Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Srovnání volně šiřitelných relačních SŘBD z pohledu databázového programátora

Plzeň 2016 David Langmaier

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 4. května 2016 . . . . . .

David Langmaier

ABSTRAKT

Cílem této práce je porovnat vybrané relační SŘBD – PostgreSQL, MariaDB, Firebird a SQLite. Předmětem srovnávání jsou klientské aplikace, grafické nástroje, webová rozhraní a dialekty jazyka SQL. Pro účely porovnávání jsou pro každou oblast vytyčena vhodná kritéria, která souvisejí se základními potřebami databázového programátora. Součástí práce je také návod k vytvoření části testovací databáze pomocí grafického rozhraní. Díky této testovací databázi lze také porovnat SQL dialekty. Na základě zjištěných údajů se začínající či středně pokročilí databázoví programátoři mohou snáze rozhodnout, který systém pro svou práci zvolit, aby pokryli své osobní požadavky.

KLÍČOVÁ SLOVA

Relační SŘBD, konzolový klient, grafický nástroj, webové rozhraní, SQL, databáze

ABSTRACT

The purpose of this thesis is to compare selected relational DBMS – PostgreSQL, MariaDB, Firebird and SQLite. The subjects of comparison are client applications, graphical tools, web interfaces and SQL dialects. For purpose of the comparison are outlined appropriate criteria that match the basic needs of a database programmer. The work also includes instructions to create a part of the test database using a graphical interface. With this test database you can also compare SQL dialects. Based on the acquired informations beginners and intermediate database programmers can easily decide which system to choose to cover their own personal requirements.

KEYWORDS

Relational DBMS, command line client, GUI tool, web interface, SQL, database

LANGMAIER, David *Srovnání volně šiřitelných relačních SŘBD z pohledu databázového programátora*: bakalářská práce. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 2016. 58 s. Vedoucí práce byl Ing. Martin Zíma, Ph.D.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Martinu Zímovi, Ph.D. za odborné vedení, vstřícný přístup, trpělivost a podnětné návrhy k práci.

OBSAH

SEZNAM TABULEK

1 ÚVOD

Cílem této práce je výběr vhodných zástupců volně šiřitelných relačních systémů řízení báze dat (SŘBD) a navržení vhodných kritérií, podle kterých budou vybrané systémy následně porovnávány. Při výběru jednotlivých SŘBD bude kladen důraz také na to, aby byl vývoj těchto systémů stále aktivní a jejich vývoj by zřejmě měl pokračovat i v následujících letech.

Kritéria budou navržena tak, aby výsledné porovnání co nejvíce odpovídalo potřebám a požadavkům databázového programátora. Systémy tedy budou zkoumány z pohledu uživatele a tato studie bude zaměřena především na samotnou práci s jednotlivými SŘBD. Mezi předpokládaná kritéria se řadí například srovnání klientských aplikací (konzolových, grafických a webových) nebo využívaný dialekt jazyka SQL.

Součástí práce bude také návrh a tvorba netriviální testovací databáze. Tato databáze bude vytvořena ve všech vybraných systémech a bude na ní následně provedena demonstrace vhodných vlastností porovnávaných SŘBD (zejména v oblasti její tvorby pomocí různých grafických rozhraní). Export databáze může také sloužit k porovnání dialektů SQL (např. odlišné pojmenování datových typů).

2 HISTORIE RELAČNÍCH SŘBD

2.1 MySQL

Relační systém řízení báze dat MySQL byl vytvořen švédskou společností MySQL AB a jeho první vydaná verze byla zveřejněna v roce 1995. V průběhu let si tento systém vytvořil skutečně skvělou pověst a stal se jedním z nejvýznamnějších produktů ve svém oboru. MySQL byl druhým nejpoužívanějším relačním SŘBD a v kategoriích open-source[1] systémů naprosto dominoval. [1]

V současné době však popularita MySQL velmi výrazně klesá a systém ztrácí své nejvýznamnější dlouholeté klienty. Tento open-source systém totiž koupila společnost Oracle, která se věnuje především komerčnímu vývoji. Licence na jejich produkty jsou tedy zpoplatněny. V minulosti již MySQL přestaly využívat například společnosti Wikimedia Foundation, Red Hat nebo Google. Všechny tyto firmy opustily MySQL především kvůli tomu, že ji Oracle odsunul do pozadí, její vývoj se výrazně zpomalil, neopravují se některé důležité chyby a díky tomu tento systém již zastarává. Budoucí vývoj a směr, kterým se bude MySQL ubírat, je tedy velmi nejistý. [2]

2.2 MariaDB

MariaDB vznikla jako fork MySQL ještě před tím, než ji odkoupila společnost Oracle. Michaelovi Wideniusovi, jednomu z hlavních zakladatelů MySQL, se totiž nelíbilo, jakým směrem se projekt ubírá, a proto se rozhodl začít pracovat na projektu novém. Založil tedy nejprve společnost The Monty Program a následně, s dalšími bývalými zakladateli MySQL, i organizaci MariaDB Foundation. V této organizaci pracuje i mnoho dalších pracovníků, kteří se podíleli na vývoji MySQL. Lze tedy říci, že původní myšlenka tohoto systému pokračuje právě v podobě MariaDB.

MariaDB je s MySQL kompatibilní, díky čemuž ji už začala nahrazovat v celé řadě projektů. Dvěma z nejvýznamnějších společností, které přešly na MariaDB, jsou již dříve zmíněné firmy Wikimedia Foundation a Google. Základními charakteristikami a nejdůležitějšími rysy, které si chce MariaDB udržet, jsou totiž její otevřený vývoj, zkušená vývojářská komunita a dobrá uživatelská podpora. Tyto vlastnosti jsou pro mnoho velkých projektů tak stěžejní, že s náhradou MySQL za MariaDB vůbec neváhají. [3]

[1] O open-source softwaru hovoříme tehdy, je-li jeho zdrojový kód volně dostupný veřejnosti (technicky i legálně). [4]

2.3 Ingres a PostgreSQL

Prvotní vývoj PostgreSQL začal na akademické půdě už v roce 1973. V té době byl projekt pod záštitou Univerzity Berkeley v Kalifornii a nesl označení Ingres (resp. post-Ingres). Na to, že byl systém vyvíjen na univerzitě a v poměrně skromných podmínkách, zaznamenal velký úspěch a velmi dobré ohlasy ze strany odborníků. Konkurence však byla veliká. Kvůli špatné kvalitě kódu (nedodržování konvencí) a tím pádem jeho složité udržitelnosti, bylo vskutku obtížné držet krok s ostatními. Projekt pokračoval pod záštitou Berkeley až do začátku 90. let. V té době už byl však další rozvoj téměř nemožný, a proto byla vydána poslední verze, jejímž úkolem bylo alespoň částečné pročištění kódu.

Projekt PostgreSQL převzala širší komunita, která dokázala kód odladit, odstranit většinu kritických chyb a vyřešit problémy s pamětí. Jelikož se od samého začátku jednalo o open-source systém, mnoho různých vývojářů dostalo možnost samostatně pracovat na své vlastní kopii. Díky tomu vznikaly projekty, které vždy implementovaly nejnovější technologie a poznatky. V současné době pracuje na tomto systému velké množství vývojářů, každý rok vychází nová verze a v průběhu roku dochází k rychlým opravám zjištěných chyb. Komunitní vývoj je tedy opravdu aktivní. PostgreSQL se stal velmi oblíbeným systémem a kromě samotných vývojářů jej začalo finančně podporovat i mnoho soukromých firem. [5]

2.4 InterBase a Firebird

Na konci minulého století vyvíjela společnost Borland vlastní SŘBD pod názvem InterBase. Ve společnosti se v té době často měnil vrcholový management, tím pádem i priority a strategie jednotlivých projektů, a proto nebyl nikdy zcela jasný směr, kterým se bude InterBase ubírat. Ačkoli byl tento systém pro firmu poměrně výnosný, Borland se rozhodl projektu omezit finance a redukovat stav vývojářů, kteří na něm pracovali. Důležitý okamžik nastal v červenci 2000. Borland zveřejnil veškeré zdrojové kódy InterBase verze 6.0 pod open-source licencí. Nejspíše tím chtěl zapojit a rozšířit řady vývojářů o veřejnou komunitu. K tomu ale nikdy nedošlo a později byl vývoj opět uzavřen.

Hned týden po zveřejnění zdrojových kódů byl ale na SourceForge založen fork InterBase s názvem Firebird. Kolem projektu se téměř okamžitě vytvořila velmi silná komunita a postupně na něm začala pracovat i spousta vývojářů InterBase, včetně Ann Harrisonové, která stála u zrodu systému ještě předtím, než jej odkoupil Borland. Firebird je i v současnosti stále open-source systémem a jeho vývoj je neustále zajišťován velmi oddanou komunitou. [6]

2.5 SQLite

Prvotní vývoj SQLite začal mnohem později, než u všech ostatních systémů, které jsou zmíněny dříve. Tvůrcem systému je Dwayne Richard Hipp, který na něm začal pracovat v roce 2000. SQLite byl vyvíjen v rámci zakázky pro námořnictvo Spojených států amerických. Měl tedy původně sloužit výhradně k vojenským účelům (konkrétně pro řízení naváděných střel). Z tohoto důvodu bylo potřebné implementovat některé vcelku neobvyklé vlastnosti (např. aby nebylo nutné zasahování administrátora databáze). Hipp použil při vývoji systému některé prvky z PostgreSQL.

Nejedná se o SŘBD, který komunikuje na bázi klient-server. SQLite je pouze knihovna, která se nalinkuje k příslušnému programu. Databázová data se ukládají do samostatného souboru. [7]

2.6 CUBRID

Vývoj tohoto relačního SŘBD začal v roce 2006 pod záštitou společnosti NHN Corporation. NHN Corporation je korejská firma, jenž obsluhuje například vyhledávací portál *Naver.com*, který je nejvýznamnějším vyhledávačem v Koreji. Prvotní vývoj systému trval 2 roky, takže první verze byla uvolněna až v roce 2008. Na projektu CUBRID se podílí okolo 40 hlavních vývojářů, kteří jsou především z Koreji, Číny nebo Rumunska. Původně byly vydávány 2 verze systému ročně, které byly vždy prokládány několika menšími aktualizacemi. Poslední verze však byla vydána 5. června 2014, z čehož vyplývá, že se vývoj velmi výrazně zpomalil a jeho budoucnost se jeví nejistá. [8]

2.7 Ostatní SŘBD

Existuje ještě mnoho dalších volně šiřitelných SŘBD, avšak jejich vlastnosti plně nevyhovují stanoveným podmínkám. Některé systémy (např. Apache Derby, HSQLDB, H2) jsou psány v jazyce Java a i práce s nimi probíhá především skrze tento jazyk. Z toho důvodu je zde tedy zavedena i podpora objektů a na tyto SŘBD se tak nedá nahlížet jako na čistě relační systémy. [9, 10, 11]

Další skupinou SŘBD, které nesplňují některou ze zmíněných podmínek, jsou systémy, jejichž vývoj již není zcela aktivní a poslední vydaná verze je i několik let stará. Typickými zástupci této skupiny jsou např. systémy Drizzle (poslední verze vyšla v roce 2012), LucidDB (2010) nebo SmallSQL (2011). [12, 13, 14]

3 NÁVRH KRITÉRIÍ

Z perspektivy databázového programátora je prvotní pohled zaměřen na samotnou práci s jednotlivými SŘBD. Primární důraz proto bude kladen především na druhy a vlastnosti uživatelských rozhraní, které daný systém poskytuje. Všechny systémy nabízejí možnost přístupu k databázi přes konzolové klienty. U některých je v základním instalačním balíčku zahrnut i nástroj pro práci s databází skrze grafické uživatelské rozhraní. Pro většinu systémů bylo vytvořeno také webové rozhraní, takže databázi lze upravovat i s použitím webového prohlížeče. Každý SŘBD využívá odlišný dialekt jazyka SQL. I tato oblast tedy bude podrobena zkoumání.

Jednotliví konzoloví klienti budou porovnáváni podle podpory českého jazyka, možnosti ověřování uživatele (zadávání jména a hesla) a podle existence klientských aplikací pro import a export databáze. Dále bude zjišťováno, zda je možný export do binárního souboru či formátu SQL.

Grafická rozhraní lze porovnávat podle mnoha různých faktorů. Mezi ty nejzákladnější patří např. seznam platforem (operačních systémů), na které lze daného klienta nainstalovat. Některé aplikace jsou také součástí instalačního balíčku daného SŘBD. Dále může být uživatelem vyžadována podpora českého jazyka nebo možnost uchování spojení pro více různých databází současně. Důležitým kritériem je také nutnost znalosti jazyka SQL pro práci s vybraným grafickým klientem. S některými nástroji totiž lze pracovat s nulovou znalostí tohoto jazyka, pro některé je alespoň částečná znalost nezbytná (např. v oblasti sestavování dotazů) a vyskytují se i nástroje, které vyžadují znalost SQL i pro spoustu základních operací. S tím souvisí i možnost sestavování dotazů grafickou formou. Také je možné porovnávat klienty s ohledem na schopnost generování skriptu, který reprezentuje vybraný objekt, podle kvality zvýrazňování SQL syntaxe nebo podle možnosti importu a exportu databází.

Kritéria pro porovnávání webových rozhraní mohou být v mnoha případech totožná s kritérii pro grafické klienty. V první řadě bude zjišťována podpora českého jazyka. Dále bude testována možnost uchování několika spojení (s různými databázemi) současně. Velmi důležitým faktorem v porovnávání bude také nutná úroveň znalosti jazyka SQL pro práci s vybranými klienty. Ta může být snížena díky existenci nástroje pro grafické sestavování dotazů. Některá webová rozhraní mohou obsahovat funkce pro grafické vytvoření modelu databáze a jejích relací (tzv. data modeler). Zásadní může být také existence formuláře, skrze který lze komunikovat s klientskými aplikacemi pro import a export databáze.

V rámci porovnávání dialektu SQL bude především zjišťováno, jakými klíčovými slovy označují systémy své základní datové typy. S tím souvisí i druh používaných typů. Dále bude testováno, které systémy poskytují možnost automatické inkrementace, případně kte-

rým klíčovým slovem toho lze dosáhnout. Také budou zaznamenány znaky, kterými lze ohraničit identifikátory a bude testována nutnost dodržování velikosti písmen mezi těmito znaky. Některé systémy umožňují v dotazu SELECT vynechat klauzuli FROM nebo podporují možnost vícenásobného vkládání dat v rámci jediného příkazu INSERT.

