okrajem.
- 26) V dialogovém okně Přidat pole vybereme položku *firt_name* a klikneme na ikonu **Vložit**.
- 27) V dialogovém okně Přidat pole vybereme položku *last_name* a klikneme na ikonu **Vložit**.
- 28) V dialogovém okně Přidat pole vybereme položku *birthday* a poté klikneme na ikonu **Vložit**.
- 29) V dialogovém okně Přidat pole vybereme položku *sport_age* a klikneme na ikonu **Vložit**.
- 30) Base vloží pole se štítkem a textové pole pro zobrazení obsahu polí *first_name*, *last_name*, *birth_date*, and *sport_age* dotazu *qryRegistration* do zvýrazněné oblasti sestavy.
- 31) Zavřeme dialogové okno Přidat pole.
- 32) Návrh sestavy by nyní měl vypadat podobně jako na obrázku 82.



Obrázek 82: Návrh sestavy v okně Návrhář sestav

- 33) Uložíme sestavu s názvem *rptList_of_entrants*.
- 34) Uložíme databázi (jinak se sestava uloží pouze dočasně).

✓ Poznámka

Při návrhu sestavy pomocí nástroje Návrhář sestav byly hlášeny chyby jako nestabilita programu. Proto je důležité uložit jak sestavu, tak databázový soubor.

Naštěstí pozdější provedení sestavy obvykle není těmito problémy ovlivněno.

Chceme-li zobrazit sestavu, klikneme na **sestavy** v oblasti *Databáze* v hlavním okně Base a poté dvakrát klikneme na název požadované sestavy uvedené v oblasti *Sestavy*.

Pokud je tato sestava spuštěna s příslušnými daty, pak by první část výstupu měla vypadat jako na obrázku 83.

sport_name	discus throw
gender	f
age_group	30.0
first_name	Jessica
last_name	McCoy
birth_date	02/12/86
sport_age	36
first_name	Patricia
last_name	Pike
birth_date	12/02/85
sport_age	37

Obrázek 83: První část návrhu výstupu sestavy

Na začátku sestavy jsou uvedeny dvě účastnice, Jessica a Patricia, které se přihlásily do soutěže v hodů diskem a patří do věkové skupiny 30-39 let. V závislosti na údajích v tabulkách databáze může následovat mnoho dalších kombinací účastník / věková skupina / sport.

Při prvním spuštění sestavy je běžné, že rozpoznáme některé funkce, které by mohly být vylepšeny, jako např.:

- Vzdálenosti mezi záhlavími skupin a před obsahem v části s podrobnostmi jsou příliš velké.
- Pohlaví se označuje jednotlivými písmeny *m* a *f*. Zkratky nejsou nutné a lepší by byla plná slova.
- Pole *age_group* zobrazuje čísla na jedno desetinné místo a tato granularita je zbytečná (pravděpodobně je to výsledkem matematických operací provedených v dotazu *qryRegistration*).
- Data v sekci detaily pro každou kombinaci sportu / pohlaví / věku (*first_name*, *last_name*, *birth_date* a *age_group*) by bylo lepší zobrazit v tabulce.

Následující podkapitoly zlepšují sestavu tím, že se postupně věnují každé z těchto otázek.

Nastavení vzdálenosti mezi poli sestavy

Chceme-li upravit výšku části sestavy, přetáhneme myší okraje této části nahoru a dolů podle potřeby. Můžeme také zvýraznit záhlaví oddílu a zadat vlastnost *Výška* v oblasti *Vlastnosti* v pravé části okna nástroje Návrhář sestav.

Není možné, aby byl oddíl menší než popisky a pole, které obsahuje.

Podrobné uspořádání každé sestavy je samozřejmě otázkou vkusu. Při výběru vzdáleností dbejme na to, aby se následující skupina neobjevila příliš blízko pod předchozí skupinou. Popisky a textová pole mohou být vzdáleny od horní části oddílu, ve kterém se nacházejí. Pokud tato vzdálenost od horního okraje není požadována, lze místo záhlaví zobrazit zápatí skupiny, které zajistí určitý odstup. Takové předvolby je možné provést v dialogovém okně Řazení

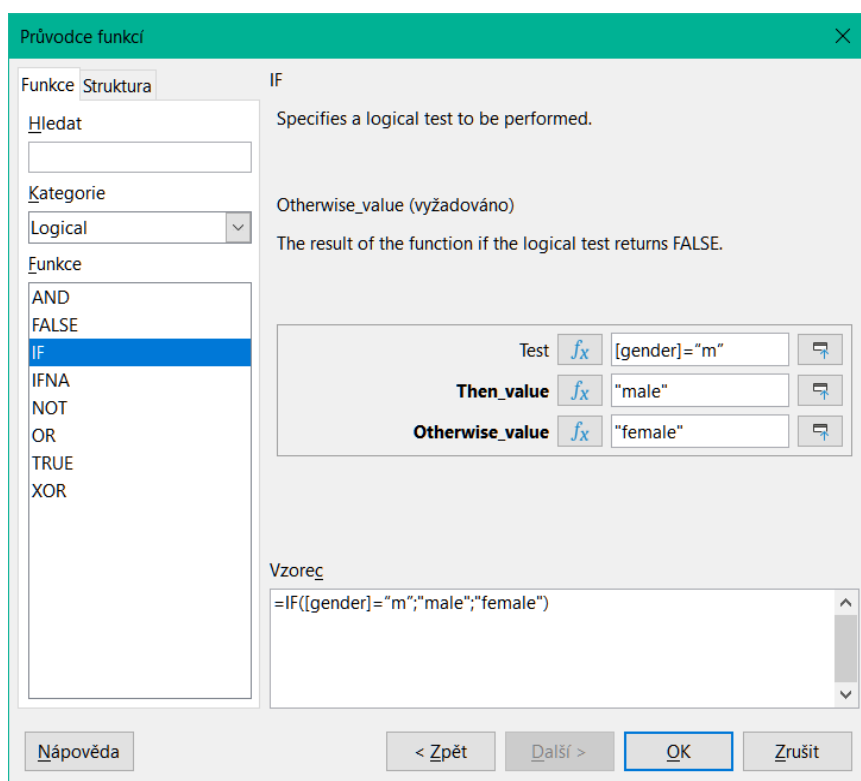
a seskupení (obrázek 80) pro každou skupinu. Pro sestavu *rptList_of_entrants* použijeme následující doporučení:

- Nastavíme výšku *Záhlaví stránky* na 1 cm.
- Nastavíme výšku záhlaví *sport_name* na 1,5 cm.
- Nastavíme výšku záhlaví *gender* na 1 cm.
- Nastavíme výšku záhlaví *age_group* na 1,7 cm.
- V současné době se nepokoušíme měnit výšku oblasti *Podrobnosti*.
- Nastavíme výšku *Zápatí stránky* na 1,3 cm.

Ovlivnění obsahu textového pole pomocí vzorce

The designation of the gender in the table is not enough for the *rptList_of_entrants* report. Přejmenování pole lze provést v dotazu. Protože však dotaz již byl vytvořen, můžeme v sestavě použít funkci.

- 1) Zvýrazníme textové pole = *gender* v záhlaví sekce *gender*.
- 2) V oblasti *Vlastnosti* na pravé straně okna nástroje Návrhář sestav klikneme na kartu *Data*.
- 3) Zkontrolujeme, zda je v rozevírací nabídce *Typ datového pole* nastavena hodnota **Pole nebo vzorec**.
- 4) Klikneme na tři tečky (...) napravo od pole *Data*.
- 5) Base zobrazí Průvodce funkcí.
- 6) V rozevírací nabídce **Kategorie** vyberte možnost *Logické*.
- 7) Dvakrát klikneme na položku **IF** v oblasti *Funkce*. Všimneme si, že po výběru detailů funkce IF na pravé straně Průvodce Funkcí se může obsah oblastí *Kategorie* a *Funkce* změnit – jedná se o prediktivní chování, které je třeba v této fázi ignorovat.
- 8) Do pole *Test* zadáme *[gender]="m"*. Tím se definuje test, který se má provést. Název pole dotazu, z něhož se data načítají, je ohraničen hranatými závorkami. Textové řetězce jsou uzavřeny v dvojitých uvozovkách.
- 9) Do pole *Then_value* zadáme *"Male"*. Pokud je pole nastaveno na „m“, nahradíme jej řetězcem „Male“ (angl. Muž).
- 10) Do pole *Další hodnota* zadáme *"Female"* (angl. Žena) (obrázek 84). Pokud pole není nastaveno na „m“, nahradíme jej řetězcem „Female“.



Obrázek 84: Použití Průvodce funkcí ke změně řetězce pohlaví

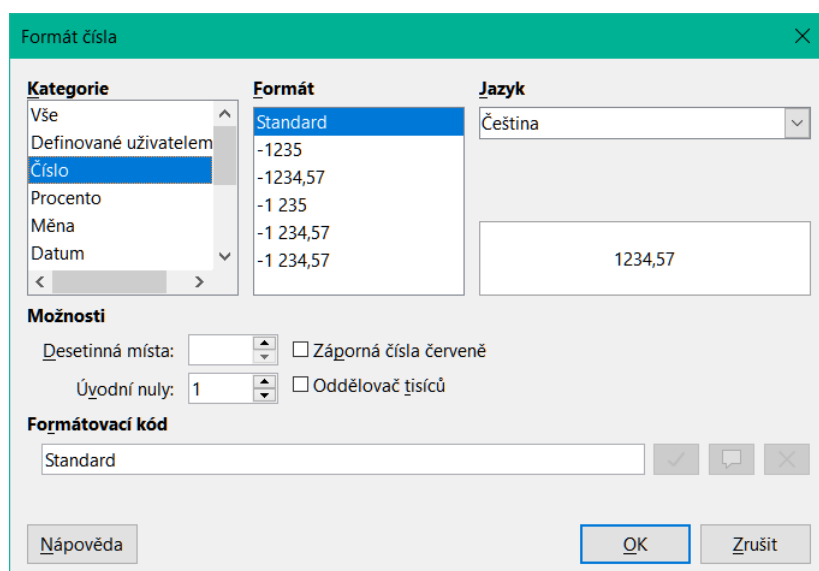
11) Klepnutím na **OK** zadání potvrdíme.

Při dalším spuštění sestavy se odpovídajícím způsobem změní znění pole *gender*.

Změna formátování textového pole

Pole, která zobrazují obsah databáze, jsou v sestavě zpočátku označena jako textová pole, ale lze je formátovat stejně jako pole v tabulkách v aplikaci Writer nebo Calc.

- 1) Zvýrazníme pole `=age_group`.
- 2) V oblasti *Vlastnosti* v pravé části okna nástroje Návrhář sestav klikneme na kartu *Obecné*.
- 3) Všimneme si, že pole *Formátování* je zpočátku nastaveno na **Text**. Klikneme na tlačítko s třemi tečkami (...) napravo od pole *Formátování*.
- 4) Base otevře dialogové okno Formát čísla. Podobná dialogová okna se používají v aplikaci Calc, Writer nebo při vytváření formulářů.
- 5) V oblasti *Kategorie* vybereme možnost **Číslo**.
- 6) Ujistíme se, že v oblasti **Formát** je zvolena volba *Obecný* (obrázek 85).



Obrázek 85: Dialogové okno Formát čísla

7) Klepneme na tlačítko **OK**.

Při dalším spuštění sestavy se formát pole věkové skupiny nyní zobrazí bez desetinných míst.

Přesouvání polí v okně Návrhář sestav

V této podkapitole je popsáno, jak sestavu trochu upravit a přesunout pole na vhodnější pozice.

Pole sestavy lze přesunout za hranice sekce do jiného sekce. V cílové sekci však musí být pro pole dostatek místa. Žádná část jednoho pole nemůže existovat na stejném místě jako část jiného pole.

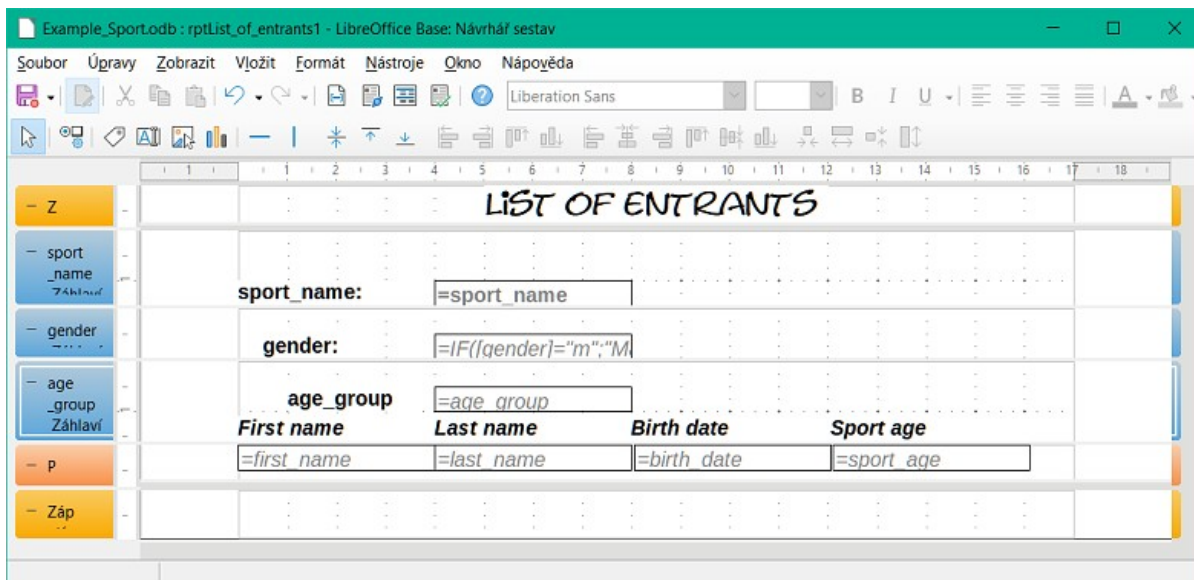
Umístění polí pomocí myši je nepřesné, proto je třeba v každé sekci zajistit dostatek místa pro pole, která tam chceme umístit. Po přetažení pole na novou pozici pomocí myši lze přesnějšího umístění dosáhnout pomocí kláves se šipkami. Dále můžeme pro umístění polí použít klávesnici, upravit hodnoty *Pozice X* a *Pozice Y* na kartě *Obecné* v oblasti *Vlastnosti*.

Popisek se přidá do oblasti *Záhlaví stránky* výběrem možnosti **Vložit > Ovládací prvky sestavy > Popisek** v hlavní nabídce. Text zobrazený na popisku je definován v poli *Popisek* v oblasti *Vlastnosti* s písmo lze vybrat v poli *Písmo* ve stejné oblasti.

Čtyři popisky původně umístěné v oblasti *Podrobnosti* lze přesunout tak, aby tvořily řadu ve spodní části oblasti záhlaví *age_group*. Čtyři textové oblasti v oblasti *Podrobnosti* lze nyní uspořádat do řady, každou pod odpovídajícím popiskem. Po dokončení lze výšku oblasti *Podrobnosti* snížit na 0,5 cm.

Chceme-li kompletně skrýt sekci *Zápatí stránky*, nastavíme *Viditelnost* na **Ne** v oblasti **Vlastnosti**. Spodní okraj dokumentu obsahuje příliš mnoho místa. Nezapomeneme, že dostupná výška je již snížena o velikost okrajů stránky.

Naše vylepšená sestava může mít podobnou strukturu jako na obrázku 86.



Obrázek 86: Upravená sestava rptList_of_entrants

Jak je vidět na obrázku 87, tyto změny zlepšily rozložení výstupu sestavy.

List of Entrants			
Sport:	discus throw		
Gender:	Female		
Age Group:	30		
Last Name	First Name	Date of Birth	Sport Age
McCoy	Jessica	1986-02-12	36
Pike	Patricia	1985-12-02	37
Gender:	Male		
Age Group:	20		
Last Name	First Name	Date of Birth	Sport Age
Putnam	Stephen	2002-06-14	20
Brown	Jerry	2002-05-13	20
Cook	Dennis	1995-04-02	27
Age Group:	30		
Last Name	First Name	Date of Birth	Sport Age
Ramirez	Gary	1987-09-11	35
Austin	Norman	1988-07-16	34
Gibbs	Zayne	1992-02-08	30
Age Group:	40		
Last Name	First Name	Date of Birth	Sport Age
McClure	Roger	1981-09-04	41
Lockwood	Andrew	1981-12-27	41

Obrázek 87: Prvních několik řádků aktualizovaného výstupu z rptList_of_entrants

Další informace o možnostech formátování sestav najdeme v kapitole 6, Sestavy.

Rozšíření databáze Example_Sport.odt

Zde uvedená příkladová databáze je pouze prvním krokem pro databázi, která má být použita ve sportovním odvětví. Pro databázi by bylo možné navrhnout mnoho užitečných rozšíření:

- Mohla by být přidána další pole pro případné další informace. Vhodným místem pro přidání takových polí je tabulka tblRel_entrant_sport.

- Pokud provozujeme několik soutěží pro stejný sport, můžeme do tabulky *tblRel_entrant_sport* zahrnout pole s datem nebo jiné pole, které lze přiřadit ke konkrétní soutěži. Pole se pak stane součástí primárního klíče tabulky.
- Možná by se mohl přidat i klub, jehož je účastník členem. Stačilo by přidat pole do tabulky *tblEntrants*. Pokud může mít více klubů stejný název, doporučuje se přidat samostatnou tabulku *tblClub* a také odpovídající cizí klíč v tabulce *tblEntrants*.
- Stejně jako u všech soutěží je třeba určit, kdo by měl být zařazen do které věkové skupiny a sportu. Zde je nutné třídění, které může opět skončit jako sestava se seznamem výsledků.
- Bylo by užitečné navrhnout sestavu, na jejímž základě by každý účastník obdržel atraktivně zpracovaný certifikát s osobním výkonem a umístěním.

Tato a mnohá další rozšíření jsou možná pomocí metod popsanych v dalších kapitolách této příručky.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 2

Vytvoření databáze

Úvod

Kapitola 1, Úvod do aplikace Base, popisuje, jak Base poskytuje grafické uživatelské rozhraní pro databáze.

Do databáze Base jsou integrovány dva relační databázové stroje, HSQLDB (HyperSQL DataBase) verze 1.8.0 a Firebird verze 3.0.7, přičemž druhý jmenovaný je v současné době dostupný pouze jako experimentální funkce. Databáze vytvořené pomocí vestavěného stroje jsou přístupné vždy pouze jednomu uživateli a všechna data pro databázi jsou uložena v souboru ODB. Základy vytváření vestavěné databáze jsou popsány v kapitole 8 příručky *Začínáme s LibreOffice*, *Začínáme s Base*, a v kapitole 1 této knihy. Příklady databází v této knize se vztahují k HSQLDB, ale jsou upraveny tak, aby většina funkcí byla přímo přenositelná do Firebirdu. V případě potřeby jsou identifikovány alternativy v programu Firebird.

Base podporuje také připojení k mnoha typům externích databází. Ty mohou být umístěny buď lokálně, nebo ve vzdáleném síťovém umístění a mohou podporovat více souběžných uživatelů. Než může Base navázat připojení k externí databázi, musí databáze existovat a případně být nakonfigurována tak, aby umožňovala síťové připojení s konkrétními přihlašovacími údaji.

Tato kapitola popisuje, jak vytvořit novou (vestavěnou) databázi pomocí vestavěného modulu HSQLDB, jak vytvořit připojení k několika různým typům externích databází a jak tato připojení později v případě potřeby upravit.



Poznámka

Vestavěný engine HSQLDB je napsán v jazyce Java. Pokud vložená databáze HSQLDB obsahuje velké množství dat, může použití Javy někdy vést ke snížení rychlosti. Tento faktor je jednou z motivací probíhajících prací na převodu výchozího vestavěného databázového stroje z HSQLDB na Firebird.

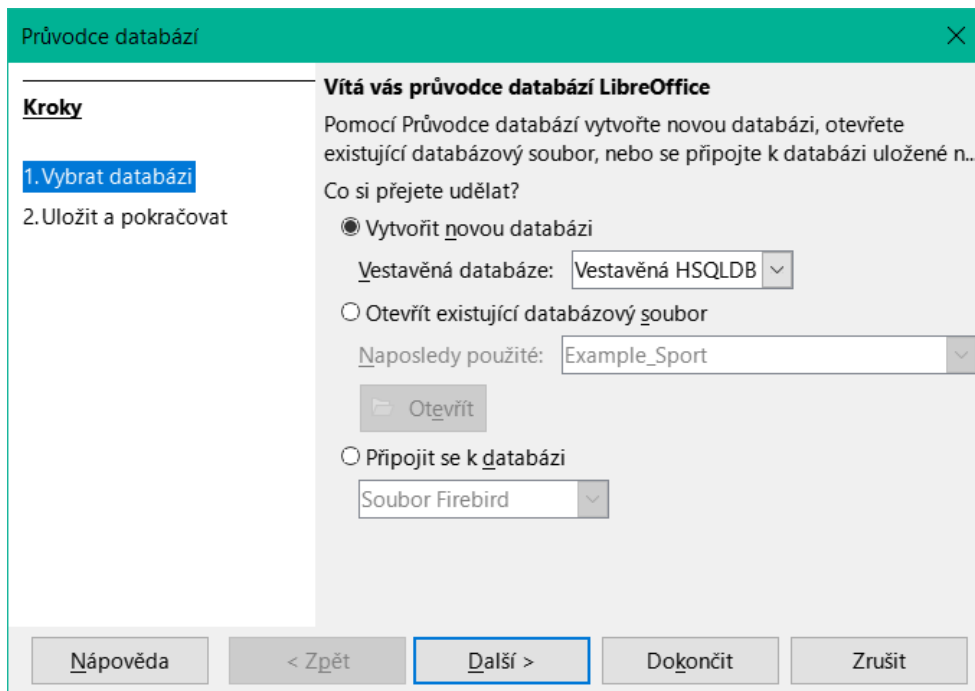
Podobné efekty mohou být pozorovány při přístupu k externím databázím, které nepoužívají Javu, v závislosti na použitém ovladači. Pokud je připojení realizováno prostřednictvím JDBC, což je technologie založená na Javě, může to mít negativní vliv na výkon.

Vytvoření databáze pomocí vestavěného databázového stroje

Pokud plánujeme databázi pro jednoho uživatele nebo pokud chceme pouze získat první zkušenosti s databázím, vestavěný engine HSQLDB bude vyhovovat našim potřebám. Pokud se později naše cíle změní, je možné databázi přenést na externí server, kde může být výkonnější a kde může mít k databázi souběžný přístup více uživatelů.

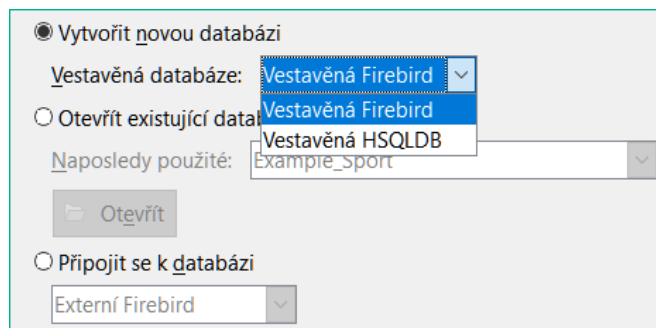
Chceme-li vytvořit vloženou databázi, provedeme následující kroky.

- 1) Přístup k Průvodci databází získáme provedením jedné z následujících akcí:
 - a) V Centru spouštění LibreOffice klepneme na možnost **Databáze Base** v oblasti *Vytvořit*.
 - b) V libovolné komponentě LibreOffice vybereme na nástrojové liště **Soubor > Nový > Databáze**.
 - c) V okně Base klepneme na ikonu **Nový** na Standardní nástrojové liště nebo použijeme odpovídající klávesovou zkratku *Ctrl + N*.
 - d) V libovolné komponentě LibreOffice klepneme na šipku dolů napravo od ikony **Nový** na Standardní nástrojové liště a v rozevírací nabídce vybereme **Databáze**.
- 2) Aplikace Base zobrazí Krok 1 (*Vybrat databázi*) Průvodce databází (obrázek 88).



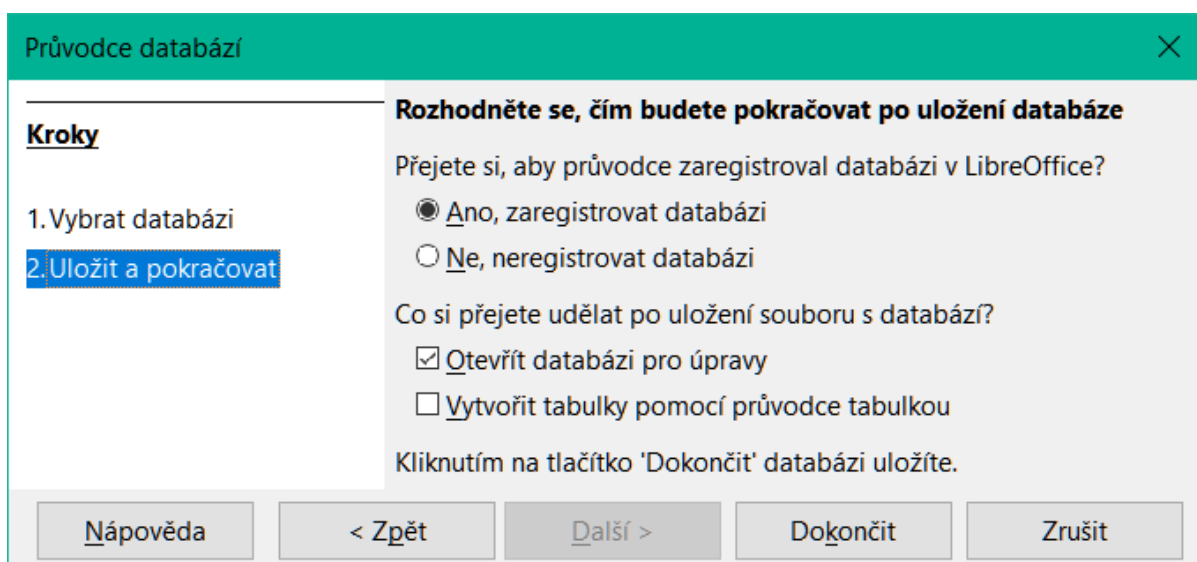
Obrázek 88: Krok 1 Průvodce databází: Výběr databáze

- 3) Klepneme na možnost **Vytvořit novou databázi**. Ve výchozím nastavení (s vypnutými experimentálními funkcemi) vytvoří Base databázi HSQLDB, což se projeví v nastavení rozevřací nabídky *Vestavěná databáze*, která neobsahuje žádné další možnosti. Pokud je však v dialogovém okně **Nástroje > Možnosti > LibreOffice > Pokročilé** vybrána možnost **Povolit experimentální funkce (mohou být nestabilní)**, chování Průvodce databází se změní tak, že nabídka *Vestavěná databáze* obsahuje dvě možnosti (obrázek 89), přičemž výchozí je možnost **Firebird Embedded**.



Obrázek 89: Nabídka *Vestavěná databáze* s povolenými experimentálními funkcemi

- 4) Klikneme na tlačítko **Další >**.
- 5) Aplikace Base zobrazí krok 2 (*Uložit a pokračovat*) Průvodce databází (obrázek 90).



Obrázek 90: Krok 2 Průvodce databází: Uložit a pokračovat

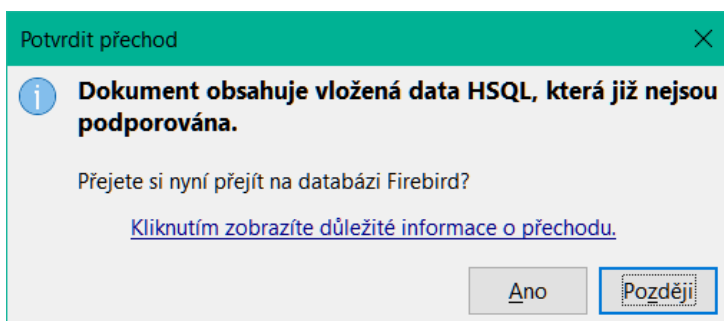
- 6) Ve výchozím nastavení je vybrána možnost **Ano, zaregistrovat databázi**. Registrovanou databázi mohou využívat další součásti LibreOffice, například pro sloučení dopisů v aplikaci Writer. Doporučuje se, aby všechny databáze byly při vytvoření zaregistrovány, ale konečná volba je na uživateli.
- 7) Ve výchozím nastavení je vybrána možnost **Otevřít databázi pro úpravy**, která je obvykle vhodná pro novou databázi. Pokud si v tuto chvíli nepřejeme databázi upravovat, můžeme tuto možnost zrušit.
- 8) Ve výchozím nastavení je možnost **Vytvořit tabulky pomocí průvodce tabulkou** zrušena. Použití Průvodce tabulkou bylo popsáno v kapitole 8 příručky *Začínáme s LibreOffice*, *Začínáme s aplikací Base*. Ve zbytku této knihy se záměrně vyhneme popisu použití průvodců pro vytváření tabulek, dotazů apod. Uživatel však může tuto možnost zvolit podle potřeby.
- 9) Klepnutím na tlačítko **Dokončit** uložíme novou databázi.
- 10) Base zobrazí dialogové okno Uložit jako, ve kterém je třeba zadat název a umístění souboru ODB.
- 11) Klepneme na tlačítko **Uložit**.

Vložená databáze je připravena pro vkládání záznamů a ukládání pohledů, dotazů, formulářů a sestav. Všimněme si, že soubor je uložen předtím, než jsme zadali jakékoli datové položky, což je odlišný přístup než v jiných komponentách LibreOffice.

Upozornění

Pokud jsou zapnuty experimentální funkce pomocí volby **Povolit experimentální funkce (mohou být nestabilní)** v dialogovém okně **Nástroje > Možnosti > LibreOffice > Pokročilé** a později otevřeme existující vestavěnou databázi HSQLDB, pak aplikace Base může zobrazit dialogové okno Potvrdit přechod (obrázek 91) s dotazem, jestli chceme databázi zmigrovat do databáze formátu Firebird. Odkaz v tomto dialogovém okně vede na následující stránku na wiki nadace The Document Foundation:
<https://wiki.documentfoundation.org/Documentation/HowTo/MigrateFromHSQLDB>.

Tato experimentální funkce automatické migrace se spustí vždy, když je otevřena vložená databáze HSQLDB a je vytvořeno první připojení k datovému schématu. Pokud se s tímto dialogovým oknem setkáme, doporučujeme maximální opatrnost, protože na vývoji této funkce se stále pracuje. Nepotvrzujeme pouze klepnutím na **Ano**; nezapomeneme klepnout na **Později**. Před pokusem o migraci se ujistíme, že jsme provedli přípravy popsané na odkazované stránce wiki.



Obrázek 91: Dialogové okno Potvrdit přechod

Přístup k externím databázím

✓ Poznámka

Při použití aplikace Base k navázání připojení přes Internet se ujistíme, že víme, jak je toto připojení strukturováno. Každá databáze, ke které se přistupuje přes internet, by měla být chráněna specifickým uživatelským jménem a heslem. Ptejme se například: Je spojení šifrované / zabezpečené? Jak se heslo přenáší? Jsou moje data v bezpečí?

Úvod

Externí databáze musí existovat, aby k ní bylo možné přistupovat pomocí Base. Pokud je požadován přístup k externí databázi, musí být nastavena tak, aby umožňovala síťová připojení s konkrétními přihlašovacími údaji, a teprve poté se k ní může připojit jakýkoli jiný software (například Base).

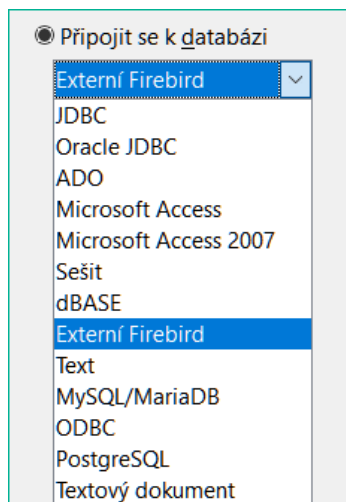
Pokud je externí databáze vhodně nastavena, můžeme v závislosti na typu databáze a dostupném softwaru ovladače vytvářet tabulky, zadávat data a dotazovat se na data. Některé externí zdroje dat (například připojení k tabulkovému procesoru nebo textové databázi) neumožňují zadávání nových dat a slouží pouze k vyhledávání nebo vykazování existujících dat.

Chceme-li se připojit k externí databázi, postupujeme podle následujících kroků.

- 1) Průvodce databází otevřeme provedením jedné z akcí uvedených v kroku 1) postupu pro vytvoření nové databáze pomocí vestavěného databázového stroje (strana 77).
- 2) V kroku 1 (*Výběr databáze*) Průvodce databází (obrázek 88), klepneme na možnost **Připojit se k databázi**. Typy databází uvedené v přilehlé rozevírací nabídce se liší v závislosti na operačním systému a dalších faktorech, ale vždy by měly být k dispozici následující:
 - dBASE
 - Firebird
 - JDBC
 - MySQL
 - ODBC

- Oracle JDBC
- PostgreSQL
- Sešit
- Text
- Textový dokument.

Obrázek 92 je příkladem možností dostupných pro systém se systémem Windows.



Obrázek 92: Možnosti připojení k externí databázi na platformě Windows

3) Vybereme položku požadovaného typu připojení a klepneme na tlačítko **Další >**.

Počet zbývajících kroků a možnosti volby v každém kroku závisí na povaze vybraného typu připojení. Nastavení připojení můžeme změnit po vytvoření přidruženého souboru ODB.

Popisy v následujících kapitolách této knihy popisují především použití databáze Base ve spojení s vestavěnou databází HSQLDB. Většinu práce s návrhem databáze lze rozšířit na databáze využívající Firebird, MySQL, PostgreSQL atd.

Ve zbývajících částech této kapitoly jsou uvedeny příklady navázání připojení k typickým externím databázím. Někteří uživatelé mohou chtít použít Base pro připojení k existující víceuživatelské databázi a měli by se poradit se správcem databáze své organizace, aby získali přístup k RDBMS. Jiní uživatelé si mohou nainstalovat systém RDBMS pro jednouživatelský přístup na svůj vlastní počítač a pak používat Base jako front-end. Pokrytí všech možných případů použití přesahuje rámec tohoto dokumentu.

✓ Poznámka

V závislosti na použitém ovladači může být na připojeném serveru možné přistupovat k několika databázím současně. Pokud je například navázáno připojení k databázi MySQL/MariaDB, mohou být v Base současně zobrazeny další databáze na serveru, ke kterým má zadáný uživatel přístupová práva. Je tedy možné kopírovat data z jedné databáze do druhé, provádět související dotazy atd.

Databáze MySQL/MariaDB

Přehled

MySQL je open-source systém pro správu relačních databází (RDBMS), který poskytuje společnost Oracle Corporation. Další informace najdeme na webových stránkách MySQL na

adrese <https://www.mysql.com/>. Aktuální verze referenční příručky MySQL je dostupná prostřednictvím odkazu *Documentation* na webových stránkách MySQL.

MariaDB je open-source RDBMS, který je kompatibilní s MySQL. Na vývoj MariaDB dohlíží nadace MariaDB Foundation a další informace najdeme na <https://mariadb.com/>. Dokumentaci serveru MariaDB najdeme na adrese <https://mariadb.com/kb/en/documentation/>.

Base se může připojit k databázím MySQL a MariaDB jedním ze tří způsobů:

- Přímé připojení pomocí vestavěného konektoru. Jedná se o nejjednodušší a nejrychlejší metodu.
- Připojení pomocí ODBC (Open Database Connectivity).
- Připojení pomocí JDBC (Java Database Connectivity).

Počáteční příprava

Podrobné pokyny ke stažení a instalaci softwaru databázového serveru pro vaši konfiguraci počítače a podrobnosti o přístupu ke konzole SQL naleznete v Referenční příručce MySQL nebo v dokumentaci serveru MariaDB. Po instalaci postupujeme podle pokynů pro nastavení uvedených v dokumentaci. Mezi ně mohou patřit:

- 1) V MySQL i MariaDB je uživatelské jméno účtu správce *root* (všimneme si, že jméno *root* v tomto kontextu nesouvisí s účtem superuživatele *root*, který je k dispozici v některých operačních systémech). Ujistíme se, že heslo pro účet *root* databázového serveru bylo nastaveno na řetězec, který je vhodný pro zachování bezpečnosti dat.
- 2) Některé uživatelské účty mohou být ve výchozím nastavení k dispozici pro testovací účely a některé testovací databáze mohou umožňovat anonymní přístup. Jako další bezpečnostní opatření se ujistíme, že jsou odstraněny všechny anonymní uživatelské účty.

Před navázáním připojení musí na serveru existovat databáze, která vás zajímá. Pokud naše databáze ještě neexistuje, provedeme následující kroky k jejímu vytvoření.

- 1) Chceme-li vytvořit (prázdnou) databázi s názvem *libretest*, zadáme do konzoly SQL následující příkaz:

```
CREATE DATABASE libretest;
```

- 2) Chceme-li udělit všechna práva v nové databázi novému uživateli s názvem *lotest* a heslem *libre*, zadáme v konzole SQL následující příkazy:

```
CREATE USER 'lotest' IDENTIFIED BY 'libre';  
GRANT ALL PRIVILEGES ON libretest.* TO 'lotest';  
FLUSH PRIVILEGES;
```

Po úspěšném dokončení těchto kroků je nová databáze k dispozici a Base se k ní může připojit, jak je popsáno v následujících podkapitolách.

Přímé připojení



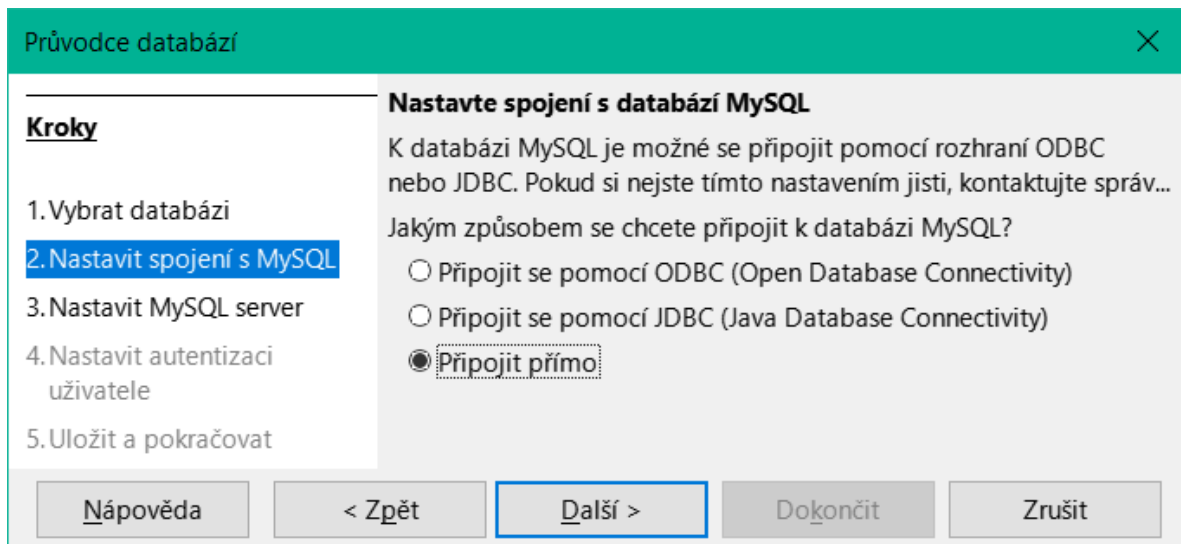
Poznámka

V dřívějších verzích Base (před verzí LibreOffice 6.2) vyžadovalo přímé připojení k MySQL nebo MariaDB instalaci rozšíření. Od verze LibreOffice 6.2 je software potřebný k přímému připojení integrován do aplikace Base.

Pokud chceme poprvé navázat přímé připojení k určité databázi MySQL nebo MariaDB, provedeme následující kroky.

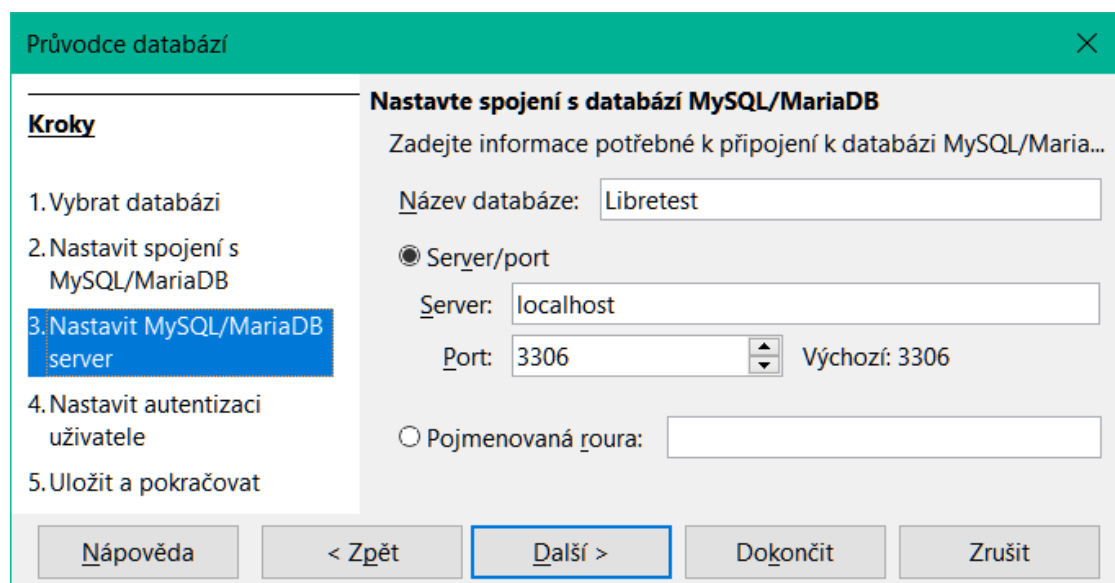
- 1) Průvodce databázím otevřeme provedením jedné z akcí uvedených v kroku 1) postupu pro vytvoření nové databáze pomocí vestavěného databázového stroje (strana 77).

- 2) Aplikace Base zobrazí Krok 1 (*Vybrat databázi*) Průvodce databází (obrázek 88).
- 3) Klepneme na možnost **Připojit se k databázi**. Base aktivuje přílehlou rozbalovací nabídku se seznamem dostupných možností (obrázek 92).
- 4) V rozevírací nabídce vybereme možnost **MySQL**.
- 5) Klikneme na tlačítko **Další >**.
- 6) Base zobrazí krok 2 (*Nastavit spojení s MySQL*) Průvodce databází (obrázek 93).



Obrázek 93: Krok 2 Průvodce databází: Nastavení připojení k MySQL

- 7) Klepneme na možnost **Připojit přímo**.
- 8) Klikneme na tlačítko **Další >**.
- 9) Základna zobrazí krok 3 (*Nastavit MySQL server*) Průvodce databází (obrázek 94).



Obrázek 94: Krok 3 Průvodce databází: Nastavení dat serveru MySQL

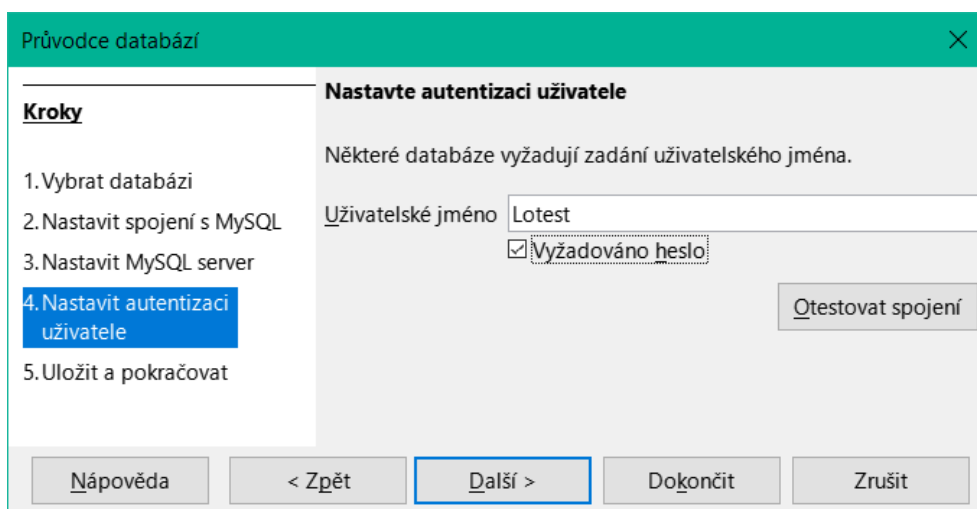
- 10) Do pole *Název databáze* zadáme název databáze. Naše ukázková databáze se jmenuje *libretest*.
- 11) Klepneme na možnost **Server/port**.

- 12) Do pole *Server* zadáme název serveru, na kterém běží databáze MySQL nebo MariaDB. Pokud se připojujeme k databázi na stejném počítači, název serveru by měl být *localhost*. Pokud se připojujeme ke vzdálenému serveru, může to být IP adresa, název síťového počítače nebo internetová adresa.
- 13) Do pole *Port* zadáme číslo portu databázového serveru. Výchozí číslo portu pro databáze MySQL a MariaDB je 3306.

Tip

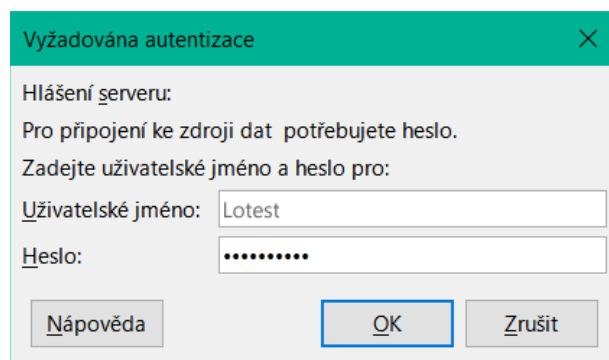
V krocích 10) až 13), pokud si nejsme jisti správností zadávaných informací, požádáme o radu správce databáze.

- 14) Klikneme na tlačítko **Další >**.
- 15) Base zobrazí krok 4 (*Nastavit autentizaci uživatele*) Průvodce databází (obrázek 95).



Obrázek 95: Krok 4 Průvodce databází: Nastavení ověřování uživatelů

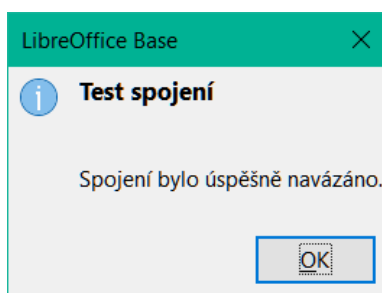
- 16) Do pole *Uživatelské jméno* zadáme platné uživatelské jméno. Náš příklad uživatele se jmenuje *lotest*.
- 17) Klepneme na možnost **Vyžadováno heslo**.
- 18) Klepneme na tlačítko **Otestovat spojení**.
- 19) Base zobrazí dialogové okno *Požadované ověření* (obrázek 96).



Obrázek 96: Dialogové okno *Vyžadována autentizace*

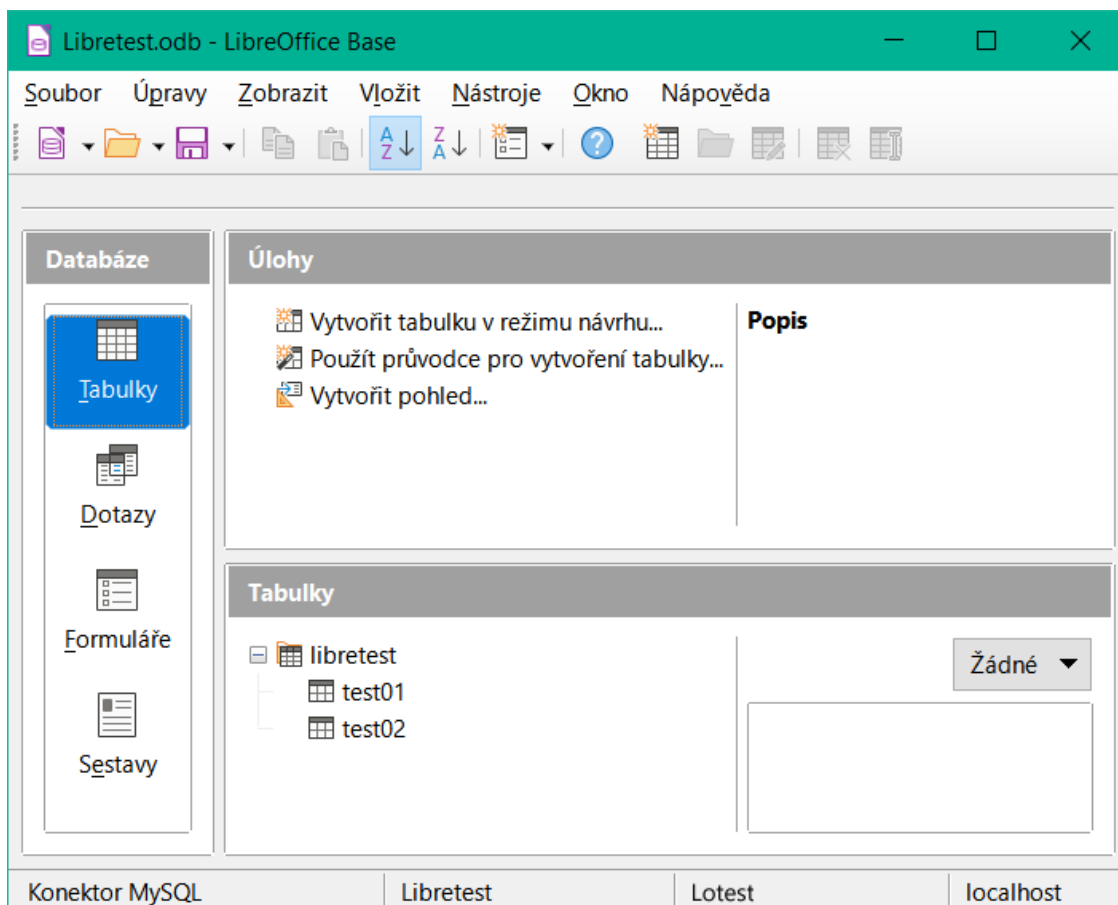
- 20) Zadáme uživatelské jméno (*lotest*) a heslo (*libre*) potřebné pro přístup do databáze.
- 21) Klepneme na tlačítko **OK**.

- 22) Base zobrazí dialogové okno s informací, že test připojení proběhl úspěšně (obrázek 97). Pokud byl test neúspěšný (například databázový server není právě spuštěn nebo jsme zadali nesprávné přihlašovací údaje), zobrazí se místo toho odpovídající chybová zpráva.



Obrázek 97: Dialogové okno úspěšného testu připojení

- 23) Stisknutím tlačítka **OK** odstraníme z obrazovky dialogové okno úspěšného testu připojení.
- 24) Klikneme na tlačítko **Další >**.
- 25) Base zobrazí krok 5 (*Uložit a pokračovat*) Průvodce databází (se stejnými možnostmi, jaké jsou uvedeny na obrázku 90).
- 26) Klepneme na možnost **Ne, neregistrovat databázi**, protože se jedná o testovací databázi.
- 27) Klepneme na možnost **Otevřít databázi pro úpravy**, protože tato možnost je obvykle vhodná pro novou databázi. Pokud si v tuto chvíli nepřejeme databázi upravovat, můžeme tuto možnost zrušit.
- 28) Pokud nechceme použít Průvodce tabulkou, ponecháme volbu **Vytvořit tabulky pomocí průvodce tabulkou** ve výchozím nezaškrtnutém stavu.
- 29) Klepneme na tlačítko **Dokončit**.
- 30) Base zobrazí dialogové okno Uložit jako, ve kterém je třeba zadat název a umístění souboru ODB.
- 31) Klepneme na tlačítko **Uložit**.
- 32) Průvodce databází se odstraní z obrazovky, vytvoří se soubor ODB obsahující informace o připojení k databázi a Base aktualizuje své hlavní okno tak, aby se pod názvem databáze zobrazovaly informace o tabulkách databáze (obrázek 98).



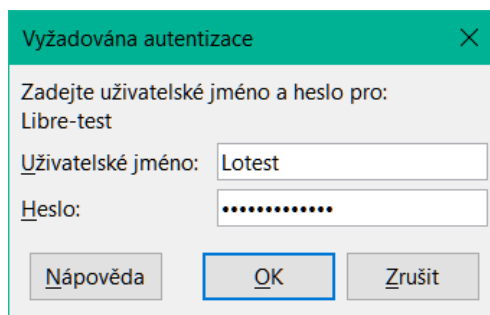
Obrázek 98: Hlavní okno Base zobrazující tabulky v databázi libretest

Tabulky lze vytvářet, upravovat a odstraňovat.

Stavový řádek ve spodní části obrázku 98 zobrazuje:

- Název konektoru použitého pro připojení k databázi MySQL nebo MariaDB (v tomto případě *Konektor MySQL*).
- Název otevřené databáze MySQL nebo MariaDB. V tomto případě mají soubor ODB a databáze MySQL společný název (*libretest*), ale nemusí tomu tak být.
- Uživatelské jméno, které se používá pro připojení k databázi MySQL nebo MariaDB (v tomto případě *lotest*).
- Název serveru, na kterém je spuštěna databáze MySQL nebo MariaDB (v tomto případě *localhost*).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k základní databázi MySQL nebo MariaDB, zobrazí se dialogové okno Vyžadována autentizace (obrázek 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.



Obrázek 99: Dialogové okno Vyžadována autentizace

Připojení pomocí ODBC

Počáteční příprava

Chceme-li se pomocí aplikace Base připojit k databázovému serveru MySQL nebo MariaDB prostřednictvím aplikačního programovacího rozhraní (API) ODBC (Open Database Connectivity), musí být v počítači nainstalován správně nakonfigurovaný ovladač ODBC. Pokud je databázový server v počítači již nainstalován, může být nainstalován i konektor ODBC. Pokud tento konektor není v počítači aktuálně k dispozici, budeme k instalaci nového softwaru potřebovat administrátorská práva.

Chceme-li se připojit k databázi MySQL prostřednictvím ODBC, doporučujeme použít MySQL Connector/ODBC. Správnou verzi konektoru ODBC pro náš počítač si můžeme stáhnout z webu MySQL Community Downloads na adrese <https://dev.mysql.com/downloads/connector/odbc/>. Pokyny k instalaci v závislosti na platformě nalezneme v příručce MySQL Connector/ODBC Developer Guide na adrese <https://dev.mysql.com/doc/connector-odbc/en/>.

Pro připojení k databázi MariaDB prostřednictvím ODBC se doporučuje použít MariaDB Connector/ODBC. Správnou verzi konektoru ODBC pro náš počítač si můžeme stáhnout z oblasti Downloads na webových stránkách MariaDB na adrese <https://mariadb.com/downloads/connectors/connectors-data-access/odbc-connector>. Pokyny k instalaci závislé na platformě najdeme ve znalostní databázi MariaDB na adrese <https://mariadb.com/kb/en/about-mariadb-connector-odbc/>.

Při vytváření názvu zdroje dat ODBC (DSN) pro naši databázi postupujeme podle pokynů uvedených v dokumentaci ke konektoru. Pro účely příkladu v následujícím pododdíle předpokládáme, že zdroj dat byl nakonfigurován takto:

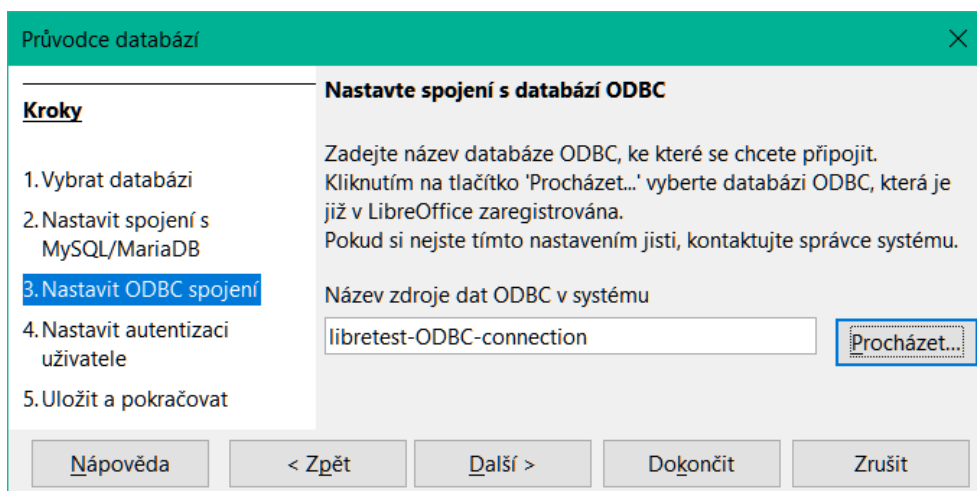
- Je vybrán ovladač MySQL ODBC Unicode.
- Název zdroje dat je nastaven na *libretest-ODBC-connection*.
- Název databáze (*libretest*) je obsažen v definici zdroje dat.
- Přihlašovací údaje pro databázový server nejsou ve zdroji dat definovány.

Použití Průvodce databází

Když chceme poprvé vytvořit připojení ODBC k určité databázi MySQL nebo MariaDB, provedeme následující kroky.

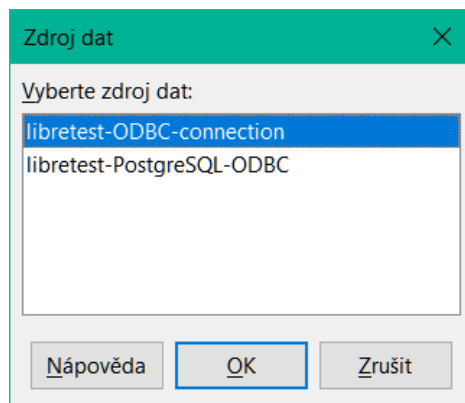
- 1) Průvodce databází otevřeme provedením jedné z akcí uvedených v kroku 1) postupu pro vytvoření nové databáze pomocí vestavěného databázového stroje (strana 77).
- 2) Aplikace Base zobrazí Krok 1 (*Vybrat databázi*) Průvodce databází (obrázek 88).
- 3) Klepneme na možnost **Připojit se k databázi**. Base aktivuje přílehlou rozbalovací nabídku se seznamem dostupných možností (obrázek 92).
- 4) V rozevírací nabídce vybereme možnost **MySQL**.

- 5) Klikneme na tlačítko **Další >**.
- 6) Base zobrazí krok 2 (*Nastavit spojení s MySQL*) Průvodce databází (obrázek 93).
- 7) Klepneme na možnost **Připojit se pomocí ODBC (Open Database Connectivity)**.
- 8) Klikneme na tlačítko **Další >**.
- 9) Base zobrazí krok 3 (*Nastavit ODBC spojení*) Průvodce databází (obrázek 100).



Obrázek 100: Krok 3 Průvodce databází: Nastavení připojení ODBC

- 10) Zadáme název dříve definovaného zdroje dat ODBC, který má být použit pro přístup k databázi MySQL nebo MariaDB (v našem příkladu *libretest-ODBC-connection*). Klepnutím na tlačítko **Procházet** zobrazíme dialog Zdroj dat (obrázek 101), v němž můžeme vybrat hledaný zdroj dat ze seznamu dostupných zdrojů a výběr potvrdit klepnutím na tlačítko **OK**.



Obrázek 101: Dialogové okno Zdroj dat

- 11) Klikneme na tlačítko **Další >**.
- 12) Base zobrazí krok 4 (*Nastavit autentizaci uživatele*) Průvodce databází (podobně jako na obrázku 95).
- 13) Do pole *Uživatelské jméno* zadáme platné uživatelské jméno. Náš příklad uživatele se jmenuje *lotest*.
- 14) Klepneme na možnost **Vyžadováno heslo**.
- 15) Pokud je potřeba, otestujeme připojení provedením kroků 18) až 23) postupu přímého připojení (strana 84).

- 16) Klikneme na tlačítko **Další >**.
- 17) Base zobrazí krok 5 (*Uložit a pokračovat*) Průvodce databází (se stejnými možnostmi, jaké jsou uvedeny na obrázku 90).
- 18) Klepneme na možnost **Ne, neregistrovat databázi**, protože se jedná o testovací databázi.
- 19) Klepneme na možnost **Otevřít databázi pro úpravy**, protože tato možnost je obvykle vhodná pro novou databázi. Pokud si v tuto chvíli nepřejeme databázi upravovat, můžeme tuto možnost zrušit.
- 20) Pokud nechceme použít Průvodce tabulkou, ponecháme volbu **Vytvořit tabulky pomocí průvodce tabulkou** ve výchozím nezaškrtnutém stavu.
- 21) Klepneme na tlačítko **Dokončit**.
- 22) Base zobrazí dialogové okno Uložit jako, ve kterém je třeba zadat název a umístění souboru ODB.
- 23) Klepneme na tlačítko **Uložit**.
- 24) Průvodce databází se zavře, vytvoří se soubor ODB obsahující informace o připojení k databázi a Base aktualizuje své hlavní okno tak, aby se pod názvem databáze zobrazovaly informace o tabulkách databáze (podobně jako na obrázku 98).

Tabulky lze vytvářet, upravovat a odstraňovat.

V tomto případě se na stavovém řádku zobrazí následující informace:

- Název konektoru používaného k připojení k databázi MySQL nebo MariaDB (například *MySQL (ODBC)*).
- Název zdroje dat ODBC, v tomto případě *libretest-ODBC-connection*.
- Uživatelské jméno, které se používá pro připojení k databázi MySQL nebo MariaDB (v tomto případě *lotest*).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k základní databázi MySQL nebo MariaDB, zobrazí se dialogové okno Vyžadována autentizace (obrázek 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

Připojení pomocí JDBC

Počáteční příprava

Chceme-li se pomocí Base připojit k databázovému serveru MySQL nebo MariaDB prostřednictvím rozhraní pro programování aplikací (API) Java Database Connectivity (JDBC), měl by být v počítači umístěn příslušný konektor JDBC. Pokud je databázový server v počítači již nainstalován, může být přítomen i konektor JDBC. Pokud konektor není v našem počítači aktuálně k dispozici, je třeba jej stáhnout a umístit konkrétní soubor JAR (Java ARchive) do přístupné složky.

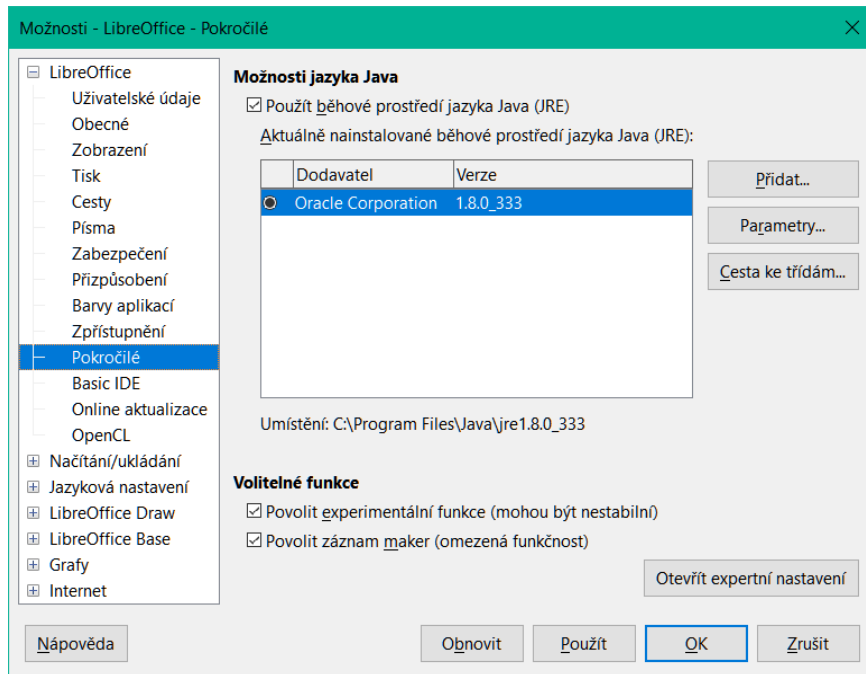
Pro připojení k databázi MySQL prostřednictvím JDBC se doporučuje použít MySQL Connector/J. Správnou verzi konektoru JDBC pro náš počítač si můžeme stáhnout z webu MySQL Community Downloads na adrese <https://dev.mysql.com/downloads/connector/j/>. Pokyny k instalaci nalezneme v příručce MySQL Connector/J Developer Guide, která je dostupná na adrese <https://dev.mysql.com/doc/index-connectors.html>.

Pro připojení k databázi MariaDB prostřednictvím JDBC se doporučuje použít MariaDB Connector/J. Správnou verzi konektoru JDBC pro náš počítač si můžeme stáhnout z oblasti Downloads na webových stránkách MariaDB na adrese <https://mariadb.org/connector-java/all-releases/>. Pokyny k instalaci nalezneme ve znalostní databázi MariaDB na adrese <https://mariadb.com/kb/en/mariadb-connector-j/>.

Při stahování konektoru JDBC se ujistíme, že jsme vybrali správnou verzi pro váš operační systém, verzi databáze a verzi Javy. Kompatibilitou se zabývá dokumentace MySQL a MariaDB uvedená v předchozích odstavcích. Konektor JDBC nelze použít, pokud náš systém nepodporuje jazyk Java.

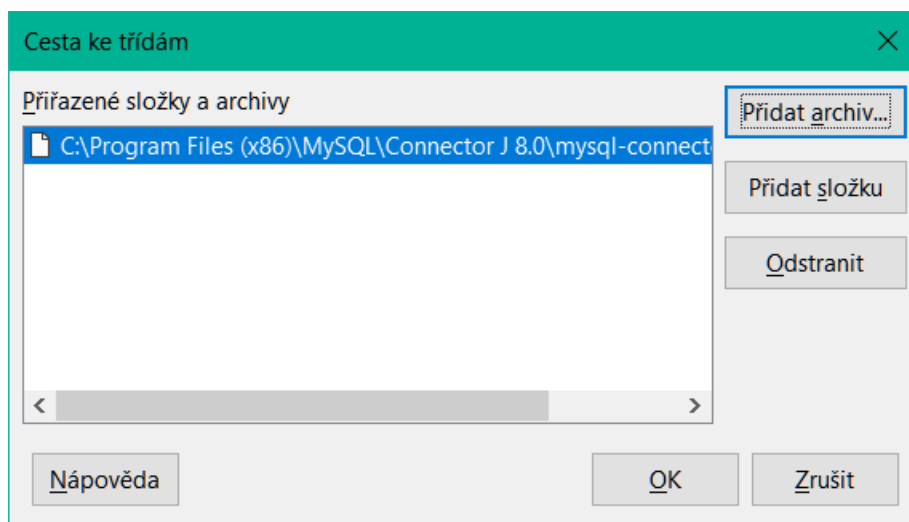
Následujícími kroky informujeme Base o tom, kde najde konektor JDBC.

- 1) V hlavní nabídce vybereme **Nástroje > Možnosti > LibreOffice > Pokročilé**.
- 2) LibreOffice zobrazí dialogové okno Možnosti – LibreOffice – Rozšířené (obrázek 102).



Obrázek 102: Možnosti – LibreOffice – dialogové okno Pokročilé

- 3) Klepneme na tlačítko **Cesta ke třídám**.
- 4) LibreOffice zobrazí dialogové okno Cesta ke třídám (obrázek 103).



Obrázek 103: Dialogové okno Cesta ke třídám

- 5) Klepneme na tlačítko **Přidat archiv**.
- 6) LibreOffice zobrazí dialogové okno Vybrat archivy (není zobrazeno).

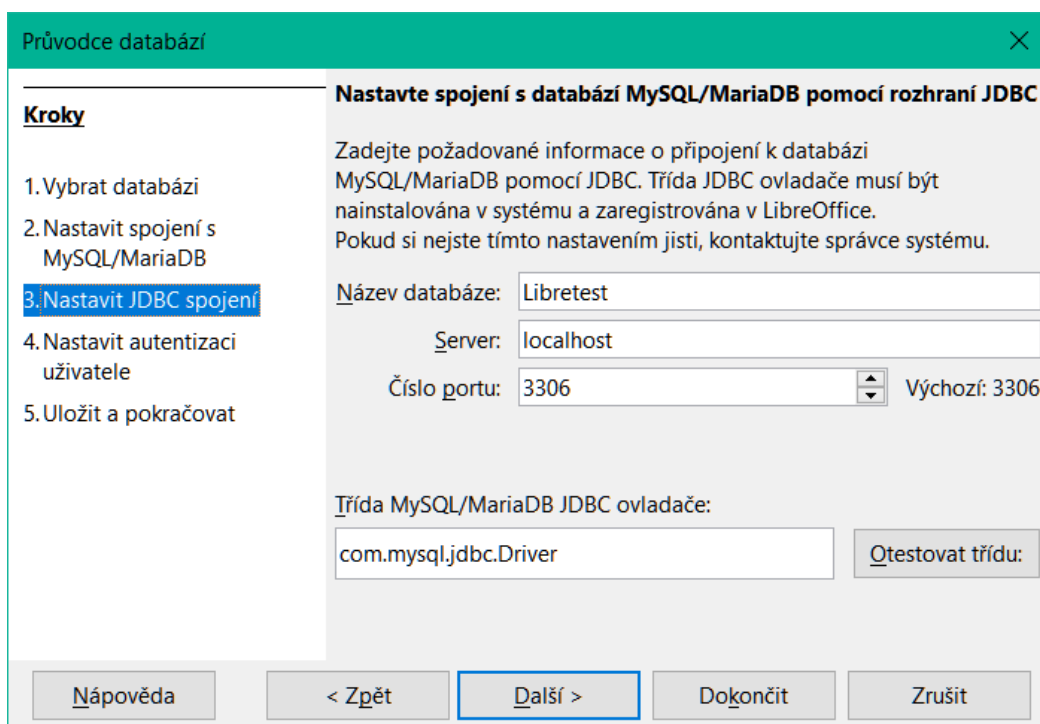
- 7) Přejdeme do složky obsahující soubor JAR pro konektor JDBC, vybereme soubor a klepneme na tlačítko **Otevřít**.
- 8) Ujistíme se, že je vybrán požadovaný soubor JAR, a v dialogovém okně Cesta ke třídám klepneme na tlačítko **OK**.
- 9) Klepneme na tlačítko **OK** v dialogovém okně Možnosti – LibreOffice – Upřesnit.

Při pozdějším pokusu o vytvoření připojení JDBC k databázi, jak je popsáno v následující podkapitole, by Base měla být schopna najít konektor JDBC.

Použití Průvodce databází

Když chceme poprvé vytvořit připojení JDBC k určité databázi MySQL nebo MariaDB, provedeme následující kroky.

- 1) Průvodce databází otevřeme provedením jedné z akcí uvedených v kroku 1) postupu pro vytvoření nové databáze pomocí vestavěného databázového stroje (strana 77).
- 2) Aplikace Base zobrazí Krok 1 (*Vybrat databázi*) Průvodce databází (obrázek 88).
- 3) Klepneme na možnost **Připojit se k databázi**. Base aktivuje přílehlou rozbalovací nabídku se seznamem dostupných možností (obrázek 92).
- 4) V rozevírací nabídce vybereme možnost **MySQL**.
- 5) Klikneme na tlačítko **Další >**.
- 6) Base zobrazí krok 2 (*Nastavit spojení s MySQL*) Průvodce databází (obrázek 93).
- 7) V případě potřeby klepneme na možnost **Připojit se pomocí JDBC (Java Database Connectivity)**. Toto je výchozí volba pro tuto sadu tří přepínačů.
- 8) Klikneme na tlačítko **Další >**.
- 9) Base zobrazí krok 3 (*Nastavit JDBC spojení*) Průvodce databází (obrázek 104).



Obrázek 104: Krok 3 Průvodce databází: Nastavení připojení JDBC

- 10) Do pole *Název databáze* zadáme název databáze. Naše ukázková databáze se jmenuje *libretest*.
- 11) Do pole *Server* zadáme název serveru, na kterém běží databáze MySQL nebo MariaDB. Pokud se připojujeme k databázi na stejném počítači, měl by být název serveru

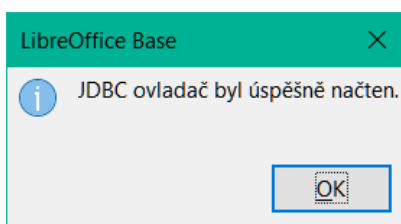
localhost. Pokud se připojujeme ke vzdálenému serveru, může to být IP adresa, název síťového počítače nebo internetová adresa.

- 12) Do pole *Číslo portu* zadáme číslo portu databázového serveru. Výchozí číslo portu pro databáze MySQL a MariaDB je 3306.

Tip

pokud si nejste jisti správností zadávaných informací v krocích 10) až 12), požádáme o radu správce databáze.

- 13) Pokud Base zná konektor JDBC, automaticky vyplní pole *Třída MySQL JDBC ovladače* názvem konektoru. Jedná se buď o *com.mysql.jdbc.Driver* (pro MySQL konektor) nebo *org.mariadb.jdbc.Driver* (pro MariaDB konektor).
- 14) Kliknutím na tlačítko **Testovací třída** zkontrolujeme, zda je soubor JAR konektoru přístupný pro Base. V případě úspěchu zobrazí Base potvrzovací dialog (obrázek 105). Kliknutím na tlačítko **OK** tento dialog zavřeme.



Obrázek 105: Úspěšně načtený dialog konektoru JDBC

- 15) Klikneme na tlačítko **Další >**.
- 16) Base zobrazí krok 4 (*Nastavit autentizaci uživatele*) Průvodce databází (podobně jako na obrázku 95).
- 17) Do pole *Uživatelské jméno* zadáme platné uživatelské jméno. Náš příklad uživatele se jmenuje *lotest*.
- 18) Klepneme na možnost **Vyžadováno heslo**.
- 19) Pokud je potřeba, otestujeme připojení provedením kroků 18) až 23) postupu přímého připojení (strana 84).
- 20) Klikneme na tlačítko **Další >**.
- 21) Base zobrazí krok 5 (*Uložit a pokračovat*) Průvodce databází (se stejnými možnostmi, jaké jsou uvedeny na obrázku 90).
- 22) Klepneme na možnost **Ne, neregistrovat databázi**, protože se jedná o testovací databázi.
- 23) Klepneme na možnost **Otevřít databázi pro úpravy**, protože tato možnost je obvykle vhodná pro novou databázi. Pokud si v tuto chvíli nepřejeme databázi upravovat, můžeme tuto možnost zrušit.
- 24) Pokud nechceme použít Průvodce tabulkou, ponecháme volbu **Vytvořit tabulky pomocí průvodce tabulkou** ve výchozím nezaškrtnutém stavu.
- 25) Klepneme na tlačítko **Dokončit**.
- 26) Base zobrazí dialogové okno *Uložit jako*, ve kterém je třeba zadat název a umístění souboru ODB.
- 27) Klepneme na tlačítko **Uložit**.

- 28) Průvodce databází se zavře, vytvoří se soubor ODB obsahující informace o připojení k databázi a Base aktualizuje své hlavní okno tak, aby se pod názvem databáze zobrazovaly informace o tabulkách databáze (podobně jako na obrázku 98).

Tabulky lze vytvářet, upravovat a odstraňovat.

V tomto případě se na stavovém řádku zobrazí následující informace:

- Název konektoru použitého pro připojení k databázi MySQL nebo MariaDB (v tomto případě *MySQL (JDBC)*).
- Název otevřené databáze MySQL nebo MariaDB (v tomto případě *libretest*).
- Uživatelské jméno, které se používá pro připojení k databázi MySQL nebo MariaDB (v tomto případě *lotest*).
- Název serveru, na kterém je spuštěna databáze MySQL nebo MariaDB (v tomto případě *localhost*).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k základní databázi MySQL nebo MariaDB, zobrazí se dialogové okno Vyžadována autentizace (obrázek 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

Tip

Pokud jsou v oblasti *Tabulky* hlavního okna Base zobrazeny tabulky `INFORMATION_SCHEMA`, měla by je většina uživatelů obvykle ignorovat. Nejjednodušší je sbalit uzel `information_schema` ve stromové struktuře tabulky. Případně z hlavní nabídky zvolíme **Nástroje > Filtr tabulky**, odznačíme uzel `information_schema` v dialogovém okně Filtr tabulek a klikneme na tlačítko **OK**. Poté v hlavní nabídce vybereme **Zobrazit > Obnovit tabulky**, poté uzel `information_schema` a jeho podřízené uzly již nebudou viditelné v oblasti *Tabulky*.

Možná omezení a řešení

Tabulky bez primárních klíčů

V MySQL a MariaDB je možné zadávat a měnit data v tabulkách bez pole primárního klíče. Uživatelské rozhraní Base umí takové tabulky zobrazit, ale nenabízí žádné možnosti zadávání nebo úprav.

Pokud chceme použít tabulky bez primárního klíče, můžeme místo toho použít volbu **Nástroje > SQL** nebo zahrnout makra do formulářů, abychom tabulkám dodali data.

Připojení přes Internet

Připojení k databázím MySQL nebo MariaDB může být k místnímu serveru („localhost“), k síťovému serveru nebo k internetovému serveru. MySQL a MariaDB poskytují systémovou proměnnou `wait_timeout`, která definuje dobu, po kterou server čeká, než se neaktivní připojení stane aktivním, a teprve poté jej uzavře. V některých případech můžeme mít pocit, že aktuální čekací doba je pro naše úlohy nevhodná, například když byl nastaven nízký `wait_timeout`, aby nedošlo k překročení maximálního počtu připojení.

V závislosti na nastavení serveru můžeme změnit hodnotu `wait_timeout`. Zkusíme následující příkaz s použitím volby **Nástroje > SQL**:

```
SET SESSION wait_timeout = 600;
```

Pokud se to podaří, `wait_timeout` se vynuluje na 600 sekund nebo 10 minut. Jakmile je proveden jakýkoli dotaz, doba čekání do dalšího přerušení se vynuluje na 10 minut.

Pokud nelze `wait_timeout` nastavit na vyhovující hodnotu, měli bychom zvážit napsání makra, které pošle jednoduchý dotaz (např. `SELECT NOW()`) v opakujících se intervalech, aby se udrželo otevřené spojení.

U připojení JDBC můžeme zkusit nastavit konfigurační parametr `autoReconnect` na hodnotu `true`. V takovém případě může dojít k prvnímu přerušení, ale při dalším pokusu o navázání kontaktu se spojení obnoví.

Při přímém připojení je nutné soubor databáze po přerušení zavřít a znovu otevřít, aby se obnovil kontakt se serverem. Údaje o připojení, jako jsou přihlašovací údaje, však není třeba zadávat znovu.

Přístup k uloženým rutinám

MySQL i MariaDB podporují uložené procedury (procedury a funkce). Uložená procedura je sada příkazů SQL, které jsou uloženy na databázovém serveru a vyvolány pomocí příkazu `CALL`. Uložená procedura může mít parametry, takže jí můžeme předávat hodnoty a získávat výsledky.

V Base lze uložené procedury volat pomocí volby **Nástroje > SQL**. Neexistuje však žádný přímý způsob, jak zobrazit výstup z uložené procedury v Base.

Jednoduchým řešením pro přístup k výstupu uložené procedury je zapsat výstup procedury do dočasné tabulky v databázi, která bude viditelná pouze pro aktuálního uživatele databáze. Tento uživatel se pak může dotazovat na dočasnou tabulku a přistupovat k výsledkům uložené procedury. Automatizovanější přístup k řešení tohoto problému je popsán v kapitole 9, Makra.

Databáze PostgreSQL

Přehled

PostgreSQL (často označovaný jednoduše jako „Postgres“) je open-source RDBMS poskytovaný společností The PostgreSQL Global Development Group. Další informace najdeme na webových stránkách PostgreSQL na adrese <https://www.postgresql.org/>. Odkazy na aktuální i minulé verze dokumentace PostgreSQL najdeme na adrese <https://www.postgresql.org/docs/>.

Base se může k databázi PostgreSQL připojit jedním ze tří způsobů:

- Přímé připojení pomocí integrovaného konektoru PostgreSQL.
- Připojení pomocí ODBC.
- Připojení pomocí JDBC. Tato možnost může být vhodná, pokud je třeba navázat zabezpečené připojení ke vzdálené databázi PostgreSQL pomocí protokolu SSH (Secure Shell).

Uživatelské interakce nutné pro připojení k databázi PostgreSQL mají určitou podobnost s interakcemi pro připojení k databázi MySQL nebo MariaDB. Existují však i významné rozdíly. Cílem následujících podkapitol je zdůraznit a vysvětlit tyto rozdíly, aniž bychom opakovali podrobnosti, které jsou shodné s těmi, které již byly popsány pro MySQL/MariaDB.

Úvod do schémat PostgreSQL

Schémata jsou důležitou součástí databáze PostgreSQL, ale noví uživatelé by si měli uvědomit, že termín *schema* se v jiných relačních systémech správy databází nepoužívá vždy ve stejném významu. Podrobnější informace najdeme v kapitole Schémata v dokumentaci PostgreSQL (<https://www.postgresql.org/docs/current/ddl-schemas.html>).

Následující poznámky upozorňují na některé vlastnosti schémat PostgreSQL, které se mohou projevit v uživatelském rozhraní Base při připojení k databázi PostgreSQL.

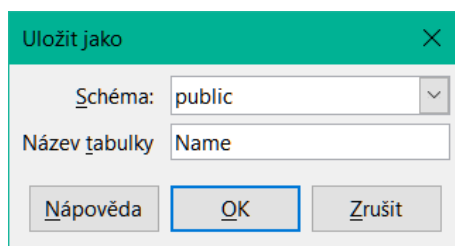
- Schéma je pojmenovaná kolekce databázových objektů, včetně tabulek a pohledů (ale nejen jich).

- Databáze obsahuje jedno nebo více schémat. Dvě schémata v rámci databáze mohou obsahovat různé objekty, které mají podobný název.
- Pro každou novou databázi vygeneruje PostgreSQL schéma nazvané *public*. Pokud vytvoříme databázový objekt bez zadání názvu schématu, bude umístěn do schématu *public*.
- Pro každou novou databázi generuje PostgreSQL další schémata určená pro systémové účely. Některá z nich mohou být v Base viditelná v závislosti na typu připojení (například schémata *information_schema* a *pg_catalog*). Uživatelé by neměli ukládat své databázové objekty do schémat, která jsou určena pro systémové použití.
- Uživatelé mohou vytvářet vlastní schémata pomocí volby **Nástroje > SQL**, ale měli by se vyhnout názvům začínajícím předponou „pg_“. Chceme-li přistupovat k objektu v určitém schématu, specifikujeme objekt pomocí syntaxe: `schema_name.object_name`.

Uživatелеm vytvořená schémata se nezobrazují v oblasti *Tabulky* hlavního okna Base, dokud neobsahují alespoň jeden uložený objekt databáze.

Pokud zjistíme, že oblast *Tabulky* hlavního okna Base je přeplněná příliš velkým množstvím schémat a/nebo tabulek, zvolíme v hlavní nabídce **Nástroje > Filtr tabulky** a pomocí dialogového okna *Filtr Tabulky* vybereme položky, které se mají zobrazit nebo skrýt. Všimneme si, že pro zobrazení účinku změn provedených v tomto dialogu bude možná nutné vybrat **Zobrazit > Obnovit tabulky**.

Při prvním uložení nové tabulky vytvořené v zobrazení návrhu zobrazí Base dialogové okno *Uložit jako*, které obsahuje nabídku *Schema*, z níž je třeba vybrat požadované schéma (obrázek 106). Podobná nabídka je k dispozici v dialogovém okně, které se zobrazí, když vybereme dotaz a poté v hlavní nabídce zvolíme **Upravit > Vytvořit jako pohled**.



Obrázek 106: Dialogové okno *Uložit jako* s rozevírací nabídkou *Schéma*

Počáteční přípravy před připojením

Podrobné pokyny ke stažení a instalaci softwaru databázového serveru pro naši konfiguraci počítače a podrobnosti o přístupu ke všem nástrojům, které potřebujeme k zadávání příkazů SQL před připojením k databázi Base, nalezneme v dokumentaci PostgreSQL. Po instalaci postupujeme podle pokynů pro nastavení uvedených v dokumentaci. Může jít například o to, aby heslo výchozího superuživatelského účtu databázového serveru (uživatelské jméno *postgres*) bylo nastaveno na řetězec, který je vhodný pro zachování bezpečnosti dat.

Před navázáním připojení musí na serveru existovat databáze, která nás zajímá, a musíme mít uživatelské jméno a heslo. Pokud databáze a uživatelské jméno ještě neexistují, měl by uživatel *postgres* provést následující kroky k jejich vytvoření. Příkazy lze zadávat prostřednictvím terminálového rozhraní (například *psql*) nebo grafického rozhraní (například *pgAdmin*).

- 1) Chceme-li vytvořit (prázdnou) databázi s názvem *libretest*, zadáme následující příkaz:
`CREATE DATABASE libretest;`
- 2) Chceme-li udělit všechna práva v nové databázi novému uživateli s názvem *lotest* a heslem *libre*, zadáme následující příkazy:

```
CREATE USER lotest WITH PASSWORD 'libre';
GRANT ALL PRIVILEGES ON DATABASE libretest TO lotest;
```

- 3) Chceme-li v případě potřeby vytvořit schéma s názvem *user_area* pro uživatele *lotest* pro ukládání objektů databáze, zadáme následující příkazy:

```
CREATE SCHEMA user_area;
ALTER SCHEMA user_area OWNER TO lotest;
```

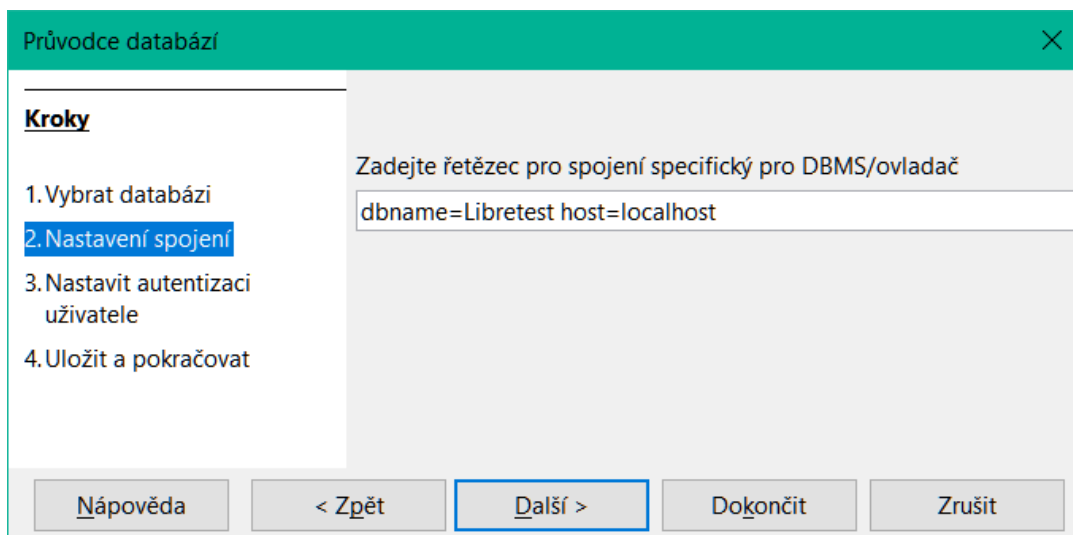
Po úspěšném dokončení těchto kroků je nová databáze k dispozici a Base se k ní může připojit, jak je popsáno v následujících podkapitolách.

Přímé připojení

Při prvním přímém připojení k určité databázi PostgreSQL provedeme podobné kroky jako při přímém připojení k databázi MySQL/MariaDB (viz podkapitola Chyba: zdroj odkazu nenalezen Přímé připojení na straně 82), ale s následujícími důležitými rozdíly:

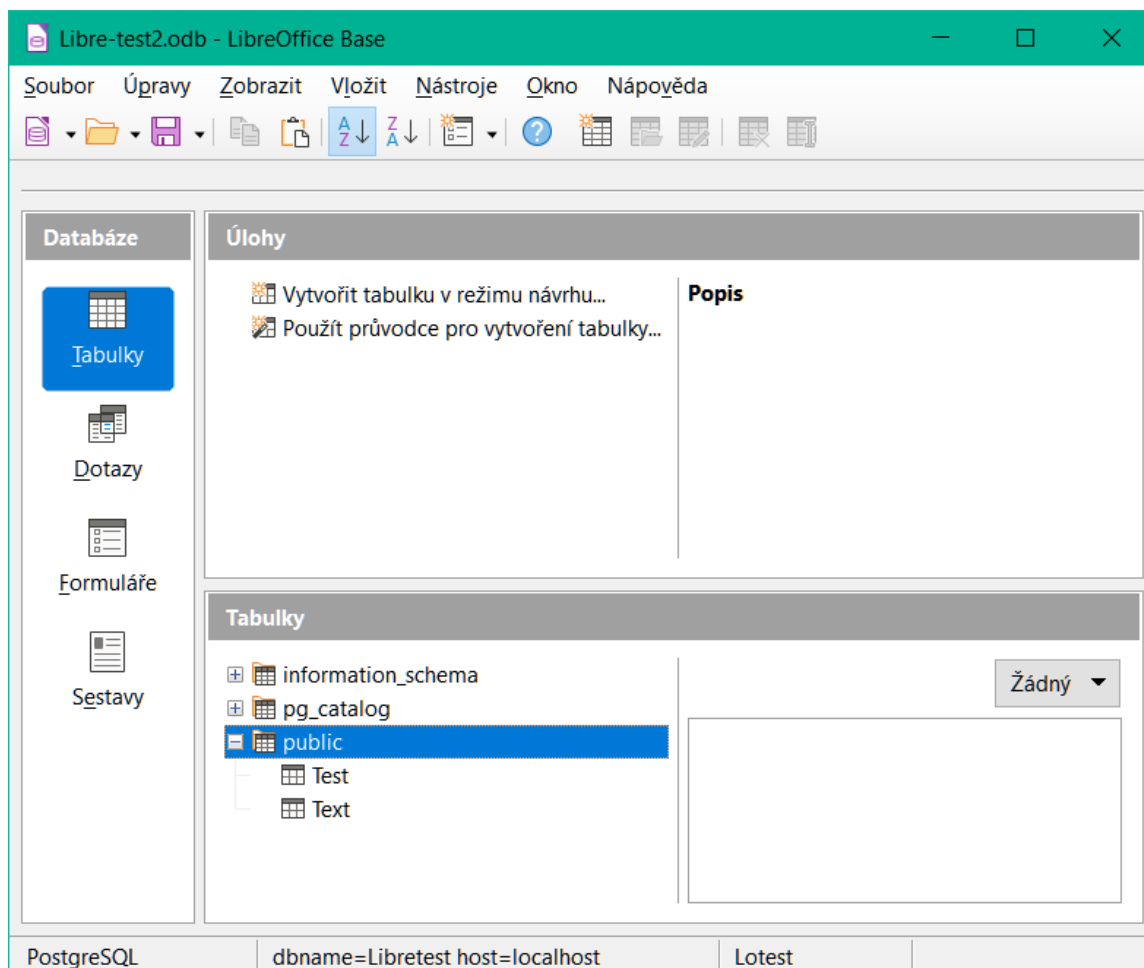
- Po kliknutí na možnost **Připojit se k databázi** v kroku 1 (*Vybrat databázi*) Průvodce databází vybereme v rozevírací nabídce možnost **PostgreSQL** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) a kroku 3 (*Nastavit MySQL server*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro PostgreSQL relevantní. Namísto toho Base zobrazí krok 2 (*Nastavení spojení*) Průvodce databází (obrázek 107).

Zadáme název databáze a název hostitele, jak je znázorněno na obrázku 107 (*dbname=libretest host=localhost*). Pro zadání parametrů připojení se používá formát prostého řetězce klíčových slov a hodnot, který je blíže popsán v části Řetězce připojení v dokumentaci PostgreSQL (<https://www.postgresql.org/docs/current/libpq-connect.html>). Za určitých okolností je nutné uvést plně kvalifikovaný název hostitele včetně názvu domény.



Obrázek 107: Krok 2 Průvodce databází: Nastavení spojení

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o databázi PostgreSQL (obrázek 108).



Obrázek 108: Hlavní okno Base při přímém připojení k PostgreSQL

Oblast *Tabulky* zobrazuje tři schémata, pojmenovaná *information_schema*, *pg_catalog* a *public*. Schéma *public* je rozbaleno, aby byly zobrazeny dvě tabulky *test* a *text*.

Stavový řádek ve spodní části obrázku 108 zobrazuje:

- Název použitého konektoru (*PostgreSQL*).
- Podrobnosti o připojovacím řetězci zadaném v kroku 2 Průvodce databází (v tomto případě *dbname=libretest host=localhost*).
- Uživatelské jméno, které bylo použito pro připojení k databázi PostgreSQL (v tomto případě *lotest*).

Když později znovu tento soubor ODB otevřeme a poté použijeme Base pro přístup k základní databázi PostgreSQL, zobrazí se dialogové okno Požadované ověření (podobné tomu na obrázku 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

Připojení pomocí ODBC

Počáteční příprava

Chceme-li se pomocí Base připojit k databázovému serveru PostgreSQL prostřednictvím rozhraní ODBC API, měl by být v počítači nainstalován správně nakonfigurovaný ovladač ODBC. Pokud ovladač není v našem počítači k dispozici, budeme k instalaci nového softwaru potřebovat administrátorská práva.

Pro připojení k databázi PostgreSQL prostřednictvím ODBC se doporučuje použít oficiální ovladač PostgreSQL ODBC (psqlODBC). Správnou verzi ovladače ODBC pro náš počítač si

můžeme stáhnout z webové stránky psqLODBC na adrese <https://odbc.postgresql.org/>, kde také najdeme příslušnou dokumentaci. Pro systém Windows je ovladač k dispozici ve standardním, snadno použitelném instalačním balíčku. Pro prostředí Unix musíte zdrojový kód zkompilovat sami podle pokynů uvedených na webových stránkách. Pro použití psqLODBC v systému macOS nejsou uvedeny žádné pokyny.

Při vytváření názvu zdroje dat ODBC (DSN) pro naši databázi postupujeme podle pokynů uvedených v dokumentaci k ovladači. Pro účely příkladu v následujícím pododdíle předpokládejme, že zdroj dat byl nakonfigurován takto:

- Je vybrán ovladač PostgreSQL Unicode.
- Název zdroje dat je nastaven na *libretest-PostgreSQL-ODBC*.
- Název databáze (*libretest*) je obsažen v definici zdroje dat.
- Ve zdroji dat nejsou zadány přihlašovací údaje k databázovému serveru.

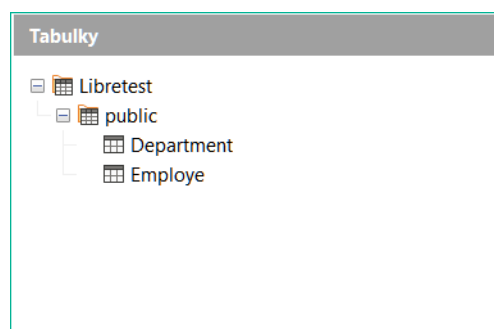
Použití Průvodce databází

Při prvním připojení ODBC k určité databázi PostgreSQL provedeme podobné kroky jako při připojení ODBC k databázi MySQL/MariaDB (viz podkapitola Použití Průvodce databází na straně 87), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **ODBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro PostgreSQL relevantní.
- V kroku 2 (*Nastavit ODBC spojení*) zadáme název dříve definovaného datového zdroje ODBC, který bude použit pro přístup k databázi PostgreSQL (*libretest-PostgreSQL-ODBC* v našem příkladu). Kliknutím na tlačítko **Procházet** zobrazíme dialog Zdroj dat (obrázek 101), v němž můžeme vybrat hledaný zdroj dat ze seznamu zdrojů dostupných a výběr potvrdit kliknutím na tlačítko **OK**.

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o databázi PostgreSQL.

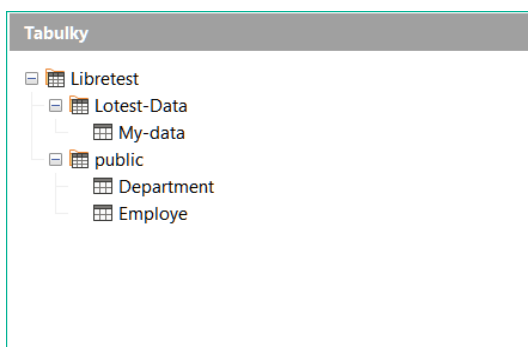
Ve výchozím nastavení připojení ODBC k PostgreSQL se v oblasti *Tabulky* zobrazují pouze objekty ve veřejném schématu, i když jsou přítomna jiná uživatelsky definovaná schémata.



Obrázek 109: Počáteční rozložení oblasti *Tabulky* pro připojení ODBC k PostgreSQL

Chceme-li zobrazit další definovaná schémata v oblasti *Tabulky*, může pomoci reset proměnné `search_path` na úroveň databáze. Jedním z možných způsobů, jak toho dosáhnout, je nakonfigurovat ovladač ODBC tak, aby při připojení k databázi vydal příslušný příkaz. Například v systému Windows lze otevřít ODBC Data Source Administrator, v jehož dialogu vybereme ovladač, klikneme na **Konfigurace**, klikneme na **Zdroj dat**, dále na **Stránka 2** a pak vložíme příkaz `SET SEARCH_PATH TO libretest;` v textovém poli *Nastavení spojení*. V systému

Unix se ujistíme, zda soubor „odbc.ini“ obsahuje položku `ConnSettings = SET SEARCH_PATH TO libretest;`. Účinek změny proměnné `search_path` vidíme na obrázku 110, na němž je tatáž základní databáze, nyní však vidíme i dodatečné schéma `lotest_data`.



Obrázek 110: Aktualizované rozložení oblastí Tabulky pro připojení ODBC k PostgreSQL

Při připojení ODBC k PostgreSQL se ve stavovém řádku hlavního okna Base zobrazují následující informace:

- Název použitého konektoru (ODBC).
- Název zdroje dat ODBC (v tomto případě `libretest-PostgreSQL-ODBC`).
- Uživatelské jméno, které bylo použito pro připojení k databázi PostgreSQL (v tomto případě `lotest`).

Když později znovu tento soubor ODB otevřeme a poté použijeme Base pro přístup k základní databázi PostgreSQL, zobrazí se dialogové okno Požadované ověření (podobné tomu na obrázku 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

Připojení pomocí JDBC

Počáteční příprava

Chceme-li se pomocí Base připojit k databázovému serveru PostgreSQL prostřednictvím rozhraní JDBC API, měl by být v počítači nainstalován příslušný ovladač JDBC. Pokud v našem počítači ovladač není, je třeba jej stáhnout a konkrétní soubor JAR (Java ARchive) umístit do přístupné složky.

Pro připojení k databázi PostgreSQL prostřednictvím JDBC doporučujeme použít oficiální ovladač PostgreSQL JDBC. Správnou verzi ovladače JDBC pro náš počítač si můžeme stáhnout z webové stránky PostgreSQL JDBC Driver na adrese <https://jdbc.postgresql.org/>. Pokyny k instalaci nalezneme v dokumentaci k ovladači PostgreSQL JDBC, která je dostupná na adrese <https://jdbc.postgresql.org/documentation/documentation.html>.

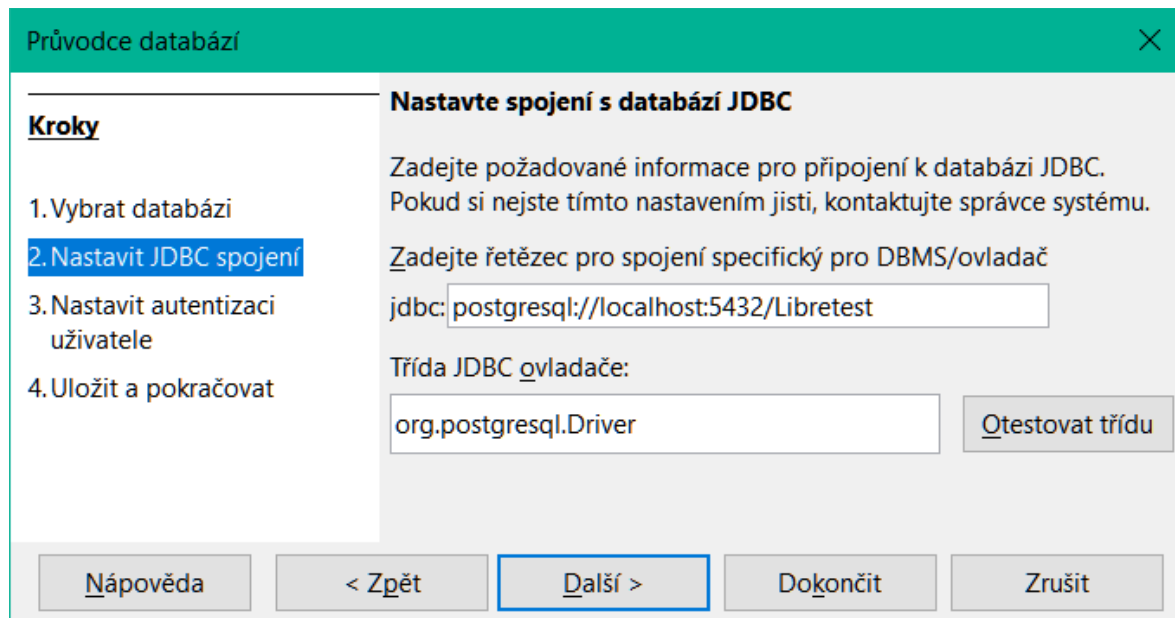
Před stažením ovladače JDBC se ujistíme, že jsme vybrali pro konfiguraci našeho počítače správnou verzi. Ovladač JDBC nelze použít, pokud náš systém nepodporuje jazyk Java.

Provedeme podobné kroky jako u připojení JDBC k databázi MySQL/MariaDB, abychom Base informovali, kde najde soubor JAR pro ovladač JDBC (viz podkapitola Počáteční příprava na straně 89).

Použití Průvodce databází

Při prvním připojení JDBC k určité databázi PostgreSQL provedeme podobné kroky jako při připojení JDBC k databázi MySQL/MariaDB (viz podkapitola Použití Průvodce databází na straně 91), ale s následujícími důležitými rozdíly:

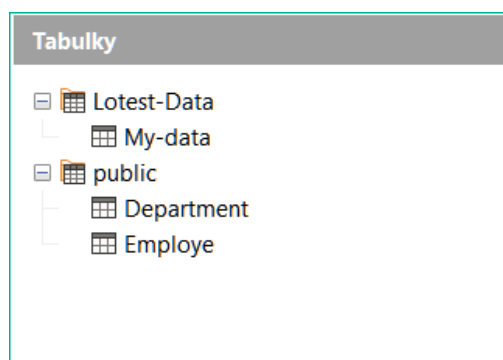
- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **JDBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro PostgreSQL relevantní.
- V případě připojení JDBC zobrazí Base krok 2 (*Nastavit JDBC spojení*) Průvodce databází (obrázek 111).



Obrázek 111: Krok 2 Průvodce databází: Nastavení připojení JDBC

- Zadáme připojovací řetězec pomocí jedné z forem uvedených v dokumentaci k ovladači PostgreSQL JDBC. Na obrázku 111 je použit tvar `jdbc:postgresql://host:port/database`.
- Do pole *Třída ovladače JDBC* zadáme název ovladače jako `org.postgresql.Driver`.
- Kliknutím na tlačítko **Test třídy** zkontrolujeme, zda je soubor JAR ovladače pro Base přístupný. V případě úspěchu zobrazí Base potvrzovací dialog (obrázek 105). Kliknutím na tlačítko **OK** tento dialog zavřeme.

Ve výchozím nastavení připojení JDBC k PostgreSQL se v oblasti *Tabulky* zobrazují pouze objekty ve schématu *public*, i když jsou přítomna jiná uživatelsky definovaná schémata (například *lotest_data* na obrázku 112).



Obrázek 112: Počáteční rozložení oblasti *Tabulky* pro připojení JDBC k PostgreSQL

Při připojení JDBC k PostgreSQL se ve stavovém řádku hlavního okna Base zobrazují následující informace:

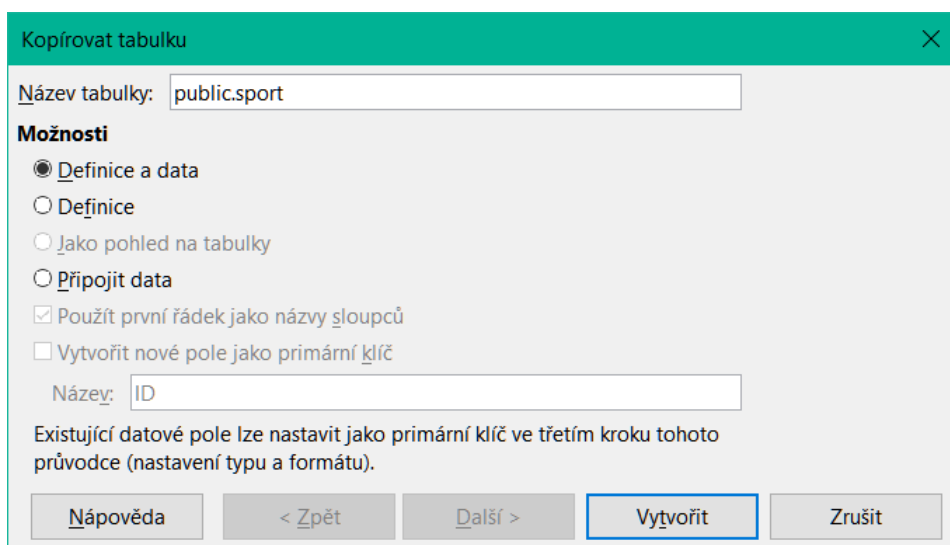
- Název použitého konektoru (JDBC).
- Připojovací řetězec zadaný v kroku 2 Průvodce databází (v tomto případě `postgresql://localhost:5432/libretest`).
- Uživatelské jméno, které bylo použito pro připojení k databázi PostgreSQL (v tomto případě `lotest`).

Když později znovu tento soubor ODB otevřeme a poté použijeme Base pro přístup k základní databázi PostgreSQL, zobrazí se dialogové okno Požadované ověření (podobné tomu na obrázku 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

Možná omezení a řešení

Kopírování tabulky z jiné databáze

Base můžeme použít ke zkopírování tabulky z nějaké databáze do databáze PostgreSQL. Máme-li obě databáze otevřené v programu Base, klikneme pravým tlačítkem myši na název tabulky ve zdrojové databázi a z kontextové nabídky vybereme **Kopírovat**. Při kopírování do PostgreSQL databáze zkontrolujeme, zda jsou vybrány **Tabulky** v oblasti **Databáze** hlavního okna Base a klikneme na tlačítko **Vložit** na standardní nástrojové liště. Base zobrazí dialogové okno Kopírovat tabulku (obrázek 113), ve kterém je třeba do pole *Název tabulky* požadovanou předponu (prefix) doplnit (například „public.“). Pro ovladač ODBC (pro který jsme nastavili `search_path` na `libretest`) může být typická předpona „libretest.public“.



Obrázek 113: Dialogové okno Kopírovat tabulku

Automatické zvyšování hodnot v PostgreSQL

Při práci s vestavěnou HSQLDB databází můžeme vytvořit v tabulce vytvořit primární klíč přidáním sloupce s názvem *id* (nebo podobným) typu **Integer**, který má parametr *AutoValue* nastaven na **Yes**. Pokud stejný přístup použijeme pro tabulku v databázi PostgreSQL, nebude to mít požadovaný efekt a hodnoty sloupců se nebudou automaticky zvyšovat – musíme je zadat.

Dlouhodobě zavedenou a běžnou metodou přidání sloupce primárního klíče s automatickým nárůstem do tabulky PostgreSQL je použití některého z nestandardních datových typů **BIGSERIAL**, **SERIAL** nebo **SMALLSERIAL**. Například následující kód SQL vytvoří jednoduchou tabulku s datovým typem **SERIAL**:

```
CREATE TABLE public.employee (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(100) NOT NULL  
);
```

Při vytváření tabulky v zobrazení návrhu s přímým nebo ODBC připojením nejsou ve sloupci Typ pole k dispozici datové typy SERIAL. Jsou však k dispozici s připojením JDBC.

V novějších verzích PostgreSQL je k dispozici alternativní přístup, který používá standardní datový typ integer. V tomto případě by bylo možné výše uvedený kód SQL nahradit takto:

```
CREATE TABLE public.employee (  
    id INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,  
    name VARCHAR(100) NOT NULL  
);
```

V případech, kdy nelze vybrat požadovaný typ primárního klíče v Zobrazení návrhu, se doporučuje vybrat v hlavní nabídce **Nástroje > SQL**, zadat kód SQL potřebný k vytvoření tabulky obsahující pouze požadovaný primární klíč. Poté můžeme přepnout do zobrazení návrhu tabulky a přidat zbývající sloupce. V prvním případě se nová tabulka nemusí v oblasti *Tabulky* hlavního okna Base zobrazit, ale to lze napravit výběrem **Zobrazení > Obnovit tabulky** v hlavní nabídce.



Poznámka

Při použití přímého připojení nebo připojení JDBC může při spuštění dotazu pomocí grafického uživatelského rozhraní nastat problém (chyba 130376). Tento problém by mohl vést ke ztrátě dat, ale měl by se vyskytnout pouze u dotazů, které zahrnují automaticky se zvyšující pole a alias tabulky (v klauzuli FROM příkazu SQL). Problému se lze vyhnout buď odstraněním aliasu tabulky, nebo přímým spuštěním takového dotazu pomocí volby **Nástroje > SQL**. Užijeme-li připojení ODBC, můžeme se potkat s problémem jen tehdy, je-li volba **Respektovat u výsledku typ nastavený databázovým ovladačem** zaškrtnuta na kartě *Speciální nastavení* dialogu **Úpravy > Databáze > Pokročilá nastavení**.

Zobrazení PostgreSQL

Obvyklým mechanismem pro uložení pohledu v programu Base je výběr dotazu a následné zvolení možnosti **Úpravy > Vytvořit jako pohled** v hlavní nabídce. Tři způsoby připojení k databázi PostgreSQL však vedou při této interakci k následujícím rozdílům v chování:

- **Přímé připojení.** Přímý ovladač neumožňuje vytváření pohledů.
- **Připojení ODBC.** Ovladač ODBC neumožňuje vytváření pohledů prostřednictvím této interakce, pokud ještě žádný pohled neexistuje. To však můžeme opravit vytvořením fiktivního pohledu z nabídky **Nástroje > SQL** (například můžeme použít následující kód z dokumentace PostgreSQL: `CREATE VIEW vista AS SELECT text 'Hello World' AS hello;`).
- **Připojení JDBC.** Ovladač JDBC vždy umožňuje vytváření pohledů prostřednictvím této interakce.

Pro všechny typy připojení lze pohledy vytvářet pomocí nabídky **Nástroje > SQL**.

Databáze Firebird4

Přehled

Firebird je open-source RDBMS poskytovaný nadací Firebird Foundation. Více informací najdeme na webových stránkách společnosti Firebird na adrese <https://firebirdsql.org/>. Odkazy

na aktuální a minulé verze dokumentace Firebirdu najdeme na adrese <https://firebirdsql.org/en/documentation/>.

Base se může připojit k externí databázi prostřednictvím serveru Firebird jedním ze dvou způsobů:

- Připojení pomocí ODBC.
- Připojení pomocí JDBC.

Uživatelské interakce nutné k navázání takového připojení mají určitou podobnost s interakcemi při připojování k databázi MySQL nebo MariaDB. Existují však i významné rozdíly. Cílem následujících podkapitol je zdůraznit a vysvětlit tyto rozdíly, aniž bychom opakovali podrobnosti, které jsou shodné s těmi, které již byly popsány pro MySQL/MariaDB.

Pokud není vyžadován přístup k databázi serveru, je možné v aplikaci Base přímo otevřít externí soubor databáze Firebird. V tomto případě bude otevřená databáze spravována pomocí engine Firebird zabudovaného v systému Base. Pro využití této funkce není nutné povolit experimentální funkce v rámci Base.

✓ Poznámka

Vestavěný engine je založen na Firebirdu 3.0.7. V příručce Firebird 3 Quick Start Guide se uvádí, že Firebird 3 neotevře databáze vytvořené pomocí dřívějších verzí. Podobně Firebird 4 vytváří databáze s odlišnou strukturou na disku a takové databáze nelze otevřít přímo pomocí vestavěného engine Firebird 3.

Počáteční přípravy před připojením k serveru Firebird

Podrobné pokyny ke stažení a instalaci softwaru databázového serveru pro naši konfiguraci počítače a podrobnosti o přístupu k nástrojům, které potřebujeme k zadávání příkazů SQL před připojením k databázi Base, najdeme v dokumentaci k Firebirdu. Po instalaci postupujeme podle pokynů pro nastavení uvedených v dokumentaci. Může jít například o to, aby heslo výchozího účtu superuživatele databázového serveru (uživatelské jméno *SYSDBA*) bylo nastaveno na řetězec, který je vhodný pro zachování bezpečnosti dat. Výchozí heslo pro uživatele *SYSDBA* je nastaveno na *masterkey*.

✓ Poznámka

V době přípravy této kapitoly nebyla k dispozici žádná verze Firebirdu 4 ke stažení pro macOS. Kromě toho nebyla k dispozici žádná příručka pro rychlé spuštění Firebirdu 4.

Před navázáním připojení musí existovat databáze, která nás zajímá, a musíme mít uživatelský účet, který nám dává oprávnění pracovat s databází a objekty v ní. Pokud databáze a uživatelské jméno ještě neexistují, měl by je vytvořit superuživatel *SYSDBA*. Příkazy lze zadávat pomocí nástroje Firebird Interactive SQL Utility (*isql*) – v prvních verzích Firebirdu byl pro správu uživatelů k dispozici nástroj *gsec*, který je však již zastaralý a tyto činnosti se provádějí pomocí příkazů SQL.

✓ Poznámka

V některých instalacích Firebirdu by mohl být *isql* přejmenován na *isql-fb*, aby se předešlo možné kolizi názvů.

- 1) Chceme-li vytvořit (prázdnou) databázi v souboru *libretest.fdb*, umístěnou v adresáři *c:\my_databases*, zadáme následující příkaz (zaměníme řetězec "masterkey" heslem uživatele *SYSDBA*):

```
CREATE DATABASE 'c:\my_databases\libretest.fdb'  
USER SYSDBA  
PASSWORD 'masterkey';
```

- 2) Chceme-li vytvořit nového uživatele s názvem *lotest* a heslem *libre*, zadáme následující příkaz:

```
. CREATE USER lotest PASSWORD 'libre';
```

Informace o uživateli oprávněných k přístupu na konkrétní server Firebird jsou uloženy v bezpečnostní databázi s názvem *security3.fdb* nebo *security4.fdb*, v závislosti na verzi Firebirdu. Tyto soubory se obvykle nacházejí v instalačním adresáři.
- 3) Chceme-li novému uživateli povolit vytváření a změnu tabulek nebo pohledů v databázi *libretest*, zadáme následující příkazy:

```
GRANT ALL PRIVILEGES TABLE TO USER lotest;  
GRANT ALL PRIVILEGES VIEW TO USER lotest;  
COMMIT;
```
- 4) Nakonec je užitečné vytvořit alias pro novou databázi. Upravíme soubor *databases.conf*, který se obvykle nachází v instalačním adresáři, tak, aby obsahoval řádek

```
libretest=c:\my_databases\libretest.fdb
```

To umožňuje odkazovat na databázi pomocí jejího aliasu *libretest*, čímž se skryje fyzické umístění databáze a sníží se možnost chyb při psaní.

Tip

V krocích 2) a 4) výše, budeme možná muset upravit oprávnění k souborům *security3.fdb*, *security4.fdb* nebo *databases.conf* před tím než provedeme nějakou akci.

Po úspěšném dokončení těchto kroků je nová databáze k dispozici a Base se k ní může připojit, jak je popsáno v následujících podkapitolách.

Poznámka

Pokud jiný uživatel než *lotest* vytváří v databázi *libretest* tabulky, potom možná uživatel *lotest* nebude mít k těmto tabulkám plný přístup. Chceme-li povolit uživateli *lotest* plný přístup k tabulce s názvem *Tabulka1*, zadáme následující příkaz:

```
. GRANT ALL ON TABLE "Table1" TO USER lotest;
```

Připojení pomocí ODBC

Počáteční příprava

Chceme-li se pomocí Base připojit k serveru Firebird prostřednictvím rozhraní ODBC API, měl by být v počítači nainstalován správně nakonfigurovaný ovladač ODBC. Pokud ovladač není v našem počítači k dispozici, budeme k instalaci nového softwaru potřebovat administrátorská práva.

Pro připojení k databázi serveru Firebird prostřednictvím ODBC se doporučuje použít oficiální ovladač Firebird ODBC. Správnou verzi ovladače Firebird ODBC pro náš počítač si můžeme stáhnout z webových stránek Firebird na adrese <https://dev.mysql.com/downloads/connector/odbc/>. Pokyny k instalaci závislé na platformě nalezneme v příručce Firebird ODBC/JDBC Driver 2.0, která je přístupná na stránce Firebird Drivers Documentation na adrese <https://firebirdsql.org/en/drivers-documentation/>.

✓ Poznámka

V době přípravy této části neobsahovala příručka k ovladači Firebird ODBC/JDBC Driver 2.0 žádné pokyny k instalaci pro systém macOS.

Při vytváření názvu zdroje dat ODBC (DSN) pro naši databázi postupujeme podle pokynů uvedených v dokumentaci k ovladači. Pro účely příkladu v následujícím pododdíle předpokládáme, že zdroj dat byl nakonfigurován takto:

- Název zdroje dat je nastaven na *libretest-firebird*.
- Alias pro databázi, která nás zajímá, je obsažen v definici zdroje dat (například *localhost:libretest*).
- Přihlašovací údaje pro databázový server nejsou ve zdroji dat definovány.
- Vlastnost *Sensitive Identifier* je povolena.

Použití Průvodce databází

Při prvním připojení ODBC k určité databázi Firebird provedeme podobné kroky jako při připojení ODBC k databázi MySQL/MariaDB (viz podkapitola Použití Průvodce databází na straně 87), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **ODBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro Firebird relevantní.
- V kroku 2 (*Nastavit ODBC spojení*) zadáme název dříve definovaného datového zdroje ODBC, který má být použit pro přístup k databázi Firebird (v našem příkladu *libretest-firebird*). Kliknutím na tlačítko **Procházet** zobrazíme dialog Zdroj dat (obrázek 101), v němž můžeme vybrat hledaný zdroj dat ze seznamu zdrojů dostupných a výběr potvrdit kliknutím na tlačítko **OK**.

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o databázi Firebird.

Při připojení ODBC k Firebirdu se ve stavovém řádku hlavního okna Base zobrazují následující informace:

- Název použitého konektoru (*ODBC*).
- Název zdroje dat ODBC (v tomto případě *libretest-firebird*).
- Uživatelské jméno, které bylo použito pro připojení k databázi Firebird (v tomto případě *lotest*).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k základní databázi Firebird, zobrazí se dialogové okno Požadované ověření (podobné tomu na obrázku 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

i Tip

Při vytváření tabulek v databázi Firebird prostřednictvím ODBC dbáme na to, aby byl každému poli od začátku přiřazen správný typ. Po uložení tabulky není možné měnit typ pole, je však možné odstranit nežádoucí pole a přidat nová.

Připojení pomocí JDBC

Počáteční příprava

Chceme-li se pomocí Base připojit k serveru Firebird prostřednictvím rozhraní JDBC API, měl by být v počítači umístěn příslušný ovladač JDBC. Pokud v našem počítači ovladač není, je třeba jej stáhnout a konkrétní soubor JAR (Java ARchive) umístit do přístupné složky.

Pro připojení k databázi Firebird prostřednictvím JDBC se doporučuje použít oficiální ovladač Firebird JDBC, který se nazývá Jaybird. Správnou verzi programu Jaybird pro náš počítač si můžeme stáhnout z webových stránek Firebird na adrese <https://firebirdsql.org/en/jdbc-driver/>. Pokyny k použití nalezneme na stránce Dokumentace ovladačů Firebird na adrese <https://firebirdsql.org/en/drivers-documentation/>.

Při stahování aplikace Jaybird se ujistíme, že jsme vybrali správnou verzi pro konfiguraci svého počítače. Ovladač JDBC nelze použít, pokud náš systém nepodporuje jazyk Java.

Provedeme podobné kroky jako v případě připojení JDBC k databázi MySQL/MariaDB, abychom informovali Base, kde najde soubor JAR pro Jaybird (viz podkapitola Počáteční příprava na straně 89). Použijeme soubor JAR s názvem ve tvaru *jaybird-full-4.0.5.java11.jar* (číslice v tomto názvu se liší v závislosti na verzích Jaybird a Java vybraných na stránce pro stažení).

Použití Průvodce databází

Při prvním připojení JDBC k určité databázi Firebird provedeme podobné kroky jako při připojení JDBC k databázi MySQL/MariaDB (viz podkapitola Použití Průvodce databází na straně 91), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **JDBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro Firebird relevantní.
- V případě připojení JDBC zobrazí Base krok 2 (*Nastavit JDBC spojení*) Průvodce databází (obrázek 111).
- Zadáme údaje o databázi jako *jdbc:firebirdsql:oo://localhost/libretest?charSet=utf-8*. Pokud zde není zadána znaková sada, zobrazí se chyba.
- Zadejte název třídy ovladače jako *org.firebirdsql.jdbc.FBDriver*.
- Kliknutím na tlačítko **Test třídy** zkontrolujeme, zda je soubor JAR ovladače pro Base přístupný. V případě úspěchu zobrazí Base potvrzovací dialog (obrázek 105). Kliknutím na tlačítko **OK** tento dialog zavřeme.

Při připojení JDBC k Firebirdu se ve stavovém řádku hlavního okna Base zobrazují následující informace:

- Název použitého konektoru (*JDBC*).
- Název zdroje dat JDBC zadaný v kroku 2 Průvodce databází (*firebirdsql:oo://localhost/libretest?charSet=utf-8*).
- Uživatelské jméno, které bylo použito pro připojení k databázi Firebird (v tomto případě *lotest*).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k základní databázi Firebird, zobrazí se dialogové okno Požadované ověření (podobné tomu na obrázku 99). Připojení není třeba znovu konfigurovat, ale je třeba znovu zadat správné přihlašovací údaje.

Tip

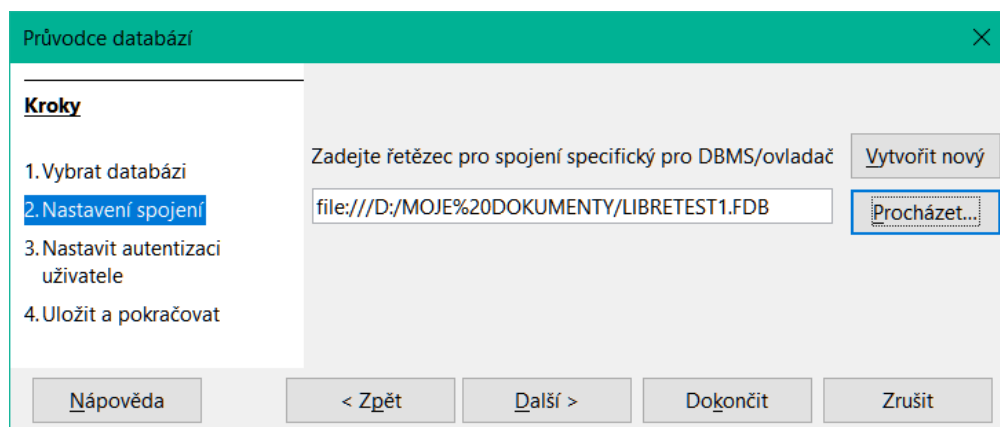
Při vytváření tabulek v databázi Firebird prostřednictvím JDBC dbáme na to, aby byl každému poli od začátku přiřazen správný typ. Po uložení tabulky není možné měnit typ pole, je však možné odstranit nežádoucí pole a přidat nová.

Přímé připojení k souboru Firebird

Chceme-li se připojit k (neserverovému) databázovému souboru Firebird 3 umístěnému v cestě, ke které máme přístup pro čtení a zápis, provedeme podobné kroky jako při nastavení přímého připojení k databázi MySQL/MariaDB (viz podkapitola Přímé připojení na straně 82), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **Externí Firebird** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) a kroku 3 (*Nastavit MySQL server*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro **Externí Firebird** relevantní. Namísto toho Base zobrazí krok 2 (*Nastavení spojení*) Průvodce databází, jak je znázorněno na obrázku 114.

Zadáme název a umístění souboru Firebird 3 do příslušného textového pole nebo podle potřeby použijeme funkce **Vytvořit nový** nebo **Procházet**.



Obrázek 114: Krok 2 Průvodce databází: Nastavení připojení (Externí Firebird)

- V kroku 3 (*Nastavte autentizaci uživatele*) Průvodce databází klikneme na tlačítko **Další >**, aniž bychom zadávali údaje o ověření uživatele – ty nejsou pro připojení k databázi Firebird 3, která není serverem, vyžadovány.

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o databázi Firebird.

Stavový řádek v dolní části obrazovky zobrazuje:

- Použitý typ připojení (*Externí Firebird*).
- Název a umístění souboru zadané v kroku 2 Průvodce databází (například `file:///C:/my_databases/firebird_v3_file.fdb`).

✓ Poznámka

Připojení k databázi na serveru je možné navázat prostřednictvím serveru Firebird pomocí možnosti **Externí Firebird** v kroku 1 (*Vybrat databázi*) Průvodce databází. Možnosti, které jsou prostřednictvím takového připojení k dispozici, však vyžadují další zkoumání a mohou být popsány v některé z příštích verzí této knihy.

Externí databáze HSQLDB

Přehled

Databáze HyperSQL (HSQLDB) je open-source RDBMS napsaná v jazyce Java a poskytovaná společností The HSQL Development Group. Více informací nalezneme na webových stránkách HSQLDB na adrese <https://hsqldb.org/>. Odkazy na nejnovější dokumentaci k HSQLDB nalezneme na adrese <https://hsqldb.org/web/hsqldbDocsFrame.html>.

Každá databáze HSQLDB obsahuje 2 až 6 souborů v adresáři, všechny se stejným názvem, ale každý s jinou příponou. Možná rozšíření jsou:

- *záloha*
- *dat*
- *loby*
- *Přihlásit se*
- *vlastnosti*
- *script*

Mechanismus integrovaný v databázi Base využívá HSQLDB verze 1.8.0, která byla poprvé vydána v červenci 2005. Existuje několik důvodů, proč se někteří uživatelé mohou rozhodnout použít externí databázi HSQLDB, včetně:

- Využití dalších funkcí poskytovaných v pozdějších verzích HSQLDB. V době aktualizace této kapitoly byla k dispozici verze HSQLDB 2.6.1 (vydaná v říjnu 2021).
- Připojení k databázi, kterou nelze integrovat, například k existující víceuživatelské databázi.
- Aby se předešlo potenciální nevýhodě vestavěných databází HSQLDB, která vyplývá z ukládání dat v archivním formátu ODB. Vzhledem k tomu, že data jsou zapsána až po zabalení archivu, existuje větší riziko ztráty dat než při práci s externí databází.

Stávající vestavěnou databázi HSQLDB je možné převést a vyjmout z balíčku ODB tak, aby ji bylo možné použít jako externí databázi. Pokyny pro tento postup však přesahují rámec této kapitoly.

Chceme-li použít externí databázi HSQLDB, připojíme se pomocí ovladače JDBC, který je součástí balíčku HSQLDB JAR. Vzhledem k tomu, že databáze Base i HSQLDB využívají jazyk Java, je logické vytvořit spojení mezi databází Base a externí databází HSQLDB pomocí JDBC. Ačkoli by mělo být možné připojit databázi Base k HSQLDB prostřednictvím ODBC (viz kapitola 17 Uživatelské příručky HyperSQL), podrobnosti o tomto přístupu nejsou v této kapitole uvedeny.

Uživatelské interakce nutné k vytvoření připojení JDBC k externí databázi HSQLDB mají určitou podobnost s odpovídajícími interakcemi při připojení k databázi MySQL nebo MariaDB. Existují však i významné rozdíly. Cílem následujících podkapitol je zdůraznit a vysvětlit tyto rozdíly, aniž bychom opakovali podrobnosti, které jsou shodné s těmi, které již byly popsány pro MySQL/MariaDB.

Poznámka

Poslední verze HSQLDB mají obvykle mnohem více funkcí než starší verze. Při pokusu o převod mezi vloženou databází vytvořenou pomocí verze 1.8.0 a externí databází vytvořenou pomocí novější verze tak pravděpodobně nastanou problémy s kompatibilitou. Zejména externí databázi, která byla upravena pomocí HSQLDB 2.0+, nelze snadno převést na vestavěnou databázi pod Base.

Pokud potřebujeme převést databázi z jedné verze HSQLDB na jinou, užitečné informace najdeme v uživatelské příručce HyperSQL v kapitolách „Upgrade databází“ a „Problémy zpětné kompatibility“.

Upozornění

Integrace externího balíčku JAR HSQLDB *hsqldb.jar* do cesty tříd Java pro LibreOffice může znamenat, že k vestavěným databázím již nelze správně přistupovat pomocí Base. Base se může pokusit použít externí HSQLDB pro vestavěné databáze. Ačkoli existují kroky, které mohou pokročilejší uživatelé podniknout, aby se takovému problému vyhnuli (například zapsání cesty ke třídě příslušného databázového souboru pomocí makra nebo vložení cesty přímo do souboru *content.xml* souboru ODB), tyto kroky přesahují rámec této kapitoly.

Úvodní přípravy před připojením k databázi HSQLDB

Pokud jsme spokojeni s funkčností HSQLDB verze 1.8.0, ale chceme ji pouze použít pro externí databázi, pak není třeba stahovat další soubory z Internetu. Můžeme jednoduše použít soubor *hsqldb.jar* dodaný s naší verzí LibreOffice, který se obvykle nachází v podsložce `/program/classes/` v instalační složce programu. Tento přístup zabraňuje problémům vyplývajícím z nekompatibility různých verzí. Jinak si můžeme aktuální i starší verze HSQLDB stáhnout z webových stránek HyperSQL.

Podrobné pokyny ke stažení a instalaci softwaru HSQLDB pro naši konfiguraci počítače a podrobnosti o přístupu ke všem nástrojům, které potřebujeme k zadávání příkazů SQL před připojením k databázi Base, najdeme v Uživatelské příručce HyperSQL. Po instalaci postupujeme podle pokynů pro nastavení uvedených v dokumentaci. Může jít například o to, aby heslo výchozího uživatele admin (SA, což je zkratka pro System Administrator) bylo nastaveno na řetězec, který je vhodný pro zachování bezpečnosti dat. Výchozí heslo pro uživatele SA je prázdné. Můžeme také raději změnit uživatelské jméno správce na jiné než SA. Pro zjednodušení se ve zbytku této části předpokládá, že k databázi HSQLDB se přistupuje pomocí výchozího uživatelského jména SA s prázdným heslem, což se však pro skutečnou databázi nedoporučuje.

HSQLDB se vyznačuje tím, že při pokusu o připojení k databázi, která v současné době neexistuje, je vytvořena nová, prázdná databáze. To znamená, že k vytvoření databáze před připojením k ní není třeba provádět žádné zvláštní přípravné akce.

Identifikujeme umístění souboru *hsqldb.jar* staženého z webu Hyper SQL (například `C:\Users\username\Downloads\hsqldb-2.6.1\hsqldb-2.6.1\hsqldb\lib\hsqldb.jar`) a přesuneme jej do jiného umístění, pokud je to pro nás vhodnější. Tento JAR obsahuje engine HyperSQL i ovladač HyperSQL JDBC. Ať už se jedná o připojení pro přístup jednoho nebo více uživatelů, provedeme podobné kroky jako při připojení JDBC k databázi MySQL/MariaDB a informujeme Base, kde najde tento soubor JAR (viz podkapitola Počáteční příprava na straně 89).

✓ Poznámka

V současné době balíček ZIP ke stažení HSQLDB obsahuje jednu verzi souboru *hsqldb.jar*, která je kompatibilní s Javu 11 nebo novější, a druhou, která je zkompileována s JDK 8. Další verzi zkompileovanou s JDK 6 lze samostatně stáhnout z webových stránek HyperSQL. Ujistíme se, že používáme správnou verzi pro konfiguraci našeho počítače.

Připojení pro jednoho uživatele

Pro přístup jedno uživatelský přístup použijeme připojení v procesu (in-process), jak je popsáno v Uživatelské příručce HyperSQL. Chceme-li nakonfigurovat tento typ připojení, zadáme cestu k databázi v adrese URL připojení.

Při prvním připojení k externí databázi HSQLDB v procesu provedeme podobné kroky jako při připojení JDBC k databázi MySQL/MariaDB (viz podkapitola Použití Průvodce databází na straně 91), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **JDBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro HSQLDB relevantní.
- V případě připojení JDBC zobrazí Base krok 2 (*Nastavit JDBC spojení*) Průvodce databází (obrázek 111).
- Zadáme údaje o cestě k databázi v následujícím formátu:
`jdbc:hsqldb:file:C:\my_databases\hsqldb\libretest;default_schema=true;shutdown=true.`
 - Nahradíme řetězec „C:\my_databases\hsqldb\libretest“ cestou k naší databázi.
 - Při připojování Base k externímu databázovému souboru HSQLDB by měl připojovací řetězec obsahovat vlastnost "*default_schema=true*", která zajistí, že Base bude pracovat pouze s PUBLIC schématem HSQLDB. Zahrnuje také vlastnost "*shutdown=true*", která zajistí, že se databáze vypne po uzavření posledního připojení. Další informace o těchto vlastnostech nalezneme v Uživatelské příručce HyperSQL.
- Zadáme název třídy ovladače jako `org.hsqldb.jdbc.JDBCDriver`.
- Kliknutím na tlačítko **Test třídy** zkontrolujeme, zda je soubor JAR ovladače pro Base přístupný. V případě úspěchu zobrazí Base potvrzovací dialog (obrázek 105). Kliknutím na tlačítko **OK** tento dialog zavřeme.
- V kroku 3 (*Nastavit autentizaci uživatele*) Průvodce databází zadáme v poli *Uživatelské jméno SA* a ujistíme se, že pole **Vyžadováno heslo** není zaškrtnuto.

Při probíhající připojení k externí databázi HSQLDB se ve stavovém řádku hlavního okna Base zobrazují následující informace:

- Název použitého konektoru (*JDBC*).
- Cesta k databázovému souboru zadaná v kroku 2 Průvodce databází
(v tomto případě `hsqldb:file:C:\my_databases\hsqldb\libretest;default_schema=true;shutdown=true`).
- Uživatelské jméno, které bylo použito pro připojení k databázi HSQLDB (v tomto případě SA).

Připojení k serveru / více uživatelů

Pro víceuživatelský přístup použijeme jeden z režimů serveru popsaných v Uživatelské příručce HyperSQL. V tomto příkladu je použit režim serveru HyperSQL HSQL.

Pokud potřebujeme server spustit, podrobnosti o potřebných krocích nalezneme v Uživatelské příručce HyperSQL. Obvykle to vyžaduje příkaz shellu nebo terminálu ve tvaru:

```
C:\> java -cp c:/my_databases/hsqldb/lib/hsqldb.jar
org.hsqldb.server.Server --database.0 file:c:/my_databases/mydb --
dbname.0 libretest
```

Následující body mohou pomoci objasnit tento příkaz:

- 1) Příkaz *java* musí ležet ve vyhledávací cestě příkazů našeho shellu.
- 2) V tomto příkladu je balíček HSQLDB JAR umístěn ve složce *c:/my_databases/hsqldb/lib*.
- 3) V tomto příkladu jsou databázové soubory pojmenovány *mydb.** a jsou umístěny ve složce *c:/my_databases*.
- 4) Název *libretest* lze použít k odkazu na databázi, když je Base připojena k serveru.

Při prvním připojení pomocí režimu HyperSQL HSQL Server provedeme podobné kroky jako při připojení JDBC k databázi MySQL/MariaDB (viz podkapitola Použití Průvodce databází na straně 91), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **JDBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro HSQLDB relevantní.
- V případě připojení JDBC zobrazí Base krok 2 (*Nastavit JDBC spojení*) Průvodce databází (obrázek 111).
- Zadáme údaje o databázi a serveru v následujícím formátu:
jdbc:hsqldb:hsql://localhost/libretest;default_schema=true;shutdown=true.
 - Všimneme si, že zde není zmínka o databázovém souboru, protože ten byl zadán při spuštění serveru. Potřebný je pouze veřejný název definovaný pro *dbname.0*.
 - V režimech serveru je *localhost* název popisující adresu místního počítače. V síťovém prostředí může být nahrazen IP adresou nebo názvem počítače.
 - Vysvětlení vlastností viz podkapitola Připojení pro jednoho uživatele nad , jedná se o vlastnosti *default_schema=true* a *shutdown=true*.
- Zadáme název třídy ovladače jako *org.hsqldb.jdbc.JDBCDriver*.
- Kliknutím na tlačítko **Test třídy** zkontrolujeme, zda je soubor JAR ovladače pro Base přístupný. V případě úspěchu zobrazí Base potvrzovací dialog (obrázek 105). Kliknutím na tlačítko **OK** tento dialog zavřeme.
- V kroku 3 (*Nastavit autentizaci uživatele*) Průvodce databází zadáme v poli *Uživatelské jméno SA* a ujistíme se, že pole **Vyžadováno heslo** není zaškrtnuto.

Při připojení k externí databázi v režimu HyperSQL HSQL Server se na stavovém řádku hlavního okna Base zobrazují následující informace:

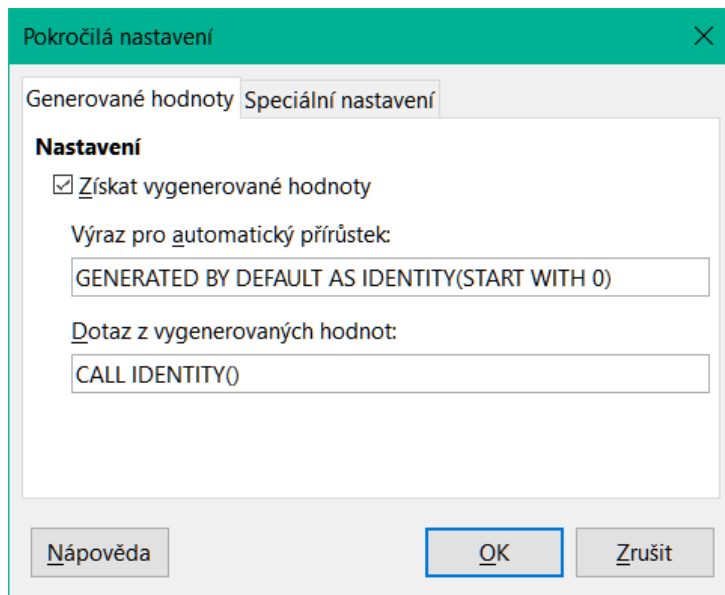
- Název použitého konektoru (*JDBC*).
- Název databázového souboru zadaný v kroku 2 Průvodce databází (v tomto případě *hsqldb:hsql://localhost/libretest;default_schema=true;shutdown=true*).
- Uživatelské jméno, které bylo použito pro připojení k databázi HSQLDB (v tomto případě *SA*).

Možná omezení a řešení

Automatické zvyšování hodnot

Před vytvořením automaticky se zvyšujícího pole primárního klíče vybereme v hlavní nabídce možnost **Upravit > Databáze > Pokročilá nastavení** a klikneme na kartu *Generované hodnoty*.

Zaškrtneme pole **Získat vygenerované hodnoty** checkbox, zadáme **GENERATED BY DEFAULT AS IDENTITY(START WITH 0)** v poli *Výraz pro automatický přírůstek*, zadáme **CALL IDENTITY()** v poli *Dotaz z vygenerovaných hodnot* a klikneme na tlačítko **OK** (obrázek 115).



Obrázek 115: Dialogové okno Pokročilá nastavení, karta Generované hodnoty pro externí připojení HSQLDB

Tyto úpravy nám mohou umožnit úspěšně definovat pole primárního klíče s automatickým nárůstem. Pokud Base při zadávání dat do tabulky zobrazí chybové hlášení o chybě kontrolního omezení NOT NULL, lze to často napravit vytvořením pole primárního klíče bez automatické hodnoty a následným provedením příkazu **ALTER TABLE "table_name" ALTER COLUMN "ID" INT GENERATED BY DEFAULT AS IDENTITY(START WITH 0)** pomocí volby **Nástroje > SQL** option. *table__name* a *ID* jsou názvy tabulky, resp. pole primárního klíče.

Žádný přístup k tabulkám

Někdy může být databáze HSQLDB otevřena v aplikaci Base, ale není k ní přístup. V tomto případě doporučujeme zvolit v hlavní nabídce **Nástroje > SQL** a zadat příkaz **SHUTDOWN SCRIPT**. Doufejme, že následné uzavření a opětovné otevření databáze problém odstraní, i když to nemusí být úspěšné, pokud již byla nahlášena „chyba v souboru skriptu“.

Databáze SQLite

Přehled

SQLite je vestavěný databázový stroj SQL s otevřeným zdrojovým kódem. Na rozdíl od databázových systémů popsaných v předchozích částech této kapitoly nemá SQLite samostatný server. Databáze SQLite se skládají z jediného souboru, který může být uložen kdekoli v souborovém systému a k databázi Base se lze připojit pomocí vhodného ovladače. Další informace najdeme na webových stránkách SQLite na adrese <https://www.sqlite.org/>. Rozsáhlou dokumentaci k SQLite najdeme na adrese <https://www.sqlite.org/docs.html>.

Tip

SQLite podporuje omezenou podmnožinu příkazu ALTER TABLE jazyka SQL. Při návrhu tabulek budme opatrní, protože po vytvoření tabulky je obtížné změnit datový typ sloupce.

Base se může připojit k externímu souboru databáze SQLite pomocí ovladače ODBC. Uživatelské interakce nutné k navázání takového připojení mají určitou podobnost s interakcemi při připojování k databázi MySQL nebo MariaDB. Existují však i významné rozdíly. Cílem následujících podkapitol je zdůraznit a vysvětlit tyto rozdíly, aniž bychom opakovali podrobnosti, které jsou shodné s těmi, které již byly popsány pro MySQL/MariaDB.

Databáze Base v současné době neumožňuje přímé připojení k databázovému souboru SQLite.

Vývojáři SQLite nedodávají ovladač JDBC ke stažení. Ačkoli je na internetu k dispozici několik ovladačů JDBC pro SQLite, v současné době nevíme o žádném, který by podporoval spolehlivé spojení mezi Base a SQLite.

Počáteční přípravy před připojením k souboru SQLite

Před vytvořením připojení musí existovat soubor databáze SQLite, který nás zajímá. Pokud náš databázový soubor ještě neexistuje, můžeme prázdný soubor vytvořit několika způsoby:

- Použijeme jednu z mnoha možností, které nabízí náš operační systém. Například v systému Windows můžeme pomocí *Průzkumníka souborů* přejít do složky, kam chceme soubor umístit, kliknout pravým tlačítkem myši, vybrat **Nový**, vybrat **Textový dokument** a zadat název souboru. Příkazový řádek systému Windows nabízí také příkazy, které při nápaditém použití dokážou vytvářet prázdné soubory (například `type nul > libretest.db`). V Linuxu jsou k dispozici ekvivalentní prostředky (například `> libretest.db`).
- Použijeme běžný textový editor, například *Notepad* v systému Windows, *Vim* v systému Linux nebo *TextEdit* v systému MacOS. Ujistíme se, že velikost souboru je po vytvoření nula bajtů.
- Použijeme příkazový řádek *sqlite3* pro SQLite, který může být nainstalován při instalaci ovladače nebo jej lze stáhnout (spolu s dalšími nástroji) z <https://www.sqlite.org/download.html>.

Podrobnou dokumentaci k *sqlite3* najdeme na adrese <https://www.sqlite.org/cli.html>. Následující příkazy by měly stačit k vytvoření nového databázového souboru s názvem *libretest.db*:

- 1) Na příkazovém řádku shellu, terminálu nebo operačního systému zadáme:
`sqlite3 libretest.db`
- 2) Na příkazovém řádku *sqlite>* zadáme:
`.databases`
sqlite3 poskytuje zpětnou vazbu o přesném umístění vytvořeného souboru.
- 3) Na příkazovém řádku *sqlite>* zadáme:
`.exit`
sqlite3 se uzavře.

SQLite nepředepisuje příponu databázového souboru – mnoho uživatelů použije ".sqlite", jiní zvolí ".db", ale je to osobní rozhodnutí.

SQLite neposkytuje kontrolu uživatelského jména a hesla pro databázový soubor.

Připojení pomocí ODBC

Počáteční příprava

Chceme-li se pomocí aplikace Base připojit k databázovému souboru SQLite prostřednictvím rozhraní ODBC API, měl by být v počítači nainstalován správně nakonfigurovaný ovladač ODBC. Pokud ovladač není v našem počítači k dispozici, budeme k instalaci nového softwaru potřebovat administrátorská práva.

Vývojáři SQLite nedodávají ovladač ODBC ke stažení. Na internetu je však k dispozici několik takových ovladačů, z nichž některé jsou open-source a jiné mohou být proprietární. Jeden takový ovladač je k dispozici na webových stránkách Christiana Wernera na adrese <http://ch-werner.de/sqliteodbc/>, kde je uvedeno, že ovladač byl testován v systémech Windows a Linux.

Pokyny k instalaci v závislosti na platformě nalezneme v dokumentaci SQLiteODBC na adrese <http://ch-werner.de/sqliteodbc/html/index.html>.

Poznámka

V době přípravy této části vývojář ovladače neposkytuje sestavení pro systém macOS. Jiní vývojáři však přispěli několika sestaveními systému macOS pro vybrané starší verze ovladače.

Při vytváření názvu zdroje dat ODBC (DSN) pro naši databázi postupujeme podle pokynů uvedených v dokumentaci k ovladači. Pro účely příkladu v následujícím pododdíle předpokládáme, že zdroj dat byl nakonfigurován takto:

- Název zdroje dat je nastaven na *libretest-SQLite*.
- Zdroj dat obsahuje název databáze *C:\my_databases\libretest.db*.
- Přihlašovací údaje pro databázový server nejsou ve zdroji dat definovány.

Použití Průvodce databází

Při prvním připojení ODBC k určitému souboru databáze SQLite provedeme podobné kroky jako při připojení ODBC k databázi MySQL/MariaDB (podkapitola Použití Průvodce databází na straně 87), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **ODBC** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) Průvodce databází při vytváření připojení k databázi MySQL/MariaDB nejsou pro SQLite relevantní.
- V kroku 2 (*Nastavit ODBC spojení*) zadáme název dříve definovaného zdroje dat ODBC, který má být použit pro přístup k databázi SQLite (v našem příkladu *libretest-SQLite*). Kliknutím na tlačítko **Procházet** zobrazíme dialog Zdroj dat (obrázek 101), v němž můžeme vybrat hledaný zdroj dat ze seznamu zdrojů dostupných a výběr potvrdit kliknutím na tlačítko **OK**.
- V kroku 3 (*Nastavit autentizaci uživatele*) Průvodce databází nezadáme žádné informace.

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o databázovém souboru SQLite.

Při připojení ODBC k databázovému souboru SQLite se ve stavovém řádku hlavního okna Base zobrazují následující informace:

- Název použitého konektoru (*ODBC*).
- Název zdroje dat ODBC (v tomto případě *libretest-SQLite*).

Databáze dBASE

Připojení k databázi dBASE

Databáze dBASE využívají typ souboru DBF, což je standardní formát pro ukládání strukturovaných dat. Base se může připojit k databázi obsahující jeden nebo více souborů DBF umístěných v jedné složce. Každý soubor DBF by měl mít příponu ".dbf" (malými písmeny v systémech, kde jsou názvy souborů citlivé na velikost písmen) a uchovává data pro jednu databázovou tabulku, včetně deskriptorů polí i datových záznamů. Tabulky definované v souborech DBF jsou koncepčně podobné listům v Calc.

Mezi poli v různých tabulkách neexistují žádné vnitřní vazby / vztahy – to například znamená, že pokud zaměstnanec opustí vaši společnost a vymažeme jeho údaje z tabulky *employee*, může stále dostávat mzdu za příští měsíc, pokud ji zapomeneme vymazat také z tabulky *salary*! Pokud

jsou takové odkazy pro připojenou databázi dBASE nutné, můžeme je vytvořit pomocí prostředků Base (například formulářů, dotazů a maker).

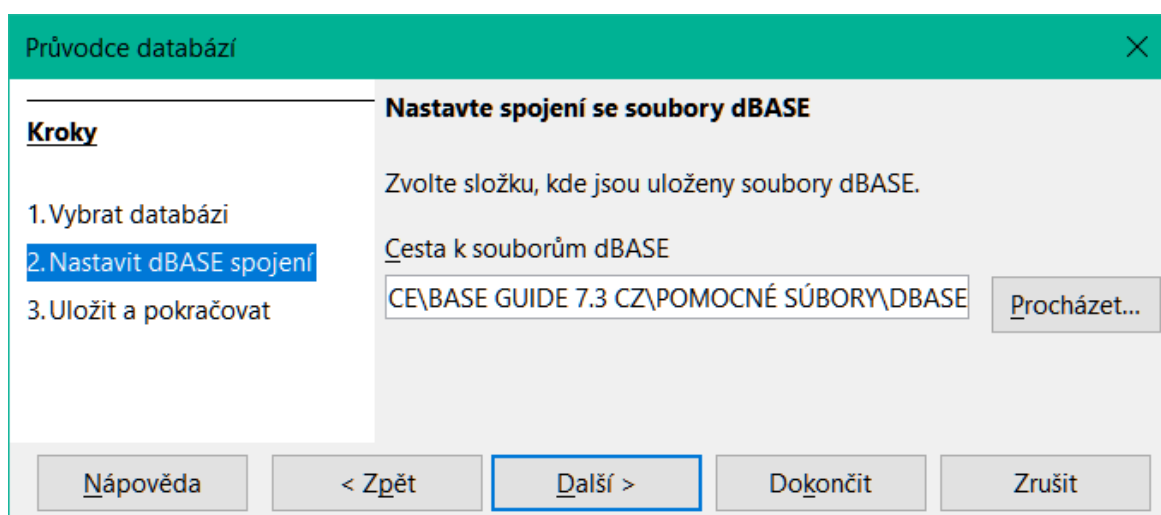
Tip

LibreOffice Calc umí uložit tabulku jako soubor DBF (ačkoli vzorce a formátování jsou ztraceny) a umí načíst soubor DBF do tabulky. Tato možnost může být užitečná v případech, kdy potřebujeme rychle zobrazit data v tabulce dBASE, která ještě není připojena v databázi Base.

Při prvním připojení k určité databázi dBASE provedeme podobné kroky jako při přímém připojení k databázi MySQL/MariaDB (viz podkapitola Přímé připojení na straně 82), ale s následujícími důležitými rozdíly:

- Po kliknutí na volbu **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází, zvolíme volbu **dBASE** v rozbalovacím seznamu (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) a kroku 3 (*Nastavit MySQL server*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro dBASE relevantní. Namísto toho Base zobrazí krok 2 (*Nastavit dBASE spojení*) Průvodce databází (obrázek 116).

Zadáme umístění složky obsahující soubory dBASE. Kliknutím na tlačítko **Procházet se** zobrazí dialogové okno Výběr složky, kde můžeme přejít na požadovanou složku a poté kliknout na tlačítko **Vybrat složku**.



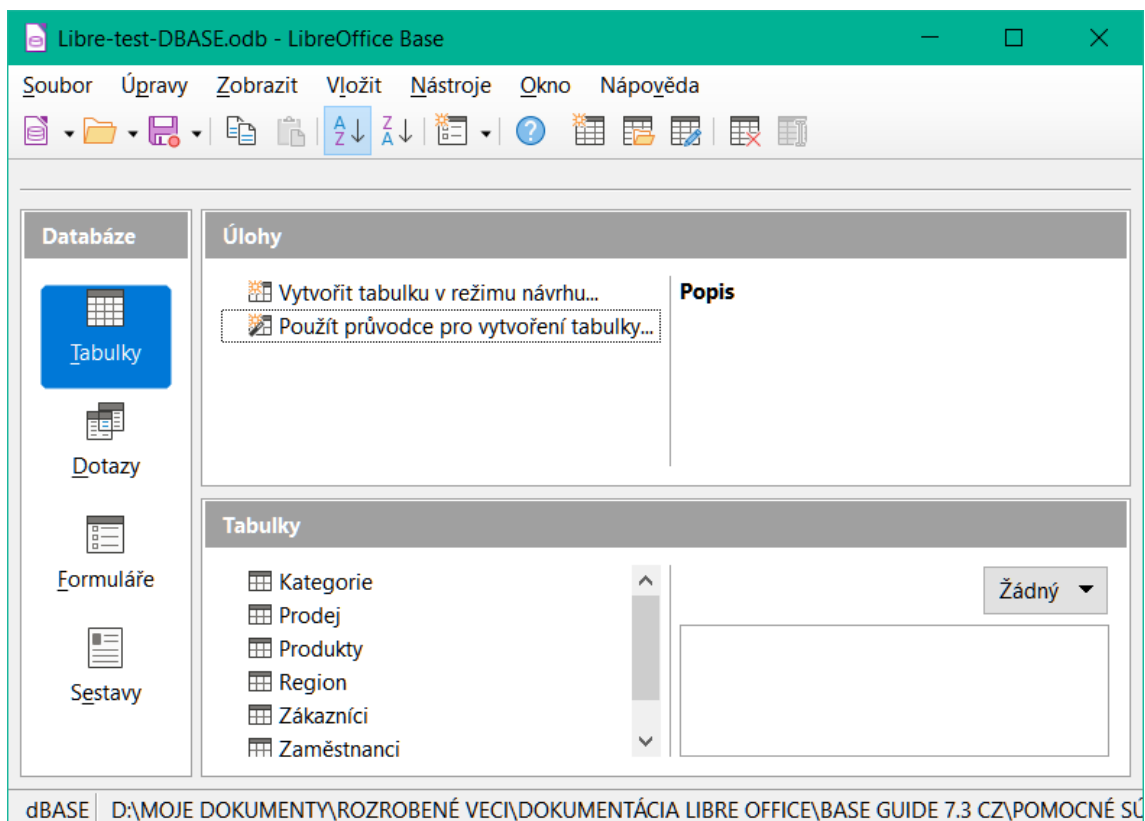
Obrázek 116: Krok 2 Průvodce databází: Nastavení připojení dBASE

- Možnosti uvedené v kroku 4 (*Nastavení autentizace uživatele*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro dBASE relevantní.

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o databázi dBASE (obrázek 117), včetně všech souborů DBF ve složce. V uvedeném případě našla databáze Base v dané složce šest souborů DBF, které jsou pojmenovány:

- category.dbf
- customer.dbf
- employee.dbf
- products.dbf
- region.dbf

- sales.dbf



Obrázek 117: Tabulky v souboru dBASE

Při připojení k databázi dBASE se ve stavovém řádku hlavního okna Base zobrazují následující informace:

- Název použitého konektoru (dBASE).
- Složka obsahující databázové soubory (v tomto případě C:\dBASE_files).

Nové tabulky lze vytvářet prostřednictvím uživatelského rozhraní Base, které se zobrazí jako další soubory DBF v zadané složce.

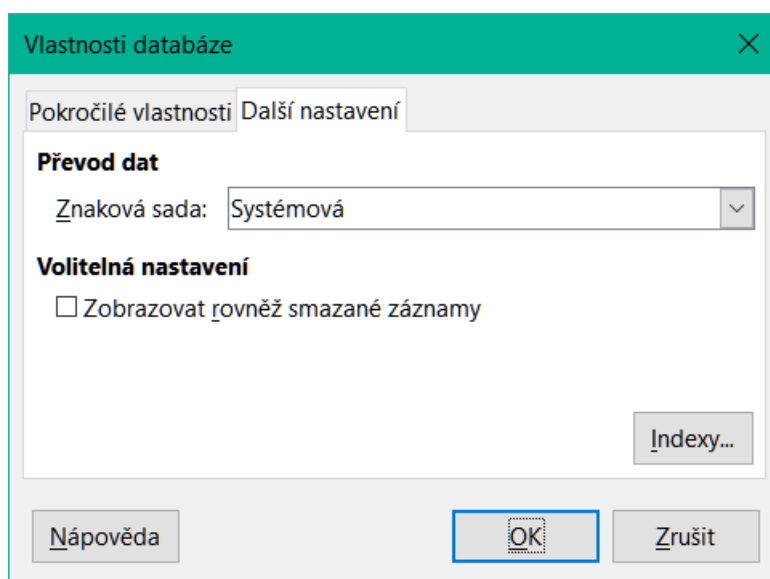
Možná omezení a řešení

- 1) Formát souboru DBF nepodporuje koncept deklarace sloupce tabulky jako primárního nebo cizího klíče.
- 2) Formát souboru DBF omezuje délku názvů polí v tabulkách na deset znaků.
- 3) Při vytváření nové tabulky dBASE je k dispozici podstatně méně různých typů polí než při vytváření tabulek pro většinu jiných databází (například při vytváření tabulky ve vložené databázi HSQLDB). To je vidět na obrázku 118.

	Název pole	Typ pole	
	ID	Integer [INTEGER]	
	Jméno	Text [VARCHAR]	
	► Příjmení	Text [VARCHAR]	▼
		Yes/No [BOOLEAN]	
		Memo [LONGVARCHAR]	
		Decimal [DECIMAL]	
		Decimal [NUMERIC]	
		Integer [INTEGER]	
		Double [DOUBLE]	
		Double [DOUBLE]	
		Text [VARCHAR]	
		Date [DATE]	
		Date/Time [TIMESTAMP]	

Obrázek 118: Typy polí dostupné při vytváření nové tabulky dBASE

- 4) Ve výchozím nastavení přebírá Base kódování znaků operačního systému. Proto je možné, že při importu speciálních znaků dojde ke vzniku chyb ve starších souborech dBASE. Znakovou sadu lze opravit pomocí nabídky *Znaková sada* na kartě *Další nastavení* v dialogovém okně *Vlastnosti databáze* (obrázek 119), které lze otevřít v hlavní nabídce **Úpravy > Databáze > Vlastnosti**.



Obrázek 119: Dialogové okno *Vlastnosti databáze*, karta *Další nastavení* – změna znakové sady

- 5) Při kopírování tabulky dBASE do jiné databáze (například do vložené databáze HSQLDB) má průvodce importem problémy s automatickým rozpoznáváním číselných typů polí a polí Ano/Ne (chyba 53027). To může vyžadovat následné opravy.

Tabulky a dokumenty Writer

Přehled

Base se může připojit k existujícímu tabulkovému procesoru nebo dokumentu textového procesoru a použít tabulková data z připojeného zdroje k vyplnění tabulky (tabulek) databáze. Tento způsob přístupu neumožňuje měnit žádná data v připojeném dokumentu. Base však lze použít k zobrazení informací, spouštění dotazů a vytváření sestav na základě dat, která jsou již

obsažena v připojeném dokumentu. Base se může připojit k sešitům Calc, sešitům Microsoft Excel, dokumentům Writer a dokumentům Microsoft Word.

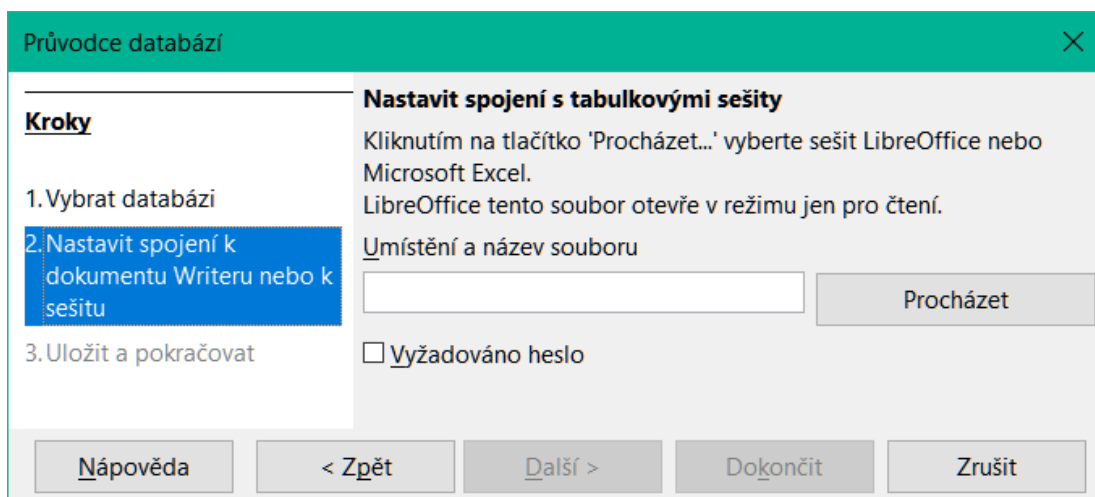
Připojení k sešitu

Při prvním připojení k určitému sešitu Calc nebo Microsoft Excel provedeme podobné kroky jako při přímém připojení k databázi MySQL/MariaDB (viz podkapitola Přímé připojení na straně 82), ale s následujícími důležitými rozdíly:

- Po kliknutí na volbu **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází, zvolíme v rozbalovací nabídce volbu **Sešit** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) a kroku 3 (*Nastavit MySQL server*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro **Sešity** relevantní. Namísto toho Base zobrazí krok 2 (*Nastavit spojení k dokumentu Writeru nebo k sešitu*) Průvodce databází (obrázek 120).

Zadáme umístění a název souboru tabulky. Kliknutím na tlačítko **Procházet** se zobrazí dialogové okno průzkumníka souborů, kde můžeme přejít na soubor, který nás zajímá, a poté kliknout na tlačítko **Otevřít**.

Pokud je sešit, která nás zajímá, chráněna heslem, klikneme na možnost **Vyžadováno heslo**.



Obrázek 120: Krok 2 Průvodce databází: Nastavení připojení dokumentu Writer nebo tabulkového procesoru

Base zpracovává tabulková data takto:

- Pro každý list v sešitu se vytvoří jedna databázová tabulka, přičemž databázová tabulka má stejný název jako příslušný list.
- Pro každou oblast databáze definovanou v sešitě Calc se vytvoří jedna databázová tabulka, přičemž databázová tabulka má stejný název jako odpovídající databázová oblast. Tato funkce se nevztahuje na „standardní“ pojmenované oblasti v sešitech Calc ani na sešity Microsoft Excel.

Kromě toho databáze Base umožňuje vytvářet tabulky, pohledy, vztahy a dotazy napříč více tabulkami.

Po navázání spojení a vytvoření souboru ODB aktualizuje databáze Base své hlavní okno tak, aby zobrazovalo informace ze sešitu.

Při připojení k tabulkovému procesoru se na stavovém řádku hlavního okna Base zobrazí následující informace:

- Název typu připojení (*Sešit*).
- Umístění a název připojeného sešitu (zadané v kroku 2 Průvodce databází).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k datům sešitu, může se zobrazit dialogové okno Vyžadována autentizace (podobná tomu na obrázku 99), v závislosti na tom, zda jsme v kroku 2 Průvodce databází vybrali možnost **Vyžadováno heslo**. Připojení nemusíme znovu konfigurovat, ale na požádání musíme znovu zadat správné přihlašovací údaje.

Tip

V tabulkovém procesoru Calc je možné zobrazit externí data získaná z internetu – více informací najdeme v kapitole 10, Propojení dat, v *Příručce programu Calc*. K těmto internetovým údajům lze zase přistupovat v aplikaci Base připojením k příslušné tabulce.

Připojení k dokumentu textového procesoru

Při prvním připojení k určitému dokumentu aplikace Writer nebo Microsoft Word provedeme podobné kroky jako při připojení k tabulkovému procesoru (viz podkapitola Připojení k sešitu na straně 118), ale s následujícím rozdílem:

- Po kliknutí na volbu **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází, zvolíme v rozbalovací nabídce volbu **Sešit** (místo **Sešit**).

Chování v kroku 2 (*Nastavit spojení k dokumentu Writeru nebo k sešitu*) Průvodce databází (Figure 120), je stejné když je v kroku 1 vybrán **Sešit** nebo **Dokument Writeru**.

Base zpracovává data v dokumentu následujícím způsobem:

- Pro každou tabulku v dokumentu se vytvoří jedna databázová tabulka, přičemž databázová tabulka má stejný název jako její tabulka v dokumentu. V programu Writer lze názvy tabulek zobrazit a upravit prostřednictvím Navigátor, který je přístupný výběrem v hlavní nabídce **Zobrazit > Navigátor**.
- Data v dokumentu, která nejsou v tabulce, jsou ignorována.

Kromě toho databáze Base umožňuje vytvářet tabulky, pohledy, vztahy a dotazy napříč více tabulkami.

Po navázání spojení a vytvoření souboru ODB aktualizuje databáze Base své hlavní okno tak, aby zobrazovalo informace z dokumentu.

Při připojení k dokumentu se na stavovém řádku hlavního okna Base zobrazují následující informace:

- Název typu připojení (*Textový dokument*).
- Umístění a název připojeného dokumentu (zadané v kroku 2 Průvodce databází).

Když později znovu otevřeme tento soubor ODB a poté použijeme Base pro přístup k datům sešitu, může se zobrazit dialogové okno Vyžadována autentizace (podobná tomu na obrázku 99), v závislosti na tom, zda jsme v kroku 2 Průvodce databází vybrali možnost **Vyžadováno heslo**. Připojení nemusíme znovu konfigurovat, ale na požádání musíme znovu zadat správné přihlašovací údaje.

Úprava dat v tabulkovém procesoru s otevřeným programem Calc i Base

Data v připojeném sešitu Calc nelze v aplikaci Base upravovat. Následující kroky ukazují, jak upravovat data v tabulkovém procesoru při současně otevřené databázi Base i připojeném tabulkovém procesoru Calc.

- 1) Ujistíme se, že přidružený sešit **NENÍ** otevřený, a poté otevřeme soubor Base.

- 2) Otevřeme související sešit. Calc otevře soubor s ochranou pouze pro čtení – všimněme si zprávy v horní části okna Calc „Tento dokument je otevřen v režimu pouze pro čtení“. Klikneme na sousední tlačítko **Upravit dokument**.
- 3) Otevřeme sešit v aplikaci Base. Pokud se v příslušné oblasti v programu Calc změnila existující data, zvolíme v hlavní nabídce **Data > Obnovit** v okně Base: Pohled na data tabulky, aby se změněná data zobrazila v souboru Base.
- 4) Pokud jsou v rámci aplikace Calc do příslušné oblasti přidána další data, nemusí se v předchozím kroku nová data zobrazit v Base. V takovém případě zavřeme okno Base: Pohled na data tabulky a v hlavním okně Base zvolíme v hlavní nabídce **Zobrazit > Obnovit tabulky**. Poté tabulku znovu otevřeme a nová data by měla být viditelná.

Kroky 3) a 4) výše data v sešitu bez uložení změn. K uložení upraveného souboru Calc se stejným názvem a do původního umístění by bylo nutné provést další kroky.

Podobnou strategii lze použít ke změně dat v souboru Writer při zobrazení změn v databázi Base.

Textové databáze

Přehled

Může se stát, že budeme potřebovat načíst data do databáze, ale zjistíme, že data nejsou ve formě přímo podporované ovladači poskytovanými v databázi Base. Taková data lze často převést do formátu CSV (hodnoty oddělené čárkou) nebo jiného vhodného textového formátu, případně pomocí programu Calc. Databáze Base nabízí dvě různé metody pro začlenění dat z textových souborů do databázových tabulek:

- 1) Pokud používáme vloženou nebo externí databázi HSQLDB, můžeme využít specifické možnosti HSQLDB pro vytváření textových tabulek.
- 2) Případně se můžeme připojit k existující sadě textových souborů pomocí Průvodce databází.

Tyto možnosti jsou popsány v následujících podkapitolách.

Textové tabulky HSQLDB

Textové tabulky HSQLDB jsou soubory CSV nebo jiné soubory s oddělenými znaky, se kterými se zachází jako s tabulkami SQL. Textovou tabulku tohoto druhu nelze vytvořit přímo pomocí grafického rozhraní Base, ale místo toho použijeme příkaz **CREATE TEXT TABLE** pomocí volby **Nástroje > SQL**. Například:

```
CREATE TEXT TABLE "company_data" (
  "Date" DATE PRIMARY KEY,
  "Open" DOUBLE,
  "High" DOUBLE,
  "Low" DOUBLE,
  "Close" DOUBLE,
  "Adj Close" DOUBLE,
  "Volume" BIGINT
);
```

vytvoří prázdnou textovou tabulku určenou k uložení historických dat o ceně akcií společnosti.

Následuje příkaz **SET TABLE**, který určuje soubor a oddělovací znak, který textová tabulka používá. Například:

```
SET TABLE "company_data"
SOURCE "share_prices.csv;encoding=UTF-8;ignore_first=true";
```


propojí soubor *share_prices.csv* s textovou tabulkou *company_data*. Tento příkaz předpokládá, že soubor CSV je ve stejné složce jako databáze.

Další informace nalezneme v kapitole Textové tabulky v Uživatelské příručce HyperSQL (<https://hsqldb.org/doc/2.0/guide/guide.pdf>).

Při používání textových tabulek HSQLDB v rámci databáze Base je třeba vzít v úvahu následující faktory:

- 1) Po vytvoření textové tabulky může být nutné zvolit v hlavní nabídce **Zobrazit > Obnovit tabulky**, než se nová tabulka zobrazí v oblasti *Tabulky* hlavního okna Base. V oblasti *Tabulky* se vzhled ikony pro textovou tabulku liší od vzhledu „normální“ tabulky.
- 2) Podkladový textový soubor je možné současně otevřít (a upravovat) v jiných externích programech, například v tabulkovém procesoru nebo v aplikacích pro úpravu textu. Toto chování by mohlo způsobit zmatek a ztrátu dat.
- 3) Při úpravě záznamu v databázi Base se změní pořadí záznamů v základním datovém souboru. Konkrétně se vyprázdní původní umístění záznamu a aktualizovaný záznam se připojí na konec souboru.
- 4) V uživatelské příručce HyperSQL je uvedeno upozornění, že textové tabulky nejsou tak odolné proti selhání stroje jako jiné typy tabulek. Je to proto, že jiné typy tabulek obsahují další mechanismy pro zachování integrity dat.

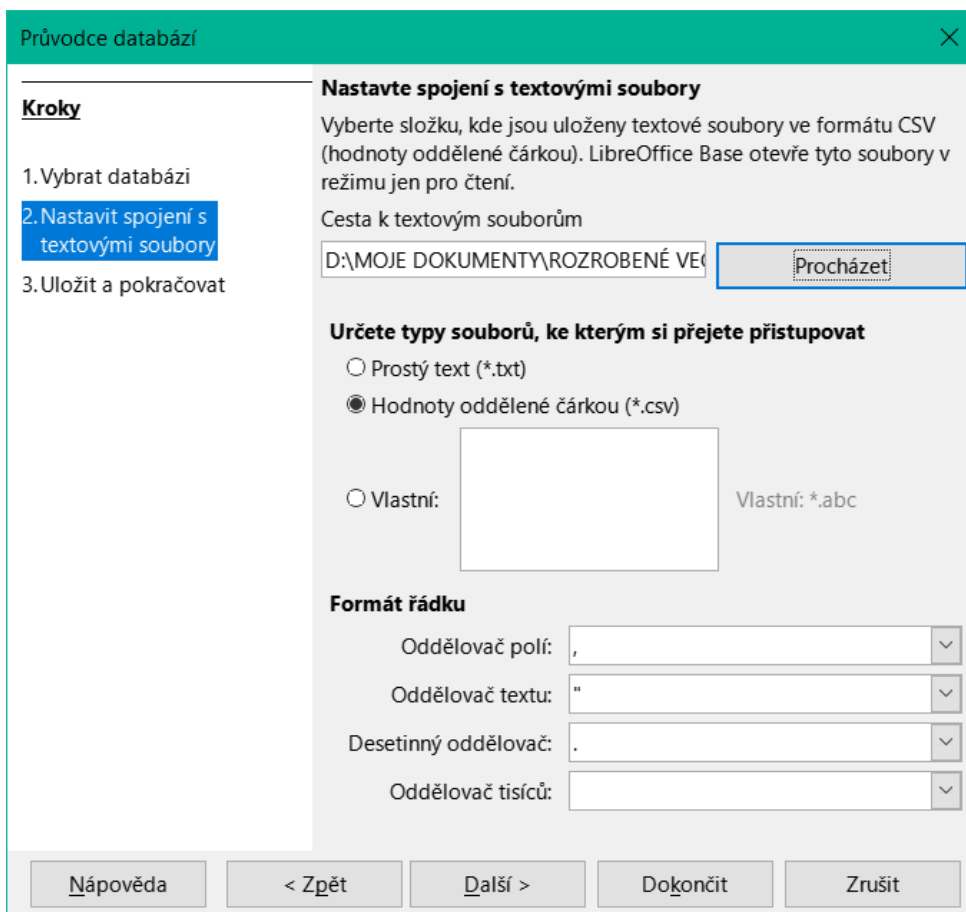
Připojení k textovým souborům

V aplikaci Base můžeme vytvořit kompletní databázi přístupem k sadě textových souborů ve složce. Tyto soubory budou často ve formátu CSV, ale není to nezbytné, i když všechny musí mít stejný formát. Soubory budou otevřeny pouze pro čtení, bez možnosti zápisu.

Textové databáze v této podobě sice často nejsou vhodné pro běžné použití, ale lze je použít k rychlému vyhledávání zajímavých záznamů v souboru CSV nebo k importu souboru CSV do jiné databáze pomocí funkce kopírování. Soubor CSV stačí přesunout do určené složky a poté jej lze snadno vyhledávat nebo kopírovat.

Při prvním připojení k určité textové databázi provedeme podobné kroky jako při přímém připojení k databázi MySQL/MariaDB (viz podkapitola Přímé připojení na straně 82), ale s následujícími důležitými rozdíly:

- Po kliknutí na **Připojit se k databázi** v kroku 1 (*Výběr databáze*) Průvodce databází vybereme z rozbalovací nabídky **Text** (místo **MySQL**).
- Možnosti uvedené v kroku 2 (*Nastavit spojení s MySQL*) a kroku 3 (*Nastavit MySQL server*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro textovou databázi relevantní. Namísto toho Base zobrazí krok 2 (*Nastavit spojení s textovými soubory*) Průvodce databází (obrázek 121).



Obrázek 121: Krok 2 Průvodce databází: Nastavení připojení k textovým souborům

- Zadáme umístění složky obsahující textové soubory. Kliknutím na tlačítko **Procházet** se zobrazí dialogové okno Výběr složky, kde můžeme přejít na požadovanou složku a poté kliknout na tlačítko **Vybrat složku**.
- Vybereme typ souboru, který chceme použít, **Prostý text (*.txt)**, **Text oddělený čárkami (*.csv)**, nebo **Vlastní**. V druhém případě zadáme příponu souboru, která se má použít.
- Upravíme (v případě potřeby) čtyři oddělovače použité v řádcích všech vybraných souborů.
- Možnosti uvedené v kroku 4 (*Nastavit autentizaci uživatele*) Průvodce databází při přímém připojení k databázi MySQL/MariaDB nejsou pro textové databáze relevantní.

Po navázání spojení a vytvoření souboru ODB aktualizuje Base své hlavní okno a zobrazí informace o textové databázi. V oblasti *Tabulky* se zobrazí seznam vytvořených tabulek, jedna pro každý soubor správného typu, který se nachází v zadané složce. Název tabulky bude nastaven na odpovídající název souboru (bez přípony). Kromě toho databáze Base umožňuje vytvářet tabulky, pohledy, vztahy a dotazy napříč více tabulkami.

Při připojení k textové databázi se ve stavovém řádku hlavního okna Base zobrazí následující informace:

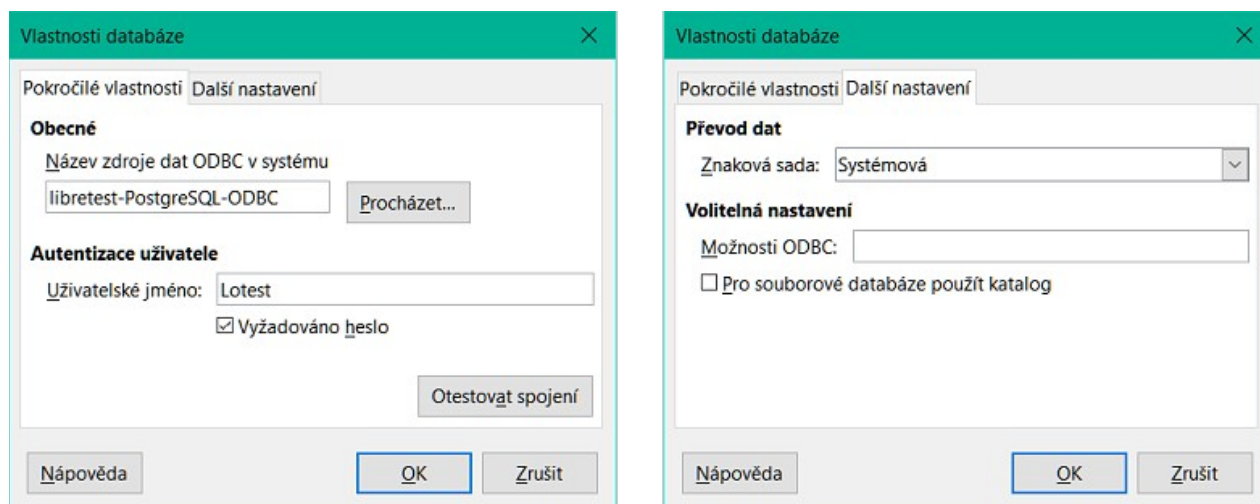
- Název použitého konektoru (*Text*).
- Složka obsahující databázové soubory (v tomto případě *C:\my_databases\csv_files*).

Následná úprava vlastností připojení

Ve většině případů povedou pokyny uvedené v předchozích částech této kapitoly k připojení k externí databázi, které bude fungovat podle očekávání. Občas se však může stát, že připojení nebude fungovat tak, jak má. Může se například změnit něco v základních parametrech připojení (například řetězec připojení, název ovladače nebo uživatelské jméno databáze), může být potřeba jiná znaková sada nebo dílčí formuláře nemusí fungovat správně. Změny vlastností připojení lze provést výběrem **Úpravy > Databáze** v hlavní nabídce a následným výběrem jedné ze tří možností:

- 1) **Vlastnosti.** Dialogové okno Vlastnosti databáze umožňuje měnit vlastnosti aktuálního připojení, ale nepodporuje výběr alternativního typu připojení. Nabízí jednu nebo více karet, na kterých lze provést úpravy parametrů připojení, z nichž většina byla původně nakonfigurována prostřednictvím Průvodce databází.
- 2) **Typ připojení.** Vyvolá Průvodce vlastnostmi databáze, který umožňuje přepnutí na jiný typ připojení. Můžeme například přepnout z připojení ODBC na JDBC nebo přímé připojení. To může být užitečné, pokud chceme vyzkoušet, který typ připojení nejlépe vyhovuje naší aplikaci. Možnosti zobrazené v jednotlivých krocích jsou podobné jako v Průvodci databází.
- 3) **Pokročilá nastavení.** Dialogové okno Rozšířená nastavení obsahuje jednu nebo více karet s ovládacími prvky pro úpravu různých nastavení, která nejsou nakonfigurována při použití Průvodce databází. Dostupné možnosti závisí na typu databáze a připojení. V závislosti na databázovém systému existují různé příkazy pro automatické generování přírůstkových hodnot – pomocí karty *Generované hodnoty* lze řídit zacházení s automaticky generovanými hodnotami pro nové datové záznamy – další informace získáme vyhledáním „Generované hodnoty“ v modulu Base systému nápovědy. Záložka *Speciální nastavení* umožňuje zadat různé podrobné volby, které ovlivňují způsob, jakým Base spolupracuje s databází – další informace nalezneme v nápovědě modulu Base.

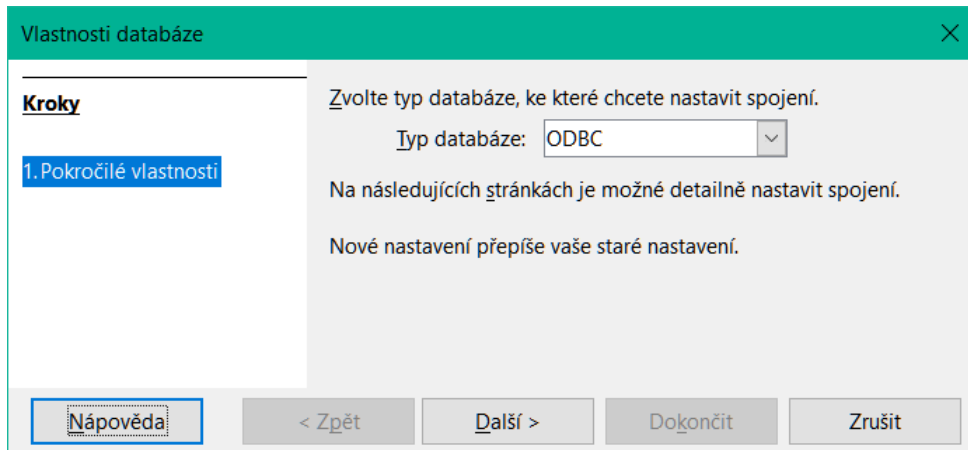
Obrázek 122 představuje typický příklad dialogového okna Vlastnosti databáze pro připojení ODBC k databázi PostgreSQL. V tomto případě dialogové okno obsahuje dvě karty, *Pokročilé vlastnosti* a *Další nastavení*. Na kartě *Pokročilé vlastnosti* jsou ovládací prvky pro výběr alternativního zdroje dat ODBC nebo jiných přihlašovacích údajů. Karta *Další nastavení* umožňuje výběr jiné znakové sady a zadání volitelných nastavení ODBC.



Obrázek 122: Dialogové okno Vlastnosti databáze – karta Rozšířené vlastnosti (vlevo) a karta Další nastavení (vpravo)

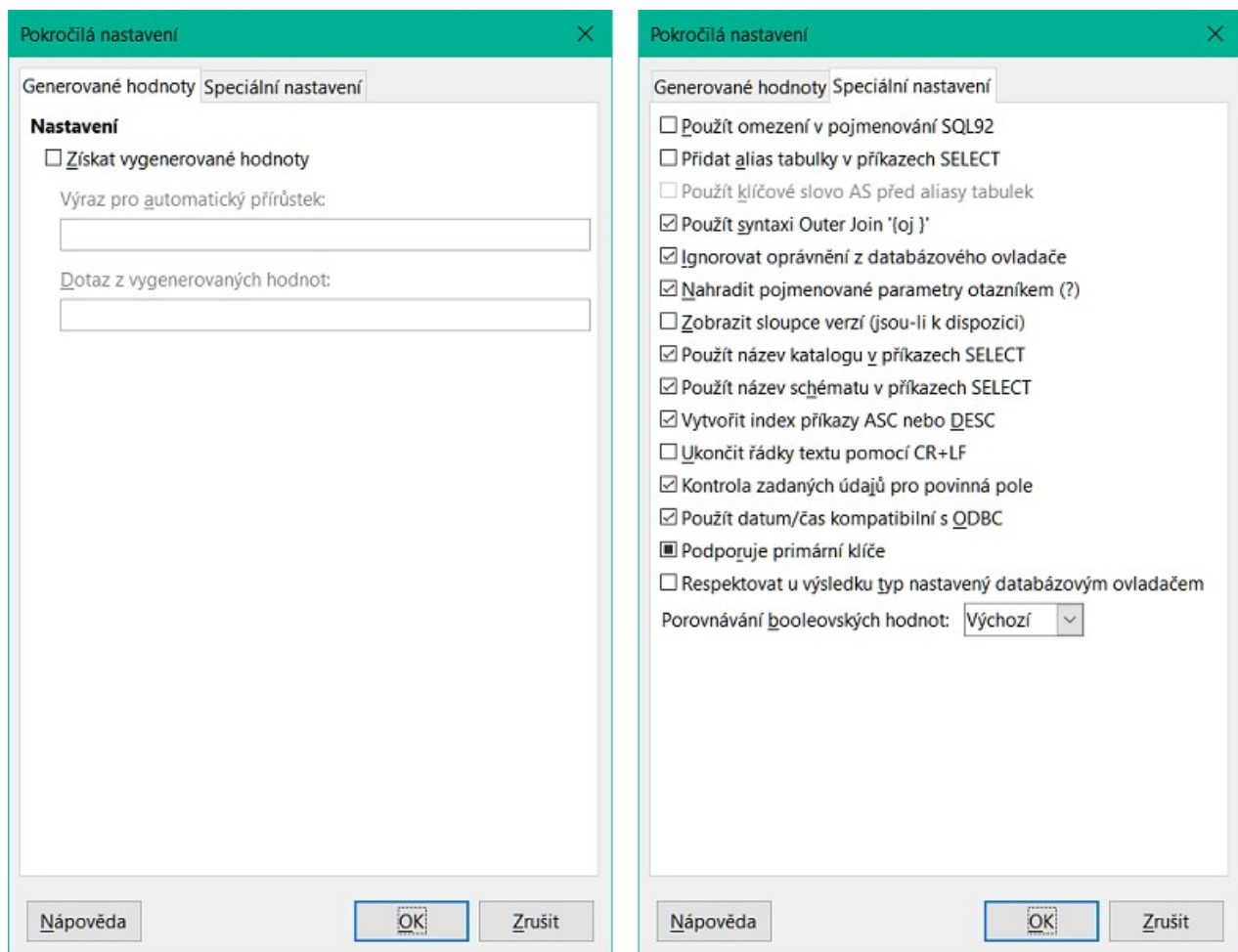
Obrázek 123 zobrazuje první krok Průvodce vlastnostmi databáze. Ve výchozím nastavení je rozevírací seznam *Typ databáze* nastaven tak, aby odrážel aktuální typ připojení (v tomto příkladu **ODBC**). Z nabídky vybereme jinou možnost pro změnu typu připojení a poté klikneme

na tlačítko **Další** a začneme vyplňovat parametry nového typu připojení, podobně jako v příslušných krocích Průvodce databází.



Obrázek 123: Krok 1 Průvodce vlastnostmi databáze: Pokročilé vlastnosti

Obrázek 124 ukazuje příklad dialogového okna Upřesnit nastavení pro připojení ODBC k databázi PostgreSQL. V tomto případě dialogové okno obsahuje dvě karty, *Generované hodnoty* a *Speciální nastavení*. Některé možnosti na kartě *Speciální nastavení* budou možná nedostupná (zašedlá) pokud je nejde pro databázi změnit (například volba **Použít klíčové slovo AS před aliasy tabulek** na obrázku 124).



Obrázek 124: Dialogové okno Rozšířená nastavení – karta Generované hodnoty (vlevo) a karta Speciální nastavení (vpravo)

Některá pokročilá nastavení nelze změnit prostřednictvím grafického uživatelského rozhraní Base. Za určitých okolností však zkušený uživatel dokáže upravit soubor *content.xml* v rámci souboru ODB databáze a problém vyřešit. Zde je příklad, který způsobil problémy při přechodu z LibreOffice 6.0 na LibreOffice 6.1: Dílčí dotazy v MySQL již nefungovaly podle očekávání, protože byl znemožněn přenos spojovací hodnoty (parametru). To lze napravit otevřením souboru *content.xml* a nalezením kódu:

```
a<db:driver-settings db:system-driver-settings="" db:base-dn=""  
db:parametername-substitution="false"/>
```

Změna tohoto řádku na:

```
<db:driver-settings db:system-driver-settings="" db:base-dn=""/>
```

vyřešila problém, který v té době existoval. Alternativně bylo možné napsat makro, které by dosáhlo stejného výsledku.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 3, Tabulky

Obecné informace o tabulkách

Databáze ukládají data do tabulek. Hlavní rozdíl oproti tabulkám v databázi a oblasti buněk v jednoduchém tabulkovém procesoru spočívá v tom, že datový typ polí musí být předem jasně definován. Databáze například neumožňuje, aby textové pole obsahovalo čísla pro použití ve výpočtech. Taková čísla se zobrazují, ale pouze jako řetězce, jejichž skutečná číselná hodnota je nula. Stejně tak nelze obrázky zahrnout do všech typů polí.

Podrobnosti o tom, které typy dat jsou k dispozici, lze získat v okně Návrh tabulky v programu Base. Typy dat jsou uvedeny v příloze A této knihy.

Jednoduché databáze jsou založeny pouze na jedné tabulce. Všechny datové prvky se zadávají nezávisle, což může vést k vícenásobnému zadávání stejných údajů. Tímto způsobem lze vytvořit jednoduchý adresář pro soukromé použití. Adresář školy nebo sportovního svazu však může obsahovat tolik opakujících se poštovních směrovacích čísel a míst, že je lepší umístit tato pole do jedné nebo dokonce dvou samostatných tabulek.

Ukládání dat do samostatných tabulek pomáhá:

- Omezit opakované zadávání stejného obsahu
- Předcházet pravopisným chybám v důsledku opakovaného zadávání
- Zlepšit filtrování dat v zobrazených tabulkách

Při vytváření tabulky bychom měli vždy zvážit, zda se v tabulce může vyskytovat více opakování, zejména textu nebo obrázků (které spotřebovávají velké množství kapacity úložiště). Pokud ano, je třeba je exportovat do jiné tabulky. Jak to v zásadě udělat, je popsáno v kapitole 1, Úvod do programu Base, v části „Jednoduchá databáze – podrobný testovací příklad“.



Poznámka

Relační databáze je skupina tabulek, které jsou vzájemně propojeny společnými atributy. Účelem relační databáze je co nejvíce zabránit duplicitnímu zadávání datových prvků. Je třeba se vyvarovat nadbytečných dat.

Toho lze dosáhnout:

- Rozdělením obsahu do co největšího počtu jedinečných polí (například místo jednoho pole pro kompletní adresu použijeme samostatná pole pro číslo domu, ulici, město a PSČ).
- Zabráněním duplicitním údajům pro jedno pole ve více záznamech (například importem PSČ a města z jiné tabulky).

Tyto postupy se nazývají *Normalizace databáze*.

Relace mezi tabulkami

V této kapitole je řada těchto kroků podrobně vysvětlena na příkladu databáze pro knihovnu: `Media_without_Macros`. Sestavení tabulek pro tuto databázi je rozsáhlá práce, protože zahrnuje nejen přidávání položek do knihovny médií, ale také jejich následné půjčování.

Relace pro tabulky v databázích

Tabulky v interní databázi HSQLDB mají vždy charakteristické, jedinečné pole, *primární klíč*. Toto pole musí být definováno před zápisem jakýchkoli dat do tabulky. Pomocí tohoto pole lze vyhledat konkrétní záznamy v tabulce.

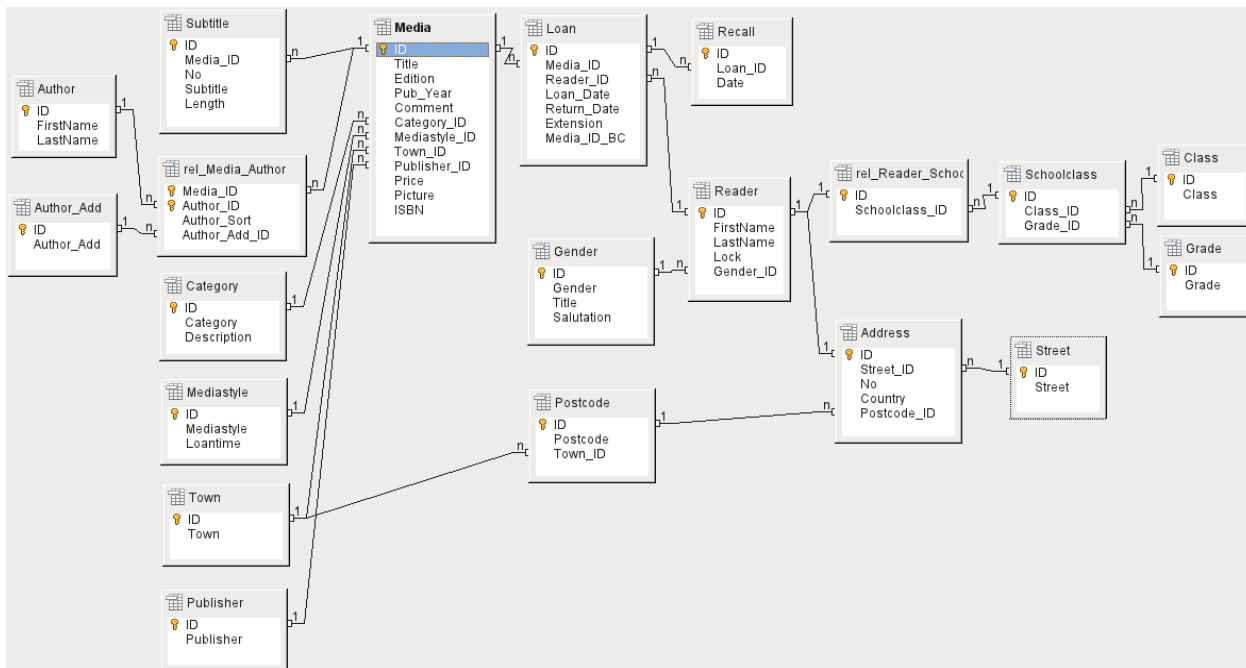
V některých případech je primární klíč tvořen kombinací několika polí. Tato pole musí být jedinečná, pokud jsou posuzována společně. To se nazývá *složený primární klíč*.

✓ Poznámka

Kombinace polí ve složeném primárním klíči je jedinečná, pokud každý záznam tabulky obsahuje jedinečnou kombinaci hodnot těchto polí.

Předpokládejme, že existují dvě tabulky, Tabulka1 a Tabulka2. Tabulka 2 může obsahovat pole, které označuje záznam v tabulce 1. Zde je primární klíč tabulky 1 zapsán jako hodnota v poli tabulky 2. Tabulka 2 má nyní pole, které ukazuje na pole klíče jiné tabulky, známé jako *cizí klíč*. Tento cizí klíč existuje v tabulce 2 vedle jejího primárního klíče.

Čím více relací je mezi tabulkami, tím složitější je úloha návrhu. Obrázek 125 znázorňuje celkovou strukturu tabulek této příkladové databáze, zmenšenou tak, aby odpovídala velikosti stránky tohoto dokumentu. Chceme-li si přečíst obsah, zvětšíme stránku na 200 %.



Obrázek 125: Diagram vztahů pro databázi Media_without_Macros

Relace typu jeden k mnoha

Databáze Media_without_Macros obsahuje názvy médií v jedné tabulce. Vzhledem k tomu, že tituly mohou mít více podtitulů nebo někdy vůbec žádné, jsou podtituly uloženy v samostatné tabulce.

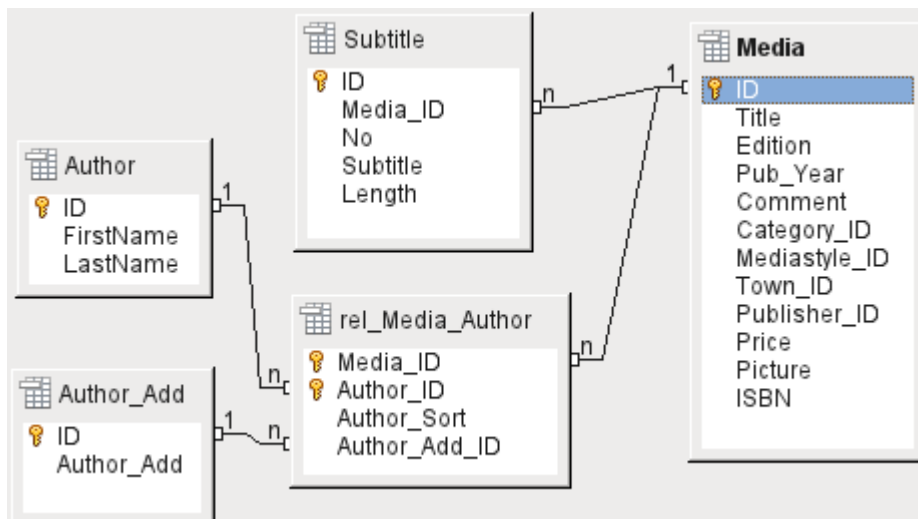
Relace, ve kterém může mít pole více než jeden cizí klíč, se nazývá relací *jeden k mnoha* (1:n). Jednomu médiu může být přiřazeno mnoho podtitulů, například mnoho názvů skladeb na hudebním CD. Primární klíč pro tabulku Media je uložen jako cizí klíč v tabulce Subtitle. Většina relací mezi tabulkami v databázi jsou vztahy typu „jeden k mnoha“.

Relace typu mnoho k mnoha

Databáze pro knihovnu může obsahovat tabulku se jmény autorů a tabulku s médii. Spojitost mezi autorem a například knihami, které napsal, je zřejmá. Knihovna může obsahovat více knih od jednoho autora. Může také obsahovat knihy s více autory. Tato relace, ve které mohou mít obě tabulky více než jeden cizí klíč, se nazývá *mnoho k mnoha* (n:m). Takové relace vyžadují tabulku, která funguje jako prostředník mezi oběma příslušnými tabulkami. To je na obrázku 126 znázorněno tabulkou rel_Media_Author.

V praxi se tedy relace n:m řeší tak, že se považuje za dvě relace 1:n. V mezitabulce se může pole Media_ID vyskytovat více než jednou, stejně jako pole Author_ID. Pokud je však použijeme

jako pár, nedochází k duplikaci: žádné dva páry nejsou identické. Tato dvojice tedy splňuje požadavky na primární klíč pro mezi tabulku.

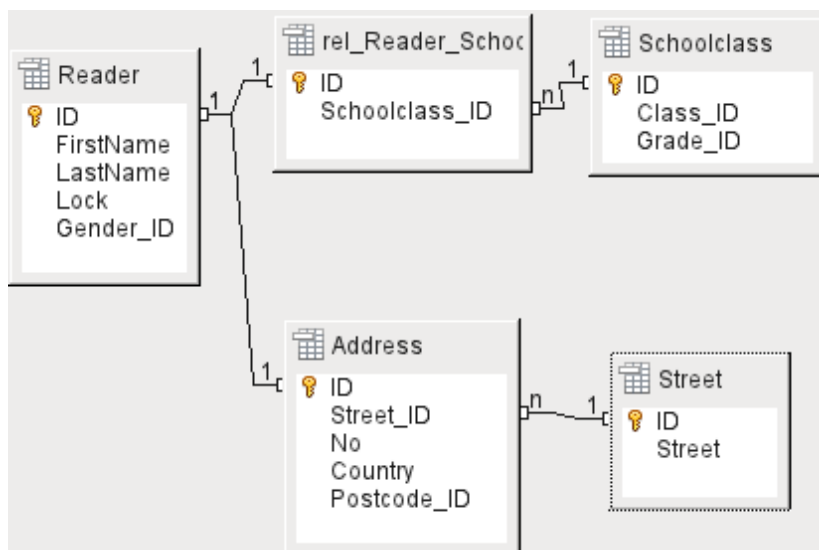


Obrázek 126: Příklad relace 1:n; relace n:m

✓ Poznámka

Pro danou hodnotu Media_ID existuje pouze jeden název média a jedno ISBN. Pro danou hodnotu Author_ID existuje pouze jedno jméno a příjmení autora. Pro danou dvojici těchto hodnot tedy existuje pouze jedno číslo ISBN a pouze jeden autor. Díky tomu je tato dvojice jedinečná.

Relace typu jeden k jednomu



Obrázek 127: Příklad relace 1:1

V tabulce Reader byla předem naplánována pouze pole, která jsou přímo nezbytná. Naše databáze však zahrnuje dva případy: školní a veřejnou knihovnu. U databáze školní knihovny je u čtenáře vyžadována také školní třída. Ve školních třídních výkazech můžeme v případě potřeby zjistit adresy dlužníků. Proto není nutné tyto adresy uvádět v databázi školní knihovny. Relace žáků ke třídě školy je rovněž oddělen od tabulky čtenářů, protože mapování na třídy není ve všech oblastech vhodné. Mezi čtenářem a adresou a čtenářem a třídou existuje vztah 1:1.

V databázi pro veřejnou knihovnu jsou vyžadovány adresy čtenářů. Pro každého čtenáře existuje jedna adresa. Pokud je na stejné adrese více čtenářů, bylo by nutné tuto strukturu zadat znovu, protože primární klíč tabulky Reader je zadán přímo jako primární klíč v tabulce Address. Primární klíč a cizí klíč jsou v tabulce Address jeden a tentýž. Jedná se tedy o relaci 1:1.

Relace 1:1 neznamená, že pro každý záznam v tabulce bude existovat odpovídající záznam v jiné tabulce. **Nejvýše** však bude existovat pouze jeden odpovídající záznam. Relace 1:1 tedy vede k tomu, že se exportují pole, která budou vyplněna obsahem pouze některých záznamů.

Tabulky a relace pro ukázkovou databázi

Příkladová databáze (Media_without_Macros) musí splňovat tři požadavky: přidávání a odebírání médií, výpůjčky a správa uživatelů.

Tabulka pro přidávání médií

Nejprve je třeba přidat média do databáze, aby s nimi knihovna mohla pracovat. Pro jednoduché shrnutí sbírky médií však můžeme vytvořit jednodušší databáze pomocí průvodce; to by mohlo být pro domácí použití dostačující.

Centrální tabulkou pro přidávání médií je tabulka Media (viz obrázek 128).

V této tabulce se předpokládá, že všechna pole, která jsou přímo zadána, se nepoužívají také pro jiná média se stejným obsahem. Proto je třeba se vyhnout duplicitě.

Z tohoto důvodu jsou v tabulce plánována pole s názvem, ISBN, obrázkem obálky a rokem vydání. Seznam polí lze v případě potřeby rozšířit. Knihovníci tak mohou například chtít zahrnout pole pro velikost (počet stran), název série atd.

Tabulka Subtitle obsahuje podrobný obsah CD. Vzhledem k tomu, že CD může obsahovat několik hudebních skladeb, záznam jednotlivých skladeb v hlavní tabulce by vyžadoval mnoho dalších polí (Subtitle 1, 'Subtitle 2 atd.) nebo by se stejná položka musela zadávat mnohokrát. Tabulka Subtitle je tedy v relaci n:1 k tabulce Media.

Pole tabulky Subtitle jsou (kromě samotného podnadpisu) pořadové číslo podnadpisu a doba trvání stopy. Pole Length musí být nejprve definováno jako pole pro čas. Tímto způsobem lze vypočítat celkovou dobu trvání CD a v případě potřeby ji zobrazit v přehledu.

Autoři mají k médiím relaci n:m. Jedna položka může mít několik autorů a jeden autor může vytvořit několik položek. Tato relace je řízena tabulkou rel_Media_Author. Primárním klíčem této propojovací tabulky je cizí klíč vytvořený z tabulek Author a Media. Tabulka rel_Media_Author obsahuje dodatečné třídění autorů (Author_Sort), například podle pořadí, v jakém jsou v knize uvedeni. Kromě toho se v případě potřeby k autorovi přidá doplňující označení, jako je producent, fotograf apod.

Category, Mediastyle, Town a Publisher mají relaci 1:n.

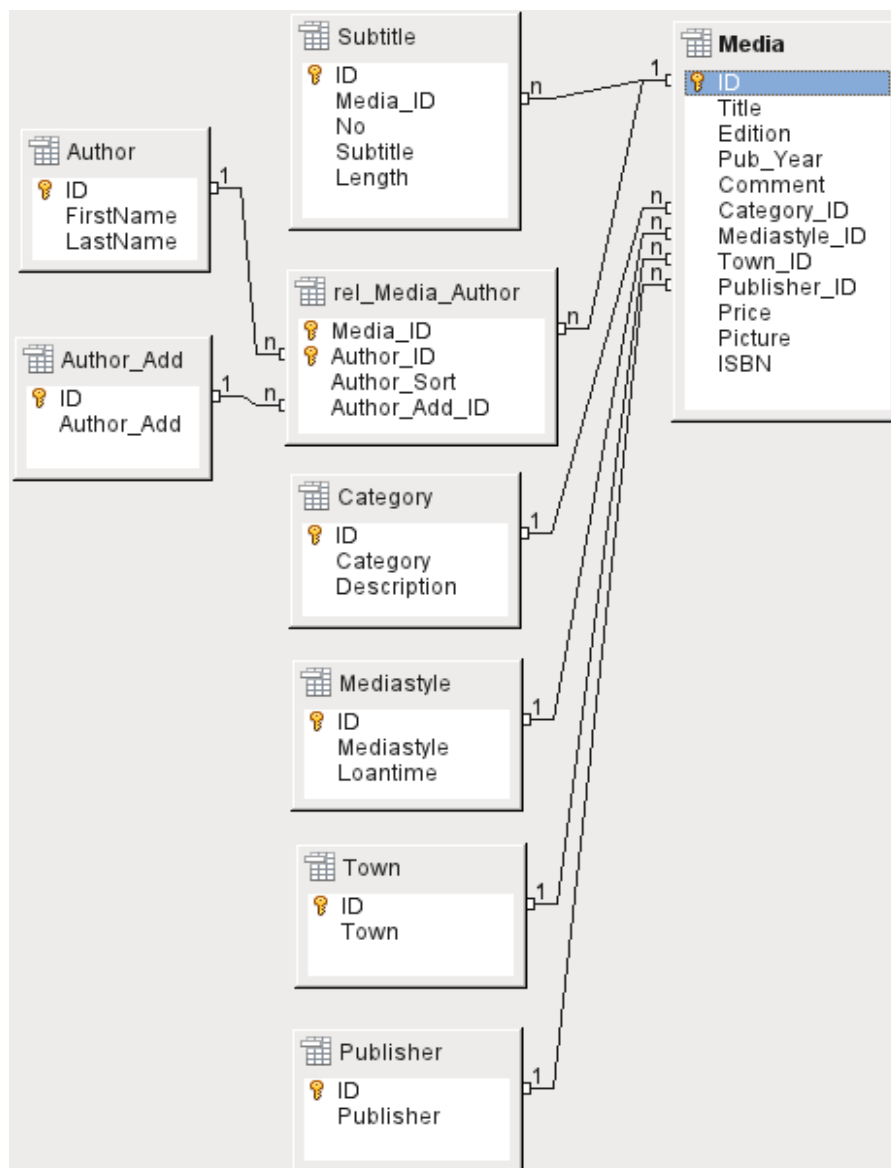
Pro pole *Category* může malá knihovna použít hodnoty jako Art nebo Biology. Pro větší knihovny jsou k dispozici obecné systémy pro knihovny. Tyto systémy poskytují jak zkratky, tak úplné popisy. Obě pole se proto zobrazují v položce Category.

Mediastyle je spojen s výpůjční dobou Loantime. Například videodisky DVD se zásadně půjčují na 7 dní, ale knihy se mohou půjčovat na 21 dní. Pokud je výpůjční doba spojena s nějakým jiným kritériem, dojde k odpovídajícím změnám v naší metodice.

Tabulka *Town* slouží nejen k ukládání údajů o poloze z médií, ale také k ukládání polohy použité v adresách uživatelů.

Protože se *Publishers* (vydavatelé) také často opakují, je pro ně vytvořena samostatná tabulka.

Tabulka Media má celkem čtyři cizí klíče a jeden primární klíč, který se používá jako cizí klíč ve dvou tabulkách, jak ukazuje obrázek 128.



Obrázek 128: Přidání médií

Tabulka půjček

Ústřední tabulkou je tabulka *Loan* (viz obrázek 129). Je to spojovací článek mezi tabulkami *Media* a *Reader*.

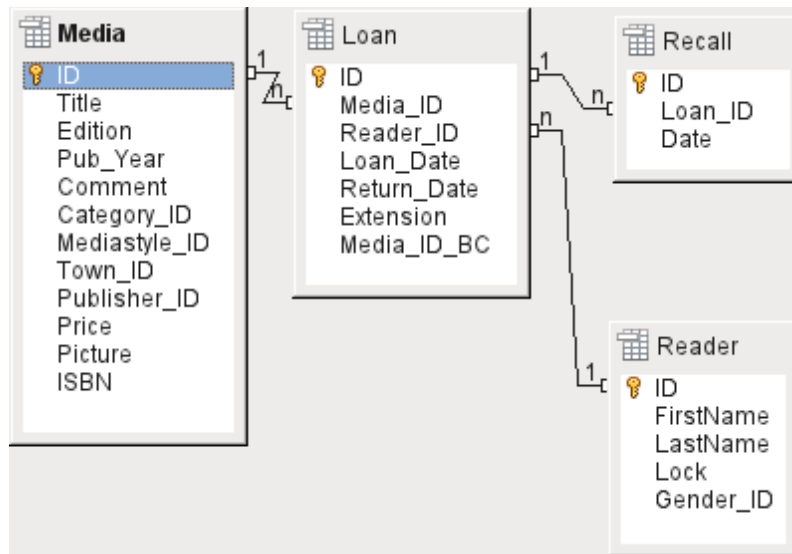
Po vrácení média lze většinu jeho dat smazat, protože již nejsou potřeba. Dvě z těchto polí by však neměla být pole: *ID* a *Loan_Date*. Prvním z nich je primární klíč. Druhá možnost je, když byl předmět zapůjčen. Slouží ke dvěma účelům. Nejprve je užitečné určit nejoblíbenější média. Za druhé, pokud je při vynášení předmětu zjištěno jeho poškození, zobrazí se v tomto poli informace o tom, kdo si předmět vypůjčil jako poslední. Kromě toho se při vrácení položky zaznamená údaj *Return_Date*.

Podobně jsou do výpůjčního procesu integrovány i *Reminders* (upomínky). Každá upomínka se zadává zvlášť do tabulky *Recall*, aby bylo možné určit celkový počet upomínek.

Kromě doby prodloužení v týdnech je v záznamu o výpůjčce další pole, které umožňuje vypůjčit médium pomocí snímače čárového kódu (*Media_ID_BC*). Čárové kódy obsahují kromě individuálního *Media_ID* také kontrolní číslici, podle které může skener určit, zda je naskenovaná hodnota správná. Toto pole čárového kódu je zde uvedeno pouze pro testovací účely. Bylo by

lepší, kdyby primární klíč tabulky Media bylo možné zadat přímo ve formě čárového kódu nebo kdyby se před uložením použilo makro, které by ze zadaného čísla čárového kódu odstranilo kontrolní číslici.

Nakonec musíme připojit čtenáře k výpůjčce. V tabulce Reader je v plánu uveden pouze název, volitelný zámeček a cizí klíč odkazující na tabulku Gender.

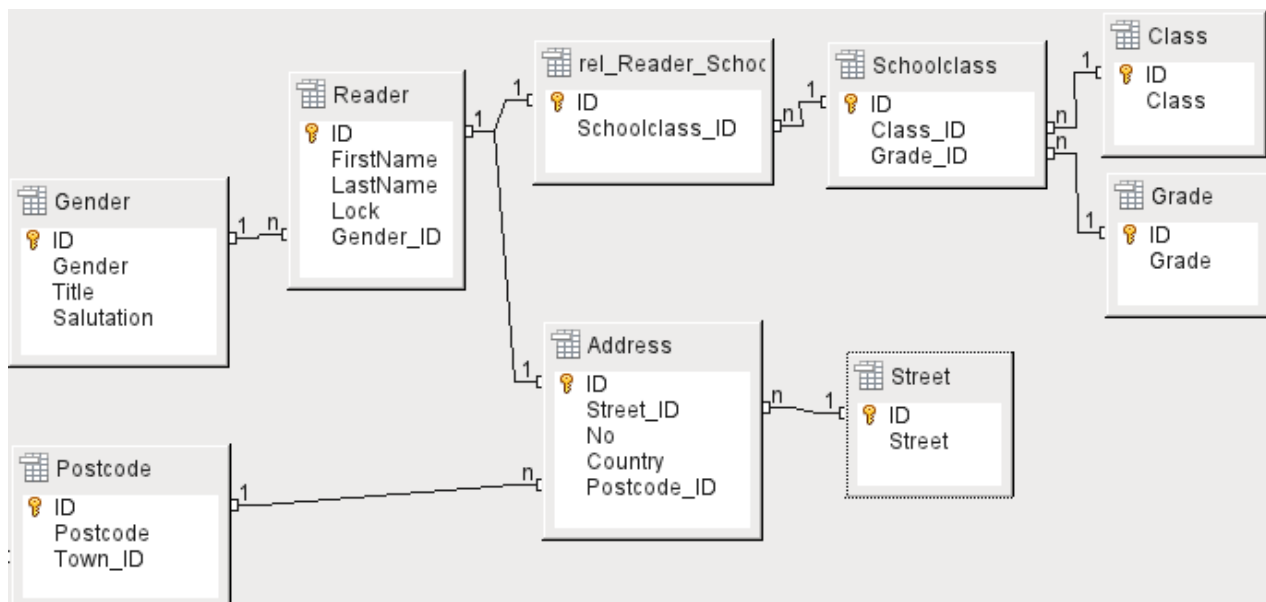


Obrázek 129: Relace v tabulce Loan

Tabulka správy uživatelů

U tohoto návrhu tabulky se počítá se dvěma scénáři. Řetězec tabulek na obrázku 130 je určen pro školní knihovny. Zde není třeba uvádět adresy, protože žáky lze kontaktovat prostřednictvím školy. Upomínky není nutné rozesílat poštou, ale lze je distribuovat interně.

Řetězec adresy je nezbytný v případě veřejných knihoven. Zde je třeba zadat údaje, které budou potřebné pro vytvoření upomínacích dopisů. Viz obrázek 130.



Obrázek 130: Čtenáři – řetězec školních tříd a řetězec adres

Tabulka *Pohlaví* zajišťuje, aby se v upomínkách používaly správné pozdravy. Psaní upomínek pak může být v maximální možné míře automatizováno. Kromě toho mohou být některá jména stejně mužská i ženská. Proto je nutné uvádět pohlaví zvlášť, i když se upomínky vypisují ručně.

Tabulka *rel_Reader_Schoolclass* má stejně jako tabulka *Address* relaci 1:1 s tabulkou *Reader*. Tato možnost byla zvolena proto, že může být vyžadována buď školní třída nebo adresa. V opačném případě by bylo možné vložit identifikátor *Schoolclass_ID* přímo do tabulky žáků; totéž by platilo pro kompletní obsah tabulky adres v systému veřejných knihoven.

Třída školy se obvykle skládá z označení ročníku a přípony paralelních tříd. Ve škole se čtyřmi paralelními třídami může tato přípona nabývat hodnot od *a* do *d*. Přípona se zadává v tabulce *Class*. Ročník je v samostatné tabulce *Grade*. Pokud se čtenáři na konci každého školního roku přesunou do vyšší třídy, můžeme jednoduše změnit zápis ročníku pro všechny.

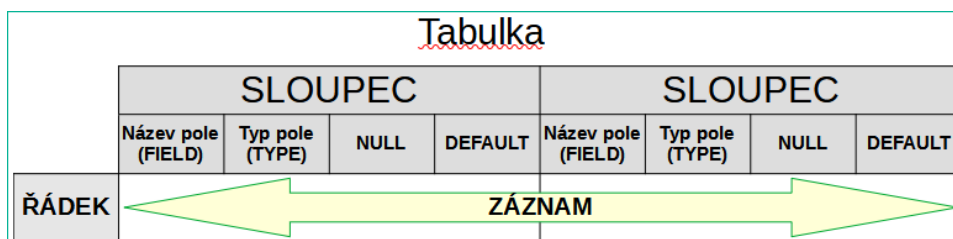
Adresa je také rozdělena. *Ulice* se ukládá zvlášť, protože názvy ulic v oblasti se často opakují. Poštovní směrovací číslo a obec jsou odděleny, protože pro jednu oblast často existuje několik poštovních směrovacích čísel, a tedy více poštovních směrovacích čísel než obcí. V porovnání s tabulkou *Address* tak tabulka *Postcode* obsahuje výrazně méně záznamů a tabulka *Town* ještě méně.

Použití této struktury tabulek je dále vysvětleno v kapitole 4, Formuláře, v této knize.

Tvorba tabulek

Většina uživatelů LibreOffice bude k vytváření tabulek obvykle používat výhradně grafické uživatelské rozhraní (GUI). Přímé zadání příkazů SQL je nutné například tehdy, když je třeba dodatečně vložit pole na určitou pozici nebo nastavit standardní hodnotu po uložení tabulky.

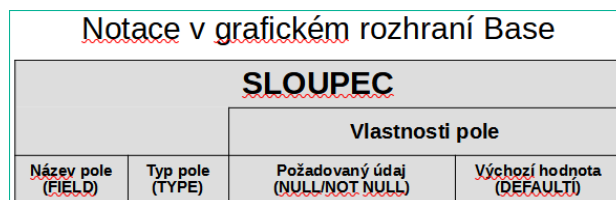
Tabulková terminologie: Obrázek 131 ukazuje standardní rozdělení tabulek na sloupce a řádky.



Obrázek 131: Standardní tabulková terminologie

Každý datový záznam je uložen ve vlastním řádku tabulky. Jednotlivé sloupce jsou z velké části definovány polem, typem a pravidly, která určují, zda může být pole prázdné. Podle typu lze také určit velikost pole ve znacích. Kromě toho lze zadat výchozí hodnotu, která se použije v případě, že do pole nebylo zadáno nic.

V základním grafickém uživatelském rozhraní jsou pojmy pro sloupec popsány poněkud odlišně, jak ukazuje obrázek 132.



Obrázek 132: Terminologie tabulky aplikace Base

Pole se změní na Název pole, Typ se změní na Typ pole. Název pole a Typ pole se zadávají do horní části okna Návrh tabulky. Další vlastnosti sloupce můžeme nastavit v části Vlastnosti pole ve spodní části. Tyto další vlastnosti závisí na typu sloupce definovaném výše a jsou také

omezeny uživatelským rozhraním. Chceme-li překonat omezení rozhraní, jako je nastavení výchozí hodnoty pole data na skutečné datum zadání, použijeme vhodný příkaz SQL (viz „Přímé zadávání SQL příkazů“ na straně 144).

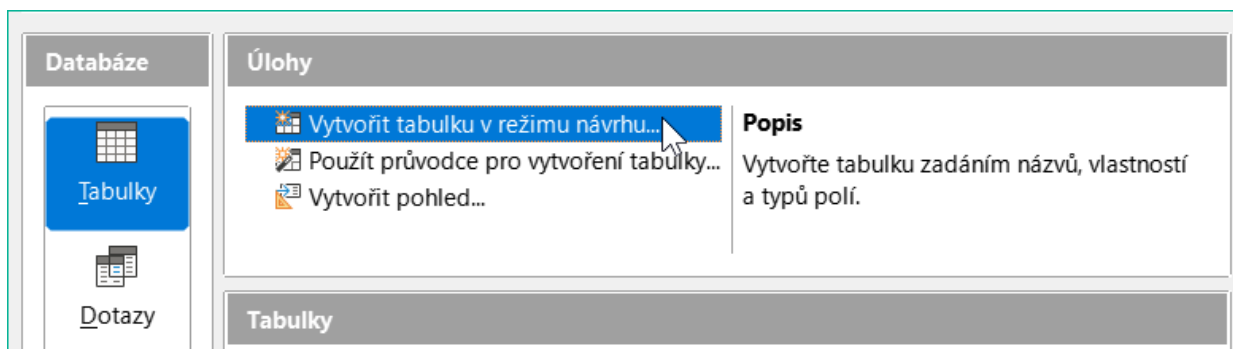
✓ Poznámka

Výchozí hodnoty: Termín „Výchozí hodnota“ v grafickém uživatelském rozhraní neznámá to, co uživatel databáze obecně chápe jako výchozí hodnotu. Grafické uživatelské rozhraní viditelně zobrazuje určitou hodnotu, která je uložena spolu s daty.

Výchozí hodnota v databázi je uložena v definici tabulky. Do pole se pak zapíše vždy, když je pole v novém datovém záznamu prázdné. Při úpravě vlastností tabulky se nezobrazují výchozí hodnoty SQL.

Tvorba pomocí grafického uživatelského rozhraní

Vytvoření databáze pomocí grafického uživatelského rozhraní (GUI) je vysvětleno krok za krokem na příkladu tabulky Media.

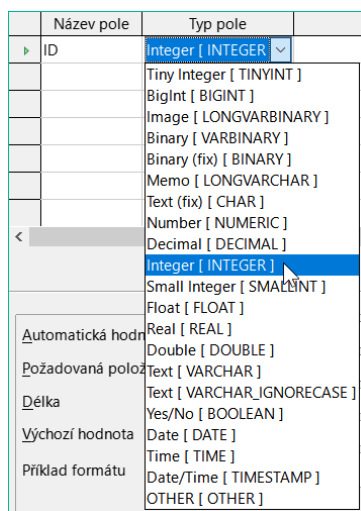


Obrázek 133: Klepneme na tlačítko Vytvořit tabulku v režimu návrhu

Spustíme editor tabulky klepnutím na **Vytvořit tabulku v režimu návrhu**.

1) Pole ID:

- a) Do prvního sloupce zadáme Název pole *ID*. Poté se klávesou Tab přesuneme do sloupce Typ pole. Případně klepneme myší na další sloupec a vybereme jej nebo stiskneme klávesu *Enter*.



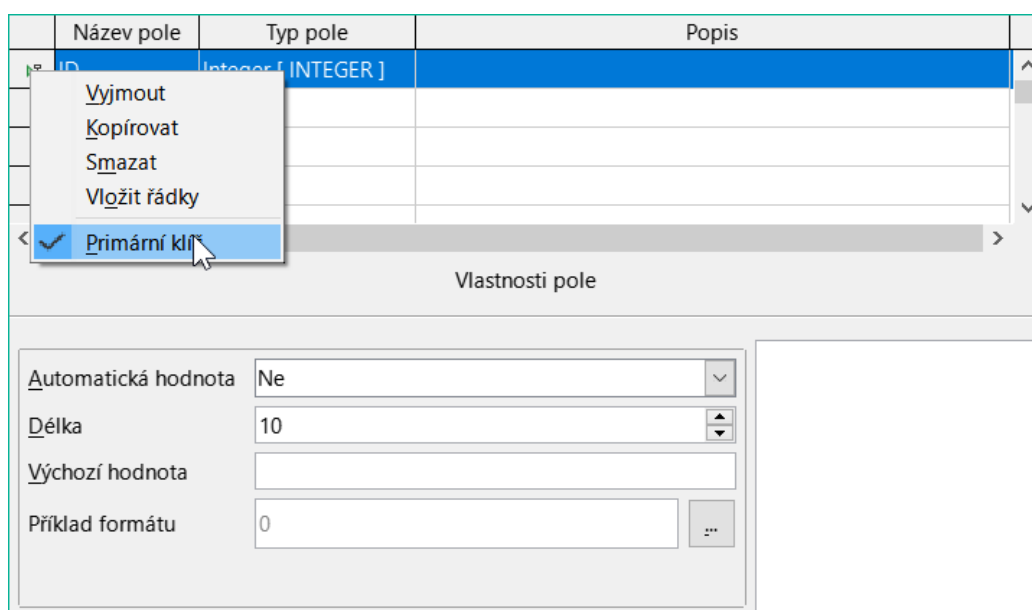
Obrázek 134: Výběr typu Integer pro pole ID

- b) Jako typ pole vybereme ze seznamu Integer [INTEGER]. Výchozí hodnota je Text [VARCHAR]. Celá čísla mohou obsahovat až 10 číslic. Kromě toho je Integer jediným typem dostupným v grafickém uživatelském rozhraní, kterému lze přiřadit automaticky se zvyšující hodnotu.

Tip

Chceme-li rychle provést výběr ze seznamu Typ pole pomocí klávesnice, zadáme znak odpovídající prvnímu písmenu výběru. Opakovaným zadáním tohoto znaku lze výběr změnit. Například zadáním D můžeme změnit výběr z Date na Date/Time nebo na Decimal.

- c) Nastavíme ID jako primární klíč klepnutím pravým tlačítkem myši na obdélník před názvem pole a výběrem možnosti Primární klíč z místní nabídky. Před ID se zobrazí symbol klíče.

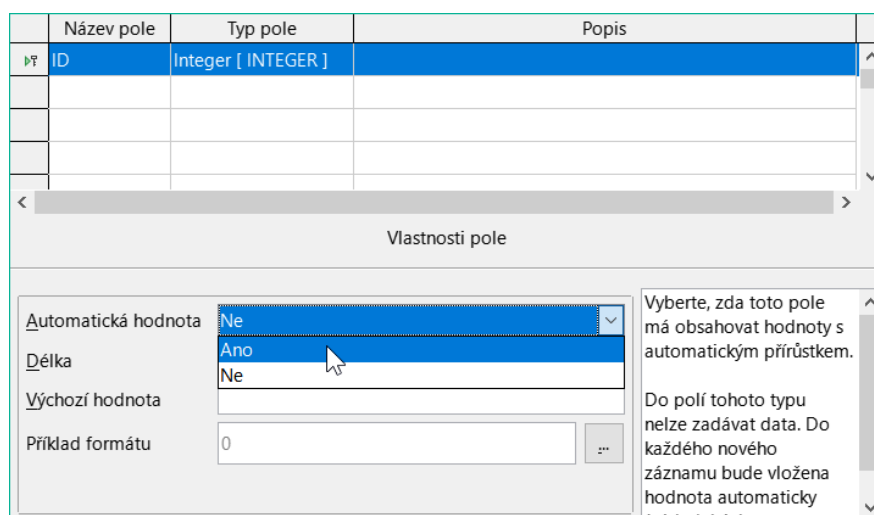


Obrázek 135: Nastavení primárního klíče pomocí místní nabídky

Poznámka

Primární klíč slouží pouze k jednomu účelu – k jednoznačné identifikaci záznamu. Proto můžeme pro toto pole použít libovolný název. V příkladu jsme použili běžně používaný název ID (identifikace).

- d) V části Vlastnosti pole pro pole ID změníme vlastnost *Automatická hodnota* z *Ne* na *Ano*. Tím se primární klíč bude automaticky zvyšovat. V interní databázi začíná počítání od 0.



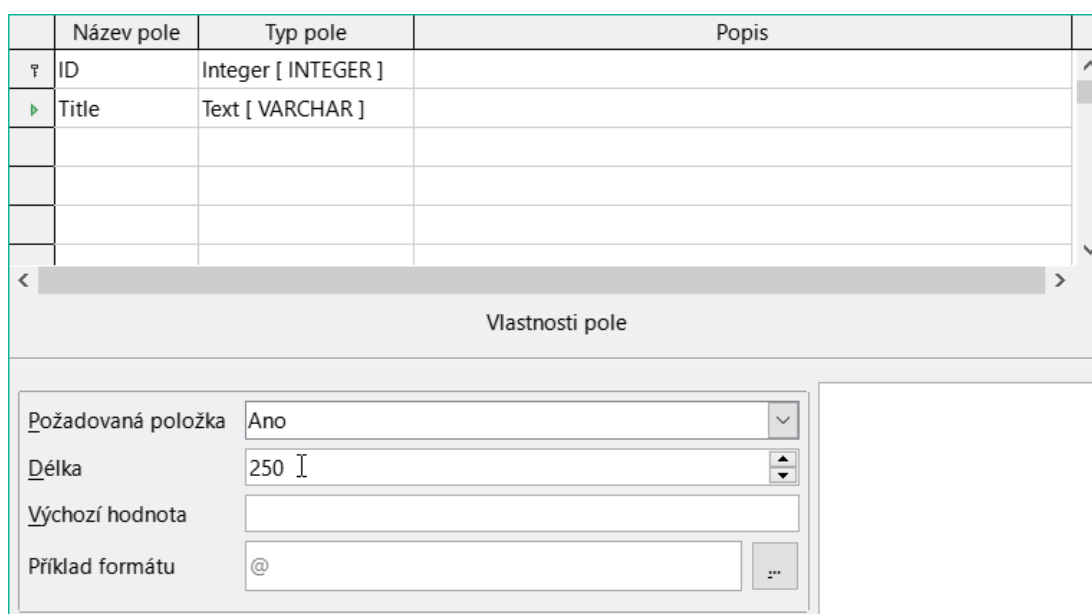
Obrázek 136: Nastavení hodnoty AutoValue na Yes pro pole ID

Automatickou hodnotu lze nastavit pouze pro jedno pole v tabulce. Výběrem možnosti **Automatická hodnota > Ano** se toto pole automaticky nastaví jako primární klíč, pokud nebyl primární klíč ještě nastaven.

2) Dalším polem je *Title*.

a) Název pole *Title* se zadává do sloupce Název pole.

b) Typ pole zde není třeba měnit, protože je již nastaven na `Text [VARCHAR]`.



Obrázek 137: Přidání pole Title

c) Ve vlastnostech pole by měla být délka pole pro názvy médií zvýšena na 250.

d) Ve vlastnostech pole změníme *Požadovaná položka* z *Ne* na *Ano*. Médium bez názvu by nemělo smysl.

	Název pole	Typ pole	Popis
⌵	ID	Integer [INTEGER]	
	Title	Text [VARCHAR]	
	Edition	Text [VARCHAR]	Nº of edition, new edition etc
▶	Pub_Year	Small Integer [SMALLINT]	Year of publication

Obrázek 138: Přidání polí Edition a Pub_Year

- 3) *Popis* může být cokoli. Tento sloupec může zůstat také prázdný. Popis slouží pouze k vysvětlení obsahu pole pro ty, kteří si chtějí definici tabulky prohlédnout později.
- 4) Pro pole Pub_Year byl zvolen typ `Small Integer [SMALLINT]`. Může obsahovat celé číslo o maximální velikosti 5 číslic. Pokud je datum zveřejnění celé číslo, je zajištěno, že nebude obsahovat žádné znaky abecedy.

	Název pole	Typ pole	Popis
⌵	ID	Integer [INTEGER]	
	Title	Text [VARCHAR]	
	Edition	Text [VARCHAR]	Nº of edition, new edition etc
	Pub_Year	Small Integer [SMALLINT]	Year of publication
	Comment	Text [VARCHAR]	
▶	Category_ID	Integer [INTEGER]	Foreign key - "Category"

Obrázek 139: Přidání polí Comment a Category_ID

- 5) Pro pole Category_ID jsme zvolili typ `Integer`. V tabulce Category by měl mít primární klíč tento typ pole, takže to, co je zde zadáno jako cizí klíč, musí mít stejný typ. To platí také pro následující cizí klíče Mediastyle_ID, Town_ID a Publisher_ID.

	Comment	Text [VARCHAR]	
	Category_ID	Integer [INTEGER]	Foreign key - "Category"
	Mediastyle_ID	Text [VARCHAR]	Foreign key - "Mediastyle"
	Town_ID	Text [VARCHAR]	Foreign key - "Town"
	Publisher_ID	Text [VARCHAR]	Foreign key - "Publisher"
▶	Price	Number [NUMERIC]	Declaration of value (€, \$, £)

Vlastnosti pole

Požadovaná položka:

Délka:

Desetinná místa:

Výchozí hodnota:

Příklad formátu:

Obrázek 140: Přidání polí Mediastyle_ID, Town_ID, Publisher_ID a Price

- 6) Pro pole Price použijeme typ `[NUMERIC]` nebo `[DECIMAL]`. Oba tyto typy polí mohou obsahovat hodnoty s desetinnou čárkou. V části Vlastnosti pole nastavíme délku 6 znaků. To by mělo být pro ceny našich médií dostačující.
 - a) Počet desetinných míst je nastaven na 2. Tím se získá maximální cena 9999,99, protože samotná desetinná čárka se do počtu nezapočítává.

b) Ve formátu není nutné uvádět znak \$, protože vzorec si s ním poradí sám.

	Publisher_ID	Text [VARCHAR]	Foreign key -"Publisher"
	Price	Number [NUMERIC]	Declaration of value (€, \$, £)
	Picture	Image [LONGVARBIN]	
▶	ISBN	Number [NUMERIC]	Max. 13-place ISBN number

Vlastnosti pole

Požadovaná položka: Ne

Délka: 13

Obrázek 141: Přidání polí Picture a ISBN

- 7) Pro pole ISBN použijeme typ [NUMERIC]. Tuto hodnotu lze nastavit přesně na správnou délku pole ISBN. Čísla ISBN jsou dlouhá 10 nebo 13 znaků. Budou uloženy jako čísla bez oddělovače. Délka je nastavena na maximálně 13 znaků. Počet desetinných míst je nastaven na nulu.
- 8) Uložíme tabulku s názvem Media.

Nyní jsme vytvořili hlavní tabulku ukázkové databáze. Všechny ostatní tabulky lze vytvořit podobným způsobem. Dbáme na to, aby typy polí a jejich vlastnosti odpovídaly tomu, co bude v těchto polích uloženo. Tím se liší od tabulkového procesoru, v němž může sloupec obsahovat směsici vlastností.

✓ Poznámka

Pořadí polí v tabulce lze měnit pouze do doby, než je tabulka poprvé uložena v grafickém uživatelském rozhraní. Při následném zadávání dat přímo do tabulky je pořadí polí pevné. Pořadí však lze v dotazech, formulářích a sestavách libovolně měnit.

Primární klíče

Pokud při návrhu tabulky není nastaven primární klíč, budeme při ukládání tabulky dotázáni, zda má být primární klíč vytvořen. To znamená, že v tabulce chybí významné pole. Bez primárního klíče nelze v grafickém uživatelském rozhraní k tabulce v databázi HSQLDB přistupovat. Toto pole se obvykle jmenuje ID a má typ INTEGER s *Automatickou hodnotou > Ano*, který hodnotu pole automaticky zvýší. Klepnutím na **Ano** v dialogovém okně primárního klíče se automaticky vytvoří pole primárního klíče. Klepnutím na tlačítko Ne nebo Zrušit v dialogovém okně primárního klíče můžeme označit existující pole jako primární klíč klepnutím pravým tlačítkem myši na zelenou šipku vlevo od příslušného pole.

Jako primární klíč můžeme použít i kombinaci polí. Pole musí být deklarována jako primární klíč společně (držíme stisknutou klávesu Control nebo Shift). Pak klepnutím pravým tlačítkem myši vytvoříme kombinaci všech zvýrazněných polí jako primární klíč.

Pokud jsou do této tabulky importovány informace z jiných tabulek (například z databáze adres s externě uloženými poštovními směrovacími čísly a obcemi), musí být zahrnuto pole se stejným datovým typem jako primární klíč jiné tabulky. Předpokládejme, že tabulka Postcode má jako primární klíč pole ID s typem Tiny Integer. V tabulce Address pak musí být pole Postcode_ID s typem Tiny Integer. Do tabulky Address se vždy zadává číslo, které slouží jako primární klíč pro dané místo v tabulce Postcode. To znamená, že tabulka Address má nyní kromě vlastního primárního klíče také cizí klíč.

Pravidlem pro pojmenování polí v tabulce je, že žádná dvě pole nemohou mít stejný název. Proto se v tabulce Address nemůže vyskytovat druhé pole s názvem ID jako cizí klíč.

Typ pole lze měnit pouze v omezeném rozsahu. Zvýšení vlastnosti (délka textového pole, větší velikost v čísle) je vždy povoleno, protože všechny již zadané hodnoty budou odpovídat novým podmínkám. Snížení vlastnosti pravděpodobně způsobí problémy a může vést ke ztrátě dat.

Časová pole v tabulkách nelze vytvořit tak, aby obsahovala zlomky sekundy. K tomu potřebujeme pole Timestamp. Grafické uživatelské rozhraní však umožňuje vytvořit pouze pole Timestamp s datem, hodinou, minutou a sekundou. Toto pole budeme muset následně upravit pomocí **Nástroje > SQL..**

```
ALTER TABLE "Název_tabulky" ALTER COLUMN "Název_pole" TIMESTAMP(6)
```

Parametr „6“ umožňuje ukládat do pole Timestamp zlomky sekundy.

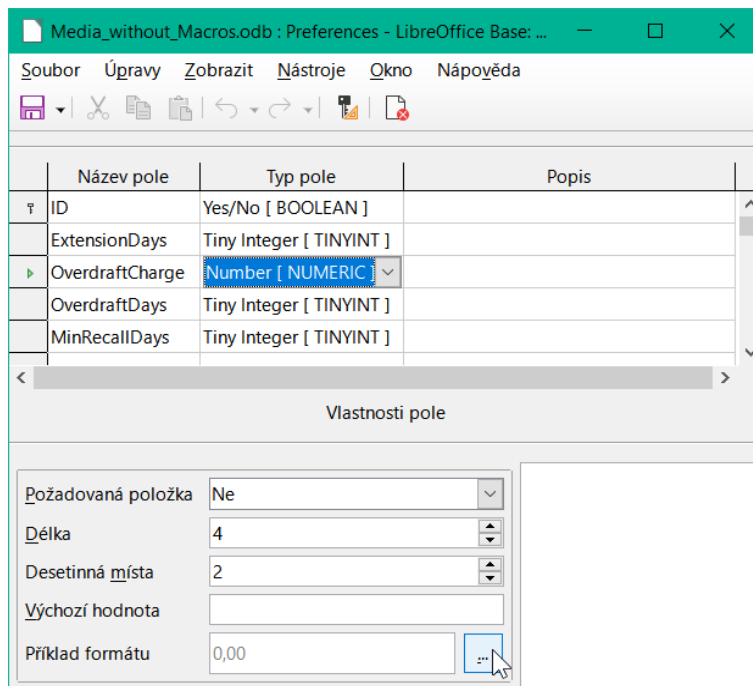
Formátování polí

Formátování představuje uživateli hodnoty v databázi a umožňuje zadávat hodnoty v závislosti na konvencích zadávání obvyklých v dané zemi. Bez formátování se desetinná místa oddělují tečkou, zatímco většina evropských zemí používá čárku (4.21 místo 4,21). Hodnoty data jsou uvedeny ve tvaru 2014-12-22. Při nastavování formátování je třeba brát ohled na místní standardy.

Formátování poskytuje pouze reprezentaci obsahu. Datum reprezentované dvoumístným číslem roku je stále uloženo jako čtyřmístný rok. Pokud je vytvořeno pole pro číslo se dvěma desetinnými místy, jako je například poplatek za prodlení (tzv. kontokorent) v následujícím příkladu, je číslo uloženo se dvěma desetinnými místy, i když je chybně nastaveno jejich nezobrazování. Do pole formátovaného bez desetinných míst lze zadat číslo se dvěma desetinnými místy. Zdá se, že desetinná část při zadávání zmizí, ale je viditelná, pokud se formátování obejde.

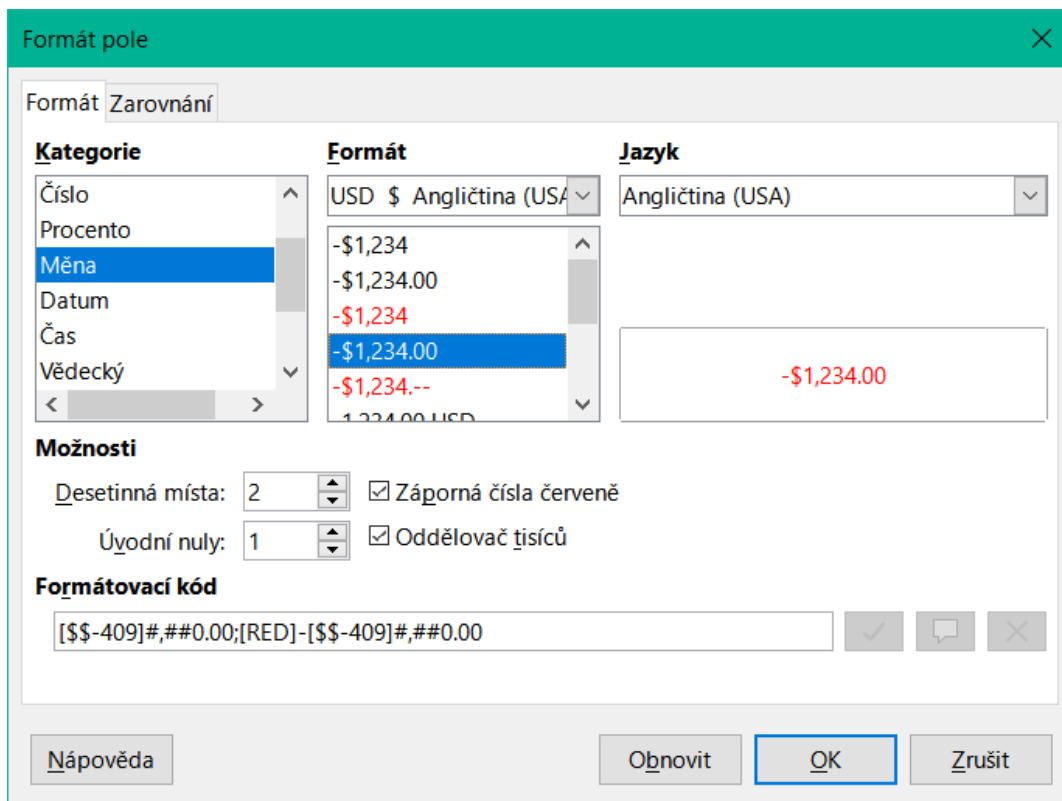
Chceme-li zobrazit pouze čas, nikoli datum, můžeme formuláře naformátovat tak, aby zobrazovaly pouze potřebné informace a zbytek pole Timestamp skryly. V případě ukládání času například ze stopky lze minuty, sekundy a zlomky sekundy v časovém razítku zobrazit pomocí formátu MM:SS.00 na displeji. Formát bez data lze nastavit později ve Formulářích pomocí formátovaného pole, nikoli však přímo do pole Timestamp.

Formátování polí při vytváření tabulky nebo následně prostřednictvím vlastností polí se provádí v samostatném dialogovém okně:



Obrázek 142: Klepnutím na tlačítko vedle příkladu Formát se zobrazí dialogové okno Formát pole

Tlačítko vedle **Vlastnosti pole > Příklad formátu** otevře dialogové okno pro změnu formátu.



Obrázek 143: Dialogové okno Formát pole

Při vytváření měnových polí dbáme na to, aby číselné pole mělo nastavena dvě desetinná místa. Při vytváření tabulky v grafickém uživatelském rozhraní lze provést formátování, aby se při zadávání použila příslušná měna. To má vliv pouze na vstup do tabulky a na dotazy, které

používají vstupní hodnotu bez přepočtu. Ve formulářích musí být označení měny samostatně formátováno.

✓ Poznámka

Program Base ukládá formátování tabulek při vytváření polí nebo během zadávání dat, pokud jsou formáty sloupců upraveny klepnutím pravým tlačítkem myši na záhlaví sloupců. Šířky sloupců na vstupní obrazovce se ukládají i při úpravě během zadávání dat.

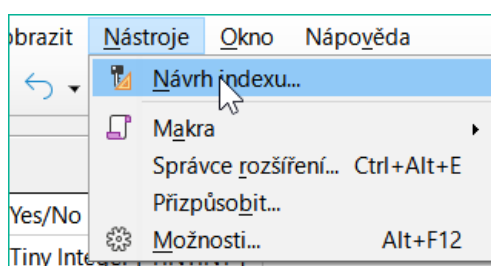
V dotazech, formulářích nebo sestavách lze formátování zobrazení upravit podle potřeby.

V případě polí, která mají obsahovat procenta, je třeba vzít na vědomí, že 1 % musí být uloženo jako 0,01. Zápis procent proto vyžaduje alespoň dvě desetinná místa. Pokud je třeba uložit zlomková procenta, například 3,45, vyžaduje uložená číselná hodnota čtyři desetinná místa.

Vytvoření indexu

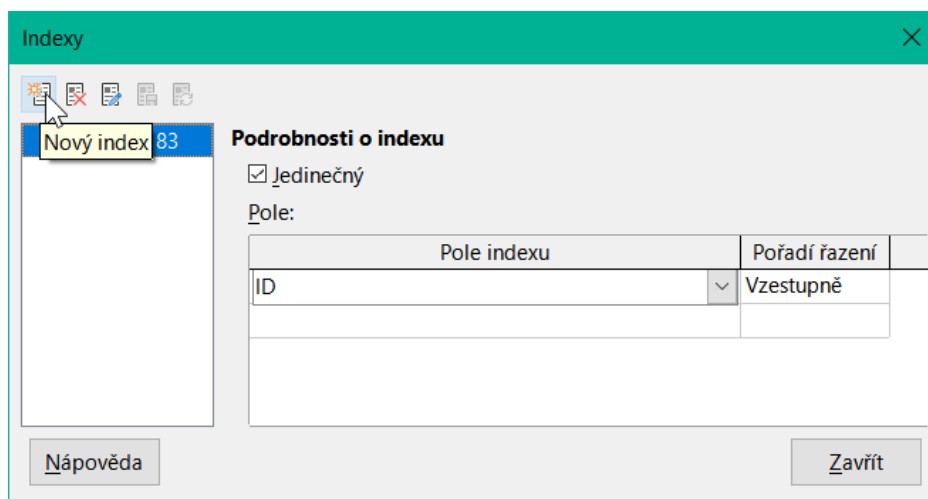
Někdy je užitečné kromě primárního klíče indexovat další pole nebo kombinaci dalších polí. Index urychluje vyhledávání a lze jej také použít k zabránění duplicitním záznamům.

Každý index má definované pořadí řazení. Pokud je tabulka zobrazena bez třídění, bude seřazení probíhat podle obsahu polí uvedených v indexu.



Obrázek 144: Výběr **Nástroje > Návrh indexu**

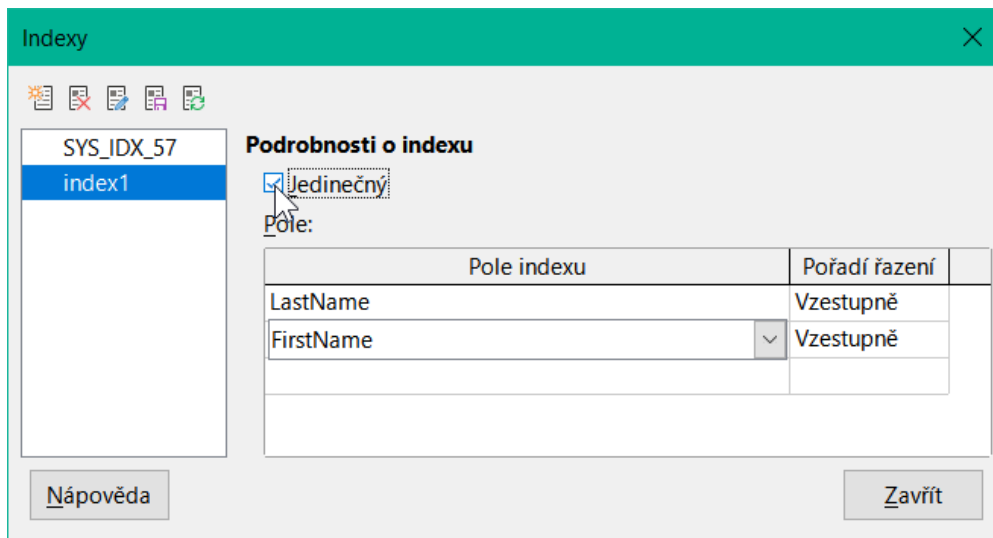
Otevřeme tabulku pro úpravy klepnutím pravým tlačítkem myši a použitím místní nabídky. Poté můžeme přistoupit k vytváření indexů pomocí **Nástroje > Návrh indexu**.



Obrázek 145: Vytvoření nového indexu

V dialogovém okně Indexy (obrázek 145) klepneme na **Nový index** a vytvoříme kromě primárního klíče také index.

Novému indexu je automaticky přiřazen název index1. **Pole indexu** určuje, které pole nebo která pole mají být použita pro tento index. Současně můžeme zvolit Pořadí řazení.



Obrázek 146: Index je definován jako Unique

V zásadě lze index vytvořit i z polí tabulky, která neobsahují jedinečné hodnoty. Na obrázku 146 byl však detail Index **Jedinečný** zaškrtnut, takže pole LastName spolu s polem FirstName může mít pouze položky, které se v této kombinaci ještě nevyskytují. Takže například jména Robert Miller a Robert Maier jsou možné, stejně tak Robert Miller a Eva Millerová.

Pokud je index vytvořen pouze pro jedno pole, vztahuje se jedinečnost pouze na toto pole. Takovým indexem je obvykle primární klíč. V tomto poli se může každá hodnota vyskytnout pouze jednou. V případě primárních klíčů navíc pole nesmí mít za žádných okolností hodnotu NULL.

Výjimečnou okolností pro jedinečný index je situace, kdy v poli není žádný záznam (pole je NULL). Protože NULL může mít libovolnou hodnotu, index používající dvě pole může mít v jednom z polí vždy opakovaně stejnou položku, pokud v druhém poli není žádná položka.

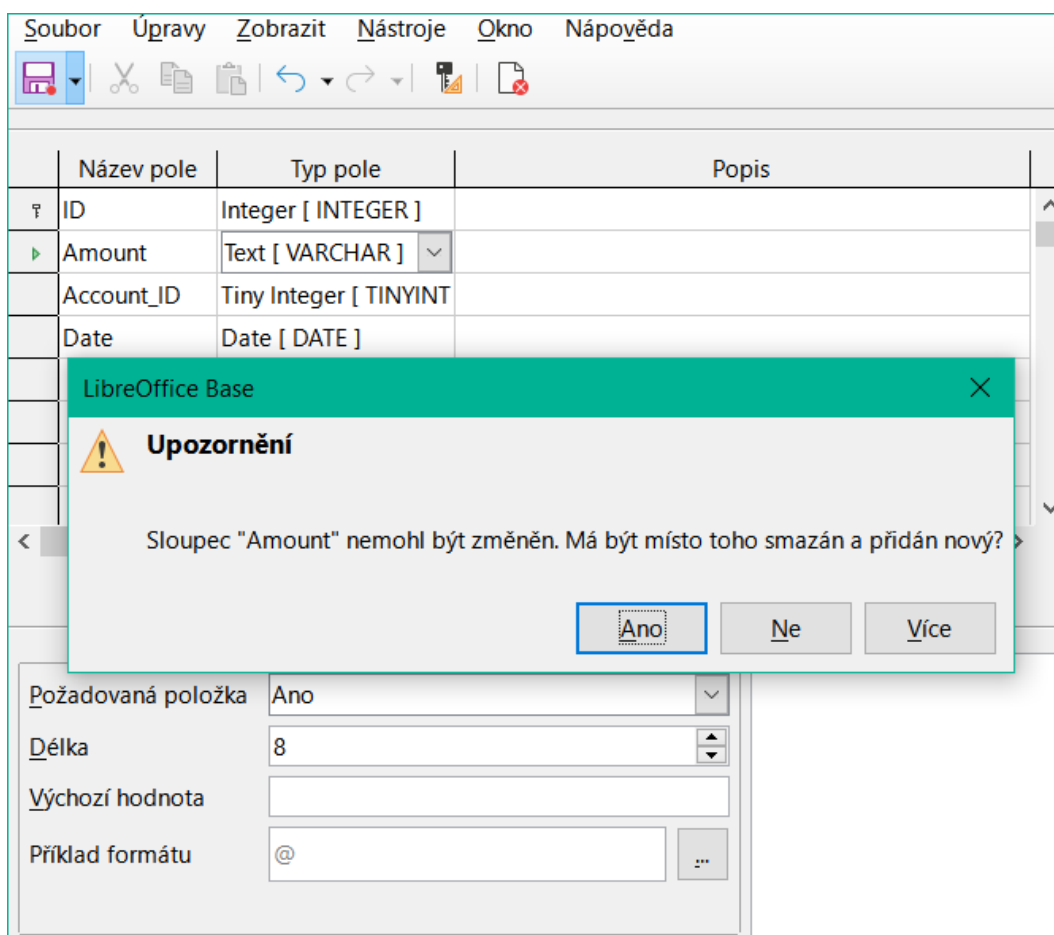
✓ Poznámka

NULL se v databázích používá pro označení prázdné buňky, která nic neobsahuje. Pomocí pole NULL nelze provést žádný výpočet. To je rozdíl oproti tabulkovým procesorům, kde prázdná pole automaticky obsahují hodnotu 0 (nula).

Příklad: V databázi médií se při výpůjčce zadává číslo média a datum výpůjčky. Při vrácení položky se zadá datum vrácení. Teoreticky by index využívající pole Media_ID a ReturnDate mohl snadno zabránit opakovanému půjčování stejné položky, aniž by bylo zaznamenáno datum vrácení. To však nebude fungovat, protože datum vrácení nemá zpočátku žádnou hodnotu. Index zabránil tomu, aby položka byla dvakrát označena jako vrácená se stejným datem, ale neudělá nic jiného.

Problémy při úpravě tabulek

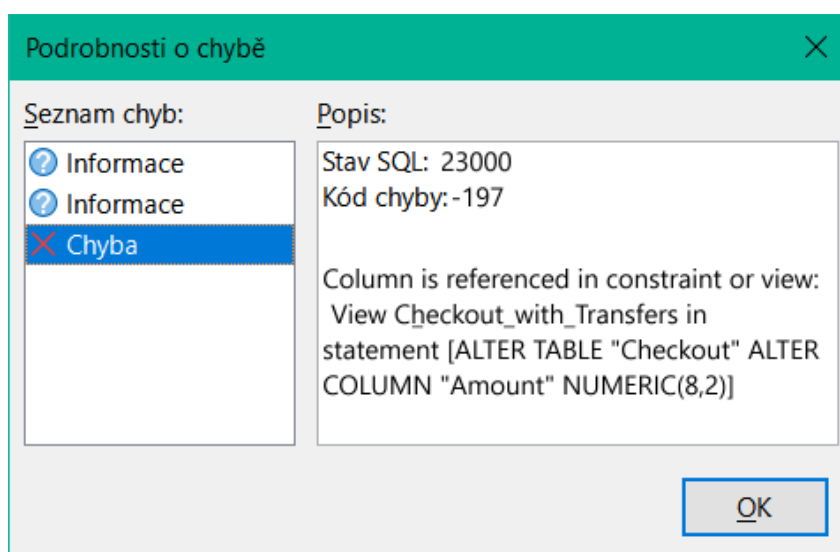
Nejllepší je vytvořit tabulky se všemi požadovanými nastaveními, aby nebylo nutné později měnit jejich konfiguraci. Při pozdější změně vlastností polí (název pole, povinný údaj apod.) může dojít k chybovým hlášením, která nejsou způsobena grafickým uživatelským rozhraním, ale pokusem o nežádoucí změnu základní databáze.



Obrázek 147: Chyba při pokusu o změnu požadované položky na Ano

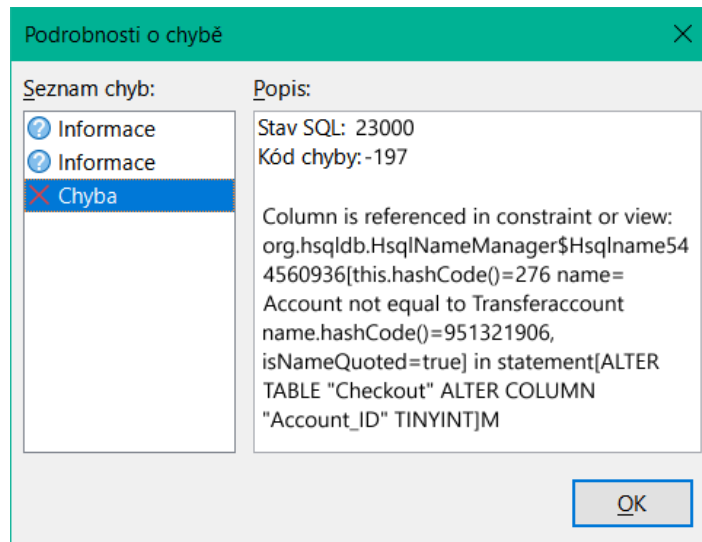
V tomto případě se pole Amount vynuluje na hodnotu Vstup požadován=ano. Výstražný symbol nás upozorňuje, že tato změna může vést ke ztrátě dat. Jednoduchá změna není možná, protože již mohou existovat záznamy, které v tomto poli nemají žádný záznam.

Klepnutí na **Ano** vede k dalšímu chybovému hlášení, protože struktura databáze neumožňuje toto pole odstranit. Klepnutím na Ne se celá operace zruší. Volba Další možnosti je uvedena vždy, když je to možné, abychom získali další informace o řešení problému.



Obrázek 148: Dialogové okno Podrobnosti o chybě, když nelze změnit pole Amount

Chybové hlášení **Sloupec** je odkazován v omezení nebo zobrazení znamená: Na sloupec s názvem pole „Amount“ se odkazuje v jiné části databáze. Může se jednat o definici omezení nebo zobrazení tabulky, které bylo vytvořeno některým uživatelem po vytvoření samotné tabulky. Z výše uvedeného obrázku vyplývá, že název omezení nebo zobrazení je **View_Checkout_with_Transfers**. Uživateli je tak jasné, kde v databázi je třeba provést změny. Například kód SQL pro zobrazení by mohl být nejprve uložen jako dotaz a poté by mohlo být zobrazení zničeno a proveden nový pokus o znovuvytvoření pole.



Obrázek 149: Dialogové okno Podrobnosti o chybě při nemožnosti změnit pole **Account_ID**

V tomto případě nás název omezení *Account not equal to Transferaccount* vede k definici tohoto omezení. Podmínkou je, že hodnota v poli **Account_ID** nesmí být stejná jako hodnota v poli **TransferAccount_ID**. Sloupec lze změnit pouze tehdy, pokud je tato podmínka odstraněna.

Pokud dojde k další chybě, je to s největší pravděpodobností způsobeno tím, že příslušné pole je propojeno s polem v jiné tabulce pomocí definovaného vztahu. V takovém případě je třeba před provedením změny přerušit vazbu pomocí **Nástroje > Relace**.

Omezení grafického návrhu tabulky

Pořadí polí v tabulce nelze po uložení databáze měnit. Zobrazení jiné sekvence vyžaduje dotaz.

Pole na konkrétní pozici v tabulce lze vložit pouze zadáním přímých příkazů SQL. Touto metodou však nelze přesouvat již vytvořená pole.

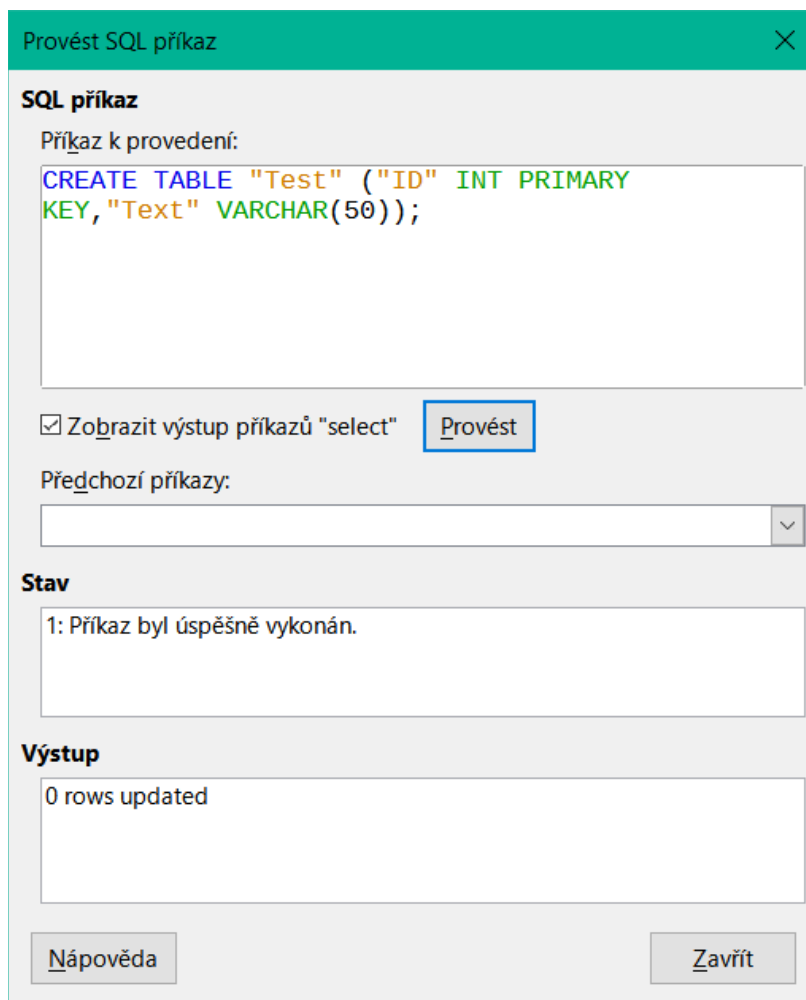
Na začátku je třeba nastavit vlastnosti tabulek: například která pole nesmí být NULL a která musí obsahovat standardní hodnotu (Default). Tyto vlastnosti nelze následně měnit pomocí grafického uživatelského rozhraní.

Výchozí hodnoty, které můžeme nastavit v grafickém uživatelském rozhraní, nejsou tak silné jako možné výchozí hodnoty v samotné databázi. Například nelze definovat výchozí hodnotu pole s datem jako datum zadání. To je možné pouze pomocí přímo zadaných příkazů SQL.

Přímé zadávání SQL příkazů

Chceme-li zadat příkazy SQL přímo, přejdeme na **Nástroje > SQL**.

Příkazy se zadávají v horní části okna (obrázek 150). V dolní části (Status) se zobrazuje úspěch nebo důvod neúspěchu. Výsledky dotazů lze zobrazit v poli Výstup, pokud je toto políčko zaškrtnuto.



Obrázek 150: Dialog pro přímé zadávání příkazů SQL

Přehled možných příkazů pro vestavěný engine HSQLDB najdeme na adrese <http://www.hsqldb.org/doc/1.8/guide/ch09.html>. Obsah je popsán v následujících kapitolách. Některé příkazy mají smysl pouze při práci s externí databází HSQLDB (Zadat uživatele atd.). V případě potřeby se jimi zabývá kapitola „Práce s externí HSQLDB“ v dodatku této příručky.

✓ Poznámka

LibreOffice je založen na verzi 1.8.0 HSQLDB. Aktuálně dostupná verze serveru je 2.5. Funkce nové verze jsou rozsáhlejší. Informace o nich můžeme nalézt na adrese <http://hsqldb.org/web/hsqldbDocsFrame.html>. Popis verze 1.8 je nyní na adrese <http://www.hsqldb.org/doc/1.8/guide/>. Další popis je uveden v instalačních balíčcích pro HSQLDB, které si můžeme stáhnout ze stránky <http://sourceforge.net/projects/hsqldb/files/hsqldb/>.

Vytvoření tabulky

Jednoduchý příkaz pro vytvoření použitelné tabulky je:

```
CREATE TABLE "Test" ("ID" INT PRIMARY KEY, "Text" VARCHAR(50));
```

Rozdělení tohoto příkazu:

```
CREATE TABLE "Test": Vytvoří tabulku s názvem "Test".
```

(): zadané názvy polí, typy polí a možnosti se vloží do závorek.

"ID" INT PRIMARY KEY, : Název pole ID s číselným typem integer jako primárním klíčem;

"Text" VARCHAR(50): název pole Text s proměnnou délkou textu typu text a velikostí textu omezenou na 50 znaků.

Parametry příkazu CREATE:

```
CREATE [MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT] TABLE  
"Table name" ( <Field definition> [, ...] [,  
<Constraint Definition>...] ) [ON COMMIT {DELETE | PRESERVE} ROWS];
```

[MEMORY | CACHED | [GLOBAL] TEMPORARY | TEMP | TEXT]:

Určuje umístění nově vytvořené tabulky. Výchozí nastavení je MEMORY: HSQLDB vytvoří všechny tabulky v základní paměti. Toto nastavení se vztahuje také na tabulky, které LibreOffice Base zapisuje do vložené databáze. Další možností by bylo zapsat tabulky na pevný disk a používat paměť pouze jako vyrovnávací paměti pro přístup na pevný disk (CACHED).

✓ Poznámka

```
CREATE TEXT TABLE "Text" ("ID" INT PRIMARY KEY, "Text"  
VARCHAR(50));
```

Vytvoří textovou tabulku v HSQLDB. Nyní musí být propojen s externím textovým souborem (například se souborem *.csv): `SET TABLE "Text" SOURCE "Text.csv";`

Soubor *Text.csv* musí samozřejmě obsahovat odpovídající pole ve správném pořadí. Při vytváření odkazu lze vybrat různé další možnosti. Podrobnosti nalezneme na http://www.hsqldb.org/doc/1.8/guide/guide.html#set_table_source-section.

Textové tabulky nejsou chráněny proti zápisu jinými programy. Může se tedy stát, že jiný program nebo uživatel změní tabulku právě ve chvíli, kdy k ní přistupuje databáze Base. Textové tabulky se používají především k výměně dat mezi různými programy.

Tabulky ve formátu TEXT (například CSV) nelze zapisovat do interních databází, které jsou nastaveny čistě v MEMORY, zatímco program Base nemá přístup k tabulkám TEMPORARY nebo TEMP. Příkazy SQL se v tomto případě provedou, ale tabulky se pomocí grafického uživatelského rozhraní nezobrazí (a nelze je tedy odstranit) a data zadaná prostřednictvím SQL nejsou pro dotazovací modul grafického uživatelského rozhraní rovněž viditelná, pokud není zabráněno automatickému odstranění obsahu po konečném odevzdání (pomocí ON COMMIT PRESERVE ROWS). Jakýkoli požadavek v tomto případě zobrazí tabulku bez jakéhokoli obsahu.

Tabulky sestavené přímo pomocí jazyka SQL se nezobrazují okamžitě. Musíme buď použít **Zobrazení > Obnovit tabulky** nebo databázi jednoduše zavřít a poté ji znovu otevřít.

<Field definition>:

```
"Field name" Data type [(Number of characters[,Decimal places])]  
[ {DEFAULT "Default value" | GENERATED BY DEFAULT AS IDENTITY (START  
WITH <n>[, INCREMENT BY <m>)} ] | [[NOT] NULL] [IDENTITY] [PRIMARY  
KEY]
```

Umožňuje zahrnout do definice pole výchozí hodnoty.

Do textových polí můžeme zadávat text v jednoduchých uvozovkách nebo NULL. Jediná povolená funkce SQL je CURRENT_USER. To má smysl pouze v případě, že se HSQLDB používá jako externí databáze serveru s několika uživateli.

U polí pro datum a čas lze datum, čas nebo jejich kombinaci zadat v jednoduchých uvozovkách nebo jako NULL. Je třeba zajistit, aby datum odpovídalo americkým konvencím (rrrr-mm-dd), aby čas měl formát hh:mm:ss a aby kombinovaná hodnota data a času měla formát rrrr-mm-dd hh:mm:ss.

Povolené funkce SQL:

pro aktuální datum – *CURRENT_DATE, TODAY, CURDATE()*

pro aktuální čas – *CURRENT_TIME, NOW, CURTIME()*

pro aktuální časové razítko dat – *CURRENT_TIMESTAMP, NOW.*

Pro logická pole (ano/ne) lze zadat výrazy *FALSE, TRUE, NULL*. Ty musí být zadány bez jednoduchých uvozovek.

U číselných polí je možné zadat libovolné platné číslo v daném rozsahu nebo NULL. Pokud zadáváme NULL, nepoužíváme uvozovky. Při zadávání desetinných míst dbáme na to, aby desetinný oddělovač byla tečka, nikoli čárka. (Někteří anglicky mluvící lidé používají čárku jako desetinnou tečku.)

Pro binární pole (obrázky atd.) je možné použít jakýkoli platný hexadecimální řetězec v jednoduchých uvozovkách nebo NULL. Příklad hexadecimálního řetězce je: 0004ff, což představuje 3 bajty: nejprve 0, pak 4 a nakonec 255 (0xff). Vzhledem k tomu, že binární pole je v praxi nutné zadávat pouze pro obrázky, je třeba znát binární kód obrázku, který má sloužit jako výchozí.



Poznámka

Šestnáctková soustava: Čísla jsou založena na 16. Smíšený systém složený z čísel 0 až 9 a písmen a až f poskytuje 16 možných číslic pro každý sloupec. Při použití dvou sloupců můžeme získat $16 \cdot 16 = 256$ možných hodnot. To odpovídá 1 Bajtu (28).

NOT NULL: Hodnota pole nemůže být NULL. Tuto podmínku lze uvést pouze v definici pole.

Příklad:

```
CREATE TABLE "Test" ("ID" INT GENERATED BY DEFAULT AS IDENTITY (START WITH 10),  
"Name" VARCHAR(50) NOT NULL, "Date" DATE DEFAULT TODAY);
```

Vytvoří se tabulka s názvem Test. Klíčové pole ID je definováno jako Automatická hodnota s hodnotami začínajícími na 10. Vstupní pole Name je textové pole s maximální velikostí 50 znaků. Nesmí být prázdné. Nakonec máme pole Date, které ve výchozím nastavení ukládá aktuální datum, pokud není zadáno jiné datum. Tato výchozí hodnota je platná pouze při vytvoření nového záznamu. Odstraněním data v existujícím záznamu zůstane pole prázdné.

<Constraint definition>:

```
[CONSTRAINT "Name"]
```

```
UNIQUE ( "Field_name 1" [, "Field_name 2"...] ) |  
PRIMARY KEY ( "Field_name 1" [, "Field_name 2"...] ) |  
FOREIGN KEY ( "Field_name 1" [, "Field_name 2"...] )  
REFERENCES "other_table_name" ( "Field_name_1" [, "Field_name 2"...])  
[ON {DELETE | UPDATE}  
{CASCADE | SET DEFAULT | SET NULL}] |  
CHECK(<Search_condition>)
```

Omezení definují podmínky, které musí být při zadávání dat splněny. Omezení lze pojmenovat.

UNIQUE ("Field_name"): hodnota pole musí být jedinečná v rámci daného pole

PRIMARY KEY ("Field_name"): hodnota pole musí být jedinečná a nesmí být NULL (primární

klíč)

```
FOREIGN KEY ("Field_name") REFERENCES <"other_table_name">
```

("Field_name"): Uvedená pole této tabulky jsou propojena s poli jiné tabulky. Hodnota pole musí být testována na referenční integritu jako cizí klíč, to znamená, že pokud je zde zadána hodnota, musí existovat odpovídající primární klíč v jiné tabulce.

[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}]: V případě cizího klíče určuje, co se má stát, pokud je například cizí záznam odstraněn. V tabulce výpůjček pro knihovnu nemá smysl uvádět číslo uživatele, který již neexistuje. Příslušný záznam musí být upraven tak, aby vztah mezi tabulkami zůstal platný. Obvykle se záznam jednoduše odstraní. K tomu dojde, pokud vybereme možnost ON DELETE CASCADE.

CHECK(<Search_condition>): Formulováno jako podmínka WHERE, ale pouze pro aktuální záznam.

```
CREATE TABLE "Time_measurement" ("ID" INT PRIMARY KEY, "Start_time" TIME, "End_time" TIME, CHECK ("Start_time" <= "End_time"));
```

Podmínka CHECK vylučuje zadání dřívější hodnoty času ukončení než času zahájení. Při pokusu o takovýto zápis se zobrazí chybová zpráva podobná:

```
Zkontrolujeme porušení omezení tabulky SYS_CT_357: Time_measurement ...
```

Omezení pro vyhledávání je přiřazen název, který není příliš informativní. Místo toho by název mohl být definován v definici tabulky:

```
CREATE TABLE "Time_measurement" ("ID" INT PRIMARY KEY, "Start_time" TIME, "End_time" TIME, CONSTRAINT "Start_time<=End_time" CHECK ("Start_time" <= "End_time"));
```

Tím se získá poněkud jasnější chybové hlášení, protože se zobrazí název příslušného omezení.

Při vytváření relací mezi tabulkami nebo indexování konkrétních polí je třeba dodržovat omezení. Omezení se vytvářejí pomocí podmínky „CHECK“, v grafickém uživatelském rozhraní pomocí **Nástroje > Relace** a také v **indexech vytvořených v návrhu tabulky v části Nástroje > Návrh indexu**.

[ON COMMIT {DELETE | PRESERVE} ROWS]:

Obsah tabulek typu TEMPORARY nebo TEMP se standardně vymaže po ukončení práce s konkrétním záznamem (ON COMMIT DELETE ROWS). To umožňuje vytvářet dočasné záznamy, které obsahují informace pro další akce, jež mají být provedeny současně.

Pokud chceme, aby tabulka tohoto typu obsahovala data dostupná po celou relaci (od otevření databáze až po její uzavření), zvolíme možnost ON COMMIT PRESERVE ROWS.

Úprava tabulky

Někdy můžeme chtít vložit další pole na určitou pozici v tabulce. Předpokládejme, že máme tabulku s názvem *Addresses* s poli ID, Name, Street a podobně. Uvědomujeme si, že by možná bylo rozumné rozlišovat křestní jména a příjmení.

```
ALTER TABLE "Addresses" ADD "FirstName" VARCHAR(25) BEFORE "Name";
```

ALTER TABLE "Addresses": Změní tabulku s názvem *Addresses*.

ADD "FirstName" VARCHAR(25): vloží pole *FirstName* s délkou 25 znaků.

BEFORE "Name": před polem *Name*.

Možnost zadat pozici dalších polí po vytvoření tabulky není v grafickém uživatelském rozhraní k dispozici.

```
ALTER TABLE "Table_name" ADD [COLUMN] <Field_definition> [BEFORE "already_existing_field_name"];
```

Dodatečné označení COLUMN není nutné v případech, kdy nejsou k dispozici žádné alternativní možnosti.

```
ALTER TABLE "Table_name" DROP [COLUMN] "Field_name";
```

Pole Field_name je vymazáno z tabulky Table_name. K tomu však nedojde, pokud je pole zapojeno do zobrazení nebo jako cizí klíč v jiné tabulce.

```
ALTER TABLE "Table_name" ALTER COLUMN "Field_name" RENAME TO  
"New_field_name"
```

Změní název pole.

```
ALTER TABLE "Table_name" ALTER COLUMN "Field_name" SET DEFAULT  
<Standard value>;
```

Nastaví konkrétní výchozí hodnotu pole. NULL odstraní existující výchozí hodnotu.

```
ALTER TABLE "Table_name" ALTER COLUMN "Field_name" SET [NOT] NULL
```

Nastaví nebo odstraní podmínku NOT NULL pro pole.

```
ALTER TABLE "Table_name" ALTER COLUMN <Field definition>;
```

Definice pole odpovídá definici z Vytvoření tabulky s následujícími omezeními:

- Pole musí být již polem primárního klíče, aby bylo možné akceptovat vlastnost IDENTITY. IDENTITY znamená, že pole má vlastnost Automatická hodnota. To je možné pouze pro pole INTEGER nebo BIGINT. Popisy těchto typů polí nalezneme v dodatku k této příručce.
- Pokud pole již má vlastnost IDENTITY, ale v definici pole se neopakuje, stávající vlastnost IDENTITY se odstraní.
- Výchozí hodnotou se stane hodnota uvedená v definici nového pole. Pokud je definice výchozí hodnoty ponechána prázdná, jsou všechny již definované výchozí hodnoty odstraněny.
- Vlastnost NOT NULL pokračuje v nové definici, pokud není definována jinak. To je rozdíl od výchozí hodnoty.
- V některých případech, v závislosti na typu změny, musí být tabulka prázdná, aby mohlo dojít ke změně. Ve všech případech se změna projeví pouze tehdy, pokud je v zásadě možná (například změna z NOT NULL na NULL) a všechny stávající hodnoty lze převést (například změna z TINYINT na INTEGER).

```
ALTER TABLE "Table_name" ALTER COLUMN "Field_name" RESTART WITH  
<New_field_value>
```

Tento příkaz se používá výhradně pro pole IDENTITY. Určuje další hodnotu pole s nastavenou funkcí Automatická hodnota. Lze ji použít například v případě, kdy je databáze nejprve používána s testovacími daty a následně je vybavena reálnými daty. To vyžaduje odstranění obsahu tabulek a nastavení nové hodnoty pole, například "1".

```
ALTER TABLE "Table_name"  
ADD [CONSTRAINT "Condition_name"] CHECK (<Search_condition>;
```

Tím se přidá podmínka vyhledávání zavedená slovem CHECK. Taková podmínka se nebude vztahovat zpětně na existující záznamy, ale bude se vztahovat na všechny následné změny a nově vložené záznamy. Pokud není název omezení definován, bude přiřazen automaticky.

Příklad:

```
ALTER TABLE "Loan" ADD CHECK (IFNULL("Return_Date", "Loan_Date")>="Loan_Date")
```

Tabulka Loan musí být chráněna před vstupními chybami. Musíme například zabránit tomu, aby bylo datum vrácení uvedeno dříve, než je datum výpůjčky. Pokud se tato chyba vyskytne během procesu vrácení, zobrazí se chybová zpráva `Kontrola porušení omezení ...`.

```
ALTER TABLE "Table_name"  
ADD [CONSTRAINT "Constraint_name"] UNIQUE ("Field_name1",  
"Field_name2"...);
```

Zde je přidána podmínka, která vynucuje, aby pojmenovaná pole měla v každém záznamu jiné hodnoty. Pokud je pojmenováno více polí, vztahuje se tato podmínka spíše na kombinaci než na jednotlivá pole. `NULL` se zde nepočítá. Pole tedy může mít opakovaně stejnou hodnotu, aniž by to způsobovalo problémy, pokud je jiné pole v každém ze záznamů `NULL`.

Tento příkaz nebude fungovat, pokud pro stejnou kombinaci polí již existuje podmínka `UNIQUE`.

```
ALTER TABLE "Table_name"  
ADD [CONSTRAINT "Constraint_name"] PRIMARY KEY ("Field_name1",  
"Field_name2"...);
```

Přidá do tabulky primární klíč, případně s omezením. Syntaxe omezení je stejná jako při vytváření tabulky.

```
ALTER TABLE "Table_name" ADD [CONSTRAINT "Constraint_name"] FOREIGN KEY  
("Field_name1", "Field_name2"...)  
REFERENCES "Table_name_of_another_table" ("Field_name1_other_table",  
"Field_name2_other_table"...)  
[ON {DELETE | UPDATE} {CASCADE | SET DEFAULT | SET NULL}];
```

Tím se do tabulky přidá cizí klíč (`FOREIGN KEY`). Syntaxe je stejná jako při vytváření tabulky.

Operace se ukončí chybovým hlášením, pokud některá hodnota v tabulce nemá odpovídající hodnotu v tabulce obsahující daný primární klíč.

Příklad: Tabulky `Name` a `Address` mají být propojeny. Tabulka `Name` obsahuje pole s názvem `Address_ID`. Hodnota tohoto pole by měla být spojena s polem `ID` v tabulce `Address`. Pokud se v poli `Address_ID` nachází hodnota **1**, ale ne v poli `ID` tabulky `Address`, odkaz nebude fungovat. Nebude to fungovat ani v případě, že obě pole jsou různých typů.

```
ALTER TABLE "Table_name" DROP CONSTRAINT "Constraint_name";
```

Tento příkaz odstraní z tabulky pojmenované omezení (`UNIQUE`, `CHECK`, `FOREIGN KEY`).

```
ALTER TABLE "Table_name" RENAME TO "new_table_name";
```

Nakonec tento příkaz změní pouze název tabulky.

Poznámka

Když změníme tabulku pomocí jazyka SQL, změna se projeví v databázi, ale nemusí být nutně viditelná nebo účinná v grafickém uživatelském rozhraní. Po zavření a opětovném otevření databáze se změny zobrazí i v grafickém uživatelském rozhraní.

Změny se také zobrazí, pokud v kontejneru tabulky zvolíme **Zobrazení > Obnovit tabulky**.

Odstranění tabulky

```
DROP TABLE "Table name" [IF EXISTS] [RESTRICT | CASCADE];
```

Odstraní tabulku *Table name*.

IF EXISTS zabrání chybě, pokud tato tabulka neexistuje.

RESTRICT je výchozí uspořádání a nemusí být explicitně zvoleno; znamená, že k vymazání nedojde, pokud je tabulka propojena s jinou tabulkou pomocí cizího klíče nebo je aktivní zobrazení této tabulky. Dotazy nejsou ovlivněny, protože nejsou uloženy v HSQLDB.

Pokud místo toho zvolíme CASCADE, budou odstraněny všechny odkazy na tabulku Table_name. V propojených tabulkách jsou všechny cizí klíče nastaveny na NULL. Všechny pohledy odkazující na pojmenovanou tabulku jsou také zcela odstraněny.

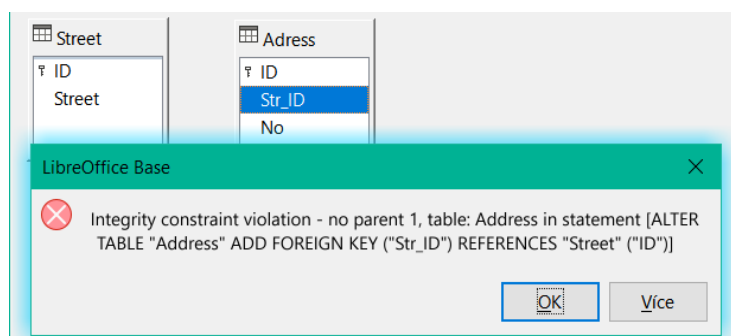
Propojení tabulek

V zásadě můžeme mít databázi bez odkazů mezi tabulkami. Uživatel pak musí při zadávání dat zajistit, aby vztahy mezi tabulkami zůstaly správné. K tomu obvykle slouží vhodné vstupní formuláře, které to umožňují.

Odstranění záznamů v propojených tabulkách není jednoduchá záležitost. Předpokládejme, že chceme odstranit konkrétní ulici z tabulky Street na obrázku 130, kde je toto pole propojeno s tabulkou Address jako cizí klíč v této tabulce. Odkazy v tabulce Address by zmizely. Databáze to po vytvoření vztahu neumožňuje. Aby bylo možné ulici odstranit, musí být splněna podmínka, že na ni již není odkazováno v tabulce Address.

Základní vazby se vytvářejí pomocí **Nástroje > Relace**. Tím se vytvoří spojovací linie z primárního klíče v jedné tabulce na definovaný cizí klíč v druhé tabulce.

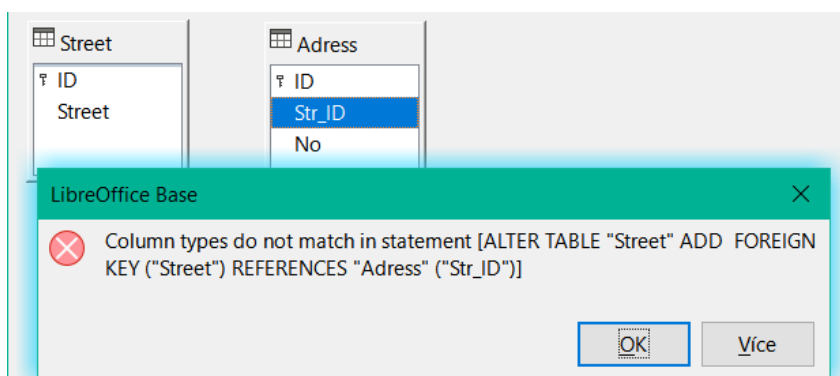
Při vytváření takového odkazu se může zobrazit následující chybová zpráva:



Obrázek 151: Typy sloupců se při vytváření relace neshodují

Tato zpráva zobrazuje chybu, ke které došlo, a interní příkaz SQL, který chybu způsobil.

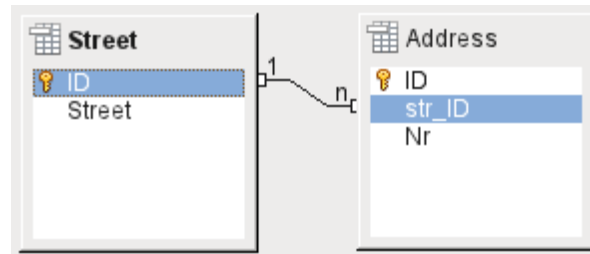
Typy sloupců se v příkazu neshodují—Při zobrazení příkazu SQL je zřejmý odkaz na sloupec Address.str_ID a Street.ID. Pro účely testu bylo jedno z těchto polí definováno jako Integer, druhé jako Tiny Integer. Proto nelze vytvořit žádné propojení, protože jedno pole nemůže mít stejnou hodnotu jako druhé.



Obrázek 152: Porušení omezení integrity při vytváření vztahu

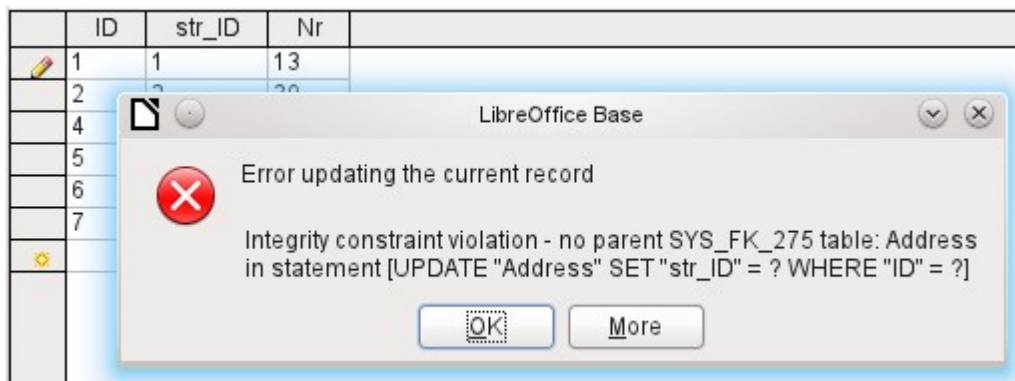
V tomto případě se typy sloupců shodují. Příkaz SQL je stejný jako v prvním příkladu. Opět je zde však chyba:

Porušení omezení integrity - žádný rodič 1, tabulka: Address... —Nelze stanovit integritu relace. V poli str_ID tabulky Address je číslo 1, které se v poli ID tabulky Street nevyskytuje. Nadřazenou tabulkou je zde Street, protože musí existovat její primární klíč. Tato chyba je velmi častá, pokud mají být propojeny dvě tabulky a některá pole v tabulce s předpokládaným cizím klíčem již obsahují data. Pokud pole cizího klíče obsahuje položku, která se nenachází v rodičovské tabulce (tabulka obsahující primární klíč), jedná se o neplatnou položku.



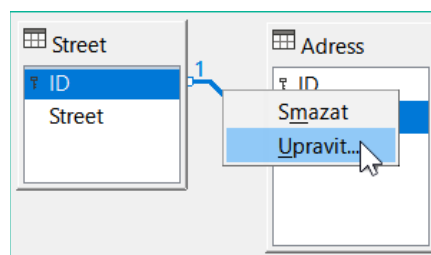
Obrázek 153: Relace úspěšně vytvořena

Pokud je propojení provedeno úspěšně a následně dojde k pokusu o zadání podobně neplatného záznamu do tabulky, zobrazí se následující chybové hlášení:



Obrázek 154: Porušení integritního omezení při zadávání dat

Opět se jedná o porušení integrity. Program Base odmítá přijmout hodnotu 1 pro pole str_ID po vytvoření odkazu, protože tabulka Street takovou hodnotu v poli ID neobsahuje.



Obrázek 155: Úprava odkazů klepnutím pravým tlačítkem myši

Vlastnosti odkazu lze upravit tak, aby se při odstranění záznamu z tabulky Street současně nastavila hodnota NULL u odpovídajících záznamů v tabulce Address.

Vlastnosti zobrazené na obrázku 155 se vždy vztahují k akci spojené se změnou záznamu z tabulky obsahující příslušný primární klíč. V našem případě je to tabulka Street. Pokud je primární klíč záznamu v této tabulce změněn (Update), může dojít k následujícím akcím.

Žádná akce

Změna primárního klíče Street.ID není v tomto případě povolena, protože by došlo k narušení vztahu mezi tabulkami.

Kaskádová aktualizace

Pokud se změní primární klíč Street.ID, cizí klíč se automaticky změní na novou hodnotu. Tím je zajištěno, že nedojde k poškození připojených tabulek. Pokud se například změní hodnota z 3 na 4, změní se u všech záznamů z tabulky Address, které obsahují cizí klíč Address.Street_ID s hodnotou 3, na 4.

Nastavit null

Všechny záznamy, které obsahují tento konkrétní primární klíč, nyní nebudou mít žádný záznam v poli cizího klíče Address.Street_ID; pole bude mít hodnotu NULL.

Relace

Použité tabulky

Address Street

Použitá pole

Address	Street
Str_ID	ID

Možnosti aktualizace

Žádná akce

Kaskádová aktualizace

Nastavit NULL

Nastavit výchozí

Možnosti smazání

Žádná akce

Kaskádové smazání

Nastavit NULL

Nastavit výchozí

Nápověda OK Zrušit

Obrázek 156: Úprava vlastností relace

Nastavení výchozí hodnoty

Pokud se změní primární klíč Street_ID, nastaví se hodnota původně přiřazeného klíče Address.Street_ID na dříve definovanou výchozí hodnotu. Za tímto účelem potřebujeme jednoznačnou definici výchozí hodnoty. Pokud je výchozí hodnota nastavena pomocí příkazu SQL:

```
ALTER TABLE "Address" ALTER COLUMN "Street_ID" SET DEFAULT 1;
```

definice odkazu zajišťuje, že se pole v případě aktualizace vrátí na tuto hodnotu. Pokud se tedy změní primární klíč v tabulce Street, odpovídající cizí klíč v tabulce Address se nastaví na hodnotu 1. To je užitečné v případě, že je u záznamu vyžadováno pole ulice, jinými slovy toto pole nesmí být NULL. Ale pozor: pokud se 1 nepoužívá, vytvořili jsme odkaz na neexistující hodnotu. Je tedy možné narušit integritu vztahu.

Upozornění

Pokud by výchozí hodnota v poli cizího klíče nebyla propojena s hodnotou primárního klíče cizí tabulky, vytvořil by se odkaz na hodnotu, která není možná. Referenční integrita databáze by byla zničena.

Lepší by bylo využít možnosti nastavit výchozí hodnotu.

Pokud je záznam *odstraněn* z tabulky Street, jsou k dispozici následující možnosti.

Žádná akce

Žádná akce se nekoná. Pokud se požadované odstranění týká záznamu v tabulce Address, bude požadavek zamítnut.

Kaskádové mazání

Pokud je odstraněn záznam z tabulky Street a ovlivní to záznam v tabulce Address, bude odstraněn i tento záznam.

V tomto kontextu se to může zdát zvláštní, ale existují i jiné struktury tabulek, ve kterých to dává smysl. Předpokládejme, že máme tabulku CD a tabulku, ve které jsou uloženy názvy těchto CD. Pokud je nyní záznam v tabulce CD smazán, mnoho titulů v jiné tabulce nemá žádný význam, protože již nejsou k dispozici. V takových případech má kaskádové mazání smysl. To znamená, že před odstraněním CD z databáze nemusíme odstranit všechny tituly.

Nastavení na hodnotu Null

Je to stejné jako u možnosti aktualizace.

Nastavit na výchozí hodnotu

Tento postup je stejný jako u možnosti aktualizace a vyžaduje stejná opatření.



Tip

Ve většině případů je třeba se vyhnout možnosti Žádná akce, aby se uživatelé nezobrazovali chybová hlášení z databáze, protože ta nemusí být pro uživatele vždy srozumitelná.

V nabídce **Nástroje > Relace** se přetažením myši vytvoří cizí klíče, které odkazují na jedno pole v jiné tabulce. Chceme-li vytvořit odkaz na tabulku, která má složený primární klíč, přejdeme na **Nástroje > Relace** a poté na **Vložit > Nová relace**, nebo použijeme příslušné tlačítko. Zobrazí se dialogové okno pro úpravu vlastností vztahu s volným výběrem dostupných tabulek.

Zadávání dat do tabulek

Databáze, které se skládají pouze z jedné tabulky, obvykle nevyžadují vstupní formulář, pokud neobsahují pole pro obrázky. Jakmile však tabulka obsahuje cizí klíče z jiných tabulek, musí si uživatelé buď pamatovat, která čísla klíčů mají zadat, nebo musí mít možnost podívat se současně do jiných tabulek. V takových případech je užitečný formulář.

Zadání pomocí grafického uživatelského rozhraní aplikace Base

Tabulky v kontejneru tabulek se otevírají poklepáním. Pokud je primárním klíčem automaticky inkrementující pole, bude jedno z viditelných polí obsahovat text *Automatická hodnota*. Do pole Automatická hodnota není možné zadávat žádné údaje. Jeho přiřazenou hodnotu lze v případě potřeby změnit, ale až poté, co byl záznam uložen.

ID	Title	Edition	Publ.	Pub. Year
0				19
1				19
2				19
3	Traditionelle und k...			19
4	Die neue deutsche			19

Obrázek 157: Zadání položky do tabulek – skrytí sloupců

ID	Title	Edition	Pub. Year	Comment
0	Der kleine Ho...			
1	Das sogennat...			
2	Eine kurze Ge...			
3	Traditionelle			
4	Die neue deutsche		1996	
5	I hear you knocking	1	1972	
6	Datenbanken mit C3., aktualisi		2009	
7	Das Bestf...		2000	

Obrázek 158: Zadání položky do tabulek – zrušení skrývání sloupců

Jednotlivé sloupce v zobrazení tabulky dat lze skrýt. Pokud například pole primárního klíče nemusí být viditelné, lze to zadat v tabulce v zobrazení pro zadávání dat klepnutím pravým tlačítkem myši na záhlaví sloupce. Toto nastavení je uloženo v grafickém uživatelském rozhraní. Sloupec v tabulce nadále existuje a lze jej vždy znovu zviditelnit.

Do tabulky se obvykle zapisuje zleva doprava pomocí klávesnice s klávesami Tab nebo Enter. Můžeme také použít myš.

Po dosažení posledního pole záznamu kurzor automaticky přejde na další záznam. Předchozí záznam se uloží do úložiště. Další ukládání pomocí **Soubor > Uložit** není nutné a ani možné. Data jsou již v databázi.

! Upozornění

V případě HSQLDB jsou data v pracovní paměti. Na pevný disk se přenesou až po zavření aplikace Base. To není z hlediska bezpečnosti dat vždy žádoucí, protože pokud se aplikace Base z nějakého důvodu neuzavře spořádaným způsobem, může to vést ke ztrátě dat.

Pokud nejsou zadána žádná data do pole, které bylo při návrhu tabulky definováno jako povinné (NOT NULL), zobrazí se příslušné chybové hlášení:
 Pokus o vložení prázdné hodnoty do povinného sloupce...

Zobrazí se také příslušný sloupec, tabulka a příkaz SQL (přeložený grafickým uživatelským rozhraním).

Změna záznamu je snadná: vyhledáme pole, zadáme jinou hodnotu a řádek opět opustíme.

	ID	Title	Edition	Pub_Year
▶	0	Der kleine Hobbit	2. Aufl.	1972
				1972
				1983
				1970
				1996
			1	1972
	6	Datenbanken mit C	3., aktualisi	2009

Obrázek 159: Odstranění řádku(ů) tabulky

Chceme-li odstranit záznam, vybereme řádek klepnutím na jeho záhlaví (šedá oblast vlevo), klepneme pravým tlačítkem myši a zvolíme **Smazat řádky**.

Existuje poměrně dobře skrytá metoda pro kopírování celých řádků. Aby to fungovalo, musí být primární klíč tabulky definován jako Automatická hodnota.

	ID	Title	Edition	Pub_Year
	0	Der kleine Hobbit	2. Aufl.	1972
▶	1	Das sogenannte Böse		1972
	2	Eine kurze Geschichte der Zeit		1983
		Traditionelle und kritische Theorie		1970
	4	Die neue deutsche Rechtschreibung		1996
	5	I hear you knocking	1	1972
	6	Datenbanken mit OpenOffice.org 3	3., aktualisi	2009
	7	Das Postfix-Buch		2008
	8	Im Augenblick		2009
+	<Auto			

Obrázek 160: Kopírování řádku tabulky

Nejprve se klepne levým tlačítkem myši na záhlaví řádku. Poté podržíme stisknuté tlačítko a přetáhneme myš. Kurzor se změní na symbol se znaménkem +. To znamená, že záznam je zkopírován do poslední položky tabulky.

	ID	Title	Edition	Pub_Year
	0	Der kleine Hobbit	2. Aufl.	1972
▶	1	Das sogenannte Böse		1972
	2	Eine kurze Geschichte der Zeit		1983
	3	Traditionelle und kritische Theorie		1970
	4	Die neue deutsche Rechtschreibung		1996
	5	I hear you knocking	1	1972
	6	Datenbanken mit OpenOffice.org 3	3., aktualisi	2009
	7	Das Postfix-Buch		2008
	8	Im Augenblick		2009
	9	Das sogenannte Böse		1972
+	<Auto			

Obrázek 161: Řádek s ID 1 zkopírován na konec tabulky jako ID 9

Záznam s primárním klíčem **1** se vloží jako nový záznam s novým primárním klíčem **9**.

Pokud je pomocí klávesy Control nebo Shift označena skupina záznamů, budou tyto záznamy zkopírovány jako skupina.

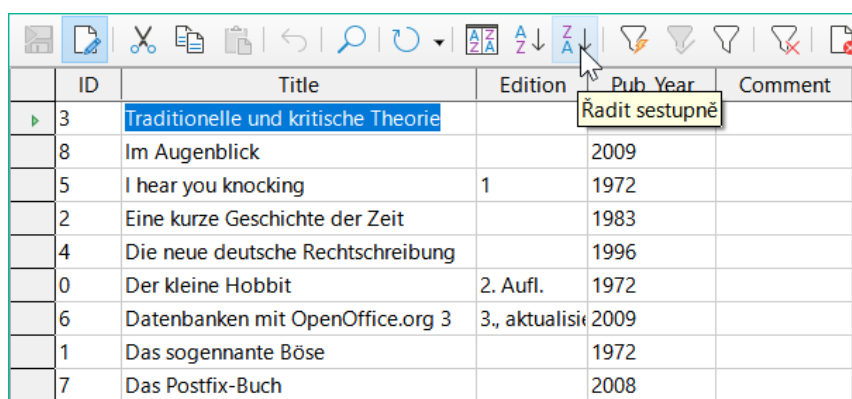
Tip

Záhlaví sloupců lze přetáhnout na šířku vhodnou pro zadávání. Pokud se to provede v tabulce, program Base automaticky uloží novou šířku sloupce do tabulky.

Šířka sloupců v tabulkách ovlivňuje šířku sloupců v dotazech. Pokud jsou sloupce v dotazu příliš úzké, bude mít jejich rozšíření pouze dočasný účinek. Nová šířka se neuloží. Je lepší sloupec v tabulce rozšířit, aby se v dotazech zobrazoval správně a nebylo nutné měnit jeho velikost.

Funkce Třídít, Hledat a Filtrovat jsou velmi užitečné pro vyhledávání konkrétních záznamů.

Třídění tabulek

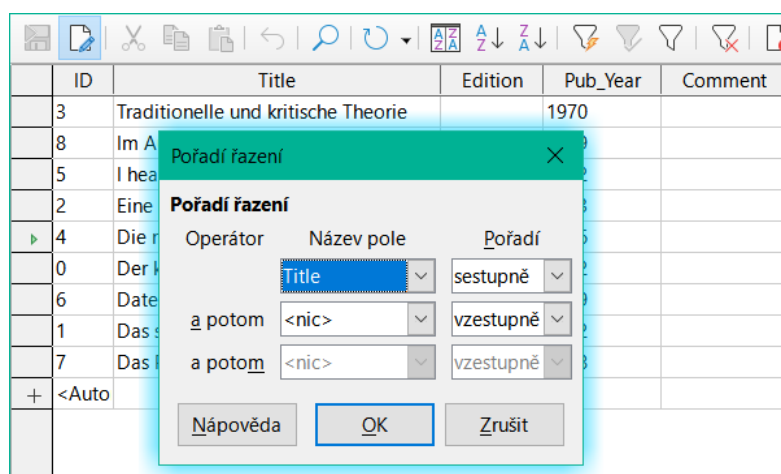


ID	Title	Edition	Pub Year	Comment
3	Traditionelle und kritische Theorie			
8	Im Augenblick		2009	
5	I hear you knocking	1	1972	
2	Eine kurze Geschichte der Zeit		1983	
4	Die neue deutsche Rechtschreibung		1996	
0	Der kleine Hobbit	2. Aufl.	1972	
6	Datenbanken mit OpenOffice.org 3	3., aktualisi	2009	
1	Das sogenannte Böse		1972	
7	Das Postfix-Buch		2008	

Obrázek 162: Rychlé třídění

Tlačítka A > Z a Z > A umožňují rychlé řazení. Nejprve vybereme pole. Poté klepneme na tlačítko odpovídající vzestupnému nebo sestupnému řazení a data se seřadí podle daného sloupce. Obrázek 162 zobrazuje sestupné řazení podle pole Title.

Rychlé třídění seřadí pouze podle jednoho sloupce. Chceme-li řadit podle více sloupců současně, nalevo od tlačítek rychlého řazení je k dispozici pokročilejší funkce řazení:

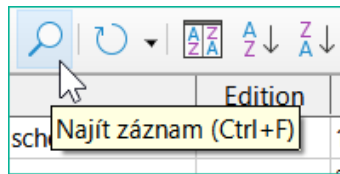


Operátor	Název pole	Pořadí
	Title	sestupně
a potom	<nic>	vzestupně
a potom	<nic>	vzestupně

Obrázek 163: Řazení podle více než jednoho sloupce

Vybere se název pole sloupce a aktuální pořadí řazení. Pokud bylo provedeno předchozí rychlé třídění, první řádek již bude obsahovat odpovídající název pole a pořadí třídění.

Vyhledávání tabulek



Obrázek 164: Ikona Najít záznam na nástrojové liště Tabulka dat

Tlačítko **Najít záznam** je jednoduchá metoda pro vyhledávání záznamů ve velké tabulce. Funkce vyhledávání je však u rozsáhlých databází velmi pomalá, protože vyhledávání nepoužívá příkaz SQL v rámci databáze. Pro rychlejší vyhledávání použijeme místo funkce Najít záznam dotaz. Pro eliminaci častých úprav dotazu lze navrhnout jeho spouštění pomocí parametrů. Viz kapitola 5, Dotazy, část „Použití parametrů v dotazech“.

Tip

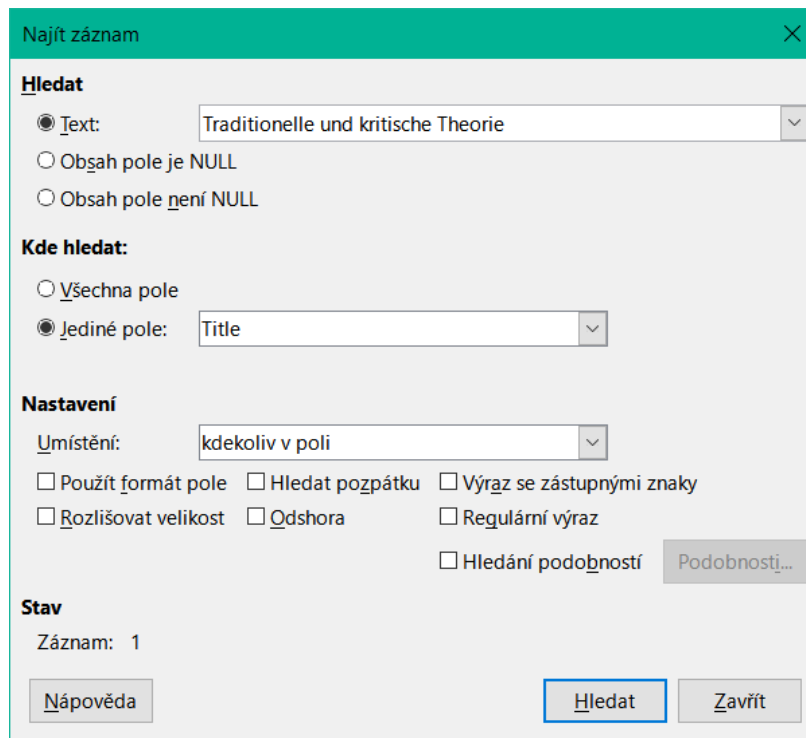
Před vyhledáváním se ujistíme, že sloupce, ve kterých budeme hledat, jsou dostatečně široké, aby správně zobrazovaly nalezené záznamy. Okno vyhledávání zůstane v popředí a nebude možné opravit nastavení šířky sloupců v základní tabulce. Chceme-li se dostat ke stolu, musíme hledání přerušit.

Tlačítko Najít záznam automaticky vyplní hledaný výraz obsahem pole, ze kterého bylo vyvoláno.

Aby bylo hledání účinné, měla by být oblast hledání co nejvíce omezena. Bylo by zbytečné vyhledávat výše uvedený text z pole Title v poli Author. Namísto toho je jméno pole Title již navrženo pro Název pole.

Další nastavení vyhledávání může usnadnit práci prostřednictvím specifických kombinací. Můžeme použít běžné zástupné znaky SQL ("_" pro proměnný znak, "%" pro libovolný počet proměnných znaků nebo "\" jako escape znak, který umožňuje vyhledávání těchto speciálních znaků).

Regulární výrazy jsou podrobně popsány v nápovědě LibreOffice.

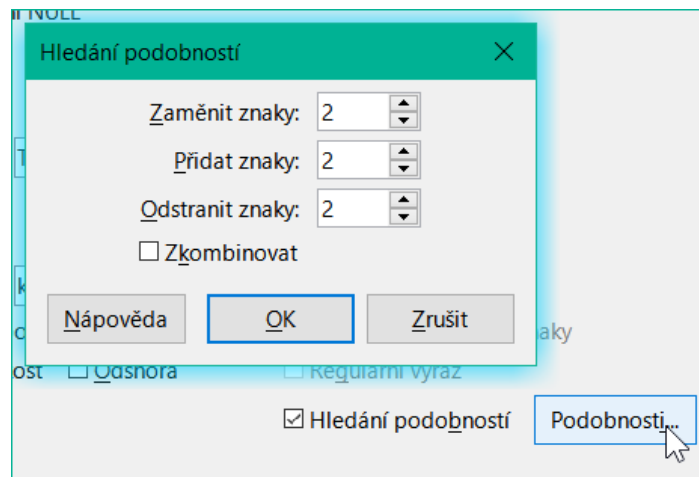


Obrázek 165: Vstupní maska pro vyhledávání záznamů

Funkce vyhledávání podobností je užitečná, pokud potřebujeme vyloučit pravopisné chyby. Čím vyšší hodnoty nastavíme, tím více záznamů se v konečném seznamu zobrazí.

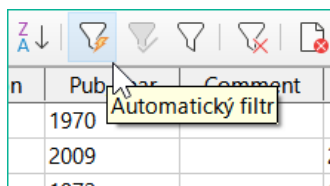
Tento vyhledávací modul je nejhodnější pro lidi, kteří z pravidelného používání přesně vědí, jak dosáhnout daného výsledku. Většina uživatelů má větší šanci na úspěšné nalezení záznamů pomocí filtru.

Kapitola 4 této knihy popisuje použití formulářů pro vyhledávání a způsob, jakým lze pomocí jazyka SQL a maker dosáhnout vyhledávání podle klíčových slov.



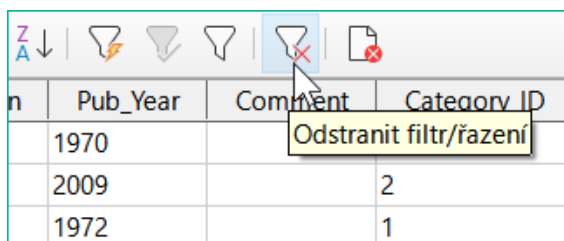
Obrázek 166: Omezení hledání podobnosti

Filtrování tabulek



Obrázek 167: Ikona AutoFilter na nástrojové liště Tabulka dat

Tabulku můžeme rychle filtrovat pomocí **Automatického filtru**. Umístíme kurzor do buňky pole a jedním klepnutím na ikonu filtr převezme obsah tohoto pole. Zobrazí se pouze ty záznamy, u nichž má vybrané pole stejný obsah jako vybraná buňka. Obrázek níže ukazuje filtrování podle položky ve sloupci **Pub_Year**.



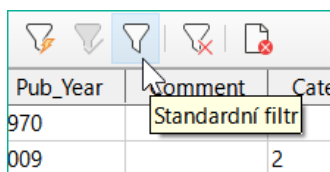
Obrázek 168: Ikona Obnovit filtr/uspořádání na nástrojové liště Tabulka dat

Filtr je aktivní, což je znázorněno ikonou filtru se zeleným zaškrtnutím. Symbol filtru je zobrazen stisknutý. Toto tlačítko je přepínací, takže pokud na něj klepneme znovu, filtr bude existovat i nadále, ale nyní se zobrazí všechny záznamy. Pokud chceme, můžeme na něj kdykoli klepnout a vrátit se do filtrovaného stavu.

Klepnutím na ikonu **Odstranit filtr/řazení** vpravo dole se odstraní všechny existující filtry a řazení. Filtry se stanou neaktivními a nelze již obnovit jejich staré hodnoty.

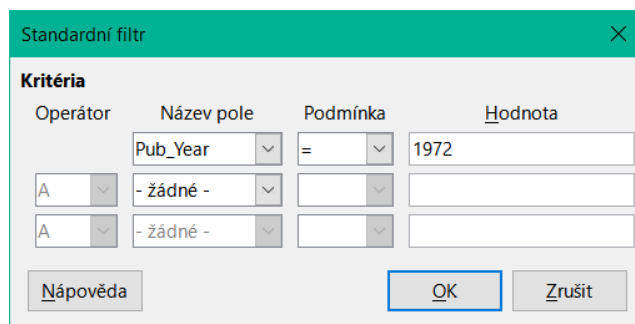
Tip

Do filtrované tabulky nebo do tabulky, která byla omezena vyhledáváním, můžeme stále zadávat záznamy normálně. V zobrazení tabulky zůstávají viditelné, dokud není tabulka aktualizována klepnutím na tlačítko **Obnovit**.



Obrázek 169: Ikona Standardní filtr na nástrojové liště Tabulka dat

Ikona **Standardní filtr** otevře dialogové okno, ve kterém můžeme filtrovat pomocí několika kritérií současně, podobně jako při třídění. Pokud se používá Automatický filtr, zobrazí se toto existující kritérium filtru již na prvním řádku standardního filtru.



Obrázek 170: Filtrování vícenásobných dat pomocí standardního filtru

Standardní filtr poskytuje mnoho funkcí filtrování dat SQL. K dispozici jsou následující příkazy SQL.

Tabulka 4: Podmínky dostupné pro standardní filtry

Grafické zobrazení podmínky	Popis
=	Přesná rovnost; odpovídá like , ale bez dalších zástupných znaků.
<>	Není rovno
<	Méně než
<=	Menší než nebo rovno
>	Větší než
>=	Větší než nebo rovno
like	Pro text psaný v uvozovkách (' '); "_" pro proměnný znak, "%" pro libovolný počet proměnných znaků.
not like	Opak like , v SQL NOT LIKE .
prázdný	Žádné zadání, dokonce ani znak mezery. V jazyce SQL je toto vyjádřeno termínem NULL .
Není prázdný	Opak prázdného, v SQL NOT NULL

Před kombinací jednoho kritéria filtru s jiným musí být na následujícím řádku vybrán alespoň jeden název pole. Na obrázku 170 je místo názvu pole zobrazeno slovo – none – (- žádné -, pozn. překl.), takže kombinace není aktivní. K dispozici jsou kombinační operátory **AND** a **OR**.

Název pole může být nový nebo dříve vybraný název pole.

I v případě velkých souborů dat lze počet vyhledaných záznamů snížit na zvládnutelnou množinu pomocí vhodného filtrování s využitím těchto tří možných podmínek.

Také v případě filtrování formulářů existují některé další možnosti (popsané v kapitole 4), které grafické uživatelské rozhraní neposkytuje.

Přímé zadání pomocí SQL

Přímé zadávání dat pomocí jazyka SQL je užitečné pro zadávání, změnu nebo odstranění více záznamů jedním příkazem.

Zadávání nových záznamů

```
INSERT INTO "Table_name" [( "Field_name" [,...] )]
{ VALUES("Field value" [,...] ) | <Select-Formula>;
```

Pokud není zadáno pole `Field_name`, musí být všechna pole vyplněna a ve správném pořadí (jak je uvedeno v tabulce). To zahrnuje i automaticky inkrementované pole primárního klíče, pokud je k dispozici. Zadané hodnoty mohou být také výsledkem dotazu (<Select-Formula>).

Přesnější informace jsou uvedeny níže.

```
INSERT INTO "Table_name" ("Field_name") VALUES ('Test');  
CALL IDENTITY();
```

V tabulce se do sloupce `Field_name` vloží hodnota `Test`. Automaticky inkrementované pole primárního klíče `ID` není dotčeno. Příslušnou hodnotu `ID` je třeba vytvořit samostatně pomocí `CALL IDENTITY()`. To je důležité při použití maker, aby bylo možné hodnotu tohoto klíčového pole použít později.

```
INSERT INTO "Table_name" ("Field_name") SELECT "Other_fieldname" FROM  
"Name_of_other_table";
```

V první tabulce se do pole `Field_name` vloží tolik nových záznamů, kolik jich je ve sloupci `Other_fieldname` v druhé tabulce. Zde lze samozřejmě použít vzorec `Select-Formula` k omezení počtu záznamů.

Úprava existujících záznamů

```
UPDATE "Table_name" SET "Field_name" = <Expression> [, ...] [WHERE  
<Expression>];
```

Při úpravě mnoha záznamů najednou je velmi důležité pečlivě zkontrolovat zadávaný příkaz SQL. Předpokládejme, že všichni žáci ve třídě budou posunuti o jeden ročník výše:

```
UPDATE "Table_name" SET "Year" = "Year"+1
```

Nic nemůže být rychlejší: Všechny datové záznamy se změní jediným příkazem. Představme si však, že nyní musíme určit, kterých studentů se tato změna neměla týkat. Bylo by jednodušší zaškrtnout pole `Yes/No` pro opakování ročníku a poté přesunout nahoru pouze ty studenty, u kterých toto pole nebylo zaškrtnuto:

```
UPDATE "Table_name" SET "Year" = "Year"+1 WHERE "Repetition" = FALSE
```

Tyto podmínky fungují pouze v případě, že dané pole může nabývat pouze hodnot `FALSE` a `TRUE`; nesmí být `NULL`. Bylo by bezpečnější, kdyby byla podmínka formulována jako `WHERE "Repetition" <> TRUE`.

Pokud chceme následně do určitého pole, které je prázdné, zadat výchozí hodnotu, můžeme to provést pomocí příkazu:

```
UPDATE "Table" SET "Field" = 1 WHERE "Field" IS NULL
```

Přímým přiřazením hodnot můžeme změnit několik polí najednou. Předpokládejme, že tabulka knih obsahuje jména jejich autorů. Zjistilo se, že `Erich Kästner` byl často zapsán jako `Eric Käschtner`.

```
UPDATE "Books" SET "Author_first_name" = 'Erich', "Author_surname" = 'Kästner'  
WHERE "Author_first_name" = 'Eric' AND "Author_surname" = 'Käschtner'
```

Pomocí funkce `Update` je možné provést i další kroky výpočtu. Pokud má být například zboží s cenou vyšší než 150,00 USD zahrnuto do speciální nabídky a cena snížena o 10 %, lze to provést takto:

```
UPDATE "Table_name" SET "Price" = "Price"*0.9 WHERE "Price" >= 150
```

Pokud zvolíme datový typ `CHAR`, pole má pevnou šířku. V případě potřeby je text doplněn nulovými znaky. Pokud jej převedeme na `VARCHAR`, tyto nulové znaky zůstanou zachovány. Chceme-li je odstranit, použijeme funkci oříznutí prázdných znaků zprava:

```
UPDATE "Table_name" SET "Field_name" = RTRIM("Field_name")
```

Odstranění existujících záznamů

```
DELETE FROM "Table_name" [WHERE <Expression>];
```

Bez podmíněného výrazu je příkaz

```
DELETE FROM "Table_name"
```

odstraní celý obsah tabulky.

Z tohoto důvodu je vhodnější, aby byl příkaz konkrétnější. Pokud je například zadána hodnota primárního klíče, bude smazán pouze tento přesný záznam.

```
DELETE FROM "Table_name" WHERE "ID" = 5;
```

Pokud má být v případě výpůjčky záznam o médiu při vrácení položky vymazán, lze to provést pomocí příkazu

```
DELETE FROM "Table_name" WHERE NOT "Return_date" IS NULL;
```

nebo alternativně pomocí

```
DELETE FROM "Table_name" WHERE "Return_date" IS NOT NULL;
```

Import dat z jiných zdrojů

Někdy jsou v jiném programu kompletní datové sady, které je třeba importovat do programu Base prostřednictvím schránky. To může zahrnovat vytvoření nové tabulky nebo přidání záznamů do stávající tabulky.



Poznámka

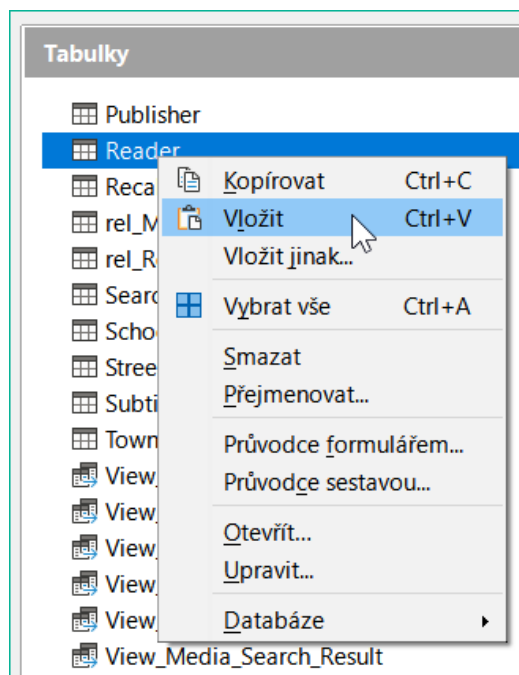
Chceme-li importovat data pomocí schránky, musí být formát dat čitelný v aplikaci Base. To platí vždy pro datové soubory otevřené v LibreOffice.

Pokud se například mají do souboru *.odb načíst tabulky z externí databáze, musí být tato databáze nejprve otevřena v LibreOffice nebo zaregistrována v LibreOffice jako zdroj dat. Viz „Přístup k externím databázím“ v kapitole 2, Vytvoření databáze.

	A	B	C
1	ID	FirstName	LastName
2	10	Robert	Großkopf
3	11	Maike	Longfoot
4	12	George	Orwell
5			

Obrázek 171: Data ke zkopírování do tabulky

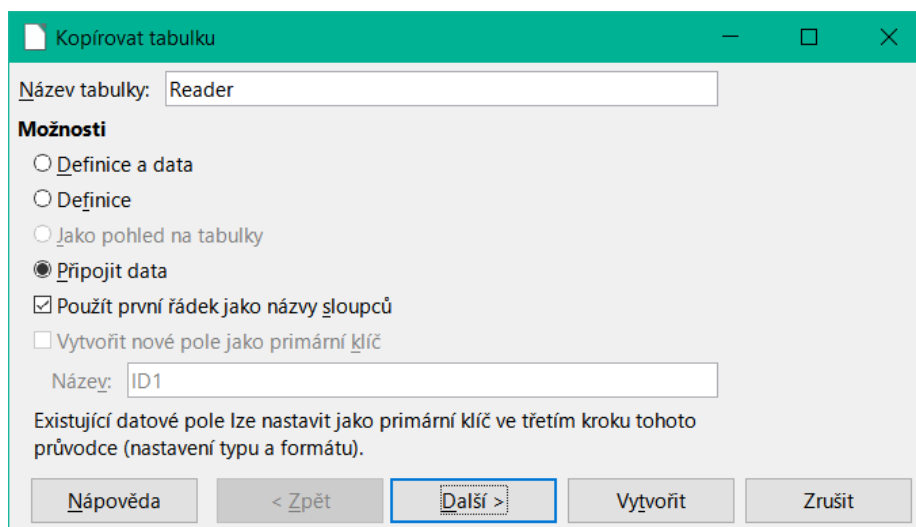
Nahoře je malá ukázková tabulka zkopírovaná z tabulkového procesoru Calc do schránky. Poté se vloží v aplikaci Base do kontejneru Table. Samozřejmě to šlo udělat i tak, že jsme ji vybrali levým tlačítkem myši a pak ji přetáhli na druhou stranu.



Obrázek 172: Vložení dat do požadované tabulky

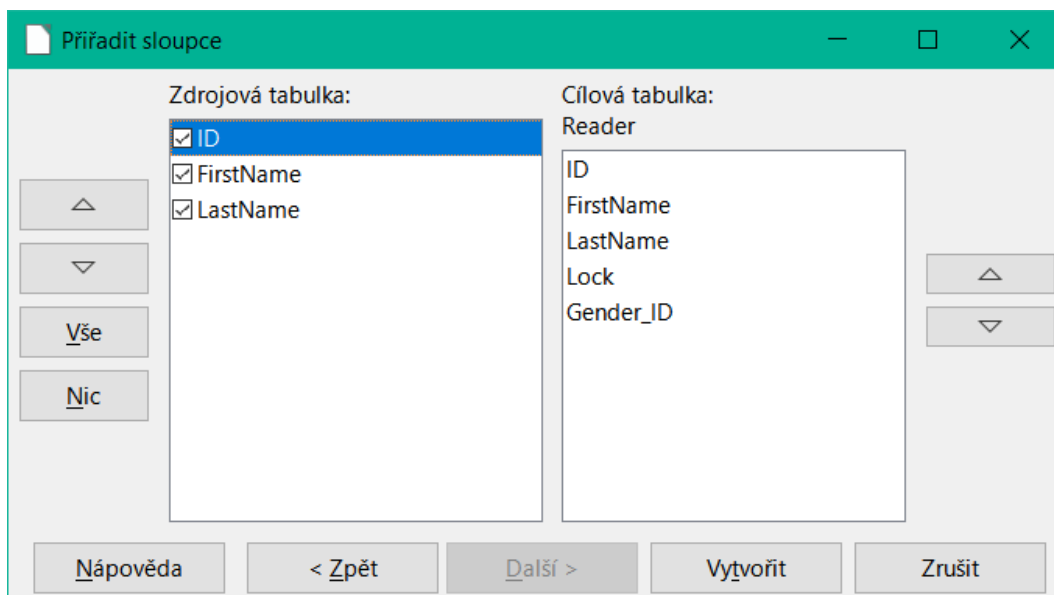
V kontejneru Table klepneme pravým tlačítkem myši a otevřeme místní nabídku pro tabulku, do které mají být záznamy přidány.

Přidání importovaných záznamů do existující tabulky



Obrázek 173: Dialogové okno Kopírovat tabulku – přidávání dat

Název tabulky se zobrazí v průvodci importem. Současně je vybrána volba *Připojit data*. *Použít první řádek jako názvy sloupců* může, ale nemusí být vyžadován v závislosti na verzi LibreOffice. Pokud mají být záznamy připojeny, není definice dat vyžadována. K dispozici musí být také primární klíč.



Obrázek 174: Přidat dat sloupců – doplnění dat

Sloupce zdrojové tabulky aplikace Calc a cílové tabulky v aplikaci Base se nemusí shodovat v pořadí, názvech ani celkovém počtu. Přenesou se pouze prvky vybrané z levé strany. Shodu mezi zdrojovou a cílovou tabulkou je třeba upravit pomocí tlačítek se šipkami na obou stranách.

Tím je import dokončen.

Import může vést k problémům, pokud:

- Pole v cílové tabulce vyžadují povinný záznam, ale zdrojová tabulka pro ně neposkytuje žádné údaje.
- Definice polí v cílové tabulce neodpovídají definicím ve zdrojové tabulce (například do číselného pole má být zadáno jméno nebo cílové pole má příliš málo znaků pro data).
- Zdrojová tabulka poskytuje data nekonzistentní s daty cílové tabulky, například nejednotné hodnoty primárních klíčů nebo jiných polí definovaných jako jedinečné.

Vytvoření nové tabulky pro importovaná data

Po spuštění Průvodce importem se automaticky zobrazí dříve vybraný název tabulky. Pokud vytváříme novou tabulku, je nutné tento název tabulky změnit, protože je zakázáno mít tabulku se stejným názvem jako již existující. Název této tabulky je *Names*. *Definice a data* mají být přeneseny. První řádek se použije jako záhlaví sloupců.

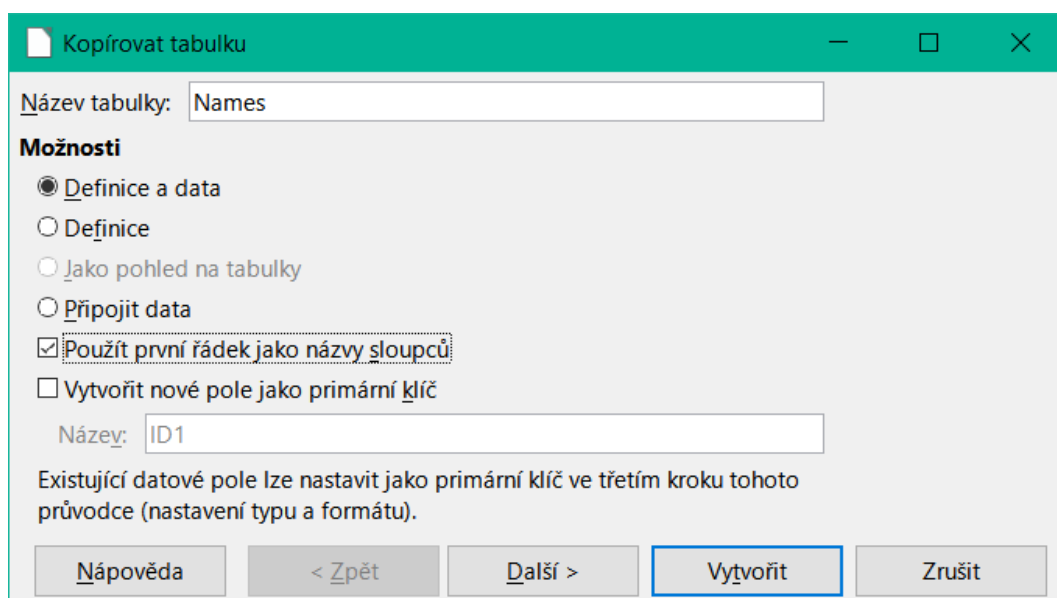
V tomto okamžiku můžeme vytvořit nové, další pole pro primární klíč. Název tohoto pole nesmí existovat jako záhlaví sloupce v tabulce aplikace Calc. V opačném případě se zobrazí chybová zpráva:

Následující pole jsou již nastavena jako primární klíče: ID

Tato zpráva však situaci nevysvětluje zcela správně.

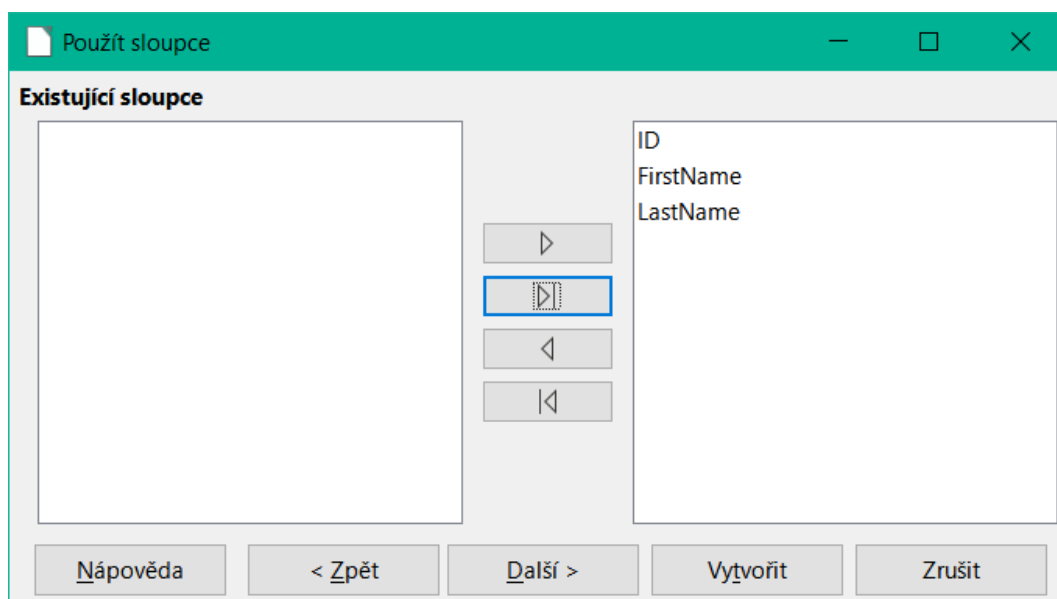
Pokud chceme, aby jako primární klíč sloužilo existující pole, nevybereme možnost *Vytvořit primární klíč*. V tomto případě vytvoříme pole primárního klíče na třetí stránce dialogového okna Průvodce.

Při importu se přenesou definice tabulky a data.



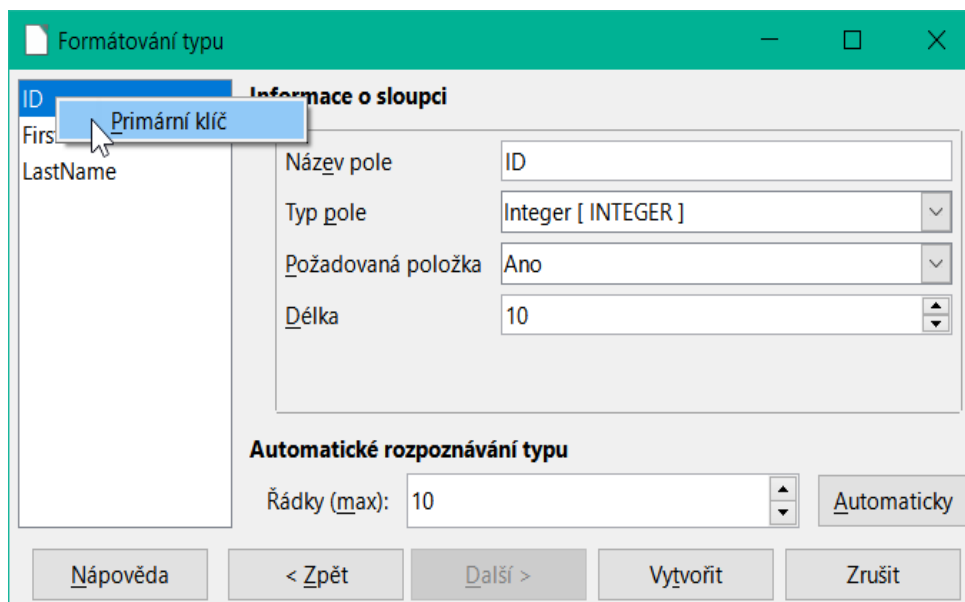
Obrázek 175: Dialog Kopírovat tabulku – vytvoření nové tabulky

Přenesou se všechny dostupné sloupce.



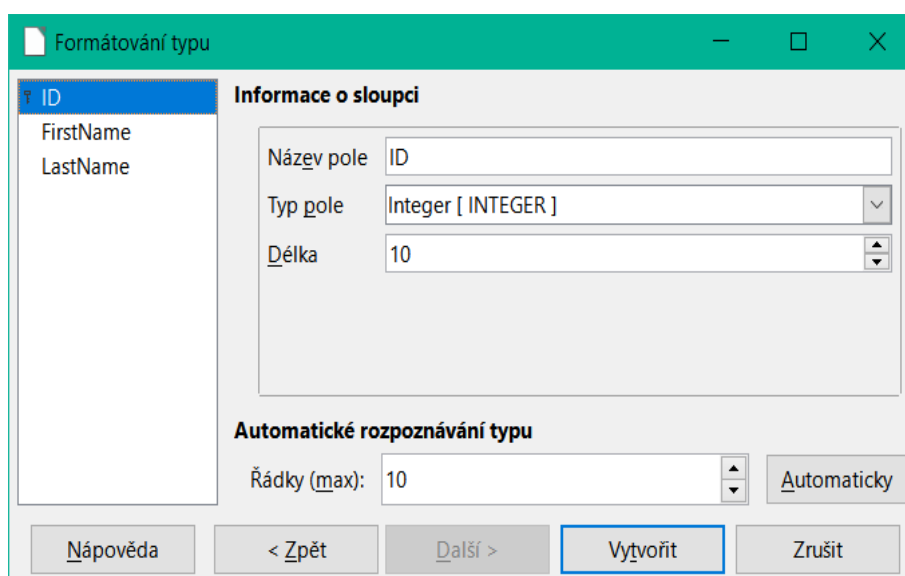
Obrázek 176: Dialogové okno Použít sloupce

Formátování typů tabulek často vyžaduje úpravu. Obvykle jsou pole předdefinována jako textová pole s velmi velkou velikostí. Číselná pole a pole s daty by proto měla být resetována pomocí **Typ formátování > Informace o sloupci > Typ pole**. V případě desetinných čísel je třeba zkontrolovat počet desetinných míst.



Obrázek 177: Dialogové okno Formátování typu – nastavení primárního klíče

Možnost výběru primárního klíče je poněkud nejasně přítomna v místní nabídce pole, které jej má obsahovat. V tomto příkladu bylo pole ID naformátováno způsobem, který umožňuje jeho použití jako primárního klíče. Pokud nebyl primární klíč vytvořen jako další pole v okně Kopírovat tabulku v průvodci, je třeba jej nyní explicitně nastavit pomocí místní nabídky názvu pole.



Obrázek 178: Dialogové okno Formátování typu – ID nastaveno jako primární klíč

Po klepnutí na tlačítko **Vytvořit** se vytvoří tabulka a vyplní se zkopírovanými daty.

Nový primární klíč není klíčem Automatické hodnoty. Chceme-li takovou tabulku vytvořit, musíme ji otevřít pro úpravy. Poté můžeme provádět další operace formátování.

Rozdělení dat při importu

Někdy nejsou zdrojová data k dispozici v požadované podobě. Například adresy se do tabulek často zadávají jako jediné pole, včetně města a poštovního směrovacího čísla. Při importu těchto

prvků je možná budeme chtít umístit do samostatné tabulky, kterou lze následně propojit s hlavní tabulkou.

Následující způsob je možný pro přímé vytvoření této relace:

- 1) Kompletní tabulka se všemi informacemi o adresách je importována do databáze aplikace Base jako tabulka s názvem Addresses. Podrobnosti nalezneme v předchozích kapitolách.
- 2) Pole Postcode a Town jsou načtena dotazem, zkopírována a uložena jako samostatná tabulka Postcode_Town. Za tímto účelem se přidá pole ID a zadá se jako primární klíč s funkcí Automatická hodnota.

Zde je dotaz:

```
SELECT DISTINCT "Postcode", "Town" FROM "Addresses"
```

- 3) Do tabulky Addresses je přidáno nové pole s názvem Postcode_ID.
- 4) Pomocí **Nástroje > SQL** se provede aktualizace této tabulky:

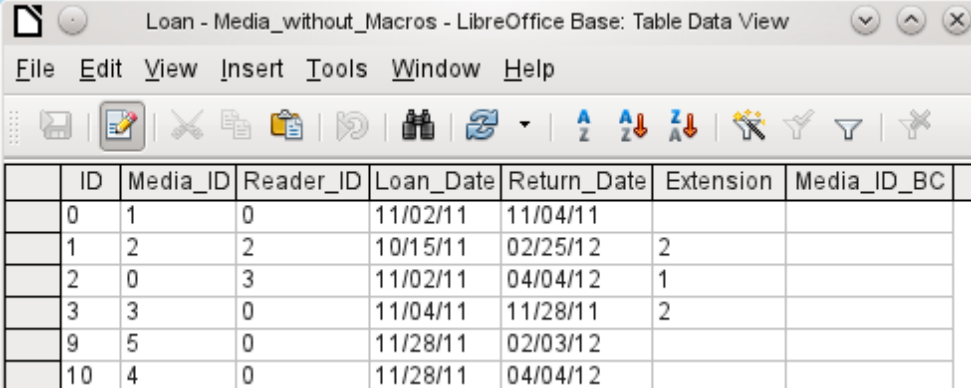
```
UPDATE "Addresses" AS "a" SET "a"."Postcode_ID" = (SELECT "ID" FROM "Postcode_Town" WHERE "Postcode" | "Town" = "a"."Postcode" | "a"."Town")
```
- 5) Otevře se tabulka Addresses k úpravám a odstraní se pole Postcode a Town. Tato změna se uloží a tabulka se opět zavře.

Tím se tabulky oddělí, aby bylo možné vytvořit relaci 1:n mezi tabulkou Postcode_Town a tabulkou Addresses. Tento vztah je definován pomocí **Nástroje > Relace**.

Podrobnosti o kódu SQL nalezneme také v kapitole 5, Dotazy.

Problémy s těmito metodami zadávání dat

Zadání pouze pomocí tabulky nebere v úvahu odkazy na jiné tabulky. To je zřejmé z příkladu půjčovny médií.



	ID	Media_ID	Reader_ID	Loan_Date	Return_Date	Extension	Media_ID_BC
	0	1	0	11/02/11	11/04/11		
	1	2	2	10/15/11	02/25/12	2	
	2	0	3	11/02/11	04/04/12	1	
	3	3	0	11/04/11	11/28/11	2	
	9	5	0	11/28/11	02/03/12		
	10	4	0	11/28/11	04/04/12		

Obrázek 179: Část tabulky Loan

Tabulka Loan se skládá z cizích klíčů pro vypůjčovanou položku (Media_ID) a odpovídajícího čtenáře (Reader_ID) a z data výpůjčky (Loan_Date). Do tabulky proto musíme v okamžiku výpůjčky zadat dvě číselné hodnoty (číslo média a číslo čtenáře) a datum. Primární klíč je automaticky zadán do pole ID. Zda čtenář skutečně odpovídá číslu, není zřejmé, pokud není současně otevřena druhá tabulka pro čtenáře. Není také zřejmé, zda byl předmět zapůjčen se správným číslem. Zde se musí výpůjčka spolehnout na štítek na položce nebo na jinou otevřenou tabulku.

Toho všeho lze mnohem snadněji dosáhnout pomocí formulářů. Zde lze uživatele a média vyhledat pomocí ovládacích prvků seznamu. Ve formulářích jsou názvy uživatele a položky viditelné a jejich číselné identifikátory jsou skryté. Kromě toho může být formulář navržen tak, že se nejprve vybere uživatel, pak datum výpůjčky a každé sadě médií se přiřadí toto jedno datum podle čísla. Jinde lze tato čísla zviditelnit přesně odpovídajícími popisy médií.

Přímý vstup do tabulek je užitečný pouze pro databáze s jednoduchými tabulkami. Jakmile existují vztahy mezi tabulkami, je lepší použít speciálně navržený formulář. Ve formulářích lze tyto vztahy lépe řešit pomocí podformulářů nebo seznamových polí.



LibreOffice
Community



Příručka aplikace Base 7.3

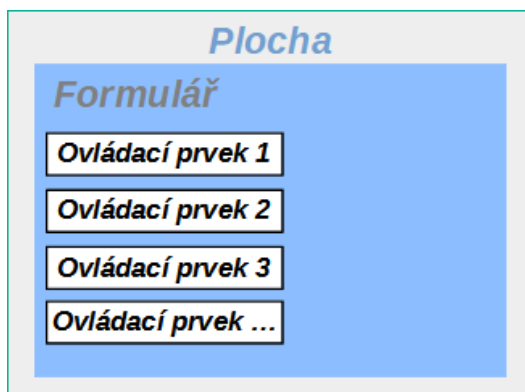
Kapitola 4 *Formuláře*

Formuláře usnadňují zadávání dat

Formuláře se používají v případech, kdy je přímý zápis do tabulky nepohodlný, k rychlému zachycení chyb při zadávání dat nebo v případech, kdy příliš mnoho tabulek znemožňuje přímou správu dat.

✓ Poznámka

Formulář v aplikaci Base je struktura neviditelná pro uživatele. V rámci aplikace Base umožňuje kontakt s databází. To, co je pro uživatele viditelné, je sada ovládacích prvků, které slouží k zadávání nebo zobrazování textu, čísel atd. Tyto ovládací prvky jsou v grafickém uživatelském rozhraní rozděleny do různých typů.



Obrázek 180: Schematické znázornění jednoduchého formuláře

Termín *Formulář* má dva významy. Může znamenat celý obsah vstupního okna, které se používá pro správu dat jedné nebo více tabulek. Takové okno může obsahovat jeden nebo více hlavních formulářů a každý z nich může obsahovat podformuláře. Pro tyto dílčí oblasti se také používá slovo *Formulář*. Z kontextu by mělo být zřejmé, o jaký význam se jedná, aby nedošlo k nedorozumění.

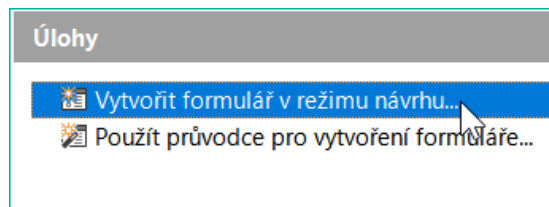
Vytváření formulářů

Nejjednodušším způsobem vytvoření formuláře je použití Průvodce formulářem. Jeho použití k vytvoření formuláře je popsáno v kapitole 8, *Začínáme s Base*, v příručce *Začínáme s aplikací Base*. V této kapitole je také vysvětleno, jak můžeme formulář po použití Průvodce dále upravovat.

Tato příručka popisuje vytvoření formuláře bez použití Průvodce. Popisuje také vlastnosti různých typů ovládacích prvků ve formuláři.

Jednoduchý formulář

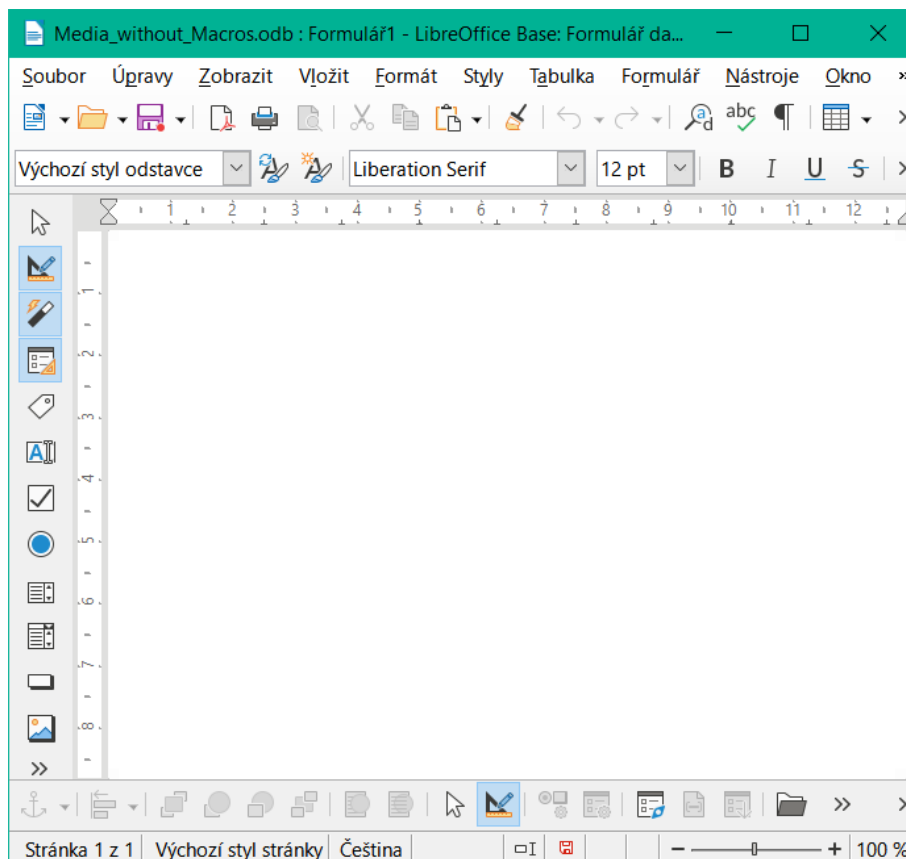
Začneme pomocí úlohy **Vytvořit formulář v zobrazení návrhu** v oblasti Formuláře hlavního okna aplikace Base.



Obrázek 181: Vytvoření nového formuláře v režimu návrhu

Tím se vyvolá Editor formuláře a formulář se zobrazí v okně režim Návrhu (obrázek 182).

Nástrojová lišta Ovládací prvky formuláře je ukotvena na levé straně. Nástrojová lišta Návrh formuláře (obrázek 184) je zakotven ve spodní části. Pokud se tyto nástrojové lišty nezobrazí automaticky, zobrazíme je pomocí **Zobrazení > Nástrojové lišty**. Bez těchto nástrojových lišt nelze formulář vytvořit.



Obrázek 182: Okno Databázový formulář podporuje návrh formulářů v režimu Návrh.

V prázdné oblasti je zobrazena mřížka bodů. Tato mřížka nám pomůže přesně umístit ovládací prvky, zejména ve vzájemném vztahu. Symboly na pravém konci nástrojové lišty Návrh formuláře ukazují, že mřížka je viditelná a aktivní. Poslední tři symboly by měly být viditelné a aktivní. Pokud nejsou všechny, klepneme na ty, které nejsou.

i Tip

Editor formuláře pracuje vlastně jako dokument Writer. Pro úpravy formuláře jsou k dispozici všechny prostředky Writeru, včetně stylů, přímého formátování, kontroly pravopisu, objektů a dalších.

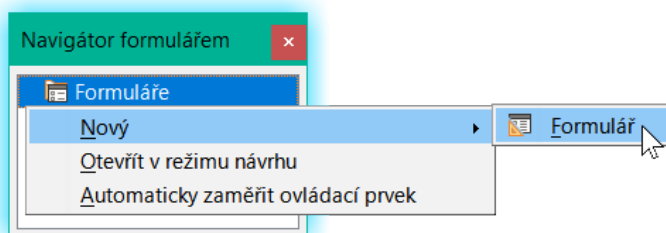
Nástrojové lišty pro návrh formulářů

Na prázdné stránce se vytvoří formulář. To lze provést dvěma způsoby:

- K nastavení formuláře použijeme nástroj Navigátor formuláře nebo.
- Navrhujeme ovládací prvky formuláře a nastavíme formulář pomocí místní nabídky.

Nastavení formuláře pomocí nástroje Navigátor formulářem

Chceme-li zobrazit nástroj Navigátor formuláře, klepneme na tlačítko Navigátor formuláře (na obrázku 184). Zobrazí se okno (obrázek 183); v něm je zobrazena pouze jedna složka označená jako Formuláře. Jedná se o nejvyšší úroveň oblasti, kterou upravujeme. Lze sem umístit několik formulářů.



Obrázek 183: Použití Navigátoru formuláře k vytvoření nového formuláře



Obrázek 184: Nástrojová lišta Návrh formuláře

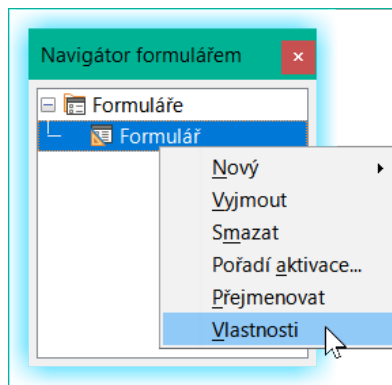
V okně Navigátor formulářů (obrázek 183) klepneme pravým tlačítkem myši na položku Formuláře a otevřeme místní nabídku. Chceme-li vytvořit nový formulář, zvolíme **Nový > Formulář**. Další volby v místní nabídce (Otevřít v režimu návrhu a Automatické zaostření ovládaní) odpovídají tlačítkům na obrázku 186; probereme je později.

i Tip

Chceme-li spustit formulář s kurzorem v prvním poli, použijeme volbu **Automaticky zaměřit ovládací prvek**. To, co se považuje za první prvek, je určeno aktivační posloupností formuláře.

Formulář nese výchozí název *Form*. Tento název můžeme změnit okamžitě nebo později. Nemá žádný význam, pokud nepotřebujeme přistupovat k některé části formuláře pomocí maker. Jediné, co je třeba zajistit, je, aby se dva prvky se stejným názvem nevyskytovaly na stejné úrovni ve stromu složek.

Místní nabídka formuláře (obrázek 185) umožňuje vytvářet vlastnosti formuláře.

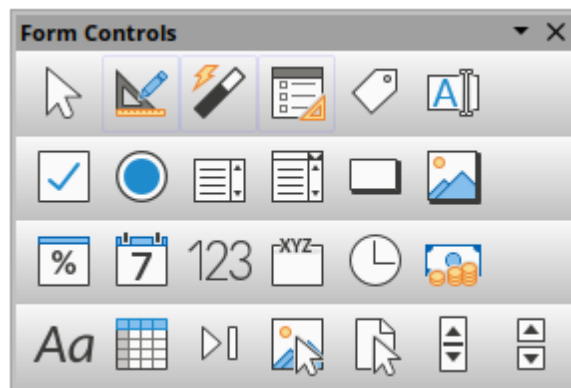


Obrázek 185: Přístup k dialogovému oknu Vlastnosti formuláře

Vytvoření formuláře pomocí formulářového pole

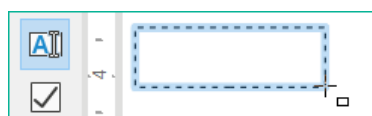
Nástrojová lišta Ovládací prvky formuláře (obrázek 186) obsahuje tlačítka, pomocí kterých lze vytvářet ovládací prvky (popisek plus pole). Ne všechna tlačítka jsou přímo viditelná na levé straně formuláře. Klepnutím na spodní tlačítko (») zobrazíme ostatní viditelná tlačítka. Chceme-li zobrazit všechna viditelná i neviditelná tlačítka, klepneme pravým tlačítkem myši na libovolnou část nástrojové lišty a vybereme možnost **Viditelná tlačítka**. Viditelná tlačítka jsou zaškrtnutá, nezaškrtnutá nejsou viditelná.

To umožňuje vybrat tlačítka, která chce uživatel pravidelně používat, a zároveň odstranit nežádoucí tlačítka. Příklad: Tlačítko Ovládací prvek tabulky není viditelné, ale lze jej zobrazit tak, že jej nejprve vyhledáme v seznamu všech tlačítek a poté jej zaškrtneme.



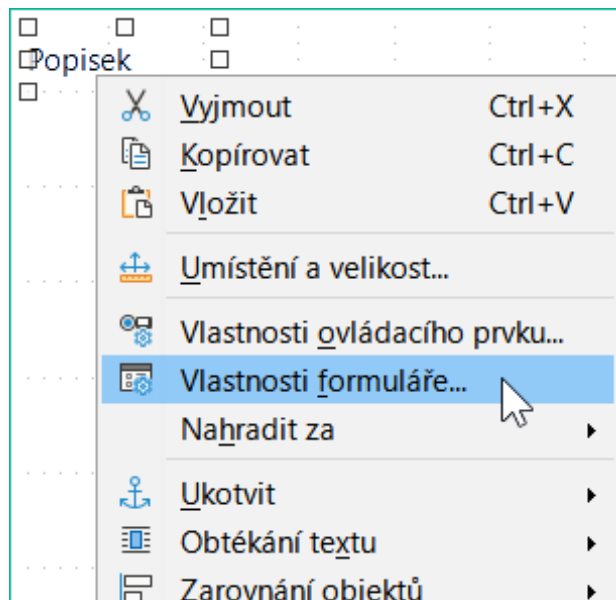
Obrázek 186: Nástrojová lišta Ovládací prvky formuláře

Výběrem ovládacího prvku formuláře se automaticky vytvoří formulář. Předpokládejme například, že vybereme textové pole: kurzor změní tvar a na bílé ploše formuláře se může vykreslit obdélníkový tvar. Poté se na tečkovaném povrchu formuláře objeví textové pole.



Obrázek 187: Vložení textového pole

Nyní můžeme vytvořit formulář klepnutím pravým tlačítkem myši a použitím místní nabídky ovládacího prvku (obrázek 188).



Obrázek 188: Přístup k dialogu Vlastnosti formuláře z místní nabídky ovládacího prvku formuláře

Výběrem možnosti nabídky **Formulář** (zvýrazněné na obrázku) nastavíme vlastnosti právě vytvořeného formuláře. Formulář má výchozí název Form.

Externí formuláře

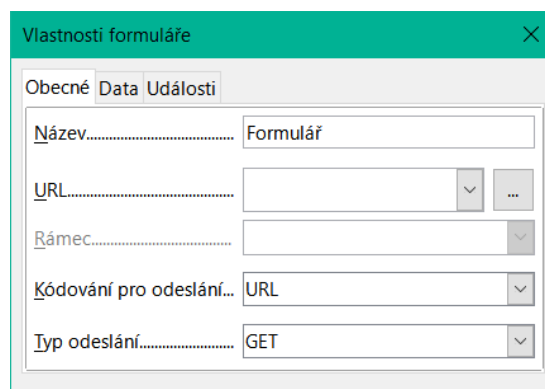
Kromě formulářů vytvořených v aplikaci Base je možné vytvářet formuláře také v aplikaci Writer nebo Calc. Ty jsou popsány v kapitole 7, Propojení s databázemi.

Vlastnosti formuláře

Po vyvolání vlastností formuláře pomocí místní nabídky v Navigátoru formulářem nebo místní nabídky ovládacího prvku formuláře se zobrazí okno Vlastnosti formuláře. Má tři karty: *Obecné*, *Data* a *Události*.

Karta Obecné

Zde můžeme změnit Název formuláře. Kromě toho existují možnosti designu, které nemají uvnitř aplikace Base žádný význam. Ukazují pouze obecnější možnosti návrhu pomocí editoru formulářů: při vytváření webového formuláře je budeme muset použít.



Obrázek 189: Dialogové okno Vlastnosti formuláře (karta Obecné)

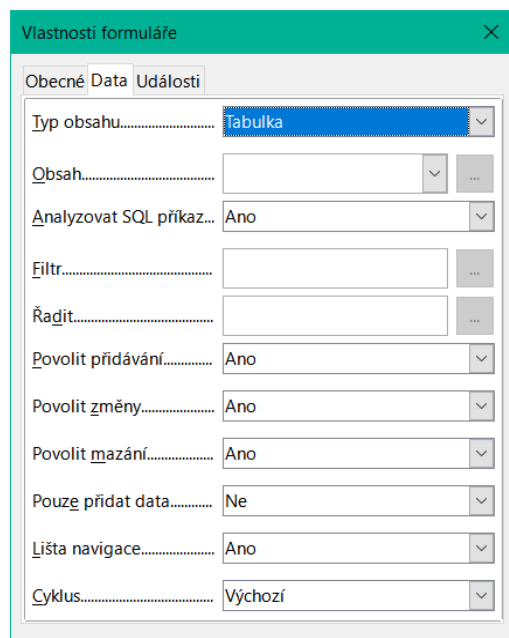
URL: Cílová adresa pro data.

Rámeček: Část cílové webové stránky, která má být v případě potřeby adresována.

Kódování pro odeslání: kromě běžného kódování znaků pro převod na adresu URL zde můžeme zadat kódování textu a kódování více částí (například pro přenos dat).

Typ odeslání: GET (viditelné přes URL připojené k názvu souboru; často se s ním můžeme setkat na webu, pokud používáme vyhledávač) nebo POST (není viditelné; vhodné pro velké objemy dat).

Karta Data



Obrázek 190: Dialogové okno Vlastnosti formuláře (karta Data)

Pro vytváření interních formulářů v aplikaci Base je tato karta nejdůležitější. Zde můžeme nastavit následující počáteční vlastnosti formuláře:

Typ obsahu: Vybereme si mezi **tabulkou**, **dotazem** a příkazem **SQL**. Zatímco tabulku lze vždy použít pro zadávání dat do formuláře, v případě dotazu (více informací nalezneme v kapitole 5, Dotazy) nebo přímého zadání příkazu SQL to vždy neplatí. Zde se jedná o dotaz, který není viditelný v kontejneru dotazů Base, ale má v zásadě stejnou strukturu.

Obsah: Podle toho, zda byla výše vybrána tabulka nebo dotaz, se zobrazí všechny dostupné tabulky a dotazy. Pokud má být vytvořen příkaz SQL, můžeme vyvolat Editor dotazů pomocí elipsy (...) vpravo vedle pole Obsah.

Analyzovat SQL příkaz: Pokud by analýza příkazů SQL neměla být povolena (například proto, že používáme kód, který grafické rozhraní nemůže správně zobrazit), měli bychom zde zvolit **Ne**. To však zabrání formuláři v přístupu k podkladovým datům pomocí filtru nebo třídění.

Filtr: Zde můžeme nastavit filtr. Chceme-li získat nápovědu, klepneme na tlačítko vpravo vedle pole. Viz také kapitola 3, Tabulky.

Třídění: Zde můžeme nastavit třídění dat. Chceme-li získat nápovědu, klepneme na tlačítko vpravo vedle pole. Viz také kapitola 3, Tabulky.

Povolit přidávání: Má být povoleno zadávání nových údajů? Ve výchozím nastavení je tato hodnota nastavena na **Ano**.

Povolit změny: Měla by být povolena editace dat? Ve výchozím nastavení také **Ano**.

Povolit mazání: Ve výchozím nastavení je povoleno i mazání dat.

Pouze přidat data: Pokud zvolíme tuto možnost a pro ostatní volby zadáme **Ne**, zobrazí se vždy prázdný formulář. Ke stávajícím datům, která nelze upravovat ani prohlížet, nebude umožněn přístup. Pokud je však vybrána možnost **Povolit přidávání** a pouze přidat data, lze data do pole přidat. Jakmile jsou však data uložena, nejsou již viditelná ačkoliv byla zapsána do tabulek.

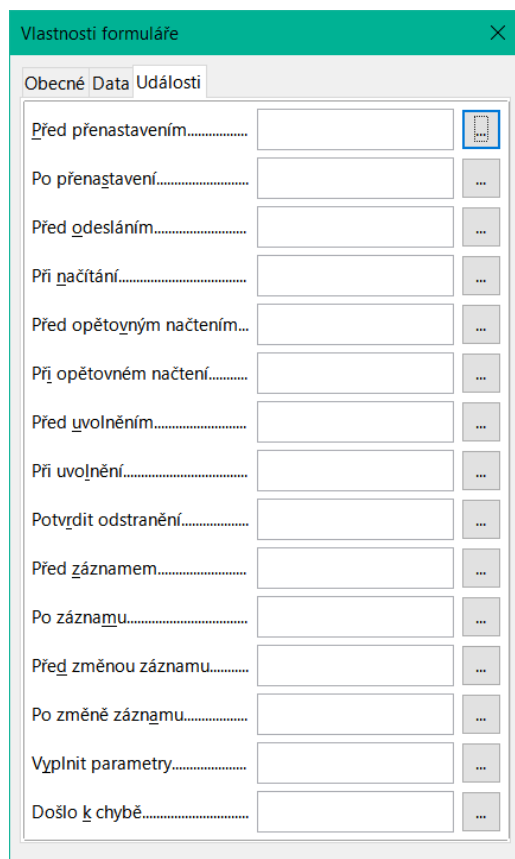
Tip

Tato vlastnost může být užitečná v případě, kdy je třeba do databáze přidat data, ale osoba, která je zadává, je nesmí pouze zadávat, ale ani nijak upravovat. Úpravu je třeba přenechat jiné osobě, která má k dispozici jiný formulář, který to umožňuje.

Lišta navigace: Zobrazení Lišty navigace ve spodní části obrazovky lze zapnout nebo vypnout. Existuje také možnost, aby se v případě, že máme podformulář, vždy zobrazil navigační panel pro hlavní formulář, takže aktivace tohoto panelu nástrojů ovlivní pouze hlavní formulář. Toto nastavení navigačního panelu se netýká interního navigačního panelu nástrojů, který lze v případě potřeby přidat jako ovládací prvek formuláře.

Cyklus: Výchozí volbou pro databáze aplikace Base je, že po zadání do posledního pole formuláře se klávesou Tab přejde na první pole dalšího záznamu – tj. vytvoří se nový záznam. U databází to má stejný účinek jako *Všechny záznamy*. Pokud naopak zvolíme *Aktivní záznam*, kurzor se bude pohybovat pouze v rámci záznamu; jakmile dosáhne posledního pole, přeskočí zpět na první pole tohoto záznamu. *Aktuální stránka* se týká zejména formulářů HTML. Kurzor přeskočí z konce formuláře na další formulář na této stránce dále.

Karta Události



Obrázek 191: Dialogové okno Vlastnosti formuláře (karta Události)

Události mohou spouštět makra. Klepnutím na tlačítko vpravo (...) lze k události připojit makra.

Obnovit: Formulář se vyprázdní od všech nových záznamů, které ještě nebyly uloženy.

Před odesláním: Před odesláním dat formuláře. To má význam pouze pro webové formuláře.

Při načítání: Pouze při otevření formuláře, nikoliv při načítání nového záznamu do formuláře.

Při opětovným načtením: K tomu dochází při obnovení obsahu formuláře, například pomocí tlačítka na Liště navigace.

Při uvolnění: Zdá se, že tato možnost nefunguje. Očekávalo by se, že bude odkazovat na uzavření formuláře.

Po záznamu: Patří sem například uložení pomocí tlačítka. Při testech se tato akce pravidelně duplikuje; makra se spouštějí dvakrát za sebou.