Byly vybrány čtyři relační SŘBD, které budou porovnávány podle navržených kritérií. Zvolenými systémy jsou PostgreSQL, MariaDB, Firebird a SQLite. Systém MySQL byl z porovnávání vynechán záměrně, jelikož podobnost se SŘBD MariaDB je vskutku veliká (z pohledu databázového programátora) a budoucí vývoj MySQL bude zřejmě mnohem méně aktivní.

4 KONZOLOVÝ KLIENT

Konzolový klient je mnohdy prvotním rozhraním, které databázový programátor pro práci s databází využije. Existuje mnoho různých druhů klientských aplikací. Nejvyužívanější a nejdůležitější je však základní terminálový klient daného SŘBD a klienti pro zálohu a následnou obnovu databáze.

V této kapitole se vyskytují především příkazy pro připojení do databáze přes příkazovou řádku. V takovém případě se může prompt jednotlivých operačních systémů lišit. Z tohoto důvodu byla zavedena obecná značka `cmd>`, která znázorňuje prompt libovolného operačního systému. Dále jsou zavedeny značky `[connection-option]` a `[option]`, které obsahují množinu příslušných přepínačů. První skupina obsahuje přepínače potřebné pro připojení do databáze, ve druhé kategorii jsou přepínače upravující funkčnost konkrétního klienta.

4.1 PostgreSQL

PostgreSQL obsahuje skutečně rozsáhlou řadu klientských aplikací. Svého základního interaktivního terminálového klienta pojmenovali vývojáři intuitivně *psql*. Mezi další velmi podstatné konzolové klienty patří *pg_dump* (export databáze do souboru) a *pg_restore* (obnova databáze ze souboru). Dále jsou k dispozici například nástroje pro vytváření a odstraňování databází (*createdb*, *dropdb*) a uživatelů (*createuser*, *dropuser*). K výpisu informací o aktuálně používané verzi PostgreSQL slouží *pg_config*. [15]

4.1.1 PSQL

Jak již bylo zmíněno, hlavní konzolový klient systému PostgreSQL se jmenuje *psql*. První věcí, které si český uživatel využívající *psql* může všimnout, je podpora českého jazyka. Klient se sice tváří, že by měl umět vypisovat např. nápovědu nebo chybová hlášení v češtině, ovšem podpora české diakritiky nefunguje zcela bezproblémově. Ve většině případů je třeba před každým spuštěním klienta vždy zadat následující dvojici příkazů:

```
cmd> SET PGCLIENTENCODING=WIN1250
cmd> chcp 1250
```

Po splnění tohoto kroku by již měla být diakritika vypisována bezchybně. Zkušenějším uživatelům je však doporučeno používat výchozí jazyk (angličtinu), tím pádem i výchozí kódování.

Pro připojení do databáze přes *psql* je potřeba znát několik klíčových parametrů. Mezi ty základní patří například `-U` (nebo `--username`) pro připojení jako konkrétní uživatel, `-h` (`--host`) určuje hostitelský server, `-p` (`--port`) označuje číslo portu, skrze který bude databázový server komunikovat. Parametr `-d` připojí uživatele ke konkrétní databázi. Lze použít i parametr `-W` (`--password`) pro zadání hesla, ale tato možnost se z bezpečnostních důvodů příliš nedoporučuje. Je mnohem bezpečnější zadávat heslo až tehdy, vyzve-li k tomu uživatele zpráva v příkazové řádce (heslo je poté skryto a není tedy možné jej tímto jednoduchým způsobem odpozorovat). [16]

Jednotlivé přepínače lze zadávat dvěma způsoby. Buďto pomocí jediného znaku, kterému předchází pomlčka a následuje mezera, nebo za použití slovní verze parametru, jemuž ale musí předcházet pomlčky dvě a za přepínačem musí následovat rovnítko. Druhá možnost byla zavedena z důvodu lepší přehlednosti a zapamatovatelnosti jednotlivých přepínačů. Připojení do databáze přes *psql* pomocí znakových přepínačů lze provést následovně:

```
cmd> psql -h host_name -p 5432 -U user_name -d db_name
```

Druhý způsob využívá přepínačů slovních:

```
cmd> psql --hostname=host_name --port=5432
     --username=user_name --dbname=db_name
```

4.1.2 PG_DUMP

Tento klient slouží k exportu databáze a výstup z této klientské aplikace tedy umožňuje její zálohu. Pomocí *pg_dump* je možné vytvořit zálohu databáze i v případě, že je databáze právě používána. Tento nástroj může zálohovat nejen jednotlivé databáze, ale i konkrétní objekty v nich. Pro kompletní export všech dostupných databází v rámci celého clusteru lze použít klienta *pg_dumpall*. Obecná syntaxe pro použití klienta *pg_dump* je následovná:

```
cmd> pg_dump [connection-option] [option] [dbname]
```

Export pomocí *pg_dump* může být proveden dvěma způsoby. První možností je export v podobě skriptů obsahujících SQL příkazy. Tato volba je sice nastavena jako implicitní, je však možné ji zvolit také pomocí přepínače `-F` (nebo `--format`) a následným nastavením hodnoty `p` (nebo také `plain`). Jedná se tedy o textové soubory, podle nichž je možné v budoucnu rekonstruovat databázi přesně do stavu, v jakém se nacházela v době vytvoření zálohy. Následná obnova databáze z takto vytvořené zálohy se provádí přes klienta *psql*.

Druhou možností je export dat v komprimovaném archivu. Tato možnost může být zvolena hned několika způsoby. Přepínači `-F` lze nastavit buď hodnotu `c` (`custom`),

`d` (`directory`) nebo `t` (`tar`). Custom je nejobvyklejší způsob exportu dat do archivu. Společně s formátem directory podporuje kompresi dat. Directory formát vytvoří pro každou tabulku samostatný soubor a dále přidá soubor popisující všechny exportované objekty. Poslední možností je formát tar, který je kompatibilní s formátem directory (extrahováním archivu lze dosáhnout validního directory formátu), ale nepodporuje kompresi dat. Všechny tři typy exportu databáze do archivu je nutné obnovovat pomocí klientské aplikace *pg_restore*, který poskytuje mnoho možností, jak s archivy pracovat. [17, 18]

4.1.3 PG_RESTORE

Jak již bylo zmíněno, *pg_restore* je nástroj sloužící k obnově dat ze zálohy vytvořené pomocí *pg_dump*. Tato obnova však může být vytvořena pouze z dat ve formátu archiv. Pro zálohy typu plain (SQL příkazy) není možné tento nástroj použít. Obecně lze obnovu databáze vykonat tímto způsobem:

```
cmd> pg_restore [connection-option] [option] [filename]
```

Možnost využití tohoto klienta je výhodná především proto, že je možné jednoduše vybrat, která konkrétní data mají být ze zálohy obnovena. Není tedy nutné kompletně obnovovat data z celé zálohy, ale lze vybrat pouze některé její části. Tyto části jsou vzápětí aplikovány do stávající databáze.

Pg_restore operuje ve dvou módech v závislosti na tom, zda bylo při volání tohoto nástroje zadáno i jméno databáze, která má být aktualizována. Pokud byl tento parametr zadán, *pg_restore* se k databázi připojí a změny se automaticky aplikují. V případě, že tento parametr není dostupný, je vytvořen skript obsahující SQL příkazy, které jsou nutné k vytvoření databáze shodné se zálohou v archivu. Tento skript je následně vypsán do souboru nebo na obrazovku a je identický se skriptem, který by byl vytvořen za pomoci klientské aplikace *pg_dump* s parametrem plain. [19]

4.2 MariaDB

Základním konzolovým klientem SŘBD MariaDB je *mysql*. Pro zálohu dat a pro jejich následnou obnovu je možné využít klienty *mysqldump* a *mysqlimport*. Dalším klientem je například *mysqlshow*, který zobrazí jména všech databází, tabulek a sloupců (tedy kompletní databázovou strukturu), ke kterým má daný uživatel nastavena dostatečná přístupová práva. Často lze také využít nástroj *mysqladmin*, který monitoruje, co dělají ostatní klienti, umožňuje vytvářet či mazat databáze nebo zastavit běh serveru. [20]

4.2.1 MYSQL

Konzolový klient využívaný systémem MariaDB nese stále ještě označení *mysql*. Klient podporuje vyobrazení výsledků požadovaných dotazů buď pomocí ASCII tabulkového formátu, nebo jsou data prezentována za použití tabulátorového oddělovače. Způsob zobrazení výstupů závisí na tom, zda je používán interaktivní (tzv. ASCII-table formát) či neinteraktivní mód (tab-separated formát). Výstupový formát je v případě potřeby možné změnit. [21]

Konzolový klient se spouští příkazem `mysql`. Jako základní přepínače mohou být použity `--host` (`-h`), `--port` (`-P`), `--user` (`-u`), `--password` (`-p`) a `--database` (`-D`). Poslední zmíněný přepínač není nutné zadávat, jelikož si *mysql* sám dohledá hodnotu, která není přidělena žádnému přepínači a přistupuje k ní jako ke jménu požadované databáze. Výsledný příkaz tedy může vypadat například následovně: [22]

```
cmd> mysql -h host_name -P 3306 -u user_name
     -p password db_name
```

4.2.2 MYSQLDUMP

Tato klientská aplikace slouží k exportu databáze, čímž je vytvořena její záloha. Tohoto klienta je vhodné využít například v případě, že je potřeba přenést databázi na jiný databázový server. Vytvořená záloha může být ve formě textového souboru obsahujícího SQL příkazy nutné k vytvoření identické databáze. Záloha vytvořená pomocí *mysqldump* může být i ve formátech CSV nebo XML (v případě exportu dat, nikoli struktury databáze).

Mysqldump lze použít jedním ze tří hlavních způsobů, které jsou voleny podle toho, jaká data chceme exportovat. Lze vytvořit zálohu v rozsahu několika tabulek, výčtu několika databází anebo je možné exportovat kompletní strukturu všech databází na daném databázovém serveru. Popsané akce je možné vykonat pomocí následujících příkazů:

```
cmd> mysqldump [options] db_name [table_name ...]
cmd> mysqldump [options] --databases db_name ...
cmd> mysqldump [options] --all-databases
```

Součástí zálohy jsou veškeré tabulky z exportované databáze nebo kompletní databázová struktura (v případě zálohování většího databázového svazku). Implicitně však do exportu nejsou zahrnuty uložené procedury, pohledy a události. Pokud je požadována i záloha některého z těchto objektů, je nutné předat *mysqldump* ještě příslušný parametr (například `--routines`, `--events`). [23]

4.2.3 MYSQLIMPORT

Mysqlimport slouží k načtení tabulek z textového souboru, přičemž je podporováno několik různých formátů. Textový soubor musí mít přesně totožný název jako tabulka (přípona souboru není brána v potaz), ve které mají být požadované změny provedeny. Je samozřejmě možné importovat hned několik textových souborů najednou. Obecná syntaxe pro import je následovná: [24]

```
cmd> mysqlimport [options] db_name textfile1
     [textfile2 ...]
```

4.3 Firebird

Základní klientská aplikace systému Firebird byla pojmenována *isql* (Interactive SQL). Drobnou zvláštností tohoto systému je skutečnost, že nepoužívá samostatné klienty pro vytvoření zálohy a pro její následnou obnovu. O obě tyto funkce se stará stejný klient v závislosti na volbě přepínače. Naproti tomu má Firebird takové klienty hned dva. První aplikace byla pojmenována *gbak*, druhá nese označení *nbackup*. [25]

4.3.1 ISQL

Konzolový klient *isql* podporuje víceřádkové zadávání dotazů. To znamená, že jakýkoli příkaz může být psán na libovolný počet řádků bez ohledu na to, kolikrát byla stisknuta klávesa ENTER. Příkaz končí tehdy, je-li přečten středník. Až poté je odeslán ke zpracování serveru, který k němu již přistupuje bez ohledu na to, na kolika řádcích byl zadán. Do té doby, než je zadán středník, je na příkazové řádce znázorněn prompt `CON>`. Víceřádkový příkaz pro připojení k databázi může vypadat například takto:

```
SQL> CONNECT "database_path"
CON> USER "user_name"
CON> PASSWORD "password"
CON> ;
SQL>
```

Tento víceřádkový způsob zadávání příkazů je sice možný v mnoha ostatních SŘBD, nicméně komunita okolo Firebirdu tuto možnost využívá velmi často. [26, 27]

Nástroj *isql* lze použít ve třech základních módech. Nejčastěji je z příkazové řádky vyvolána tzv. interaktivní relace, která trvá až do její ukončení příkazem `QUIT` nebo `EXIT`. Druhou možností je spuštění *isql* z příkazové řádky bez zahájení interaktivní relace. V takovém případě je vykonán zadaný příkaz a následně je řízení automaticky předáno zpět

operačnímu systému. Dále je možné využít také shellový skript nebo dávkový soubor, které provedou požadované operace. [28]

4.3.2 GBAK a NBACKUP

Zpočátku existoval pouze nástroj *gbak*. *Nbackup* se v instalačních balíčcích Firebirdu začal objevovat až od verze 2.0 a měl původní zálohovací nástroj nahradit. Obsahoval totiž možnosti, které *gbak* neposkytoval. Postupem času se však ukázalo, že každý nástroj má své silné a slabé stránky. Oba tak zřejmě budou koexistovat i nadále. [29]

Mezi oběma nástroji jsou 2 zásadní rozdíly. *Gbak* kontroluje data, ale tato kontrola se samozřejmě projeví na rychlosti zálohy. Naproti tomu *nbackup* se kontrolou dat vůbec nezabývá, za to je však výrazně rychlejší než jeho předchůdce. *Nbackup* dále podporuje inkrementální zálohy. Ani jeden z klientů neumožňuje export databáze ve formátu SQL. Výstupní soubor po použití klienta *gbak* má obvykle příponu .fbk, při provedení zálohy pomocí *nbackup* je koncovka exportu .nbk. [32, 33]

Pokud chce uživatel využít služeb nástroje *gbak*, je nutné znát tři hlavní přepínače, od nichž se dále odvíjí, jaká akce bude provedena. Pro vytvoření zálohy databáze existuje přepínač `-b` (backup database), který ale není nutné zadávat. To znamená, že klient automaticky předpokládá, že se jedná o vytvoření zálohy, pokud není uvedeno jinak. Pro obnovení databáze ze zálohy se používá přepínač `-c` (create database), který vytvoří zcela novou databázi, nebo přepínač `-r o` (recreate database and overwrite), který přepíše stávající databázi. Základní provedení zálohy databáze pomocí klienta *gbak* lze provést následujícím příkazem: [30]

```
cmd> gbak -b <source> <destination>
```

Při použití klienta *nbackup* může být využito pět základních přepínačů. Přepínač `B` (backup) pro vytvoření zálohy, `R` (restore) pro obnovení databáze ze souboru, `L` (lock) a `N` (unlock) pro uzamčení a následné odemčení databáze. Přepínač `F` odemkne databázi po úspěšně dokončené obnově. Obnova databáze ze souboru pomocí *nbackup* lze vyvolat následovně: [31]

```
cmd> nbackup -R <database> [<backupfile>]
```

4.4 SQLite

Systém SQLite používá konzolového klienta, jehož aktuální verze je nazvána *sqlite3*. Díky tomu, že SQLite nekomunikuje na bázi klient-server, není nutné, aby tento SŘBD obsahoval velké množství obslužných knihoven a souborů. V podstatě lze používat pouze jediný

spustitelný soubor. Tím souborem je právě konzolový klient *sqlite3* (případně starší verze). Součástí základního balíčku SQLite tedy není žádná klientská aplikace pro zálohování a následnou práci s vytvořenými zálohami. Veškeré operace tohoto druhu jsou prováděny v rámci *sqlite3*.

4.4.1 SQLITE3

Díky výše zmíněné skutečnosti není tedy ani nutná žádná forma autorizace při přístupu do databáze. Databáze se totiž nenachází na serveru, nýbrž je uchována v samostatném externím souboru. Odpadá zde tedy nutnost existence jakýchkoli přepínačů a jediný parametr, který je možné nástroji *sqlite3* předat, je název databáze. K přístupu k požadované databázi stačí tedy napsat následující příkaz:

```
cmd> sqlite3 database_name
```

V případě, že uživatel požaduje přepínání mezi více databázemi (nebo pokud nebyl klient spuštěn přes příkazovou řádku a tím pádem pracuje v režimu „dočasné databáze“), je potřeba využít příkazu `.open`:

```
sqlite> .open database_name
```

V případě vytváření zálohy databáze lze postupovat dvěma různými způsoby. Pomocí příkazu `.backup` lze vytvořit zálohu v podobě binárního souboru, po využití příkazu `.dump` je dosaženo exportu databáze v SQL formátu. Obnovu databáze lze provést využitím volby `.restore`. [34]

4.5 Výsledky porovnávání konzolových klientů

Při porovnávání konzolových aplikací došlo k zjištění mnoha skutečností, jejichž shrnutí se nachází v tab. 4.1. SQLite viditelně zaostává za ostatními systémy kvůli snaze o zachování maximální míry jednoduchosti. Také je to proto, že obsahuje jedinou konzolovou aplikaci.

	PostgreSQL	**MariaDB**	**Firebird**	**SQLite**
Čeština	ANO	NE	NE	NE
Autentizace uživatele	ANO	ANO	ANO	NE
Klient pro import	ANO	ANO	ANO	NE
Klient pro export	ANO	ANO	ANO	NE
Export do bin. souboru	ANO	NE	ANO	ANO
Export do SQL	ANO	ANO	NE	ANO

Tab. 4.1: Porovnání konzolových klientů

5 GRAFICKÉ ROZHRANÍ

Konzolový klient neposkytuje při práci s databází dostatečný komfort, a proto existují nástroje, které tuto činnost výrazně usnadňují. Jedná se o aplikace, díky kterým lze s databází manipulovat pomocí grafického rozhraní. Program obsahuje algoritmy, které zaznamenají činnosti vykonané uživatelem a následně je převedou do jazyka SQL. Výsledný skript je poté promítnut do databáze. Nutnost znalosti jazyka SQL je tedy v některých oblastech zcela eliminována (v závislosti na konkrétním klientovi). Pro plné využití potenciálu daného SŘBD je však jistá znalost SQL nezbytností.

Grafické rozhraní usnadňuje především proces definice struktury databáze, vkládání a editaci dat a sestavování dotazů. Po provedení dotazu jsou data zobrazena přehledněji, což umožňuje jejich snazší interpretaci.

5.1 PgAdmin

Aktuální vydaná verze grafického klienta systému PostgreSQL byla nazvána *pgAdmin*. Pro účely testování byla použita verze 1.20.0. Tento klient je dostupný pro celou řadu operačních systémů včetně Windows, Mac OS X, Linuxu, FreeBSD nebo Solarisu. Je tedy možné tvrdit, že *pgAdmin* je skutečně multiplatformní, což je jedna z jeho velikých výhod. Na některých platformách je dokonce importován do instalačního balíčku tohoto SŘBD, čímž i samotní vývojáři nabádají k jeho využívání. S každou novou vydanou verzí PostgreSQL je vzápětí aktualizován i tento klient, takže všechny změny se ve velmi krátkém časovém intervalu projeví i v grafickém rozhraní. [35, 36]

Celý program je kompletně lokalizován do mnoha různých jazyků (včetně češtiny). Po spuštění aplikace je možné vypozorovat, že celé hlavní okno programu je rozděleno na 3 základní sekce (viz příloha A.1). V levé části se nachází seznam objektů, vyobrazen do přehledné stromové struktury. Zbytek okna je rozdělen vertikálně na další 2 oblasti. Horní část obsahuje přehled vlastností vybraného objektu a další informace, spodní část je tzv. SQL panel.

Pro připojení k databázi je potřeba znát IP adresu nebo DNS jméno serveru, na kterém je databáze umístěna, a port, skrze který bude následně komunikace probíhat (výchozí port je 5432). Také je nutné vlastní pojmenování vytvořeného spojení pro jeho následnou identifikaci. Položka *Služba* představuje jméno služby nastavené v souboru pg_service.conf a ve většině případů ji není třeba vyplňovat. *Servisní DB* představuje jméno databáze, do které jsou ukládány objekty vytvořené přihlášeným uživatelem, a i zde stačí zpravidla ponechat výchozí hodnotu. Dále už je potřeba vyplnit pouze uživatelské jméno a heslo, čímž je dosaženo ověření uživatele.

Po úspěšném přihlášení je možné se ve stromu objektů v levé části připojit k požadované databázi na konkrétním serveru, spravovat tabulkové prostory (prostor, kam se ukládají data), přihlašovací role a skupinové role. Po provedení výběru databáze jsou dále zpřístupněny další možnosti, související s aktuální databází. Jsou zde například tzv. katalogy, které obsahují informace o databázi a o jejích objektech. Dále jsou zde rozšíření, díky nimž je možné přidat či nastavit jazyky pro vytváření databázových objektů. Nejpodstatnější položkou celé této stromové struktury pro jsou však schémata. Schémata totiž reprezentují prostor, kam uživatel vytváří databázové objekty. Jednotlivých typů objektů je celá řada, nejdůležitější jsou však tabulky, u kterých lze dále zobrazovat jednotlivé sloupce a jejich další vlastnosti (např. primární a cizí klíče). Ostatními typy objektů jsou například pohledy nebo domény (nastavení možných hodnot intervalem, výčtem atd.). [37]

Na každou položku ve stromové struktuře lze kliknout pravým tlačítkem, čímž je možné zobrazit kontextové menu. Tato nabídka je specifická pro každý typ objektu a obsahuje tedy pouze ty možnosti, které jsou pro zvolený objekt vhodné. Díky tomu není téměř vůbec nutné využívat hlavní (vodorovné) menu v horní části okna programu a většinu akcí lze provádět právě přes strom objektů (vytváření nových objektů, editace nebo odstranění objektů atd.).

Pro tabulky je kontextové menu velmi obsáhlé a nabízí mnoho funkcí. Skrze jednu z jeho položek *Nový objekt* lze např. přidat sloupec do vybrané tabulky nebo vytvořit nový cizí klíč. Kontextové menu může sloužit také pro zobrazení dat, jimiž je tabulka naplněna. V následně zobrazeném okně je možné jednoduše přidávat do tabulky nové řádky. Zajímavým nástrojem je také možnost vytvoření vlastního skriptu (create, select, insert, update, delete), jehož podstatnou část předem vyplní program samotný a uživatel poté pouze zadá nezbytné hodnoty, které program nezná (např. kritéria výběru nebo hodnoty pro daný řádek). Nechybí ani položky pro zálohu a následnou obnovu dat nebo pro kompletní odstranění veškerých dat z tabulky a zachování tak pouze její struktury.

Panel vlastností obsahuje podrobný přehled informací o objektu vybraném v panelu objektů. Také zde může docházet k modifikacím některých informací. Tato akce se provádí dvojklikem na vybranou položku. Dále tento panel obsahuje záložku se statistikami a záložky *Závislosti* (objekty, na kterých daná položka závisí) a *Závisející* (objekty, které závisejí na vybrané položce).

Poslední částí okna programu je panel SQL a i tato sekce obsahuje velmi užitečné informace. Jedná se především o to, jak by vybraný objekt vypadal v jazyce SQL, respektive jak by bylo možné jej v tomto jazyce vytvořit.

Vývojáři *pgAdmin* se skutečně snažili, aby bylo možné program ovládat i s prakticky nulovou znalostí jazyka SQL. Proto byla přidána i podpora grafického sestavování dotazů, která bývá v programech obdobného typu často opomíjena či záměrně přehlížena.

V případě potřeby využití této funkce stačí pouze v dialogovém okně pro zadávání dotazu přepnout na kartu *Grafické sestavení dotazu*. Poté lze jednoduše vybrat tabulky, které bude třeba v dotazu použít, propojit je skrze příslušné sloupce a zadat názvy sloupců, které uživatel požaduje ve výsledku zobrazit. Volba podmínek lze také definovat velmi intuitivní formou bez znalosti SQL a podobným způsobem lze také přednastavit způsob řazení výsledných dat. Dotaz je poté proveden stisknutím tlačítka *Provést dotaz* (zelený trojúhelník). Výsledek je možné vypsat na obrazovku i do souboru.

Grafický klient *pgAdmin* je velmi komplexní nástroj, který umožňuje práci s databází pro všechny typy uživatelů. Lze v něm pracovat kompletně skrze jazyk SQL (v hlavním menu dokonce obsahuje i ikonu pro rychlé spuštění konzolového klienta *psql*), avšak v žádném případě není znalost SQL nutnou podmínkou. Veškerá funkčnost je totiž obsažena i v grafických prvcích a s databází je tedy možné manipulovat i bez znalosti jazyka. Nejlepším způsobem práce s klientem je však kombinace obou přístupů, která umožňuje značně urychlit některé úkony, oproti užívání pouze jedné z metod.

Díky tomu, že tento klient obsahuje skutečně obsáhlé množství funkcí a nástrojů, se stává velmi nepřehledným a pro nového uživatele může být složité se v něm zorientovat. Program není příliš intuitivní a zpočátku je téměř nezbytné dohledávání informací v manuálech.

5.2 HeidiSQL

HeidiSQL (verze 9.1.0) je grafický klient systému MariaDB a je v mnoha ohledech podobný klientu *pgAdmin* (viz příloha A.2). Je také součástí instalačního balíčku systému, je kompletně lokalizován do českého jazyka a především rozložení hlavních prvků v okně programu působí téměř totožně. Také je zde zřejmá inspirace klientem *MySQL Workbench* (grafický klient systému MySQL). V levé sekci aplikace se nachází stromová struktura znázorňující seznam databází a typy objektů v nich. Napravo od ní je umístěna hlavní část okna, skrze které je možné pracovat s vybranou databází. Na spodní straně aplikace je pak SQL sekce, která převádí veškerou manipulaci s databází do jazyka SQL.

Pro připojení k databázi je opět potřeba znát některé základní údaje. Je nutné vyplnit vlastní identifikátor serveru, IP adresu nebo DNS jméno serveru, komunikační port (výchozí je 3306), uživatelské jméno a heslo. Dále lze z roletky vybrat druh sítě, přičemž je zde často využívána varianta *MySQL (TCP/IP)*, která je nastavena jako výchozí. Ostatní položky mohou zůstat nevyplněné.

Levá část programu zde však nemá ani zdaleka tak stěžejní funkci, jako tomu bylo u *pgAdminu*. Nachází se zde totiž podstatně méně položek, které se dají rozdělit do dvou skupin. Prvním typem jsou databáze, které obsahují informace (metadata) o všech ostat-

ních databázích na daném serveru. Jedná se tedy o informační databáze, jejichž obsahem nejsou přímo základní tabulky, nýbrž pohledy (nejsou spojeny s žádným souborem a obsahují pouze předpis, jakým způsobem mají být data z tabulek získávána). Mohou obsahovat například údaje o jménech databází a tabulek, datových typech jednotlivých sloupců nebo o uživatelských oprávněních. Druhým typem položek jsou pak samotné databáze vytvořené uživatelem.

Při kliknutí pravým tlačítkem na některou z položek ve stromové struktuře je zobrazeno kontextové menu. To sice skýtá širokou škálu možností a nástrojů, avšak na rozdíl od *pgAdminu* jej není tolik nutné využívat. Všechny tyto možnosti lze totiž mnohem jednodušeji zobrazit pomocí některé ze záložek v hlavní sekci okna programu nebo přes jeho vlastní kontextovou nabídku.

Na vrchní hraně hlavní části programu se nachází skupina karet znázorňujících položky, ke kterým je vybraný objekt vázán. Například pro libovolnou tabulku, vybranou v seznamu objektů v levé části, obsahují karty informace o serveru a databázi, v nichž se tabulka nachází, a o tabulce samotné. Dále jsou zde karty zobrazující data v dané tabulce a záložka pro vytvoření dotazu. Vedle poslední zmíněné záložky je umístěno ještě tlačítko, které umožňuje přidat více karet pro tvorbu dotazů. Po výběru některých záložek (hostitel, tabulka) je dostupná i druhá úroveň karet, které obsahují informace o daném objektu. Informace o tabulce lze skrze tyto karty jednoduše editovat. Každá změna je zapsána do jazyka SQL v záložce *Kód ALTER* a je aplikována po stisku tlačítka pro uložení. Také je možné zobrazit SQL kód pro tvorbu vybrané tabulky.

V SQL okně ve spodní části obrazovky jsou zobrazovány pouze SQL příkazy, které znázorňují akce prováděné uživatelem. Ten má tedy možnost sledovat v reálném čase, jak je veškerý jeho pohyb v rámci klienta interpretován jazykem SQL. Vzniklé posloupnosti příkazů lze poté exportovat do souboru či zapnout automatické logování do souboru.

Při používání klienta *HeidiSQL* je po uživateli požadována alespoň základní znalost jazyka SQL. Především v oblasti zadávání dotazů je tato znalost doslova nezbytností. Klient sice nabízí nápovědu pro sestavení dotazu, avšak ta není příliš přehledná. Obsahuje např. seznam sloupců ve vybrané tabulce nebo funkce a klíčová slova SQL. Sekce s nápovědou však nabízí navíc ještě přehled o historii vykonaných dotazů, což je často velmi užitečná možnost. Je zde zřetelně zobrazeno, kdy byl vybraný dotaz použit naposledy, jak dlouho trvalo jeho provedení a lze jej jednoduše použít znovu.