Je to proto, že se zde provádějí různé funkce („implementace“). Názvy funkcí jsou: `org.openoffice.comp.svx.FormController` a `com.sun.star.comp.forms.ODatabaseForm`. Pokud se uvnitř makra, které používá `oForm.ImplementationName`, dotazuje na odpovídající název, může být makro omezeno na jedno spuštění.

Změna záznamu: Otevření formuláře se považuje za změnu záznamu. Kdykoli se v rámci formuláře změní jeden záznam na jiný, tato akce se provede dvakrát. Makra se proto spouštějí dvakrát po sobě. I zde můžeme rozlišit příčiny tohoto výsledku.

Vyplnit parametry: Toto makro se spustí, pokud má být v dílčím formuláři vyvolán dotaz na parametr, ale z nějakého důvodu není parametr správně přenesen z hlavního formuláře. Pokud tato událost není zachycena, bude po načtení formuláře následovat dotaz na parametry.

Došlo k chybě: Tuto událost se nepodařilo rekonstruovat.

Vlastnosti ovládacích prvků

Po vytvoření formuláře jej lze naplnit viditelnými ovládacími prvky. Některé ovládací prvky umožňují zobrazit obsah databáze nebo do ní zadávat data. Ostatní ovládací prvky slouží výhradně k navigaci, vyhledávání a provádění příkazů (interakci). Některé ovládací prvky slouží k dalšímu grafickému zpracování formuláře.

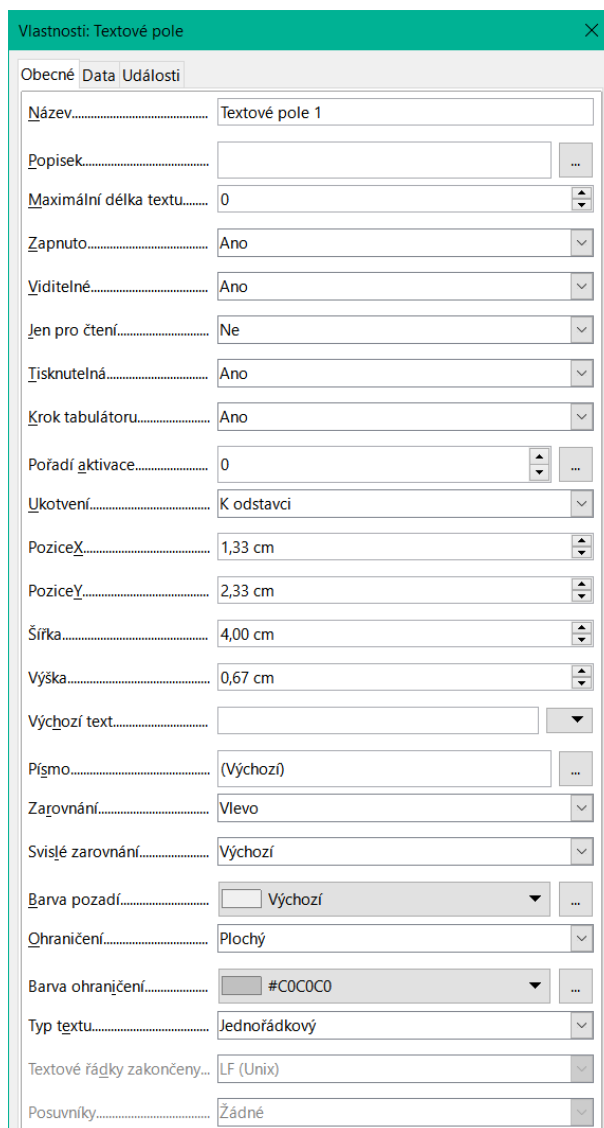
Tabulka 5: Ovládací prvky formuláře

Zadávání a zobrazování dat	
Ovládací prvek	Použití
Textové pole	Zadávání textu
Číselné pole	Zadávání čísel
Pole pro datum	Zadávání dat
Pole pro čas	Zadávání časů
Měnové pole	Číselný údaj, předpřipravený pro měnu
Formátované pole	Zobrazení a zadání s dalším formátováním, například pomocí měrných jednotek.
Seznam	Volba mezi několika různými možnostmi, také pro přenos jiných než zobrazených hodnot do databáze.
Pole se seznamem	Podobně jako pole seznamu, ale přenáší se pouze zobrazená hodnota, nebo můžeme zadat nové hodnoty ručně.
Zaškrtačivé pole	Pole Ano/Ne
Přepínače	Tlačítko volby; umožňuje vybrat z malého počtu možností.
Ovládací pole obrázku	Zobrazení obrázků z databáze a zadání obrázků do databáze pomocí výběru cesty
Pole vzorku	Zadání do přednastavené masky; omezuje možnosti zadání na určité kombinace znaků.
Ovládací prvek tabulky	Univerzální vstupní modul, který dokáže zobrazit celou tabulku. Do tohoto ovládacího prvku je integrováno mnoho z výše uvedených ovládacích prvků.
Vzhled	
Ovládací prvek	Použití
Pole popisku	Záhlaví formuláře, popis dalších ovládacích prvků
Seskupení	Rámeček například kolem sady tlačítek možností
Interakce	
Ovládací prvek	Použití
Tlačítko	Tlačítko s popiskem
Obrázkové tlačítko	Jako tlačítko, ale se zobrazeným obrázkem (grafikou).

Interakce	
Ovládací prvek	Použití
Lišta navigace	Nástrojová lišta velmi podobná tomu na spodním okraji obrazovky
Výběr souboru	Pro výběr souborů, například pro nahrání do formuláře HTML – není dále popsáno.
Rolovací tlačítko	Lze použít pouze s makrem - není dále popsáno.
Posuvník	Lze použít pouze s makrem - není dále popsáno.
Skrytý ovládací prvek	Zde lze hodnotu uložit pomocí maker a poté ji znovu načíst,

Výchozí nastavení mnoha ovládacích prvků

Stejně jako u formulářů jsou vlastnosti ovládacích prvků rozděleny do tří kategorií: *Obecné*, *Data* a *Události*. Obecné zahrnuje vše, co je pro uživatele viditelné. Kategorie Data určuje vazbu na pole v databázi. Kategorie Události řídí akce, které mohou být vázány na některé makro. V databázi bez maker nehraje tato kategorie žádnou roli.



Obrázek 192: Dialogové okno Vlastnosti (karta Obecné) pro textové pole

Karta Obecné

Název..... Textové pole 1

Popisek.....

Název ovládacího prvku musí být v rámci formuláře jedinečný – používá se pro přístup pomocí maker. [Name]

Má pole popisek? Tím se pole a štítek seskupí.

Popisek umožňuje dosáhnout pole formuláře přímo pomocí klávesové zkratky. [LabelControl]

Zapnuto..... Ano

Viditelné..... Ano

Jen pro čtení..... Ne

Tisknutelná..... Ano

Krok tabulátoru..... Ano

Pořadí aktivace..... 0

Ukotvení..... K odstavci

PoziceX..... 1,33 cm

PoziceY..... 2,33 cm

Šířka..... 4,00 cm

Výška..... 0,67 cm

Písmo..... (Výchozí)

Zarovnání..... Vlevo

Nepovolená pole nelze použít a jsou šedá. Užitečné pro ovládání pomocí maker. (Příklad: Pokud Field 1 obsahuje hodnotu, nesmí Field 2 obsahovat žádnou hodnotu; Field 2 je deaktivováno.)
[Enabled]

Obvykle ano; neviditelná pole lze použít jako mezisklad, například při vytváření kombinovaných polí s makry. Viz kapitola 9, Makra.
[EnableVisible]

Ano vyloučí jakoukoli změnu hodnoty. To je užitečné například pro automaticky generovaný primární klíč.
[ReadOnly]

Někdy je užitečné vytisknout stránku z formuláře místo samostatné sestavy. V tomto případě nemusí být nutné zobrazit všechna pole.
[Printable]

Ve formuláři se k navigaci obvykle používá klávesa Tab. Pole, které je určeno pouze pro čtení, nepotřebuje zarážku tabulátoru; může být vynecháno.
[Tabstop]

Má pole zarážku tabulátoru? Zde je zadána aktivační posloupnost v rámci formuláře.
[TabIndex]

Ukotvení grafiky v textovém poli.

Pozice od levého horního rohu vzhledem k levé straně formuláře.
[PosSize.X]

Pozice od levého horního rohu vzhledem k horní části formuláře.
[PosSize.Y]

Šířka pole.
[PosSize.Width]

Výška pole.
[PosSize.Height]

Zde lze nastavit písmo, velikost písma a efekty písma.
[Fontxxx]

Zarovnání. Zde je zadávaný text zarovnán vlevo.
[Align]

Svislé zarovnání..... Výchozí

Svislé zarovnání:
Standardní | Nahoře | Uprostřed | Dole.[VerticalAlign]

Barva pozadí..... Výchozí

Barva pozadí textového pole.
[BackgroundColor]

Ohraničení..... Plochý

Ohraničení: Bez rámce | 3D vzhled | Plochý.
[Border]

Barva ohraničení..... #C0C0C0

Pokud existuje ohraničení, lze zde nastavit jeho barvu pouze v případě, že je jako ohraničení vybráno *Plochý*.
[BorderColor]

Skrýt výběr..... Ano

Zvýrazněný text ztratí zvýraznění, když textové pole ztratí fokus.
[HideInactiveSelection]

Další informace.....

Používá se pro informace, které mají být načteny makry. Viz kapitola 9, Makra.
[Tag]

Pomocný text.....

Zobrazí se jako nápověda při najetí myši na textové pole.
[HelpText]

URL nápovědy.....

Ukazuje na soubor nápovědy, užitečný především pro HTML. Lze vyvolat pomocí klávesy F1, když je pole aktivováno.
[HelpURL]

Číselná pole, pole pro datum atd. mají navíc tyto vlastnosti.

Přesný formát..... Ano

Při zapnutém testování lze zadávat pouze čísla a desetinné čárky.
[EnforceFormat]

Posun kolečkem myši... Nikdy

Nikdy neumožňuje změny pomocí kolečka myši; Když je vybráno, umožňuje tyto změny, když je pole vybráno a myš je nad polem; Vždy znamená vždy, když je myš nad polem.
[MouseWheelBehavior]

Rolovací tlačítko..... Ano

V pravé části pole je symbol otáčení.
[Spin]

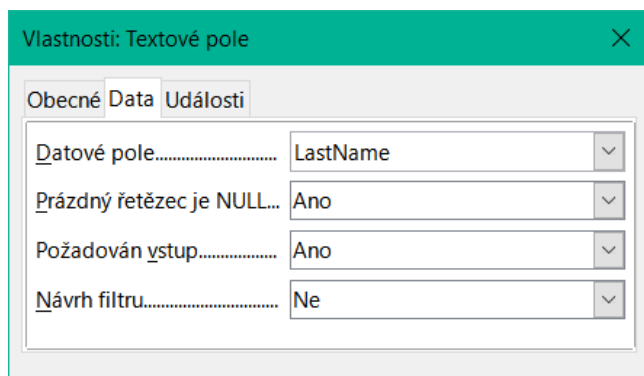
Opakovat..... Ne

Pokud je stisknuta a podržena šipka otáčení, určuje se, zda se má údaj v poli zvýšit o další hodnotu.
[Repeat]

Prodleva..... 50 ms

Určuje minimální prodlevu po stisknutí tlačítka myši, která spustí opakování.
[RepeatDelay]

Karta Data



Obrázek 193: Dialogové okno Vlastnosti (karta Data) pro textové pole

Datové pole: Zde vytvoříme vazbu s tabulkou, na které je formulář založen.
[Model.DataField]

Prázdný řetězec je NULL: Zda se má prázdný řetězec považovat za (NULL) nebo se má obsah jednoduše vymazat.

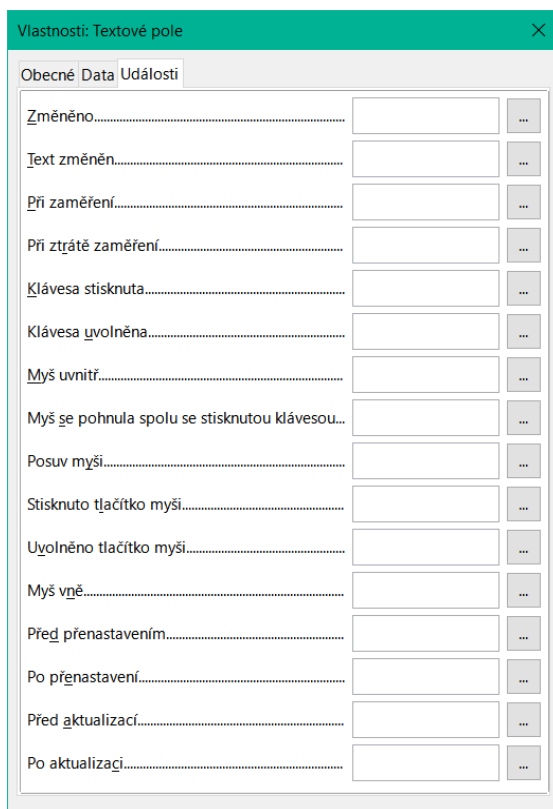
Požadovaný vstup: Tato podmínka by měla odpovídat podmínce v tabulce. Pokud uživatel nezadal žádnou hodnotu, grafické uživatelské rozhraní ho vyzve k zadání.
[Model.InputRequired]

Návrh filtru: Obsah tohoto pole je při filtrování dat dočasně uložen jako návrh.
[Model.UserValueFilterProposal]

Upozornění

Upozornění – při velkém obsahu může tato volba spotřebovat hodně úložného prostoru.

Karta Události



Obrázek 194: Dialogové okno Vlastnosti (karta Události) pro textové pole

Změněno: Tato událost nastane, když je ovládací prvek upraven a poté ztratí aktivaci. Pokud se přepneme přímo na jiný záznam, událost se ztratí. Za těchto okolností je změna uložena, aniž by byla dříve zjištěna. [com.sun.star.lang.EventObject]

Text změněn: Odkazuje na obsah, který může být ve skutečnosti textový, číselný nebo jakýkoli jiný. Objeví se po zadání každého dalšího znaku. [com.sun.star.awt.TextEvent]

Při zaměření: Kurzor vstoupí do pole.

Upozornění

Makro v žádném případě nesmí na obrazovce vytvořit dialogové okno se zprávou; klepnutí v takovém dialogu způsobí, že formulářové pole ztratí zaměření a poté jej znovu získá, čímž se makro opět spustí. Vytvoří se smyčka, kterou lze přerušit pouze pomocí klávesnice.

Při ztrátě zaměření: Kurzor opustí pole. To může vést ke stejnému druhu souhry, kdy zpracování události způsobí její opakování.

Klávesa: Odkazuje na klávesnici. Klávesa se zadává například při pohybu ve formuláři pomocí klávesy Tab. To způsobí, že pole získá zaměření. Poté se klávesa uvolní.

Událost se předává pomocí `keyCode` nebo `KeyChar` uvolněné klávesy (písmeno, číslo, speciální klávesa). [com.sun.star.awt.KeyEvent]

Myš: Samozřejmě; tyto události se odehrají pouze tehdy, pokud se myš nachází nebo již nacházela v poli ("mimo" odpovídá javascriptovému `onMouseOut`). [com.sun.star.awt.MouseEvent]

Přenasavení: Ve formuláři se vyprázdní všechna data (při vytváření nového záznamu) nebo se vrátí do původního stavu (při editaci existujícího záznamu). U formulářového pole se tato událost spustí pouze tehdy, když je zadání dat zrušeno pomocí tlačítka na liště navigace. [com.sun.star.lang.EventObject] Při prvním načtení formuláře se postupně vyskytnou dvě události *Před přenasavením* a *Po přenasavení*, a to ještě předtím, než je formulář k dispozici pro zadávání.

Aktualizace: Pokud je událost vázána na ovládací prvek formuláře, aktualizace proběhne při ztrátě zaměření a přeskočení na jiný ovládací prvek formuláře po změně obsahu pole. Změny ve formuláři jsou přijaty a zobrazeny. Při uzavření formuláře dojde postupně ke dvěma událostem *Před aktualizací* a *Po aktualizaci*. [com.sun.star.lang.EventObject]

Textové pole

Stejně jako vlastnosti uvedené v části "Výchozí nastavení mnoha ovládacích prvků". nad, mohou mít textová pole následující další vlastnosti:

Karta Obecné

Maximální délka textu..... 0

Pokud je tato hodnota 0, vstup není povolen. Obvykle se zde používá délka databázového pole, kterému textové pole odpovídá.
[MaxTextLen]

Výchozí text.....

Měl by být výchozí text vložen do prázdného pole? Tento text musí být vymazán, pokud má být úspěšně proveden jakýkoli další záznam.
[DefaultText]

Typ textu..... Jednořádkový

Možné typy: (poslední dva typy se liší v chování při stisku tabulátoru a navíc pole se vzorem nelze vázat na databázi): Jednořádkový | Víceřádkový | Víceřádkový s formátováním. Svislé zarovnání není aktivní pro víceřádková pole.
[MultiLine]

Textové řádky zakončeny... CR+LF (Windows)

Unix nebo Windows? To se týká především konců řádků. Interně jsou řádky systému Windows ukončeny dvěma řídicími znaky (CR a LF).
[LineEndFormat]

Posuvníky..... Žádné

Pouze pro víceřádková pole:
Vodorovné | Svislé | Obojí.[HScroll], [VScroll]

Znak hesla.....

Aktivní pouze pro jednořádková pole. Změní znaky tak, aby se zobrazovaly pouze body.
[EchoChar]

Karta Data

Nic významného.

Karta Události

Nic významného.

Číselné pole

Kromě již popsaných vlastností existují následující vlastnosti:

Karta Obecné

Min. hodnota..... -1000000,00

Minimální hodnota pole. Měla by souhlasit s minimální hodnotou definovanou v tabulce.
[ValueMin]

Max. hodnota..... 1000000,00

Maximální hodnota.
[ValueMax]

Zvýšit/snížit hodnoty... 1

Přírůstek rolování při použití kolečka myši nebo v rámci rolovacího pole.
[ValueStep]

Výchozí hodnota.....

Hodnota zobrazená při vytváření nového záznamu.
[DefaultValue]

Desetinná přesnost..... 2

Počet desetinných míst, pro celá čísla nastaveno na 0.
[DecimalAccuracy]

Oddělovač tisíců..... Ne

Oddělovač tisíců, obvykle čárka.
[ShowThousandsSeparator]

Karta Data

Neexistuje žádná kontrola, zda pole může být NULL. Pokud neexistuje žádný záznam, bude pole NULL a nikoli 0.

Žádný návrh filtru se nenabízí.

Karta Události

Událost *Změněno* není přítomna. Změny je třeba provádět pomocí události *Text změněn* (slovo *text* zde nelze brát doslova).

Pole pro datum

Stejně jako vlastnosti popsané v části "Výchozí nastavení mnoha ovládacích prvků". nad, je třeba si uvědomit následující.

Karta Obecné

Min. datum..... 01.01.1800

Minimální hodnota pole, kterou lze vybrat pomocí rozevřacího seznamu kalendář.
[DateMin]

Max. datum..... 31.12.2200

Maximální hodnota.
[DateMax]

Formát data..... Standardní (krátké)

Zkrácená forma jako 10.02.12 nebo různé formy s použitím '/' místo '.' nebo '-' v americkém stylu.
[DateFormat]

Výchozí datum.....

Zde můžeme zadat doslovné datum, nikoli však (zatím) aktuální datum (Today) v okamžiku otevření formuláře.
[DefaultDate]

Rozbalovací..... Ne

Součástí může být měsíční kalendář pro výběr dat.
[DropDown]

Karta Data

Neexistuje žádná kontrola, zda pole může být NULL. Pokud neexistuje žádný záznam, bude pole NULL a nikoli 0. Žádný návrh filtru se nenabízí.

Karta Události

Událost *Změněno* není přítomna. Změny je třeba provádět pomocí události *Text změněn* (slovo *text* zde nelze brát doslova).

Pole pro čas

Stejně jako vlastnosti uvedené v části "Výchozí nastavení mnoha ovládacích prvků". nad, jsou k dispozici následující vlastnosti.

Karta Obecné

Min. čas..... 00:00:00

Minimální hodnota pole, ve výchozím nastavení nastavena na 0.
[TimeMin]

Max. čas..... 23:59:59

Maximální hodnota, standardně nastavená na 1 sekundu před 24:00.
[TimeMax]

Formát času..... 13:45

Krátká forma bez sekund, dlouhá forma se sekundami a také 12hodinové formáty s AM a PM.
[TimeFormat]

Výchozí čas..... 00:00:00

Můžeme nastavit pevný čas, ale ne (zatím) skutečný čas uložení formuláře.
[DefaultTime]

Karta Data

Neexistuje žádná kontrola, zda pole může být NULL. Pokud neexistuje žádný záznam, bude pole NULL a nikoli 0.

Žádný návrh filtru se nenabízí.

Karta Události

Událost *Změněno* není přítomna. Změny je třeba provádět pomocí události *Text změněn* (slovo *text* zde nelze brát doslova).

Měnové pole

Kromě vlastností již uvedených v části "Výchozí nastavení mnoha ovládacích prvků". nad, jsou k dispozici následující vlastnosti:

Karta Obecné

Min. hodnota, Max. hodnota, Zvýšit/snížit hodnoty, Výchozí hodnota, Desetinná přesnost a Oddělovač tisíců. odpovídají obecným vlastnostem uvedeným v části "Číselné pole". nad. Kromě nich existuje pouze:

Symbol měny..... Kč

Symbol je zobrazen, ale není uložen v tabulce, která je základem formuláře.
[CurrencySymbol]

Symbol pro předponu... Ne

Má být symbol umístěn před nebo za číslem?
[PrependCurrencySymbol]

Karta Data

Neexistuje žádná kontrola, zda pole může být NULL. Pokud neexistuje žádný záznam, bude pole NULL a nikoli 0.

Žádný návrh filtru se nenabízí.

Karta Události

Událost *Změněno* není přítomna. Změny je třeba provádět pomocí události *Text změněn* (slovo *text* zde nelze brát doslova).

Formátované pole

Kromě vlastností uvedených v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou nabízeny následující vlastnosti:

Karta Obecné

Minimální a maximální hodnota a výchozí hodnota závisí na formátování. Za tlačítkem Formátování se nachází flexibilní pole, díky kterému je většina měnových a číselných polí zbytečná. Na rozdíl od jednoduchého pole měny mohou být v poli vzorku záporné částky zobrazeny červeně.



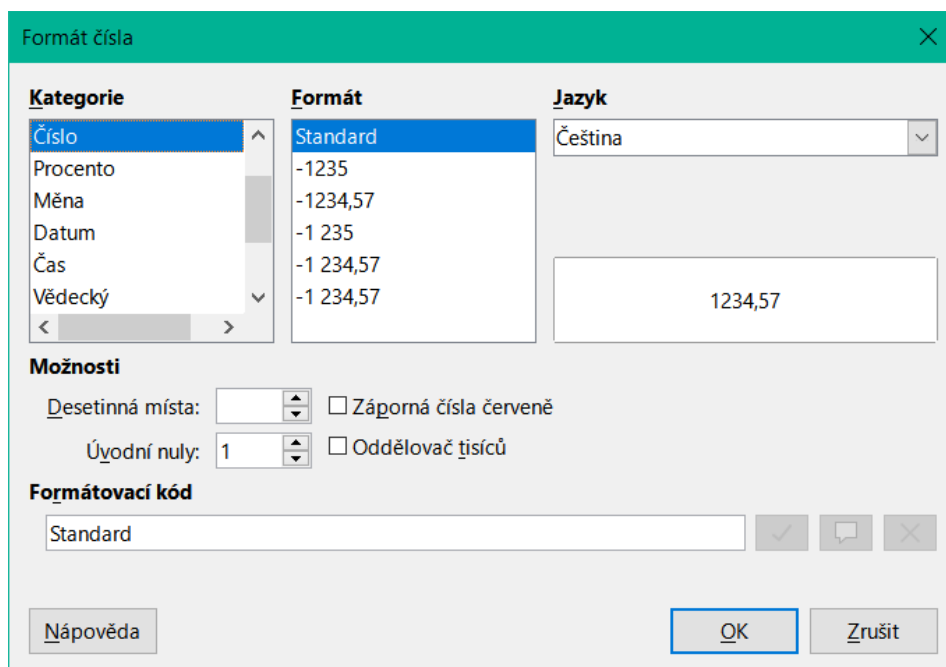
Formátování..... [] [...]

Tlačítko vpravo se třemi tečkami nabízí volbu číselných formátů, jak je to obvyklé v aplikaci Calc.
[FormatKey]

Mezi číselnými formáty lze vedle formátu Datum, Čas, Měna nebo normálního číselného formátu vidět možnosti použití polí s měnou jednotkou, například kg (viz obrázek 195). Viz také obecná nápověda k číselným kódům formátu.

Formátované pole umožňuje vytvářet a zapisovat do polí časových razítek v tabulkách pomocí jediného pole. Průvodce formulářem k tomu používá kombinaci pole data a času.

Pokud chceme do pole časového razítka zadat data ve tvaru minuty:sekundy:setiny sekundy, musíme použít makra.



Obrázek 195: Dialogové okno Formát čísla s vybraným obecným formátem čísla

Karta Data

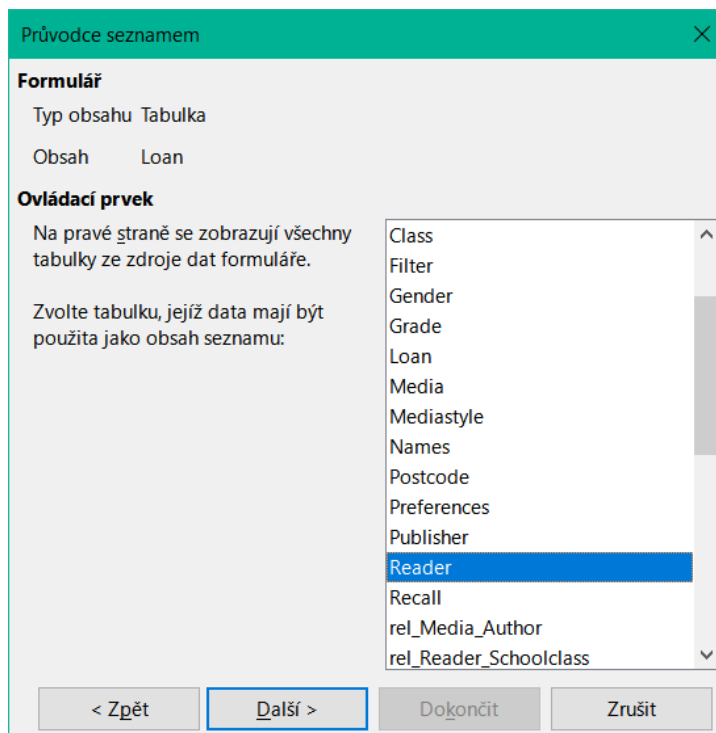
Nic zvláštního.

Karta Události

Událost *Změněno* není přítomna. Změny je třeba provádět pomocí události *Text změněn* (slovo *text* zde nelze brát doslova).

Seznam

Při vytváření pole seznamu se ve výchozím nastavení zobrazí Průvodce seznamem. Tento automatický vzhled lze v případě potřeby vypnout pomocí ikony **Přepnout průvodce ovládacími prvky** na nástrojové liště Ovládací prvky formuláře (na obrázku 186).



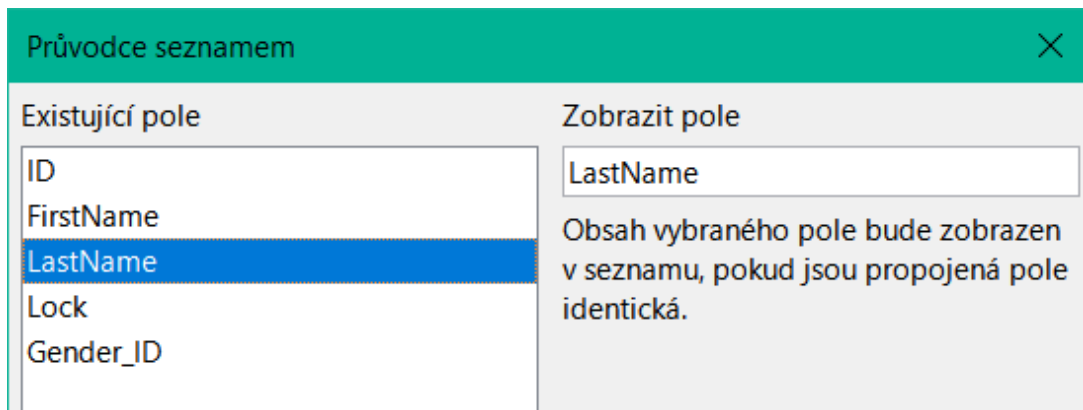
Obrázek 196: Průvodce seznamem, výběr tabulky

Průvodce

Formulář je již definován. Je svázán s tabulkou s názvem Loans. Pole seznamu zobrazuje uživateli jiné údaje, než které se skutečně přenášejí do tabulky. Tato data obvykle pocházejí z jiné tabulky v databázi, nikoli z tabulky, ke které je formulář vázán.

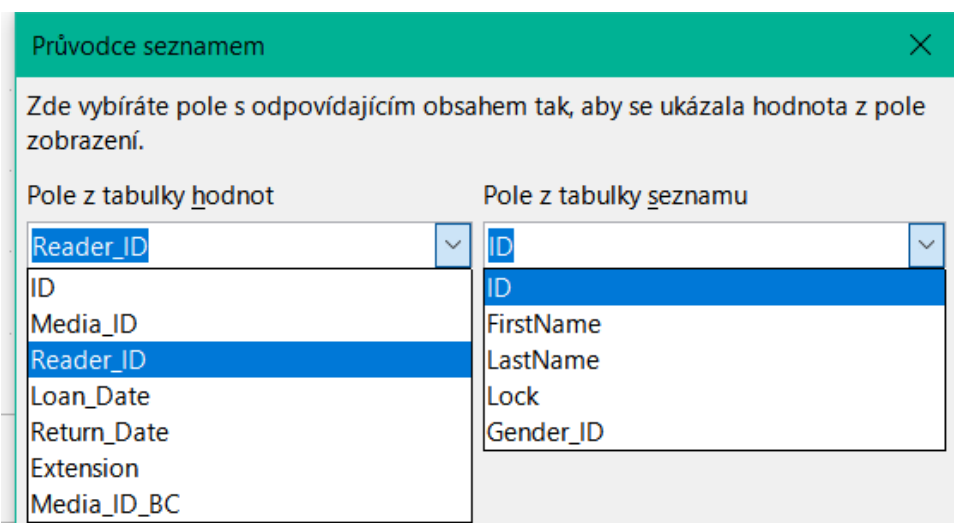
Tabulka Loans má ukázat, který čtenář si vypůjčil která média. Tato tabulka však neukládá jméno čtenáře, ale odpovídající primární klíč z tabulky Reader. Proto je základem pole seznamu právě tabulka Reader.

V seznamu by mělo být viditelné pole LastName z tabulky Reader. Slouží jako pole Display.



Obrázek 197: Průvodce pole se seznamem, výběr pole

Pole Reader_ID se vyskytuje v tabulce Loan, která je základem formuláře. Tato tabulka je zde popsána jako tabulka Value. Na toto pole musí být navázán primární klíč ID z tabulky Reader. Tabulka Reader je zde popsána jako tabulka List.



Obrázek 198: Průvodce pole se seznamem, propojení polí v tabulkách hodnot a seznamů

Pole seznamu je nyní vytvořeno s daty a výchozí konfigurací a je plně funkční.

Kromě vlastností uvedených v části "Výchozí nastavení mnoha ovládacích prvků". nad, jsou k dispozici následující vlastnosti.

Karta Obecné

Položky seznamu..... ▼

Položky seznamu již byly nastaveny pomocí Průvodce. Zde můžeme přidat další položky, které nejsou z žádné tabulky v databázi. Položky seznamu zde znamenají viditelné položky, nikoli ty, které formulář přenesou do tabulky. [StringItemList]

Rozbařlovací..... Ano

Pokud pole není zadáno jako rozevřací, zobrazí se po načtení formuláře na pravé straně pole seznamu rolovací šipky. Z pole seznamu se pak automaticky stane víceřádkové pole, ve kterém je zvýrazněna aktuální vybraná hodnota.
[Dropdown]

Počet řádek..... 20

Pokud je pole rozevřací, udává tato vlastnost maximální viditelný počet řádků. Pokud obsah přesahuje více řádků, zobrazí se při poklesu seznamu posuvník.
[LineCount]

Vícenásobný výběr..... Ne

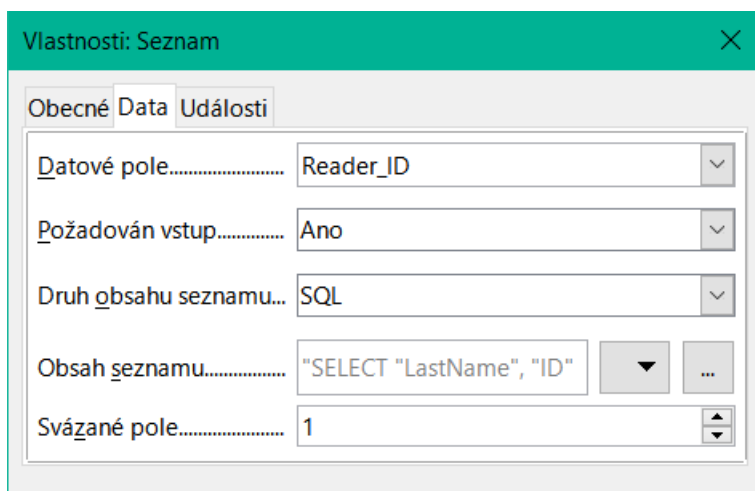
Lze vybrat více než jednu hodnotu? Ve výše uvedeném příkladu to není možné, protože se ukládá cizí klíč. Obvykle se tato funkce nepoužívá pro databáze, protože každé pole by mělo obsahovat pouze jednu hodnotu. V případě potřeby mohou makra pomoci při interpretaci více položek v poli seznamu.
[MultiSelection]
[MultiSelectionSimpleMode]

Výchozí výběr.....

Jak je z deaktivovaného tlačítka zřejmé, výchozí výběr nemá v kontextu vazby s databázovou tabulkou, jak ji vytvořil Průvodce polem seznamu, příliš smysl. Může se stát, že záznam odpovídající výchozímu výběru v příkladu v tabulce Readers již neexistuje.
[DefaultSelection]

Karta Data

Kromě obvyklých datových vlastností Datové pole a Vstupní pole jsou zde ještě další významné vlastnosti, které ovlivňují vazbu mezi zobrazenými daty a daty, která se mají zadat do tabulky, která je základem formuláře.



Vlastnosti: Seznam

Obecné Data Události

Datové pole..... Reader_ID

Požadován vstup..... Ano

Druh obsahu seznamu... SQL

Obsah seznamu..... "SELECT "LastName", "ID"

Svázané pole..... 1

Obrázek 199: Dialogové okno Vlastnosti (karta Data) pro pole se seznamem

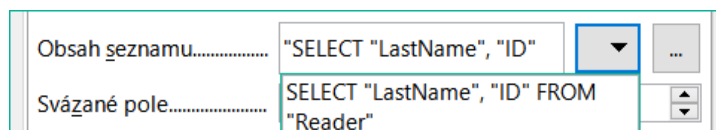
Druh obsahu seznamu: Seznam hodnot | Tabulka | Dotaz | SQL | SQL [Nativní] | Pole tabulky [ListSourceType]

Obsah seznamu: Pokud byly položky seznamu vytvořeny v části *Obecné*, zadávají se zde příslušné hodnoty, které se mají uložit. Obsah seznamu se načítá s jednotlivými položkami oddělenými pomocí *Shift + Enter*. V obsahovém poli Seznam se pak zobrazí jako "Hodnota1"; "Hodnota2"; "Hodnota3" ... Vlastnost Svázané pole je neaktivní.

Obsah seznamu Tabulka: Zde lze vybrat jednu z databázových tabulek. To je však možné jen zřídka, protože to vyžaduje, aby obsah tabulky byl strukturován tak, že první pole tabulky obsahuje hodnoty, které se mají zobrazit v poli seznamu, a jedno z následujících polí obsahuje primární klíč, který tabulka, na níž je formulář založen, používá jako cizí klíč. Pozice tohoto pole v rámci tabulky je uvedena v poli Bound Field, kde *Číslování začíná 0 pro první pole databázové tabulky*. Tato 0 je však vyhrazena pro zobrazovanou hodnotu, ve výše uvedeném příkladu Surname, zatímco 1 se vztahuje k poli ID.

Obsah seznamu Dotaz: Zde je dotaz nejprve vytvořen samostatně a uložen. Vytváření takových dotazů je popsáno v kapitole 5, Dotazy. Pomocí dotazu je možné přesunout pole ID z první pozice v podkladové tabulce na druhou pozici, kterou zde představuje vázané pole 1.

Obsah seznamu SQL: Toto pole vyplní Průvodce seznamem. Dotaz sestavený Průvodcem vypadá takto:



Obrázek 200: Typický kód SQL vygenerovaný Průvodce pole se seznamem

Dotaz je nejjednodušší možný. Pole Surname se vyskytuje na pozici 0, pole ID na pozici 1. Obojí se načítá z tabulky Reader. Vzhledem k tomu, že vázaným polem je pole 1, funguje tento vzorec SQL. Zde by mělo být přidáno *ORDER BY "LastName" ASC*. Abychom nemuseli dlouho procházet seznam, než někoho najdeme. Dalším problémem může být, že hodnota pole LastName může být stejná pro více než jednoho čtenáře. Proto musí být v zobrazení pole seznamu přidáno FirstName. Pokud existují čtenáři se stejným příjmením (LastName) a stejným jménem (FirstName), musí se zobrazit také ID primárního klíče. Informace o tom, jak to funguje, najdeme v kapitole 5, Dotazy.

Obsah seznamu SQL [Nativní]: Vzorec SQL se zadává přímo, nikoli pomocí Průvodce. Program Base dotaz nevyhodnotí. To je vhodné v případě, že dotaz obsahuje funkce, kterým by grafické uživatelské rozhraní aplikace Base nemuselo rozumět. V tomto případě se dotaz nekontroluje na přítomnost chyb. Více informací o *přímém režimu SQL* najdeme v kapitole 5, Dotazy.

Seznam obsahových polí tabulek: Zde jsou uvedeny názvy polí z tabulky, nikoli jejich obsah. Pro tabulku Reader by obsah seznamu byl ID, Given name, Surname, Lock, Gender_ID.



Poznámka

Pokud chceme pole času, které umí pracovat s časem v milisekundách, budeme potřebovat pole časové značky, jak je popsáno v části „Pole pro čas“. Reprezentace milisekund je nekompatibilní se způsobem, jakým jsou znaky sestavovány v polích seznamu. Chceme-li to obejít, musíme časové razítko převést na text:

```
SELECT REPLACE(LEFT(RIGHT(CONVERT("Required_service(??)".
"Time", VARCHAR), 15), 8), '.', ',') AS "Listcontent", "ID" FROM
"Required_service"
```

Zobrazí se minuty:sekundy:setiny.

Svázané pole: Pole seznamu zobrazují obsah, který nemusí být nutně totožný s tím, co bude uloženo ve formuláři. Obvykle se zobrazí název nebo něco podobného a odpovídající primární klíč se stane uloženou hodnotou tohoto pole.

```
SELECT "Name", "ID" FROM "Table" ORDER BY "Name"
```

Pole ID je uloženo v podkladové tabulce jako cizí klíč, Počet polí v databázích začíná nulou, takže pole s ním spojené ve formuláři má číslo 1.

Pokud místo toho vybereme možnost „0“, obsah pole Název se uloží uvnitř formuláře. V takovém případě můžeme pole ID z dotazu odstranit.

Je možné zvolit i polohu „-1“. Pak se nejedná o obsah dotazu, ale o pozici položky, která je uložena v seznamu. První záznam pak má pozici 1.

Výše uvedený dotaz vede k následujícímu výsledku:

Tabulka 6: Výsledek dotazu vybírajícího data z tabulky

Název	ID
Anneliese	2
Dorothea	3
Sieglinde	1

V polích seznamu lze vybrat pouze název. Pole seznamu je nastaveno na jméno „Dorothea“. Pro toto pole je možné uložit následující údaje:

Svázané pole=1 znamená, že je uložen „3“, obsah pole ID.

Svázané pole=0 znamená, že je uložen obsah pole Jméno „Dorothea“.

Svázané pole=-1 znamená, že je uloženo „2“, protože „Dorothea“ je v seznamu na druhém místě.

Poznámka

Změna svázaného pole na „0“ nebo „-1“ byla zavedena ve verzi LO 4.1. Dříve bylo možné vybrat pouze hodnoty ≥ 1 .

Karta Události

Kromě standardních událostí jsou k dispozici následující události:

Provést akci: Pokud je hodnota vybrána pomocí klávesnice nebo myši, pole seznamu provede tuto akci.

Změna stavu položky: Může se jednat o změnu zobrazeného obsahu pole seznamu pomocí rozbalovacího tlačítka. Může to být také klepnutí na rozevírací tlačítko pole.

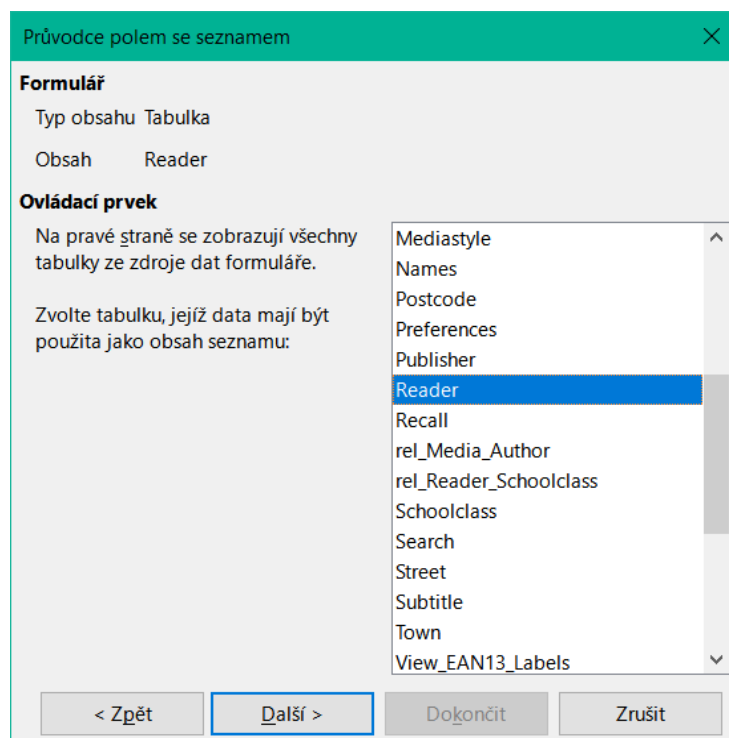
Došlo k chybě: Tato událost se nevyskytuje u seznamů.

Pole se seznamem

Jakmile je vytvořen prvek Pole se seznamem, zobrazí se ve výchozím nastavení Průvodce, stejně jako u seznamu. Toto automatické chování lze v případě potřeby vypnout pomocí tlačítka **Průvodci zapnuto/vypnuto**.

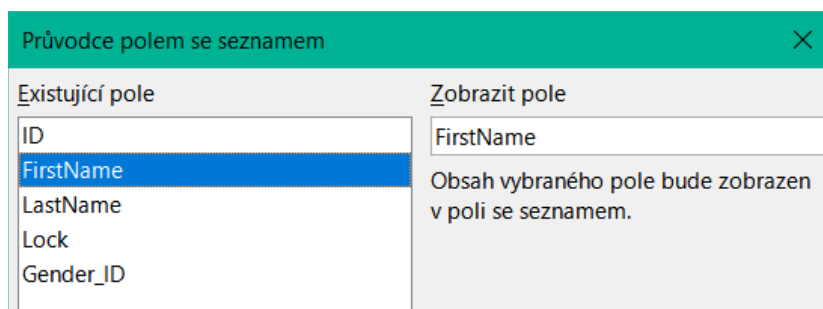
Pole se seznamem zapisují vybraný text přímo do tabulky, která je základem formuláře. Proto je v následujícím příkladu tabulka propojená s formulářem i tabulka vybraná pro ovládací prvek tabulka Reader.

Průvodce



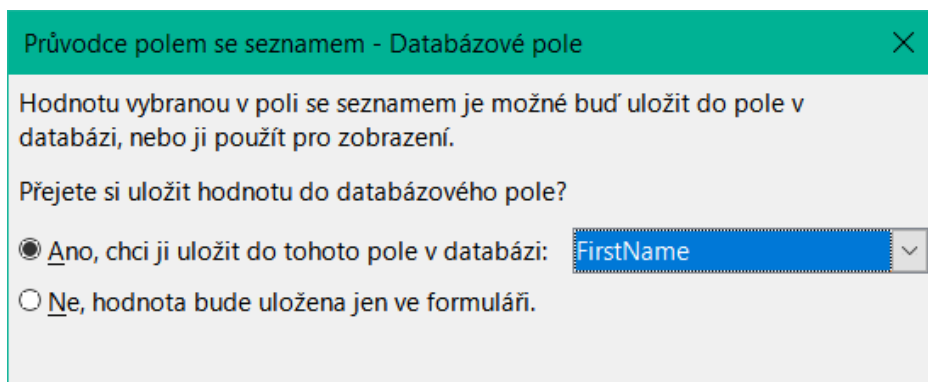
Obrázek 201: Průvodce pole se seznamem, výběr tabulky

Formulář je opět předdefinován, tentokrát s tabulkou Reader. Vzhledem k tomu, že data, která se mají zobrazit v poli se seznamem, mají být také uložena v této tabulce, je zdrojem vybraných dat pro seznam rovněž tabulka Reader.



Obrázek 202: Průvodce výběrovým polem, výběr pole

V tabulce Reader se vyskytuje pole FirstName. Tato hodnota by se měla zobrazit v poli se seznamem.



Obrázek 203: Průvodce pole se seznamem, výběr, zda se má hodnota uložit do databázového pole

Zdá se, že v databázi nemá smysl ukládat hodnotu pole se seznamem v rámci pole. Chceme načíst daná jména z tabulky Reader a také je zpřístupnit novým čtenářům, aby nebylo nutné vytvářet nové záznamy pro dané jméno, které již v databázi existuje. Ve poli se seznamem se zobrazuje křestní jméno a není nutné zadávat text.

Pokud je třeba zadat novou hodnotu, lze to snadno provést v poli se seznamem, protože pole přesně zobrazuje, co se má do podkladové tabulky formuláře zadat.

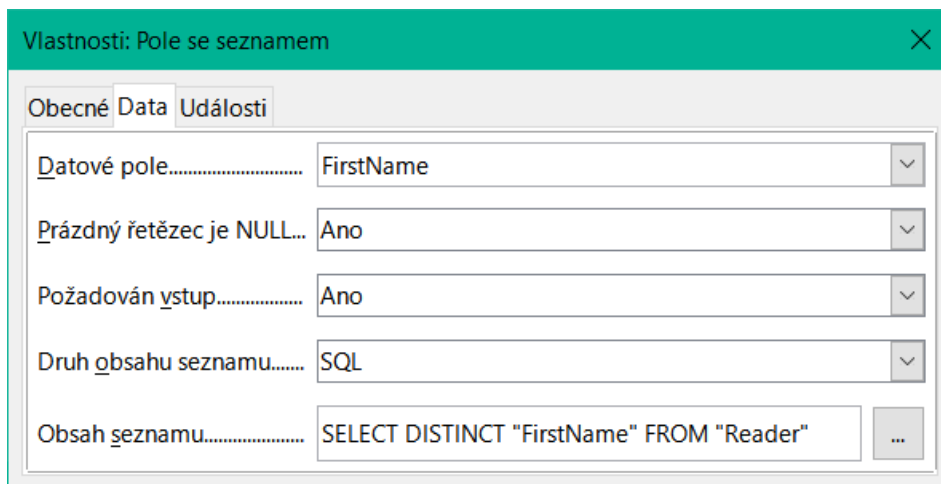
Kromě vlastností uvedených v části "Výchozí nastavení mnoha ovládacích prvků". nad a popsaných pro pole se seznamem, jsou k dispozici následující vlastnosti.

Karta Obecné

Automatické vyplňování... Ano

Při zadávání nových hodnot se zobrazí seznam odpovídajících hodnot (pokud existují) pro případný výběr. [AutoComplete]

Karta Data



Obrázek 204: Dialogové okno Vlastnosti (karta Data) pro pole se seznamem

Datová pole odpovídají stávajícímu výchozímu nastavení a nastavení pro pole seznamu. Příkaz SQL však vykazuje zvláštní vlastnost:

```
SELECT DISTINCT "FirstName" FROM "Reader"
```

Přidáním klíčového slova DISTINCT zajistíme, že se duplicitní zadaná jména zobrazí pouze jednou. Vytvoření pomocí Průvodce však opět znemožňuje třídění obsahu.

Karta Události

Události odpovídají událostem pro pole seznamu.

Zaškrťovací pole

Zaškrťovací políčko se okamžitě zobrazí jako kombinace zaškrťovacího pole a popisku políčka.

Kromě vlastností popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou k dispozici následující vlastnosti.

Karta Obecné

Popisek..... Zaškrťovací pole ▼

Popisek tohoto pole se ve výchozím nastavení zobrazuje vpravo od pole. Kromě toho jej můžeme svázat se samostatným polem popisku. [Label]

Výchozí stav..... Nevybráno ▼

Zde jsou v závislosti na volbě v poli Trojstav k dispozici až tři možnosti: Není vybráno | Vybráno | Není definováno. Není definováno odpovídá položce NULL v tabulce, která je základem formuláře. [State]

Zalomení slova..... Ne ▼

Ve výchozím nastavení není popisek rozdělen. Pokud pole není dostatečně velké, je popisek zkrácen. [MultiLine]

Obrázek..... ▼ ...

Zde můžeme zadat grafiku namísto popisku nebo jako doplněk k němu. Měli bychom si uvědomit, že tato grafika není ve výchozím nastavení vázána na dokument *.odb. U malých grafických prvků je vhodné grafiku vložit, nikoliv odkazovat. [Graphic]

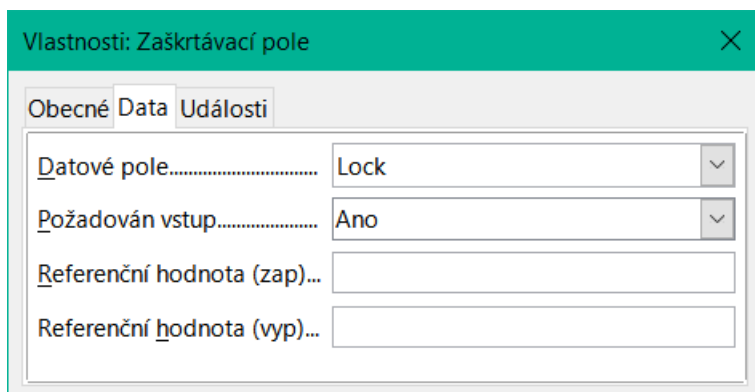
Zarovnání obrázku... Uprostřed ▼

Pokud jsme se rozhodli použít grafiku, můžeme zde nastavit její zarovnání s popiskem. [ImagePosition] (0=vlevo | 1=na střed | 2=vpravo)

Trojstav..... Ne ▼

Ve výchozím nastavení mají zaškrťovací políčka pouze dva stavy: Vybráno (Hodnota: 1) a Není vybráno (Hodnota: 0). U třístavového systému je přidána definice prázdného pole (NULL). [TriState]

Karta Data



Obrázek 205: Dialogové okno Vlastnosti (karta Data) pro zaškrťovací políčko

Zaškrťovacímu políčku lze přiřadit referenční hodnotu. Do podkladového datového pole však lze přenést pouze hodnoty 1 (pro Zapnuto) nebo 0 (pro Vypnuto) (zaškrťovací políčka fungují jako pole pro volbu Ano a Ne).

Karta Události

Pole *Změněno*, *Text změněn*, *Před aktualizací* a *Po aktualizaci* schází.

Další pole pro zaškrťovací políčko jsou *Provést akci* (viz pole *Seznam*) a *Změna stavu položky* (odpovídá *Změněno*).

Přepínač

Přepínač je podobný výše popsanému zaškrťovacímu políčku, s výjimkou jeho obecných vlastností a vnějšího (kulatého) tvaru.

Pokud je několik přepínačů ve formuláři propojeno se stejným polem tabulky, lze vybrat pouze jednu z možností.

Karta Obecné

Název skupiny.....

Přepínač je určen k použití ve skupinách. Poté lze vybrat jednu z několika možností. Proto se zde objevuje název skupiny, pod nímž lze tyto možnosti řešit.
[GroupName]

Karta Data

Viz část Zaškrťovací políčko. Zde se však zadané referenční hodnoty skutečně přenášejí do datového pole.

Karta Události

Viz část Zaškrťovací políčko.

Ovládací pole obrázku

Grafický (obrázkový) ovládací prvek spravuje zadávání a zobrazování obrázků pro databázi. Pokud má být obrázek uložen přímo, musí být podkladové datové pole binární. Může to být také textové pole, ve kterém je uložena relativní cesta k obrázku. V takovém případě je třeba dbát na to, aby cesta k obrázkům zůstala platná i v případě kopírování databáze.

! Upozornění

Obrázky by měly být při ukládání do databáze v každém případě zmenšeny. U 3MB fotografií v interní databázi HSQLDB se velmi rychle objeví chyby Javy (NullPointerException), které znemožní uložení záznamů. To může vést ke ztrátě všech záznamů, které byly zadány v aktuální relaci.

Vstup do ovládacího prvku obrázku se provede buď dvojitým klepnutím myši, čímž se otevře dialogové okno pro výběr souboru, nebo klepnutím pravým tlačítkem myši, kterým se vybere, zda má být stávající grafika odstraněna nebo nahrazena.

Grafický ovládací prvek ve výchozím nastavení nemá krok tabulátoru.

Kromě vlastností popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou k dispozici následující vlastnosti.

Karta Obecné

Obrázek.....

Zde vybraná grafika se zobrazuje pouze uvnitř ovládacího prvku, zatímco se formulář upravuje. Pro pozdější zadávání nemá žádný význam. [Graphic]

Měřítko.....

Ne: Obrázek nebude přizpůsoben poli. Pokud je příliš velký, zobrazí se v poli pouze část obrázku. Obraz není zkreslený.
Zachovejte poměr: Obraz je přizpůsoben ovládacímu prvku, ale není deformován (poměr stran je zachován).
Autom. Velikost: Obrázek je přizpůsoben ovládacímu prvku a může být zobrazen ve zkrácené podobě. [ScaleImage] [ScaleMode]

Karta Data

Nic zvláštního.

Karta Události

Schází události *Změněno*, *Text změněn*, *Před aktualizací* a *Po aktualizaci*.

Pole vzorku

Vstupní maska slouží ke kontrole vstupu do pole. Znaky jsou předem určeny pro konkrétní pozice a určují vlastnosti zadávaných znaků. Přednastavené znaky se ukládají společně se zadanými znaky.

Kromě vlastností popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou k dispozici následující vlastnosti.

Karta Obecné

Upravit masku.....

To určuje, jaké znaky lze zadat. [EditMask]

Znaková maska.....

To je to, co vidí uživatel formuláře. [LiteralMask]

Délka masky úprav určuje, kolik znaků lze zadat. Pokud zadání uživatele neodpovídá masce, je zadání při opuštění kontroly odmítnuto. Pro definici masky úprav jsou k dispozici následující znaky.

Tabulka 7: Dostupné znaky pro definici masky úprav

Znak	Význam
L	Textová konstanta. Tuto pozici nelze upravovat. Skutečný znak se zobrazí na odpovídající pozici ve znakové masce.
a	Představuje libovolné písmeno a-z/A-Z. Velká písmena se nepřevádějí na malá.
A	Představuje libovolné písmeno od A do Z. Pokud jsou zadána malá písmena, budou automaticky převedena na velká.
c	Představuje libovolný znak a-z/A-Z a číslice 0-9. Velká písmena se nepřevádějí na malá.
C	Představuje libovolná písmena A-Z a číslice 0-9. Pokud jsou zadána malá písmena, budou automaticky převedena na velká.
N	Zadávat lze pouze číslice 0-9.
x	Povoleny jsou všechny tisknutelné znaky.
X	Povoleny jsou všechny tisknutelné znaky. Pokud jsou zadána malá písmena, budou automaticky převedena na velká.

Můžeme tedy například definovat znakovou masku jako "__/__/2012" a masku upravit na "NNLNNLLLLL", aby uživatel mohl zadat pouze čtyři znaky pro datum.

Karta Data

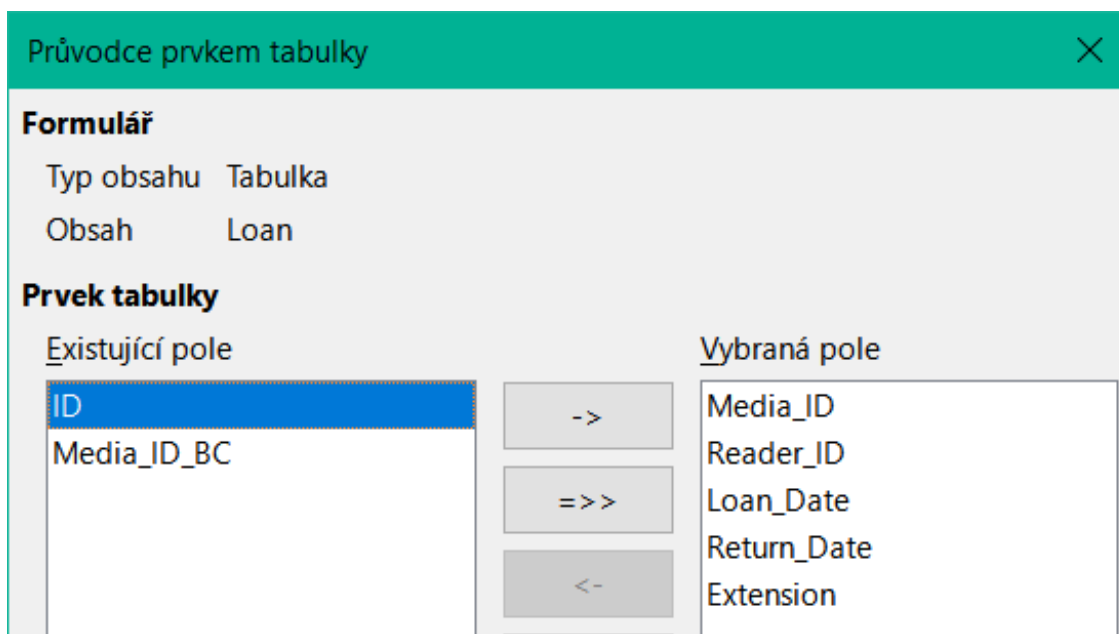
Nic zvláštního.

Karta Události

Událost *Změněno* není přítomna.

Ovládací prvek tabulky

Jedná se o nejkomplexnější kontrolu. Poskytuje tabulku, která pak může být opatřena ovládacími prvky pro jednotlivé sloupce. To umožňuje zobrazit nejen aktuální data během zadávání, ale také dříve zadaná data, aniž by bylo nutné procházet záznamy pomocí Lišty navigace.



Obrázek 206: Průvodce prvkem tabulky, výběr pole

Ne každé pole, které je možné vložit do formuláře, lze vybrat jako kontrolní pole tabulky. Tlačítka, tlačítka obrázků a tlačítka možností vybrat nelze.

Průvodce ovládacími prvky tabulky sestaví v okně pole, která se poté v tabulce objeví.

Tabulka Loans je k dispozici k úpravám v ovládacím prvku. Kromě pole ID (primární klíč) a pole Media_ID_BC (zadávaní médií pomocí snímače čárových kódů) se pro kontrolu použijí všechna pole.

Dříve vytvořenou kontrolu tabulky je nyní třeba dále rozvinout, aby bylo možné vstoupit do tabulky Loans. U polí, jako je Reader_ID nebo Media_ID, by bylo užitečnější, kdyby bylo možné vybrat přímo čtenáře nebo médium, a ne číslo představující čtenáře nebo médium. Za tímto účelem lze do ovládacího prvku tabulky umístit ovládací prvky, jako jsou seznamy. To je deklarováno později. Formátování pole Extension se dvěma desetinnými místy rozhodně nebylo zamýšleno.

	Media_ID	Reader_ID	Loan_Date	Return_Date	Extension	
▶	2,00	2,00	15.10.21	25.02.21	2,00	^
	0,00	3,00	02.11.21	04.04.21	1,00	
	3,00	0,00	04.11.21	28.11.21	2,00	
	5,00	0,00	28.11.21	03.02.21		
	4,00	0,00	28.11.21	04.04.21		
	4,00	0,00	09.11.21	05.04.21		
	3,00	0,00	09.12.21	17.11.21		
	7,00	0,00	09.12.21	04.04.21		▼

Záznam 1 z 22

Obrázek 207: Výstup Průvodce prvkem tabulky

Kromě vlastností uvedených v části "Výchozí nastavení mnoha ovládacích prvků". nad, jsou k dispozici následující vlastnosti.

Karta Obecné

Výška řádku.....

Výška jednotlivých řádků. Pokud zde není uvedena žádná hodnota, výška se automaticky přizpůsobí velikosti písma. Víceřádková textová pole se pak zobrazují jako jednotlivé řádky, které lze posouvat.

[RowHeight]

Lišta navigace.....

Stejně jako u tabulek se na spodním okraji ovládacího prvku zobrazuje číslo záznamu a navigační pomůcky.

[HasNavigationBar]

Značka záznamu...

Ve výchozím nastavení je značka záznamu na levém okraji ovládacího prvku. Označuje aktuální záznam. Pomocí značky záznamu můžeme přistupovat k funkci mazání celého záznamu.

[HasRecordMarker]

Karta Data

Protože se jedná o pole, které samo o sobě neobsahuje žádná data, ale spravuje jiná pole, nejsou zde žádné vlastnosti dat.

Karta Události

Chybí události *Změněno* a *Text změněn*. Je přidána událost *Došlo k chybě*.

Pole popisku

Kromě vlastností popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou k dispozici následující vlastnosti.

Karta Obecné

Název slouží jako popis jiného ovládacího prvku. Pokud je nadpis vázán na ovládací prvek, lze jej použít jako jeho zkratku. Vložené „~“ označuje konkrétní písmeno, které se stane zkratkou „Na~me“ definuje „m“ jako písmeno zkratky. Pokud je kurzor kdekoli ve formuláři, přejdeme stiskem klávesové zkratka Alt + m přímo do pole Name.

Ve výchozím nastavení není popisek zabalen. Pokud je pro pole příliš dlouhý, je zkrácen. Pozor: obtékání slov nerozpoznává mezery, takže pokud je pole příliš malé, může dojít k přerušení uvnitř slova. [MultiLine]

Karta Data

Není.

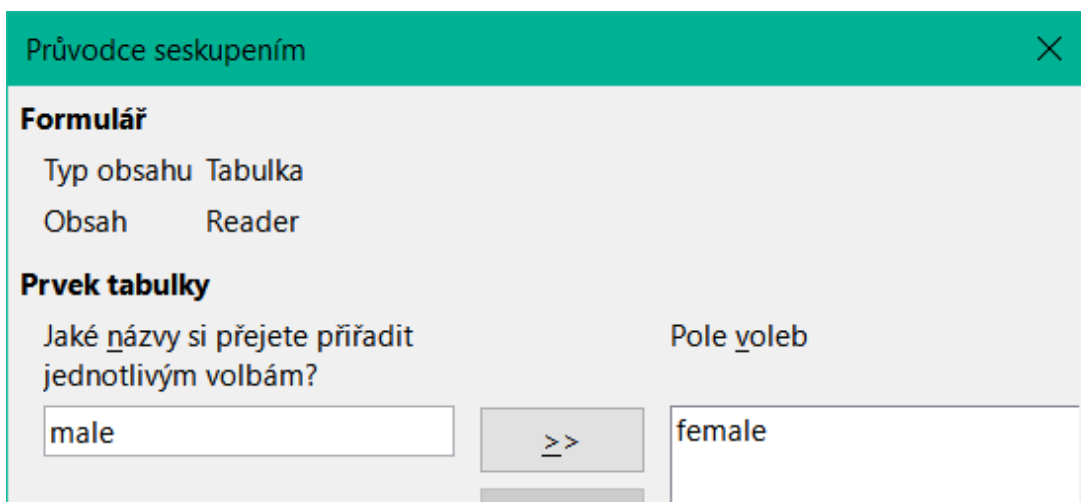
Karta Události

Pole popisku reaguje pouze na události spojené s myší, klávesou nebo fokusem.

Seskupení

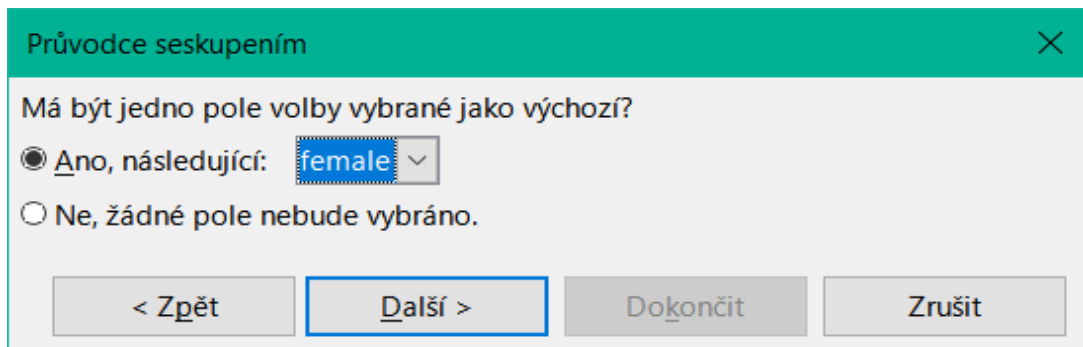
Seskupení graficky seskupuje několik ovládacích prvků a opatřuje je společným označením.

Pokud je vytvořeno seskupení s aktivními Průvodci, vychází Průvodce z předpokladu, že se v tomto seskupení bude vyskytovat několik tlačítek možností společně.



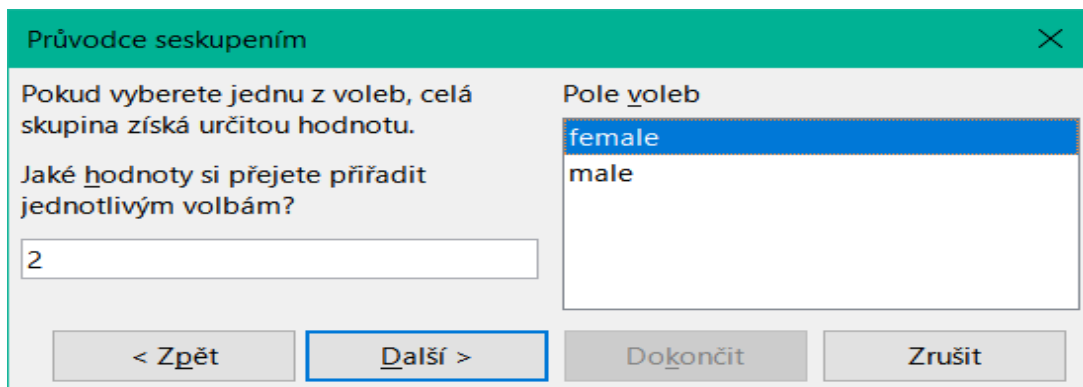
Obrázek 208: Průvodce skupinovým prvkem, výběr volitelného pole

Tento formulář vychází z tabulky Reader. Zabýváme se volbou pohlaví. Tyto položky jsou popisky tlačítek možností.



Obrázek 209: Průvodce skupinovým prvkem, výběr výchozího pole volby

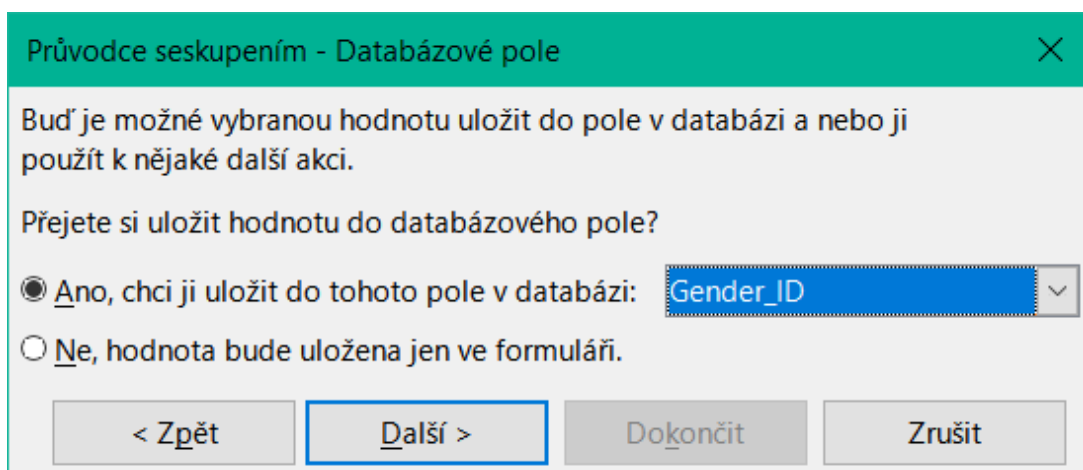
Zde je výchozí možností "female". Pokud nemá být výchozí pole, je výchozí položka v základní tabulce NULL.



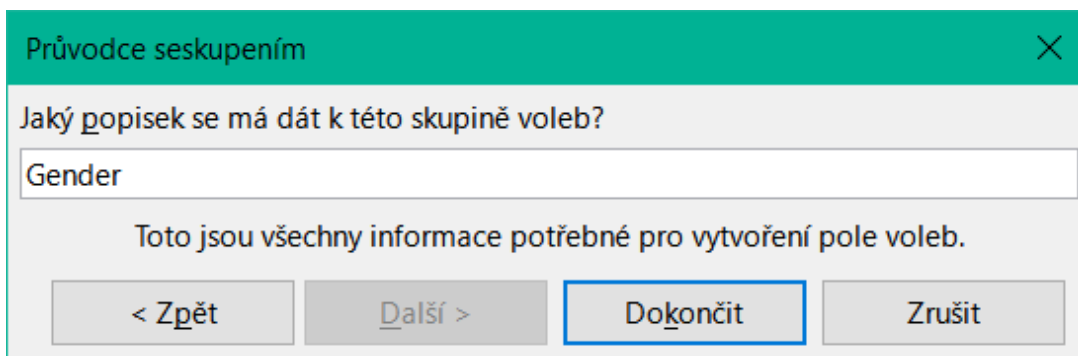
Obrázek 210: Průvodce skupinovým prvkem, přiřazení hodnoty volitelného pole

Průvodce nabízí ve výchozím nastavení oddělené hodnoty tlačítek volby, zde 1 pro ženu a 2 pro muže. Tyto hodnoty odpovídají příkladům polí primárního klíče v tabulce Gender.

Hodnota vybraná klepnutím na tlačítko volby se přenesou do pole Gender_ID základní tabulky formuláře Readers. Tímto způsobem je tabulka Readers opatřena odpovídajícím cizím klíčem z tabulky Gender pomocí tlačítka volby.

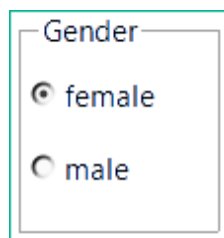


Obrázek 211: Průvodce skupinovým prvkem, výběr, zda se má hodnota uložit do databázového pole



Obrázek 212: Průvodce skupinovým prvkem, definice nadpisu

Skupině tlačítek možností je přiřazeno seskupení (rámeček) s označením Gender.



Obrázek 213: Příklad skupiny volitelných tlačítek

Pokud je v aktivním formuláři vybrána žena, výběr muže se zruší. Jedná se o vlastnost přepínačů, které jsou vázány na stejné pole v podkladové tabulce. Ve výše uvedeném příkladu nahrazují přepínače dvouprvkový seznam.

Kromě vlastností popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou k dispozici následující vlastnosti.

Karta Obecné

Popisek lze změnit z výchozí hodnoty. V současné době nelze měnit vlastnosti rámečku (tloušťka čáry, barva čáry), ale můžeme změnit formátování písma.

Karta Data

Žádné, protože tento ovládací prvek slouží pouze k vizuálnímu seskupení polí.

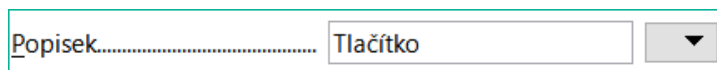
Karta Události

Seskupení reaguje na události týkající se myši, klávesy nebo zaměření.

Tlačítko

Kromě vlastností popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou k dispozici následující vlastnosti.

Karta Obecné



Popisek na tlačítku.

Vložení znaku „~“ lze určité písmeno popisku změnit na klávesovou zkratku. „Push ~Button“ definuje „b“ jako klávesovou zkratku. Pokud se kurzor nachází kdekoli ve formuláři, kombinace kláves Alt + b nás přenesou na tlačítko. [Label]

Při kliknutí získat zaměření... Ano

Přepnout..... Ne

Výchozí stav..... Nevybráno

Zařazení slova..... Ne

Činnost..... Žádné

URL.....

Rámec.....

Tlačítko získá zaměření, když na něj klepneme.

To nemusí být vždy žádoucí. Můžeme například chtít, aby se po klepnutí na tlačítko vložil nějaký obsah do jiného pole formuláře, a v takovém případě by měl kurzor po klepnutí na tlačítko zůstat v tomto poli.

[FocusOnClick]

Pokud ano, lze tlačítko zobrazit stisknuté. Stav tlačítka je zobrazen jako u přepínače. Po druhém stisknutí se zobrazí nestisknuté tlačítko.

[Toggle]

Aktivní, pokud je přepínač nastaven na hodnotu Ano. Vybrané odpovídá stisknutému tlačítku.

[DefaultState]

Obtékání slov, pokud je tlačítko příliš úzké.

[MultiLine]

K dispozici je celá řada akcí podobných těm, které se provádějí na navigačním panelu.

Pokud má akce otevřít dokument nebo webovou stránku, zadáme adresu URL do dalšího pole.

HTML: Soubor, který má být vyvolán tímto tlačítkem. Zde je třeba vybrat zdroj, který se má otevřít pomocí „Otevřít dokument/webovou stránku“ ve volbě „Činnost“.

Kromě HTML lze pomocí volby „Činnost“ otevřít i další moduly LO.

Pokud zde tedy zadáme

.uno:RecSearch, bude tlačítku přiřazena vyhledávací funkce navigátoru.

Můžeme také otevřít soubory aplikace Writer tak, že stisknutím tlačítka provedeme sloučení pošty pomocí záznamů z databáze.

Příkazy, které jsou zde k dispozici, lze zjistit pomocí záznamníku maker.

[TargetURL]

Pouze pro formuláře HTML:

Cílový rámec (uspořádání rámce pro různé stránky HTML), ve kterém se má soubor otevřít.[TargetFrame]

Výchozí tlačítko..... Ne

Pokud je tato možnost nastavena na Ano, je výchozí tlačítko orámováno. Pokud je na formuláři několik alternativních tlačítek, mělo by tuto vlastnost mít to, které se používá nejčastěji. Aktivuje se stisknutím klávesy Enter, pokud na této klávese nezávisí žádná jiná akce. Pouze jedno tlačítko na formuláři může být výchozím tlačítkem.
[DefaultButton]

Obrázek.....

Měl by se na tlačítku objevit obrázek?
[Graphic] [ImageURL]

Zarovnání obrázku..... Uprostřed

Aktivní pouze v případě, že byl vybrán obrázek. Určuje zarovnání obrázku k textu.
[ImagePosition] [ImageAlign]

Karta Data

Není. Tlačítko pouze provádí akce.

Karta Události

Schválit akci, Provést akci a Změna stavu položky.

Obrázkové tlačítko

Kromě vlastností již popsaných v části "Výchozí nastavení mnoha ovládacích prvků". nad, jsou k dispozici následující vlastnosti.

Karta Obecné

Podobně jako normální tlačítko. Toto tlačítko však nemá žádný text a samotné tlačítko není viditelné. Kolem grafiky se zobrazí pouze rámeček.

Ve výchozím nastavení nemá obrázkové tlačítko nastaven krok tabulátoru.

Upozornění: v době psaní tohoto text s tímto tlačítkem nefungují téměř žádné akce. Je prakticky použitelný pouze s makry.

Karta Data

Žádné; tento ovládací prvek pouze provádí akce.

Karta Události

Schválit akci a všechny události zahrnující myš, klávesu nebo zaměření.

Lišta navigace



Záznam 12 z 15

Obrázek 214: Ovládání lišty navigace

Standardní lišta Navigátor formulářem se vkládá do formulářů na spodní okraj obrazovky. Vložení této nástrojové lišty může způsobit malý posun formuláře směrem doprava, jak se na obrazovce vytváří. To může být rušivé v případech, kdy je lišta navigace pro některé části viditelného formuláře opět vypnutá, například když jsou ve viditelném formuláři podformuláře nebo více než jeden formulář.

Naproti tomu ovládací prvek lišty navigace, který je součástí formuláře, oddělený od příslušných položek, umožňuje jasně určit, přes které položky se pomocí panelu nástrojů pohybujeme. Například formulář pro výpůjčky musí nejprve vyhledat čtenáře a poté zobrazit média, která jsou

čtenáři půjčena. Ovládací prvek navigačního panelu je umístěn v blízkosti čtenáře, takže si uživatel všimne, že lišta navigace slouží pro čtenáře, a nikoli pro média, která jsou čtenáři zapůjčena.

Standardní navigační lišta formuláře zpřístupňuje tlačítka zobrazená na obrázku 215. Ovládací prvek navigační lišty zobrazuje stejná tlačítka kromě tlačítek Najít záznam, Filtry na základě formuláře a Zdroj dat, jako tabulka.

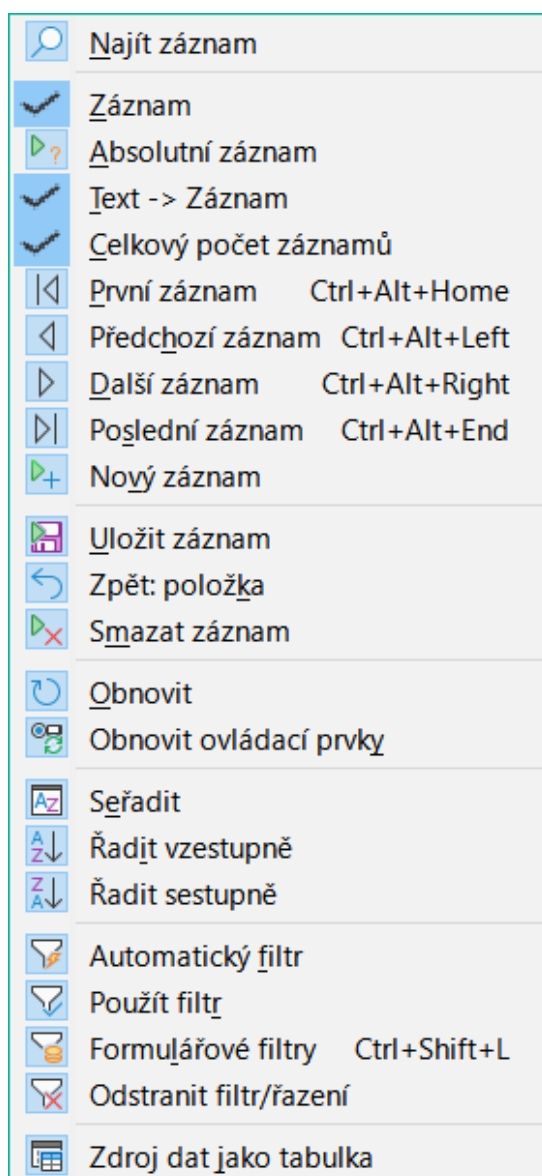
Kromě vlastností uvedených v části "Výchozí nastavení mnoha ovládacích prvků". nad jsou pro ovládací prvek Pruh navigátoru k dispozici následující vlastnosti.

Karta Obecné

Velikost ikony.....	Malá	▼
Umístění.....	Zobrazit	▼
Navigace.....	Zobrazit	▼
Činnost na záznamu...	Zobrazit	▼
Filtrování/řazení.....	Zobrazit	▼

Velikost ikony je nastavitelná. Kromě toho můžeme zvolit, které skupiny se zobrazí. Ty jsou zobrazeny na obrázku 214 zleva doprava pomocí svislé čáry jako oddělovače skupin: Umístění, Navigace, Akce na záznam a skupiny příkazů pro Filtrování a Třídění.

[IconSize]
[ShowPosition]
[ShowNavigation]
[ShowRecordActions]
[ShowFilterSort]



Obrázek 215: Výchozí rozdělení ikon na nástrojové liště Navigace na formuláři

Karta Data

Žádné, protože tento ovládací prvek pouze provádí akce.

Karta Události

Všechny události, které se týkají myši, klávesy nebo zaměření.

Nezávisle na tomto ovládacím prvku formuláře samozřejmě nadále existuje navigační lišta *vložitelná* se stejnými položkami jako na obrázku výše.

Tato vložitelná lišta navigace navíc umožňuje obecné vyhledávání záznamů, filtrování na základě formuláře a zobrazení základního zdroje dat formuláře v tabulkovém zobrazení nad formulářem.

Pokud pracujeme nejen s formulářem, ale i s podformuláři a pomocnými formuláři, musíme si dát pozor, aby tato vložitelná lišta navigace při přepínání formulářů nezmizela. To vytváří na obrazovce rušivý efekt.

Rolovací tlačítka a posuvníky

Žádné z těchto polí nemá přímé spojení s daty v databázi. Lze je využít pouze pomocí maker, kde je rolovací tlačítko integrováno například do číselného pole.

Posuvník se od rolovacího tlačítka liší tím, že má kromě tlačítek pro zvyšování a snižování hodnoty v pevně daném rozsahu také posuvník.

Při otevření formuláře musí být posuvníku sdělena aktuální hodnota pole, ke kterému je připojen. Toto makro je vázáno na **Vlastnosti formuláře > Události > Po změně záznamu**.

```
SUB scrollbar_form(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oField AS OBJECT
    oForm = oEvent.Source
    oField = oForm.getByName("scrollbar")
    oField.ScrollValue = oForm.getInt(oForm.findColumn("numeric_value"))
END SUB
```

Nejprve se deklaruje proměnná. Událost, kterou bude zpracovávat, je zadána na formuláři, na který bude makro nakonec navázáno. Posuvník najdeme podle jeho názvu na formuláři. Jeho aktuální hodnota je nastavena na číslo uložené v poli numeric_field v základní tabulce.

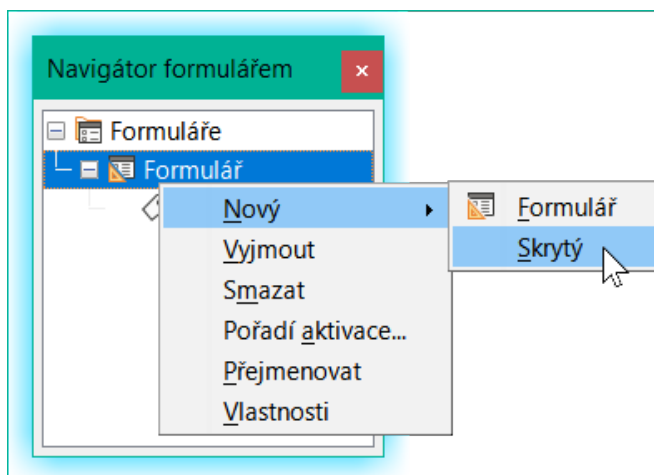
Pokud je hodnota změněna pomocí posuvníku, musí být tato změna přenesena do podkladového pole. Toto makro je vázáno na **Vlastnosti: Události > Při úpravě**.

```
SUB Scrollbar_change(oEvent AS OBJECT)
    DIM oForm AS OBJECT
    DIM oField AS OBJECT
    DIM inValue AS INTEGER
    oField = oEvent.Source.Model
    inValue = oEvent.Value
    oForm = oField.Parent
    oForm.updateInt(oForm.findColumn("numeric_value"), inValue)
END SUB
```

Nejprve se deklarují proměnné. Vybraná hodnota se z pole načte jako celé číslo. Na tuto hodnotu se pak aktualizuje odpovídající pole v podkladové tabulce numeric_value.

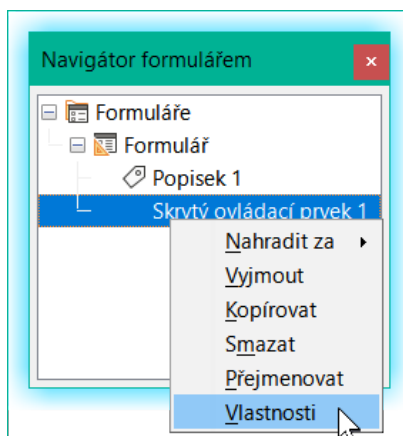
Podrobný výklad tohoto druhu kódu je uveden v kapitole 9, Makra.

Skrytý ovládací prvek



Obrázek 216: Vytvoření skrytého ovládacího prvku pomocí Navigátoru formulářem

Jedním z typů ovládacích prvků, které nelze vložit z panelu nástrojů ovládacích prvků, je skrytý ovládací prvek. Musí být vytvořen v navigátoru formulářem pomocí místní nabídky formuláře.

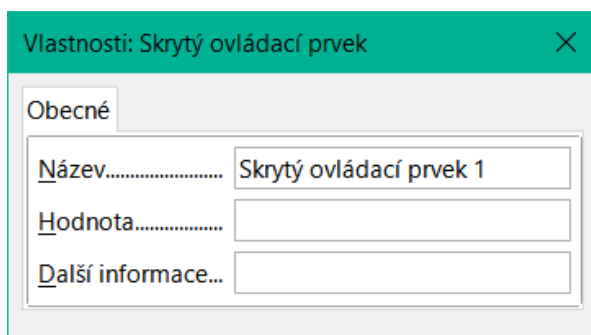


Obrázek 217: Přístup k dialogovému oknu Vlastnosti skrytého ovládacího prvku

Stejně jako jakýkoli jiný ovládací prvek je i skrytý ovládací prvek součástí formuláře. V uživatelském rozhraní však není vidět.

Skrytý ovládací prvek má pouze několik vlastností. Název a doplňující informace mají stejný význam jako u ostatních ovládacích prvků. Kromě toho lze ovládacímu prvku přiřadit hodnotu [Hidden Value].

Pokud nepoužíváme makra, nemá skrytý ovládací prvek smysl. V případě maker je však často užitečné mít možnost ukládat mezihodnoty někde ve formuláři, aby bylo možné k nim přistupovat později. Příklad tohoto způsobu použití maker je popsán v kapitole 9, Makra, v části „Hierarchické seznamy“.



Obrázek 218: Dialogové okno Vlastnosti pro skrytý ovládací prvek

Vícenásobný výběr

Pokud pomocí tlačítka **Vybrat** vybereme velkou oblast nebo několik prvků formuláře, mohou být provedeny následující úpravy.

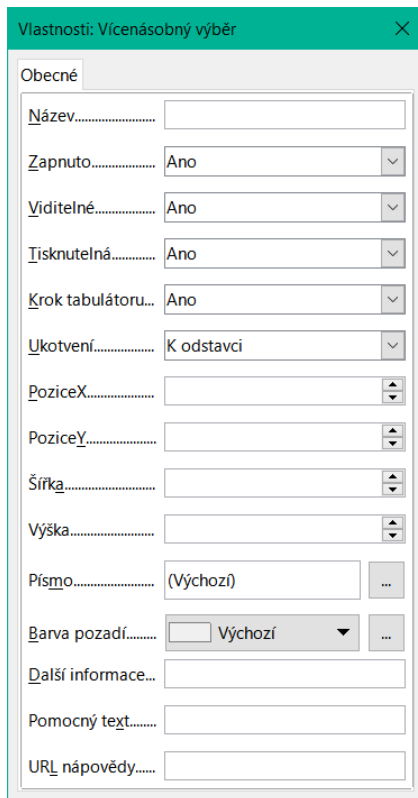
Název bychom neměli měnit. To by způsobilo, že by všechny vybrané prvky náhle získaly stejný název. Ztížilo by to vyhledávání jednotlivých prvků pomocí nástroje Navigátor formulářem a znemožnilo by to správu formuláře pomocí pojmenovaných ovládacích prvků v makrech.

Vícenásobný výběr je užitečnější pro změnu písma, výšky nebo barvy pozadí ovládacích prvků. Všimneme si, že změna barvy pozadí má vliv i na popisky.

Pokud chceme změnit pouze popisky, podržíme klávesu *Control* a klepneme na tyto ovládací prvky přímo nebo v Navigátoru, případně klepneme pravým tlačítkem myši na pole a vyvoláme

vlastnosti ovládacího prvku. Nyní je výběr vlastností, které můžeme měnit, větší, protože se jedná pouze o podobná pole. Zde můžeme změnit cokoli, co je k dispozici v poli popisku.

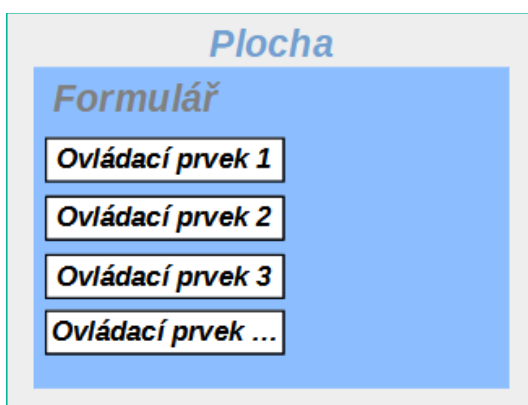
Možnosti vícenásobného výběru tedy závisí na výběru polí. Můžeme současně měnit ovládací prvky stejného druhu, které mají všechny vlastnosti, které existují pro jednu instanci.



Obrázek 219: Dialogové okno Vlastnosti pro vícenásobný výběr

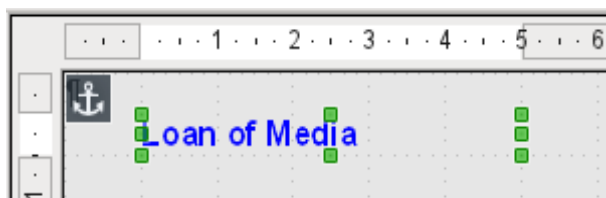
Vyplněný jednoduchý formulář

Jednoduchý formulář má ovládací prvky formuláře pro zápis nebo čtení záznamů z jedné tabulky nebo dotazu. Jeho konstrukci ukazuje následující příklad.



Obrázek 220: Schematické znázornění jednoduchého tvaru

Příklad jednoduchého formuláře pro výpůjčky v knihovně je zde uveden v několika variantách. Rychlý způsob použití Průvodce formulářem je popsán v kapitole 8, Začínáme s aplikací Base v příručce *Začínáme s LibreOffice*. Zde popíšeme vytvoření v Zobrazení návrhu.

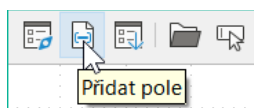


Obrázek 221: Vložení štítku pro nadpis formuláře

Záhlaví formuláře bylo vytvořeno pomocí pole popisku. Písmo bylo změněno. Pole popisku je ukotveno k odstavci v levém horním rohu dokumentu. Pomocí místní nabídky pole popisku byl vytvořen formulář, který byl propojen s tabulkou Loans (viz „Vlastnosti formuláře“ nad). Stránka má také jednotně barevné pozadí.

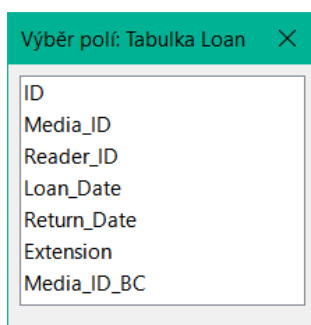
Přidání skupin polí

Rychlou variantou přímého zadání polí s popisky je použití funkce Přidat pole.



Obrázek 222: Ikona Přidat pole na nástrojové liště Návrh vzorce

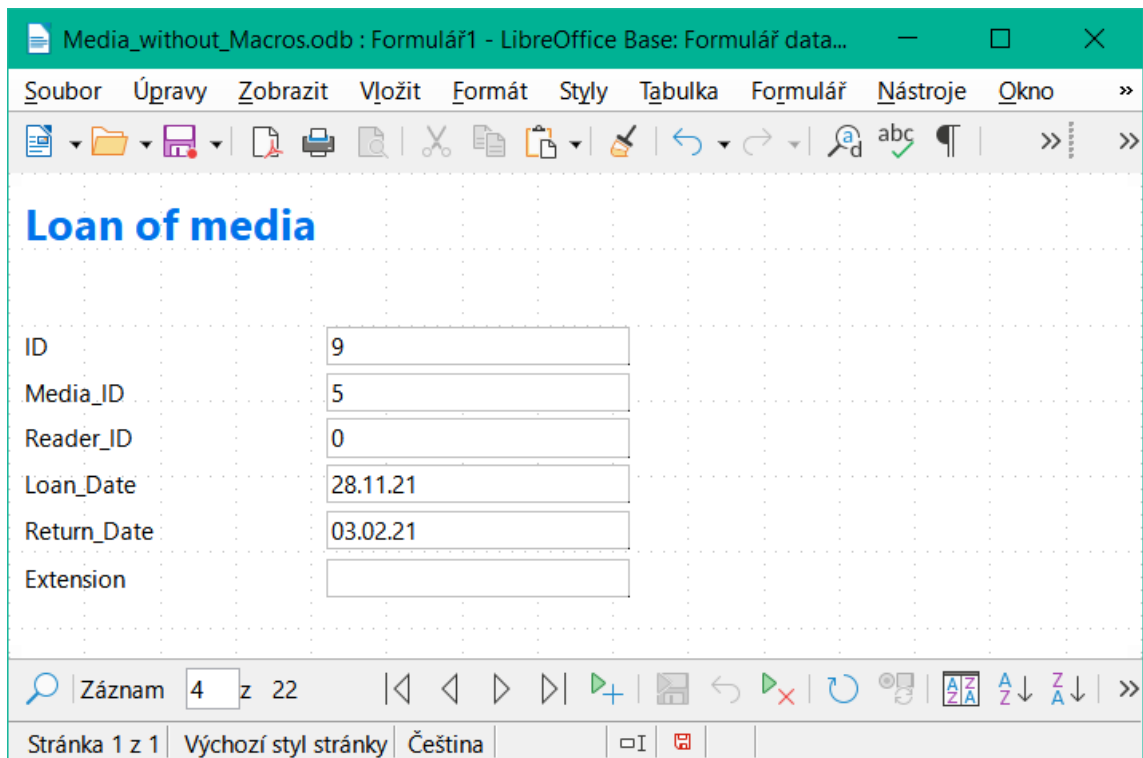
Tato funkce, dostupná na nástrojové liště Návrh formuláře (viz obrázek 184), umožňuje vybrat všechna pole podkladové tabulky.



Obrázek 223: Použití dialogového okna Přidat pole pro výběr pole podkladové tabulky, které má být přidáno

Dvojitým klepnutím na pole je vložíme do formuláře jako skupinu s popisky, zpočátku všechna naskládaná na stejném místě. Skupinu je třeba oddělit tak, aby formulář nakonec vypadal jako na obrázku 224. Pro lepší přehlednost byly z okna odstraněny všechny nepotřebné nástrojové lišty a okno bylo také zkomprimováno, takže nejsou vidět všechny prvky lišty navigace.

Byla vybrána všechna pole kromě pole Media_ID_BC, které je určeno pouze pro použití se snímačem čárových kódů.



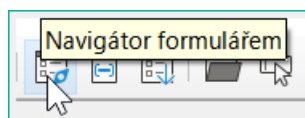
Obrázek 224: Jednoduchý formulář vytvořený pomocí funkce Přidat pole

Pro každé pole tabulky byl automaticky vybrán vhodný ovládací prvek formuláře. Čísla jsou v číselných polích a jsou deklarována jako celá čísla bez desetinných míst. Pole s datem jsou správně reprezentována jako ovládací prvky data. Všechna pole mají stejnou šířku. Pokud by bylo zahrnuto grafické ovládání, bylo by mu přiděleno čtvercové pole.

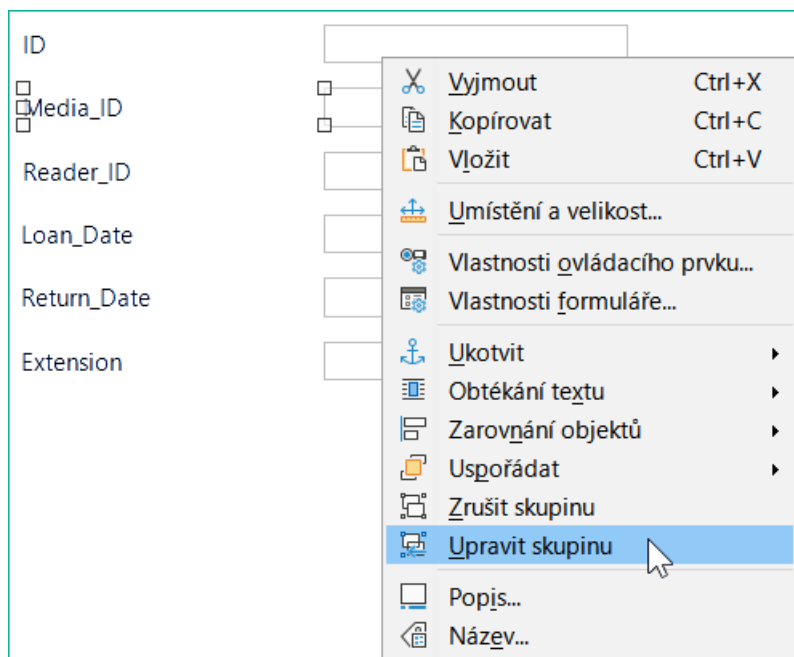
Úprava proporcí pole

Nyní můžeme provádět některé kreativní činnosti, včetně úpravy délky polí a vytvoření dat jako rozevíracích polí. Ještě důležitější je, aby pole Media_ID a Reader_ID byla uživatelsky přívětivější, pokud každý uživatel knihovny nemá čtenářský průkaz a každé médium není při příjmu opatřeno ID. To se v následujícím textu nepředpokládá.

Chceme-li upravit jednotlivá pole, musíme upravit skupinu. To lze provést klepnutím pravým tlačítkem myši na skupinu a následným vyvoláním místní nabídky (obrázek 226). Pro budoucí práci bude přehlednější, pokud použijeme Navigátor formulářem.



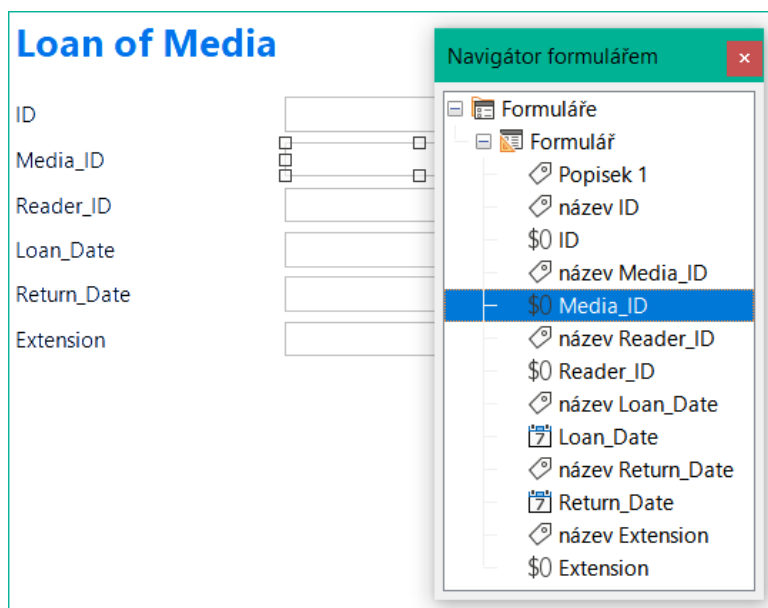
Obrázek 225: Ikona Navigátor formuláře na nástrojové liště Návrh formuláře



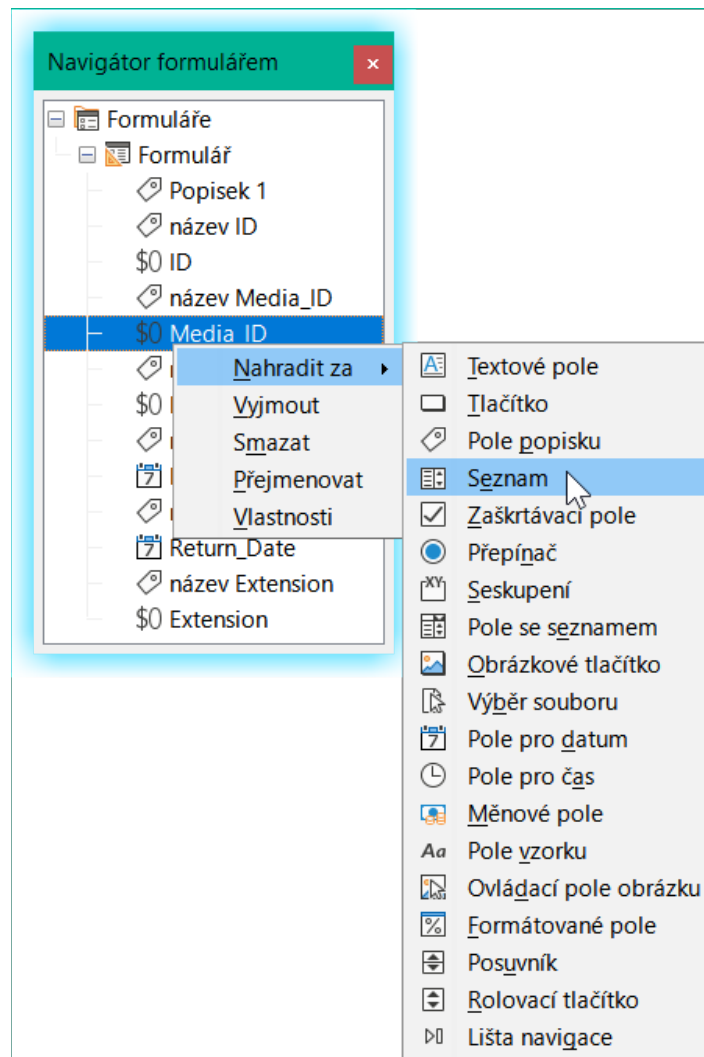
Obrázek 226: Použití místní nabídky pro rozřazení do skupin nebo vstup/výstup ze skupiny ovládacích prvků formuláře

Navigátor formulářem zobrazuje všechny prvky formuláře s jejich popisky. U ovládacích prvků jsou názvy převzaty přímo z názvů polí v podkladové tabulce. Názvy popisků mají příponu Label.

Klepnutím na Media_ID se toto pole vybere (obrázek 227). Klepnutím pravým tlačítkem myši můžeme vybrané pole nahradit jiným typem pole pomocí místní nabídky (obrázek 228).

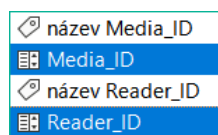


Obrázek 227: Výběr ovládacích prvků formuláře přímo pomocí Navigátoru formulářem



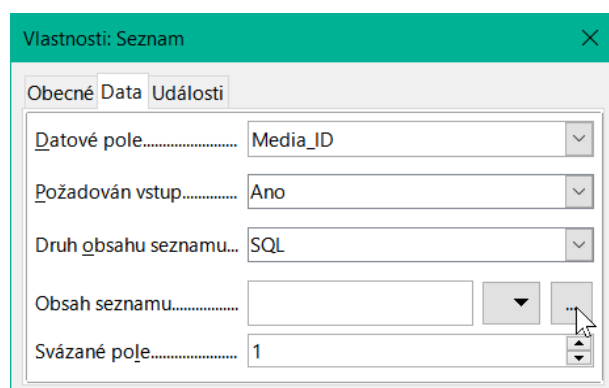
Obrázek 228: Nahrazení jednoho typu ovládacího prvku jiným pomocí Navigátoru formulářem

Tato výměna se provádí pro ovládací prvky Media_ID a Reader_ID.



Obrázek 229: Media_ID a Reader_ID nahrazeny seznamem polí

Změna je v Navigátoru formulářem viditelná změnou doprovodné ikony.



Obrázek 230: Dialogové okno Vlastnosti (karta Data) pro pole se seznamem

Dotaz SQL pro pole se seznamem lze nyní vytvořit prostřednictvím grafického uživatelského rozhraní klepnutím na tlačítko vpravo. To se provádí automaticky, pokud je rámeček seznamu vytvořen přímo, ale ne, pokud je vytvořen konverzí z jiného typu ovládacího prvku. Příkaz SQL nalezneme v kapitole 5, Dotazy.

Vzhledem k tomu, že pole seznamu mají být rozbalovací, lze současně opravit následující závady:

- Popisky pro pole seznamu by měly být Media místo Media_ID a Reader místo Reader_ID.
- Ovládací prvek ID by měl být nastaven pouze pro čtení.
- Pole, která nejsou nezbytně nutná pro poskytování půjček pro nové médium, nepotřebují mít nastaven krok tabulátoru. Bez něj lze formulářem procházet mnohem rychleji. V případě potřeby lze kroky tabulátoru nastavit také pomocí aktivační sekvence (viz strana "Výchozí nastavení mnoha ovládacích prvků" nad). Pomocí klávesy Tab musí být ve všech případech přístupná pouze pole Media, Reader a Loan_date.
- Pokud je formulář určen k provádění výpůjček, je zbytečné a také matoucí, aby se v něm zobrazovala vrácená média. Média s datem návratu by měla být odfiltrována. Kromě toho by pořadí zobrazení mohlo být řazeno podle pole Reader, takže média zapůjčená téže osobě by se zobrazovala postupně. Viz poznámka „Vlastnosti formuláře“ nad. Problém je však v tom, že čtenáře lze řadit pouze podle ID, nikoli podle abecedy, protože tabulka, která je základem formuláře, obsahuje pouze ID.

Přidání jednotlivých polí

Přidání jednotlivých polí je o něco složitější. Pole je třeba vybrat, přetáhnout na plochu formuláře a zadat příslušné pole z podkladové tabulky. Kromě toho je třeba správně zvolit typ pole; například číselná pole mají standardně dvě desetinná místa.

Pouze při vytváření polí seznamu se do hry zapojí Průvodce, který začátečníkům usnadňuje provádění kroků pro vytvoření správných polí – do určité míry. Po překročení tohoto bodu přestává Průvodce splňovat požadavky, protože:

- Zadané hodnoty nejsou automaticky tříděny.
- Kombinace několika polí v obsahu pole seznamu není možná.

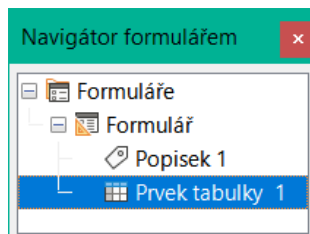
I zde je třeba provést zpětná vylepšení, aby bylo možné potřebný kód SQL vytvořit poměrně rychle pomocí vestavěného editoru dotazů.

Při přidávání jednotlivých ovládacích prvků musí být pole a jeho popis explicitně přiřazeny (viz „Výchozí nastavení mnoha ovládacích prvků“ nad). V praxi by mohlo být lepší, kdybychom pole nepřipřazovali k popiskům, abychom před změnou vlastností pole nemuseli použít skupinu úprav.

Ovládací prvek tabulky

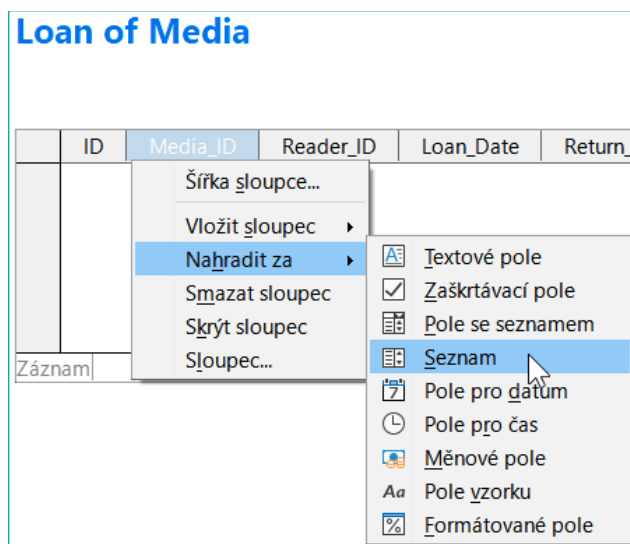
Použití Průvodce tabulkou k vytvoření ovládacího prvku tabulky již bylo popsáno v části "Ovládací prvek tabulky". nad. Má však některé nedostatky, které je třeba zlepšit:

- Pole Media_ID a Reader_ID se musí stát poli seznamu.
- Číselná pole musí být zbavena desetinných míst, protože Průvodce u čísel vždy uvádí dvě desetinná místa.



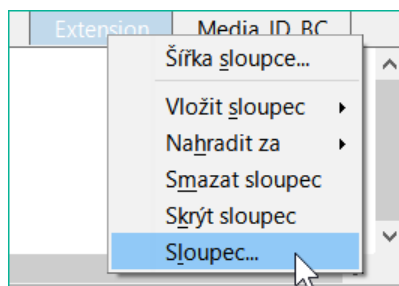
Obrázek 231: Ovládací prvek tabulky zobrazený v nástroji Form Navigator

Změna polí v ovládacím prvku tabulka není možná stejným způsobem, jaký je popsán u ostatních ovládacích prvků. Popis polí v Navigátoru končí u ovládacího prvku tabulky. Navigátor neví nic o ovládacích prvcích, které se nacházejí uvnitř ovládacího prvku tabulky a odkazují na pole v podkladové tabulce. To platí i později při pokusech o přístup k polím pomocí maker. Nelze k nim přistupovat podle názvu.



Obrázek 232: Nahrazení jednoho typu ovládacího prvku jiným pro sloupec v ovládacím prvku tabulky

Ovládací prvky v rámci ovládacího prvku tabulky se nazývají sloupce. Pomocí místní nabídky je nyní možné nahradit jeden typ pole jiným. Není však k dispozici celá řada typů. Nejsou zde žádná tlačítka, pole možností ani grafické ovládací prvky.



Obrázek 233: Přístup k dialogovému oknu Vlastnosti pro sloupec v ovládacím prvku tabulky

Vlastnosti polí jsou skryty v místní nabídce za konceptem sloupců. Zde lze například změnit číselné pole Extension tak, aby se nezobrazovala desetinná místa. Také minimální výchozí hodnota –1 000 000,00 sotva dává smysl pro prodloužení úvěru. Číslo by mělo zůstat vždy malé a kladné.

Jakmile jsou vlastnosti sloupce vyvolány, můžeme vybrat jiný sloupec, aniž by se dialogové okno vlastností zavřelo. Tímto způsobem můžeme pracovat se všemi poli postupně, aniž bychom museli mezitím ukládat.

✓ Poznámka

Hodnota se v dialogu uloží, jakmile dojde k pohybu mezi vlastnostmi. Tak je tomu i při přechodu z jednoho sloupce do druhého.

Nakonec uložíme celý formulář a nakonec i samotnou databázi.

Vlastnosti polí zabudovaných do ovládacího prvku tabulky nejsou tak komplexní jako u polí mimo něj. Například písmo lze nastavit pouze pro celý ovládací prvek tabulky. Kromě toho nemáme možnost přeskočit jednotlivé sloupce odstraněním kroků tabulátoru.

i Tip

Ve formuláři se můžeme pohybovat pomocí myši nebo klávesy Tab. Pokud do ovládacího prvku tabulky vložíme tabulátor, kurzor se při každém dalším tabulátoru posune o jedno pole doprava; na konci řádku se přesune zpět na první pole dalšího záznamu v ovládacím prvku tabulky. Chceme-li ovládnutí tabulky ukončit, použijeme klávesovou zkratku Ctrl + Tab.

Pokud mají být během používání viditelná pouze některá pole, můžeme ve formuláři použít několik různých ovládacích prvků tabulky, protože tabulátor je ve výchozím nastavení zachycen každým ovládacím prvkem tabulky.

Formulář uvedený na obrázku 234 je určen pro výpůjčku médií. V horním ovládacím prvku tabulky jsou zobrazena pouze bezprostředně potřebná pole. V dolní části se zobrazují všechna pole, aby bylo zřejmé, pro kterou osobu a médium je příznání určeno.

Loan fo Media

	Reader	Medium	Loan_Date	
▶	Lisa Gerd	Eine kurze Geschichte der Zeit	15.10.21	^
	Monika Mirinda	Der kleine Hobbit	02.11.21	
	Bert Lederstrumpf	Traditionelle und kritische Theorie	04.11.21	
	Bert Lederstrumpf	I hear you knocking	28.11.21	
	Bert Lederstrumpf	Die neue deutsche Rechtschreibung	28.11.21	
	Bert Lederstrumpf	Die neue deutsche Rechtschreibung	09.11.21	
	Bert Lederstrumpf	Traditionelle und kritische Theorie	04.11.21	▼

Záznam 1 z 22

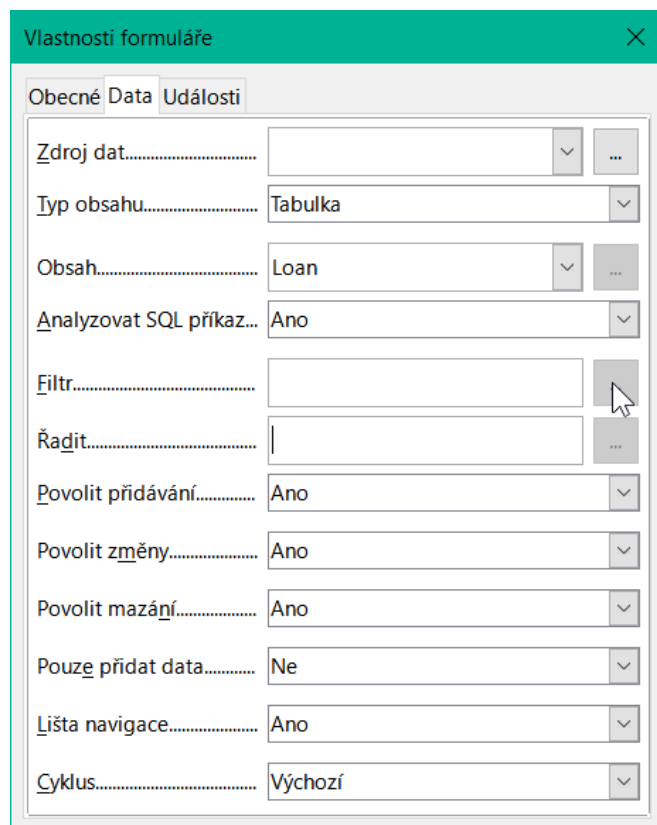
	Reader	Medium	Loan_Date	Extension	Return_Date	
▶	Lisa Gerd	Eine kurze Ge	15.10.21	2,00	25.02.21	^
	Monika Mirinda	Der kleine Ho	02.11.21	1,00	04.04.21	
	Bert Lederstrum	Traditionelle u	04.11.21	2,00	28.11.21	
	Bert Lederstrum	I hear you kno	28.11.21			
	Bert Lederstrum	Die neue deut	28.11.21		04.04.21	
	Bert Lederstrum	Die neue deut	09.11.21			
	Bert Lederstrum	Traditionelle	04.11.21		17.11.21	▼

Záznam 1 z 22

Obrázek 234: Formulář s více ovládacími prvky tabulky

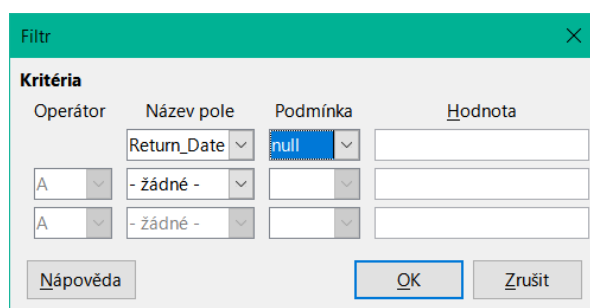
Tento obrázek ukazuje estetický nedostatek, kterému je třeba věnovat naléhavou pozornost. V ovládacím prvku horní tabulky se stejné médium někdy vyskytuje více než jednou. K tomu dochází proto, že tabulka zobrazuje i média, která byla vrácena dříve. Proto je třeba data filtrovat, aby se zobrazovaly pouze zápůjčky. Záznamy s datem vrácení by se zobrazovat neměly.

Toto filtrování je možné buď pomocí dotazu, nebo přímo pomocí vlastností formuláře. Pokud je to provedeno pomocí vlastností formuláře, lze filtr během zadávání dočasně vypnout. Filtrování pomocí dotazu je popsáno v kapitole 5, Dotazy. Zde popíšeme, jak to provést pomocí vlastností formuláře.



Obrázek 235: Klepneme na ikonu elipsy vedle pole Filtr na kartě Data v dialogovém okně Vlastnosti formuláře.

Filtrování se provádí pomocí tlačítka se třemi tečkami (elipsou), které otevře níže uvedený dialog. Pokud známe kódování jazyka SQL, můžeme filtr zadat také přímo do textového pole Filtr.



Obrázek 236: Dialog filtru

Pomocí grafického uživatelského rozhraní nyní můžeme vybrat pole s názvem Return_Date. Zobrazí pouze záznamy, u nichž je pole prázdné, přičemž „empty“ znamená označení NULL v jazyce SQL.

Vyčištěný formulář (na obrázku 237) nyní vypadá poněkud jednodušeji.

Samozřejmě je stále co zlepšovat, ale ve srovnání s dřívější podobou má tato verze jasnou výhodu v tom, že všechna média jsou vidět na první pohled.

Zpracování dat pomocí ovládacích prvků tabulky je podobné jako u skutečné tabulky. Klepnutí pravým tlačítkem myši na záhlaví existujícího záznamu způsobí jeho smazání a v případě nových záznamů lze záznam zrušit nebo uložit.

Po opuštění řádku se záznam automaticky uloží.

Loan fo Media			
	Reader	Medium	Loan_Date
▶	Lisa Gerd	Eine kurze Geschichte der Zeit	15.10.21
	Monika Mirinda	Der kleine Hobbit	02.11.21
	Bert Lederstrumpf	Traditionelle und kritische Theorie	04.11.21
	Bert Lederstrumpf	I hear you knocking	28.11.21
	Bert Lederstrumpf	Die neue deutsche Rechtschreibung	28.11.21
	Bert Lederstrumpf	Die neue deutsche Rechtschreibung	09.11.21
	Bert Lederstrumpf	Traditionelle und kritische Theorie	04.11.21

Záznam 1 z 22

	Reader	Medium	Loan_Date	Extension	Return_Date
	Bert Lederstr	Das Postfix-Buch	25.02.21		
	Lisa Gerd	Eine kurze Geschic	25.02.21		
	Terence Nob	Der kleine Hobbit	04.03.21		
	Heinrich Müll	Eine kurze Geschic	04.04.21	1,00	
	Heinrich Müll	Im Augenblick	12.09.21		
	Heinrich Müll	I hear you knockir	29.04.21		
	Bert Lederstr	Traditionelle und	04.11.21		

Záznam 2 z 11

Obrázek 237: Upravený formulář s více ovládacími prvky tabulky

Formulář Loan of Media můžeme ještě několika způsoby vylepšit.

- Bylo by hezké, kdyby výběr čtenáře v jedné části formuláře způsobil, že se média zapůjčená tomuto čtenáři zobrazí v jiné části.
- Ve výše uvedené tabulce vidíme mnoho záznamů, které nejsou nutné, protože tato média jsou již zapůjčena. Tabulka byla vytvořena za účelem umožnění výpůjček, takže by bylo lepší, kdyby se objevila pouze prázdná stránka, kterou by bylo možné vyplnit novou výpůjčkou.

Taková řešení jsou k dispozici pomocí dalších formulářů, které jsou hierarchicky uspořádány a umožňují oddělené pohledy na data.

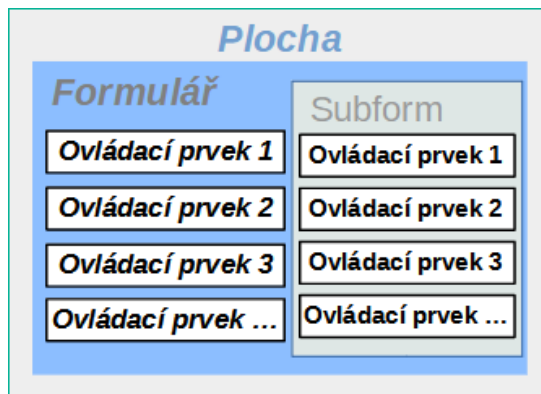
i Tip

Ovládací prvky tabulky lze kombinovat s jinými poli ve formuláři. Například ovládací pole tabulky zobrazuje pouze názvy těch médií, u nichž byly provedeny změny v základních polích formuláře.

Při kombinaci ovládacích prvků tabulky s jinými formulářovými poli se objevuje drobný problém, který lze snadno obejít. Pokud se ve formuláři vyskytují oba typy polí společně, kurzor se automaticky přesune z ostatních polí do ovládacího prvku tabulky, přestože ve výchozím nastavení má tento typ pole nastavení Krok tabulátoru > Ne. To lze napravit přepnutím vlastnosti tabulátoru na Ano a poté zpět na Ne. Tím se zaregistruje hodnota „Ne“.

Hlavní formuláře a podformuláře

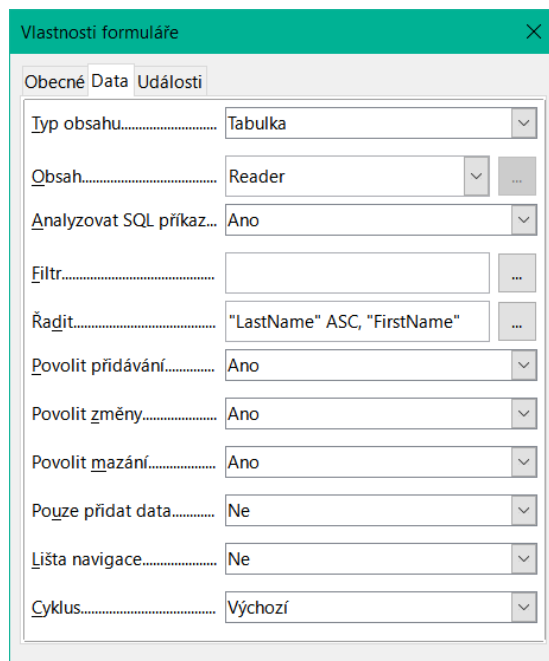
Podformulář leží uvnitř formuláře stejně jako ovládací prvek formuláře. Stejně jako ovládací prvek formuláře je vázán na data z hlavního formuláře. Zdrojem dat však může být jiná tabulka nebo dotaz (nebo příkaz SQL). Důležité pro dílčí formulář je, aby byl jeho zdroj dat nějakým způsobem propojen se zdrojem dat hlavního formuláře.



Obrázek 238: Schematické znázornění formuláře s podformulářem

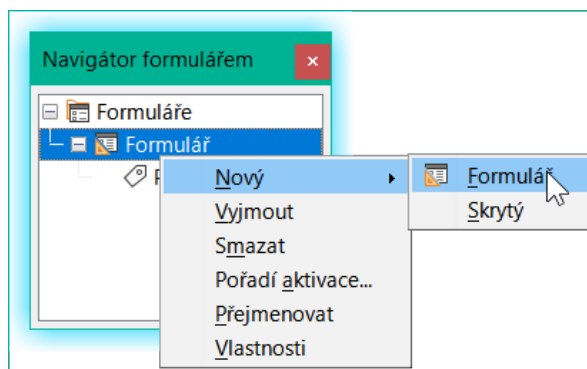
Typickou strukturou tabulek, které lze použít jako dílčí formuláře, jsou tabulky s relací jeden k více (viz kapitola 3, Tabulky). Hlavní formulář zobrazuje tabulku se záznamy, ke které lze připojit a zobrazit mnoho závislých záznamů v podformuláři.

Nejprve použijeme vztah tabulky Reader k tabulce Loan (viz kapitola 3, Tabulky). Tabulka Reader bude tvořit základ hlavního formuláře a tabulka Loan bude zobrazována v podformuláři.



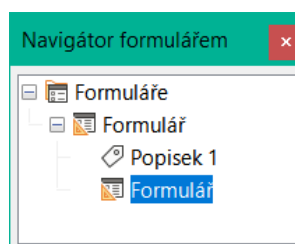
Obrázek 239: Dialogové okno Vlastnosti formuláře (karta Data) pro formulář propojený s tabulkou Reader, která je řazena abecedně

Zde je hlavní formulář propojen s tabulkou Reader. Pro urychlení vyhledávání čtenářů je tabulka řazena abecedně. Obejdeme se bez lišty navigace, protože obsah podformuláře by se nacházel mezi hlavním formulářem a lištou navigace. Místo toho použijeme vestavěný ovládací prvek formuláře (obrázek 214).



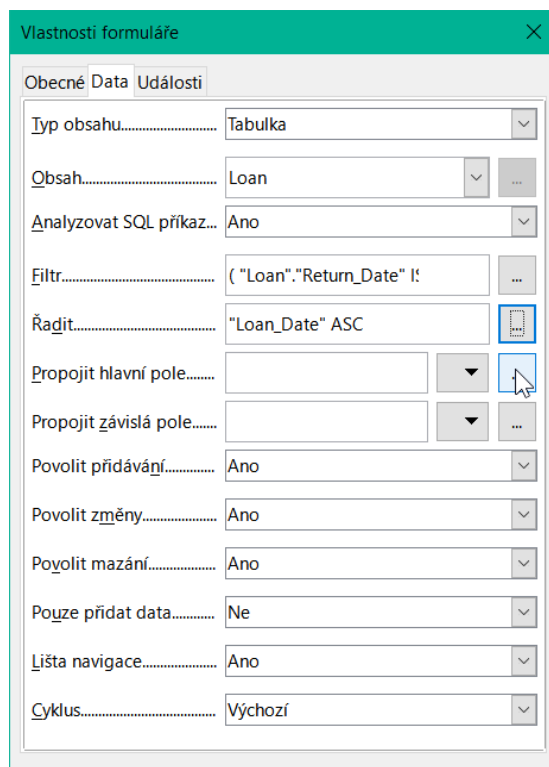
Obrázek 240: Vytvoření nového podformuláře pomocí Navigátoru formulářem

Klepeme pravým tlačítkem myši na hlavní formulář v Navigátoru formulářem a pomocí místní nabídky vytvoříme nový formulář. Tento formulář má opět výchozí název Form, ale nyní je prvkem v podsložce hlavního formuláře.



Obrázek 241: Jeden formulář v podsložce jiného formuláře

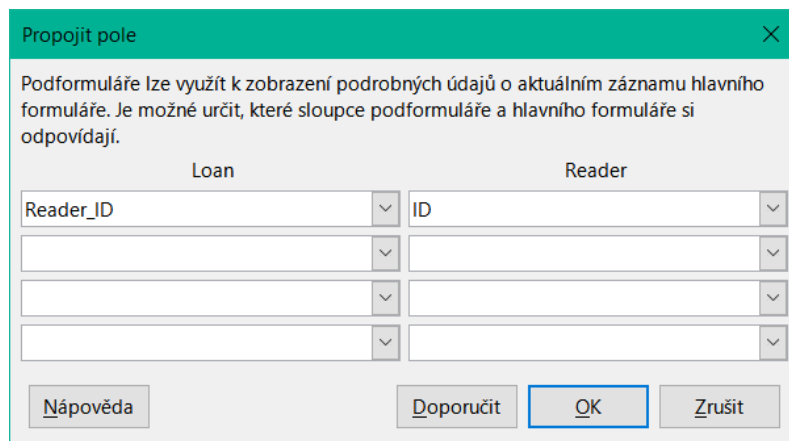
Nyní je třeba nastavit vlastnosti podformuláře tak, aby mu byl přidělen správný zdroj dat, aby bylo možné reprodukovat data pro správného čtenáře.



Obrázek 242: Dialogové okno Vlastnosti formuláře (karta Data) pro dílčí formulář

Pro podformulář je vybrána tabulka Loans. Pro filtr určíme, že pole Return_Date má být prázdné ("Return_Date" IS NULL). Tím se zabrání zobrazení již vrácených médií. Záznamy by měly být seřazeny podle data výpůjčky. Vzestupné řazení zobrazuje nahoře médium vypůjčené na nejdelší dobu.

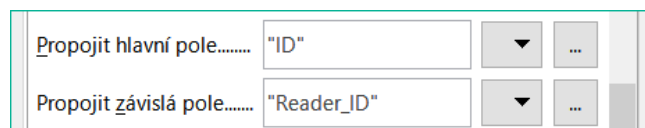
Propojení hlavních polí a propojení závislých polí slouží k vytvoření vazby na hlavní hlavní formulář, ve kterém se podformulář nachází. Tlačítko se třemi tečkami opět ukazuje, že je k dispozici užitečný dialog pro jejich vytvoření.



Obrázek 243: Použití dialogového okna Propojit pole k propojení sloupců v hlavním formuláři a dílčím formuláři

V části Loans se zobrazují pole tabulky Loans, v části Readers se zobrazují pole tabulky Reader. Reader_ID z tabulky Loans by mělo být nastaveno jako ekvivalent ID z tabulky Reader.

Přestože toto propojení již bylo v databázi vytvořeno pomocí Nástroje > Relace (viz kapitola 3, Tabulky), funkce, která se skrývá za tlačítkem Doporučit v tomto dialogovém okně, na něj neodkazuje a navrhuje, aby byl první cizí klíč v tabulce Loans, tedy Media_ID, propojen s polem ID z tabulky Reader. Průvodce vytvořením formuláře to řeší lépe tím, že načte relaci z relace databáze.



Obrázek 244: Pole ID v tabulce Reader propojené s polem Reader_ID v tabulce Loan

Zvolené propojení mezi tabulkou pro podformulář a tabulkou pro hlavní formulář je nyní zadáno pomocí polí z tabulek.

Abychom mohli vytvořit ovládací prvek tabulky pro hlavní formulář, musíme nyní vybrat hlavní formulář v Navigátoru formulářem. Pokud je povolen Průvodce ovládním tabulky, zobrazí pole dostupná v hlavním formuláři. Podobným způsobem pracujeme s podformulářem.

Po nastavení ovládacích prvků tabulky je třeba provést úpravy, o kterých jsme již hovořili při vytváření jednoduššího formuláře:

- Nahrazení číselného pole Media_ID v podformuláři seznamem.
- Přejmenování pole Media_ID na Media.
- Úprava číselných polí na formát bez desetinných míst.
- Omezení minimálních a maximálních hodnot.
- Přejmenování jiných polí za účelem úspory místa nebo přidání znaků jiných než ASCII, které by se neměly používat v názvech polí v databázových tabulkách.

Funkce řazení a filtrování jsou v hlavním formuláři doplněny o lištu navigace. Ostatní pole na liště navigace nejsou potřeba, protože jsou většinou dostupná z ovládacího prvku tabulky (zobrazení záznamu, navigace po záznamu) nebo se provádějí pohybem přes ovládací prvek tabulky (ukládání dat). Konečný tvar může vypadat jako na obrázku 245.

Loan of Media

ID	LastName	FirstName	Lock	Gender
3,00	Mirinda	Monika	<input type="checkbox"/>	female
15,00	Müller	Heinrich	<input type="checkbox"/>	male
1,00	Müller	Heinrich	<input checked="" type="checkbox"/>	male
7,00	MüBiggang	Kerstin	<input type="checkbox"/>	female
0,00	Mebodu	Terence	<input type="checkbox"/>	male

Záznam 8 z 11 (1)

Loaned media of the chosen reader

Media	Loan_Date	Return_Date	Extension
Eine kurze Geschichte der Zeit	04.04.21		1,00
I hear you knocking	29.04.21		
Im Augenblick	12.09.21		

Záznam 1 z 3

Obrázek 245: Formulář sestávající z hlavního formuláře (nahore) a dílčího formuláře (dole)

Pokud je nyní v hlavním formuláři vybrán čtenář, zobrazí se v podformuláři média půjčená tomuto čtenáři. Když je položka vrácena, zobrazuje se ve formuláři až do obnovení formuláře. K tomu dojde automaticky při načtení dalšího záznamu do hlavního formuláře. Pokud je opět vybrána původní čtenář, vrácená média se již nezobrazují.

Tato zpožděná aktualizace je v tomto případě vlastně žádoucí, protože umožňuje zkontrolovat média, která právě leží na pultu knihovny, a ihned zjistit, zda byla zaregistrována.

Tato struktura formulářů je podstatně jednodušší než předchozí struktura s jediným formulářem. Stále však existují detaily, které lze vylepšit:

- V případě delší výpůjčky médií a termínů výpůjček může dojít ke změně. Změna data média by mohla znemožnit dohledání, která položka je v knihovně stále k dispozici a která je vypůjčená. Změna data výpůjčky by mohla vést k chybám. Oznámení o stažení nebylo možné ověřit.
- Pokud není záznam čtenáře vybrán klepnutím na záhlaví záznamu vlevo, pouze malá zelená šipka v záhlaví ukazuje, který záznam je právě aktivní. Je docela dobře možné, že aktivní záznam bude posunut přímo z okna ovládaní tabulky.
- Místo textu „Loaned Media of the chosen Reader“ („Vypůjčená média vybraného čtenáře“, pozn. překl.) by bylo lepší uvést jméno čtenáře.
- Stejně médium je možné půjčit dvakrát, aniž by bylo vráceno.
- Záznam o vypůjčeném předmětu je možné poměrně snadno vymazat.
- Údaje lze měnit nebo mazat v hlavním formuláři. To může být užitečné pro malé knihovny s malou návštěvností. Pokud je však na přepážce výpůjček rušno, neměla by se editace údajů o uživatelích provádět současně s vydáváním výpůjček. Bylo by lepší, kdyby bylo možné registrovat nové uživatele, ale stávající uživatelská data by zůstala nedotčena. Pro knihovny to platí i pro mazání nebo úplné změny názvů.

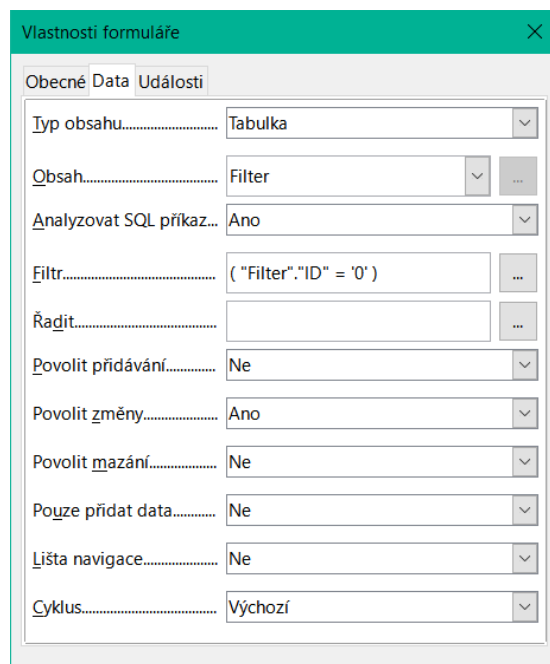
Nejprve zlepšíme výběr čtenářů. To by nás mělo ochránit před změnami v záznamech o výpůjčkách. Jednoduchým řešením by bylo nepovolit žádné změny s výjimkou vkládání nových záznamů. To stále vyžaduje funkci vyhledávání, když si čtenář přeje vypůjčit položku. Lepší by bylo použít k vyhledání čtenáře pole seznamu a operace vydávání a vracení provádět v samostatných ovládacích prvcích tabulky.

Pro hlavní formulář potřebujeme tabulku, do které může pole seznamu zapsat hodnotu navázanou na tuto tabulku. Tabulka vyžaduje pole celých čísel a primární klíč. Vždy bude obsahovat pouze jeden záznam, takže pole primárního klíče ID může být bezpečně deklarováno jako Tiny Integer. Proto by měla být vytvořena následující tabulka s názvem Filter.

Tabulka 8: Definice tabulky filtrů

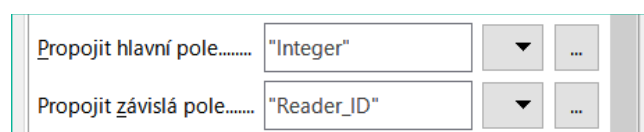
Název tabulky: Filter	
Název pole	Typ pole
ID	Tiny Integer, primární klíč
Integer	Integer

Tabulka má primární klíč s hodnotou 0. Tento záznam bude opakovaně načítán a přepisován hlavním formulářem.



Obrázek 246: Dialogové okno Vlastnosti formuláře (karta Data) pro hlavní formulář

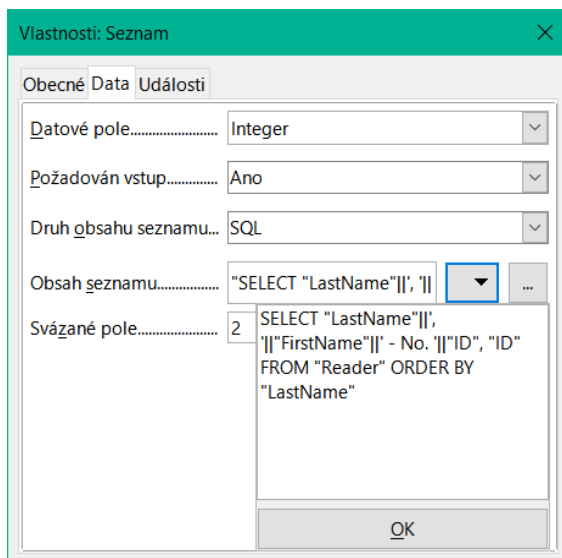
Hlavní formulář je založen na tabulce Filter. Z tabulky se pouze načte hodnota, která je spojena s primárním klíčem (ID) 0. Žádná data nebudou přidána, aktuální záznam bude pouze opakovaně přepsán. Jelikož jsou povoleny pouze úpravy jednoho záznamu, byla by lišta navigace zbytečná.



Obrázek 247: Pole Integer v tabulce Filter propojené s polem Reader_ID v tabulce Loan

Tento hlavní formulář je propojen s dílčím formulářem tak, že hodnota pole Integer v tabulce Filter je stejná jako hodnota pole Reader_ID v tabulce Loan. Vlastnosti podformuláře se oproti výše uvedené verzi nezměnily.

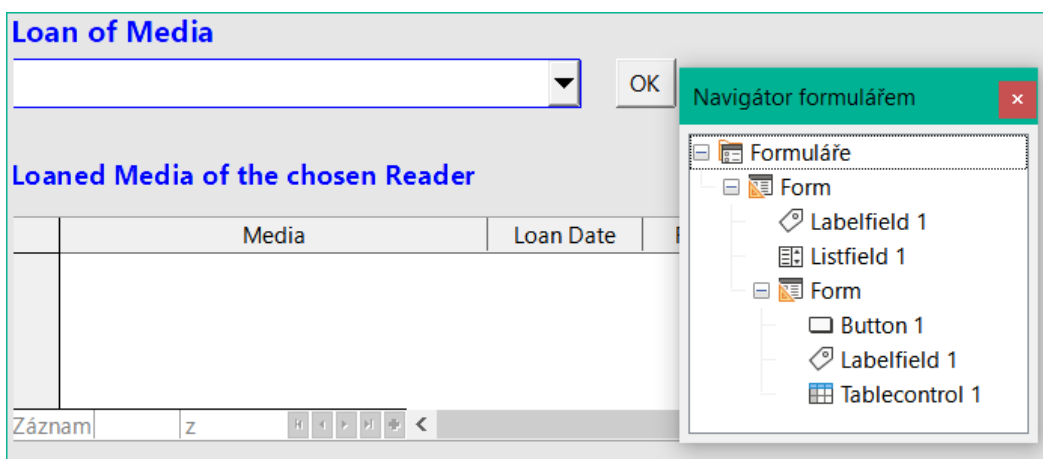
Před vytvořením pole seznamu v hlavním formuláři musíme vypnout průvodce. Průvodce seznamem umožňuje vytvořit pouze pole, které zobrazuje obsah jediného pole; nebylo by možné mít v oblasti zobrazení pole seznamu příjmení, jméno a další číslo. Stejně jako v jednodušším formuláři nyní zadáme pro obsah pole seznamu Příjmení, Jméno – číslo ID. Pole seznamu přenáší ID do podkladové tabulky.



Obrázek 248: Dialogové okno Vlastnosti pro pole seznamu zobrazující obsah pole Obsah seznamu na kartě Data

Vedle pole seznamu se vytvoří tlačítko. Toto tlačítko je ve skutečnosti součástí podformuláře. Převezme dvě funkce: uložení záznamu v hlavním formuláři a aktualizaci tabulky v podformuláři. Stačí svěžit aktualizaci tlačítku v podformuláři. Proces uložení upraveného hlavního formuláře se pak provede automaticky.

Tlačítko může být jednoduše označeno jako OK. V obecných vlastnostech tlačítka je mu přiřazena akce Obnovit formulář.



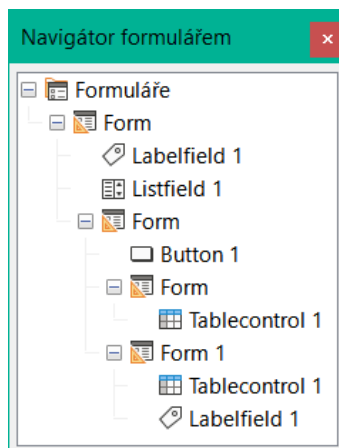
Obrázek 249: Hlavní formulář jako filtr pro dílčí formulář

Hlavní formulář se skládá pouze ze záhlaví a pole seznamu; podformulář obsahuje další záhlaví, ovládací prvek tabulky z předchozí verze a tlačítko.

Formulář nyní funguje lépe:

- Žádného čtenáře nyní nelze upravovat, měnit ani mazat a
- Čtenáře lze rychleji najít zadáním do ovládacího prvku než pomocí filtru.

Pro větší funkčnost (návraty bez změny předchozích dat) je třeba vytvořit druhý podformulář, který bude propojen se stejnou tabulkou Loans. Aby byla zajištěna funkčnost seznamu na obrázku 249, musí být oba dílčí formuláře umístěny o úroveň níže, jako dílčí formuláře dílčího formuláře. Data jsou aktualizována hierarchicky od hlavního formuláře směrem dolů přes podformuláře. Tlačítko v dříve popsaném formuláři musí být nyní umístěno v prvním podformuláři, nikoli ve dvou podformulářích, které jsou pod ním.



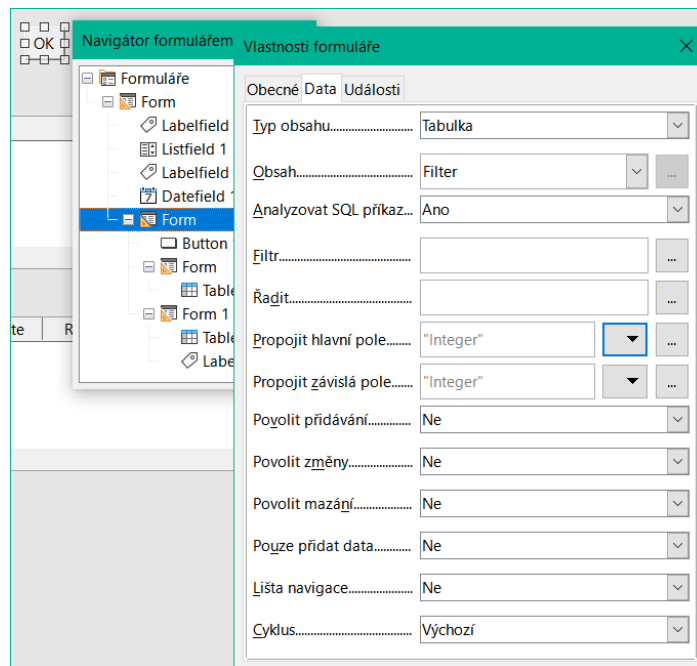
Obrázek 250: Hierarchie formulářů a podformulářů v nástroji Form Navigátor

Zde se k zobrazení různých úrovní používá Navigátor formulářem. V hlavním formuláři máme textové pole pro název formuláře a pole seznamu pro vyhledání čtenáře. Pole seznamu se zobrazí v dolní části formuláře, protože je deklarováno po podformulářem. Tuto sekvenci zobrazení nelze změnit. Podformulář má pouze jedno tlačítko, které slouží k aktualizaci jeho obsahu a zároveň k uložení hlavního formuláře. O úroveň níže se nacházejí další dva podformuláře. Ty jsou při vytváření pojmenovány různě, aby nedošlo k záměně v žádné úrovni stromu.

✓ Poznámka

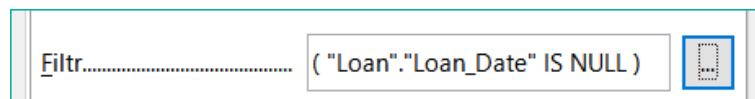
Názvy formulářů a ovládacích prvků jsou v podstatě bez významu. Pokud však mají být tyto názvy přístupné pomocí maker, musí být rozlišitelné. Nelze rozlišit shodné názvy na stejné úrovni.

Při vytváření rozsáhlejších struktur formulářů má samozřejmě smysl používat smysluplné názvy formulářů a jejich ovládacích prvků. V opačném případě by se hledání správného pole mohlo rychle stát problematickým.



Obrázek 251: Dialogové okno Vlastnosti formuláře (karta Data) pro dílčí formulář, kde je mnoho polí nastaveno na hodnotu Ne

Hlavní formulář a podformulář používají stejnou tabulku. V dílčím formuláři se nezadávají žádná data. Proto jsou všechna pole tohoto formuláře nastavena na hodnotu Ne. Hlavní formulář a podformulář jsou propojeny polem, jehož hodnota se má přenášet do podformulářů: pole Integer v tabulce Filter.



Obrázek 252: Podmínka filtru pro hodnotu přenášenou do dílčího formuláře

V prvním podformuláři se nezobrazují žádná existující data; slouží pouze k vytváření nových dat. K tomu je vhodný navrhovaný filtr. Zobrazí se pouze záznamy odpovídající poli Reader_ID a s prázdným polem data výpůjčky ("Loan_Date" IS NULL). V praxi to znamená prázdný ovládací prvek tabulky. Protože ovládací prvek tabulky není průběžně aktualizován, nově zapůjčená média v něm zůstanou, dokud se pomocí aktualizací tlačítka OK nevybere nový název nebo se data nepřenesou do druhého podformuláře.

Obrázek 253: Dialogové okno Vlastnosti formuláře (karta Data) pro druhý dílčí formulář

Druhý podformulář vyžaduje více nastavení. I tento formulář obsahuje údaje z tabulky "Loans". Zde jsou data filtrována pro prázdné datum vrácení ("Return_Date" IS NULL). Údaje jsou seřazeny stejně jako v předchozím formuláři tak, aby byla okamžitě vidět média, která jsou zapůjčena nejdéle.

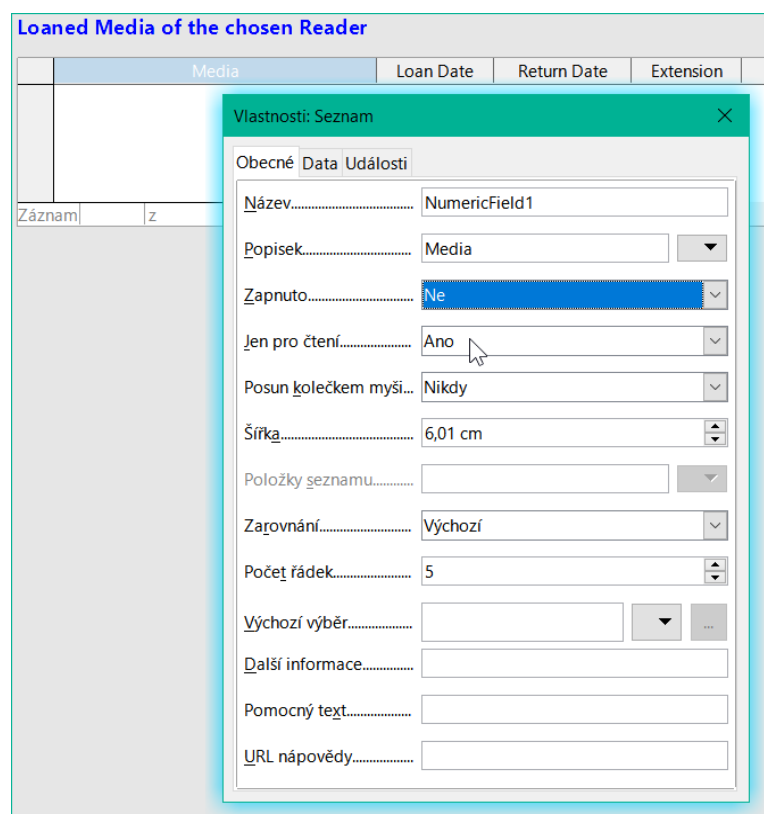
Důležité jsou také následující body. Staré záznamy lze měnit, ale nelze přidávat nové. Vymazání je rovněž nemožné. Jedná se o první nezbytný krok, který zabrání pozdějšímu jednoduchému vymazání záznamů o výpůjčkách. Stále by však bylo možné změnit médium a datum výpůjčky. Proto bude nutné upravit vlastnosti sloupců. Nakonec by se mělo zobrazit médium a datum půjčky, ale mělo by být chráněno před změnou.

Ovládací prvek tabulky se po vytvoření formuláře jednoduše duplikuje. To se provádí tak, že jej vybereme, zkopírujeme, zrušíme výběr a poté jej vložíme ze schránky. Kopie se bude nacházet na stejném místě jako originál, a proto ji bude třeba odtáhnout. Poté lze oba ovládací prvky tabulky upravovat samostatně. Ovládací prvek tabulky pro návrat médií lze ponechat prakticky stejné. Je třeba změnit pouze přístup k zápisu pro sloupce Media a Loan date.

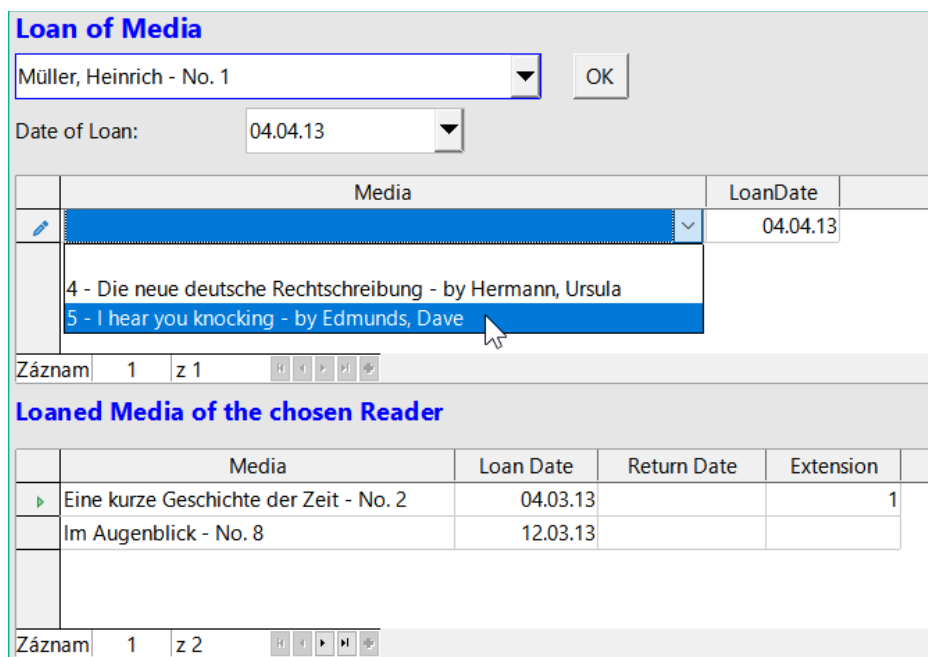
Zatímco pro Loan_Date je nutné zvolit pouze Jen pro čtení, pro pole seznamu to nestačí. Toto nastavení nebrání tomu, aby bylo pole seznamu použito k provádění změn. Pokud je však možnost Zapnuto nastavena na hodnotu Ne, nelze zde provést volbu. Pole seznamu obsažené v ovládacím prvku tabulky se pak zobrazí jako textové pole určené pouze pro čtení.

Ve výše uvedeném ovládacím prvku tabulky jsou odstraněna všechna pole, která nemají s výpůjčkou nic společného. Zůstává pouze médium jako výběrové pole a datum výpůjčky Loan_Date.

Pokud je nakonec vybrán dotaz pro pole seznamu v horním ovládacím prvku tabulky, zobrazí se pouze ta média, která lze skutečně vypůjčit. Více informací o této problematice najdeme v kapitole 5, Dotazy.



Obrázek 254: Dialogové okno Vlastnosti (karta Obecné) pro sloupec Media zobrazuje hodnotu Povolenou nastavenou na Ne



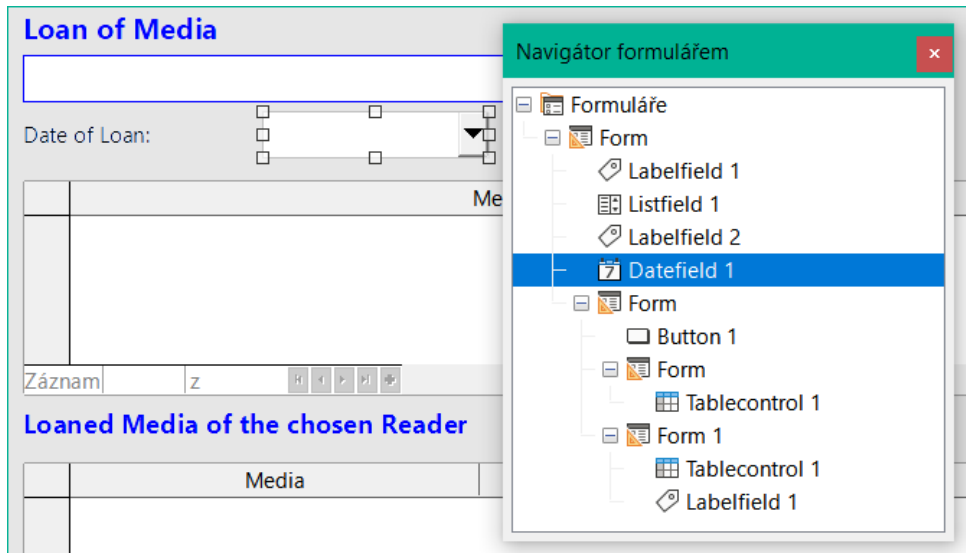
Obrázek 255: Výběrové pole v horním podformuláři zobrazuje pouze položky médií, které nejsou vypůjčené.

Formulář pro výpůjčku médií je již výrazně užitečnější. Když čtenář přijde k výpůjční přepážce, vyhledá se jeho jméno. Média, která mají být zapůjčena, lze vybrat pomocí seznamu a zadat datum zapůjčení. Klávesou Tab přejdeme na další záznam.

Žádoucí je také poslední vylepšení: v současné době je třeba pokaždé zvolit datum výpůjčky. Představme si den v knihovně, kdy se uskuteční třeba 200 výpůjčních transakcí, třeba jen

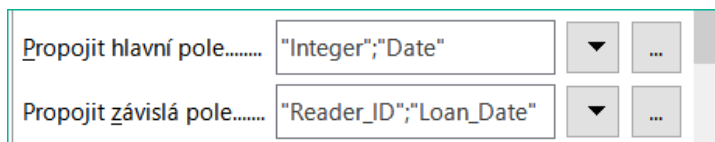
s jednou osobou, která musí pokaždé půjčit asi 10 médií. To by vyžadovalo opakované zadávání stejného pole data. Musí existovat způsob, jak to zjednodušit.

Náš hlavní formulář je propojen s tabulkou Filter. Hlavní formulář pracuje pouze se záznamem, který má jako primární klíč "ID" 0. Do tabulky Filter lze však zabudovat další pole. V současné době neexistuje žádné pole, které by mohlo ukládat datum, ale můžeme snadno vytvořit nové pole s názvem Date a typem pole Date. V tabulce Filter máme nyní uloženo nejen Reader_ID ("Filter". "Integer"), ale také Loan_Date ("Filter". "Date").



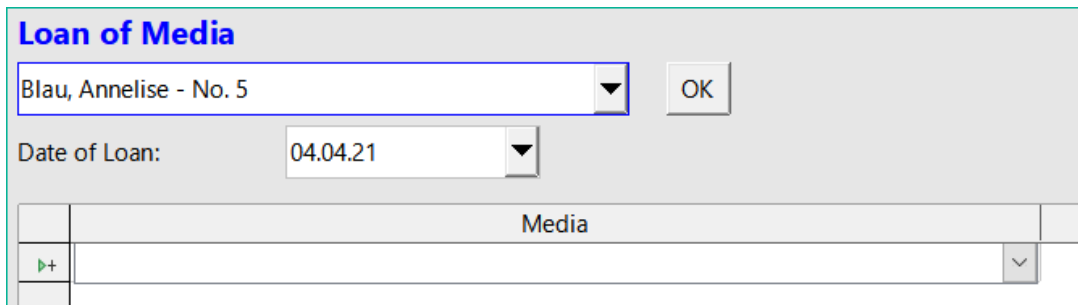
Obrázek 256: Dodatečné pole data v hlavním formuláři

V hlavním formuláři se objeví další pole s datem a popisem odkazující na jeho obsah. Hodnota z pole datum je uložena v tabulce Filter a přenáší se pomocí vazeb z podformuláře do podformuláře.



Obrázek 257: Pole Celé číslo a Datum v tabulce Filter propojená s poli Reader_ID a Loan_Date v tabulce Loan

Vazba mezi oběma formuláři se nyní týká dvou polí. Pole Integer je vázáno na pole Reader_ID podformuláře. Pole Date je vázáno na pole Loan_Date. Tím je zajištěno, že se údaj Loan_Date automaticky přenesou z tabulky Filter do tabulky Loans při vytvoření zápůjčky.



Obrázek 258: Datum výpůjčky se zadává pouze jednou; při změně čtenáře je třeba jej zadat znovu.

Z ovládacího prvku tabulky je nyní odstraněno pole s datem, takže tabulka obsahuje pouze jedno vyhledávací pole. To by byl ideální požadavek pro ještě větší urychlení práce knihovny. Každé

médium bude mít totiž vytištěné identifikační číslo, tak proč se musí hledat? Číslo můžeme zadat přímo. Nebo, ještě lépe, je možné jej naskenovat pomocí čtečky čárových kódů. Média pak mohou být půjčována tak rychle, jak rychle si je vypůjčitel dokáže uložit do tašky.

To je znázorněno v příkladu databáze. Výše uvedený příklad by měl stačit pro pochopení počátečního návrhu formuláře, ale protože v příkladu databáze Media_without_Macros.odt je formulář dále rozvíjen, jsou níže stručně uvedena další vylepšení.

Loan

	First Name	Last Name	ID
	Annelise	Blau	5
	Greta	Garbo	6
▶	Lisa	Gerd	2
	Hein	Keindurchblick	4
	Bert	Lederstrumpf	0

Filter (Last Name)

Záznam 3 z 10 (1)

Loan for Reader Gerd, Lisa

Loan Date:

current Loan

	Medium	Loan Date
▶	1 - Das sogenannte Böse - by Lorenz, Konrad	06.11.21

Záznam 1 z 1

Return

	Medium	Loan Date	Return Date	Extension	Loan Days	Balance Time
▶	6 - Datenbanken mit OpenOffice.org 3 - by Open Document	05.11.21			1 Days	13 Days

Záznam 1 z 1

Obrázek 259: Výpůjční formulář

Formulář Loan zobrazuje následující vlastnosti:

- Čtenáři jsou zobrazeni v ovládacím prvku tabulky. Zde můžeme také zadat nové čtenáře.
- Pomocí filtru propojeného s tabulkou Filter lze filtrovat jména podle jejich počátečního písmene. Pokud tedy zadáme A, zobrazí se pouze osoby, jejichž příjmení začíná na A. Toto filtrování je nezávislé na velikosti zadaného písmene.
- V nadpisu je opět uvedeno jméno osoby, které má být půjčka poskytnuta. Pokud byl na tohoto dlužníka uvalen zámeček, zobrazí se také tato informace.
- Datum výpůjčky je nastaveno na aktuální datum. To se provádí v tabulce filtru pomocí jazyka SQL tak, že pokud není zadáno žádné datum, je výchozí hodnotou, která se uloží, aktuální datum.
- Vypůjčitelná média se vybírají pomocí seznamu. Po stisknutí tlačítka Update se výpůjčka přenesou do níže uvedeného ovládacího prvku tabulky.
- Ovládací prvek tabulky uprostřed slouží pouze k zobrazení aktuálního data výpůjčky média. Zde je možné chybu opravit i zpětně vymazáním řádku.

- V dolním ovládacím prvku tabulky, stejně jako ve výše uvedeném příkladu, není možné měnit data médií a výpůjček. Záznamy nelze ani vymazat.
- Kromě zadání data vrácení nebo případného prodloužení výpůjčky se v této tabulce zobrazuje také počet dní, na které lze médium vypůjčit, a kolik dní zbývá do konce aktuální výpůjční lhůty.
- Pokud je tento zbývajících čas záporný, musí být médium okamžitě vráceno. Číslo je poté uzamčeno. Znovu je to možné až po navrácení média. Po návratu stačí tlačítko Update stisknout pouze jednou.

Tento formulář, vytvořený pomocí dotazů, má podstatně složitější strukturu než dříve zobrazená verze. Více informací o tomto najdeme v kapitole 5, Dotazy.

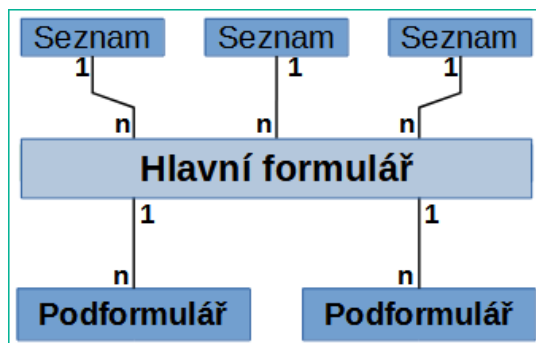
Jeden pohled – více formulářů

Zatímco příklad formuláře Loans zahrnuje pouze vstupy do jedné tabulky (tabulka Loans) a navíc umožňuje vstup do podformuláře pro nové čtenáře, postup zadávání pro média je podstatně rozsáhlejší. V konečné podobě je tabulka médií obklopena celkem osmi dalšími tabulkami (viz kapitola 3, Tabulky).

Tabulky Subtitle a rel_Media_author jsou propojeny s podformuláři formuláře Media prostřednictvím relace n:1. Naproti tomu tabulky s relací 1:n k tabulce Media by měly být reprezentovány formuláři, které leží nad tabulkou Media. Protože takových tabulek je několik, jejich hodnoty se do hlavního formuláře zadávají pomocí polí seznamu.

Tabulka hlavního formuláře a tabulka podformuláře jsou v relaci 1:n nebo v některých výjimečných případech 1:1. Proto po delším používání databáze hlavní formulář obvykle spravuje tabulku, která má podstatně méně záznamů než tabulka patřící podformuláři.

Více hlavních formulářů nemůže obsahovat stejný podformulář. Proto není možné vytvořit uspořádání formuláře pro mnoho souběžných relací 1:n, ve kterých má podformulář stejný obsah. Pokud existuje relace 1:n pro tabulku patřící k formuláři, můžeme použít pole seznamu. Zde je z jiné tabulky k dispozici pouze několik termínů, jejichž cizí klíče lze tímto způsobem zadat do tabulky hlavního formuláře.

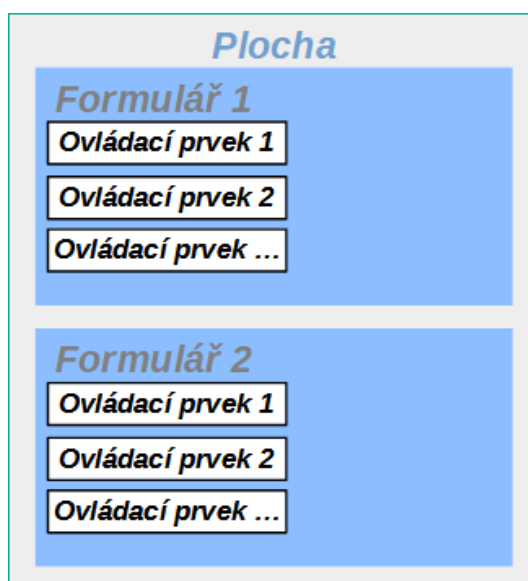


Obrázek 260: Schematické znázornění, které ukazuje, jak se pracuje se vztahy 1:n a n:1.

Pomocí polí seznamu lze hlavnímu formuláři založenému na tabulce Media přiřadit hodnoty z tabulek Category, Town nebo Publisher. Podformuláře slouží k propojení tabulek rel_Media_Author a Subtitle s hlavním formulářem a jeho prostřednictvím s tabulkou Media.

Podformulář pro tabulku rel_Media_Author se opět skládá ze dvou polí seznamu, aby nebylo nutné zadávat cizí klíče z polí Authors a Author_Add_ID (doplňky mohou být například editor, fotograf atd.) přímo jako čísla.

V případě formuláře pro zadávání médií je obvykle třeba postupně vyplňovat políčka seznamu v průběhu zadávání. Za tímto účelem jsou vedle hlavního formuláře zabudovány další formuláře. Existují nezávisle na hlavním formuláři.



Obrázek 261: Schematické znázornění uspořádání vícenásobného formuláře

Celkový formulář pro zadávání médií vypadá takto:

Obrázek 262: Formulář Media

Na levé straně je hlavní formulář s ohledem na vyhledávání a zadávání nových médií. Na pravé straně se nachází skupinové pole s popisem pole seznamu Editing Contents (Úprava obsahu, pozn. překl.), které poskytuje samostatnou oblast určenou pro vyplňování polí seznamu v hlavním formuláři. Pokud databáze neexistuje dlouho, bude často nutné do těchto polí provést

záznamy. Čím více položek je však k dispozici pro pole seznamu hlavního formuláře, tím méně často je třeba přistupovat k ovládacím prvkům tabulky ve skupinovém poli.

Všechny následující ovládací prvky tabulky jsou podřízeny hlavnímu formuláři, vstupnímu formuláři, jako jednotlivé podformuláře.

The screenshot shows a window titled "Editing Content of Listfields" containing six sub-forms, each representing a table with a list of items. Each sub-form has a table with columns and a "Záznam" (Record) control at the bottom.

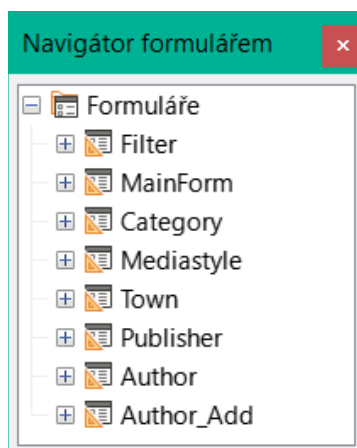
- Category:** Table with columns "Category" and "Description". Items: Fantasy, Liedermacher, Rock. Record: 1 z 3.
- Mediastyle:** Table with columns "Mediastyle" and "Loantime". Items: Buch (14 Days), CD (7 Days), DVD (7 Days). Record: 1 z 3.
- Town:** Table with columns "Town" and "Description". Items: Downtown, Dresden, Hamburg. Record: 1 z 5.
- Publisher:** Table with columns "Publisher" and "Description". Item: (empty). Record: 1 z 1.
- First Name Author / Last Name Author:** Table with columns "First Name Author" and "Last Name Author". Items: Dave Edmunds, Dr. Lutz Götze, Steven W. Hawking. Record: 1 z 12.
- Author Add:** Table with columns "Author Add" and "Description". Items: Geleitwort, Hrsg., Illustration. Record: 1 z 4.

Obrázek 263: Ovládací prvky tabulky na pravé straně formuláře Media

Zde se v každém případě zadávají kompletní údaje pro tabulku. V počátečních fázích je často nutné využít těchto vedlejších formulářů, protože v příslušné tabulce ještě není uloženo mnoho autorů.

Pokud je v některém z ovládacích prvků tabulky uložen nový záznam, je třeba v hlavním formuláři vyhledat příslušné pole seznamu a pomocí ovládacího prvku Aktualizovat (viz navigační panel) načíst nové hodnoty.

Navigátor formulářem zobrazuje odpovídající velký seznam formulářů.



Obrázek 264: Navigátor formulářem zobrazující komponenty nejvyšší úrovně formuláře Media

Formuláře byly jednotlivě pojmenovány, aby se usnadnilo jejich rozpoznání. Pouze hlavní formulář má stále název MainForm, který mu byl přidělen Průvodcem. Celkem existuje osm paralelních formulářů. Formulář Filter je hostitelem vyhledávací funkce, zatímco formulář MainForm je hlavním vstupním rozhraním. Všechny ostatní formuláře se vztahují k jednomu nebo druhému z výše uvedených ovládacích prvků tabulky.

Bez ovládacích prvků tabulky vypadá hlavní formulář poněkud jednodušeji.

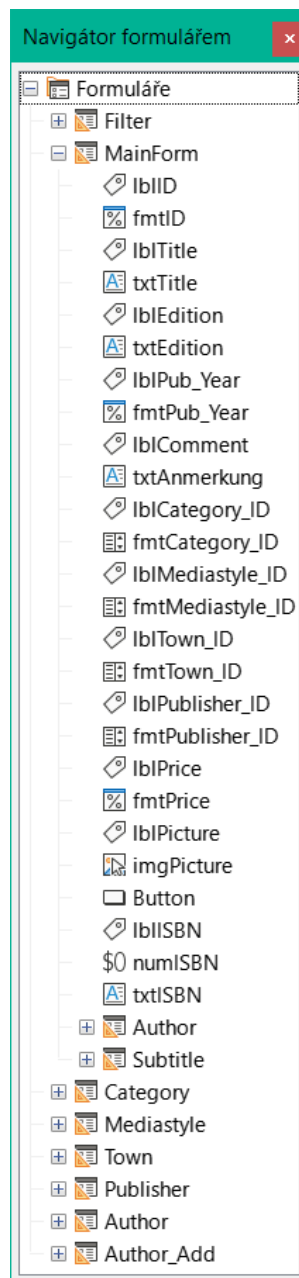
Author Sort.	Author	Author Add
1	van Veen, Herman	

No	Subtitle	Length
1	Amsterdam	
2	Hier unten am Deich	
3	Köln-Ehrenfeld	
4	Bei Mir	

Obrázek 265: Formulář médií bez šesti ovládacích prvků tabulky umístěných vpravo

Pole pro hledaný výraz se nachází ve formuláři Filter, dva ovládací prvky tabulky (pro autora a podtitul) se nacházejí v podformuláři hlavního formuláře Media Entry.

V okně Navigátor formulářem vypadá tento formulář mnohem složitěji, protože se zde zobrazují všechny ovládací prvky a popisky. V předchozím formuláři byla většina polí ve skutečnosti sloupci v ovládacích prvcích tabulky, a proto byla pro Navigátor formulářem neviditelná.

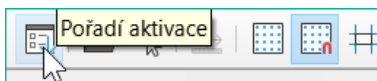


Obrázek 266: Navigátor formulářem zobrazující strukturu formuláře MainForm

Pořadí v Navigátoru formulářem nelze snadno změnit. Proto by se například zdálo rozumnější umístit podformuláře Subtitle a Author jako větve MainForm hned na začátku. V Navigátoru formulářem se však jednotlivé ovládací prvky a podformuláře zobrazují v pořadí, v jakém byly vytvořeny.

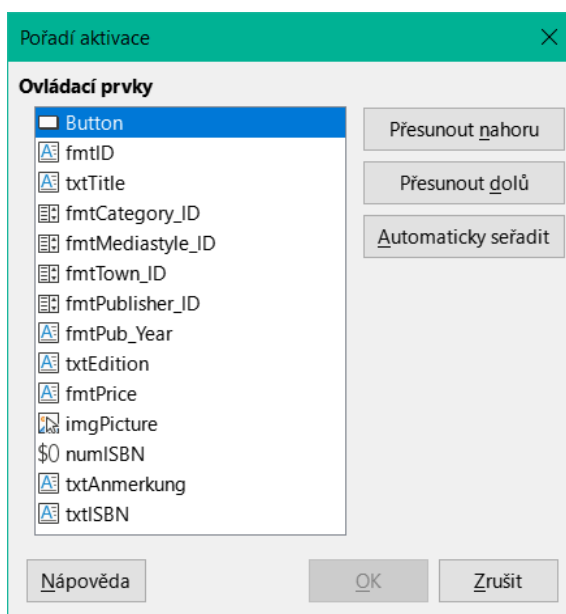
Průvodce formulářem opatřuje ovládací prvky určitými zkratkami, které se zobrazují vedle ikon a označují, o jaký typ ovládacího prvku se jedná. Popisky začínají písmenem 'lbl', textová pole písmenem 'txt' atd. Celkově poskytují popisky pouze sekundární informace pro zadávání údajů. Lze je také umístit přímo nad textové rámečky, které se pak v Navigátoru formulářem nezobrazují.

Pořadí prvků v navigátoru nemá nic společného s pořadím tabulátorů. To je určeno aktivačním příkazem.



Obrázek 267: Ikona Aktivační příkaz na nástrojové liště Návrh formuláře

Aktivační sekvence pro určitý formulář se vyvolá po výběru prvku formuláře a následném klepnutí na tlačítko Pořadí aktivace. U jednoduchého formuláře není nutné provádět úkony v tomto pořadí, ale pokud existuje mnoho paralelních formulářů, musí funkce vědět, který formulář je určen. Jinak je ve výchozím nastavení prvním formulářem v Navigátoru formulářem. Ve výše uvedeném příkladu obsahuje pouze jedno textové pole.



Obrázek 268: Dialog Pořadí na kartě

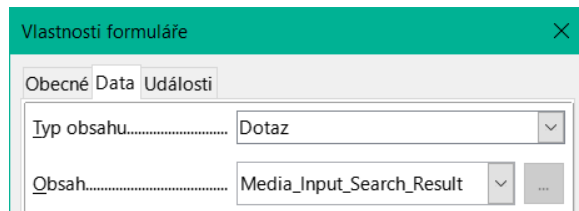
Pořadí aktivace umožňuje seřadit všechny prvky, které přenášejí data do podkladové tabulky formuláře nebo mohou provádět akce. To odpovídá nastavení aktivační sekvence ve vlastnostech ovládacího prvku uvedených v části "Výchozí nastavení mnoha ovládacích prvků".

Všimneme si, že v pořadí aktivace se objeví některé ovládací prvky, u kterých je krok tabulátoru ve skutečnosti vypnutý. Jsou sice zahrnuty v seznamu, ale ve skutečnosti nejsou při práci s formulářem přístupné z klávesnice.

Pole lze automaticky řadit v pořadí, v jakém se zobrazují na pozadí formuláře. Čím výše pole leží, tím dříve je v pořadí při použití automatického řazení. U polí se stejnou výškou je na prvním místě pole nejvíce vlevo. Toto třídění funguje bezchybně pouze tehdy, pokud byly prvky při vytváření formuláře přesně umístěny pomocí mřížky. V opačném případě je budeme muset upravit. Stačí vybrat ovládací prvek a pomocí tlačítka Přesunout nahoru nebo Přesunout dolů jej posunout v sekvenci výše nebo níže.

Pokud je k dispozici podformulář, přejde automatické řazení po dokončení hlavního formuláře přímo do podformuláře. V případě ovládacího prvku tabulky to způsobí, že kurzor je při zadávání z klávesnice uvězněn v tomto podformuláři; uvolnit jej lze pouze pomocí myši nebo stisknutím kláves Ctrl + Tab.

Funkce Automaticky seřadit funguje pro ovládací prvek tabulky pouze jednou. Následný podformulář s ovládacím prvkem tabulky nebude zahrnut. V úvahu se neberou ani paralelní formuláře. Automaticky seřadit nelze provést dodatečně pro podformulář s ovládacím prvkem tabulky. Podformulář musí být zcela odstraněn (dočasně přesunut do jiného formuláře).



Obrázek 269: Dialogové okno Vlastnosti formuláře (karta Data) pro formulář MainForm, zobrazující typ obsahu dotazu

Datovým podkladem pro formulář pro zadávání médií není v tomto případě tabulka, ale dotaz. To je nezbytné, protože formulář má sloužit nejen k zadávání záznamů, ale také k vyhledávání. Formulář obsahuje také textové pole, které po uložení zobrazuje, zda bylo zadáno číslo ISBN správné. I to je možné pouze pomocí komplexního dotazu. Pro pochopení souvislostí je proto nutné probrat základy dotazů, které jsou popsány v kapitole 5.

Chybové zprávy při zadávání do formulářů

Zde jsou stručně vysvětleny některé chybové zprávy, které se běžně vyskytují při prvním návrhu formuláře.

Pokus o vložení nuly do nenulovatelného sloupce: sloupec: ID tabulky: Název tabulky v příkazu ...

Některá pole nesmějí být prázdná. Pokud jsou v tabulce takto definovány (*NOT NULL*), zobrazí se toto chybové hlášení. Pokud je ve formuláři také deklarováno jako nenulové, zobrazí se také zpráva v mateřském jazyce s přesným názvem pole, které je třeba vyplnit.

Výše uvedená zpráva se velmi často objevuje v případě, že primární klíč, v tomto případě *ID*, nebyl do formuláře importován. Bylo navrženo jako automaticky přírůstkové pole, ale bohužel jej návrhář zapomněl definovat jako Automatická hodnota. Takže je třeba to buď opravit, nebo se pole musí objevit ve formuláři, aby bylo možné zadat hodnotu.

Definovat pole zpětně jako Automatickou hodnotu je někdy problematické, pokud má tabulka relace s jinými tabulkami nebo pokud se k tabulce přistupuje pomocí zobrazení. Aby bylo pole primárního klíče tabulky editovatelné, je třeba odstranit všechny odkazy. Obsah příkazu SQL, který byl použit k vytvoření zobrazení, lze dočasně uložit do dotazu.

Porušení omezení integrity – neexistuje nadřazená tabulka SYS_FK_95: Název tabulky v příkazu ...

Zde máme odkaz mezi tabulkou, která je základem formuláře, a jinou tabulkou. Tato tabulka má poskytnout svůj primární klíč pro použití v poli cizího klíče. Obvykle se to provádí pomocí pole se seznamem, které provede správný dotaz pro získání klíče. Pokud nepoužíváme pole se seznamem nebo pokud bylo pole se seznamem nesprávně zkonstruováno, může pole cizího klíče získat nesprávnou hodnotu, která ve zdrojové tabulce jako primární klíč ve skutečnosti není. To je to, co je míněno "*Integritní porušení omezení*". '*no parent SYS_FK_95*' znamená, že ve druhé tabulce, zdrojové tabulce „*Parents*“, není odpovídající hodnota indexu s názvem '*SYS_FK_95*'.