Prostředí *HeidiSQL* je velmi intuitivní a nový uživatel se v něm dokáže snadno a rychle zorientovat. Tvorbu nových objektů a plnění tabulek daty lze taktéž provést velmi jednoduchým způsobem a tyto možnosti jsou přehledně zakomponované do uživatelského rozhraní. I přesto se však uživatel bez znalosti SQL nejspíše neobejde (díky již zmíněnému sestavování dotazů), což je zřejmou nevýhodou tohoto klienta.

5.3 FlameRobin

Grafický klient pro systém Firebird se jmenuje *FlameRobin*. Testování bylo provedeno na verzi 0.9.2. Tento klient není součástí instalačního balíčku systému Firebird a je tedy nutné jeho stažení zvlášť. Jeho vývojáři se snažili především o to, aby byl program velmi jednoduchý a vykonávané příkazy byly prováděny co nejrychleji. Zaměření na jednoduchost a minimalistické pojetí aplikace jsou zřejmé na první pohled (viz příloha A.3) a klient může zpočátku působit velmi stroze. *FlameRobin* není lokalizován do češtiny a je dostupný pouze v anglickém jazyce. [38]

Skrze *FlameRobin* se lze připojit ke dvěma druhům databází. Může se jednat o databázi umístěnou na serveru nebo o databázi uloženou na disku počítače. V prvním případě je potřeba registrovat nový server. To lze provést pomocí kontextového menu po kliknutí pravým tlačítkem na položku *Home*. Poté již stačí pouze vyplnit název serveru, jeho adresu a číslo portu. Po vykonání registrace serveru je již postup pro připojení k oběma typům databází totožný. Po kliknutí pravým tlačítkem na název serveru lze vytvořit novou databázi, či připojit se k existující. V dialogovém okně je poté nutné vyplnit název a cestu k databázi, typ ověření uživatele, případně jméno a heslo.

Klient je primárně ovládán přes okno se seznamem objektů, které tvoří přehlednou stromovou strukturu. Položky v tomto seznamu jsou obdobné jako u klienta *pgAdmin*. Lze zde nalézt například domény, role, pohledy a samozřejmě také tabulky. Tabulky se dále dělí na 2 typy – tabulky s daty a tzv. systémové tabulky, obsahující informace o celé databázi. Po kliknutí pravým tlačítkem na některý z objektů je zobrazeno kontextové menu, které však většinou nabízí pouze možnost vytvoření nového příslušného objektu.

Prostředí klienta se neskládá z jediného hlavního okna, nýbrž hned z několika menších oken, které na sebe vzájemně navazují. Vždy jsou tedy zobrazena taková okna, která se vztahují k vykonávaným akcím.

Po poklepání na některou z tabulek dojde k jejímu výběru a v nově vzniklém okně jsou zobrazeny veškeré informace o dané tabulce. Zde lze přidávat nové řádky, přeorganizovat jejich pořadí či některé řádky odstranit. Dále je možné spravovat primární klíče, cizí klíče, uživatelská oprávnění nebo lze zobrazit závislosti tabulky. Položka *DDL* (Data Definition Language) poskytuje náhled, jak by daná tabulka (či jakýkoli jiný objekt) mohla být definována pomocí jazyka SQL.

Jedním z hlavních předpokladů pro práci s tímto klientem je znalost jazyka SQL. Nelze zde totiž například vytvořit tabulku čistě pomocí grafického rozhraní. Při vytváření tabulky je klientem pouze vygenerován SQL skript v obecném formátu, který si musí uživatel sám upravit a doplnit podle svých požadavků. Na tomto principu funguje naprostá většina obdobných operací. Zajímavostí je, že pro přidání nového sloupce do již existující tabulky (nebo pro jeho editaci) nabízí *FlameRobin* obvyklé dialogové okno a není tedy třeba psát

tuto aktualizaci v SQL. Program může působit neuspořádaným dojmem i v jiných případech. Pro přidání řádku do tabulky je zcela logicky a intuitivně vybírána z kontextového menu volba *Insert into*, která nabízí opět obecný předpis v SQL, do kterého musí uživatel na vhodná místa doplnit data ke vložení. Tuto akci lze však provádět i mnohem jednodušším způsobem. Stačí pouze vybrat v kontextové nabídce položku *Select from* a v nově otevřeném okně lze poté zaznamenat ikony pro přidání a odebrání řádků mnohem komfortnějším způsobem.

Klient pracuje v režimu potvrzování vykonaných operací a téměř každou vykonanou akci (nebo skupinu akcí) je tedy potřeba vzápětí buď potvrdit (commit) nebo odvolat (rollback).

Klient *FlameRobin* poskytuje velmi strohé uživatelské rozhraní s nepříliš rozsáhlou funkční výbavou. Bez základní znalosti jazyka SQL je práce s aplikací velmi obtížná. Navíc jsou některé funkce programu umístěny poměrně nelogicky. I tak však nabízí oproti konzolovému klientovi mnohem větší komfort a umožňuje snadnější a rychlejší manipulaci s celou databází.

5.4 SQLiteBrowser

Grafických klientů pro SŘBD SQLite je celá řada. Za nejvhodnější z nich byl však zvolen projekt *SQLiteBrowser* (viz příloha A.4), který je volně dostupný, má otevřený zdrojový kód a je vyvíjen speciálně a pouze pro systém SQLite. Program sice obsahuje na výběr z několika jazyků, čeština však bohužel podporována není. Při testování byla použita verze programu 3.8.0. [39]

Jelikož SQLite není založený na principu komunikace klient-server, není nutné pro připojení do databáze zadávat adresu serveru nebo znát přihlašovací údaje. Stačí pouze z disku vybrat příslušný soubor, ve kterém je požadovaná databáze uložena. V případě vytváření nové databáze je potřeba pouze zvolit umístění souboru a na tomto místě je databáze následně vytvořena.

Po spuštění aplikace je zřejmé, že se vývojáři tohoto klienta snažili o zachování maximální míry jednoduchosti. Všechny důležité prvky jsou reprezentovány příslušnými tlačítky a jsou přehledně zakomponovány do uživatelského rozhraní. Každá provedená akce (nebo skupina operací) musí být následně potvrzena či odvolána. *SQLiteBrowser* se tedy v tomto ohledu podobá klientu *FlameRobin*. Strukturu databáze tvoří především tabulky a pohledy.

Importovat lze databáze v SQL formátu. Je také možné importovat pouze vybrané tabulky, které musí být ve formátu CSV. Obdobná pravidla platí i pro export objektů.

Funkční vybavení aplikace není příliš objemné, což přispívá k lepší přehlednosti a k intuitivnější orientaci v rámci klienta. Vytváření, editace a odstraňování tabulek, přidávání a modifikace dat, zapisování vlastních SQL příkazů a další podobné operace jsou vhodně umístěny a uživatel si práci v programu *SQLiteBrowser* velmi rychle osvojí. Celkově klient odpovídá požadavkům, které jsou kladeny na rozhraní pro práci s tímto SŘBD.

5.5 Výsledky porovnávání grafických rozhraní

Každý grafický klient byl podroben testování podle předem stanovených kritérií. Zjištěné skutečnosti jsou k dispozici v tab. 5.1. Z výsledků porovnávání je zřejmé, že funkční výbava aplikací *pgAdmin* a *HeidiSQL* je na velmi vysoké úrovni.

	PostgreSQL	**MariaDB**	**Firebird**	**SQLite**
Platformy	Windows, Linux, Solaris, Mac OS X, FreeBSD	Windows, Linux, Mac OS X[1]	Windows, Linux, Solaris, Mac OS X, FreeBSD	Windows, Linux, Mac OS X, FreeBSD
Součást instal. balíčku	ANO	ANO	NE	NE
Čeština	ANO	ANO	NE	NE
Více databází současně	ANO	ANO	ANO	NE
Nutná úroveň znalosti SQL	Nízká	Střední	Vysoká	Střední
GUI pro sestavení dotazu	ANO	NE	NE	NE
Náhled na objekt v SQL	ANO	ANO	ANO	ANO
Zvýrazňování a našeptávání syntaxe SQL	ANO	ANO	ANO	ANO
Import a export	ANO	ANO	ANO	ANO

Tab. 5.1: Srovnání grafických rozhraní

[1] Pro spuštění programu na OS Linux a Mac OS X je potřeba využít služeb aplikace WINE, která umožňuje zprovoznění programů pro Windows i na jiných operačních systémech. [40]

6 WEBOVÉ ROZHRANÍ

Díky webovým rozhraním jednotlivých SŘBD je možné připojit se k databázi prakticky z kteréhokoli počítače či mobilního zařízení na světě. Stačí k tomu pouze přístup k internetu. To je mnohdy doslova nedocenitelná vlastnost webových rozhraní.

Tyto webové aplikace jsou zpravidla tvořeny pomocí jazyka PHP. Pro správné zprovoznění se předpokládá existence webového serveru. Ten musí být správně nakonfigurován tak, aby umožnil bezproblémovou komunikaci mezi webovým rozhraním a databází. Většina aplikací nabízí tzv. konfigurační skript, s jehož pomocí lze jednoduchým způsobem vyplnit příslušné konfigurační soubory. Jedná se např. o zadání cesty ke konzolovým aplikacím vybraného SŘBD, s nimiž webové rozhraní mnohdy komunikuje a využívá jejich funkce. Webová rozhraní zpravidla nepřizpůsobují svůj obsah stupni oprávnění přihlášeného uživatele.

6.1 PhpPgAdmin

Webový klient systému PostgreSQL se jmenuje *phpPgAdmin*. Testování proběhlo na verzi 5.1. Jeho uživatelské rozhraní se velmi podobá klientu *pgAdmin* (viz příloha B.1), což výrazně usnadňuje orientaci uživateli, který pracuje s oběma klienty.

V levé části se opět nachází seznam objektů uspořádaný do stromové struktury. Tento seznam obsahuje obdobné položky, jako tomu bylo u grafického klienta (např. databáze, tabulky, pohledy nebo domény). Některé položky zde chybí, jiné však byly přidány navíc. Zásadní je především přidání možností pro import a export vybrané tabulky do této sekce. V rámci webového rozhraní totiž není možné využít kontextovou nabídku, a proto musely být takovéto funkce přeorganizovány. Z tohoto důvodu také není seznam objektů tak stěžejním prvkem, jako tomu bylo u grafického klienta.

Pomocí tohoto webového rozhraní lze velmi jednoduše spravovat celou databázi. Po výběru některé z databází je možné např. zobrazit souhrn informací, spravovat schémata a uživatelská oprávnění nebo ji exportovat. Pokud je vybrána některá z tabulek, lze pomocí *phpPgAdmin* vykonávat všechny standardní operace – vytváření nových tabulek a editace a odstraňování stávajících. Dále je samozřejmě možná i práce s daty v tabulkách, především se jedná o procházení, vkládání, editaci a odstraňování dat.

Pro export nabízí webové rozhraní několik předvoleb. Výstupní formát muže zahrnovat strukturu s daty, strukturu samotnou nebo čistě jen data. Export lze provést do různých typů formátů – např. SQL, CSV nebo XML. Pokud je potřeba exportovat strukturu databáze či tabulky, je nutné využít služeb konzolového klienta *pg_dump*, který provede export podle zvolených parametrů.

V pravém horním rohu je možné otevřít SQL okno, do kterého je možné zadat libovolný příkaz v jazyce SQL. Dále je zde možnost zobrazit historii provedených dotazů a je možné některý z nich vykonat znovu. Zajímavá je také funkce pro vyhledávání objektů ve vybrané databázi.

Klient *phpPgAdmin* nabízí v podstatě velmi obdobnou funkční výbavu, která byla k dispozici u grafického klienta *pgAdmin*. Chybí zde sekce, která jednoduchou formou nabízí náhled v SQL na vybraný objekt. Do webového rozhraní také nebyla zakomponována možnost grafického sestavení dotazu. Kvůli tomu je uživatel nucen ovládat jazyk SQL (alespoň v základní míře). Klient je velmi přehledný a orientace v něm je v některých případech jednodušší, než tomu bylo u grafického klienta.

6.2 PhpMyAdmin

Jak již název napovídá, *phpMyAdmin* byl původně vyvíjen pro systém MySQL. Jelikož MariaDB z tohoto systému vychází, byla do ní přenesena většina základních vlastností MySQL. Díky tomu byla také zachována plná kompatibilita s webovým rozhraním *phpMyAdmin*.

Tento webový klient je zřejmě natolik komplexní a dostačující, že vývojáři MariaDB neměli zapotřebí tvorbu vlastního rozhraní. Komunita okolo MySQL si navíc na klienta *phpMyAdmin* velmi silně zvykla, byl velmi hojně využíván a jeho zachování tedy výrazně usnadňuje přechod mezi oběma systémy. Při testování byla využita verze 4.5.1. [41]

Toto rozhraní poskytuje možnost individuálního nastavení. Uživatelským požadavkům lze přizpůsobit např. některé vlastnosti navigačního panelu nebo navigačního stromu. Tato nastavení jsou dostupná skrze jednu z ikon pod logem klienta *phpMyAdmin* (viz příloha B.2).

Struktura jednotlivých položek navigačního stromu je v podstatě totožná jako u grafického klienta *HeidiSQL*. Jediným rozdílem je přidání položek potřebných pro správný chod webového rozhraní.

Webové rozhraní samozřejmě poskytuje nástroje pro snadnou manipulaci s databází a s tabulkami v ní. Součástí je také možnost jednoduchého vkládání a úpravy dat. Dále lze spravovat i uživatelská oprávnění. Nechybí ani nástroje pro import a export, které podporují velmi širokou škálu formátů. Všechny tyto možnosti však patří mezi standardní výbavu většiny webových i grafických klientů. *PhpMyAdmin* však nabízí i mnoho dalších neobvyklých funkcí. Lze zde například pomocí grafického editoru propojovat jednotlivé tabulky, čímž je dosaženo vytvoření relačních spojení.

Při sestavování dotazu je možné postupovat dvěma různými metodami. Je zde zahrnuta podpora pro sestavení dotazu grafickou formou. To znamená, že není nutné příliš ovládat

jazyk SQL, ale stačí pouze zadat některé parametry. Mezi takové parametry se řadí sloupce, které mají být ve výsledku zobrazeny (případně jejich aliasy), řazení a směr řazení vybraných sloupců a podmínka, podle které budou výsledná data vyhledána. Tato metoda však neumožňuje sestavení komplikovanějších dotazů. Druhou možností je pak napsání dotazu kompletně v jazyce SQL. Tímto způsobem není nutné psát pouze dotazy, ale lze takto manipulovat s celou databází (příkazy create, alter, insert, update atd.).