*.Chyby při zadávání nových záznamů
Chyby při spouštění funkcí*

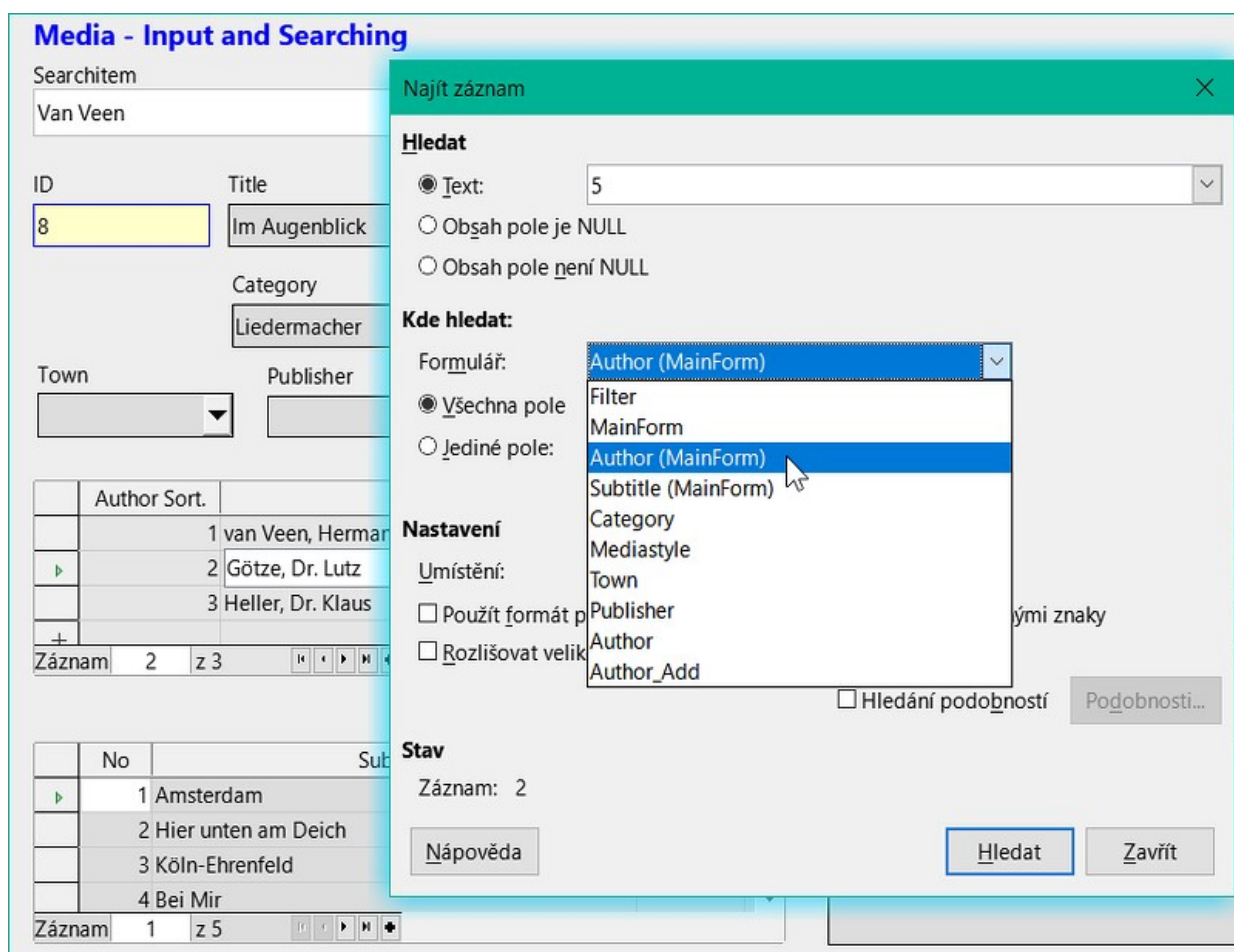
Předpokládejme, že formulář poskytuje data podformuláři. Aplikace Base to nepovažuje za změnu hodnoty pole. Grafické uživatelské rozhraní nabízí uložení hodnoty, ale záznam se zobrazí jako prázdný. Řešením je zahrnout do základní tabulky jednoduché pole, například pole ano/ne. Nyní před uložením toto pole odstraníme a záznam bude možné bez problémů uložit.

Hodnoty předávané prostřednictvím jiných formulářů se zřejmě do příslušných polí zadávají pouze tehdy, když je ve formuláři alespoň jedna další akce.

Vyhledávání a filtrování ve formulářích pomocí navigační lišty

Lišta navigace formuláře nabízí různé možnosti vyhledávání a filtrování. Jednotlivé prvky na liště navigace již byly uvedeny výše.

Vyhledávání záznamů pomocí parametrů



Obrázek 270: Vyhledávání v podformuláři Author formuláře MainForm pomocí dialogového okna Vyhledávání záznamů

Jednoduché vyhledávání záznamů s parametry je popsáno v kapitole 3, Tabulky. Nabízí značné množství nastavení. Charakteristickým rysem tohoto typu vyhledávání je, že se v podstatě prosívají všechny záznamy, místo aby se zobrazovaly pouze záznamy obsahující parametry vyhledávání.

Vyhledávání záznamů ve formulářích umožňuje prohledávat jak formulář, tak všechny podformuláře, ale není možné prohledávat všechny formuláře najednou.

Obrázek ukazuje přístup k podformuláři Form_author v hlavním formuláři MainForm. Jedná se o pole seznamu obsahující jména autorů. Nehledá se však text v poli seznamu, ale hodnoty, které jsou při použití pole zadány jako cizí klíče.

Tento typ vyhledávání má následující nedostatky:

- Vyhledávání parametrů je pro běžné použití příliš složité.
- Samotná funkce vyhledávání je velmi pomalá, protože nevyužívá dotazovací funkce databáze.
- Vyhledávání funguje vždy pouze pro jeden formulář, i když pro všechna pole. Dílčí formuláře musí být prohledávány samostatně.
- Hledáme podformulář, který patří k aktuálnímu hlavnímu formuláři. Vstupy do jiných podformulářů se nezobrazí. Nelze tedy například najít podtituly jiného média, než které je právě zobrazeno.
- Vyhledávání použité pro funkce pole seznamu na základě klíčových polí (cizích klíčů) uložených v tabulce. Zobrazená data nelze použít.

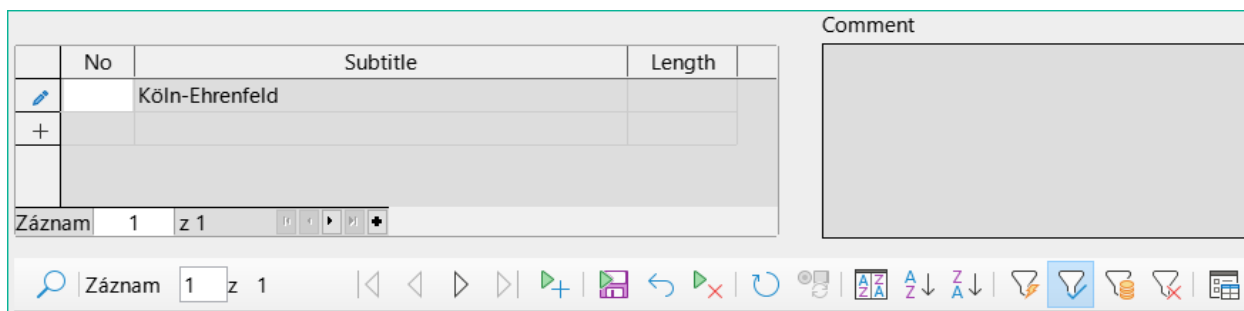
Filtrování pomocí automatického filtru

Automatický filtr lze vybrat přímo na liště navigace. Klepnutím na pole v hlavním formuláři se provede filtrování obsahu tohoto pole. Automatický filtr funguje také s poli seznamu a má zjevnou výhodu oproti ručnímu zadávání cizích klíčů.

The screenshot shows a web application interface for media management. The main form is titled "Media - Input and Searching". It contains several input fields: ID (8), Title (Im Augenblick), Category (Liedermacher), Mediastyle (CD), Town, Publisher, Pub-Year (2009), Edition, Price [\$] (\$20,00), and Picture (a portrait of Herman Van Veen). Below the form is a table with columns: Author Sort., Author, Author Add, and ISBN/ISSN. The table shows one record for "1 van Veen, Herman". At the bottom, there is a navigation bar with a search icon, a "Záznam 3 z 3" indicator, and a button labeled "Automatický filtr". A green arrow points from the "Automatický filtr" button to the "Mediastyle" dropdown menu.

Obrázek 271: Použití automatického filtru

Ve výše uvedeném příkladu je načten záznam, který má v poli Mediastyle hodnotu „CD“. Pak se přepne tlačítko automatického filtru. Z původních 9 záznamů v tabulce Media se nyní v počítadle záznamů zobrazují pouze 3 záznamy.



Obrázek 272: Výstup automatického filtru

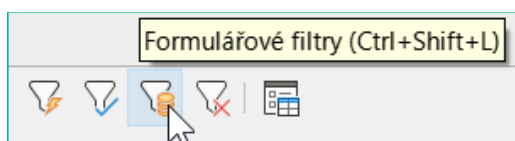
Přestože pouze dva záznamy mají v poli číslo „3“, hlavní formuláře všech ostatních záznamů se budou zobrazovat i nadále. V podformuláři se zobrazí pouze záznamy, které odpovídají hodnotě „3“.

Použitím automatického filtru však nelze vyřešit následující problémy:

- Pokud hledáme v podformuláři, zobrazí se v něm pouze zadaná hodnota, ale v hlavním formuláři se zobrazí všechny záznamy, i když danou hodnotu neobsahují. Pokud tedy například hledáme médium s druhým autorem, zobrazí se všechna média, která mají pouze jednoho autora. V těchto záznamech bude pole autor prázdné.
- Hodnota použitá pro filtr musí být konkrétní. Neexistuje žádná možnost vyhledávání podobností pro nalezení souvisejících materiálů.

Filtrování pomocí filtru založeného na formuláři

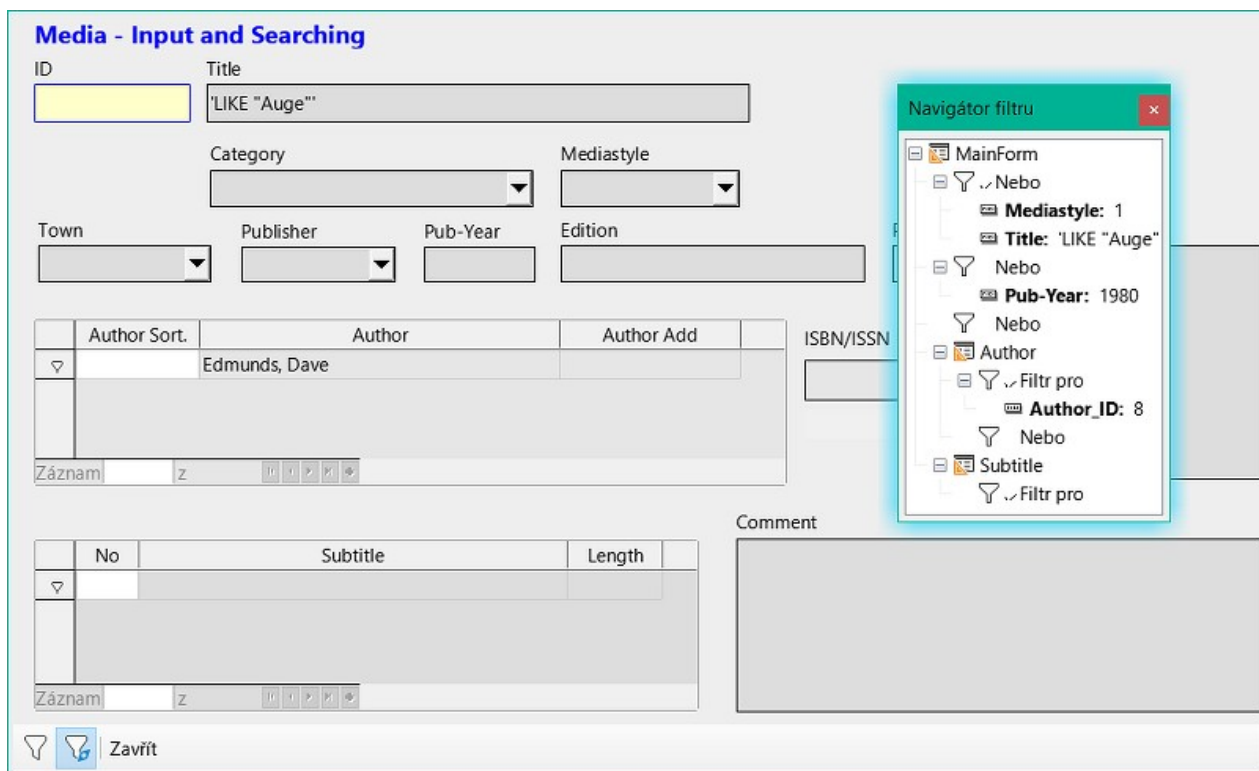
Filtr založený na formuláři nabízí formulář pro vložení skutečných hodnot, které se použijí pro filtrování, a nikoli pouze tlačítko filtru. Zde můžeme filtrovat několik hodnot najednou. Tyto hodnoty lze kombinovat buď jako společnou podmínku (AND) nebo jako alternativy (OR).



Obrázek 273: Ikona Filtry na základě formuláře na nástrojové liště Navigace formulářem

Zadané hodnoty filtru nemusí být jedinečné. Zde tedy můžeme formulovat podmínky podobné těm, které jsou popsány ve filtrování tabulek. Takže `LIKE '%eye%'` zadané do pole pro název média poskytne všechny záznamy, jejichž název obsahuje slovo „eye“.

Filtrování polí seznamu se provádí přímým výběrem obsahu zobrazeného pole seznamu. Podkladové cizí klíče není třeba zadávat.



Obrázek 274: Nastavení filtru založeného na formuláři

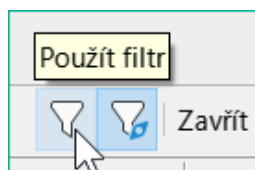
Zde hledáme všechny záznamy s mediastylem „CD“ (odpovídá hodnotě cizího klíče „1“) a zároveň mají někde v názvu sekvenci znaků „eye“. Při zadání `LIKE '*eye*'` nahradí grafické uživatelské rozhraní obvyklé zástupné symboly SQL „%“ znakem „*“, který je většině uživatelů známější.

Přidáme také podmínku, že se zobrazí všechny záznamy, u nichž je rok vydání „1980“.

Autor „Edmunds, Dave“ je vybrán z pole seznamu v podformuláři. Odpovídající hodnota cizího klíče je „8“.

Při zadávání se ve formulářovém filtru zobrazí pouze pole, která mají před tlačítkem NEBO zatržítka. Tímto způsobem lze pro pole nastavit několik podmínek. Filtr však potlačí zobrazení obsahu pole seznamu Mediastyle, zatímco v ovládacím prvku tabulky podformuláře to funguje.

Během vytváření filtru se nezobrazuje lišta navigace, ale volby filtru založené na formuláři. Filtrujeme podle svých požadavků, vložíme hodnoty filtru a pak jednoduše zavřeme zobrazení.



Obrázek 275: Ikona Použít filtr na základě formuláře na nástrojové liště Filtr formuláře

Filtr se použije a formulář se filtruje podle zadání.

Media - Input and Searching

ID: 0 Title: Der kleine Hobbit

Category: Fantasy Mediastyle: Buch

Town: Publisher: Pub-Year: 1972 Edition: 2. Aufl. Price [\$\$]: \$7,20 Picture:

Author Sort.	Author	Author Add	ISBN/ISSN
1			

Záznam 1 z 1

No	Subtitle	Length

Záznam 1 z 1

Comment:

Navigation: Záznam 1 z 5

Obrázek 276: Výsledky filtru založeného na formuláři

Je-li zapnuto filtrování, počet záznamů je omezen na ty, které odpovídají. V tomto případě existuje pět shod. Na obrázku se neshoduje ani název, ani typ média; podmínka, že rok vydání je menší než 1980, poskytuje shodu.

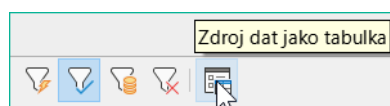
Podformulář ukazuje něco zajímavého. Protože i to má být filtrováno, autor knihy „Malý Hobit“ není zobrazen. Hodnota filtru je spojena s autorem Dave Edmunds. To opět zobrazuje logiku filtrování: používá vlastnosti filtru aktuálního formuláře. Podformuláře jsou filtrovány samostatně a filtrování podformuláře nemá vliv na hlavní formulář – a naopak.

Tento poměrně úspěšný filtr má stále následující nevýhody:

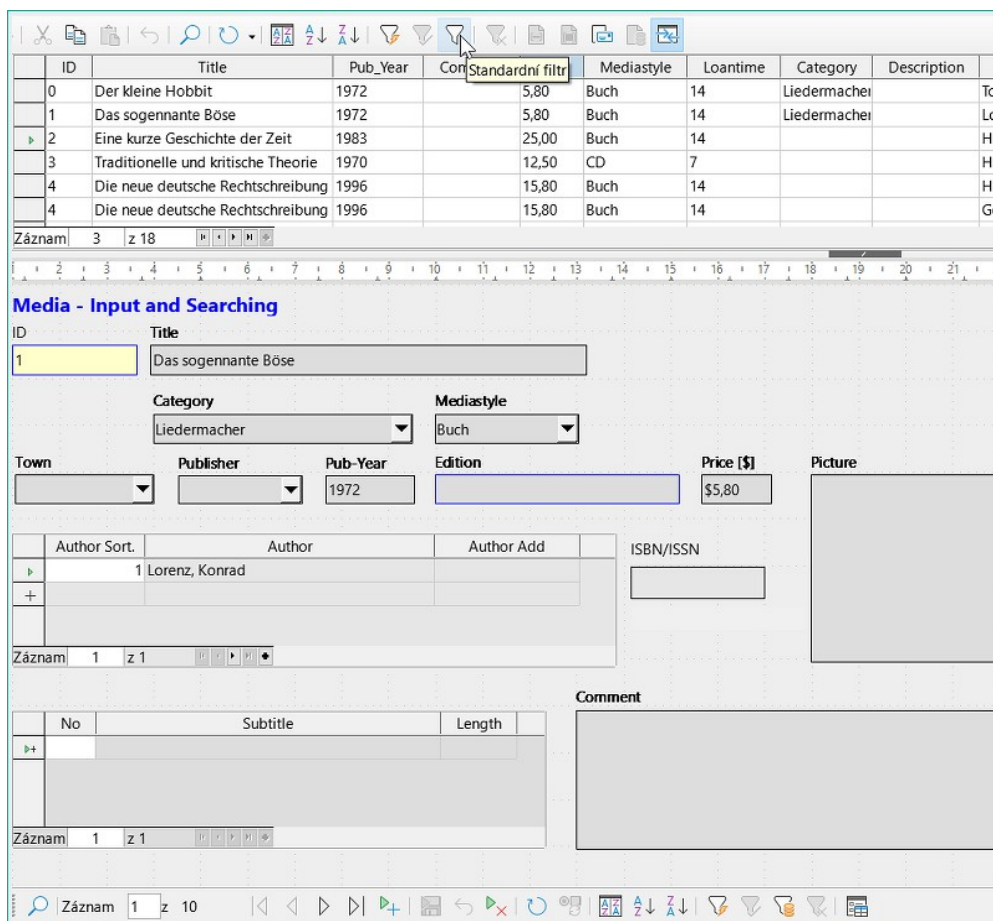
- Filtrování opět funguje pouze v hlavním formuláři. V podformuláři se stejně jako v případě automatického filtru nezobrazí hodnoty, které se neshodují. Na zobrazení hlavního formuláře to nemá žádný vliv. Proto se v hlavním formuláři zobrazuje obsah, který nemůže generovat žádný filtrovaný obsah pro podformulář.
- Pokud hledáme něco složitějšího než kompletní pole, potřebujeme podrobné znalosti o tom, jak nastavit podmínky. Většina uživatelů, i když jsou zvyklí používat vyhledávače, nemá potřebné dovednosti.

Filtrování pomocí výchozího filtru

Výchozího filtru lze dosáhnout vyžádáním zobrazení zdroje dat jako tabulky. Pak jsou postupy stejné jako při použití výchozího filtru v tabulce.



Obrázek 277: Zdroj dat jako ikona tabulky na nástrojové liště Navigace formulářem



Obrázek 278: Použití standardního filtru

V pravé části navigátoru formulářem se nachází tlačítko **Zdroj dat jako tabulka**. V pohledu na tabulku je k dispozici výchozí filtr. Na rozdíl od zobrazení skutečné tabulky Media jsou cizí klíče zobrazeny nikoli se svými skutečnými hodnotami, ale s obsahem, který jim odpovídá. Klepnutím na pole Mediastyle se ukáže, že se jedná o zdroj pole seznamu, který se zobrazí ve formuláři.

✓ Poznámka

Výše uvedené zobrazení upozorňuje na chybu, protože 13místné číslo ISBN není správně naformátováno. Místo toho je v příslušném poli uveden minimální možný počet číslic. Pokud má číslo více než 9 číslic, musíme být opatrní: úpravy mohou vést ke ztrátě dat (chyba 82411).

Tato chyba se projevuje také v ovládacích prvcích tabulky. Správně se zobrazí, pokud místo číselného pole použijeme pole formátované.

Zobrazení tabulky formuláře má pro tuto metodu vyhledávání následující nevýhody:

- Vyhledávání funguje pouze v rámci jedné tabulky nebo dotazu, který je základem formuláře, nikoli v podformuláři, který k němu patří.
- V zobrazení dat se zobrazují pole seznamu, ale nelze je použít k filtrování. Opět je třeba znát skutečné hodnoty cizích klíčů pro tato pole. Pokud je tedy například typ média „Book“ a tento typ má primární klíč hodnotu „1“, pak je v tabulce Media v tomto poli uložena hodnota „1“. Tuto hodnotu je třeba vyhledat, i když je ve složeném zobrazení uvedena hodnota „Book“.

Souhrn

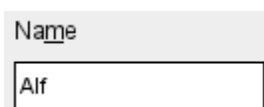
Funkce vyhledávání a filtrování jsou pro formuláře použitelné jen v omezené míře. Vyhledávání je příliš pomalé a neomezuje celkový počet záznamů na ty, které se skutečně shodují. Filtry fungují vždy pouze v rámci jednoho formuláře. Pokud chceme filtrovat v podformuláři, ovlivní to pouze tento podformulář. Správný počet odpovídajících záznamů jednoduše nelze získat.

Proto byla do formulářů ukázkové databáze integrována uživatelsky přívětivá funkce vyhledávání, což je postup, který je třeba vypracovat ručně a vyžaduje určité znalosti jazyka SQL.

Zadávání záznamů a navigace

Pokud je formulář navržen tak, že jeho otevření aktivuje zaměření ovládacího prvku, kurzor se zobrazí v prvním vstupním poli. Místo toho, abychom se myší přesouvali z pole do pole, přejdeme na další pole v pořadí pomocí klávesy tabulátoru.

Někdy formulář umožňuje přejít na jiné pole buď myší, nebo pomocí klávesové zkratky.



Obrázek 279: Pole Label zobrazující podtržený znak „m“

Pomocí **Vlastnosti: Textové pole > Popisek** je popisek přiřazen. Popisek tohoto pole se změní z „Name“ na „Na~me“. Ve formuláři je nyní písmeno „m“ na popisku podtrženo. Nyní můžeme na toto textové pole okamžitě přejít pomocí klávesové zkratky *Alt+m*. To funguje pouze v případě, že se kurzor již nachází v některém poli formuláře.

Jako zkratku lze v zásadě použít libovolné písmeno, protože skok se odehrává výhradně v rámci formuláře. Pokud však kurzor není zpočátku v ovládacím prvku formuláře, aktivují zkratky místo toho uživatelské rozhraní LibreOffice. Například klávesová zkratka „a“ by tedy nevedla ke skoku do pole Název, ale otevřela by nabídku Tabulka.

Přeskoky polí fungují pouze v rámci jednoho formuláře, nikoli ve strukturách, které zahrnují dílčí formuláře. V současné době není možné přejít z podformuláře zpět do hlavního formuláře.

Pokud je kurzor v ovládacím prvku tabulky, měl by vyskočit při použití *Ctrl + Tab*. Někdy se stává, že tato kombinace kláves nemá požadovaný účinek.

✓ Poznámka

Viz také příklad databáze Example_cursorjump_subform_mainform.odt

Zde je tedy alternativa:

V podformuláři se vytvoří tlačítko. Jeho název je „~New“. Znak tilda ("~") se používá, jak je popsáno výše, pro vytvoření klávesové zkratky. Takže *Alt + N* lze použít k aktivaci tlačítka. Pole Doplňkové informace pro tlačítko pojmenovává pole v hlavním formuláři, do kterého se má kurzor při aktivaci přenést. Tlačítko je vázáno na následující makro pomocí **Vlastnosti > Události > Provést akci**:

```
SUB JumpToMainform(oEvent AS OBJECT)
    DIM oField AS OBJECT
    DIM oForm AS OBJECT
    DIM oDoc AS OBJECT
    DIM oController AS OBJECT
```

```

DIM oView AS OBJECT
DIM stShortcut AS STRING
oField = oEvent.Source.Model
stShortcut = Mid(oField.Label, InStr(oField.Label, "~") + 1, 1)
oForm = oField.Parent.Parent
SELECT CASE stShortcut
    CASE "n"
        oForm.MoveToInsertRow()
    CASE "a"
        IF oForm.isLast() THEN
            oForm.MoveToInsertRow()
        ELSE
            oForm.Next()
        END IF
END SELECT

oDoc = thisComponent
oController = oDoc.getCurrentController()
oView = oController.getControl(oForm.getByName(oField.Tag))
oView.setFocus
END SUB

```

Makro lze použít pouze pro přechod zpět do hlavního formuláře. Lze jej také použít s výše uvedenými nastaveními k okamžitému vytvoření nového záznamu nebo pomocí dalšího tlačítka na navigačním panelu Další záznam k přeskočení na další záznam. Pokud se kurzor v hlavním formuláři nachází na posledním záznamu, skok povede na nový záznam. Další informace o makrech nalezneme v kapitole 9.

Přeskočení na tlačítko vyžaduje, aby bylo tlačítko viditelné. Může však být poměrně malé nebo mít dokonce šířku a výšku 0 cm. Pokud šířka a výška nejsou definovány, může se stát, že se tlačítko zobrazí jako čára přes celou obrazovku.

Tisk z formulářů

Formuláře mohou být konstruovány tak, aby je bylo možné přímo vytisknout. Chceme-li dosáhnout použitelných výsledků, musíme dbát na to, abychom prvky neumístili mimo tisknutelnou oblast. Zvolíme **Zobrazení > Normální**. Poté lze nastavit vlastnosti stránky. Barevné pozadí zde není výhodou.

Každý jednotlivý prvek lze z tisku vyloučit pomocí **Vlastnosti > Obecné > Tisknutelná**.

Pokud je databáze zaregistrována v LibreOffice (pomocí **Nástroje > Možnosti > LibreOffice Base > Databáze** nebo přímo v rámci procesu vytváření), lze formulář použít pro hromadnou korespondenci. Formulář se otevře k úpravám. Zdroje dat jsou přístupné pomocí **Pohled > Zdroje dat** nebo stisknutím *F4*. Databázová pole lze nyní přetáhnout do formuláře podle jejich záhlaví tabulky. Poté se formulář uloží. Pokud je nyní otevřen stejný formulář pro zadání, LibreOffice rozpozná, že se jedná o pole pro hromadnou korespondenci, a zeptá se, zda chceme dopisy vytisknout.

Podrobnosti o tom, jak zpracovat hromadnou korespondenci, jsou uvedeny v kapitole 7, Propojení s databázemi.

Vlastnosti: Textové pole

Obecné Data Události

Název..... Textové pole 1

Popisek.....

Maximální délka textu..... 0

Zapnuto..... Ano

Viditelné..... Ano

Jen pro čtení..... Ne

Tisknutelná..... Ano

Krok tabulátoru..... Ano

Pořadí aktivace..... 0

Ukotvení..... K odstavci

PoziceX..... 1,67 cm

PoziceY..... 4,17 cm

Šířka..... 3,00 cm

Výška..... 1,00 cm

Výchozí text..... Alf

Písmo..... (Výchozí)

Zarovnání..... Vlevo

Svislé zarovnání..... Výchozí

Barva pozadí..... Výchozí

Ohraničení..... Ploché

Barva ohraničení..... #C0C0C0

Typ textu..... Jednořádkový

Textové řádky zakončeny... CR+LF (Windows)

Posuvníky..... Žádné

Znak hesla.....

Skrýt výběr..... Ano

Další informace.....

Pomocný text.....

URL nápovědy.....

Obrázek 280: Dialogové okno Vlastnosti (karta Obecné) pro textové pole, kde je vlastnost Tisknutelné nastavena na hodnotu Ano.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 5 *Dotazy*

Obecné informace o dotazech

Dotazy do databáze jsou nejmocnějším nástrojem, který máme k dispozici pro praktické využití databází. Mohou spojit data z různých tabulek, v případě potřeby vypočítat výsledky a rychle vyfiltrovat konkrétní záznamy z velkého množství dat. Rozsáhlé internetové databáze, které lidé denně využívají, existují především proto, aby na základě promyšleného výběru klíčových slov poskytl uživateli rychlý a praktický výsledek z obrovského množství informací – včetně reklamy související s vyhledáváním, která lidi vybízí k nákupu.

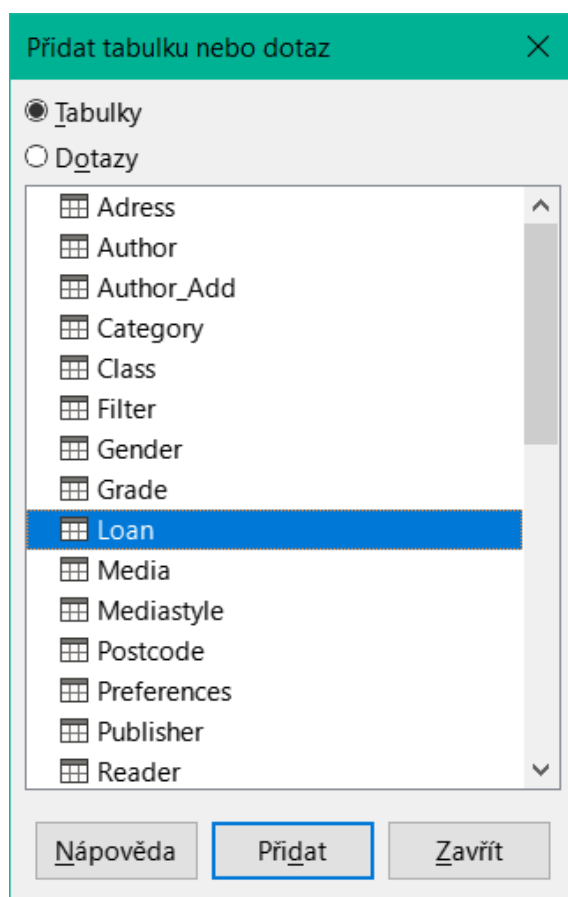
Base používá jazyk SQL (Structured Query Language), vyslovovaný jako "sequel", který je v podporovaných databázích podobný. Dotazy lze zadávat jak v grafickém uživatelském rozhraní, tak přímo jako kód SQL. V obou případech se otevře okno, ve kterém můžeme dotaz vytvořit a v případě potřeby také opravit.

Vytváření dotazů pomocí dialogového okna Návrh dotazu

Vytváření dotazů pomocí Průvodce je stručně popsáno v kapitole 8 příručky *Začínáme s LibreOffice*, *Začínáme s aplikací Base*. Zde vysvětlíme přímé vytváření dotazů v Režimu návrhu.

V této kapitole budeme nadále používat ukázkovou databázi *Media_without_Macros.odt*. V hlavním okně databáze klepneme v části Databáze na ikonu **Dotazy** a poté v části Úlohy klepneme na **Vytvořit dotaz v zobrazení návrhu**. Zobrazí se dvě dialogová okna. Jedno z nich poskytuje základ pro vytvoření návrhu pohledu na dotaz; druhý slouží k přidání tabulek, pohledů nebo dotazů k aktuálnímu dotazu.

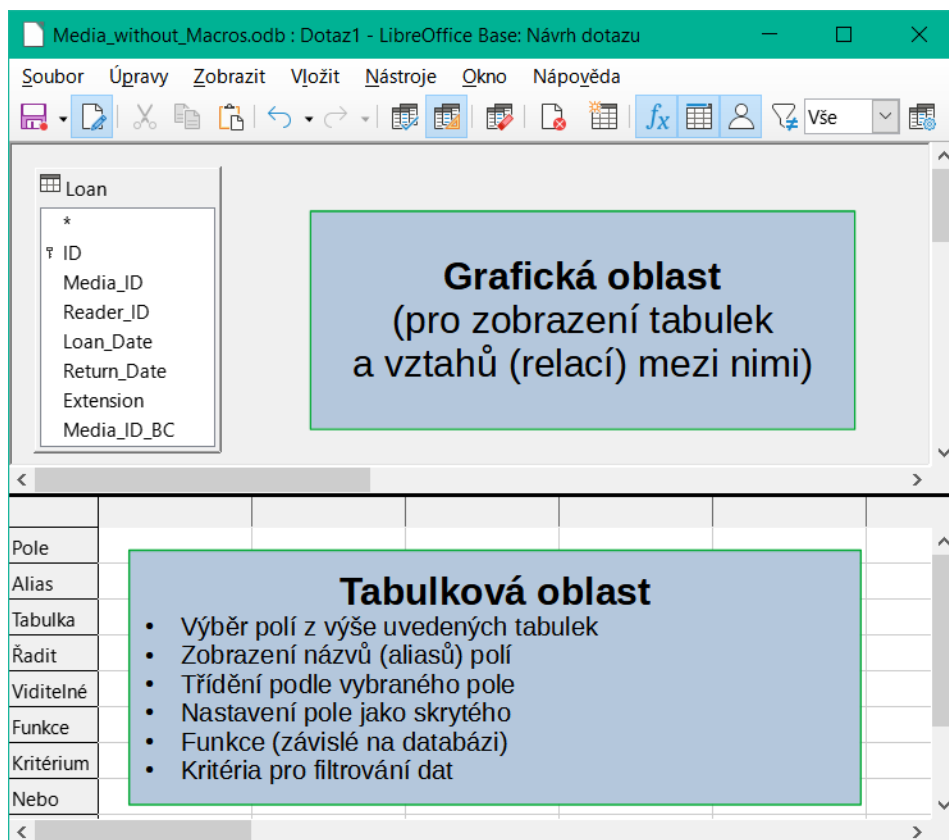
Protože náš jednoduchý formulář odkazuje na tabulku *Loan*, vysvětlíme si nejprve vytvoření dotazu pomocí této tabulky.



Obrázek 281: Dialogové okno Přidat tabulku nebo dotaz

Z dostupných tabulek vybereme tabulku Úvěr. Toto okno umožňuje kombinovat více tabulek (a také pohledů a dotazů). Chceme-li vybrat tabulku, klepneme na její název a poté na tlačítko Přidat. Nebo dvakrát klepneme na název tabulky. Obě metody přidají tabulku do grafické oblasti dialogového okna Návrh dotazu (obrázek 282).

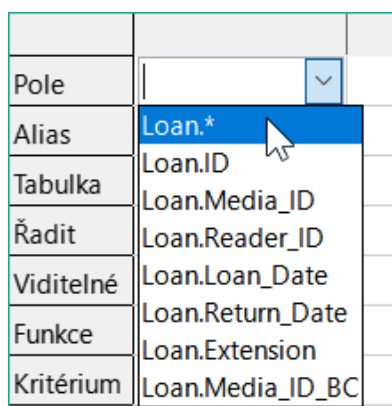
Po výběru všech potřebných tabulek klepneme na tlačítko **Zavřít**. V případě potřeby lze později přidat další tabulky a dotazy. Žádný dotaz však nelze vytvořit bez alespoň jedné tabulky, proto je třeba provést výběr na začátku.



Obrázek 282: Oblasti okna návrhu dotazu

Obrázek 282 zobrazuje základní rozdělení dialogového okna Návrh dotazu: v grafické oblasti jsou zobrazeny tabulky, které mají být s dotazem propojeny. Mohou být také zobrazeny jejich vzájemné relace ve vztahu k dotazu. Tabulková oblast slouží k výběru polí pro zobrazení nebo k nastavení podmínek týkajících se těchto polí.

Klepnutím na pole v prvním sloupci v tabulkové oblasti se zobrazí šipka dolů, jak je znázorněno na obrázku 283. Klepnutím na tuto šipku otevřeme rozevírací seznam dostupných polí. Formát je Název_tabulky.Název_pole – proto zde všechny názvy polí začínají slovem Loan.



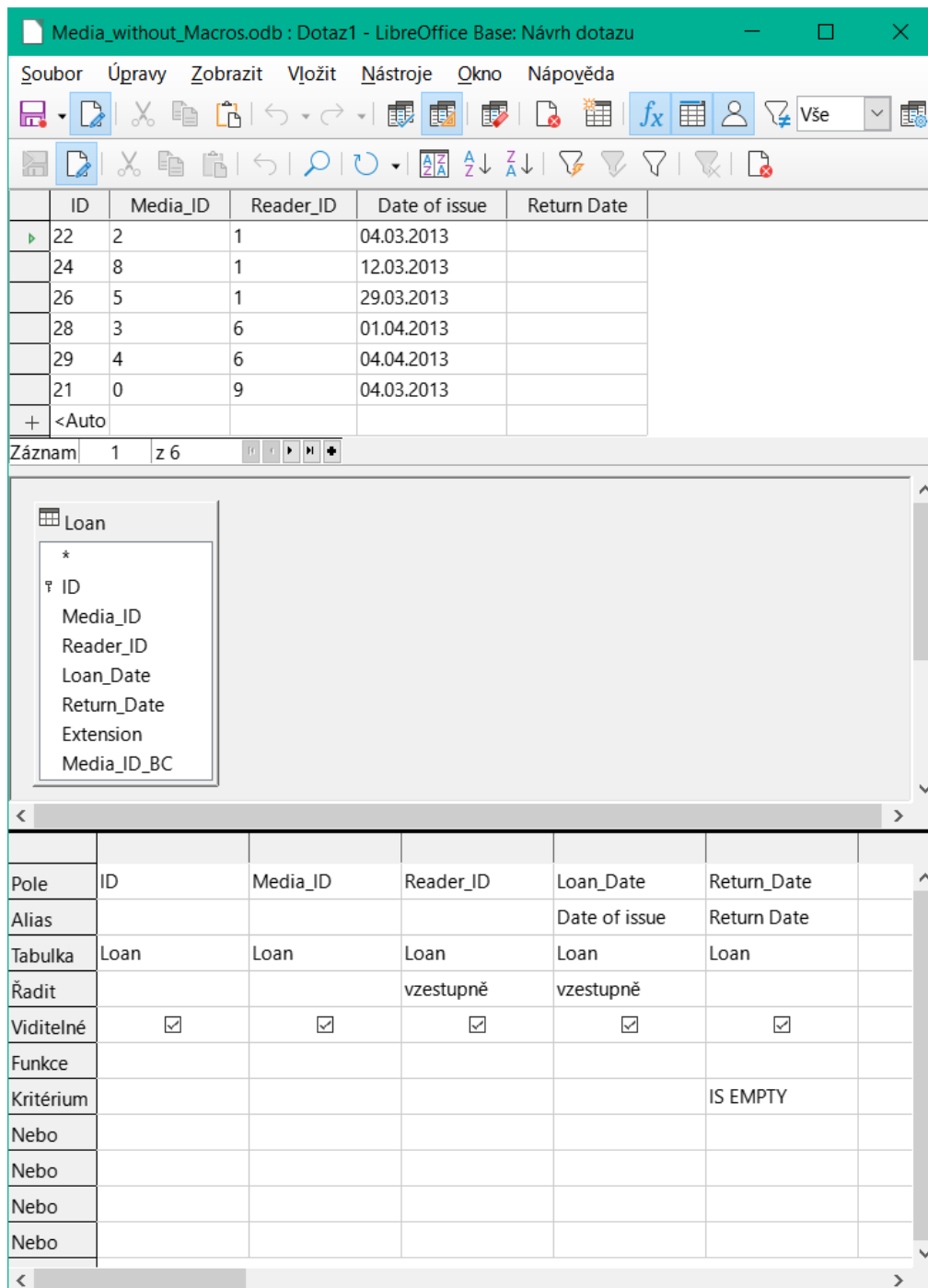
Obrázek 283: Rozevírací seznam dostupných polí

Označení vybraného pole Loan.* má zvláštní význam. Zde můžeme jedním klepnutím přidat do dotazu všechna pole ze zdrojové tabulky. Použijeme-li toto označení polí se zástupným znakem * pro všechna pole, dotaz se od tabulky nerozezná.

Tip

Chceme-li rychle přenést všechna pole tabulky do dotazu, stačí klepnout na zobrazení tabulky v grafickém rozhraní (Loan.* výše).

Dvojitým klepnutím na pole se toto pole vloží do tabulkové oblasti na nejbližší volnou pozici.



Obrázek 284: Okno návrhu dotazu zobrazující oblast náhledu obsahující výsledky dotazu

Prvních pět polí tabulky Loan je vybráno podle obrázku 284. Dotazy v režimu návrhu lze vždy spustit jako testy klepnutím na tlačítko Spustit dotaz. Tím se nad grafickým zobrazením tabulky

Loan se seznamem polí zobrazí tabulka s daty. Před uložením dotazu je vždy vhodné provést jeho testovací spuštění, aby bylo jasné, zda dotaz skutečně dosáhne svého cíle. Logická chyba často zabrání tomu, aby dotaz získal vůbec nějaká data. V jiných případech se může stát, že se zobrazí právě ty záznamy, které jsme chtěli vyloučit.

Dotaz, který v základní databázi vyvolá chybové hlášení, nelze v zásadě uložit, dokud není chyba opravena.

	ID	Media_ID
▶	22	2
	24	8
	26	5
	28	3
	29	4
	21	0
☀	<AutoF	

Obrázek 285: Upravitelný dotaz

	Media_ID	Rea
▶	2	1
	8	1
	5	1
	3	6
	4	6
	0	9

Obrázek 286: Needitovatelný dotaz

Ve výše uvedeném testu věnujeme zvláštní pozornost prvnímu sloupci výsledku dotazu. Značka aktivního záznamu (zelená šipka) se vždy zobrazuje na levé straně tabulky, zde ukazuje na první záznam jako na aktivní záznam. Zatímco první pole prvního záznamu na obrázku 285 je zvýrazněno, odpovídající pole na obrázku 286 zobrazuje pouze čárkovaný okraj. Zvýraznění označuje, že toto pole lze upravit. Jinými slovy, záznamy jsou editovatelné. Čárkovaný rámeček označuje, že toto pole nelze upravovat. Obrázek 285 obsahuje také další řádek pro zadání nového záznamu, přičemž pole ID je již označeno jako <AutoField>. To také ukazuje, že je možné zadávat nové položky.

i Tip

Základním pravidlem je, že pokud není v dotazu uveden primární klíč dotazované tabulky, není možné provést žádný nový záznam.

Pole	ID	Media_ID	Reader_ID	Loan_Date	Return_Date
Alias				Date of issue	Return Date

Obrázek 287: Definice aliasů pro pole Loan_Date a Return_Date

Pole Loan_Date a Return_Date mají aliasy, jak je znázorněno na obrázku 287. Tím nedojde k jejich přejmenování, ale pouze k tomu, že se uživateli dotazu zobrazí pod těmito názvy.

	ID	Media_ID	Reader_ID	Date of Issue	Return Date
▶	22	2	1	04.03.13	

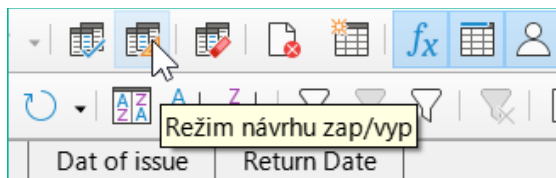
Obrázek 288: Pseudonymy používané v záhlavích sloupců výsledků dotazů

Zobrazení tabulky na obrázku 288 ukazuje, jak aliasy nahrazují skutečné názvy polí.

Return_Date
Return Date
Loan
<input checked="" type="checkbox"/>
IS EMPTY

Obrázek 289: Omezení zobrazených záznamů na záznamy s prázdným Return_Date

Pole Return_Date zobrazené na obrázku 289 je rovněž opatřeno vyhledávacím kritériem, které způsobí, že se zobrazí pouze ty záznamy, u nichž je pole Return_Date prázdné. (Do řádku Kritérium v poli Return_Date zadáme IS EMPTY.) Toto vylučovací kritérium způsobí, že se zobrazí pouze ty záznamy, které se týkají médií, která ještě nebyla vrácena.



Obrázek 290: Ikona Zapnout/vypnout zobrazení návrhu na nástrojové liště Návrh dotazu

Občasné přepínání mezi režimem návrhu a režimem SQL pomocí nástroje na obrázku 290 může pomoci při učení jazyka SQL.

The screenshot shows the 'Dotaz1 - Návrh dotazu' window in LibreOffice Base. It displays a table with the following data:

ID	Media_ID	Reader_ID	Date of issue	Return Date
22	2	1	04.03.2013	
24	8	1	12.03.2013	
26	5	1	29.03.2013	
28	3	6	01.04.2013	
29	4	6	04.04.2013	
21	0	9	04.03.2013	

Below the table, the SQL query is displayed:

```
SELECT "ID", "Media_ID", "Reader_ID", "Loan_Date" AS "Date of issue",
"Return_Date" AS "Return Date" FROM "Loan" WHERE "Return_Date" IS NULL
ORDER BY "Reader_ID" ASC, "Date of issue" ASC
```

Obrázek 291: Okno návrhu dotazu s vypnutým zobrazením návrhu, zobrazující základní kód SQL

Zde je na obrázku 291 zobrazen vzorec SQL vytvořený našimi předchozími volbami. Pro snazší čtení byly zařazeny některá zalomení řádků. Editor tyto zalomení řádků neukládá, takže při opětovném vyvolání dotazu se zobrazí jako jediný souvislý řádek zalomený na okraji okna.

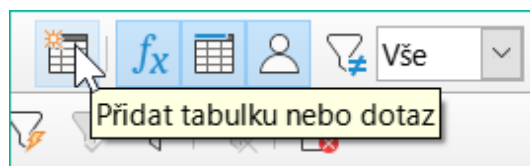
SELECT začíná výběrová kritéria. AS určuje aliasy polí, které se mají použít. FROM zobrazuje tabulku, která má být použita jako zdroj dotazu. WHERE udává podmínky dotazu, konkrétně, že pole Return_Date má být prázdné (IS NULL). ORDER BY definuje kritéria řazení, konkrétně vzestupné pořadí (ASC – ascending) pro dvě pole Reader_ID a Loan_Date. Tato specifikace třídění ilustruje, jak lze alias pole Loan_Date použít v samotném dotazu.

i Tip

Při práci v režimu zobrazení návrhu můžeme pomocí příkazu IS EMPTY požadovat, aby bylo pole prázdné. Při práci v režimu SQL použijeme IS NULL, což vyžaduje jazyk SQL (Structured Query Language).

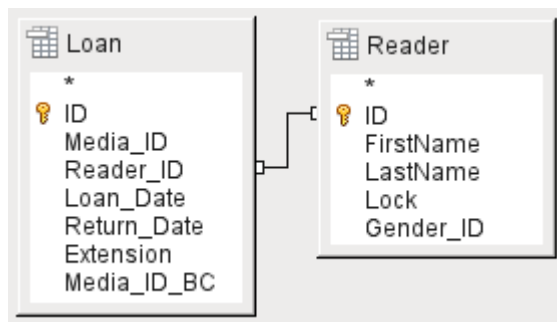
Pokud chceme řadit sestupně pomocí jazyka SQL, použijeme místo ASC volbu DESC.

Pole Media_ID a Reader_ID jsou zatím viditelná pouze jako číselná pole. Jména čtenářů jsou nejasná. Chceme-li je zobrazit v dotazu, musí být do dotazu zahrnuta tabulka Reader. Za tímto účelem se vrátíme do režimu návrhu. Poté lze do zobrazení Návrh přidat novou tabulku pomocí nástroje na obrázku 292.



Obrázek 292: Ikona Přidat tabulku nebo dotaz na panelu nástrojů Návrh

Zde lze následně přidat další tabulky nebo dotazy a zviditelnit je v grafickém uživatelském rozhraní. Pokud byly vazby mezi tabulkami deklarovány při jejich vytváření (viz kapitola 3, Tabulky), pak jsou tyto tabulky zobrazeny s příslušnými přímými vazbami na obrázku 293.



Obrázek 293: Tabulka čtenářů přidaná do grafické oblasti okna Návrh dotazu

Pokud odkaz chybí, lze jej v tomto okamžiku vytvořit přetažením myši z položky "Loan". "Reader_ID" na položku "Reader". "ID".

✓ Poznámka

Propojení tabulek funguje pouze v interní databázi nebo v externích relačních databázích. Například tabulky z tabulkového procesoru propojit nelze. Nejprve je třeba je importovat do interní databáze.

K vytvoření propojení mezi tabulkami stačí prostý import bez dodatečného vytvoření primárního klíče.

Nyní lze do tabulkové oblasti zadat pole z tabulky Reader. Pole jsou zpočátku přidána na konec dotazu, jak je znázorněno na obrázku 294.

	←		→	
Reader_ID	Loan_Date	Return_Date	FirstName	LastName
	Date of Issue	Return Date		
Loan	Loan	Loan	Reader	Reader

Obrázek 294: Změna pozice pole dotazu

Polohu polí lze v tabulkové oblasti editoru opravit pomocí myši. Takže například pole FirstName bylo na obrázku 295 přetaženo na pozici přímo před pole Loan_Date.

	ID	Media_ID	Reader_ID	FirstName	LastName	Date of issue	Return Date
▶	22	2	1	Heinrich	Müller	04.03.2013	
	24	8	1	Heinrich	Müller	12.03.2013	
	26	5	1	Heinrich	Müller	29.03.2013	
	28	3	6	Greta	Garbo	01.04.2013	
	29	4	6	Greta	Garbo	04.04.2013	
	21	0	9	Terence	Nobody	04.03.2013	

Záznam 1 z 6

Obrázek 295: Aktualizovaná tabulková oblast okna Návrh dotazu

Nyní jsou jména viditelná. Pole Reader_ID se stalo nadbytečným. Také řazení podle LastName a FirstName dává větší smysl než řazení podle Reader_ID.

Tento dotaz již není vhodný pro použití jako dotaz, který umožňuje zadávat nové záznamy do výsledné tabulky, protože v něm chybí primární klíč pro přidanou tabulku Reader. Teprve pokud je tento primární klíč zabudován, je dotaz opět editovatelný. Ve skutečnosti je pak zcela upravitelný, takže lze měnit i jména čtenářů. Z tohoto důvodu je možnost upravovat výsledky dotazů možností, kterou je třeba používat s nejvyšší opatrností, v případě potřeby pod kontrolou formuláře.

! Upozornění

Problémy může způsobit dotaz, který můžeme upravovat. Úpravou dat v dotazu se upravují také data v podkladové tabulce a záznamy v ní obsažené. Údaje nemusí mít stejný význam. Například změním jméno čtenáře, a tím změním i to, jaké knihy si čtenář vypůjčil a vrátil.

Pokud musíme data upravovat, prováděme to ve formuláři, abychom viděli účinky úprav dat.

I když lze dotaz dále upravovat, není tak snadné jej používat jako formulář s poli se seznamem, který zobrazuje jména čtenářů, ale obsahuje Reader_ID z tabulky. Pole se seznamem nelze přidat do dotazu; lze je použít pouze ve formulářích.

```
SELECT "Loan"."ID", "Loan"."Media_ID", "Loan"."Reader_ID",
"Reader"."FirstName", "Reader"."LastName", "Loan"."Loan_Date" AS "Date of
Issue", "Loan"."Return_Date" AS "Return Date"
FROM "Loan", "Reader"
WHERE "Loan"."Reader_ID" = "Reader"."ID" AND "Loan"."Return_Date" IS NULL
ORDER BY "Loan"."Reader_ID" ASC, "Date of Issue" ASC
```

Obrázek 296: Kód SQL pro dotaz obsahující pole z více tabulek

Pokud se nyní přepneme zpět do zobrazení SQL, vidíme na obrázku 296, že všechna pole jsou nyní zobrazena v dvojitém uvozovkách: "Název_tabulky". "Název_pole" oddělená tečkou. To je nutné, aby databáze věděla, ze které tabulky pocházejí dříve vybraná pole. Vždyť pole v různých tabulkách mohou mít snadno stejné názvy. Ve výše uvedené struktuře tabulky to platí zejména pro pole ID.

✓ Poznámka

Následující dotaz funguje bez uvedení názvů tabulek před názvy polí:

```
SELECT "ID", "Number", "Price" FROM "Stock", "Dispatch" WHERE
"Dispatch"."stockID" = "Stock"."ID"
```

Zde je ID převzato z tabulky, která je v definici FROM na prvním místě. Definice tabulky ve vzorci WHERE je také zbytečná, protože stockID se vyskytuje pouze jednou (v tabulce Dispatch) a ID bylo jednoznačně převzato z tabulky Stock (z pozice tabulky v dotazu).

Pokud má pole v dotazu alias, lze na něj odkazovat – například při třídění – pomocí tohoto aliasu, aniž by byl uveden název tabulky. Třídění se provádí v grafickém uživatelském rozhraní podle pořadí polí v tabulkovém zobrazení. Pokud chceme místo toho řadit nejprve podle "Loan_Date" a poté podle "Loan". "Reader_ID", lze to provést, pokud:

- Změní se pořadí polí v oblasti tabulky grafického uživatelského rozhraní (přetáhneme "Loan_Date" vlevo od "Loan". "Reader_ID" nebo "Loan_ID").
- Přidá se další pole, nastavené jako neviditelné, pouze pro třídění (editor jej však zaregistruje pouze dočasně, pokud pro něj nebyl definován alias) [přidáme další pole "Loan_Date" těsně před "Loan". "Reader_ID" nebo přidáme další pole "Loan". "Reader_ID" těsně za "Loan_Date"], nebo
- Text příkazu ORDER BY v SQL editoru se odpovídajícím způsobem změní (ORDER BY "Loan_Date", "Loan". "Reader_ID").

Tip

Dotaz může vyžadovat pole, které není součástí výstupu dotazu. V grafice v následující části je příklad dotazu Return_Date. Tento dotaz vyhledává záznamy, které neobsahují datum vrácení. Toto pole poskytuje kritérium pro dotaz, ale neobsahuje žádné užitečné viditelné údaje.

Použití funkcí v dotazu

Použití funkcí umožňuje dotazu poskytnout více než jen filtrovaný pohled na data v jedné nebo více tabulkách. Dotaz na obrázku 297 vypočítá, kolik médií bylo vypůjčeno v závislosti na identifikátoru Reader_ID.

Pole	ID	Reader_ID	Return_Date
Alias	Count		
Tabulka	Loan	Loan	Loan
Řadit			
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Funkce	Count	Seskupit	
Kritérium			IS EMPTY

Obrázek 297: Použití funkce Count k výpočtu počtu vypůjčených položek pro každé Reader_ID

Pro ID tabulky Loan je vybrána funkce Count. V zásadě je jedno, které pole tabulky je pro tento účel vybráno. Jedinou podmínkou je, že pole nesmí být v žádném ze záznamů prázdné. Z tohoto důvodu je nejvhodnější volbou pole primárního klíče, které není nikdy prázdné. Započítávají se všechna pole s jiným obsahem než NULL.

Pro identifikátor Reader_ID, který umožňuje přístup k informacím o čtenáři, je zvolena funkce Seskupení. Tímto způsobem jsou záznamy se stejným Reader_ID seskupeny dohromady. Výsledek na obrázku 298 ukazuje počet záznamů pro každé Reader_ID.

Jako kritérium vyhledávání je Return_Date nastaveno na "IS EMPTY", stejně jako v předchozím příkladu. (Níže je uveden následující SQL příkaz WHERE "Return_Date" IS NULL.)

	Count	Reader_ID
▶	1	9
	3	1
	2	6

Záznam 1 z 3

```
SELECT COUNT( "ID" ) AS "Count", "Reader_ID"
FROM "Loan"
WHERE "Return_Date" IS NULL
GROUP BY "Reader_ID"
```

Obrázek 298: Výsledky a kód SQL pro dotaz na zjištění počtu vypůjčených položek pro každé Reader_ID

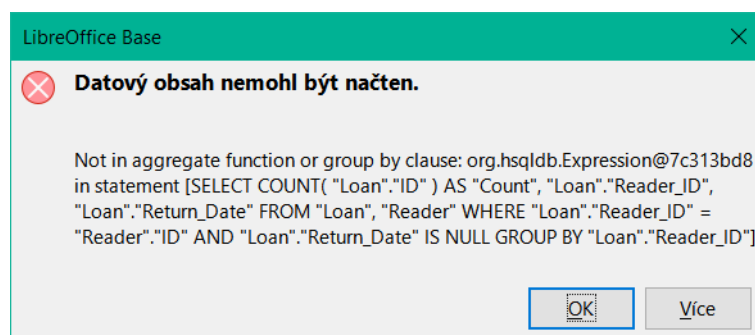
Výsledek dotazu ukazuje, že Reader_ID '0' má celkem 3 vypůjčená média. Kdyby byla funkce Count přiřazena k Return_Date místo k ID, měl by každý Reader_ID na vypůjčce médium '0', protože Return_date je předdefinován jako NULL.

Odpovídající vzorec v kódu SQL je rovněž uveden na obrázku 298.

Celkově grafické uživatelské rozhraní poskytuje funkce uvedené vpravo, které odpovídají funkcím v základní databázi HSQLDB.

Vysvětlení funkcí nalezneme v části „Vylepšení dotazu pomocí režimu SQL“ na straně 269.

Pokud je jedno pole v dotazu spojeno s funkcí, musí být všechna zbývající pole uvedená v dotazu také spojena s funkcemi, pokud mají být zobrazena. Pokud to není zajištěno, zobrazí se chybové hlášení na obrázku 299:



Obrázek 299: Dialogové okno s chybou zobrazené v případě, že pole není přiřazeno k funkci

Poněkud volný překlad by zněl: Následující výraz neobsahuje žádnou agregační funkci ani seskupení.

Tip

Při použití režimu zobrazení návrhu je pole viditelné pouze tehdy, pokud je v řádku Viditelné zaškrtnuto. Při použití SQL režimu je pole viditelné pouze tehdy, když následuje za klíčovým slovem SELECT.

✓ Poznámka

Pokud pole není spojeno s funkcí, je počet řádků ve výstupu dotazu určen podmínkami vyhledávání. Pokud je pole spojeno s funkcí, je počet řádků ve výstupu dotazu určen tím, zda je či není provedeno seskupení. Pokud nedojde k seskupení, je ve výstupu dotazu pouze jeden řádek. Pokud existuje seskupení, počet řádků odpovídá počtu různých hodnot, které má pole se zvolenou funkcí seskupení. Všechna viditelná pole tedy musí být buď spojena s funkcí, nebo musí být spojena s příkazem seskupení, aby se zabránilo tomuto konfliktu ve výstupu dotazu.

Poté je v chybové zprávě uveden celý dotaz, ale bez konkrétního názvu chybného pole. V tomto případě bylo pole Return_Date přidáno jako zobrazované pole. Toto pole nemá žádnou přidruženou funkci a není zahrnuto ani v příkazu pro seskupení.

Informace poskytnuté pomocí tlačítka Více zahrnují kód chyby SQL, ale pro běžného uživatele databáze nemusí být příliš poučné.

Chceme-li chybu opravit, odstraníme zaškrtnutí v řádku Viditelné u pole Return_Date. Jeho vyhledávací podmínka (Kritérium) je použita při spuštění dotazu, ale není viditelná ve výstupu dotazu.

Pomocí grafického uživatelského rozhraní lze provádět základní výpočty a používat další funkce.

Pole	ID	Media_ID	Reader_ID	Date	Count("Recall"."Date") * 2	Return_Date	
Alias				RecallCount	RecallAmount		
Tabulka	Loan	Loan	Loan	Recall		Loan	
Řadit							
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Funkce	Seskupit	Seskupit	Seskupit	Count			
Kritérium						IS EMPTY	

Obrázek 300: Dotaz na výpočet pokuty za položku u opožděných médií

Předpokládejme, že knihovna nevydává upomínky, když má být položka vrácena, ale vydává upomínky za zpoždění v případech, kdy výpůjční lhůta uplynula a položka nebyla vrácena. To je běžná praxe ve školních a veřejných knihovnách, které půjčují pouze na krátkou, pevně stanovenou dobu. V tomto případě vydání oznámení o prodloužení automaticky znamená, že musí být zaplacená pokuta. Jak tyto pokuty vypočítáme?

V dotazu na obrázku 300 jsou tabulky Loan a Recalls dotazovány společně. Z počtu datových záznamů v tabulce Recalls se zjistí celkový počet oznámení o vrácení. Pokuta za zpožděná média je v dotazu stanovena na 2,00 USD. Místo názvu pole je označení pole uvedeno jako Count(Recalls.Date)*2. Grafické uživatelské rozhraní přidá uvozovky a převede výraz „count“ na příslušný příkaz SQL.

! Upozornění

Pouze pro ty, kteří používají čárku jako oddělovač desetinných míst:

Pokud chceme zadávat čísla s desetinnými místy pomocí grafického uživatelského rozhraní, musíme zajistit, aby se v konečném příkazu SQL použila jako oddělovač desetinných míst desetinná tečka, nikoli čárka. Jako oddělovače polí se používají čárky, takže pro desetinnou část se vytvoří nové pole dotazu.

Položka s čárkou v zobrazení SQL vždy vede k dalšímu poli obsahujícímu číselnou hodnotu desetinné části, jak je znázorněno na obrázku 301.

	ID	Media_ID	Reader_ID	RecallCount	RecallAmount
▶	24	8	1	1	2
	22	2	1	1	2

Record 1 of 2

```

SELECT "Loan"."ID", "Loan"."Media_ID", "Loan"."Reader_ID",
COUNT( "Recall"."Date" ) AS "RecallCount",
COUNT( "Recall"."Date" ) * 2 AS "RecallAmount"
FROM "Recall", "Loan"
WHERE "Recall"."Loan_ID" = "Loan"."ID"
AND "Loan"."Return_Date" IS NULL
GROUP BY "Loan"."ID", "Loan"."Media_ID", "Loan"."Reader_ID"

```

Obrázek 301: Výsledky a kód SQL pro dotaz na výpočet pokuty za položku pro opožděné vrácení médií

Dotaz nyní pro každé médium, které je stále zapůjčeno, zobrazuje pokuty, které narostly na základě vydaných oznámení o vrácení, a pole pro dodatečnou multiplikaci. Struktura dotazu na obrázku 302 bude užitečná i pro výpočet dlužných pokut od jednotlivých uživatelů.

Pole	Reader_ID	Date	Count("Recall"."Date") * 2	Return_Date
Alias		RecallCount	RecallAmount	
Tabulka	Loan	Recall		Loan
Řadit				
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Funkce	Seskupit	Count		
Kritérium				IS EMPTY

Obrázek 302: Dotaz na výpočet pokut za zpoždění médií pro každé Reader_ID

Pole "Loan". "ID" a "Loan". "Media_ID" byla odstraněna. Byly použity v předchozím dotazu k vytvoření samostatného záznamu pro každé médium. Nyní budeme seskupovat pouze podle čtenáře. Výsledek dotazu je zobrazen na obrázku 303:

	Reader_ID	RecallCount	RecallAmount
▶	1	2	4

Záznam 1 z 1

Obrázek 303: Výsledky dotazu na výpočet pokut za opožděné vrácení médií pro každé Reader_ID

Namísto samostatného výpisu médií pro Reader_ID = 0 byla započítána všechna pole "Recalls". "Date" a celková částka 8,00 USD byla zadána jako dlužná pokuta.

Definice relace v dotazu

Při vyhledávání dat v tabulkách nebo formulářích je vyhledávání obvykle omezeno na jednu tabulku nebo jeden formulář. Dokonce ani cesta z hlavního formuláře do podformuláře není navigovatelná pomocí vestavěné vyhledávací funkce. Takové údaje je lepší shromažďovat pomocí dotazu.

	Title
▶	Der kleine Hobbit
	Das sogenannte Böse
	Eine kurze Geschichte der Zeit
	Traditionelle und kritische Theorie
	Die neue deutsche Rechtschreibung
	I hear you knocking
	Datenbanken mit OpenOffice.org 3
	Das Postfix-Buch
	Im Augenblick

Záznam 1 z 9

Pole	Title
Alias	
Tabulka	Media
Řadit	
Viditelné	<input checked="" type="checkbox"/>

Obrázek 304: Dotaz na zobrazení názvů z tabulky Media

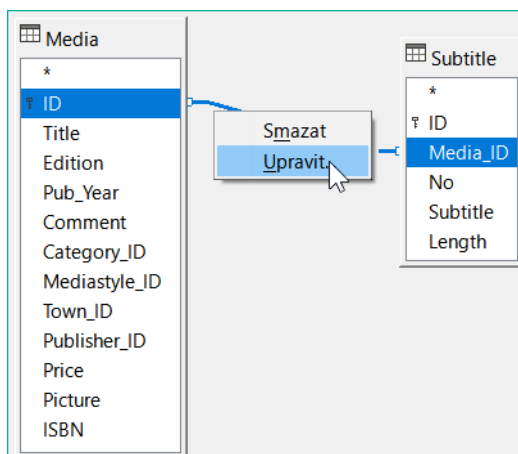
	Title	Subtitle
▶	I hear you knocking	Youn can't catch me
	I hear you knocking	The stumble
	I hear you knocking	Sabre dance (Single version)
	Im Augenblick	Amsterdam
	Im Augenblick	Hier unten am Deich
	Im Augenblick	Köln-Ehrenfeld
	Im Augenblick	Bei Mir
	Im Augenblick	Gott sei Dank

Záznam 1 z 8

Pole	Title	Subtitle
Alias		
Tabulka	Media	Subtitle
Řadit		
Viditelné	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

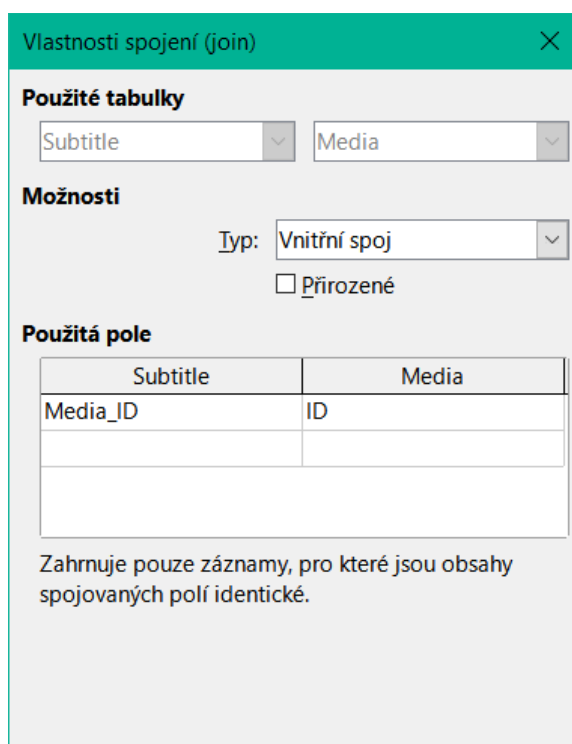
Obrázek 305: Dotaz na zobrazení názvů a titulků z tabulky Media

Jednoduchý dotaz na pole Title z tabulky Media na obrázku 304 zobrazuje testovací záznamy pro tuto tabulku, celkem 9 záznamů. Pokud však do dotazovací tabulky zadáte položku Subtitle, obsah záznamu v tabulce Media se zmenší na pouhé 2 tituly. Pouze pro tyto dva Titles (tituly, pozn. překl.) jsou v tabulce uvedeny také Subtitles (podtituly, pozn. překl.), jak ukazuje obrázek 305. U všech ostatních titulů podtituly neexistují. To odpovídá podmínce spojení uvedené na obrázku 306, že se mají zobrazit pouze ty záznamy, u nichž se pole Media_ID v tabulce Subtitles rovná poli ID v tabulce Media. Všechny ostatní záznamy jsou vyloučeny.



Obrázek 306: Úprava spojení v rámci dotazu

Aby se zobrazily všechny požadované záznamy, je třeba otevřít podmínky spojení pro úpravy. V tomto případě nemáme na mysli spojení mezi tabulkami v rámci návrhu relací, ale spojení v rámci dotazů.



Obrázek 307: Dialogové okno Vlastnosti spojení zobrazující výchozí nastavení Typ vnitřního spojení

Ve výchozím nastavení jsou relace nastaveny jako vnitřní spojení, což znamená, že dotaz vrátí pouze záznamy, jejichž obsah souvisejících polí obou tabulek je totožný. Na obrázku 307 je zobrazeno okno s informacemi o tom, jak tento typ spojení funguje v praxi.

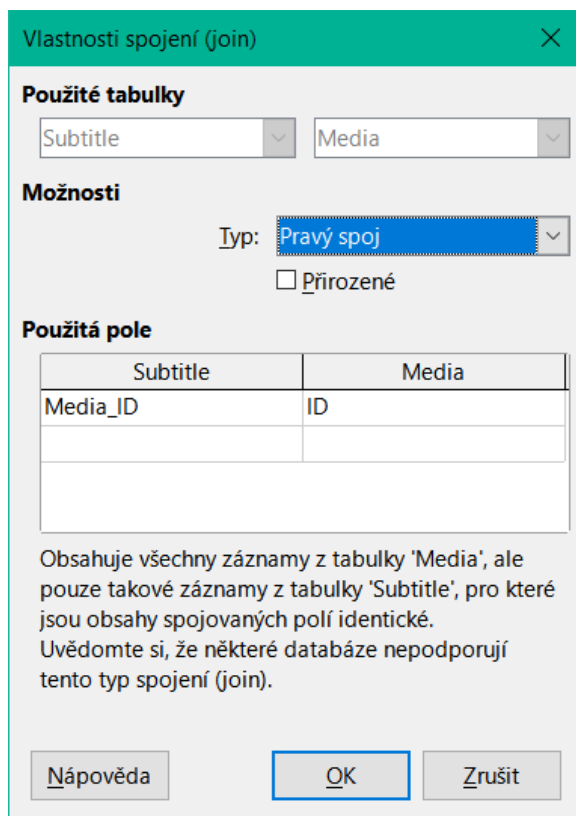
Dvě dříve vybrané tabulky jsou uvedeny v seznamu Použité tabulky. Zde je nelze vybrat. Příslušná pole z obou tabulek se načtou z definic tabulek. Pokud není v definici tabulky zadán žádný vztah, lze jej v tomto okamžiku pro dotaz vytvořit. Pokud jsme však databázi naplánovali přehledně pomocí HSQLDB, nemělo by být nutné tato pole měnit.

Nejdůležitějším nastavením je možnost Join. Zde lze relace zvolit tak, aby byly vybrány všechny záznamy z tabulky Subtitles, ale pouze ty záznamy z Media, které mají v tabulce Subtitles zadaný titulek.

Nebo můžeme zvolit opačný postup: že se v každém případě zobrazí všechny záznamy z tabulky Media bez ohledu na to, zda mají podtitul.

Volba Přirozené určuje, že propojená pole v tabulkách jsou považována za rovnocenná. Tomuto nastavení se můžeme vyhnout také správným definováním relací na samém začátku plánování databáze.

Pro typ Pravý spoj popis ukazuje, že se zobrazí všechny záznamy z tabulky Media (Subtitle RIGHT JOIN Media). Vzhledem k tomu, že v Media neexistuje žádný Subtitle, kterému by chyběl název, ale v Media určitě existují Titles, kterým chybí Subtitles, je tato volba správná, jak ukazuje obrázek 308.



Obrázek 308: Dialogové okno Vlastnosti spojení s typem nastaveným na Pravé spojení

Po potvrzení Pravého spoje vypadají výsledky dotazu tak, jak jsme chtěli. Title a Subtitle se zobrazují společně v jednom dotazu, který je znázorněn na obrázku 309. Samozřejmě se tituly objevují vícekrát, stejně jako u předchozí relace. Pokud se však nepočítají shody, lze tento dotaz dále použít jako základ vyhledávací funkce. Viz fragmenty kódu v této kapitole, v kapitole 8, Úlohy databáze, a v kapitole 9, Makra.

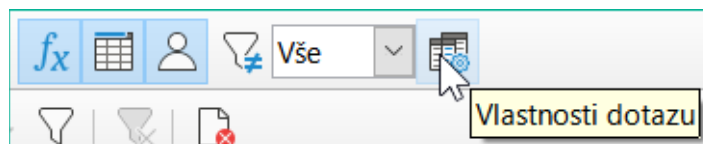
	Title	Subtitle
▶	Der kleine Hobbit	
	Das sogenannte Böse	
	Eine kurze Geschichte der Zeit	
	Traditionelle und kritische Theorie	
	Die neue deutsche Rechtschreibung	
	I hear you knocking	Youn can't catch me
	I hear you knocking	The stumble
	I hear you knocking	Sabre dance (Single version)
	Datenbanken mit OpenOffice.org 3	
	Das Postfix-Buch	
	Im Augenblick	Amsterdam
	Im Augenblick	Hier unten am Deich
	Im Augenblick	Köln-Ehrenfeld
	Im Augenblick	Bei Mir
	Im Augenblick	Gott sei Dank

Záznam 1 z 15

Obrázek 309: Výsledek dotazu zobrazující všechny názvy a titulky (pokud existují)

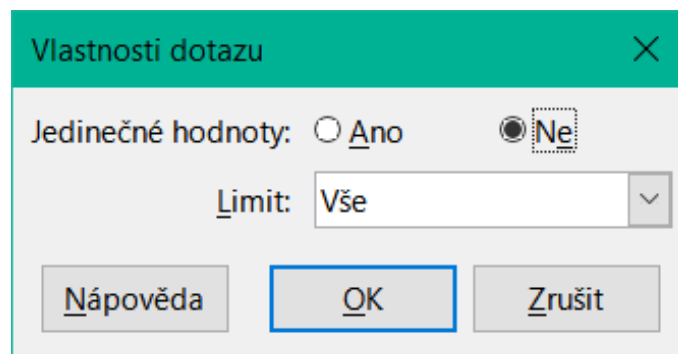
Definování vlastností dotazu

Od LibreOffice verze 4.1 je možné v editoru dotazů definovat další vlastnosti.



Obrázek 310: Ikona Vlastnosti dotazu na nástrojové liště Návrh

Vedle tlačítka pro otevření okna Vlastnosti dotazu se nachází pole se seznamem pro regulaci počtu zobrazených záznamů a tlačítko Rozlišit hodnoty (viz obrázek 310). Tyto funkce se opakují v dialogovém okně na obrázku 311:



Obrázek 311: Dialogové okno Vlastnosti dotazu

Nastavení Jedinečné hodnoty určuje, zda má dotaz potlačit duplicitní záznamy.

	FirstName	LastName	Return_Date
▶	Greta	Garbo	
	Greta	Garbo	
	Lisa	Gerd	
	Lisa	Gerd	
	Lisa	Gerd	
	Lisa	Gerd	
	Heinrich	Müller	
	Heinrich	Müller	
	Heinrich	Müller	
	Terence	Nobody	

Obrázek 312: Výstup dotazu s hodnotou Rozlišovat hodnoty nastavenou na Ne

Předpokládejme, že se provede dotaz na čtenáře, kteří mají stále vypůjčené knihy. Jejich názvy se zobrazí, pokud je pole pro datum vrácení prázdné. Pro čtenáře, kteří mají vypůjčeno více položek, se názvy zobrazují vícekrát, jak je znázorněno na obrázku 312.

	FirstName	LastName	Return_Date
▶	Greta	Garbo	
	Lisa	Gerd	
	Heinrich	Müller	
	Terence	Nobody	

Obrázek 313: Výstup dotazu s hodnotou Rozlišovat hodnoty nastavenou na Ano

Pokud zvolíme možnost Odlišné hodnoty, zmizí záznamy se stejným obsahem, jak je znázorněno na obrázku 313.

Dotaz pak vypadá takto:

```
SELECT DISTINCT
  "Reader"."FirstName", "Reader"."LastName", "Loan"."Return_Date"
FROM "Loan", "Reader"
WHERE "Loan"."Reader_ID" = "Reader"."ID" AND "Loan"."Return_Date" IS NULL
ORDER BY "Reader"."LastName" ASC
```

Původní dotaz:

```
SELECT "Reader"."FirstName", "Reader"."LastName" ...
```

Přidáním DISTINCT do dotazu se duplicitní záznamy potlačí.

```
SELECT DISTINCT "Reader"."FirstName", "Reader"."LastName" ...
```

V dřívějších verzích bylo pro zadání jedinečných záznamů nutné přepnout z režimu návrhu do režimu SQL a vložit kvalifikátor DISTINCT. Tato vlastnost je zpětně kompatibilní s předchozími verzemi LO a nezpůsobuje žádné problémy.

Nastavení Limit určuje, kolik záznamů se má v dotazu zobrazit. Zobrazen je pouze omezený počet záznamů.

ID	Title	Pub_Year
0	Der kleine Hobbit	1972
1	Das sogenannte Böse	1972
2	Eine kurze Geschichte der Zeit	1983
3	Traditionelle und kritische Theorie	1970
4	Die neue deutsche Rechtschreibung	1996
5	I hear you knocking	1972
6	Datenbanken mit OpenOffice.org 3	2009
7	Das Postfix-Buch	2008
8	Im Augenblick	2009
+ <Auto		

Obrázek 314: Výstup dotazu zobrazující všechny záznamy v tabulce Media

Všechny záznamy v tabulce Media jsou zobrazeny na obrázku 314. Dotaz je upravitelný, protože obsahuje primární klíč.

ID	Title	Pub_Year
0	Der kleine Hobbit	1972
1	Das sogenannte Böse	1972
2	Eine kurze Geschichte der Zeit	1983
3	Traditionelle und kritische Theorie	1970
4	Die neue deutsche Rechtschreibung	1996
+ <Auto		

Obrázek 315: Výstup dotazu omezený na zobrazení prvních pěti záznamů v tabulce Media

Nastavením limitu uvedeného na obrázku 315 se zobrazí pouze prvních pět záznamů (ID 0-4). Nebylo požadováno řazení, proto je použito výchozí řazení podle primárního klíče. I přes omezení výstupu lze dotaz dále upravovat. Tím se vstup v grafickém rozhraní liší od toho, co bylo v předchozích verzích dostupné pouze pomocí jazyka SQL.

```
SELECT "ID", "Title", "Pub_Year" FROM "Media" LIMIT 5
```

Do původního dotazu bylo jednoduše přidáno „LIMIT 5“. Velikost limitu může být libovolná.

! Upozornění

Nastavení limitů v grafickém rozhraní není zpětně kompatibilní. Ve všech verzích LO před verzí 4.1 bylo možné limit nastavit pouze v režimu přímého SQL. Omezení pak vyžadovalo třídění (ORDER BY ...) nebo podmínku (WHERE ...).

Vylepšení dotazu pomocí režimu SQL

Pokud při grafickém zadávání vypneme zobrazení návrhu pomocí **Zobrazit > Režim návrhu zap/vyp**, zobrazí se příkaz SQL pro to, co se dříve zobrazovalo v zobrazení návrhu. To může začátečníkům pomoci naučit se standardní dotazovací jazyk (Standard Query Language) pro databáze. Někdy je to také jediný způsob, jak zadat dotaz do databáze, pokud grafické uživatelské rozhraní nedokáže převést vaše požadavky na potřebné příkazy SQL.

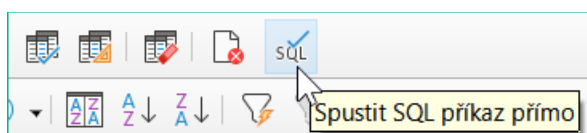
```
SELECT * FROM "Table_name"
```

Zobrazí se vše, co je v tabulce Table_name. Znak "*" představuje všechna pole tabulky.

```
SELECT * FROM "Table_name" WHERE "Field_name" = 'Karl'
```

Zde existuje významné omezení. Zobrazí se pouze ty záznamy, u nichž pole Field_name obsahuje výraz „Karl“ – přesný výraz, nikoli například „Karl Egon“.

Někdy nelze dotazy v aplikaci Base provádět pomocí grafického uživatelského rozhraní, protože určité příkazy nemusí být rozpoznány. V takových případech je nutné vypnout zobrazení návrhu a pro přímý přístup k databázi použít příkaz **Úpravy > Spustit SQL příkaz přímo**. Tato metoda má tu nevýhodu, že s dotazem lze pracovat pouze v režimu SQL.



Obrázek 316: Ikona přímého spuštění příkazu SQL na panelu nástrojů SQL

Přímé použití příkazů SQL je dostupné také pomocí grafického uživatelského rozhraní, jak ukazuje obrázek 316. Klepnutím na zvýrazněnou ikonu (**Spustit SQL příkaz přímo**) vypneme ikonu Režim návrhu zap/vyp.. Nyní se po klepnutí na ikonu Spustit, spustí přímo příkazy SQL.

Tvorba dotazů v režimu SQL

Zde je příklad rozsáhlých možností, které jsou k dispozici pro zadávání dotazů do databáze a určování typu požadovaného výsledku:

```
SELECT [{LIMIT <offset> <limit> | TOP <limit>}][ALL | DISTINCT]
{ <Select-Formulation> | "Table_name".* | * } [, ...]
[INTO [CACHE | TEMP | TEXT] "new_Table"]
FROM "Table_list"
[WHERE SQL-Expression]
[GROUP BY SQL-Expression [, ...]]
[HAVING SQL-Expression]
[ { UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT] }
|
INTERSECT [DISTINCT] } Query statement]
[ORDER BY Order-Expression [, ...]]
[LIMIT <limit> [OFFSET <offset>]];
```

[{LIMIT <offset> <limit> | TOP <limit>}]:

Tím se omezí počet zobrazovaných záznamů. LIMIT 10 20 začíná od 11. záznamu a zobrazí následujících 20 záznamů. TOP 10 zobrazí vždy prvních 10 záznamů. Všechny zadané hodnoty musí být celá čísla. To je stejné jako LIMIT 0 10. LIMIT 10 0 vynechá prvních 10 záznamů a zobrazí všechny záznamy počínaje 11. záznamem.

[ALL | DISTINCT]

SELECT ALL je výchozí nastavení. Zobrazí se všechny záznamy, které splňují podmínky vyhledávání. Příklad: SELECT ALL "Name" FROM "Table_name" zobrazí všechna jména; pokud se v tabulce vyskytuje třikrát "Peter" a čtyřikrát "Egon", zobrazí se tato jména třikrát, resp. čtyřikrát. SELECT DISTINCT "Name" FROM "Table_name" potlačí výsledky dotazu, které mají stejný obsah. V tomto případě se slova „Peter“ a „Egon“ vyskytují pouze jednou. DISTINCT se vztahuje k celému záznamu, ke kterému dotaz přistupuje. Pokud je tedy například požadováno i příjmení, budou záznamy pro „Peter Müller“ a „Peter Maier“ považovány za odlišné. I když zadáme podmínku DISTINCT, zobrazí se obě.

<Select-Formulation>

```
{ Expression | COUNT(*) |  
{ COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP  
| STDDEV_POP | STDDEV_SAMP }  
([ALL | DISTINCT]] Expression) } [[AS] "display_name"]
```

Názvy polí, výpočty, součty záznamů jsou všechny možné položky.

✓ Poznámka

Výpočty v databázi někdy vedou k neočekávaným výsledkům. Předpokládejme, že databáze obsahuje známky udělené za nějakou práci ve třídě a že chceme vypočítat průměrnou známku.

Nejprve musíme sečíst všechny známky. Předpokládejme, že součet je 80. Nyní je třeba tuto hodnotu vydělit počtem žáků (řekněme 30). Výsledkem dotazu je 2.

K tomu dochází, protože pracujeme s poli typu INTEGER. Výsledkem výpočtu proto musí být celočíselná hodnota. Ve výpočtu musíme mít alespoň jednu hodnotu typu DECIMAL. Toho můžeme dosáhnout buď pomocí převodní funkce HSQLDB nebo (jednodušeji) můžeme dělit číslem 30,0 namísto 30. Tím získáme výsledek s jedním desetinným místem. Dělením 30.00 získáme dvě desetinná místa. Dbáme na používání desetinných míst v anglickém stylu. Použití čárek v dotazech je vyhrazeno pro oddělovače polí.

Kromě toho jsou pro zobrazené pole k dispozici různé funkce. Kromě funkce COUNT(*) (která spočítá všechny záznamy) žádná z těchto funkcí nepřistupuje k polím NULL.

```
COUNT | MIN | MAX | SUM | AVG | SOME | EVERY | VAR_POP | VAR_SAMP |  
STDDEV_POP | STDDEV_SAMP
```

COUNT("Name") spočítá všechny záznamy pro pole Name.

MIN("Name") zobrazí první jméno v abecedním pořadí. Výsledek této funkce je vždy formátován tak, jak se vyskytuje v poli. Text se zobrazuje jako text, celá čísla jako celá čísla, desetinná čísla jako desetinná čísla atd.

MAX("Name") zobrazí příjmení v abecedním pořadí.

SUMA("Number") může sčítat pouze hodnoty v číselných polích. Funkce selhává v případech polí s datem.

i Tip

Tato funkce selhává také v polích pro čas. Zde mohou být užitečné následující informace:

```
SELECT (SUM( HOUR("Time") ) * 3600 + SUM( MINUTE("Time") )) * 60 + SUM(  
SECOND("Time") ) AS "seconds" FROM "Table"
```

Zobrazený součet se skládá z jednotlivých hodin, minut a sekund. Poté se upraví tak, aby se získal celkový součet v jedné jednotce, v tomto případě v sekundách. Zde se pak funkcí sum sčítají pouze celá čísla. Pokud nyní použijeme:

```
SELECT ((SUM( HOUR("Time") ) * 3600 + SUM( MINUTE("Time") )) * 60 +  
SUM( SECOND("Time") )) / 3600.0000 AS "hours" FROM "Table"
```

získáme čas v hodinách s minutami a sekundami jako desetinnými místy. Při vhodném **formátování** můžeme tuto hodnotu v dotazu nebo formuláři změnit zpět na normální časovou hodnotu.

AVG("Number") zobrazí průměr obsahu sloupce. I tato funkce je omezena na číselná pole.

SOME("Field_Name"), EVERY("Field_Name"): Pole používaná těmito funkcemi musí mít typ pole Yes/No[BOOLEAN] (obsahuje pouze 0 nebo 1). Kromě toho vytvářejí přehled obsahu pole, na které jsou aplikovány.

SOME vrací *TRUE* (nebo 1), pokud je alespoň jedna položka pole 1, a vrací *FALSE* (nebo 0) pouze tehdy, pokud jsou všechny položky 0. EVERY vrací 1 pouze v případě, že každá položka pole je 1, a vrací *FALSE*, pokud je alespoň jedna položka 0.

Tip

Typ pole logického typu je Yes/No[BOOLEAN]. Toto pole však obsahuje pouze 0 nebo 1. V podmínkách vyhledávání dotazu použijeme buď *TRUE*, 1, *FALSE* nebo 0. Pro podmínku Yes můžeme použít hodnotu *TRUE* nebo 1. Pro podmínku No použijeme buď *FALSE*, nebo 0. Pokud se místo toho pokusíme použít „Yes“ nebo „No“, zobrazí se chybová zpráva. Pak budeme muset svou chybu opravit.

Příklad:

```
SELECT "Class", EVERY("Swimmer")
FROM "Table1"
GROUP BY "Class";
```

Třída obsahuje názvy plavecké třídy. Swimmer je logické pole popisující, zda student umí plavat (1 nebo 0). Students obsahuje jména studentů. Table1 obsahuje tato pole: primární klíč, Class, Swimmer a Students. Pro tento dotaz jsou potřeba pouze údaje Class a Swimmer.

Protože je dotaz seskupen podle záznamů pole Class, vrátí EVERY pro každou třídu hodnotu pole Swimmer. Když každý účastník plavecké třídy umí plavat, EVERY vrací *TRUE*. V opačném případě EVERY vrátí *FALSE*, protože alespoň jeden žák třídy neumí plavat. Vzhledem k tomu, že výstupem pro pole Swimmer je zaškrťovací políčko, značka zaškrtnutí znamená *TRUE* a žádná značka znamená *FALSE*.

VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP jsou statistické funkce a ovlivňují pouze celočíselná a desetinná pole.

Všechny tyto funkce vracejí hodnotu 0, pokud jsou všechny hodnoty ve skupině stejné.

Statistické funkce neumožňují použít omezení DISTINCT. V podstatě počítají se všemi hodnotami, na které se dotaz vztahuje, zatímco DISTINCT vylučuje ze zobrazení záznamy se stejnými hodnotami.

[AS] "display_name": Pole mohou mít v rámci dotazu jiné označení (alias).

"Table_name".* | * [, ...]

Symbol „*“ použijeme pro zobrazení celkového obsahu tabulky, místo abychom zadávali všechna pole jednotlivě. Pokud se výsledky vztahují pouze na jednu tabulku, je zbytečné používat název tabulky. Pokud však dotaz obsahuje všechna pole jedné tabulky a alespoň jedno pole z druhé tabulky, použijeme:

```
"Název_tabulky 1".*, "Název_tabulky 2". "Název_pole".
```

[INTO [CACHED | TEMP | TEXT] "new_table"]

Výsledek tohoto dotazu se zapíše přímo do nové tabulky, která je zde pojmenována. Vlastnosti polí pro novou tabulku jsou definovány z definic polí obsažených v dotazu. Zápis do nové tabulky nefunguje v režimu SQL, protože ten zpracovává pouze zobrazitelné výsledky. Místo toho musíme použít **Nástroje > SQL**. Výsledná tabulka není zpočátku editovatelná, protože v ní chybí primární klíč.

FROM <Table_list>

```
"Table_name 1" [{CROSS | INNER | LEFT OUTER | RIGHT OUTER} JOIN  
"Table_name 2" ON Expression] [, ...]
```

Tabulky, které mají být společně prohledávány, jsou obvykle v seznamu odděleném čárkami. Vzájemný vztah tabulek je pak dodatečně definován klíčovým slovem `WHERE`.

Pokud jsou tabulky svázány pomocí `JOIN`, nikoli čárkou, je jejich vztah definován výrazem začínajícím `ON`, který se vyskytuje přímo za druhou tabulkou.

Jednoduché `JOIN` má za následek, že se zobrazí pouze ty záznamy, pro které platí podmínky v obou tabulkách.

Příklad:

```
SELECT "Table1"."Name", "Table2"."Class"  
FROM "Table1", "Table2"  
WHERE "Table1"."ClassID" = "Table2"."ID"
```

je shodné s:

```
SELECT "Table1"."Name", "Table2"."Class"  
FROM "Table1"  
    JOIN "Table2"  
ON "Table1"."ClassID" = "Table2"."ID"
```

Zde jsou zobrazeny jména a odpovídající třídy. Pokud není u jména uvedena žádná třída, není toto jméno do zobrazení zahrnuto. Pokud třída nemá žádná jména, také se nezobrazí. Přidání `INNER` na tom nic nemění.

```
SELECT "Table1"."Name", "Table2"."Class"  
FROM "Table1"  
    LEFT JOIN "Table2"  
ON "Table1"."ClassID" = "Table2"."ID"
```

Pokud je přidáno klíčové slovo `LEFT`, zobrazí se všechna Jména z `Table1`, i když nemají žádnou hodnotu v poli `Class`. Pokud je naopak přidáno klíčové slovo `RIGHT`, zobrazí se všechny Třídy, i když v nich nejsou žádná jména. Použití klíčového slova `OUTER` zde není třeba uvádět. (Pravý vnější spoj je totéž co Pravý spoj; Levý vnější spoj je totéž co Levý spoj.)

```
SELECT "Table1"."Player1", "Table2"."Player2"  
FROM "Table1" AS "Table1"  
    CROSS JOIN "Table2" AS "Table1"  
WHERE "Table1"."Player1" <> "Table2"."Player2"
```

Křížové spojení `CROSS JOIN` vyžaduje, aby byla tabulka opatřena aliasem, ale přidání výrazu `ON` není vždy nutné. Všechny záznamy z první tabulky jsou spárovány se všemi záznamy z druhé tabulky. Výše uvedený dotaz tedy poskytne všechny možné dvojice záznamů z první tabulky se záznamy z druhé tabulky s výjimkou dvojic mezi záznamy pro stejného hráče. V případě `CROSS JOIN` nesmí podmínka obsahovat vazbu mezi tabulkami uvedenými v termínu `ON`. Místo toho lze zadat podmínky `WHERE`. Pokud jsou podmínky formulovány přesně jako v případě jednoduchého spojení `JOIN`, získáme stejný výsledek:

```
SELECT "Table1"."Name", "Table2"."Class"  
FROM "Table1"  
    JOIN "Table2"  
ON "Table1"."ClassID" = "Table2"."ID"
```

dává stejný výsledek jako

```
SELECT "Table1"."Name", "Table2"."Class"
FROM "Table1" AS "Table1"
CROSS JOIN "Table2" AS "Table2"
WHERE "Table1"."ClassID" = "Table2"."ID"
```

[WHERE SQL-Expression]

Standardní úvod pro podmínky, které vyžadují přesnější filtrování dat. I zde se obvykle definují relace mezi tabulkami, pokud nejsou propojeny pomocí JOIN.

[GROUP BY SQL-Expression [, ...]]

Tuto funkci použijeme, pokud chceme data dotazu rozdělit do skupin a teprve poté použít funkce na každou skupinu zvlášť. Rozdělení je založeno na hodnotách pole nebo polí obsažených ve výrazu GROUP BY.

Příklad:

```
SELECT "Name", SUM("Input"-"Output") AS "Balance"
FROM "Table1"
GROUP BY "Name";
```

Záznamy se stejným názvem jsou sečteny. Ve výsledku dotazu je pro každou osobu uveden součet Vstup – Výstup (Input - Output). Toto pole se nazývá Balance. Každý řádek výsledku dotazu obsahuje hodnotu z tabulky Name a vypočtený zůstatek pro tuto konkrétní hodnotu.



Tip

Pokud jsou pole zpracována pomocí určité funkce (například COUNT, SUM ...), všechna pole, která nejsou zpracována pomocí funkce, ale mají být zobrazena, jsou seskupena pomocí GROUP BY.

[HAVING SQL-Expression]

Vzorec HAVING se velmi podobá vzorci WHERE. Rozdíl je v tom, že vzorec WHERE se vztahuje na hodnoty vybraných polí v dotazu. Vzorec HAVING se vztahuje na vybrané vypočtené hodnoty. Konkrétně vzorec WHERE nemůže jako součást vyhledávací podmínky použít agregační funkci, vzorec HAVING ano.

Vzorec HAVING slouží ke dvěma účelům, jak je uvedeno ve dvou příkladech níže. V prvním případě podmínka vyhledávání vyžaduje, aby minimální doba běhu byla kratší než 40 minut. V druhém příkladu podmínka hledání vyžaduje, aby zůstatek jednotlivce byl kladný.

Ve výsledcích prvního dotazu jsou uvedena jména lidí, jejichž doba běhu byla alespoň jednou kratší než 40 minut, a minimální doba běhu. Lidé, jejichž doba běhu byla delší než 40 minut, uvedeni nejsou.

Ve výsledcích druhého dotazu jsou uvedena jména osob, jejichž celkový výstup je vyšší než vstup, a jejich zůstatek. Lidé, jejichž zůstatek je 0 nebo méně, nejsou uvedeni.

Příklady:

```
SELECT "Name", "Runtime"
FROM "Table1"
GROUP BY "Name", "Runtime"

HAVING MIN("Runtime") < '00:40:00';
SELECT "Name", SUM("Input"-"Output") AS "Balance"
FROM "Table1"
GROUP BY "Name"
HAVING SUM("Input"-"Output") > 0;
```

[SQL Expression]

Výrazy SQL se kombinují podle následujícího schématu:

```
[NOT] podmínka [{ OR | AND } podmínka]
```

Příklad:

```
SELECT *  
FROM "Table_name"  
WHERE NOT "Return_date" IS NULL AND "ReaderID" = 2;
```

Z tabulky se načtou záznamy, u nichž bylo zadáno datum vrácení (Return_date) a hodnota RecordID je 2. V praxi to znamená, že získáme všechna média zapůjčená konkrétní osobě a vrácená zpět. Podmínky jsou spojeny pouze pomocí AND. NOT se vztahuje pouze na první podmínku.

```
SELECT *  
FROM "Table_name"  
WHERE NOT ("Return_date" IS NULL AND "ReaderID" = 2);
```

Závorky kolem podmínky, mimo které je uvedeno NOT, zobrazí pouze ty záznamy, které podmínku v závorce zcela nesplňují. To by se týkalo všech záznamů kromě záznamů pro ReaderID číslo 2, které dosud nebyly vráceny.

[SQL Expression]: podmínky

```
{ hodnota [| | hodnota]
```

Hodnota může být buď jedna nebo několik hodnot spojených dvěma svislými čarami ||. To samozřejmě platí i pro obsah polí.

```
SELECT "Surname" || ', ' || "First_name" AS "Name"  
FROM "Table_name"
```

Obsah polí Surname a First_name se zobrazí společně v poli s názvem Name. Všimneme si, že mezi hodnoty Surname a First_name se vkládá čárka a mezera.

```
| hodnota { = | < | <= | > | >= | <> | != } hodnota
```

Tyto znaky odpovídají známým matematickým operátorům:

```
{ Rovná se | Menší než | Menší nebo rovno | Větší než | Větší nebo  
rovno | Není rovno | Není rovno }
```

```
| hodnota IS [NOT] NULL
```

Odpovídající pole nemá žádný obsah, protože do něj nebylo nic zapsáno. To nelze v grafickém uživatelském rozhraní jednoznačně určit, protože vizuálně prázdné textové pole neznamená, že je zcela bez obsahu. Ve výchozím nastavení databáze Base jsou však prázdná pole v databázi nastavena na hodnotu NULL.

```
| EXISTS(Query_result)
```

Příklad:

```
SELECT "Name"  
FROM "Table1"  
WHERE EXISTS  
  (SELECT "First_name"  
   FROM "Table2"  
   WHERE "Table2"."First_name" = "Table1"."Name")
```

Zobrazí se jména z Table1, jejichž křestní jména jsou uvedena v Table2.

```
| Hodnota BETWEEN Hodnota AND Hodnota
```

BETWEEN hodnota1 AND hodnota2 zobrazí všechny hodnoty od hodnota1 až po hodnota2 včetně. Pokud jsou hodnoty tvořeny písmeny, použije se abecední řazení, při kterém mají malá písmena stejnou hodnotu jako odpovídající velká písmena.

```
SELECT "Name"  
FROM "Table_name"  
WHERE "Name" BETWEEN 'A' AND 'E';
```

Výsledkem tohoto dotazu budou všechna jména začínající na A, B, C nebo D (a také na odpovídající malá písmena). Jelikož je E nastaveno jako horní hranice, nejsou zahrnuta jména začínající na E. Samotné písmeno E se vyskytuje těsně před jmény začínajícími na E.

| hodnota [NOT] IN ({hodnota [, ...] | výsledek dotazu })

To vyžaduje buď seznam hodnot, nebo dotaz. Podmínka je splněna, pokud je hodnota obsažena v seznamu hodnot nebo ve výsledku dotazu.

| hodnota [NOT] LIKE hodnota [ESCAPE] hodnota }

Operátor LIKE je operátor, který je potřebný v mnoha jednoduchých vyhledávacích funkcích. Hodnota se zadává podle následujícího vzoru:

'%' znamená libovolný počet znaků (včetně 0),
'_' nahrazuje přesně jeden znak.

Chceme-li vyhledat samotný znak '%' nebo '_', musí tyto znaky následovat bezprostředně za jiným znakem definovaným jako ESCAPE.

```
SELECT "Name"  
FROM "Table_name"  
WHERE "Name" LIKE '\_%' ESCAPE '\'
```

Tento dotaz zobrazí všechna jména začínající podtržítkem. '\' je zde definováno jako znak ESCAPE.

[SQL Expression]: hodnoty

[+ | -] { Výraz [{ + | - | * | / | | } Výraz]

Hodnoty mohou mít předchozí znaménko. Je povoleno sčítání, odčítání, násobení, dělení a spojování výrazů. Příklad spojování:

```
SELECT "Surname" || ', ' || "First_name"  
FROM "Table"
```

Tímto způsobem se v dotazu zobrazí záznamy s polem obsahujícím "Surname, First_name". Operátor spojování lze kvalifikovat pomocí následujících výrazů.

| (Podmínka)

Viz předchozí oddíl.

| Funkce ([Parametr] [, ...])

Viz oddíl Funkce v příloze.

Následující dotazy se také označují jako poddotazy (podvýběry).

| Výsledek dotazu, který dává přesně jednu odpověď

Protože záznam může mít v každém poli pouze jednu hodnotu, pouze dotaz, který dává právě jednu hodnotu, může být zobrazen celý.

| {ANY|ALL} (Výsledek dotazu, který dává přesně jednu odpověď z celého sloupce)

Často se vyskytuje podmínka, která porovnává výraz s celou skupinou hodnot.

V kombinaci s **ANY** to znamená, že se výraz musí ve skupině vyskytovat alespoň jednou. To lze zadat také pomocí podmínky **IN**. **= ANY** vede ke stejnému výsledku jako **IN**.

V kombinaci s **ALL** znamená, že všechny hodnoty skupiny musí odpovídat jednomu výrazu.

[SQL Expression]: Výraz

```
{ 'Text' | Celé číslo | Číslo s plovoucí desetinnou čárkou  
| ["Table"."]Field" | TRUE | FALSE | NULL }
```

Hodnoty v zásadě slouží jako argumenty pro různé výrazy v závislosti na zdrojovém formátu. Chceme-li vyhledat obsah textových polí, umístíme obsah do uvozovek. Celá čísla se zapisují bez uvozovek, stejně jako čísla s pohyblivou čárkou.

Pole znamenají hodnoty, které se v tabulce vyskytují v daných polích. Obvykle se pole porovnávají buď mezi sebou, nebo s určitými hodnotami. V jazyce SQL by měly být názvy polí umístěny do dvojitých uvozovek, protože jinak nemusí být správně rozpoznány. SQL obvykle předpokládá, že text bez dvojitých uvozovek je bez speciálních znaků, tj. jedno slovo bez mezer a s velkými písmeny. Pokud je v dotazu obsaženo více tabulek, musí být kromě názvu pole uveden i název tabulky, oddělený od názvu pole tečkou.

Hodnoty **TRUE** a **FALSE** se obvykle odvozují od polí **Yes/No**.

Hodnota **NULL** znamená žádný obsah. Není to totéž co 0, ale spíše odpovídá „prázdnému“.

UNION [ALL | DISTINCT] Query_result

Tím se dotazy propojí tak, že obsah druhého dotazu je zapsán pod prvním dotazem. Aby to fungovalo, musí se všechna pole v obou dotazech shodovat v typu. Toto propojení několika dotazů funguje pouze v režimu přímého příkazu SQL.

```
SELECT "First_name"  
FROM "Table1"  
UNION DISTINCT  
  SELECT "First_name"  
  FROM "Table2";
```

Tento dotaz zobrazí všechna křestní jména z **Table1** a **Table2**; doplňkový výraz **DISTINCT** znamená, že se nezobrazí žádná duplicitní křestní jména. **DISTINCT** je v tomto kontextu výchozí hodnota. Ve výchozím nastavení jsou křestní jména řazena vzestupně podle abecedy. **ALL** způsobí, že se zobrazí všechna křestní jména v **Table1** a za nimi křestní jméno v **Table2**. V tomto případě je výchozí třídění podle primárního klíče.

Použití této techniky dotazování umožňuje vypsát hodnoty ze záznamu přímo ve sloupci pod sebe. Předpokládejme, že máme tabulku s názvem **Stock**, ve které jsou pole **Sales_price**, **Rebate_price_1** a **Rebate_price_2**. Z těchto údajů chceme vypočítat kombinační pole, ve kterém budou tyto ceny uvedeny přímo pod sebou.

```
SELECT  
  "Sales_price"  
FROM "Stock" WHERE "Stock_ID" = 1  
UNION  
SELECT  
  "Rebate_price_1"  
FROM "Stock" WHERE "Stock_ID" = 1  
UNION  
SELECT  
  "Rebate_price_2"  
FROM "Stock" WHERE "Stock_ID" = 1;
```

Primární klíč tabulky **Stock** musí být samozřejmě nastaven pomocí makra, protože pole kombinace bude mít odpovídající položku.

MINUS [DISTINCT] | EXCEPT [DISTINCT] Query_result

```
SELECT "First_name"  
FROM "Table1"  
EXCEPT  
    SELECT "First_name"  
    FROM "Table2";
```

Zobrazí všechna křestní jména z tabulky Table1 s výjimkou křestních jmen obsažených v tabulce Table2. MINUS a EXCEPT vedou ke stejnému výsledku. Řazení je abecední.

INTERSECT [DISTINCT] Query_result

```
SELECT "First_name"  
FROM "Table1"  
INTERSECT  
    SELECT "First_name"  
    FROM "Table2";
```

Zobrazí se křestní jména, která se vyskytují v obou tabulkách. Řazení je opět abecední. V současné době to funguje pouze v režimu přímého příkazu SQL.

[ORDER BY Ordering-Expression [, ...]]

Výrazem může být název pole, číslo sloupce (počínaje 1 zleva), alias (formulovaný například pomocí AS) nebo složený výraz hodnoty (viz [SQL Expression]: hodnoty). Řazení je obvykle vzestupné (ASC). Pokud chceme sestupné třídění, musíme výslovně zadat DESC.

```
SELECT "First_name", "Surname" AS "Name"  
FROM "Table1"  
ORDER BY "Surname";
```

je totožný s

```
SELECT "First_name", "Surname" AS "Name"  
FROM "Table1"  
ORDER BY 2;
```

je totožný s

```
SELECT "First_name", "Surname" AS "Name"  
FROM "Table1"  
ORDER BY "Name";
```

[LIMIT <limit> [OFFSET <offset>]]:

LIMIT 10 zobrazí pouze 10 záznamů. Přidání OFFSET 20 způsobí, že zobrazení začne od 21. záznamu. Tato poslední forma omezení zobrazení vyžaduje buď instrukci pro řazení (ORDER BY...), nebo podmínku (WHERE...).

Všechny zadané hodnoty limitu musí být celá čísla. Není možné nahradit záznam poddotazem tak, aby se například pokaždé zobrazilo pět posledních záznamů řady.

Použití aliasu v dotazu

Dotazy mohou reprodukovat pole se změněnými názvy.

```
SELECT "First_name", "Surname" AS "Name"  
FROM "Table1"
```

Pole Surname se v zobrazení nazývá Name.

Pokud dotaz zahrnuje dvě tabulky, musí být před názvem každého pole uveden název tabulky:

```
SELECT "Table1"."First_name", "Table1"."Surname" AS "Name",  
"Table2"."Class"
```

```
FROM "Table1", "Table2"  
WHERE "Table1"."Class_ID" = "Table2"."ID"
```

Názvu tabulky lze také přidělit alias, ale tato změna se projeví pouze v dotazu a neprojeví se v zobrazení tabulky. Pokud je takový alias nastaven, je třeba odpovídajícím způsobem změnit všechny názvy tabulek v dotazu:

```
SELECT "a"."First_name", "a"."Surname" AS "Name", "b"."Class"  
FROM "Table1" AS "a", "Table2" AS "b"  
WHERE "a"."Class_ID" = "b"."ID"
```

Přirazení aliasu tabulce lze provést stručněji bez použití termínu AS:

```
SELECT "a"."First_name", "a"."Surname" "Name", "b"."Class"  
FROM "Table1" "a", "Table2" "b"  
WHERE "a"."Class_ID" = "b"."ID"
```

Tím se však kód stává méně čitelným. Z tohoto důvodu by se zkrácený formulář měl používat pouze ve výjimečných případech.



Poznámka

Kód editoru dotazů v grafickém uživatelském rozhraní byl nedávno změněn tak, aby se označení aliasů vytvářela bez použití předpony AS. Je to proto, že při použití externích databází, jejichž způsob zadávání aliasů nelze předvídat, vedlo zahrnutí AS k chybovým hlášením.

Pokud je kód dotazu otevřen k úpravám nikoli přímo v SQL, ale pomocí grafického uživatelského rozhraní, stávající aliasy ztratí předponu AS. Pokud je to pro nás důležité, musíme dotazy vždy upravovat v režimu SQL.

Aliasový název také umožňuje použít tabulku s odpovídajícím filtrováním více než jednou v rámci dotazu:

```
SELECT "KasseAccount"."Balance", "Account"."Date",  
       "a"."Balance" AS "Actual",  
       "b"."Balance" AS "Desired"  
FROM "Account"  
LEFT JOIN "Account" AS "a"  
ON "Account"."ID" = "a"."ID" AND "a"."Balance" >= 0  
LEFT JOIN "Account" AS "b"  
ON "Account"."ID" = "b"."ID" AND "b"."Balance" < 0
```

Dotazy na vytvoření polí seznamu

V polích se seznamem se zobrazuje hodnota, která neodpovídá obsahu podkladové tabulky. Používají se k zobrazení hodnoty, kterou uživatel přiřadil cizímu klíči, a nikoli samotného klíče. Hodnota, která je nakonec uložena ve formuláři, se nesmí vyskytovat na první pozici pole seznamu.

```
SELECT "FirstName", "ID"  
FROM "Table1";
```

Tento dotaz by zobrazil všechna křestní jména a hodnoty primárního klíče "ID", které poskytuje základní tabulka formuláře. Samozřejmě to ještě není optimální. Křestní jména nejsou seřazena a v případě stejných křestních jmen nelze určit, o kterou osobu se jedná.

```
SELECT "FirstName" || ' ' || "LastName", "ID"  
FROM "Table1"  
ORDER BY "FirstName" || ' ' || "LastName";
```

Nyní se zobrazí jméno i příjmení oddělené mezerou. Jména se stanou rozlišitelnými a jsou také seřazena. Řazení se však řídí obvyklou logikou, kdy se začíná prvním písmenem řetězce, takže se řadí podle jména a teprve potom podle příjmení. Jiné pořadí řazení, než v jakém jsou pole zobrazena, by bylo pouze matoucí.

```
SELECT "LastName" || ', ' || "FirstName", "ID"
FROM "Table1"
ORDER BY "LastName" || ', ' || "FirstName";
```

To nyní vede k řazení, které lépe odpovídá běžným zvyklostem. Členové rodiny se objevují společně, jeden pod druhým; různé rodiny se stejným příjmením se však prolínají. Abychom je mohli rozlišit, museli bychom je v tabulce seskupit jinak.

Je tu ještě jeden problém: pokud mají dva lidé stejné příjmení a jméno, stejně je nelze rozlišit. Jedním z řešení by mohlo být použití přípony jména. Ale představme si, jak by to vypadalo, kdyby na pozdravu stálo Pan "Müller II"!

```
SELECT "LastName" || ', ' || "FirstName" || ' - ID:' || "ID", "ID"
FROM "Table1"
ORDER BY "LastName" || ', ' || "FirstName" || "ID";
```

Zde jsou všechny záznamy rozlišitelné. Ve skutečnosti se zobrazí "LastName, FirstName – ID:ID hodnota". Například osoba jménem Heinrich Müller s ID 1 se zobrazí jako „Müller, Heinrich – ID: 1“.

Ve formuláři pro výpůjčky se v seznamu zobrazí pouze média, která ještě nebyla vypůjčena. Vytváří se pomocí následujícího SQL příkazu:

```
SELECT "Title" || ' - Nr. ' || "ID", "ID"
FROM "Media"
WHERE "ID" NOT IN
  (SELECT "Media_ID"
   FROM "Loan"
   WHERE "Return_Date" IS NULL)
ORDER BY "Title" || ' - Nr. ' || "ID" ASC
```

Je důležité, aby toto pole se seznamem bylo vždy aktualizováno, pokud je médium v něm zapůjčeno.

Následující pole se seznamem na obrázku 317 zobrazuje obsah několika polí v tabulkové podobě tak, že prvky, které patří k sobě, jsou přímo pod sebou.

	Quantity	Goods	
	2	Schnellhefter, Pappe	- 0,46 €
	5	Papier, 500 blat	- 5,65 €
	10	Bleistift	- 0,25 €
	1	Hefter, Tischgerät	- 11,25 €
	▶ 1	Locher, Registratur	- 15,48 €
	+	Bleistift HB	- 0,25 €
		Desktop-PC Prozessor 17 A	- 599,00 €
		Hefter, Tischgerät	- 11,25 €
		Locher, Registratur	- 15,48 €
		MiniPC Nano Prozessor: 15	- 398,00 €
		Papier, 500 blat	- 5,65 €
Záznam	5	Schnellhefter, Pappe	- 0,46 €
		Ultrabook Bildschirm: 15"	- 899,00 €

Obrázek 317: Příklad pole se seznamem

Aby takové zobrazení fungovalo, musíme nejprve zvolit vhodné písmo s pevnou šířkou. Zde lze použít Courier nebo jiné bezpatkové písmo, například Liberation Mono. Tabulkový formulář vytvoříte pomocí kódu SQL:


```
SELECT
  LEFT("Stock" || SPACE(25), 25) || ' - ' ||
  RIGHT(SPACE(8) || "Price", 8) || ' €',
  "ID"
FROM "Stock"
ORDER BY ("Stock" || ' - ' || "Price" || ' $') ASC
```

Obsah pole Stock byl doplněn mezerami tak, aby celý řetězec měl minimální délku 25 znaků. Poté se tam umístí prvních 25 písmen a přebytek se odstříhne.

Situace se komplikuje, pokud obsah pole seznamu obsahuje netisknutelné znaky, jako jsou nové řádky. Pak je třeba kód upravit takto:

```
SELECT
  LEFT(REPLACE("Stock", CHAR(10), ' ') || SPACE(25), 25) || ' - ' || ...
```

Tím se v systému Linux nahradí nový řádek mezerou. V systému Windows musíme navíc odstranit návrat kurzoru na začátek řádku (CHAR(13)).

Počet potřebných mezer lze také určit na základě jednotlivých dotazů. Tím se zabrání náhodnému zkrácení hodnoty „Stock“.

```
SELECT
  LEFT("Stock" || SPACE((SELECT MAX(LENGTH("Stock")) FROM "Stock")),
  (SELECT MAX(LENGTH("Stock")) FROM "Stock")) || ' - ' ||
  RIGHT(' ' || "Price", 8) || ' €',
  "ID"
FROM "Stock"
ORDER BY ("Stock" || ' - ' || "Price" || ' $') ASC
```

Protože se cena má zobrazovat vpravo, je vlevo doplněna mezerami a umístěna maximálně osm znaků zprava. Vybrané zobrazení bude fungovat pro všechny ceny do \$ 99999,99.

Pokud chceme v jazyce SQL nahradit desetinnou tečku čárkou, budeme potřebovat další kód:

```
REPLACE(RIGHT(' ' || "Price", 8), '.', ',')
```

Dotazy jako základ pro další informace ve formulářích

Pokud chceme, aby formulář zobrazoval další informace, které by jinak nebyly viditelné, máme k dispozici různé možnosti dotazů. Nejjednodušší je získat tyto informace pomocí nezávislých dotazů a výsledky vložit do formuláře. Nevýhodou této metody je, že změny v záznamech mohou ovlivnit výsledek dotazu, ale tyto změny se automaticky nezobrazují.

Tabulka 9 je příkladem z oblasti skladového hospodářství pro jednoduchou pokladnu.

Tabulka pokladny obsahuje součty a cizí klíče pro skladové položky a číslo příjímky. Zákazník má jen velmi málo informací, pokud na účtence není vytištěn žádný další výsledek dotazu. Koneckonců, položky jsou identifikovány pouze načtením čárového kódu. Bez dotazu se zobrazí pouze formulář:

Tabulka 9: Počet položek pro každý čárový kód

Celkem	Čárový kód
3	17
2	24

To, co se skrývá za čísly, nelze zviditelnit pomocí pole se seznamem, protože cizí klíč se zadává přímo pomocí čárového kódu. Stejně tak není možné použít pole se seznamem vedle položky, aby se zobrazovala alespoň jednotková cena.

Zde může pomoci dotaz.

```
SELECT "Checkout"."Receipt_ID", "Checkout"."Total", "Stock"."Item",  
"Stock"."Unit_Price", "Checkout"."Total"*"Stock"."Unit_price" AS  
"Total_Price"  
FROM "Checkout", "Stock"  
WHERE "Stock"."ID" = "Checkout"."Item_ID";
```

Nyní po zadání informací alespoň víme, kolik je třeba zaplatit za 3 * Item'17'. Kromě toho je třeba ve formuláři filtrovat pouze informace relevantní pro příslušné Receipt_ID. Stále chybí to, co musí zákazník celkově zaplatit.

```
SELECT "Checkout"."Receipt_ID",  
SUM("Checkout"."Total"*"Stock"."Unit_price") AS "Sum"  
FROM "Checkout", "Stock"  
WHERE "Stock"."ID" = "Checkout"."Item_ID"  
GROUP BY "Checkout"."Receipt_ID";
```

Formulář navrhne tak, aby zobrazoval vždy jeden záznam dotazu. Protože je dotaz seskupen podle Receipt_ID, formulář zobrazuje informace vždy o jednom zákazníkovi.

Tip

Pokud formulář potřebuje zobrazit hodnoty data, které závisí na jiném datu (například výpůjční doba média může být 21 dní, takže jaké je datum vrácení?), nelze použít vestavěné funkce HSQLDB. Funkce „DATEADD“ neexistuje.

Dotaz

```
SELECT "Date", DATEDIFF('dd', '1899-12-30', "Date")+21 AS  
"ReturnDate" FROM "Table"
```

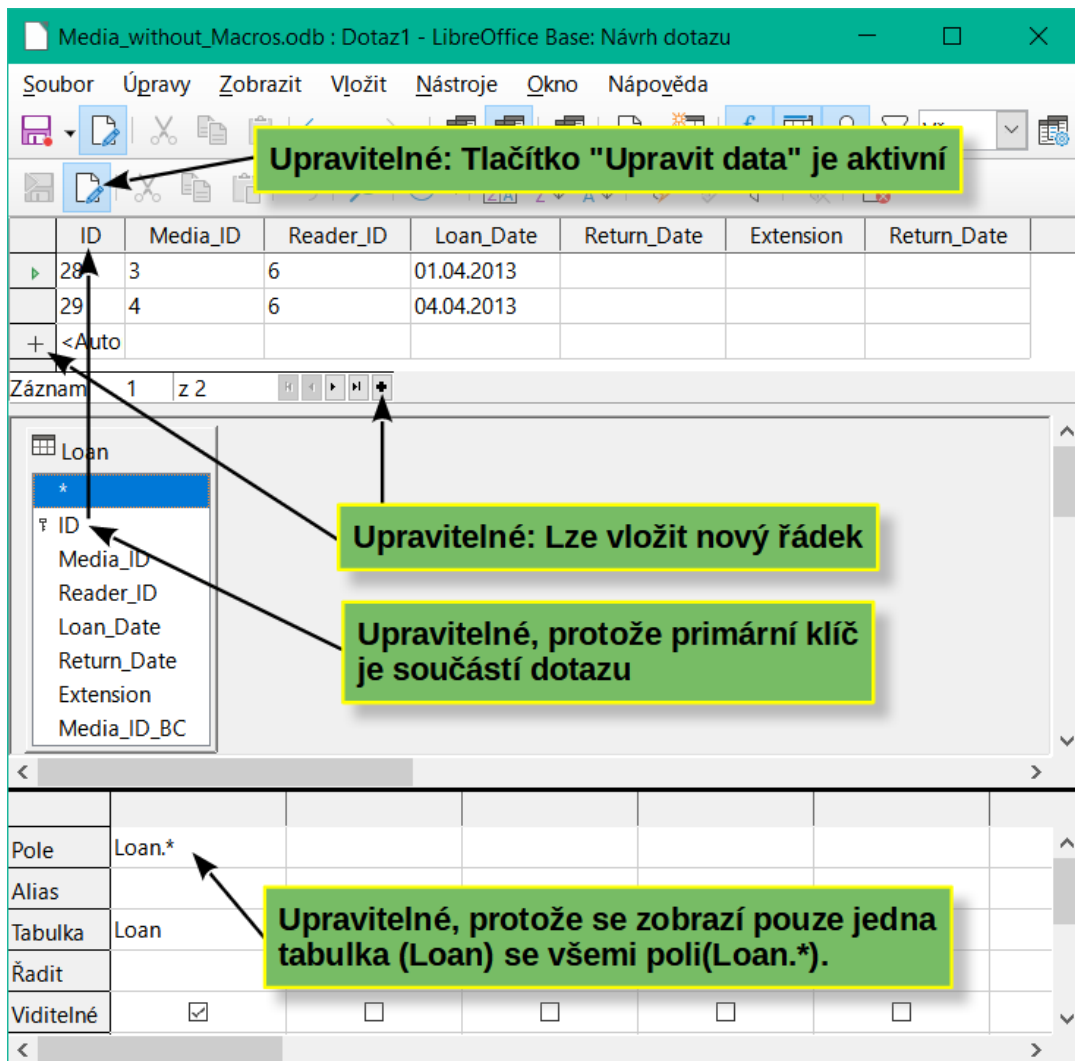
poskytne správné cílové datum pro vrácení ve formuláři. Tento dotaz počítá dny od 30.12.1899. Jedná se o výchozí datum, které Calc používá také jako nulovou hodnotu.

Vrácená hodnota je však pouze číslo, nikoli datum, které lze použít v dalším dotazu.

Vrácené číslo není vhodné pro použití v dotazech, protože formátování dotazů se neukládá. Místo toho je třeba vytvořit zobrazení.

Možnosti zadávání dat v rámci dotazů

Abychom mohli provádět záznamy do dotazu, který vyvolává pouze jednu tabulku, musí být přítomen primární klíč tabulky, která je základem dotazu, jak je znázorněno na obrázku 318.



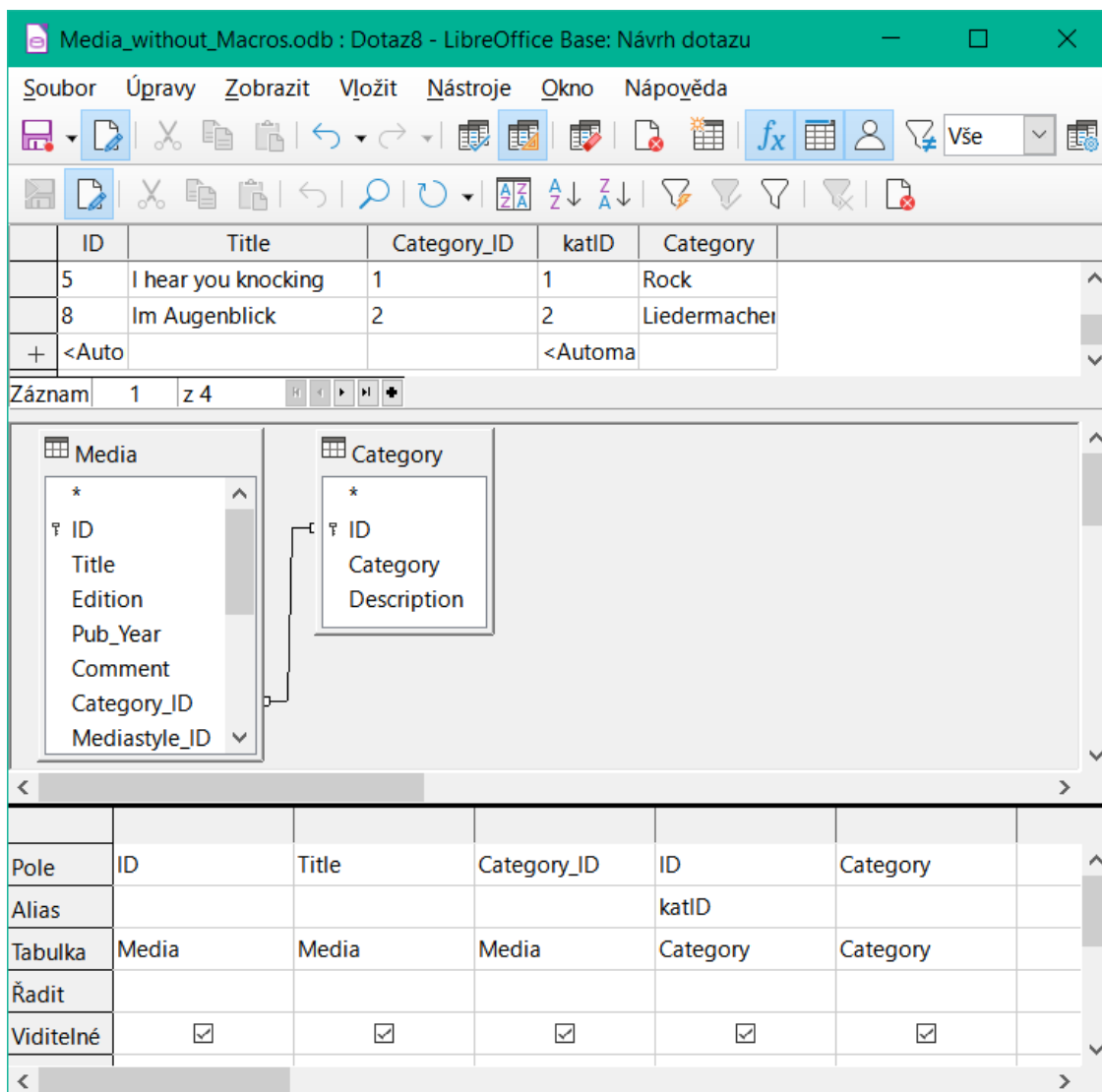
Obrázek 318: Podmínky pro to, aby byl dotaz editovatelný

Při půjčování médií nemá smysl vystavovat předměty, které již byly před časem vráceny.

```
SELECT "ID", "Reader_ID", "Media_ID", "Loan_Date"
FROM "Loan"
WHERE "Return_Date" IS NULL;
```

Formulář tak může v rámci kontrolního pole tabulky zobrazit vše, co si konkrétní čtenář v průběhu času vypůjčil. I zde musí dotaz filtrovat pomocí příslušné struktury formuláře (čtenář v hlavním formuláři, dotaz v podformuláři), aby se zobrazovala pouze média, která jsou skutečně vypůjčena. Dotaz je vhodný pro zadávání dat, protože je v něm obsažen primární klíč.

Pokud se dotaz skládá z více než jedné tabulky, je možné jej upravovat, pokud jsou přítomny primární klíče všech tabulek a k tabulkám se nepřistupuje prostřednictvím aliasu.

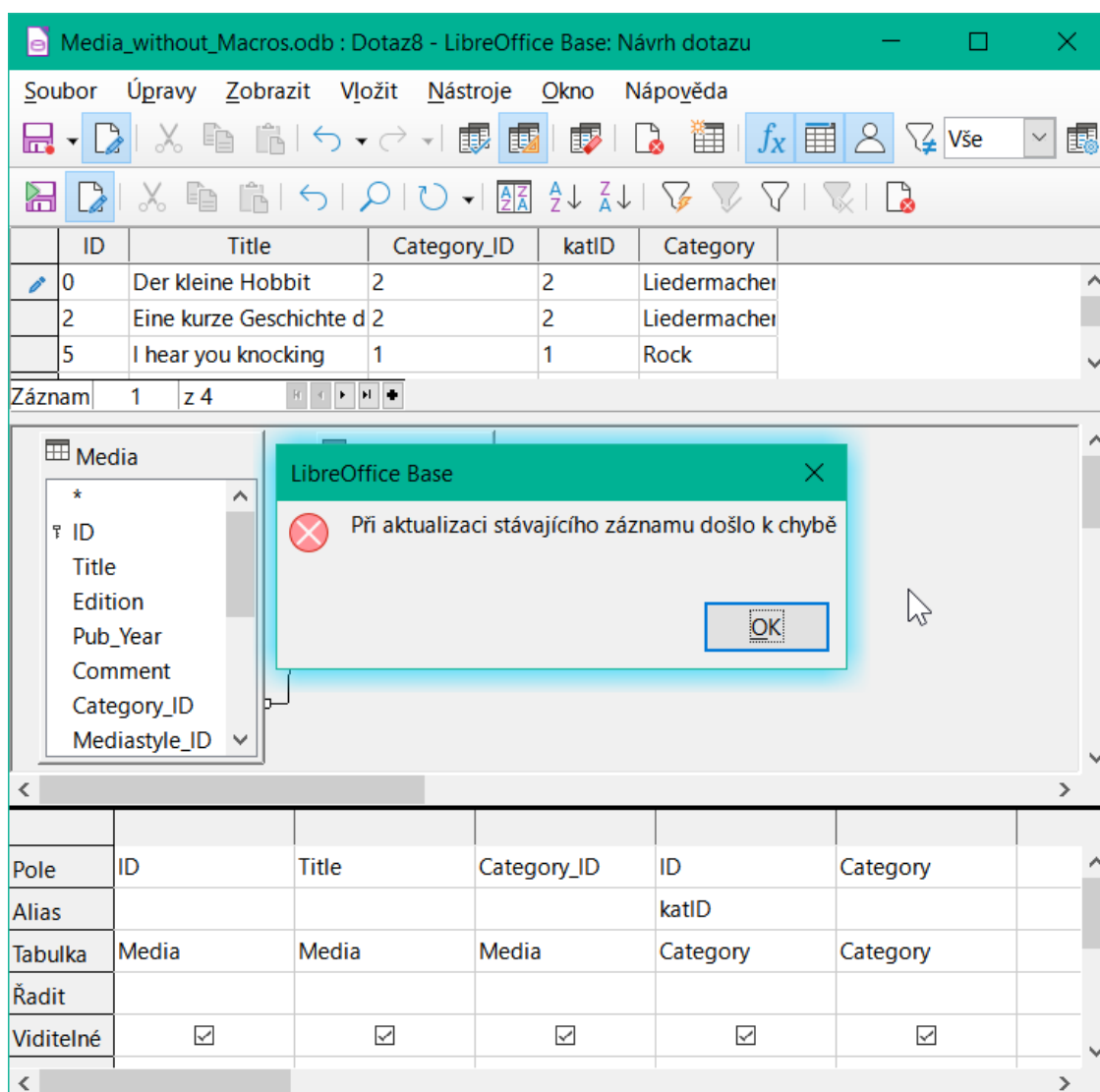


Obrázek 319: Příklad upravitelného dotazu zahrnujícího dvě tabulky

```
SELECT "Media"."ID", "Media"."Title", "Media"."Category_ID",
"Category"."ID" AS "katID", "Category"."Category"
FROM "Media", "Category"
WHERE "Media"."Category_ID" = "Category"."ID";
```

Dotaz na obrázku 319 je upravitelný, protože jsou v něm zahrnuty oba primární klíče a lze k nim přistupovat v tabulkách bez použití aliasu. Dotaz, který spojuje několik tabulek dohromady, rovněž vyžaduje přítomnost všech primárních klíčů.

V dotazu, který zahrnuje několik tabulek, není možné změnit pole cizího klíče v jedné tabulce, které odkazuje na záznam v jiné tabulce. V záznamu na obrázku 320 byl proveden pokus o změnu kategorie pro titul „The Little Hobbit“. Pole Category_ID bylo změněno z 0 na 2. Zdálo se, že změna proběhla a nová kategorie se objevila v záznamu. Ukázalo se však, že ji nelze uložit.



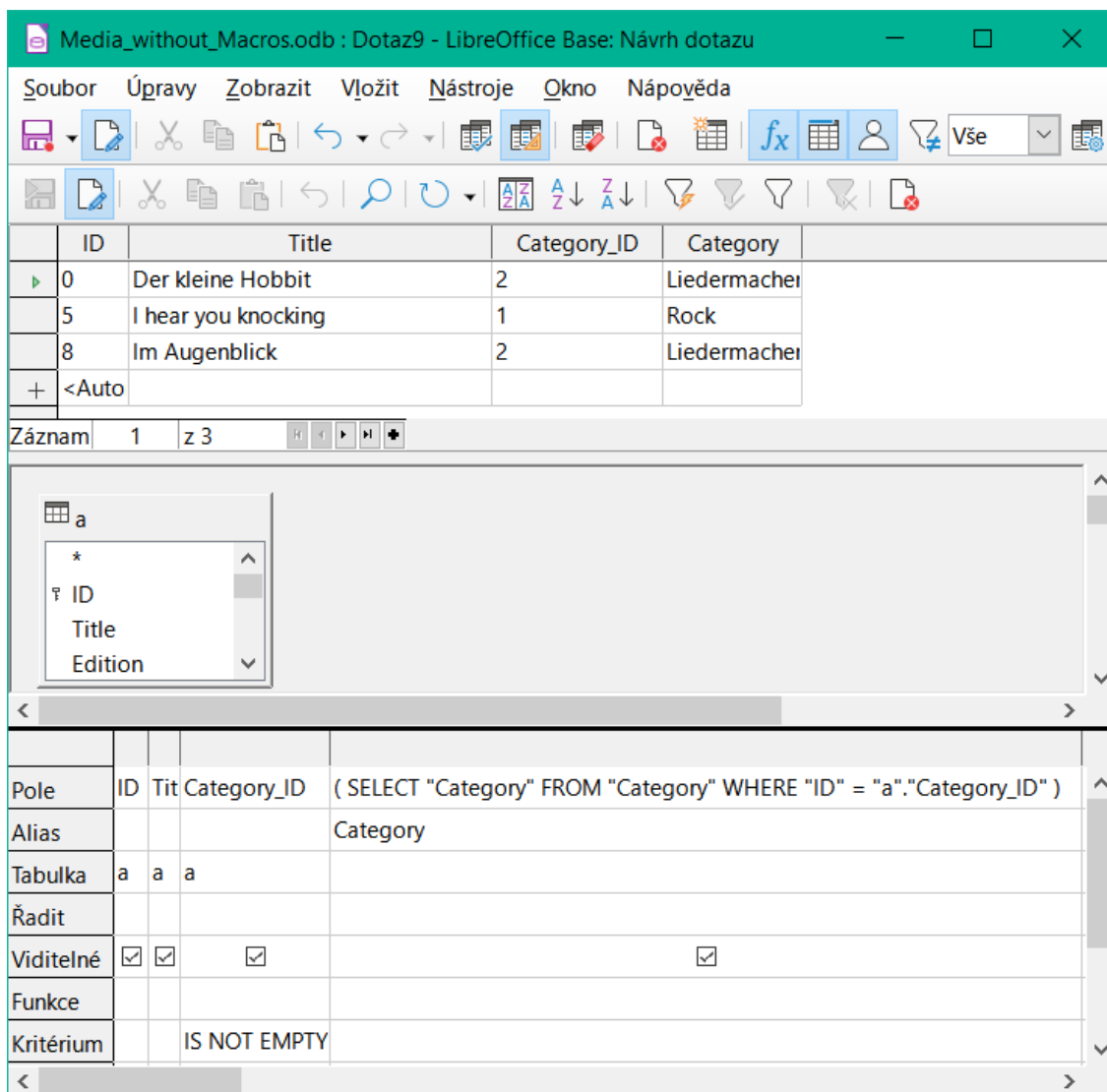
Obrázek 320: Chyba, když upravený dotaz nelze uložit

Je však možné upravit obsah příslušného záznamu kategorie, například nahradit „Fantasie“ za „Fantasy“. Název kategorie se pak změní pro všechny záznamy, které jsou s touto kategorií propojeny.

```
SELECT "m"."ID", "m"."Title", "Category"."Category", "Category"."ID" AS
"katID"
FROM "Media" AS "m", "Category"
WHERE "m"."Category_ID" = "Category"."ID";
```

V tomto dotazu se k tabulce Media přistupuje pomocí aliasu. Dotaz nelze upravovat. V tomto případě je jedno, zda je primární klíč v dotazu přítomen, nebo ne.

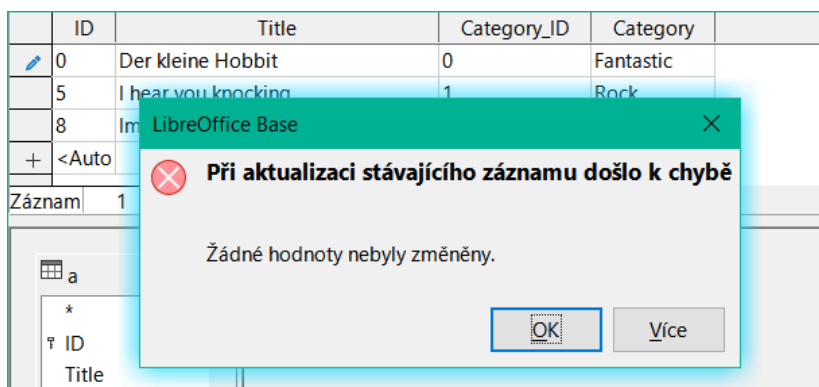
Ve výše uvedeném příkladu se tomuto problému snadno vyhneme. Pokud však použijeme korelovaný poddotaz (viz strana 289), musíme použít alias tabulky. Dotaz je v tomto případě upravitelný pouze tehdy, pokud obsahuje pouze jednu tabulku v hlavním dotazu.



Obrázek 321: Upravitelný dotaz pomocí korelovaného poddotazu s aliasem

V režimu zobrazení na obrázku 321 se zobrazí pouze jedna tabulka. Tabulce Media je přidělen alias, aby bylo možné přistupovat k obsahu pole Category_ID pomocí souvisejícího poddotazu.

V takovém dotazu je nyní možné změnit pole cizího klíče Category_ID na jinou kategorii. Ve výše uvedeném příkladu se pole Category_ID změnil z 0 na 2. Titul „The Little Hobbit“ je tedy zařazen do kategorie „Songwriter“.



Obrázek 322: Chyba při pokusu o změnu hodnoty v poli korelovaného poddotazu

Není však již možné změnit hodnotu v poli, které získalo svůj obsah prostřednictvím souvisejícího poddotazu. Pokus o změnu kategorie z „Fantasy“ na „Fantastic“ je znázorněn na obrázku 322. Tato změna není zaregistrována a nelze ji ani uložit. V tabulce, která se zobrazí v zobrazení návrhu, není pole Category přítomno.

Použití parametrů v dotazech

Pokud často používáme stejný základní dotaz, ale pokaždé s jinými hodnotami, můžeme použít dotazy s parametry. Dotazy s parametry fungují v zásadě stejně jako dotazy pro dílčí formulář:

```
SELECT "ID", "Reader_ID", "Media_ID", "Loan_Date"
FROM "Loan"
WHERE "Return_Date" IS NULL AND "Reader_ID"=2;
```

Tento dotaz zobrazuje pouze média půjčená čtenáři s číslem 2.

```
SELECT "ID", "Reader_ID", "Media_ID", "Loan_Date"
FROM "Loan"
WHERE "Return_Date" IS NULL AND "Reader_ID" =: Readernumber;
```

Nyní se při spuštění dotazu zobrazí vstupní pole. Vyzve nás k zadání čísla čtenáře. Ať už zde zadáme jakoukoli hodnotu, zobrazí se média, která jsou pro daného čtenáře aktuálně vypůjčena.

```
SELECT
  "Loan"."ID",
  "Reader"."LastName" || ', ' || "Reader"."FirstName",
  "Loan"."Media_ID",
  "Loan"."Loan_date"
FROM "Loan", "Reader"
WHERE "Loan"."Return_Date" IS NULL
  AND "Reader"."ID" = "Loan"."Reader_ID"
  AND "Reader"."LastName" LIKE '%' ||: Readername || '%'
ORDER BY "Reader"."LastName" || ', ' || "Reader"."FirstName" ASC;
```

Tento dotaz je zjevně uživatelsky přívětivější než předchozí. Číslo čtenáře již není nutné znát. Stačí zadat část příjmení a zobrazí se všechna média, která jsou půjčena odpovídajícím čtenářům.

Pokud vyměníme

```
"Reader"."LastName" LIKE '%' || :Readername || '%'
```

za

```
LOWER("Reader"."LastName") LIKE '%' || LOWER(: Readername) || '%'
```

Již nezáleží na tom, zda je jméno zadáno velkými nebo malými písmeny.

Pokud by parametr ve výše uvedeném dotazu zůstal prázdný, všechny verze LibreOffice až do verze 4.4 by zobrazily **všechny** čtenáře, protože pouze ve verzi 4.4 se prázdné pole parametru čte jako NULL, nikoli jako prázdný řetězec. Pokud toto chování nechceme, musíme mu zabránit pomocí triku:

```
LOWER ("Reader"."LastName") LIKE '%' || IFNULL(NULLIF (LOWER
(:Readername), ''), '$$') || '%'
```

Pole prázdného parametru vrací do dotazu prázdný řetězec, nikoli NULL. Proto musí být prázdnému poli parametru přiřazena vlastnost NULL pomocí NULLIF. Pak, protože položka parametru nyní dává hodnotu NULL, lze ji resetovat na hodnotu, která se normálně nevyskytuje v žádném záznamu. Ve výše uvedeném příkladu je to '\$\$'. Tuto hodnotu samozřejmě ve vyhledávání nenajdeme.

Od verze 4.4 jsou nutné úpravy této techniky dotazování:

```
LOWER ("Reader"."LastName") LIKE '%' || LOWER (:Readername) || '%'
```

musí v případě neexistence záznamu nevyhnutelně vést ke kombinaci:

```
'%' || LOWER (:Readername) || '%' a the NULL value.
```

Abychom tomu zabránili, přidáme další podmínku, že pro prázdné pole se skutečně zobrazí všechny hodnoty:

```
(LOWER ("Reader"."LastName") LIKE '%' || LOWER (:Readername) || '%'  
OR: Readername IS NULL)
```

Celou věc je třeba dát do závorek. Pak se buď hledá jméno, nebo pokud je pole prázdné (NULL z LibreOffice 4.4), použije se druhá podmínka.

Při použití formulářů lze parametr předat z hlavního formuláře do podformuláře. Někdy se však stává, že se dotazy s parametry v dílčích formulářích neaktualizují, pokud jsou data změněna nebo nově zadána.

Často by bylo vhodné měnit obsah polí se seznamem pomocí nastavení v hlavním formuláři. Mohli bychom například zabránit půjčování médií v knihovně osobám, které mají v současné době zakázáno si média půjčovat. Ovládání nastavení seznamu tímto personalizovaným způsobem pomocí parametrů není možné.

Poddotazy

Poddotazy zabudované do polí mohou vždy vrátit pouze jeden záznam. Pole může také vracet pouze jednu hodnotu.

```
SELECT "ID", "Income", "Expenditure",  
      ( SELECT SUM( "Income" ) - SUM( "Expenditure" )  
        FROM "Checkout" ) AS "Balance"  
FROM "Checkout";
```

Tento dotaz umožňuje zadávání dat (včetně primárního klíče). Dílčí dotaz poskytuje přesně jedna hodnotu, a to celkový zůstatek. To umožňuje po každém zadání odečíst zůstatek na pokladně. To ještě není srovnatelné s formulářem pokladny supermarketu popsáným v části „Dotazy jako základ pro další informace ve formulářích“ na straně 281. Samozřejmě chybí jednotlivé výpočty Total * Unit_price, ale také přítomnost čísla účtenky. Uvádí se pouze celkový součet. Pomocí parametru dotazu lze uvést alespoň číslo účtenky:

```
SELECT "ID", "Income", "Expenditure",  
      ( SELECT SUM( "Income" ) - SUM( "Expenditure" )  
        FROM "Checkout"  
        WHERE "Receipt_ID" = :Receipt_Number ) AS "Balance"  
FROM "Checkout" WHERE "Receipt_ID" =: Receipt_Number;
```

V dotazu s parametry musí být parametr v obou příkazech dotazu stejný, pokud má být rozpoznán jako parametr.

U podformulářů lze tyto parametry zahrnout. Podformulář pak místo názvu pole obdrží odpovídající název parametru. Tento odkaz lze zadat pouze ve vlastnostech podformuláře, nikoli při použití Průvodce.



Poznámka

Dílčí formuláře založené na dotazech nejsou automaticky aktualizovány na základě jejich parametrů. Vhodnější je předávat parametr přímo z hlavního formuláře.

Související poddotazy

Pomocí ještě dokonalejšího dotazu, editovatelného dotazu, můžeme dokonce přenášet průběžný zůstatek pokladny:

```
SELECT "ID", "Income", "Expenditure",  
  ( SELECT SUM( "Income" ) - SUM( "Expenditure" )  
    FROM "Checkout"  
    WHERE "ID" <= "a"."ID" ) AS "Balance"  
FROM "Checkout" AS "a"  
ORDER BY "ID" ASC
```

Tabulka Checkout je stejná jako tabulka "a". "a" však poskytuje **pouze** vztah k aktuálním hodnotám v tomto záznamu. Tímto způsobem lze v rámci poddotazu vyhodnotit aktuální hodnotu ID z vnějšího dotazu. V závislosti na ID se tedy určí předchozí zůstatek v příslušném čase, pokud vycházíme z toho, že ID, které je automatická hodnota, se zvyšuje samo o sobě.



Poznámka

Pokud má být obsah poddotazu filtrován pomocí funkce filtru editoru dotazů, funguje to v současné době pouze tehdy, pokud na začátku a na konci poddotazu použijeme dvojité závorky místo jednoduchých: ((SELECT ...)) AS "Saldo"

Dotazy jako zdrojové tabulky pro dotazy

Dotazem je nutné nastavit zámeček proti všem čtenářům, kteří obdrželi třetí oznámení o zpoždění pro médium.

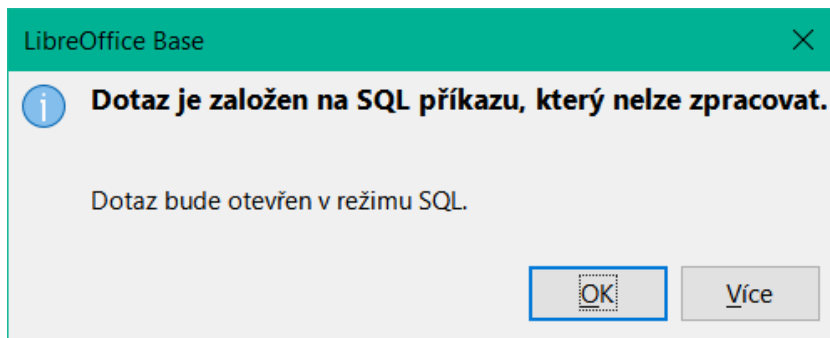
```
SELECT "Loan"."Reader_ID", '3rd Overdue - the reader is blacklisted' AS  
"Lock"  
FROM  
  (SELECT COUNT( "Date" ) AS "Total_Count", "Loan_ID"  
    FROM "Recalls" GROUP BY "Loan_ID") AS "a",  
  "Loan"  
WHERE "a"."Loan_ID" = "Loan"."ID" AND "a"."Total_Count" > 2
```

Nejprve prozkoumejme vnitřní dotaz, ke kterému se vnější dotaz vztahuje. V tomto dotazu se zjišťuje počet záznamů data seskupených podle cizího klíče Loan_ID. To nesmí být závislé na Reader_ID, protože by se tak započítávala nejen tři oznámení o zpoždění pro jedno médium, ale také tři média, každé s jedním oznámením o zpoždění. Vnitřnímu dotazu je přidělen alias, aby mohl být propojen s identifikátorem Reader_ID z vnějšího dotazu.

Vnější dotaz se v tomto případě vztahuje pouze k podmíněnému vzorci z vnitřního dotazu. Zobrazuje pouze Reader_ID a text pro pole Lock, pokud se "Loan"."ID" a "a"."Loan_ID" rovnají a "a"."Total_Count" > 2.

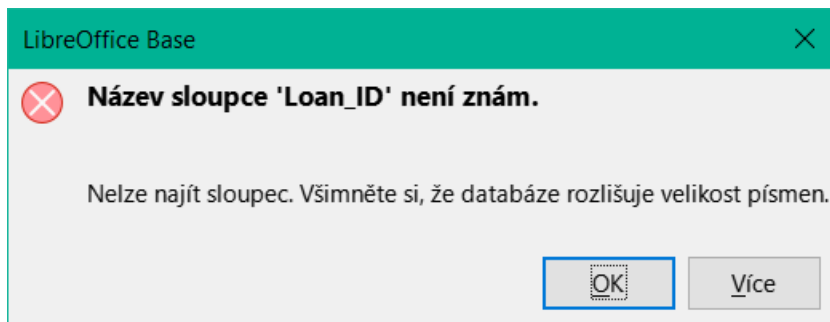
V zásadě jsou všechna pole vnitřního dotazu dostupná vnějšímu dotazu. Takže například součet "a"."Total_Count" může být sloučen do vnějšího dotazu, aby se získal skutečný součet pokut.

V dialogovém okně Návrh dotazu se však může stát, že po takové konstrukci již nebude režim zobrazení návrhu fungovat. Pokud se pokusíme dotaz znovu otevřít pro úpravy, zobrazí se upozornění na obrázku 323:



Obrázek 323: Chyba při pokusu o otevření dotazu, který nelze otevřít v režimu návrhu

Pokud pak otevřeme dotaz k úpravám v režimu SQL a pokusíme se z něj přepnout do režimu Návrh, zobrazí se chybová zpráva na obrázku 324:



Obrázek 324: Chyba při nemožnosti přepnout z režimu SQL do zobrazení návrhu

V režimu zobrazení návrhu nelze najít pole obsažené ve vnitřním dotazu "Loan_ID", kterým se řídí vztah mezi vnitřním a vnějším dotazem.

Při spuštění dotazu v režimu SQL se odpovídající obsah poddotazu reprodukuje bez chyby. Proto v tomto případě nemusíme používat přímý režim SQL.

Vnější dotaz použil výsledky vnitřního dotazu k vytvoření konečných výsledků. Jedná se o seznam hodnot "Loan_ID", které by měly být uzamčeny a proč. Pokud chceme konečné výsledky dále omezit, použijeme funkce řazení a filtrování v grafickém uživatelském rozhraní.

Následující snímky obrazovky ukazují, jakými různými způsoby lze dojít k výsledku dotazu s poddotazy. Zde se dotaz do skladové databáze snaží zjistit, co má zákazník zaplatit u pokladny. Jednotlivé ceny se vynásobí počtem zakoupených položek a získá se mezisoučet. Pak je třeba určit součet těchto dílčích součtů. To vše musí být možné upravovat, aby bylo možné dotaz použít jako základ formuláře.

	ID	quantity	article_ID	articleID	price	subtotal
	106	1	27	27	78,89	78,89
	107	2	28	28	1,25	2,5
	108	1	29	29	0,85	0,85
	109	2	30	30	0,65	1,3
+	<Auto			<Automatick		

Záznam 1 z 110 (1)

```

SELECT
"sale"."ID",
"sale"."quantity",
"article_ID",
"articles"."ID" AS "articleID",
"articles"."price",
"quantity" * "price" AS "subtotal"
FROM
"sale",
"articles"
WHERE
"sale"."article_ID" = "articles"."ID"

```

Obrázek 325: Jednoduchý dotaz pro vytvoření mezisoučtů

Dotaz znázorněný na obrázku 325 používá dvě tabulky (*sale* and *articles*). Aby bylo možné dotaz upravovat, musí být zahrnuti oba primární klíče.

✓ Poznámka

Kvůli chybě 61871 aplikace Base neaktualizuje dílčí výsledek automaticky.

	ID	quantity	article_ID	price	subtotal
	105	2	26	65,89	131,78
	106	1	27	78,89	78,89
	107	2	28	1,25	2,5
	108	1	29	0,85	0,85
▶	109	2	30	0,65	1,3
+	<AutomatickéPole>				

Záznam 110 z 110

```

SELECT
"sale"."ID",
"sale"."quantity",
"sale"."article_ID",
"articles"."price",
"quantity" * "price" AS "subtotal"
FROM "sale",
(SELECT "ID", "price" FROM "articles")
AS "articles"
WHERE "sale"."article_ID" = "articles"."ID"

```

Obrázek 326: Tabulka článků je přesunuta do poddotazu

Na obrázku 326 je tabulka *articles* přesunuta do poddotazu, který je vytvořen v oblasti tabulky (za výrazem „FROM“) a je mu přidělen alias. Nyní již není primární klíč tabulky *articles* nezbytně nutný k tomu, aby byl dotaz editovatelný.

	bill_ID	total
▶	0	1439,17
	1	728,24
	2	710,93

Záznam 1 z 3

```
SELECT "sale"."bill_ID",
SUM( "quantity" * "price" ) AS "total"
FROM "sale", "articles"
WHERE "sale"."article_ID" = "articles"."ID"
GROUP BY "sale"."bill_ID"
```

Obrázek 327: Výpočet hodnoty prodeje podle bill_id

Na obrázku 327 se v dotazu musí objevit vypočtená částka. Již jednoduchý dotaz pro výpočet součtu není editovatelný, takže je zde seskupen a sečten.

	ID	quantity	articles_ID	price	subtotal	total
▶	47	1	6	398,00	398,00	3607,70
	48	1	7	599,00	599,00	3607,70
+	<Auto					

Záznam 1 z 49

```
SELECT
    "sale"."ID",
    "sale"."quantity",
    "sale"."article_ID",
    "articles"."price",
    "quantity" * "price" AS "subtotal",
    "billtotal"."total"
FROM "sale",
    ( SELECT "ID", "price" FROM "articles" )
    AS "articles",
    ( SELECT "sale"."bill_ID",
        SUM( "quantity" * "price" ) AS "total"
      FROM "sale", "articles"
      WHERE "sale"."article_ID" = "articles"."ID"
      GROUP BY "sale"."bill_ID" ) AS "billtotal"
WHERE "sale"."article_ID" = "articles"."ID"
      AND "sale"."bill_ID" = "billtotal.bill_ID"
ORDER BY "sale"."bill_ID", "sale"."ID"
```

Obrázek 328: Použití dvou poddotazů

Na obrázku 328 se druhým poddotazem stane zdánlivě nemožné možným. Předchozí dotaz je vložen jako poddotaz do definice tabulky hlavního dotazu (za „FROM“). Celý dotaz tak zůstává upravitelný. V tomto případě je možné provádět záznamy pouze ve sloupcích „Sum“ a „WarID“. To je následně jasně uvedeno ve formuláři dotazu.

Shrnutí dat pomocí dotazů

Při vyhledávání dat v celé databázi vede použití jednoduchých formulářových funkcí často k problémům. Formulář se koneckonců vztahuje pouze k jedné tabulce a vyhledávací funkce prochází pouze podkladové záznamy tohoto formuláře.

Získání všech dat je jednodušší, pokud používáme dotazy, které mohou poskytnout obraz o všech záznamech. V části „Definice relace v dotazu“ je navržena taková konstrukce dotazu. Ten je pro ukázkovou databázi vytvořen takto:

```
SELECT "Media"."Title", "Subtitle"."Subtitle", "Author"."Author"
FROM "Media"
  LEFT JOIN "Subtitle"
    ON "Media"."ID" = "Subtitle"."Media_ID"
  LEFT JOIN "rel_Media_Author"
    ON "Media"."ID" = "rel_Media_Author"."Media_ID"
  LEFT JOIN "Author"
    ON "rel_Media_Author"."Author_ID" = "Author"."ID"
```

Zde jsou všechny tituly (Titles), podtituly (Subtitles) a autoři (Authors) zobrazení společně.

Tabulka Media obsahuje celkem 9 titulů. U dvou z těchto titulů je k dispozici celkem 8 podtitulů. Bez spojení LEFT JOIN se v obou tabulkách zobrazí dohromady pouze 8 záznamů. Pro každý podtitul se vyhledá odpovídající název a tím dotaz končí. Tituly bez podtitulů se nezobrazují.

Nyní se zobrazí všechna média včetně těch bez podtitulů: hodnota pole Media je na levé straně zadání, Subtitles na pravé straně. LEFT JOIN zobrazí všechny hodnoty Title z tabulky Media, ale hodnoty Subtitle pouze pro ty, které mají hodnotu Title. Tabulka Media se stává rozhodující tabulkou pro určení, které záznamy se mají zobrazit. To bylo plánováno již při sestavování tabulky (viz kapitola 3, Tabulky). Protože pro dva z devíti titulů existují podtituly, dotaz nyní zobrazí $9 + 8 - 2 = 15$ záznamů.



Poznámka

Běžné propojení tabulek po vypsání všech tabulek následuje po klíčovém slově WHERE.

V případě spojení LEFT JOIN nebo RIGHT JOIN se přiřazení definuje přímo za názvy obou tabulek pomocí ON. Pořadí je tedy vždy

```
Tabulka1 LEFT JOIN Tabulka2 ON Tabulka1.Pole1 = Tabulka2.Pole1
LEFT JOIN Tabulka3 ON Tabulka2.Pole1 = Tabulka3.Pole1...
```

Dva tituly v tabulce Media zatím nemají položku Author nebo Subtitle. Zároveň má jeden titul celkem tři autory. Pokud je tabulka Author propojena bez spojení LEFT JOIN, nezobrazí se dvě Media bez autora. Protože však jedno médium má tři autory místo jednoho, celkový počet zobrazených záznamů bude stále 15.

Pouze pomocí spojení LEFT JOIN bude dotaz instruován, aby použil tabulku Media k určení záznamů, které se mají zobrazit. Nyní se opět objeví záznamy bez hodnoty Subtitle nebo Author, celkem tedy 17 záznamů.

Použití vhodných spojů obvykle zvyšuje množství zobrazených dat. Tento rozšířený soubor dat lze však snadno skenovat, protože kromě názvů se zobrazují i autoři a podtituly. V ukázkové databázi lze přistupovat ke všem tabulkám závislým na médiích.

Rychlejší přístup k dotazům pomocí zobrazení tabulek

Pohledy v SQL jsou rychlejší než dotazy, zejména pro externí databáze, protože jsou ukotveny přímo v databázi a server vrací pouze výsledky. Naproti tomu dotazy jsou nejprve odeslány na server a tam zpracovány.

Pokud se nový dotaz vztahuje k jinému dotazu, režim SQL v databázi Base vytvoří z jiného dotazu tabulku. Pokud z něj vytvoříte View, uvidíte, že ve skutečnosti pracujete s poddotazem (Select použitý v rámci jiného Selectu). Z tohoto důvodu nelze Query 2, který se vztahuje k jinému Query 1, spustit přímo pomocí příkazu **Edit > Spustit SQL**, protože o Query 1 ví pouze grafické uživatelské rozhraní, a nikoli samotná databáze.

Databáze neumožňuje přímý přístup k dotazům. To platí i pro přístup pomocí maker. Naproti tomu k pohledům lze přistupovat jak z maker, tak z tabulek. V zobrazení však nelze upravovat žádné záznamy. (Musí být upraveny v tabulce nebo formuláři.)

Tip

Dotaz vytvořený pomocí funkce Vytvořit dotaz v režimu SQL má tu nevýhodu, že jej nelze řadit ani filtrovat pomocí grafického uživatelského rozhraní. Jeho použití je proto omezeno.

Naproti tomu Pohledy lze spravovat v aplikaci Base stejně jako běžnou tabulku – s tím rozdílem, že není možné měnit data. Zde jsou tedy i v přímých příkazech SQL k dispozici všechny možnosti řazení a filtrování.

Kromě toho se formátování sloupců v pohledu zachovává i po uzavření databáze, na rozdíl od sloupců v dotazu.

Pohledy jsou řešením mnoha dotazů, pokud chceme vůbec získat nějaké výsledky. Pokud se má například na výsledky dotazu použít podvýběr, vytvořte Pohled, který vám tyto výsledky poskytne. Poté použijeme dílčí výběr v okně Pohled. Odpovídající příklady najdeme v kapitole 8, Úlohy databáze.

Vytvoření pohledu z dotazu je poměrně snadné a přímočaré.

- 1) Klepneme na objekt **Tabulka** v části Databáze.
- 2) Klepneme na **Vytvořit pohled**.
- 3) Zavřeme dialogové okno Přidat tabulku.
- 4) Klepneme na ikonu **Režim návrh zap/vyp**. (Jedná se o režim SQL pro Pohled.)
- 5) Získání SQL pro Pohled:
 - a) Upravíme dotaz v režimu SQL.
 - b) Pomocí *Control+A* zvýrazníme SQL dotaz.
 - c) Pomocí *Control+C* zkopírujeme SQL.
- 6) V zobrazení režimu SQL použijeme klávesy *Control+V* pro vložení SQL.
- 7) Pohled zavřeme, uložíme a pojmenujeme.

Chyby ve výpočtech v dotazech

K výpočtu hodnot se používají také dotazy. Někdy interní databáze HSQLDB vykazuje zdánlivé chyby, které se při bližším zkoumání ukáží jako logicky správné interpretace dat. Objevují se také problémy se zaokrouhlováním, které mohou snadno způsobit zmatek.

Časy v HSQLDB jsou správně formátovány pouze do rozdílu 23:59:59 hodin. Pokud je třeba sečíst více časů, například pro výpočet odpracovaných hodin, je třeba najít jiný způsob. Zde existuje několik složitých přístupů:

- Čas se přímo vyjadřuje pouze v součtu minut nebo dokonce sekund. Výhoda: hodnoty umožňují následný bezproblémový výpočet.
- Čas je rozdělen na části hodiny, minuty a sekundy a znovu sestaven jako text s použitím znaku ':' jako oddělovače. Výhoda: text se v dotazech od začátku zobrazuje jako správně naformátovaný čas.
- Čas je vytvořen jako desetinné číslo. Den je 1, hodina je 1/24 atd. Výhoda: hodnoty lze následně v dotazu přeformátovat na čas a prezentovat je jako formátovatelné formulářové pole.

	DateTime_Start	DateTime_End	TimeDifference_Hours
▶	10.01.21 08:59	10.01.21 09:00	1

Záznam 1 z 1

```
SELECT "DateTime_Start", "DateTime_End",
DATEDIFF( 'hh', "DateTime_Start", "DateTime_End" )
AS "TimeDifference_Hours" FROM "Start-End"
```

Obrázek 329: Použití funkce DATEDIFF pro výpočet časového rozdílu v hodinách

Funkce DATEDIFF umožňuje určit časové intervaly. Výslovně se ptá na rozdíl, který má být určen. V příkladu na obrázku 329 nebyly požadovány minuty, ale jsou zohledněny všechny prvky, které jsou delší než minuta. Rozdíl mezi 8:59 a 9:00 je tedy jedna hodina (!). Naproti tomu rozdíl dat se počítá jako časový rozdíl v hodinách. Pokud je například pole Date_time_end nastaveno na 2.10.14 09:00, je to počítáno jako 25 hodin.

	DateTime_Start	DateTime_End	TimeDifference_Hours
▶	10.01.21 08:59	10.01.21 09:00	0

Záznam 1 z 1

```
SELECT "DateTime_Start", "DateTime_End",
DATEDIFF( 'hh', "DateTime_Start", "DateTime_End" ) / 60
AS "TimeDifference_Hours" FROM "Start-End"
```

Obrázek 330: Vypočítáme časový rozdíl v minutách a vydělíme jej 60.

Pokud se místo toho časový interval vypočítá v minutách a poté se vydělí 60, časový rozdíl v hodinách se stane nulovým, jak je znázorněno na obrázku 330. Tohle vypadá spíš jako správná hodnota, až na to, kam se poděla ta jedna minuta?

	DateTime_Start	DateTime_End	TimeDifference_Hours
▶	10.01.21 08:59	10.01.21 09:00	0,02

Záznam 1 z 1

```
SELECT "DateTime_Start", "DateTime_End",
DATEDIFF( 'hh', "DateTime_Start", "DateTime_End" ) / 60.00
AS "TimeDifference_Hours" FROM "Start-End"
```

Obrázek 331: Vypočítáme časový rozdíl v minutách a vydělíme jej číslem 60,00.

Na obrázku 331 je časový interval zadán jako celé číslo. Celé číslo bylo děleno celým číslem. Výsledkem dotazu musí být i v těchto případech celé, nikoli desetinné číslo. To lze snadno

opravit. Časový rozdíl v minutách se vydělí desetinným číslem se dvěma desetinnými místy (60,00). Výsledek má rovněž dvě desetinná místa. 0,02 hodiny stále není přesně jedna minuta, ale je jí mnohem blíže než dříve. Počet desetinných míst by bylo možné zvýšit použitím více nul. Perioda 0,016 je ještě bližší přiblížení, ale pozdější chyby ve výpočtu nelze vždy vyloučit.

Místo toho, abychom museli pracovat se spoustou přidaných nul, lze datový typ DATEDIFF ovlivnit přímo. Pomocí (CONVERT(DATEDIFF('mi', "Date_time_start", "Date_time_end"), DECIMAL(50, 49))) / 60 lze dosáhnout přesnosti 49 desetinných míst.

Při používání výpočetních funkcí si musíme vždy uvědomit, že datové typy v HSQLDB mají pouze omezenou přesnost. Ať už použijeme jakýkoli počet desetinných míst, faktem zůstává, že mezilehlé výsledky zahrnující počítání času lze pro další výpočty použít pouze v omezené míře.

Pokud má být časová hodnota následně použita ve formuláři nebo sestavě jako formátovaný čas, musíme zajistit, aby byl den platný jako základ pro formátování času.

	DateTime_Start	DateTime_End	Time_formatable
▶	10.01.21 08:59	10.01.21 09:00	00:01

Záznam 1 z 1

```
SELECT "DateTime_Start", "DateTime_End",
DATEDIFF( 'mi', "DateTime_Start", "DateTime_End" ) / 1440.0000
AS "Time_formatable" FROM "Table"
```

Obrázek 332: Vypočítáme časový rozdíl v minutách a vydělíme jej číslem 1440.0000.

Na obrázku 332 je rozdíl vypočítán v minutách. Výsledek se udává jako zlomek dne. Jeden den má 60 * 24 minut. Pokud bychom jednoduše vydělili 1440, výsledek by byl nula, takže opět musíme explicitně uvést desetinná místa. Pak se zobrazí jako formátovaný čas 0 hodin a 1 minuta.

Kód formátu pro čas delší než jeden den je [HH]:MM. Pokud použijeme nesprávný formát, může se časový rozdíl 1 den a 1 minuta zobrazit pouze jako 1 minuta.

	DateTime_Start	DateTime_End	TimeDifference_Minutes	Time_formatable
▶	10.01.21 08:59	10.01.21 09:09	10	00:09

Záznam 1 z 1

```
SELECT "DateTime_Start", "DateTime_End",
DATEDIFF( 'mi', "DateTime_Start", "DateTime_End" )
AS "TimeDifference_Minutes",
DATEDIFF( 'mi', "DateTime_Start", "DateTime_End" ) / 1440.0000
AS "Time_formatable" FROM "Table"
```

Obrázek 333: Vypočítáme časový rozdíl v minutách kromě časového rozdílu v minutách vyděleného 1440,0000.

Chybový ďábel znovu udeřil! Časový rozdíl 10 minut by se při správném formátování podle obrázku 333 neměl zobrazit jako 9 minut. Abychom zjistili, v čem spočívá problém, musíme se zabývat tím, jak přesně se výpočet provádí:

- 1) $10/1440 = 0.00694$. Výsledek je zaokrouhlen na 0,0069, protože byla zadána pouze čtyři desetinná místa.
- 2) $0,0069 * 1440 = 9,936$ minuty, což je 9 minut 56,16 sekundy. A sekundy se nezobrazují ve zvoleném formátu!

	DateTime_Start	DateTime_End	TimeDifference_Minutes	Time_formatable
	10.01.21 08:59	10.01.21 09:09	10	00:10

Záznam 1 z 1

```

SELECT "DateTime_Start", "DateTime_End",
DATEDIFF( 'mi', "DateTime_Start", "DateTime_End" )
AS "TimeDifference_Minutes",
DATEDIFF( 'mi', "DateTime_Start", "DateTime_End" ) / 1440.00000
AS "Time_formatable" FROM "Table"

```

Obrázek 334: Vypočítáme časový rozdíl v minutách kromě časového rozdílu v minutách vyděleného 1440,00000.

Konečně na obrázku 334 prodloužení dělitele o jedno desetinné místo (z 1440,0000 na 1440,00000) tuto chybu vyřeší. Nyní se zaokrouhluje na 0,00694. $0,00694 \cdot 1440$ dává 9,9936, což je 59,616 sekundy. Počet vteřin se zaokrouhluje na 60 vteřin, takže k 9 minutám se přičte 1 minuta, celkem tedy 10 minut.

I zde se mohou vyskytnout další problémy. Mohou existovat další desetinná místa, která při formátování nedávají 1 minutu? K vyřešení tohoto problému může pomoci krátký výpočet pomocí programu Calc s podobně zaokrouhlenými čísly. Sloupec A obsahuje posloupnost čísel od 1 (pro minuty). Sloupec B obsahuje vzorec $=\text{ROUND}(A1/1440;4)$ a je formátován tak, aby zobrazoval hodiny a minuty. Pokud budeme pokračovat směrem dolů, uvidíme vedle 10 minut ve sloupci A hodnotu 00:09 ve sloupci B. Podobně pro 28 minut atd. Pokud zaokrouhlíme na 5 míst, tyto chyby zmizí.

Jakkoli by bylo příjemné mít ve formuláři vhodně naformátované zobrazení, je třeba si uvědomit, že se jedná o zaokrouhlené hodnoty, které nejsou vhodné pro další použití při slepých mechanických výpočtech. Pokud je třeba hodnotu použít pro další výpočet, je lepší použít pouze vyjádření časového rozdílu v nejmenších dostupných jednotkách, v tomto případě v minutách.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 6 *Sestavy*

Vytváření sestav pomocí nástroje Návrhář sestav

Sestavy slouží k prezentaci dat způsobem, který je snadno pochopitelný i lidé bez znalosti databáze. Sestavy mohou:

- Prezentovat data v přehledných tabulkách.
- Vytvářet grafy pro zobrazení dat.
- Umožnit použití dat pro tisk štítků.
- Vytvářet formáty dopisů, jako jsou účty, oznámení o odvolání nebo oznámení lidem, kteří vstupují do sdružení nebo z něj vystupují.

Vytvoření sestavy vyžaduje pečlivou přípravu podkladové databáze. Na rozdíl od formuláře nemůže sestava obsahovat dílčí sestavy, a tedy ani další zdroje dat. Ani sestava nemůže prezentovat jiné datové prvky, než které jsou k dispozici v podkladovém zdroji dat, jako to může dělat formulář pomocí polí se seznamem.

Sestavy se nejlépe připravují pomocí dotazů. Tímto způsobem lze určit všechny proměnné. Zejména pokud je v rámci sestavy vyžadováno řazení, vždy použijeme dotaz, který řazení umožňuje. To znamená, že za těchto podmínek je třeba se vyhnout dotazům v přímém režimu SQL. Pokud musíme v databázi použít dotaz tohoto typu, můžeme řazení provést tak, že nejprve vytvoříme z tohoto dotazu pohled. Takové zobrazení lze vždy řadit a filtrovat pomocí grafického uživatelského rozhraní (GUI) aplikace Base.

Upozornění

Při používání nástroje Návrhář sestav bychom měli svou práci během úprav často ukládat. Kromě ukládání v samotném nástroji Návrhář sestav bychom měli po každém významném kroku uložit také celou databázi.

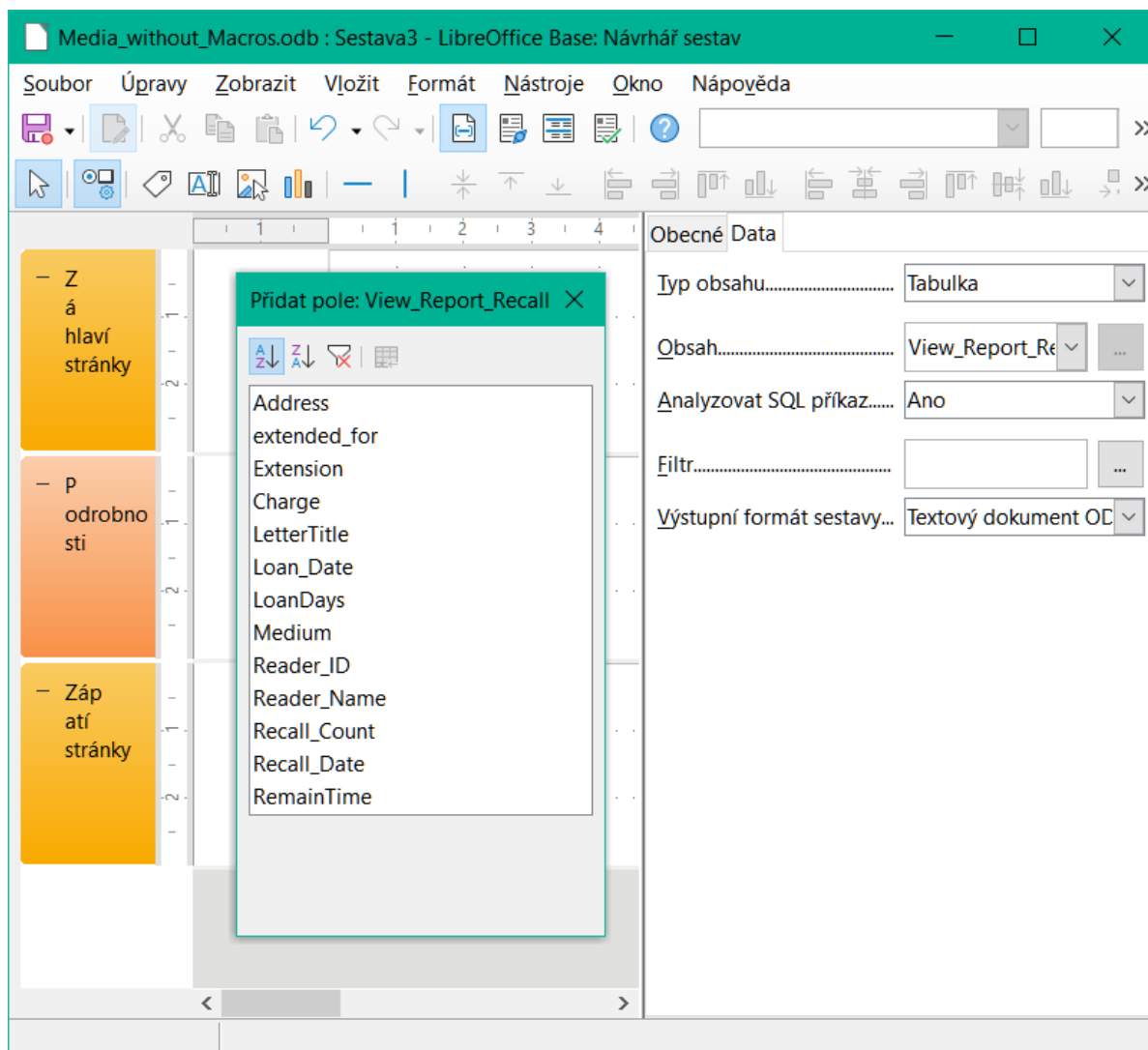
Uživatelské rozhraní nástroje Návrhář sestav

Chceme-li spustit nástroj Návrhář sestav z prostředí aplikace Base, použijeme příkaz **Sestavy > Vytvořit sestavu v režimu návrhu**.

Úvodní okno nástroje Návrhář sestav (obrázek 335) obsahuje tři části. Vlevo je aktuální rozdělení sestavy na Záhloví stránky, Podrobnosti a Zápatí stránky, uprostřed jsou příslušné oblasti, do kterých se bude vkládat obsah, a vpravo jsou zobrazeny vlastnosti těchto oblastí.

Současně se zobrazí dialogové okno Přidat pole. Tento dialog odpovídá dialogu při vytváření formuláře. Vytvoří pole s odpovídajícími popisky polí.

Bez obsahu z databáze nemá sestava správnou funkci. Z tohoto důvodu se dialogové okno otevře na kartě Data, kde lze nastavit obsah sestavy. Jako příklad se používá tabulka View_Report_Recall. Pokud je příkaz Analyzovat SQL nastaven na hodnotu Ano, lze sestavu třídit, seskupovat a filtrovat. Jako základ této sestavy byl vybrán pohled, takže nebude použit žádný filtr; ten již byl zahrnut do dotazu, který je základem pohledu.



Obrázek 335: Počáteční rozložení okna nástroje Návrhář sestav

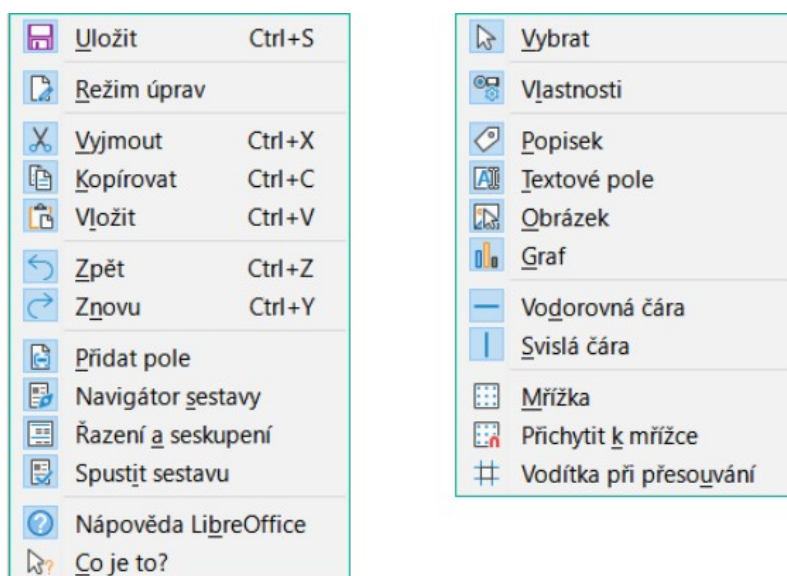
✓ Poznámka

Zadaným typem obsahu může být tabulka, pohled, dotaz nebo přímé kódování SQL. Nástroj Návrhář sestav pracuje nejlépe, pokud má k dispozici data, která byla pokud možno předem připravena. Tak lze například předem provést výpočty v dotazech a v případě potřeby omezit rozsah záznamů, které se mají v sestavě objevit.

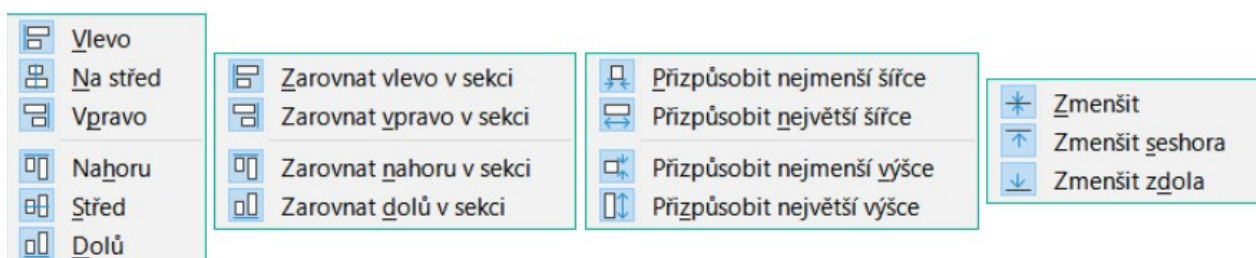
Na výběr jsou dva výstupní formáty sestav: Textový dokument ODF (dokument Writer) nebo Tabulkový procesor ODF (dokument Calc). Pokud chceme pouze tabulkové zobrazení dat, rozhodně bychom měli pro sestavu zvolit dokument Calc. Jeho vytvoření je výrazně rychlejší a je také snazší jej následně formátovat, protože je třeba zvážit méně možností a sloupce lze snadno přetáhnout na požadovanou šířku.

Ve výchozím nastavení hledá nástroj Návrhář sestav zdroj dat v první tabulce databáze. Tím je zajištěno, že je možné provést alespoň test funkcí. Před tím, než je možné sestavu vybavit poli, je třeba zvolit zdroj dat.

Nástroj Návrhář sestav nabízí mnoho dalších tlačítek, která jsou spolu s popisem zobrazena na obrázcích 336 a 337. Tlačítka pro zarovnání prvků nejsou v této kapitole dále popsána. Jsou užitečná pro rychlou úpravu polí v jedné oblasti nástroje Návrhář sestav, ale v zásadě lze vše provést přímou úpravou vlastností polí.

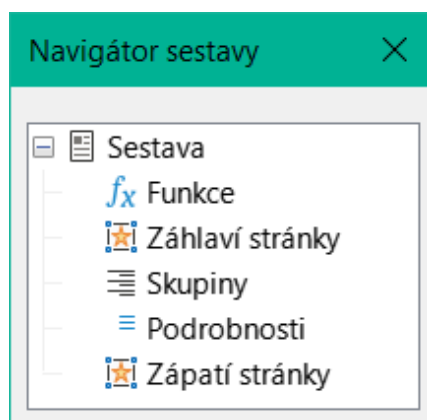


Obrázek 336: Tlačítka pro úpravu obsahu – Standardní nástrojová lišta (vlevo) a nástrojová lišta Ovládací prvky sestavy (vpravo)



Obrázek 337: Tlačítka pro zarovnání prvků – nástrojové lišty Zarovnat, Změna velikosti objektu, Zarovnat v sekci a Zmenšení v sekci (zleva doprava).

Stejně jako u formulářů je užitečné použít příslušný navigátor. Tak například neopatrné klepnutí na začátku nástroje Návrhář sestav může ztížit nalezení vlastností dat pro sestavu. Taková data mohou být dostupná pouze prostřednictvím navigátoru sestavy. Klepneme levým tlačítkem myši na položku Sestava a vlastnosti sestavy jsou opět přístupné.



Obrázek 338: Navigátor sestavy

Zpočátku se v navigátoru (viz obrázek 338) zobrazují části sestavy, a to kromě viditelných (Záhlaví stránky, Skupiny, Detail a Zápatí stránky) i neviditelné (Funkce). Skupiny lze použít například k přiřazení všech médií zapůjčených osobě, která si je vypůjčila, aby se předešlo vícenásobnému volání dotazu. Oblasti podrobností zobrazují záznamy patřící do skupiny. Funkce se používají k výpočtům, například k součtům.

Aby bylo možné v příkladu získat užitečný výstup, musí být obsah zobrazení reprodukován s vhodným seskupením. Každý čtenář by měl být propojen s oznámeními o stažení všech svých vypůjčených a nevrácených médií.

Zobrazit > Řazení a seskupení nebo odpovídajícím tlačítkem spustíme funkci seskupování.

Pole/výraz	
▶	Reader_Name

Akce skupin ↑ ↓ ✕

Vlastnosti

Řazení Vzestupně

Záhlaví skupiny Přítomno

Zápatí skupiny Není přítomno

Seskupení Každá hodnota

Interval seskupení 1

Udržet pohromadě Ne

Nápověda

Vyberte pole nebo typ výrazu k řazení nebo seskupení.

Obrázek 339: Dialogové okno Řazení a seskupení

Zde se seskupování a řazení provádí podle pole Reader_Name. Do výše uvedené tabulky lze zahrnout i další pole. Chceme-li například seskupovat a třídit také podle pole Loan_Date, zvolíme toto pole jako druhý řádek.

Přímo pod tabulkou je k dispozici několik akcí seskupení. Skupinu můžeme přesunout nahoru nebo dolů v seznamu nebo ji úplně odstranit. Vzhledem k tomu, že pro plánovanou sestavu je zapotřebí pouze jedna skupina, je na obrázku 339 zobrazen jako dostupný pouze symbol Odstranit v pravém krajním rohu skupinových akcí.

Vlastnost Řazení je zřejmá.

Po vytvoření položky se v levé části nástroje Návrhář sestav okamžitě zobrazilo nové rozdělení. Vedle popisu pole Reader_Name nyní vidíme položku Header. Tato část je určena pro záhlaví skupiny v sestavě. Záhlaví může obsahovat jméno osoby, která obdrží upomínku k vrácení médií. V tomto případě není zápatí skupiny. Taková patička by mohla obsahovat dlužnou pokutu nebo místo a aktuální datum a prostor pro podpis osoby, která oznámení odesílá.

Ve výchozím nastavení je pro každou hodnotu vytvořena nová skupina. Pokud se tedy změní hodnota pole `Reader_Name`, založí se nová skupina. Můžeme je také seskupit podle počátečního písmene. V případě upomínky by však všichni čtenáři se stejnou iniciálou byli zařazeni do jedné skupiny. Schmidt, Schulze a Schulte by obdrželi společnou upomínku, což by v tomto příkladu bylo zcela zbytečné.

Při seskupování podle počátečního písmene můžeme navíc určit, o kolik písmen později má začínat další skupina. Lze si představit například seskupení pro malý telefonní seznam. Podle velikosti seznamu kontaktů si lze představit seskupení na každé druhé počáteční písmeno. Takže A a B by tvořily první skupinu, pak C a D a tak dále.

Skupinu lze nastavit buď tak, aby byla vedena společně s prvním oddílem údajů, nebo pokud možno jako kompletní skupina. Ve výchozím nastavení je tato možnost nastavena na hodnotu `Ne`. Pro oznámení o odvolání bychom pravděpodobně chtěli skupinu uspořádat tak, aby se pro každou osobu, která má obdržet odvolání, vytiskla samostatná stránka. V další nabídce můžeme zvolit, aby za každou skupinou (v tomto případě za každým jménem čtenáře) následoval před zpracováním další hodnoty zlom stránky.

Pokud jsme zvolili záhlaví skupiny a případně i zápatí skupiny, zobrazí se tyto prvky jako sekce v navigátoru sestavy pod odpovídajícím názvem pole `Reader_Name`. I zde máme možnost používat funkce, které pak budou omezeny na tuto skupinu.

Chceme-li přidat pole, použijeme funkci `Přidat pole`, stejně jako u formulářů. V tomto případě však popisek a obsah pole nejsou svázány dohromady. Oběma lze nezávisle pohybovat, měnit jejich velikost a přetahovat je do různých částí.

Obrázek 340 zobrazuje návrh sestavy pro oznámení o odvolání. V záhlaví stránky je nadpis `Knihovna Libre Office`, vložený jako pole s popisem. Zde můžeme mít také hlavičkový papír s logem, protože může obsahovat grafiku. Tato úroveň se nazývá `Záhlaví stránky`, ale to neznamená, že nad ní není žádný prostor. To závisí na nastavení stránky; pokud byl nastaven horní okraj, nachází se tento okraj nad záhlavím stránky.

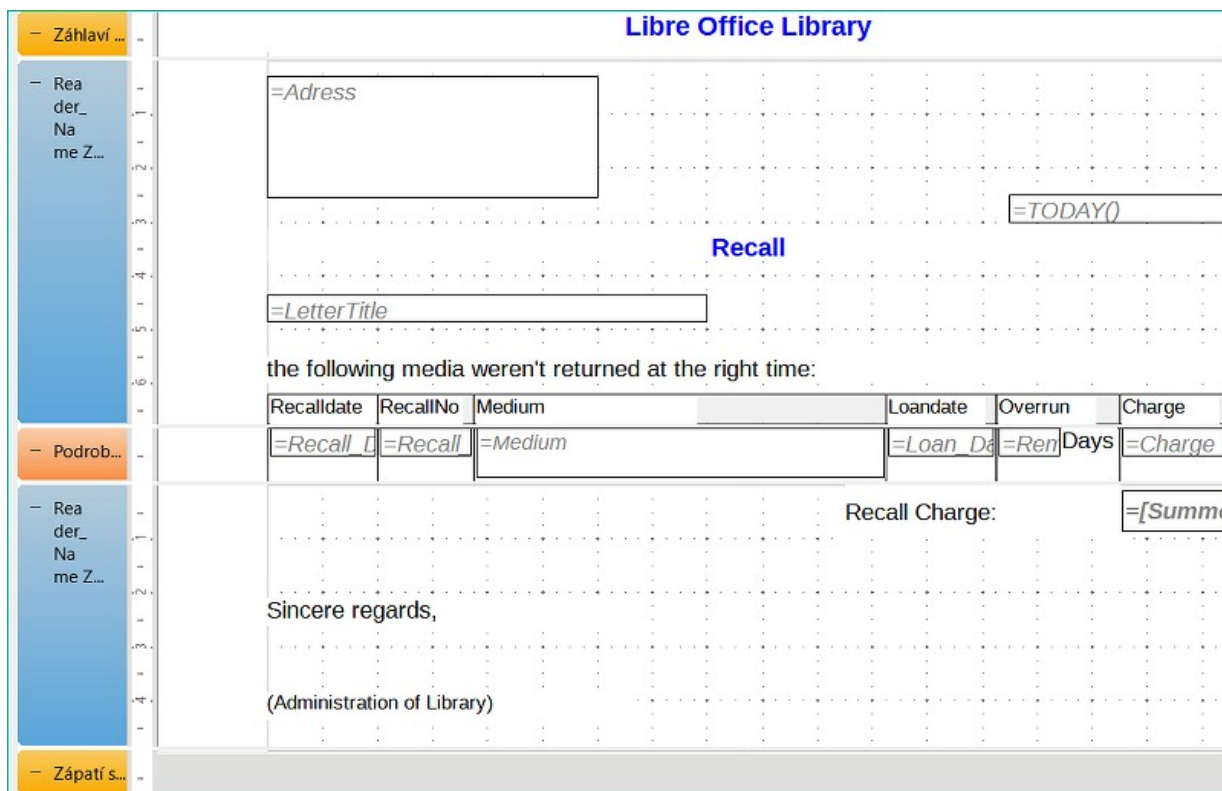
Záhlaví `Reader_Name` je záhlaví seskupených a seřazených dat. V polích, která mají obsahovat data, jsou názvy příslušných datových polí zobrazeny světle šedou barvou. Tak například zobrazení, které je základem sestavy, má pole s názvem `Address`, které obsahuje kompletní adresu příjemce s ulicí a městem. Vložení tohoto údaje do jediného pole vyžaduje v dotazu zalomení řádku. K jejich vytvoření můžeme použít `CHAR(13) || CHAR(10)`.

Příklad:

```
SELECT "Salutation" || CHAR(13) || CHAR(10) || "FirstName" || ' ' || "LastName" ||  
CHAR(13) || CHAR(10) || "Street" || ' ' || "No" || CHAR(13) || CHAR(10) || "Postcode" || '  
' || "Town" AS "Adress" FROM "Reader"
```

Pole `=TODAY()` představuje vestavěnou funkci, která na tuto pozici vloží aktuální datum.

V Záhlaví `Reader_Name` se kromě pozdravu zobrazují záhlaví sloupců pro následující zobrazení tabulky. Tyto prvky by se měly objevit pouze jednou, i když je v seznamu uvedeno několik médií.



Obrázek 340: Návrh zprávy pro příklad oznámení upomínky

Na pozadí těchto záhlaví sloupců je šedý obdélník, který slouží také jako rámeček pro data.

Oblast Podrobnosti se opakuje tolikrát, kolikrát existují samostatné záznamy se stejným údajem Reader_Name. Zde jsou uvedena všechna média, která nebyla vrácena včas. V pozadí je další obdélník, který rámuje obsah. Tento obdélník je vyplněn bílou barvou, nikoli šedou.

✓ Poznámka

LibreOffice v zásadě umožňuje přidávat vodorovné a svislé čáry. Tyto linie mají tu nevýhodu, že jsou interpretovány pouze jako vlasové linie. Lze je lépe reprodukovat, pokud se použijí obdélníky. Nastavíme pozadí obdélníku na černou barvu a jeho velikost například na šířku 17 cm a výšku 0,03 cm. Tím se vytvoří vodorovná čára o tloušťce 0,03 cm a délce 17 cm.

Tato varianta má však také nevýhodu: grafické prvky nelze správně umístit, pokud se oblast rozprostírá na více než jedné stránce.

Zápatí Reader_Name uzavírá dopis pozdravem a prostorem pro podpis. Zápatí je definováno tak, že za touto oblastí dojde k dalšímu zalomení stránky. Na rozdíl od výchozího nastavení je také stanoveno, že tato oblast by měla být ve všech případech pohromadě. Koneckonců by vypadalo poněkud podivně, kdyby více upomínek obsahovalo podpis na samostatné stránce.

Udržet pohromadě zde odkazuje na zalomení stránky. Pokud chceme, aby obsah záznamu zůstal pohromadě nezávisle na zalomení, je to v současné době možné pouze v případě, že záznam není načten jako Podrobnosti, ale je použit jako základ pro seskupení. Můžeme zvolit možnost Udržet pohromadě = Ano, ale nefunguje to; oblast Podrobnosti se oddělí. Abychom udrželi obsah Podrobnosti pohromadě, musíme jej zařadit do samostatné skupiny.

Obecné

Název..... Group Footer

Vynutit novou stránku..... Po sekci

Udržet pohromadě..... Ano

Opakovat sekci..... Ne

Viditelné..... Ano

Výška..... 4,87 cm

Výraz pro podmíněný tisk..

Průhlednost pozadí..... Ano

Barva pozadí..... Východí

Obrázek 341: Vlastnosti pro zápatí
Reader_Name

K výpočtu celkových pokut se používá vestavěná funkce.

Upomínka je zobrazena na obrázku 342. Oblast s podrobnostmi obsahuje položky médií, které si čtenář vypůjčil. Zápatí skupiny obsahuje celkovou dlužnou pokutu.

Libre Office Library

Mr.
Heinrich Müller
Nowhereroad 14 b
GB 3P67Q Downtown

11/24/12

Recall

Dear Mr. Müller,

the following media weren't returned at the right time:

Recalldate	RecallNo	Medium	Loandate	Overrun	Charge
24.11.12	1	5 - I hear you knocking - by Edmunds, Dave	29.04.12	202 Days	\$7.00
24.11.12	1	8 - Im Augenblick - by van Veen, Herman	22.04.12	209 Days	\$7.25
24.11.12	1	2 - Eine kurze Geschichte der Zeit - by Hawking, Steven W.	04.04.12	213 Days	\$7.50

Recall Charge: **\$21.75**

Sincere regards,

(Administration of Library)

Obrázek 342: Sestava upomínky

✓ Poznámka

Sestavy pro jednotlivé záznamy mohou být také rozsáhlejší než jedna stránka. Velikost reportu je zcela odlišná od velikosti stránky. Roztažení oblasti s podrobnostmi na více než jednu stránku však může vést k chybným zlomům. Zde má nástroj Návrhář sestav stále problémy se správným výpočtem rozestupů. Pokud jsou zahrnuty jak oblasti seskupení, tak grafické prvky, může to vést k nepředvídatelným velikostem některých oblastí.

Jednotlivé prvky lze zatím přesouvat na pozice mimo velikost jedné stránky pouze pomocí myši a kurzorových kláves. Vlastnosti prvků udávají vždy stejnou maximální vzdálenost od horního rohu libovolné oblasti, která leží na první stránce.

Obecné vlastnosti polí

Existují pouze tři typy polí pro prezentaci dat. Kromě textových polí (která mohou v rozporu se svým názvem obsahovat také čísla a formátování) existuje také typ pole, které může obsahovat obrázky z databáze. V poli Charge se zobrazí souhrn údajů na obrázku 343.

Obecné Data	
Název.....	Charge
Viditelné.....	Ano
Pozice X.....	15,55 cm
Pozice Y.....	0,00 cm
Šířka.....	1,85 cm
Výška.....	0,50 cm
Automaticky.....	Ne
Výraz pro podmíněný tisk.....	
Tisk opakovaných hodnot.....	Ano
Tisknout opakované hodnoty při změně skupiny...	Ano
Průhlednost pozadí.....	Ano
Barva pozadí.....	Výchozí
Písmo.....	Liberation Sans, Běžné, 12
Vodorovné zarovnání.....	Vlevo
Svislé zarovnání.....	Nahoru
Formátování.....	1,234.57 €

Obrázek 343: Vlastnosti (karta Obecné) pro pole Charge

Stejně jako u formulářů jsou pole pojmenována. Ve výchozím nastavení se použije název podkladového databázového pole.

Pole lze nastavit jako neviditelné. V případě polí se to může zdát poněkud zbytečné, ale je to užitečné pro záhlaví a zápatí skupin, které mohou být vyžadovány k provádění jiných funkcí seskupení, aniž by obsahovaly cokoli, co je třeba zobrazit.

Pokud je možnost Tisk opakovaných hodnot deaktivována, zobrazení pole se zablokuje, pokud je bezprostředně předtím načteno pole se stejným obsahem. To funguje správně pouze pro datová

pole, která obsahují text. Číselná pole nebo pole s datem pokyn k deaktivaci ignorují, pole Popisku jsou při deaktivaci zcela vybledlá, i když se vyskytnou pouze jednou.

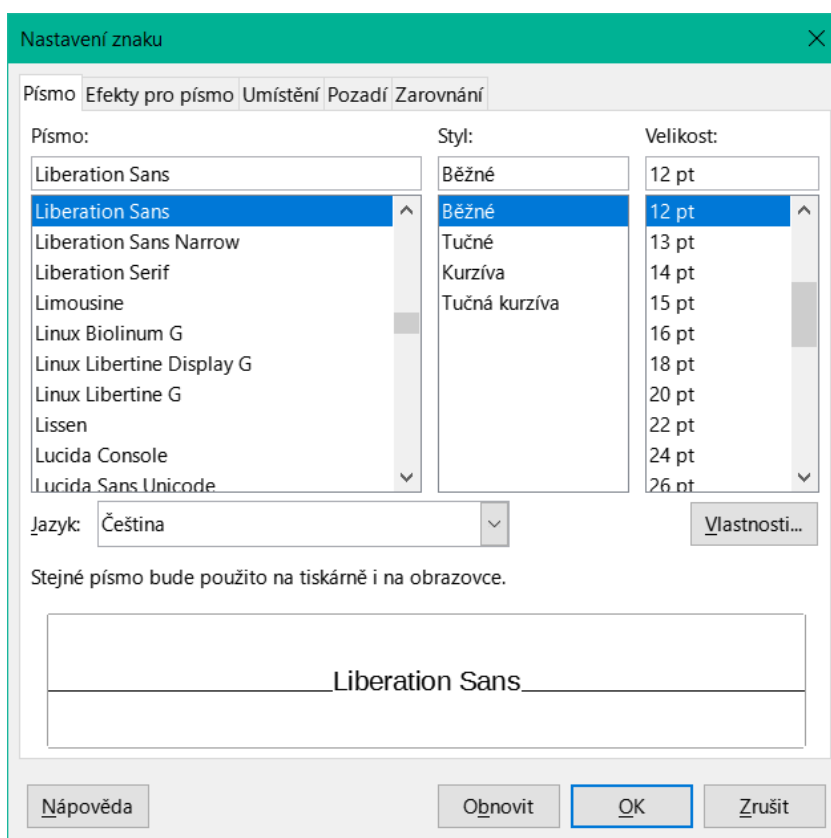
V nástroji Návrhář sestav lze pomocí podmíněného tiskového výrazu zakázat zobrazení určitého obsahu nebo lze hodnotu pole použít jako základ pro formátování textu a pozadí. Více o podmíněných výrazech je uvedeno v části „Podmíněný tisk“ na straně 321.

Nastavení kolečka myši nemá žádný vliv, protože pole sestavy nelze upravovat. Zdá se, že jde o pozůstatek z dialogového okna definice formuláře.

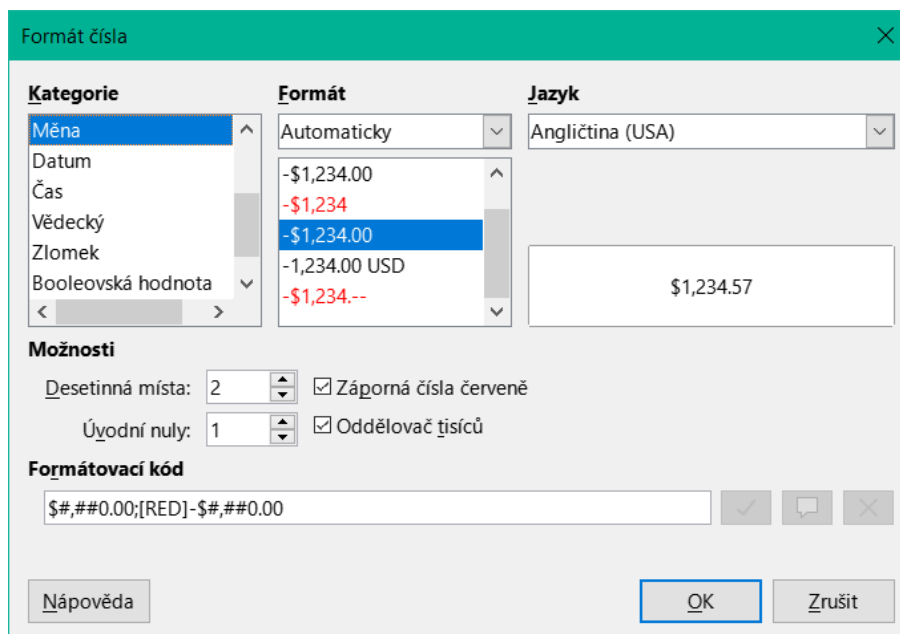
Funkci Tisk při změně skupiny nebylo možné reprodukovat ani v sestavách.

Pokud není pozadí definováno jako průhledné, lze pro každé pole definovat barvu pozadí.

Ostatní položky se týkají vnitřního obsahu daného pole. Jedná se o písmo (pro barvu písma, váhu písma atd., viz obrázek 344), zarovnání textu v poli a formátování pomocí příslušného dialogového okna Znaky (viz obrázek 345).

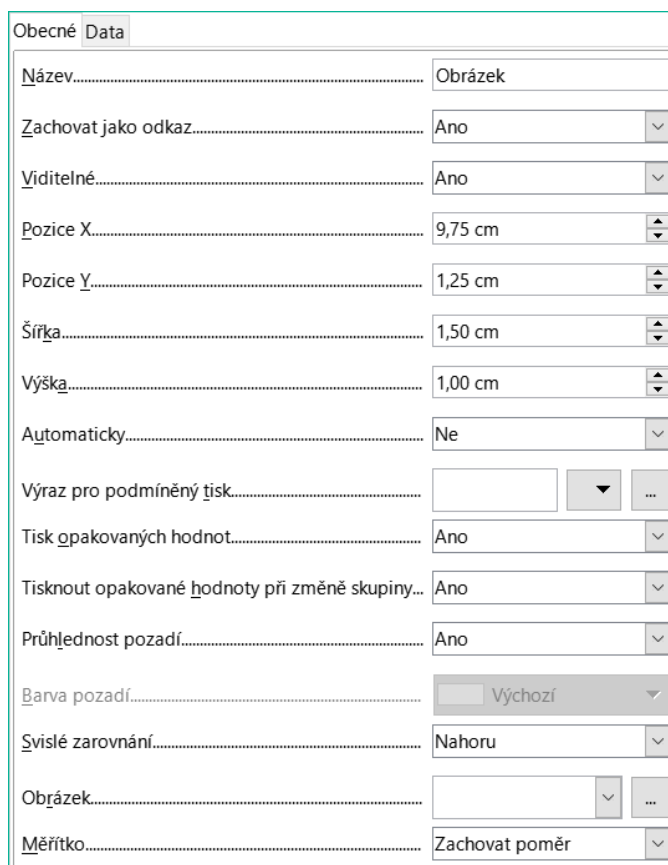


Obrázek 344: Dialogové okno Nastavení znaků



Obrázek 345: Dialogové okno Formát čísla

Zvláštní vlastnosti grafických ovládacích prvků



Obrázek 346: Vlastnosti (karta Obecné) pro ovládací prvek obrázku

Grafický ovládací prvek (viz obrázek 346) může obsahovat grafiku z databáze i mimo ni. V současné době však není možné uložit grafiku, například logo, trvale do databáze aplikace Base. Proto je nezbytné, aby grafika byla k dispozici v cestě vyhledávání, i když je nám nabídnuta volba vložení, nikoli propojení obrázků, a první pole Nastavit jako odkaz lze nastavit

(doslova uzavřít) na odpovídající plánovanou funkci. Jedná se o jednu z několika funkcí, které jsou plánovány pro aplikaci Base a jsou v grafickém uživatelském rozhraní, ale ve skutečnosti ještě nebyly implementovány – takže tlačítka a zaškrťovací políčka nemají žádný účinek.

Případně může být grafika uložena v samotné databázi, takže je k dispozici interně. V takovém případě však musí být přístupný prostřednictvím jednoho z polí v dotazu, který je základem sestavy.

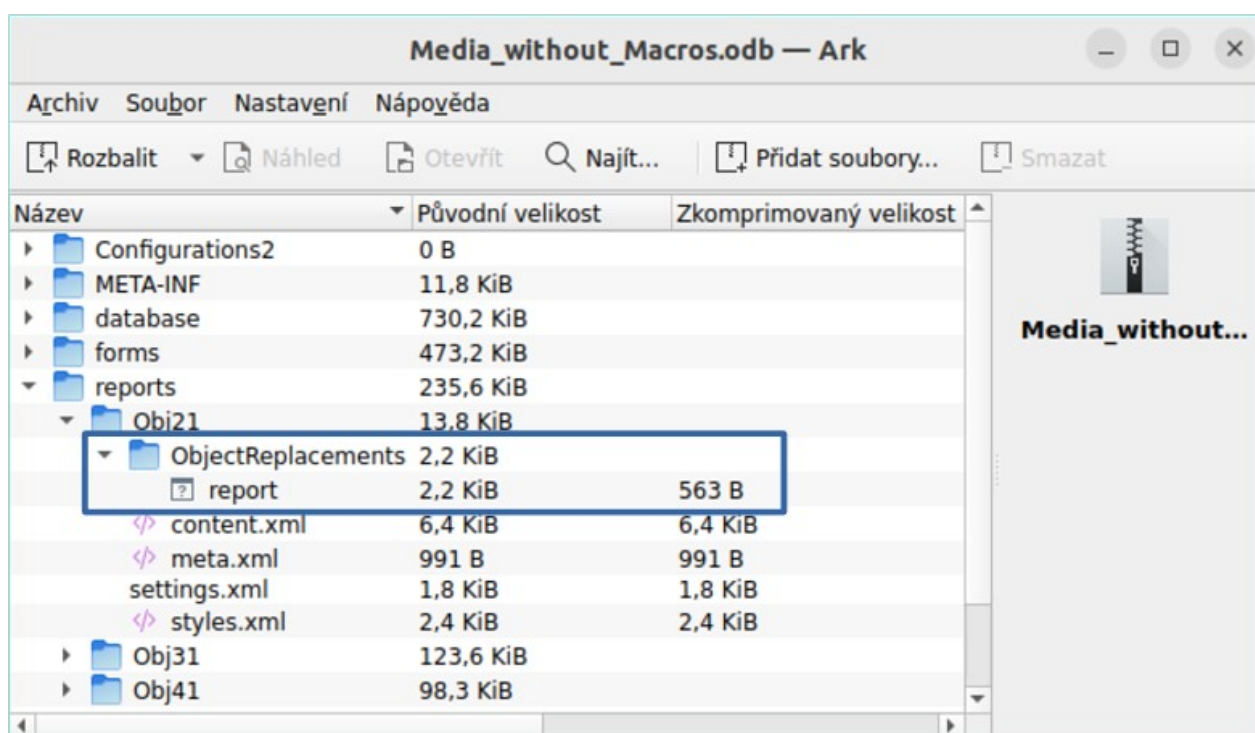
Chceme-li načíst externí grafiku, použijeme tlačítko výběru vedle pole Grafika. Chceme-li načíst grafické databázové pole, zadáme pole na kartě Data.

Zdá se, že nastavení vertikálního zarovnání nemá ve fázi návrhu žádný vliv. Po vyvolání sestavy se však grafika zobrazí na správném místě.

Při škálování můžeme vybrat možnost *Ne*, *Zachovat poměr stran* nebo *Automatická velikost*. To odpovídá nastavení formuláře:

- **Ne:** Obrázek není přizpůsoben ovládacímu prvku. Pokud je příliš velký, zobrazí se oříznutá verze. Původní obrázek tím není ovlivněn.
- **Zachovat poměr stran:** Obrázek je přizpůsoben ovládacímu prvku, ale není zkreslený.
- **Automatická velikost:** Obrázek se přizpůsobí ovládacímu prvku a v některých případech může být zkreslený.

Při úpravách sestav obsahujících obrázky se může stát, že se databáze výrazně zvětší. Uvnitř souboru *.odb zobrazeného na obrázku 347 umístí Base z ne zcela pochopitelných důvodů do adresáře sestavy složku ObjectReplacements. Tato složka obsahuje soubor „report“, který je zodpovědný za zvětšení.



Obrázek 347: Obsah souboru ODB zobrazený pomocí správce archivu souborů Ark

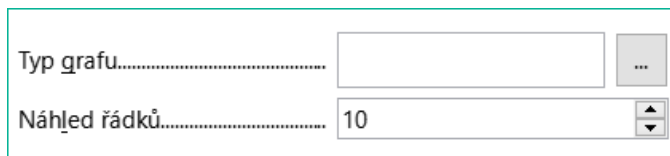
Pokud je databáze otevřena v archivačním programu, je tato složka s obsahem viditelná v podadresáři Reports. Složku můžeme bezpečně najít a odstranit pomocí archivačního programu.

✓ Poznámka

Pokud se sestavy nebudou opakovaně upravovat, stačí složku ObjectReplacements odstranit jednou. Velikost této složky může velmi rychle růst. To závisí na počtu a velikosti zahrnutých souborů. Test ukázal, že jediný soubor jpg o velikosti 2,8 MB zvětšil soubor *.odb o 11 MB! Viz [Bug 80320](#).

Začlenění grafů do zprávy

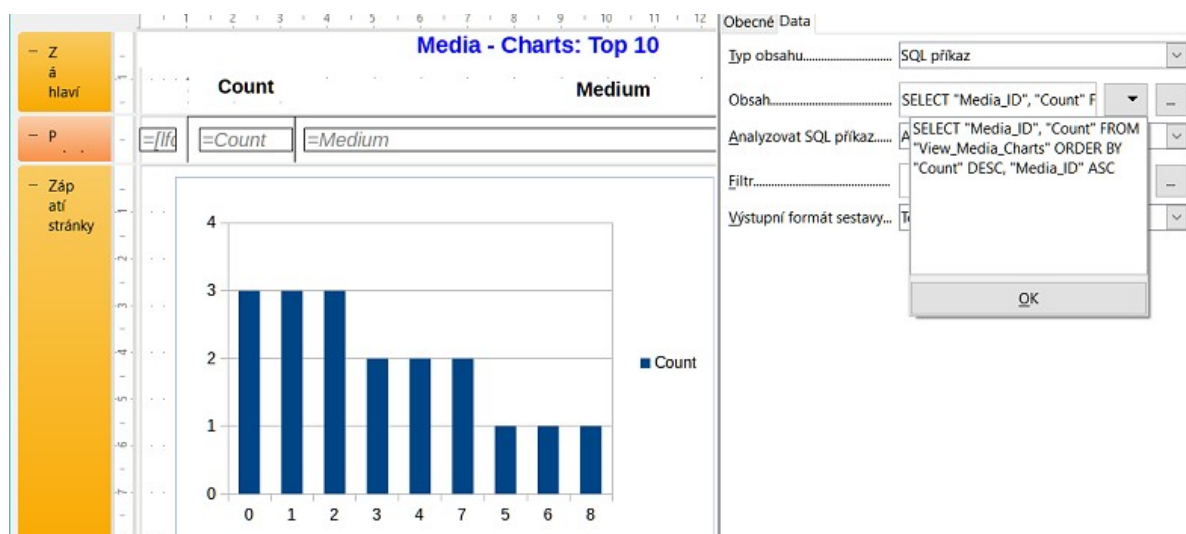
Grafy můžeme do sestavy vložit pomocí příslušného ovládacího prvku nebo pomocí **Vložit > Ovládací prvky sestavy > Graf**. Graf je jediným způsobem, jak reprodukovat data, která se nenacházejí ve zdroji dat zadaném pro sestavu. Graf lze tedy považovat za jakousi dílčí sestavu, ale také za samostatnou součást sestavy.



Obrázek 348: Výběr typu grafu na kartě Vlastnosti (karta Obecné)

Místo pro graf musíme nakreslit myší. V obecných vlastnostech (viz obrázek 348) můžeme kromě známých polí vybrat typ grafu (viz příslušné typy v Calc). Kromě toho můžeme nastavit maximální počet záznamů pro náhled, který poskytne představu o tom, jak bude graf nakonec vypadat.

Grafy lze formátovat stejným způsobem jako v aplikaci Calc (poklepeme na graf). Další informace nalezneme v popisu v příručce *Průvodce programem LibreOffice Calc*.



Obrázek 349: Použití příkazu SQL k vyplnění grafu zobrazujícího četnost výpůjček médií

Graf je propojen v části Data s potřebnými datovými poli. Příklad seznamu Media Top 10 na obrázku 349 grafu ukazuje četnost výpůjček jednotlivých médií. Editor dotazů slouží k vytvoření vhodného příkazu SQL, stejně jako v případě pole se seznamem ve formuláři. První sloupec v dotazu se použije k označení svislých sloupců v grafu, zatímco druhý sloupec poskytuje celkový počet výpůjčních transakcí, který je uveden ve výšce sloupců.

Ve výše uvedeném příkladu graf zpočátku ukazuje jen velmi málo, protože před vydáním příkazu SQL byly provedeny pouze omezené testovací výpůjčky.

Obrázek 350: Naplnění grafu pomocí dotazu

Graf lze vyplnit pomocí dotazu ve Vlastnostech dat, jak je znázorněno na obrázku 350. Tento graf z databáze Example_sport.odt ukazuje kromě základů tvorby grafů v sestavách i něco zvláštního: náhled grafu zobrazuje více sloupců, než se předpokládalo. To vyplývá z obsahu dotazu, který vytváří další sloupce, které se všechny nezobrazí v samotném grafu.

✓ Poznámka

Databáze Example_sport.odt je součástí balíčku ukázkových databází pro tuto příručku.

Filtrování a třídění pomocí interních nástrojů nástroje Návrhář sestav není nutné, protože to již bylo v rámci možností provedeno v dotazu.

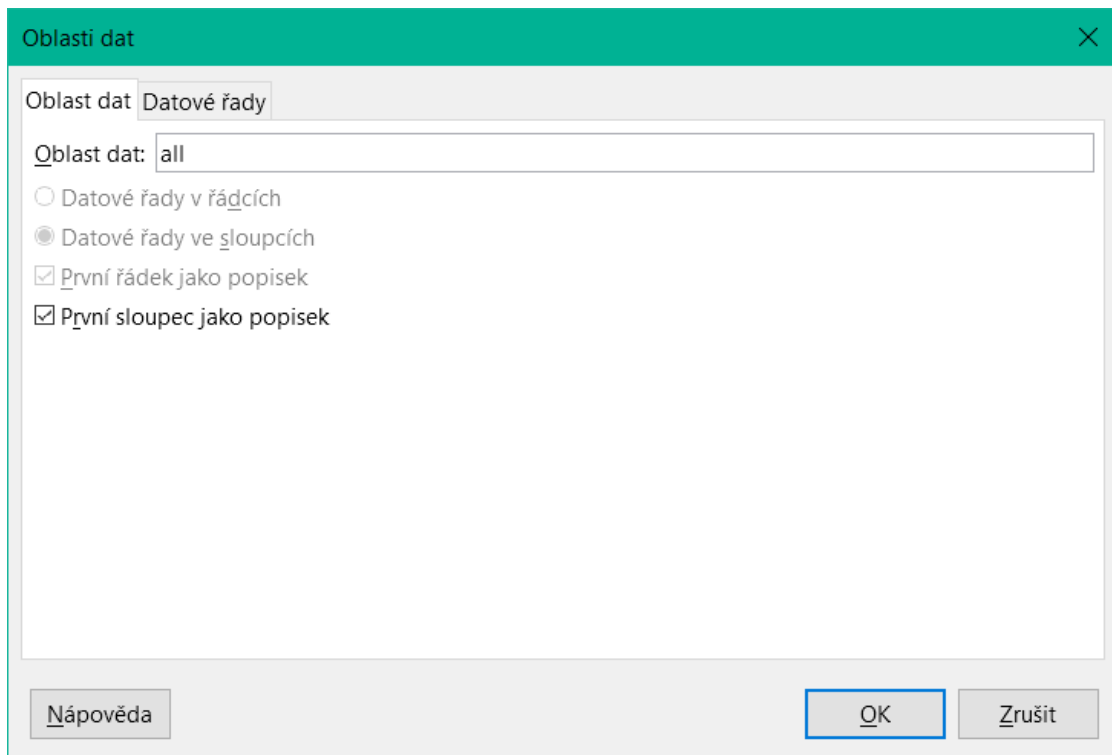
i Tip

V zásadě chceme z vytváření sestav odstranit co nejvíce úkolů. Cokoli lze zvládnout na začátku procesu pomocí dotazů, není třeba znovu provádět během relativně pomalého procesu vytváření samotné sestavy.

Stejně jako u hlavních formulářů a podformulářů jsou nyní některá pole propojena. V aktuální zprávě jsou věkové skupiny účastníků a účastnic sportovních táborů uvedeny v tabulce. Jsou rozděleny do skupin podle pohlaví. V každé skupině je nyní samostatný graf. Aby bylo zajištěno, že graf používá pouze údaje pro správné pohlaví, jsou obě pole nazvaná „Gender“ – ve zprávě a v grafu – propojena.

Osa X grafu je automaticky propojena s prvním sloupcem v tabulce zdroje dat. Pokud jsou v tabulce více než dva sloupce, do grafu se automaticky vloží další sloupce. Další nastavení grafu jsou dostupná po výběru celého grafu dvojitým klepnutím. Klepnutím pravým tlačítkem myši

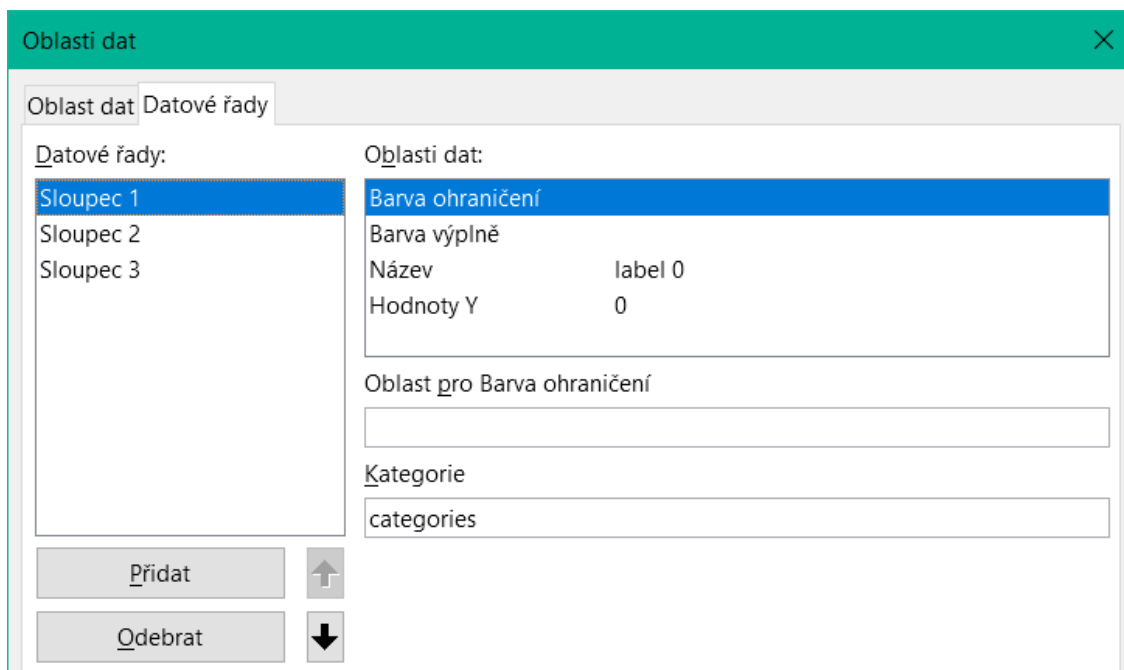
se nad diagramem otevře místní nabídka, jejíž obsah závisí na tom, který prvek byl vybrán. Obsahuje možná nastavení pro datové rozsahy uvedené na obrázku 351:



Obrázek 351: Dialogové okno Oblasti dat (karta Oblasti dat)

Volba *Datové řady ve sloupcích* je šedá, a proto ji nelze změnit. Nelze změnit ani zaškrtačací políčko *První řádek jako popis*. Zbývající nastavení na kartě Oblasti dat by se neměla měnit, protože zde existuje více možností, než kolik jich sestava Návrhář sestav skutečně zvládne.

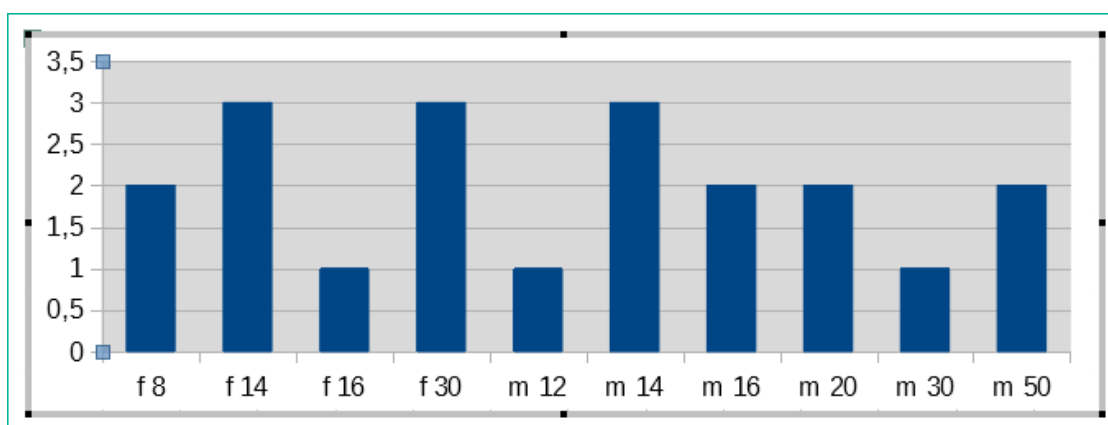
Karta *Datové řady* naopak skrývá několik nastavení, která mohou výrazně změnit výchozí vzhled grafu. Zobrazí všechny datové řady, které jsou k dispozici pro první sloupec dotazu. Všechny, které nechceme zobrazit, můžeme v tomto okamžiku odstranit.



Obrázek 352: Dialogové okno Oblasti dat (karta Datové řady)

V příkladu na obrázku 352 bylo v grafu viditelných příliš mnoho sloupců. To vyžaduje zlepšení! Gender ani age_group_sort, jejichž názvy pocházejí z podkladového dotazu, zde nejsou k ničemu. Řada pohlaví (gender) slouží k provázání grafu se zdrojem dat sestavy a v žádném případě ji nelze znázornit číselně. A age_group_sort slouží pouze k zajištění správného seřazení hodnot v dotazu, protože jinak by kód jako „m8“ přišel bezprostředně před „m80“ místo na začátek (řazení v textových polích často vede k podobným nežádoucím výsledkům).

Po odstranění všech řádků před řádkem Celkem vypadá náhled grafu tak, jak je znázorněno na obrázku 353:



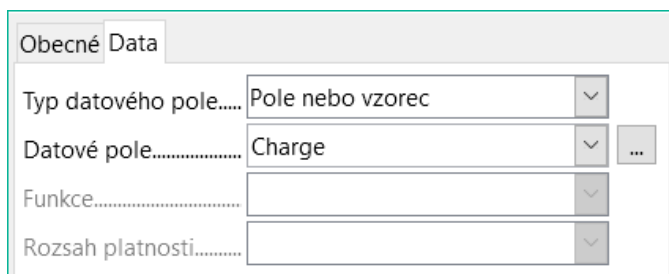
Obrázek 353: Náhled grafu zobrazujícího počet účastníků soutěže pro každou kombinaci pohlaví / věková skupina

Tento náhled zobrazuje 10 sloupců – prvních deset sloupců dotazu. Při skutečném spuštění se zobrazí pouze ty sloupce, které patří ke správnému pohlaví: sloupce „m“ pro muže a „f“ pro ženy.

Osa Y stále vykazuje nešťastnou vlastnost. Koneckonců neexistuje poloviční člověk! I to by se dalo zlepšit. Při automatickém spuštění s tímto nastavením se však tato čísla změní na celá čísla, pokud se rozsah hodnot nezastaví, jako ve výše uvedeném příkladu, na hodnotě „3“. Pokud by se toto automatické zpracování vypnulo, bylo by skutečně nutné provést určité ruční vylepšení.

Všechna další nastavení jsou podobná těm, která Calc používá pro vytváření grafů.

Datové vlastnosti polí



Obrázek 354: Vlastnosti (záložka Data) pro pole Charge

V dialogovém okně Vlastnosti je na kartě Data na obrázku 354 ve výchozím nastavení zobrazeno pouze pole databáze, ze kterého budou načtena data pro toto pole sestavy. Kromě typů polí Pole nebo vzorec jsou však k dispozici také typy Funkce, Počítadlo a Uživatelsky definovaná funkce.

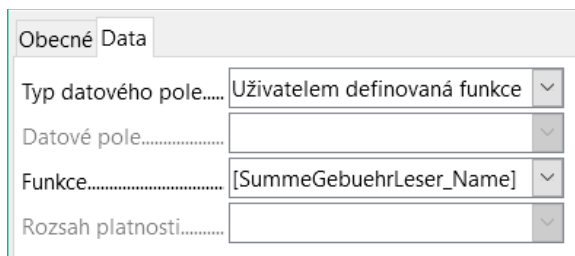
Můžeme předem vybrat funkce Nárůst, Minimum a Maximum. Budou se vztahovat buď na aktuální skupinu, nebo na celou sestavu. Tyto funkce mohou vést k problémům, pokud je pole prázdné (NULL). Pokud jsou tato pole formátována jako čísla, zobrazí se NaN, tj. není v nich uvedena žádná číselná hodnota. U prázdných polí se neprovádí žádný výpočet a výsledkem je vždy 0.

Taková pole lze přeformátovat tak, aby zobrazovala hodnotu 0, a to pomocí následujícího vzorce v oblasti Data zobrazení.

```
IF([numericfield];[numericfield];0)
```

Tato funkce počítá se skutečnou hodnotou pole, které nemá žádnou hodnotu. Zdá se, že by bylo jednodušší formulovat základní dotaz pro sestavu tak, aby se u číselných polí místo NULL uváděla 0.

Počítadlo počítá pouze záznamy, které se vyskytnou buď ve skupině, nebo v celé sestavě. Pokud je počítadlo vloženo do oblasti Podrobnosti, bude každý záznam opatřen průběžným číslem. Číslování se použije pouze pro záznamy ve skupině nebo v celé sestavě.



Obrázek 355: Vlastnosti (karta Data) s typem datového pole nastaveným na hodnotu Uživatelem definovaná funkce

Nakonec je k dispozici podrobná funkce definovaná uživatelem, jak je znázorněno na obrázku 355. Může se stát, že nástroj Návrhář sestav sám zvolí tuto variantu, pokud byl požadován výpočet, ale z nějakého důvodu nedokáže správně interpretovat zdroj dat.

Funkce v nástroji Návrhář sestav

Nástroj Návrhář sestav nabízí řadu funkcí pro zobrazení dat i pro nastavení podmínek. Pokud tyto funkce nestačí, lze pomocí jednoduchých výpočetních kroků vytvořit uživatelsky definované funkce, které jsou užitečné zejména v zápatí skupin a v souhrnech.

Zadávání vzorců

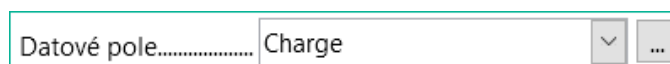
Nástroj Návrhář sestav je založen na nástroji Pentaho Report Builder. Malá část jeho dokumentace je na adrese <http://wiki.pentaho.com/display/Reporting/9.+Report+Designer+Formula+Expressions>.

Dalším zdrojem jsou specifikace pro standard OpenFormula: <http://www.oasis-open.org/committees/download.php/16826/openformula-spec-20060221.html> se základními zásadami pro zadávání vzorců shrnutými v tabulce 10.

Tabulka 10: Základní zásady pro zadávání vzorců

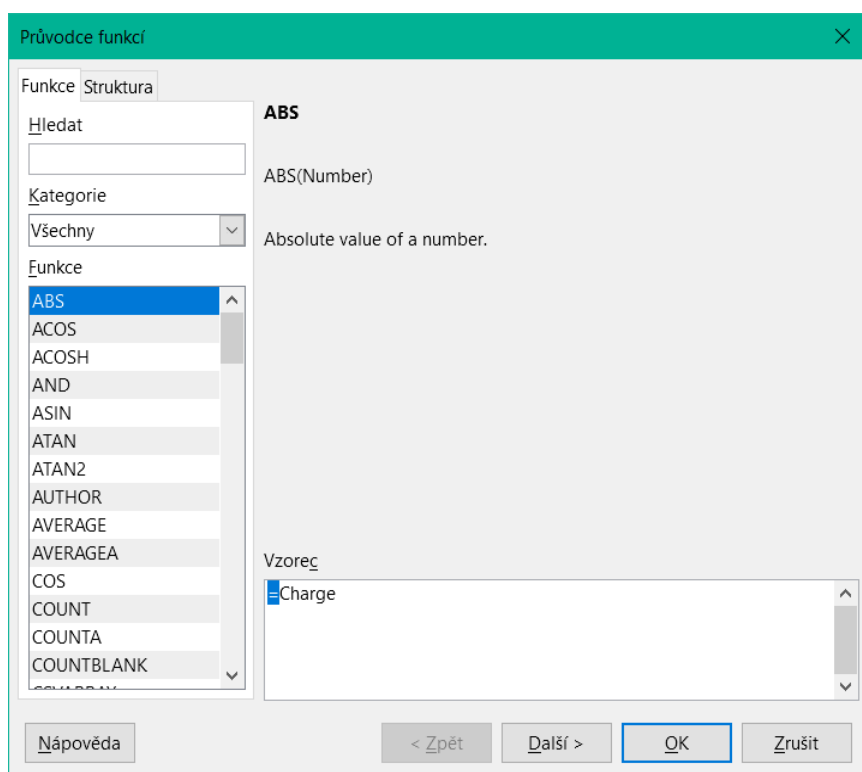
Princip	Detail
Vzorci začínají znaménkem rovnítka.	=
Odkazy na datová pole se uvádějí v hranatých závorkách.	[Název pole]
Pokud datová pole obsahují speciální znaky (včetně mezer), musí být název pole rovněž uzavřen v uvozovkách.	["Tento název pole by měl být v uvozovkách"]
Zadávání textu musí být vždy v dvojitých uvozovkách.	"Zadaný textu"
Povoleny jsou následující operátory.	+, -, * (násobení), / (dělení), % (předchozí číslo vydělíme 100), ^ (Zvýšení na mocninu následujícího čísla), & (zřetězení textu),
Možné jsou následující relace.	=, <>, <, <=, >, >=
Kulaté závorky jsou povoleny.	()
Výchozí chybová zpráva.	NA (není k dispozici)
Chybová zpráva pro prázdné pole, které bylo definováno jako číslo.	NaN (Možná "not a number" ("není číslo", pozn. překl.?)

Všechny zadané vzorce se vztahují pouze k aktuálnímu záznamu. Relace s předchozími nebo následujícími záznamy proto nejsou možné.



Obrázek 356: Klepnutím na tlačítko elipsy vedle pole Data přejdeme do Průvodce funkcí.

Vedle datového pole je tlačítko se třemi tečkami, kdykoli lze zadat vzorec, jak je znázorněno na obrázku 356. Tímto tlačítkem se spustí Průvodce funkcí zobrazený na obrázku 357.



Obrázek 357: Dialogové okno Průvodce funkcí

Je zde mnohem méně funkcí než v Calcu, i když mnoho funkcí má ekvivalenty z Calcu. Zde Průvodce vypočítá výsledek funkce přímo.

Průvodce funkcemi nefunguje vždy dokonale. Například textové položky nejsou zabírány dvojitými uvozovkami. Při spuštění funkce se však zpracovávají pouze položky s dvojitými uvozovkami.

K dispozici jsou funkce uvedené v tabulce 11.

Tabulka 11: Funkce dostupné v Průvodci funkcí

Funkce	Popis
Funkce data a času	
DATE	Vytvoří platné datum z číselných hodnot roku, měsíce a dne.
DATEDIF (DAY MONTH YEAR)	Vrátí celkový počet let, měsíců nebo dnů mezi dvěma hodnotami data.
DATEVALUE	Převede americký údaj o datu v textové podobě (uvozovky) na hodnotu data. Vytvořenou americkou variantu data lze následně přeformátovat.
DAY	Vrátí den v měsíci pro zadané datum. DAY([Pole pro datum])
DAYS	Vrátí počet dní mezi dvěma daty.
HOUR	Vrátí hodiny zadaného času ve 24hodinovém formátu. HOUR([Pole pro datum a čas]) vypočítá hodiny v poli.
MINUTE	Vrací minuty data v interním číselném formátu MINUTE([Pole pro čas]) vypočítá minutovou část času.
MONTH	Vrátí měsíc pro zadané datum jako číslo. MONTH([Pole pro datum])
NOW	Vrátí aktuální datum a čas.

Funkce	Popis
SECOND	Vrací sekundy data v interním číselném formátu SECOND(NOW()) zobrazuje sekundovou část času, kdy je příkaz proveden.
TIME	Zobrazuje aktuální čas.
TIMEVALUE	Převede textový údaj o čase na hodnotu času pro výpočty.
TODAY	Zobrazuje aktuální datum.
WEEKDAY	Vrátí den v týdnu jako číslo. Den číslo 1 je neděle.
YEAR	Vrátí část roku v položce data.
Logické funkce	
AND	Výsledek je TRUE, pokud jsou všechny jeho argumenty TRUE.
FALSE	Definuje logickou hodnotu jako FALSE.
IF	Pokud je podmínka TRUE, pak tato hodnota, jinak jiná hodnota.
IFNA	
NOT	Obrátí logickou hodnotu argumentu.
OR	Výsledek je TRUE, pokud je jedna z jeho podmínek TRUE.
TRUE	Definuje logickou hodnotu jako TRUE.
XOR	Výsledek je TRUE, pokud je pouze jedna z propojených hodnot TRUE.
Funkce zaokrouhlování	
INT	Zaokrouhluje směrem dolů na předchozí celé číslo.
Matematické funkce	
ABS	Vrací absolutní (nezápornou) hodnotu čísla.
ACOS	Vypočítá arcsin čísla. – argumenty mezi –1 a 1.
ACOSH	Vypočítá areakosinus (inverzní hyperbolický kosinus) – argument ≥ 1 .
ASIN	Vypočítá arcsin čísla – argument mezi –1 a 1.
ATAN	Vypočítá arkustangens čísla.
ATAN2	Vypočítá arkustangens souřadnice x a souřadnice y.
AVERAGE	Uvádí průměr zadaných hodnot.
AVERAGEA	Uvádí průměr zadaných hodnot. Text je považován za nulu.
COS	Argumentem je úhel v radiánech, jehož kosinus se má vypočítat.
EVEN	Zaokrouhluje kladné číslo nahoru nebo záporné číslo dolů na nejbližší sudé celé číslo.
EXP	Vypočítá exponenciální funkci (základ „e“).
LN	Vypočítá přirozený logaritmus čísla.
LibreOfficeG10	Vypočítá logaritmus čísla (se základem '10').
MAX	Vrátí maximum z řady čísel.

Funkce	Popis
MAXA	Vrací maximální hodnotu v řádku. Jakýkoli text je nastaven na hodnotu nula.
MIN	Vrací nejmenší z řady hodnot.
MINA	Vrací minimální hodnotu v řádku. Jakýkoli text je nastaven na hodnotu nula.
MOD	Vrátí zbytek pro dělení po zadání dividendy a dělitele.
ODD	Zaokrouhuje kladné číslo nahoru nebo záporné číslo dolů na nejbližší liché celé číslo.
PI	Udává hodnotu čísla 'π'.
POWER	Zvyšuje základ na mocninu exponentu.
SIN	Vypočítá sinus čísla.
SQRT	Vypočítá druhou odmocninu čísla.
SUM	Sčítá seznam číselných hodnot.
SUMA	Sčítá seznam číselných hodnot. Povolena jsou textová pole a pole Ano/Ne. Tato funkce bohužel končí chybovou zprávou.
VAR	Vypočítá rozptyl, počínaje vzorkem.
Textové funkce	
EXACT	Zobrazí, zda se dva textové řetězce přesně rovnají.
FIND	Udává odsazení textového řetězce v rámci jiného řetězce.
LEFT	Zadaný počet znaků textového řetězce se reprodukuje zleva.
LEN	Udává počet znaků v textovém řetězci.
LOWER	Převéde text na malá písmena.
MESSAGE	Zformátuje hodnotu do zadaného výstupního formátu.
MID	Zadaný počet znaků textového řetězce se reprodukuje od zadané pozice znaku.
REPLACE	Nahradí podřetězec jiným podřetězcem. Musí být zadána počáteční pozice a délka nahrazovaného podřetězce.
REPT	Opakuje text tolikrát, kolikrát je zadáno v argumentu funkce.
RIGHT	Zadaný počet znaků textového řetězce se reprodukuje od pravého okraje.
SUBSTITUTE	Nahradí určité části zadaného textového řetězce novým textem. Navíc můžeme určit, který z několika výskytů cílového řetězce má být nahrazen.
T	Vrací text nebo prázdný textový řetězec, pokud hodnota není text (například číslo).
TEXT	Převod čísel nebo časů na text.
TRIM	Odstraňuje počáteční a koncové mezery a redukuje více mezer na jednu.
UNICHAR	Převéde desetinné číslo Unicode na znak Unicode. Například 196 se změní na „Ä“ („Ä“ má hexadecimální hodnotu 00C4, což je 196 v desetinných číslech bez počátečních nul).
UNICODE	Převéde znak Unicode na desetinné číslo Unicode. 'Ä' se změní na 196.

Funkce	Popis
UPPER	Vrací textový řetězec napsaný velkými písmeny.
URLENCODE	Převede zadaný text na text, který odpovídá platné adrese URL. Pokud není zadán žádný konkrétní standard, postupuje se podle ISO-8859-1.
Informační funkce	
CHOOSE	Prvním argumentem je index, za kterým následuje seznam hodnot. Vrací se hodnota reprezentovaná indexem. CHOOSE(2; "Apple"; "Pear"; "Banana") returns Pear. CHOOSE([age_level_field]; "Milk"; "Cola"; "Beer") vrátí možný nápoj pro zadanou hodnotu 'age_level_field'.
COUNT	Započítávají se pouze pole obsahující číslo nebo datum. COUNT([time]; [number]) vrací 2, pokud obě pole obsahují hodnotu (non-NULL), jinak 1 nebo 0.
COUNTA	Zahrnuje také pole obsahující text. Započítává se i NULL a logická pole.
COUNTBLANK	Spočítá prázdná pole v oblasti.
HASCHANGED	Zkontroluje, zda se pojmenovaný sloupec změnil. O sloupci však nejsou uvedeny žádné informace.
INDEX	Pracuje s regiony
ISBLANK	Testuje, zda má pole hodnotu NULL (je prázdné).
ISERR	Vrací hodnotu TRUE, pokud má záznam jinou chybu než NA. ISERR(1/0) vrací TRUE.
ISERROR	Stejně jako ISERR s tím rozdílem, že NA také vrací TRUE.
ISEVEN	Testuje, zda je číslo sudé.
ISLOGICAL (ISTLOG)	Testuje, zda se jedná o hodnotu Ano/Ne. ISLOGICAL(TRUE()) or ISLOGICAL(FALSE()) vrací hodnotu TRUE, Textové hodnoty jako např. ISLOGICAL("TRUE") vrací hodnotu FALSE.
ISNA	Testuje, zda je výraz chybou typu NA.
ISNONTEXT	Testuje, zda hodnota není text.
ISNUMBER	Testuje, zda je něco číselné. ISNUMBER(1) dává výsledek TRUE, ISNUMBER("1") dává výsledek FALSE.
ISODD	Testuje, zda je číslo liché.
ISREF	Testuje, zda je něco odkazem na pole. ISREF([Názevpo le]) dává hodnotu TRUE, ISREF(1) dává hodnotu FALSE.
ISTEXT	Testuje, zda je obsahem pole text.
NA (NV)	Vrací kód chyby NA.
VALUE	
Definováno uživatelem	
CSVARRAY	Převede text CSV na pole.

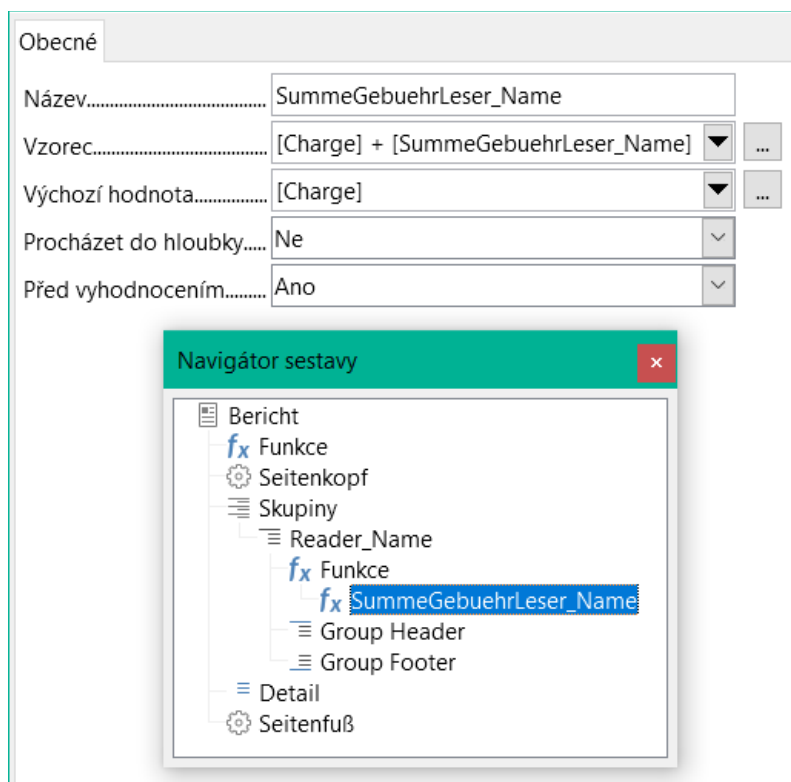
Funkce	Popis
CSVTEXT	Převede pole na text CSV.
NORMALIZEARRAY	
NULL	Vrací hodnotu NULL.
PARSEDATE	Převede text na datum. Používá formát SimpleDateFormat. Vyžaduje datum v textu, jak je popsáno v tomto formátu data. Příklad: PARSEDATE("9.10.2012" ; "dd.MM.rrrr") dává interně použitelné číslo pro datum.
Informace o dokumentu	
AUTHOR	Autor, jak je načten z nastavení Nástroje > Možnosti > LibreOffice > Uživatelské údaje . Nejedná se tedy o skutečného autora, ale o aktuálního uživatele databáze.
TITLE	Vrací název sestavy.

Uživatelsky definované funkce

Pomocí uživatelsky definovaných funkcí můžeme vracet konkrétní mezivýsledky pro skupinu záznamů. V našem příkladu byla taková funkce použita k výpočtu pokut v oblasti Reader_Name_Footer.

V Navigátoru sestavy se funkce zobrazí ve skupině Reader_Name. Klepnutím pravým tlačítkem myši na tuto funkci můžeme definovat další funkce podle názvu (viz obrázek 358).

Vlastnosti funkce SummeGebuehrLeser_Name jsou uvedeny výše. Vzorec přidá pole Charge k hodnotě již uložené v samotné funkci. Počáteční hodnota je hodnota pole Charge při prvním průchodu skupinou. Tato hodnota je uložena ve funkci pod názvem funkce a je znovu použita ve vzorci, dokud není ukončena smyčka a zapsána patička skupiny.



Obrázek 358: Výběrem uživatelsky definované funkce v navigátoru sestavy získáme přístup k jejím vlastnostem.

Zdá se, že procházení do hloubky zatím nemá žádnou funkci, pokud se zde grafy nepovažují za dílčí sestavy.

Pokud je pro funkci aktivováno předběžné vyhodnocení, může být výsledek umístěn také v záhlaví skupiny. Jinak záhlaví skupiny obsahuje pouze odpovídající hodnotu prvního pole skupiny.

Uživatelsky definované funkce mohou také odkazovat na jiné uživatelsky definované funkce. V takovém případě je třeba zajistit, aby použité funkce již byly vytvořeny. Předběžný výpočet ve funkcích, které odkazují na jiné funkce, musí být vyloučen.

```
[SumMarksClass] / ([ClassNumber]+1)
```

Vztahuje se ke skupině Class. Obsah pole Marks se sečte a vrátí se součet za všechny záznamy. Součet známek se vydělí součtem záznamů. Abychom získali správné číslo, je třeba přičíst 1, jak je uvedeno v položce [ClassNumber]. Z toho pak vyplynou průměrné známky.

Zadání vzorce pro pole

Pomocí **Data > Datové pole** můžeme zadávat vzorce, které ovlivňují pouze jedno pole v oblasti Podrobnosti.

```
IF([boolean_field];"Yes";"No")
```

nastaví přípustné hodnoty na "Yes" nebo "No" namísto TRUE a FALSE.

Může se stát, že se v poli se vstupním vzorcem objeví pouze jedno číslo. V textových polích je to nula. Chceme-li to napravit, musíme změnit textové pole z výchozího „Number“ na „Text“.

Podmíněný tisk



Obrázek 359: Pole podmíněného tiskového výrazu na kartě Obecné v okně Vlastnosti

Obecné vlastnosti záhlaví skupin, zápatí skupin a polí zahrnují pole Podmíněný tiskový výraz (viz obrázek 359). Vzorce zapsané v tomto poli ovlivňují obsah pole nebo zobrazení celé oblasti. I zde můžeme využít Průvodce funkcí.

```
[Názevpo le]="true"
```

způsobí, že se obsah pojmenovaného pole zobrazí pouze v případě, že je true.

✓ Poznámka

Viz také databáze `Example_Report_conditional_Overlay_Graphics.odt`, která je součástí ukázkových databází k této knize.

Mnoho forem podmíněného zobrazení není plně určeno zadanými vlastnostmi. Pokud má být například za desáté místo seznamu výsledků soutěže vložena grafická oddělovací čára, nelze pro grafické zobrazení jednoduše použít následující příkaz podmíněného zobrazení:

```
[Place]=10
```

Tento příkaz nefunguje. Namísto toho se grafika bude nadále zobrazovat v části Podrobnosti po každém dalším záznamu. Viz chyba [73707](#).

Pokud chceme v tomto místě pouze překrýt obdélníkový tvar, lze to provést pomocí grafického ovládacího prvku, kterému lze zadat adresu (černobílého) grafického souboru. V obecných vlastnostech tohoto ovládacího prvku by měla být vybrána možnost **Měřítko > Autom..** Pak bude grafika odpovídat formuláři a podmínka bude splněna.

Není-li to jinak nutné, je bezpečnější vázat podmíněné zobrazení na zápatí skupiny než na grafiku. Čára je umístěna v zápatí skupiny. Pak se čára skutečně objeví za 10. místem, pokud je formulována výše uvedeným způsobem. Protože je podmínka spojena se zápatím skupiny, řádek se zobrazí pouze tehdy, je-li to skutečně nutné. V opačném případě může podmíněné zobrazení vést k tomu, že se místo čáry objeví prázdné místo.

Zde můžeme také použít tvary poskytované **Vložit > Tvary > Standardní tvary**, například vodorovnou čáru, která se prolíná se zápatím skupiny.

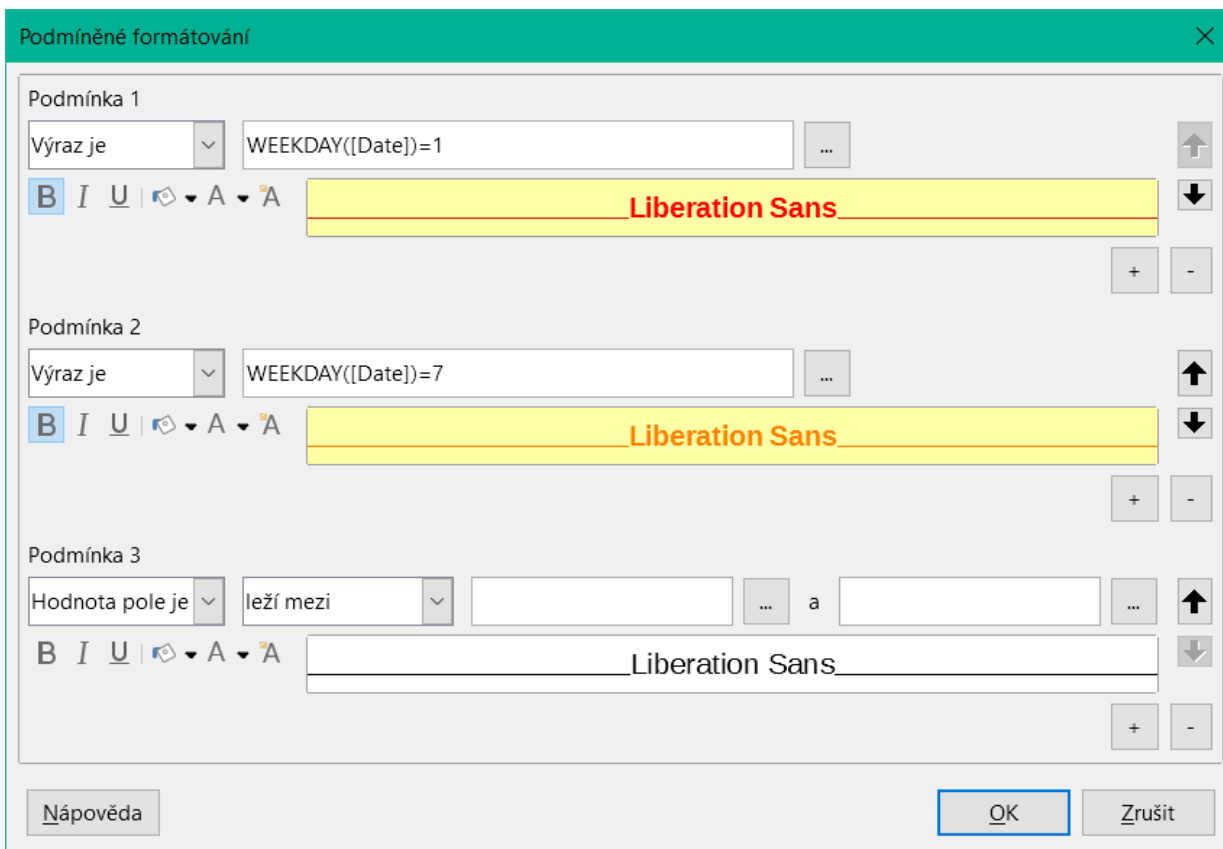
Podmíněné formátování

Podmíněné formátování lze použít například na obrázku 360 k formátování kalendáře tak, aby se víkendy zobrazovaly odlišně. Zvolíme **Formát > Podmíněné formátování** a zadáme:

```
WEEKDAY([Date])=1
```

a odpovídající formátování pro neděle.

Pokud v podmíněném formátování použijeme výraz „Výraz je“, můžeme zadat vzorec. Jak je v programu Calc obvyklé, lze formulovat několik podmínek, které se vyhodnocují postupně. Ve výše uvedeném příkladu se nejprve testuje neděle a poté sobota. Nakonec se může objevit dotaz na obsah pole. Například obsah „Holiday“ by vedl k jinému formátu.



Obrázek 360: Dialogové okno Podmíněné formátování

✓ Poznámka

Pokud se vyskytnou další neopravitelné chyby (neimplementované vzorce, příliš dlouhý text zobrazený jako prázdné pole apod.), je někdy nutné části sestavy odstranit nebo ji jednoduše vytvořit znovu.

Příklady sestav vytvořených pomocí nástroje Návrhář sestav

Použití nástroje Návrhář sestav může být obtížné, protože některé funkce se zdají být k dispozici, ale nefungují tak, jak mají. Systém nápovědy navíc v současné době tyto problémové oblasti neřeší. Z tohoto důvodu je zde uvedeno několik příkladů, které ukazují, jak lze nástroj Návrhář sestav použít pro různé typy sestav.

Tisk účtů

Vytvoření účtu vyžaduje následující úvahy:

- Jednotlivé položky musí být očíslovány.
- Účty, které vyžadují více než jednu stranu, musí být očíslovány.
- Účty, které vyžadují více než jednu stranu, by měly mít na každé straně průběžnou sumu, která se přenesse na další stranu.

Zdá se, že několik současných chyb to znemožňuje:

- **Bug 51452:** Je-li skupina nastavena na „Opakovat sekci“, vloží se před a za skupinu automaticky zlom stránky.
- **Bug 51453:** Skupiny s individuálním stránkováním jsou povoleny, ale ve skutečnosti nefungují.
- **Bug 51959:** Zápatí skupiny nelze opakovat. Může se objevit pouze na konci skupiny, nikoli na konci každé stránky. A pokud zvolíme možnost „Opakovat sekci“, zmizí úplně.

Kromě toho se objevují problémy s vkládáním čar do sestav. Vestavěné vodorovné a svislé čáry se objevují pouze ve verzích LibreOffice 4.0.5 a 4.1.1. Jako náhradu můžeme použít obdélníky, které však nelze správně umístit, pokud je v sekci zalomení stránky.

Zpráva v jednoduchém tvaru by měla vypadat jako na obrázku 361:

Quantity	Article	Price	Quantity*Price
2	paper, 500 sheet	\$4.23	\$8.46
1	rubber	\$0.75	\$0.75
4	sharpener	\$1.27	\$5.08
2	pencil HB	\$0.23	\$0.46
1	spiral-bound notepad	\$0.98	\$0.98
1	envelopes, 25 pcs.	\$1.25	\$1.25
4	CD-envelopes, 100 pcs.	\$1.89	\$7.56
1	wax crayons, 6 pcs.	\$3.85	\$3.85
1	folder, 2 inch width	\$1.89	\$1.89
2	clipboard	\$4.45	\$8.90
1	ring binder A4	\$5.76	\$5.76
1	CD-pens, 4 pcs.	\$4.56	\$4.56
2	CD-blanks, 50 pcs.	\$12.34	\$24.68
1	paper, recycling, 500 sheet	\$3.76	\$3.76
4	briecase with register	\$6.87	\$27.48
2	receipt book, 50 sheet	\$1.35	\$2.70
10	bill book, 50 Blatt	\$3.15	\$31.50
1	datestamp, simple	\$18.50	\$18.50
1	till, small	\$12.00	\$12.00
12	hanging file folder, 25 pcs.	\$14.75	\$177.00
2	universallabels 70x32, 2700 pcs.	\$20.50	\$41.00
1	universallabels smallpack 12x30, 700 pcs.	\$4.15	\$4.15
2	printerhead, black	\$24.00	\$48.00
1	printerhead, color	\$26.30	\$26.30
3	ink cartridge, black, 32ml	\$5.40	\$16.20
5	ink cartridge, color, 17ml	\$5.20	\$26.00
		Carryover:	\$508.77

Quantity	Article	Price	Quantity*Price
2	toner laserprinter black	\$65.89	\$131.78
1	toner laserprinter color	\$78.89	\$78.89
2	register for folders, A-Z	\$1.25	\$2.50
1	staples	\$0.85	\$0.85
2	file recycling	\$0.65	\$1.30
1	paper, recycling, 500 sheet, color	\$4.15	\$4.15
1	pencils, 10 pcs., div. thickness	\$4.85	\$4.85
2	ballpen	\$1.35	\$2.70
1	watercolors, 12 colors	\$8.75	\$8.75
1	aquarelle, 24 colors	\$17.15	\$17.15
2	aquarelle brush, 3 pcs., div. thickness	\$8.34	\$16.68
1	drawing pad, A3, 20 sheet	\$3.85	\$3.85
4	desk pad 20*30 inch	\$15.67	\$62.68
2	paper, div. colors, 20*30 inch	\$0.45	\$0.90
10	cardboard, div. colors, 20*30 inch	\$0.89	\$8.90
1	datestamp, simple	\$18.50	\$18.50
1	till, small	\$12.00	\$12.00
12	hanging file folder, 25 pcs.	\$14.75	\$177.00
2	universallabels 70x32, 2700 pcs.	\$20.50	\$41.00
1	universallabels smallpack 12x30, 700 pcs.	\$4.15	\$4.15
2	printerhead, black	\$24.00	\$48.00
1	printerhead, color	\$26.30	\$26.30
3	ink cartridge, black, 32ml	\$5.40	\$16.20
5	ink cartridge, color, 17ml	\$5.20	\$26.00
2	toner laserprinter black	\$65.89	\$131.78
1	toner laserprinter color	\$78.89	\$78.89
		Carryover:	\$1,434.52

Obrázek 361: Počáteční návrh zprávy o činnosti úřadu Officeshop

Aby bylo možné překonat výše popsaná omezení, je při tvorbě odpovídajícího účtu nutné přesně dbát na rozměry stránek ve výsledném tištěném dokumentu. Tento příklad na obrázku 362 vychází z formátu DIN-A4. Celková výška každé stránky je tedy 29,7 cm.

Sestavu je třeba rozdělit do několika skupin. Dvě skupiny se vztahují ke stejnému poli tabulky a obsahují stejný prvek tabulky.

Quantity	Article	Price	Quantity*Price
=quantity	=article	=price	=quantity*price
Carryover:			=TotalQuantity
e "&[CounterBillNum]			
"&([CounterBillNumbe			
Quantity	Article	Price	Quantity*Price
Carryover:			=TotalQuantity
Total:			=total

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE612345675678500000456

Obrázek 362: Úprava sestavy Officeshopu

Tabulka 12 ukazuje rozdělení stránky na jednotlivé části sestavy.

Tabulka 12: Rozdělení stránky do oddílů pro sestavu Officeshopu

Karta Umístění	Popis	Hodnota
A	Horní okraj	2,00 cm
B	Záhlaví stránky (zobrazuje se na každé stránce, neobsahuje žádný obsah databáze, pouze materiál, jako je logo společnosti nebo adresa dodavatele).	3,00 cm
C	Záhlaví skupiny pro účet (později by měly být přidány pouze položky, které patří k číslu účtu. Záhlaví skupiny se objevuje pouze na začátku účtu).	2,50 cm
D	Záhlaví skupiny pro jednotlivé položky. (Sekce Detail je vyžadována pro jiný účel. Proto zde vzniká skupina, která také třídí obsah po zadání. Tato skupina obsahuje pouze jednu hodnotu.)	0,70 cm
E	Záhlaví skupiny, rovněž vázané na položky. (Tato sekce se zobrazí pouze v případě, že se vyskytuje tolik položek, že je nutná další stránka. Obsahuje průběžný součet a číslo stránky v dolní části stránky. Po této části následuje přerušení stránky.)	2,00 cm
F	Sekce s podrobnostmi. (Tato sekce se zobrazuje pouze v případě, že se vyskytuje tolik položek, že je potřeba další stránka. Obsahuje přenesenou částku a číslo stránky v horní části stránky.)	2,50 cm

Karta Umístění	Popis	Hodnota
G	Zápatí skupiny pro číslo účtu. (Zde je uvedena celková částka za účet, případně s připočtenou DPH. Zápatí skupiny se objevuje pouze na poslední straně účtu.)	1,60 cm
H	Zápatí stránky (např. bankovní údaje)	1,00 cm
I	Okraj stránky	1,00 cm

Zalomení stránky následuje pouze v případě, že je položek příliš mnoho. Pro položky máme následující volná místa, jak je uvedeno v tabulce 13:

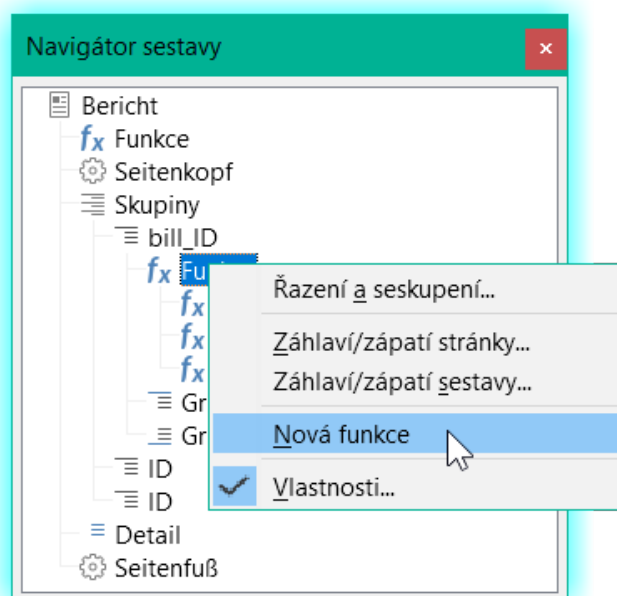
Tabulka 13: Volné místo pro položky v sestavě Officeshop

29,70 cm	(DIN A 4)
- 2,00 cm	(Poz. A)
- 3,00 cm	(Poz. B)
- 2,50 cm	(Poz. C)
- 1,60 cm	(Poz. G)
- 1,00 cm	(Poz. H)
- 1,00 cm	(Poz. I)
= 18,60 cm	

Zbývající volný prostor je tedy maximálně $18,60 \text{ cm} / 0,70 \text{ cm} = 26,57$. Zaokrouhlením nahoru získáme 26 řádků položek.

Jakmile se objeví 27. položka, musí okamžitě následovat zalomení stránky. To znamená, že se musí zobrazit záhlaví skupiny E a oddíl Podrobnosti. Potřebujeme tedy počítadlo položek (oddíl C). Jakmile toto počítadlo dosáhne hodnoty 27, zobrazí se sekce Podrobnosti (F).

Počítadlo položek je definováno takto:



Obrázek 363: Vytvoření nové funkce pomocí navigátoru sestavy

Pomocí nástroje Navigátor sestavy vyhledáme skupinu Bill_ID (viz obrázek 363). Naše nová funkce se jmenuje CounterBillNumber. Vzorec je [CounterBillNumber] + 1. Počáteční hodnota je 1. Žádné podsestavy nejsou vázány (taková funkce neexistuje). Není ani předem vypočtena. Pro pokročilé výpočty používáme samostatné počítadlo CounterTotal.

Záhlaví skupiny E a sekce Podrobnosti F se zobrazí, pokud je na účtu celkem více než 26 položek a aktuální pozice záznamu dosáhla 26. Výraz pro podmíněné zobrazení je tedy pro obě varianty stejný:

```
AND([CounterBillNumber]=26; [CounterTotal]>26)
```

Obsah této sekce se proto zobrazí pouze v případě, že se očekává alespoň 27 položek. Na první straně se zobrazí záhlaví skupiny E. Následuje přerušení stránky a obsah oddílu Podrobnosti se zobrazí na další stránce.

Nyní musíme vypočítat počet oddílů, které se musí objevit na první stránce, jak je uvedeno v tabulce 14.

Tabulka 14: Výpočet počtu oddílů na první stránce

	29,70 cm	(DIN A 4)
-	2,00 cm	(Poz. A)
-	3,00 cm	(Poz. B)
-	2,50 cm	(Poz. C)
-	18,20 cm	(Poz. D * 26)
-	1,00 cm	(Poz. H)
-	1,00 cm	(Poz. I)
=	2,00 cm	

Zápatí skupiny není pro první stránku povinné. Celkem je k dispozici 26 řádků položek. Záhlaví skupiny E tak může na první straně zabírat maximálně 2 cm. Do těchto 2 cm se musí vejít průběžný součet a číslo stránky. Aby byl za všech okolností zajištěn správný zlom stránky, měla by být tato část ve skutečnosti o něco menší než 2 cm. V příkladu použijeme 1,90 cm.

V horní části další stránky se nachází sekce Podrobnosti. Protože se na této stránce nevyskytuje záhlaví skupiny položek (C), může sekce Podrobnosti zabírat stejně místa jako záhlaví, tedy 2,50 cm. Poté začneme další sérii položek se stejným uspořádáním jako na předchozí straně.

Přenesená částka se vypočítá prostým sečtením předchozích položek.

K vyhledání skupiny Bill_number se používá nástroj Navigátor sestavy. Nově vytvořená funkce se bude jmenovat TotalPrice. Vzorec je [Price] + [TotalPrice]. Počáteční hodnota je [Price]. Žádné podsestavy nejsou vázány. Ani tento údaj není třeba předem vypočítávat.

Přenášená částka se zobrazuje jak v záhlaví skupiny E, tak v sekci Podrobnosti. V záhlaví skupiny je hned na začátku stránky. Na první stránce se objeví v dolní části. V sekci Podrobnosti je součet uveden úplně dole. Je uveden na straně 2 přímo pod záhlavím tabulky.

Dotaz pro určení čísla stránky je podobný dotazu pro určení zobrazení záhlaví skupiny a části s podrobnostmi:

```
IF([CounterBillNumber]=26; "Page 1"; "")
```

Tímto způsobem získáme číslo první stránky. Další podmínky IF lze použít pro ostatní stránky.

Stejným způsobem lze snadno nastavit číslo stránky pro následující stránku na „Page 2“.

Pokud vzorce pokračují stejným způsobem, mohou pokrýt libovolný počet stran.

Výraz pro podmíněné zobrazení se mění z

```
AND([CounterBillNumber]=26; [CounterBillComplete]>26)
```

na

```
AND(MOD([CounterBillNumber];26)=0;  
[CounterBillComplete]>[CounterBillNumber])
```

Záhlaví skupiny E a sekce Podrobnosti F se zobrazí pouze tehdy, když vydělením počítadla položek číslem 26 nevznikne žádný zbytek a celkový počet položek je větší než počítadlo položek.

Výraz pro číslo stránky se změní z

```
IF([CounterBillNumber]=26; "Page 1"; "")
```

na

```
"Pag "&[CounterBillNumber]/26
```

pro aktuální stránku a

```
"Page "&([CounterBillNumber]/26)+1
```

pro následující stránku.

Sestava zobrazená na obrázku 364 při použití těchto nastavení ještě není zcela připravena k použití:

Officeshop
Norderstr. 17, 43219 Phantastica

Officeshop - Norderstr. 17 - 43219 Phantastica
Mr.
Marko Patternman
Palace Avenue 42
06741 Behind the Mountain

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
2	paper, 500 sheet	\$4.23	\$8.46
1	rubber	\$0.75	\$0.75
4	sharpener	\$1.27	\$5.08
2	pencil HB	\$0.23	\$0.46
1	spiral-bound notepad	\$0.98	\$0.98
1	envelopes, 25 pcs.	\$1.25	\$1.25
4	CD-envelopes, 100 pcs.	\$1.89	\$7.56
1	wax crayons, 6 pcs.	\$3.85	\$3.85
1	folder, 2 inch width	\$1.89	\$1.89
2	clipboard	\$4.45	\$8.90
1	ring binder A4	\$5.76	\$5.76
1	CD-pens, 4 pcs.	\$4.56	\$4.56
2	CD-blanks, 50 pcs.	\$12.34	\$24.68
1	paper, recycling, 500 sheet	\$3.76	\$3.76
4	briefcase with register	\$6.87	\$27.48
2	receipt book, 50 sheet	\$1.35	\$2.70
10	bill book, 50 Blatt	\$3.15	\$31.50
1	datestamp, simple	\$18.50	\$18.50
1	till, small	\$12.00	\$12.00
12	hanging file folder, 25 pcs.	\$14.75	\$177.00
	Carryover:		\$347.12

Page 1

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567850000456

Officeshop
Norderstr. 17, 43219 Phantastica

Page 2

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
	Carryover:		\$347.12
2	universallabels 70x32, 2700 pcs.	\$20.50	\$41.00
1	universallabels smallpack 12x30, 700 pcs.	\$4.15	\$4.15
2	printerhead, black	\$24.00	\$48.00
1	printerhead, color	\$26.30	\$26.30
3	ink cartridge, black, 32ml	\$5.40	\$16.20
5	ink cartridge, color, 17ml	\$5.20	\$26.00
2	toner laserprinter black	\$65.89	\$131.78
1	toner laserprinter color	\$78.89	\$78.89
2	register for folders, A-Z	\$1.25	\$2.50
1	staples	\$0.85	\$0.85
2	file recycling	\$0.65	\$1.30
1	paper, recycling, 500 sheet, color	\$4.15	\$4.15
1	pencils, 10 pcs., div. thickness	\$4.85	\$4.85
2	ballpen	\$1.35	\$2.70
1	watercolors, 12 colors	\$8.75	\$8.75
1	aquarelle, 24 colors	\$17.15	\$17.15
2	aquarelle brush, 3 pcs., div. thickness	\$8.34	\$16.68
1	drawing pad, A3, 20 sheet	\$3.85	\$3.85
4	desk pad 20*30 inch	\$15.67	\$62.68
2	paper, div. colors, 20*30 inch	\$0.45	\$0.90
10	cardboard, div. colors, 20*30 inch	\$0.89	\$8.90
1	datestamp, simple	\$18.50	\$18.50
1	till, small	\$12.00	\$12.00
12	hanging file folder, 25 pcs.	\$14.75	\$177.00
	Carryover:		\$1,062.20

Page 2

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE61234567567850000456

Obrázek 364: Revidovaný návrh sestavy OfficeShop

Kvůli adresnímu poli je na první straně účtu méně položek než na druhé straně. Sekce Podrobnosti vpravo nahoře na druhé straně je proto výrazně menší než záhlaví skupiny účtu (C). Aby bylo možné na prvních dvou stranách zohlednit různé počty položek, je třeba upravit vzorce. Následující výpočet zajistí správné zobrazení příslušných sekcí.


```
AND(MOD([CounterBillNumber]-20;24)=0;  
[CounterBillComplete]>[CounterBillNumber])
```

Od počítadla položek odečteme počet položek na první stránce. Tento rozdíl se vydělí možným celkovým počtem položek na druhé stránce. Pokud je dělení přesné beze zbytku, je splněna první podmínka pro zobrazení záhlaví skupiny E a sekce Podrobnosti F. Kromě toho, jak bylo uvedeno výše, musí být aktuální hodnota počítadla položek menší než očekávaná celková hodnota. V opačném případě by na aktuální stránce bylo místo pro vypočtený celkový součet.

Možný celkový počet položek na druhé stránce je nyní menší, protože tato stránka nyní obsahuje číslo účtu a datum.

Číslo stránky lze nyní vypočítat jednodušeji:

```
"Page "&INT([CounterBillNumber]/24)+1
```

Funkce INT zaokrouhluje dolů na nejbližší celé číslo. První stránka obsahuje maximálně 20 položek. Výsledkem dělení je tedy hodnota <1 pro tuto první stránku. Tato hodnota se zaokrouhluje směrem dolů na nulu. Proto musíme k vypočtenému číslu stránky přidat 1, aby se 1 objevilo na první stránce. Podobně je tomu na straně 2.

Sestava stále vykazuje estetickou chybu. Při pečlivém pohledu na položky účtu zjistíme, že tři spodní položky na stránkách jsou stejné. Je to proto, že záznamy byly jednoduše zkopírovány. Ve skutečnosti se nejedná o stejné záznamy, ale o různé záznamy pro stejný produkt, které byly zpracovány nezávisle na sobě. Bylo by lepší seskupit je podle typu výrobku, aby se každý výrobek zobrazil pouze jednou s celkovým počtem objednaného množství.

V zásadě bychom se měli snažit odstranit z nástroje Návrhář sestav co nejvíce výpočtů a seskupení. Namísto použití skupin nástroje Návrhář sestav je lepší použít funkce seskupování v editoru dotazů. Aby nástroj Návrhář sestav mohl dotaz snadno zpracovat, změníme jej na pohled. V opačném případě se nástroj Návrhář sestav pokusí dotaz vylepšit vlastními funkcemi pro seskupování a třídění, což může rychle vést ke zcela nepraktickému kódování.

Konečný výsledek je zobrazen na obrázku 365:

Officeshop

Norderstr. 17, 43219 Phantastica

Officeshop - Norderstr. 17 - 43219 Phantastica

Mr.
Marko Patternman
Palace Avenue 42
06741 Behind the Mountain

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
2	paper, 500 sheet	\$4.23	\$8.46
1	rubber	\$0.75	\$0.75
4	sharpener	\$1.27	\$5.08
2	pencil HB	\$0.23	\$0.46
1	spiral-bound notepad	\$0.98	\$0.98
1	envelopes, 25 pcs.	\$1.25	\$1.25
4	CD-envelopes, 100 pcs.	\$1.89	\$7.56
1	wax crayons, 6 pcs.	\$3.85	\$3.85
1	folder, 2 inch width	\$1.89	\$1.89
2	clipboard	\$4.45	\$8.90
1	ring binder A4	\$5.76	\$5.76
1	CD-pens, 4 pcs.	\$4.56	\$4.56
2	CD-blanks, 50 pcs.	\$12.34	\$24.68
1	paper, recycling, 500 sheet	\$3.76	\$3.76
4	briefcase with register	\$6.87	\$27.48
2	receipt book, 50 sheet	\$1.35	\$2.70
10	bill book, 50 Blatt	\$3.15	\$31.50
2	datesamp, simple	\$18.50	\$37.00
2	stiff, small	\$12.00	\$24.00
24	hanging file folder, 25 pcs.	\$14.75	\$354.00

Carryover: \$554.62

Page 1

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE612345678900000456

Officeshop

Norderstr. 17, 43219 Phantastica

Page 2

Bill number: 2013-0
Date: 03/11/13

Quantity	Article	Price	Quantity*Price
		Carryover:	\$554.62
4	universallabels 70x32, 2700 pcs.	\$20.50	\$82.00
2	universallabels smallpack 12x30, 700 pcs.	\$4.15	\$8.30
4	printerhead, black	\$24.00	\$96.00
2	printerhead, color	\$26.30	\$52.60
6	ink cartridge, black, 32ml	\$5.40	\$32.40
10	ink cartridge, color, 17ml	\$5.20	\$52.00
4	toner laserprinter black	\$65.89	\$263.56
2	toner laserprinter color	\$78.89	\$157.78
4	register for folders, A-Z	\$1.25	\$5.00
2	staples	\$0.85	\$1.70
4	file recycling	\$0.65	\$2.60
1	paper, recycling, 500 sheet, color	\$4.15	\$4.15
1	pencils, 10 pcs., div. thickness	\$4.85	\$4.85
2	ballpen	\$1.35	\$2.70
1	watercolors, 12 colors	\$8.75	\$8.75
1	aquarelle, 24 colors	\$17.15	\$17.15
2	aquarelle brush, 3 pcs., div. thickness	\$8.34	\$16.68
1	drawing pad, A3, 20 sheet	\$3.85	\$3.85
4	desk pad 20*30 inch	\$15.67	\$62.68
2	paper, div. colors, 20*30 inch	\$0.45	\$0.90
10	cardboard, div. colors, 20*30 inch	\$0.89	\$8.90
	Total:		\$1,439.17

Bank details: Bank Phantastica - BIC WORTUE2X - IBAN DE612345678900000456

Obrázek 365: Konečný návrh sestavy Officeshopu

Zde se všechny položky vyskytují pouze jednou. Z původních 12 závěsných složek v dolní části strany 1 je nyní 24 závěsných složek. Quantity*Price bylo odpovídajícím způsobem přepočítáno.

Tisk sestav pro aktuální záznam ve formuláři

Zejména v případě výroby účtů, jak je uvedeno v předchozím příkladu, může být užitečné připravit a zobrazit náhled nového tisku po každém zadání položky. Obsah účtu by měl být stanoven pomocí formuláře a následovat by měl tisk jednotlivých dokumentů.

Sestavy nelze spouštět s filtrem pomocí maker. Dotaz, který se použije k sestavení sestavy, však lze předem filtrovat. To se provádí buď pomocí parametrizovaného dotazu ve tvaru

```
SELECT * FROM "bill" WHERE "ID" = :ID
```

nebo dotazem, který je opatřen daty pomocí jednořádkové filtrační tabulky:

```
SELECT * FROM "bill" WHERE "ID" = (SELECT "Integer" FROM "Filter" WHERE "ID" = TRUE)
```

V dotazu s parametry je třeba po jeho spuštění vložit obsah do příslušného dialogového pole.

Pokud se k řízení procesu používá filtrační tabulka, její obsah se zapisuje pomocí makra. Proto již není nutný samostatný záznam, což uživateli usnadňuje život. Postup je popsán níže.

Vytvoření filtru tabulky

Filtrační tabulka by měla obsahovat vždy pouze jeden záznam. To znamená, že jeho primárním klíčem může být pole Ano/Ne. Ostatní pole v tabulce jsou pojmenována tak, aby bylo zřejmé, jaký typ obsahu obsahují. V příkladu uvedeném v tabulce 15 se pole, které má filtrovat primární klíč tabulky Bill, nazývá Integer, protože samotný klíč je tohoto typu. Pro další možnosti filtrování lze později přidat další pole. Filtr Integer lze použít v různých časech pro několik různých tabulek, protože starý obsah se před tiskem vždy přepíše novým. Toto vícenásobné použití však funguje pouze v databázi pro jednoho uživatele (aplikace Base s interní HSQLDB). Ve víceuživatelské

databázi vždy existuje možnost, že některý jiný uživatel změní hodnotu filtru v některé z běžných tabulek, zatímco je prováděn dotaz, který filtr používá.

Tabulka 15: Filtrovací tabulka určená k filtrování primárního klíče tabulky Bill

Název pole	Typ pole
ID	Ano/Ne [BOOLEAN]
Integer	Integer [INTEGER]

Tato tabulka je na začátku vyplněna jedním záznamem. Za tímto účelem musí být v poli ID nastavena hodnota Ano (v jazyce SQL „TRUE“).

Vytvoření makra pro spuštění filtrované sestavy

Pro vyvolání jedné sestavy musí formulář někde obsahovat primární klíč tabulky Bill. Tento primární klíč je načten a přenesen do tabulky filtrů pomocí makra. Makro pak spustí požadovanou sestavu.

```
SUB Filter_and_Print
  DIM oDoc AS OBJECT
  DIM oDrawpage AS OBJECT
  DIM oForm AS OBJECT
  DIM oField AS OBJECT
  DIM oDatasorce AS OBJECT
  DIM oConnection AS OBJECT
  DIM oSQL_Command AS OBJECT
  DIM stSQL AS STRING
  oDoc = thisComponent
  oDrawpage = oDoc.Drawpage
  oForm = oDrawpage.Forms.getByName("MainForm")
  oField = oForm.getByName("fmtID")
  oDatasource = ThisComponent.Parent.CurrentController
  If NOT (oDatasorce.isConnected()) THEN
    oDatasource.connect()
  END IF
  oConnection = oDatasource.ActiveConnection()
  oSQL_Command = oConnection.createStatement()
  stSql = "UPDATE ""Filter"" SET ""Integer"" = '"+oField.GetCurrentValue()+"' WHERE
    ""ID"" = TRUE"
  oSQL_Command.executeUpdate(stSql)
  ThisDatabaseDocument.ReportDocuments.getByName("bill").open
END SUB
```

V tomto příkladu se formulář nazývá MainForm. Primární klíč se nazývá fmtID. Toto klíčové pole nemusí být viditelné, aby k němu makro mělo přístup. Hodnota pole se načte a příkaz UPDATE ji zapíše do tabulky. Poté je sestava spuštěna. Zobrazení, na které se hlášení vztahuje, je rozšířeno na podmínku:

```
... WHERE "bill_ID" = IFNULL((SELECT "Integer" FROM "Filter" WHERE "ID" =
TRUE), "bill_ID") ...
```

Odečte se pole Integer. Pokud nemá žádnou hodnotu, nastaví se na bill_ID. To znamená, že se zobrazí všechny záznamy, nikoli pouze filtrovaný záznam. Ze stejného zobrazení lze tedy vytisknout všechny uložené záznamy.

Alternativní zbarvení čar na pozadí

Při čtení jednotlivých řádků v tabulce v rámci sestavy může oko snadno sklouznout o řádek nahoru nebo dolů. Zabránit tomu pomáhá obarvení pozadí alespoň jednoho řádku. V příkladu na obrázku 366 jsou čáry jednoduše střídavě barevné. Výsledek vypadá takto:

Birthdaylist	
Kathrin	01/17/84
Sally	02/15/91
Mick	03/03/53
Hanne	04/13/70
Meike	04/13/71
Lara	04/23/85

Obrázek 366: Příklad zobrazující alternativní řádky barevně zvýrazněné pro zvýšení použitelnosti

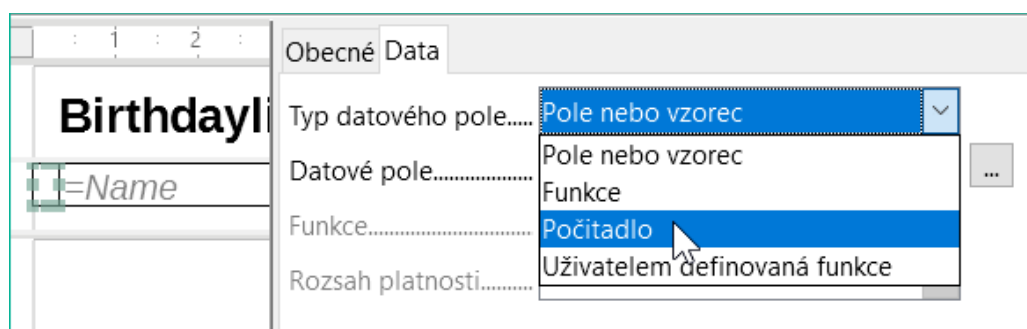
Základem zprávy je dotaz se jmény a daty. Původní tabulka je dotazována s řazením podle data (měsíc a den), aby se zobrazilo pořadí narozenin v průběhu roku. K tomu slouží:

```
... ORDER BY MONTH("birthday") ASC, DAY("birthday") ASC
```

Abychom získali střídavé barvy, musíme vytvořit funkci, která může později pomocí určité hodnoty nastavit podmínky, které následně určí barvu pozadí. V sestavě vytvoříme textové pole a pomocí **Vlastnosti > Data > Typ datového pole** definujeme počítadlo, jak je znázorněno na obrázku 367.

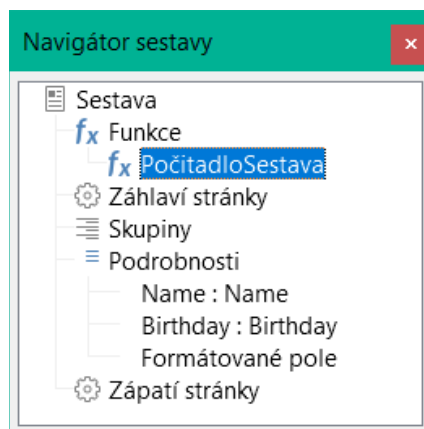
✓ Poznámka

Databáze Example_Report_Rows_Color_change_Columns.odt je součástí ukázkových databází k této knize.



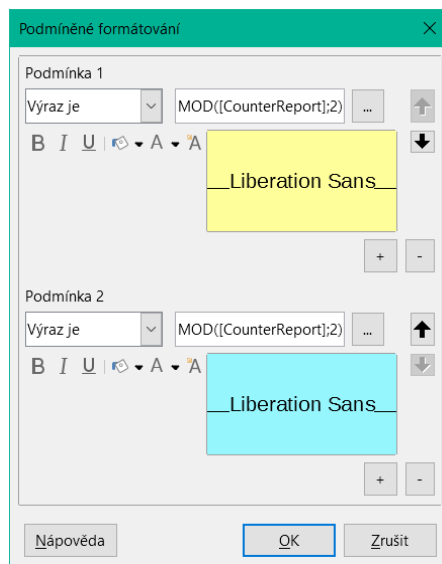
Obrázek 367: Nastavení vlastnosti Typ datového pole textového pole na hodnotu Counter

Potřebujeme název funkce pro podmíněné formátování. Skutečný čítač se ve výrazu nemusí objevit. Název lze přečíst přímo z pole. Pokud je pole opět smazáno, je název funkce přístupný pomocí **Zobrazení > Navigátor sestavy** (viz obrázek 368).



Obrázek 368: Název funkce zvýrazněný v navigátoru sestavy

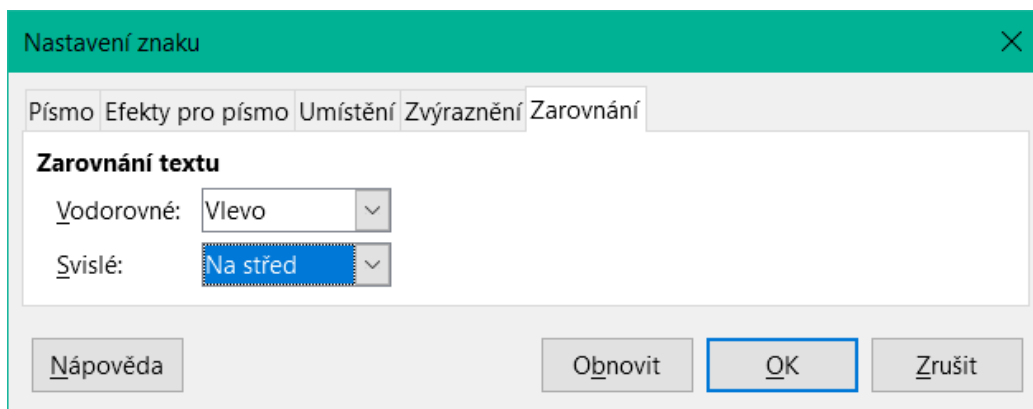
Nyní je třeba použít čítač, aby každé textové pole mělo svůj vlastní formát, jak je znázorněno na obrázku 369. Podmínkou je výraz, který není přímo spojen s polem. Proto je nastaven jako Podmínka 1 > Výraz je > $\text{MOD}([\text{CounterReport}]; 2) > 0$. MOD vypočítá zbytek po dělení. Pro všechna lichá čísla je větší než 0; pro sudá čísla je to přesně 0. Řádkům 1, 3 a 5 bude proto přiřazen tento formát.



Obrázek 369: Podmíněné formátování pro střídání barev mezi sousedními řádky

Druhá podmínka je formulována přesně opačně a je jí přiřazen odpovídající formát. Ve skutečnosti by bylo možné tuto druhou podmínku vynechat a ve vlastnostech každého pole nastavit výchozí formát. Podmíněný formát zadáný v první podmínce by pak nahradil toto výchozí nastavení, kdykoli by byla podmínka splněna.

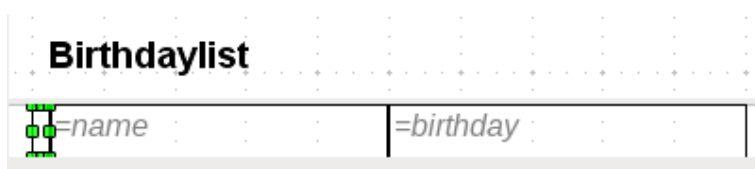
Vzhledem k tomu, že podmíněný formát přepíše veškeré výchozí formátování, je třeba zarovnání textu pro tento formát zahrnout pomocí nastavení znaků uvedených na obrázku 370.



Obrázek 370: Použití dialogu Nastavení znaků pro zarovnání textu

Zde mají být písmena zarovnána svisle na střed na barevné textové pole. Vodorovné zarovnání je standardní, ale je třeba přidat odsazení, aby se písmena netlačila na sebe na levé straně textového pole.

Pokusy s přidáváním mezer do obsahu dotazu nebo vzorce nevedly ke správnému odsazení textu. Úvodní mezery se jednoduše vyříznou.



Obrázek 371: Použití samostatného textového pole k dosažení konzistentního odsazení

Úspěšnější metodou je umístění textového pole před vlastní text, ale bez vazby na datové pole. Toto textové pole je podmíněně formátováno stejným způsobem jako ostatní, takže se při tisku objeví konzistentní viditelné odsazení (viz obrázek 371).

Dvousloupcové sestavy

Pomocí chytrých technik dotazování můžeme vytvořit sestavu s více sloupci tak, aby vodorovná posloupnost odpovídala po sobě jdoucím záznamům, jak je znázorněno na obrázku 372:

Birthdaylist			
Kathrin	01/17/84	Sally	02/15/91
Mick	03/03/53	Hanne	04/13/70
Meike	04/13/71	Lara	04/23/85
Monika	06/05/86	Karl	07/01/67
Paul	07/11/89	Egon	07/23/67
Susanne	08/02/65	Georg	08/28/95
Ysabelle	09/17/89	Maik	10/28/93
John	11/19/97	Erikan	12/17/75
Johann	12/23/91		

Obrázek 372: Příklad zobrazující dvousloupcový seznam narozenin

První záznam se vloží do levého sloupce, druhý do pravého. Záznamy jsou řazeny podle pořadí narozenin v rámci roku.

Třídění podle data narození způsobuje, že dotaz pro tuto zprávu je poměrně dlouhý. Pokud by se třídění provádělo podle primárního klíče základní tabulky, bylo by mnohem kratší. Kritérium třídění používá opakující se blok textu, který je vysvětlen níže.

Zde je dotaz, který je základem této sestavy:

```
SELECT "T1"."name" "name1", "T1"."birthday" "birthday1", "T2"."name" "name2",
"T2"."birthday" "birthday2"
FROM
  ( SELECT "name", "birthday", "rowsNr" AS "row" FROM
    ( SELECT "a".*,
      ( SELECT COUNT( "ID" ) FROM "birthdays" WHERE
        RIGHT( '0' || MONTH( "birthday" ), 2 ) ||
        RIGHT( '0' || DAY( "birthday" ), 2 ) || "ID"
        <= RIGHT( '0' || MONTH( "a"."birthday" ), 2 ) ||
        RIGHT( '0' || DAY( "a"."birthday" ), 2 ) || "a"."ID" )
      AS "rowsNr"
      FROM "birthdays" AS "a" )
    WHERE MOD( "rowsNr", 2 ) > 0 )
  AS "T1"
LEFT JOIN
  ( SELECT "name", "birthday", "rowsNr" - 1 AS "row" FROM
    ( SELECT "a".*,
      ( SELECT COUNT( "ID" ) FROM "birthdays" WHERE
        RIGHT( '0' || MONTH( "birthday" ), 2 ) ||
        RIGHT( '0' || DAY( "birthday" ), 2 ) || "ID"
        <= RIGHT( '0' || MONTH( "a"."birthday" ), 2 ) ||
        RIGHT( '0' || DAY( "a"."birthday" ), 2 ) || "a"."ID" )
      AS "rowsNr"
      FROM "birthdays" AS "a" )
    WHERE MOD( "rowsNr", 2 ) = 0 )
  AS "T2"
ON "T1"."row" = "T2"."row"
ORDER BY "T1"."row"
```

V tomto dotazu jsou dva stejné poddotazy. První dva sloupce se vztahují k poddotazu s aliasem T1 a poslední dva k poddotazu T2.

Poddotazy poskytují kromě polí v tabulce Birthdays další pole (rowsNr), které umožňuje rozlišovat řádky a třídit záznamy. Hlavní dotaz to využívá spolu s možností číslování řádků v dotazech (viz kapitola 5, Dotazy).

```
RIGHT( '0' || MONTH( "birthday" ), 2 ) || RIGHT( '0' || DAY( "birthday" ),
2 ) || "ID"
```

Tento vzorec umožňuje vytvořit jedinečnou posloupnost záznamů. Vzhledem k tomu, že záznamy příkladu mají být seřazeny podle data, bylo by snadné si myslet, že při porovnávání by se mělo použít pouze datum. V praxi je to obtížnější, protože nezáleží na samotných datech narození, ale na jejich pořadí v rámci jednoho roku. Další problémy způsobují shodné hodnoty dat, které brání vytvoření jedinečné posloupnosti. Třídění se tedy provádí nejen podle měsíce a dne, ale také podle jedinečného primárního klíče. A aby se zabránilo tomu, že měsíc 10 bude před měsícem 2, je před každý měsíc při sestavování kritéria řazení umístěna úvodní nula pomocí ||; pokud má měsíc již 2 číslice, je tato nula následně odstraněna pomocí RIGHT(. . . , 2).

SELECT COUNT("ID") zobrazí počet záznamů, jejichž kombinace měsíce, dne a primárního klíče je menší nebo rovna kombinaci pro aktuální záznam v tabulce Birthday. Zde máme korelační poddotaz (viz kapitola „Dotazy“).

MOD("rowsNr", 2) určuje, zda je toto číslo sudé nebo liché. MOD udává zbytek po dělení, v příkladu dělení dvěma. To způsobí, že rowsNr bude střídavě nabývat hodnot '1' a '0'. Tím se zase rozlišují dotazy pro T1 a T2.

V další úrovni dotazu T2 je řádek definován jako "rowsNr" -1. Tímto způsobem jsou T1 a T2 přímo srovnatelné.

T1 je propojen s T2 pomocí `LEFT JOIN`, takže všechna data v tabulce Birthdays jsou zobrazena i pro lichá čísla záznamů. Při kombinaci T1 a T2 lze nyní přímo odkazovat na skutečné řádky: "T1". "řádek" = "T2". "řádek".

Nakonec se celý obsah seřadí podle hodnoty řádek, která je stejná pro T1 a T2. Tuto funkci může sestava využít také přímo pomocí funkce seskupování, která je znázorněna na obrázku 373.

Birthdaylist

January		February	
Kathrin	01/17/84	Sally	02/15/91
March		April	
Mick	03/03/53	Hanne	04/13/70
		Meike	04/13/71
		Lara	04/23/85
May		June	
		Monika	06/05/86
July		August	
Karl	07/01/67	Susanne	08/02/65
Paul	07/11/89	Georg	08/28/95
Egon	07/23/67		
September		October	
Ysabelle	09/17/89	Maik	10/28/93
November		December	
John	11/19/97	Erkan	12/17/75
		Johann	12/23/91

Obrázek 373: Příklad seznamu narozenin s použitím dvou sloupců pro zobrazení narozenin seskupených podle měsíců

Vytvořit techniky dotazování pro vytváření podsekcí kromě dvousloupcového formátu je mnohem složitější. V takových případech se prázdné řádky vyskytují uprostřed sestavy, zatímco v předchozím dvousloupcovém příkladu se vyskytovaly až na konci. Nástroj Návrhář sestav s tímto typem dotazu nepracuje dobře. Proto místo toho použijeme dva propojené pohledy.

Nejprve vytvoříme následující zobrazení:

```
SELECT "a"."ID", "a"."name", "a"."birthday",
       MONTH( "a"."birthday" ) AS "monthnumber",
       ( SELECT COUNT( "ID" ) FROM "birthdays"
         WHERE MONTH( "birthday" ) = MONTH( "a"."birthday" )
           AND RIGHT( '0' || DAY( "birthday" ), 2 ) || "ID" <=
           RIGHT( '0' || DAY( "a"."birthday" ), 2 ) || "a"."ID" )
       AS "monthcounter"
FROM "birthdays" AS "a"
```

Jsou zahrnuta všechna pole tabulky Birthdays. Kromě toho je měsíc uveden jako číslo. Tabulce Birthdays je přidělen alias „a“, aby bylo možné k tabulce přistupovat pomocí odpovídajícího poddotazu.

Tento poddotaz spočítá všechny záznamy v měsíci, jejichž hodnoty data mají menší nebo stejné číslo dne. U stejných dat určuje primární klíč, které je na prvním místě. Tato technika je stejná jako v předchozím příkladu.

Pohled report_month_two_columns zpřístupňuje zobrazení monthnumber. Zde jsou uvedeny pouze výňatky z tohoto pohledu:


```

SELECT
  "Tab1"."name" AS "name1",
  "Tab1"."birthday" AS "birthday1",
  1 AS "monthnumber1",
  IFNULL("Tab1"."monthcounter", 999) AS "monthcounter1",
  'January' AS "month1",
  "Tab2"."name" AS "name2",
  "Tab2"."birthday" AS "birthday2",
  2 AS "monthnumber2",
  IFNULL("Tab2"."monthcounter", 999) AS "monthcounter2",
  'February' AS "month2"
FROM
  (SELECT * FROM "monthnumber" WHERE "monthnumber" = 1) AS "Tab1"
  RIGHT JOIN (SELECT * FROM "monthnumber" WHERE "monthnumber" = 2) AS
  "Tab2" ON "Tab1"."monthcounter" = "Tab2"."monthcounter"
UNION
SELECT "Tab1"."name" AS "name1",
  "Tab1"."birthday" AS "birthday1",
  1 AS "monthnumber1",
  IFNULL("Tab1"."monthcounter", 999) AS "monthcounter1",
  'January' AS "month1",
  "Tab2"."name" AS "name2",
  "Tab2"."birthday" AS "birthday2",
  2 AS "monthnumber2",
  IFNULL("Tab2"."monthcounter", 999) AS "monthcounter2",
  'February' AS "month2"
FROM
  (SELECT * FROM "monthnumber" WHERE "monthnumber" = 1) AS "Tab1"
  LEFT JOIN (SELECT * FROM "monthnumber" WHERE "monthnumber" = 2) AS
  "Tab2" ON "Tab1"."monthcounter" = "Tab2"."monthcounter"
UNION
...
ORDER BY "monthnumber1", "monthcounter1", "monthcounter2"

```

Nejprve se z čísla měsíce načtou všechna data, pro která číslo měsíce = 1. Tomuto výběru je přiřazen alias Tab1. Současně se do tabulky Tab2 načtou všechna data pro číslo měsíce = 2. Tyto tabulky jsou propojeny pomocí RIGHT JOIN, takže se zobrazí všechny záznamy z Tab2, ale pouze ty záznamy z Tab1, které mají stejné počítadlo měsíce jako Tab2.

Sloupce zobrazení musí mít různé názvy, aby každý sloupec měl alias. Jako číslo měsíce je také přímo zadána jednička pro Tab1 a leden jako měsíc1. Tyto záznamy se objevují také v případě, že na kartě Tab1 není žádný záznam, ale na kartě Tab2 je stále několik záznamů. Pokud není k dispozici žádné počítadlo měsíců, zadá se hodnota 999. Protože je karta Tab1 propojena s kartou Tab2 pomocí RIGHT JOIN, může se stát, že pokud je na kartě Tab1 méně záznamů, zobrazí se místo nich prázdná pole. Vzhledem k tomu, že při pozdějším třídění by se takové záznamy dostaly před všechny záznamy s obsahem, je místo toho zadáno velmi vysoké číslo.

Vytvoření sloupců pro Tab2 je podobné. Zde však není nutné použít kód IFNULL("Tab2"."monthcounter", 999), protože při RIGHT JOIN pro Tab2 se zobrazí všechny řádky z Tab2, ale již nebude platit, že by z Tab1 mohlo vzniknout více řádků než z Tab2.

Právě tento problém se řeší propojením dvou dotazů. Při použití UNION se zobrazí všechny záznamy z prvního dotazu a všechny záznamy z druhého dotazu. Záznamy z druhého dotazu se zobrazí pouze v případě, že nejsou totožné s předchozím záznamem. To znamená, že UNION funguje jako DISTINCT.

Pomocí UNION je znovu položen stejný dotaz, pouze Tab1 a Tab2 jsou nyní propojeny pomocí LEFT JOIN. To způsobí, že se zobrazí všechny záznamy z Tab1, i když Tab2 obsahuje méně záznamů než karta Tab1.

Pro měsíce 3 a 4, 5 a 6 atd. se použijí přesně ekvivalentní dotazy, které se opět připojí k předchozímu dotazu pomocí UNION.

Nakonec je výsledek zobrazení seřazen podle čísla monthnumber1, monthcounter1 a monthcounter2. Není třeba řadit podle monthnumber2, protože monthnumber1 již poskytuje stejnou posloupnost záznamů.

Zdroje chyb ve zprávách

Nástroj Návrhář sestav někdy skrývá chyby, jejichž přesnou příčinu nelze dodatečně snadno určit. Zde je několik zdrojů chyb a užitečných protiopatření.

Obsah pole z dotazu se nezobrazuje

Pro simulaci prodeje zásob je vytvořena databáze. Dotaz vypočítá celkovou cenu z počtu zakoupených položek a jednotkové ceny.

```
SELECT "sales"."sum", "stock"."stock", "stock"."Price",  
"sales"."sum"*"stock"."Price" FROM "sales", "stock"  
WHERE "sales"."stock_ID" = "stock"."ID"
```

Tento dotaz slouží jako základ sestavy. Pokud je však v sestavě vyvoláno pole "sales"."sum"*"stock"."Price", zůstane prázdné. Pokud je naopak v dotazu uveden alias, může k němu nástroj Návrhář sestav snadno přistupovat:

```
SELECT "sales"."sum", "stock"."stock", "stock"."Price",  
"sales"."sum"*"stock"."Price" AS "TPrice"  
FROM "sales", "stock" WHERE "sales"."stock_ID" = "stock"."ID"
```

Pole nyní přistupuje k položce "Tprice" a zobrazuje odpovídající hodnotu.

Sestavu nelze vypracovat

Někdy se může stát, že sestava je připravena, ale pak není vytvořena nebo dokonce uložena. Zobrazí se poměrně neinformativní chybová zpráva:

„Sestavu nebylo možné vytvořit. Byla zachycena výjimka typu com.sun.star.lang.WrappedTargetException“.

Zde může být užitečné přečíst si doplňující informace připojené k sestavě. Pokud se zde objeví odkaz na SQL, znamená to, že nástroj Návrhář sestav nedokáže správně interpretovat kód SQL ve zdroji dat.

Zde může pomoci použít nástroj Navigátor sestavy a nastavit příkaz Data > Analyzovat SQL > Ne. Toto řešení však způsobí, že stávající skupiny přestanou fungovat.

Lepší způsob je použít jako základ sestavy spíše pohled než dotaz. Ten je nastaven tak, že se pro nástroj Návrhář sestav tváří jako tabulka a lze jej bez problémů upravovat. Dokonce i požadavky na třídění zobrazení fungují bez problémů.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 7

Propojení s databázemi

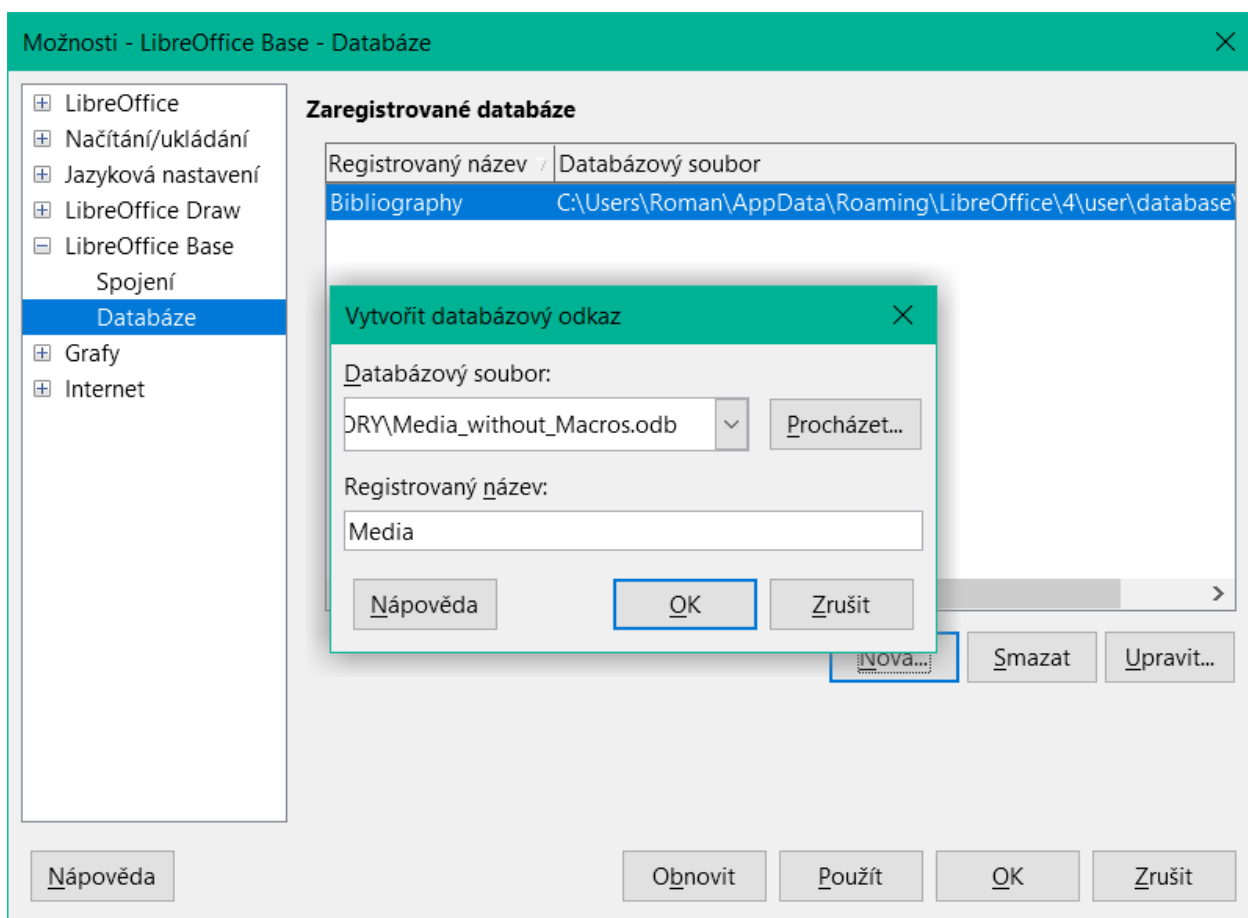
Obecné poznámky k propojení databází

Pomocí aplikace Base můžeme dokumenty v aplikacích LibreOffice Writer a Calc používat různými způsoby jako zdroje dat. To znamená, že použití aplikace Base není nutně vázáno na registraci databází v konfiguraci LibreOffice. Externí formuláře mohou také přímo komunikovat s aplikací Base, pokud je zadána cesta ke zdrojům dat.

Registrace databází

Mnoho funkcí, jako je tisk štítků nebo použití dat pro dopisy s poli, vyžaduje registraci databáze v konfiguraci LibreOffice.

Pomocí **Nástroje > Možnosti > LibreOffice Base > Databáze > Nový** můžeme zaregistrovat databázi pro následné použití jinými komponentami LibreOffice.



Obrázek 374: Registrace databáze pro použití jinými komponentami LibreOffice

Vyhledáme databázi pomocí prohlížeče souborů a připojíme ji k LibreOffice podobným způsobem jako v případě jednoduchého formuláře. Samotné databázi dáme vhodný informativní registrovaný název, například název souboru databáze. Název slouží jako alias, který lze použít i v dotazech do databáze.

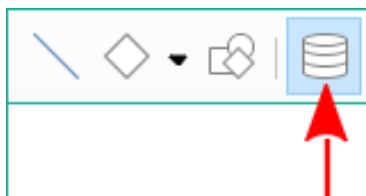
Prohlížeč zdrojů dat

Prohlížeč zdrojů dat v aplikaci Writer a Calc umožňuje přístup k tabulkám a dotazům všech registrovaných databází pod jejich registrovanými názvy. Prohlížeč otevřeme pomocí **Zobrazit > Zdroje dat** nebo stisknutím kláves **Ctrl+Shift+F4**, případně klepnutím na ikonu na Standardní nástrojové liště.

Ikona Zdroje dat není na Standardní nástrojové liště obvykle viditelná. Chceme-li ji zviditelnit, klepneme pravým tlačítkem myši na tlačítko Uložit a otevřeme tuto nástrojovou lištu. Přejdeme dolů na *Viditelná tlačítka*. Tím se otevře seznam všech tlačítek. Přejdeme dolů až téměř k dolní části a najdeme tlačítko Zdroje dat. Klepnutím na něj jej zviditelníme na pravém konci nástrojové lišty.

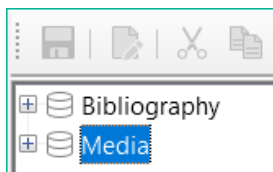
Tip

Pokud používáme notebook, bude možná nutné stisknout klávesy *Ctrl + fn + shift + F4*. Klávesa fn (funkce) umožňuje používat klávesy F pro více než jednu funkci.



Obrázek 375: Tlačítko Zdroje dat na Standardní nástrojové liště

Registrované zdroje dat se zobrazují na levé straně prohlížeče zdrojů dat, který je ve výchozím nastavení umístěn v horní části pracovní plochy. Zdroj dat použité literatury je ve výchozím nastavení součástí LibreOffice. Ostatní zdroje dat jsou uvedeny pod svými registrovanými názvy.



Obrázek 376: Registrované zdroje dat zobrazené ve stromu názvů na levé straně prohlížeče zdrojů dat

Klepnutím na rozbalovací znaménko před názvem databáze otevřeme databázi a zobrazíme podsložky pro dotazy a tabulky. Ostatní podsložky databáze zde nejsou k dispozici. Interní formuláře a sestavy jsou přístupné pouze otevřením samotné databáze.

Teprve po klepnutí na složku Tabulky je databáze skutečně zpřístupněna. U databází chráněných heslem je třeba v tomto okamžiku zadat heslo.

Vpravo od stromu názvů se zobrazí vybraná tabulka. Lze ji upravovat stejně jako v aplikaci Base. Přímý vstup do tabulek je však třeba ve velmi složitých relačních databázích provádět opatrně, protože tabulky jsou propojeny cizími klíči. Například níže uvedená databáze má samostatné tabulky pro názvy ulic, poštovních směrovacích čísel a obcí.

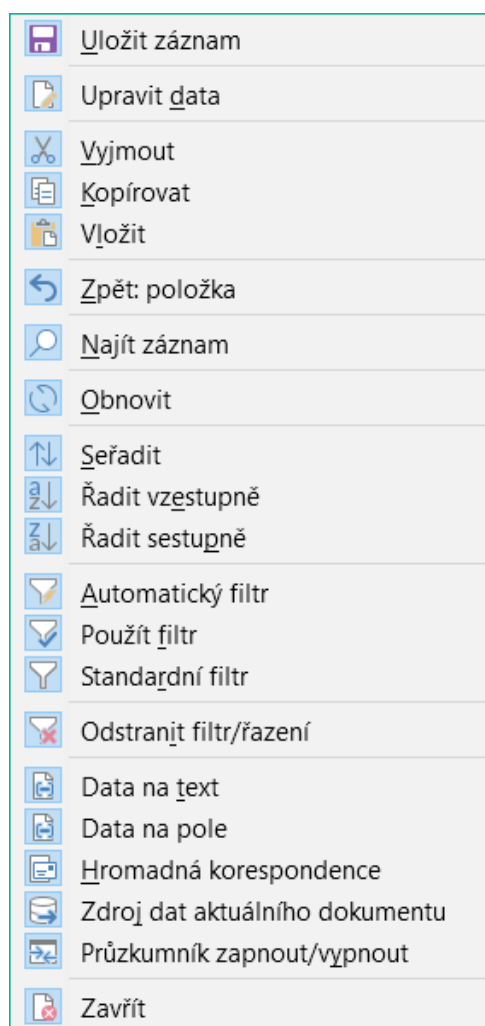
ID	FirstName	LastName
0	Max	Horkheimer
1	Steven W.	Hawking
2	Konrad	Lorenz
3	J.R.R.	Tolkien
4	Ursula	Hermann
5	Dr. Lutz	Götze
6	Dr. Klaus	Heller
7	Erich	Kästner
8	Dave	Edmunds
9	Herman	van Veen
+	<Auto	

Obrázek 377: Výběrem tabulky ve stromu názvů zobrazíme údaje o složkách na pravé straně prohlížeče zdrojů dat.

Pro správné zobrazení dat (ale bez možnosti úprav) jsou vhodnější dotazy nebo pohledy.

Mnohé ikony na nástrojové liště (obrázek 378) budou známé ze zadávání dat do tabulek. (Ikony na vašem displeji se mohou lišit od ikon na obrázku.)

Hlavní nové ikony jsou uvedeny v poslední části: Data na text, Data na pole, Hromadná korespondence, Zdroj dat aktuálního dokumentu, Průzkumník zapnout/vypnout. Jejich použití je popsáno níže s využitím tabulky Reader v databázi Media.



Obrázek 378: Výchozí přiřazení tlačítek na nástrojové liště Data tabulky prohlížeče zdrojů dat

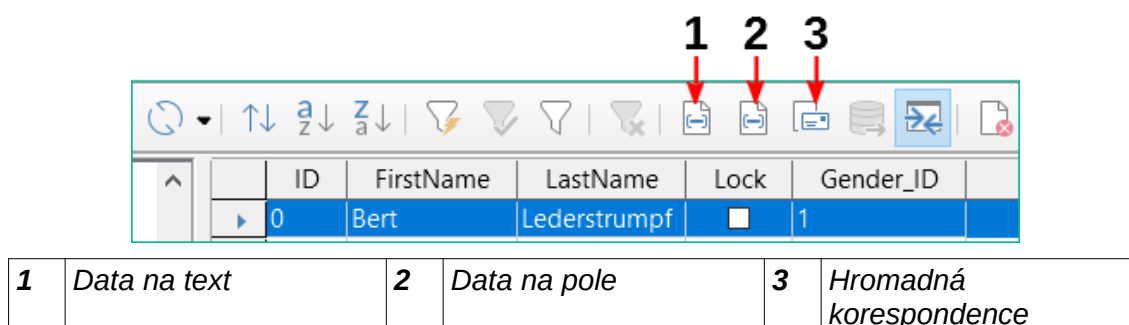
Data na text

Tip

Pomocí této metody lze data vkládat přímo na konkrétní místa v textovém dokumentu nebo do konkrétních buněk tabulky. I když lze data do těchto míst zadat, jejich vložení zaručuje jejich přesnost. To je důležité při použití sloučení pošty, o kterém bude řeč později.

Při odesílání stejného dokumentu různým osobám je zaručeno, že všichni obdrží naprosto stejné údaje.

Výběrem jednoho nebo více záznamů aktivujeme funkce Data na text a Data na pole.

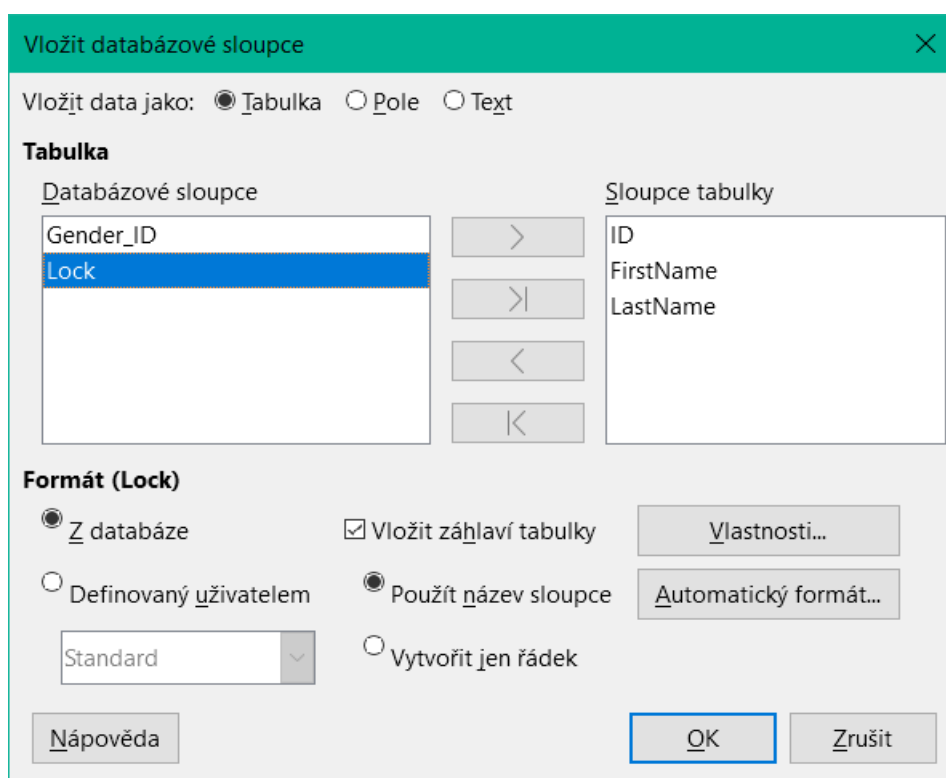


Obrázek 379: Výběrem datového záznamu aktivujeme ikonu Data do textu na nástrojové liště Data tabulky.

Pokud nyní zvolíme **Data do textu**, zobrazí se Průvodce provedením potřebného formátování (obrázek 380).

Tři možnosti zadávání dat jako textu jsou: jako tabulka, jako jednotlivá pole, nebo jako běžný text.

Na obrázku 380 je zobrazena možnost **Vložit data jako tabulku**. V případě číselných polí a polí s daty lze formát databáze změnit na zvolený formát. V opačném případě se formátování provede automaticky při výběru polí tabulky. Pořadí polí se nastavuje pomocí šipek.



Obrázek 380: Dialogové okno Vložit sloupce databáze s vybraným přepínačem Tabulka

Jakmile jsou vybrány sloupce tabulky, aktivuje se tlačítko **Vlastnosti** pro danou tabulku. To umožňuje nastavit obvyklé vlastnosti tabulky aplikace Writer (šířka tabulky, šířka sloupců atd.).

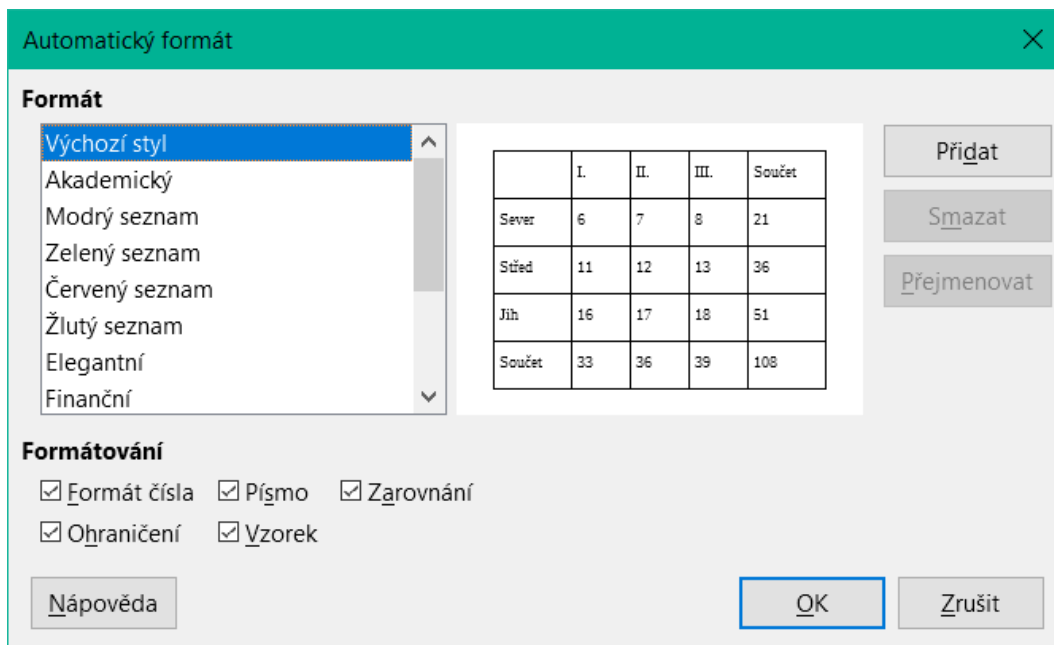
Zaškrtnuté políčko **Vložit záhlaví tabulky** určuje, zda je vyžadováno záhlaví tabulky. Pokud není zaškrtnuto, nebude pro nadpisy vyhrazen žádný samostatný řádek.

Řádek vybraný pro záhlaví tabulky lze převzít z názvů sloupců nebo lze záznam vypsát s ponecháním místa pro pozdější úpravu záhlaví. Vybereme možnost **Vytvořit jen řádek**.

Pomocí tlačítka **Automatický formát** můžeme otevřít dialogové okno s několika předformátovanými styly tabulek. Kromě navrhovaného výchozího stylu lze všechny formáty

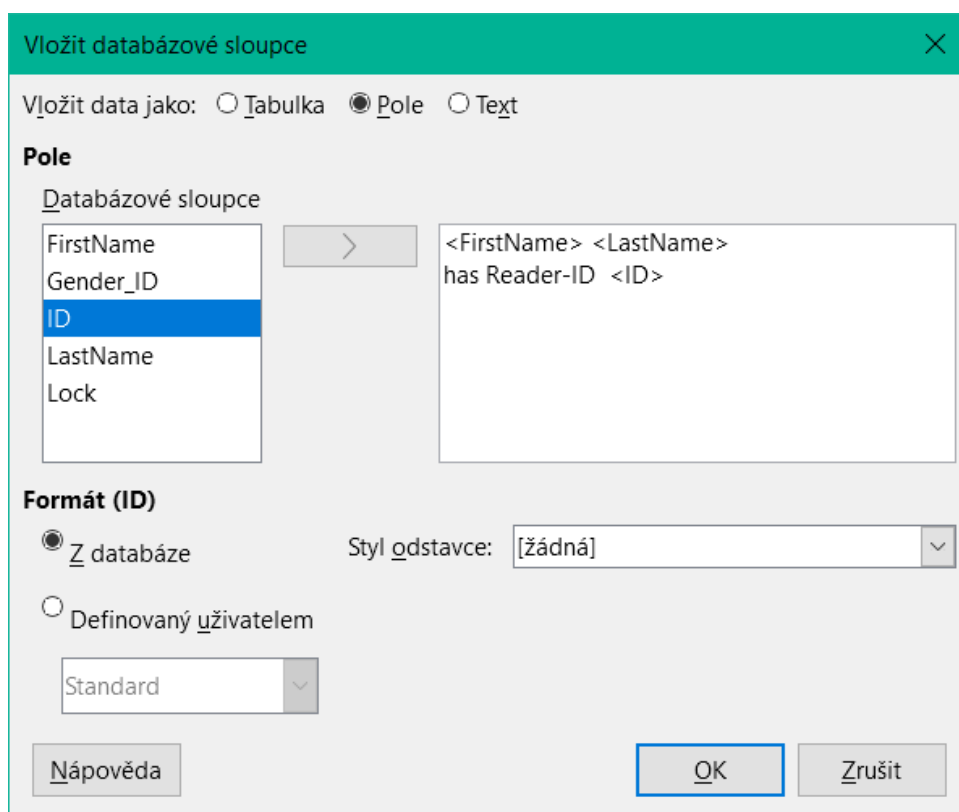
přejmenovat. (Můžeme také přidat autoformáty; za tímto účelem nejprve vytvoříme tabulku v požadovaném formátu. Poté vybereme tabulku a klepnutím na tlačítko **Přidat** přidáme její formát do seznamu.) Tabulku můžeme také formátovat ve Writeru tak, že ji vybereme a zvolíme formát ze seznamu Styly tabulky na kartě Styly v postranní liště; seznam stylů tabulky je stejný jako seznam formátů v dialogovém okně Automatický formát.

Chceme-li vytvořit tabulku s vybranými záznamy a sloupci, klepneme na tlačítko **OK** v dialogovém okně Vložit databázové sloupce.



Obrázek 381: Dialogové okno Automatický formát

Vkládání dat jako polí umožňuje pomocí minieditoru umístit v textu postupně různá pole tabulky. Takto vytvořený text lze také opatřit stylem odstavce. I v tomto případě lze formátování dat a čísel zadat samostatně nebo je lze načíst přímo z nastavení tabulky v databázi.

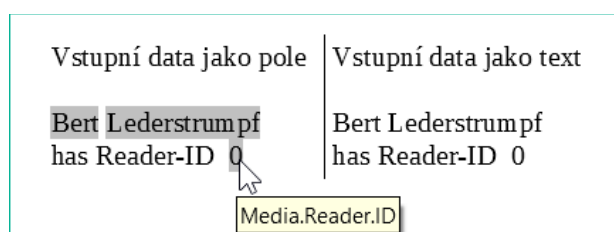


Obrázek 382: Dialogové okno Vložit sloupce databáze se zaškrtnutým přepínačem Pole

Takto vložená pole do textu lze následně jednotlivě vymazat nebo použít pro hromadnou korespondenci.

Pokud zvolíme možnost **Vložit data jako text**, jediný rozdíl oproti použití polí je ten, že pole zůstávají propojena s databází. Při vkládání jako text se přenese pouze obsah zadaných polí, nikoli odkaz na skutečnou databázi.

Výsledky obou postupů jsou porovnány níže.

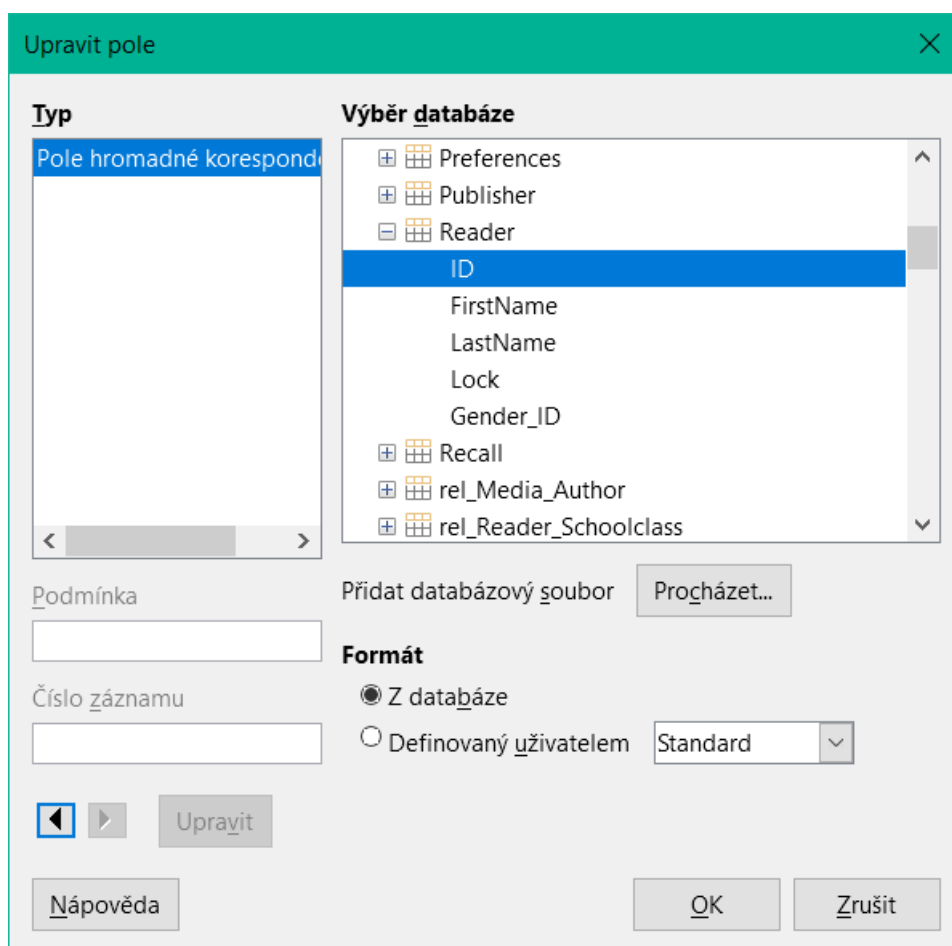


Obrázek 383: Porovnání možností Data jako pole a Data jako text

Pole mají šedá pozadí. Pokud na pole najedeme kurzorem myši, zobrazí se nápověda, že pole jsou propojena s databází Media, s tabulkou Reader a v rámci této tabulky s polem ID.

Tak například dvojitě klepnutí na pole ID otevře následující přehled. Tím je zřejmé, které pole bylo vytvořeno pomocí procedury Vložit data jako pole. Jedná se o stejný typ pole, který je zobrazen pomocí **Vložit > Pole > Další pole > Databáze**.

Jednodušší je vytvořit takové pole výběrem záhlaví sloupce tabulky v prohlížeči zdrojů dat a jeho přetažením do dokumentu pomocí myši. Tímto způsobem můžeme vytvořit přímo formulářový dopis.



Obrázek 384: Dvojitým klepnutím na vložené pole se otevře dialogové okno Upravit pole

Data na pole

Tip

Tato metoda je užitečná, když se dokument posílá několika osobám, z nichž každé se pošlou některé údaje, které jsou specifické pouze pro ni. To se provádí pomocí hromadné korespondence.

Knihovny například rozesílají lidem oznámení se seznamem médií, která si vypůjčili a nevrátili včas. Seznam se obvykle u každé osoby liší, ale všichni dostanou nějaké varování. Typ varování samozřejmě závisí na době, která uplynula od doby, kdy mělo být médium vráceno. Všichni, kteří spadají do určitého časového období, obdrží stejné upozornění.

Vložení dat do polí (viz obrázek 380):

- 1) Klepnutím na levý okraj jednoho z testovacích řádků jej zvýrazníme.
- 2) Klepnutím na tlačítko **Data na textu** otevřeme průvodce **Vložit databázové sloupce**.
- 3) Zvolíme přepínač, **Pole**.
- 4) Přesuneme pole databáze, která chceme použít, z levého seznamu do pravého v požadovaném pořadí. Text lze přidat stejně jako na obrázku 382.
- 5) Chceme-li na tato pole použít konkrétní styl odstavce, vybereme jej z rozevíracího seznamu Styly odstavce.
- 6) Klepneme na **OK**.

Tip

Vkládání dat do textu se provádí stejným způsobem jako vkládání dat do polí. Jediný rozdíl je v tom, zda vybereme volbu Text, nebo Pole. Rozdíl je ve vzhledu v dokumentu nebo tabulce, jak je uvedeno níže. Vlevo jsou to data na text, vpravo data na pole.

Bert Lederstrumpf has ID 0 Bert Lederstrumpf has ID 0

Obrázek 385: Levá strana vložena jako text; pravá strana vložena jako pole

Tip

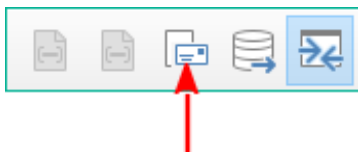
Po vložení pole lze hodnoty, které se v něm zobrazují, měnit. Vybereme záznam (řádek) s ID = 1 a klepneme na tlačítko **Data na pole**.

Bert Lederstrumpf has ID 0. Heinrich Müller has ID 1.

Obrázek 386: Data vložena jako pole se mohou měnit

Hromadná korespondence

Tlačítko Hromadná korespondence spustí Průvodce hromadnou korespondencí. Formulářový dopis sestavuje svá data z různých tabulek, takže je třeba nejprve spustit databázi. V databázi pak vytvoříme nový dotaz, který zpřístupní požadovaná data.



Obrázek 387: Ikona Hromadná korespondence na nástrojové liště Tabulka dat

Chceme-li spustit databázi, klepneme pravým tlačítkem myši na samotnou databázi nebo na některou z jejích tabulek či dotazů; tím se okamžitě obnoví zobrazení v prohlížeči zdrojů dat. Poté lze vyvolat Průvodce hromadnou korespondencí klepnutím na příslušné tlačítko.

Zdroj dat aktuálního dokumentu

Klepnutím na tlačítko **Zdroj dat aktuálního dokumentu** otevřeme přímé zobrazení tabulky, která tvoří základ pro data vložena do dokumentu. Ve výše uvedeném příkladu se zobrazí tabulka Reader z databáze Media.

Průzkumník zapnout/vypnout

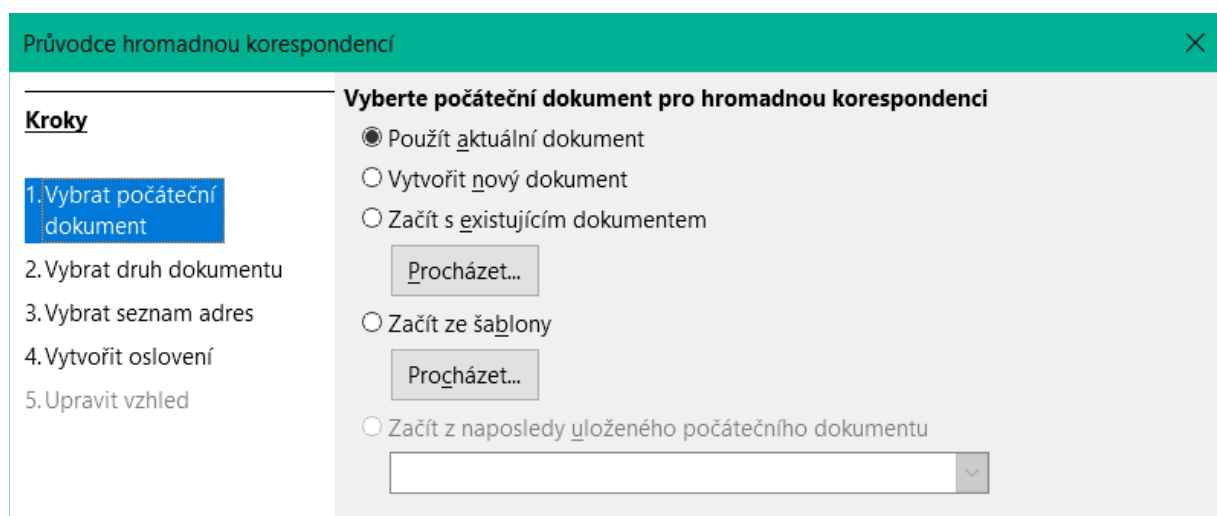
Přepnutím tlačítka **Průzkumník zapnout/vypnout** se zobrazí nebo skryje strom adresářů v levé části zobrazení tabulky. To v případě potřeby umožňuje větší prostor pro zobrazení dat. Chceme-li přistupovat k jiné tabulce, musíme Průzkumník znovu zapnout.

Vytváření dokumentů hromadné korespondence

Průvodce hromadnou korespondencí je přístupný také z prohlížeče databáze. Tento průvodce umožňuje v malých krocích vytvořit pole adresy a pozdravu ze zdroje dat. V zásadě můžeme tato pole vytvořit bez použití Průvodce. Zde si na příkladu projdeme jednotlivé kroky Průvodce. V následujících krocích k tomu použijeme Průvodce. Tentokrát bude zdrojem dat dotaz, konkrétně *Readeraddresses* společnosti Media. Vyhledáme ji v rozevíracím seznamu Dotaz stejně jako dříve tabulku Reader v rozevíracím seznamu Tabulka.

Tip

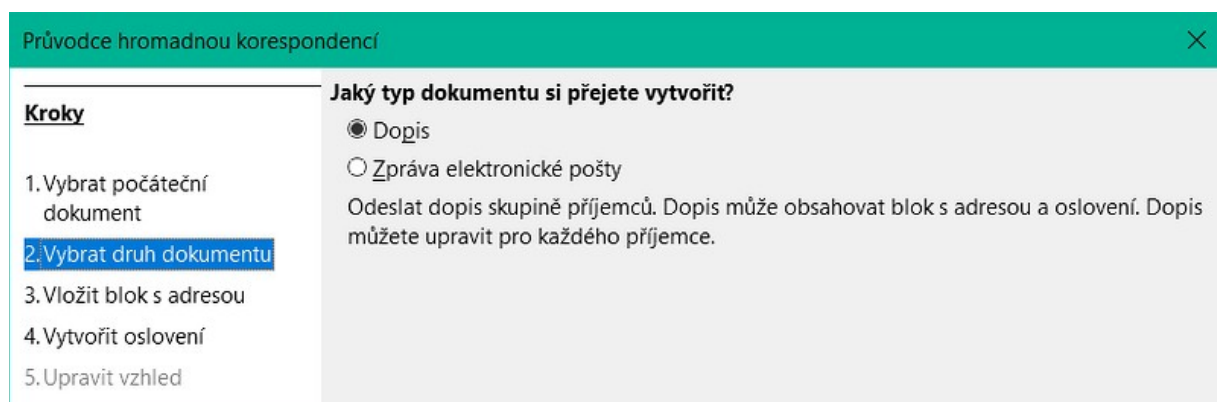
Odebrání libovolného textového dokumentu v aplikaci Writer, pokud obsahuje odkazy na databázi. Nelze vytvořit nový odkaz na tento dokument, pokud je starý odkaz stále aktivní. Začneme novým dokumentem, který může být bez názvu, nebo šablonou dopisu, která neobsahuje žádné odkazy.



Obrázek 388: Průvodce hromadnou korespondencí – krok 1

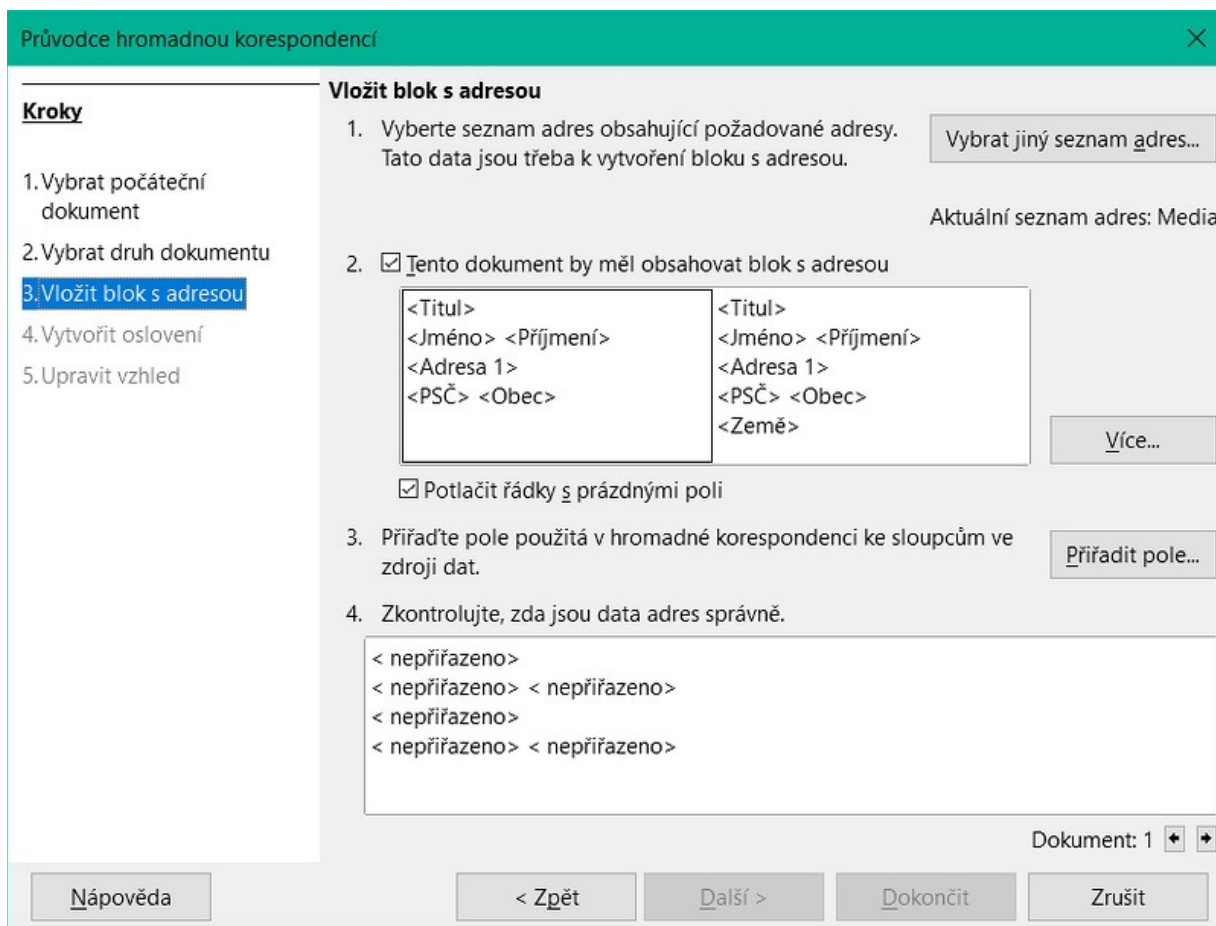
Výchozí dokument pro formulářový dopis je dokument, ke kterému budou připojena pole databáze.

Sloučený dokument je dokument obsahující údaje pro různé osoby, kterým mají být zaslány formulářové dopisy. Ve sloučeném dokumentu není žádná vazba na zdroj dat. Je podobný výstupu funkce Vložit data jako text.



Obrázek 389: Průvodce hromadnou korespondencí – krok 2

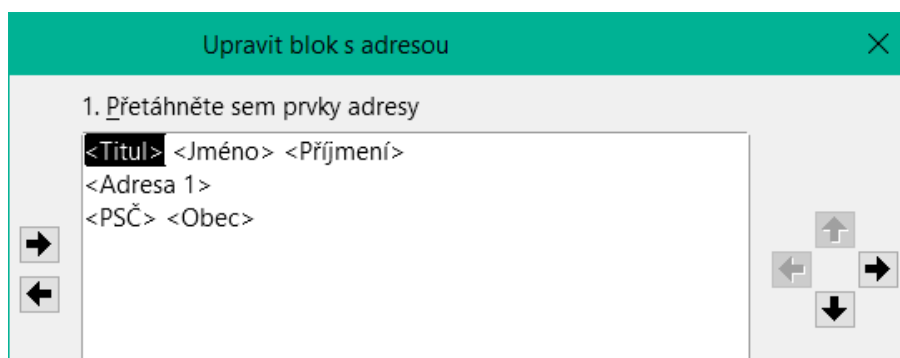
Průvodce hromadnou korespondencí může vytvářet dopisy nebo e-maily pomocí záznamů z databáze. V tomto příkladu vytvoříme dopisy pomocí tabulky Reader databáze Media.



Obrázek 390: Průvodce hromadnou korespondencí – krok 3

Zadání bloku adres umožňuje nejrozsáhlejší konfiguraci. Navrhovaný seznam adres pochází z aktuálně vybraného dotazu nebo tabulky v aktuálně vybrané databázi.

Krok 3 určuje celkový vzhled bloku adres, který lze dále přizpůsobit klepnutím na tlačítko **Další**. Viz obrázek 390. Levá Adresa je již vybrána a tento blok bude použit.



Obrázek 391: Změna obsahu bloku adres pomocí dialogového okna Upravit blok adres

Je třeba doplnit jeden prvek: <Address Line 2>. Provedeme to následovně:

- 1) Klepneme na tlačítko **Další**.
- 2) Přetáhneme prvek <Address Line 2> v seznamu Address elements a umístíme jej napravo od prvku <Address Line 1>.
- 3) Pokud mezi těmito dvěma prvky není mezera, klepneme na šipku vpravo na pravé straně dialogového okna a vytvoříme mezery.

- 4) Klepneme na **OK**.
- 5) Vybrat blok s adresou: klepneme na **OK**.

Prvek <Title> je třeba přesunout o řádek níže a umístit jej před prvky <First Name> <Last Name>. Nezapomeneme mezi <Title> a <First Name> udělat mezeru. Pomocí čtyř šipek vpravo přesuneme nejprve <Last Name> a poté <First Name>.

Krok 3 slouží k propojení pojmenovaných polí v bloku adres se správnými poli v databázi. Průvodce zpočátku rozpozná pouze ta databázová pole, která mají přesně stejné názvy jako ta, která Průvodce používá. V tomto příkladu se žádné z polí neshoduje, takže v tomto kroku bude nutné vybrat všechna pole z rozevíracích seznamů.

- Pro položku <Title> vybereme možnost Salutation.
- Pro položku <FirstName> vybereme First Name.
- Pro položku <LastName> vybereme Last Name.
- Pro položku <Address Line 1> vybereme Street.
- Pro položku <Address Line 2> vybereme možnost No.
- Pro položku <City> vybereme Town.
- Pro položku <Zip> vybereme Postal code.

Prvky adresy	Odpovídá poli:
<Titul>	Title Dear Mr.
<Jméno>	FirstName Bert
<Příjmení>	LastName Lederstrumpf
<Firma>	
<Adresa 1>	Street Neuenkirchener Str.
<Adresa 2>	No 72
<Obec>	Town Pusemuckel

Obrázek 392: Přiřazení polí zdroje dat k prvkům adresy pomocí dialogového okna Přiřadit pole

Zde jsou prvky adresy přiřazeny k odpovídajícím prvkům z dotazu databáze úspěšně přenesené pomocí Průvodce hromadnou korespondencí. Pro náhled se opět použije první záznam v dotazu.

Nastavení databáze v podstatě končí krokem 4. Zde je třeba pouze vybrat, z jakého pole se má pohlaví příjemce převzít. Toto pole již bylo pojmenováno, takže je třeba zadat pouze obsah pole pro příjemce ženského pohlaví.

✓ Poznámka

Protože průvodce má v tomto bodě chybu, osobní pozdrav se vytvoří pomocí převodu dat na text, jak je popsáno výše. Konkrétně pole Salutation poskytne správný titul pro každou osobu. Zbytek pozdravu se vytvoří zadáním do sloučeného dokumentu.

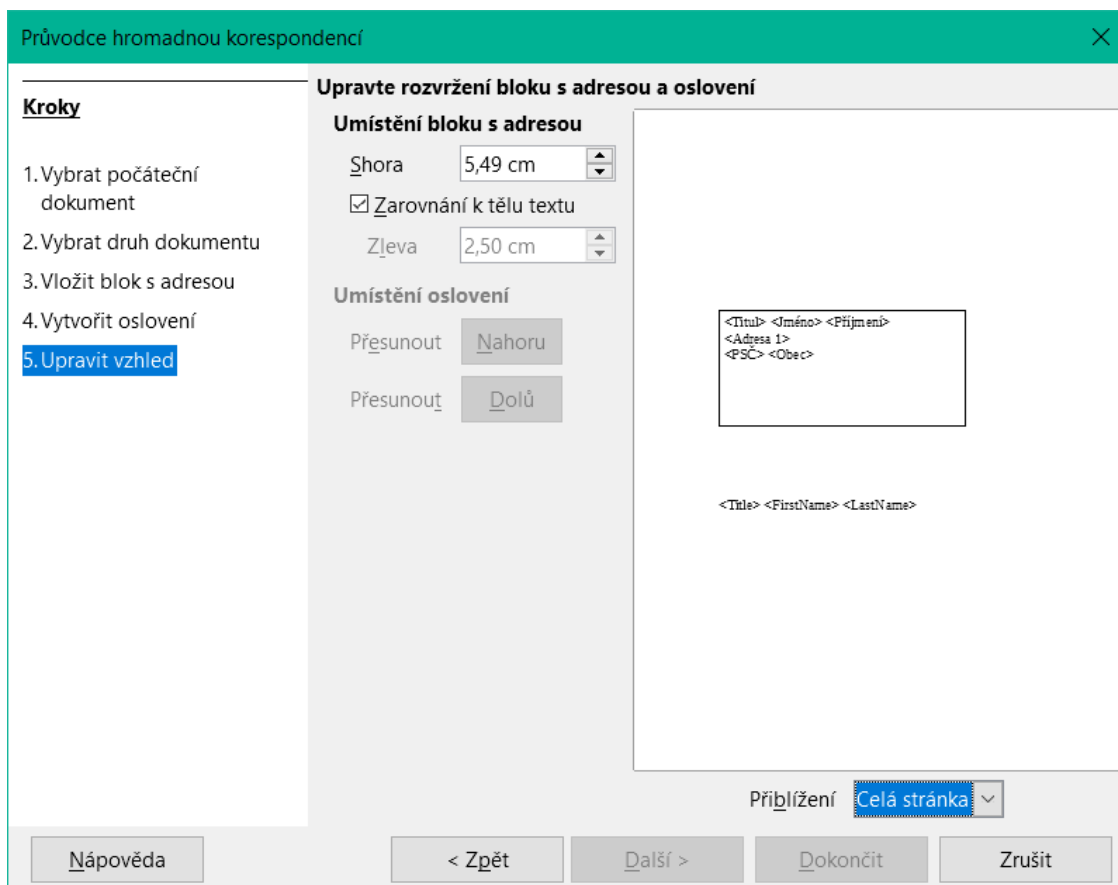
- 1) Chceme-li tuto stránku dokončit, zrušíme zaškrtnutí políčka *Vložit personalizované oslovení*.
- 2) V *Obecné oslovení* neprovádíme žádné změny. Bude nahrazen později, ale je nutný k určení místa, kde má být v dopise pozdrav.

Vytvořit oslovení
 Tento dokument by měl obsahovat oslovení
 Vložit personalizované oslovení
Žena
Muž
Pole v seznamu adres, podle kterého se pozná žena
Název pole
Hodnota pole
Obecné oslovení

Náhled

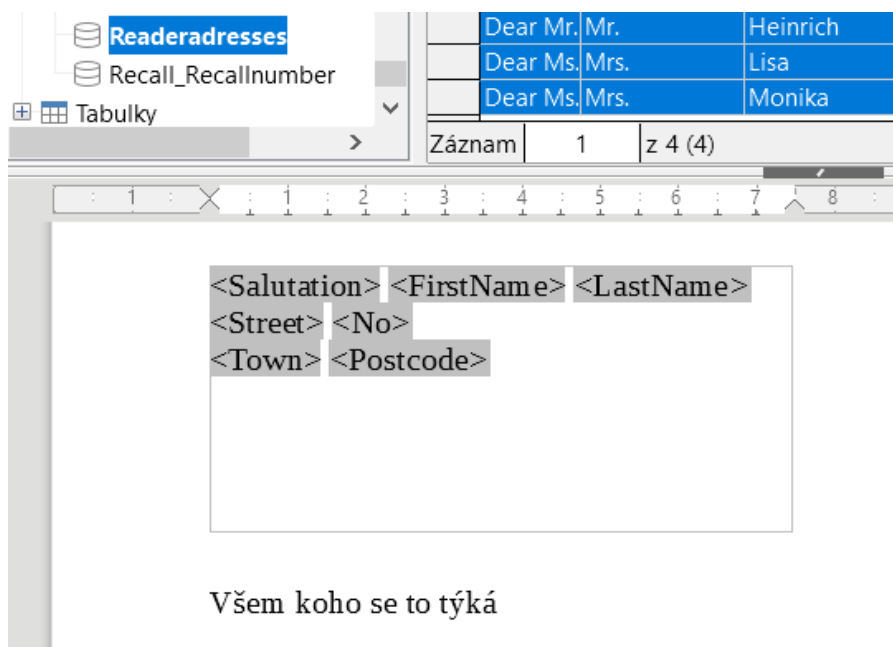
Obrázek 393: Vytvoření pozdravu v kroku 4 Průvodce hromadnou korespondencí

Klepneme na **Další >**. V kroku 5 můžeme upravit umístění bloku adresy a pozdravu na stránce. (Viz obrázek 394.) Poté klepneme na **Dokončit**.



Obrázek 394: Průvodce hromadnou korespondencí – krok 5

Nyní dokončíme rozvržení dokumentu hromadné korespondence. Obsahuje pole Address Block, kde jsme je umístili.



Obrázek 395: Dokument Writer po dokončení interakce s Průvodcem hromadnou korespondencí

Nyní pomocí funkce Data na text nahradíme pozdrav.

- 1) Nahradíme *To whom it may concern* za *Dear*.

- 2) Přetáhneme <Salutation> na jedno místo za Dear.
- 3) Přetáhneme <FirstName> na jedno místo za <Salutation>.
- 4) Přetáhneme položku <LastName> na jedno místo za položku <FirstName>.

```
<Salutation> <FirstName> <LastName>
<Street> <No>
<Town> <Postcode>
```

Dear <Salutation> <FirstName> <LastName>.

Obrázek 396: Aktualizovaný pozdrav

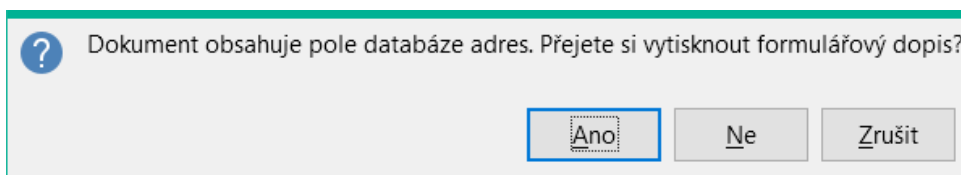
Vybereme horní záznam v okně Zdroj dat na jeho začátku. Poté klepneme na tlačítko Data na text a zobrazíme data zadaná do polí.

```
Mr. Bert Lederstrumpf
Neuenkirchener Str. 72
Pusemuckel 45793
```

Dear Mr. Bert Lederstrumpf,

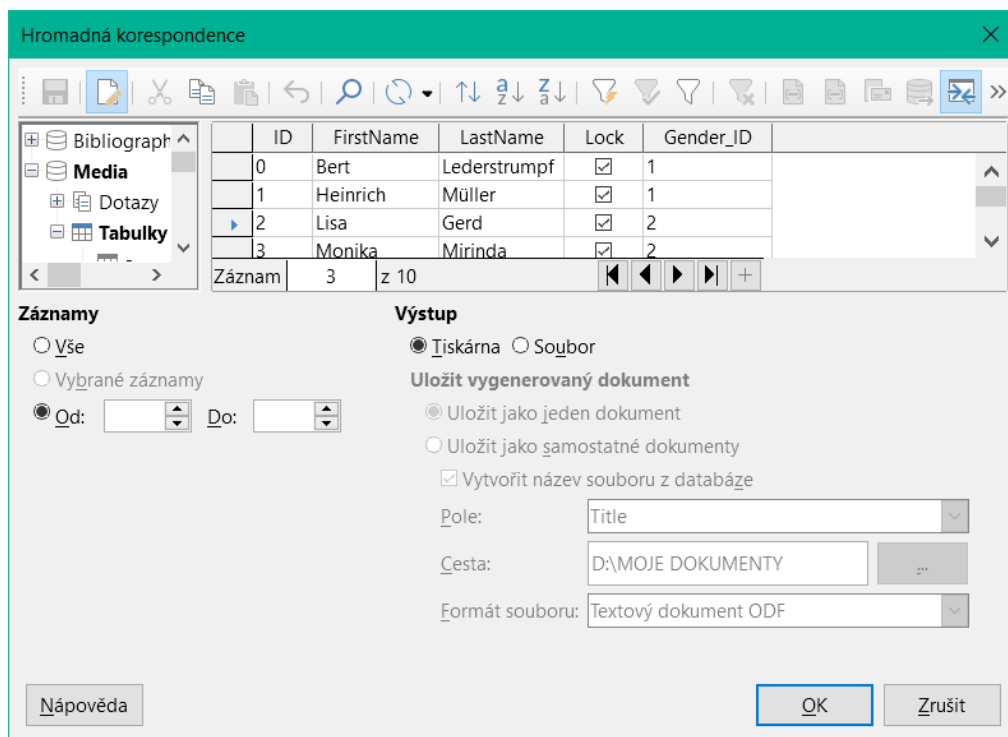
Obrázek 397: Data z prvního záznamu zdroje dat zobrazeného v dokumentu Writeru

Nyní máme dokument Writeru, do kterého můžeme psát obsah dopisu. Chceme-li sloučit pole a vytisknout dopisy, zvolíme nabídku **Soubor > Tisk**. Zobrazí se následující zpráva. Stiskneme tlačítko **Ano**.



Obrázek 398: Dialog pro potvrzení tisku dopisů hromadné korespondence

Nyní se zobrazí dialogové okno Hromadná korespondence (obrázek 399), kde můžeme volitelně vybrat záznamy, které chceme zahrnout nebo vyloučit, a zvolit tisk dopisů nebo jejich uložení do souboru. Další podrobnosti nalezneme v kapitole 14, Hromadná korespondence, v příručce *Průvodce aplikací Writer*.



Obrázek 399: Dialogové okno Hromadná korespondence

Tisk štítků

Soubor > Nový > Štítky spustí Průvodce štítky. Otevře dialogové okno, které zahrnuje všechny otázky formátování a obsahu štítků, a to ještě předtím, než se vytvoří samotné štítky. Nastavení v tomto dialogovém okně se uloží do osobních nastavení uživatele.

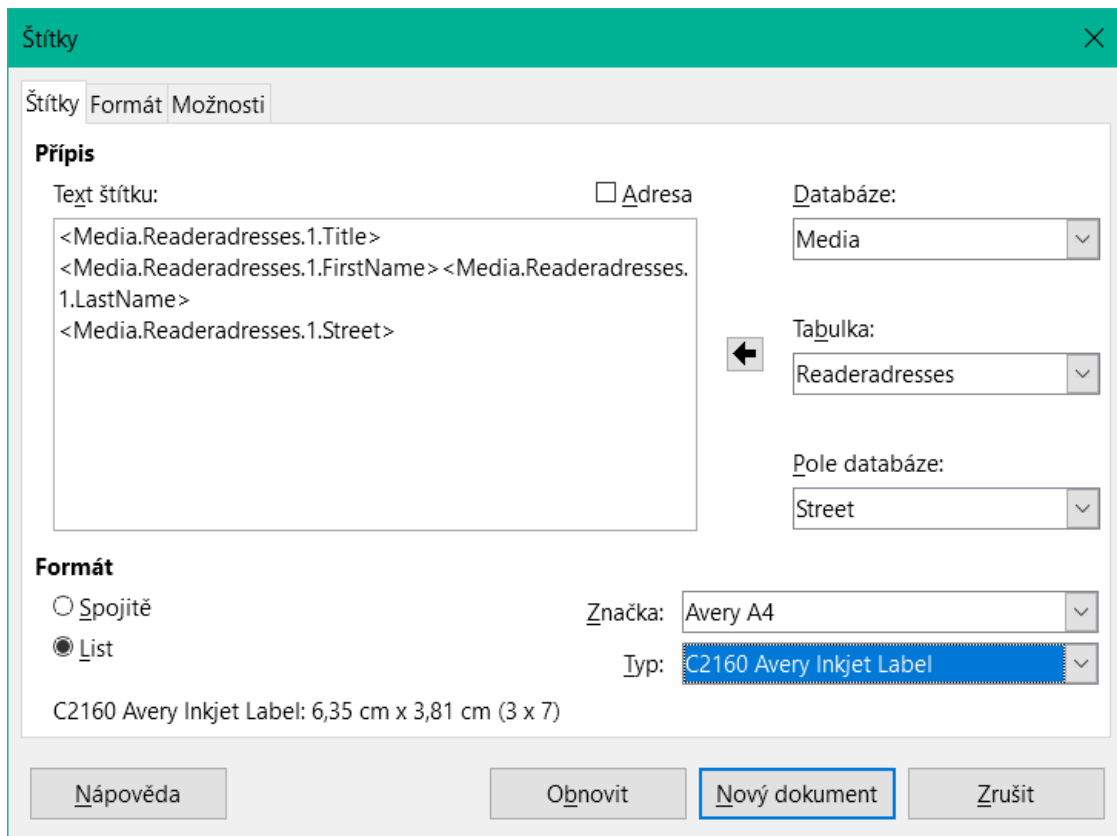
Základní nastavení obsahu se nachází na kartě Popisky (obrázek 400). Pokud u textu štítku zaškrtneme políčko Adresa, budou mít všechny štítky stejný obsah převzatý z nastavení LibreOffice pro uživatele programu.

Jako příklad opět použijeme databázi Addresses. Přestože další výběrové pole má nadpis Tabulky, jsou zde uvedeny jak tabulky, tak dotazy, stejně jako v prohlížeči zdrojů dat.

Pomocí tlačítek se šipkami můžeme do editoru vkládat jednotlivá pole databáze. Název databázového pole Surname je zde nastaven na <Addresses.MailMergeQuery.1.Surname>. Sekvence je tedy <database.Table.1.database field>.

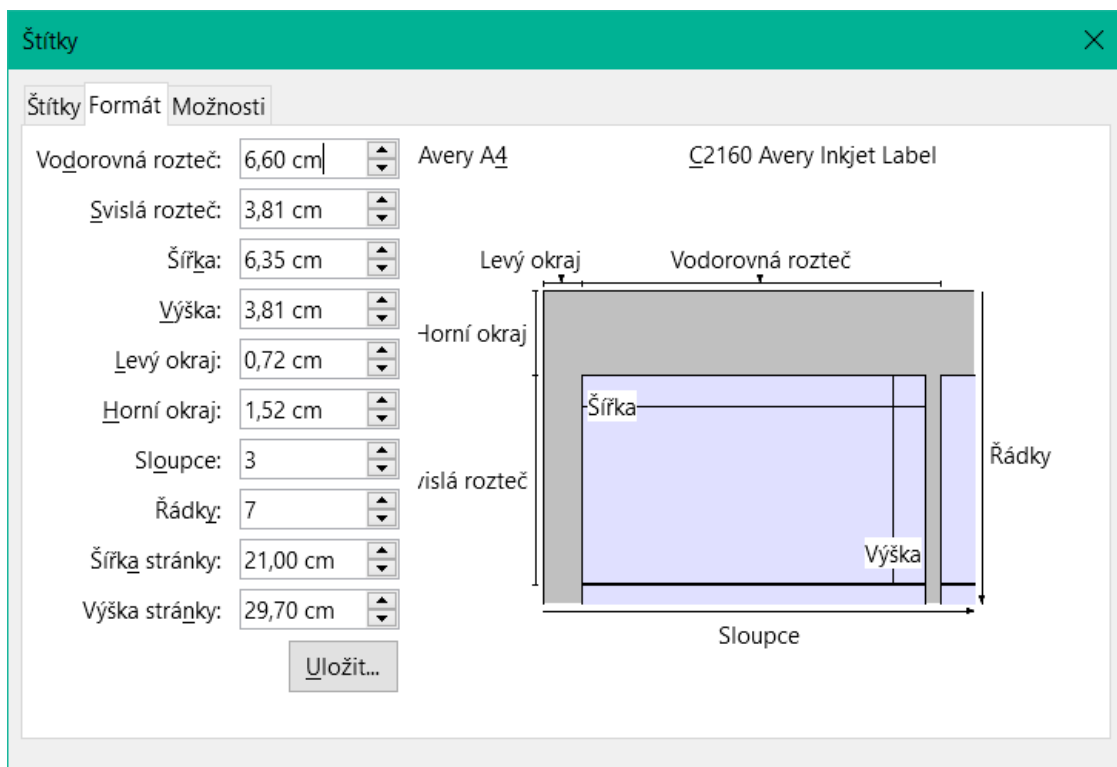
V editoru můžeme pracovat s klávesnicí. Tak například můžeme na začátek vložit zalomení řádku, aby se štítky netiskly přímo na horní okraj, ale aby byl obsah vytištěn zcela a jasně viditelný.

Formát lze vybrat na kartě Štítky. Zde je začleněno velké množství značek štítků, takže většina ostatních nastavení na kartě Formát není nutná.

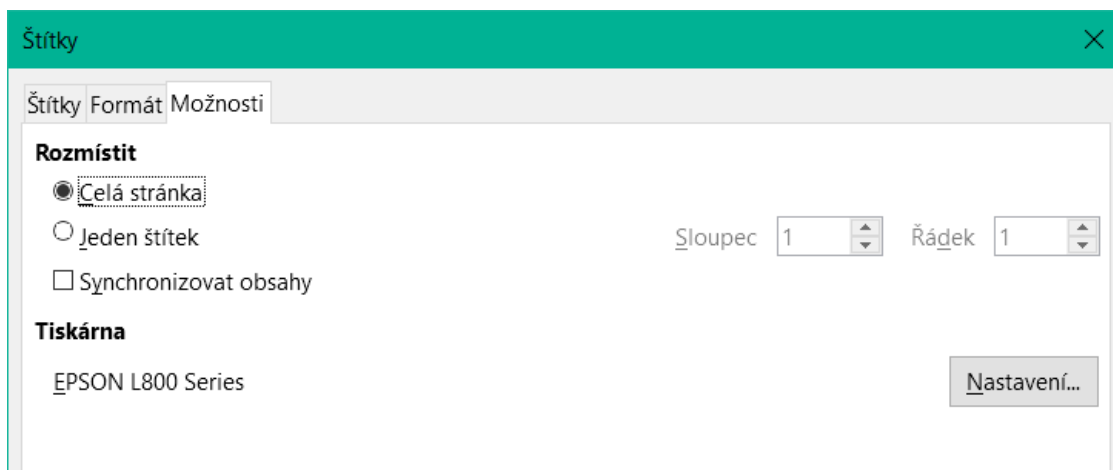


Obrázek 400: Dialogové okno Štítky, karta Štítky

Pomocí karty Formát (obrázek 401) můžeme přesně nastavit velikost štítku. Nastavení má význam pouze v případě, že není známa značka a typ štítků. Pro tisk štítků o šířce 7,00 cm potřebujeme šířku stránky o něco větší než $3 \times 7,00 \text{ cm} = 21,00 \text{ cm}$. Teprve pak se na stránce vytisknou tři štítky za sebou.