Ve spodní části rozhraní se nachází záložka, která umožňuje zobrazení alternativní konzole přímo ve webovém rozhraní. Zde lze sledovat veškeré akce, které uživatel pomocí *phpMyAdmin* provede. Jedná se o obdobu SQL okna u klienta *HeidiSQL*. Na rozdíl od něj lze však ve webovém rozhraní do této konzole přímo zapisovat požadované SQL příkazy. Pro jejich vykonání je nutné stisknout kombinaci kláves CTRL + ENTER. Obrovskou výhodou této konzole oproti klasickému konzolovému klientovi je to, že *phpMyAdmin* nabízí možnost našeptávání klíčových SQL slov, tabulek, sloupců atd. Díky tomu lze pracovat s databází velmi efektivně a rychle.

Webové rozhraní *phpMyAdmin* je výsledkem dlouholeté práce a vynaložené úsilí je znatelné na první pohled. Není divu, že jej vývojáři systému MariaDB doposud nenahradili vlastním projektem. Součástí jeho výbavy jsou takové funkce, které lze jen stěží nalézt u jiných webových rozhraní. Často nejsou obsaženy ani v grafických klientech jednotlivých systémů. Tyto funkce jsou navíc velmi logicky umístěny. Při základním používání tohoto rozhraní se uživatel obejde i bez znalosti jazyka SQL.

6.3 FirebirdWebAdmin

FirebirdWebAdmin není jediným dostupným webovým rozhraním pro práci se systémem Firebird. Existuje ještě projekt *ibWebAdmin*, který byl dlouhou dobu velmi oblíbeným nástrojem. Jeho vývoj je však již více než 3 roky neaktivní. Poslední verze byla vydána v roce 2013. [42] Projekt *FirebirdWebAdmin* je jím navíc znatelně inspirován a obě rozhraní jsou si tedy velmi podobná. *FirebirdWebAdmin* byl vytvořen s využitím frameworku Bootstrap, díky čemuž působí uživatelsky mnohem přívětivěji než jeho předchůdce. Pro účely testování byla použita verze 3.3.0.

Uživatelské rozhraní tohoto nástroje se na první pohled podstatně liší od rozhraní *phpPgAdmin* nebo *phpMyAdmin* (viz příloha B.3). Není zde zřetelně vyobrazena struktura databáze do navigačního stromu. Místo toho je využíván systém záložek (karet), z nichž každá slouží ke specifickým účelům. Tyto karty jsou umístěny v horní části okna a jsou seřazeny horizontálně vedle sebe.

První karta obsahuje nástroje pro obsluhu databáze. Lze se zde připojit k existující databázi. Dále je možné skrze tuto kartu databáze vytvářet či odstraňovat. Jsou zde také

systémové tabulky a možnost zobrazení SQL skriptu, který představuje strukturu databáze.

Pod další záložkou jsou prostředky pro práci s tabulkami ve vybrané databázi. Tabulky zde lze zobrazovat, vytvářet nebo mazat. Také je možné vybrat některou z tabulek a upravit její vlastnosti.

Skrze další důležitou záložku lze spravovat data. Tato položka obsahuje obvyklé možnosti. Jedná se o vkládání dat, jejich modifikaci nebo odstraňování. Pod touto kartou lze také jednoduše zobrazit data, která jsou uchována ve vybrané tabulce. Jsou zde i funkce pro import a export dat, které však obsahují pouze základní možnosti nastavení.

Pod ostatními záložkami lze nalézt také okno pro zápis příkazu (např. dotaz na databázi) v jazyce SQL. Skript je možné nahrát i z externího souboru. Dále lze spravovat např. pohledy, domény, funkce, role a uživatelská oprávnění pro přístup k databázi. Poslední karta umožňuje zobrazení statistik o databázi, pokročilou správu databáze nebo její zálohu a následnou obnovu.

FirebirdWebAdmin je sice velmi jednoduchý nástroj, poskytuje však dostatečné funkční vybavení pro znatelné urychlení práce s databází. Rozložení uživatelského rozhraní je ale poměrně nepřehledné. Nevykazuje totiž žádné známky podobnosti s jinými rozhraními, jejichž rozložení je již zaběhnuté a velmi důmyslně uzpůsobené potřebám uživatele. Při práci s tímto rozhraním není po uživateli požadována znalost jazyka SQL v takovém rozsahu, jako tomu je u grafického klienta Firebirdu *FlameRobin*. Jistá znalost SQL je však i v tomto případě nezbytná.

6.4 PhpLiteAdmin

Webové rozhraní systému SQLite se jmenuje *phpLiteAdmin* a na první pohled se vyznačuje stejnými vlastnostmi, jako celý systém SQLite. Toto rozhraní je velmi jednoduché a minimalistické (viz příloha B.4). Při testování byla využita verze 1.9.6.

Na rozdíl od ostatních webových rozhraní lze v *phpLiteAdmin* pracovat pouze se třemi typy objektů – s databázemi, tabulkami a pohledy. Tyto objekty jsou zobrazeny v levé části okna, kde mezi nimi lze jednoduše a rychle přepínat. Hlavní část rozhraní se mění podle toho, jaký typ objektu byl právě vybrán.

Po výběru databáze jsou zobrazeny hlavní informace (např. velikost a umístění souboru s databází nebo datum poslední modifikace) a seznam objektů, které jsou k databázi vázány. Dále zde lze také vytvořit nové tabulky či pohledy. V horní části rozhraní se nachází záložky, které umožňují zadávání SQL příkazů a import a export databáze. Import i export podporují 2 typy formátů (SQL a CSV), které jsou voleny v závislosti na tom, zda uživatel potřebuje přenést strukturu, data či obojí. Zajímavou funkcí, kterou *phpLiteAdmin* nabízí,

je tzv. *Vacuum*. Díky této možnosti lze odstranit volná místa, která vznikají častým mazáním dat, čímž je dosaženo defragmentace databáze. Následkem této akce může být výrazné snížení velikosti souboru s databází. Možnost *Vacuum* je vhodná především pro obsáhlé databáze. Tato funkce je implementována ve většině ostatních SŘBD a většina z nich ji využívá automaticky.

Pokud je vybrána některá z tabulek, *phpLiteAdmin* umožňuje měnit její strukturu, procházet data, vkládat je a editovat. Nachází se zde také okno, do kterého je možné zapsat SQL dotaz. Také je podporována funkce, která umožňuje vytvoření velmi jednoduchého dotazu (nad jednou tabulkou) pomocí primitivního rozhraní.

Webové rozhraní *phpLiteAdmin* působí na první pohled velmi strohým dojmem. Nový uživatel se v něm však dokáže velmi snadno zorientovat. Především v oblasti manipulace s daty dokáže ušetřit značné množství času a poskytuje dostatečný komfort. Pokud se navíc uživatel spokojí pouze s využíváním velmi jednoduchých dotazů, vystačí si i s nulovou znalostí jazyka SQL.

6.5 Výsledky porovnávání webových rozhraní

V tab. 6.1 jsou zobrazeny veškeré výsledky porovnávání webových rozhraní. Je zřejmé, že velké množství vlastností a funkcí grafických klientů nebylo přeneseno do webových rozhraní. Uživatel si tedy může pro vybrané operace zvolit rozhraní, které mu více usnadní manipulaci s databází. Dostupné webové aplikace pro systémy PostgreSQL a MariaDB jsou i v tomto případě mnohem lépe zpracované, než je tomu u zbylých dvou systémů.

	PostgreSQL	**MariaDB**	**Firebird**	**SQLite**
Čeština	ANO	ANO	Částečně přeloženo	NE
Více databází současně	ANO	ANO	NE	NE
Nutná úroveň znalosti SQL	Střední	Nízká	Střední	Střední
GUI pro sestavení dotazu	NE	ANO	NE	NE
Data modeler	NE	ANO	NE	NE
Zvýrazňování a našeptávání syntaxe SQL	NE	ANO	ANO	NE
Import a export	ANO	ANO	ANO	ANO

Tab. 6.1: Porovnání webových rozhraní

7 DIALEKT SQL

Přestože všechny vybrané SŘBD využívají dotazovací jazyk SQL, jeho dialekt se u nich liší. Nejedná se však pouze o rozdílnou syntaxi. Dialekty se liší i v závislosti na funkčním vybavením jednotlivých systémů. Pokud tedy některý systém obsahuje funkci, která se v jiných systémech nevyskytuje, musí pro tuto funkci existovat příslušný ekvivalent v SQL. Velmi zásadní rozdíly jsou také mezi datovými typy jednotlivých SŘBD. Ty jsou často odlišně pojmenovány, některé základní typy dokonce nejsou vybranými systémy podporovány.

Vzhledem k tomu, že se tato práce vztahuje pouze k relačním SŘBD, není nutné porovnávat dialekty SQL s aktuálním standardem. Od jeho verze z roku 1992 (SQL 2) totiž v oblasti relačních databází nenastaly žádné zásadní změny. [43, 44]

7.1 Datové typy

V systémech PostgreSQL, MariaDB a Firebird lze nalézt pro většinu základních datových typů příslušný ekvivalent. Mezi základní datové typy lze řadit např. celé číslo, reálné číslo (s pohyblivou a pevnou řádovou čárkou), textový řetězec, obsáhlý text, binární objekt, datum, datum a čas nebo tzv. `BOOLEAN` (pravdivostní typ). Poslední zmíněný typ existuje pouze v PostgreSQL, v ostatních systémech je potřeba jej nahradit[1] (např. pomocí `TINYINT(1)`). Všechny 3 vyjmenované systémy využívají tzv. statické datové typy. Datový typ každé hodnoty uložené v tabulce je ovlivněn tím, jaký typ sloupce byl zadán uživatelem při jeho definici. [45]

SQLite pracuje s dynamickými datovými typy. Příslušný typ je určen přímo vloženou hodnotou. V SQLite tedy nejsou při tvorbě sloupce zadávány konkrétní datové typy. Lze zde vybrat pouze z pěti tzv. typových afinit – `TEXT`, `NUMERIC`, `INTEGER`, `REAL`, `BLOB`. Každá obsahuje podmnožinu datových typů, z nichž je automaticky zvolen takový, který nejlépe reprezentuje zadanou hodnotu. [46]

Zvláštností SQLite je absence speciálního typu pro datum a čas. Ten však lze definovat pomocí 3 různých afinit:

- `TEXT` – řetězec (ve formátu „YYYY-MM-DD HH:MM:SS.SSS“).
- `REAL` – počet dní od poledne 24. listopadu 4714 před Kristem (podle gregoriánského kalendáře).
- `INTEGER` – počet sekund od půlnoci 1. 1. 1970.

[1] Některé systémy sice umožňují uživateli zvolit datový typ `BOOLEAN`, avšak vzápětí si jej sami pomocí interních procesů převedou na příslušný ekvivalent.

V tab. 7.1 jsou přehledně zaznamenány datové typy pro všechny základní druhy datových formátů. Z této tabulky je patrné, že jejich pojmenování se u různých systémů velmi výrazně odlišuje, což může být při využívání více systémů poměrně matoucí.

	PostgreSQL	**MariaDB**	**Firebird**	**SQLite**
Celé číslo	`INTEGER`	`INT`	`INTEGER`	`INTEGER`
Reálné číslo (pohyblivá řádová čárka)	`REAL`, `DOUBLE PRECISION`	`FLOAT`, `DOUBLE`	`FLOAT`, `DOUBLE PRECISION`	`REAL`
Reálné číslo (pevná řádová čárka)	`NUMERIC`	`DECIMAL`	`DECIMAL`, `NUMERIC`	`NUMERIC`
Řetězec	`CHARACTER VARYING`	`VARCHAR`	`VARCHAR`	`TEXT`
Text (CLOB)	`TEXT`	`TEXT`	`BLOB SUB_TYPE TEXT`	`TEXT`
Binární objekt	`BYTEA`	`BLOB`	`BLOB SUB_TYPE BINARY`	`BLOB`
Datum	`DATE`	`DATE`	`DATE`	-
Datum a čas	`TIMESTAMP WITHOUT TIME ZONE`	`DATETIME`	`TIMESTAMP`	-
Pravdivostní typ	`BOOLEAN`	`BOOLEAN`[2]	-	-

Tab. 7.1: Základní datové typy porovnávaných SŘBD

7.2 Automatická inkrementace

Tato funkce umožňuje vložení celočíselné hodnoty do příslušného sloupce bez nutnosti jejího definování. Operace se tedy provede automaticky. Doplněná hodnota je s každým záznamem zvyšována, čímž je dosaženo její unikátnosti. Z toho důvodu se tato funkce zpravidla používá pro sloupce označené jako primární klíč (jednoduchým způsobem je zabráněno tvorbě duplicit). [47]

Systém PostgreSQL tuto možnost podporuje. Při definování sloupce, který má být primárním klíčem, stačí pouze zadat klíčové slovo `SERIAL` (případně `BIGSERIAL` nebo `SMALLSERIAL`). Tím se vytvoří sloupec typu `INTEGER` (`BIGINT` nebo `SMALLINT`), který bude doplňovat a zvyšovat svou hodnotu automaticky. [48]

[2]Bezprostředně po zvolení této hodnoty si ji MariaDB automaticky převede na datový typ `TINYINT(1)`.

MariaDB využívá pro tuto funkci označení `AUTO_INCREMENT`. Stejně jako u PostgreSQL lze také využít klíčového slova `SERIAL`, které je však v tomto případě aliasem pro skupinu klíčových slov `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE`. [49]

Systém Firebird tuto funkci neumožňuje v žádné formě. Jediným způsobem, kterým lze požadovanou funkčnost nahradit, je vytvoření speciální spouště. Ta se aktivuje vždy při přidání nového řádku do tabulky a doplní vhodnou hodnotu podle předem zvolených parametrů. [50]

Funkční vybavení systému SQLite možnost automatické inkrementace také umožňuje. V tomto případě je funkce definována klíčovým slovem `AUTOINCREMENT`. [51]

7.3 Identifikátory

Každý databázový objekt (tabulka, sloupec, pohled, spoušť atd.) je jednoznačně pojmenován v rámci celé databáze. Tomuto jménu se říká identifikátor. Podle standardu SQL by měl být ohraničen dvojitými uvozovkami, měl by obsahovat maximálně 128 znaků a jeho součástí by neměly být mezery ani jiné oddělovací znaky (tečky, čárky, středníky). [52]

Existují dva základní typy identifikátorů. Prvním druhem jsou tzv. regulární identifikátory, které nemusí být ohraničeny žádnými znaky. Většina objektů v databázi je reprezentována tímto typem a zpravidla u nich nemusí být dodržována velikost písmen.