Obrázek 401: Dialogové okno Štítky, karta Formát



Obrázek 402: Dialogové okno Štítky, karta Možnosti

Na kartě Možnosti můžeme určit, zda se má vytvořit pouze jeden štítek, nebo celá stránka štítků. Stránka se pak vyplní postupně údaji ze záznamů databáze, počínaje prvním záznamem. Pokud je záznamů více, než se vejde na stránku, další stránka se automaticky zaplní další sadou záznamů.

Zaškrtnutí políčko Synchronizovat obsahy propojí všechny štítky dohromady, takže následné změny v rozvržení kteréhokoli štítku se použijí na všechny ostatní štítky. Chceme-li přenést upravený obsah, použijeme tlačítko Synchronizovat, které se zobrazí při vytváření štítku, pokud jsme toto políčko zaškrtnuli.

Tlačítkem **Nový dokument** vytvoříme dokument obsahující vybraná pole.

Po zahájení procesu tisku se zobrazí potvrzovací dialogové okno znázorněné na obrázku 398 (jako u dopisů hromadné korespondence). Výběrem možnosti **Ano** vyplníme pole databáze adres odpovídajícím obsahem.

Zdroj dat pro tisk štítků není vyhledáván automaticky, pouze je předem vybrána databáze. Vlastní dotaz musí zadat uživatel, protože v tomto případě se nejedná o tabulku.

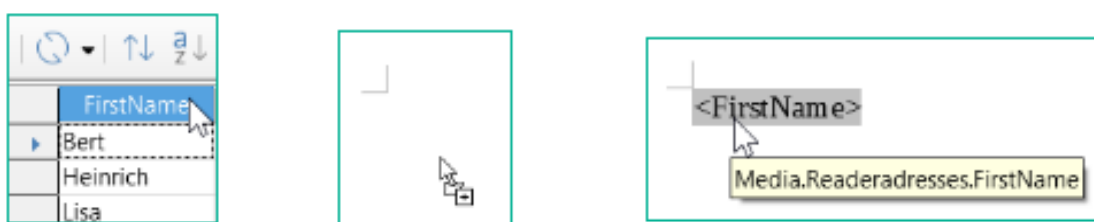
Po výběru dotazu a příslušných záznamů (v tomto případě *Všechny*) lze zahájit tisk. Zejména při prvních testech je vhodné v dialogovém okně Hromadné korespondence zvolit *Výstup do souboru* (obrázek 399), čímž se štítky uloží jako dokument. Možnost uložení do několika dokumentů není vhodná pro tisk štítků, ale spíše pro dopisy různým příjemcům, se kterými lze následně pracovat.

Přímé vytváření dokumentů hromadná korespondence a štítků

Místo Průvodce můžeme vytvářet dokumenty hromadné korespondence a štítky přímo.

Hromadná korespondence pomocí myši

Pole pro hromadnou korespondenci lze přebírat z prohlížeče databáze pomocí myši.

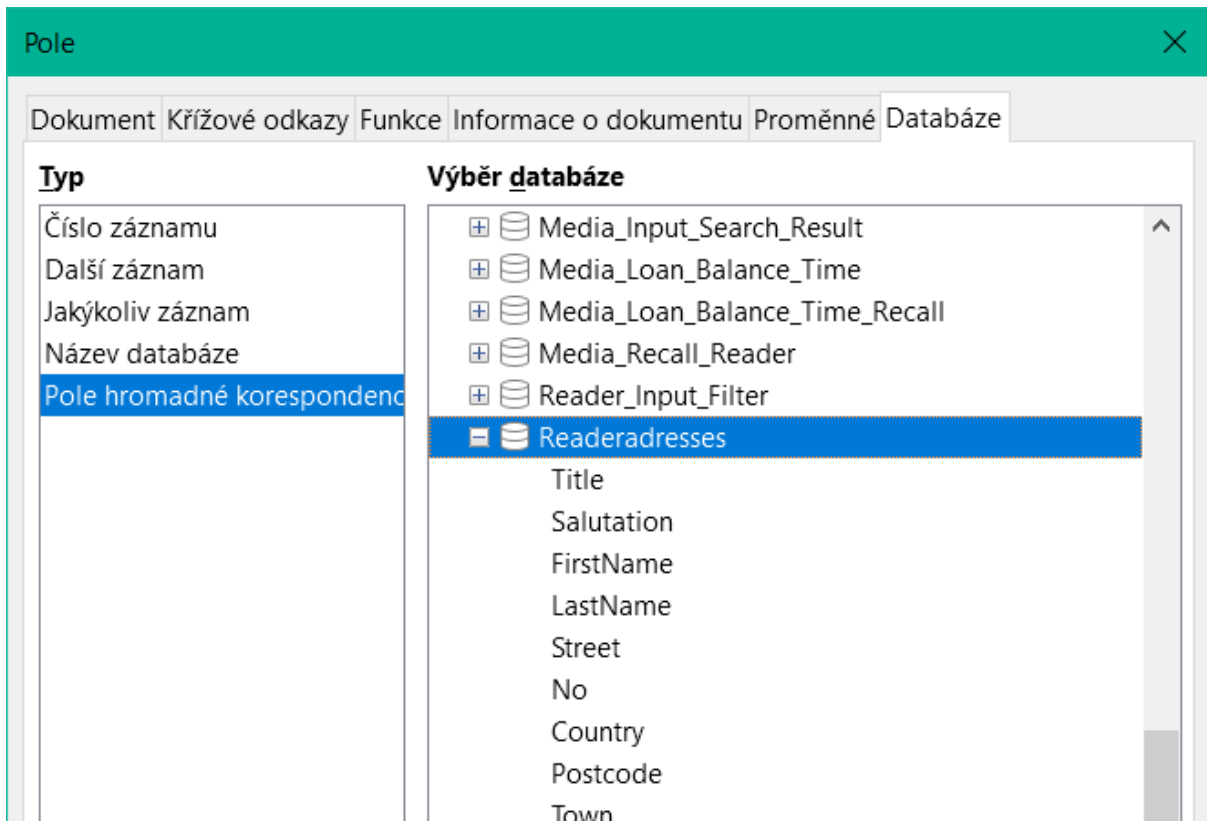


Obrázek 403: Použití myši k vytvoření hromadné korespondence

Levým tlačítkem myši vybereme záhlaví tabulky. Podržíme tlačítko stisknuté a přetáhneme kurzor v textovém dokumentu. Kurzor změní svůj tvar na symbol vložení. Pole MailMerge je vloženo do textového dokumentu, zde je zobrazeno v úplném popisu, který je viditelný pomocí **Zobrazit > Názvy polí**.

Vytváření hromadných dopisů výběrem polí

Pole pro hromadnou korespondenci lze vložit pomocí **Vložit > Pole > Další pole > Databáze**.



Obrázek 404: Použití dialogového okna Pole, karta Databáze pro vložení polí pro hromadnou korespondenci

Zde jsou k dispozici všechny tabulky a dotazy ve vybrané databázi. Pomocí tlačítka **Vložit** můžeme vkládat různá pole jedno po druhém přímo do textu na aktuální pozici kurzoru.

Pokud chceme vytvořit pozdrav, který je v dopisech hromadné korespondence obvyklý, můžeme použít skrytý odstavec nebo skrytý text: **Vložit > Pole > Další pole > Funkce > Skrytý odstavec**. U obou variant dbáme na to, aby nebyla splněna námi formulovaná podmínka, protože chceme, aby byl odstavec viditelný.

Aby se formule Vážená paní <Surname> objevila pouze v případě, že se jedná o ženu, postačí splnit následující podmínku:

```
[Media.Readeraddresses.Salutation] != "Mrs."
```

Jediný problém, který zbývá, je, že možná neexistuje žádné příjmení. Za těchto okolností by se mělo objevit "Vážený pane/paní", takže tuto podmínku musíme vložit. Celkový výraz vypadá takto:

```
[Media.Readeraddresses.Salutation] ! "Mrs." OR NOT  
[Media.Readeraddresses.Salutation]
```

To vylučuje možnost, že by se tento odstavec objevil v případě, že se nejedná o ženu nebo není uvedeno příjmení.

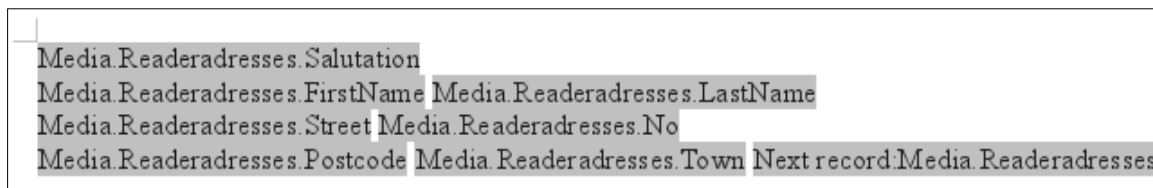
Stejným způsobem můžeme vytvořit položky pro mužský rod a chybějící položky pro zbývající dva typy pozdravů.

Pozdrav v poli adresy lze samozřejmě vytvořit úplně stejným způsobem, ať už je pohlaví uvedeno kdekoli.

Další informace jsou uvedeny v nápovědě LibreOffice v části Skrytí textu a *Podmíněný text*.

Samozřejmě by bylo ještě jednodušší, kdyby někdo, kdo rozumí databázím, vložil celý pozdrav přímo do dotazu. To lze provést pomocí korelovaného poddotazu (viz kapitola 5, Dotazy).

Pro štítky je zajímavý zejména typ pole **Další záznam**. Pokud je tento typ pole zvolen na konci štítku, bude následující štítek vyplněn daty z následujícího záznamu. Typické štítky pro sekvenční tisk štítků vypadají jako na následujícím obrázku, když pomocí **Zobrazení > Názvy polí** zviditelníme příslušná označení polí:



Obrázek 405: Typický výběr polí pro sekvenční tisk štítků

U posledního štítku na stránce je třeba zohlednit skutečnost, že po přerušení stránky se automaticky vyvolá další záznam. Zde by se typ pole *Další záznam* neměl vyskytovat. V opačném případě dojde k vynechání záznamu, protože dojde k dvojímu skoku záznamu.

Tip

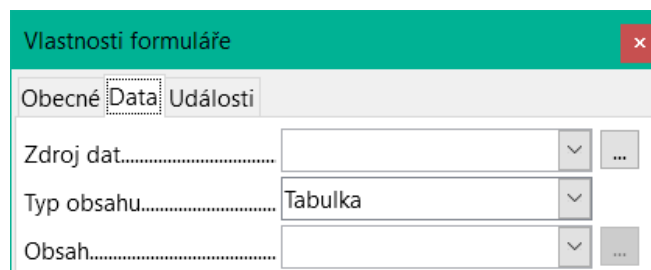
Vytváření dopisů pro hromadnou korespondenci je možné také přímo z databázového formuláře. Jediným požadavkem je, aby byla databáze zaregistrována v LibreOffice.

Při tvorbě hromadné korespondence nezapomeňte zvolit **Zobrazení > Normální**. Tím se zajistí správné umístění prvků na stránce. Pokud je pak formulář vytištěn, zobrazí se obvyklý dotaz pro hromadnou korespondenci.

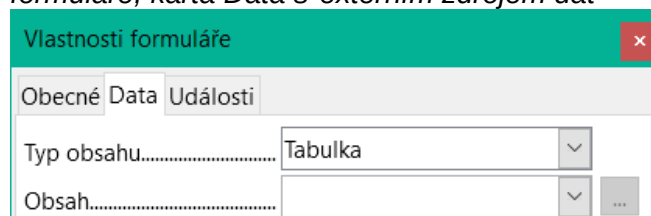
Tento typ hromadné korespondence má tu výhodu, že k tisku nepotřebujeme žádné jiné soubory než soubor *.odb.

Externí formuláře

Pokud mají být jednoduché vlastnosti formuláře dostupné v LibreOffice použity v jiných komponentách, jako jsou Writer a Calc, stačí zobrazit nástrojovou lištu Návrh formuláře pomocí **Zobrazit > Nástrojové lišty > Návrh formuláře** a poté otevřít Navigátor formulářem. Můžeme vytvořit formulář, nebo, jak je popsáno v kapitole 4, Formuláře, vytvořit formulářové pole. Karta Data v dialogovém okně Vlastnosti formuláře vypadá trochu jinak než při vytváření formulářů přímo v databázovém souboru ODB.



Obrázek 406: Horní část dialogu Vlastnosti formuláře, karta Data s externím zdrojem dat



Obrázek 407: Horní část dialogu Vlastnosti formuláře, karta Data s interním zdrojem dat.

Při použití externího formuláře je nutné zvolit zdroj dat samostatně. Tlačítkem vpravo od pole se seznamem zdrojů dat otevřeme prohlížeč souborů. Lze vybrat libovolný soubor ODB. Pole pro zdroj dat navíc obsahuje odkaz začínající `file:///`.

Pokud se místo toho podíváme do obsahu seznamu, uvidíme databáze, které jsou již v LibreOffice zaregistrovány pod svými registrovanými názvy.

Formuláře se vytvářejí úplně stejným způsobem jako v samotné aplikaci Base.

Takto vytvořené formuláře se ve výchozím nastavení zobrazují při každém otevření souboru v režimu úprav, nikoliv chráněné proti zápisu jako v aplikaci Base. Abychom zabránili náhodné úpravě formuláře, můžeme soubor otevřít pouze pro čtení pomocí **Soubor > Vlastnosti > Zabezpečení**. Soubor můžeme dokonce chránit před změnou pomocí hesla. V kancelářských systémech je také možné prohlásit celý soubor za chráněný proti zápisu. To stále umožňuje vstup do polí formuláře, ale ne pohyb v polích nebo zadávání textu mezi nimi.

Tip

Formuláře lze také rychle vytvářet přetažením. Za tímto účelem otevřeme databázi, vyhledáme příslušnou tabulku nebo dotaz a vybereme záhlaví tabulky.

V aplikaci Writer vybereme pomocí levého tlačítka myši příslušné nadpisy polí, podržíme stisknuté klávesy Shift a Ctrl a kurzor myši se změní na symbol odkazu. Poté přetáhneme nadpisy do dokumentu Writer.

Pole můžeme přetáhnout do souborů Calc bez použití dalších kláves. Symbol kopírování se zobrazí jako kurzor myši.

V obou případech se vytvoří vstupní pole s přiřazeným štítkem. Odkaz na zdroj dat se vytvoří při prvním skutečném zadání dat, takže zadávání dat do takového formuláře může začít ihned po operaci přetažení.

Výhody externích formulářů

Pro práci s databází není nutné nejprve otevřít aplikaci Base. Proto nepotřebujeme další otevřené okno na pozadí.

V již hotové databázi lze stávajícím uživatelům databáze následně bez problémů zaslat vylepšený formulář. Při vývoji dalších formulářů mohou nadále používat databázi a nemusí kopírovat složité externí formuláře z jedné databáze do druhé.

Formuláře pro databázi lze měnit podle potřeb uživatele. Uživatelům, kteří nemají oprávnění opravovat data nebo vytvářet nové záznamy, mohou ostatní uživatelé zaslat aktuální soubor dat a jednoduše nahradit svůj soubor *.odb, aby měli k dispozici aktuální zobrazení. To by mohlo být užitečné například pro databázi organizace, kde všichni členové výboru dostanou databázi, ale pouze jedna osoba může údaje upravovat; ostatní mohou stále prohlížet adresy svých oddělení.

Nevýhody externích formulářů

Uživatelé musí vždy instalovat formuláře a aplikaci Base se stejnou adresářovou strukturou. Jen tak může být přístup do databáze bez chyb. Vzhledem k tomu, že odkazy jsou uloženy relativně k formulářům, stačí uložit databázi a její formuláře do společného adresáře.

Externě lze vytvářet pouze formuláře, nikoli dotazy nebo sestavy. Pouhý pohled na dotaz proto musí projít formulářem. Naproti tomu sestava vyžaduje otevření databáze. Případně by bylo možné ji alespoň částečně vytvořit pomocí hromadné korespondence.

Použití databáze v aplikaci Calc

Data lze použít v aplikaci Calc pro účely výpočtu. Za tímto účelem je nejprve nutné zpřístupnit data v listu aplikace Calc.

Zadávání dat do aplikace Calc

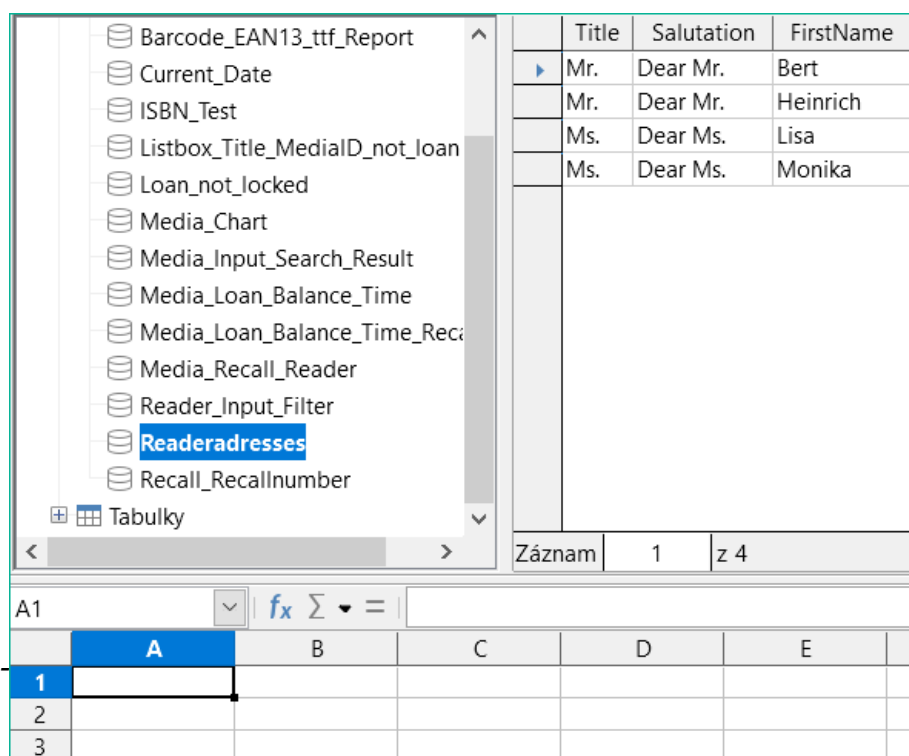
Do aplikace Calc lze zadávat data různými způsoby.

Vybereme tabulku levým tlačítkem myši a přetáhneme ji do pracovního listu aplikace Calc. Kurzor nastaví levý horní roh tabulky. Vytvoří se tabulka s názvy polí. Prohlížeč zdrojů dat v tomto případě nenabízí možnosti Data na text nebo Data na pole.

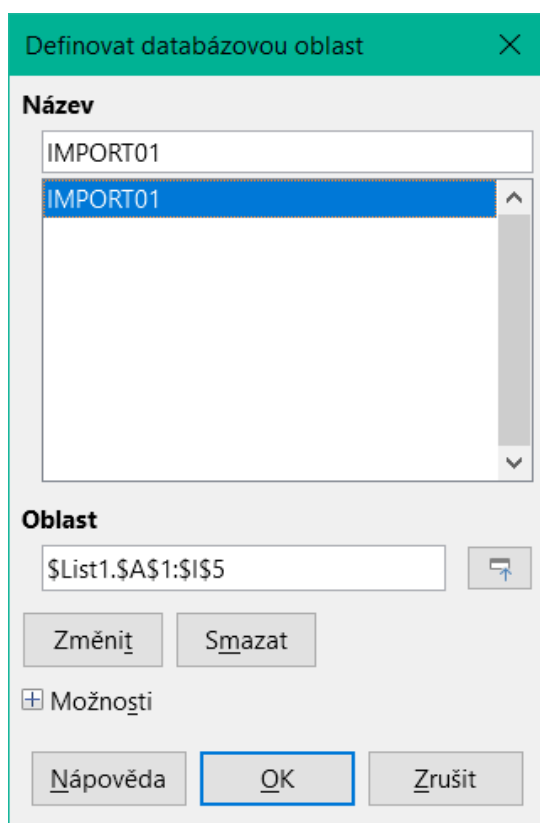
Data přetažená do aplikace Calc tímto způsobem vykazují následující vlastnosti:

Při importu se neimportují pouze data, ale také vlastnosti polí, které se během importu načítají a působí na ně. Pole, jako jsou čísla domů, která byla deklarována jako textová pole, jsou po vložení do Calc formátována jako text.

Z importu se stane oblast aplikace Calc, které je ve výchozím nastavení přiřazen název Import1. K datům lze později přistupovat pomocí této oblasti. Volba **Data > Obnovit oblast** umožňuje, aby byl rozsah případně doplněn o nová data z databáze.



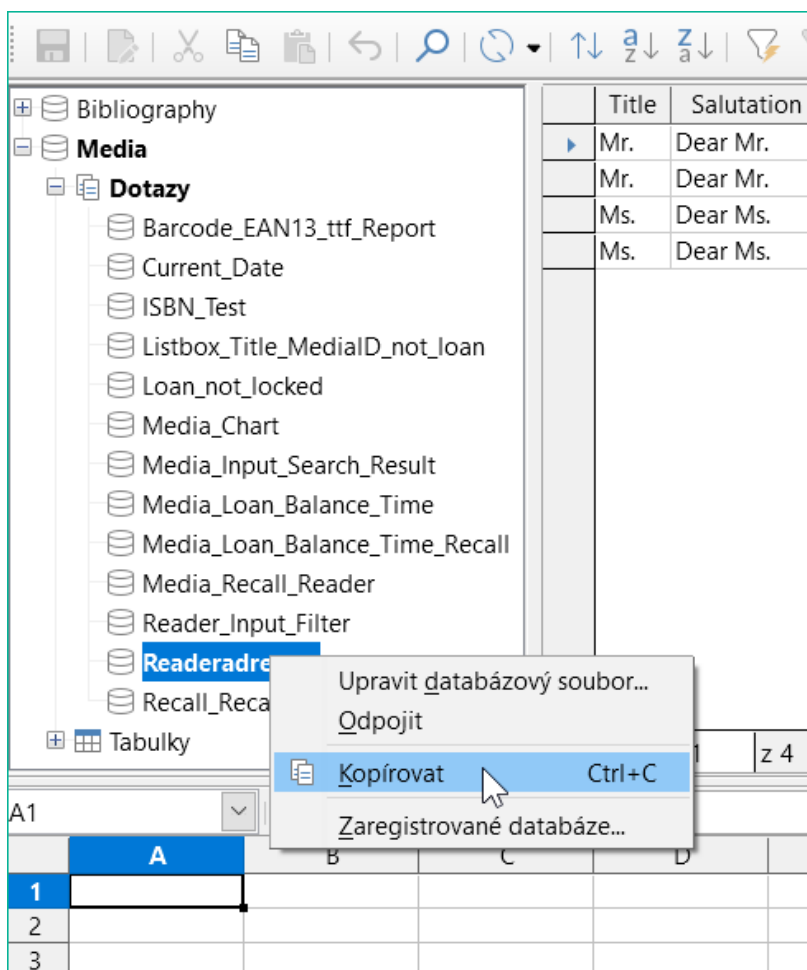
Obrázek 408: Přetažení tabulky zdroje dat do listu Calc



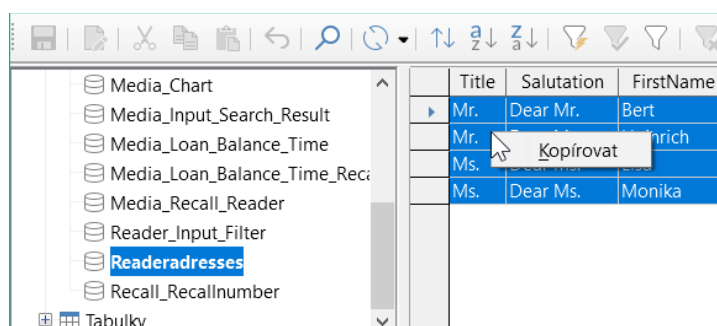
Obrázek 409: Dialogové okno Definice databázové oblasti zobrazující automaticky vytvořenou databázovou oblast Import1

Importovaná data nejsou formátována jinak, než jak to vyžadují vlastnosti databázových polí.

Pomocí místní nabídky tabulky můžeme také vytvořit *kopii* dat. V tomto případě však nejde o import, ale pouze o kopii. Vlastnosti datových polí nejsou načítány společně s nimi, ale jsou určeny aplikací Calc. Kromě toho jsou názvy polí formátovány jako záhlaví tabulky.



Obrázek 410: Kopírování tabulky z levé strany prohlížeče zdrojů dat



Obrázek 411: Kopírování tabulky z pravé strany prohlížeče zdrojů dat

Rozdíl je vidět zejména v databázových polích, která jsou formátována jako text. Při importu je aplikace Calc změní na textová pole, která jsou zarovnána vlevo jako ostatní text. Tato čísla pak již nelze použít při výpočtech.

Pokud je znovu exportujeme, zůstanou data zachována v původní podobě.

	A	B	C	D
1	Import		Copy	
2	Street	No	Street	No
3	Neuenkirchener Str.	72	Neuenkirchener Str.	72
4	Nowhereroad	14 b	Nowhereroad	14 b
5	Berliner Allee	364	Berliner Allee	364
6	Bäckerstr.	13 a	Bäckerstr.	13 a
7				

Obrázek 412: Porovnání importovaných a zkopírovaných dat v aplikaci Calc

i Tip

Importem dat do aplikace Calc se přepíše předchozí obsah a také veškeré předchozí formátování. Pokud mají být data důsledně exportována do stejné tabulky, měli bychom pro import dat použít samostatný list. Data se pak načtou do druhého listu pomocí výrazu jménotabučky.názevpo le. Pole v tomto listu lze vhodně naformátovat bez rizika přepsání formátování.

i Tip

Záznamy lze také kopírovat přímo z databáze pomocí schránky nebo přetažením myší. Pokud je tabulka nebo dotaz přetažen do tabulky aplikace Calc, vloží se celý obsah. Pokud je otevřena tabulka nebo dotaz a je vybrán jeden nebo více záznamů, zkopírují se po přetažení pouze tyto záznamy spolu s názvy polí.

Export dat z aplikace Calc do databáze

Vybereme data v listu aplikace Calc. Podržíme stisknuté levé tlačítko myši a přetáhneme data, která chceme převést do databáze, do oblasti tabulky v prohlížeči databází.

	Title	Salutation	FirstName
▶	Mr.	Dear Mr.	Bert
	Mr.	Dear Mr.	Heinrich
	Ms.	Dear Ms.	Lisa
	Ms.	Dear Ms.	Monika
Záznam 1 z 4			

	A	B	C	D	E	F	G
1	ID	Title	FirstName	LastName	Street	No	C
2	0	Mr.	Bert	Lederstrumpf	Neuenkirchener Str.	72	D
3	1	Mr.	Heinrich	Müller	Nowhereroad	14 b	GE
4	2	Ms.	Lisa	Gerd	Berliner Allee	364	D
5	3	Ms.	Monika	Mirinda	Bäckerstr.	13 a	D
6							

Obrázek 413: Kopírování dat z Calc do databázové tabulky

Kurzor změni svůj vzhled a ukáže, že je možné něco vložit.

Otevře se první okno Průvodce importem. Další kroky průvodce jsou popsány v kapitole 3, Tabulky, v části „Import dat z jiných zdrojů“.

Převod dat z jedné databáze do druhé

V průzkumníku prohlížeče zdrojů dat lze tabulky kopírovat z jedné databáze do druhé tak, že vybereme zdrojovou tabulku levým tlačítkem myši, podržíme tlačítko stisknuté a přetáhneme ji do cílové databáze v kontejneru tabulek. Tím se zobrazí dialogové okno pro kopírování tabulek.

Tímto způsobem lze například databáze určené pouze pro čtení (zdroje dat, jako jsou adresáře z programu elektronické pošty nebo tabulky v tabulkovém procesoru) použít jako základ pro databázi, ve které jsou data editovatelná. Data lze také přímo kopírovat při přechodu na jiný databázový program (například při přechodu z PostgreSQL na MySQL).

Pokud chceme, aby nová databáze měla jiné relace než původní, můžeme to zařídit pomocí vhodných dotazů. Ti, kteří nejsou dostatečně odborně zdatní, mohou místo toho použít aplikaci Calc. Stačí přetáhnout data do tabulky a připravit je k importu do cílové databáze pomocí prostředků, které aplikace Calc nabízí.

Aby byl import do nové databáze co nejčistší, měly by být tabulky připraveny předem. Díky tomu lze s dostatečným předstihem rozpoznat problémy s formátováním a problémy s vytvářením primárních klíčů.

Import záznamů do tabulky pomocí schránky

Pokud jsou záznamy k dispozici v tabulkové podobě, lze je vložit do databáze aplikace Base pomocí schránky a průvodce.

V aplikaci Base se import zahájí klepnutím pravým tlačítkem myši na cílovou tabulku. V místní nabídce pod volbou **Kopírovat** jsou příkazy **Importovat** a **Importovat obsah**. Pokud vybereme **Vložit**, Průvodce importem již vybere tabulku a *Připojit data*. **Vložit jinak** poskytuje pouze dotaz na importní filtr. K dispozici jsou možnosti HTML a RTF.

Pokud místo toho klepneme pravým tlačítkem myši do kontejneru tabulky, Průvodce importem nám nabídne pouze možnost vytvořit novou tabulku.

Importování PDF záznamů

Pokud chceme importovat data z různých externích zdrojů, je nejlepší zvolit formát, který zabrání úpravám formuláře během zadávání dat. Pomocí Writeru můžeme vytvářet formuláře ve formátu PDF, umístit je online a nechat si vyplněné formuláře vrátit, například jako přílohy e-mailu. Chybí jen co nejjednodušší zadání údajů do databáze aplikace Base. Příklad ilustruje takový způsob importu.



Poznámka

Databáze Example_PDFFormular_Import.odt pro tuto sestavu je obsažena v ukázkových databázích k této knize.

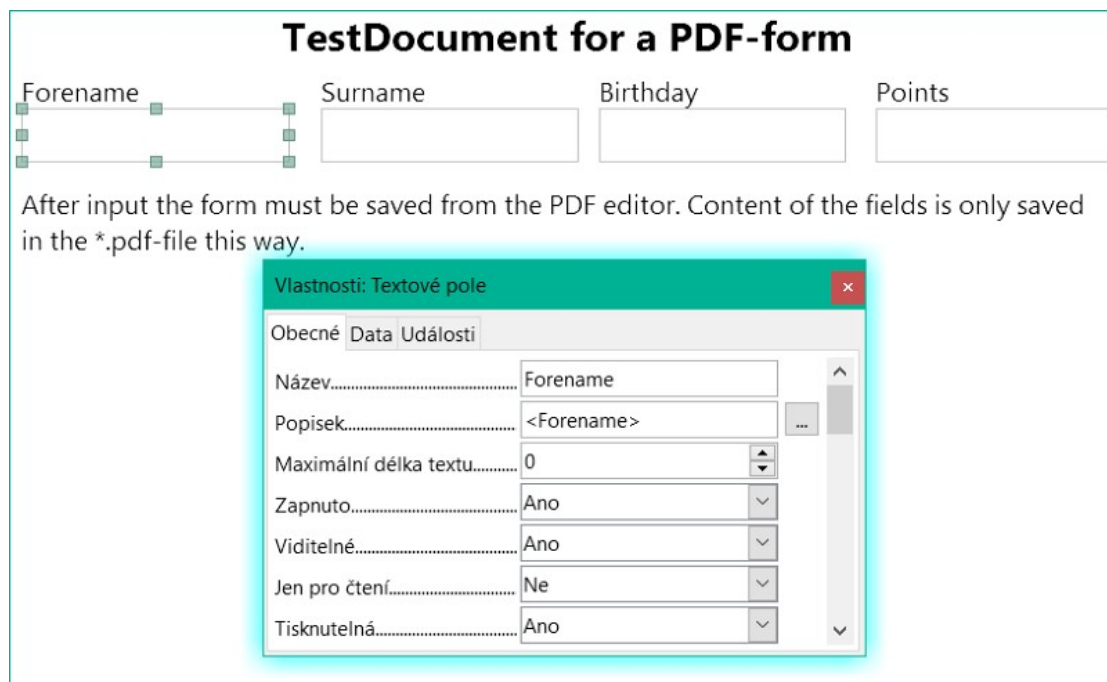
Vytvoření PDF formuláře

Formulář PDF je vytvořen jako externí formulář bez propojení s databází. Pomocí **Zobrazit > Nástrojové lišty > Ovládací prvky formuláře** se zobrazí potřebné prvky formuláře, které lze vložit podle potřeby.

Formát PDF bohužel nerozlišuje mezi číselnými poli, daty a textovými poli. Pro zde uvedený příklad stačí pro všechny položky použít textová pole. Ostatní formáty polí ve formuláři aplikace Writer budou při exportu do PDF nevyhnutelně ztraceny.

Formuláře PDF mohou mít v zásadě následující pole:

- Tlačítka
- Textová pole
- Zaškrťovací políčka
- Rozevírací seznamy
- Seznamy



Obrázek 414: Testovací dokument PDF, který má být načten do databáze Base

Testovací formulář obsahuje celkem 4 textová pole. V části **Vlastnosti: Textové pole > Obecné > Název** při použití následující metody importu bychom měli vždy zvolit název pole použitý v databázové tabulce, abychom se vyhnuli problémům s názvy polí a jejich obsahem.

Nápovědy se zobrazují při čtení záznamů, ale nezobrazují se v každém prohlížeči PDF.

Aby bylo zajištěno, že formulář skutečně obsahuje záznamy, měl by být po zadání dat uložen v prohlížeči PDF pomocí možnosti nabídky **Soubor > Uložit jako**. Konkrétní příkaz pro provedení tohoto úkonu se může u různých prohlížečů lišit. Bez tohoto postupu prohlížeč zobrazí záznamy po otevření formuláře v našem počítači, ale ve skutečnosti je načte z dočasně uloženého souboru prohlížeče, nikoli přímo ze souboru PDF. Pokud je pak formulář přenesen do jiného počítače, bude prázdný.

Čtení záznamů z PDF formuláře

Formulář pro databázi aplikace Base je vzhledově velmi jednoduchý. Je propojen s tabulkou a zobrazuje právě načtené záznamy. Nejnovější záznamy jsou uvedeny v kontrolní tabulce výše.

	ID	Forename	Surname	Birthday	Poits
	40	Geraldine	Much Wind	31.01.01	14,40
	41	Bill	Bigbox	23.11.87	17,34
+	pole>				
Záznam 1 z 2					

Import PDF-form

Obrázek 415: Data formuláře načtená do aplikace Base z dokumentu PDF

Makro pro načítání záznamů se zadává v části **Vlastnosti: Tlačítko > Události > Provést akci**.

K vyčítání záznamů používáme open source program `pdftk`. Program je volně dostupný pro Windows i Linux. Linuxové distribuce jej mají většinou jako balíček ve svých repozitářích. Uživatelé systému Windows jej najdou na adrese <https://www.pdfabs.com/tools/pdftk-the-pdf-toolkit/>.

Záznamy načtené pomocí `pdftk` se zapisují do textového souboru, který vypadá takto:

```

1 ---
2 FieldType: Text
3 FieldName: forename
4 FieldFlags: 0
5 FieldValue: Bill
6 FieldJustification: Left
7 ---
8 FieldType: Text
9 FieldName: surname
10 FieldFlags: 0
11 FieldValue: Bigbox
12 FieldJustification: Left
13 ---
14 FieldType: Text
15 FieldName: birthday
16 FieldNameAlt: Date, year minimum 2 places
17 FieldFlags: 0
18 FieldValue: 11/23/87
19 FieldJustification: Left
20 ---
21 FieldType: Text
22 FieldName: points
23 FieldNameAlt: Decimal value, 2 decimal places
24 FieldFlags: 0
25 FieldValue: 17.34
26 FieldJustification: Left
27

```

Každé pole je v souboru zastoupeno pěti až šesti řádky. Pro makro jsou důležité řádky **FieldName** (měl by být stejný jako `FieldName` v cílové tabulce), **FieldValue** (obsah pole po uložení souboru PDF) a **FieldJustification** (poslední řádek položky).

Celý proces importu je řízen pomocí maker. Formulář PDF musí být uložen ve stejné složce jako databáze. Záznamy se z něj načtou do textového souboru a poté se z něj načtou do databáze. Takto se pokračuje u všech souborů PDF ve složce. Staré záznamy by proto měly být ze složky pokud možno odstraněny, protože funkce nekontroluje duplicitu.

```

SUB PDF_Form_Import(oEvent AS OBJECT)
    DIM inNumber AS INTEGER
    DIM stRow AS STRING
    DIM i AS INTEGER
    DIM k AS INTEGER
    DIM oDatasource AS OBJECT

```

```

DIM oConnection AS OBJECT
DIM oSQL_Command AS OBJECT
DIM oResult AS OBJECT
DIM stSql AS STRING
DIM oDB AS OBJECT
DIM oFileAccess AS OBJECT
DIM inFields AS INTEGER
DIM stFieldName AS STRING
DIM stFieldValue AS STRING
DIM stFieldType AS STRING
DIM stDir AS STRING
DIM stDir2 AS STRING
DIM stPDFForm AS STRING
DIM stFile AS STRING
DIM stTable AS STRING
DIM inNull AS INTEGER
DIM aFiles()
DIM aNull()
DIM stCommand AS STRING
DIM stParameter AS STRING
DIM oShell AS OBJECT

```

Po deklaraci proměnných je uveden počet polí ve formuláři PDF. Počet začíná od 0, takže hodnota 3 ve skutečnosti znamená celkem čtyři pole. Pomocí tohoto počtu lze zjistit, zda byla načtena všechna data pro daný záznam, takže je připraven k přenosu do tabulky.

```

inFields = 3
stTable = "Name"
oDatasource = ThisComponent.Parent.CurrentController
If NOT (oDatasource.isConnected()) THEN
    oDatasource.connect()
END IF
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()

```

Připojení k databázi je vytvořeno. Čte se cesta k databázovému souboru v souborovém systému. Pomocí této cesty se obsah složky načte do pole aFiles. Smyčka kontroluje každý název souboru v poli, zda nekončí příponou .pdf. Velká a malá písmena se nerozlišují, protože všechny výsledky hledání jsou převedeny na malá písmena pomocí Lcase.

```

oDB = ThisComponent.Parent
stDir = Left(oDB.Location, Len(oDB.Location) - Len(oDB.Title))
oFileAccess = createUnoService("com.sun.star.ucb.SimpleFileAccess")
aFiles = oFileAccess.getFolderContents(stDir, False)
FOR k = 0 TO uBound(aFiles())
    IF LCase(Right(aFiles(k), 4)) = ".pdf" THEN
        stDir2 = ConvertFromUrl(stDir)
        stPDFForm = ConvertFromUrl(aFiles(k))
    END IF
NEXT k

```

Pro určení příkazu pro načtení dat je nutné znát konvence adresování souborů v operačním systému. Původní adresa URL začínající na file:// se proto musí přizpůsobit aktuálnímu systému. Příkaz pro spuštění programu pdftk závisí na operačním systému. Může mít příponu .exe nebo třeba úplnou cestu k programu, například `.C:\Program Files (x86)\pdftk\pdftk.exe` nebo přípona nemusí být vůbec vyžadována. GetGuiType slouží k určení typu používaného systému: 1 znamená Windows, 3 macOS a 4 Linux. Následující kroky rozlišují pouze mezi systémem Windows a ostatními systémy.

Poté se použije funkce Shell(), která konzoli předá příslušný příkaz pro spuštění pdftk. Argument True zajistí, že LibreOffice počká, dokud se proces shellu neukončí.


```

IF GetGuiType = 1 THEN '()
    stCommand = "pdftk.exe"
ELSE
    stCommand = "pdftk"
END IF
stParameter = stPDFForm & " dump_data_fields_utf8 output "
    & stDir2 & "PDF_Form_Data.txt"
Shell(stCommand, 0, stParameter, True)
stFile = stDir & "PDF_Form_Data.txt"
i = -1
inNumber = FreeFile

```

Funkce FreeFile určuje, který další volný datový kanál je v operačním systému k dispozici. Tento kanál se načte jako celé číslo a použije se k přímému připojení k právě vytvořenému datovému souboru PDF. Ke čtení souboru se používá instrukce INPUT. To probíhá mimo kancelářský balík LibreOffice. Externí záznamy se pak načtou do LibreOffice.

```

OPEN stFile FOR INPUT AS inNumber
DO WHILE NOT Eof(inNumber)
    LINE INPUT #inNumber, stRow

```

Datový soubor PDF se nyní načítá řádek po řádku. Kdykoli se vyskytne výraz **FieldName**, je zbývající obsah řádku považován za název pole ve formuláři PDF a vzhledem ke způsobu, jakým byl formulář definován, také za název databázového pole, do kterého se mají data zapsat.

Všechny názvy polí jsou přímo kombinovány pro použití v pozdějších příkazech SQL. V praxi to znamená, že názvy polí jsou uzavřeny ve dvojitých uvozovkách a odděleny čárkami.

Kromě toho dotaz pro každý název pole určuje typ pole v tabulce. Datum a desetinné hodnoty se musí přenášet jiným způsobem než text.

```

IF instr(stRow, "FieldName: ") THEN
    IF stFieldName = "" THEN
        stFieldName = "" + mid(stRow, 12) + ""
    ELSE
        stFieldName = stFieldName & ", " +
mid(stRow, 12) + ""
    END IF
    stSql = "SELECT TYPE_NAME FROM
INFORMATION_SCHEMA.SYSTEM_COLUMNS
    WHERE TABLE_NAME = '" + stTable + "' AND
    COLUMN_NAME = '" + mid(stRow, 12) + "'"
    oResult = oSQL_Command.executeQuery(stSql)
    WHILE oResult.next
        stFieldType = oResult.getString(1)
    WEND
END IF

```

Stejně jako u názvů polí, tak i u hodnot polí. Nesmí však být uvozeny dvojitými uvozovkami, ale musí být připraveny v souladu s požadavky kódu SQL. To znamená, že text musí být uvozen jednoduchými uvozovkami, data musí být převedena tak, aby odpovídala konvencím SQL atd. K tomu slouží dodatečná externí funkce SQL_Value.

```

IF instr(stRow, "FieldValue: ") THEN
    IF stFieldValue = "" THEN
        stFieldValue = SQL_Value(mid(stRow, 13), stFieldType)
    ELSE
        stFieldValue = stFieldValue & ", " &
SQL_Value(mid(stRow, 13), stFieldType)
    END IF

```

```
END IF
```

Pokud je nalezen výraz **FieldJustification**, znamená to konec kombinovaného bloku názvu pole a vlastností. Čítač *i*, který bude následně porovnán s dříve deklarovaným čítačem polí *inFields*, se tedy zvýší o 1.

Pokud se *i* a *inFields* rovnají, lze příkaz SQL dát dohromady. Musíme však zajistit, aby se z prázdných formulářů nevytvářely prázdné záznamy. Proto se předtím kontroluje, zda jsou všechny hodnoty polí NULL. V takových případech se příkaz SQL spustí okamžitě. V opačném případě je záznam zařazen pro vložení do tabulky *Name*. Poté se proměnné vrátí na výchozí hodnoty a lze načíst další formulář PDF.

```
IF instr(stRow, "FieldJustification:") THEN
    i = i + 1
END IF
IF i = inFields THEN
    aNull = Split(stFieldValue, ",")
    FOR n = 0 TO Ubound(aNull())
        IF aNull(n) = "NULL" THEN inNull = inNull + 1
    NEXT
    IF inNull < inFields THEN
        stSql = "INSERT INTO "" + stTable + ""(" + stFieldName
+ ")"
        stSql = stSql + "VALUES (" + stFieldValue + ")"
        oSQL_Command.executeUpdate(stSql)
    END IF
    stFieldName = ""
    stFieldValue = ""
    stFieldType = ""
    i = -1
    inNull = 0
END IF
LOOP
CLOSE inNumber
```

Na konci postupu zůstane jeden soubor *PDF_Form_Data.txt*. Tento soubor je vymazán. Poté se formulář znovu načte, aby bylo možné zobrazit načtené záznamy.

```
Kill(stFile)
END IF
NEXT
oEvent.Source.Model.Parent.reload()
END SUB
```

Pokud text obsahuje znak `""`, bude tento znak při vkládání SQL považován za značku konce textu. Kód SQL pro příkaz vložení selže, pokud následuje jakýkoli další text, který není uzavřen v jednoduchých uvozovkách. Aby se tomu zabránilo, musí být každá jednoduchá uvozovka v textu maskována další jednoduchou uvozovkou. To je úkolem funkce `String_to_SQL`.

```
FUNCTION String_to_SQL(st AS STRING) AS STRING
    IF Instr(st, "") THEN
        st = Join(Split(st, ""), "")
    END IF
    String_to_SQL = st
END FUNCTION
```

Data v souboru PDF se načítají jako text. Jejich správnost nelze předem zkontrolovat.

Při psaní dat v angličtině se den, měsíc a rok oddělují tečkami nebo častěji pomlčkami. Den a měsíc mohou mít jednu nebo dvě číslice. Rok může mít dvě nebo čtyři číslice.

V kódu SQL musí data začínat čtyřmístným rokem a musí být zapsána ve formátu RRRR-MM-DD. Zadaná data proto musí projít procesem převodu.

Zadané datum je rozděleno na části den, měsíc a rok. Den a měsíc jsou opatřeny počáteční nulou a poté zkráceny na dvě číslice. Tím je ve všech případech zajištěn dvoumístný údaj.

Pokud je část roku již čtyřmístná (větší než 1000), hodnota se nemění. V opačném případě, pokud je rok větší než 30, se předpokládá, že datum patří do minulého století, a je třeba přidat rok 1900. Všechna ostatní data jsou přiřazena k aktuálnímu století.

```
FUNCTION Date_to_SQLDate(st AS STRING) AS STRING
  DIM stDay AS STRING
  DIM stMonth AS STRING
  DIM stDate AS STRING
  DIM inYear AS INTEGER
  stDay = Right("0" & Day(CDate(st)), 2)
  stMonth = Right("0" & Month(CDate(st)), 2)
  inYear = Year(CDate(st))
  IF inYear = 0 THEN
    inYear = Year(Now())
  END IF
  IF inYear > 1000 THEN
  ELSEIF inYear > 30 THEN
    inYear = 1900 + inYear
  ELSE
    inYear = 2000 + inYear
  END IF
  stDate = inYear & "-" & stMonth & "-" & stDay
  Date_to_SQLDate = stDate
END FUNCTION
```

Funkce SQL_Value kombinuje tuto funkci s níže uvedeným nastavením NULL a poskytuje tak volající funkci správně formátované hodnoty pro vstup do databáze.

Prázdná pole mají hodnotu NULL. Odpovídající pole v tabulce bude rovněž prázdné.

```
FUNCTION SQL_Value(st AS STRING, stType AS STRING) AS STRING
  DIM stValue AS STRING
  IF st = "" THEN
    SQL_Value = "NULL"
  ELSEIF stType = "DATE" THEN
    IF isDate(st) THEN
      SQL_Value = "" & Date_to_SQLDate(st) & ""
    ELSE
      SQL_Value = "NULL"
    END IF
  ELSE
    SQL_Value = "NULL"
  END IF
```

Pokud se jedná o pole s datem a jeho obsah má být rozpoznatelný jako datum, musí být jeho obsah převeden do formátu data SQL. Pokud není rozpoznatelné jako datum, mělo by pole zůstat prázdné.

```
ELSEIF stType = "DATE" THEN
  IF isDate(st) THEN
    SQL_Value = "" & Date_to_SQLDate(st) & ""
  ELSE
    SQL_Value = "NULL"
  END IF
```

Desetinné pole může obsahovat čárky místo desetinných míst a za nimi desetinná místa. V jazycích Basic a SQL je desetinným oddělovačem vždy tečka. Číslo obsahující čárku se proto musí převést. Pole musí obsahovat číslo, takže ostatní znaky, jako jsou jednotky, musí být odstraněny. To provádí funkce Val().

```
ELSEIF stType = "DECIMAL" THEN
  stValue = Str(Val(Join(Split(st, ","), ".")))
```

Veškerý ostatní obsah je považován za text. Jednoduché uvozovky jsou maskovány další jednoduchou uvozovkou a celý výraz je opět uzavřen v jednoduchých uvozovkách.

```
ELSE
    SQL_Value = "'" & String_to_SQL(st) & "'"
END IF
END FUNCTION
```

Další podrobnosti o konstrukci maker nalezneme v kapitole 9, Makra. Tento příklad jednoduše ukazuje, že je možné přenášet data z formulářů PDF do databáze aplikace Base, aniž by bylo nutné kopírovat hodnoty pole po poli pomocí schránky. Konstrukce výše uvedeného postupu byla záměrně ponechána velmi obecná a bylo by třeba ji přizpůsobit konkrétním situacím.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 8

Úlohy databáze

Obecné poznámky k databázovým úlohám

Tato kapitola popisuje některá řešení problémů, které se vyskytují u mnoha uživatelů databáze.

Filtrování dat

Filtrování dat pomocí grafického uživatelského rozhraní je popsáno v kapitole 3, Tabulky. Zde popíšeme řešení problému, na který upozornilo mnoho uživatelů: jak pomocí seznamových polí vyhledávat obsah polí v tabulkách, které se pak zobrazí filtrované v základní části formuláře a lze je upravovat.

Základem tohoto filtrování je upravitelný dotaz (viz kapitola 5, Dotazy) a další tabulka, ve které jsou uložena filtrovaná data. Dotaz zobrazí z podkladové tabulky pouze záznamy, které odpovídají hodnotám filtru. Pokud není zadána žádná hodnota filtru, dotaz zobrazí všechny záznamy.

Následující příklad vychází z ukázkové tabulky 'Media', která obsahuje mimo jiné následující pole: ID (primární klíč), Title, Category. Typy polí jsou INTEGER, VARCHAR a VARCHAR.

Nejprve potřebujeme ukázkovou tabulku „Filtr“. Tato tabulka obsahuje primární klíč a dvě filtrační pole (samozřejmě jich můžeme mít více, pokud chceme): ID (primární klíč), Filter_1, Filter_2. Protože pole příkladové tabulky „Media“, která má být filtrována, jsou typu VARCHAR, jsou pole Filter_1 a Filter_2 také tohoto typu. Pole ID může být nejmenší číselný typ TINYINT, protože tabulka Filter nikdy nebude obsahovat více než jeden záznam.

Můžeme také filtrovat pole, která se v ukázkové tabulce „Media“ vyskytují pouze jako cizí klíče. V takovém případě je třeba dát odpovídajícím polím v ukázkové tabulce „Filter“ typ odpovídající cizím klíčům, obvykle INTEGER.

Následující dotaz je určitě možné upravit:

```
SELECT * FROM "Media"
```

Zobrazí se všechny záznamy z ukázkové tabulky Media včetně primárního klíče.

```
SELECT * FROM "Media"  
WHERE "Title" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), 'Title' )
```

Pokud pole Filtr není NULL, zobrazí se ty záznamy, jejichž Název je stejný jako Filter_1. Pokud je pole Filter NULL, použije se hodnota pole Title. Protože je Title stejný jako "Title", zobrazí se všechny záznamy. Tento předpoklad však neplatí, pokud je pole Title některého záznamu prázdné (obsahuje NULL). To znamená, že se nikdy nezobrazí ty záznamy, které nemají žádný záznam názvu (hodnotu v poli Title). Proto musíme dotaz vylepšit:

```
SELECT "Media".* , IFNULL( "Media"."Title", '' ) AS "T" FROM "Media"  
WHERE "T" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), 'T' )
```

Tip

IFNULL(výraz, 'hodnota') vyžaduje, aby výraz měl stejný typ pole jako hodnota.

Pokud má výraz pole typu VARCHAR, použijeme jako hodnotu dvě jednoduché uvozovky.

Pokud je typ pole DATE, zadáme jako hodnotu datum, které není obsaženo v poli filtrované tabulky. Použijeme tento formát: {D 'RRRR-MM-DD'}.

Pokud se jedná o některý z typů číselných polí, použijeme pro hodnotu typ pole NUMERIC. Zadáme číslo, které se nevyskytuje v poli filtrované tabulky.

Tato varianta povede k požadovanému cíli. Místo přímého filtrování pole Title se filtruje pole, které nese alias T. Ani toto pole nemá žádný obsah, ale není NULL. V podmínkách se uvažuje pouze pole T. Všechny záznamy se tedy zobrazí, i když má pole Title hodnotu NULL.

Tento příkaz je k dispozici pouze přímo v jazyce SQL a nelze jej provést pomocí grafického uživatelského rozhraní. Aby jej bylo možné upravovat v grafickém uživatelském rozhraní, je třeba provést další úpravy:

```
SELECT "Media".* , IFNULL( "Media"."Title", '' ) AS "T"  
FROM "Media"  
WHERE "T" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), 'T' )
```

Pokud je nyní nastaven vztah tabulky k polím, je dotaz v grafickém uživatelském rozhraní upravitelný. Jako test můžeme vložit název do pole "Filter". "Filter_1".

Protože "Filter". "ID" nastaví hodnotu "0", záznam se uloží a filtrování je pochopitelné. Pokud je položka "Filter". "Filter_1" prázdná, grafické uživatelské rozhraní ji považuje za NULL. Při novém testu se zobrazí všechna média. V každém případě by měl být před vytvořením a otestováním formuláře do tabulky Filter zadán pouze jeden záznam s primárním klíčem. Musí se jednat pouze o jeden záznam, protože poddotazy, jak je uvedeno výše, mohou přenášet pouze jednu hodnotu.

Dotaz lze nyní rozšířit o filtrování dvou polí:

```
SELECT "Media".* , IFNULL( "Media"."Title", '' ) AS "T", IFNULL(  
"Media"."Category", '' ) AS "K"  
FROM "Media"  
WHERE "T" = IFNULL( ( SELECT "Filter_1" FROM "Filter" ), "T" ) AND "K" =  
IFNULL( ( SELECT "Filter_2" FROM "Filter" ), "K" )
```

Tím je tvorba upravitelného dotazu ukončena. Nyní základní dotaz pro dva seznamy:

```
SELECT DISTINCT "Title", "Title"  
FROM "MediaExample" ORDER BY "Title" ASC
```

Pole se seznamem by mělo zobrazit hodnotu pole Title a poté také přenést tuto hodnotu do pole Filter_1 v tabulce Filter, která je základem formuláře. Také by se neměly zobrazovat žádné duplicitní hodnoty (podmínka DISTINCT). A celou věc je samozřejmě třeba seřadit do správného pořadí.

Pro pole Category je pak vytvořen odpovídající dotaz, který má zapsat jeho data do pole Filter_2 v tabulce Filter.

Pokud jedno z těchto polí obsahuje cizí klíč, je dotaz upraven tak, aby byl cizí klíč předán podkladové tabulce Filter.

Formulář se skládá ze dvou částí. Form 1 je formulář založený na tabulce Filter. Form 2 je formulář založený na dotazu. Form 1 nemá lištu navigace a cyklus je nastaven na Aktuální záznam. Kromě toho je vlastnost *Povolit přidávání* nastavena na *Ne*. První a jediný záznam pro tento formulář již existuje.

Form 1 obsahuje dva seznamy s příslušnými popisky. Listbox 1 vrací hodnoty pro Filter_1 a je propojeno s dotazem pro pole Title. Listbox 2 vrací hodnoty pro Filter_2 a vztahuje se k dotazu pro pole Category.

Form 2 obsahuje kontrolní pole tabulky, ve kterém lze uvést všechna pole z dotazu kromě polí T a K. Formulář by fungoval i v případě, že by tato pole byla přítomna. Jsou vynechána, aby nedocházelo k matoucí duplikaci obsahu polí. Form 2 navíc obsahuje tlačítko propojené s funkcí *Aktualizovat formulář*. Aby se zabránilo blikání obrazovky při každé změně formuláře v důsledku přítomnosti lišty navigace v jednom formuláři a ne ve druhém, lze do něj zabudovat další lištu navigace.

Po dokončení formuláře začíná testovací fáze. Při změně pole se seznamem se pomocí tlačítka na Form 2 uloží tato hodnota a aktualizuje se Form 2. To se nyní vztahuje k hodnotě, kterou poskytuje pole se seznamem. Filtrování lze provést zpětně výběrem prázdného pole v seznamu.

Vyhledávání dat

Hlavní rozdíl mezi vyhledáváním dat a filtrováním dat je v technice dotazování. Cílem je poskytnout v reakci na volné jazykové výrazy výsledný seznam záznamů, které mohou tyto aktuální výrazy obsahovat pouze částečně. Nejprve jsou popsány podobné přístupy k tabulce a formuláři.

Vyhledávání pomocí LIKE

Tabulka pro obsah vyhledávání může být stejná jako tabulka, která již obsahuje hodnoty filtru. Tabulka Filter je jednoduše rozšířena o pole s názvem Searchterm. V případě potřeby lze tedy přistupovat ke stejné tabulce a pomocí formulářů ji současně filtrovat a vyhledávat. Searchterm má typ pole VARCHAR.

Formulář je vytvořen stejně jako pro filtrování. Namísto pole se seznamem potřebujeme pole pro zadání textu pro hledaný výraz a také pole s popiskem a názvem Hledat. Pole pro hledaný výraz může být ve formuláři samostatné nebo společně s poli pro filtrování, pokud jsou obě funkce žádoucí.

Rozdíl mezi filtrováním a vyhledáváním spočívá v technice dotazování. Zatímco filtrování používá termín, který se již vyskytuje v základní tabulce, vyhledávání používá libovolné položky. (Koneckonců pole se seznamem je vytvořeno z obsahu tabulky.)

```
SELECT * FROM "Media"  
WHERE "Title" = ( SELECT "Searchterm" FROM "Filter" )
```

Tento dotaz obvykle vede k prázdnému seznamu výsledků z těchto důvodů:

- Při zadávání vyhledávacích výrazů lidé málokdy vědí úplně a přesně, o jaký název se jedná. Proto se nezobrazí správný název. Chceme-li najít knihu "The Hitchhiker's Guide to the Galaxy", stačí do pole pro vyhledávání zadat "Hitchhiker" nebo dokonce jen "Hit".
- Pokud je pole Searchterm prázdné, zobrazí se pouze záznamy, u nichž není uveden žádný název. K tomu dojde pouze v případě, že položka nemá název nebo někdo nezadal její název.

Poslední podmínku lze odstranit, pokud je podmínka filtrování:

```
SELECT * FROM "Media"  
WHERE "Title" = IFNULL( ( SELECT "Searchterm" FROM "Filter" ), "Title" )
```

Po tomto zpřesnění filtrování (co se stane, když je název NULL?) získáme výsledek, který více odpovídá očekávání. První podmínka však stále není splněna. Vyhledávání by mělo dobře fungovat, i když jsou k dispozici pouze dílčí znalosti. Technika dotazování proto musí používat podmínku LIKE:

```
SELECT * FROM "Media"  
WHERE "Title" LIKE ( SELECT '%' || "Searchterm" || '%' FROM "Filter" )
```

nebo ještě lépe:

```
SELECT * FROM "Media" WHERE "Title" LIKE IFNULL( ( SELECT '%' ||  
"Searchterm" || '%' FROM "Filter" ), "Title" )
```


LIKE ve spojení s % znamená, že se zobrazí všechny záznamy, které obsahují hledaný výraz. % je zástupný znak pro libovolný počet znaků před nebo za hledaným výrazem. Po sestavení této verze dotazu stále zůstávají různé otázky:

- Pro vyhledávací výrazy se běžně používají malá písmena. Jaký výsledek dostanu, když místo "Hitch" zadám "hitch"?
- Jaké další písemné konvence je třeba vzít v úvahu?
- A co pole, která nejsou formátována jako textová pole? Můžeme vyhledávat data nebo čísla pomocí stejného vyhledávacího pole?
- A co když chceme, jako v případě filtru, zabránit tomu, aby hodnoty NULL v poli způsobily zobrazení všech záznamů?

Následující varianta zahrnuje jednu nebo dvě z těchto možností:

```
SELECT * FROM "Media" WHERE
LOWER("Title") LIKE IFNULL( ( SELECT '%' || LOWER("Searchterm") || '%' FROM
"Filter" ), LOWER("Title") )
```

Podmínka změní hledaný výraz a obsah pole na malá písmena. To také umožňuje porovnávat celé věty.

```
SELECT * FROM "Media" WHERE
LOWER("Title") LIKE IFNULL( ( SELECT '%' || LOWER("Searchterm") || '%' FROM
"Filter" ), LOWER("Title") ) OR
LOWER("Category") LIKE ( SELECT '%' || LOWER("Searchterm") || '%' FROM
"Filter" )
```

Funkce IFNULL se musí vyskytnout pouze jednou, takže když Searchterm je NULL, dotazuje se LOWER("Title") LIKE LOWER("Title"). A protože název by měl být polem, které nemůže být NULL, zobrazí se v takových případech všechny záznamy. V případě vyhledávání ve více polích se tento kód samozřejmě odpovídajícím způsobem prodlouží. V takových případech je lepší použít makro, aby kód pokryl všechna pole najednou.

Funguje však tento kód i v případě polí, která nejsou textová? Ačkoli je podmínka LIKE skutečně přizpůsobena pro text, funguje také pro čísla, data a časy, aniž by bylo nutné ji jakkoli upravovat. Konverze textu tedy ve skutečnosti nemusí proběhnout. Pole s časem, které je směsí textu a čísel, však nemůže s výsledky vyhledávání interagovat – pokud není dotaz rozšířen tak, aby byl jeden hledaný výraz rozdělen na všechny mezery mezi textem a čísly. Tím se však dotaz značně rozšíří.

Tip

Dotazy, které se používají pro filtrování a vyhledávání záznamů, lze vložit přímo do formuláře.

Celou podmínku, která byla sestavena výše, lze zadat do filtru řádků pomocí vlastností formuláře.

```
SELECT * FROM "Media" WHERE "Title" = IFNULL( ( SELECT "Searchterm" FROM
"Filter" ), "Title" )
```

se pak stane formulářem, který používá obsah tabulky Media.

V části "Filter" máme

```
("Media"."Title" = IFNULL( ( SELECT "Searchterm" FROM "Filter" ),
"Media"."Title" ) )
```

V zadání filtru dbáme na to, aby podmínka byla uvedena v závorkách a pracovala s pojmem "Tabulka". "Pole".

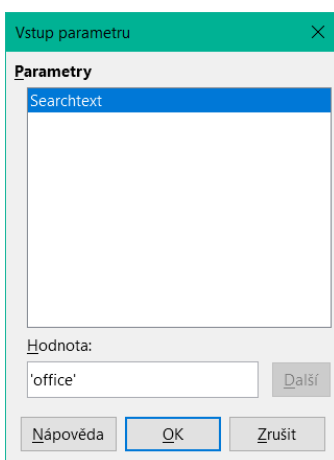
Výhodou této varianty je, že filtr lze zapínat a vypínat při otevřeném formuláři.

Vyhledávání pomocí funkce LOCATE

Vyhledávání pomocí LIKE je obvykle vyhovující pro databáze s poli obsahujícími text v množství, které lze snadno prohledat okem. Ale co pole Memo, která mohou obsahovat několik stránek textu? V takovém případě musí vyhledávání určit, kde lze zadaný text nalézt.

Pro přesné vyhledání textu má HSQLDB funkci LOCATE. Funkce LOCATE přijímá jako argument hledaný výraz (text, který chceme vyhledat). Můžeme také přidat pozici, která má být vyhledávána. Stručně: LOCATE(Hledaný výraz, Textové pole databáze, Pozice).

Následující výklad používá tabulku s názvem Table. Primární klíč se nazývá ID a musí být jedinečný. Existuje také pole s názvem Memo, které bylo vytvořeno jako pole typu Memo (LONGVARCHAR). Toto pole Memo obsahuje několik vět z této příručky.



Obrázek 416: Zadání řetězce „office“ v dialogovém okně Zadávání parametrů

✓ Poznámka

Snímky obrazovky pro tuto kapitolu pocházejí ze souboru Example_Autotext_Searchmark_Spelling.odt, který je součástí ukázkových databází pro tuto knihu.

Příklady dotazů jsou prezentovány jako dotazy s parametry. Hledaný text, který se má zadat, je vždy „office“.

ID	memo
1	What are all these things called? The terms used in LibreOffice for most parts of the user interface (the parts of the
2	Introduction In everyday office operation, spreadsheets are regularly used to aggregate sets of data
5	General notes on the creation of a database The basics of creating a database in LibreOffice are described in Chapter 8 of the
+	

Záznam | z 3

```
SELECT "ID", "memo" FROM "table"
WHERE LOWER ( "memo" ) LIKE '%' || LOWER ( :Searchtext ) || '%'
```

Obrázek 417: Výsledky hledání a kód SQL pomocí LIKE

Nejprve použijeme LIKE. LIKE lze použít pouze v podmínkách. Pokud je hledaný text kdekoli nalezen, zobrazí se odpovídající záznam. Porovnává se verze obsahu pole s malými písmeny pomocí LOWER("Memo") a verze textu hledání s malými písmeny pomocí LOWER(:Searchtext), aby se při hledání nerozlišovala velká a malá písmena. Čím delší je text v poli typu Memo, tím obtížnější je zobrazit výraz ve vyhledaném textu.

Funkce LOCATE nám přesněji ukáže, kde se hledaný výraz vyskytuje. V záznamech 1 a 2 se tento pojem nevyskytuje. V tomto případě LOCATE udává polohu '0'. Obrázek uvedený u záznamu 5 lze snadno potvrdit: řetězec „Office“ začíná na pozici 6. Samozřejmě by bylo možné zobrazit výsledky z LOCATE stejným způsobem jako u LIKE.

Ve sloupci Hits se výsledky hledání zobrazují přesněji. Předchozí dotaz byl použit jako základ pro tento dotaz. Díky tomu lze ve vnějším dotazu použít slovo "Position", místo aby bylo nutné pokaždé opakovat LOCATE(LOWER(:Searchtext),LOWER("Memo")). V zásadě se to neliší od uložení předchozího dotazu a jeho použití jako zdroje pro tento dotaz.

ID	memo	position
1	What are all these things called? The terms used in LibreOffice for most parts of the user interface (the	58
2	Introduction In everyday office operation, spreadsheets are regularly used to	26
3	The Base environment contains four work areas: Tables, Queries, Forms, and Reports. Depending on the work area selected, various tasks -	0
4	Reports – presentation of data Before an actual report in the form of a recall notice can be printed, the	0
5	General notes on the creation of a database The basics of creating a database in LibreOffice are described in Chapter	87
6	Accessing external databases An external database must exist before it can be accessed. Assuming that	0
+		

Záznam | z 6

```
SELECT "ID", "memo",
       LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ) ) "position"
FROM "table"
```

Obrázek 418: Výsledky hledání a kód SQL pomocí LOCATE (verze 1 – bez pokusu o zobrazení textu kolem místa shody)

"Position" = 0 znamená, že není žádný výsledek. V tomto případě se zobrazí zpráva `**not found**`.

"Position" < 10 znamená, že se hledaný výraz vyskytuje hned na začátku textu. 10 znaků lze snadno naskenovat pouhým okem. Proto se zobrazí celý text. Zde jsme místo SUBSTRING("Memo", 1) mohli použít pouze "Memo".

Ve všech ostatních případech hledání se hledá mezera ' ' až do 10 znaků před hledaným výrazem. Zobrazený text nezačíná uprostřed slova, ale až za mezerou. SUBSTRING("Memo", LOCATE(' ', "Memo", "Position" - 10) + 1) zajistí, že text začíná na začátku slova, které leží maximálně 10 znaků před hledaným výrazem.

V praxi bychom chtěli použít více znaků, protože existuje mnoho delších slov a hledaný výraz se může nacházet v jiném slově, které má před sebou více než 10 znaků. LibreOffice obsahuje hledaný výraz „office“ s písmenem „O“ jako šestým znakem. Pokud by hledaný výraz byl „hand“, záznam 4 by byl pro zobrazení fatální. Obsahuje slovo „LibreOffice-Handbooks“, které má 12 znaků nalevo od slova „hand“. Pokud by se hledaly mezery maximálně 10 znaků vlevo, byl by prvním nalezeným znakem znak následující za čárkou. Ve sloupci „Hits“ se zobrazí jako začátek slova „the built-in help system...“.

ID	memo	position	hit
1	What are all these things called?	58	in LibreOffice for most p
2	Introduction	26	everyday office operation
3	The Base environment contains four work areas: Tables,	0	**not found**
4	Reports – presentation of data	0	**not found**
5	General notes on the creation of a database	87	in LibreOffice are descri
6	Accessing external databases	0	**not found**

Záznam | z 6

```

SELECT "ID", "memo", "position",
CASE
WHEN "position" = 0 THEN '**not found**'
WHEN "position" < 10 THEN SUBSTRING ( "memo", 1, 25 )
ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position" - 10 ) + 1, 25 )
END AS "hit"
FROM
( SELECT "ID", "memo", LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ) )

```

Obrázek 419: Výsledky hledání a kód SQL pomocí LOCATE (verze 2 – zobrazuje text kolem místa shody, ale ve sloupci shody se zobrazují částečná slova)

Technika dotazování je stejná jako u předchozího dotazu. Pouze délka zobrazovaného záznamu byla zkrácena na 25 znaků. Funkce SUBSTRING vyžaduje jako argumenty text, který má být vyhledán, dále počáteční pozici pro výsledek a jako třetí nepovinný argument délku textového řetězce, který má být zobrazen. Zde je pro demonstrační účely nastaven poměrně krátce. Výhodou zkrácení je, že se sníží nároky na ukládání velkého počtu záznamů a umístění je snadno viditelné. Viditelnou nevýhodou tohoto typu zkracování řetězců je, že se zkracování provádí striktně podle limitu 25 znaků, aniž by se bralo v úvahu, kde slova začínají.

ID	memo	position	hit
1	What are all these things called?	58	in LibreOffice for most parts
2	Introduction	26	everyday office operation, spreadsheets
3	The Base environment contains four work areas:	0	**not found**
4	Reports – presentation of data	0	**not found**
5	General notes on the creation of a database	87	in LibreOffice are described
6	Accessing external databases	0	**not found**

Záznam | z 6

```

SELECT "ID", "memo", "position",
CASE
WHEN "position" = 0 THEN '**not found**'
WHEN "position" < 10 THEN SUBSTRING ("memo",1, LOCATE( ' ', "memo", 25 ) )
ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position" - 10 ) + 1,
(LOCATE(' ', "memo", "position" + 20)-(LOCATE(' ', "memo", "position" -10)+1))
END AS "hit"
FROM ( SELECT "ID", "memo", LOCATE( LOWER ( :Searchtext ), LOWER
( "memo" ) ) "position" FROM "table" )

```

Obrázek 420: Výsledky hledání a kód SQL pomocí LOCATE (verze 3 – zobrazuje text kolem místa shody bez dílčích slov ve sloupci shody)

Zde hledáme od 25. znaku v poli „Hits“ po další znak mezery. Obsah, který se má zobrazit, leží mezi těmito dvěma pozicemi.

Je mnohem jednodušší, když se shoda objeví na začátku pole. Zde `LOCATE(' ', "Memo", 25)` udává přesnou pozici, kde text začíná. Protože chceme, aby se text zobrazoval od začátku, odpovídá to přesně délce zobrazitelného výrazu.

Hledání prostoru následujícího za hledaným výrazem není o nic složitější, pokud výraz leží dále v poli. Hledání jednoduše začíná tam, kde se nachází shoda. Poté se počítá dalších 20 znaků, které mají následovat za všech okolností. Následující mezera je umístěna pomocí `LOCATE(' ', "Memo", "Position" +20)`. Tímto způsobem se zadává pouze umístění v rámci pole jako celku, nikoli délka řetězce, který se má zobrazit. K tomu je třeba odečíst pozici, na které má začít zobrazování odpovídajícího textu. To již bylo nastaveno v rámci dotazu pomocí `LOCATE(' ', "Memo", "Position" -10)+1`. Tímto způsobem lze zjistit správnou délku textu.

Stejnou techniku lze použít i pro řetězení dotazů. Předchozí dotaz se nyní stane zdrojem dat pro nový dotaz. Byl vložen v závorce pod výraz `FROM`. Pouze pole jsou do jisté míry přejmenována, protože nyní existuje více pozic a shod. Kromě toho je další pozici přiřazen odkaz pomocí `LOCATE(LOWER(:Searchtext), LOWER("Memo"), "Position01"+1)`. To znamená, že vyhledávání začíná znovu na pozici za předchozím odpovídajícím textem.

ID	memo	position01	hit01	position02	hit02	position03
1	What are all these	58	in LibreOffice for most parts	0	**not found**	0
2	Introduction	26	everyday office operation,	0	**not found**	0
3	The Base	0	**not found**	0	**not found**	0
4	Reports –	0	**not found**	0	**not found**	0
5	General notes on the	87	in LibreOffice are described	209	of LibreOffice, called	307
6	Accessing external	0	**not found**	0	**not found**	0

Záznam | z 6

```

SELECT "ID", "memo", "position01", "hit01", "position02",
CASE
WHEN "position02" = 0 THEN '**not found**'
WHEN "position02" < 10 THEN SUBSTRING ( "memo", 1, LOCATE( ' ', "memo",
25 ) )
ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position02" - 10 ) + 1,
( LOCATE( ' ', "memo", "position02" + 20 ) - ( LOCATE( ' ', "memo",
"position02" - 10 ) + 1 ) ) )
END AS "hit02",
CASE
WHEN "position02" = 0 THEN 0
ELSE LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ), "position02" + 1 )
END AS "position03"
FROM ( SELECT "ID", "memo", "position01",
CASE
WHEN "position01" = 0 THEN '**not found**'
WHEN "position01" < 10 THEN SUBSTRING ( "memo", 1, LOCATE( ' ', "memo",
25 ) )
ELSE SUBSTRING ( "memo", LOCATE( ' ', "memo", "position01" - 10 ) + 1,
( LOCATE( ' ', "memo", "position01" + 20 ) - ( LOCATE( ' ', "memo",
"position01" - 10 ) + 1 ) ) )
END AS "hit01",
CASE
WHEN "position01" = 0 THEN 0
ELSE LOCATE( LOWER ( :Searchtext ), LOWER ( "memo" ), "position01" + 1 )
END AS "position02"
FROM ( SELECT "ID", "memo", LOCATE( LOWER ( :Searchtext ), LOWER
( "memo" ) ) AS "position01" FROM "table" ) )

```

Obrázek 421: Výsledky hledání a kód SQL pomocí LOCATE (verze 4 – zobrazí až tři shody na záznam)

Nejvzdálenější dotaz nastaví odpovídající pole pro ostatní dva dotazy a také poskytne „hit02“ stejnou metodou, jaká byla použita pro „hit01“. Tento nejvzdálenější dotaz navíc určuje, zda existují další shody. Odpovídající pozice je uvedena jako „Pozice03“. Další shody má pouze záznam 5, který by mohl být nalezen v dalším poddotazu.

Zde uvedené skládání dotazů lze v případě potřeby provádět dále. Přidání každého nového vnějšího dotazu však představuje další zátěž pro systém. Bylo by nutné provést několik testů, aby se zjistilo, jak daleko je užitečné a reálné zajít. Kapitola 9, Makra, ukazuje, jak lze makra použít k vyhledání všech odpovídajících textových řetězců v poli pomocí formuláře.

Zpracování obrázků a dokumentů v aplikaci Base

Formuláře aplikace Base používají grafické ovládací prvky pro práci s obrázky. Pokud používáme interní databázi HSQLDB, jsou grafické ovládací prvky jediným způsobem, jak načíst obrázky z databáze bez použití maker. Lze je také použít jako odkazy na obrázky mimo databázový soubor.

Načtení obrázků do databáze

Databáze vyžaduje tabulku, která splňuje alespoň následující podmínky:

Tabulka 16: Požadavky na tabulku pro načítání obrázků do databáze

Název pole	Typ pole	Popis
ID	Integer	ID je primárním klíčem této tabulky.

Název pole	Typ pole	Popis
Image	Image	Obsahuje obrázek jako binární data.

Primární klíč musí být přítomen, ale nemusí to být celé číslo. Měla by být přidána další pole, která přidávají informace o obrázku.

Data, která budou načtena do pole obrázku, nejsou v tabulce viditelná. Místo toho se zobrazí slovo <OBJECT>. Stejně tak nelze obrázky zadávat přímo do tabulky. Musíme použít formulář, který obsahuje grafický ovládací prvek. Po klepnutí se otevře grafický ovládací prvek, který zobrazí dialogové okno pro výběr souboru. Následně se zobrazí obrázek, který byl načten z vybraného souboru.

Obrázky, které mají být vloženy přímo do databáze, by měly být co nejmenší. Vzhledem k tomu, že aplikace Base neposkytuje žádný způsob (kromě použití maker), jak exportovat obrázky v jejich původní velikosti, má smysl používat pouze nezbytnou velikost, například pro tisk v sestavě. Původní snímky v rozsahu megapixelů jsou zcela zbytečné a zahlcují databázi. Po přidání pouze několika obrázků interní HSQLDB vyhodí `java.NullPointerException` a nemůže již záznam uložit. I když obrázky nejsou tak velké, může se stát, že se databáze stane nepoužitelnou.

Obrázky by navíc neměly být integrovány do tabulek, které jsou určeny k vyhledávání. Máme-li například personální databázi a chceme-li do ní zahrnout obrázky pro použití v průkazech, je nejvhodnější uložit je do samostatné tabulky s cizím klíčem v hlavní tabulce. To znamená, že v hlavní tabulce lze vyhledávat podstatně rychleji, protože samotná tabulka nevyžaduje tolik paměti.

Odkazování na obrázky a dokumenty

Díky pečlivě navržené struktuře složek je pohodlnější přistupovat k externím souborům přímo. Soubory mimo databázi mohou být libovolně velké, aniž by to mělo vliv na fungování samotné databáze. To však také znamená, že přejmenování složky ve vlastním počítači nebo na internetu může způsobit ztrátu přístupu k souboru.

Pokud nechceme načítat obrázky přímo do databáze, ale pouze na ně odkazovat, musíme provést malou změnu v předchozí tabulce:

Tabulka 17: Požadavky na tabulku pro odkaz na externí obrázky

Název pole	Typ pole	Popis
ID	Integer	ID je primárním klíčem této tabulky.
Image	Text	Obsahuje cestu k obrázku.

Pokud je typ pole nastaven na text, grafický ovládací prvek na formuláři přenesse cestu k souboru. K obrázku lze stále přistupovat pomocí grafického ovládání stejně jako k internímu obrázku.

Totéž nelze provést s dokumentem. Není možné ani načíst cestu, protože grafické ovládací prvky jsou určeny pro grafické obrázky a dialogové okno pro výběr souborů zobrazuje pouze soubory s grafickým formátem.

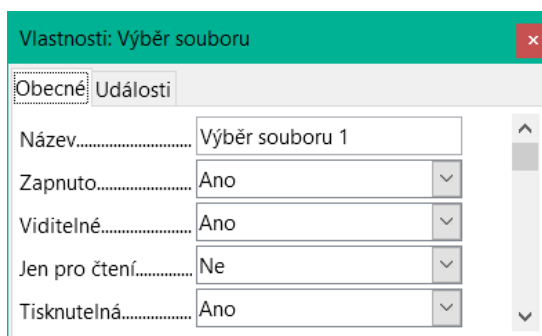
V případě obrázku lze obsah zobrazit alespoň v grafickém ovládacím prvku pomocí cesty k souboru. U dokumentu nelze zobrazit cestu, i když je uložena v tabulce. Nejprve musíme tabulku poněkud zvětšit, aby bylo vidět alespoň malé množství informací o dokumentu.

Tabulka 18: Požadavky na tabulku pro propojení s externími dokumenty

Název pole	Typ pole	Popis
ID	Integer	ID je primárním klíčem této tabulky.

Název pole	Typ pole	Popis
Popis	Text	Popis dokumentu, výrazy pro vyhledávání
Soubor	Text	Obsahuje cestu k dokumentu.

Aby byla cesta k dokumentu viditelná, musíme do formuláře zabudovat pole pro výběr souboru.



Obrázek 422: Dialogové okno Vlastnosti (karta Obecné) pro pole pro výběr souboru

Pole pro výběr souboru nemá v dialogu vlastností žádnou kartu pro data. Není tedy vázán na žádné pole v podkladové tabulce.

Propojení dokumentů pomocí absolutní cesty

Pomocí pole pro výběr souboru lze cestu zobrazit, ale nelze ji uložit. K tomu je nutný zvláštní postup, který je vázán na **Události > Text změněn**:

```
SUB PathRead(oEvent AS OBJECT)
```

```
    DIM oForm AS OBJECT
```

```
    DIM oField AS OBJECT
```

```
    DIM oField2 AS OBJECT
```

```
    DIM stUrl AS STRING
```

```
    oField = oEvent.Source.Model
```

```
    oForm = oField.Parent
```

```
    oField2 = oForm.GetByName("graphical_control")
```

```
    IF oField.Text <> "" THEN
```

```
        stUrl = ConvertToUrl(oField.Text)
```

```
        oField2.BoundField.updateString(stUrl)
```

```
    END IF
```

```
END SUB
```

Je jí předána událost, která proceduru spouští, a pomáhá najít formulář a pole, do kterého má být cesta uložena. Použití `oEvent AS OBJECT` zjednodušuje přístup, pokud chce jiný uživatel použít makro se stejným názvem v podformuláři. Zpřístupňuje pole pro výběr souboru prostřednictvím `oEvent.Source.Model`. Formulář je přístupný jako nadřazené pole pro výběr souboru. Název formuláře je proto irelevantní. Z formuláře je nyní přístupné pole s názvem „graphical_control“. Toto pole se obvykle používá k ukládání cest k souborům obrázků. V tomto případě se do něj zapíše adresa URL vybraného souboru. Aby bylo zajištěno, že adresa URL bude fungovat v souladu s konvencemi operačního systému, převede se text v poli pro výběr souboru do obecně platné podoby pomocí funkce `ConvertToUrl`.

Databázová tabulka nyní obsahuje cestu v absolutním formátu: `file:///...`

Pokud jsou položky cesty načítány pomocí grafického ovládání, získáme relativní cestu. Aby byla použitelná, je třeba ji vylepšit. Tento postup je mnohem zdlouhavější, protože zahrnuje porovnání vstupní a skutečné cesty.

Propojení dokumentů pomocí relativní cesty

Následující makro je vázáno na vlastnost „Text modified“ pole pro výběr souboru.

```
SUB PathRead
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oField AS OBJECT
    DIM oField2 AS OBJECT
    DIM arUrl_Start()
    DIM ar()
    DIM ar1()
    DIM ar2()
    DIM stText AS STRING
    DIM stUrl_complete AS STRING
    DIM stUrl_Text AS STRING
    DIM stUrl AS STRING
    DIM stUrl_cut AS STRING
    DIM ink AS INTEGER
    DIM i AS INTEGER
    oDoc = thisComponent
    oDrawpage = oDoc.Drawpage
    oForm = oDrawpage.Forms.getByName("Form")
    oField = oForm.getByName("graphical_control")
    oField2 = oForm.getByName("filecontrol")
```

Nejprve se stejně jako ve všech procedurách deklarují proměnné. Poté se vyhledají pole, která jsou důležitá pro zadávání cest. Celý následující kód se pak provede pouze v případě, že v poli pro výběr souboru skutečně něco je, tj. nebylo vyprázdněno změnou záznamu.

```
IF oField2.Text <> "" THEN
    arUrl_Start = split(oDoc.Parent.Url, oDoc.Parent.Title)
    ar = split(ConvertToUrl(oField2.Text), "/")
    stText = ""
```

Čte se cesta k databázovému souboru. To se provádí, jak je uvedeno výše, nejprve načtením celé adresy URL a jejím rozdělením do pole tak, aby první prvek pole obsahoval přímou cestu.

Poté se všechny prvky cesty nalezené v poli pro výběr souboru načtou do pole ar. Oddělovač je /. Takto to lze provést přímo v systému Linux. V systému Windows musí být obsah oField2 převeden na adresu URL, která jako oddělovač cesty použije lomítko (nikoli zpětné lomítko).

Účelem rozdělení je získat cestu k souboru jednoduchým odříznutím názvu souboru na konci. Proto se v dalším kroku opět sestaví cesta k souboru a vloží se do proměnné stText. Smyčka nekončí posledním prvkem pole ar, ale předchozím prvkem.

```
FOR i = LBound(ar()) TO UBound(ar()) - 1
    stText = stText & ar(i) & "/"
NEXT
stText = Left(stText, Len(stText)-1)
arUrl_Start(0) = Left(arUrl_Start(0), Len(arUrl_Start(0))-1)
```

Konečné / je opět odstraněno, protože jinak by se v následujícím poli objevila prázdná hodnota, což by narušilo porovnávání cest. Pro správné porovnání musí být text převeden na správnou adresu URL začínající file:/// . Nakonec se porovná cesta k databázovému souboru s cestou, která byla vytvořena.

```
stUrl_Text = ConvertToUrl(stText)
ar1 = split(stUrl_Text, "/")
ar2 = split(arUrl_Start(0), "/")
stUrl = ""
ink = 0
stUrl_cut = ""
```

Pole ar1ar2 se porovnává krok za krokem ve smyčce.

```
FOR i = LBound(ar2()) TO UBound(ar2())
    IF i <= UBound(ar1()) THEN
```

Následující kód se provede pouze tehdy, pokud číslo i není větší než počet prvků v ar1. Pokud je hodnota v ar2 stejná jako odpovídající hodnota v ar1 a do této chvíle nebyla nalezena žádná nekompatibilní hodnota, je společný obsah uložen do proměnné, kterou lze nakonec od hodnoty cesty odříznout.

```
IF ar2(i) = ar1(i) AND ink = 0 THEN
    stUrl_cut = stUrl_cut & ar1(i) & "/"
ELSE
```

Pokud je v některém bodě mezi oběma poli rozdíl, pak se pro každou odlišnou hodnotu přidá do proměnné stUrl znaménko pro přechod o jeden adresář výše.

```
stUrl = stUrl & "../"
ink = 1
END IF
```

Jakmile je index uložený v i větší než počet prvků v ar1, každá další hodnota v ar2 způsobí uložení dalšího ../ do proměnné stUrl.

```
ELSE
    stUrl = stUrl & "../"
END IF
NEXT
stUrl_complete = ConvertToUrl(oFeld2.Text)
oFeld.boundField.UpdateString(stUrl &
Right(stUrl_complete, Len(stUrl_complete)-Len(stUrl_cut)))
```

```
END IF
```

```
END SUB
```

Po dokončení smyčky ar2 jsme zjistili, zda a o kolik je soubor, ke kterému se má přistupovat, ve stromu výše než databázový soubor. Nyní lze z textu v poli pro výběr souboru vytvořit `stUrl_complete`. Obsahuje také název souboru. Nakonec se hodnota přenesla do grafického ovládacího prvku. Hodnota URL začíná znakem `stUrl`, který obsahuje potřebný počet teček (`../`). Pak je začátek `stUrl_complete`, část, která se ukázala být stejná pro databázi i externí soubor, odříznut. Způsob rozřezání řetězce je uložen v `stUrl_cut`.

Zobrazení propojených obrázků a dokumentů

Propojené obrázky lze zobrazit přímo v grafickém ovládacím prvku. Větší displej by však lépe zobrazoval detaily.

Dokumenty nejsou v aplikaci Base běžně viditelné.

Aby bylo možné tento typ zobrazení provést, musíme opět použít makra. Toto makro se spouští pomocí tlačítka na formuláři, které obsahuje grafický ovládací prvek.

```
SUB View(oEvent AS OBJECT)
```

```
    DIM oDoc AS OBJECT
```

```
    DIM oForm AS OBJECT
```

```
    DIM oField AS OBJECT
```

```
    DIM oShell AS OBJECT
```

```
    DIM stUrl AS STRING
```

```
    DIM stField AS STRING
```

```
    DIM arUrl_Start()
```

```
    oDoc = thisComponent
```

```
    oForm = oEvent.Source.Model.Parent
```

```
    oField = oForm.GetName("graphical_control")
```

```
    stUrl = oField.BoundField.GetString
```

Grafický ovládací prvek je ve formuláři umístěn. Protože tabulka neobsahuje samotný obrázek, ale pouze cestu k němu uloženou jako textový řetězec, načte se tento text pomocí `getString`.

Poté se určí cesta k databázovému souboru. Soubor `odb`, kontejner pro formuláře, je přístupný pomocí `oDoc.Parent`. Celá adresa URL včetně názvu souboru se načte pomocí `oDoc.Parent.Url`. Název souboru je také uložen v `oDoc.Parent.Title`. Text je oddělen pomocí funkce `split` s názvem souboru jako oddělovačem. Jako první a jediný prvek pole je uvedena cesta k databázovému souboru.

```
    arUrl_Start = split(oDoc.Parent.Url, oDoc.Parent.Title)
```

```
    oShell = createUnoService("com.sun.star.system.SystemShellExecute")
```

```
    stField = convertToUrl(arUrl_Start(0) + stUrl)
```

```
    oShell.execute(stField, , 0)
```

```
END SUB
```

Externí programy lze spouštět pomocí struktury `com.sun.star.system.SystemShellExecute`. Absolutní cesta k souboru, sestavená z cesty k databázovému souboru a interně uložené relativní cesty z databázového souboru, je předána externímu programu. Grafické rozhraní operačního systému určuje, který program bude zavolán k otevření souboru.

Příkaz oShell.execute přijímá tři argumenty. Prvním je spustitelný soubor nebo cesta k datovému souboru, který je systémem propojen s programem. Druhým je seznam argumentů pro program. Třetí je číslo, které určuje, jak se mají chyby hlásit. Možnosti jsou 0 (výchozí chybová zpráva), 1 (žádná zpráva) a 2 (povolit pouze otevření absolutních URL adres).

Načtení dokumentů do databáze

Při čtení v dokumentech je třeba vždy dodržovat následující podmínky:

- Čím větší je počet dokumentů, tím je databáze těžkopádnější. Proto je pro rozsáhlé dokumenty vhodnější externí databáze než interní.
- Stejně jako v případě obrázků nelze v dokumentech vyhledávat. Jsou uloženy jako binární data, a proto je lze vložit do pole obrázku.
- Dokumenty načtené do interní databáze HSQLDB lze načíst pouze pomocí maker. Pomocí SQL dotazů to provést nelze.

Následující makra pro načítání a odesílání závisí na tabulce, která obsahuje popis dat a původní název souboru a také binární verzi souboru. Název souboru není automaticky ukládán spolu se souborem, ale může poskytnout užitečné informace o typu dat uložených v souboru, který má být načten. Teprve potom mohou soubor bezpečně číst jiné programy.

Tabulka obsahuje následující pole:

Tabulka 19: Požadavky na tabulku pro načítání dokumentů dovnitř a ven

Název pole	Typ pole	Popis
ID	Integer	ID je primárním klíčem této tabulky.
Popis	Text	Popis dokumentu, vyhledávací termíny atd.
Soubor	Image	Obrázek nebo soubor v binární podobě.
Filename	Text	Název souboru včetně jeho přípony. Důležité pro další čtení.

Formulář pro načítání a odesílání souborů vypadá takto:

Obrázek 423: Formulář pro načítání souborů dovnitř a ven

Pokud jsou v databázi přítomny soubory obrázků, lze je zobrazit v grafickém ovládacím prvku formuláře. Všechny ostatní typy souborů jsou skryté.

Následující makro pro načtení souboru je spuštěno příkazem Vlastnosti: Výběr souboru → Události → Text změněn.

SUB FileInput_withName(oEvent AS OBJECT)

```

DIM oForm AS OBJECT
DIM oField AS OBJECT
DIM oField2 AS OBJECT
DIM oField3 AS OBJECT
DIM oStream AS OBJECT
DIM oSimpleFileAccess AS OBJECT
DIM stUrl AS STRING
DIM stName AS STRING
    oField = oEvent.Source.Model
oForm = oField.Parent
oField2 = oForm.GetByName("txt_filename")
oField3 = oForm.GetByName("graphical_control")
IF oField.Text <> "" THEN
    stUrl = ConvertToUrl(oField.Text)
    ar = split(stUrl, "/")
    stName = ar(UBound(ar))
    oField2.BoundField.updateString(stName)
    oSimpleFileAccess =
createUnoService("com.sun.star.ucb.SimpleFileAccess")
    oStream = oSimpleFileAccess.openFileRead(stUrl)
    oField3.BoundField.updateBinaryStream(oStream, oStream.getLength())
END IF
END SUB

```

Vzhledem k tomu, že spouštěcí událost makra poskytuje název jiného pole formuláře, není nutné kontrolovat, zda se pole nachází v hlavním formuláři nebo v podformuláři. Je pouze nutné, aby všechna pole byla ve stejném tvaru.

V poli „txt_filename“ je uložen název souboru, který se má vyhledat. V případě obrázků je třeba tento název zadat ručně bez použití makra. Místo toho je název souboru určen pomocí adresy URL a automaticky zadán při načítání dat.

V poli „graphical_control“ jsou uložena aktuální data jak pro obrázky, tak pro ostatní soubory.

Kompletní cesta včetně názvu souboru se načte z pole pro výběr souboru pomocí `oField.Text`. Aby se zajistilo, že adresa URL nebude ovlivněna podmínkami specifickými pro operační systém, převede se načtený text do standardního formátu URL pomocí příkazu `ConvertToUrl`. Tato univerzálně platná adresa URL je rozdělena do pole. Oddělovač je `/`. Posledním prvkem cesty je název souboru. `UBound(ar)` udává index tohoto posledního prvku. Skutečný název souboru pak lze přečíst pomocí `ar(UBound(ar))` a přenést do pole jako řetězec.

Pro samotné čtení souboru je vyžadována `UnoService com.sun.star.ucb.SimpleFileAccess`. Tato služba může číst obsah souboru jako datový proud. Ten je dočasně uložen v objektu `oStream` a poté vložen jako datový tok do pole vázaného na pole „File“ v tabulce. To vyžaduje zadání délky datového toku a objektu `oStream`.

Údaje jsou nyní uvnitř pole formuláře stejně jako při běžném zadávání. Pokud je však formulář v tomto okamžiku jednoduše uzavřen, data se neuloží. Uložení vyžaduje stisknutí tlačítka Uložit na liště navigace; k uložení dojde také automaticky při přechodu na další záznam.

Určování názvů obrazových souborů

Ve výše uvedené metodě bylo stručně zmíněno, že název souboru používaného pro vstup do grafického ovládání nelze přímo určit. Zde je makro pro určení tohoto názvu souboru, které odpovídá výše uvedenému formuláři. Název souboru nelze s jistotou určit pomocí události přímo vázané na grafický ovládací prvek. Makro se tedy spouští pomocí **Vlastnosti formuláře > Události > Před záznamem**.

```
SUB ImagenameRead(oEvent AS OBJECT)
    oForm = oEvent.Source
    IF InStr(oForm.ImplementationName, "ODatabaseForm") THEN
        oField = oForm.getByName("graphical_control")
        oField2 = oForm.getByName("txt_filename")
        IF oField.ImageUrl <> "" THEN
            stUrl = ConvertToUrl(oField.ImageUrl)
            ar = split(stUrl, "/")
            stName = ar(UBound(ar))
            oField2.BoundField.updateString(stName)
        END IF
    END IF
END SUB
```

Před záznamem se provedou dvě implementace s různými názvy implementací. Formulář je nejnázveu přístupný pomocí implementace ODatabaseForm.

V grafickém ovládacím prvku lze adresu URL zdroje dat získat pomocí ImageUrl. Tato URL adresa je načtena, název souboru je určen pomocí předchozí procedury FileInput_withName a je přenesen do pole txt_filename.

Odstranění názvů souborů obrázků z paměti

Pokud po spuštění výše uvedeného makra přejdeme na další záznam, cesta k původnímu obrázku je stále k dispozici. Pokud je nyní pomocí pole pro výběr souboru načten jiný soubor než obrázek, název souboru pro obrázek se přepíše názvem tohoto souboru, pokud nepoužijeme následující makro.

Cestu nelze odstranit pomocí předchozího makra, protože soubor s obrázkem se načte až při ukládání záznamu. Odstraněním cesty v tomto bodě by se obrázek odstranil.

Makro se spouští pomocí **Vlastnosti formuláře > Události > Po záznamu**.

```
SUB ImagenameReset(oEvent AS OBJECT)
    oForm = oEvent.Source
    IF InStr(oForm.ImplementationName, "ODatabaseForm") THEN
        oField = oForm.getByName("graphical_control")
        IF oField.ImageUrl <> "" THEN
            oField.ImageUrl = ""
        END IF
    END IF
END SUB
```

Stejně jako v proceduře „ImageRead“ se přistupuje k ovládacímu prvku grafiky. Pokud existuje položka imageUrl, je odstraněna.

Čtení a zobrazování obrázků a dokumentů

U negrafických souborů i obrázků původní velikosti je třeba stisknout tlačítko **Otevřít soubor pomocí externího programu**. Soubory v dočasné složce pak lze číst a zobrazovat pomocí programu spojeného s příponou souboru v operačním systému.

Makro se spouští pomocí **Vlastnosti: Tlačítko > Události > Provést akci**.

```
SUB FileDisplay_withName(oEvent AS OBJECT)
    DIM oDoc AS OBJECT
    DIM oDrawpage AS OBJECT
    DIM oForm AS OBJECT
    DIM oField AS OBJECT
    DIM oField2 AS OBJECT
    DIM oStream AS OBJECT
    DIM oShell AS OBJECT
    DIM oPath AS OBJECT
    DIM oSimpleFileAccess AS OBJECT
    DIM stName AS STRING
    DIM stPath AS STRING
    DIM stField AS STRING
    oForm = oEvent.Source.Model.Parent
    oField = oForm.getByname("graphical_control")
    oField2 = oForm.getByname("txt_filename")
    stName = oField2.Text
    IF stName = "" THEN
        stName = "file"
    END IF
    oStream = oField.BoundField.getBinaryStream
    oPath = createUnoService("com.sun.star.util.PathSettings")
    stPath = oPath.Temp & "/" & stName
    oSimpleFileAccess =
createUnoService("com.sun.star.ucb.SimpleFileAccess")
    oSimpleFileAccess.writeFile(stPath, oStream)
    oShell = createUnoService("com.sun.star.system.SystemShellExecute")
    stField = convertToUrl(stPath)
    oShell.execute(stField,,0)
END SUB
```

Pozice ostatních dotčených polí ve formuláři je dána tlačítkem. Pokud název souboru chybí, soubor se jednoduše pojmenuje „File“.

Obsah ovládacího prvku formuláře „graphical_control“ odpovídá obsahu pole File v tabulce. Čte se jako datový proud. Cesta k dočasné složce se používá jako cesta k těmto datům; lze ji nastavit pomocí **Nástroje > Možnosti > LibreOffice > Cesty**. Pokud mají být data následně použita k jiným účelům, a ne pouze zobrazena, lze je z této cesty zkopírovat. V rámci makra se soubor po úspěšném přečtení otevře přímo pomocí programu, který byl grafickým uživatelským rozhraním operačního systému vázán na příponu souboru.

Úryvky kódu

Tyto úryvky kódu pocházejí z dotazů do poštovních konferencí. Vystávají konkrétní problémy, které by mohly být užitečné jako řešení pro vaše vlastní databázové experimenty.

Zjištění aktuálního věku

Dotaz potřebuje vypočítat skutečný věk osoby z data narození. Viz také funkce v dodatku k této příručce Průvodce aplikací Base.

```
SELECT DATEDIFF('yy', "Birthdate", CURDATE()) AS "Age" FROM "Person"
```

Tento dotaz uvádí věk jako rozdíl v letech. Věk dítěte narozeného 31. prosince 2011 by se však 1. ledna 2012 počítal jako 1 rok, protože se jedná o přestupný rok. Musíme tedy vzít v úvahu také postavení dne v rámci roku. To je dostupné pomocí funkce DAYOFYEAR(). Porovnání provede jiná funkce.

```
SELECT CASEWHEN  
(DAYOFYEAR("Birthdate") > DAYOFYEAR(CURDATE())),  
DATEDIFF('yy', "Birthdate", CURDATE())-1,  
DATEDIFF('yy', "Birthdate", CURDATE())  
AS "Age" FROM "Person"
```

Nyní získáme správný aktuální věk v letech.

CASEWHEN lze také použít k tomu, aby se text Birthday dnes objevil v jiném poli, pokud DAYOFYEAR("Birthdate") = DAYOFYEAR(CURDATE()).

Nyní se může objevit jemná námitka: "A co přestupné roky?". U osob narozených po 28. únoru se jedná o chybu jednoho dne. Při každodenním používání to není závažný problém, ale neměli bychom se snažit o přesnost?

```
CASEWHEN (  
(MONTH("Birthdate") > MONTH(CURDATE())) OR  
((MONTH("Birthdate") = MONTH(CURDATE())) AND (DAY("Birthdate") >  
DAY(CURDATE()))),  
DATEDIFF('yy', "Birthdate", CURDATE())-1,  
DATEDIFF('yy', "Birthdate", CURDATE())
```

Výše uvedený kód tohoto cíle dosahuje. Pokud je měsíc data narození větší než aktuální měsíc, odečte funkce rozdíl roku jeden rok. Stejně tak se odečte jeden rok, pokud jsou oba měsíce stejné, ale den v měsíci pro datum narození je větší než den v aktuálním datu. Tento vzorec však není pro grafické uživatelské rozhraní srozumitelný. Tento dotaz úspěšně zpracuje pouze *Přímý SQL příkaz*, který by zabránil úpravě našeho dotazu. Dotaz však musí být editovatelný, takže zde je návod, jak grafické uživatelské rozhraní obelstít:

```
CASE  
WHEN MONTH("Birthdate") > MONTH(CURDATE())
```



```

THEN DATEDIFF('yy', "Birthdate", CURDATE())-1
WHEN (MONTH("Birthdate") = MONTH(CURDATE()) AND DAY("Birthdate") >
DAY(CURDATE()))
THEN DATEDIFF('yy', "Birthdate", CURDATE())-1
ELSE DATEDIFF('yy', "Birthdate", CURDATE())
END

```

Při této formulaci již grafické uživatelské rozhraní nereaguje chybovou zprávou. Věk je nyní uváděn přesně i v přestupných letech a dotaz je stále editovatelný.

Zobrazení narozenin, které nastanou v nejbližších dnech

Pomocí malého úryvku výpočtu můžeme z tabulky zjistit, kdo bude v následujících osmi dnech slavit narozeniny.

```

SELECT *
FROM "Table"
WHERE
    DAYOFYEAR("Date") BETWEEN DAYOFYEAR(CURDATE()) AND
        DAYOFYEAR(CURDATE()) + 7
    OR DAYOFYEAR("Date") < 7 -
        DAYOFYEAR(CAST(YEAR(CURDATE()) || '-12-31' AS DATE)) +
        DAYOFYEAR(CURDATE())

```

Dotaz zobrazí všechny záznamy, jejichž datum leží mezi aktuálním dnem roku a následujícími 7 dny.

Aby bylo i na konci roku uvedeno 8 dní, je třeba důkladně zkontrolovat den, kdy rok začal. Tato kontrola se provádí pouze pro čísla dnů, která jsou nejvýše o 7 dní pozdější než poslední číslo dne v aktuálním roce (obvykle 365) plus číslo dne v aktuálním datu. Pokud je aktuální datum vzdáleno od konce roku více než 7 dní, je součet <1. Žádný záznam v tabulce nemá takové datum, takže v takových případech není tato dílčí podmínka splněna.

Ve výše uvedeném vzorci budou přestupné roky dávat špatný výsledek, protože jejich data jsou posunuta o výskyt 29. února. Kód musí být rozsáhlejší, aby se této chybě předešlo:

```

SELECT *
FROM "Table"
WHERE
    CASE
        WHEN
            DAYOFYEAR(CAST(YEAR("Date") || '-12-31' AS DATE)) = 366
            AND DAYOFYEAR("Date") > 60 THEN DAYOFYEAR("Date") - 1
        ELSE
            DAYOFYEAR("Date")
    END
BETWEEN
    CASE
        WHEN

```

```

        DAYOFYEAR(CAST(YEAR(CURDATE()) || '-12-31' AS DATE)) = 366
        AND DAYOFYEAR(CURDATE()) > 60 THEN DAYOFYEAR(CURDATE()) - 1
    ELSE
        DAYOFYEAR(CURDATE())
    END
AND
CASE
    WHEN
        DAYOFYEAR(CAST(YEAR(CURDATE()) || '-12-31' AS DATE)) = 366
        AND DAYOFYEAR(CURDATE()) > 60 THEN DAYOFYEAR(CURDATE()) + 6
    ELSE
        DAYOFYEAR(CURDATE()) + 7
    END
OR DAYOFYEAR("Datum") < 7 -
    DAYOFYEAR(CAST(YEAR(CURDATE()) || '-12-31' AS DATE)) +
    DAYOFYEAR(CURDATE())

```

Přestupný rok poznáme podle toho, že celkový počet dní je 366, nikoli 365. To se používá pro odpovídající určení.

Jednak je třeba u každé hodnoty data ověřit, zda leží v přestupném roce, a také správný počet 60. dne (31 dní v lednu a 29 dní v únoru). V tomto případě je třeba všechny následující hodnoty DAYOFYEAR pro datum zvýšit o 1. Pak bude 1. březen přestupného roku přesně odpovídat 1. březnu běžného roku.

Na druhou stranu je třeba otestovat, zda je aktuální rok (CURDATE()) skutečně přestupný. I zde je třeba zvýšit počet dní o 1.

Zobrazení koncové hodnoty pro následujících 8 dní také není tak jednoduché, protože rok stále není v dotazu zahrnut. Tuto podmínku by však bylo snadné přidat: YEAR("Date") = YEAR(CURDATE()) pro aktuální rok nebo YEAR("Date") = YEAR(CURDATE()) + 1 pro následující rok.

Přidání dnů k hodnotě data

Při půjčování médií by knihovna mohla chtít znát přesný den, kdy má být médium vráceno. Interní HSQLDB neposkytuje funkci DATEADD(), která je k dispozici v mnoha externích databázích a také v interním Firebirdu. Zde je uveden nepřímý způsob, jak toho dosáhnout na omezený časový úsek.

Nejprve se vytvoří tabulka obsahující posloupnost dat pokrývající požadované časové období. Za tímto účelem se otevře program Calc a do pole A1 se vloží název „ID“ a do pole B1 „Date“. Do pole A2 zadáme 1 a do pole B2 počáteční datum, například 15. 1. 2015 (anglický formát 01/15/2015) Vybereme A2 a B2 a přetáhneme je dolů. Tím se vytvoří posloupnost čísel ve sloupci A a posloupnost dat ve sloupci B.

Poté je celá tato tabulka včetně nadpisů sloupců vybrána a importována do databáze Base: **klepnutí pravým tlačítkem myši > Vložit > Název tabulky > Date**. V části Možnosti klepneme na možnost Definice a data a na možnost Použít první řádek jako názvy sloupců. Všechny sloupce jsou přeneseny. Poté se ujistíme, že pole ID má typ Integer [INTEGER] a pole Date typ Date [DATE]. Primární klíč není nutný, protože záznamy nebudou později měněny. Protože nebyl definován primární klíč, je tabulka chráněna proti zápisu.



Tip

K vytvoření takového zobrazení můžeme použít také techniku dotazování. Pokud použijeme filtrační tabulku, můžeme dokonce kontrolovat počáteční datum a rozsah hodnot data.

```
SELECT DISTINCT CAST
( "Y"."Nr" + (SELECT "Year" FROM "Filter" WHERE "ID" = True) - 1 || '-' ||
CASEWHEN( "M"."Nr" < 10, '0' || "M"."Nr", '' || "M"."Nr" ) || '-' ||
CASEWHEN( "D"."Nr" < 10, '0' || "D"."Nr", '' || "D"."Nr" )
AS DATE ) AS "Date"
FROM "Nrto31" AS "D", "Nrto31" AS "M", "Nrto31" AS "Y"
WHERE "Y"."Nr" <= (SELECT "Year" FROM "Filter" WHERE "ID" = True) AND
"M"."Nr" <= 12 AND "D"."Nr" <= 31
```

Toto zobrazení přistupuje k tabulce, která obsahuje pouze čísla od 1 do 31 a je chráněna proti zápisu. Další filtrační tabulka obsahuje počáteční rok a rozsah roků, které má zobrazení pokrýt. Z nich se sestaví datum a vytvoří se textový výraz pro datum (rok, měsíc, den), který lze následně převést na datum. HSQLDB přijímá všechny dny až do 31. dne v měsíci a řetězce jako 31. 02. 2015. Datum 31. 2. 2015 je však přeneseno jako datum 3. 3. 2015. Při přípravě zobrazení proto musíme použít funkci DISTINCT, abychom vyloučili duplicitní hodnoty dat.

Zde je účinný následující pohled:

```
SELECT "a"."Date",
(SELECT COUNT(*) FROM "View_date" WHERE "Date" <=
"a"."Date")
AS "lfdNr"
FROM "View_Date" AS "a"
```

Pomocí číslování řádků se hodnota data převede na číslo. Vzhledem k tomu, že data v zobrazení nelze odstranit, není třeba žádná další ochrana proti zápisu.

Pomocí dotazu nyní můžeme určit konkrétní datum, například datum za 14 dní:

```
SELECT "a"."Loan_Date",
(SELECT "Date" FROM "Date" WHERE "ID" =
(SELECT "ID" FROM "Date" WHERE "Date" = "a"."Loan_Date")+14)
AS "Returndate"
FROM "Loans" AS "a"
```

V prvním sloupci je uvedeno datum půjčky. K tomuto sloupci se přistupuje pomocí korelačního poddotazu, který je opět rozdělen do dvou dotazů. SELECT "ID" FROM "Date" poskytne hodnotu pole ID, která odpovídá datu vydání. K hodnotě se připočítává 14 dní. Výsledek je přiřazen poli ID vnějším poddotazem. Toto nové ID pak určuje, které datum se vloží do pole data.

Při zobrazení tohoto dotazu není automaticky rozpoznán typ data, takže je nutné použít formátování. Ve formuláři lze uložit odpovídající zobrazení, takže každý dotaz poskytne hodnotu data.

Přímá varianta určení hodnoty data je možná kratší cestou:

```
SELECT "Loan_Date",  
       DATEDIFF( 'dd', '1899-12-30', "Loan_Date" ) + 14  
       AS "Returndate"  
FROM "Table"
```

Vrácenou číselnou hodnotu lze ve formuláři formátovat jako datum pomocí formátovaného pole. Zpřístupnění pro další zpracování SQL v dotazu však vyžaduje mnoho práce.

Přidání času k časovému razítku

MySQL má funkci `TIMESTAMPADD()`. Podobná funkce v HSQLDB neexistuje. K sečtení nebo odečtení však lze použít interní číselnou hodnotu časového razítka pomocí formátovaného pole ve formuláři.

Na rozdíl od přidávání dnů k datu způsobují časy problém, který nemusí být na začátku zřejmý.

```
SELECT "DateTime"  
       DATEDIFF( 'ss', '1899-12-30', "DateTime" ) / 86400.0000000000 +  
       36/24 AS "DateTime+36hours"  
FROM "Table"
```

Nový vypočtený čas je založen na rozdílu od nulového času systému. Při výpočtu data se jedná o datum 30. 12. 1899.



Poznámka

Nulté datum 30. 12. 1899 bylo údajně zvoleno proto, že rok 1900 nebyl na rozdíl od většiny let dělitelný čtyřmi přestupný. Značka „1“ interního výpočtu byla tedy přesunuta zpět na 31.12.1899, a nikoli na 1.1.1900.

Rozdíl je vyjádřen v sekundách, ale interní číslo počítá dny jako čísla před desetinnou čárkou a hodiny, minuty a sekundy jako desetinná místa. Protože den obsahuje $60 \cdot 60 \cdot 24$ sekund, je třeba pro správný výpočet dnů a zlomků dnů vydělit počet sekund číslem 86400. Pokud má interní HSQLDB vůbec uvádět desetinná místa, musí být do výpočtu zahrnuta, takže místo 86400 musíme dělit 86400.0000000000. Desetinná místa v dotazu musí být oddělena desetinnou tečkou bez ohledu na místní zvyklosti. Výsledek bude mít za tečkou 10 desetinných míst.

K tomuto výsledku je třeba přičíst celkový počet hodin jako zlomek dne. Vypočtený údaj, vhodně naformátovaný, lze vytvořit v dotazu. Formátování se neuloží, ale lze je přenést ve správném formátu pomocí formátovaného pole ve formuláři nebo sestavě.

Pokud je třeba přidat minuty nebo sekundy, dbejme na to, aby byly uvedeny jako zlomky dne.

Pokud datum spadá do listopadu, prosince, ledna atd., nejsou s výpočtem žádné problémy. Zdá se, že jsou poměrně přesné: přičtením 36 hodin k časovému razítku 20.01.2015 13:00:00 získáme 22.01.2015 00:00:00. Pro datum 20.04.2015 13:00:00 je to ale jinak. Výsledek je 04/22/2015 00:00:00. Výpočet je špatný kvůli letnímu času. Hodina „ztracená“ nebo „získaná“ změnou času se nezohledňuje. V rámci jednoho časového pásma existují různé způsoby, jak získat „správný“ výsledek. Zde je jednoduchá varianta:

```
SELECT "DateTime"  
       DATEDIFF( 'dd', '1899-12-30', "DateTime" ) +  
       HOUR( "DateTime" ) / 24.0000000000 +
```

```

MINUTE( "DateTime" ) / 1440.0000000000 +
SECOND( "DateTime" ) / 86400.0000000000 +
36/24
AS "DateTime+36hours"

```

```
FROM "Table"
```

Místo počítání hodin, minut a sekund od počátku data se počítají od aktuálního data. Dne 20.5.2015 je čas 13:00, ale bez letního času by byl zobrazen jako 12:00. Funkce HOUR zohledňuje letní čas a jako hodinovou část času uvádí 13 hodin. Tu pak můžeme správně přidat do denní části. S minutami a sekundami se pracuje úplně stejně. Nakonec se hodiny navíc přičtou jako zlomek dne a vše se zobrazí jako vypočtené časové razítko pomocí formátování buněk.

Při tomto výpočtu je třeba mít na paměti dvě věci:

- Při přechodu ze zimního času na letní se hodinové hodnoty nezobrazují správně. To lze opravit pomocí pomocné tabulky, která převezme data začátku a konce letního času a opraví hodinový počet. Poněkud komplikovaná záležitost.
- Zobrazení časů je možné pouze u formátovaných polí. Výsledkem je desetinné číslo, nikoli časová značka, kterou by bylo možné uložit přímo do databáze. Buď se musí zkopírovat v rámci formuláře, nebo převést z desetinného čísla na časovou značku pomocí složitějšího dotazu. Zlomovým bodem převodu je hodnota data, protože se může jednat o přestupné roky nebo měsíce s různým počtem dní.

Získání průběžné bilance podle kategorií

Namísto domácího deníku může databáze v počítači zjednodušit únavné sčítání výdajů na jídlo, oblečení, dopravu atd. Chceme, aby většina těchto údajů byla v databázi okamžitě viditelná, takže náš příklad předpokládá, že příjmy a výdaje budou uloženy jako podepsané hodnoty v jednom poli nazvaném Částka (Amount). V zásadě lze celou věc rozšířit tak, aby zahrnovala samostatná pole a příslušný součet pro každé z nich.

```

SELECT "ID", "Amount", (SELECT SUM("Amount") FROM "Cash" WHERE "ID" <=
"a"."ID") AS "Balance" FROM "Cash" AS "a" ORDER BY "ID" ASC

```

Tento dotaz způsobí pro každý nový záznam přímý výpočet zůstatku běžného účtu. Dotaz přitom zůstává editovatelný, protože pole Balance je vytvořeno prostřednictvím korelujícího poddotazu. Dotaz závisí na automaticky vytvořeném primárním klíči ID pro výpočet stavu účtu. Zůstatky se však obvykle počítají denně. Potřebujeme tedy dotaz na datum.

```

SELECT "ID", "Date", "Amount", ( SELECT SUM( "Amount" ) FROM "Cash" WHERE
"Date" <= "a"."Date" ) AS "Balance" FROM "Cash" AS "a" ORDER BY "Date",
"ID" ASC

```

Výdaje se nyní zobrazují seřazené a sečtené podle data. Zbývá ještě otázka kategorie, protože chceme odpovídající zůstatky pro jednotlivé kategorie výdajů.

```

SELECT "ID", "Date", "Amount", "Acct_ID",
( SELECT "Acct" FROM "Acct" WHERE "ID" = "a"."Acct_ID" ) AS "Acct_name",
( SELECT SUM( "Amount" ) FROM "Cash" WHERE "Date" <= "a"."Date" AND
"Acct_ID" = "a"."Acct_ID" ) AS "Balance",
( SELECT SUM( "Amount" ) FROM "Cash" WHERE "Date" <= "a"."Date" ) AS
"Total_balance"
FROM "Cash" AS "a" ORDER BY "Date", "ID" ASC

```

Tím se vytvoří editovatelný dotaz, ve kterém se kromě vstupních polí (Date, Amount, Acct_ID) zobrazí společně název účtu, příslušný zůstatek a celkový zůstatek. Vzhledem k tomu, že korelační poddotazy jsou částečně založeny na předchozích položkách ("Date" <= "a"."Date"), projdou bez problémů pouze nové položky. Změny předchozího záznamu jsou zpočátku zjistitelné pouze v tomto záznamu. Má-li být později proveden výpočet závislý na dotazu, musí být dotaz aktualizován.

Číslování řádků

Pole s automatickým přírůstkem jsou v pořádku. Neříkají však s určitostí, kolik záznamů se v databázi nachází nebo kolik je jich skutečně k dispozici pro dotazování. Záznamy jsou často mazány a mnozí uživatelé se marně snaží zjistit, která čísla již neexistují, aby se průběžné číslo shodovalo.

```
SELECT "ID", ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <= "a"."ID" ) AS "Nr." FROM "Table" AS "a"
```

Pole ID je načteno a druhé pole je určeno korelačním poddotazem, který se snaží zjistit, kolik hodnot pole ID je menší nebo rovno aktuální hodnotě pole. Z toho se vytvoří číslo průběžného řádku.

Každý záznam, na který chceme tento dotaz použít, obsahuje pole. Chceme-li tento dotaz použít na záznamy, musíme nejprve do dotazu přidat tato pole. V klauzuli SELECT je můžeme umístit v libovolném pořadí. Pokud máme záznamy ve formuláři, musíme formulář upravit tak, aby data pro formulář pocházela z tohoto dotazu.

Například záznam obsahuje field1, field2 a field3. Úplný dotaz by tedy byl:

```
SELECT "ID", "field1", "field2", "field3", ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <= "a"."ID" ) AS "Nr." FROM "Table" AS "a"
```

Je možné také číslování pro odpovídající seskupení:

```
SELECT "ID", "Calculation", ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <= "a"."ID" AND "Calculation" = "a"."Calculation" ) AS "Nr." FROM "Table" AS "a" ORDER BY "ID" ASC, "Nr." ASC
```

Zde jedna tabulka obsahuje různá vypočtená čísla ("Calculation"). Pro každé vypočtené číslo je "Nr." vyjádřeno zvlášť vzestupně po seřazení podle pole ID. Tím se vytvoří číslování od 1 nahoru.

Má-li skutečné pořadí v dotazu souhlasit s čísly řádků, je třeba zmapovat vhodný typ řazení. Za tímto účelem musí mít pole řazení ve všech záznamech jedinečnou hodnotu. Jinak budou mít dvě uložená čísla stejnou hodnotu. To může být užitečné například při zobrazování pořadí v soutěži, protože shodné výsledky pak vedou ke společnému umístění. Aby bylo možné pořadí míst vyjádřit tak, že v případě společných pozic bude další hodnota vynechána, je třeba dotaz sestavit poněkud jinak:

```
SELECT "ID", ( SELECT COUNT( "ID" ) + 1 FROM "Table" WHERE "Time" < "a"."Time" ) AS "Place" FROM "Table" AS "a"
```

Vyhodnocují se všechny záznamy, pro které má pole Time menší hodnotu. To se týká všech sportovců, kteří dosáhli vítězného postu před současným sportovcem. K této hodnotě se přičte číslo 1. Tím se určí místo aktuálního sportovce. Pokud je čas shodný s časem jiného sportovce, umístí se společně. To umožňuje pořadí míst, jako je 1. místo, 2. místo, 2. místo, 4. místo.

Problematictější by bylo, kdyby se kromě pořadí míst vyžadovala i čísla řádků. To by mohlo být užitečné, pokud by bylo třeba spojit několik záznamů do jednoho řádku.

```
SELECT "ID", ( SELECT COUNT( "ID" ) + 1 FROM "Table" WHERE "Time" < "a"."Time" ) AS "Place",  
CASE WHEN
```

```
( SELECT COUNT( "ID" ) + 1 FROM "Table" WHERE "Time" = "a"."Time" ) = 1
THEN ( SELECT COUNT( "ID" ) + 1 FROM "Table" WHERE "Time" < "a"."Time" )
ELSE (SELECT ( SELECT COUNT( "ID" ) + 1 FROM "Table" WHERE "Time" <
"a"."Time" ) + COUNT( "ID" ) FROM "Table" WHERE "Time" = "a"."Time" "ID" <
"a"."ID"
END
AS "LineNumber" FROM "Table" AS "a"
```

Ve druhém sloupci je stále uvedeno pořadí míst. Ve třetím sloupci se nejprve zkontroluje, zda čáru s tímto časem překročila pouze jedna osoba. Pokud ano, pořadí míst se převede přímo na číslo řádku. V opačném případě se k umístění přidá další hodnota. Pro stejný čas ("Time" = "a"."Time") se přidá alespoň 1, pokud existuje další osoba s primárním klíčem ID, jejíž primární klíč je menší než primární klíč v aktuálním záznamu ("ID" < "a"."ID"). Tento dotaz tedy poskytuje stejné hodnoty pro pořadí míst, pokud neexistuje druhá osoba se stejným časem. Pokud existuje druhá osoba se stejným časem, ID určí, která osoba má menší číslo řádku.

Mimochodem, toto řazení podle čísla řádku může sloužit k jakémukoli účelu, který si uživatelé databáze přejí. Pokud je například řada záznamů řazena podle jména, nejsou záznamy se stejným jménem řazeny náhodně, ale podle svého primárního klíče, který je samozřejmě jedinečný. I tímto způsobem může číslování vést k třídění záznamů.

Číslování řádků je také dobrou předehrou pro spojování jednotlivých záznamů do jednoho záznamu. Pokud je dotaz na číslování řádků vytvořen jako pohled, lze na něj aplikovat další dotaz, aniž by způsobil nějaký problém. Jako jednoduchý příklad je zde opět první dotaz na číslování s jedním polem navíc:

```
SELECT "ID", "Name", ( SELECT COUNT( "ID" ) FROM "Table" WHERE "ID" <=
"a"."ID" ) AS "Nr." FROM "Table" AS "a"
```

Z tohoto dotazu se vytvoří pohled View1. Dotaz lze použít například pro spojení prvních tří jmen do jednoho řádku:

```
SELECT "Name" AS "Name_1", ( SELECT "Name" FROM "View1" WHERE "Nr." = 2 )
AS "Name_2", ( SELECT "Name" FROM "View1" WHERE "Nr." = 3 ) AS "Name_3"
FROM "View1" WHERE "Nr." = 1
```

Tímto způsobem lze několik záznamů převést do sousedních polí. Toto číslování probíhá jednoduše od prvního do posledního záznamu.

Pokud mají mít všechny tyto osoby stejné příjmení, lze to provést takto:

```
SELECT "ID", "Name", "Surname", ( SELECT COUNT( "ID" ) FROM "Table" WHERE
"ID" <= "a"."ID" AND "Surname" = "a"."Surname" ) AS "Nr." FROM "Table" AS "a"
```

Po vytvoření zobrazení lze sestavit rodinu.

```
SELECT "Surname", "Name" AS "Name_1", ( SELECT "Name" FROM "View1" WHERE
"Nr." = 2 AND "Surname" = "a"."Surname" ) AS "Name_2", ( SELECT "Name" FROM
"View1" WHERE "Nr." = 3 AND "Surname" = "a"."Surname" ) AS "Name_3" FROM
"View1" AS "a" WHERE "Nr." = 1
```

V adresáři tak mohou být shromážděni všichni členové jedné rodiny ("Surname"), takže při odesílání dopisu je třeba každou adresu zohlednit pouze jednou, ale jsou v ní uvedeni všichni, kteří mají dopis obdržet.

Zde musíme být opatrní, protože nechceme, aby se funkce donekonečna opakovala ve smyčce. Dotaz ve výše uvedeném příkladu omezuje počet paralelních záznamů, které mají být převedeny do polí, na 3. Tento limit byl zvolen záměrně. Žádná další jména se nezobrazí, i když je hodnota "Nr." větší než 3.

V několika případech je takové omezení jasně pochopitelné. Pokud například vytváříme kalendář, řádky mohou představovat týdny v roce a sloupce dny v týdnu. Stejně jako v původním kalendáři určuje obsah pole pouze datum, číslování řádků slouží k průběžnému číslování dnů každého týdne a týdny v roce se pak stávají záznamy. To vyžaduje, aby tabulka obsahovala pole data s průběžnými daty a pole pro události. Také nejstarší datum vždy vytvoří „Nr.“ = 1. Pokud tedy chceme, aby kalendář začínal v pondělí, musí být nejbližší datum v pondělí. Sloupec 1 je pak pondělí, sloupec 2 úterý atd. Poddotaz pak končí na "Nr." = 7. Tímto způsobem lze zobrazit všech sedm dní v týdnu vedle sebe a vytvořit odpovídající zobrazení kalendáře.

Získání zlomu řádku v dotazu

Někdy je užitečné sestavit několik polí pomocí dotazu a oddělit je zalomením řádku, například při načítání kompletní adresy do sestavy.

Zalomení řádku v dotazu je reprezentováno znakem Char (13). Příklad:

```
SELECT "Firstname" || ' ' || "Surname" || Char(13) || "Road" || Char(13) || "Town" FROM "Table"
```

Tím se získá:

```
Firstname Surname
Road
Town
```

Takový dotaz s číslováním řádků do 3 umožňuje tisknout adresní štítky ve třech sloupcích vytvořením sestavy. Číslování je v této souvislosti nutné, aby bylo možné v jednom záznamu umístit tři adresy vedle sebe. Jen tak zůstanou při čtení do zprávy vedle sebe.

V některých operačních systémech je nutné do kódu vedle char(13) zahrnout i char(10).

```
SELECT "Firstname" || ' ' || "Surname" || Char(13) || Char(10) || "Street" || Char(13) || Char(10) || "Town" FROM "Table"
```

Seskupování a souhrn

Pro ostatní databáze a novější verze HSQLDB je k dispozici příkaz Group_Concat(). Lze jej použít ke seskupení jednotlivých polí v záznamu do jednoho pole. Je tedy například možné ukládat křestní jména a příjmení do jedné tabulky a pak prezentovat data tak, že v jednom poli jsou příjmení zobrazena jako rodová jména, zatímco druhé pole obsahuje všechna příslušná křestní jména postupně oddělená čárkami.

Tento příklad je v mnoha ohledech podobný číslování řádků. Seskupení do společného pole je jakýmsi doplňkem.

Tabulka 20: Názvy osob, které mají být seskupeny

Surname	Firstname
Müller	Karin
Schneider	Gerd
Müller	Egon
Schneider	Volker
Müller	Monika
Müller	Rita

je dotazem převedeno na:

Tabulka 21: Jména seskupená podle příjmení

<i>Surname</i>	<i>Firstnames</i>
Müller	Karin, Egon, Monika, Rita
Schneider	Gerd, Volker

Tento postup lze v rámci možností vyjádřit v HSQLDB. Následující příklad se týká tabulky s názvem Name s poli ID, Firstname a Surname. Následující dotaz je nejprve spuštěn na tabulku a uložen jako pohled s názvem View_Group.

```
SELECT "Surname", "Firstname", ( SELECT COUNT( "ID" ) FROM "Name" WHERE "ID" <= "a"."ID" AND "Surname" = "a"."Surname" ) AS "GroupNr" FROM "Name" AS "a"
```

V kapitole Dotazy si můžeme přečíst, jak tento dotaz přistupuje k obsahu pole ve stejném řádku dotazu. Výsledkem je vzestupně číslovaná sekvence seskupená podle Surname. Toto číslování je nutné pro následující dotaz, takže v příkladu je uvedeno maximálně 5 křestních jmen.

```
SELECT "Surname",
( SELECT "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname" AND "GroupNr" = 1 ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname" AND "GroupNr" = 2 ), '' ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname" AND "GroupNr" = 3 ), '' ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname" AND "GroupNr" = 4 ), '' ) ||
IFNULL( ( SELECT ', ' || "Firstname" FROM "View_Group" WHERE "Surname" = "a"."Surname" AND "GroupNr" = 5 ), '' )
AS "Firstnames"
FROM "View_Group" AS "a"
```

Pomocí dílčích dotazů se postupně vyhledají a zkombinují křestní jména členů skupiny. Od druhého poddotazu musíme zajistit, aby hodnoty „NULL“ nenastavily celou kombinaci na hodnotu „NULL“. Proto se zobrazí výsledek „,“ a ne 'NULL'.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 9 *Makra*

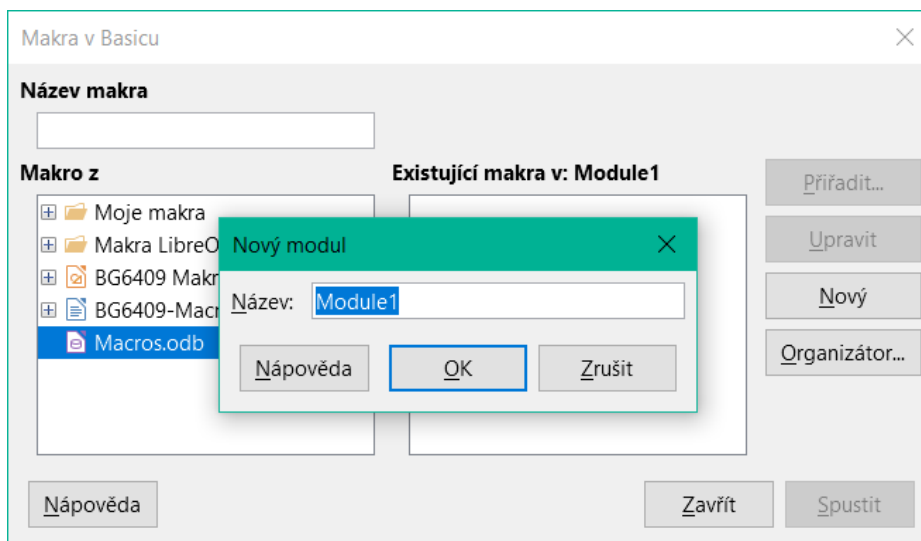
Obecné poznámky k makrům

Databázi v systému Base lze v zásadě spravovat bez maker. Někdy však mohou být nezbytná pro:

- Účinnější prevence chyb při zadávání.
- Zjednodušení některých úloh zpracování (přechod z jednoho formuláře do druhého, aktualizace dat po zadání do formuláře atd.).
- Snadnější vyvolání některých příkazů SQL než v samostatném editoru SQL.

Sami se musíme rozhodnout, jak intenzivně chceme makra v Base používat. Makra mohou zlepšit použitelnost, ale vždy jsou spojena s malým snížením rychlosti programu a někdy i s větším (při špatném kódování). Z dlouhodobého hlediska může také způsobit problémy s údržbou kódu. Vždy je lepší začít plným využitím možností databáze a nastavením konfigurace formulářů, než se pokusíme zajistit další funkce pomocí maker. Makra by se měla vždy testovat na větších databázích, aby se zjistil jejich vliv na výkon.

Makra jsou vytvořena pomocí **Nástroje > Makra > Správce maker > Základní**. Zobrazí se okno s přístupem ke všem makrům. V případě Base odpovídá důležitá oblast názvu souboru Base.



Obrázek 424: Dialogové okno Nový modul

Tlačítko **Nový** v dialogovém okně makra LibreOffice Basic otevře dialogové okno Nový modul, ve kterém je požadován název modulu (složka, ve které bude makro uloženo). Název lze v případě potřeby později změnit.

Jakmile je zadán, zobrazí se editor maker. Jeho vstupní oblast již obsahuje klauzule Start a End for podprogramu:

```
REM ***** BASIC *****  
  
Sub Main  
  
End Sub
```

Pokud chceme použít makra, je nutné provést následující kroky:

- Pomocí **Nástroje > Možnosti > Zabezpečení > Zabezpečení maker** by měla být úroveň zabezpečení snížena na střední. V případě potřeby můžeme na kartě Důvěryhodné zdroje navíc nastavit cestu k vlastním souborům maker, abychom zabránili pozdějším dotazům na aktivaci maker.
- Po vytvoření prvního modulu makra je nutné databázový soubor zavřít a znovu otevřít.

Některé základní zásady pro použití kódu Basic v LibreOffice:

- Řádky nemají ve výchozím nastavení žádná čísla řádků (ačkoli je možné je povolit) a musí končit zakončením řádku (hard return).
- U funkcí, rezervovaných výrazů a podobných prvků se nerozlišují velká a malá písmena. "String" je tedy totéž co "STRING" nebo "string" nebo jakákoli jiná kombinace velkých a malých písmen. Velká písmena by se měla používat pouze pro zlepšení čitelnosti. Názvy konstant a výčtů však při prvním průchodu překladačem maker rozlišují malá a velká písmena, proto je lepší je vždy psát se správnými velkými písmeny.
- Rozlišujeme procedury (začínající **Sub**) a funkce (začínající **Function**). Procedury byly původně programové segmenty bez návratové hodnoty, zatímco funkce vracejí hodnoty, které lze dále zpracovávat. Toto rozlišení však stále více ztrácí na významu. Lidé dnes používají termíny jako „metoda“ nebo „rutina“ bez ohledu na to, zda existuje návratová hodnota, nebo ne. Procedura může mít také návratovou hodnotu (kromě „Variant“).

`Sub myProcedure As Integer` 'Tato syntaxe pro Sub není oficiálně podporována.

`End Sub`

Další podrobnosti nalezneme v kapitole 13, Začínáme s makry, v příručce *Začínáme s LibreOffice*.

✓ Poznámka

Makra ve verzích PDF a ODT této kapitoly jsou barevně odlišena podle pravidel editoru maker LibreOffice:

Popis makra

Komentář makra

Operátor makra

Rezervovaný výraz makra

Číslo makra

Řetězec znaků makra

✓ Poznámka

V této kapitole je několik odkazů na *Odkazy na LibreOffice API*. V elektronické verzi kapitoly budou tyto odkazy hypertextové. Pro ty, kteří čtou tištěnou verzi, je odkaz na rozhraní API k dispozici na adrese <https://api.libreoffice.org/docs/idl/ref/>.

Makra v Base

Používání maker

„Přímý způsob“ pomocí **Nástroje > Makra > Spustit makro** je možný, ale pro základní makra není obvyklý. Makro je obvykle přiřazeno k události a spouští se, když tato událost nastane. Použití maker je na následující:

- Zpracování událostí ve formulářích.
- Úprava zdroje dat uvnitř formuláře.
- Přepínání mezi ovládacími prvky formuláře.
- Reakce na to, co uživatel dělá uvnitř ovládacího prvku.

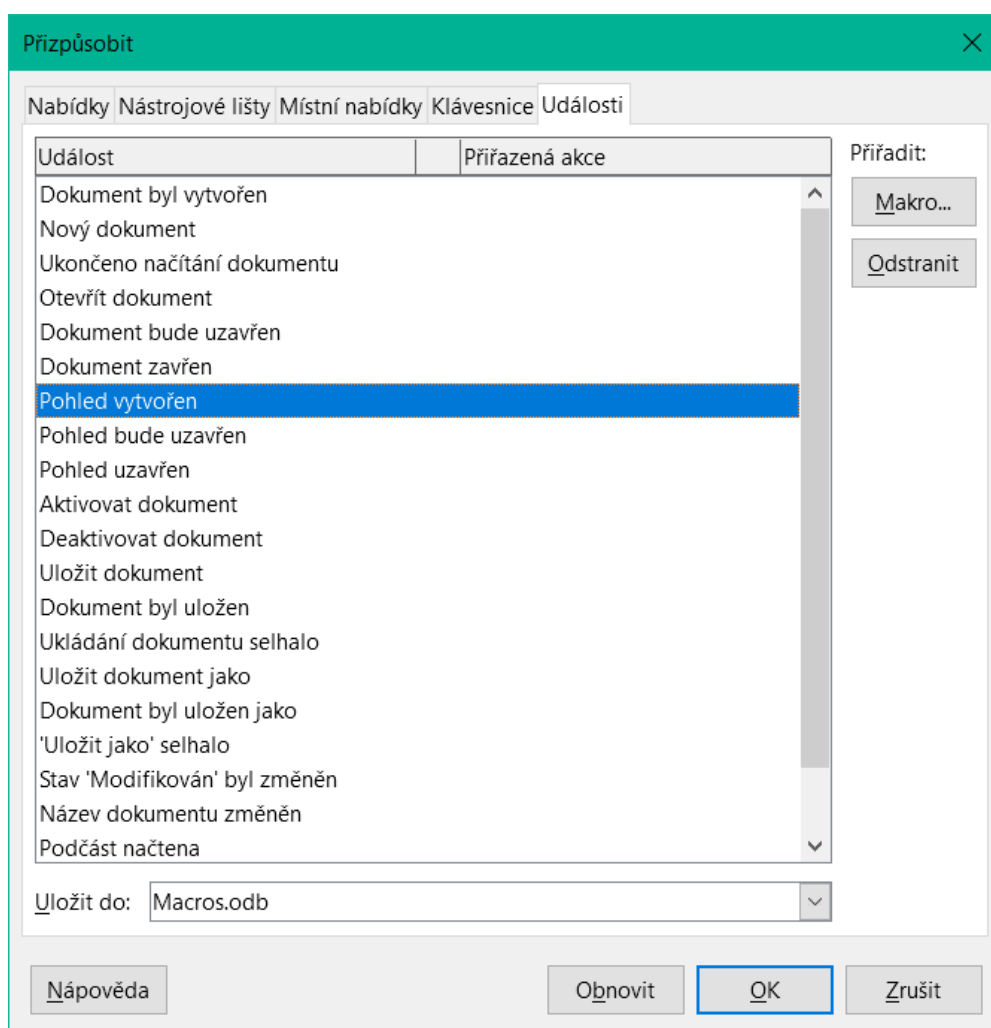
„Přímá cesta“ není možná, a to ani pro testování, když má být používán jeden z objektů **thisComponent** (viz „Přístup k formulářům“ na straně 409) nebo **oEvent** (viz „Přístup k prvkům formuláře“ na straně 410).

Přiřazení maker

Má-li být makro spuštěno událostí, musí být nejprve definováno. Poté jej lze přiřadit k události. K těmto událostem lze přistupovat na dvou místech.

Události, které nastanou ve formuláři při otevření nebo zavření okna.

Akce, které se provádějí při otevření nebo zavření formuláře, se zaznamenávají následujícím způsobem:



Obrázek 425: Dialogové okno Přizpůsobení, karta Události

- 1) Při návrhu formuláře otevřeme kartu Události v **Nástroje > Přizpůsobit**.
- 2) Vybereme příslušnou událost. Některá makra lze spustit pouze při zvolení události **Pohled vytvořen**. Jiná makra, například pro vytvoření celoobrazovkového formuláře, by měla být spuštěna příkazem **Otevřít dokument**.

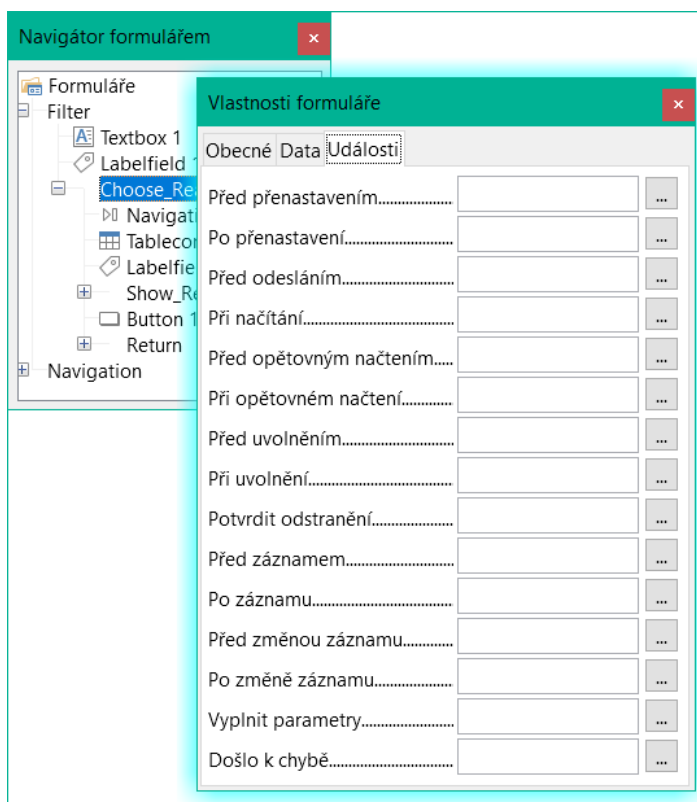
- 3) Pomocí tlačítka **Makro** vyhledáme požadované makro a potvrdíme výběr.
- 4) V části Uložit do zadáme název formuláře.
- 5) Potvrdíme pomocí **OK**.

Události ve formuláři v otevřeném okně

Po otevření okna, které zobrazuje celkový obsah formuláře, lze přistupovat k jednotlivým prvkům formuláře. To zahrnuje prvky, které jsme formuláři přiřadili.

K prvkům formuláře lze přistupovat pomocí Navigátoru formulářem, jak je znázorněno na obrázku níže. Stejně dobře k nim lze přistupovat pomocí místních nabídek jednotlivých ovládacích prvků v rozhraní formuláře.

Všechny události uvedené pod **Vlastnosti formuláře > Události** se odehrávají při otevření okna formuláře. Lze je nastavit samostatně pro každý formulář nebo podformulář v okně formuláře.



Obrázek 426: Dialogové okno Vlastnosti formuláře, karta Události

✓ Poznámka

Base používá slovo „formulář“ jak pro okno otevřené pro zadávání dat, tak pro prvky v tomto okně, které jsou vázány na konkrétní zdroj dat (tabulku nebo dotaz).

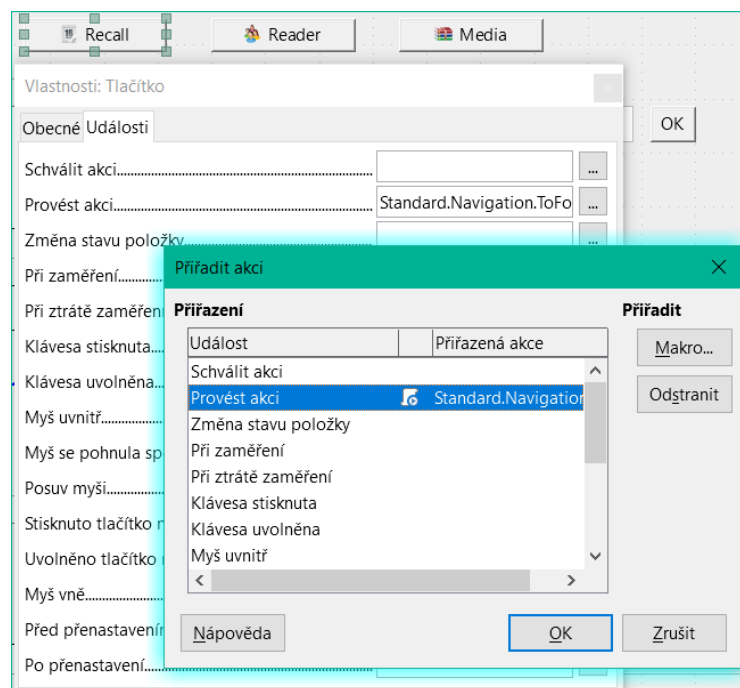
Jedno okno formuláře může obsahovat několik formulářů s různými zdroji dat. V Navigátoru formulářem se vždy nejprve zobrazí termín Formuláře, který v případě jednoduchého formuláře obsahuje pouze jednu podřízenou položku.

Události ve formuláři

Všechna ostatní makra se registrují pomocí vlastností podformulářů a ovládacích prvků na kartě Události.

- 1) Otevřeme okno vlastností ovládacího prvku (pokud jsme tak ještě neučinili).

- 2) Na kartě Události vybereme vhodnou událost.
- 3) K úpravě zdroje dat použijeme události, které odkazují na položku *Záznam*, *Aktualizovat* nebo *Obnovit*.
 - Pro tlačítka nebo volby v polích seznamu nebo možností by byla událost *Provést akci* první volbou.
 - Všechny ostatní události závisí na typu ovládání a požadované akci.
- 4) Kliknutím na tlačítko ... vpravo otevřeme dialogové okno *Přiřadit akci*.
- 5) Kliknutím na tlačítko **Makro** vybereme makro definované pro danou akci.
- 6) Kliknutím na **OK** potvrdíme přiřazení.



Obrázek 427: Dialogové okno *Přiřadit akci*

Součásti maker

V této části jsou vysvětleny některé makrojazyky, které se v Base běžně používají, zejména ve formulářích. Pokud je to možné (a rozumné), jsou příklady uvedeny ve všech následujících oddílech.

„Framework“ makra

Definice makra začíná jeho typem – **Sub** nebo **Function** – a končí **End Sub** nebo **End Function**. Makro, které je přiřazeno události, může přijímat argumenty (hodnoty); jediný užitečný je argument **oEvent**. Všechny ostatní rutiny, které by mohly být takovým makrem volány, mohou být definovány s návratovou hodnotou nebo bez ní, v závislosti na jejich účelu, a v případě potřeby opatřeny argumenty.

```
Sub update_loan
End Sub
Sub from_Form_to_Form(oEvent As Object)
End Sub
Function confirm_delete(oEvent As Object) As Boolean
    confirm_delete = False
```

End Function

Je užitečné si tento framework ihned zapsat a následně do něj vložit obsah. Nezapomeňte přidat komentáře k vysvětlení makra a pamatujte na pravidlo „Tolik, kolik je třeba, co nejméně je možné“. Jazyk Basic navíc nerozlišuje mezi velkými a malými písmeny. Obvykle se pevně stanovené pojmy jako **SUB** píšou přednostně velkými písmeny, ostatní pojmy smíšenými.

Definování proměnných

V dalším kroku, na začátku rutiny, se pomocí příkazu **Dim** definují proměnné, které se budou v rutině vyskytovat, každá s příslušným datovým typem. Samotný Basic to nevyžaduje; akceptuje všechny nové proměnné, které se v programu vyskytnou. Programový kód je však „bezpečnější“, pokud jsou proměnné, zejména jejich datové typy, deklarovány. Mnozí programátoři to vyžadují a při psaní modulu používají volbu Explicit jazyka Basic. To znamená „Neuznávej žádnou starou proměnnou, ale pouze tu, kterou jsem předem deklaroval“.

```
Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm As Object
Dim sName As String
Dim bOKEnabled As Boolean
Dim iCounter As Integer
Dim dBirthday As Date
```

V názvech proměnných lze používat pouze abecední znaky (A-Z nebo a-z), čísla a znak podtržení „_“. Žádné speciální znaky nejsou povoleny. Mezery jsou za určitých podmínek povoleny, ale je lepší se jim vyhnout. První znak musí být abecední.

Běžnou praxí je uvádět datový typ v prvním znaku. Pak ji lze rozpoznat všude, kde se proměnná v kódu vyskytuje. Doporučují se také „expresivní názvy“, aby byl význam proměnné zřejmý z jejího názvu.



Poznámka

Pokud je to možné, měli bychom to upřesnit, protože pouze jedno písmeno neumožňuje rozlišit mezi datovými typy „Double“ a „Date“ nebo „Single“ a „String“.

Seznam možných datových typů v jazyce Star Basic najdeme v příloze A této knihy. Na různých místech se liší od typů v databázi a v rozhraní API (Application Programming Interface) LibreOffice. Tyto změny jsou jasně uvedeny v příkladech.

Definování polí

Zejména u databází je důležité sestavení několika proměnných do záznamu. Pokud je několik proměnných uloženo společně na jednom společném místě, nazývá se pole. Před zápisem dat do pole musí být pole definováno.

```
Dim arData()
```

vytvoří prázdné pole.

```
arData = Array("Lisa", "Schmidt")
```

vytvoří pole určité velikosti (2 prvky) a předá mu hodnoty.

Použití

```
Print arData(0), arData(1)
```

způsobí, že se na obrazovce zobrazí dva definované prvky. Počet prvků začíná číslem 0.


```
Dim arData(2)
arData(0) = "Lisa"
arData(1) = "Schmidt"
arData(2) = "Cologne"
```

Tím se vytvoří pole, do kterého lze uložit tři prvky libovolného typu, například záznam pro "Lisa""Schmidt""Cologne". Do tohoto pole nelze vložit více než tři prvky. Pokud chceme uložit více prvků, musíme pole zvětšit. Pokud je však velikost pole předdefinováno za běhu makra, je pole zpočátku prázdné, stejně jako nové pole.

```
ReDim Preserve arData(3)
arData(3) = "18.07.2003"
```

Přidáním **Preserve** se zachovávají předchozí údaje, takže pole je skutečně rozšířeno o položku data (zde ve formě textu).

Výše uvedené pole může uchovávat pouze jeden záznam. Pokud chceme uložit několik záznamů, jako je tomu u tabulky v databázi, musíme definovat dvourozměrné pole.

```
Dim arData(2,1)
arData(0,0) = "Lisa"
arData(1,0) = "Schmidt"
arData(2,0) = "Cologne"
arData(0,1) = "Egon"
arData(1,1) = "Müller"
arData(2,1) = "Hamburg"
```

I zde je možné rozšířit dříve definované pole a zachovat stávající obsah pomocí **Preserve**.

Přístup k formulářům

Formulář leží v aktuálně aktivním dokumentu. Oblast, která je zde reprezentována, se nazývá **drawpage**. Kontejner, ve kterém jsou uloženy všechny formuláře, se nazývá **forms**; v Navigátoru formulářů se zobrazuje jako primární záhlaví s připojenými jednotlivými formuláři. Výše uvedené proměnné získávají své hodnoty takto:

```
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("Filter")
```

Formulář, ke kterému se má přistupovat, se nazývá Filter. Jedná se o název, který je viditelný v nejvyšší úrovni Navigátoru formulářem (ve výchozím nastavení se první formulář jmenuje MainForm). Dílčí formuláře jsou v rámci hlavního formuláře hierarchicky uspořádány a lze k nim přistupovat postupně:

```
Dim oSubForm As Object
Dim oSubSubForm As Object
oSubForm = oForm.getByName("Readerselect")
oSubSubForm = oSubForm.getByName("Readerdisplay")
```

Namísto použití přechodných proměnných můžeme přejít přímo na konkrétní formulář. Přechodný objekt, který může být použit více než jednou, je třeba deklarovat a přiřadit mu samostatnou hodnotu. V následujícím příkladu se **oSubForm** již nepoužívá.

```
oForm = thisComponent.drawpage.forms.getByName("Filter")
```

```
oSubSubForm =  
oForm.getByName("readerselect").getByName("readerdisplay")
```

✓ Poznámka

Pokud se název skládá pouze z písmen ascii a čísel bez mezer a speciálních znaků, lze jej použít přímo v příkazu přiřazení.

```
oForm = thisComponent.drawpage.forms.Filter  
oSubSubForm = oForm.readerselect.readerdisplay
```

Na rozdíl od běžného používání jazyka Basic je nutné tyto názvy psát se správnými velkými a malými písmeny.

Jiný způsob přístupu k formuláři poskytuje událost, která makro spouští.

Pokud je makro spuštěno z události formuláře, například **Vlastnosti formuláře > Před záznamem**, lze se k samotnému formuláři dostat následujícím způsobem:

```
Sub MacroexampleCalc(oEvent As Object)  
    oForm = oEvent.Source  
    ...  
End Sub
```

Pokud je makro spuštěno z události na ovládacím prvku formuláře, například **Textové pole > Při ztrátě zaměření**, zpřístupní se formulář i pole:

```
Sub MacroexampleCalc(oEvent As Object)  
    oField = oEvent.Source.Model  
    oForm = oField.Parent  
    ...  
End Sub
```

Přístup k událostem má tu výhodu, že se nemusíme starat o to, zda se jedná o hlavní nebo podformulář. Také název formuláře nemá pro fungování makra žádný význam.

Přístup k prvkům formuláře

K prvkům ve formulářích se přistupuje podobným způsobem: deklaruujeme vhodnou proměnnou jako **objekt** a vyhledáme příslušný ovládací prvek ve formuláři:

```
Dim btnOK As Object ' Button »OK«  
btnOK = oSubSubForm.getByName("button 1") ' z formuláře readerdisplay
```

Tato metoda funguje vždy, když víme, se kterým prvkem má makro pracovat. Pokud je však prvním krokem určení, která událost makro spustila, je užitečná výše uvedená metoda **oEvent**. Proměnná je deklarována v rámci makra „framework“ a při spuštění makra je jí přiřazena hodnota. Vlastnost **Source** vždy uvádí prvek, který makro spustil, zatímco vlastnost **Model** podrobně popisuje ovládací prvek:

```
Sub confirm_choice(oEvent As Object)  
    Dim btnOK As Object  
    btnOK = oEvent.Source.Model  
End Sub
```

Pokud chceme, můžeme s objektem získaným touto metodou provádět další akce.

Upozorňujeme, že podformuláře se počítají jako součásti formuláře.

Přístup do databáze

Přístup k databázi se obvykle řídí pomocí formulářů, dotazů, sestav nebo funkce hromadné korespondence (mailmerge), jak bylo popsáno v předchozích kapitolách. Pokud se tyto možnosti ukážou jako nedostatečné, může makro přistupovat k databázi několika způsoby.

Připojení k databázi

Nejjednodušší metoda používá stejné připojení jako formulář. **oForm** se určí podle výše uvedeného.

```
Dim oConnection As Object
oConnection = oForm.activeConnection()
```

Nebo můžeme zdroj dat (tj. databázi) načíst prostřednictvím dokumentu a pro makro použít jeho existující připojení:

```
Dim oDatasource As Object
Dim oConnection As Object
oDatasource = thisComponent.Parent.dataSource
oConnection = oDatasource.getConnection("", "")
```

Další způsob umožňuje vytvořit připojení k databázi za běhu:

```
Dim oDatasource As Object
Dim oConnection As Object
oDatasource = thisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then oDatasource.connect()
oConnection = oDatasource.ActiveConnection()
```

Podmínka **If** řídí pouze jeden řádek, takže **End If** není nutná.

Pokud má být makro spuštěno prostřednictvím uživatelského rozhraní, a ne z události ve formuláři, je vhodná následující varianta:

```
Dim oDatasource As Object
Dim oConnection As Object
oDatasource = thisDatabaseDocument.CurrentController
If Not (oDatasource.isConnected()) Then oDatasource.connect()
oConnection = oDatasource.ActiveConnection()
```

Přístup k databázím mimo aktuální databázi je možný následujícím způsobem:

```
Dim oDatabaseContext As Object
Dim oDatasource As Object
Dim oConnection As Object
oDatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
oDatasource = oDatabaseContext.getByName("registered name of Database in LO")
oConnection = oDatasource.GetConnection("", "")
```

Připojení k databázím, které nejsou registrovány v LibreOffice, jsou také možná. V takových případech musí být místo registrovaného názvu uvedena cesta k databázi jako `file:///...../database.odt`.

Rozšířené pokyny pro připojení k databázi jsou uvedeny v části „Vytvoření připojení k databázi“ (strana 471).

Příkazy SQL

S databází pracujeme pomocí příkazů SQL. Ty je třeba vytvořit a odeslat do databáze; výsledek se určí podle typu příkazu a výsledky lze dále zpracovávat. Direktiva `createStatement` vytvoří k tomuto účelu vhodný objekt.

```
Dim oSQL_Statement As Object ' objekt, který provede SQL příkaz
Dim stSql As String          ' Text aktuálního SQL příkazu
Dim oResult As Object       ' výsledek executeQuery
Dim iResult As Integer      ' výsledek executeUpdate
oSQL_Statement = oConnection.createStatement()
```

Pro *dotaz na data* zavoláte metodu `executeQuery`; výsledek se pak vyhodnotí. Názvy tabulek a polí se obvykle uvádějí s dvojitými uvozovkami. Makro je musí maskovat dalšími dvojitými uvozovkami, aby se v příkazu objevily.

```
stSql = "SELECT * FROM ""Table1""
oResult = oSQL_Statement.executeQuery(stSql)
```

Chceme-li *upravit data* – tedy `INSERT`, `UPDATE` nebo `DELETE` – *nebo ovlivnit* strukturu databáze, zavoláme metodu `executeUpdate`. V závislosti na příkazu a databázi se buď nezjistí nic užitečného (nula), nebo se zjistí počet změněných záznamů.

```
stSql = "DROP TABLE ""Suchtmp"" IF EXISTS"
iResult = oSQL_Statement.executeUpdate(stSql)
```

Pro úplnost je třeba zmínit ještě jeden speciální případ: má-li být `oSQL_Statement` používán různými způsoby pro `SELECT` nebo pro jiné účely, je k dispozici ještě jedna metoda, a to `execute`. Nebudeme ji zde používat. Další informace nalezneme v části [Odkazy API](#).

Předpřipravené příkazy SQL s parametry

Ve všech případech, kdy je třeba do příkazu SQL převést ruční zadání uživatele, je jednodušší a bezpečnější nevytvářet příkaz jako dlouhý řetězec znaků, ale připravit si jej předem a použít jej s parametry. To usnadňuje formátování čísel, dat a řetězců (zmizí neustálé dvojitě uvozené) a zabraňuje ztrátě dat při nekorektním vstupu.

Při použití této metody se vytvoří a připraví objekt pro konkrétní příkaz SQL:

```
Dim oSQL_Statement As Object ' objekt, který provede SQL příkaz
Dim stSql As String          ' Text aktuálního SQL příkazu
stSql = "UPDATE author " _
        & "SET lastname = ?, firstname = ?" _
        & "WHERE ID = ?"
oSQL_Statement = oConnection.prepareStatement(stSql)
```

Objekt je vytvořen pomocí `prepareStatement`, takže příkaz SQL je předem znám. Každý otazník označuje pozici, která později – před provedením příkazu – získá skutečnou hodnotu. Protože je příkaz připraven předem, databáze ví, jaký typ záznamu – v tomto případě dva řetězce a číslo – se očekává. Jednotlivé pozice jsou rozlišeny číslem (počítáno od 1).

Poté se hodnoty přenesou pomocí vhodných příkazů a provede se příkaz SQL. Zde jsou hodnoty převzaty z ovládacích prvků formuláře, ale mohou pocházet i z jiných maker nebo mohou být zadány jako prostý text:

```
oSQL_Statement.setString(1, oTextfeld1.Text) ' Text pro příjmení
(surname)
oSQL_Statement.setString(2, oTextfeld2.Text) ' Text pro křestní jméno
(first name)
oSQL_Statement.setLong(3, oZahlenfeld1.Value) ' hodnota pro odpovídající
ID
iResult = oSQL_Statement.executeUpdate
```

Úplný seznam přiřazení je uveden v části "Parametry pro připravené příkazy SQL" (strana 429).

Další informace o výhodách této metody nalezneme níže (externí odkazy):

- SQL injection (https://en.wikipedia.org/wiki/SQL_injection)
- Proč používat PreparedStatement v Java JDBC (<https://javarevisited.blogspot.com/2012/03/why-use-preparedstatement-in-java-jdbc.html>)
- Structured Query Language (https://en.wikibooks.org/wiki/Structured_Query_Language)

Čtení a používání záznamů

V závislosti na požadavcích existuje několik způsobů, jak přenést informace z databáze do makra, aby mohly být dále zpracovány.

Upozornění: odkazy na formulář zahrnují i podformuláře. Tím je myšlen formulář nebo část formuláře, která je vázána na určitý zdroj dat.

Používání formulářů

Aktuální záznam a jeho data jsou vždy k dispozici prostřednictvím formuláře, který zobrazuje příslušná data (tabulka, dotaz, SELECT). Existuje několik metod typu `getData_`, jako např.:

```
Dim ID As Long
Dim sName As String
Dim dValue As Currency
Dim dEntry As New com.sun.star.util.Date
ID = oForm.getLong(1)
sName = oForm.getString(2)
dValue = oForm.getDouble(4)
dEntry = oForm.getDate(7)
```

Všechny tyto metody vyžadují číslo sloupce ze zdroje dat; počítadlo začíná od 1.



Poznámka

U všech metod, které pracují s databázemi, začíná počítání od hodnoty 1. To platí pro sloupce i řádky.

Pokud pro práci s podkladovým zdrojem dat (tabulkou, dotazem, pohledem) dáváme přednost použití názvů sloupců před jejich čísly, lze číslo sloupce určit pomocí `findColumn`. Zde je příklad pro vyhledání sloupce s názvem Name.

```
Dim sName As String
```

```
nName = oForm.findColumn("Name")
sName = oForm.getString(nName)
```

Typ vrácené hodnoty vždy odpovídá typu metody, je však třeba upozornit na následující zvláštní případy:

- Pro data typu **Decimal**, **Currency** apod., která se používají pro komerčně přesné výpočty, neexistují žádné metody. Protože Basic automaticky provede příslušný převod, můžeme použít **getDouble**.
- Při použití **getBoolean** je třeba vzít v úvahu způsob, jakým jsou v databázi definovány hodnoty TRUE a FALSE. Obvyklé definice (logické hodnoty, 1 jako TRUE) jsou zpracovány správně.
- Hodnoty data lze definovat nejen pomocí datového typu **Date**, ale také (jak je uvedeno výše) jako **util.Date**. To usnadňuje čtení a úpravu roku, měsíce a dne.
- U celých čísel si dáme pozor na různé datové typy. Výše uvedené příklady používají **getLong**; ID proměnné Basicu musí mít také datový typ **Long**, protože to odpovídá datovému typu **Integer** v databázi.

Úplný seznam těchto metod najdeme v části "Úprava řádků dat" (strana 427).

Tip

Pokud mají být hodnoty z formuláře použity přímo pro další zpracování v SQL (například pro vstup do jiné tabulky), je mnohem jednodušší nedotazovat se na typ pole.

Následující makro, které je vázáno na **Vlastnosti: Tlačítko > Události > Provést akci** načte první pole ve formuláři nezávisle na typu, který je nutný pro další zpracování v Basicu.

```
SUB ReadValues(oEvent As Object)
    Dim oForm As Object
    Dim stFeld1 As String
    oForm = oEvent.Source.Model.Parent
    stFeld1 = oForm.getString(1)
End Sub
```

Pokud jsou pole načtena pomocí **getString()**, je zachováno veškeré formátování potřebné pro další zpracování SQL. Datum, které se zobrazí jako 08.03.19, se načte ve formátu 2019-03-08 a lze jej použít přímo v SQL.

Čtení ve formátu odpovídající typu je povinné pouze tehdy, má-li být hodnota dále zpracována v rámci makra, například při výpočtu.

Výsledek dotazu

Stejným způsobem lze použít i sadu výsledků dotazu. V sekci *Příkazy SQL* najdeme pro tuto sadu výsledků proměnnou **oResult**, která se obvykle čte takto:

```
While oResult.next ' jeden záznam za druhým
    REM převádí výsledek do proměnných
    stVar = oResult.getString(1)
    inVar = oResult.getLong(2)
    boVar = oResult.getBoolean(3)
    REM udělá něco s těmito hodnotami
Wend
```

Podle typu SQL příkazu, očekávaného výsledku a jeho účelu lze smyčku **WHILE** zkrátit nebo zcela vypustit. V zásadě lze ale vždy takto vyhodnotit množinu výsledků.

Pokud se má vyhodnotit pouze první záznam.

```
oResult.next
```

přístupuje k řádku tohoto záznamu a pomocí

```
stVar = oResult.getString(1)
```

přečte obsah prvního pole. Zde smyčka končí.

Dotaz pro výše uvedený příklad má v prvním sloupci text, ve druhém celé číslo (**Integer** v databázi odpovídá **Long** v jazyce Basic) a ve třetím pole Ano/Ne. K polím se přístupuje pomocí indexu pole, který na rozdíl od indexu pole začíná od 1.

Navigace přes takový výsledek není možná. Povoleny jsou pouze jednotlivé kroky k dalšímu záznamu. Aby bylo možné se v záznamu pohybovat, musí být při vytváření dotazu znám **ResultSetType**. K tomu se přístupuje pomocí

```
oSQL_Result.ResultSetType = 1004
```

nebo

```
oSQL_Result.ResultSetType = 1005
```

Typ **1004** – **SCROLL_INTENSIVE** umožňuje volný pohyb, ale nezachycuje změny v původních datech. Typ **1005** – **SCROLL_SENSITIVE** rozpozná změny v původních datech, které mohou ovlivnit výsledek dotazu.

Celkový počet řádků v sadě výsledků lze určit až po zadání číselného typu výsledku. Provádí se takto:

```
Dim iResult As Long
If oResult.last           ' přejde na poslední záznam, pokud je to možné
    iResult = oResult.getRow ' průběžné číslo je součet
Else
    iResult = 0
End If
```

Použití ovládacího prvku

Pokud je ovládací prvek vázán na zdroj dat, lze hodnotu načíst přímo, jak je popsáno v následující části. To však může vést k problémům. Bezpečnější je použít postup popsany v kapitole „Používání formulářů“ (strana 413) nebo následující metodu, která je uvedena pro několik různých typů ovládaní:

```
sValue = oTextField.BoundField.Text           ' příklad pro Textové pole
nValue = oNumericField.BoundField.Value       ' příklad číselného pole
dValue = oDateField.BoundField.Date           ' příklad pole pro datum
```

BoundField představuje spojení mezi viditelným ovládacím prvkem a skutečným obsahem datové sady.

Navigace v datové sadě

V předposledním příkladu byla použita metoda **Next** pro přechod z jednoho řádku výsledkové sady na další. Existují další podobné metody a testy, které lze použít jak pro data ve formuláři – reprezentovaná proměnnou **oForm** –, tak pro množinu výsledků. Například pomocí metody

popsané v části „Automatická aktualizace formulářů“ (strana 430) lze znovu vybrat předchozí záznam:

```
Dim loRow As Long
loRow = oForm.getRow() ' uloží aktuální číslo řádku
oForm.reload()        ' znovu načte množinu záznamů
oForm.absolute(loRow) ' vrátí se zpět ke stejnému řádku
```

V části „Automatická aktualizace formulářů“ jsou uvedeny všechny metody, které jsou k tomu vhodné.

Poznámka

Existuje chyba, která ovlivňuje formuláře. Při změně dat ve formuláři nastaví aktuální číslo řádku na hodnotu „0“. Viz https://bugs.documentfoundation.org/show_bug.cgi?id=82591. Chceme-li získat správné číslo aktuálního řádku, přiřadíme k události **Formulář > Vlastnosti > Události > Po změně záznamu** následující makro.

```
Global loRow As Long
Sub RowCounter(oEvent As Object)
    loRow = oEvent.Source.Row
End Sub
```

Nové číslo řádku se načte a přiřadí do globální proměnné **loRow**. Tato proměnná má být umístěna na začátku všech modulů a zachová si svůj obsah, dokud neukončíme aplikaci Base nebo nezměníme hodnotu opětovným voláním **RowCounter**.

Editace záznamů – přidávání, úprava, mazání

Aby bylo možné záznamy upravovat, musí spolupracovat několik věcí:

- Informace musí uživatel zadat do ovládacího prvku pomocí klávesnice.
- Datová sada za formulářem musí být o změně informována. K tomu dochází, když se přesuneme z pole do nového pole.
- Je třeba upravit samotnou databázi. K tomu dochází při přechodu z jednoho záznamu na druhý.

Při provádění tohoto postupu pomocí makra je třeba vzít v úvahu všechny tyto dílčí kroky. Pokud některý z nich chybí nebo je proveden chybně, změny se ztratí a v databázi se neobjeví. Především se změna nesmí týkat zobrazené hodnoty ovládacího prvku, ale samotné sady dat. Proto je zbytečné měnit vlastnost **Text** ovládacího prvku.

Upozorňujeme, že tabulky jsou jediné datové sady, které lze měnit, aniž by to způsobilo problémy. U ostatních datových sad jsou úpravy možné pouze za zvláštních okolností.

Změna obsahu ovládacího prvku

Pokud chceme změnit pouze jednu hodnotu, lze použít vlastnost **BoundField** ovládacího prvku s vhodnou metodou. Poté musí být změna přenesena do databáze. Zde je příklad pole pro datum, do kterého se zadává skutečné datum:

```
Dim unoDate As New com.sun.star.util.Date
unoDate.Year = Year(Date)
unoDate.Month = Month(Date)
unoDate.Day = Day(Date)
oDateField.BoundField.updateDate( unoDate )
```



```
oForm.updateRow() ' změna je přenesena do databáze
```

Pro **BoundField** se použije metoda **updateXxx**, která odpovídá datovému typu pole. V tomto příkladu se jedná o pole **Date**. Jako argument se předává nová hodnota – v tomto případě aktuální datum převedené do formátu, který makro vyžaduje.

Změna řádků v datové sadě

Předchozí metoda je nevhodná, pokud je třeba změnit několik hodnot v řádku. Pro každé pole by totiž musel existovat ve formuláři ovládací prvek, což často není žádoucí ani užitečné. Pro každé pole musí být také načten objekt. Jednoduchý a přímý způsob používá tento formulář:

```
Dim unoDate As New com.sun.star.util.Date
unoDate.Year = Year(Date)
unoDate.Month = Month(Date)
unoDate.Day = Day(Date)
oForm.updateDate(3, unoDate )
oForm.updateString(4, "ein Text")
oForm.updateDouble(6, 3.14)
oForm.updateInt(7, 16)
oForm.updateRow()
```

Pro každý sloupec v datové sadě se zavolá metoda **updateXxx** odpovídající jeho typu. Argumenty jsou číslo sloupce (počítáno od 1) a požadovaná hodnota. Poté se změny přenesou do databáze.

Vytváření, úprava a odstraňování řádků

Pojmenované změny se vztahují k aktuálnímu řádku datové sady, která je základem formuláře. Za určitých okolností je nutné zavolat metodu z kapitoly „Navigace v datové sadě“ (strana 426). Je třeba provést následující kroky:

- 1) Vybereme aktuální záznam.
- 2) Změníme hodnoty podle popisu v předchozí části.
- 3) Změnu potvrdíme následujícím příkazem:
`oForm.updateRow()`
- 4) Ve zvláštních případech je možné zrušit a vrátit se do předchozího stavu:
`oForm.cancelRowUpdates()`

Pro nový záznam existuje speciální metoda srovnatelná s přechodem na nový řádek v ovládacím prvku tabulky. To se provádí následujícím způsobem:

- 1) Připravíme se na nový záznam:
`oForm.moveToInsertRow()`
- 2) Zadáme všechny potřebné/požadované hodnoty. To se provádí pomocí metod **updateXxx**, jak je uvedeno v předchozí části.
- 3) Nová data potvrdíme následujícím příkazem:
`oForm.insertRow()`
- 4) Nový záznam nelze snadno vrátit zpět. Místo toho budeme muset nový záznam odstranit.

Pro odstranění záznamu existuje jednoduchý příkaz; postupujeme takto:

- 1) Zvolíme požadovaný záznam a provedeme jeho aktualizaci, stejně jako v případě změny.
- 2) K jeho odstranění použijeme následující příkaz:
`oForm.deleteRow()`

Tip

Aby bylo zajištěno, že se změny přenesou do databáze, je třeba je explicitně potvrdit pomocí **updateRow** nebo **insertRow**. Zatímco při stisknutí tlačítka Uložit se automaticky použije příslušná funkce, u makra je třeba před uložením určit, zda se jedná o nový záznam (**Insert**) nebo o úpravu stávajícího (**Update**).

```
If oForm.isNew Then
    oForm.insertRow()
Else
    oForm.updateRow()
End If
```

Testování a změna ovládacích prvků

Kromě obsahu datové sady lze z kontroly vyčíst, upravit a modifikovat mnohem více informací. To platí zejména pro vlastnosti, jak je popsáno v kapitole 4, Formuláře.

Několik příkladů v části „Zlepšení použitelnosti“ (strana 430) používá doplňkové informace v poli:

```
Dim stTag As String
stTag = oEvent.Source.Model.Tag
```

Jak bylo uvedeno v předchozí části, vlastnost **Text** lze užitečně měnit pouze tehdy, pokud ovládací prvek není vázán na datovou sadu. Existují však i další vlastnosti, které jsou určeny jako součást definice formuláře, ale lze je upravit za běhu. Například popisek může mít jinou barvu textu, pokud představuje spíše varování než informaci:

```
Sub showWarning(oField As Object, iType As Integer)
    Select Case iType
        Case 1
            oField.TextColor = RGB(0, 0, 255) ' 1 = blue
        Case 2
            oField.TextColor = RGB(255, 0, 0) ' 2 = red
        Case Else
            oField.TextColor = RGB(0, 255, 0) ' 0 = green (neither 1 nor 2)
    End Select
End Sub
```

Anglické názvy v makrech

Zatímco návrhář formuláře může pro vlastnosti a přístup k datům používat označení v rodném jazyce, v jazyce Basic lze používat pouze anglické názvy. Ty jsou uvedeny v následujícím přehledu.

Vlastnosti, které se obvykle nastavují pouze v definici formuláře, zde nejsou obsaženy. Stejně tak metody (funkce a/nebo procedury), které se používají jen zřídka nebo jsou vyžadovány pouze u složitějších deklamací.

Přehled obsahuje následující informace:

- Název – Název, který se použije pro vlastnost v makrokódu
- Datový typ – Datový typ Basic. U funkcí je to návratový typ. Není zahrnuto pro procedury.
- R/W – Označuje způsob použití hodnoty:

- R – pouze pro čtení
- W – pouze zápis (úprava)
- (R) – Čtení možné, nevhodné k úpravám
- (W) – Psaní je možné, ale není vhodné
- R+W – vhodné pro čtení a psaní

Další informace nalezneme v [Přehledu API](#) po vyhledání anglického názvu ovládacího prvku. Existuje užitečný nástroj Xray, který umožňuje zjistit, které vlastnosti a metody jsou pro daný prvek k dispozici.

```
Sub Main(oEvent)
    Xray(oEvent)
End Sub
```

Tím se spustí rozšíření Xray pro daný argument.

Vlastnosti formulářů a ovládacích prvků

Model ovládacího prvku popisuje jeho vlastnosti. Podle situace může být hodnota vlastnosti přístupná pouze pro čtení nebo pouze pro zápis. Pořadí je stejné jako v seznamech „Vlastnosti řídicích polí“ v kapitole 4, Formuláře.

Písmo

V každém ovládacím prvku, který zobrazuje text, lze přizpůsobit vlastnosti písma.

Tabulka 22: Vlastnosti písma

Název	Datový typ	R/W	Vlastnost
FontName	string	R+W	Název písma
FontHeight	single	R+W	Velikost písma
FontWeight	single	R+W	Tučné nebo normální
FontSlant	integer	R+W	Kurzíva nebo latinka
FontUnderline	integer	R+W	Podtržené či nikoliv
FontStrikeout	integer	R+W	Přeškrtnuté či nikoliv

Vzorec

Anglický termín: Form

Tabulka 23: Vlastnosti vzorce

Název	Datový typ	R/W	Vlastnost
ApplyFilter	boolean	R+W	Použitý filtr
Filtr	string	R+W	Aktuální filtr pro záznam
FetchSize	long	R+W	Počet záznamů načtených najednou
Row	long	R	Aktuální číslo řádku
RowCount	long	R	Počet záznamů

Tyto vlastnosti platí pro všechny ovládací prvky

Ovládací prvek – viz také FormComponent

Tabulka 24: Vlastnosti platné pro všechny ovládací prvky

Název	Datový typ	R/W	Vlastnost
Název	string	R+(W)	Název pole
Enabled	boolean	R+W	Aktivní: Pole lze vybrat.
EnableVisible	boolean	R+W	Zobrazí se pole.
ReadOnly	boolean	R+W	Obsah pole nelze měnit.
TabStop	boolean	R+W	Do pole se dostaneme pomocí klávesy Tab
Zarovnání	integer	R+W	Vodorovné zarovnání: 0 = vlevo, 1 = na střed, 2 = vpravo
BackgroundColor	long	R+W	Barva pozadí
Tag	string	R+W	Další informace
HelpText	string	R+W	Text nápovědy jako Tip zobrazený po najetí

Ty se vztahují na mnoho typů ovládacích prvků

Tabulka 25: Vlastnosti platné pro mnoho ovládacích prvků

Název	Datový typ	R/W	Vlastnost
Text	string	(R+W)	Zobrazený obsah pole. V textových polích je lze přečíst a dále zpracovat, ale u jiných typů to obvykle nefunguje.
Spin	boolean	R+W	Číselník začleněný do formátovaného pole.
TextColor	long	R+W	Barva textu (popředí).
DataField	string	R	Název pole v datové sadě.
BoundField	objekt	R	Objekt představující připojení k datové sadě a poskytující přístup k obsahu pole.

Textové pole – další vlastnosti (TextField)

Tabulka 26: Další vlastnosti textového pole

Název	Datový typ	R/W	Vlastnost
String	string	R+W	Zobrazený obsah pole.
MaxTextLen	integer	R+W	Maximální délka textu.
DefaultText	string	R+W	Výchozí text.
MultiLine	boolean	R+W	Označuje, zda existuje více než jeden řádek.
EchoChar	(integer)	R+W	Znak zobrazený při zadávání hesla.

Číselné pole (NumericField)

Tabulka 27: Vlastnosti číselného pole

Název	Datový typ	R/W	Vlastnost
ValueMin	double	R+W	Minimální přijatelná vstupní hodnota
ValueMax	double	R+W	Maximální přijatelná vstupní hodnota

Název	Datový typ	R/W	Vlastnost
Hodnota	double	R+(W)	Aktuální hodnota (Nepoužívejte pro hodnoty ze souboru dat).
ValueStep	double	R+W	Interval odpovídající jednomu kliknutí kolečkem myši nebo spinboxem.
DefaultValue	double	R+W	Výchozí hodnota.
DecimalAccuracy	integer	R+W	Počet desetinných míst.
ShowThousandsSeparator	boolean	R+W	Zobrazení oddělovače tisíců podle nastavení národního prostředí.

Pole pro datum (DateField)

Hodnoty data jsou definovány datovým typem **long** a zobrazují se ve formátu ISO: RRRRMMDD, například 20190304 pro 04. března 2019. Chceme-li použít tento typ s funkcí **getDate** a **updateDate**, a s typem **com.sun.star.util.Date**, podíváme se na příklady.

Tabulka 28: Vlastnosti pole Datum

Název	Datový typ	Datový typ od LO 4.1.1	R/W	Vlastnost
DateMin	long	com.sun.star.util.Date	R+W	Minimální akceptované zadané datum.
DateMax	long	com.sun.star.util.Date	R+W	Maximální akceptované zadané datum.
Datum	long	com.sun.star.util.Date	R+(W)	Aktuální hodnota (Nepoužívejte pro hodnoty ze souboru dat).
DateFormat	integer		R+W	Formát data specifický pro operační systém: 0 = krátké datum (jednoduché) 1 = krátké Datum dd.mm.yy (rok zobrazen pomocí dvou číslic) 2 = krátké Datum dd.mm.yyyy (čtyřmístný rok) 3 = dlouhé Datum (obsahuje název dne v týdnu a měsíce) Další možnosti nalezneme v definici formuláře nebo v Popisu API .
DefaultDate	long	com.sun.star.util.Date	R+W	Výchozí hodnota.
DropDown	boolean		R+W	Zobrazení rozevíracího měsíčního kalendáře

Pole pro čas (TimeField)

Hodnoty času jsou také typu **long**.

Tabulka 29: Vlastnosti pole pro čas

Název	Datový typ	Datový typ z LO 4.1.1	R/W	Vlastnost
TimeMin	long	com.sun.star.util.Time	R+W	Minimální přijatelná vstupní hodnota.

Název	Datový typ	Datový typ z LO 4.1.1	R/W	Vlastnost
TimeMax	long	com.sun.star.util.Time	R+W	Maximální přijatelná vstupní hodnota.
Čas	long	com.sun.star.util.Time	R+(W)	Aktuální hodnota (Nepoužívejte pro hodnoty ze souboru dat).
TimeFormat	integer		R+W	Formát času: 0 = krátký, formát hh : mm (hodiny, minuty, 24hodinové hodiny) 1 = dlouhý, formát h : mm : ss (totéž s vteřinami, 24hodinové hodiny) 2 = krátký, formát h : mm (12hodinové hodiny s AM/PM) 3 = dlouhý, formát h : mm : ss (12hodinové hodiny s AM/PM) 4 = krátké zadání na dobu trvání 5 = dlouhý záznam pro časové období
DefaultTime	long	com.sun.star.util.Time	R+W	Výchozí hodnota.

Měnové pole (CurrencyField)

Měnové pole je číselné pole s následujícími dalšími možnostmi.

Tabulka 30: Vlastnosti měnového pole

Název	Datový typ	R/W	Vlastnost
CurrencySymbol	string	R+W	Symbol měny pouze pro zobrazení.
PrependCurrencySymbol	boolean	R+W	Před číslem se zobrazí symbol.

Formátované pole (FormattedControl)

Formátovaný ovládací prvek lze použít podle potřeby pro čísla, měnu nebo datum/čas. Platí zde velmi mnoho již popsaných vlastností, ale s jinými názvy.

Tabulka 31: Vlastnosti formátovaného pole

Název	Datový typ	L/S	Vlastnost
CurrentValue	varianta	R	Aktuální hodnota obsahu. Skutečný typ dat závisí na jejich obsahu a formátu.
EffectiveValue		R+(W)	
EffectiveMin	double	R+W	Minimální přijatelná vstupní hodnota.
EffectiveMax	double	R+W	Maximální přijatelná vstupní hodnota.
EffectiveDefault	varianta	R+W	Výchozí hodnota.
FormatKey	long	R+(W)	Formát pro zobrazení a zadání. Pomocí makra jej nelze jednoduše změnit.
EnforceFormat	boolean	R+W	Formát se testuje při zadávání. Povoleny jsou pouze určité znaky a kombinace.

Pole se seznamem (ListBox)

Přístup ke čtení a zápisu hodnoty ležící za vybraným řádkem je poněkud komplikovaný, ale možný.

Tabulka 32: Vlastnosti seznamu

Název	Datový typ	R/W	Vlastnost
ListSource	pole řetězců	R+W	Zdroj dat: Zdroj obsahu seznamu nebo název datové sady, která poskytuje viditelnou položku.
ListSourceType	integer	R+W	Typ zdroje dat: 0 = Seznam hodnot 1 = Tabulka 2 = Dotaz 3 = Sada výsledků z SQL příkazu 4 = Výsledek databázového příkazu 5 = Názvy polí z tabulky databáze
StringItemList	pole řetězců	R	Seznam položek, které lze vybrat.
ItemCount	integer	R	Počet dostupných položek seznamu
ValueItemList	pole řetězců	R	Seznam hodnot, které mají být předány z formuláře do tabulky.
DropDown	boolean	R+W	Rozevírací seznam.
LineCount	integer	R+W	Celkový počet zobrazených řádků při úplném rozbalení.
MultiSelection	boolean	R+W	Určeno pro vícenásobný výběr.
SelectedItems	pole celých čísel	R+W	Seznam vybraných položek jako seznam pozic v celkovém seznamu položek.

První vybraný prvek z pole seznamu se získá takto:

```
oControl = oForm.GetByName("Name of the Listbox")
sEintrag = oControl.ValueItemList( oControl.SelectedItems(0) )
```

Poznámka

Od verze LibreOffice 4.1 lze hodnotu předanou do databáze určit přímo.

```
oControl = oForm.GetByName("Name of the Listbox")
id = oControl.GetCurrentValue()
```

`GetCurrentValue()` vrací hodnotu, která bude uložena v databázové tabulce. U polí se seznamem to závisí na polí, ke kterému jsou vázány (`BoundField`).

Až do verze LibreOffice 4.0 včetně vracela tato funkce zobrazený obsah, nikoli základní hodnotu v tabulce.

Vezměme prosím na vědomí, že položka je „pole řetězců“, pokud se dotaz na pole seznamu vymění za omezení možnosti výběru:

```
Sub Listenfeldfilter
Dim stSql(0) As String
Dim oDoc As Object
Dim oDrawpage As Object
```

```

Dim oForm As Object
Dim oFeld As Object
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("MainForm")
oFeld = oForm.getByName("Listenfeld")
stSql(0) = "SELECT ""Name"", ""ID"" FROM ""Filter_Name"" ORDER BY
""Name""""
oFeld.ListSource = stSql
oFeld.refresh
End Sub

```

Pole se seznamem (ComboBox)

Přestože mají podobné funkce jako seznamy, vlastnosti polí se seznamem se poněkud liší. Viz příklad „Pole se seznamem jako seznamy s možností zadání“ na straně 449.

Tabulka 33: Vlastnosti pole se seznamem

Název	Datový typ	R/W	Vlastnost
Automatické dokončování	boolean	R+W	Vyplňuje se automaticky.
StringItemList	pole řetězců	R+W	Položky seznamu, které jsou k dispozici k použití.
ItemCount	integer	R	Počet dostupných položek seznamu.
DropDown	boolean	R+W	Rozevírací seznam.
LineCount	integer	R+W	Počet řádků zobrazených při rozbalení.
Text	string	R+W	Aktuálně zobrazený text.
DefaultText	string	R+W	Výchozí položka.
ListSource	string	R+W	Název zdroje dat, který poskytuje položky seznamu.
ListSourceType	integer	R+W	Typ zdroje dat. Stejně možnosti jako u polí se seznamem (pouze výběr seznamu hodnot je ignorován).

Zaškrtačací políčka (CheckBox) a přepínače (RadioButton)

Lze použít také tlačítka volby.

Tabulka 34: Vlastnosti zaškrtačacích políček a přepínačů

Název	Datový typ	R/W	Vlastnost
Popisek	string	R+W	Název (štítek)
Stav	short	R+W	Stav 0 = není vybráno 1 = vybráno 2 = nedefinováno
MultiLine	boolean	R+W	Zalomení řádků pro dlouhý text.

Pole vzoru (PatternField)

Kromě vlastností pro jednoduchý text jsou zajímavé následující:

Tabulka 35: Vlastnosti pole vzorku

Název	Datový typ	R/W	Vlastnost
EditMask	string	R+W	Vstupní maska.
LiteralMask	string	R+W	Maska znaků.
StrictFormat	boolean	R+W	Testování formátu při zadávání.

Ovládání prvek tabulky (GridControl)

Tabulka 36: Vlastnosti prvku tabulky

Název	Datový typ	R/W	Vlastnost
Počet	long	R	Počet sloupců.
ElementNames	pole řetězců	R	Seznam názvů sloupců.
HasNavigationBar	boolean	R+W	K dispozici je navigační panel.
RowHeight	long	R+W	Výška řádku.

FixedText – také se nazývá Label

Tabulka 37: Vlastnosti pevného textu / popisku

Název	Datový typ	R/W	Vlastnost
Popisek	string	R+W	Zobrazený text.
MultiLine	boolean	R+W	Zalomení řádků pro dlouhý text.

Skupinové rámečky (GroupBox)

Pro skupinové rámečky, které se běžně zpracovávají pomocí maker, nejsou k dispozici žádné vlastnosti. Důležitý je stav jednotlivých polí možností.

Tlačítka

CommandButton nebo ImageButton

Tabulka 38: Vlastnosti příkazového a obrazového tlačítka

Název	Datový typ	R/W	Vlastnost
Popisek	string	R+W	Nadpis – Text popisku.
Stav	short	R+W	Výchozí stav vybraný pro přepínání.
MultiLine	boolean	R+W	Zalomení řádků pro dlouhý text.
DefaultButton	boolean	R+W	Zda se jedná o výchozí tlačítko.

Lišta navigace (NavigationBar)

Další vlastnosti a metody spojené s navigací – například filtry a změna ukazatele záznamu – se ovládají pomocí formuláře.

Tabulka 39: Vlastnosti lišty navigace

Název	Datový typ	R/W	Vlastnost
IconSize	short	R+W	Velikost ikon.
ShowPosition	boolean	R+W	Polohu lze zadat a zobrazí se.
ShowNavigation	boolean	R+W	Umožňuje navigaci.
ShowRecordActions	boolean	R+W	Umožňuje provádět záznamové akce.

Název	Datový typ	R/W	Vlastnost
ShowFilterSort	boolean	R+W	Umožňuje třídění podle filtru.

Metody pro formuláře a ovládací prvky

Datový typ parametru je označen zkratkou:

- číslo sloupce pro požadované pole v datovém souboru, počítáno od 1.
- číselná hodnota – může to být celé nebo desetinné číslo.
- s – String; maximální délka závisí na definici tabulky.
- b – Boolean (logický) – true nebo false
- d – Hodnota data

Navigace v datové sadě

Tyto metody fungují jak ve formulářích, tak v sadě výsledků dotazu.

„Kurzor“ v popisu znamená ukazatel záznamu.

Tabulka 40: Navigační metody: Testování polohy kurzoru

Název	Datový typ	Popis
isBeforeFirst	boolean	Kurzor je před prvním záznamem. To platí v případě, že po zadání ještě nebyl vynulován.
isFirst	boolean	Zobrazí, zda je kurzor na prvním záznamu.
isLast	boolean	Zobrazí, zda je kurzor na posledním záznamu.
isAfterLast	boolean	Kurzor je po přesunutí na další řádek za posledním řádkem.
getRow	long	Aktuální číslo řádku.

Tabulka 41: Navigační metody: Nastavení kurzoru

Pro logické datové typy znamená True, že navigace proběhla úspěšně.

Název	Datový typ	Popis
beforeFirst	–	Přesune se před první řádek.
first	boolean	Přesune se na první řádek.
previous	boolean	Vrací se o jeden řádek zpět.
next	boolean	Přesune se o jeden řádek dopředu.
last	boolean	Přejde na poslední záznam.
afterLast	–	Přejde po posledním záznamu.
absolute(n)	boolean	Přejde na řádek se zadaným číslem řádku.
relative(n)	boolean	Přejde zpět nebo dopředu o zadanou hodnotu: dopředu kladná hodnota a dozadu pro záporná hodnota argumentu.

Tabulka 42: Navigační metody: Metody ovlivňující aktuální stav záznamu

Název	Datový typ	Popis
refreshRow	–	Zpětně načte původní hodnoty řádku.
rowInserted	boolean	Označuje, zda se jedná o nový řádek.
rowUpdated	boolean	Ukazuje, zda byl aktuální řádek změněn.
rowDeleted	boolean	Ukazuje, zda byl aktuální řádek smazán.

Úprava řádků dat

Metody používané pro čtení jsou k dispozici pro jakýkoli formulář nebo soubor dat. Metody pro změnu a ukládání lze použít pouze pro upravitelné datové sady (obvykle tabulky, nikoli dotazy).

Tabulka 43: Metody pro úpravu řádků dat: Metody pro celý řádek

Název	Datový typ	Popis
insertRow	–	Uloží nový řádek.
updateRow	–	Potvrdí změnu aktuálního řádku.
deleteRow	–	Odstraní aktuální řádek.
cancelRowUpdates	–	Obrátí změny v aktuálním řádku.
moveToInsertRow	–	Přesune kurzor na řádek odpovídající novému záznamu.
moveToCurrentRow	–	Po zadání nového záznamu vrátí kurzor na předchozí pozici.

Tabulka 44: Metody pro úpravu řádků dat: Čtení hodnot

Název	Datový typ	Popis
getString(c)	string	Předává obsah sloupce jako řetězec znaků.
getBoolean(c)	boolean	Předává obsah sloupce jako logickou hodnotu.
getByte(c)	byte	Udává obsah sloupce jako jeden bajt.
getShort(c)	short	Udává obsah sloupce jako celé číslo.
getInt(c)	integer	Udává obsah sloupce jako celé číslo.
getLong(c)	long	Udává obsah sloupce jako celé číslo.
getFloat(c)	float	Udává obsah sloupce jako desetinné číslo s jednoduchou přesností.
getDouble(c)	double	Udává obsah sloupce jako desetinné číslo s dvojnásobnou přesností. Díky automatickým převodům prováděným Basicem je tento typ vhodný pro desetinná a měnová pole.
getBytes(c)	pole bajtů	Předává obsah sloupce jako pole jednotlivých bajtů.
getDate(c)	Datum	Předává obsah sloupce jako datum.
getTime(c)	Čas	Předává obsah sloupce jako časovou hodnotu.
getTimestamp(c)	DateTime	Předává obsah sloupce jako časové razítko (datum a čas).
wasNull	boolean	Ukazuje, zda hodnota naposledy načteného sloupce byla NULL.



Poznámka

V samotném jazyce Basic mají hodnoty data a času typ DATE. Pro přístup k datům v datových sadách jsou k dispozici různé typy: **com.sun.star.util.Date** pro datum, **com.sun.star.util.Time** pro čas a **com.sun.star.util.DateTime** pro časovou značku.

Tabulka 45: Metody ukládání hodnot

Název	Datový typ	Popis
updateNull(c)	–	Nastaví obsah sloupce na hodnotu NULL.

updateBoolean(c,b)	–	Změní obsah sloupce c na logickou hodnotu b.
updateByte(c,x)	–	Uloží bajt x do sloupce c.
updateShort(c,n)	–	Uloží celé číslo n do sloupce c.
updateInt(c,n)	–	Uloží celé číslo n do sloupce c.
updateLong(c,n)	–	Uloží celé číslo n do sloupce c.
updateFloat(c,n)	–	Uloží desetinné číslo n do sloupce c.
updateDouble(c,n)	–	Uloží desetinné číslo n do sloupce c.
updateString(c,s)	–	Uloží řetězec s do sloupce c.
updateBytes(c,x)	–	Uloží pole bajtů x do sloupce c.
updateDate(c,d)	–	Uloží datum d do sloupce c.
updateTime(c,d)	–	Uloží čas d do sloupce c.
updateTimestamp(c,d)	–	Uloží časové razítko d do sloupce c.

Úprava jednotlivých hodnot

Tato metoda používá vlastnost **BoundField** ovládacího prvku ke čtení nebo úpravě obsahu příslušného sloupce. Odpovídá téměř přesně metodě popsané v předchozí části s tím rozdílem, že se neuvádí číslo sloupce.

Tabulka 46: Metody úprav jednotlivých hodnot: čtení hodnot

Název	Datový typ	Popis
getString	string	Předává obsah pole jako řetězec znaků.
getBoolean	boolean	Předává obsah pole jako logickou hodnotu.
getByte	byte	Předává obsah pole jako jeden bajt.
getShort	short	Udává obsah pole jako celé číslo.
getInt	integer	Udává obsah pole jako celé číslo.
getLong	long	Udává obsah pole jako celé číslo.
getFloat	float	Udává obsah pole jako desetinnou hodnotu s přesností na jedno desetinné místo.
getDouble	double	Udává obsah pole jako desetinné číslo s přesností na dvě desetinná místa. Díky automatickým převodům prováděným Basicem je tento typ vhodný pro desetinná a měnová pole.
getBytes	pole bajtů	Předává obsah pole jako pole bajtů.
getDate	Datum	Udává obsah pole jako datum.
getTime	Čas	Udává obsah pole jako čas.
getTimestamp	DateTime	Udává obsah pole jako časové razítko.
wasNull	boolean	Ukazuje, zda hodnota naposledy načteného sloupce byla NULL.



Poznámka

V samotném jazyce Basic mají hodnoty data a času typ DATE. Pro přístup k datům v datových sadách jsou k dispozici různé typy: **com.sun.star.util.Date** pro datum, **com.sun.star.util.Time** pro čas a **com.sun.star.util.DateTime** pro časovou značku.

Tabulka 47: Metody úprav jednotlivých hodnot: ukládání hodnot

Název	Datový typ	Popis
updateNull	–	Nastaví obsah sloupce na hodnotu NULL.
updateBoolean(b)	–	Nastaví obsah sloupce na logickou hodnotu b.
updateByte(x)	–	Uloží bajt x do sloupce.
updateShort(n)	–	Uloží celé číslo n do sloupce.
updateInt(n)	–	Uloží celé číslo n do sloupce.
updateLong(n)	–	Uloží celé číslo n do sloupce.
updateFloat(n)	–	Uloží do sloupce desetinné číslo n.
updateDouble(n)	–	Uloží do sloupce desetinné číslo n.
updateString(s)	–	Uloží řetězec znaků s do sloupce.
updateBytes(x)	–	Uloží pole bajtů x do sloupce.
updateDate(d)	–	Uloží datum d do sloupce.
updateTime(d)	–	Uloží čas d do sloupce.
updateTimestamp(d)	–	Uloží časové razítko d do sloupce.

Parametry pro připravené příkazy SQL

Metody, které přenášejí hodnotu předpřipraveného příkazu SQL (viz „Předpřipravené příkazy SQL s parametry“ na straně 412), jsou podobné jako v předchozí části. První parametr (označený l) je číslována pozice v příkazu SQL.

Tabulka 48: Metody přenosu hodnoty předpřipraveného příkazu SQL

Název	Datový typ	Popis
setNull(i, n)	–	Nastaví obsah sloupce na hodnotu NULL. N je datový typ SQL, jak je uveden v Popisu API .
setBoolean(i, b)	–	Vloží do příkazu SQL zadanou logickou hodnotu b.
setByte(i, x)	–	Vloží zadaný bajt x do příkazu SQL.
setShort(i, n)	–	Vloží do příkazu SQL zadané celé číslo n.
setInt(i, n)	–	Vloží do příkazu SQL zadané celé číslo n.
setLong(i, n)	–	Vloží do příkazu SQL zadané celé číslo n.
setFloat(i, n)	–	Vloží zadané desetinné číslo do příkazu SQL.
setDouble(i, n)	–	
setString(i, s)	–	Vloží zadaný řetězec znaků do příkazu SQL.
setBytes(i, x)	–	Vloží zadané pole bajtů x do příkazu SQL.
setDate(i, d)	–	Vloží zadané datum d do příkazu SQL.
setTime(i, d)	–	Vloží do příkazu SQL zadaný čas d.
setTimestamp(i, d)	–	Vloží do příkazu SQL zadané časové razítko d.
clearParameters	–	Odstraní předchozí hodnoty všech parametrů příkazu SQL.

Zlepšení použitelnosti

U této první kategorie použití maker si ukážeme různé možnosti, jak zlepšit použitelnost formulářů aplikace Base.

Automatická aktualizace formulářů

Často se ve formuláři něco změní a tato změna se musí objevit v druhém formuláři na stejné stránce. Následující úryvek kódu zavolá metodu Reload druhého formuláře a způsobí jeho obnovení.

Sub Update

Nejprve je makro pojmenováno. Výchozí označení makra je **Sub**. Může být napsáno velkými nebo malými písmeny. **Sub** umožňuje spuštění podprogramu bez vrácení hodnoty. Dále je naopak popsána funkce, která vrací hodnotu.

Makro má název Update. Proměnné nemusíme deklarovat, protože LibreOffice Basic automaticky vytváří proměnné při jejich použití. Pokud proměnnou napíšeme špatně, LibreOffice Basic vytvoří novou proměnnou, aniž by si stěžoval. Použijeme **Option Explicit** chceme-li LibreOffice Basicu zabránit v automatickém vytváření proměnných; většina programátorů tento postup doporučuje.

Proto obvykle začínáme deklarací proměnných. Všechny zde deklarované proměnné jsou objekty (nikoli čísla nebo text), proto na konec deklarace přidáme **As Object**. Abychom si později připomněli typ proměnných, uvádíme jejich názvy s písmenem "o". V zásadě si však můžeme zvolit téměř libovolné názvy proměnných.

```
Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm As Object
```

Formulář leží v aktuálně aktivním dokumentu. Kontejner, ve kterém jsou uloženy všechny formuláře, se jmenuje **drawpage**. V navigátoru formulářem se jedná o koncept nejvyšší úrovně, kterému jsou všechny formuláře podřízeny.

V tomto příkladu je formulář, ke kterému se má přistupovat, pojmenován Display. Display je název viditelný v navigátoru formulářem. Tak například první formulář se ve výchozím nastavení jmenuje Form1.

```
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("Display")
```

Protože je nyní formulář zpřístupněn a bod, ve kterém je přístupný, je uložen v proměnné **oForm**, je nyní znovu načten (obnoven) příkazem **reload()**.

```
oForm.reload()
```

End Sub

Podprogram začíná **SUB**, takže musí končit **End Sub**.

Toto makro lze nyní vybrat ke spuštění při uložení jiného formuláře. Například v pokladně, pokud se do jednoho formuláře zadá celkový počet prodaných položek a jejich skladová čísla (načtená snímačem čárových kódů), může se v jiném formuláři ve stejném otevřeném okně zobrazit název všech položek a celková cena, jakmile se formulář uloží.

Filtrování záznamů

Samotný filtr může bez problémů fungovat v podobě popsané v kapitole 8, Úlohy databáze. Níže uvedená varianta nahrazuje tlačítko Uložit a znovu načítá seznamy, takže zvolený filtr z jednoho seznamu může omezit možnosti dostupné v druhém seznamu.

✓ Poznámka

Viz také databáze `Example_Search_and_Filter.odt` související s touto knihou.

Sub `Filter`

```
Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm1 As Object
Dim oForm2 As Object
Dim oFieldList1 As Object
Dim oFieldList2 As Object
oDoc = thisComponent
oDrawpage = oDoc.drawpage
```

Nejprve jsou definovány a nastaveny proměnné pro přístup k sadě formulářů. Tato sada obsahuje dva formuláře "Filter" a "Display". Pole seznamu jsou ve formuláři "Filter" a mají názvy "List_1" a "List_2".

```
oForm1 = oDrawpage.forms.getByName("filter")
oForm2 = oDrawpage.forms.getByName("display")
oFieldList1 = oForm1.getByName("listbox1")
oFieldList2 = oForm1.getByName("listbox2")
```

Nejprve se obsah polí se seznamem přeneso do základního formuláře pomocí `commit()`. Přenos je nutný, protože jinak by se změna v poli se seznamem při ukládání neprojevila. Instrukci `commit()` je třeba použít pouze na poli se seznamem, ke kterému byl právě uskutečněn přístup. Poté se záznam uloží pomocí `updateRow()`. Naše filtrační tabulka obsahuje v zásadě pouze jeden záznam, který je zapsán jednou na začátku. Tento záznam je proto průběžně přepisován pomocí příkazu `update`.

```
oFieldList1.commit()
oFieldList2.commit()
oForm1.updateRow()
```

Pole se seznamem se mají navzájem ovlivňovat. Pokud například jedno pole se seznamem slouží k omezení zobrazených médií na CD, druhé pole se seznamem by nemělo zahrnovat všechny autory knih ve svém seznamu autorů. Výběr v druhém seznamu by pak příliš často vedl k prázdnému filtru. To je důvod, proč je třeba seznamy znovu načíst. Přesně řečeno, příkaz `refresh()` je třeba provést pouze na seznamu, který nebyl otevřen.

Poté se znovu načte `form2`, který má zobrazit filtrovaný obsah.

```
oFieldList1.refresh()
oFieldList2.refresh()
oForm2.reload()
```

End Sub

Pole se seznamem, která mají být touto metodou ovlivněna, mohou být doplněna pomocí různých dotazů.

Nejjednodušší variantou je, že pole seznamu přebírá svůj obsah z výsledků filtru. Jediný filtr pak určuje, který obsah dat bude dále filtrován.

```
SELECT "Field_1" || ' - ' || "Count" AS "Display", "Field_1"
FROM ( SELECT COUNT( "ID" ) AS "Count", "Field_1" FROM "searchtable" GROUP
BY "Field_1" )
ORDER BY "Field_1"
```

Zobrazí se obsah pole a počet nálezů. K získání počtu shod se používá dílčí dotaz. To je nezbytné, protože jinak se v seznamu zobrazí pouze počet shod bez dalších informací z pole.

Makro touto akcí poměrně rychle vytvoří seznamy, které jsou vyplněny pouze jednou hodnotou. Pokud pole se seznamem není NULL, bere se při filtrování v úvahu. Po aktivaci druhého pole se seznamem jsou v obou polích se seznamem k dispozici pouze prázdná pole a jedna zobrazená hodnota. To se může zdát praktické pro omezené vyhledávání. Ale co když knihovní katalog jasně ukazuje zařazení položky, ale neukazuje jednoznačně, zda se jedná o knihu, CD nebo DVD? Pokud je nejprve vybrána klasifikace a poté je druhé pole seznamu nastaveno na "CD", musí být pro následné vyhledávání zahrnující knihy nastaveno na NULL. Bylo by praktičtější, kdyby se v druhém seznamu přímo zobrazovaly různé dostupné typy médií s odpovídajícím počtem položek.

K dosažení tohoto cíle je sestaven následující dotaz, který již nevychází přímo z výsledků filtru. Počet odpovídajících položek je třeba získat jiným způsobem.

```
SELECT
IFNULL( "Field_1" || ' - ' || "Count", 'empty - ' || "Count" ) AS "Display",
"Field_1"
FROM
( SELECT COUNT( "ID" ) AS "Count", "Field_1" FROM "Table"
WHERE "ID" IN
( SELECT "Table"."ID" FROM "Filter", "Table"
WHERE "Table"."Field_2" = IFNULL( "Filter"."Filter_2",
"Table"."Field_2" ) )
GROUP BY "Field_1" )
ORDER BY "Field_1"
```

Tento velmi složitý dotaz lze rozdělit. V praxi se běžně používá **VIEW** pro poddotaz. Pole se seznamem získává svůj obsah z dotazu vztahujícího se k tomuto **VIEW**.

Podrobný dotaz: Dotaz obsahuje dva sloupce. První sloupec obsahuje zobrazení, které vidí osoba s otevřeným formulářem. Tento pohled ukazuje obsah pole a shody pro obsah tohoto pole oddělené pomlčkou. Druhý sloupec přenáší svůj obsah do základní tabulky formuláře. Zde máme k dispozici pouze obsah pole. Pole se seznamem tak čerpají svůj obsah z dotazu, který je ve formuláři prezentován jako výsledek filtru. Pouze tato pole jsou k dispozici pro další filtrování.

Tabulka, ze které se tyto informace čerpají, je vlastně dotaz. V tomto dotazu se počítají pole primárního klíče (**SELECT COUNT("ID") AS "Count"**). Ta se pak seskupí podle hledaného výrazu v poli (**GROUP BY "Field_1"**). Tento dotaz zobrazuje termín v samotném poli jako druhý sloupec. Tento dotaz je zase založen na dalším poddotazu:

```
SELECT "Table"."ID" FROM "Filter", "Table"
```



```
WHERE "Table"."Field_2" =
  IFNULL( "Filter"."Filter_2", "Table"."Field_2" )
```

Tento poddotaz se zabývá dalším filtrovaným polem. Toto další pole musí v zásadě také odpovídat primárnímu klíči. Pokud existují další filtry, lze tento dotaz rozšířit:

```
SELECT "Table"."ID" FROM "Filter", "Table" WHERE
"Table"."Field_2" = IFNULL( "Filter"."Filter_2", "Table"."Field_2" )
AND
"Table"."Field_3" = IFNULL( "Filter"."Filter_3", "Table"."Field_3" )
```

To umožňuje, aby všechna další pole, která mají být filtrována, kontrolovala, co se nakonec zobrazí v seznamu prvního pole "Field_1".

Nakonec je celý dotaz seřazen podle základního pole.

Jak vlastně vypadá výsledný dotaz, který je základem zobrazeného formuláře, se dozvíme v kapitole 8, Úlohy databáze.

Následující makro může prostřednictvím pole se seznamem řídit, která pole se seznamem se musí uložit a která se musí znovu načíst.

Následující podprogram předpokládá, že vlastnost Další informace pro každé pole se seznamem obsahuje čárkou oddělený seznam všech názvů polí se seznamem bez mezer. První název v seznamu musí být název tohoto pole se seznamem.

```
Sub Filter_more_info(oEvent As Object)
  Dim oDoc As Object
  Dim oDrawpage As Object
  Dim oForm1 As Object
  Dim oForm2 As Object
  Dim sTag As String
  sTag = oEvent.Source.Model.Tag
```

Vytvoří se pole (kolekce dat přístupná pomocí indexového čísla) a naplní se názvy polí se seznamem. První název v seznamu je název pole seznamu spojeného s událostí.

```
aList() = Split(sTag, ",")
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm1 = oDrawpage.forms.getByName("filter")
oForm2 = oDrawpage.forms.getByName("display")
```

Pole je procházeno od své dolní hranice ('**Lbound()**') k horní hranici ('**Ubound()**') v jedné smyčce. Všechny hodnoty, které byly v doplňkových informacích odděleny čárkami, se nyní přenášejí postupně.

```
For i = LBound(aList()) To UBound(aList())
  If i = 0 Then
```

Pole se seznamem, které makro spustilo, musí být uloženo. Nachází se v proměnné **aList(0)**. Nejprve se informace pro pole seznamu přenesou do základní tabulky a poté se záznam uloží.

```
oForm1.getByName(aList(i)).commit()
oForm1.updateRow()
```

```
Else
```

Ostatní seznamy je třeba obnovit, protože nyní obsahují různé hodnoty v závislosti na prvním seznamu.

```
oForm1.getByname(aList(i)).refresh()  
End If  
Next  
oForm2.reload()
```

```
End Sub
```

Dotazy pro toto použitelnější makro jsou přirozeně stejné jako ty, které již byly uvedeny v předchozí části.

Příprava dat z textových polí tak, aby odpovídala konvencím SQL

Při ukládání dat do příkazu SQL mohou apostrofy v názvech, například „O'Connor“, způsobit problémy. Je to proto, že jednoduché uvozovky (' ') se používají k uzavření textu, který má být vložen do záznamů. V takových případech potřebujeme zprostředkující funkci, která data vhodně připraví.

```
Function String_to_SQL(st As StringString)  
If Instr(st, "'") Then  
st = Join(Split(st, "'"), "''")  
End If  
String_to_SQL = st  
End Function
```

Všimneme si, že se jedná o funkci, nikoli o podřízený prvek. Funkce přijímá jako argument hodnotu a poté vrací hodnotu.

V textu, který se má přenést, se nejprve vyhledá, zda neobsahuje apostrof. V takovém případě se text v tomto místě rozdělí – apostrof je sám o sobě oddělovačem – a opět se spojí dvěma apostrofy. Tím se zakryje kód SQL. Výsledek funkce se získá následujícím voláním:

```
stTextnew = String_to_SQL(stTextold)
```

To jednoduše znamená, že proměnná stTextold se přepracuje a výsledek se uloží do proměnné stTextnew. Tyto dvě proměnné nemusí mít ve skutečnosti odlišné názvy. Volání lze provést pomocí:

```
stText = String_to_SQL(stText)
```

Tato funkce se opakovaně používá v následujících makrech, aby bylo možné apostrofy ukládat také pomocí příkazů SQL.

Výpočet hodnot ve formuláři předem

Hodnoty, které lze vypočítat pomocí databázových funkcí, nejsou v databázi uloženy samostatně. Výpočet neprobíhá během zadávání do formuláře, ale až po uložení záznamu. Pokud se formulář skládá pouze z jednoho ovládacího prvku tabulky, je to jen malý rozdíl. Vypočtenou hodnotu lze odečíst ihned po zadání údajů. Pokud však formuláře obsahují sadu různých jednotlivých polí, nemusí být předchozí záznam viditelný. V takových případech má smysl, aby se hodnoty, které se jinak počítají uvnitř databáze, zobrazovaly v příslušných polích.

✓ Poznámka

Viz databáze Example_direct_Calculation_Form.odt spojená s touto knihou.

Následující tři makra ukazují, jak lze takovou věc v zásadě provést. Obě makra jsou spojena s výstupem z daného pole. To také umožňuje zohlednit skutečnost, že hodnota v existujícím poli může být následně změněna.

```
Sub Calculation_without_Tax(oEvent As Object)
    Dim oForm As Object
    Dim oField As Object
    Dim oField2 As Object
    oField = oEvent.Source.Model
    oForm = oField.Parent
    oField2 = oForm.GetByName("price_without_tax")
    oField2.BoundField.UpdateDouble(oField.GetCurrentValue / 1.19)
    If Not IsEmpty(oForm.GetByName("quantity").GetCurrentValue()) Then
        total_calc2(oForm.GetByName("quantity"))
    End If
End Sub
```

Pokud je do pole Price zadána hodnota, makro se spustí při opuštění tohoto pole. Ve stejném formuláři jako pole Price je pole nazvané price_without_tax. Pro toto pole se použije **BoundField.UpdateDouble** pro výpočet ceny bez DPH. Datové pole je odvozeno z dotazu, který v zásadě provádí stejný výpočet, ale s použitím uložených dat. Tímto způsobem je vypočtená hodnota viditelná při zadávání dat a také později při procházení záznamu, aniž by byla uložena.

Pokud pole quantity obsahuje hodnotu, provede se další výpočet pro pole, která jsou s ním svázána.

```
Sub Calculation_Total(oEvent As Object)
    oField = oEvent.Source.Model
    Calculation_Total2(oField)
End Sub
```

Tento krátký postup slouží pouze k přenosu řešení následujícího postupu při opuštění pole quantity na formuláři.

```
Sub Calculation_Total2(oField As Object)
    Dim oForm As Object
    Dim oField2 As Object
    Dim oField3 As Object
    Dim oField4 As Object
    oForm = oField.Parent
    oField2 = oForm.GetByName("price")
    oField3 = oForm.GetByName("total")
    oField4 = oForm.GetByName("tax_total")
End Sub
```

```

    oField3.BoundField.UpdateDouble(oField.getCurrentValue *
oField2.getCurrentValue)
    oField4.BoundField.UpdateDouble(oField.getCurrentValue *
oField2.getCurrentValue -
    oField.getCurrentValue * oField2.getCurrentValue / 1.19)
End Sub

```

Tento postup je pouze způsob, jak ovlivnit několik polí najednou. Postup se spouští z jednoho pole quantity, které obsahuje počet nakoupených položek. Pomocí tohoto pole a pole price se vypočítá celková částka a tax_total (daň celkem, pozn. překl.) a přenesou se do příslušných polí.

Tyto postupy a dotazy mají jeden nedostatek: sazba DPH je v programu zakódována napevno. Bylo by lepší použít argument související s cenou, protože DPH se může lišit a nemusí být u všech výrobků stejná. V takových případech by bylo třeba příslušnou hodnotu DPH vyčíst z pole formuláře.

Poskytnutí aktuální verze LibreOffice

LibreOffice verze 4.1 přinesl některé změny v polích se seznamem a hodnotách data, kvůli kterým je nutné při spouštění maker v těchto oblastech určit aktuální verzi. K tomuto účelu slouží následující kód:

```

Function OfficeVersion()
    Dim aSettings, aConfigProvider
    Dim aParams2(0) As New com.sun.star.beans.PropertyValue
    Dim sProvider$, sAccess$
    sProvider = "com.sun.star.configuration.ConfigurationProvider"
    sAccess = "com.sun.star.configuration.ConfigurationAccess"
    aConfigProvider = createUnoService(sProvider)
    aParams2(0).Name = "nodepath"
    aParams2(0).Value = "/org.openoffice.Setup/Product"
    aSettings = aConfigProvider.CreateInstanceWithArguments(sAccess,
aParams2())
    OfficeVersion() =
Array(aSettings.ooName, aSettings.ooSetupVersionAboutBox)
End Function

```

Tato funkce vrací pole, jehož prvním prvkem je LibreOffice a druhým úplné číslo verze, například 4.1.5.2.

Vrácení hodnoty polí seznamu

Od verze LibreOffice 4.1 se do databáze ukládá hodnota vrácená polem se seznamem do pole CurrentValue. V předchozích verzích, ani v OpenOffice nebo Apache OpenOffice tomu tak nebylo. Výpočet provede následující funkce. Je třeba zkontrolovat, zda verze LibreOffice není novější než LibreOffice 4.0.

```

Function ID_Determination(oField As Object) As Integer
    a() = OfficeVersion()
    If a(0) = "LibreOffice" And (LEFT(a(1),1) = 4 And
RIGHT(LEFT(a(1),3),1) > 0) Or LEFT(a(1),1) > 4 Then

```

```
stContent = oField.currentValue  
Else
```

Před verzí LibreOffice 4.1 se předávaná hodnota načítala ze seznamu hodnot pole se seznamem. Viditelně vybraný záznam je SelectedItems(0). '0', protože v seznamu lze vybrat několik dalších hodnot.

```
stContent = oField.ValueItemList(oField.SelectedItems(0))  
End If  
If IsEmpty(stContent) Then
```

-1 je hodnota, která se nepoužívá jako automatická hodnota, a proto nebude existovat ve většině tabulek jako cizí klíč.

```
ID_Determination = -1  
Else  
ID_Determination = Cint(stContent)
```

Převod na celé číslo

```
End If  
End Function
```

Funkce přenáší hodnotu jako celé číslo. Většina primárních klíčů jsou automaticky se zvětšující celá čísla. Pokud cizí klíč toto kritérium nespĺňuje, musí být návratová hodnota upravena na příslušný typ.

Zobrazenou hodnotu pole se seznamem lze dále určit pomocí vlastnosti zobrazení pole.

```
Sub Listfielddisplay  
Dim oDoc As Object  
Dim oForm As Object  
Dim oListbox As Object  
Dim oController As Object  
Dim oView As Object  
oDoc = thisComponent  
oForm = oDoc.Drawpage.Forms(0)  
oListbox = oForm.getByName("Listbox")  
oController = oDoc.getCurrentController()  
oView = oController.getControl(oListbox)  
print "Displayed content: " & oView.SelectedItem  
End Sub
```

Kontrolér slouží k přístupu k zobrazení formuláře. To určuje, co se zobrazí ve vizuálním rozhraní. Vybraná hodnota je **SelectedItem**.

Omezení seznamů zadáním počátečních písmen

Někdy se může stát, že obsah polí se seznamem naroste do příliš velkých rozměrů. V takových případech je pro urychlení vyhledávání vhodné omezit obsah pole seznamu na hodnoty uvedené zadáním jednoho nebo více počátečních znaků. Samotné pole se seznamem je opatřeno příkazem SQL, který slouží jako zástupný příkaz. Může to být:

```
SELECT "Name", "ID" FROM "Table" ORDER BY "Name" LIMIT 5
```

Tím se zabrání tomu, aby aplikace Base musela při otevření formuláře načítat obrovský seznam hodnot.

Následující makro je propojeno s **Vlastnosti: Vlastnosti: Pole se seznamem > Události > Klávesa uvolněna**.

```
Global stListStart As String
```

```
Global lTime As Long
```

Nejprve se vytvoří globální proměnné. Tyto proměnné jsou nezbytné k tomu, aby bylo možné vyhledávat nejen jednotlivá písmena, ale po stisknutí dalších kláves také kombinace písmen.

Zadaná písmena se postupně ukládají do globální proměnné **stListStart**.

Globální proměnná **lTime** slouží k uložení aktuálního času v sekundách. Pokud je mezi jednotlivými stisky kláves dlouhá pauza, měla by se proměnná **stListStart** vynulovat. Z tohoto důvodu je dotazován časový rozdíl mezi po sobě následujícími záznamy.

```
Sub ListFilter(oEvent As Object)
```

```
oField = oEvent.Source.Model
```

```
If oEvent.KeyCode < 538 Then
```

Makro se spouští stiskem klávesy. V rámci rozhraní API má každý klíč číselný kód, který lze vyhledat v `com::sun::star::awt::Key` (skupiny konstant části rozhraní API). Speciální znaky jako ä, ö, a ü má kód 0. Všechna ostatní písmena a číslice mají kód `KeyCode` menší než 538.

Je důležité zkontrolovat **KeyCode**, protože stisknutí klávesy Tab pro přechod do jiného pole spustí také makro. **KeyCode** pro klávesu Tab je 1282, takže další kód v makru nebude proveden.

```
Dim stSql(0) As String
```

Kód SQL pro pole se seznamem je uložen v poli. Příkazy SQL se však počítají jako jednotlivé datové prvky, takže pole je dimenzováno jako **stSql(0)**.

Při čtení kódu SQL z pole se seznamem si uvědomíme, že kód SQL není přímo přístupný jako text. Místo toho je kód k dispozici jako jeden prvek pole: **oField.ListSource(0)**.

Po deklaraci proměnných pro budoucí použití se příkaz SQL rozdělí. Abychom získali pole, které má být filtrováno, rozdělíme kód na první čárku. Pole proto musí být v příkazu uvedeno jako první. Poté je tento kód opět rozdělen na první znak uvozovky, který zavádí název pole. Zde je to provedeno pomocí jednoduchých polí. Proměnná **stField** musí mít na začátku uvozovky. Kromě toho se používá **Rtrim**, aby se na konci výrazu nevyskytovala mezera.

```
Dim stText As String
```

```
Dim stField As String
```

```
Dim stQuery As String
```

```
Dim ar0()
```

```
Dim ar1()
```

```
ar0() = Split(oField.ListSource(0), ",", 2)
```

```
ar1() = Split(ar0(0), "''", 2)
```

```
stField = "''" & Rtrim(ar1(1))
```

Dále se v kódu SQL očekává instrukce třídění. Příkazy v jazyce SQL však mohou být ve velkých, malých nebo smíšených písmenech, proto se k nalezení řetězce znaků **ORDER** používá funkce **InStr** místo funkce **Split**. Poslední parametr této funkce je 1, což znamená, že vyhledávání by nemělo rozlišovat velká a malá písmena. Vše nalevo od řetězce **ORDER** se použije pro

konstrukci nového kódu SQL. Tím je zajištěno, že kód může obsluhovat i pole se seznamem, která pocházejí z různých tabulek nebo byla definována v kódu SQL pomocí dalších podmínek.

```
stQuery = Left(oField.ListSource(0), InStr(1,oField.ListSource(0),
"ORDER",1)-1)
If InStr(stQuery, "LOWER") > 0 Then
    stQuery = Left(stQuery, InStr(stQuery, "LOWER")-1)
ElseIf InStr(1,stQuery, "WHERE",1) > 0 Then
    stQuery = stQuery & " AND "
Else
    stQuery = stQuery & " WHERE "
End If
```

Pokud dotaz obsahuje výraz LOWER, znamená to, že byl vytvořen pomocí této procedury **ListFilter**. Proto při konstrukci nového dotazu musíme jít pouze do této pozice.

Pokud tomu tak není a dotaz již obsahuje výraz WHERE (velkými nebo malými písmeny), je třeba k dalším podmínkám dotazu připojit **AND**.

Pokud není splněna ani jedna z těchto podmínek, připojí se ke stávajícímu kódu **WHERE**.

```
If lTime > 0 And Timer() - lTime < 5 Then
    stListStart = stListStart & oEvent.KeyChar
Else
    stListStart = oEvent.KeyChar
End If
lTime = Timer()
```

Pokud je v globální proměnné uložena hodnota času a rozdíl mezi ní a aktuálním časem je menší než 5 sekund, připojí se zadané písmeno k předchozímu. V opačném případě je písmeno považováno za nový jednopísmenný záznam. Pole se seznamem bude poté znovu filtrováno podle této položky. Poté se aktuální čas uloží do **lTime**.

```
stText = LCase( stListStart & "%")
stSql(0) = stQuery + "LOWER("+stField+") LIKE '"+stText+"' ORDER BY
"+stField+"
oField.ListSource = stSql
oField.refresh
End If
End Sub
```

Kód SQL je konečně sestaven. Verze obsahu pole s malými písmeny se porovná s verzí zadaného písmene (písmen) s malými písmeny. Kód se vloží do pole se seznamem a pole se aktualizuje tak, aby bylo možné vyhledat pouze filtrovaný obsah.

Převod dat z formuláře do proměnné data

```
Function DateValue(oField As Object) As Date
    a() = OfficeVersion()
    If a(0) = "LibreOffice" And (LEFT(a(1),1) = 4 And
RIGHT(LEFT(a(1),3),1) > 0)
```

```
Or LEFT(a(1),1) > 4 Then
```

Zde jsou zachyceny všechny verze LibreOffice od verze 4.1. Za tímto účelem se číslo verze rozdělí na jednotlivé prvky a zkontroluje se hlavní a vedlejší číslo verze. Tato funkce bude fungovat až do verze LibreOffice 9.

```
Dim stMonth As String
Dim stDay As String
stMonth = Right(Str(0) & Str(oField.CurrentValue.Month),2)
stDay = Right(Str(0) & Str(oField.CurrentValue.Day),2)
Datumswert = CDateFromIso(oField.CurrentValue.Year & stMonth &
stDay)
Else
DateValue = CDateFromIso(oField.CurrentValue)
End If
End Function
```

Od verze LibreOffice 4.1.2 jsou data v ovládacích prvcích formuláře ukládána jako pole. To znamená, že aktuální hodnotu ovládacího prvku nelze použít k přístupu k samotnému datu. Má-li být datum dále použito v makrech, je třeba jej znovu vytvořit ze dne, měsíce a roku.

Vyhledávání datových záznamů

Záznamy v databázi můžeme prohledávat bez použití makra. Odpovídající dotaz, který je třeba nastavit, však může být velmi složitý. Makro může tento problém vyřešit pomocí smyčky.

Následující podprogram načte pole v tabulce, interně vytvoří dotaz, a nakonec vypíše seznam čísel primárních klíčů záznamů v tabulce, které jsou vyhledány pomocí tohoto vyhledávacího výrazu. V následujícím popisu je tabulka s názvem Searchtmp, která se skládá z pole primárního klíče s automatickým nárůstem (ID) a pole s názvem Nr., které obsahuje všechny primární klíče získané z prohledávané tabulky. Název tabulky je podprogramu zpočátku předáván jako proměnná.

Chceme-li získat správný výsledek, musí tabulka obsahovat hledaný obsah jako text, nikoli jako cizí klíče. V případě potřeby můžeme vytvořit VIEW, které makro použije.



Poznámka

Viz databáze Example_Search_and_Filter.odt související s touto knihou.

```
Sub Searching(stTable As String)
Dim oDataSource As Object
Dim oConnection As Object
Dim oSQL_Command As Object
Dim stSql As String
Dim oResult As Object
Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm As Object
Dim oForm2 As Object
Dim oField As Object
```



```

Dim stContent As String
Dim arContent() As String
Dim inI As Integer
Dim inK As Integer
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("searchform")
oField = oForm.getByName("searchtext")
stContent = oField.getCurrentValue()
stContent = LCase(stContent)

```

Obsah vyhledávacího textového pole je zpočátku převeden na malá písmena, takže následná vyhledávací funkce musí porovnávat pouze malá písmena.

```

oDataSource = ThisComponent.Parent.DataSource
oConnection = oDataSource.GetConnection("", "")
oSQL_Command = oConnection.createStatement()

```

Nejprve je třeba zjistit, zda byl vyhledávací výraz skutečně zadán. Pokud je pole prázdné, předpokládá se, že vyhledávání není vyžadováno. Všechny záznamy se zobrazí bez dalšího vyhledávání.

Pokud byl zadán hledaný výraz, načtou se názvy sloupců z prohledávané tabulky, aby mohl dotaz přistupovat k polím.

```

If stContent <> "" Then
    stSql = "SELECT ""COLUMN_NAME"" FROM
""INFORMATION_SCHEMA"". ""SYSTEM_COLUMNS"" WHERE ""TABLE_NAME"" = '" + stTable
+ "' ORDER BY ""ORDINAL_POSITION"""
    oResult = oSQL_Statement.executeQuery(stSql)

```

Poznámka

Vzorce SQL v makrech musí být nejprve umístěny do dvojitého uvozovky jako běžné znakové řetězce. Názvy polí a tabulek jsou uvnitř vzorce SQL již ve dvojitém uvozovkách. Pro vytvoření konečného kódu, který správně přenáší dvojité uvozovky, musí být názvy polí a tabulek opatřeny dvěma sadami těchto uvozovek.

```
stSql = "SELECT ""Name"" FROM ""Table"";"
```

se při zobrazení příkazem `MsgBox stSql`,
`.SELECT "Name" FROM "Table";`

Index pole, do kterého se zapisují názvy polí, je zpočátku nastaven na 0. Poté se začne číst dotaz. Protože velikost pole není známa, je třeba ji průběžně upravovat. Proto smyčka začíná '**ReDim Preserve arContent(inI)**', aby se nastavila velikost pole a zároveň se zachoval jeho stávající obsah. Poté se načtou pole a index pole se zvýší o 1. Poté se pole znovu dimenzuje a lze do něj uložit další hodnotu.

```

inI = 0
While oResult.next
    ReDim Preserve arContent(inI)

```

```

        arContent(inI) = oResult.getString(1)
        inI = inI + 1
    Wend
    stSql = "DROP TABLE ""searchtmp"" IF EXISTS"
    oSQL_Command.executeUpdate (stSql)

```

Nyní je dotaz sestaven v rámci smyčky a následně aplikován na tabulku definovanou na začátku. Jsou povoleny všechny kombinace malých a velkých písmen, protože obsah pole v dotazu je převeden na malá písmena.

Dotaz je sestaven tak, aby výsledky skončily v tabulce "searchtmp". Předpokládá se, že primárním klíčem je první pole tabulky (**arContent(0)**).

```

    stSql = "SELECT """+arContent(0)+""" INTO ""searchtmp"" FROM "" +
stTable
    + "" WHERE "
    For inK = 0 To (inI - 1)
        stSql = stSql+"LCase("""+arContent(inK)+"") LIKE '%" + stContent + "%'"
        If inK < (inI - 1) Then
            stSql = stSql + " OR "
        End If
    Next
    oSQL_Command.executeQuery(stSql)
Else
    stSql = "DELETE FROM ""searchtmp""
    oSQL_Command.executeUpdate (stSql)
End If

```

Zobrazovací formulář je třeba znovu načíst. Jeho zdrojem dat je dotaz, v tomto případě Searchquery.

```

oForm2 = oDrawpage.forms.getByName("display")
oForm2.reload()

```

End Sub

Tím se vytvoří tabulka, která bude vyhodnocena dotazem. Pokud je to možné, dotaz by měl být sestaven tak, aby jej bylo možné následně upravovat. Je zobrazen vzorový dotaz:

```

SELECT * FROM "searchtable" WHERE "Nr." IN ( SELECT "Nr." FROM "searchtmp"
) OR "Nr." = CASE WHEN ( SELECT COUNT( "Nr." ) FROM "searchtmp" ) > 0
THEN '0' ELSE "Nr." END

```

Jsou zahrnuty všechny prvky **searchtable** včetně primárního klíče. V přímém dotazu se neobjevuje žádná jiná tabulka, proto není potřeba žádný primární klíč z jiné tabulky a výsledek dotazu zůstává editovatelný.

Primární klíč je v tomto příkladu uložen pod názvem **Nr.** Makro načte právě toto pole. Provede se počáteční kontrola, zda se obsah pole **Nr.** objeví v tabulce **searchtmp**. Operátor **IN** je kompatibilní s více hodnotami. Dílčí dotaz může poskytnout i několik záznamů.

Při větším množství dat se přiřazování hodnot pomocí operátoru **IN** rychle zpomaluje. Proto není dobré použít prázdné vyhledávací pole, abychom jednoduše přenesli všechna pole primárního klíče z tabulky **searchtable** do tabulky **searchtmp** a poté data zobrazovali stejným

způsobem. Místo toho prázdné vyhledávací pole vytvoří prázdnou tabulku **searchtmp**, takže nejsou k dispozici žádné záznamy. To je účelem druhé poloviny podmínky:

```
OR "Nr." = CASE WHEN ( SELECT COUNT( "Nr." ) FROM "searchtmp" ) > 0 THEN '-1' ELSE "Nr." END
```

Pokud je v tabulce Searchtmp nalezen záznam, znamená to, že výsledek prvního dotazu je větší než 0. V tomto případě: "Nr." = '-1' (zde potřebujeme číslo, které se nemůže vyskytovat jako primární klíč, takže "-1" je dobrá hodnota). Pokud je výsledkem dotazu přesně 0 (což je případ, kdy nejsou přítomny žádné záznamy), pak "Nr." = "Nr.". Tímto se vypíše každý záznam, který má Nr. a Nr. je primární klíč, což znamená všechny záznamy.

Zvýraznění vyhledávacích výrazů ve formulářích a výsledcích

U rozsáhlého textového pole často není jasné, kde se vyskytují shody s hledaným výrazem. Bylo by hezké, kdyby formulář uměl zvýraznit zápasy. Mělo by to vypadat jako na obrázku 428.

Searchitem:

General notes on the creation of a database
The basics of creating a database in LibreOffice are described in Chapter 8 of the Getting Started guide, Getting Started with Base.
The database component of LibreOffice, called Base, provides a graphical interface for working with databases. In addition, LibreOffice contains a version of the HSQL database engine. This HSQLDB database can only be used by a single user. The entire data set is stored in an ODB file which has no file locking mechanism when opened by a user.

Obrázek 428: Formulář pro zvýraznění shody při vyhledávání

K tomu, aby formulář takto fungoval, potřebujeme několik dalších položek v naší sadě triků.

Poznámka

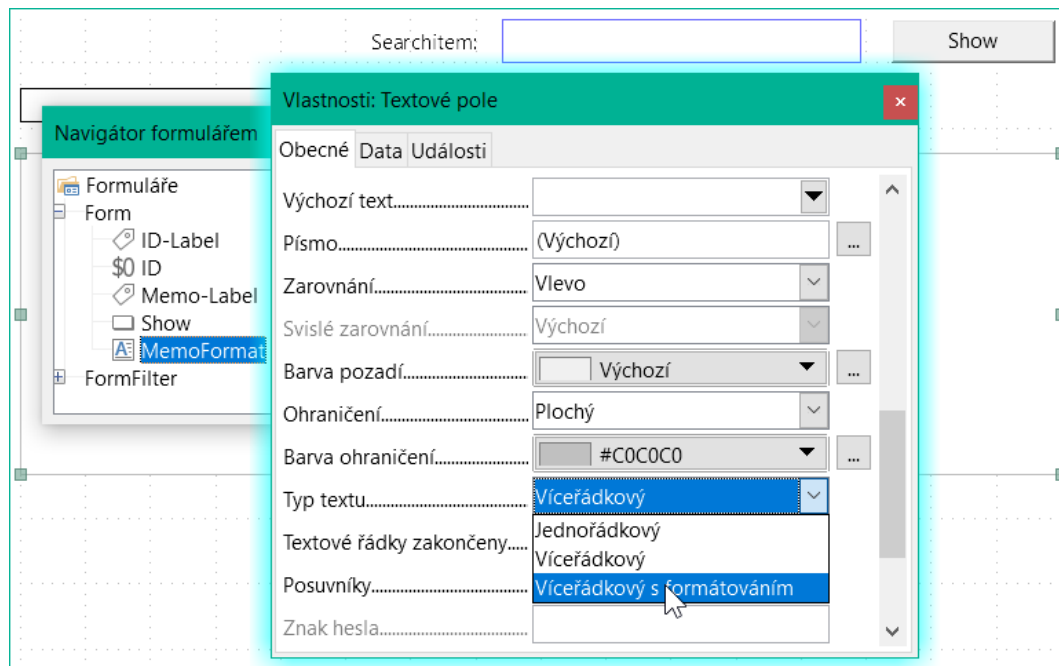
Viz databáze Example_Autotext_Searchmarking_Spelling.odt související s touto knihou.

Fungování takového vyhledávacího pole již bylo vysvětleno. Je vytvořena tabulka filtrů a formulář slouží k zápisu aktuálních hodnot jednoho záznamu do této tabulky. Hlavní formulář je opatřen obsahem pomocí dotazu, který vypadá takto:

```
SELECT "ID", "memo"  
FROM "table"  
WHERE LOWER ( "memo" ) LIKE '%' || LOWER (  
  ( SELECT "searchtext" FROM "filter" WHERE "ID" = TRUE ) ) || '%'
```

Po zadání hledaného textu se zobrazí všechny záznamy v tabulce „Table“, které mají hledaný text v poli „memo“. Při vyhledávání se nerozlišují velká a malá písmena.

Pokud není zadán žádný vyhledávací text, zobrazí se všechny záznamy v tabulce. Jelikož je primární klíč této tabulky obsažen v dotazu, lze jej upravovat.



Obrázek 429: Nastavení typu textu na víceřádkový s formátováním

Ve formuláři je kromě pole ID pro primární klíč také pole s názvem MemoFormat, které bylo nakonfigurováno (pomocí **Vlastnosti > Obecné > Typ textu > Víceřádkový s formátováním**) tak, aby zobrazovalo barevný i černý text. Při pozorném zkoumání vlastností textového pole zjistíme, že karta Data nyní zmizela. Je to proto, že do pole nelze zadávat data s dodatečným formátováním, které databáze sama o sobě nedokáže uložit. Přesto je možné do tohoto pole dostat text, označit jej a po aktualizaci jej přenést pomocí makra.

Procedura ContentRead slouží k přenosu obsahu databázového pole „memo“ do formátovaného textového pole MemoFormat a k jeho formátování tak, aby byl zvýrazněn text odpovídající textu ve vyhledávacím poli.

Postup je vázán na **Formulář > Události > Po změně záznamu**.

```
Sub ContentRead(oEvent As Object)
    Dim inMemo As Integer
    Dim oField As Object
    Dim stSearchtext As String
    Dim oCursor As Object
    Dim inSearch As Integer
    Dim inSearchOld As Integer
    Dim inLen As Integer
    oForm = oEvent.Source
    inMemo = oForm.findColumn("memo")
    oField = oForm.getByName("MemoFormat")
    oField.Text = oForm.getString(inMemo)
```

Nejprve jsou definovány proměnné. Poté se z formuláře vyhledá pole tabulky „memo“ a pomocí funkce **getString()** se načte text z očíslovaného sloupce. Ten se přenáší do pole, které lze formátovat, ale které nemá vazbu na databázi: MemoFormat.

První testy ukázaly, že se formulář otevřel, ale nástrojová lišta formuláře v dolní části se již nevytvořila. Proto byla do systému zabudována velmi krátká čekací doba 5/1000 sekund. Poté se zobrazený obsah načte z formuláře FormFilter (který je v hierarchii formulářů paralelní s formulářem Form).

```
Wait 5
stSearchtext =
oForm.Parent.getByname("FormFilter").getbyname("Search").Text
```

Aby bylo možné formátovat text, musí být v poli, které obsahuje text, vytvořen (neviditelný) **TextCursor**. Výchozí zobrazení textu používá 12bodové písmo serif, které se nemusí vyskytovat v jiných částech formuláře a nelze jej přímo přizpůsobit pomocí vlastností ovládacího prvku formuláře. Při tomto postupu se hned na začátku nastaví požadovaný vzhled textu. Pokud se tak nestane, mohou rozdíly ve formátování způsobit odříznutí horního okraje textu v poli. Při prvních testech byly čitelné pouze 2/3 prvního řádku.

Aby neviditelný kurzor označil text, nastaví se nejprve na začátek pole a poté na jeho konec. Argumentem je v obou případech **true**. Poté následují specifikace velikosti, tvaru, barvy a gramáže písma. Pak se kurzor vrátí na začátek.

```
oCursor = oField.createTextCursor()
oCursor.gotoStart(true)
oCursor.gotoEnd(true)
oCursor.CharHeight = 10
oCursor.CharFontName = "Arial, Helvetica, Tahoma"
oCursor.CharColor = RGB(0,0,0)
oCursor.CharWeight = 100.000000 'com::sun::star::awt::FontWeight
oCursor.gotoStart(false)
```

Pokud je v poli text a byl zadán požadavek na vyhledávání, je nyní tento text prohledáván, aby byl nalezen hledaný řetězec. Vnější smyčka se nejprve ptá, zda jsou tyto podmínky splněny; vnitřní smyčka zjišťuje, zda se hledaný řetězec skutečně nachází v textu v poli MemoFormat. Tato nastavení lze ve skutečnosti vynechat, protože dotaz, na němž je formulář založen, zobrazuje pouze text, který tyto podmínky splňuje.

```
If oField.Text <> "" And stSearchtext <> "" Then
  If instr(oField.Text, stSearchtext) Then
    inSearch = 1
    inSearchOld = 0
    inLen = Len(stSearchtext)
```

V textu se hledá hledaný řetězec. To probíhá ve smyčce, která skončí, jakmile se nezobrazí žádná další shoda. **InStr()** vrací umístění prvního znaku hledaného řetězce v zadaném formátu zobrazení, nezávisle na velikosti písmen. Smyčka je řízena požadavkem, aby na konci každého cyklu byl začátek **inSearch** zvýšen o 1 (-1 v prvním řádku smyčky a +2 v posledním řádku). V každém cyklu se kurzor přesune na počáteční pozici bez označení pomocí **oCursor.goRight(Position, false)** a poté doprava s označením podle délky hledaného řetězce. Poté se použije požadované formátování (modré a poněkud tučnější) a kurzor se přesune zpět do výchozího bodu pro další běh.

```

Do While InStr(inSearch, oField.Text, stSearchtext) > 0
    inSearch = InStr(inSearch, oField.Text, stSearchtext) - 1
    oCursor.goRight(inSearch-inSearchOld, false)
    oCursor.goRight(inLen, true)
    oCursor.CharColor = RGB(102, 102, 255)
    oCursor.CharWeight = 110.000000
    oCursor.goLeft(inLen, false)
    inSearchOld = inSearch
    inSearch = inSearch + 2
Loop
End If
End If
End Sub

```

Procedura ContentWrite slouží k přenosu obsahu textového pole MemoFormat do databáze. To probíhá nezávisle na tom, zda dojde k nějaké změně.

Postup je vázán na **Formulář > Události > Před změnou záznamu**.

```

Sub ContentWrite(oEvent As Object)
    Dim oForm As Object
    Dim inMemo As Integer
    Dim loID As Long
    Dim oField As Object
    Dim stMemo As String
    oForm = oEvent.Source
    If InStr(oForm.ImplementationName, "ODatabaseForm") Then

```

Spouštěcí událost je implementována dvakrát. Správný přístup k záznamu umožňuje pouze název implementace, který končí na OdatabaseForm (implementace jsou vysvětleny na straně 468).

```

    If Not oForm.isBeforeFirst() And Not oForm.isAfterLast() Then

```

Při načítání formuláře nebo při opětovném načtení stojí kurzor před aktuálním záznamem. Pokud se o to pokusíme, zobrazí se zpráva „Invalid cursor status“.

```

        inMemo = oForm.findColumn("memo")
        loID = oForm.findColumn("ID")
        oField = oForm.getByname("MemoFormat")
        stMemo = oField.Text
        If stMemo <> "" Then
            oForm.updateString(inMemo, stMemo)
        End If
        If stMemo <> "" And oForm.getString(loID) <> "" Then
            oForm.UpdateRow()
        End If
    End If

```

```
End If
```

```
End If
```

```
End Sub
```

Pole tabulky „memo“ se nachází ve zdroji dat formuláře spolu s polem „ID“. Pokud pole MemoFormat obsahuje text, přenesení se do pole Memo zdroje dat pomocí `oForm.updateString()`. Pouze pokud je v poli ID záznam (jinými slovy byl nastaven primární klíč), následuje aktualizace. V opačném případě je nový záznam vložen běžnou prací s formulářem; formulář změnu rozpozná a uloží ji samostatně.

Kontrola pravopisu při zadávání dat

Toto makro lze použít pro víceřádková formátovaná textová pole. Stejně jako v předchozí kapitole je třeba nejprve zapsat obsah každého záznamu a poté lze nový záznam načíst do ovládacího prvku formuláře. Procedury TransferContent a WriteContent se liší pouze v tom, v jakém okamžiku lze funkci vyhledávání vyřadit ze závorky.

Introduction

In everyday office operation, spreadsheets are regularly used to aggregate sets of data and to perform some kind of analyses on them. As the data in a spreadsheet is laid out in a table view, plainly visible and able to be edited or added to, many users ask why they should use a database instead of a spreadsheet. This handbook explains the differences between the two, beginning with a short section on what can be done with a database.

This chapter introduces two database examples and the entire Handbook is built around these. One database is named Media_without_macros.odt and the other, extended with the inclusion of macros, is named Media_with_macros.odt.

Obrázek 430: Formulář pro kontrolu pravopisu

Kontrola pravopisu se ve výše uvedeném formuláři spustí vždy, když je v ovládacím prvku formuláře stisknuta mezera nebo návrat. Jinými slovy, spustí se na konci každého slova. Mohlo by to být také spojeno se ztrátou zaměření ovládacího prvku, aby se zajistilo, že bude zkontrolováno poslední slovo.

Postup je vázán na **Formulář > Události > Klávesa uvolněna**.

```
SUB MarkWrongWordsDirect(oEvent As Object)
```

```
GlobalScope.BasicLibraries.LoadLibrary("Tools")
```

Funkce `RTriMStr` slouží k odstranění interpunkčního znaménka na konci řetězce. Jinak by se všechna slova, která končí čárkou, tečkou nebo jiným interpunkčním znaménkem, zobrazovala nahoře jako pravopisné chyby. Kromě toho se funkce `LTrimChar` používá k odstranění závorek na začátku slov.

```
Dim aProp() As New com.sun.star.beans.PropertyValue
```

```
Dim oLinuSvcMgr As Object
```

```
Dim oSpellChk As Object
```

```
Dim oField As Object
```

```
Dim arText()
```

```
Dim stWord As String
```

```

Dim inlenWord As Integer
Dim ink As Integer
Dim i As Integer
Dim oCursor As Object
Dim stText As Object
oLinguSvcMgr =
createUnoService("com.sun.star.linguistic2.LinguServiceManager")
If Not IsNull(oLinguSvcMgr) Then
    oSpellChk = oLinguSvcMgr.getSpellChecker()
End If

```

Nejprve se deklarují všechny proměnné. Poté je zpřístupněn modul Basic pro kontrolu pravopisu **SpellChecker**. Právě tento modul bude skutečně kontrolovat správnost jednotlivých slov.

```

oField = oEvent.Source.Model
ink = 0
If oEvent.KeyCode = 1280 Or oEvent.KeyCode = 1284 Then

```

Událostí, která makro spustí, je stisknutí klávesy. Tato událost obsahuje kód **KeyCode** pro každou jednotlivou klávesu. **KeyCode** pro klávesu Return je 1280, pro mezeru 1284. Stejně jako mnoho dalších informací se tyto položky získávají pomocí nástroje Xray. Po stisknutí mezerníku nebo klávesy Enter se zkontroluje pravopis. Jinými slovy se spouští na konci každého slova. Pouze test posledního slova neprobíhá automaticky.

Při každém spuštění makra se zkontrolují všechna slova v textu. Kontrola jednotlivých slov by byla také možná, ale vyžadovala by mnohem více práce.

Text je rozdělen na jednotlivá slova. Oddělovačem je znak mezery. Předtím je třeba slova rozdělená zalomeními řádků opět spojit, jinak by mohlo dojít k záměně těchto částí za celá slova.

```

stText = Join(Split(oField.Text,CHR(10)), " ")
stText = Join(Split(stText,CHR(13)), " ")
arText = Split(RTrim(stText)," ")
For i = LBound(arText) To Ubound(arText)
    stWord = arText(i)
    inlenWord = len(stWord)
    stWord = Trim( RtrimStr( RtrimStr( RtrimStr( RtrimStr(
RtrimStr(
    RtrimStr(stWord,","), "."), "?"), "!"), "."), ")"))
    stWord = LTrimChar(stWord,"(")

```

Jednotlivá slova jsou přečtena. Jejich nezkrácená délka je potřebná pro následující krok úprav. Jen tak lze určit pozici slova v rámci celého textu (což je nezbytné pro konkrétní zvýraznění pravopisných chyb).

Funkce **Trim** se použije pro odstranění mezer, zatímco **RTrimStr** odstraní čárky a tečky na konci textu a **LTrimChar** jakákoliv interpunkční znaménka na začátku.

```

If stWord <> "" Then
    oCursor = oField.createTextCursor()
    oCursor.gotoStart(false)

```



```

oCursor.goRight(ink, false)
oCursor.goRight(inLenWord, true)
If Not oSpellChk.IsValid(stWord, "en", aProp()) Then
    oCursor.CharUnderline = 9
    oCursor.CharUnderlineHasColor = True
    oCursor.CharUnderlineColor = RGB(255, 51, 51)
Else
    oCursor.CharUnderline = 0
End If
End If
ink = ink + inLenWord + 1
Next
End If
End Sub

```

Pokud slovo není nulové, vytvoří se textový kurzor. Tento kurzor se přesune bez zvýraznění na začátek textu v zadávaném poli. Pak skočí dopředu doprava, stále bez zvýraznění, na výraz uložený v proměnné **ink**. Tato proměnná začíná jako 0, ale po spuštění první smyčky je rovna délce slova (+1 za následující mezeru). Pak se kurzor posune doprava o délku aktuálního slova. Vlastnosti písma jsou upraveny tak, aby vytvořily zvýrazněnou oblast.

Spustí se **kontrola pravopisu**. Jako argumenty vyžaduje slovo a kód země; bez kódu země se vše považuje za správné. Argument pole je obvykle prázdný.

Pokud slovo není ve slovníku, je označeno červenou vlnovkou. Tento typ podtržení je zde reprezentován '9'. Pokud je slovo nalezeno, není podtrženo ('0'). Tento krok je nutný, protože jinak by se slovo rozpoznalo jako chybné a následně opravené nadále zobrazovalo červenou vlnovkou. Nikdy by nebyl odstraněn, protože nebyl dán žádný konfliktní formát.

Pole se seznamem jako seznamy s možností zadání

Tabulku s jedním záznamem lze vytvořit přímo pomocí polí se seznamem a neviditelných číselných polí a odpovídající primární klíč zadat do jiné tabulky.

Tip

Použití polí se seznamem namísto seznamů nalezneme v databázi Example_Combobox_Listfield.odb, která je přiřazena k této knize.

Ovládací prvek Pole se seznamem zachází s formulářovými poli pro kombinované zadávání a výběr hodnot (pole se seznamem) jako se seznamem s možností zadání. Za tímto účelem se kromě polí se seznamem ve formuláři ukládají hodnoty klíčových polí, které se mají přenést do podkladové tabulky, do samostatných číselných polí. Pole lze deklarovat jako neviditelná. Klíče z těchto polí jsou načteny při načtení formuláře a pole se seznamem je nastaveno tak, aby zobrazovalo odpovídající obsah. Pokud se obsah pole se seznamem změní, uloží se a nový primární klíč se přenesou do příslušného číselného pole, které se uloží do hlavní tabulky.

Pokud se místo tabulek používají upravitelné dotazy, lze text, který se má zobrazit v kombinačních polích, určit přímo z dotazu. Pro tento krok pak není nutné používat makro.

Předpokladem pro fungování makra je, že primárním klíčem tabulky, která je zdrojem dat pro kombinační pole, je automaticky se zvyšující celé číslo. Předpokládá se také, že název pole pro primární klíč je ID.

Zobrazení textu v polích se seznamem

Tento podprogram má zobrazit text v poli se seznamem podle hodnoty neviditelných polí cizího klíče z hlavního formuláře. Lze ji použít i pro seznamy, které odkazují na dvě různé tabulky. K tomu může dojít, pokud je například poštovní směrovací číslo v poštovní adrese uloženo odděleně od města. V takovém případě lze poštovní směrovací číslo načíst z tabulky, která obsahuje pouze cizí klíč pro město. V seznamu by se mělo zobrazit poštovní směrovací číslo a město společně.

```
Sub ShowText(oEvent As Object)
```

Toto makro by mělo být vázáno na následující událost formuláře: 'Po změně záznamu'.

Makro se volá přímo z formuláře. Spouštěcí událost je zdrojem všech proměnných, které makro potřebuje. Některé proměnné již byly deklarovány globálně v samostatném modulu a nejsou zde znovu deklarovány.

```
Dim oForm As Object
Dim oFieldList As Object
Dim stFieldValue As String
Dim inCom As Integer
Dim stQuery As String
oForm = oEvent.Source
```

Ve formuláři je skrytý ovládací prvek, ze kterého lze získat názvy všech různých polí se seznamem. Tato pole se seznamem jsou jedno po druhém zpracovávána makrem.

```
aComboboxes() = Split(oForm.GetName("combofields").Tag, ",")
For inCom = LBound(aComboboxes) TO UBound(aComboboxes)
    ...
Next inCom
```

Doplňující informace (Tag) připojené ke skrytému ovládacímu prvku obsahují tento seznam názvů polí se seznamem oddělených čárkami. Názvy jsou zapsány do pole a poté zpracovány v rámci smyčky. Smyčka končí výrazem **NEXT**.

Pole se seznamem, které nahradilo seznam, se nazývá **oFieldList**. Abychom získali cizí klíč, potřebujeme správný sloupec v tabulce, která je základem formuláře. To je dostupné pomocí názvu pole tabulky, který je uložen v doplňkových informacích pole se seznamem.

```
oFieldList = oForm.GetName(Trim(aComboboxes(inCom)))
stFieldID = oForm.GetString(oForm.FindColumn(oFieldList.Tag))
oFieldList.Refresh()
```

Pole se seznamem se znovu načte pomocí **Refresh()** v případě, že se obsah pole změnil vložením nových dat.

Dotaz potřebný k zobrazení viditelného obsahu pole se seznamem je založen na poli, které je základem ovládacího prvku, a na hodnotě určené pro cizí klíč. Aby byl kód SQL použitelný, jsou odstraněny všechny případné operace třídění. Poté se zkontrolují případné definice relací (které budou začínat slovem **WHERE**). Ve výchozím nastavení funkce **InStr()** nerozlišuje mezi velkými a malými písmeny, takže zahrnuje všechny kombinace velkých a malých písmen. Pokud existuje relace, znamená to, že dotaz obsahuje pole ze dvou různých tabulek. Potřebujeme najít tabulku,

kteřá poskytuje cizí klíč pro odkaz. Makro zde závisí na tom, že primární klíč v každé tabulce se nazývá ID.

Pokud není definována žádná relace, dotaz přistupuje pouze k jedné tabulce. Informace o tabulce lze vynechat a podmínku formulovat přímo pomocí hodnoty cizího klíče.

```
If stFieldID <> "" Then
    stQuery = oFieldList.ListSource
    If InStr(stQuery, "order by") > 0 Then
        stSql = Left(stQuery, InStr(stQuery, "order by")-1)
    Else
        stSql = stQuery
    End If
    If InStr(stSql, "where") Then
        st = Right(stSql, Len(stSql)-InStr(stSql, "where")-4)
        If InStr(Left(st, InStr(st, "=")), ".ID") Then
            a() = Split(Right(st, Len(st)-InStr(st, "=")-1), ".")
        Else
            a() = Split(Left(st, InStr(st, "=")-1), ".")
        End If
        stSql = stSql + "AND "+a(0)+".ID" = "+stFieldID
    Else
        stSql = stSql + "WHERE ID" = "+stFieldID
    End If
```

Každý název pole a tabulky musí být do příkazu SQL zadán se dvěma sadami uvozovek. Uvozovky jsou normálně interpretovány Basicem jako oddělovače textových řetězců, takže se při předávání kódu do jazyka SQL již nezobrazují. Zdvojení uvozovek zajistí, že se předá jedna sada. "ID" značí, že v dotazu bude použito pole "ID" s jednou sadou uvozovek, kterou SQL vyžaduje.

Nyní se provede dotaz uložený v proměnné **stSql** a jeho výsledek se uloží do proměnné **oResult**.

```
oDatasource = ThisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
oResult = oSQL_Command.executeQuery(stSql)
```

Výsledek dotazu je načten ve smyčce. Stejně jako u dotazu v grafickém uživatelském rozhraní lze zobrazit několik polí a záznamů. Konstrukce tohoto dotazu však vyžaduje pouze jeden výsledek, který se nachází v prvním sloupci (**1**) množiny výsledků dotazu. Jedná se o záznam, který poskytuje zobrazený obsah pole se seznamem. Obsahem je text (**getString()**), proto příkaz **oResult.getString(1)**.

```

While oResult.next
    stFieldValue = oResult.getString(1)
Wend

```

Nyní musí být pole se seznamem nastaveno na textovou hodnotu získanou dotazem.

```

oFieldList.Text = stFieldValue
Else

```

Pokud v poli cizího klíče **oField** není žádná hodnota, dotaz se nezdařil a pole se seznamem je nastaveno na prázdný řetězec.

```

oFieldList.Text = ""
End If
Next inCom

```

```
End Sub
```

Tento postup spravuje kontakt mezi polem se seznamem a cizím klíčem dostupným v poli zdroje dat formuláře. To by mělo stačit k zobrazení správných hodnot v polích se seznamem. Uložení nových hodnot by vyžadovalo další postup.

Přenos hodnoty cizího klíče z pole se seznamem do číselného pole

Pokud je do pole se seznamem zadána nová hodnota (a to je koneckonců účel, pro který bylo toto makro vytvořeno), musí být odpovídající primární klíč zadán do základní tabulky formuláře jako cizí klíč.

```
Sub TextSelectionSaveValue(oEvent As Object)
```

Toto makro by mělo být vázáno na následující událost formuláře: 'Před akcí záznamu'.

Po deklarování proměnných (zde nejsou uvedeny) musíme nejprve přesně určit, která událost má makro spustit. Před akcí záznamu se postupně zavolají dvě implementace. Je důležité, aby makro samo načetlo objekt formuláře. To lze provést v obou implementacích, ale různými způsoby. Zde je odfiltrována implementace s názvem OdatabaseForm.

```

If InStr(oEvent.Source.ImplementationName, "ODatabaseForm") Then
    ...
End If

```

```
End Sub
```

Tato smyčka se vytvoří na stejném začátku jako procedura **Display_text**:

```

oForm = oEvent.Source
aComboboxes() = Split(oForm.getByname("combofields").Tag, ",")
For inCom = LBound(aComboboxes) To Ubound(aComboboxes)
    ...
Next inCom

```

Pole **oFieldList** zobrazuje text. Může se nacházet uvnitř ovládacího prvku tabulky a v takovém případě k němu není možné přistupovat přímo z formuláře. V takových případech by doplňkové informace pro skryté ovládací prvky polí se seznamem měly obsahovat cestu k poli pomocí pole se seznamem „tablecontrol“. Rozdělení této položky odhalí, jakým způsobem se má k poli se seznamem přistupovat.

```

a() = Split(Trim(aComboboxen(inCom)), ">")
If Ubound(a) > 0 Then

```

```

oFieldList = oForm.GetByName(a(0)).GetByName(a(1))
Else
oFieldList = oForm.GetByName(a(0))
End If

```

Poté je dotaz načten z pole se seznamem a rozdělen na jednotlivé části. U jednoduchých polí se seznamem jsou nezbytnými informacemi název pole a název tabulky:

```
SELECT "Field" FROM "Table"
```

To by v některých případech mohlo být doplněno o pokyn k třídění. Kdykoli mají být v poli se seznamem dvě pole pohromadě, bude třeba více práce s jejich oddělením.

```
SELECT "Field1" || ' ' || "Field2" FROM "Table"
```

Tento dotaz spojí dvě pole a vloží mezi ně mezeru. Protože oddělovačem je mezeru, makro ji vyhledá a rozdělí text na dvě části. To samozřejmě bude spolehlivě fungovat pouze tehdy, pokud Field již neobsahuje text, ve kterém jsou mezery povoleny. V opačném případě, pokud je křestní jméno „Anne Marie“ a příjmení „Müller“, bude makro považovat „Anne“ za křestní jméno a „Marie Müller“ za příjmení. V takových případech je třeba použít vhodnější oddělovač, který pak může být nalezen makrem. V případě jmen by to mohlo být „Příjmení, Jméno“.

Situace se ještě více zkomplikuje, pokud obě pole pocházejí z různých tabulek:

```

SELECT "Table1"."Field1" || ' > ' || "Table2"."Field2"
FROM "Table1", "Table2"
WHERE "Table1"."ID" = "Table2"."ForeignID"
ORDER BY "Table1"."Field1" || ' > ' || "Table2"."Field2" ASC

```

Zde je třeba pole od sebe oddělit, určit tabulku, do které každé pole patří, a určit odpovídající cizí klíče.

```

stQuery = oFieldList.ListSource
aFields() = Split(stQuery, " ")
stContent = ""
For i=LBound(aFields)+1 To UBound(aFields)

```

Obsah dotazu je zbaven zbytečného balastu. Jednotlivé části jsou znovu sestaveny do pole s neobvyklou kombinací znaků jako oddělovačem. FROM odděluje zobrazení viditelných polí od názvů tabulek. WHERE odděluje podmínku od názvů tabulek. Spojení nejsou podporována.

```

If Trim(UCASE(aFields(i))) = "ORDER BY" Then
Exit For
ElseIf Trim(UCASE(aFields(i))) = "FROM" Then
stContent = stContent+" §§ "
ElseIf Trim(UCASE(aFields(i))) = "WHERE" Then
stContent = stContent+" §§ "
Else
stContent = stContent+Trim(aFields(i))
End If
Next i
aContent() = Split(stContent, " §§ ")

```

V některých případech pochází obsah viditelného zobrazení polí z různých polí:

```
aFirst() = Split(aContent(0), "||")
If UBound(aFirst) > 0 Then
    If UBound(aContent) > 1 Then
```

První část obsahuje nejméně dvě pole. Pole začínají názvem tabulky. Druhá část obsahuje dva názvy tabulek, které lze určit z první části. Třetí část obsahuje vztah s cizím klíčem, oddělený znakem =:

```
aTest() = Split(aFirst(0), ".")
NameTable1 = aTest(0)
NameTableField1 = aTest(1)
Erase aTest
stFieldSeparator = Join(Split(aFirst(1), ""), "")
aTest() = Split(aFirst(2), ".")
NameTable2 = aTest(0)
NameTableField2 = aTest(1)
Erase aTest
aTest() = Split(aContent(2), "=")
aTest1() = Split(aTest(0), ".")
If aTest1(1) <> "ID" Then
    NameTab12ID = aTest1(1)
    IF aTest1(0) = NameTable1 Then
        Position = 2
    Else
        Position = 1
    End If
Else
    Erase aTest1
    aTest1() = Split(aTest(1), ".")
    NameTab12ID = aTest1(1)
    If aTest1(0) = NameTable1 Then
        Position = 2
    Else
        Position = 1
    End If
End If
Else
```

První část obsahuje dva názvy polí bez názvů tabulek, případně s oddělovači. Druhá část obsahuje názvy tabulek. Třetí část neexistuje:

```
If UBound(aFirst) > 1 Then
```

```

        NameTableField1 = aFirst(0)
        stFieldSeparator = Join(Split(aFirst(1), ""), "")
        NameTableField2 = aFirst(2)
    Else
        NameTableField1 = aFirst(0)
        NameTableField2 = aFirst(1)
    End If
    NameTable1 = aContent(1)
End If
Else

```

Z jedné tabulky je pouze jedno pole:

```

        NameTableField1 = aFirst(0)
        NameTable1 = aContent(1)
    End If

```

Maximální délka znaku, kterou může záznam mít, je dána funkcí **ColumnSize**. Pole se seznamem nelze použít k omezení velikosti, protože může být nutné, aby obsahovalo dvě pole současně.

```

    LengthField1 = ColumnSize(NameTable1, NameTableField1)
    If NameTableField2 <> "" Then
        If NameTable2 <> "" Then
            LengthField2 = ColumnSize(NameTable2, NameTableField2)
        Else
            LengthField2 = ColumnSize(NameTable1, NameTableField2)
        End If
    Else
        LengthField2 = 0
    End If

```

Obsah pole se seznamem se načte:

```

    stContent = oFieldList.getCurrentValue()

```

V případě potřeby se odstraní počáteční a koncové mezery a netisknutelné znaky.

```

    stContent = Trim(stContent)
    If stContent <> "" Then
        If NameTableField2 <> "" Then

```

Pokud existuje druhé pole tabulky, musí být obsah pole se seznamem rozdělen. Pro určení místa, kde má dojít k rozdělení, použijeme oddělovač polí, který je funkcí zadán jako argument.

```

        a_stParts = Split(stContent, FieldSeparator, 2)

```

Poslední parametr znamená, že maximální počet částí je 2.

Podle toho, který údaj odpovídá poli 1 a který poli 2, se nyní obsah pole se seznamem přiřadí jednotlivým proměnným. "Position = 2" zde slouží jako označení, že druhá část obsahu znamená pole 2.

```

If Position = 2 Then
    stContent = Trim(a_stParts(0))
    If UBound(a_stParts()) > 0 Then
        stContentField2 = Trim(a_stParts(1))
    Else
        stContentField2 = ""
    End If
    stContentField2 = Trim(a_stParts(1))
Else
    stContentField2 = Trim(a_stParts(0))
    If UBound(a_stParts()) > 0 Then
        stContent = Trim(a_stParts(1))
    Else
        stContent = ""
    End If
    stContent = Trim(a_stParts(1))
End If
End If

```

Může se stát, že při dvou oddělitelných obsazích se instalovaná velikost pole se seznamem (délka textu) nevejde do polí tabulky, která se mají uložit. U polí se seznamem, která představují jediné pole, se to obvykle řeší vhodnou konfigurací ovládacího prvku formuláře. Zde naopak potřebujeme nějaký způsob, jak takové chyby zachytit. Kontroluje se maximální přípustná délka příslušného pole.

```

If (LengthField1 > 0 And Len(stContent) > LengthField1) Or
(LengthField2 > 0 And Len(stContentField2) > LengthField2) Then

```

Pokud je délka pole první nebo druhé části příliš velká, uloží se do jedné z proměnných výchozí řetězec. Znak **Chr(13)** se používá pro zalomení řádku.

```

    stmsgbox1 = "The field " + NameTableField1 + " must not exceed " +
Field1Length + "characters in length." + Chr(13)
    stmsgbox2 = "The field " + NameTableField2 + " must not exceed " +
Field2Length + "characters in length." + Chr(13)

```

Pokud jsou oba obsahy polí příliš dlouhé, zobrazí se oba texty.

```

If (LengthField1 > 0 And Len(stContent) > LengthField1) And
(LengthField2 > 0 And Len(stContentField2) > LengthField2) Then
    MsgBox("The entered text is too long." + Chr(13) + stmsgbox1 +
stmsgbox2 + "Please shorten it.", 64, "Invalid entry")

```

Zobrazení používá funkci **MsgBox()**. Jako první argument se očekává textový řetězec, dále volitelně číslo (které určuje typ zobrazeného okna zprávy) a nakonec volitelný textový řetězec jako název okna. V okně se proto zobrazí nadpis "Invalid entry" (Neplatné zadání, pozn. překl.) a číslo "64" poskytuje pole obsahující symbol Informace.

Následující kód se vztahuje na všechny další případy, kdy se může vyskytnout příliš dlouhý text.

```

ElseIf (Field1Length > 0 And Len(stContent) > Field1Length) Then

```



```

        MsgBox("The entered text is too long." + Chr(13) + stmsgbox1 +
        "Please shorten it.", 64, "Invalid entry")
    Else
        MsgBox("The entered text is too long." + Chr(13) + stmsgbox2 +
        "Please shorten it.", 64, "Invalid entry")
    End If
Else

```

Pokud není text příliš dlouhý, může funkce pokračovat. V opačném případě se zde ukončí. Nyní jsou položky maskovány tak, aby případné uvozovky nevyvolaly chybu.

```

        stContent = String_to_SQL(stContent)
        If stContentField2 <> "" Then
            stContentField2 = String_to_SQL(stContentField2)
        End If

```

Nejprve jsou předalokovány proměnné, které lze následně dotazem měnit. Proměnné inID1 a inID2 uchovávají obsah polí primárních klíčů obou tabulek. Pokud dotaz neposkytne žádné výsledky, přiřadí Basic této celočíselné proměnné hodnotu 0. Tato hodnota by však také mohla znamenat úspěšný dotaz vracející hodnotu primárního klíče 0; proto je proměnná přednastavena na -1. HSQLDB nemůže tuto hodnotu nastavit pro pole automatické hodnoty.

Dále se nastaví připojení k databázi, pokud ještě neexistuje.

```

inID1 = -1
inID2 = -1
oDatasource = ThisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
If NameTableField2 <> "" And Not IsEmpty(stContentField2) And
NameTable2 <> "" Then

```

Pokud existuje druhé pole tabulky, musí být nejprve deklarována druhá závislost.

```

        stSql = "SELECT ""ID"" FROM "" + NameTable2 + "" WHERE "" +
NameTableField2 + ""="" + stContentField2 + """
        oResult = oSQL_Command.executeQuery(stSql)
        While oResult.next
            inID2 = oResult.getInt(1)
        Wend
        If inID2 = -1 Then
            stSql = "INSERT INTO "" + NameTable2 + "" ("" +
NameTableField2 + "" ) VALUES ('" + stContentField2 + "') "
            oSQL_Command.executeUpdate(stSql)
            stSql = "CALL IDENTITY()"

```

Pokud se obsah v poli se seznamem nenachází v příslušné tabulce, vloží se tam. Výsledná hodnota primárního klíče se pak přečte. Pokud je přítomen, načte se existující primární klíč stejným způsobem. Funkce používá automaticky generovaná pole primárního klíče (**IDENTITY**).

```

oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inID2 = oResult.getInt(1)
Wend
End If

```

Primární klíč pro druhou hodnotu je dočasně uložen v proměnné **inID2** a poté zapsán jako cizí klíč do tabulky odpovídající první hodnotě. Podle toho, zda byl záznam z první tabulky již k dispozici, se obsah čerstvě uloží (**INSERT**) nebo změní (**UPDATE**):

```

If inID1 = -1 Then
    stSql = "INSERT INTO "" + NameTable1 + "" ("" +
NameTableField1 + "", "" + NameTab12ID + "") VALUES (' + stContent +
', ' + inID2 + ')"
    oSQL_Command.executeUpdate(stSql)

```

A odpovídající ID se přímo načte:

```

stSql = "CALL IDENTITY()"
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inID1 = oResult.getInt(1)
Wend

```

Primární klíč první tabulky je nakonec nutné znovu načíst, aby mohl být přenesen do základní tabulky formuláře.

```

Else
    stSql = "UPDATE "" + NameTable1 + "" SET "" + NameTab12ID
+ ""=' + inID2 + ' WHERE "" + NameTableField1 + "" = ' + stContent +
'"
    oSQL_Command.executeUpdate(stSql)
End If
End If

```

V případě, že obě pole, která jsou základem pole se seznamem, jsou ve stejné tabulce (například Surname a Firstname v tabulce Name), je třeba použít jiný dotaz:

```

If NameTableField2 <> "" And NameTable2 = "" Then
    stSql = "SELECT ""ID"" FROM "" + NameTable1 + "" WHERE "" +
NameTableField1 + ""=' + stContent + ' AND "" + NameTableField2 +
""=' + stContentField2 + '"
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
        inID1 = oResult.getInt(1)
    Wend
    If inID1 = -1 Then

```

... a druhá tabulka neexistuje:

```
stSql = "INSERT INTO "" + NameTable1 + "" (" + NameTableField1 + ""," + NameTableField2 + "" ) VALUES (' + stContent + ',' + stContentField2 + ')"
oSQL_Command.executeUpdate(stSql)
```

Poté se znovu načte primární klíč.

```
stSql = "CALL IDENTITY()"
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inID1 = oResult.getInt(1)
Wend
End If
End If
IF NameTableField2 = "" Then
```

Nyní se budeme zabývat nejjednodušším případem: Druhé pole tabulky neexistuje a záznam v tabulce ještě není. Jinými slovy, do pole se seznamem byla zadána jedna nová hodnota.

```
stSql = "SELECT ""ID"" FROM "" + NameTable1 + "" WHERE "" + NameTableField1 + ""=' + stContent + '"
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inID1 = oResult.getInt(1)
Wend
If inID1 = -1 Then
```

Pokud druhé pole neexistuje, vloží se obsah pole jako nový záznam.

```
stSql = "INSERT INTO "" + NameTable1 + "" (" + NameTableField1 + "" ) VALUES (' + stContent + ')"
oSQL_Command.executeUpdate(stSql)
```

... a výsledné ID se přímo odečte.

```
stSql = "CALL IDENTITY()"
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inID1 = oResult.getInt(1)
Wend
End If
End If
```

Je třeba určit hodnotu pole primárního klíče, aby ji bylo možné přenést do hlavní části formuláře.

Poté se hodnota primárního klíče, která vznikla na základě všech těchto smyček, přenesou do neviditelného pole v hlavní tabulce a v základní databázi. Pole tabulky propojené s polem formuláře se získá pomocí '**BoundField**'. '**updateInt**' umístí do tohoto pole celé číslo (viz definice číselného typu).

```
oForm.updateLong(oForm.findColumn(oFldList.Tag), inID1)
```

```
End If
ELSE
```

Pokud nemá být zadán žádný primární klíč, protože v poli se seznamem nebyl žádný záznam nebo byl tento záznam odstraněn, musí být odstraněn i obsah neviditelného pole.

updateNull() se použije k vyplnění pole výrazem specifickým pro databázi pro prázdné pole, **NULL**.

```
oForm.updateNULL(oForm.findColumn(oFeldList.Tag), NULL)
End If
NEXT inCom
End If
End Sub
```

Funkce pro měření délky položky pole se seznamem

Následující funkce udává počet znaků v příslušném sloupci tabulky, aby se příliš dlouhé záznamy pouze nezkracovaly. Pro poskytování návratových hodnot je zde zvoleno **Function**. Příkaz **SUB** nemá žádnou návratovou hodnotu, kterou by bylo možné předat a zpracovat jinde.

```
Function ColumnSize(TableName As String, Fieldname As String) As Integer
```

```
oDataSource = ThisComponent.Parent.CurrentController
If Not (oDataSource.isConnected()) Then
oDataSource.connect()
End If
oConnection = oDataSource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
stSql = "SELECT ""COLUMN_SIZE"" FROM
""INFORMATION_SCHEMA"". ""SYSTEM_COLUMNS"" WHERE ""TABLE_NAME"" = ' " +
TableName + "' AND ""COLUMN_NAME"" = ' " + Fieldname + "'
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
i = oResult.getInt(1)
Wend
ColumnSize = i
End Function
```

Generování akcí databáze

```
Sub GenerateRecordAction(oEvent As Object)
```

Toto makro by mělo být vázáno na událost *Při zaměření* pole seznamu. Je nutné, aby se ve všech případech, kdy se mění pole seznamu, změna uložila. Bez tohoto makra by v aktuální tabulce nedošlo k žádné změně, kterou by aplikace Base rozpoznala, protože pole se seznamem není vázáno na formulář.

Toto makro přímo mění vlastnosti formuláře:

```
Dim oForm As Object
oForm = oEvent.Source.Model.Parent
```

```
oForm.IsModified = TRUE
```

```
End Sub
```

Toto makro není nutné pro formuláře, které používají dotazy pro obsah polí se seznamem. Změny v polích se seznamem se zaznamenávají přímo.

Navigace z jednoho formuláře do druhého

Formulář se má otevřít, když nastane určitá událost.

Ve vlastnostech ovládacího prvku formuláře zadáme na řádku "Další informace" (značka) název formuláře. Zde lze také zadat další informace a následně je oddělit pomocí funkce **Split()**.

```
Sub From_form_to_form(oEvent As Object)
```

```
Dim stTag As String
```

```
stTag = oEvent.Source.Model.Tag
```

```
aForm() = Split(stTag, ",")
```

Pole je deklarováno a naplněno názvy formulářů; jednak formuláře, který má být otevřen, a jednak aktuálního formuláře, který bude po otevření druhého zavřen.

```
ThisDatabaseDocument.FormDocuments.GetByName( Trim(aForm(0)) ).open
```

```
ThisDatabaseDocument.FormDocuments.GetByName( Trim(aForm(1)) ).close
```

```
End Sub
```

Pokud se má jiný formulář otevřít až po zavření aktuálního formuláře, například pokud existuje hlavní formulář a všechny ostatní formuláře se z něj ovládají pomocí tlačítek, mělo by být následující makro vázáno na formulář s **Nástroje > Přizpůsobit > Události > Dokument zavřen**:

```
Sub Mainform_open
```

```
ThisDatabaseDocument.FormDocuments.GetByName( "Mainform" ).open
```

```
End Sub
```

Pokud jsou dokumenty formuláře v rámci souboru ODB řazeny do adresářů, musí být makro pro změnu formuláře rozsáhlejší:

```
Sub From_form_to_form_with_folders(oEvent As Object)
```

```
REM Nejprve se zadá formulář, který se má otevřít.
```

```
REMPokud se formulář nachází ve složce, použijeme "/" pro definování relace
```

```
REM, aby bylo možné podsložku najít.
```

```
Dim stTag As String
```

```
stTag = oEvent.Source.Model.Tag 'Tag is entered in the additional  
information
```

```
aForms() = Split(stTag, ",") 'Here the form name for the new form  
comes first, then the one for the old form
```

```
aForms1() = Split(aForms(0), "/")
```

```
aForms2() = Split(aForms(1), "/")
```

```
If UBound(aForms1()) = 0 Then
```

```
ThisDatabaseDocument.FormDocuments.GetByName( Trim(aForms1(0))  
) .open
```

```
Else
```

```

        ThisDatabaseDocument.FormDocuments.getByName( Trim(aForms1(0))
).getByName( Trim(aForms1(1)) ).open
    End If
    If UBound(aForms2()) = 0 Then
        ThisDatabaseDocument.FormDocuments.getByName( Trim(aForms2(0))
).close
    Else
        ThisDatabaseDocument.FormDocuments.getByName( Trim(aForms2(0))
).getByName( Trim(aForms2(1)) ).close
    End If
End Sub

```

Dokumenty formuláře, které leží v adresáři, se zadávají do pole Další informace jako adresář/formulář. To musí být převedeno na:

```
...getByName("Directory").getByName("Form").
```

Hierarchické seznamy

Nastavení v jednom poli seznamu mají přímo ovlivňovat nastavení jiného pole. Pro jednoduché případy to již bylo popsáno výše v části o filtrování záznamů. Předpokládejme však, že první pole seznamu má ovlivnit obsah druhého pole seznamu, které pak ovlivní obsah třetího pole seznamu atd.

Ročník	Třída	Jméno
1	a	Karl Müller
2	b	Evelyn Maier
3	c	Maria Gott
4	d	Eduard Abgefahren
5	e	Kurt Drechsler
6	f	Kunigunde Schimmel
7	g	
8		
9		
10		
11		
12		
13		

Obrázek 431: Příklad polí se seznamem pro hierarchické uspořádání polí se seznamem

V tomto příkladu obsahuje první pole seznamu (Jahrgang = Years) všechny školní roky. Classes (Třídy, pozn. překl.) v jednotlivých ročnících jsou označeny písmeny. Names (Jména, pozn. překl.) jsou jmény členů třídy.

Za normálních okolností by se v seznamu Years zobrazilo všech 13 ročníků, v seznamu Classes všechna písmena tříd a v seznamu Names všichni žáci školy.

Pokud se jedná o hierarchické seznamy, je výběr tříd po výběru roku omezen. Zobrazena jsou pouze ta písmena tříd, která se v daném roce skutečně vyskytují. To se může lišit, protože pokud se zvyšuje počet žáků, může se zvýšit i počet tříd v ročníku. Poslední pole seznamu, Names, je velmi omezené. Namísto více než 1000 žáků jich bude jen 30.

Na začátku lze vybrat pouze rok. Po provedení této operace se zpřístupní (omezený) seznam tříd. Teprve na konci je uveden seznam jmen.

Pokud je pole seznamu Years změněno, musí se sekvence začít znovu. Pokud se změní pouze pole Classes, zůstane v platnosti číslo roku pro poslední pole seznamu.

Aby bylo možné takovou funkci vytvořit, musí být ve formuláři možné uložit prostřední proměnnou. To probíhá ve skrytém ovládacím prvku.

Makro je vázáno na změnu obsahu pole se seznamem: **Vlastnosti pole se seznamem > Události > Změněno**. Potřebné proměnné jsou uloženy v doplňkových informacích pole seznamu.

Zde je příklad doplňujících informací:

MainForm, Year, hidden control, Listbox_2

Formulář se nazývá MainForm. Aktuální pole seznamu se jmenuje Listbox1. Toto pole seznamu zobrazuje obsah pole tabulky Year a následující pole seznamu musí být filtrována podle této položky. Skrytý ovládací prvek je označen hidden_control a existence druhého pole seznamu (Listbox_2) je předána filtrovací proceduře.

```
Sub Hierarchical_control(oEvent As Object)
```

```
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oForm As Object
    Dim oFieldHidden As Object
    Dim oField As Object
    Dim oField1 As Object
    Dim stSql As String
    Dim acontent()
    Dim stTag As String
    oField = oEvent.Source.Model
    stTag = oField.Tag
    oForm = oField.Parent
```

```
    REM Značka přechází do pole Další informace
```

```
    REM Obsahuje:
```

```
    REM 0. Název pole, které se má v tabulce filtrovat
```

```
    REM 1. Název pole skrytého ovládacího prvku, do kterého bude uložena  
    filtrovaná hodnota
```

```
    REM 2. Případné další pole se seznamem
```

```
    REM Značka se načte z prvku, který makro spouští. Proměnná je
```

```
    REM předáno proceduře a případně všem dalším polím se seznamem
```

```
    aFilter() = Split(stTag, ",")
```

```
stFilter = ""
```

Po deklaraci proměnných se obsah značky předá do pole, aby bylo možné přistupovat k jednotlivým prvkům. Poté se deklaruje přístup k jednotlivým polím formuláře.

Určí se pole seznamu, které makro vyvolalo, a přečte se jeho hodnota. Pouze pokud tato hodnota není NULL, bude zkombinována s názvem filtrovaného pole, v našem příkladu Year, a vytvoří příkaz SQL. Jinak zůstane filtr prázdný. Pokud jsou pole seznamu určena k filtrování formuláře, není k dispozici žádný skrytý ovládací prvek. V tomto případě je hodnota filtru uložena přímo ve formuláři.

```
If Trim(aFilter(1)) = "" Then
  If oField.GetCurrentValue <> "" Then
    stFilter = """"+Trim(aFilter(0))
+""""=''+oField.GetCurrentValue()+""""
```

Pokud již existuje filtr (například filtr, který se zabývá seznamem Listbox 2, ke kterému se nyní přistupuje), nový obsah se připojí k předchozímu obsahu uloženému ve skrytém ovládacím prvku.

```
If oForm.Filter <> ""
```

K tomu musí dojít pouze v případě, že stejné pole ještě nebylo filtrováno. Například pokud filtrujeme podle pole Year, opakování filtru nenajde žádné další záznamy pro pole seznamu Name. Osoba může být nalezena pouze v jednom roce. Musíme proto vyloučit možnost, že název filtru již byl použit.

```
And InStr(oForm.Filter, """"+Trim(aFilter(0))+""""='') = 0
Then
  stFilter = oForm.Filter + " AND " + stFilter
```

Pokud filtr existuje a pole, které bude použito pro filtrování, se již ve filtru nachází, je třeba předchozí filtrování na tomto poli smazat a vytvořit nový filtr.

```
ElseIf oForm.Filter <> "" Then
  stFilter = Left(oForm.Filter,
  InStr(oForm.Filter, """"+Trim(aFilter(0))+""""='')-1) +
stFilter
End If
End If
```

Poté se filtr zadá do formuláře. Tento filtr může být také prázdný, pokud bylo vybráno první pole seznamu a nemá žádný obsah.

```
oForm.Filter = stFilter
oForm.reload()
```

Stejný postup se spustí, pokud formulář není třeba filtrovat okamžitě. V tomto případě je hodnota filtru uložena v průměrném čase ve skrytém ovládacím prvku.

```
Else
  oFieldHidden = oForm.GetByName(Trim(aFilter(1)))
  If oField.GetCurrentValue <> "" Then
    stFilter = """"+Trim(aFilter(0))
+""""=''+oField.GetCurrentValue()+""""
    If oFieldHidden.HiddenValue <> ""
```



```

And InStr(oFieldHidden.HiddenValue, """"+Trim(aFilter(0))
+""""='') = 0 Then
    stFilter = oFieldHidden.HiddenValue + " AND " + stFilter
ElseIf oFieldHidden.HiddenValue <> "" Then
    stFilter = Left(oFieldHidden.HiddenValue,
        InStr(oFieldHidden.HiddenValue, """"+Trim(aFilter(0))
+""""='')-1) +
        stFilter
End If
End If
oFieldHidden.HiddenValue = stFilter
End If

```

Pokud má pole Další informace záznam s číslem 4 (číslování začíná od 0), musí být následující pole se seznamem nastaveno na odpovídající položku ze seznamu volajícího.

```

If UBound(aFilter()) > 1 Then
    oField1 = oForm.getByname(Trim(aFilter(2)))
    aFilter1() = Split(oField1.Tag, ",")

```

Potřebné údaje pro filtrování se načtou z pole Další informace (Tag) v příslušném seznamu. Není možné zapsat pouze čerstvý kód SQL do pole se seznamem a poté načíst hodnoty pole se seznamem. Místo toho je třeba hodnoty odpovídající dotazu zapsat přímo do pole seznamu.

Vytvoření kódu vychází z toho, že tabulka, na kterou se formulář odkazuje, je stejná jako tabulka, na kterou se odkazují seznamy. Takové pole se seznamem není určeno k přenosu cizích klíčů do tabulky.

```

If oField.getCurrentValue <> "" Then
    stSql = "SELECT DISTINCT """"+Trim(aFilter1(0))+"""" FROM
""""+oForm.Command+
"""" WHERE "+stFilter+" ORDER BY
""""+Trim(aFilter1(0))+""""
    oDatasource = ThisComponent.Parent.CurrentController
    If Not (oDatasource.isConnected()) Then
        oDatasource.connect()
    End If
    oConnection = oDatasource.ActiveConnection()
    oSQL_Statement = oConnection.createStatement()
    oQuery_result = oSQL_Statement.executeQuery(stSql)

```

Hodnoty se načtou do pole. Pole se přenáší přímo do pole seznamu. Příslušné indexy pole rostou v rámci smyčky.

```

inIndex = 0
While oQuery_result.next
    ReDim Preserve aContent(inIndex)
    acontent(inIndex) = oQuery_result.getString(1)
    inIndex = inIndex+1
WEnd

```

```

Else
    aContent(0) = ""
End If
oField1.StringItemList = aContent()

```

Obsah pole se seznamem byl vytvořen nově. Nyní je třeba ho znovu načíst. Poté se pomocí vlastnosti Další informace aktualizovaného pole seznamu vyprázdní každé následující závislé pole seznamu a spustí se smyčka pro všechny následující pole seznamu, dokud se nedosáhne takového, který nemá ve svých Dalšíších informacích čtvrtý výraz.

```

oField1.refresh()
While UBound(aFilter1()) > 1
    Dim aLeer()
    oField2 = oForm.GetByName(Trim(aFilter1(2)))
    Dim aFilter1()
    aFilter1() = Split(oField2.Tag, ",")
    oField2.StringItemList = aEmpty()
    oField2.refresh()
Wend
End If
End Sub

```

Viditelný obsah polí se seznamem je uložen v `oField1.StringItemList`. Pokud je třeba uložit nějakou další hodnotu pro přenos do podkladové tabulky jako cizí klíč, jak je obvyklé u polí se seznamem ve formulářích, musí být tato hodnota předána do dotazu samostatně a poté uložena pomocí `oField1.ValueItemList`.

Takové rozšíření vyžaduje další proměnné, jako je kromě tabulky, do které se mají ukládat hodnoty formuláře, také tabulka, ze které se vykresluje obsah pole seznamu.

Formulaci dotazu na filtr je třeba věnovat zvláštní pozornost.

```
stFilter = """"+Trim(aFilter(1))+""""=''+oField.GetCurrentValue()+""""
```

To bude fungovat pouze v případě, že základní verze LibreOffice je 4.1 nebo novější, protože jako `CurrentValue()` se zadává hodnota, která má být uložena, a ne hodnota, která se zobrazuje. Chceme-li zajistit, aby fungoval v různých verzích, nastavíme Vlastnost: Nastavte vlastnost Seznam > Data > Vázané pole > '0'.

Zadávání časů pomocí milisekund

Ukládání časů s přesností na milisekundy vyžaduje v tabulce pole časové značky, které je pro tento účel zvlášť upraveno v jazyce SQL (viz „Vytvoření tabulky“ v kapitole 3). Takové pole může být ve formuláři reprezentováno formátovaným polem ve formátu **MM:SS,00**. Při prvním pokusu o zápis do něj však záznam selže. To lze napravit pomocí následujícího makra, které by mělo být navázáno na vlastnost formuláře „Před záznamem“:

```

SUB Timestamp
    Dim unoStmp As New com.sun.star.util.DateTime
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oForm As Object

```

```

Dim oFeld As Object
Dim stZeit As String
Dim ar()
Dim arMandS()
Dim loNano As Long
Dim inSecond As Integer
Dim inMinute As Integer
oDoc = thisComponent
oDrawpage = oDoc.Drawpage
oForm = oDrawpage.Forms.getByName("MainForm")
oField = oForm.getByName("Time")
stTime = oField.Text

```

Proměnné se deklarují jako první. Zbytek kódu se provede pouze tehdy, když je v poli Time něco uvedeno. V opačném případě vnitřní mechanismus formuláře nastaví pole na **NULL**.

```

If stTime <> "" Then
    ar() = Split(stTime, ".")
    loNano = CLng(ar(1) & "0000000")
    arMandS() = Split(ar(0), ":")
    inSecond = CInt(arMandS(1))
    inMinute = CInt(arMandS(0))

```

Položka v poli Time je rozdělena na jednotlivé prvky.

Nejprve se oddělí desetinná část a doplní se nulovými znaky vpravo na celkem devět číslic. Takto vysoké číslo lze uložit pouze do dlouhé proměnné.

Zbytek času se pak rozdělí na minuty a sekundy pomocí dvojtečky jako oddělovače a převede se na celá čísla.

```

With unoStmp
    .NanoSeconds = loNano
    .Seconds = inSecond
    .Minutes = inMinute
    .Hours = 0
    .Day = 30
    .Month = 12
    .Year = 1899
End With

```

Hodnoty časových razítek jsou přiřazeny standardnímu datu LibreOffice 30.12.1899. Vedle něj lze samozřejmě uložit i aktuální datum.



Poznámka

Získání a uložení aktuálního data:

```

Dim now As Date
    now = Now()
    With unoStmp
        .NanoSeconds = loNano
        .Seconds = inSecond
        .Minutes = inMinute
        .Hours = Hour(now)
        .Day = Day(now)
        .Month = Month(now)
        .Year = Year(now)
    End With
    oField.BoundField.updateTimestamp(unoStmp)
End If
End Sub

```

Nyní se vytvořené časové razítko přeneso do pole pomocí **updateTimestamp** a uloží se do formuláře.

V dřívějších výukových materiálech se **NanoSekundy** nazývaly **HundrethSeconds**. To neodpovídá rozhraní API LibreOffice a způsobí to chybové hlášení.

Jedna událost – několik implementací

Při použití formulářů se může stát, že makro spojené s jednou událostí bude spuštěno dvakrát. K tomu dochází proto, že s uložením například upraveného záznamu je spojeno více procesů současně. Různé příčiny takové události lze určit následujícím způsobem:

```

Sub Determine_eventcause(oEvent As Object)
    Dim oForm As Object
    oForm = oEvent.Source
    MsgBox oForm.ImplementationName
End Sub

```

Při ukládání upraveného záznamu se používají dvě implementace s názvy **org.openoffice.comp.svx.FormController** a **com.sun.star.comp.forms.ODatabaseForm**. Pomocí těchto názvů můžeme zajistit, aby makro prošlo svým kódem pouze jednou. Duplicitní spuštění obvykle způsobí jen (malou) přestávku ve vykonávání programu, ale může vést k tomu, že se kurzor vrátí o dva záznamy místo jednoho. Každá implementace umožňuje pouze určité příkazy, takže znalost názvu implementace může být důležitá.

Uložení s potvrzením

U složitých změn záznamů má smysl se před provedením zeptat uživatele, zda má být změna skutečně provedena. Pokud je odpověď v dialogovém okně Ne, uložení se přeruší, změna se zahodí a kurzor zůstane na aktuálním záznamu.

```

Sub Save_confirmation(oEvent As Object)
    Dim oFormFeature As Object
    Dim oFormOperations As Object
    Dim inAnswer As Integer
    oFormFeature = com.sun.star.form.runtime.FormFeature
    Select Case oEvent.Source.ImplementationName

```

```

Case "org.openoffice.comp.svx.FormController"
    inAnswer = MsgBox("Should the record be changed?" ,4,
"Change_record")
    Select Case inAnswer
        Case 6 ' Yes, no further action
        Case 7 ' No, interrupt save
            oFormOperations = oEvent.Source.FormOperations
            oFormOperations.execute(oFormFeature.UndoRecordChanges)
        Case Else
    End Select
Case "com.sun.star.comp.forms.ODatabaseForm"
End Select
End Sub

```

Existují dva spouštěcí momenty s různými názvy implementace. Tyto dvě implementace jsou rozlišeny v příkazu **SELECT CASE**. Kód se provede pouze pro implementaci **FormController**. Je to proto, že pouze **FormController** má proměnnou **FormOperations**.

Kromě Ano a Ne může uživatel také klepnout na tlačítko zavřít. Tím však získáme stejnou hodnotu jako u Ne, tedy 7.

Pokud se ve formuláři pohybujeme pomocí klávesy tabelátoru, zobrazí se uživateli pouze dialogové okno s výzvou k potvrzení. Uživatelům, kteří používají lištu navigace, se však také zobrazí zpráva, že záznam nebude změněn.

Primární klíč z běžného čísla a roku

Při přípravě faktur jsou ovlivněny roční zůstatky. To často vede k touze oddělit tabulky faktur v databázi podle jednotlivých let a každý rok začít novou tabulku.

Následující řešení pomocí makra používá jinou metodu. Do tabulky se automaticky zapíše hodnota pole ID, ale také se zohlední pole Year, které v tabulce existuje jako sekundární primární klíč. V tabulce se tedy mohou vyskytovat následující primární klíče:

Tabulka 49: Příklad primárních klíčů kombinujících hodnoty rok a ID

year	ID
2014	1
2014	2
2014	3
2015	1
2015	2

Tímto způsobem lze snáze získat přehled o dokumentech za daný rok.

```

Sub Current_Date_and_ID
    Dim oDatasource As Object
    Dim oConnection As Object
    Dim oSQL_Command As Object
    Dim stSql As String
    Dim oResult As Object

```

```

Dim oDoc As Object
Dim oDrawpage As Object
Dim oForm As Object
Dim oField1 As Object
Dim oField2 As Object
Dim oField3 As Object
Dim inIDnew As Integer
Dim inYear As Integer
Dim unoDate
oDoc = thisComponent
oDrawpage = oDoc.drawpage
oForm = oDrawpage.forms.getByName("MainForm")
oField1 = oForm.getByName("fmt_year")
oField2 = oForm.getByName("fmtID")
oField3 = oForm.getByName("dat_date")
If IsEmpty(oField2.getCurrentValue()) Then
    If IsEmpty(oField3.getCurrentValue()) Then
        unoDate = createUnoStruct("com.sun.star.util.Date")
        unoDate.Year = Year(Date)
        unoDate.Month = Month(Date)
        unoDate.Day = Day(Date)
        inYear = Year(Date)
    Else
        inYear = oField3.CurrentValue.Year
    End If
oDatasource = ThisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
stSql = "SELECT MAX( ""ID"" )+1 FROM ""orders"" WHERE ""year"" = '"
    + inYear + "'"
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inIDnew = oresult.getInt(1)
Wend
If inIDnew = 0 Then

```

```

        inIDnew = 1
    End If
    oField1.BoundField.updateInt(inYear)
    oField2.BoundField.updateInt(inIDnew)
    If IsEmpty(oField3.GetCurrentValue()) Then
        oField3.BoundField.updateDate(unoDate)
    End If
End If
End Sub

```

Všechny proměnné jsou deklarovány. Jsou zpřístupněny ovládací prvky formuláře v hlavním formuláři. Zbytek kódu se spustí pouze v případě, že položka pole fmtID je stále prázdná. Pokud nebylo zadáno žádné datum, vytvoří se struktura data, aby se aktuální datum a rok přenesly do příslušných polí. Poté se vytvoří připojení k databázi, pokud ještě neexistuje. Nejvyšší hodnota pole ID pro aktuální rok se zvýší o 1. Pokud je sada výsledků prázdná, znamená to, že v poli ID nejsou žádné položky. V tomto okamžiku by mohla být v ovládacím prvku fmtID zadána hodnota 0, ale číslování příkazů by mělo začínat hodnotou 1, takže proměnná inIDnew získá hodnotu 1.

Do formuláře se přeneše vrácená hodnota year, ID a aktuální datum (pokud nebylo zadáno žádné datum).

Ve formuláři jsou pole primárních klíčů ID a Year chráněna proti zápisu. Proto jim lze zadat hodnoty pouze pomocí tohoto makra.

Úlohy databáze rozšířené pomocí maker

Vytvoření připojení k databázi

```

oDataSource = ThisComponent.Parent.DataSource
If Not oDataSource.IsPasswordRequired Then
    oConnection = oDataSource.GetConnection("", "")

```

Zde by bylo možné zadat uživatelské jméno a heslo, pokud by to bylo nutné. V takovém případě by závorky obsahovaly ("Username", "Password"). Namísto zadání uživatelského jména a hesla v otevřeném textu se vyvolá dialogové okno pro ochranu heslem:

```

Else
    oAuthentication =
createUnoService("com.sun.star.sdb.InteractionHandler")
    oConnection = oDataSource.ConnectWithCompletion(oAuthentication)
End If

```

Pokud však k databázi přistupuje formulář v rámci stejného souboru aplikace Base, stačí:

```

oDataSource = ThisComponent.Parent.CurrentController
If Not (oDataSource.isConnected()) Then
    oDataSource.connect()
End If
oConnection = oDataSource.ActiveConnection()

```

Zde je databáze známá, takže uživatelské jméno a heslo nejsou nutné, protože jsou již vypnuty v základní konfiguraci HSQLDB pro interní verzi.

V případě formulářů mimo aplikaci Base se připojení provádí prostřednictvím prvního formuláře:

```
oDataSource = ThisComponent.Drawpage.Forms(0)
oConnection = oDataSource.activeConnection
```

Kopírování dat z jedné databáze do druhé

Interní databáze je databáze pro jednoho uživatele. Záznamy jsou uloženy v souboru *.odb. Výměna dat mezi různými databázovými soubory nebyla povolena, nicméně je možná pomocí exportu a importu.

Soubory *.odb jsou však často nastaveny tak, aby umožňovaly automatickou výměnu dat mezi databázemi. Zde může být užitečný následující postup.

Po deklaraci proměnných se z tlačítka na formuláři načte cesta k aktuální databázi. Název databáze je oddělen od zbytku cesty. V této složce se nachází také cílový soubor pro záznamy. Název tohoto souboru je připojen k cestě, aby bylo možné navázat spojení s cílovou databází.

Připojení ke zdrojové databázi je určeno vzhledem k formuláři, který obsahuje tlačítko: **ThisComponent.Parent.CurrentController**. Připojení k externí databázi se nastavuje pomocí **DatabaseContext** a cesty.

```
Sub DataCopy
    Dim oDatabaseContext As Object
    Dim oDataSource As Object
    Dim oDataSourceZiel As Object
    Dim oConnection As Object
    Dim oConnectionZiel As Object
    Dim oDB As Object
    Dim oSQL_Command As Object
    Dim oSQL_CommandTarget As Object
    Dim oResult As Object
    Dim oResultTarget As Object
    Dim stSql As String
    Dim stSqlTarget As String
    Dim inID As Integer
    Dim inIDTarget As Integer
    Dim stName As String
    Dim stTown As String
    oDB = ThisComponent.Parent
    stDir = Left(oDB.Location, Len(oDB.Location) - Len(oDB.Title))
    stDir = ConvertToUrl(stDir & "TargetDB.odb")
    oDataSource = ThisComponent.Parent.CurrentController
    If Not (oDataSource.isConnected()) Then
        oDataSource.connect()
    End If
    oConnection = oDataSource.ActiveConnection()
```



```

oDatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
oDatasourceTarget = oDatabaseContext.getByname(stDir)
oConnectionTarget = oDatasourceTarget.GetConnection("", "")
oSQL_Command = oConnection.createStatement()
stSql = "SELECT * FROM ""table""
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    inID = oResult.getInt(1)
    stName = oResult.getString(2)
    stTown = oResult.getString(3)
    oSQL_CommandTarget = oConnectionTarget.createStatement()
    stSqlTarget = "SELECT ""ID"" FROM ""table"" WHERE ""ID"" = '"+inID+'"'
    oResultTarget = oSQL_CommandZiel.executeQuery(stSqlTarget)
    inIDZiel = - 1
    While oResultTarget.next
        inIDTarget = oResultTarget.getInt(1)
    Wend
    If inIDTarget = - 1 Then
        stSqlTarget = "INSERT INTO ""table"" (""ID"", ""name"", ""town"")
VALUES
    ('"+inID+', '"+stName+', '"+stTown+')"'
        oSQL_CommandTarget.executeUpdate(stSqlZiel)
    End If
Wend
End Sub

```

Kompletní tabulky zdrojové databáze jsou načteny a vloženy řádek po řádku do tabulek cílové databáze pomocí nastaveného připojení. Před vložením se zkontroluje, zda byla nastavena hodnota primárního klíče. Pokud ano, záznam se nekopíruje.

Lze také zařídit, aby se místo nového záznamu zkopíroval stávající záznam. Ve všech případech se tím zajistí, aby cílová databáze obsahovala záznamy se správným primárním klíčem zdrojové databáze.

Přístup k dotazům

Vytvářet dotazy v grafickém uživatelském rozhraní je jednodušší než přenášet jejich text do maker, navíc s komplikací, že pro všechny názvy tabulek a polí jsou nutné duplicitní dvojité uvozovky.

```

Sub aQueryContent
    Dim oDatabaseFile As Object
    Dim oQuery As Object
    Dim stQuery As String
    oDatabaseFile = ThisComponent.Parent.CurrentController.DataSource

```

```
oQuery = oDatabaseFile.getQueryDefinitions()
stQuery = oQuery.getByName("Query").Command
MsgBox stQuery
```

End Sub

Zde je obsah souboru *.odb zpřístupněn z formuláře. K dotazu se dostanete pomocí **getQueryDefinitions()**. Kód SQL pro dotaz je v jeho poli **Command**. Tento příkaz pak lze dále použít v makru.

Při použití kódu SQL dotazu je třeba dbát na to, aby kód neodkazoval na jiný dotaz. To nevyhnutelně vede ke zprávě, že (zdánlivá) tabulka z databáze je neznámá. Z tohoto důvodu je jednodušší vytvářet pohledy z dotazů a poté k nim přistupovat v makru.

Zabezpečení databáze

Někdy se může stát, zejména při vytváření databáze, že se soubor ODB neočekávaně zkrátí. Časté ukládání po úpravách je proto užitečné, zejména při používání modulu Sestavy.

Pokud je databáze používána, může být poškozena selháním operačního systému, pokud k němu dojde právě při ukončování souboru aplikace Base. V tomto okamžiku se obsah databáze zapisuje do souboru.

Kromě toho existují obvyklé příčiny, proč se soubory náhle odmítají otevřít, například selhání pevného disku. Není proto na škodu mít záložní kopii, která je co nejaktuálnější. Stav dat se nemění, dokud je soubor ODB otevřený. Z tohoto důvodu mohou být bezpečnostní podprogramy přímo spojeny s otevřením souboru. Soubor jednoduše zkopírujeme pomocí cesty *Zálohy* uvedené v **Nástroje > Možnosti > LibreOffice > Cesty**. Makro začne přepisovat nejstarší verzi po určitém počtu kopií (**inMax**).

```
Sub Databasebackup(inMax As Integer)
```

```
Dim oPath As Object
Dim oDoc As Object
Dim sTitle As String
Dim sUrl_End As String
Dim sUrl_Start As String
Dim i As Integer
Dim k As Integer
oDoc = ThisComponent
sTitle = oDoc.Title
sUrl_Start = oDoc.URL
Do While sUrl_Start = ""
    oDoc = oDoc.Parent
    sTitle = oDoc.Title
    sUrl_Start = oDoc.URL
Loop
```

Pokud je makro spuštěno při spuštění souboru ODB, budou hodnoty sTitle a sUrl_Start správné. Pokud však makro provádí formulář, musí nejprve zjistit, zda je k dispozici adresa URL. Pokud je adresa URL prázdná, vyhledá se hodnota na vyšší úrovni (oDoc.Parent).

```
oPath = createUnoService("com.sun.star.util.PathSettings")
```

```

For i = 1 To inMax + 1
    If Not FileExists(oPath.Backup & "/" & i & "_" & sTitle) Then
        If i > inMax Then
            For k = 1 To inMax - 1 To 1 Step -1
                If FileDateTime(oPath.Backup & "/" & k & "_" & sTitle) <=
FileDateTime(oPath.Backup & "/" & k+1 & "_" & sTitle) Then
                    If k = 1 Then
                        i = k
                        Exit For
                    End If
                Else
                    i = k + 1
                    Exit For
                End If
            Next
        End If
        Exit For
    End If
Next
sUrl_End = oPath.Backup & "/" & i & "_" & sTitle
FileCopy(sUrl_Start, sUrl_End)
End Sub

```

Zálohování můžeme provést i za běhu databáze Base, pokud je možné data z mezipaměti zapsat zpět do souboru před provedením podprogramu Databasebackup. To by mohlo být užitečné třeba po uplynutí určitého času nebo po stisknutí tlačítka na obrazovce. Toto vyčištění mezipaměti se provádí pomocí následujícího podprogramu:

```

Sub Write_data_out_of_cache
    Dim oData As Object
    Dim oDataSource As Object
    oData = ThisDatabaseDocument.CurrentController
    If Not ( oData.isConnected() ) Then oData.connect()
    oDataSource = oData.DataSource
    oDataSource.flush
End Sub

```

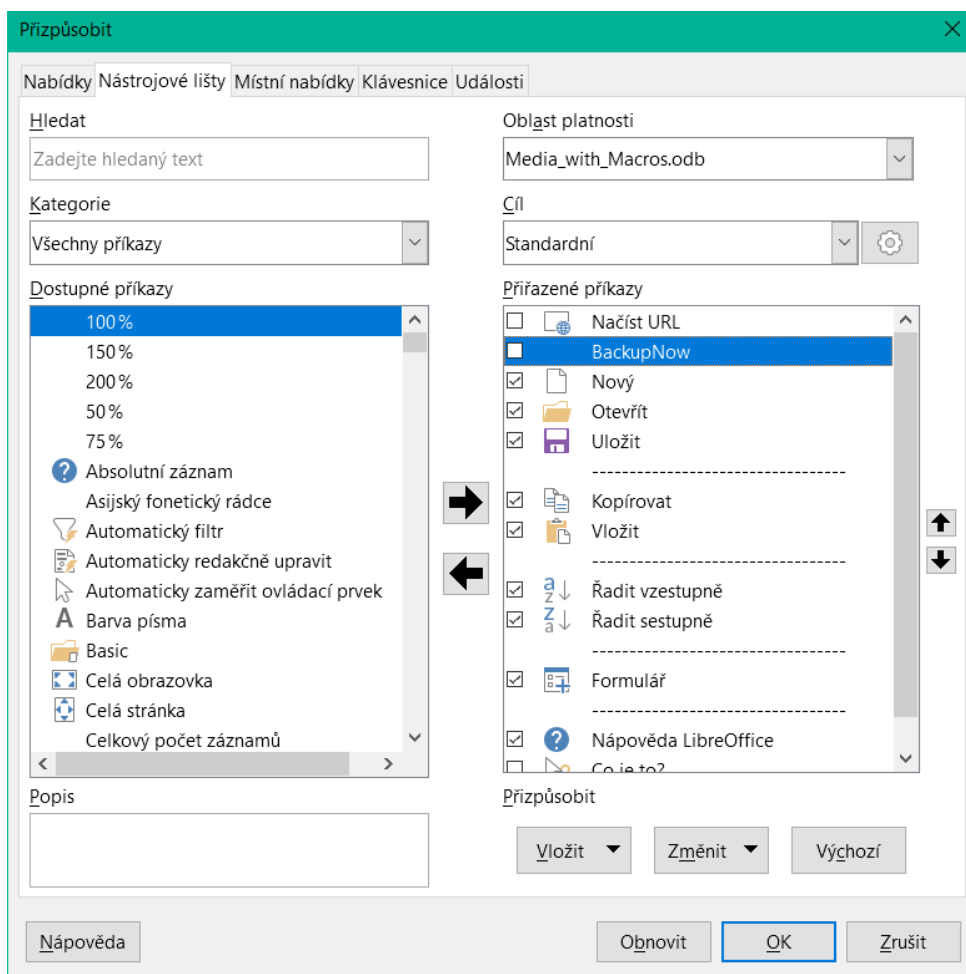
Má-li být vše spuštěno z jediného tlačítka na formuláři, musí být obě procedury volány další procedurou:

```

Sub BackupNow
    Write_data_out_of_cache
    DatabaseBackup(10)
End Sub

```

Zejména v případě bezpečnostního makra by bylo vhodné, aby bylo makro přístupné prostřednictvím nástrojové lišty databáze. To se provádí v hlavním okně souboru Base pomocí **Nástroje > Přizpůsobit > Nástrojové lišty**.



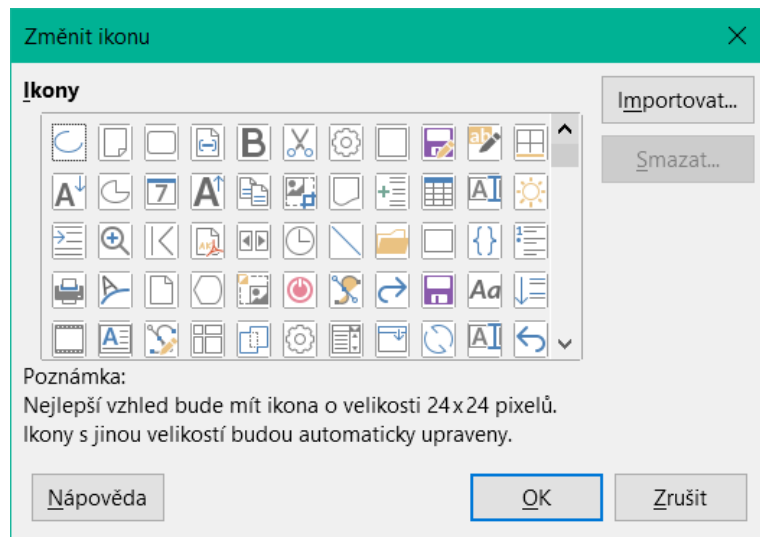
Obrázek 432: Použití karty **Nástrojové lišty** v dialogovém okně **Přizpůsobit** pro zpřístupnění makra na standardní nástrojové liště

V dialogovém okně **Přizpůsobit** v části **Rozsah platnosti** musí být příkaz uložen v souboru Base, což je v tomto případě `Media_with_Macros.odt`.

V části **Cíl** vybereme **Standardní** nástrojová lišta, který funguje ve všech částech aplikace Base.

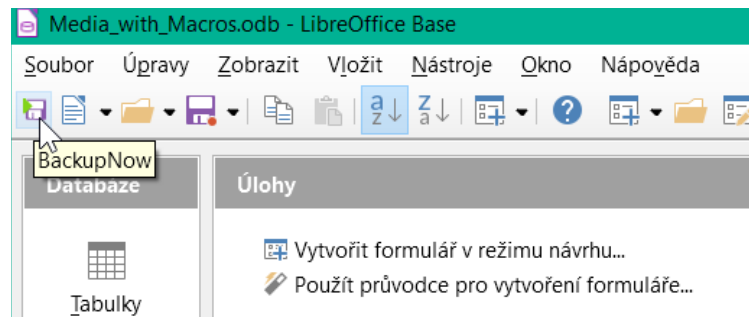
Dialogové okno nyní zobrazuje příslušné funkce v pravém seznamu. Vybereme proceduru **BackupNow**.

Příkaz je nyní k dispozici pro použití na nástrojové liště nejvyšší úrovně. Chceme-li příkazu přiřadit ikonu, zvolíme **Upravit > Změnit ikonu** a otevřeme dialogové okno zobrazené na obrázku 433.



Obrázek 433: Dialogové okno Změnit ikonu

Vybereme vhodnou ikonu. Můžeme si také vytvořit a přidat vlastní ikonu.



Obrázek 434: Nová ikona viditelná na Standardní nástrojové liště

Místo názvu procedury se nyní zobrazí ikona. Název se stane nápovědou.

Proceduru provedeme klepnutím na ikonu na nástrojové liště.

Komprimace databáze

Jedná se o jednoduchý příkaz SQL (**SHUTDOWN COMPACT**), který by se měl čas od času provést, zejména po odstranění velkého množství dat. Databáze ukládá nová data, ale stále si vyhrazuje místo pro smazaná data. V případech, kdy byla data podstatně změněna, je proto nutné databázi zkomprimovat.

✓ Poznámka

Od verze LibreOffice 3.6 se tento příkaz automaticky provede pro interní HSQLDB při uzavření databáze. Proto již toto makro není pro interní databázi nutné.

Po zkomprimování již nejsou tabulky přístupné. Soubor je třeba znovu otevřít. Proto toto makro uzavře formulář, ze kterého je voláno. Samotný dokument však nelze zavřít, aniž by došlo k obnovení při jeho opětovném otevření. Proto je tato funkce zakomentována.

Sub Database_compaction

```
Dim stMessage As String
oDataSource = ThisComponent.Parent.CurrentController ' Accessible
from the form
```

```

If Not (oDataSource.isConnected()) Then
    oDataSource.connect()
End If
oConnection = oDataSource.ActiveConnection()
oSQL_Statement = oConnection.createStatement()
stSql = "SHUTDOWN COMPACT" ' The database is being compacted and shut
down
oSQL_Statement.executeQuery(stSql)
stMessage = "The database is being compacted." + Chr(13) + "The form will
now close."
stMessage = stMessage + Chr(13) + "Following this, the database file
should be closed."
stMessage = stMessage + Chr(13) + "The database can only be accessed
after reopening the database file."
MsgBox stMessage
ThisDatabaseDocument.FormDocuments.getByName( "Maintenance" ).close
REM Uzavření databázového souboru způsobí při jeho opětovném otevření
operaci obnovení.
' ThisDatabaseDocument.close(True)
End Sub

```

Snížení indexu tabulky pro pole automatických hodnot

Pokud je z tabulky odstraněno velké množství dat, uživatelé se často obávají, že posloupnost automaticky generovaných primárních klíčů jednoduše pokračuje směrem nahoru, místo aby začala znovu od nejvyšší aktuální hodnoty klíče. Následující podprogram načte aktuálně nejvyšší hodnotu pole ID v tabulce a nastaví další počáteční hodnotu klíče o 1 vyšší, než je tato maximální hodnota.

Pokud se pole primárního klíče nejmenuje ID, je třeba makro odpovídajícím způsobem upravit.

```

Sub Table_index_down(stTable As String)
    REM Tento podprogram nastaví pole primárního klíče s automatickým
přírůstkem s přednastaveným názvem "ID" na nejnižší možnou hodnotu.
    Dim inCount As Integer
    Dim inSequence_Value As Integer
    oDataSource = ThisComponent.Parent.CurrentController ' Accessible
through the form
    If Not (oDataSource.isConnected()) Then
        oDataSource.connect()
    End If
    oConnection = oDataSource.ActiveConnection()
    oSQL_Statement = oConnection.createStatement()
    stSql = "SELECT MAX(""ID"") FROM ""+stTable+"" ' The highest value
in "ID" is determined
    oQuery_result = oSQL_Statement.executeQuery(stSql) ' Dotaz je
spuštěn a návratová hodnota je uložena v proměnné oQuery_result

```

```

If Not IsNull(oQuery_result) Then
    While oQuery_result.next
        inCount = oQuery_result.getInt(1) ' First data field is read
    Wend ' další záznam, v tomto případně žádný, protože existuje pouze
jeden záznam
        If inCount = "" Then ' Pokud nejvyšší hodnota není hodnota, což
znamená, že tabulka je prázdná, je nejvyšší hodnota nastavena na -1.
            inCount = -1
        End If
        inSequence_Value = inCount+1 ' Nejvyšší hodnota se zvýší o 1
REM Pro databázi je připraven nový příkaz. ID začne znovu od inCount+1.
REM Tento příkaz nemá žádnou návratovou hodnotu, protože se nečte žádný
záznam
        oSQL_statement = oConnection.createStatement()
        oSQL_statement.executeQuery("ALTER TABLE "" + stTable + "" ALTER
COLUMN ""ID"" RESTART WITH " + inSequence_Value + "")
    End If
End Sub

```

Tisk z Base

Standardním způsobem, jak získat dokument k tisku ze systému Base, je použití sestavy. Tabulky a dotazy lze také zkopírovat do aplikace Calc a připravit je tam k tisku. Samozřejmě je možný i přímý tisk formuláře z obrazovky.

Tisk sestavy z interního formuláře

Generování sestav se obvykle provádí z uživatelského rozhraní aplikace Base. Klepnutím na název sestavy se spustí její příprava. Bylo by samozřejmě jednodušší, kdyby bylo možné sestavu spustit přímo z formuláře.

```

Sub ReportLaunch
    ThisDatabaseDocument.ReportDocuments.getByname("Report").open
End Sub

```

Všechny sestavy jsou přístupné podle názvu ze svého kontejneru **ReportDocuments**. Otvírají se pomocí **open**. Pokud je sestava vázána na dotaz, který je filtrován prostřednictvím formuláře, umožňuje tato metoda vypsát aktuální záznam.

Spuštění, formátování, přímý tisk a uzavření sestavy

Ještě lepší by bylo, kdyby bylo možné sestavu odeslat přímo do tiskárny. Následující kombinace postupů přidává několik drobností. Nejprve vybere aktivní záznam ve formuláři, přeformátuje sestavu tak, aby se textová pole automaticky nastavila na správnou výšku, a poté sestavu spustí. Nakonec se sestava vytiskne a volitelně uloží ve formátu pdf. To vše se děje téměř zcela na pozadí, protože sestava se po otevření formuláře přepne do neviditelného režimu a po vytištění se opět zavře. Návrhy na různé postupy předložili Andrew Pitonyak, Thomas Krumbein a Lionel Elie Mamane.

```

Sub ReportStart(oEvent As Object)
    Dim oForm As Object

```

```

Dim stSql As String
Dim oDatasource As Object
Dim oConnection As Object
Dim oSQL_command As Object
Dim oReport As Object
Dim oReportView As Object
oForm = oEvent.Source.model.parent
stSql = "UPDATE ""Filter"" SET ""Integer"" = '" +
    oForm.getInt(oForm.findColumn("ID")) + "' WHERE ""ID"" = TRUE"
oDatasource = ThisComponent.Parent.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_command = oConnection.createStatement()
oSQL_command.executeUpdate(stSql)
oReport =
ThisDatabaseDocument.ReportDocuments.getByName("Reportname").open
oReportView = oReport.CurrentController.Frame.ContainerWindow
oReportView.Visible = False
ReportLineHeightAuto(oReport)
End Sub

```

Procedura ReportStart je propojena s tlačítkem ve formuláři. Pomocí tohoto tlačítka lze načíst primární klíč aktuálního záznamu. Z události, která makro spustí, se dostaneme do formuláře (**oForm**). Název pole primárního klíče je zde uveden jako **"ID"**. Pomocí **oForm.getInt(oForm.findColumn("ID"))** se klíč načte z pole jako celé číslo. Tato hodnota je uložena v tabulce filtrů. Tabulka filtru řídí dotaz, aby bylo zajištěno, že pro sestavu bude použit pouze aktuální záznam.

Sestavu lze otevřít bez odkazu na formulář. Pak je přístupný jako objekt (**oReport**). Okno sestavy je neviditelné. Všimneme si, že při vyvolání nemůže být neviditelný, takže se zobrazí jen na chvíli a pak se na pozadí vyplní příslušným obsahem.

Dále se spustí procedura ReportLineHeightAuto. Této proceduře je jako argument předán odkaz na otevřenou sestavu.

Výšku řádku záznamu lze nastavit automaticky při tisku. Pokud je v určitém poli pravděpodobně příliš mnoho textu, text se zkrátí a zbytek se označí červeným trojúhelníkem. Pokud tato funkce nefunguje, následující postup zajistí, že ve všech tabulkách s názvem Detail bude zapnuta automatická kontrola výšky.

```

Sub ReportLineHeightAuto(oReport As Object)
    Dim oTables As Object
    Dim oTable As Object
    Dim inT As Integer
    Dim inI As Integer

```



```

Dim oRows As Object
Dim oRow As Object
oTables = oReport.getTextTables()
For inT = 0 TO oTables.count() - 1
    oTable = oTables.getByIndex(inT)
    If Left$(oTable.name, 6) = "Detail" Then
        oRows = oTable.Rows
        For inI = 0 To oRows.count - 1
            oRow = oRows.getByIndex(inI)
            oRow.IsAutoHeight = True
        Next inI
    End If
Next inT
PrintCloseReport(oReport)

```

End Sub

Při vytváření sestavy je třeba dbát na to, aby všechna pole na stejném řádku sekce Detail měla stejnou výšku. V opačném případě může automatická kontrola výšky náhle nastavit dvojnásobnou výšku řádku.

Jakmile je u všech tabulek s názvem Detail nastavena automatická kontrola výšky, je sestava odeslána na tiskárnu pomocí procedury PrintCloseReport.

Pole Props obsahuje hodnoty, které jsou v dokumentu přiřazeny k tiskárně. Pro příkaz tisk je důležitý název výchozí tiskárny. Sestava by měla zůstat otevřená až do skutečného dokončení tisku. To je zajištěno zadáním názvu tiskárny a příkazu „Počkej, až budu hotový“ (**Wait**) jako argumentů.

```

Sub PrintCloseReport(oReport As Object)
    Dim Props
    Dim stPrinter As String
    Props = oReport.getPrinter()
    stPrinter = Props(0).value
    Dim arg(1) As New com.sun.star.beans.PropertyValue
    arg(0).name = "Name"
    arg(0).value = "<" & stPrinter & ">"
    arg(1).name = "Wait"
    arg(1).value = True
    oReport.print(arg())
    oReport.close(true)

```

End Sub

Teprve po úplném odeslání tisku do tiskárny se dokument uzavře.

Nastavení tiskárny nalezneme v části Nastavení tiskárny a tisku na wiki.

Pokud chceme místo tisku (nebo jako doplněk k tisku) uložit dokument ve formátu pdf jako bezpečnostní kopii, můžeme použít metodu **storeToURL()**:

```
Sub ReportPDFstore(oReport As Object)
    Dim stUrl As String
    Dim arg(0) As New com.sun.star.beans.PropertyValue
    arg(0).name = "FilterName"
    arg(0).value = "writer_pdf_Export"
    stUrl = "file:///...."
    oReport.storeToURL(stUrl, arg())
End Sub
```

Adresa URL musí být samozřejmě úplná adresa URL. Ještě lépe by tato adresa měla být spojena s trvalým záznamem vytištěného dokumentu, například s číslem faktury. V opačném případě by se mohlo stát, že bezpečnostní soubor bude při dalším tisku jednoduše přepsán.

Tisk sestav z externího formuláře

Při použití externích formulářů dochází k problémům. Sestavy se nacházejí v souboru *.odb a nejsou dostupné pomocí prohlížeče datových zdrojů.

```
Sub Reportstart(oEvent As Object)
    Dim oField As Object
    Dim oForm As Object
    Dim oDocument As Object
    Dim oDocView As Object
    Dim Arg()
    oField = oEvent.Source.Model
    oForm = oField.Parent
    sURL = oForm.DataSourceName
    oDocument = StarDesktop.loadComponentFromURL(sURL, "_blank", 0, Arg()
)
    oDocView = oDocument.CurrentController.Frame.ContainerWindow
    oDocView.Visible = False
    oDocument.getCurrentController().connect
    Wait(100)
    oDocument.ReportDocuments.getByname("Report").open
    oDocument.close(True)
End Sub
```

Sestava se spouští z tlačítka na externím formuláři. Tlačítko sdělí formuláři cestu k souboru *.odb: **oForm.DataSourceName**. Poté se soubor otevře pomocí **loadComponentFromURL**. Soubor by měl zůstat na pozadí, takže se zpřístupní zobrazení dokumentu a rozhraní se nastaví na **Visible = False**. Ideální by bylo použít přímo seznam argumentů **Arg()**, ale testy ukázaly, že to nedává správný výsledek.

Sestavu nelze z otevřeného dokumentu okamžitě vyvolat, protože připojení ještě není připraveno. Sestava se zobrazí s šedým pozadím a poté LibreOffice spadne. Krátké čekání

o 100 milisekundách tento problém řeší. Pro stanovení minimální čekací doby jsou nezbytné praktické zkoušky. Nyní je sestava spuštěna. Protože zpráva bude v samostatném textovém souboru, otevřený soubor *.odb lze opět zavřít. Metoda **oDocument.Close(True)** předá tento pokyn souboru *.odb. Soubor se uzavře pouze tehdy, když již není aktivní, tj. do sestavy již nemají být předávány žádné další záznamy.

Podobný přístup lze spustit z formulářů v souboru *.odb, ale v tomto případě by dokument neměl být uzavřen.

Pomocí maker v kombinaci s funkcí hromadné korespondence nebo textových polí můžeme získat kvalitní výtisky podstatně rychleji než pomocí nástroje Návrhář sestav.

Provedení hromadné korespondence z Base

Někdy je sestava pro vytvoření kvalitních dopisů adresátům prostě nedostatečná. Textová pole v sestavě mají v praxi velmi omezený význam. Místo toho lze v aplikaci Writer vytvořit dopis typu hromadné korespondence. Není však nutné nejprve otevřít Writer, provést v něm veškeré zadávání a přizpůsobení a teprve poté tisknout. To vše můžeme provést přímo z aplikace Base pomocí makra.

```
Sub MailmergePrint
```

```
    Dim oMailMerge As Object
```

```
    Dim aProps()
```

```
    oMailMerge = CreateUnoService("com.sun.star.text.MailMerge")
```

Název uvedený pro zdroj dat je název, pod kterým je databáze zaregistrována v LibreOffice. Tento název nemusí být totožný s názvem souboru. Registrovaný název v tomto příkladu je Addresses.

```
oMailMerge.DataSourceName = "Addresses"
```

Cesta k souboru hromadné korespondence musí být naformátována podle zvyklostí našeho operačního systému, v tomto příkladu je to absolutní cesta v systému Linux.

```
oMailMerge.DocumentURL =  
ConvertToUrl("home/user/Dokuments/mailmerge.odt")
```

Je stanoven typ příkazu. 0 znamená tabulku, 1 dotaz a 2 přímý příkaz SQL.

```
oMailMerge.CommandType = 1
```

Zde byl zvolen dotaz s názvem MailmergeQuery.

```
oMailMerge.Command = "MailmergeQuery"
```

Filtr slouží k určení záznamů, které mají být použity pro tisk hromadné korespondence. Tento filtr může být například zadán pomocí ovládacího prvku formuláře a předán z aplikace Base do makra. Použití primárního klíče záznamu by mohlo způsobit vytištění jediného dokumentu.

V tomto příkladu je vybráno pole Gender v dotazu MailmergeQuery a poté jsou vyhledány záznamy, které mají v tomto poli hodnotu 'm'.

```
oMailMerge.Filter = ""'"Gender"'='m'""
```

Dostupné typy výstupu jsou Tiskárna (1), Soubor (2) a E-mail (3). Zde je pro testovací účely vybrán výstupní soubor. Tento soubor je uložen v zadané cestě. Pro každý záznam hromadné korespondence bude proveden jeden tisk. Pro odlišení tohoto tisku je pole příjmení začleněno do názvu souboru.

```
oMailMerge.OutputType = 2  
oMailMerge.OutputUrl = ConvertToUrl("home/user/Documents")  
oMailMerge.FileNameFromColumn = True
```

```
oMailMerge.FileNamePrefix = "Surname"  
oMailMerge.execute(aProps())
```

End Sub

Pokud jsou filtru poskytnuta data prostřednictvím formuláře, umožňuje to provádět hromadnou korespondenci bez otevření Writeru.

Tisk prostřednictvím textových polí

Pomocí **Vložit > Pole > Další pole > Funkce > Zástupný znak** lze ve Writeru vytvořit vzor dokumentu, který má být v budoucnu vytištěn. Zástupné znaky by měly mít stejné názvy jako pole v databázové tabulce nebo dotazu, na kterém je založen formulář, z něhož je makro voláno.

V jednoduchém případě je typem zástupného symbolu Text.

V makru musí být uvedena cesta k modelu. Vytvoří se nový dokument Unknown1.odt. Makro vyplní zástupné symboly obsahem aktuálního záznamu z dotazu. Otevřený dokument pak můžeme podle potřeby upravovat.

Příklad databáze Example_database_mailmerge_direct.odt ukazuje, jak lze vytvořit kompletní fakturu pomocí textových polí a přístupu k připravené tabulce v rámci vzorového dokumentu. Na rozdíl od faktur vytvořených pomocí nástroje Návrhář sestav nemá tento typ vytváření faktur výškové omezení pro pole z tabulky. Zobrazí se veškerý text.

Zde je část kódu, kterou dodal především DPunch:

<http://de.openoffice.info/viewtopic.php?f=8&t=45868#p194799>

```
Sub Filling_Textfields
```

```
oForm = thisComponent.Drawpage.Forms.MainForm  
If oForm.RowCount = 0 Then  
    MsgBox "No available record for printing"  
    Exit Sub  
End If
```

Hlavní formulář je aktivován. Tlačítko, kterým se makro spustí, by mohlo sloužit také k vyhledání formuláře. Makro pak zjistí, že formulář skutečně obsahuje tisknutelná data.

```
oColumns = oForm.Columns  
oDB = ThisComponent.Parent
```

Přímý přístup na adresu URL z formuláře není možný. Musí být provedeno pomocí odkazu na databázi vyšší úrovně.

```
stDir = Left(oDB.Location, Len(oDB.Location) - Len(oDB.Title))
```

Název databáze je oddělen od adresy URL.

```
stDir = stDir & "Beispiel_Textfelder.ott"
```

Model je nalezen a otevřen.

```
Dim args(0) As New com.sun.star.beans.PropertyValue  
args(0).Name = "AsTemplate"  
args(0).Value = True  
oNewDoc = StarDesktop.loadComponentFromURL(stDir, "_blank", 0, args)
```

Textová pole jsou zapsána.

```
oTextfields = oNewDoc.Textfields.createEnumeration
```

```

Do While oTextfields.hasMoreElements
    oTextField = oTextfields.nextElement
    If oTextField.supportsService("com.sun.star.text.TextField.JumpEdit")
Then
        stColumnName = oTextField.PlaceHolder

```

Zástupný symbol představuje textové pole.

```

If oColumns.hasByName(stColumnName) Then

```

Pokud je název textového pole stejný jako název sloupce v podkladové datové sadě, přenesse se obsah databáze do pole v textovém dokumentu.

```

        inIndex = oForm.findColumn(stColumnName)
        oTextField.Anchor.String = oForm.getString(inIndex)
    End If
End If
Loop
End Sub

```

Volání aplikací pro otevírání souborů

Tato procedura umožňuje jediným klepnutím do textového pole vyvolat program, který je v operačním systému spojen s příponou podle názvu souboru. Tímto způsobem lze sledovat internetové odkazy nebo spustit e-mailový program pro konkrétní adresu uloženou v databázi.

Pro tuto část viz také příklad databáze Example_Mail_File_activate.odt.

```

Sub Website_Mail_activate
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oForm As Object
    Dim oField As Object
    Dim oShell As Object
    Dim stField As String
    oDoc = thisComponent
    oDrawpage = oDoc.Drawpage
    oForm = oDrawpage.Forms.getByName("form")
    oField = oForm.getByName("url_mail")

```

Obsah pojmenovaného pole je načten. Může to být webová adresa začínající '**http://**', e-mailová adresa začínající '@' nebo cesta k dokumentu (například externě uložený obrázek nebo soubor PDF).

```

    stFeld = oField.Text
    If stField = "" Then
        Exit Sub
    End If

```

Pokud je pole prázdné, makro se okamžitě ukončí. Při zadávání dat se často stává, že se k polím přistupuje pomocí myši, ale klepnutí na pole za účelem prvního zápisu do něj by nemělo vést ke spuštění kódu makra.

Nyní se v poli hledá znak '@'. To by znamenalo e-mailovou adresu. Na tuto adresu by měl být spuštěn e-mailový program pro odesílání pošty.

```
If InStr(stField, "@") Then
    stField = "mailto:" + stField
```

Pokud není '@', je výraz převeden na adresu URL. Pokud začíná 'http://', nejedná se o soubor v místním souborovém systému, ale o internetový zdroj, který je třeba vyhledat pomocí webového prohlížeče. Jinak bude cesta začínat výrazem 'file:///'.
End If

```
Else
    stField = convertToUrl(stField)
End If
```

Nyní se vyhledá program, který operační systém těmto souborům přiřadil. Pro klíčové slovo 'mailto:' je to poštovní program, pro 'http://' prohlížeč a jinak musí systém rozhodnout pomocí přípony jména souboru.

```
oShell = createUnoService("com.sun.star.system.SystemShellExecute")
oShell.execute(stField, , 0)
```

End Sub

Volání poštovního programu s předdefinovaným obsahem

Předchozí příklad lze rozšířit na spuštění poštovního programu s předem definovaným předmětem a obsahem.

Pro tuto část viz také příklad databáze Example_Mail_File_activate.odt.

Poštovní program se spouští pomocí 'mailto:recipient?subject= &body= &cc= &bcc='. Poslední dvě položky se ve formuláři nevyskytují. Přílohy nejsou v definici 'mailto' uvedeny, ale někdy funguje 'attachment='.

```
Sub Mail_activate
    Dim oDoc As Object
    Dim oDrawpage As Object
    Dim oForm As Object
    Dim oField1 As Object
    Dim oField2 As Object
    Dim oField3 As Object
    Dim oField4 As Object
    Dim oShell As Object
    Dim stField1 As String
    Dim stField2 As String
    Dim stField3 As String
    Dim stField4 As String
    oDoc = thisComponent
```

```

oDrawpage = oDoc.Drawpage
oForm = oDrawpage.Forms.getByName("form")
oField1 = oForm.getByName("mail_to")
oField2 = oForm.getByName("mail_subject")
oField3 = oForm.getByName("mail_body")
stField1 = oField1.Text
If stField1 = "" Then
    MsgBox "Missing email address." & Chr(13) &
        "Email program would not be activated" , 48, "Send Email"
Exit Sub
End If

```

Převod na adresu URL je nutný, aby se zabránilo rušení volání speciálními znaky a zalomením řádků. Tím se však k cestě připojí předpona 'file:///'. Těchto 8 znaků na začátku se nepřenáší.

```

stField2 = Mid(ConvertToUrl(oField2.Text), 9)
stField3 = Mid(ConvertToUrl(oField3.Text), 9)

```

Na rozdíl od prostého spuštění programu jsou zde podrobnosti o volání e-mailu uvedeny jako součást volání execute.

```

oShell = createUnoService("com.sun.star.system.SystemShellExecute")
oShell.execute("mailto:" + stField1 + "?subject=" + stField2 + "&body="
+ stField3,,0)
End Sub

```



Poznámka

Odeslání e-mailu pomocí poštovního programu lze provést také pomocí následujícího kódu, ale vlastní obsah e-mailu tímto způsobem vložit nelze.

```

Dim attachs(0)
oMailer = createUnoService("com.sun.star.system.SimpleSystemMail")
oMailProgramm = oMailer.querySimpleMailClient()
oNewmessage = oMailProgramm.createSimpleMailMessage()
oNeemessage.setRecipient(stField1)
oNewmessage.setSubject(stField2)
attachs(0) = "file:///..."
oNeueNachricht.setAttachement(attachs())
oMailprogramm.sendSimpleMailMessage(oNeuenachricht, 0 )

```

Možné parametry nalezneme v podrobnostech rozhraní *com::sun::star::system::XSimpleMailMessage* v [Referenci API](#).

Změna ukazatele myši při přechodu přes odkaz

Ukazatel myši prochází odkazem a mění se na ukazující ruku, což je na internetu běžné a aplikace Base tuto funkcionalitu používá také. Text odkazu může také změnit své vlastnosti a může být modrý a podtržený. Podobnost s internetovým odkazem je dokonalá. Každý uživatel očekává, že klepnutím otevře externí program.

Pro tuto část viz příklad databáze Example_Mail_File_activate.odt.

Tato krátká procedura by měla být vázána na událost textového pole **'Mouse inside'**.

```
Sub Mouse_pointer(Event As Object)
    REM Viz také Standardlibraries: Nástroje → ModuleControls →
    SwitchMousePointer
    Dim oPointer As Object
    oPointer = createUnoService("com.sun.star.awt.Pointer")
    oPointer.setType(27) 'Typy viz com.sun.star.awt.SystemPointer
    Event.Source.Peer.SetPointer(oPointer)
End Sub
```

Zobrazení formulářů bez panelu nástrojů

Nové uživatele Base často rozčiluje, že nástrojová lišta existuje, ale není možné ji ve formuláři použít. Tyto nástrojové lišty lze odstranit různými způsoby. Nejlepší způsoby ve všech verzích LibreOffice jsou dva a jsou popsány dále.

Velikost oken a nástrojových lišt se obvykle řídí makrem, které se spouští z dokumentu formuláře pomocí **Nástroje > Přizpůsobit > Události > Otevřít dokument**. Týká se to celého dokumentu, nikoli jednotlivých hlavních formulářů nebo podformulářů.

Formuláře bez panelu nástrojů v okně

Velikost okna lze měnit. Pomocí příslušného tlačítka ji lze také zavřít. Tyto úkoly provádí správce oken systému. Polohu a velikost okna na obrazovce lze zadat makrem při spuštění programu.

```
Sub Hide_toolbar
    Dim oFrame As Object
    Dim oWin As Object
    Dim oLayoutMng As Object
    Dim aElements()
    oFrame = StarDesktop.GetCurrentFrame()
```

Název formuláře se zobrazí v záhlaví okna.

```
oFrame.setTitle "My Form"
oWin = oFrame.getContainerWindow()
```

Okno je maximalizováno. To není totéž jako celoobrazovkový režim, protože hlavní lišta je stále viditelná a okno má titulkový pruh, který lze použít ke změně jeho velikosti nebo k jeho zavření.

```
oWin.IsMaximized = true
```

Je možné vytvořit okno s určitou velikostí a polohou. To se provádí pomocí **'oWin.setSize(0, 0, 600, 400, 15)'**. Zde se okno zobrazí v levém horním rohu obrazovky o šířce 600 pixelů a výšce 400. Poslední číslo znamená, že jsou uvedeny všechny pixely. Nazývá se **'Flag'** (Značka, pozn. překl.). **'Flag'** se vypočítá ze součtu následujících hodnot: x=1, y=2, šířka=4, výška=8. Protože jsou dány hodnoty x, y, šířka a výška, má **'Flag'** velikost 1+2+4+8=15.

```
oLayoutMng = oFrame.LayoutManager
aElements = oLayoutMng.getElements()
For i = LBound(aElements) To UBound(aElements)
    If aElements(i).ResourceURL =
```



```

        "private:resource/toolbar/formsnavigationbar" Then
    Else
        oLayoutMng.hideElement(aElements(i).ResourceURL)
    End If
Next
End Sub

```

V případě lišty navigace formuláře není třeba dělat nic. Formulář musí zůstat použitelný i v případech, kdy není zabudován ovládací prvek lišty navigace (který by stejně způsobil skrytí lišty navigace). Skryté by měly být pouze jiné nástrojové lišty než lišta navigace. Z tohoto důvodu není v tomto případě podána žádná akce.

Pokud nástrojové lišty neobnovíme přímo po opuštění formuláře, budou stále skryté. Lze je samozřejmě obnovit pomocí **Zobrazení > Nástrojové lišty**. Bylo by však poněkud nepříjemné, kdyby chyběla standardní nástrojová lišta (**Zobrazit > Nástrojové lišty > Standardní**) nebo stavový řádek (**Zobrazit > Stavový řádek**).

Tato procedura obnoví ('showElement') nástrojové lišty ze skrytého stavu ('hideElement'). V komentářích jsou uvedeny lišty, jejichž absence by byla nejpravděpodobněji zaznamenána.

```

Sub Show_toolbar
    Dim oFrame As Object
    Dim oLayoutMng As Object
    Dim aElements()
    oFrame = StarDesktop.getCurrentFrame()
    oLayoutMng = oFrame.LayoutManager
    aElements = oLayoutMng.getElements()
    For i = LBound(aElements) To UBound(aElements)
        oLayoutMng.showElement(aElements(i).ResourceURL)
    Next
    ' důležité prvky, které mohou chybět:
    ' "private:resource/toolbar/standardbar"
    ' "private:resource/statusbar/statusbar"
End Sub

```

Makra jsou vázána na: **Nástroje > Přizpůsobit > Události > Otevřít dokument > Hide_toolbar** a **Zavřít dokument > Show_toolbar**.

Nástrojové lišty se často nevracejí. V nejhorších případech může být užitečné nenačítat ty prvky, které správce rozvržení již zná, ale nejprve vytvořit konkrétní nástrojové lišty a pak je zobrazit:

```

Sub Hide_toolbar
    Dim oFrame As Object
    Dim oLayoutMng As Object
    Dim i As Integer
    Dim aElements(5) As String
    oFrame = StarDesktop.getCurrentFrame()
    oLayoutMng = oFrame.LayoutManager

```

```

aElements(0) = "private:resource/menubar/menubar"
aElements(1) = "private:resource/statusbar/statusbar"
aElements(2) = "private:resource/toolbar/formsnavigationbar"
aElements(3) = "private:resource/toolbar/standardbar"
aElements(4) = "private:resource/toolbar/formdesign"
aElements(5) = "private:resource/toolbar/formcontrols"
For Each i In aElemente()
    IF Not(oLayoutMng.requestElement(i)) Then
        oLayoutMng.createElement(i)
    End If
oLayoutMng.showElement(i)
Next i
End Sub

```

Nástrojové lišty, které mají být vytvořeny, jsou explicitně pojmenovány. Pokud správce rozvržení nemá k dispozici odpovídající nástrojovou lištu, je vytvořen pomocí **createElement** a poté zobrazen pomocí **showElement**.

Formuláře v režimu celé obrazovky

V celoobrazovkovém režimu je formulářem pokryta celá obrazovka. Není zde žádná hlavní lišta ani jiné prvky, které by mohly zobrazovat, zda jsou spuštěny jiné programy.

```

Function Fullscreen(boSwitch As Boolean)
    Dim oDispatcher As Object
    Dim Props(0) As New com.sun.star.beans.PropertyValue
    oDispatcher = createUnoService("com.sun.star.frame.DispatchHelper")
    Props(0).Name = "FullScreen"
    Props(0).Value = boSwitch
    oDispatcher.executeDispatch(ThisComponent.CurrentController.Frame,
        ".uno:FullScreen", "", 0, Props())
End Function

```

Tato funkce se spouští následujícím postupem. Při tomto postupu se současně provede i předchozí postup pro odstranění nástrojových lišt – jinak se nástrojová lišta zobrazí a lze pomocí ní vypnout celoobrazovkový režim. Jedná se rovněž o nástrojovou lištu, i když má pouze jeden symbol.

```

Sub Fullscreen_on
    Fullscreen(true)
    Hide_toolbar
End Sub

```

Režim celé obrazovky ukončíme stisknutím klávesy 'ESC'. Pokud má být místo toho pro tento příkaz použito konkrétní tlačítko, lze použít následující řádek:

```

Sub Fullscreen_off
    Fullscreen(false)

```

```
Show_toolbar
```

```
End Sub
```

Spouštění formulářů přímo z otevření databáze

Pokud jsou nástrojové lišty pryč nebo má být formulář zobrazen v celoobrazovkovém režimu, musí se při otevření databázového souboru spustit přímo formulář. Jednoduchý příkaz k otevření formuláře nebude fungovat, protože v okamžiku otevření souboru připojení k databázi ještě neexistuje.

Následující makro se spouští z **Nástroje > Přizpůsobit > Události > Otevřít dokument**. Použijte možnost **Uložit do > Databasefile.odt**.

```
Sub Form_Directstart
```

```
Dim oDataSource As Object
```

```
oDataSource = ThisDatabaseDocument.CurrentController
```

```
If Not (oDataSource.isConnected()) Then
```

```
oDataSource.connect()
```

```
End If
```

```
ThisDatabaseDocument.FormDocuments.getByName("Formname").open
```

```
End Sub
```

Nejprve je třeba navázat spojení s databází. Kontrolér je součástí **ThisDatabaseDocument**, stejně jako formulář. Pak lze formulář spustit a načíst jeho data z databáze.

Přístup k databázi MySQL pomocí maker

Všechna dosud zobrazená makra byla součástí interní databáze HSQLDB. Při práci s externími databázemi je nutné provést několik změn a rozšíření.

Kód MySQL v makrech

Při přístupu k interní databázi musí být tabulky a pole uzavřeny v duplicitních dvojitých uvozovkách oproti SQL:

```
SELECT "Field" FROM "Table"
```

Protože tyto příkazy SQL musí být připraveny uvnitř maker, musí být dvojitě uvozovky maskovány:

```
stSQL = "SELECT ""Field"" FROM ""Table"""
```

Dotazy MySQL používají jinou formu maskování:

```
SELECT `Field` FROM `Database`.`Table`
```

Uvnitř kódu makra se tato forma maskování zobrazuje jako:

```
stSql = "SELECT `Field` FROM `Database`.`Table`"
```

Dočasné tabulky jako samostatné mezisklady

V předchozí kapitole byla často používána jednořádková tabulka pro vyhledávání nebo filtrování tabulek. To nebude fungovat v systému s více uživateli, protože ostatní uživatelé by pak byli závislí na hodnotě filtru někoho jiného. Dočasné tabulky v systému MySQL jsou přístupné pouze uživateli aktivního připojení, takže k těmto tabulkám lze přistupovat za účelem vyhledávání a filtrování.

Takové tabulky samozřejmě nelze vytvořit předem. Musí být vytvořeny při otevření souboru aplikace Base. Proto by se na otevření souboru *.odb mělo vázat následující makro.

```
Sub CreateTempTable
    oDatasource = thisDatabaseDocument.CurrentController
    If Not (oDatasource.isConnected()) Then oDatasource.connect()
    oConnection = oDatasource.ActiveConnection()
    oSQL_Statement = oConnection.createStatement()
    stSql = "CREATE TEMPORARY TABLE IF NOT EXISTS `Searchtmp` (`ID` INT
PRIMARY KEY,
    `Name` VARCHAR(50))"
    oSQL_Statement.executeUpdate(stSql)
End Sub
```

Při prvním otevření souboru *.odb neexistuje žádné připojení k externí databázi MySQL. Připojení musí být vytvořeno. Poté lze vytvořit dočasnou tabulku s potřebnými poli.

Dialogová okna

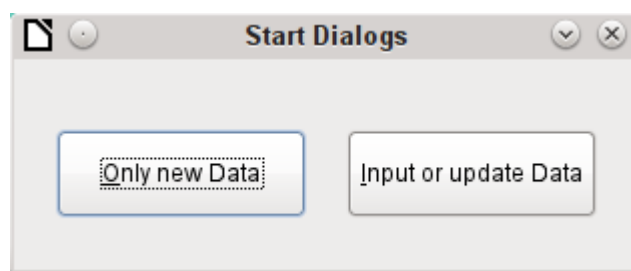
V aplikaci Base můžeme pro zadávání dat, jejich úpravu nebo údržbu databáze používat spíše dialogová okna než formuláře. Dialogová okna lze přímo přizpůsobit aktuálnímu prostředí aplikace, ale přirozeně nejsou předem definována tak pohodlně jako formuláře. Zde je krátký úvod zakončený poměrně složitým příkladem pro použití při údržbě databáze.

Spouštění a ukončování dialogů

Nejprve je třeba vytvořit dialogové okno v příslušném počítači. To se provádí pomocí **Nástroje > Makra > Uspořádat dialogová okna > Název souboru databáze > Standard > Nový**.

Dialogové okno se zobrazí se souvislou šedou plochou a záhlavím s ikonou zavření. Toto prázdné dialogové okno lze nyní vyvolat a poté opět zavřít.

Po klepnutí na dialogové okno je možné v části Obecné vlastnosti nastavit velikost a polohu. Lze zadat také obsah nadpisu Dialogového okna Start.



Obrázek 435: Dialog Spustit dialogy

Nástrojová lišta na spodním okraji okna obsahuje různé ovládací prvky formuláře. Z toho byla pro naše dialogové okno vybrána dvě tlačítka, která umožňují spouštět další dialogová okna. Úprava obsahu a vazba maker na události se provádí stejným způsobem jako u tlačítek ve formulářích.

Umístění deklarací proměnných pro dialogová okna vyžaduje zvláštní péči. Dialogové okno je deklarováno jako globální proměnná, aby k němu mohly přistupovat různé procedury. V tomto případě se dialogové okno nazývá oDialog0, protože budou existovat další dialogová okna s vyššími pořadovými čísly.

```
Dim oDialog0 As Object
```

Nejprve se načte knihovna pro dialogové okno. Pokud nebyl při vytváření dialogového okna zvolen jiný název, nachází se v adresáři Standard. Samotné dialogové okno je v této knihovně dostupné pod názvem Dialog0. **Execute()** spustí dialogové okno.

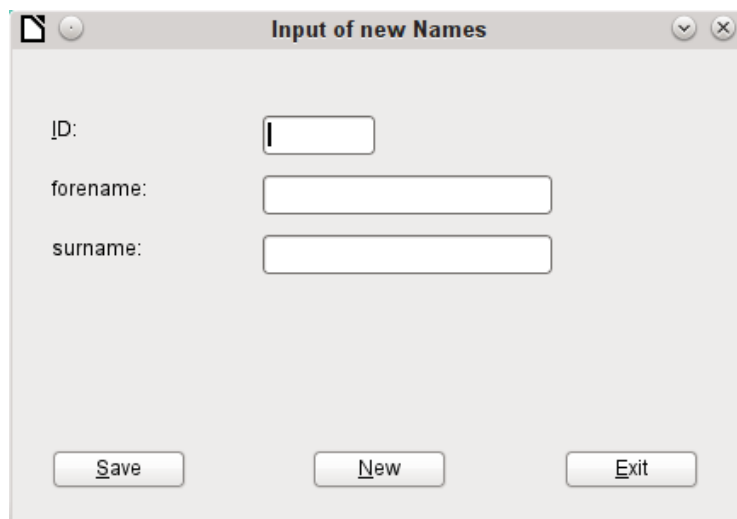
```
Sub Dialog0Start
    DialogLibraries.LoadLibrary("Standard")
    oDialog0 = createUnoDialog(DialogLibraries.Standard.Dialog0)
    oDialog0.Execute()
End Sub
```

Dialogové okno lze v zásadě zavřít pomocí tlačítka Zavřít na rámečku okna. Pokud však k tomu chceme použít jiné specifické tlačítko, je třeba v rámci procedury použít příkaz **EndExecute()**.

```
Sub Dialog0Ende
    oDialog0.EndExecute()
End Sub
```

V tomto rámci lze spustit a opět zavřít libovolný počet dialogů.

Jednoduchý dialog pro zadávání nových záznamů



Obrázek 436: Jednoduchý dialog pro zadávání nových záznamů

Toto dialogové okno je prvním krokem pro následující dialogové okno pro úpravu záznamů. Nejprve je objasněn základní přístup ke správě tabulek. Zde se jedná o uložení záznamů s novými primárními klíči nebo o úplně nové zadání záznamů. Nakolik může takové malé dialogové okno stačit pro zadávání do konkrétní databáze, závisí na požadavcích uživatele.

```
Dim oDialog1 As Object
```

přímo vytvoří globální proměnnou pro dialogové okno na nejvyšší úrovni modulu před všemi procedurami.

Dialogové okno se otevírá a zavírá stejným způsobem jako předchozí dialogové okno. Pouze se změní název z Dialog0 na Dialog1. Procedura pro zavření dialogového okna je vázána na tlačítko Exit.

Tlačítko New pomocí procedury DatafieldsClear vymaže v dialogovém okně všechny ovládací prvky z dřívějších záznamů.

```
Sub DatafieldsClear
```

```

oDialog1.getControl("NumericField1").Text = ""
oDialog1.getControl("TextField1").Text = ""
oDialog1.getControl("TextField2").Text = ""

```

End Sub

Každý ovládací prvek vložený do dialogového okna je přístupný podle názvu. Uživatelské rozhraní zajistí, aby se názvy neduplikovaly, což se u ovládacích prvků ve formuláři neděje.

Metoda **getControl** se používá s názvem ovládacího prvku. Také číselná pole mají vlastnost **Text**, kterou zde lze použít. To je jediný způsob, jak lze číselné pole vyprázdnit. Prázdný text existuje, ale prázdné číslo neexistuje. Místo toho musí být v poli primárního klíče zapsána 0.

Tlačítko **Uložit** spustí proceduru Data1Save:

Sub Data1Save

```

Dim oDatasource As Object
Dim oConnection As Object
Dim oSQL_Command As Object
Dim loID As Long
Dim stForename As String
Dim stSurname As String
loID = oDialog1.getControl("NumericField1").Value
stForename = oDialog1.getControl("TextField1").Text
stSurname = oDialog1.getControl("TextField2").Text
If loID > 0 And stSurname <> "" Then
    oDatasource = thisDatabaseDocument.CurrentController
    If Not (oDatasource.isConnected()) Then
        oDatasource.connect()
    End If
    oConnection = oDatasource.ActiveConnection()
    oSQL_Command = oConnection.createStatement()
    stSql = "SELECT ""ID"" FROM ""name"" WHERE ""ID"" = '"+loID+'"'
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
        MsgBox ("The value for field 'ID' already exist",16,
            "Duplicate Value")
        Exit Sub
    Wend
    stSql = "INSERT INTO ""name"" (""ID"", ""forename"", ""surname"")
        VALUES ('"+loID+"', '"+stForename+"', '"+stSurname+"')"
    oSQL_Command.executeUpdate(stSql)
    DatafieldsClear
End If

```

End Sub

Stejně jako v postupu DatafieldsClear se přistupuje ke vstupním polím. Tentokrát je přístup pouze pro čtení. Záznam bude předán pouze v případě, že v poli ID je hodnota větší než 0 a pole Surname obsahuje také text. Nulovou hodnotu ID lze vyloučit, protože číselná proměnná pro celá čísla je vždy inicializována na 0. Prázdné pole je proto uloženo s nulovou hodnotou.

Pokud je obsah obou polí zadán, vytvoří se spojení s databází. Protože ovládací prvky nejsou ve formuláři, musí být spojení s databází vytvořeno pomocí **thisDatabaseDocument.CurrentController**.

Nejprve se provede dotaz na databázi, zda již neexistuje záznam s daným primárním klíčem. Pokud tento dotaz vyprodukuje výsledek, zobrazí se okno se zprávou obsahující symbol Stop (kód: **16**) a zprávu „Duplicitní záznam“. Pak se procedura ukončí příkazem **Exit SUB**.

Pokud dotaz nenajde žádný záznam se stejným primárním klíčem, vloží se nový záznam do databáze pomocí příkazu insert. Poté se zavolá procedura DatafieldsClear, která vytvoří nový prázdný formulář.

Dialog pro úpravu záznamů v tabulce



Obrázek 437: Dialog pro vytváření, úpravy a mazání záznamů

Toto dialogové okno zjevně nabízí více možností než předchozí. Zde lze zobrazit všechny záznamy a procházet jimi, vytvářet nové záznamy nebo je mazat. Kód je samozřejmě mnohem složitější.

Tlačítko Exit je vázáno na proceduru, upravenou pro Dialog2, který byl popsán v předchozím dialogovém okně pro zadávání nových záznamů. Zde jsou popsána zbývající tlačítka a jejich funkce.

Zadávání dat v dialogovém okně je omezeno tím, že pole ID musí mít minimální hodnotu 1. Toto omezení souvisí s manipulací s proměnnými v jazyce Basic: číselné proměnné jsou z definice inicializovány na hodnotu 0. Pokud jsou tedy načteny číselné hodnoty z prázdných polí a z polí obsahujících 0, Basic mezi nimi nezjistí žádný rozdíl. To znamená, že pokud by v poli ID byla použita nula, musela by být nejprve načtena jako text a později případně převedena na číslo.

Dialogové okno se načte za stejných podmínek jako dříve. Postup načítání je však závislý na nulové hodnotě proměnné předané procedurou DataLoad.

```
Sub DataLoad(loID As Long)
    Dim oDataSource As Object
    Dim oConnection As Object
```

```

Dim oSQL_Command As Object
Dim stForename As String
Dim stSurname As String
Dim loRow As Long
Dim loRowMax As Long
Dim inStart As Integer
oDatasource = thisDatabaseDocument.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
If loID < 1 Then
    stSql = "SELECT MIN(""ID"") FROM ""name""
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
        loID = oResult.getInt(1)
    Wend
    inStart = 1
End If

```

Proměnné jsou deklarovány. Připojení k databázi pro dialogové okno se vytvoří, jak je popsáno výše. Na začátku má **loID** hodnotu **0**. Tento případ poskytuje nejnižší hodnotu primárního klíče povolenou v jazyce SQL. Příslušný záznam se později zobrazí v dialogovém okně. Zároveň je proměnná **inStart** nastavena na hodnotu 1, aby bylo možné dialogové okno spustit později. Pokud tabulka neobsahuje žádné záznamy, **loID** zůstane **0**. V takovém případě nebude třeba vyhledávat počet a obsah odpovídajících záznamů.

Pouze v případě, že **loID** je větší než 0, bude dotaz testovat, které záznamy jsou v databázi k dispozici. Druhý dotaz pak spočítá všechny záznamy, které se mají zobrazit. Třetí dotaz udává pozici aktuálního záznamu spočítáním všech záznamů s aktuálním primárním klíčem nebo menším.

```

If loID > 0 Then
    stSql = "SELECT * FROM ""name"" WHERE ""ID"" = '"+loID+'"'
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
        loID = oResult.getInt(1)
        stForename = oResult.getString(2)
        stSurname = oResult.getString(3)
    Wend
    stSql = "SELECT COUNT(""ID"") FROM ""name""
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next

```



```

        loRowMax = oResult.getInt(1)
    Wend
    stSql = "SELECT COUNT(""ID"") FROM ""name"" WHERE ""ID"" <=
'"+loID+"'"
    oResult = oSQL_Command.executeQuery(stSql)
    While oResult.next
        loRow = oResult.getInt(1)
    Wend
    oDialog2.getControl("NumericField1").Value = loID
    oDialog2.getControl("TextField1").Text = stForename
    oDialog2.getControl("TextField2").Text = stSurname
End If
oDialog2.getControl("NumericField2").Value = loRow
oDialog2.getControl("NumericField3").Value = loRowMax
If loRow = 1 Then
    ' previous Row (předchozí řádek, pozn. překl.)
    oDialog2.getControl("CommandButton4").Model.enabled = False
Else
    oDialog2.getControl("CommandButton4").Model.enabled = True
End If
If loRow <= loRowMax Then
    ' next Row | new Row | delete (další řádek | nový řádek | odstranit,
pozn. překl.)
    oDialog2.getControl("CommandButton5").Model.enabled = True
    oDialog2.getControl("CommandButton2").Model.enabled = True
    oDialog2.getControl("CommandButton6").Model.enabled = True
Else
    oDialog2.getControl("CommandButton5").Model.enabled = False
    oDialog2.getControl("CommandButton2").Model.enabled = False
    oDialog2.getControl("CommandButton6").Model.enabled = False
End If
IF inStart = 1 Then
    oDialog2.Execute()
End If
End Sub

```

Získané hodnoty se přenesou do polí dialogového okna. Vždy se zapisují údaje o aktuálním čísle záznamu a celkovém počtu vyhledaných záznamů, které nahrazují výchozí číselnou hodnotu 0.

Navigační tlačítka (CommandButton5 a CommandButton4) jsou použitelná pouze tehdy, když je možné se dostat k příslušnému záznamu. V opačném případě jsou dočasně deaktivovány pomocí **enabled = False**. Totéž platí pro tlačítka New (Nový, pozn. překl.) a Delete (Odstranit,

pozn. překl.). Neměly by být dostupné, pokud je číslo zobrazeného řádku vyšší než stanovený maximální počet řádků. Toto je výchozí nastavení tohoto dialogového okna při zadávání záznamů.

Dialogové okno by se mělo pokud možno spouštět pouze tehdy, když má být vytvořeno přímo ze startovacího souboru pomocí **DataLoad(0)**. Proto má speciální proměnná **inStart** na začátku procedury hodnotu 1.

Tlačítko < slouží k přechodu na předchozí záznam. Proto je toto tlačítko aktivní pouze v případě, že zobrazený záznam není první v seznamu. Navigace vyžaduje, aby byl primární klíč aktuálního záznamu načten z pole NumericField1.