V databázi se však mohou vyskytovat i takové objekty, jejichž pojmenování koliduje s některým klíčovým slovem jazyka SQL. Sloupec v tabulce může být nazván např. `order` nebo `group`. Objekty také mohou být reprezentovány víceslovnými názvy (této variantě by se však měl tvůrce databáze vyhnout). V takových případech je tedy nutné zdůraznit, že se skutečně jedná o databázový objekt. Toho je dosaženo pomocí tzv. ohraničených identifikátorů (delimited identifiers). Vybraný objekt je ohraničen příslušnými znaky. [53]

Tyto znaky se u různých systémů liší. Pokud jsou identifikátory ohraničeny, některé systémy vyžadují rozlišování velkých a malých písmen (tzv. case-sensitive).

Systémy PostgreSQL a Firebird využívají jako ohraničovací znaky dvojité uvozovky (`"id"`) a jsou case-sensitive. MariaDB využívá znak zpětný apostrof (`` `id` ``) a mezi těmito znaky není nutné dodržovat velká a malá písmena. Systém SQLite umožňuje používání několika znaků. Ohraničovat jednotlivé identifikátory lze pomocí hranatých závorek (`[id]`), dvojitých uvozovek (`"id"`), zpětných apostrofů (`` `id` ``) a lze použít také apostrofy klasické (`'id'`). Poslední dva jmenované znaky však mohou sloužit i k jiným účelům, a proto by měly být používány především hranaté závorky. SQLite není case-sensitive. [54]

7.4 SELECT bez tabulky

V některých případech je potřeba provést příkaz SELECT bez nutnosti využití obsahu některé z tabulek. Tímto způsobem je možné např. počítat matematické operace nebo zobrazit aktuální datum či čas. V dotazu tedy ani nefiguruje žádná tabulka, ze které by bylo možné požadovaná data získat.

V systémech PostgreSQL, MariaDB a SQLite lze z dotazu bez problému vynechat klauzuli FROM. [55] Firebird však takovouto možnost nenabízí a klauzule FROM musí být obsažena v každém výrazu SELECT. Z toho důvodu byla vytvořena speciální tabulka, která je pojmenována `RDB$DATABASE`. Tato tabulka je pouze pro čtení, má jediný řádek, je součástí metadat každé databáze (systémová tabulka) a lze se na ni v potřebných případech odkazovat. [56]

7.5 Vícenásobný INSERT

Všechny porovnávané systémy umožňují vícenásobné vkládání řádků v rámci jediného příkazu INSERT. Při větším objemu vkládaných dat tedy není nutné stále dokola zapisovat v SQL některé části kódu, které by byly pro každý příkaz totožné. Tím může být dosaženo výrazného urychlení této činnosti.

Systémy PostgreSQL, MariaDB a SQLite tuto možnost poskytují. Více řádků lze vložit za pomoci výrazu INSERT, přičemž není nutné využívat žádných jiných příkazů. Syntaxe pro všechny tyto systémy je totožná: [57]

```
INSERT INTO table_name (col1, col2, col3)
VALUES (val1, val2, val3),
       (val4, val5, val6);
```

I když je ve Firebirdu možnost vícenásobného vkládání dat obsažena také, její provedení je o poznání komplikovanější. Kromě výrazu INSERT je nutné využít ještě příkazu SELECT a operátoru UNION ALL[3]. Vložení několika řádků do tabulky pomocí jediného příkazu INSERT může vypadat následovně: [58]

```
INSERT INTO table_name (col1, col2, col3)
SELECT val1, val2, val3 FROM RDB$DATABASE UNION ALL
SELECT val4, val5, val6 FROM RDB$DATABASE;
```

Z uvedených příkladů je zřejmé, že ve Firebirdu je provedení této činnosti výrazně náročnější (v porovnání s ostatními systémy).

[3]Operátor UNION umožňuje sjednocovat výsledky dvou a více příkazů SELECT. Každý SELECT musí obsahovat stejné množství sloupců (ve shodném pořadí) s obdobnými datovými typy. Ve výchozím nastavení zabraňuje výběru shodných hodnot. Pro povolení duplicit je nutné přidat klíčové slovo ALL. [59]

7.6 Výsledky porovnávání SQL dialektů

Výsledky porovnání vybraných systémů podle dialektu SQL jsou dostupné v tab. 7.2. Z tabulky je zřejmé, že systémy PostgreSQL, MariaDB a SQLite obsahují veškeré testované funkce. Firebird však v tomto ohledu značně zaostává.

	PostgreSQL	**MariaDB**	**Firebird**	**SQLite**
Druh datových typů	Statický	Statický	Statický	Dynamický
Auto. inkrementace	ANO	ANO	NE	ANO
Auto. inkrementace (klíčové slovo)	`SERIAL`	`AUTO_INCREMENT`, `SERIAL`	-	`AUTOINCREMENT`
Ohraničené ID (znak)	`"id"`	`` `id` ``	`"id"`	`[id]`, `"id"`, `` `id` ``, `'id'`
Ohraničené ID (case-sensitive)	ANO	NE	ANO	NE
SELECT bez klauzule FROM	ANO	ANO	NE	ANO
Vícenásobný INSERT	ANO	ANO	ANO	ANO

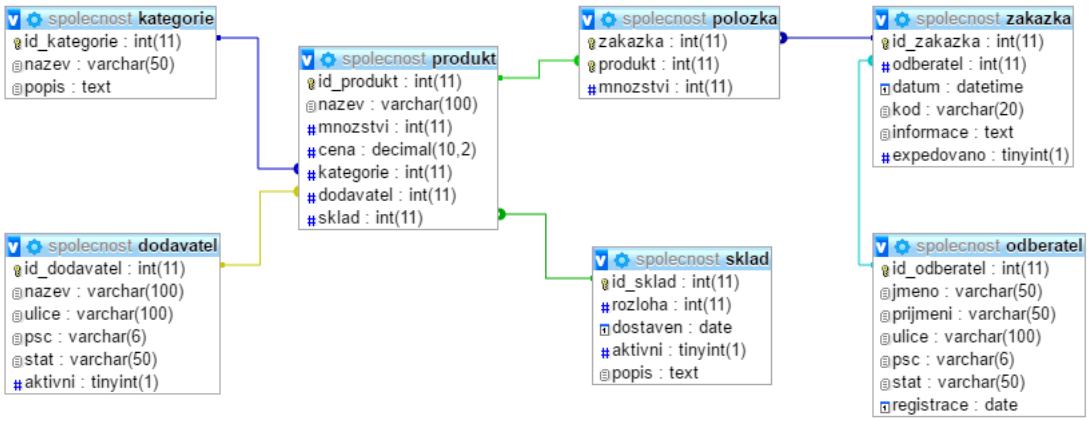
Tab. 7.2: Srovnání dialektů SQL

8 TESTOVACÍ DATABÁZE

Pro účely porovnávání jednotlivých systémů bude vytvořena testovací databáze. Ta bude sloužit především pro demonstraci tvorby struktury databáze pomocí grafických klientů. Do vytvořených tabulek budou následně vkládány také data. Díky tomu bude možné porovnat, nakolik vybraní klienti usnadní tyto činnosti. Také bude možné sledovat, jak se liší názvy jednotlivých datových typů v závislosti na využívaném dialektu.

Testovací databáze bude sloužit pro sledování pohybu zásob ve fiktivní společnosti. Pro lepší přehlednost není do modelu zahrnuta část pro obchodní styk s dodavatelem a každý produkt je odebírán od jediného dodavatele. Bude možné sledovat pouze obchodní transakce mezi společností a odběratelem (sekce pro evidenci zakázek).

Databáze je pojmenována obecným názvem `spolecnost` a navržený databázový model obsahuje 7 tabulek – `produkt`, `kategorie`, `dodavatel`, `sklad`, `odberatel`, `zakazka` a `polozka` (pomocná tabulka pro rozklad relace M:N). V tabulkách je obsažena většina základních datových typů (viz obr. 8.1).



Obr. 8.1: Návrh modelu testovací databáze[1]

[1] Export modelu byl proveden z webového rozhraní *phpMyAdmin*.

9 POROVNÁNÍ SŘBD NA TESTOVACÍ DATABÁZI

V rámci porovnávání grafických klientů bude pomocí každého z nich zaznamenán postup, díky kterému lze vytvořit část testovací databáze. Pro tento účel byla zvolena tabulka `zakazka`, jelikož obsahuje většinu základních datových typů. Poté bude do tabulky vložen jeden vzorový řádek.

9.1 PostgreSQL

Pro vytvoření nové databáze v aplikaci *pgAdmin* je potřeba kliknout v navigačním stromu na položku s databázemi a následně vybrat z nabídky položku *Nová databáze*. Tuto operaci je také možné provést výběrem položky s databázemi a kliknutím na ikonu pro vytvoření nového objektu. Tato ikona se nachází v horním horizontálním menu. Poté stačí pouze vyplnit název databáze a vybrat kódování UTF-8.

Po přepnutí se do databáze je možné vytvořit jednotlivé tabulky ve schématu public. Opět lze tuto operaci provést pomocí stejných dvou metod, jako tomu bylo u tvorby databáze. Rozdíl je pouze ten, že tentokrát jsou ve stromu objektů vybrány tabulky. Po zadání názvu tabulky je nutné přidat do tabulky sloupce. Tato možnost je dostupná při vytváření tabulky v kartě *Sloupce*. V okně, které se zobrazí po stisknutí tlačítka *Přidat*, lze definovat název sloupce, datový typ a další vlastnosti vytvářeného sloupce.

Po vytvoření všech sloupců je potřeba definovat příslušné klíče. Toho je možné dosáhnout skrze kartu *Omezení*. Pro vytvoření primárního klíče (nebo skupiny klíčů) musí uživatel vybrat tuto možnost z dostupné roletky a použít tlačítko *Přidat*. Následuje pojmenování omezení (např. `zakazka_primary`) a zvolení příslušného sloupce v kartě *Sloupce* (`id_zakazky`). Obdobným způsobem jsou definovány i cizí klíče. Zde je však v kartě *Sloupce* nutné nastavit více parametrů. Nejprve je nutné zvolit tabulku, na kterou bude odkazováno (`odberatel`), poté místní sloupec (`odberatel`) a nakonec odkazovaný sloupec (`id_odberatel`). Po potvrzení operace je klíč úspěšně spojen s požadovaným sloupcem a je možné jej nalézt u příslušné tabulky v sekci *Omezení* (v navigačním stromu).

Pro vkládání dat do tabulek existují 2 způsoby. První metoda využívá předem definovaných SQL skriptů, které *pgAdmin* poskytuje. Tento způsob je dostupný po kliknutí pravým tlačítkem na položku s tabulkami. V sekci *Skripty* se nachází možnost *Skript INSERT*. Zde stačí pouze nahradit znaky „?“ příslušnými daty a výsledný skript potvrdit pomocí tlačítka *Provést dotaz*. Tento způsob však není příliš efektivní. Existuje tedy ještě druhý způsob, který lze zvolit opět pomocí kontextového menu položky s tabulkami. Možnost *Prohlížet data* umožňuje zobrazit požadovaný počet řádků ve vybrané tabulce. V otevřeném okně lze tato data jednoduchým způsobem přidávat či editovat.

9.2 MariaDB

Tvorba databáze v *HeidiSQL* se příliš neliší od klienta *pgAdmin*. Kliknutím pravým tlačítkem na název serveru je otevřena kontextová nabídka. Zde lze vybrat položku *Vytvořit nový objekt* a následně *Databáze* (jediná aktivní možnost). Po zvolení jména a kódování je databáze vytvořena.

Tabulku je možné vytvořit opět skrze kontextové menu v případě, že je vybrána vytvořená databáze. Po zadání názvu v sekci *Obecné* lze níže přidávat sloupce (tlačítko *Přidat*). Klient také umožňuje jednoduchou změnu pořadí sloupců.

Definovat primární a cizí klíče zde lze snadněji, než tomu bylo u *pgAdmina*. Primární klíč se přiřazuje po vyvolání kontextové nabídky pro příslušný sloupec. V menu stačí vybrat položku *Vytvořit nový index* a následně *PRIMARY*. Zvolený sloupec nesmí v kolonce *Výchozí* obsahovat hodnotu `NULL`. Cizí klíč lze přidat v kartě *Cizí klíče*. Je nutné vyplnit název zdrojového sloupce, cílové tabulky a cílového sloupce. Každý vytvořený klíč je uchován v kartě *Indexy*.

Do vytvořené tabulky lze řádek přidat několika způsoby. Vždy se však uživatel musí nacházet v kartě *Data*. Nejjednodušším způsobem pro přidání nového řádku je stisknutí klávesy INSERT nebo použití zelené ikony *Vložit řádek do tabulky*. Tuto možnost lze také nalézt v kontextovém menu.

9.3 Firebird

Pomocí klienta *FlameRobin* lze databázi vytvořit v hlavním menu po kliknutí na položku *Database* a následně *Create new database*. Také lze tuto akci provést skrze kontextové menu. V obou případech však musí být vybráno místo, na kterém má být databáze vytvořena. Je potřeba vybrat cestu k souboru s budoucí databází (pojmenování databáze se doplní automaticky podle názvu souboru). Je také možné zadat autentizační údaje a kódování.

Tabulku lze vytvořit skrze kontextové menu po kliknutí na *Tables* a následně zvolením položky *Create new*. Na první pohled je zřejmé, že rozhraní pro tuto činnost se velmi výrazně liší od ostatních klientů. Je vygenerován obecný skript, ve kterém je třeba změnit název tabulky, názvy sloupců a jejich datové typy a zvolit primární a cizí klíč. Výsledný příkaz nutný k vytvoření testovací tabulky `zakazka` je zaznamenán ve skriptu 9.1. Po sestavení příkazu je nutné stisknout tlačítko *Execute statement(s)* (nebo klávesu F4) a poté *Commit transaction* (F5).

```
CREATE TABLE zakazka
(
   id_zakazka integer,
   odberatel integer,
   datum timestamp,
   kod varchar(20),
   informace blob sub_type text,
   expedovano smallint,

   PRIMARY KEY (id_zakazka),
   FOREIGN KEY (odberatel)
   REFERENCES odberatel (id_odberatel)
);
```

Skript 9.1: Tvorba tabulky v programu *FlameRobin*

Vložení dat do tabulky je také možné pomocí připraveného skriptu (kontextové menu tabulky, položka *Insert into*). Podobně jako v klientu *pgAdmin* však existuje i uživatelsky přívětivější způsob. V kontextové nabídce se nachází také položka *Select from*. Po jejím zvolení lze v nově otevřeném okně stisknout tlačítko *Insert row(s) into recordset*, které umožňuje zadávat data do tabulky snazším způsobem.