Zde existují dva možné případy:

- 1) Přecházíme na nový záznam, takže příslušné pole nemá žádnou hodnotu. V tomto případě má **loID** výchozí hodnotu, která je podle definice celočíselné proměnné 0.
- 2) Jinak bude **loID** obsahovat hodnotu větší než 0. Pak lze dotazem určit hodnotu ID přímo pod aktuální hodnotou.

```
Sub PreviousRow
Dim loID As Long
Dim loIDnew As Long
loID = oDialog2.getControl("NumericField1").Value
oDatasource = thisDatabaseDocument.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
If loID < 1 Then
    stSql = "SELECT MAX(""ID"") FROM ""name""""
Else
    stSql = "SELECT MAX(""ID"") FROM ""name"" WHERE ""ID"" < '"+loID+'""
End If
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    loIDnew = oResult.getInt(1)
Wend
If loIDnew > 0 Then
    DataLoad(loIDnew)
End If
End Sub
```

Pokud je pole ID prázdné, mělo by se zobrazení změnit na nejvyšší hodnotu čísla primárního klíče. Pokud se naopak pole ID vztahuje k záznamu, měla by být vrácena předchozí hodnota ID.

Výsledek tohoto dotazu se použije k opětovnému spuštění procedury DataLoad s odpovídající hodnotou klíče.

Tlačítko > slouží k přechodu na další záznam. Tato možnost by měla existovat pouze v případě, že dialogové okno nebylo pro zadání nového záznamu vyprázdněno. To se přirozeně projeví při spuštění dialogového okna a také při prázdné tabulce.

Hodnota v poli NumericField1 je povinná. Na základě této hodnoty může SQL určit, který primární klíč je v tabulce další nejvyšší. Pokud je množina výsledků dotazu prázdná, protože neexistuje žádný odpovídající záznam, hodnota **loIDnew** = 0. V opačném případě se obsah dalšího záznamu načte pomocí DataLoad.

Sub NextRow

```
Dim loID As Long
Dim loIDnew As Long
loID = oDialog2.getControl("NumericField1").Value
oDatasource = thisDatabaseDocument.CurrentController
If Not (oDatasource.isConnected()) Then
    oDatasource.connect()
End If
oConnection = oDatasource.ActiveConnection()
oSQL_Command = oConnection.createStatement()
stSql = "SELECT MIN(""ID"") FROM ""name"" WHERE ""ID"" > '"+loID+'""
oResult = oSQL_Command.executeQuery(stSql)
While oResult.next
    loIDnew = oResult.getInt(1)
Wend
If loIDnew > 0 Then
    DataLoad(loIDnew)
Else
    Datafields2Clear
End If
```

End Sub

Pokud při přechodu na další záznam není k dispozici žádný další záznam, navigační klávesa spustí následující proceduru Datafields2Clear, která slouží k přípravě na zadání nového záznamu.

Procedura Datafields2Clear nevyprázdňuje pouze samotná datová pole. Pozice aktuálního záznamu je nastavena na hodnotu o jedna vyšší, než je maximální číslo záznamu, čímž je zřejmé, že záznam, na kterém se právě pracuje, ještě není zařazen do databáze.

Jakmile je spuštěna funkce Datafields2Clear, je aktivována možnost skoku na předchozí záznam, Skoky na následující záznam a použití procedur New a Delete jsou deaktivovány.

Sub Datafields2Clear

```
loRowMax = oDialog2.getControl("NumericField3").Value
oDialog2.getControl("NumericField1").Text = ""
oDialog2.getControl("TextField1").Text = ""
oDialog2.getControl("TextField2").Text = ""
```

```

oDialog2.getControl("NumericField2").Value = loRowMax + 1
oDialog2.getControl("CommandButton4").Model.enabled = True ' Previous
record
oDialog2.getControl("CommandButton5").Model.enabled = False ' Next
record oDialog2.getControl("CommandButton2").Model.enabled = False ' New
record
oDialog2.getControl("CommandButton6").Model.enabled = False ' Delete
End Sub

```

Uložení záznamů by mělo být možné pouze tehdy, pokud pole ID a Surname obsahují záznamy. Pokud je tato podmínka splněna, postup testuje, zda se jedná o nový záznam. Využívá se přitom ukazatel záznamu, který je pro nové záznamy nastaven tak, aby byl o jedničku vyšší než maximální počet záznamů.

U nových záznamů se provádí kontrola toho, že operace uložení proběhne úspěšně. Pokud bylo číslo použité pro primární klíč použito již dříve, zobrazí se varování. Pokud je na zobrazenou otázku odpovězeno Ano, stávající záznam s tímto číslem se přepíše. V opačném případě bude ukládání přerušeno. Pokud v databázi nejsou žádné existující záznamy (**loRowMax = 0**), je tento test zbytečný a nový záznam lze uložit přímo. Pro nový záznam se počet záznamů zvýší o 1 a zadané hodnoty se vymažou pro další záznam.

Existující záznamy se jednoduše přepíší příkazem update.

```

Sub Data2Save(oEvent As Object)
    Dim oDatasource As Object
    Dim oConnection As Object
    Dim oSQL_Command As Object
    Dim oDlg As Object
    Dim loID As Long
    Dim stForename As String
    Dim stSurname As String
    Dim inMsg As Integer
    Dim loRow As Long
    Dim loRowMax As Long
    Dim stSql As String
    oDlg = oEvent.Source.getContext()
    loID = oDlg.getControl("NumericField1").Value
    stForename = oDlg.getControl("TextField1").Text
    stSurname = oDlg.getControl("TextField2").Text
    If loID > 0 And stSurname <> "" Then
        oDatasource = thisDatabaseDocument.CurrentController
        If Not (oDatasource.isConnected()) Then
            oDatasource.connect()
        End If
        oConnection = oDatasource.ActiveConnection()
        oSQL_Command = oConnection.createStatement()
    End If
End Sub

```

```

loRow = oDlg.getControl("NumericField2").Value
loRowMax = oDlg.getControl("NumericField3").Value
If loRowMax < loRow Then
    If loRowMax > 0 Then
        stSql = "SELECT ""ID"" FROM ""name"" WHERE ""ID"" = '"+loID+'"'
        oResult = oSQL_Command.executeQuery(stSql)
        While oResult.next
            inMsg = MsgBox ("The value for field 'ID' already exist." &
                CHR(13) & "Should the row be updated?",20,
                "Duplicate Value")
            If inMsg = 6 Then
                stSql = "UPDATE ""name"" SET
""forename""='"+stForename+'',
                ""surname""='"+stSurname+' WHERE ""ID"" = '"+loID+'"'
                oSQL_Command.executeUpdate(stSql)
                DataLoad(loID) ' Pomocí aktualizace byl přepsán řádek..
Rowcount          musí být vynulován
            End If
        Exit Sub
    Wend
End If
VALUES
stSql = "INSERT INTO ""name"" (""ID"", ""forename"", ""surname"")
('"+loID+'','"+stForename+'','"+stSurname+'')"'
oSQL_Command.executeUpdate(stSql)
oDlg.getControl("NumericField3").Value = loRowMax + 1
' Po vložení se přidá jeden řádek
Datafields2Clear
' After insert would be moved to next insert
Else
    stSql = "UPDATE ""name"" SET ""forename""='"+stForename+'',
        ""surname""='"+stSurname+' WHERE ""ID"" = '"+loID+'"'
    oSQL_Command.executeUpdate(stSql)
End If
End If
End Sub

```

Postup mazání je opatřen doplňující otázkou, která má zabránit náhodnému smazání. Vzhledem k tomu, že toto tlačítko je deaktivováno, když jsou vstupní pole prázdná, nemělo by se prázdné pole NumericField1 nikdy objevit. Proto lze kontrolní podmínku **IF loID > 0** vynechat.

Při mazání se počet záznamů sníží o 1. To je třeba opravit pomocí **loRowMax - 1**. Poté se zobrazí záznam následující po aktuálním.

```
Sub DataDelete(oEvent As Object)
```

```

Dim oDatasource As Object
Dim oConnection As Object
Dim oSQL_Command As Object
Dim oDlg As Object
Dim loID As Long
oDlg = oEvent.Source.getContext()
loID = oDlg.getControl("NumericField1").Value
If loID > 0 Then
    inMsg = MsgBox ("Should current data be deleted?",20,
        "Delete current row")
    If inMsg = 6 Then
        oDatasource = thisDatabaseDocument.CurrentController
        If Not (oDatasource.isConnected()) Then
            oDatasource.connect()
        End If
        oConnection = oDatasource.ActiveConnection()
        oSQL_Command = oConnection.createStatement()
        stSql = "DELETE FROM ""name"" WHERE ""ID"" = '"+loID+'"'
        oSQL_Command.executeUpdate(stSql)
        loRowMax = oDlg.getControl("NumericField3").Value
        oDlg.getControl("NumericField3").Value = loRowMax - 1
        NextRow
    End If
ELSE
    MsgBox ("No row deleted." & CHR(13) &
        "No data selected.",64,"Delete impossible")
End If
End Sub

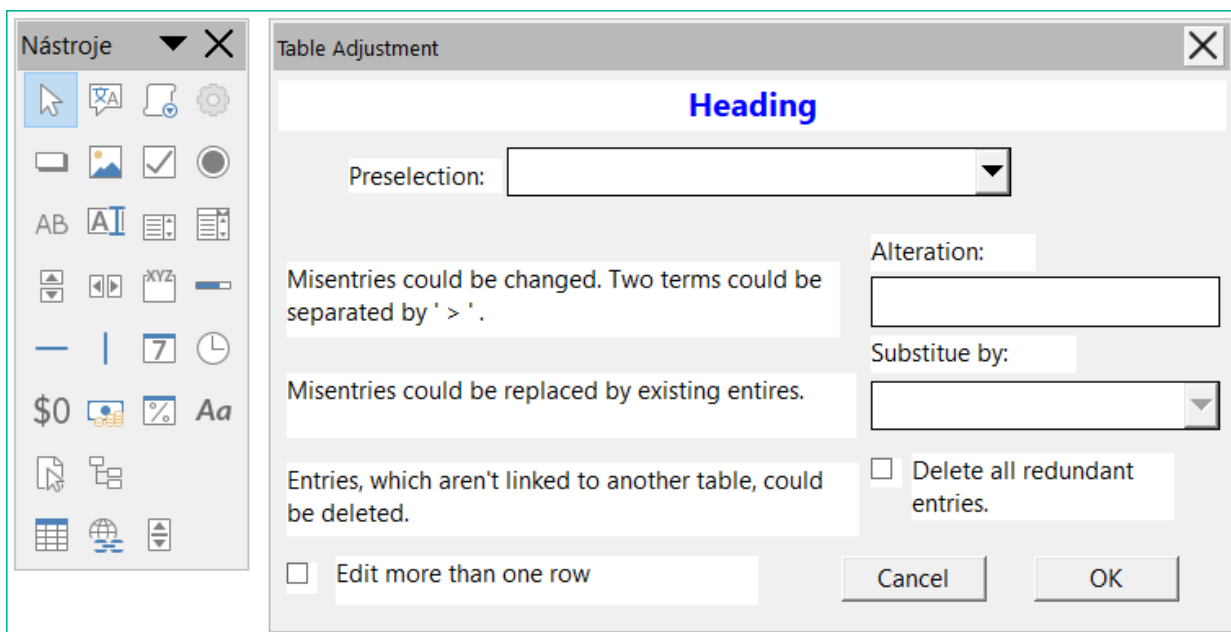
```

Tento malý dialog ukázal, že použití makrokódu může poskytnout základ pro zpracování záznamů. Přístup prostřednictvím formulářů je mnohem jednodušší, ale dialogové okno se může velmi flexibilně přizpůsobit požadavkům programu. Není však vhodný pro rychlé vytvoření databázového rozhraní.

Použití dialogového okna k vyčištění chybných záznamů v tabulkách

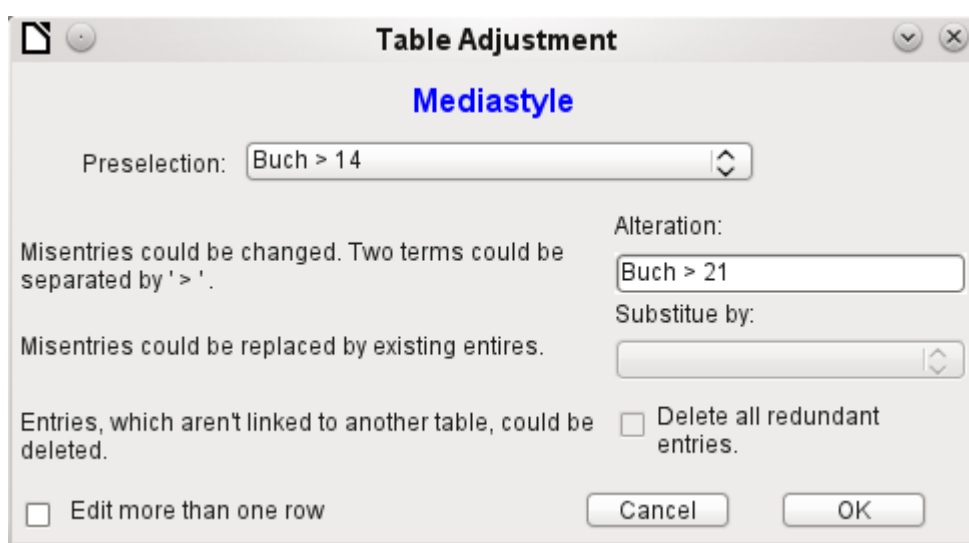
Chyby v polích se často objeví až později. Často je nutné upravit shodné záznamy v několika záznamech současně. V běžném zobrazení tabulky je to nepohodlné, zejména když je třeba upravit několik záznamů, protože každý záznam vyžaduje samostatný zápis.

Formuláře mohou k tomuto účelu používat makra, ale abychom to mohli provést pro několik tabulek, potřebovali bychom stejně konstruované formuláře. Dialogová okna to zvládnou. Dialogové okno může být na začátku opatřeno potřebnými údaji pro příslušné tabulky a může být vyvoláno několika různými formuláři.



Obrázek 438: Návrh dialogového okna pro úpravu tabulky pomocí nástrojové lišty Nástroje (vlevo)

Dialogová okna se ukládají spolu s moduly pro makra. Jejich tvorba je podobná tvorbě formuláře. K dispozici jsou velmi podobná kontrolní pole. Chybí pouze ovládací prvky tabulky pro formuláře jako zvláštní možnost zadání.



Obrázek 439: Použití dialogového okna Úprava tabulky

Vzhled ovládacích prvků dialogového okna je určen nastavením grafického uživatelského rozhraní.

Výše uvedené dialogové okno slouží v ukázkové databázi k úpravě tabulek, které nejsou přímo použity jako základ formuláře. Tak například typ média je přístupný pouze prostřednictvím pole se seznamem (ve verzi s makry se z něj stane kombinované pole). Ve verzi s makry lze obsah pole rozšířit o nový obsah, ale změna stávajícího obsahu není možná. Ve verzi bez maker se změny provádějí pomocí samostatného ovládacího prvku tabulky.

Zatímco změny v tomto případě lze snadno provést bez maker, je poměrně obtížné změnit typ média u mnoha médií najednou. Předpokládejme, že jsou k dispozici následující typy: "Book, bound", "Book, hard-cover", "Paperback" a "Ringfile". Nyní, po dlouhé době používání databáze, se ukazuje, že aktivnější současníci předpokládali podobné další typy médií pro tištěná díla.

Jejich rozlišování se stalo obtížným úkolem. Chceme je proto omezit, nejlépe na jediný termín. Bez maker by bylo nutné záznamy v tabulce médií vyhledat (pomocí filtru) a jednotlivě změnit. Pokud známe jazyk SQL, můžeme to udělat mnohem lépe pomocí příkazu SQL. Všechny záznamy v tabulce Media můžeme změnit jediným záznamem. Druhý příkaz SQL pak odstraní nyní přebytečné typy médií, které již nemají žádnou vazbu na tabulku Média. Přesně tato metoda se použije pomocí dialogového okna Nahradit za – pouze příkaz SQL se nejprve přizpůsobí tabulce Media Type pomocí makra, které může upravovat i jiné tabulky.

Často se do tabulky dostanou položky, které lze zpětně ve formuláři změnit, a proto již nejsou potřeba. Není na škodu takové osiřelé položky jednoduše odstranit, ale pomocí grafického uživatelského rozhraní se poměrně špatně hledají. I v tomto případě je užitečný vhodný příkaz SQL spojený s příkazem k odstranění. Tento příkaz pro dotčené tabulky je obsažen v dialogovém okně v části Odstranit všechny nadbytečné položky.

Pokud má být dialogové okno použito k provedení více změn, je to označeno zaškrtnutím políčkem Upravit více záznamů. Pak se dialogové okno po klepnutí na tlačítko OK jednoduše neukončí.

Celý kód makra pro toto dialogové okno si můžeme prohlédnout v databázi příkladů. Níže jsou vysvětleny pouze úryvky.

```
Sub Table_purge(oEvent As Object)
```

Makro by mělo být spuštěno zadáním do sekce Další informace pro příslušná tlačítka:

0: Formulář, 1: Podformulář, 2: Podformulář, 3: Rozevírací seznam nebo ovládací prvek tabulky, 4: Pole cizího klíče ve formuláři, prázdné pro ovládací prvek tabulky, 5: Název pomocné tabulky, 6: Pole Field1 pomocné tabulky, 7: Pole Field2 pomocné tabulky nebo 8: Název pomocné tabulky pro pole Field2 tabulky.

Položky v této oblasti jsou uvedeny na začátku makra jako komentáře. Přenesou se čísla, která jsou s nimi svázána, a příslušný záznam se načte z pole. Makro může upravovat seznamy, které mají dvě položky oddělené znakem ">". Tyto dva záznamy mohou také pocházet z různých tabulek a mohou být spojeny pomocí dotazu, jako například v tabulce Postcode, která má pro město pouze pole cizího klíče Town_ID, což vyžaduje tabulku Town pro zobrazení názvů měst.

```
Dim aForeignTable(0, 0 to 1)
Dim aForeignTable2(0, 0 to 1)
```

Mezi proměnnými definovanými na začátku jsou dvě pole. Zatímco normální pole lze vytvořit příkazem **Split()** během provádění podprogramu, dvourozměrná pole je třeba definovat předem. Dvourozměrná pole jsou nutná pro uložení několika záznamů z jednoho dotazu, pokud se dotaz sám vztahuje k více než jednomu poli. Obě výše deklarovaná pole musí být schopna interpretovat dotazy, které se vztahují ke dvěma polím tabulky. Proto jsou definována pro dva různé obsahy s použitím hodnot 0 až 1 pro druhý rozměr.

```
stTag = oEvent.Source.Model.Tag
aTable() = Split(stTag, ">")
For i = LBound(aTable()) To UBound(aTable())
    aTable(i) = trim(aTable(i))
Next
```

Poskytnuté proměnné se čtou. Pořadí je takové, jaké je uvedeno v komentáři výše. Existuje maximálně devět záznamů a je třeba deklarovat, zda existuje osmý záznam pro tabulku field2 a devátý záznam pro druhou tabulku.

Pokud mají být hodnoty z tabulky odstraněny, je třeba nejprve zkontrolovat, zda neexistují jako cizí klíče v jiné tabulce. V jednoduchých strukturách tabulek bude mít daná tabulka pouze jedno spojení cizího klíče s jinou tabulkou. V uvedené příkladové databázi však existuje tabulka Town, která se používá jak pro místo vydání médií, tak pro město a pro adresy. Primární klíč tabulky

Town je tedy zadán dvakrát do různých tabulek. Tyto tabulky a názvy cizích klíčů lze samozřejmě zadat také pomocí pole Další informace. Bylo by však lepší, kdyby mohly být poskytovány univerzálně pro všechny případy. To lze provést pomocí následujícího dotazu.

```
stSql = "SELECT ""FKTABLE_NAME"", ""FKCOLUMN_NAME"" FROM  
""INFORMATION_SCHEMA"". ""SYSTEM_CROSSREFERENCE"" WHERE ""PKTABLE_NAME"" = '' +  
aTable(5) + ''"
```

V databázi obsahuje oblast INFORMATION_SCHEMA všechny informace o tabulkách databáze, včetně informací o cizích klíčích. K tabulkám, které obsahují tyto informace, lze přistupovat pomocí "INFORMATION_SCHEMA". "SYSTEM_CROSSREFERENCE". "KTABLE_NAME" uvádí tabulku, která poskytuje primární klíč pro připojení. "FKTABLE_NAME" udává tabulku, která používá tento primární klíč jako cizí klíč. Konečně "FKCOLUMN_NAME" udává název pole cizího klíče.

Tabulka, která poskytuje svůj primární klíč pro použití jako cizí klíč, je v dříve vytvořeném poli na pozici 6. A počet začíná 0, hodnota se načte z pole pomocí **aTable(5)**.

```
inCount = 0  
stForeignIDTab1Tab2 = "ID"  
stForeignIDTab2Tab1 = "ID"  
stAuxiltable = aTable(5)
```

Před zahájením čtení polí je třeba nastavit některé výchozí hodnoty. Jedná se o index pole, do kterého se budou zapisovat hodnoty z pomocné tabulky, výchozí primární klíč, pokud nepotřebujeme cizí klíč pro druhou tabulku, a výchozí pomocnou tabulku, propojenou s hlavní tabulkou, pro PSČ a obec, tabulku Postcode.

Pokud jsou dvě pole propojena pro zobrazení v seznamu, mohou, jak je popsáno výše, pocházet ze dvou různých tabulek. Pro zobrazení PSČ a města je dotaz následující:

```
SELECT "Postcode"."Postcode" || ' > ' || "Town"."Town" FROM "Postcode",  
"Town" WHERE "Postcode"."Town_ID" = "Town"."ID"
```

Tabulka pro první pole (PSČ) je propojena s druhou tabulkou cizím klíčem. Makru se předávají pouze informace z těchto dvou tabulek a polí Postcode a Town. Všechny primární klíče se v ukázkové databázi ve výchozím nastavení nazývají ID. Cizí klíč Town v položce Postcode je proto třeba určit pomocí makra.

Stejným způsobem musí makro přistupovat ke každé tabulce, s níž je obsah pole seznamu spojen cizím klíčem.

```
oQuery_result = oSQL_Statement.executeQuery(stSql)  
If Not IsNull(oQuery_result) Then  
    While oQuery_result.next  
        ReDim Preserve aForeignTable(inCount, 0 to 1)
```

Pole musí být pokaždé nově dimenzováno. Aby se zachoval stávající obsah, zálohuje se pomocí (Preserve).

```
aForeignTables(inCount, 0) = oQuery_result.getString(1)
```

Načtení prvního pole s názvem tabulky, která obsahuje cizí klíč. Výsledkem pro tabulku Postcode je tabulka Address.

```
aForeignTables(inCount, 1) = oQuery_result.getString(2)
```

Načtení druhého pole s názvem pole cizího klíče. Výsledkem pro tabulku Postcode je pole Postcode_ID v tabulce Address.

Pokud volání podprogramu obsahuje název druhé tabulky, spustí se následující smyčka. Pouze pokud se název druhé tabulky vyskytuje jako tabulka cizího klíče pro první tabulku, je změněn výchozí záznam. V našem případě k tomu nedojde, protože tabulka Town nemá cizí klíč z tabulky Postcode. Výchozí položkou pomocné tabulky proto zůstává Postcode; kombinace PSČ a obce je nakonec základem tabulky Address, která obsahuje cizí klíč z tabulky Postcode.

```
If UBound(aTable()) = 8 Then
    If aTable(8) = aForeignTable(inCount, 0) Then
        stForeignIDTab2Tab1 = aForeignTable(inCount, 1)
        stAuxiltable = aTable(8)
    End If
End If
inCount = inCount + 1
```

Protože může být nutné načíst další hodnoty, index se zvýší, aby se změnila velikost polí. Pak smyčka skončí.

```
Wend
End If
```

Pokud při volání podprogramu existuje druhý název tabulky, spustí se stejný dotaz pro tuto tabulku:

```
If UBound(aTable()) = 8 Then
```

Probíhá identicky s tím rozdílem, že smyčka testuje, zda se snad první název tabulky nevyskytuje jako název tabulky cizího klíče. To je tento případ: tabulka Postcode obsahuje cizí klíč Town_ID z tabulky Town. Tento cizí klíč je nyní přiřazen proměnné stForeignIDTab1Tab2, aby bylo možné definovat vztah mezi tabulkami.

```
If aTable(5) = aForeignTable2(inCount, 0) Then
    stForeignIDTab1Tab2 = aForeignTable2(inCount, 1)
End If
```

Po několika dalších nastaveních, která zajistí návrat na správný formulář po spuštění dialogu (určení čísla řádku formuláře, abychom se po novém načtení mohli vrátit zpět na toto číslo řádku), se spustí smyčka, která znovu vytvoří dialog po dokončení první akce, ale dialog musí zůstat otevřený pro další akce. Nastavení opakování se provádí pomocí příslušného zaškrtačacího políčka.

```
Do
```

Před spuštěním dialogu se nejprve určí obsah polí se seznamem. Je třeba dbát na to, aby seznamová pole představovala dvě pole tabulky a možná se dokonce vztahovala ke dvěma různým tabulkám.

```
If UBound(aTable()) = 6 Then
```

Pole se seznamem se vztahuje pouze k jedné tabulce a jednomu poli, protože pole argumentů končí u pole Tablefield1 pomocné tabulky.

```
stSql = "SELECT "" + aTable(6) + "" FROM "" + aTable(5) + ""
ORDER BY "" + aTable(6) + """
```

```
ElseIf UBound(aTable()) = 7 Then
```

Pole se seznamem se vztahuje ke dvěma polím tabulky, ale pouze k jedné tabulce, protože pole argumentů končí na poli Tablefield2 pomocné tabulky.

```

        stSql = "SELECT "" + aTable(6) + ""||' > '||"" + aTable(7) +
"" FROM "" + aTable(5) + "" ORDER BY "" + aTable(6) + """"
    Else

```

Pole se seznamem je založeno na dvou polích ze dvou tabulek. Tento dotaz odpovídá příkladu s poštovním směrovacím číslem a městem.

```

        stSql = "SELECT "" + aTable(5) + ""."" + aTable(6) + ""||' >
'||"" + aTable(8) + ""."" + aTable(7) + "" FROM "" + aTable(5) + "",
"" + aTable(8) + "" WHERE "" + aTable(8) + ""."" +
stForeignIDTab2Tab1 + "" = "" + aTable(5) + ""."" +
stForeignIDTab1Tab2 + "" ORDER BY "" + aTable(6) + """"
    End If

```

Zde máme první vyhodnocení pro určení cizích klíčů. Proměnné stForeignIDTab2Tab1 a stForeignIDTab1Tab2 začínají hodnotou ID. Pro stForeignIDTab1Tab2 je výsledkem vyhodnocení předchozího dotazu jiná hodnota, a to hodnota Town_ID. Tímto způsobem předchozí konstrukce dotazu poskytuje přesně ten obsah, který byl již formulován pro PSČ a obec – pouze rozšířený o řazení.

Nyní je třeba navázat kontakt s poli se seznamem, abychom jim dodali obsah vrácený dotazy. Tato pole se seznamem ještě neexistují, protože dialogové okno ještě nebylo vytvořeno. Toto dialogové okno se nejprve vytvoří v paměti pomocí následujících řádků a teprve poté se vykreslí na obrazovku.

```

    DialogLibraries.LoadLibrary("Standard")
    oDlg =
CreateUnoDialog(DialogLibraries.Standard.Dialog_Table_purge)

```

Dále následuje nastavení polí dialogového okna. Zde je například pole se seznamem, do kterého se vloží výsledky výše uvedeného dotazu:

```

oCtlList1 = oDlg.GetControl("ListBox1")
oCtlList1.addItem(aContent(), 0)

```

Přístup k polím dialogového okna se provádí pomocí **GetControl** s příslušným názvem. V dialogových oknech není možné, aby dvě pole měla stejný název, protože by to způsobilo problémy při vyhodnocování dialogu.

Do rozevíracího seznamu je vložen obsah dotazu, který je uložen v poli aContent(). Pole se seznamem obsahuje pouze obsah, který se má zobrazit jako pole, takže je vyplněna pouze pozice 0.

Po vyplnění všech polí s požadovaným obsahem se spustí dialogové okno.

```

Select Case oDlg.Execute()
Case 1 'Case 1 znamená, že bylo klepnuto na tlačítko "OK"
Case 0 'Pokud bylo použito tlačítko "Zrušit"
    inRepetition = 0
End Select
Loop While inRepetition = 1

```

Dialogové okno se spouští opakovaně, dokud je hodnota "inRepetition" 1. Nastavuje se pomocí příslušného zaškrťovacího políčka.

Zde je ve zkratce uveden obsah po klepnutí na tlačítko "OK":

```

Case 1

```

```
stInhalt1 = oCtlList1.getSelectedItemAt() 'Přečteme hodnotu prvního  
pole se seznamem Listbox1 ...
```

```
REM ... a určíme odpovídající hodnotu ID.
```

Hodnota ID prvního pole se seznamem je uložena v proměnné "inLB1".

```
stText = oCtlText.Text ' Read the field value.
```

Pokud textové pole není prázdné, je zadání v textovém poli zpracováno. Pole se seznamem pro náhradní hodnotu ani zaškrtačkové pole pro odstranění všech osířelých záznamů se nezohledňují. To je zřejmé z toho, že zadávání textu nastavuje tato ostatní pole jako neaktivní.

```
If stText <> "" Then
```

Pokud textové pole není prázdné, zapíše se nová hodnota na místo staré hodnoty pomocí dříve načteného pole ID v tabulce. Je zde možnost dvou záznamů, stejně jako v případě pole se seznamem. Oddělovač je >. Pro dva záznamy v různých tabulkách je třeba spustit dva příkazy UPDATE, které se zde vytvoří současně a předají se oddělené středníkem.

```
ElseIf oCtlList2.getSelectedItemAt() <> "" Then
```

Pokud je textové pole prázdné a pole listbox 2 obsahuje hodnotu, musí být hodnota z pole listbox 1 nahrazena hodnotou v poli listbox 2. To znamená, že všechny záznamy v tabulkách, pro které jsou záznamy v polích seznamu cizími klíči, musí být zkontrolovány a v případě potřeby zapsány se změněným cizím klíčem.

```
stInhalt2 = oCtlList2.getSelectedItemAt()
```

```
REM Čtení hodnoty z pole se seznamem.
```

```
REM Určení ID pro hodnotu pole se seznamu.
```

Hodnota ID druhého pole se seznamem je uložena v proměnné inLB2. I zde se situace vyvíjí odlišně v závislosti na tom, zda je v poli se seznamem obsaženo jedno nebo dvě pole, a také na tom, zda je základem obsahu pole se seznamem jedna nebo dvě tabulky.

Proces nahrazení závisí na tom, která tabulka je definována jako tabulka poskytující cizí klíč pro hlavní tabulku. Ve výše uvedeném příkladu se jedná o tabulku Postcode, protože Postcode_ID je cizí klíč, který je předáván přes Listbox 1 a Listbox 2.

```
If stAuxilTable = aTable(5) Then
```

```
For i = LBound(aForeignTables()) To UBound(aForeignTables())
```

Nahrazení staré hodnoty ID novou hodnotou ID se stává problematickým v relacích n:m, protože v takových případech může být stejná hodnota přiřazena dvakrát. To možná chceme, ale je třeba tomu zabránit, pokud je cizí klíč součástí primárního klíče. V tabulce rel_Media_Author tedy nemůže mít médium dvakrát stejného autora, protože primární klíč je vytvořen z Media_ID a Author_ID. V dotazu jsou prohledána všechna klíčová pole, která mají společně vlastnost UNIQUE nebo byla definována jako cizí klíče s vlastností UNIQUE pomocí indexu.

Pokud má tedy cizí klíč vlastnost UNIQUE a je v něm již zastoupen požadovaný budoucí klíč inLB2, nelze jej nahradit.

```
stSql = "SELECT ""COLUMN_NAME"" FROM  
""INFORMATION_SCHEMA"". ""SYSTEM_INDEXINFO"" WHERE ""TABLE_NAME"" = ' " +  
aForeignTables(i,0) + "' AND ""NON_UNIQUE"" = False AND ""INDEX_NAME"" =  
(SELECT ""INDEX_NAME"" FROM ""INFORMATION_SCHEMA"". ""SYSTEM_INDEXINFO"" WHERE  
""TABLE_NAME"" = ' " + aForeignTables(i,0) + "' AND ""COLUMN_NAME"" = ' " +  
aForeignTables(i,1) + "')"
```

' "NON_UNIQUE" = False ' uvádí názvy sloupců, které jsou UNIQUE. Nejsou však potřeba všechny názvy sloupců, ale pouze ty, které tvoří index s polem cizího klíče. To se řeší pomocí podvýběru se stejnými názvy tabulek (které obsahují cizí klíč) a názvy polí cizího klíče.

Pokud je nyní cizí klíč přítomen v sadě, lze hodnotu klíče nahradit pouze tehdy, pokud jsou ostatní pole použita k definování příslušného indexu jako UNIQUE. Při výměně je třeba dbát na to, aby nebyla narušena jedinečnost kombinace indexů.

```
If aForeignTables(i,1) = stFieldName Then
    inUnique = 1
Else
    ReDim Preserve aColumns(inCount)
    aColumns(inCount) = oQuery_result.getString(1)
    inCount = inCount + 1
End If
```

Všechny názvy sloupců, kromě známých názvů sloupců pro pole cizího klíče jako Index s vlastností UNIQUE, jsou uloženy v poli. Protože název sloupce pole cizího klíče patří také do skupiny, lze jej použít k určení, zda se má při úpravě dat kontrolovat jedinečnost.

```
If inUnique = 1 Then
    stSql = "UPDATE "" + aForeignTables(i,0) + "" AS ""a"" SET "" +
aForeignTables(i,1) + ""=" + inLB2 + "" WHERE "" + aForeignTables(i,1)
+ ""=" + inLB1 + "" AND ( SELECT COUNT(*) FROM "" + aForeignTables(i,0)
+ "" WHERE "" + aForeignTables(i,1) + ""=" + inLB2 + "" )"
    If inCount > 0 Then
        stFieldgroup = Join(aColumns(), ""|| ||"" )
```

Pokud kromě pole cizího klíče existuje několik polí, která dohromady tvoří UNIQUE index, jsou zde spojena pro seskupení SQL. Jinak se jako stFieldgroup zobrazí pouze aColumns(0).

```
stFieldName = ""
For ink = LBound(aColumns()) To UBound(aColumns())
    stFieldName = stFieldName + " AND "" + aColumns(ink) + "" =
""a"". "" + aColumns(ink) + "" "
```

Části jazyka SQL jsou kombinovány pro korelovaný poddotaz.

```
Next ink
stSql = Left(stSql, Len(stSql) - 1)
```

Předchozí dotaz končí závorkou. Nyní má být do poddotazu přidán další obsah, proto je třeba tento uzávěr opět odstranit. Poté je dotaz rozšířen o další podmínky.

```
stSql = stSql + stFeldbezeichnung + "GROUP BY ("" + stFeldgruppe +
"" ) < 1"
End If
```

Pokud cizí klíč není spojen s primárním klíčem nebo s indexem UNIQUE, nezáleží na tom, zda je obsah duplicitní.

```
Else
    stSql = "UPDATE "" + aForeignTables(i,0) + "" SET "" +
aForeignTables(i,1) + ""=" + inLB2 + "" WHERE "" + aForeignTables(i,1)
+ ""=" + inLB1 + "" "
End If
oSQL_Statement.executeQuery(stSql)
NEXT
```

Aktualizace se provádí tak dlouho, dokud se vyskytují různá spojení s jinými tabulkami, tj. dokud je aktuální tabulka zdrojem cizího klíče v jiné tabulce. V případě tabulky Town je to dvakrát: v tabulce Media a v tabulce Postcode.

Poté lze starou hodnotu z pole listbox 1 vymazat, protože již nemá žádnou vazbu na ostatní tabulky.

```
stSql = "DELETE FROM "" + aTable(5) + "" WHERE ""ID""=''' + inLB1 + ""'"
oSQL_Statement.executeQuery(stSql)
```

V některých případech je nyní třeba stejnou metodu provést pro druhou tabulku, která poskytl data pro pole se seznamem. V našem příkladu je první tabulkou tabulka Postcode a druhou tabulka Town.

Pokud je textové pole prázdné a pole listbox 2 také nic neobsahuje, zkontrolujeme, zda je v příslušném zaškrťávacím políčku uvedeno, že mají být odstraněny všechny přebytečné položky. To znamená záznamy, které nejsou vázány na jiné tabulky cizím klíčem.

```
ElseIf oCtlCheck1.State = 1 Then
    stCondition = ""
    If stAuxilTable = aTable(5) Then
        For i = LBound(aForeignTables()) To UBound(aForeignTables())
            stCondition = stCondition + ""ID"" NOT IN (SELECT "" +
aForeignTables(i,1) + "" FROM "" + aForeignTables(i,0) + "") AND "
        Next
    Else
        For i = LBound(aForeignTables2()) To UBound(aForeignTables2())
            stCondition = stCondition + ""ID"" NOT IN (SELECT "" +
aForeignTables2(i,1) + "" FROM "" + aForeignTables2(i,0) + "") AND "
        Next
    End If
```

Poslední AND musí být odstraněn, protože jinak by instrukce delete končila AND.

```
stCondition = Left(stCondition, Len(stCondition) - 4)
stSql = "DELETE FROM "" + stAuxilTable + "" WHERE " + stCondition +
""
oSQL_Statement.executeQuery(stSql)
```

Vzhledem k tomu, že tabulka již byla jednou vyčištěna, lze index tabulky zkontrolovat a případně opravit směrem dolů. Viz podprogram popsáný v jedné z předchozích částí.

```
Table_index_down(stAuxilTable)
```

Poté lze v případě potřeby aktualizovat pole seznamu ve formuláři, z něhož bylo dialogové okno Table_purge vyvoláno. V některých případech je třeba celý formulář znovu načíst. Za tímto účelem se na začátku podprogramu určí aktuální záznam, aby bylo možné po obnovení formuláře aktuální záznam obnovit.

```
oDlg.endExecute() 'End dialog ...
oDlg.Dispose() '... and remove from storage
End Sub
```

Dialogová okna jsou ukončena příkazem endExecute() a zcela odstraněna z paměti příkazem Dispose().

Psaní maker pomocí Access2Base

Verze LibreOffice od verze 4.2 mají integrovanou Access2Base. Tato knihovna zavádí vrstvu Basicu se svým specifickým API mezi kód uživatele a obvyklé rozhraní UNO. Poskytované API samo o sobě nepřináší nové funkce, ale v mnoha případech je čitelnější, stručnější a snadněji použitelné než UNO.

Rozhraní API vypadá velmi podobně jako rozhraní navržené společností Microsoft pro software Access. Databáze Base a Access mají mnoho společného, ale rozhodně ne jejich nativní programovací styly. Access2Base tuto mezeru vyplňuje.

Anglickou dokumentaci s příklady nalezneme na adrese www.access2base.com.

Stručně ilustrujeme, jak Access2Base skrývá složitost UNO:

- Vlastnost (Access2Base simple) *Value* ovládacího prvku má v UNO v závislosti na typu ovládacího prvku nebo na jeho umístění ve formuláři ekvivalenty v podobě ovládacího prvku mřížky nebo dialogového okna: *CurrentValue*, *Date*, *EffectiveValue*, *HiddenValue*, *ProgressValue*, *RefValue*, *ScrollValue*, *SpinValue*, *State*, *StringItemList*, *Text*, *Time*, *ValueItemList* nebo ... *Value*.
- Chceme-li získat N prvních záznamů tabulky nebo dotazu do pole Basic, můžeme jednoduše použít metodu *GetRows(N)* na objektu *Recordset*. Porovnejme s metodami *getString*, *getNull*, *getDouble*, *getLong*, ... v UNO, které bychom měli použít na pole v závislosti na jejich typu a použitém databázovém systému.

Access2Base zpracovává dvě hlavní kategorie objektů, které se zaměřují buď na:

- Uživatelské rozhraní. Typickými třídami těchto objektů jsou: *Form*, *SubForm*, *Dialog*, *Control*, *CommandBar*, *CommandBarControl* a *Event*. Jejich metody se obvykle volají z aplikace Base.
- Přístupy k databázi. Typickými třídami těchto objektů jsou: *Database*, *TableDef*, *QueryDef*, *Recordset* a *Field*. Jejich metody se vyvolávají buď z aplikace Base nebo z jakékoli jiné aplikace LibreOffice.

API Access2Base se tradičně vyvolává z jazyka Basic. Od verze LibreOffice 6.4 umožňuje brána přístup k rozhraní API také ze skriptů Python, a to bez omezení oproti Basicu. V jedné aplikaci lze bezproblémově integrovat skripty Basicu a Pythonu, a to i sdílením stejných instancí objektů.

V následujících odstavcích budou všechny příklady uvedeny jak v jazyce Basic, tak v jazyce Python. Jsou zcela rovnocenné.

Chceme-li ke knihovně přistupovat z aplikace Base, připojíme následující postup k události *OpenDocument* souboru Base:

(BASIC)

```
Sub DBOpen(Optional oEvent As Object)
    If GlobalScope.BasicLibraries.HasByName("Access2Base") Then
        GlobalScope.BasicLibraries.LoadLibrary("Access2Base")
    End If
    Call Application.OpenConnection(ThisDatabaseDocument)
End Sub
```

(PYTHON)

```
from access2base import *
def DBOpen(event = None):
    Application.OpenConnection()
```

```
g_exportedScripts = (DBOpen, )
```

Chceme-li získat přístup k databázi z jiné aplikace než Base, můžeme také spustit:

(BASIC)

```
Function DBOpen() As Object
```

```
    If GlobalScope.BasicLibraries.hasByName("Access2Base") then
```

```
        GlobalScope.BasicLibraries.loadLibrary("Access2Base")
```

```
    End If
```

```
    Set myDb = Application.OpenDatabase(" ... database file name ... ")
```

```
End Function
```

(PYTHON)

```
from access2base import *
```

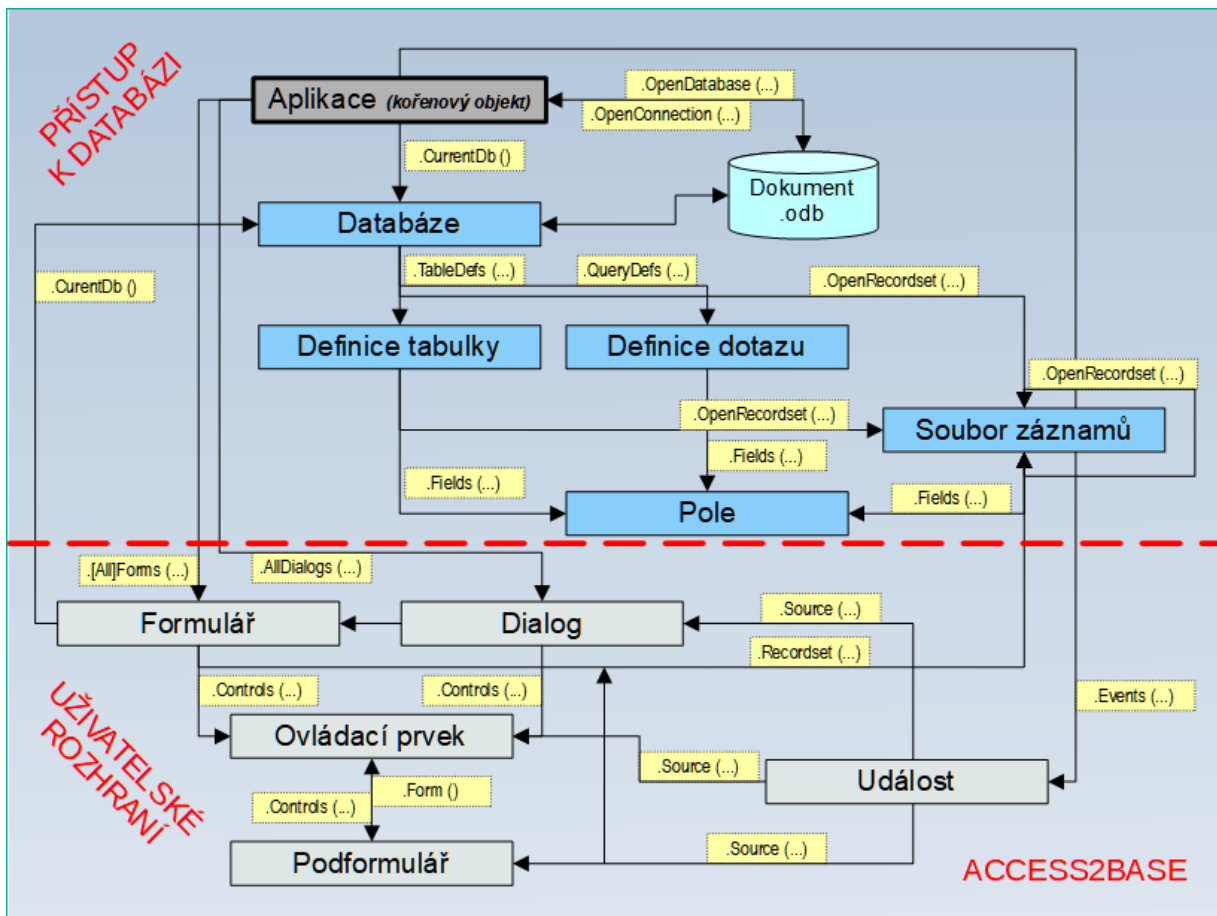
```
def DBOpen():
```

```
    return Application.OpenDatabase(' ... database file name ... ')
```

Cílem této knihy není kopírovat dokumentaci výše uvedených webových stránek. V tomto dokumentu se omezíme na shrnutí hlavních konceptů rozhraní API.

Objektový model

Na obrázku 440, počínaje kořenovým objektem *Aplikace*, je schéma popisující navigaci po nejpoužívanějších objektech.



Obrázek 440: Objektový model Access2Base

Příklad pro usnadnění čtení schématu:

- Aplikace je jeden ze dvou hlavních kořenových objektů.
- Metody `CurrentDb()` a `OpenDatabase()` poskytují objekt **Database**.
- Kolekce `Database.TableDefs()` obsahuje seznam všech tabulek uložených v databázi. Každá tabulka je reprezentována instancí objektu **TableDef**.
- Kolekce `TableDef.Fields()` obsahuje seznam všech polí, která se nacházejí v databázové tabulce. Každé pole je reprezentováno instancí objektu **Pole**.
- Pole mohou být také potomky uložených dotazů (**QueryDefs**) nebo datových sad (**Recordsets**) se stejnými atributy.

Několik příkladů

Tisk seznamu názvů tabulek a polí

(BASIC)

```
Sub ScanTables()  
Dim oDatabase As Object, oTable As Object, oField As Object  
Dim i As Integer, j As Integer  
Set oDatabase = Application.CurrentDb()  
With oDatabase  
    For i = 0 To .TableDefs.Count - 1  
        Set oTable = .TableDefs(i) ' Získání jednotlivých definic  
tabulek  
        DebugPrint oTable.Name  
        For j = 0 To oTable.Fields.Count - 1  
            Set oField = oTable.Fields(j) ' Získáme jednotlivá pole  
            DebugPrint "", oField.Name, oField.TypeName  
        Next j  
    Next i  
End With  
End Sub
```

(PYTHON)

```
def ScanTables():  
    oDatabase = Application.OpenDatabase("/home/somedir/TT NorthWind.odbc")  
    for oTable in oDatabase.TableDefs():  
        DebugPrint(oTable.Name)  
        for oField in oTable.Fields():  
            DebugPrint("", oField.Name, oField.TypeName)
```

Uložení dat vytvořených dotazem do pole jazyka Basic nebo do n-tice jazyka Python.

(BASIC)

```
Sub LoadQuery()
```

```
Dim oRecords As Object, vData As Variant
    Set oRecords = Application.CurrentDb().OpenRecordset("myQuery")
    vData = oRecords.GetRows(1000)
    orecords.mClose()
End Sub
```

(PYTHON)

```
def LoadQuery():
    oRecords= Application.CurrentDb().Openrecordset("myQuery")
    vData = orecords.GetRow(1000)
    oRecords.Close()
```

Nastavení výchozích hodnot v položkách formuláře

Chceme-li určit, že po každém záznamu bude nějaký ovládací prvek předvyplněn poslední nastavenou hodnotou, přiřadíme další rutinu události *Po změně záznamu* formuláře:

(BASIC)

```
Sub SetDefaultNewRec(poEvent As Object)
Dim oForm As Object, oControl As Object
    Set oForm = Application.Events(poEvent).Source ' Get the current
form
    Set oControl = oForm.Controls("txtCountry")
    oControl.DefaultValue = oControl.Value
End Sub
```

(PYTHON)

```
def SetDefaultNewRec(poEvent):
    oForm = Application.Events(poEvent).Source
    oControl = oForm.Controls("txtCountry")
    oControl.DefaultValue = oControl.Value
```

Databázové funkce

Pro zkrácení přístupu k hodnotám databáze na jeden řádek je k dispozici kolekce funkcí: *DLookup*, *DMax*, *DMin*, *Dsum*. Všechny přijímají stejné argumenty: název pole nebo výraz založený na názvech polí, název tabulky nebo dotazu a klauzuli SQL-where bez klíčového slova *WHERE*. Například:

(BASIC)

```
Function Lookup(psField As String, psSearchField As String,
psSearchValue As String) As Variant
    Lookup = Application.DLookup(psField, "myTable", _
        psSearchField & "=" & psSearchValue & "")
End Function
```

(PYTHON)

```
def Lookup(psField, psSearchField, psSearchValue):
```

```
return Application.Dlookup(psField, "myTable"  
    , psSearchField + "=" + psSearchValue + "'")
```

Speciální příkazy

Třída *DoCmd* (2. kořenová třída) nabízí sadu praktických funkcí, které umožňují provádět v jednom příkazu jazyka Basic složité, i když časté a praktické akce. Jmenujme alespoň některé:

Tabulka 50: Třída *Access2Base DoCmd*, speciální členské funkce

Funkce	Popis
CopyObject	Kopírování tabulky nebo dotazu v rámci stejné databáze nebo mezi dvěma databázemi.
OpenSQL	Provedení zadaného SQL příkazu SELECT a zobrazení výsledku v datovém listu.
OutputTo	Uložení dat z tabulky nebo dotazu do souboru HTML. Uložení skutečného obsahu formuláře do souboru PDF.
SelectObject	Aktivace daného okna (formuláře, sestavy, ...)
SendObject	Pošleme e-mailem s daným formulářem v příloze.

Objekt „Basic“ v jazyce Python

Z jazyka Python byl do brány *Access2Base* zaveden další objekt, nazvaný jednoduše „Basic“, který umožňuje spouštět řadu známých vestavěných funkcí jazyka Basic. Jsou vyvolány přesně se stejnými argumenty a chovají se v obou prostředích striktně stejně. Patří mezi ně:

- Systémové funkce: *Basic.ConvertToUrl* a *Basic.ConvertFromUrl*, *Basic.GlobalScope*, *Basic.CreateUnoService*.
- Funkce uživatelského rozhraní: *Basic.MsgBox*, *Basic.InputBox*
- Funkce data: *Basic.DateAdd*, *Basic.DateDiff*
- Introspekce objektů UNO: *Basic.Xray*

Knihovna ScriptForge

Programátoři maker často potřebují provádět úlohy, jako je vytváření a otevírání souborů, přístup k ovládacím prvkům formulářů, čtení dat z databází vložených do dokumentů Base atd. Cílem knihovny *ScriptForge* je usnadnit provádění takových příkazů, aniž by bylo nutné se učit potřebná rozhraní API (Application Programming Interfaces) a příkazy *LibreOffice*, což může být pro příležitostné programátory obtížné.

Knihovna *ScriptForge* je uspořádána do sady služeb, z nichž každá poskytuje metody a vlastnosti související s určitým tématem. Například služba *Dialog* umožňuje přístup k dialogovým oknům dostupným ve skriptovacích modulech a služba *Databáze* umožňuje provádět příkazy SQL v dokumentech Base. Mnoho metod je společných pro různé služby.

Od verze 7.2 je anglická verze dokumentace knihoven *ScriptForge* plně integrována do nápovědy *LibreOffice*

(https://help.libreoffice.org/7.3/en-US/text/sbasic/shared/03/lib_ScriptForge.html?DbPAR=BASIC).

Tabulka 51: Služby poskytované knihovnou *ScriptForge*

Kategorie	Služby	
LibreOffice Basic	Array Dictionary Exception	FileSystem String TextStream
Obsah dokumentu	Base Calc Chart	Database Document
Uživatelské rozhraní	Dialog DialogControl Form	FormControl PopupMenu UI
Nástroje	Basic L10N Platform	Services Session Timer

Ukázka ScriptForge s Base pomocí Pythonu

Tato část je stručnou ukázkou některých hlavních služeb, které jsou pro Base zajímavé, s využitím jazyka Python. Tato ukázka přesně navazuje na příklady nápovědy, na které se lze odkázat v případě základního jazyka. Pro práci s dokumenty Base je zapotřebí řada služeb, včetně: UI (User Interface), FileSystem, Base, Database, Form a FormControl. Na podporu demonstrace budou využity další služby.

Vytváření skriptů Python pomocí ScriptForge

Potřebujeme funkční prostředí Pythonu, takže použijeme nápovědu a projdeme si [Vytváření skriptů Pythonu pomocí ScriptForge](#).

V první části je vysvětlena řada rozdílů mezi spuštěním ScriptForge v jazycích Basic a Python. Python nemá v LibreOffice vestavěné IDE, takže je třeba jej nainstalovat a určit umístění Moje makra. Klíčovou částí je Spuštění skriptů uvnitř procesu LibreOffice:

- 1) Použití rozšíření APSO
- 2) Vytváření souborů skriptů Python

Pokud makro `increment_cell` proběhlo, úspěšně jsme vytvořili funkční prostředí Pythonu se ScriptForge. Pokud tomu tak není, vrátíme se zpět k jednotlivým krokům, dokud nebude fungovat.

Inicializace ukázky

Stejný postup lze použít i pro vývoj demonstračního modulu. Pro spuštění skriptů z LibreOffice musíme otevřít dokument, stačí nový tabulkový procesor bez názvu, protože nemusí být uložen. Dále vytvoříme soubor `SF_Base_Demo.py` Module přímo v sekci Moje makra.

Je efektivnější umístit společné deklarace na začátek modulu. Všechna makra používají knihovnu ScriptForge, takže modul začneme importem metody `CreateScriptService`, aby bylo možné přistupovat ke službám knihovny. Mnoho maker zobrazuje okno se zprávou a používá okna aplikací LibreOffice, takže je také definujeme:

```
# Ukázkový modul ScriptForge aplikace Base
from scriptforge import CreateScriptService
bas = CreateScriptService("Basic")
ui = CreateScriptService("UI")
```

Demonstrace potřebuje funkční složku, ale různé operační systémy používají různé souborové systémy. Naštěstí se SourceForge o tuto složitost stará pomocí služby FileSystem. Stačí iniciovat službu a přidat proměnnou pro pracovní cestu, další podrobnosti jsou uvedeny v dokumentaci.

```
fs = CreateScriptService("FileSystem")
fs.FileNameing = "SYS"
workpath = fs.BuildPath(fs.HomeFolder, "demo")
```

První makro slouží k rychlému otestování, zda SourceForge funguje správně, takže přidáme níže uvedené makro sf_HomeFolder a spustíme jej, aby se v systému zobrazila domovská složka. Toto je poslední připomínka k aktualizaci g_exportedScripts.

```
def sf_HomeFolder(args=None):
    bas.MsgBox("Home Folder: " + str(fs.HomeFolder))

g_exportedScripts = (sf_HomeFolder, )
```

Vytvoření, otevření a používání databáze

Jediným způsobem, jak vytvořit základní soubor, je metoda služby uživatelského rozhraní CreateBaseDocument. Možnosti nalezneme v dokumentaci SourceForge, ale pokud cesta neexistuje, bude vytvořena spolu se souborem a stávající soubor bude bez varování přepsán. Spustíme následující příklad:

```
def CreateBaseDocumentUI(workpath = workpath):
    doc = ui.CreateBaseDocument(fs.BuildPath(workpath, "myDB.odt"))
    bas.MsgBox("Created now closing: " + fs.BuildPath(workpath,
"myDB.odt"))
    doc.CloseDocument()
```

Jakmile soubor Base existuje, lze k otevření souboru Base použít metodu služby uživatelského rozhraní OpenBaseDocument. Spustíme toto makro:

```
def OpenBaseDocumentUI(workpath = workpath):
    ui.OpenBaseDocument(fs.BuildPath(workpath, "myDB.odt"), \
        macroexecution = ui.MACROEXECALWAYS)
```

Otevřený základní dokument lze aktivovat pomocí metody GetDocument služby uživatelského rozhraní. Soubor myDB.odt by měl být stále otevřený, takže jej minimalizujeme, ale nezavíráme. Pokud je zavřený, stačí jej znovu otevřít a minimalizovat. Pro aktivaci spustíme následující makro:

```
def GetDocumentUI(workpath = workpath):
    myDoc = ui.GetDocument(fs.BuildPath(workpath, "myDB.odt"))
    bas.MsgBox("Bye!")
    myDoc.CloseDocument(True)
```

Služba Database také umožňuje přímý přístup k databázi bez dokumentu Base. To umožňuje přístup k tabulkám a dotazům a k datům, která obsahují. V dokumentu Base není přístup k formulářům a sestavám, ke kterým je nutné přistupovat prostřednictvím služeb Base. Spustíme níže uvedené makro, které vrátí data přímo z databáze:

```
def database(workpath = workpath):
    myDatabase = CreateScriptService("Database", fs.BuildPath(workpath,
"myDB.odt"))
    # Spouštění dotazů, příkazů SQL,
    bas.MsgBox("Closing: " + fs.BuildPath(workpath, "myDB.odt"))
```

```
myDatabase.CloseDatabase()
```

Soubor myDB.odt ve složce demo ve vaší domovské složce již nebudeme potřebovat, takže jej můžeme odstranit.

Ukázková databáze zaměstnanců

Nyní přejdeme k demonstraci. Chceme-li používat formuláře v ukázce bez jejich nastavení, stačí stáhnout soubor employees.odt do složky s ukázkou v naší domovské složce. Je čas otevřít databázi, a proto vytvoříme další makro a spustíme jej, čímž povolíme makra:

```
def open_database(workpath = workpath):  
    doc = ui.OpenBaseDocument(fs.BuildPath(workpath, "Employees.odt"))  
    # Uživatel a heslo jsou v případě potřeby uvedeny níže.  
    myDatabase = doc.GetDatabase()
```

Soubor employees.odt z příručky Začínáme s aplikací Base nejspíše známe jako databázi Automobile.odt s dodatečným formulářem sf_demo pro spouštění maker. Od této chvíle jej můžeme používat místo APSO.

Zadáme a spustíme následující makro pro načtení dat do databáze. Tato ukázka není určena k výuce jazyka SQL, pouze demonstruje, že jej lze spustit pomocí databázové služby s metodou RunSql. S daty můžeme experimentovat. Při každém spuštění tohoto makra příkaz Drop Table odstraní tabulku, pokud existuje, před vytvořením tabulky a načtením dat.

```
# SQL Demo - Vytvoření tabulky s testovacími daty  
def create_table_with_values(args=None):  
    # Příprava SQL - Odstranit; Vytvořit; Přidat data  
    mysql = ""  
Drop Table "EmployeeData" IF EXISTS;  
  
CREATE TABLE "EmployeeData" (  
    "ID" INT IDENTITY,  
    "FirstName" VARCHAR(50),  
    "LastName" VARCHAR(50),  
    "Position" VARCHAR(50),  
    "Salary" DECIMAL(10, 2),  
    "City" VARCHAR (50)  
);  
  
-- If no Field_name specified, all fields must be completed in right  
order  
INSERT INTO "EmployeeData"  
VALUES  
    (NULL, 'Paul', 'Mejia', 'Family Therapist', 50000, 'Hogworth');  
  
INSERT INTO "EmployeeData"  
VALUES
```

```

(NULL, 'Jordin', 'Wise', 'Sales', 83954, 'Springstead');

INSERT INTO "EmployeeData"
VALUES
(NULL, 'Kaitlin', 'Palmer', 'Manager', 83250, 'Chicago');

INSERT INTO "EmployeeData"
VALUES
(NULL, 'Paul', 'Smith', 'Mechanic', 74600, 'Goldview');

INSERT INTO "EmployeeData"
VALUES
(NULL, 'Conrad', 'Palmer', 'Manager', 115000, 'North Dodgestead');

INSERT INTO "EmployeeData"
VALUES
(NULL, 'Kate', 'Wordsmith', 'Sales', 75000, 'Chicago');
"""
# get base document
# doc = CreateScriptService("Document", bas.ThisDatabaseDocument)
doc = ui.GetDocument(bas.ThisDatabaseDocument)
# Vytvoří databázovou službu s aktuální databází
myDatabase = doc.GetDatabase()
# Spustí SQL
myDatabase.RunSql(mysql, directsql=True)
# nefunguje
doc.RunCommand("DBRefreshTables")
myDatabase.CloseDatabase()
# Zobrazení zprávy
bas.MsgBox("Table Created")

```

Statistiky jsou běžně potřebné z databází, takže použijme několik metod Database pro tabulku Employees. Nejprve získáme přístup k právě načteným datům pomocí **View > Obnovit tabulky**.

```

# Výpočet statistik zaměstnanců
def table_stats(args=None):
    # Vytvoří databázovou službu s aktuální databází
    doc = ui.GetDocument(bas.ThisDatabaseDocument)
    Employees = doc.GetDatabase()
    # Pro alternativní zobrazení řetězce
    item_a = "Employee Count"

```

```

# Metoda Dcount
item_b = str(Employees.DCount("[ID]", "EmployeeData"))
bas.MsgBox("Employee Count" + "\n" + item_b.center(len(item_a), "_"))
# Metoda DSum - Selektivní
bas.MsgBox("Total Managers Salary - " + str(Employees.Dsum( \
    "[Salary]", "EmployeeData", "[Position] = 'Manager'")))

```

Podrobnosti o jednom záznamu lze vrátit pomocí služby Dlookup. Všimneme si, že pokud je v dotazu vráceno více záznamů, vrátí pouze první záznam.

```

def get_first_match(args=None):
    # Vytvoří databázovou službu s aktuální databází
    myDB = CreateScriptService("Database", bas.ThisDatabaseDocument.URL,
    "", False)
    # Metoda Dlookup
    bas.MsgBox(
        "Select First Occurance FirstName with" + "\n"
        + "Descending Seq of FirstName" + "\n"
        + "and LastName like Smith" + "\n\n"
        + (myDB.DLookup("[FirstName]", "EmployeeData", \
            Criteria = "[LastName] LIKE LOWER ('%Smith%')", \
            OrderClause = "[FirstName] DESC"))
    )

```

Služba Database nakonec umožňuje načíst celou tabulku do pole pomocí metody GetRows.

```

# Získat záznamy o zaměstnancích
def get_employee_records(args=None):
    # Vytvoří databázovou službu s aktuální databází
    doc = ui.GetDocument(bas.ThisDatabaseDocument)
    Employees = doc.GetDatabase()
    # queryResults je vrácené pole
    # záhlaví jsou názvy sloupců
    queryResults = Employees.GetRows("EmployeeData", header=True)
    bas.MsgBox(
        "Retrieve table data; Show:" + "\n"
        + "FirstName of first record" + "\n\n"
        + (queryResults[1][1])
    )
    # queryResults je vrácené pole
    queryResults = Employees.GetRows(
        "SELECT * FROM EmployeeData ORDER BY [FirstName]", maxrows=5
    )
    bas.MsgBox(

```



```

    "Retrieve table data; Show:" + "\n"
    + "FirstName of first record" + "\n"
    + "Select statement ordered by FirstName" + "\n\n"
    + (queryResults[0][1])
)

```

Většina služeb SourceForge Base se týká formulářů. Formuláře se používají v případech, kdy je přímý zápis do tabulky nepohodlný, k rychlému zachycení chyb při zadávání dat nebo v případech, kdy příliš mnoho tabulek znemožňuje přímou správu dat. Další makro demonstruje použití formuláře s podformulářem:

```

# get_form a subform - zobrazení dat z každého z nich
def get_form(args=None):
    # Vytvoří službu DB pomocí aktuálního dokumentu
    doc = CreateScriptService("Document", bas.ThisDatabaseDocument)
    # Níže uvedené prohlášení je nutné pouze v případě, že formulář ještě
    # nebyl otevřen
    form_doc = doc.OpenFormDocument("Fuel")
    # Formuláře - V dokumentu `Fuel` přistupujeme k `Mainform`
    form = doc.Forms("Fuel", "MainForm")
    # získání kontroly nad `MainForm`
    form_control = form.Controls("fmtOdometer")
    # Zobrazení hodnoty pole
    bas.MsgBox("Odometer - " + str(form_control.Value))
    # Získání dílčího formuláře pomocí hlavního formuláře
    subform = form.SubForms("SubForm")
    # Získání ovládací mřížky tabulky
    subform_control = subform.Controls("SubForm_Grid")
    # Získání sloupce v ovládacím prvku mřížky
    grid_control = subform_control.Controls("FuelCost")
    # Zobrazení hodnoty pole ve vybraném záznamu
    bas.MsgBox("First FuelCost in Grid - " + str(grid_control.Value))
    # Zavření formuláře
    form.CloseFormDocument()

def get_form2(args=None):
    bas = CreateScriptService("Basic")
    # Vytvoří službu DB pomocí aktuálního dokumentu
    doc = CreateScriptService("Document", bas.ThisDatabaseDocument)
    # Níže uvedené prohlášení je nutné pouze v případě, že formulář ještě
    # nebyl otevřen
    form_doc = doc.OpenFormDocument("Fuel")

```

```
# Formuláře - V dokumentu `Fuel` přistupujeme k `Mainform`
form = doc.Forms(thisFormDocument, "MainForm")
# získání kontroly nad `MainForm`
form_control = form.Controls("fmtOdometer")
```

Službu FormEvent lze použít k přístupu k podrobnostem ovládacího prvku FormControl, který událost vyvolal. Další makro je přiřazeno tlačítku formuláře a spuštěno pomocí akce execute. Všimneme si, že dokument formuláře není ve ScriptForge přímo k dispozici, ale je k dispozici jeho název, který se používá k vyhledání dokumentu formuláře.

```
# zavření formulář
def close_form(evt): # přiřazení vykonání akce tlačítku formuláře události makra
    control = CreateScriptService("FormEvent", evt) # tlačítko
    form = control.Parent
    fdoc_name = form.BaseForm if CreateScriptService("Platform" \
).OfficeVersion.split()[1] >= "7.3.2" else "sf_demo"
    doc = CreateScriptService("Document", bas.ThisDatabaseDocument)
    fdoc = doc.Forms(fdoc_name, form.Name)
    # Zavření formuláře
    fdoc.CloseFormDocument()
```

Dokument lze přímo zavřít:

```
# zavření aktuálního dokumentu
def close_doc(args=None):
    # Vytvoří službu DB pomocí aktuálního dokumentu
    doc = CreateScriptService("Document", bas.ThisDatabaseDocument)
    # Zavření aktuálního dokumentu
    doc.CloseDocument(True)
```

Další zdroje

Podrobnosti o službách a další příklady najdeme v dokumentaci ScriptForge nebo v nápovědě LibreOffice. Můžeme si stáhnout úplnější verzi této ukázky.



LibreOffice
Community



Příručka aplikace Base 7.3

Kapitola 10

Údržba databáze

Obecné poznámky k údržbě databází

Dynamické databáze – zejména ty, které často mažou a mění data – mají dva problematické negativní efekty. Za prvé, databáze se neustále zvětšuje, i když ve skutečnosti nemusí obsahovat více dat. Za druhé, automaticky vytvořený primární klíč se nadále inkrementuje bez ohledu na to, jaká je hodnota dalšího potřebného klíče. V této kapitole jsou popsány důležité úkoly údržby databáze, které je třeba vzít v úvahu pro dobrý výkon a správu.

Komprimace databáze

Databáze HSQLDB se chovají tak, že zachovávají úložný prostor i pro smazané záznamy. Databáze naplněné testovacími daty, zejména obrázky, si zachovávají stejnou velikost, i když jsou všechny tyto záznamy následně odstraněny. Důvodem je vlastnost primárních klíčů každé tabulky. Soubor databázových dokumentů obsahuje poslední hodnotu použitou pro každý primární klíč. Při vytvoření záznamu v tabulce se přiřadí další hodnota klíče.

Aby se uvolnilo místo v úložišti, je třeba přepsat záznamy v databázi (tabulky, popisy tabulek atd.). To lze provést otevřením každé tabulky a odstraněním všech jejích záznamů. Při práci s propojenými tabulkami je třeba dbát zvýšené opatrnosti.

Pomocí **Nástroje > Relace** určíme, ze které tabulky mají být data odstraněna. Podívejme se na dvě tabulky. Tabulka, jejíž primární klíč je součástí vztahu, je tabulka, jejíž data je třeba odstranit. Zavřete dialogové okno Relace. V hlavním okně databáze vybereme ikonu Tabulky. Poté dvakrát klepneme na tabulku a zobrazíme její data. Vymažeme její data. Uložíme tabulku a poté databázi. Poté je třeba tyto změny zapsat do souboru dokumentu databáze. Chceme-li to provést, zavřeme LibreOffice. Tím se také zkomprimují databázové soubory.

Pokud budeme LibreOffice znovu používat, zavřeme jej a znovu otevřeme.

Obnovení automatických hodnot

Při vytváření databáze a následném testování všech možných funkcí pomocí příkladů a prováděných opravách, dokud vše nebude fungovat, budeme mít pravděpodobně primární klíče s hodnotami výrazně vyššími, než byla původní hodnota před uvedením databáze do produkce. Často jsou primární klíče skutečně nastaveny na automatický přírůstek. Pokud jsou tabulky vyprázdněny v rámci přípravy na běžné používání nebo před předáním databáze jiné osobě, primární klíč se nadále inkrementuje z aktuální pozice, místo aby se vynuloval.

Následující příkaz SQL, zadaný pomocí **Nástroje > SQL**, umožňuje resetovat počáteční hodnotu:

```
ALTER TABLE "Table_name" ALTER COLUMN "ID" RESTART WITH New value
```

To předpokládá, že pole primárního klíče má název ID a bylo definováno jako pole s automatickou hodnotou. Nová hodnota by měla být ta, která se má automaticky vytvořit pro další nový záznam. Pokud se tedy například aktuální záznamy zvýší na 4, nová hodnota by měla být 5, aniž by se změnilo pole ID. První hodnotou ID bude *New value* ve výše uvedeném příkazu SQL.

Dotazování na vlastnosti databáze

Všechny informace o tabulkách databáze jsou uloženy ve formě tabulek v samostatné části HSQLDB. Do této samostatné oblasti se dostaneme pomocí názvu INFORMATION_SCHEMA.

Následující dotaz lze použít ke zjištění názvů polí, typů polí, velikostí sloupců a výchozích hodnot. Zde je příklad pro tabulku s názvem Searchtable.

```
SELECT "COLUMN_NAME", "TYPE_NAME", "COLUMN_SIZE", "COLUMN_DEF" AS "Default Value" FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" WHERE "TABLE_NAME" = 'Searchtable' ORDER BY "ORDINAL_POSITION"
```

Všechny speciální tabulky v HSQLDB jsou popsány v příloze A této knihy. Informace o obsahu těchto tabulek lze nejnázne získat přímými dotazy:

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS"
```

Hvězdička zajišťuje zobrazení všech dostupných sloupců tabulky. Výše hledaná tabulka poskytuje základní informace o primárních klíčích různých tabulek.

Tyto informace jsou užitečné především pro makra. Místo toho, aby bylo nutné poskytovat podrobné informace o každé čerstvě vytvořené tabulce nebo databázi, jsou napsány procedury, které tyto informace získávají přímo z databáze, a jsou proto univerzálně použitelné. V ukázkové databázi je to mimo jiné vidět v jednom z modulů údržby, kde se určují cizí klíče.

Export dat

Existuje mnohem jednodušší metoda exportu dat než standardní metoda exportu dat otevřením souboru *.odb. Přímo v rozhraní Base můžeme pomocí **Nástroje > SQL** zadat jednoduchý příkaz, který je v databázích serverů vyhrazen správci systému.

```
SCRIPT 'my_exported_database_file'
```

Tím se vytvoří kompletní výpis databáze SQL se všemi definicemi tabulek, relacemi mezi tabulkami a záznamy. Dotazy a formuláře nejsou extrahovány, protože byly vytvořeny v uživatelském rozhraní a nejsou uloženy v interní databázi. Zahrnutý jsou však všechny pohledy.



Poznámka

Tento postup lze použít k aktualizaci vložené databáze pro připojení k databázi pomocí HSQLDB 2.50. Opět je třeba nahradit dotazy a formuláře.

Ve výchozím nastavení je exportovaným souborem běžný textový soubor s názvem 'my_exported_database_file'. Soubor lze také poskytnout v binární nebo komprimované (zazipované) podobě, což je užitečné pro velké databáze. To však poněkud komplikuje jeho zpětný import do LibreOffice Base, viz níže.

Formát exportovaného souboru lze změnit pomocí jedné z hodnot nastavení SCRIPTFORMAT:

```
SET SCRIPTFORMAT {TEXT | BINARY | COMPRESSED};
```

Export souboru vyžaduje použití tohoto kódu SQL po jednotlivých řádcích:

```
SCRIPT 'my_exported_database_file';
```

```
SET SCRIPTFORMAT TEXT;
```

```
SHUTDOWN SCRIPT;
```

```
CHECKPOINT;
```

Tím se do domovské složky exportuje textový soubor *my_exported_database_file* s informacemi o databázi.



Poznámka

Ujistíme se, že 'my_exported_database_file' neexistuje ve složce, jinak se zobrazí chybová zpráva.

Soubor lze importovat pomocí **Nástroje > SQL** a vytvořit novou databázi se stejnými údaji. V případě interní databáze LibreOffice je třeba před importem odstranit ze souboru 'my_exported_database_file' následující řádky:

```
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
CREATE USER SA PASSWORD ""
GRANT DBA TO SA
SET WRITE_DELAY 60
SET SCHEMA PUBLIC
```

Tyto položky se týkají uživatelského profilu a dalších výchozích nastavení, která jsou již pro interní databáze LibreOffice nastavena. Pokud se některý z těchto řádků vyskytne, zobrazí se chybové hlášení. Nacházejí se přímo před obsahem, který bude vložen do tabulek pomocí příkazu `INSERT`.

Pro import tohoto souboru je třeba obsah rozdělit do několika textových souborů vytvořených jednoduchým programem pro úpravu textu. První soubor by měl obsahovat všechny příkazy pro vytváření tabulek a pohledů. Zkopírujeme všechny řádky od začátku s `CREATE TABLE` a zastavíme jeden řádek nad řádkem obsahujícím `INSERT INTO`. Toto vložíme do prvního souboru. Zkopírujeme a vložíme zbývající řádky do druhého souboru.

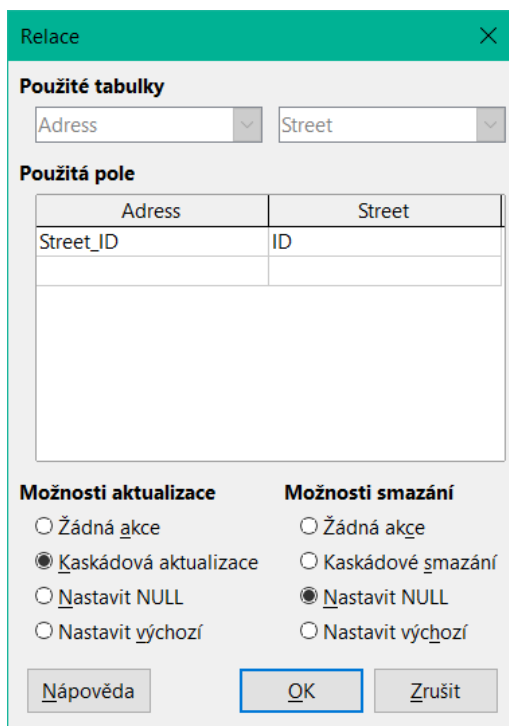
Velikost druhého souboru je však omezena: musí být menší než 65 KB. Pokud je větší, měl by být také rozdělen do menších textových souborů vyjmutím a vložením. Jen se ujistíme, že horní řádek každého z těchto nových souborů začíná `INSERT INTO`. Jedním ze způsobů, jak toho dosáhnout, je vyjímat odspodu nahoru až k takovému řádku.

Testování tabulek na nepotřebné položky

Databáze se skládá z jedné nebo více hlavních tabulek, které obsahují cizí klíče z jiných tabulek. V ukázkové databázi jsou to tabulky Media a Address. V tabulce Address se primární klíč PSČ vyskytuje jako cizí klíč. Pokud se osoba přestěhuje do nového domova, změní se i adresa. Výsledkem může být, že již neexistuje cizí klíč Postcode_ID odpovídající tomuto PSČ. V zásadě by tedy mohlo být vymazáno samotné poštovní směrovací číslo. Při běžném používání však není zřejmé, zda záznam již není potřeba. Existují různé způsoby, jak těmto problémům předcházet.

Testování záznamů pomocí definice vztahu

Při definování relací lze zajistit integritu dat. Jinými slovy, můžeme zabránit tomu, aby odstranění nebo změna klíčů vedla k chybám v databázi. Následující dialogové okno je přístupné prostřednictvím **Nástroje > Relace** a následným klepnutím pravým tlačítkem myši na spojnici mezi dvěma tabulkami.



Obrázek 441: Nastavení možností aktualizace a odstranění v dialogovém okně Relace

Zde jsou zohledněny tabulky Address a Street. Všechny zadané akce se vztahují na tabulku Address, která obsahuje cizí klíč Street_ID. Možnosti aktualizace se týkají aktualizace pole ID v tabulce Street. Pokud je číselný klíč v poli "Street". "ID" změněn, **Žádná akce** znamená, že databáze této změně odolá, pokud se v tabulce Address vyskytuje jako cizí klíč pole "Street". "ID" s tímto číslem klíče.

Kaskádová aktualizace znamená, že se číslo klíče jednoduše přenesou. Pokud má ulice "Burgring" v tabulce Street ID "3" a je také uvedena v položce "Address". "Street_ID", lze ID bezpečně změnit. Pokud je například změněna na "67", odpovídající hodnoty "Address". "Street_ID" se automaticky změny na "67".

Pokud je vybrána možnost **Nastavit null**, změnou ID se z pole "Address". "Street_ID" stane prázdné pole.

S možnostmi Odstranit se pracuje podobně.

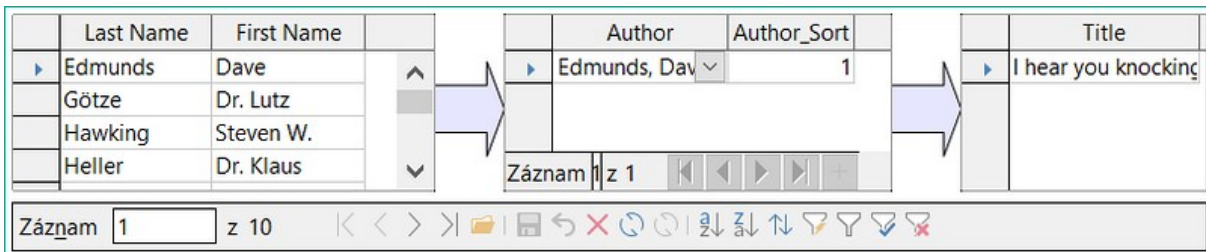
U obou možností grafické uživatelské rozhraní v současné době neumožňuje možnost **Nastavit výchozí**, protože výchozí nastavení grafického uživatelského rozhraní se liší od nastavení databáze. Viz kapitola 3, Tabulky.

Definování relací pomáhá udržovat samotné relace čisté, ale neodstraňuje zbytečné záznamy, které poskytují svůj primární klíč jako cizí klíč v relaci. Může existovat libovolný počet ulic bez odpovídajících adres.

Úprava záznamů pomocí formulářů a podformulářů

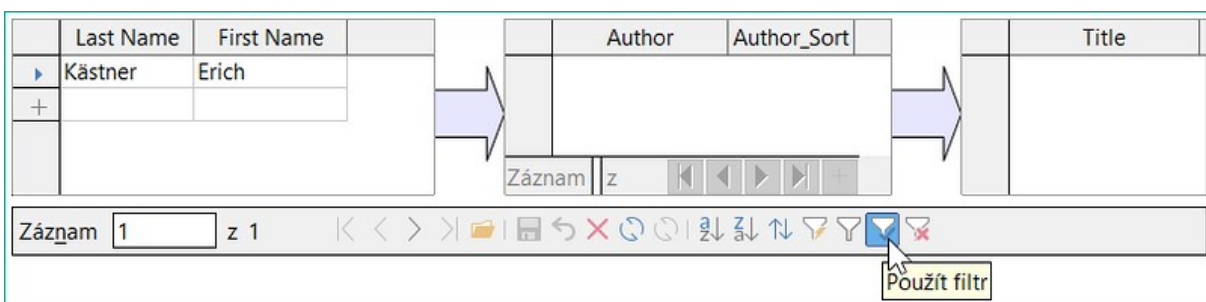
V zásadě lze ve formulářích zobrazit celou vzájemnou relaci mezi tabulkami. To je samozřejmě nejjednodušší, pokud je tabulka spojena pouze s jednou další tabulkou. V následujícím příkladu se tedy primární klíč autora stane cizím klíčem v tabulce rel_Media_Author. rel_Media_Author obsahuje také cizí klíč z Media, takže následující uspořádání ukazuje relaci n:m se třemi formuláři. Každá z nich je uvedena v tabulce.

Obrázek 442 ukazuje, že titlu *I hear you knocking* patří autorovi *Dave Edmunds*. Proto nesmí být *Dave Edmunds* smazán – jinak budou chybět informace potřebné pro médium *I hear you knocking*. Pole se seznamem však umožňuje vybrat jiný záznam místo *Dave Edmunds*..



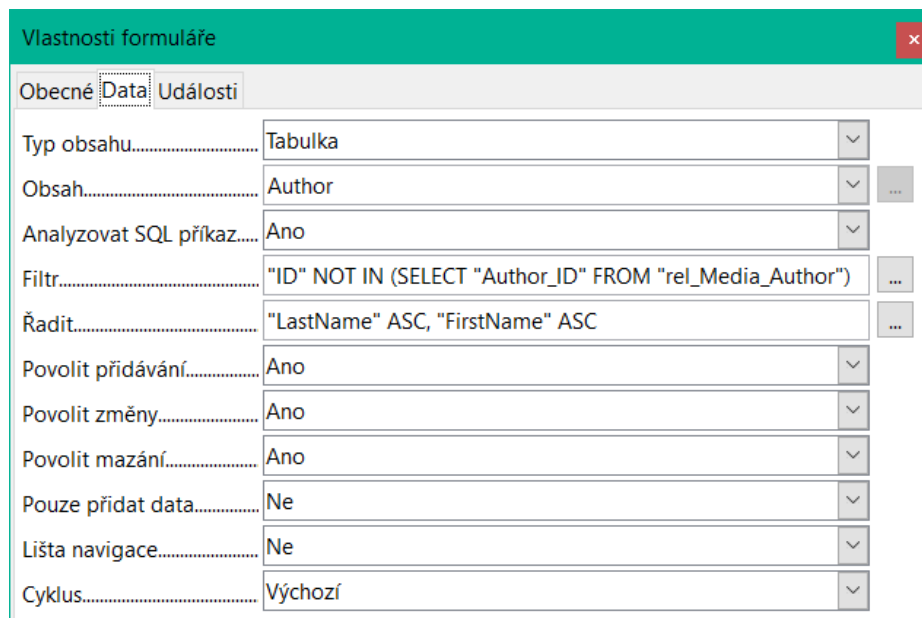
Obrázek 442: Záznam pro *Dave Edmunds* by neměl být z tabulky *Author* odstraněn.

Ve formuláři je vestavěný filtr, jehož aktivací můžeme zjistit, které kategorie nejsou v tabulce *Media* potřeba. V právě popsaném případě se používají téměř všechny vzorky autorů. Pouze záznam *Erich Kästner* může být vymazán bez jakýchkoli následků pro ostatní záznamy v tabulce *Media*.



Obrázek 443: Záznam pro *Ericha Kästnera* mohl být odstraněn z tabulky *Author*.

Filtr je v tomto případě pevně zakódován. Nachází se ve vlastnostech formuláře. Takový filtr se aktivuje automaticky při spuštění formuláře. Lze jej vypnout a zapnout. Pokud je smazán, lze k němu znovu přistoupit úplným znovunačtením formuláře. To znamená víc než jen aktualizaci dat; celý dokument formuláře musí být uzavřen a poté znovu otevřen.



Obrázek 444: Pole filtru na kartě *Data* v dialogovém okně *Vlastnosti formuláře*

Dotazy na vyhledání osiřelých záznamů

Filtr zobrazený na obrázku 444 je součástí dotazu, který lze použít k vyhledání osiřelých záznamů.

```
SELECT "Surname", "Firstname" FROM "Author" WHERE "ID" NOT IN (SELECT "Author_ID" FROM "rel_Media_Author")
```

Pokud tabulka obsahuje cizí klíče z několika jiných tabulek, je třeba dotaz odpovídajícím způsobem rozšířit. To se týká například tabulky Town, která má cizí klíče v tabulce Media i v tabulce Postcode. Proto by se na záznamy v tabulce Town, které mají být odstraněny, nemělo odkazovat v žádné z těchto tabulek. To se zjistí následujícím dotazem:

```
SELECT "Town" FROM "Town" WHERE "ID" NOT IN (SELECT "Town_ID" FROM "Media") AND "ID" NOT IN (SELECT "Town_ID" FROM "Postcode")
```

Osiřelé záznamy pak lze odstranit výběrem všech záznamů, které prošly nastaveným filtrem, a použitím možnosti Odstranit v místní nabídce ukazatele záznamu, vyvolané klepnutím pravým tlačítkem myši.

Rychlost vyhledávání v databázi

Vliv dotazů

Právě tyto dotazy, použité v předchozí části k filtrování dat, se ukazují jako nevyhovující s ohledem na maximální rychlost prohledávání databáze. Problém spočívá v tom, že v rozsáhlých databázích získává poddotaz odpovídající velké množství dat, s nimiž je třeba porovnat každý jednotlivý zobrazitelný záznam. Pouze porovnání se vztahem IN umožňuje porovnat jednu hodnotu se souborem hodnot. Dotaz

```
... WHERE "ID" NOT IN (SELECT "Author_ID" FROM "rel_Media_Author")
```

může obsahovat velké množství možných cizích klíčů z tabulky rel_Media_Author, které je třeba nejprve porovnat s primárními klíči v tabulce Authors pro každý záznam v této tabulce. Takový dotaz proto není vhodný pro každodenní použití, ale může být potřebný pro údržbu databáze. Pro každodenní použití je třeba vyhledávací funkce konstruovat jinak, aby vyhledávání dat nebylo příliš dlouhé a nenarušovalo každodenní práci s databází.

Vliv pole se seznamy a rozevíracích seznamů

Čím více seznamů je do formuláře zabudováno a čím více dat obsahují, tím déle trvá načítání formuláře, protože tyto seznamy je třeba načíst a vytvořit.

Čím lépe program Base nastaví grafické rozhraní a zpočátku načte obsah pole se seznamem jen částečně, tím rychleji poběží.

Pole se seznamy se vytvářejí pomocí dotazů a tyto dotazy se musí spustit při načítání formuláře pro každé pole se seznamem.

Pokud se stejná struktura dotazu používá pro několik polí se seznamem, je lepší použít společné Pohledy, než opakovaně vytvářet pole se stejnou syntaxí pomocí uložených příkazů SQL v polích se seznamem. Pohledy jsou výhodné především pro externí databázové systémy, protože zde server běží výrazně rychleji než dotaz, který musí být sestaven grafickým uživatelským rozhraním a čerstvě odeslán na server. Server považuje Pohledy za dokončené místní dotazy.

Vliv použitého databázového systému

Interní databáze HSQLDB je nastavena tak, aby byla zajištěna dobrá spolupráce databází Base a Java. Při použití vestavěného systému správy databází Base, jako je například databázový

stroj HSQLDB, je velikost a odezva databáze omezená ve srovnání s použitím externího databázového stroje. Zejména pokud je tento databázový server spuštěn na samostatném počítači. Pokud se funkce naší databáze začnou zpomalovat, postupujeme nejprve podle pokynů v této příručce pro vyčištění prázdného místa, odstraněných nebo dočasných dat a zkontrolujeme, zda používáme indexy tam, kde to má smysl. Pokud se odezva nevyplatí, zvážíme přesunutí dat z dat Base ODB na externí databázový server.

Externí databáze běží výrazně rychleji. Přímé připojení k MySQL nebo PostgreSQL a připojení pomocí ODBC probíhají prakticky stejně rychle. JDBC také závisí na spolupráci s Javou, ale stále funguje rychleji než interní připojení pomocí HSQLDB.



Příručka aplikace Base 7.3

Příloha A

Běžné úlohy databáze

Úvod

Jak se naše databáze rozrůstá a začleňuje do našeho podnikání, bude nutné řešit administrativní úkoly, abychom databázi udrželi v rozumných a bezpečných podmínkách a mohli ji co nejlépe využívat. V této kapitole jsou uvedeny některé informace o datových typech, funkcích a uložených procedurách databáze, příkazech Base UNO, administrativních a opravných procedurách.

Čárové kódy

Aby bylo možné použít funkci tisku čárového kódu, musí být nainstalováno písmo `ean13.ttf`. Toto písmo je volně dostupné. Například jako v následující adrese vrácené vyhledávačem: <https://graphicore.github.io/librebarcode/documentation/ean13.html>

Čárové kódy EAN13 lze vytvořit pomocí souboru `ean13.ttf` takto:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Číslo	Velká písmena, A=0 B=1 atd.						*	Malá písmena, a=0 b=1 atd.						+

Obrázek 445: Schematické uspořádání třináctimístného čárového kódu EAN-13

Viz také dotaz `Barcode_EAN13_ttf_command` v ukázkové databázi `Media_without_Macros`.

Datové typy pro editor tabulek

Tabulka 52: Datové typy celých čísel

Typ	Možnost	HSQLDB	Rozsah	Velikost v paměti
Tiny Integer	TINYINT	TINYINT	28 = 256 – 128 to + 127	1 bajt
Small Integer	SMALLINT	SMALLINT	216 = 65536 – 32768 do + 32767	2 bajty
Integer	INTEGER	INTEGER INT	232 = 4294967296 – 2147483648 do + 2147483647	4 bajty
BigInt	BIGINT	BIGINT	264	8 bajtů

Tabulka 53: Datové typy čísel s plovoucí desetinnou čárkou

Typ	Možnost	HSQLDB	Rozsah	Velikost v paměti
Desetinné číslo	DECIMAL	DECIMAL	Neomezený počet, až 50 míst v grafickém uživatelském rozhraní, pevná desetinná čárka, dokonalá přesnost	proměnná

Číslo	NUMERIC	NUMERIC	Neomezený počet, až 50 míst v grafickém uživatelském rozhraní, pevná desetinná čárka, dokonalá přesnost	proměnná
Float	FLOAT	(místo toho se používá DOUBLE)		
Reálná část	REAL	REAL		
Double	DOUBLE	DOUBLE [PRECISION] FLOAT	Nastavitelný, ne přesný, maximálně 15 desetinných míst	8 bajtů

Tabulka 54: Datové typy textu

Typ	Možnost	HSQLDB	Rozsah	Velikost v paměti
Text	VARCHAR	VARCHAR	Nastavitelný	proměnná
Text	VARCHAR_IGNORECASE	VARCHAR_IGNORECASE	Nastavitelný, rozsah ovlivňuje třídění	proměnná
Text (pevný)	CHAR	CHAR CHARACTER	Nastavitelný, zbytek aktuálního textu nahrazen mezerami	pevná
Memo	LONGVARCHAR	LONGVARCHAR		proměnná

Tabulka 55: Datové typy data a času

Typ	Možnost	HSQLDB	Rozsah	Velikost v paměti
Datum	DATE	DATE		4 bajty
Čas	TIME	TIME		4 bajty
Datum/čas	TIMESTAMP	TIMESTAMP DATETIME	Nastavitelný (0,6 – znamená 6 milisekund)	8 bajtů

Tabulka 56: Další datové typy

Typ	Možnost	HSQLDB	Rozsah	Velikost v paměti
Ano/Ne	BOOLEAN	BOOLEAN BIT		
Binární pole (pevné)	BINARY	BINARY	Jako Integer	pevná
Binární pole	VARBINARY	VARBINARY	Jako Integer	proměnná

Image	LONGVARBINARY	LONGVARBINARY	Jako Integer	proměnná, určená pro větší obrázky
OTHER	OTHER	OTHER OBJECT		

V definicích tabulek a při změně datových typů v dotazech pomocí funkcí „convert“ nebo „cast“ se u některých datových typů očekává informace o počtu znaků (a), přesnosti (g, odpovídající celkovému počtu znaků) a počtu desetinných míst (d). Typy jsou CHAR(a), VARCHAR(a), DOUBLE(g), NUMERIC(g, d), DECIMAL(g, d) a TIMESTAMP(g).

TIMESTAMP(g) může nabývat pouze dvou hodnot: '0' a '6'. '0' znamená, že v desetinné části (desetiny, setiny...) nebudou uloženy žádné sekundy. Přesnost časových značek lze zadat pouze přímo pomocí příkazů SQL. Pokud tedy ukládáme časy z nějakého sportu, musíme předem nastavit TIMESTAMP(6) pomocí **Nástroje > SQL**.

Datové typy v LibreOffice Basic

Tabulka 57: Číselné datové typy

Typ	Odpovídá v HSQLDB	Výchozí hodnota	Poznámky	Požadavky na paměť
Integer	SMALLINT	0	$2^{16} = -32768$ až $+32767$	2 bajty
Long	INTEGER	0	$2^{32} = -2147483648$ až $+2147483647$	4 bajty
Single		0.0	Desetinný oddělovač: .	4 bajty
Double	DOUBLE	0.0	Desetinný oddělovač: .	8 bajtů
Currency	Připomíná DECIMAL, NUMERIC	0.0000	4 pevná desetinná místa	8 bajtů

Tabulka 58: Další datové typy

Typ	Odpovídá v HSQLDB	Výchozí hodnota	Poznámky	Požadavky na paměť
Boolean	BOOLEAN	False	1 = ano, vše ostatní: ne.	1 bajt
Datum	TIMESTAMP	00:00:00	Datum a čas	8 bajtů
String	VARCHAR	Prázdný řetězec	Až 65536 znaků	proměnná
Objekt	OTHER	Null		proměnná
Variant		Prázdné	Může přijmout jakýkoli (jiný) datový typ	proměnná

Převod dat představuje velké riziko, zejména v případě číselných hodnot. Například primární klíče v databázích jsou nejčastěji typu INTEGER. Pokud jsou tyto hodnoty načítány makrem, musí být proměnná, ve které jsou uloženy, typu Long, protože velikostně odpovídá typu INTEGER v Base. Odpovídající instrukce pro čtení je `getLong`.

Vestavěné funkce a uložené procedury

Ve vestavěné HSQLDB jsou k dispozici následující funkce. Jednu nebo dvě funkce však lze použít pouze při volbě **Spustit SQL příkaz přímo**. Tím se zabrání úpravě těchto dotazů.

Funkce, které pracují s grafickým uživatelským rozhraním, jsou označeny [Funguje v grafickém uživatelském rozhraní]. Funkce, které fungují pouze v přímých příkazech SQL, jsou označeny [Direct SQL – nefunguje v grafickém uživatelském rozhraní].

Číselné funkce



Poznámka

Protože se zde jedná o čísla s pohyblivou řádovou čárkou, dbejme na nastavení polí v dotazech. Zobrazení desetinných míst je většinou omezeno, takže v některých případech může dojít k neočekávaným výsledkům. Například ve sloupci 1 může být uvedeno 0,00, ale ve skutečnosti obsahuje 0,001 a ve sloupci 2 1000. Pokud je sloupec 3 nastaven tak, aby zobrazoval sloupec 1 * sloupec 2, zobrazí se ve skutečnosti 1.

Tabulka 59: Číselné funkce

Funkce	Popis
ABS(d)	Vrátí absolutní hodnotu čísla. [Pracuje v grafickém uživatelském rozhraní]
ACOS(d)	Vrací arkuskosinus. [Pracuje v grafickém uživatelském rozhraní]
ASIN(d)	Vrací arkussinus. [Pracuje v grafickém uživatelském rozhraní]
ATAN(d)	Vrací arkustangens. [Pracuje v grafickém uživatelském rozhraní]
ATAN2(a,b)	Vrátí arkustangens pomocí souřadnic, kde a je hodnota osy x, b hodnota osy y. [Pracuje v grafickém uživatelském rozhraní]
BITAND(a,b)	Jak binární tvar a, tak binární tvar b musí mít na stejné pozici 1, aby výsledek byl 1. BITAND(3,5) dává 1; 0011 AND 0101 = 0001 [Pracuje v grafickém uživatelském rozhraní]
BITOR(a,b)	Buď binární tvar a nebo binární tvar b musí mít na dané pozici jedničku, aby výsledkem byla jednička. BITOR(3,5) yields 7; 0011 OR 0101 = 0111 [Pracuje v grafickém uživatelském rozhraní]
CEILING(d)	Vrací nejmenší celé číslo, které není menší než d. [Pracuje v grafickém uživatelském rozhraní]
COS(d)	Vrací kosinus. [Pracuje v grafickém uživatelském rozhraní]
COT(d)	Vrací kotangens. [Pracuje v grafickém uživatelském rozhraní]
DEGREES(d)	Převádí radiány na stupně. [Pracuje v grafickém uživatelském rozhraní]

Funkce	Popis
EXP(d)	Vrací hodnotu d-té mocniny čísla e (e: (2.718...)). [Pracuje v grafickém uživatelském rozhraní]
FLOOR(d)	Vrací největší celé číslo, které není větší než d. [Pracuje v grafickém uživatelském rozhraní]
LOG(d)	Vrací přirozený logaritmus se základem e. [Pracuje v grafickém uživatelském rozhraní]
LOG10(d)	Vrátí logaritmus se základem 10. [Pracuje v grafickém uživatelském rozhraní]
MOD(a,b)	Vrátí zbytek jako celé číslo při dělení 2 celých čísel. MOD(11,3) vrátí 2, protože $3 \cdot 3 + 2 = 11$ [Pracuje v grafickém uživatelském rozhraní]
PI()	Vrací π (3.1415...). [Pracuje v grafickém uživatelském rozhraní]
POWER(a,b)	ab , POWER(2,3) = 8, protože $2 \text{ na } 3 = 8$ [Pracuje v grafickém uživatelském rozhraní]
RADIANS(d)	Převádí stupně na radiány. [Pracuje v grafickém uživatelském rozhraní]
RAND()	Vrátí náhodné číslo větší nebo rovno 0,0 a menší než 1,0. [Pracuje v grafickém uživatelském rozhraní]
ROUND(a,b)	Zaokrouhluje číslo a na b desetinných míst. [Pracuje v grafickém uživatelském rozhraní]
ROUNDMAGIC(d)	Řeší problémy se zaokrouhlováním, které vznikají při používání čísel s pohyblivou řádovou čárkou. 3.11-3.1-0.01 není přesně 0, ale v grafickém rozhraní se zobrazuje jako 0. ROUNDMAGIC z ní udělá skutečnou nulovou hodnotu. [Pracuje v grafickém uživatelském rozhraní]
SIGN(d)	Vrací -1, pokud je d menší než 0, 0, pokud je d rovno 0, a 1, pokud je d větší než 0. [Pracuje v grafickém uživatelském rozhraní]
SIN(A)	Vrací sinus úhlu v radiánech. [Pracuje v grafickém uživatelském rozhraní]
SQRT(d)	Vrací druhou odmocninu. [Pracuje v grafickém uživatelském rozhraní]
TAN(A)	Vrací tangens úhlu v radiánech. [Pracuje v grafickém uživatelském rozhraní]
TRUNCATE(a,b)	Zkrátí číslo a na b desetinných míst. TRUNCATE(2.37456,2) = 2.37 [Pracuje v grafickém uživatelském rozhraní]

Textové funkce

Tabulka 60: Textové funkce

Funkce	Popis
ASCII(s)	Vrací kód ASCII prvního písmene řetězce. [Pracuje v grafickém uživatelském rozhraní]
BIT_LENGTH(str)	Vrací délku textového řetězce str v bitech. [Pracuje v grafickém uživatelském rozhraní]

Funkce	Popis
CHAR(c)	Vrátí písmeno odpovídající ASCII kódu c. [Pracuje v grafickém uživatelském rozhraní]
CHAR_LENGTH(str)	Vrací délku řetězce str ve znacích. [Pracuje v grafickém uživatelském rozhraní]
CONCAT(str1,str2)	Spojí řetězce str1 a str2. [Pracuje v grafickém uživatelském rozhraní]
'str1' 'str2' 'str3' nebo 'str1'+ 'str2'+ 'str3'	Spojí řetězce str1, str2 a str3. Jednodušší alternativa ke CONCAT. [Pracuje v grafickém uživatelském rozhraní]
DIFFERENCE(s1,s2)	Vrací rozdíl v anglické výslovnosti mezi řetězci s1 a s2. Výstupem je pouze celé číslo. 0 znamená, že znějí stejně. Například „for“ a „four“ dávají 0, „king“ a „wing“ dávají 1, „see“ a „sea“ dávají 0. [Pracuje v grafickém uživatelském rozhraní]
HEXTORAW(s1)	Převádí hexadecimální kód na jiné znaky. [Pracuje v grafickém uživatelském rozhraní]
INSERT(s,start,len,s2)	Vrací textový řetězec s nahrazenou částí textu. Počínaje znakem "start" se z textu "s" vyřízne délka "len" a nahradí se textem "s2". INSERT(„Bundesbahn“, 3, 4, mme) převede Bundesbahn na Bummelbahn, přičemž délka vloženého textu může být větší než délka odstraněného textu, aniž by to způsobilo nějaké problémy. Takže INSERT(„Bundesbahn“, 3, 5, s a B) dává výsledek „Bus und Bahn“. [Pracuje v grafickém uživatelském rozhraní]
LCASE(s)	Převede řetězec na malá písmena. [Pracuje v grafickém uživatelském rozhraní]
LEFT(s,count)	Vrátí počet znaků zadaný pomocí count od začátku řetězce s. [Pracuje v grafickém uživatelském rozhraní]
LENGTH(s)	Vrací délku řetězce s ve znacích. [Pracuje v grafickém uživatelském rozhraní]
LOCATE(search,s,[start])	Vrátí první shodu hledaného výrazu v řetězci s. Shoda je uvedena jako číslo posunu: (1 = vlevo, 0 = nenalezeno). Nastavení počátečního bodu v rámci textového řetězce není povinné. [Pracuje v grafickém uživatelském rozhraní]
LTRIM(s)	Odstraní počáteční mezery a netisknutelné znaky ze začátku textového řetězce. [Pracuje v grafickém uživatelském rozhraní]
OCTET_LENGTH(str)	Vrací délku textového řetězce v bajtech. To odpovídá dvojnásobku délky řetězce. [Pracuje v grafickém uživatelském rozhraní]
RAWTOHEX(s1)	Převádí na šestnáctková čísla, obráceně než HEXTORAW(). [Pracuje v grafickém uživatelském rozhraní]
REPEAT(s,count)	Opakuje textový řetězec s-krát. [Pracuje v grafickém uživatelském rozhraní]

Funkce	Popis
REPLACE(s,replace,s2)	Nahradí všechny existující výskyty slova replace v textovém řetězci s, řetězcem s2. [Pracuje v grafickém uživatelském rozhraní]
RIGHT(s,count)	Opak funkce LEFT; vrací poslední počet znaků na konci textového řetězce. [Pracuje v grafickém uživatelském rozhraní]
RTRIM(s)	Odstraní všechny mezery a netisknutelné znaky z konce textového řetězce. [Pracuje v grafickém uživatelském rozhraní]
SOUNDEX(s)	Vrací čtyřznakový kód odpovídající zvuku s. Shoduje se s funkcí DIFFERENCE(). [Pracuje v grafickém uživatelském rozhraní]
SPACE(count)	Vrací počet mezer. [Pracuje v grafickém uživatelském rozhraní]
SUBSTR(s,start[,len])	Zkratka pro SUBSTRING. [Pracuje v grafickém uživatelském rozhraní]
SUBSTRING(s,start[,len])	Vrací text s od počáteční pozice (1=vlevo). Pokud je délka len vynechána, je vrácen celý řetězec. [Pracuje v grafickém uživatelském rozhraní]
UCASE(s)	Převede řetězec na velká písmena. [Pracuje v grafickém uživatelském rozhraní]
LOWER(s)	Jako LCASE(s). [Pracuje v grafickém uživatelském rozhraní]
UPPER(s)	Jako UCASE(s). [Pracuje v grafickém uživatelském rozhraní]

Funkce data a času

Tabulka 61: Funkce data a času

Funkce	Popis
CURDATE()	Vrací aktuální datum. [Pracuje v grafickém uživatelském rozhraní]
CURTIME()	Vrací aktuální čas. [Pracuje v grafickém uživatelském rozhraní]
DATEDIFF(string, datetime1, datetime2)	Rozdíl data mezi dvěma daty – porovnává hodnoty data a času. Položka v řetězci určuje jednotky, ve kterých se rozdíl vrací: ms=milisekunda, ss=sekunda, mi=minuta, hh=hodina, dd=den, mm=měsíc, yy=rok. Pro řetězec lze použít jak dlouhou, tak krátkou formu. [Pracuje v grafickém uživatelském rozhraní]
DAY(date)	Vrací den v měsíci (1-31). [Pracuje v grafickém uživatelském rozhraní]
DAYNAME(date)	Vrací anglický název dne. [Pracuje v grafickém uživatelském rozhraní]
DAYOFMONTH(date)	Vrátí den v měsíci (1-31). Synonymum pro DAY(). [Pracuje v grafickém uživatelském rozhraní]

Funkce	Popis
DAYOFWEEK(date)	Vrátí den v týdnu jako číslo (1 představuje neděli). [Pracuje v grafickém uživatelském rozhraní]
DAYOFYEAR(date)	Vrátí den v roce (1-366). [Pracuje v grafickém uživatelském rozhraní]
HOUR(time)	Vrací hodinu (0-23). [Pracuje v grafickém uživatelském rozhraní]
MINUTE(time)	Vrací minutu (0-59). [Pracuje v grafickém uživatelském rozhraní]
MONTH(date)	Vrací měsíc (1-12). [Pracuje v grafickém uživatelském rozhraní]
MONTHNAME(date)	Vrací anglický název měsíce. [Pracuje v grafickém uživatelském rozhraní]
NOW()	Vrátí aktuální datum a aktuální čas společně jako časové razítko. Alternativně lze použít CURRENT_TIMESTAMP. [Pracuje v grafickém uživatelském rozhraní]
QUARTER(date)	Vrátí čtvrtletí roku (1-4). [Pracuje v grafickém uživatelském rozhraní]
SECOND(time)	Vrací sekundovou část času (0-59). [Pracuje v grafickém uživatelském rozhraní]
WEEK(date)	Vrátí týden v roce (1-53). [Pracuje v grafickém uživatelském rozhraní]
YEAR(date)	Vrátí část roku v položce data. [Pracuje v grafickém uživatelském rozhraní]
CURRENT_DATE	Synonymum pro CURDATE(), SQL-Standard. [Pracuje v grafickém uživatelském rozhraní]
CURRENT_TIME	Synonymum pro CURTIME(), SQL-Standard. [Pracuje v grafickém uživatelském rozhraní]
CURRENT_TIMESTAMP	Synonymum pro NOW(), SQL-Standard. [Pracuje v grafickém uživatelském rozhraní]

Funkce připojení k databázi

Poznámka

Kromě funkce IDENTITY(), která nemá v Base žádný význam, lze všechny tyto příkazy provést pomocí přímého příkazu SQL.

Tabulka 62: Funkce připojení k databázi

Funkce	Popis
DATABASE()	Vrací název databáze, ke které toto připojení patří. [Pracuje v grafickém uživatelském rozhraní]
USER()	Vrací uživatelské jméno tohoto připojení. [Přímý SQL – nefunguje s grafickým uživatelským rozhraním]
CURRENT_USER	Standardní funkce SQL, synonymum pro USER(). [Pracuje v grafickém uživatelském rozhraní]

Funkce	Popis
IDENTITY()	Vrací poslední hodnotu pole automatické hodnoty, které bylo vytvořeno v aktuálním spojení. Používá se v makrech k převodu primárního klíče v jedné tabulce na cizí klíč pro jinou tabulku. [Pracuje v grafickém uživatelském rozhraní]

Funkce systému

Tabulka 63: Systémové funkce

Funkce	Popis
IFNULL(exp,value)	Pokud je exp NULL, vrátí se value, jinak se vrátí exp. Alternativně lze jako rozšíření použít funkci COALESCE(). Exp a value musí mít stejný datový typ. [Funguje v grafickém uživatelském rozhraní]
CASEWHEN(exp,v1,v2)	Pokud je exp true, vrátí se v1, jinak v2. Alternativně lze použít CASE WHEN. CASE WHEN funguje lépe s grafickým uživatelským rozhraním. [Funguje v grafickém uživatelském rozhraní]
CONVERT(term,type)	Převede term na jiný datový typ. [Funguje v grafickém uživatelském rozhraní]
CAST(term AS type)	Synonymum pro CONVERT(). [Funguje v grafickém uživatelském rozhraní]
COALESCE(expr1,expr2,expr3,...)	Pokud expr1 není NULL, vrátí se expr1, jinak se zkontroluje expr2, pak expr3 atd. [Funguje v grafickém uživatelském rozhraní]
NULLIF(v1,v2)	Pokud se v1 rovná v2, vrátí se NULL, jinak se vrátí v1. [Funguje v grafickém uživatelském rozhraní]
CASE v1 WHEN v2 THEN v3 [ELSE v4] END	Pokud se v1 rovná v2, je vrácena hodnota v3. V opačném případě je vrácena hodnota v4 nebo NULL, pokud neexistuje podmínka ELSE. [Přímý SQL – nefunguje s grafickým uživatelským rozhraním]
CASE WHEN expr1 THEN v1[WHEN expr2 THEN v2] [ELSE v4] END	Pokud je expr1 true, vrátí se v1 [volitelně lze nastavit další podmínky]. V opačném případě je vrácena hodnota v4 nebo NULL, pokud neexistuje podmínka ELSE. [Funguje v grafickém uživatelském rozhraní]
EXTRACT ({YEAR MONTH DAY HOUR MINUTE SECOND} FROM <datum nebo čas>)	Může nahradit mnoho funkcí data a času. Vrací rok, měsíc, den atd. z data nebo hodnoty data a času. [Funguje v grafickém uživatelském rozhraní]
POSITION(<řetězcový výraz> IN <řetězcový výraz>)	Pokud je první řetězec obsažen v druhém, je uvedena odchylka prvního řetězce, jinak je vrácena 0. [Funguje v grafickém uživatelském rozhraní]
SUBSTRING(<řetězcový výraz> FROM <číselný výraz> [FOR <číselný výraz>])	Vytvoří část textového řetězce od pozice zadané ve FROM, volitelně až do délky zadané ve FOR. [Funguje v grafickém uživatelském rozhraní]

Funkce	Popis
TRIM({LEADING TRAILING BOTH}) FROM <řetězcový výraz>	Netisknutelné speciální znaky a mezery jsou odstraněny. [Funguje v grafickém uživatelském rozhraní]

Řídící znaky pro použití v dotazech

Pole lze v dotazech propojovat. Dvě pole v

```
SELECT "Jméno", "Příjmení" FROM "Tabulka"
```

se stanou jedním polem pomocí:

```
SELECT "Jméno" || ' ' || "Příjmení" FROM "Tabulka"
```

Zde se vkládá další mezera. Může to být jakýkoli znak; pokud je uzavřen v ", bude interpretován jako text. Někdy je však nutné vložit netisknutelné znaky, například nové řádky, například při přípravě zpráv. Zde je krátký seznam řídicích znaků. Více informací o nich najdeme na https://en.wikipedia.org/wiki/Control_character.

Tabulka 64: Řídící znaky pro použití v dotazech

Kód	Název	Poznámky
CHAR(9)	Vodorovný tabulátor	
CHAR(10)	Posun o řádku	V dopisech hromadné korespondence a v nástroji Návrhář sestav vytvoří zalomení řádku (Linux, Unix, Mac).
CHAR(13)	CR - carriage return (návrát vozíku)	Zalomení řádku v kombinaci s Carriage return v systému Windows CHAR(13) CHAR(10). Lze použít i v systémech Linux a Mac, proto univerzální varianta.

Některé příkazy UNO pro použití s tlačítkem

Tlačítko může mít přímo navázány různé příkazy UNO. Za tímto účelem je třeba zvolit **Vlastnosti: Tlačítko > Činnost > Otevřít dokument / webovou stránku** a pak například **URL > .uno:RecSearch**, aby se otevřela funkce vyhledávání. Často je třeba zvolit **Při kliknutí získat zaměření > Ne**, pokud akce přistupuje přímo k jinému ovládacímu prvku způsobem, který vyžaduje, aby byl v zaměření, například **.uno:Paste**, který může vložit obsah schránky.

Následující seznam obsahuje pouze několik příkazů. Všechny příkazy z nástrojové lišty navigace jsou již použitelné v tlačítku, ale lze je také vytvořit pomocí příkazů UNO. Mnoho příkazů lze zjistit pomocí záznamníku maker, který k nim často používá dispečera.

Tabulka 65: Ukázkové příkazy UNO

Příkaz UNO	Používá se pro ...
.uno:RecSearch	Otevření funkce vyhledávání ve formuláři.
.uno:Paste	Vložit ze schránky. Funguje pouze pro Při kliknutí získat zaměření > Ne
.uno:Copy	Zkopíruje vybraný obsah do schránky. Funguje pouze pro Při kliknutí získat zaměření > Ne
.uno:Print	Otevření dialogového okna pro tisk formuláře.
.uno:PrintDefault	Tisk na výchozí tiskárně bez zobrazení dialogového okna.

Informační tabulky pro HSQLDB

Uvnitř databáze jsou informace o všech vlastnostech tabulek a jejich vzájemných vazbách uloženy v oblasti *INFORMATION_SCHEMA*. Tyto informace umožňují vytvářet makra v aplikaci Base, která pro své procedury vyžadují jen velmi málo argumentů. Aplikace je uvedena v příkladové databázi v modulu Údržba – procedura *Table_purge* pro kontrolu dialogových oken.

V dotazu lze jednotlivé informace a všechna pole, která k nim patří, zadat následujícím způsobem:

```
SELECT * FROM "INFORMATION_SCHEMA"."SYSTEM_ALIASES"
```

Na rozdíl od běžné tabulky je zde nutné použít *INFORMATION_SCHEMA* jako předponu k příslušnému názvu z následujícího seznamu:

```
SYSTEM_ALIASES
SYSTEM_ALLTYPEINFO
SYSTEM_BESTROWIDENTIFIER
SYSTEM_CACHEINFO
SYSTEM_CATALOGS
SYSTEM_CHECK_COLUMN_USAGE
SYSTEM_CHECK_CONSTRAINTS
SYSTEM_CHECK_ROUTINE_USAGE
SYSTEM_CHECK_TABLE_USAGE
SYSTEM_CLASSPRIVILEGES
SYSTEM_COLUMNPRIVILEGES
SYSTEM_COLUMNS
SYSTEM_CROSSREFERENCE
SYSTEM_INDEXINFO
SYSTEM_PRIMARYKEYS
SYSTEM_PROCEDURECOLUMNS
SYSTEM_PROCEDURES
SYSTEM_PROPERTIES
SYSTEM_SCHEMAS
SYSTEM_SEQUENCES
SYSTEM_SESSIONINFO
SYSTEM_SESSIONS
SYSTEM_SUPERTABLES
SYSTEM_SUPERTYPES
SYSTEM_TABLEPRIVILEGES
SYSTEM_TABLES
SYSTEM_TABLETYPES
SYSTEM_TABLE_CONSTRAINTS
SYSTEM_TEXTTABLES
```

```
SYSTEM_TRIGGERCOLUMNS
SYSTEM_TRIGGERS
SYSTEM_TYPEINFO
SYSTEM_UDTATTRIBUTES
SYSTEM_UDTS
SYSTEM_USAGE_PRIVILEGES
SYSTEM_USERS
SYSTEM_VERSIONCOLUMNS
SYSTEM_VIEWS
SYSTEM_VIEW_COLUMN_USAGE
SYSTEM_VIEW_ROUTINE_USAGE
SYSTEM_VIEW_TABLE_USAGE
```

Následující dotaz poskytuje úplný přehled všech tabulek v databázi s typy polí, primárními klíči a cizími klíči:

```
SELECT
"A"."TABLE_NAME",
"A"."COLUMN_NAME",
"A"."TYPE_NAME",
"A"."NULLABLE",
"B"."KEY_SEQ" AS "PRIMARYKEY",
"C"."PKTABLE_NAME" || '.' || "C"."PKCOLUMN_NAME" AS "FOREIGNKEY FOR"
FROM "INFORMATION_SCHEMA"."SYSTEM_COLUMNS" AS "A"
LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_PRIMARYKEYS" AS "B"
ON ( "B"."TABLE_NAME" = "A"."TABLE_NAME" AND "B"."COLUMN_NAME" =
"A"."COLUMN_NAME" )
LEFT JOIN "INFORMATION_SCHEMA"."SYSTEM_CROSSREFERENCE" AS "C"
ON ( "C"."FKTABLE_NAME" = "A"."TABLE_NAME" AND
"C"."FKCOLUMN_NAME" = "A"."COLUMN_NAME" )
WHERE "A"."TABLE_SCHEM" = 'PUBLIC'
```

Oprava databáze pro soubory *.odb

Pravidelné zálohování dat by mělo být při používání počítače standardem. Záložní kopie jsou nejjednodušším způsobem, jak se vrátit k napůl aktuálnímu stavu dat. V praxi to však často chybí.

Formuláře, dotazy a sestavy lze vždy zkopírovat pomocí schránky do nové databáze, pokud byla uložena předchozí verze databáze. Pokud však z nějakého důvodu již nelze aktuální databázi otevřít, hlavním problémem se stává přístup k datům.

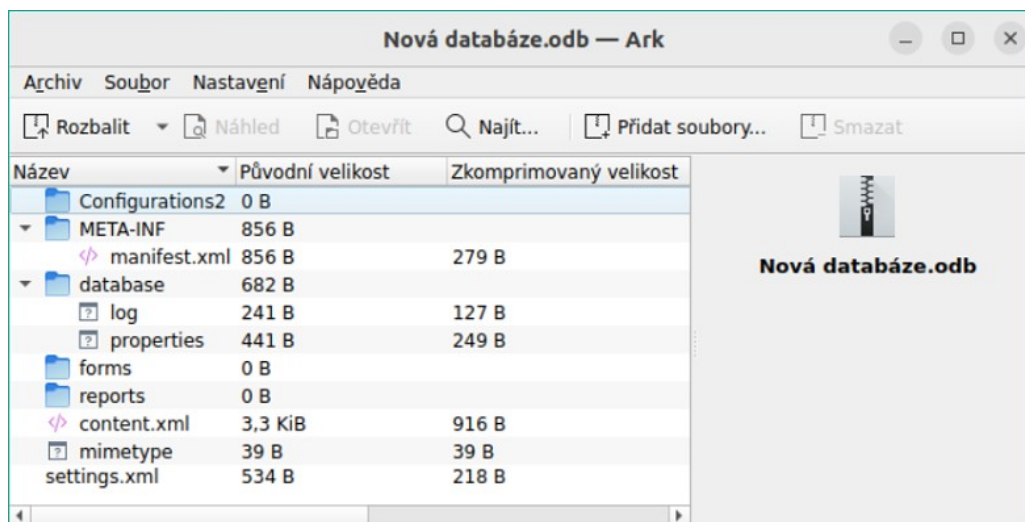
V případě náhlého selhání počítače se může stát, že otevřená databáze (interní databáze HSQLDB) již nelze v LibreOffice otevřít. Místo toho se při pokusu o otevření databáze zobrazí dotaz na filtr odpovídající formátu.

Problém spočívá v tom, že část dat v otevřené databázi je obsažena v pracovní paměti a do mezipaměti se kopíruje pouze dočasně. Teprve po uzavření souboru je celá databáze zapsána zpět do souboru a znovu zabalena.

Obnovení archivačního souboru databáze

Pro opětovné získání přístupu k datům může být užitečný následující postup:

- 1) Vytvoříme kopii databáze pro následující kroky.
- 2) Zkusíme kopii otevřít pomocí archivačního programu. V případě souborů *.odb se jedná o komprimovaný formát, archiv Zip. Pokud soubor nelze otevřít přímo, zkusíme jej přejmenovat z *.odb na *.zip. Pokud se neotevře, je naše databáze již neuložená.
- 3) Po otevření databázového souboru v archivačním programu se vždy zobrazí následující složky:



Obrázek 446: Základní struktura souborů komprimovaných do souboru ODB pro "prázdnou" databázi

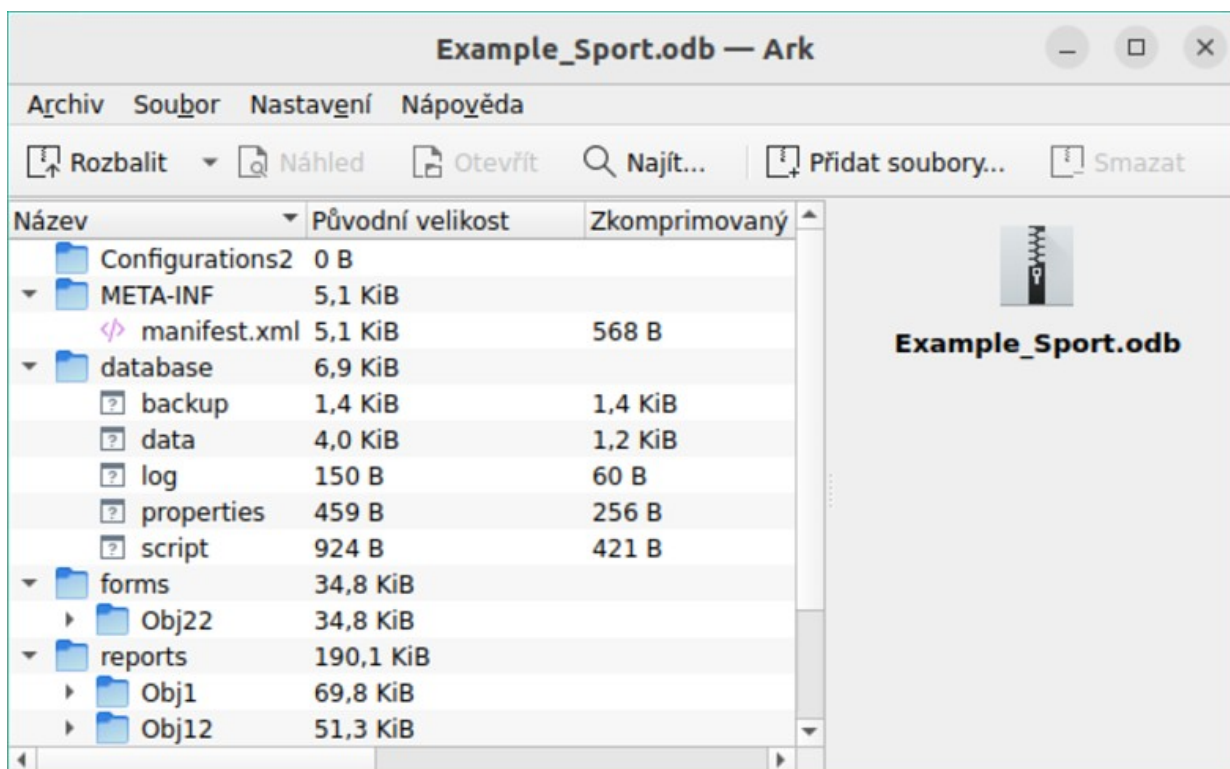
- 4) Soubor databáze musí být dekomprimován. Nejdůležitější informace, pokud jde o data, jsou v podsložce databáze v souborech data a script.
- 5) Možná bude nutné se podívat do souboru script a otestovat, zda v něm nejsou rozpory. Tento krok však lze ponechat na fázi testování. Soubor script obsahuje především popis struktury tabulky.
- 6) Vytvoříme nový prázdný databázový soubor a otevřeme jej pomocí archivačního programu.
- 7) Nahradíme soubory data a script v novém databázovém souboru soubory rozbalenými v kroku 4.
- 8) Zavřeme archivační program. Pokud bylo nutné soubor před otevřením v archivačním programu přejmenovat na *.zip (záleží na operačním systému), přejmenujeme jej nyní znovu na *.odb.
- 9) Otevřeme databázový soubor v LibreOffice. Měli bychom mít opět přístup ke svým tabulkám.
- 10) Jak velkou část dotazů, formulářů a sestav lze podobným způsobem obnovit, je třeba dále testovat.

Viz také: <http://forum.openoffice.org/en/forum/viewtopic.php?f=83&t=17125>

Další informace o archivačních souborech databáze

V praxi obsahuje archivní soubor databáze nejen základní složku pro databázi a složku META-INF, která je určena pro formát OpenDocument, ale také další složky pro ukládání formulářů a sestav. Popis základní struktury formátu OpenDocument najdeme na adrese https://en.wikipedia.org/wiki/OpenDocument_technical_specification.

Následující pohled zobrazuje databázi obsahující tabulky, formulář a sestavu. Není zřejmé, že databáze obsahuje také dotaz. Dotazy nejsou uloženy v samostatných složkách, ale v souboru content.xml. Informace potřebné ke spuštění dotazu jsou jednoduché části kódu SQL.



Obrázek 447: Databázový soubor obsahující kromě tabulek a dotazů také uložené informace pro formulář a sestavu

Zde je náhled jednoho z archivních souborů databáze.

mimetype

```
application/vnd.oasis.opendocument.base
```

eine

Tento malý textový soubor obsahuje pouze upozornění, že tento archivní soubor je databázový soubor ve formátu OpenDocument.

content.xml pro databázi bez obsahu

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content
xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0"
xmlns:fo="urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0"
xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0"
xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0"
```

```

xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0"
xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0"
xmlns:ooo="http://openoffice.org/2004/office"
xmlns:ooow="http://openoffice.org/2004/writer"
xmlns:oooc="http://openoffice.org/2004/calc"
xmlns:dom="http://www.w3.org/2001/xml-events"
xmlns:db="urn:oasis:names:tc:opendocument:xmlns:database:1.0"
xmlns:xforms="http://www.w3.org/2002/xforms"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rpt="http://openoffice.org/2005/report"
xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2"
xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:grddl="http://www.w3.org/2003/g/data-view#"
xmlns:tableooo="http://openoffice.org/2009/table"
xmlns:drawooo="http://openoffice.org/2010/draw"
xmlns:calcext="urn:org:documentfoundation:names:experimental:calc:xmlns:calce
xt:1.0" xmlns:field="urn:openoffice:names:experimental:ooo-ms-
interop:xmlns:field:1.0"
xmlns:formx="urn:openoffice:names:experimental:ooxml-odf-
interop:xmlns:form:1.0" xmlns:css3t="http://www.w3.org/TR/css3-text/"
office:version="1.2">
  <office:scripts/>
  <office:font-face-decls/>
  <office:automatic-styles/>
  <office:body>
    <office:database>
      <db:data-source>
        <db:connection-data>
          <db:connection-resource xlink:href="sdbc:embedded:hsqldb"/>
          <db:login db:is-password-required="false"/>
        </db:connection-data>
        <db:driver-settings
          db:system-driver-settings=""
          db:base-dn=""
          db:parameter-name-substitution="false"/>
        <db:application-connection-settings
          db:is-table-name-length-limited="false"
          db:append-table-alias-name="false"
          db:max-row-count="100">
          <db:table-filter>
            <db:table-include-filter>
              <db:table-filter-pattern%</db:table-filter-pattern>
            </db:table-include-filter>
          </db:table-filter>
        </db:application-connection-settings>
      </db:data-source>
    </office:database>
  </office:body>
</office:document-content>

```

Začíná verzí XML a použitou znakovou sadou. Vše, co následuje, je vlastně jeden nezalomený řádek. Výše připravený pohled by měl vše objasnit. Prvky, které patří k sobě, jsou označeny značkami.

Počáteční definice začínající `xmlns:` (jmenný prostor XML) udávají jmenné prostory, ke kterým lze přistupovat zevnitř souboru. Pak se uvažuje o poněkud konkrétnějších pojmech. Zde je zřejmé, že máme co do činění s interní databází HSQLDB a že pro přístup není vyžadováno heslo.

content.xml pro databázi s obsahem

Následující obsah je pouze výňatkem ze souboru content.xml, aby byla jasná jeho struktura.

```
<office:scripts/>
<office:font-face-decls>
  <style:font-face style:name="F" svg:font-family=""/>
</office:font-face-decls>
<office:automatic-styles>
  <style:style
    style:name="co1"
    style:family="table-column"
    style:data-style-name="N0"/>
  <style:style
    style:name="co2"
    style:family="table-column"
    style:data-style-name="N107"/>
  <style:style style:name="ce1" style:family="table-cell">
    <style:paragraph-properties fo:text-align="start"/>
  </style:style>
  <number:number-style style:name="N0" number:language="de"
number:country="DE">
    <number:number number:min-integer-digits="1"/>
  </number:number-style>
  <number:currency-style
    style:name="N107P0"
    style:volatile="true"
    number:language="de"
    number:country="DE">
    <number:number
      number:decimal-places="2"
      number:min-integer-digits="1"
      number:grouping="true"/>
    <number:text> </number:text>
    <number:currency-symbol
      number:language="de"
      number:country="DE">€
    </number:currency-symbol>
  </number:currency-style>
```

Zde je pole definováno jako pole měny. Uvádí se počet desetinných míst, odstup mezi čísly a symbolem měny a samotný symbol měny. V níže uvedeném příkladu se měna a národní prostředí vztahují k Německu.

```
<number:currency-style
  style:name="N107"
  number:language="de"
  number:country="DE">
  <style:text-properties fo:color="#ff0000"/>
  <number:text>-</number:text>
  <number:number
    number:decimal-places="2"
    number:min-integer-digits="1"
    number:grouping="true"/>
```

```

<number:text> </number:text>
<number:currency-symbol
  number:language="de"
  number:country="DE">€
</number:currency-symbol>
<style:map style:condition="value()>=0" style:apply-style-
name="N107P0"/>
</number:currency-style>

```

Druhý výpis uvádí, že do určité hodnoty by se měna měla zobrazovat červeně („ff0000“).

```

</office:automatic-styles>
<office:body>
  <office:database>
    <db:data-source>

```

Tato položka z výše uvedeného souboru content.xml se všemi svými podpoložkami odpovídá prázdnému souboru archivu databáze.

```

</db:data-source>
<db:forms>
  <db:component
    db:name="Receipts"
    xlink:href="forms/Obj12"
    db:as-template="false"/>
</db:forms>

```

Soubor archivu databáze obsahuje podsekcí, ve kterém jsou uloženy podrobnosti formuláře. Formulář je v uživatelském rozhraní označen jako *Receipts*.

```

<db:reports>
  <db:component
    db:name="Receipts"
    xlink:href="reports/Obj12"
    db:as-template="false"/>
</db:reports>

```

Soubor archivu databáze obsahuje také podsekcí, ve kterém jsou uloženy podrobnosti o sestavě. Tato sestava je v uživatelském rozhraní označena také jako *Receipts*.

```

<db:queries>
  <db:query
    db:name="Sales_calc"
    db:command="SELECT &quot;a&quot;.*, ( SELECT &quot;Price&quot; *
      &quot;a&quot;.&quot;Total&quot; FROM &quot;Stock&quot; WHERE
      &quot;ID&quot; = &quot;a&quot;.&quot;Stock_ID&quot; ) AS
      &quot;Total*Price&quot; FROM &quot;Sales&quot; AS
&quot;a&quot;"/>
</db:queries>

```

Všechny dotazy jsou uloženy přímo v souboru content.xml. " znamená dvojité uvozovky. Výše uvedený dotaz v tomto příkladu je ve skutečnosti poměrně složitý a obsahuje mnoho korelovaných poddotazů. Zde jej uvádíme ve zkrácené podobě.

```

<db:table-representations>
  <db:table-representation db:name="Receipts"/>
  <db:table-representation db:name="Sales"/>
  <db:table-representation db:name="Stock">
    <db:columns>
      <db:column
        db:name="ID"

```

```

        db:style-name="co1"
        db:default-cell-style-name="ce1"/>
    <db:column
        db:name="MWSt"
        db:style-name="co1"
        db:default-cell-style-name="ce1"/>
    <db:column
        db:name="Price"
        db:style-name="co2"
        db:default-cell-style-name="ce1"/>
    <db:column
        db:name="Stock"
        db:style-name="co1"
        db:default-cell-style-name="ce1"/>
</db:columns>
</db:table-representation>
</db:table-representations>

```

To ukazuje, jak se mají zobrazovat různé tabulky. Zde jsou uloženy vlastnosti zobrazení jednotlivých sloupců: v tomto příkladu jsou uložena nastavení pro tabulku Stock s jejími poli – ID, MWSt atd.. Zřejmě zde bylo něco přímo zadáno, čímž se trochu změnilo sloupce tabulky.

```

</office:database>
</office:body>

```

V souboru content.xml je v podstatě přímo uložen obsah dotazů a informace o vizuálním vzhledu tabulek. Dále je zde definice připojení k databázi. Nakonec přicházejí informace o formulářích a sestavách.

```

settings.xml
<?xml version="1.0" encoding="UTF-8"?>
<office:document-settings
xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:table="urn:oasis:names:tc:opendocument:xmlns:table:1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:number="urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0"
xmlns:svg="http://www.w3.org/2000/svg"
xmlns:config="urn:oasis:names:tc:opendocument:xmlns:config:1.0"
xmlns:ooo="http://openoffice.org/2004/office"
xmlns:db="urn:oasis:names:tc:opendocument:xmlns:database:1.0"
office:version="1.2"/>

```

U databáze bez dalšího obsahu jsou zde uloženy pouze základní definice. S obsahem se ukládají také různá nastavení. Po spuštění výše uvedené definice se uloží následující nastavení z ukázkové databáze.

```

<office:settings>
  <config:config-item-set config:name="ooo:view-settings">
    <config:config-item-set config:name="Queries">
      <config:config-item-set config:name="Calculate_sales">
        <config:config-item-set config:name="Tables">
          <config:config-item-set config:name="Table1">
            <config:config-item config:name="WindowName"
              config:type="string">Verkauf</config:config-item>
            <config:config-item config:name="WindowLeft"
              config:type="int">153</config:config-item>
            <config:config-item config:name="ShowAll"
              config:type="boolean">>true</config:config-item>
            <config:config-item config:name="WindowTop"

```

```

        config:type="int">17</config:config-item>
    <config:config-item config:name="WindowWidth"
        config:type="int">120</config:config-item>
    <config:config-item config:name="WindowHeight"
        config:type="int">120</config:config-item>
    <config:config-item config:name="ComposedName"
        config:type="string">Verkauf</config:config-item>
    <config:config-item config:name="TableName"
        config:type="string">Verkauf</config:config-item>
    </config:config-item-set>
</config:config-item-set>
<config:config-item config:name="SplitterPosition"
    config:type="int">105</config:config-item>
<config:config-item config:name="VisibleRows"
    config:type="int">1024</config:config-item>
</config:config-item-set>
</config:config-item-set>
</config:config-item-set>
<config:config-item-set config:name="ooo:configuration-settings">
    <config:config-item-set config:name="layout-settings">
        <config:config-item-set config:name="Tables">
            <config:config-item-set config:name="Table1">
                <config:config-item config:name="WindowName"
                    config:type="string">Verkauf</config:config-item>
                <config:config-item config:name="WindowLeft"
                    config:type="int">186</config:config-item>
                <config:config-item config:name="ShowAll"
                    config:type="boolean">>false</config:config-item>
                <config:config-item config:name="WindowTop"
                    config:type="int">17</config:config-item>
                <config:config-item config:name="WindowWidth"
                    config:type="int">120</config:config-item>
                <config:config-item config:name="WindowHeight"
                    config:type="int">120</config:config-item>
                <config:config-item config:name="ComposedName"
                    config:type="string">Verkauf</config:config-item>
                <config:config-item config:name="TableName"
                    config:type="string">Sales</config:config-item>
            </config:config-item-set>
            <config:config-item-set config:name="Table2">
                ... (identical config:type-Points as "Table1"
                <config:config-item config:name="TableName"
                    config:type="string">Ware</config:config-item>
            </config:config-item-set>
            <config:config-item-set config:name="Table3">
                ... (identical config:type-Points as "Table1"
                <config:config-item config:name="TableName"
                    config:type="string">Receipts</config:config-item>
            </config:config-item-set>
        </config:config-item-set>
    </config:config-item-set>
</config:config-item-set>
</office:settings>

```

Celý přehled se týká různých zobrazení oken pro dotaz Calculate_sales a tabulek Sales, Stock a Receipts. Poslední dvě jsou zde uvedeny ve zkrácené podobě. Pokud by tato nastavení

v chybném souboru *.odb chyběla, nevadilo by to. Při dalším otevření příslušného okna by se znovu vytvořila.

META-INF/manifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest
  xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
  <manifest:file-entry
    manifest:full-path="/"
    manifest:media-type="application/vnd.oasis.opendocument.base"/>
  <manifest:file-entry
    manifest:full-path="database/script"
    manifest:media-type=""/>
  <manifest:file-entry
    manifest:full-path="database/properties"
    manifest:media-type=""/>
  <manifest:file-entry
    manifest:full-path="settings.xml"
    manifest:media-type="text/xml"/>
  <manifest:file-entry
    manifest:full-path="content.xml"
    manifest:media-type="text/xml"/>
</manifest:manifest>
```

Tento soubor ve složce META-INF udává složku s obsahem celého archivu databáze. Protože tento soubor pracuje s prázdnou databází, je v něm pouze pět položek. Databázový archiv, který obsahuje formuláře a sestavy, bude mít mnohem složitější soubor META-INF.

database/properties

```
#HSQL Database Engine 1.8.0.10
#Sun Jul 14 18:02:08 CEST 2013
hsqldb.script_format=0
runtime.gc_interval=0
sql.enforce_strict_size=true
hsqldb.cache_size_scale=8
readonly=false
hsqldb.nio_data_file=false
hsqldb.cache_scale=13
version=1.8.0
hsqldb.default_table_type=cached
hsqldb.cache_file_scale=1
hsqldb.lock_file=true
hsqldb.log_size=10
modified=no
hsqldb.cache_version=1.7.0
hsqldb.original_version=1.8.0
hsqldb.compatible_version=1.8.0
```

Soubor vlastností obsahuje základní nastavení interní databáze HSQLDB.

database/script

```
SET DATABASE COLLATION "German"  
CREATE SCHEMA PUBLIC AUTHORIZATION DBA  
CREATE USER SA PASSWORD ""  
GRANT DBA TO SA  
SET WRITE_DELAY 60
```

Soubor skriptu obsahuje výchozí nastavení pro připojení k databázi, nastavení jazyka atd. Zde se objeví uživatel SA, který bude popsán později.

V databázi s obsahem obsahuje tento soubor základní definice tabulek.

```
SET DATABASE COLLATION "German"  
CREATE SCHEMA PUBLIC AUTHORIZATION DBA
```

Tabulky jsou definovány před definicí uživatele databáze. Nejprve se v mezipaměti vytvoří tabulky se svými poli.

```
CREATE CACHED TABLE "Stock"  
  ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL  
   PRIMARY KEY, "Stock" VARCHAR(50), "Price" DECIMAL(8,2), "MWSt" TINYINT)  
CREATE CACHED TABLE "Sales"  
  ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL  
   PRIMARY KEY, "Total" TINYINT, "Stock_ID" INTEGER, "Receipt_ID" INTEGER,  
   CONSTRAINT SYS_FK_59 FOREIGN KEY("Stock_ID") REFERENCES "Stock"("ID"))  
CREATE CACHED TABLE "Receipts"  
  ("ID" INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 0) NOT NULL  
   PRIMARY KEY, "Date" DATE)
```

Poté se provedou změny v tabulce, aby se zajistila konzistence relací (REFERENCES).

```
ALTER TABLE "Sales" ADD CONSTRAINT SYS_FK_76 FOREIGN KEY("Receipt_ID")  
  REFERENCES "Receipts"("ID")  
SET TABLE "Stock" INDEX'608 20'  
SET TABLE "Sales" INDEX'1872 1656 1872 12'  
SET TABLE "Receipts" INDEX'2232 1'
```

Po nastavení pozice indexu v datovém souboru (zde se objevuje pouze v souboru skriptu, ale nikdy se nezadáva přímo v SQL) se automaticky inkrementující pole v tabulkách (automatické hodnoty) nastaví tak, aby při zadání nového záznamu poskytovala další hodnotu. Předpokládejme, že poslední zadaná hodnota v poli ID tabulky Stock je 19. Automatické zvyšování pak začíná na hodnotě 20.

```
ALTER TABLE "Stock" ALTER COLUMN "ID" RESTART WITH 20  
ALTER TABLE "Sales" ALTER COLUMN "ID" RESTART WITH 12  
ALTER TABLE "Receipts" ALTER COLUMN "ID" RESTART WITH 1  
CREATE USER SA PASSWORD ""  
GRANT DBA TO SA  
SET WRITE_DELAY 60
```


Správa interní databáze Firebird

Interní databáze Firebird je v současné době k dispozici pouze jako experimentální funkce. Chceme-li vytvořit takovou databázi nebo upravit již vytvořenou, musíme vybrat **Nástroje > Možnosti > LibreOffice > Pokročilé > Volitelné (Nestabilní) možnosti > Povolit experimentální funkce**.

Následující odkaz umožňuje nahlásit významné chyby v interní databázi Firebird a řešit je společně s týmem LibreOffice: [Hlášení chyb pro Firebird v aplikaci Base](#).

Uživatelé si oproti HSQLDB všimnou následujících rozdílů:

- 1) Pokud je poli přidělen typ Integer a je deklarováno jako primární klíč, zdá se, že je možné mu přidělit automaticky se zvyšující hodnotu. Při uložení však toto nastavení bez upozornění zmizí.
- 2) Při zadávání nových záznamů se tyto záznamy automaticky neukládají do databáze. Tlačítko Uložit je třeba použít pro každou položku. Ve vestavěné HSQLDB není explicitní ukládání záznamů nutné.
- 3) Aliasy jsou v dotazech zcela ignorovány. Alias lze vytvořit, ale nezobrazí se v záhlaví tabulky dotazu.
- 4) Není možné vytvářet podmínky, ačkoli je externí databáze Firebird podporují.
- 5) Desetinné a číselné datové typy jsou v současné době chybné. Tyto typy jsou jediné, které zajišťují přesné hodnoty, zejména pokud se jedná o desetinná místa. Proto jsou preferovanými poli pro hodnoty měn. V současné době lze zadávat pouze hodnoty s nejméně jedním desetinným místem.

Zpřístupnění automatických hodnot

Následující kód zadaný pomocí **Nástroje > SQL** může pomoci vyřešit problém s neposkytováním automatických hodnot.

```
CREATE TABLE "Table1" ( "ID" INTEGER NOT NULL PRIMARY KEY, "Name"
VARCHAR(20) NOT NULL );
CREATE GENERATOR GEN_T1_ID;
SET GENERATOR GEN_T1_ID TO 0;
```

Poté je třeba zavřít okno pro zadávání SQL a vybrat **Pohled > Obnovit tabulky**. Teprve když se tabulka objeví a v některých případech až po (neúspěšném) pokusu o vytvoření záznamu, lze vytvořit následující spouštěč.

```
CREATE TRIGGER T1_BI FOR "Table1" ACTIVE BEFORE INSERT POSITION 0 AS
BEGIN
IF (NEW.ID IS NULL) THEN NEW.ID = GEN_ID(GEN_T1_ID, 1);
END;
```

I poté lze v tabulce vytvořit mnoho záznamů v položce Name, aniž by byl vytvořen záznam v položce ID. V poli ID se zobrazuje pouze 0. Teprve po stisknutí tlačítka **Update** se zobrazí skutečně přiřazené hodnoty. Spouštěč poskytuje hodnoty začínající číslicí 1.



LibreOffice
Community



Příručka aplikace Base 7.3

Dodatek B *Porovnání HSQLDB* *a Firebird*

Datové typy a funkce

Datové typy a funkce v HSQLDB a Firebird

Tabulky v tomto dodatku jsou převzaty z příruček pro HSQLDB a Firebird.

- <http://hsqldb.org/doc/guide/>
- <https://www.firebirdsql.org/en/documentation/>

Informace pro interní HSQLDB jsou stejné jako v dodatku A této knihy.

Dodatečná interní databáze Firebird je klasifikována jako experimentální.

V tabulkách je nejprve uvedeno srovnání funkcí, zejména funkcí, které jsou na fórech oblíbené, jako např.:

- Přidání určitého počtu dní k datu (DATEADD)
- Hodnoty z více datových řádků seskupené do jednoho datového řádku (LIST)

jsou v současné době k dispozici pouze v externí databázi Firebird, ale ne v interní verzi.

Vestavěné funkce a uložené procedury

Ve vestavěných databázích jsou k dispozici následující funkce. Jednu nebo dvě funkce lze použít pouze při volbě **Spustit SQL příkaz přímo**. V tomto režimu nelze dotazy upravovat.

Funkce, které pracují s grafickým uživatelským rozhraním, jsou označeny [funguje v grafickém uživatelském rozhraní]. Funkce, které fungují pouze v přímých příkazech SQL, jsou označeny [nefunguje v grafickém uživatelském rozhraní].

Číselné funkce



Upozornění

Protože se zde jedná o čísla s pohyblivou řádovou čárkou, dbejme na nastavení polí v dotazech. Většinou je omezeno zobrazování desetinných míst, což může v některých případech vést k neočekávanému chování. Například ve sloupci 1 může být uvedeno 0,00, ale ve skutečnosti obsahuje 0,001 a ve sloupci 2 1000. Pokud je sloupec 3 nastaven tak, aby zobrazoval sloupec 1 * sloupec 2, zobrazí se ve skutečnosti 1.

Tabulka 66: Číselné funkce

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
ABS(d)	Vrací absolutní hodnotu čísla. [funguje v grafickém uživatelském rozhraní]	ABS(d)	
ACOS(d)	Vrací arkuskosinus. [funguje v grafickém uživatelském rozhraní]	ACOS(d)	
ASIN(d)	Vrací arkussinus. [funguje v grafickém uživatelském rozhraní]	ASIN(d)	
ATAN(d)	Vrací arkustangens. [funguje v grafickém uživatelském rozhraní]	ATAN(d)	
ATAN2(a, b)	Vrací arkustangens prostřednictvím souřadnic. 'a' je hodnota osy x, 'b' je hodnota osy y. [funguje v grafickém uživatelském rozhraní]	ATAN2(x, y)	
BITAND(a, b)	"a" i "b" jsou uvedeny formou binárního zápisu. Musí mít na stejné pozici '1', aby výsledek byl '1'. BITAND(3,5) vrací 1; 0011 AND 0101 = 0001 [funguje v grafickém uživatelském rozhraní]	BIN_AND(x, y [, z . . .])	
BITOR(a, b)	"a" i "b" jsou uvedeny formou binárního zápisu. Aby výsledek pro danou pozici byl „1“, musí být na stejné pozici v "a" nebo v "b" hodnota „1“. BITOR(3,5) vrací 7; 0011 OR 0101 = 0111 [funguje v grafickém uživatelském rozhraní]	BIN_OR(x, y [, z . . .])	
		BIN_SHL(n, exp)	n · 2 ^{exp} [funguje v grafickém uživatelském rozhraní]

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
		BIN_SHR(n, exp)	n / 2 exp Výsledek se zobrazí jako zaokrouhlené celé číslo. [funguje v grafickém uživatelském rozhraní]
		BIN_XOR(x, y [, z . . .])	"a" i "b" jsou uvedeny formou binárního zápisu. Aby výsledek pro danou pozici byl „1“, musí být na stejné pozici v "a" nebo v "b" hodnota „1“. BIN_XOR(3,5) vrátí 6; 0011 X OR 0101 = 011 0 [funguje v grafickém uživatelském rozhraní]
CEILING(d)	Určuje nejmenší celé číslo, které není menší než d. [funguje v grafickém uživatelském rozhraní]	CEIL(d) CEILING(d)	
COS(d)	Vrací kosinus. [funguje v grafickém uživatelském rozhraní]	COS(radians)	Radiány lze také znázornit pomocí úhlu (zde pro jednotkovou kružnici): radiány = (2 * PI () * úhly / 360)
		COSH(d)	Vrací hyperbolický kosinus.
COT(d)	Vrací kotangens. [funguje v grafickém uživatelském rozhraní]	COT(d)	
DEGREES(d)	Převádí radiány na stupně. [funguje v grafickém uživatelském rozhraní]		
EXP(d)	Vrací hodnotu d-té mocniny čísla e (e: (2.718...)). [funguje v grafickém uživatelském rozhraní]	EXP(d)	
FLOOR(d)	Vrací největší celé číslo, které není větší než d. [funguje v grafickém uživatelském rozhraní]	FLOOR(d)	

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
LOG(d)	Vrací přirozený logaritmus se základem 'e'. [funguje v grafickém uživatelském rozhraní]	LN(d)	
LOG10(d)	Vrátí logaritmus o základu 10. [funguje v grafickém uživatelském rozhraní]	LOG10(d)	
		LOG(base, d)	Vrátí logaritmus čísla d při základu base.
MOD(a, b)	Vrací zbytek jako celé číslo, které vznikne dělením 2 celých čísel. MOD(11,3) vrátí 2, protože $3 * 3 + 2 = 11$ [funguje v grafickém uživatelském rozhraní]	MOD(a, b)	
PI()	Vrací π (3.1415...) [funguje v grafickém uživatelském rozhraní]	PI()	
POWER(a, b)	a^b , POWER(2,3) = 8, protože $2^3 = 8$ [funguje v grafickém uživatelském rozhraní]	POWER(x, y)	
RADIANS(d)	Převádí stupně na radiány. [funguje v grafickém uživatelském rozhraní]		
EDGE()	Vrátí náhodné číslo x větší nebo rovno 0,0 a menší než 1,0. [funguje v grafickém uživatelském rozhraní]	EDGE()	
ROUND(a, b)	Zaokrouhluje číslo a na b číslic za desetinnou čárkou. [funguje v grafickém uživatelském rozhraní]	ROUND(d [, místa])	Zaokrouhluje po zadaném počtu číslic od desetinné čárky. ROUND(123.45, 1) vrátí 123.50 ROUND(123.45, -2) vrací 100.00 [funguje v grafickém uživatelském rozhraní]

<i>HSQLDB</i>		<i>Firebird</i>	
<i>Syntaxe</i>	<i>Popis</i>	<i>Syntaxe</i>	<i>Popis</i>
ROUNDMAGIC (d)	Řeší problémy se zaokrouhlováním způsobené čísla s pohyblivou řádovou čárkou. 3,11 – 3,1 – 0,01 nemusí být přesně 0, ale v grafickém uživatelském rozhraní se zobrazuje jako 0. ROUNDMAGIC ji změní na skutečnou hodnotu 0. [funguje v grafickém uživatelském rozhraní]		
SIGN(d)	Vrací -1, pokud je 'd' menší než 0; 0, pokud je 'd' rovno 0; a 1, pokud je 'd' větší než 0. [funguje v grafickém uživatelském rozhraní]	SIGN(d)	
SIN(A)	Vrací sinus úhlu v radiánech. [funguje v grafickém uživatelském rozhraní]	SIN(radians)	
		SINH(d)	Vrací hyperbolický sinus.
SQRT(d)	Vrací druhou odmocninu. [funguje v grafickém uživatelském rozhraní]	SQRT(d)	
TAN(A)	Vrací tangens úhlu v radiánech. [funguje v grafickém uživatelském rozhraní]	TAN(radians)	
		TANH(d)	Vrací hyperbolický tangens.
TRUNCATE (a,b)	Ořízne znaky „a“ až „b“ za desetinnou čárkou. TRUNCATE(2.37456.2) = 2.37 [funguje v grafickém uživatelském rozhraní]	TRUNC(d[,jobs])	Nastaví na 0 po zadaném počtu číslic od desetinné čárky. TRUNC (123.45, 1) vrací 123. 4 0 ROUND (123.45, -2) vrací 100.00 [funguje v grafickém uživatelském rozhraní]

Textové funkce

Tabulka 67: Textové funkce

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
ASCII(s)	Vrací kód ASCII prvního písmene řetězce. [funguje v grafickém uživatelském rozhraní]	ASCII_VAL ('s')	Vrátí číselnou hodnotu, která odpovídá zadanému znaku. [funguje v grafickém uživatelském rozhraní]
BIT_LENGTH (str)	Vrací délku textu str v bitech. [funguje v grafickém uživatelském rozhraní]	BIT_LENGTH ('s')	
CHAR(c)	Vrátí písmeno, které patří zadanému kódu ASCII. Nejde jen o písmena, ale také o řídicí znaky. Znak CHAR (13) vytvoří v dotazu zalomení řádku, které je viditelné ve víceřádkových polích formuláře nebo v sestavách. [funguje v grafickém uživatelském rozhraní]	ASCII_CHAR (n)	Vrátí písmeno, které patří zadanému kódu ASCII. Nejde jen o písmena, ale také o řídicí znaky. [funguje v grafickém uživatelském rozhraní]

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
CHAR_LENGTH (str)	Vrací délku textu v písmenech. [funguje v grafickém uživatelském rozhraní]	CHAR_LENGTH ('s') CHARACTER_LENGTH('s')	
		CHAR_TO_UUID	Převede 36znakový univerzálně jedinečný identifikátor (UUID) na 16bajtový UUID .(vypíše nečitelné znaky).
CONCAT (STR1,STR2)	Spojí str1 + str2. [funguje v grafickém uživatelském rozhraní]		
DIFFERENCE (s1,s2)	Zobrazuje zvukový rozdíl mezi s1 a s2. Výstupem je pouze celé číslo. 0 znamená stejný zvuk. Takže „for“ a „four“ je rovno 0 (zní stejně), shortening a seasoning je rovno 1 a mouth a moon je opět 0. [funguje v grafickém uživatelském rozhraní]		
HEXTORAW (s1)	Převádí hexadecimální kód na jiné znaky. [funguje v grafickém uživatelském rozhraní]		
INSERT(s, start, len,s2)	Vrací textový řetězec s nahrazenou částí textu. Počínaje „start“ se z textu „s“ vyřízne délka „len“ a nahradí se textem „s2“. INSERT('Bundesbahn', 3, 4, 'mmel') převede Bundesbahn na Bummelbahn, přičemž délka vloženého textu může být větší než délka odstraněného textu, aniž by to způsobilo nějaké problémy. Takže INSERT('Bundesbahn', 3, 5, 's und B') dává výsledek 'Bus und Bahn'. [funguje v grafickém uživatelském rozhraní]	OVERLAY ('s' PLACING 's2' FROM pos [FOR délka])	Pokud je počáteční pozice nastavena tak, že je větší nebo rovna aktuálnímu textu „s“, pak se k „s“ přímo připojí „s2“. OVERLAY „Bundesbahn“ PLACING „mmel“ FROM 3 FOR 4) změní „Bundesbahn“ na „Bummelbahn“, přičemž délka vloženého textu může být delší než délka vystřiženého textu. Takže OVERLAY („Bundesbahn“ PLACING ' s a B ' FROM 3 FOR 5) vede k „Bus und Bahn“ . [nefunguje v grafickém uživatelském rozhraní]
LCASE(s)	Převede řetězec na malá písmena. [funguje v grafickém uživatelském rozhraní]		

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
LEFT(s, count)	Vrátí počet znaků zadaných pomocí count od začátku řetězce s. [funguje v grafickém uživatelském rozhraní]	LEFT('s', length)	
LENGTH(s)	Vrací délku řetězce. [funguje v grafickém uživatelském rozhraní]		
LOCATE (search,s [,start])	Vrátí první shodu výrazu search v řetězci s. Shoda je uvedena číselně: (1 = nalezeno, 0 = nenalezeno). Zadání počátečního bodu v řetězci je nepovinné. [funguje v grafickém uživatelském rozhraní]	POSITION ('s2' IN 's') POSITION („s2“, „s“ [, S tartpos ition])	
POSITION (<řetězcový výraz> IN <řetězcový výraz>)	Pokud je první řetězec obsažen v druhém, vrátí se pozice prvního řetězce, jinak se zobrazí 0. To lze použít místo možnosti vyhledávání pomocí LIKE. [funguje v grafickém uživatelském rozhraní]		
LTRIM(s)	Odstraní z textu počáteční mezery a netisknutelné znaky. [funguje v grafickém uživatelském rozhraní]		
OCTET_LENGTH (str)	Vrací délku textového řetězce v bajtech. V zásadě to odpovídá dvojnásobku délky ve znacích. [funguje v grafickém uživatelském rozhraní]	OCTET_LENGTH (str)	Vrátí skutečný počet znaků se započtením mezer. Počet závisí na znakové sadě. UTF-8 potřebuje dva bajty pro speciální znaky.
RAWTOHEX(s1)	Převede na hexadecimální zápis; opačná funkce než HEXTORAW(). [funguje v grafickém uživatelském rozhraní]		
REPEAT(s, count)	Opakuje textový řetězec 's' count-krát. [funguje v grafickém uživatelském rozhraní]		

<i>HSQLDB</i>		<i>Firebird</i>	
<i>Syntaxe</i>	<i>Popis</i>	<i>Syntaxe</i>	<i>Popis</i>
REPLACE(s, replace,s2)	Nahradí všechny existující výskyty slova 'replace' v řetězci 's' textem 's2'. [funguje v grafickém uživatelském rozhraní]	REPLACE('s', 's2', replacement)	REPLACE ('Schule', 'ul', 'eib') převede 'Schule' na 'Schieibe'. Pokud se 's2' v 's' nevyskytuje, nebude nahrazeno nic. Pokud se v položce 's2' nebo náhradě objeví N2, výsledkem je NULL.
		LPAD('s', celková délka [,znaků])	LPAD ('Hello', 8 , '+') = +++ Hello Umístí libovolné znaky před řetězec, dokud není dosaženo celkové délky. Pokud je celková délka menší než délka řetězce, řetězec se vpravo odřízne. Pokud třetí parametr zůstane prázdný, umístí se před něj mezery.
		RPAD('s', celková délka [,znaků])	RPAD ('Hello', 8 , '+') = Hello +++ Umístí libovolné znaky za řetězec, dokud není dosaženo celkové délky. Pokud je celková délka menší než délka řetězce, řetězec se vpravo odřízne. Pokud třetí parametr zůstane prázdný, umístí se za něj mezery.
		REVERSE('s')	Úplně invertuje řetězec. To může být užitečné, například pokud chceme řadit podle koncovek znaků (např. koncovek domén).
RIGHT(s, count)	Obrácená funkce k LEFT; vrací počet znaků zadaný počtem od konce řetězce. [funguje v grafickém uživatelském rozhraní]	RIGHT('s', length)	
RTRIM(s)	Odstraní všechny mezery a netisknutelné znaky na konci řetězce. [funguje v grafickém uživatelském rozhraní]		

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
SOUNDEX(s)	Vrací kód o 4 znacích, který by měl odpovídat zvuku řetězce 's' – odpovídá funkci DIFFERENCE (). [funguje v grafickém uživatelském rozhraní]		
SPACE(count)	Vrací počet mezer zadaný v parametru count. [funguje v grafickém uživatelském rozhraní]		
SUBSTR(s, start [,len])	Zkratka pro SUBSTRING. [funguje v grafickém uživatelském rozhraní]		
SUBSTRING (s,start [,len])	Vrací řetězec 's' od počáteční pozice. (1 = vlevo). Pokud je délka vynechána, je vrácen celý řetězec. [funguje v grafickém uživatelském rozhraní]		
SUBSTRING (<řetězcový výraz> FROM <číselný výraz> [FOR <číselný výraz>])	Vrátí část řetězce od počáteční pozice zadané ve FROM, volitelně v délce zadané ve FOR. Pokud se například v poli "Name" objeví "Roberta", výsledkem SUBSTRING ("Name" FROM 3 FOR 3) je podřetězec "bert". [funguje v grafickém uživatelském rozhraní]	SUBSTRING ('s' FROM start position [FOR length])	
TRIM ({{{LEADING TRAILING BOTH}} FROM <řetězcový výraz>)	Speciální znaky a mezery, které nelze vytisknout, jsou odstraněny. [funguje v grafickém uživatelském rozhraní]	TRIM ([Where 's2' FROM] 's')	Where: BOTH LEADING TRAILING standardem je zde BOTH pro obě strany 's'. s2: Znak, který má být odstraněn. Ve výchozím nastavení jsou to mezery (' '), ale mohou to být i jiné znaky. TRIM (TRAILING '!' FROM 'Hallo!') vrátí 'Hallo'
UCASE(s)	Převede řetězec na velká písmena. [funguje v grafickém uživatelském rozhraní]		
LOWER(s)	Jako LCASE(s) [pracuje v grafickém uživatelském rozhraní]	LOWER('s')	

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
UPPER(s)	Jako UCASE(s). [funguje v grafickém uživatelském rozhraní]	UPPER('s')	
		UUID_TO_CHAR ('s')	Převede 16bajtový UUID na 36znakový formát ASCII.

Funkce data a času

Tabulka 68: Funkce data a času

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
		DATEADD (n DAY TO date) DATEADD (DAY, n, date)	n je celé číslo a pro odčítání může být i záporné. YEAR MONTH WEEK DAY HOUR MINUTE SECOND MILLISECOND se používají jako termíny pro časový interval. Jako termín data použijeme buď datum / datumové pole, čas / časové pole, nebo časové razítko.
DATEDIFF (string, datetime1, datetime2)	Rozdíl mezi dvěma daty nebo časy. Položka v řetězci rozhoduje o tom, v jaké jednotce se rozdíl zobrazí: ms“ = „milisekunda“, „ss“ = „sekunda“, „mi“ = „minuta“, „hh“ = „hodina“, „dd“ = „den“, „mm“ = „měsíc“, „yy“ = „rok“. Lze použít dlouhou i krátkou verzi řetězce. [funguje v grafickém uživatelském rozhraní]	DATEDIFF (DAY FROM date TO date) DATEDIFF (DAY, date, date)	Viz DATEADD.

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
EXTRACT ({YEAR MONTH DAY HOUR MINUTE SECOND} FROM <date or time value>)	Může nahradit mnoho funkcí data a času. Vrátí rok, měsíc, den atd. z hodnoty data nebo denního času. EXTRACT (DAY FROM "date") zobrazí den v měsíci. [funguje v grafickém uživatelském rozhraní]	EXTRACT ({YEAR MONTH WEEK DAY WEEKDAY YEARDAY HOUR MINUTE SECOND MILLISECOND } FROM <hodnota data nebo času>)	WEEKDAY Sunday = 0 YEARDAY January 1st = 0 WEEK 1. týden: min. 4 dny v roce
DAY(date)	Vrátí den v měsíci (1-31). [funguje v grafickém uživatelském rozhraní]		
DAYNAME (date)	Vrátí anglický název dne. [funguje v grafickém uživatelském rozhraní]		
DAYOFMONTH (date)	Vrací den v měsíci (1-31), synonymum pro DAY(). [funguje v grafickém uživatelském rozhraní]		
DAYOFWEEK (date)	Vrátí den v týdnu jako číslo (1 znamená neděli.) [funguje v grafickém uživatelském rozhraní]		
DAYOFYEAR (date)	Vrátí den v roce (1-366). [funguje v grafickém uživatelském rozhraní]		
HOUR(time)	Vrátí hodinu (0-23). [funguje v grafickém uživatelském rozhraní]		
MINUTE(time)	Vrátí minutu (0-59). [funguje v grafickém uživatelském rozhraní]		
MONTH(date)	Vrátí měsíc (1-12). [funguje v grafickém uživatelském rozhraní]		
MONTHNAME (date)	Vrátí anglický název měsíce. [funguje v grafickém uživatelském rozhraní]		

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
QUARTER (date)	Vrátí čtvrtletí roku (1-4). [funguje v grafickém uživatelském rozhraní]		
SECOND(time)	Vrátí sekundy času (0-59). [funguje v grafickém uživatelském rozhraní]		
WEEK(date)	Vrátí týden v roce (1-53). [funguje v grafickém uživatelském rozhraní]		
YEAR(date)	Vrátí rok ze zadaného data. [funguje v grafickém uživatelském rozhraní]		

Funkce připojení k databázi

Tabulka 69: Funkce připojení k databázi

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
DATABASE ()	Vrací cestu a název databáze patřící k tomuto připojení. [funguje v grafickém uživatelském rozhraní]		
		CURRENT_TRANSACTION	SELECT CURRENT_TRANSACTION FROM RDB \$ DATABASE vrátí jedinečný identifikátor transakce jako celé číslo.
		CURRENT_CONNECTION	SELECT CURRENT_CONNECTION FROM RDB \$ DATABASE vrátí celočíselnou hodnotu pro aktuální připojení.
		CURRENT_ROLE	SELECT CURRENT_ROLE FROM RDB \$ DATABASE odráží roli aktuálního uživatele. Pokud není definována žádná role, výsledkem je NONE.

HSQldb		Firebird	
Syntaxe	Popis	Syntaxe	Popis
		RDB \$ SET_CONTEXT ('<namespace>', '<název proměnné>', value NULL)	Namespace: USER_SESSION USER_TRANSACTION Název proměnné může mít maximálně 80 znaků a value může mít maximálně 255 znaků.
CURRENT_USER	Standardní funkce SQL, synonymum funkce USER(). Je třeba poznamenat, že zde nejsou závorky. [funguje v grafickém uživatelském rozhraní]	CURRENT_USER	
USER()	Vrací uživatelské jméno tohoto připojení. Uživatelské jméno je důležité, pokud má být databáze převedena do externí databáze. [SQL direct – nefunguje s grafickým uživatelským rozhraním]	USER	
IDENTITY()	Vrací poslední hodnotu pole automatická hodnota, které bylo vytvořeno v aktuálním spojení. Používá se při programování maker k vytvoření cizího klíče pro jinou tabulku z primárního klíče vytvořeného pro jednu tabulku. [funguje v grafickém uživatelském rozhraní]	GEN_ID (název generátoru, <krok>)	Automatické hodnoty se vytvářejí pomocí generátoru. Velikost kroku by zde měla být uvedena jako 1. V zásadě je ale možná jakákoliv celočíselná hodnota. new.rec_id = gen_id (gen_recnum, 1);

Funkce systému

Tabulka 70: Systémové funkce

HSQldb		Firebird		
Syntaxe	Popis	Syntaxe	Popis	
IFNULL (exp, value)	<p>Pokud má parametr exp hodnotu NULL, je vrácena hodnota parametru value, jinak je vrácena hodnota parametru exp. Místo toho lze jako rozšíření použít také COALESCE (). Parametry exp a value musí mít stejný datový typ.</p> <p>Funkce IFNULL je důležitá, pokud jsou pole vzájemně propojena pomocí faktury nebo CONCAT. Obsah výsledku by byl NULL, i když je NULL pouze jedna hodnota.</p> <p>"Last name" ',' "first name" by vedlo k prázdnému poli pro osoby, které například nemají položku "first name", tj. NULL.</p> <p>"Last Name" IFNULL (',' "First Name", ") by místo toho vypsalo jen "Last Name".</p> <p>[funguje v grafickém uživatelském rozhraní]</p>			
CASE WHEN (exp, v1, v2)	<p>Pokud má parametr exp hodnotu true, vrátí se v1, jinak v2. Místo toho lze použít také CASE WHEN. CASEWHEN ("a" > 10, 'goal reached', 'still practice') vrátí 'goal reached', pokud je obsah pole "a" větší než 10.</p> <p>[funguje v grafickém uživatelském rozhraní]</p>	IIF (<podmínka> , v1, v2)		
CONVERT (term, type)	<p>Převede term na jiný datový typ uvedený v type. CONVERT ("a", DECIMAL (5,2)) vytvoří z pole "a" pole s 5 číslicemi včetně 2 desetinných míst. Pokud je číslo příliš velké, vypíše se chyba.</p> <p>[funguje v grafickém uživatelském rozhraní]</p>			
CAST (term AS type)	<p>Synonymum pro CONVERT ()</p> <p>[funguje v grafickém uživatelském rozhraní]</p>	CAST (term AS type)	From Číselné typy	To Číselné typy [VAR] CHAR BLOB

<i>HSQLDB</i>		<i>Firebird</i>	
<i>Syntaxe</i>	<i>Popis</i>	<i>Syntaxe</i>	<i>Popis</i>
			[VAR] CHAR BLOB
			[VAR] CHAR BLOB Numeric types DATE TIME TIMESTAMP
		DATE TIME	[VAR] CHAR BLOB TIMESTAMP
		TIMESTAMP	[VAR] CHAR BLOB DATE TIME
COALESCE (expr1, expr2, expr3, ...)	Pokud expr1 není NULL, zobrazí se expr1, jinak se zkontroluje expr2, pak expr3 atd. Všechny výrazy musí mít alespoň podobný datový typ. Jedná se o alternativní reprezentaci celých čísel a čísel s plovoucí desetinnou čárkou, nikoli však také data nebo hodnoty času. [funguje v grafickém uživatelském rozhraní]	COALESCE (expr1, expr2 [, expr3 ...]	
NULLIF (v1, v2)	Pokud se v1 rovná v2, vrátí se null, jinak se vrátí hodnota v1. Hodnoty v1 a v2 musí být typově srovnatelné. [funguje v grafickém uživatelském rozhraní]	NULLIF (v1, v2)	

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
CASE v1 WHEN v2 THEN v3 [ELSE v4] END	Pokud se v1 rovná v2, provede se v3. V opačném případě se provede v4 nebo NULL, pokud není definováno ELSE. [SQL direct – nefunguje s grafickým uživatelským rozhraním]	DECODE (test expression , expression n , result [, expressi on2 , Earni ngs2 ...] [, default expression)	DECODE (UPPER ("gender"), 'M', 'male', 'F', 'female', 'unknown')
CASE WHEN expr1 THEN v1 [WHEN expr2 THEN v2] [ELSE v4] END	Pokud má expr1 hodnotu true, je vrácena hodnota v1 [Volitelně lze zadat další případy]. V opačném případě je reprodukováno v4 nebo NULL, pokud není formulováno ELSE. CASE WHEN DAYOFWEEK ("date") = 1 THEN 'Sunday' WHEN DAYOFWEEK ("date") = 2 THEN 'Monday'... END by mohl vypsát den v týdnu pomocí SQL, který je jinak ve funkci k dispozici pouze v angličtině. [funguje v grafickém uživatelském rozhraní]		DECODE (EXTRACT (WEEK DAY FROM "date"), 0 , ' Sunday ', 1 , ' Monday ', ' etc. ')
		GEN_UUID ()	Vrátí jedinečné ID jako 16bajtovou sadu znaků.
		HASH (s)	Vrací hodnotu hash libovolně dlouhého řetězce. Hodnoty hash stejných řetězců znaků musí být stejné.
		MAXVALUE (expr [, expr ...])	Vrací maximální hodnotu seznamu hodnot. Pracuje s řetězci, číselnými hodnotami, daty nebo časovými hodnotami.
		MINVALUE (expr [, expr ...])	Vrací minimální hodnotu seznamu hodnot. Pracuje s řetězci, číselnými hodnotami, daty nebo časovými hodnotami.

Agregační funkce

Tabulka 71: Agregací funkce (zejména s GROUP BY)

HSQLDB		Firebird	
Syntaxe	Popis	Syntaxe	Popis
		MAX (expr)	Maximální hodnota pole v tabulce.
		MIN (expr)	Minimální hodnota pole v tabulce.
		LIST ([ALL DISTINCT] 's' , [' s2'])	Připojí pole několika datových záznamů k jednomu poli s odpovídajícím termínem připojení „s2“. [funguje v grafickém uživatelském rozhraní]

Proměnné

Tabulka 72: Proměnné (v závislosti na počítači)

HSQLDB		Firebird	
Proměnná	Popis	Proměnná	Popis
CURRENT_TIME	Synonymum pro CURTIME (), standard SQL. [funguje v grafickém uživatelském rozhraní]	CURRENT_TIME	Čas v hodinách, minutách a sekundách. CURRENT_TIME (3) rovněž udává milisekundy.
CURTIME ()	Vrací aktuální čas. [funguje v grafickém uživatelském rozhraní]		
CURRENT_TIMES TAMP	Synonymum pro NOW (), standard SQL. [funguje v grafickém uživatelském rozhraní]	CURRENT_TIMES TAMP [(přesnost)]	Zadání času s datem a milisekundami. Přesnost lze nastavit pomocí [0 1 2 3]. Standardem jsou 3 desetinná místa. SELECT CURRENT_TIMESTAMP (2) FROM RDB \$ DATABASE vrátí časové razítko s desetinnými a setinami sekundy (2 desetinná místa).

HSQLDB		Firebird	
Proměnná	Popis	Proměnná	Popis
NOW ()	Vrátí aktuální datum a čas společně jako časové razítko. Místo toho lze použít také CURRENT_TIMESTAMP. [funguje v grafickém uživatelském rozhraní]	CAST ('NOW' AS DATE TIME TIMESTAMP) nebo DATE 'NOW'	Samotné 'NOW' je chápáno jako řetězec. Vhodným převodem se z něj stane datum, čas nebo časové razítko (každé s 1/1000 s). Zkrácená forma v grafickém uživatelském rozhraní nefunguje.
CURRENT_DATE	Synonymum pro CURDATE (), standard SQL. [funguje v grafickém uživatelském rozhraní]	CURRENT_DATE	
CURDATE ()	Vrací aktuální datum. [funguje v grafickém uživatelském rozhraní]		

Operátory a příkazy

Tabulka 73: Operátory a příkazy

HSQLDB		Firebird	
Operátor nebo příkazy	Popis	Operátor nebo příkazy	Popis
'str1' 'str2' 'str3' or 'str1' + 'str2' + 'str3'	Spojí str1 + str2 + str3; jednodušší alternativa ke CONCAT. [funguje v grafickém uživatelském rozhraní]	's1' ' s 2 ' [' s3 '...]	Připojí s1, s2 atd. k novému řetězci [funguje v grafickém uživatelském rozhraní]
		ALL	
		ANY / SOME	
		IN ()	
		IS [NOT] DISTINCT FROM	Výsledkem je „ano“ nebo „ne“.

<i>HSQLDB</i>		<i>Firebird</i>	
<i>Operátor nebo příkazy</i>	<i>Popis</i>	<i>Operátor nebo příkazy</i>	<i>Popis</i>
		NEXT VALUE FOR název sekvence	Viz GEN_ID (), ale neumožňuje jiné kroky než 1.
		SOME	

Datové typy pro editor tabulek

Celočíselné datové typy

Tabulka 74: Datové typy celých čísel

<i>Typ</i>	<i>Možnost</i>	<i>HSQLDB</i>	<i>Firebird</i>	<i>Rozsah</i>	<i>Velikost v paměti</i>
Tiny integer	TINYINT	TINYINT		28 = 256 – 128 to + 127	1 bajt
Small integer	SMALLINT	SMALLINT	SMALLINT	216 = 65536 – 32768 to + 32767	2 bajty
integer	INTEGER	INTEGER INT	INTEGER	232 = 4294967296 – 2147483648 to + 2147483647	4 bajty
BigInt	BIGINT	BIGINT	BIGINT	264 (–263 to +263)	8 bajtů

Datové typy s pohyblivou řádovou čárkou

Tabulka 75: Datové typy čísla s pohyblivou desetinnou čárkou

<i>Typ</i>	<i>Možnost</i>	<i>HSQLDB</i>	<i>Firebird</i>	<i>Rozsah</i>	<i>Požadavky na paměť</i>
Desetinné číslo	DECIMAL	DECIMAL	DECIMAL (n, m)	Neomezeně, přes GUI až na 50 číslic, nastavitelné, pevná desetinná místa, vysoká přesnost	2, 4 nebo 8 bajtů
Číslo	NUMERIC	NUMERIC	NUMERIC (n, m)	Neomezeně, přes GUI až na 50 číslic, nastavitelné, pevná desetinná místa, vysoká přesnost	2, 4 nebo 8 bajtů

Typ	Možnost	HSQLDB	Firebird	Rozsah	Požadavky na paměť
Float	FLOAT	(místo toho se používá DOUBLE)	FLOAT	3.4 * 10 ⁻³⁸ to 3.4 * 10 ³⁸ nastavitelný, ne přesný, maximálně 7 desetinných míst	4 bajty
Reálná část	REAL	REAL			
Double	DOUBLE	DOUBLE [PRECISION] FLOAT	DOUBLE PRECISION	1, 7 * 10 ⁻³⁰⁸ až 1, 7 * 10 ³⁰⁸ nastavitelné, ne přesné, maximálně 15 desetinných míst	8 bajtů

Datové typy textu

Tabulka 76: Datové typy textu

Typ	Možnost	HSQLDB	Firebird	Rozsah	Požadavky na paměť
Text	VARCHAR	VARCHAR	VARCHAR (n)	Nastavitelný	Proměnná Firebird: 1 až 32767 bajtů
Text	VARCHAR_IGNORE CASE	VARCHAR_IGNORE CASE		Nastavitelné, ovlivňuje třídění, ignoruje rozdíly mezi velkými a malými písmeny	proměnná
Text (pevný)	CHAR	CHAR CHARACTER		Nastavitelný, zbytek textu je vyplněn mezerami.	pevná
Memo	LONGVARCHAR	LONGVARCHAR	BLOB (BLOB SUB_TYPE 1)		proměnná Firebird: <32 GB

Datové typy datum/čas

Tabulka 77: Datové typy datum/čas

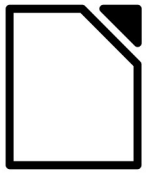
Typ	Možnost	HSQLDB	Firebird	Rozsah	Požadavky na paměť
Datum	DATE	DATE	DATE		4 bajty

Typ	Možnost	HSQLDB	Firebird	Rozsah	Požadavky na paměť
Čas	TIME	TIME	TIME	Firebird: 0:00 až 23:59,9999	4 bajty
Datum/čas	TIME STAMP	TIMESTAMP DATE TIME	TIME STAMP	Nastavitelné (HSQLDB: 0, 6 – 6 znamená s milisekundami)	8 bajtů

Ostatní typy dat

Tabulka 78: Další datové typy

Typ	Možnost	HSQLDB	Firebird	Rozsah	Požadavky na paměť
Ano Ne	BOOLEAN	BOOLEAN BIT			
Binární pole (pevné)	BINARY	BINARY		Stejně jako integer	pevná
Binární pole	VARBINARY	VARBINARY		Stejně jako integer	proměnná
Image	LONGVARBINARY	LONGVARBINARY	BLOB SUB_TYPE 0 BLOB SUB_TYPE binary	Stejně jako integer	proměnná, určená pro větší obrázky Firebird: <32 GB
OTHER	OTHER	OTHER OBJECT			



Příručka aplikace Base

Správa dat

O této knize:

Tato kniha je určena začátečníkům i pokročilým uživatelům LibreOffice Base. Pokud jste Base nikdy předtím nepoužívali nebo se chcete seznámit se všemi součástmi LibreOffice, přečtěte si nejprve knihu Začínáme s LibreOffice.

Tato kniha vás seznámí s nejběžnějšími funkcemi LibreOffice Base:

Vytvoření databáze,

Vstup a výstup dat,

Práce s tabulkami, formuláři, dotazy a sestavami,

Používání maker,

Údržba databáze,

a mnoho dalšího.

O autorech:

Tato kniha byla napsána dobrovolníky z komunity LibreOffice. Zisk z prodeje tištěného vydání bude použit ve prospěch komunity.

Verzi této knihy ve formátu PDF si můžete zdarma stáhnout z:

<https://documentation.libreoffice.org>

O LibreOffice:

LibreOffice je bezplatný, svobodný a otevřený balík aplikací pro osobní produktivitu od nadace The Document Foundation. Funguje v systémech Windows, MacOS a GNU/Linux. Podpora a dokumentace od rozsáhlé a oddané komunity uživatelů, přispěvatelů a vývojářů je zdarma.

I vy se můžete zapojit do dobrovolnické práce v mnoha oblastech: vývoj, zajištění kvality, dokumentace, překlady, podpora uživatelů a další.

LibreOffice Community si můžete stáhnout ze stránek:

<https://cs.libreoffice.org/download/>

Pro komerční/podnikové použití: Aplikace LibreOffice Enterprise mohou vypadat a chovat se jinak než edice Community popsaná v této knize.

Průvodce edicemi LibreOffice Enterprise naleznete u odborných partnerů, kteří je vytvořili.