9.4 SQLite

Novou databázi lze vytvořit pomocí tlačítka *New Database*. Jediná další potřebná akce je výběr cesty k souboru, do kterého bude databáze ukládána.

Tabulku lze vytvořit pomocí tlačítka *Create table*. V nově otevřeném okně lze zvolit název tabulky a po stisknutí tlačítka *Add field* je možné přidávat sloupce. Zaškrtávací políčko *PK* umožňuje jednoduše zvolit primární klíč.

SQLite má však v oblasti tvorby tabulek jedno specifikum, kterému je třeba věnovat pozornost. Jeho SQL dialekt totiž nepodporuje u příkazu `ALTER TABLE` zadání varianty `ADD CONSTRAINT`. [60] Není tedy možné zvolit již vytvořený sloupec jako cizí klíč. Problémem je, že prostředí klienta *SQLiteBrowser* neposkytuje možnost volby cizího klíče ani při definici sloupce. Tabulku je tedy nutné nejprve vytvořit bez sloupce, který bude v budoucnu cizím klíčem, a ten později definovat dodatečně pomocí jazyka SQL. Takový příkaz může vypadat například takto:

```
ALTER TABLE zakazka ADD COLUMN odberatel INTEGER
REFERENCES odberatel(id_odberatel);
```

Také je nutné cizí klíče aktivovat, aby vše fungovalo korektně. Tato možnost se nachází v záložce *Edit Pragmas* (zaškrtnutí položky *Foreign Keys*). [61]

Záložka *Browse Data* umožňuje data nejen procházet, ale také vkládat. Nejprve je z roletky nutné vybrat příslušnou tabulku, do které se po stisknutí tlačítka *New Record* přidá nový řádek. Všechny provedené změny je nutné zapsat do databáze pomocí tlačítka *Write Changes*.

10 VÝHODY A NEVÝHODY SYSTÉMŮ

Systémy PostgreSQL a MariaDB jsou si z pohledu databázového programátora v mnoha ohledech velmi podobné. Pro oba jsou dostupná velmi kvalitní grafická i webová rozhraní, která obsahují širokou škálu implementovaných funkcí a usnadňují uživateli manipulaci s databází v mnoha různých ohledech. Rozsáhlá funkční výbava však může zpočátku působit problémy z důvodu obtížné orientace v aplikacích[1]. Jejich dialekty SQL umožňují všechny zkoumané operace. Oba systémy obsahují velké množství datových typů, což může taktéž působit potíže při výběru vhodného typu.

Bezkonkurenční výhodou systému Firebird je možnost práce s databázemi typu klient-server i s embedded databázemi. V porovnání s ostatními systémy je však jeho dostupné grafické a webové rozhraní velmi podprůměrné. To je zřejmě kvůli nevelké komunitě vývojářů těchto aplikací. Také jeho SQL dialekt neposkytuje takové množství funkcí, které bylo možné zaznamenat u jiných systémů. Pokud některou z funkcí obsahuje, je v mnoha případech její provedení znatelně složitější.

Systém SQLite je založen na principu maximální jednoduchosti. Jeho grafické a webové rozhraní proto neposkytuje příliš širokou funkční výbavu. Pro práci se systémem je však tato výbava naprosto dostačující. Aplikace jsou velmi přehledné, intuitivní a je snadné se v nich zorientovat. Možnosti SQL dialektu jsou také dostatečné, je však potřeba věnovat pozornost jistým specifikům tohoto systému. Jako výhoda může být vnímán dynamický způsob volby datových typů. Někteří uživatelé však mohou tuto vlastnost hodnotit spíše negativně.

[1] Orientace je obtížná především v nástrojích pro PostgreSQL. Uživatelská rozhraní aplikací pro systém MariaDB se zdají intuitivnější.

11 ZÁVĚR

V práci bylo představeno několik relačních SŘBD, které jsou volně dostupné a jejichž vývoj je v současné době stále aktivní. Z nich byli vybráni 4 zástupci – PostgreSQL, MariaDB, Firebird a SQLite.

Následně došlo k vytyčení několika oblastí, které jsou pro databázového programátora velmi zásadní. Mezi vybrané oblasti se řadí konzoloví klienti, grafická a webová rozhraní a používaný dialekt jazyka SQL. V rámci vyjmenovaných sekcí byla následně zvolena vhodná kritéria, která umožnila porovnávání vybraných systémů.

Byl vytvořen návrh databázového modelu pro fiktivní společnost (7 tabulek), který obsahuje většinu základních datových typů. Podle tohoto modelu byla následně vytvořena kompletní databáze v každém ze systémů. Byl zaznamenán a podrobně popsán postup tvorby části testovací databáze pomocí všech grafických rozhraní. Výsledek byl exportován ve formátu SQL. Díky tomu lze porovnat využívané dialekty dotazovacího jazyka.

Zjištěné výsledky a další skutečnosti byly řádně zdokumentovány. Vhodná data byla zaznamenána do příslušných tabulek. Po dokončení porovnání vybraných systémů došlo k poukázání na jejich silné a slabé stránky.

Tato studie je vhodná především pro začínající či středně pokročilé databázové programátory. Ti se na základě zjištěných výsledků mohou snáze rozhodnout, který ze systémů nejlépe pokryje jejich osobní potřeby a požadavky.

LITERATURA

[1] MySQL. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2015-10-02 [cit. 2015-10-03]. Dostupné z: https://en.wikipedia.org/wiki/MySQL

[2] JELÍNEK, Lukáš. Google migruje na MariaDB. LinuxEXPRES [online]. 2013 [cit. 2015-10-03]. Dostupné z: http://www.linuxexpres.cz/novinky/google-migruje-na-mariadb

[3] ŠTRAUCH, Adam. Databáze MariaDB válcuje MySQL. Root.cz [online]. 2012, 2012-12-21 [cit. 2015-10-03]. Dostupné z: http://www.root.cz/clanky/databaze-mariadb-valcuje-mysql

[4] Open-source software. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016, 2016-05-01 [cit. 2016-05-01]. Dostupné z: https://en.wikipedia.org/wiki/Open-source_software

[5] STĚHULE, Pavel. Historie projektu PostgreSQL. Root.cz [online]. 2012, 2012-05-22 [cit. 2015-10-03]. Dostupné z: http://www.root.cz/clanky/historie-projektu-postgresql

[6] Firebird: Historical Reference. Firebird [online]. 2015 [cit. 2015-10-03]. Dostupné z: http://www.firebirdsql.org/en/historical-reference

[7] SQLite. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2015-11-30 [cit. 2015-12-07]. Dostupné z: https://en.wikipedia.org/wiki/SQLite

[8] CUBRID Release History. CUBRID.org [online]. 2014 [cit. 2015-12-07]. Dostupné z: http://www.cubrid.org/release_history

[9] Apache Derby. Apache Derby [online]. 2015 [cit. 2015-12-07]. Dostupné z: https://db.apache.org/derby

[10] HyperSQL. HyperSQL [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://hsqldb.org

[11] H2 Database Engine. H2 [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.h2database.com/html/main.html

[12] Drizzle: Project information. Drizzle [online]. 2012 [cit. 2015-12-07]. Dostupné z: https://launchpad.net/drizzle

[13] LucidDB: Project News. LucidDB [online]. 2010 [cit. 2015-12-07]. Dostupné z: http://luciddb.sourceforge.net

[14] SourceForge: SmallSQL Database. SourceForge [online]. 2013 [cit. 2015-12-07]. Dostupné z: http://sourceforge.net/projects/smallsql/files/smallsql

[15] PostgreSQL Client Applications. PostgreSQL [online]. 2015 [cit. 2016-04-29]. Dostupné z: http://www.postgresql.org/docs/9.4/static/reference-client.html

[16] Psql. PostgreSQL [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.postgresql.org/docs/9.4/static/app-psql.html

[17] Pg_dump. PostgreSQL [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.postgresql.org/docs/9.4/static/app-pgdump.html

[18] Pg_dumpall. PostgreSQL [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.postgresql.org/docs/9.4/static/app-pg-dumpall.html

[19] Pg_restore. PostgreSQL [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.postgresql.org/docs/9.4/static/app-pgrestore.html

[20] MariaDB Clients and Utilities. MariaDB [online]. 2015 [cit. 2015-12-07]. Dostupné z: https://mariadb.com/kb/en/mariadb/clients-and-utilities

[21] The MySQL Command-Line Tool. MySQL [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://dev.mysql.com/doc/refman/5.7/en/mysql.html

[22] Mysql Command-line Client. MariaDB [online]. 2015 [cit. 2015-12-07]. Dostupné z: https://mariadb.com/kb/en/mariadb/mysql-command-line-client

[23] Mysqldump. MariaDB [online]. 2015 [cit. 2015-12-07]. Dostupné z: https://mariadb.com/kb/en/mariadb/mysqldump

[24] Mysqlimport. MariaDB [online]. 2015 [cit. 2015-12-07]. Dostupné z: https://mariadb.com/kb/en/mariadb/mysqlimport

[25] Backup Mode. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/gbak-backup.html

[26] Command Line Switches. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/isql-prompts.html

[27] Isql Prompts. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/isql-prompts.html

[28] Isql - Interactive SQL. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/isql-overview.html

[29] Firebird's nbackup tool. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/nbackup.html

[30] Command-line Options. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/gbak-cmdline.html

[31] Functions and parameters. Firebird [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.firebirdsql.org/manual/nbackup-functions-params.html

[32] Firebird and InterBase command-line utilities. IB Expert [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.ibexpert.net/ibe/index.php?n=Doc.FirebirdAndInterBaseCommandLineUtilities#Gbak

[33] Making and restoring backups. IB Expert [online]. 2015 [cit. 2015-12-07]. Dostupné z: http://www.ibexpert.net/ibe/index.php?n=Doc.MakingAndRestoringBackups

[34] Command Line Shell For SQLite. SQLite [online]. 2015 [cit. 2015-12-07]. Dostupné z: https://www.sqlite.org/cli.html

[35] PgAdmin - Introduction. PgAdmin [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.pgadmin.org/index.php

[36] JELÍNEK, Lukáš. PgAdmin III: mocný klient pro databázi PostgreSQL. LinuxEXPRES [online]. 2015, 2015-06-17 [cit. 2016-04-29]. Dostupné z: http://www.linuxexpres.cz/software/pgadmin-iii-mocny-klient-pro-databazi-postgresql

[37] PostgreSQL - Úvod a příprava prostředí. ITnetwork.cz [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.itnetwork.cz/postgresql/postgresql-uvod-a-priprava-prostredi

[38] FlameRobin Manual. FlameRobin [online]. 2015 [cit. 2016-04-29]. Dostupné z: http://www.flamerobin.org/dokuwiki/wiki/manual

[39] DB Browser for SQLite. SQLiteBrowser [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://sqlitebrowser.org

[40] ŠIŠKA, David. Seznamte se s WINE. Root.cz [online]. 2001, 2001-04-25 [cit. 2016-04-29]. Dostupné z: http://www.root.cz/clanky/seznamte-se-s-wine

[41] Bringing MySQL to the web. PhpMyAdmin [online]. 2016 [cit. 2016-04-29]. Dostupné z: https://www.phpmyadmin.net

[42] IbWebAdmin. SourceForge [online]. 2013 [cit. 2016-04-29]. Dostupné z: https://sourceforge.net/projects/ibwebadmin

[43] SQL. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2016-04-29 [cit. 2016-04-29]. Dostupné z: https://en.wikipedia.org/wiki/SQL

[44] SQL-92. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2015-08-26 [cit. 2016-04-29]. Dostupné z: https://en.wikipedia.org/wiki/SQL-92

[45] Comparison of relational database management systems. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2016-04-29 [cit. 2016-04-29]. Dostupné z: https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

[46] Datatypes In SQLite Version 3. SQLite [online]. 2016 [cit. 2016-04-29]. Dostupné z: https://www.sqlite.org/datatype3.html

[47] SQL AUTO INCREMENT Field. W3Schools.com [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.w3schools.com/sql/sql_autoincrement.asp

[48] PostgreSQL - AUTO INCREMENT. TutorialsPoint.com [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.tutorialspoint.com/postgresql/postgresql_using_autoincrement.htm

[49] AUTO_INCREMENT. MariaDB [online]. 2016 [cit. 2016-04-29]. Dostupné z: https://mariadb.com/kb/en/mariadb/auto_increment

[50] How to create an autoincrement column. The Firebird FAQ [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.firebirdfaq.org/faq29

[51] SQLite - AUTO INCREMENT. TutorialsPoint.com [online]. 2016 [cit. 2016-04-29]. Dostupné z: http://www.tutorialspoint.com/sqlite/sqlite_using_autoincrement.htm

[52] Information Technology - Database Language SQL. Second Informal Review Draft. Maynard, Massachusetts: Digital Equipment Corporation, 1992. Dostupné také z: http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

[53] Database Identifiers. Microsoft [online]. 2016 [cit. 2016-04-29]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms175874.aspx

[54] Delimited identifiers. Wikibooks.com [online]. 2015, 2015-09-25 [cit. 2016-04-30]. Dostupné z: https://en.wikibooks.org/wiki/SQL_Dialects_Reference/Data_structure_definition/Delimited_identifiers

[55] Select without tables. Wikibooks.com [online]. 2015, 2015-03-23 [cit. 2016-04-30]. Dostupné z: https://en.wikibooks.org/wiki/SQL_Dialects_Reference/Select_queries/Select_without_tables

[56] How to run a select without table. The Firebird FAQ [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.firebirdfaq.org/faq30

[57] Insert (SQL). Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2016-04-13 [cit. 2016-04-30]. Dostupné z: https://en.wikipedia.org/wiki/Insert_(SQL)

[58] How to insert multiple rows in a single statement. The Firebird FAQ [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.firebirdfaq.org/faq336

[59] SQL UNION Operator. W3Schools.com [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.w3schools.com/sql/sql_union.asp

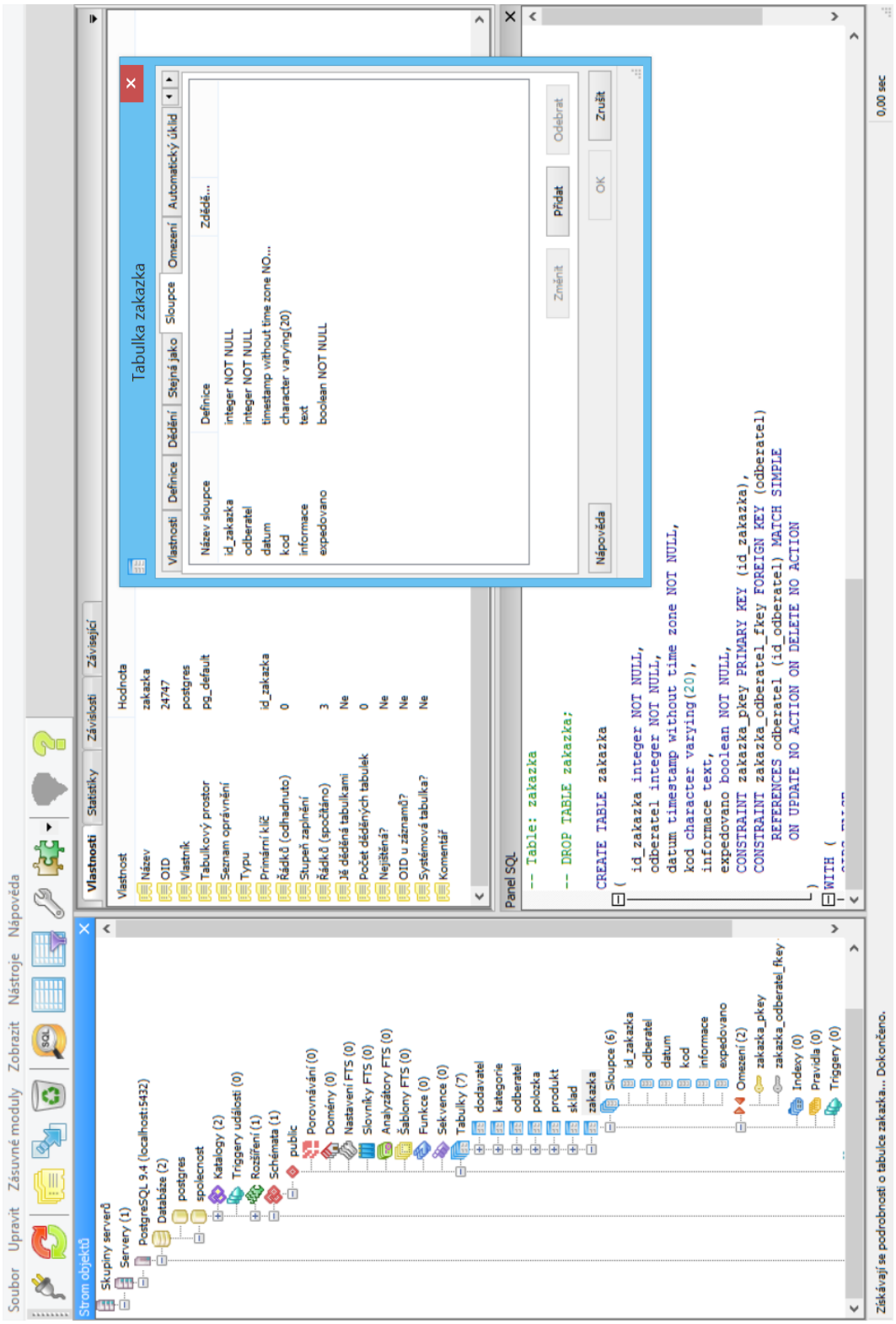
[60] SQL Features That SQLite Does Not Implement. SQLite [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.sqlite.org/omitted.html

[61] PRAGMA Statements. SQLite [online]. 2016 [cit. 2016-04-30]. Dostupné z: http://www.sqlite.org/pragma.html#pragma_foreign_keys

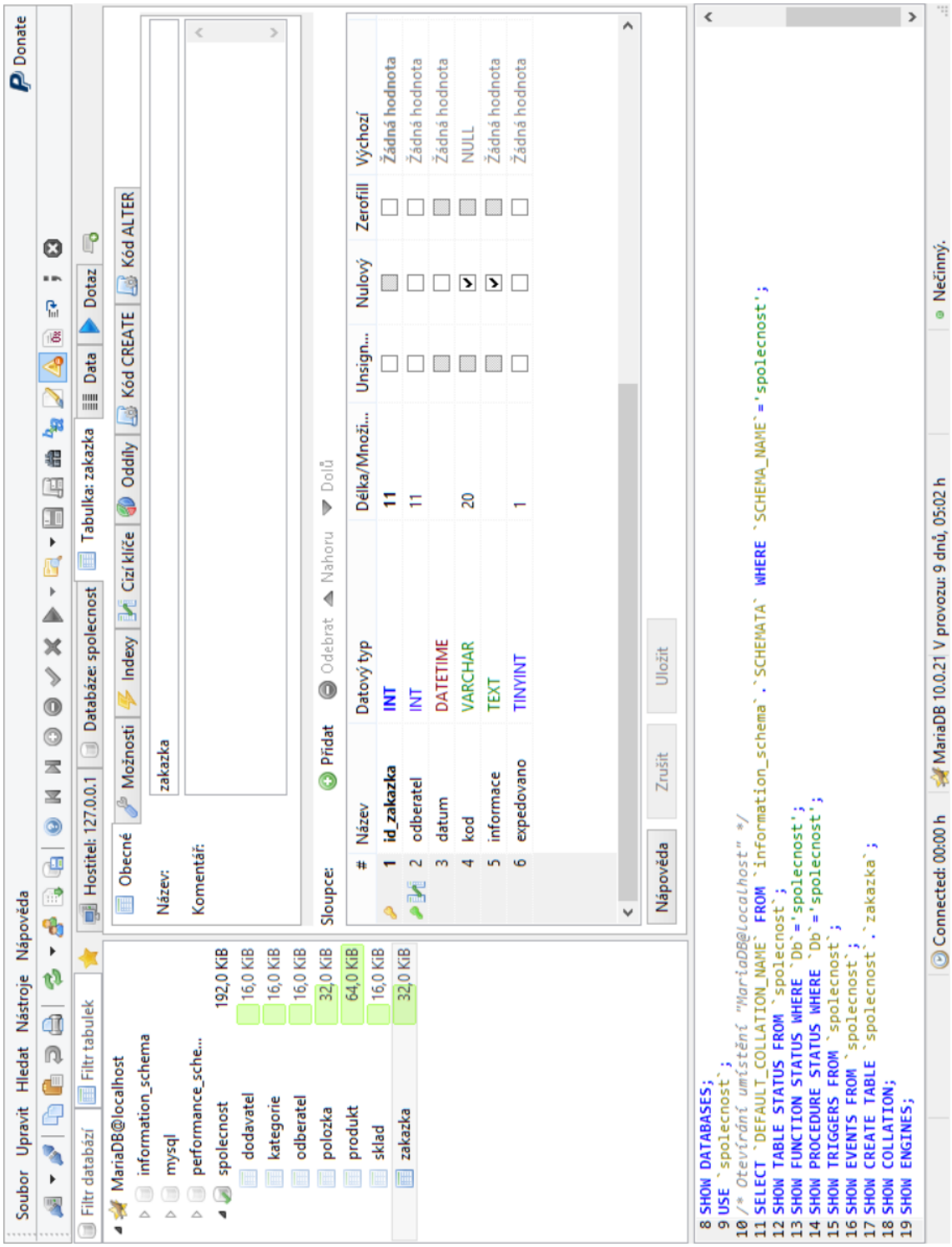
SEZNAM PŘÍLOH

A GRAFICKÁ ROZHRANÍ

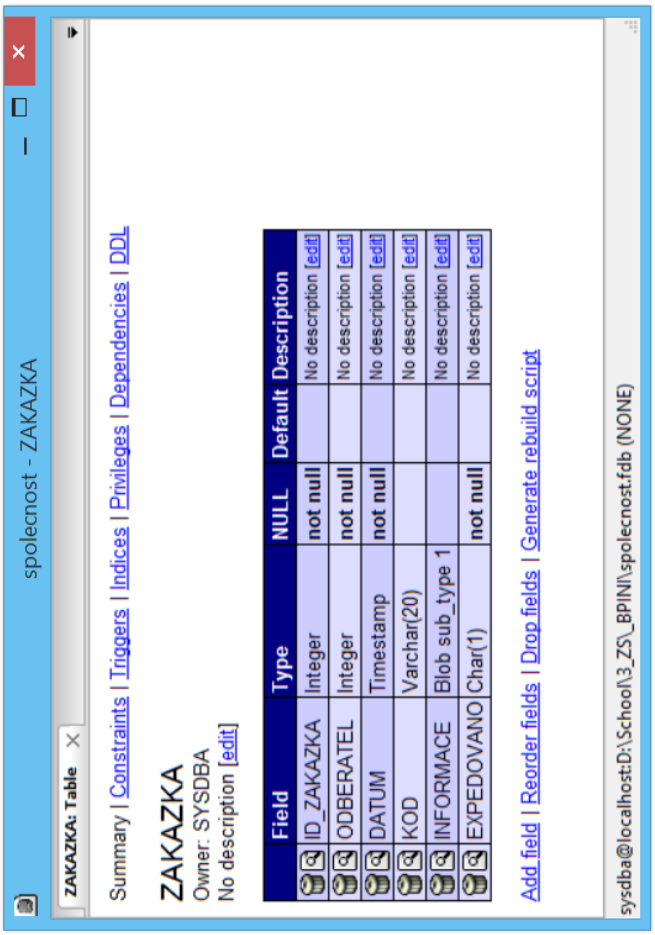
A.1 Prostředí programu pgAdmin

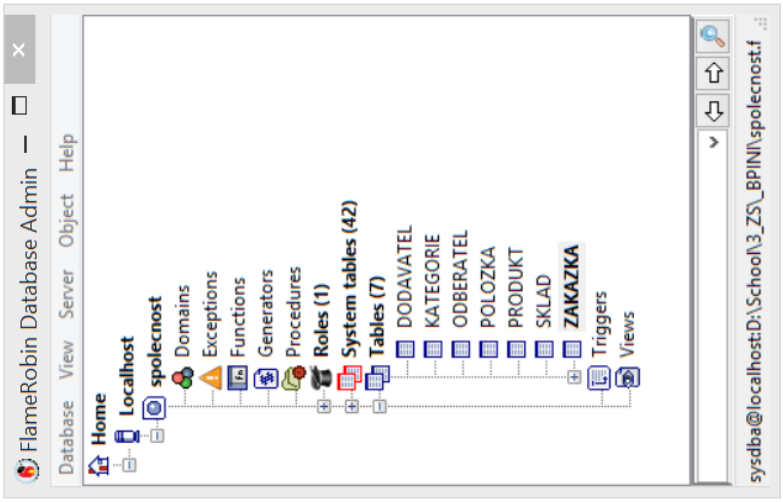


A.2 Grafické rozhraní HeidiSQL

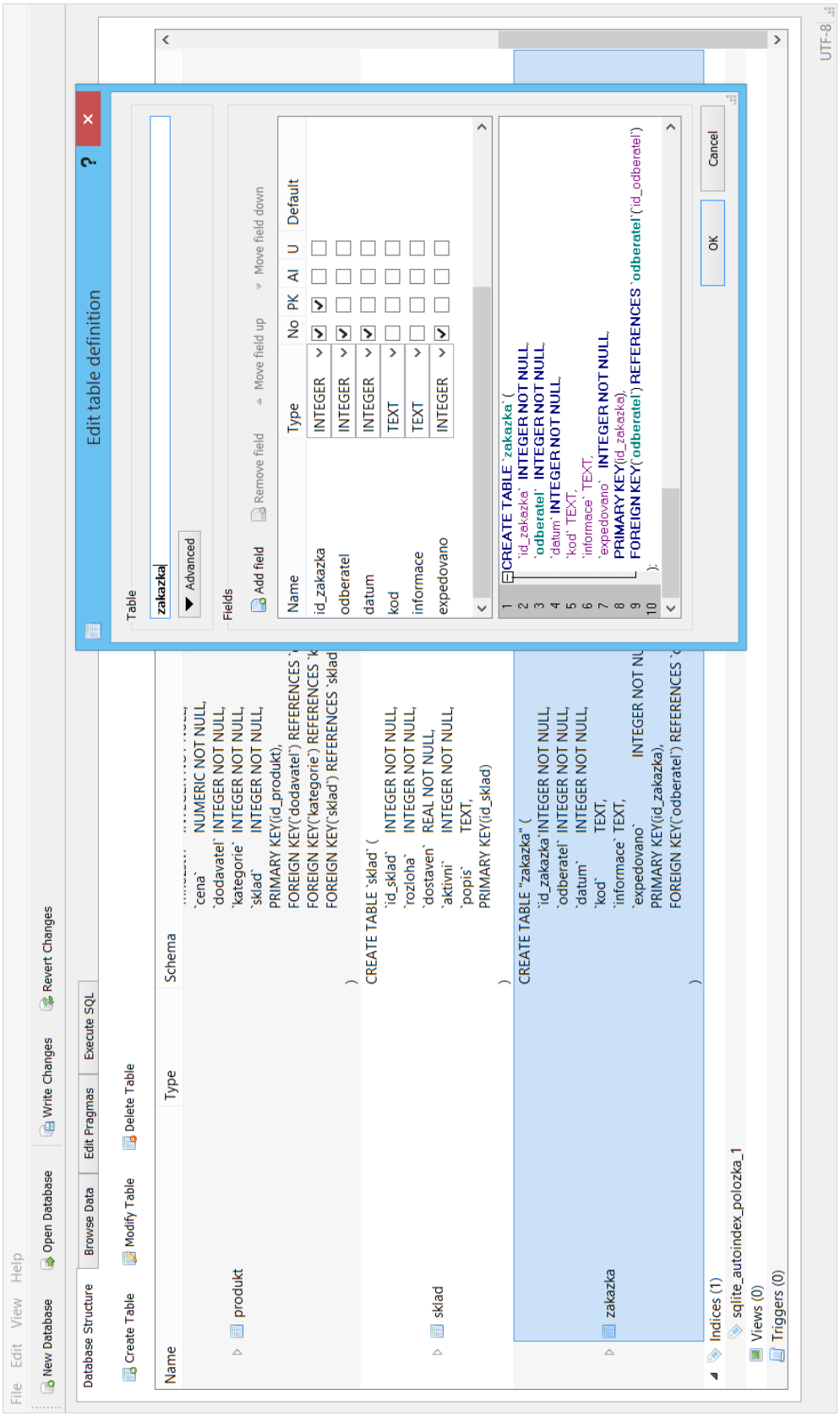


A.3 Aplikace FlameRobin



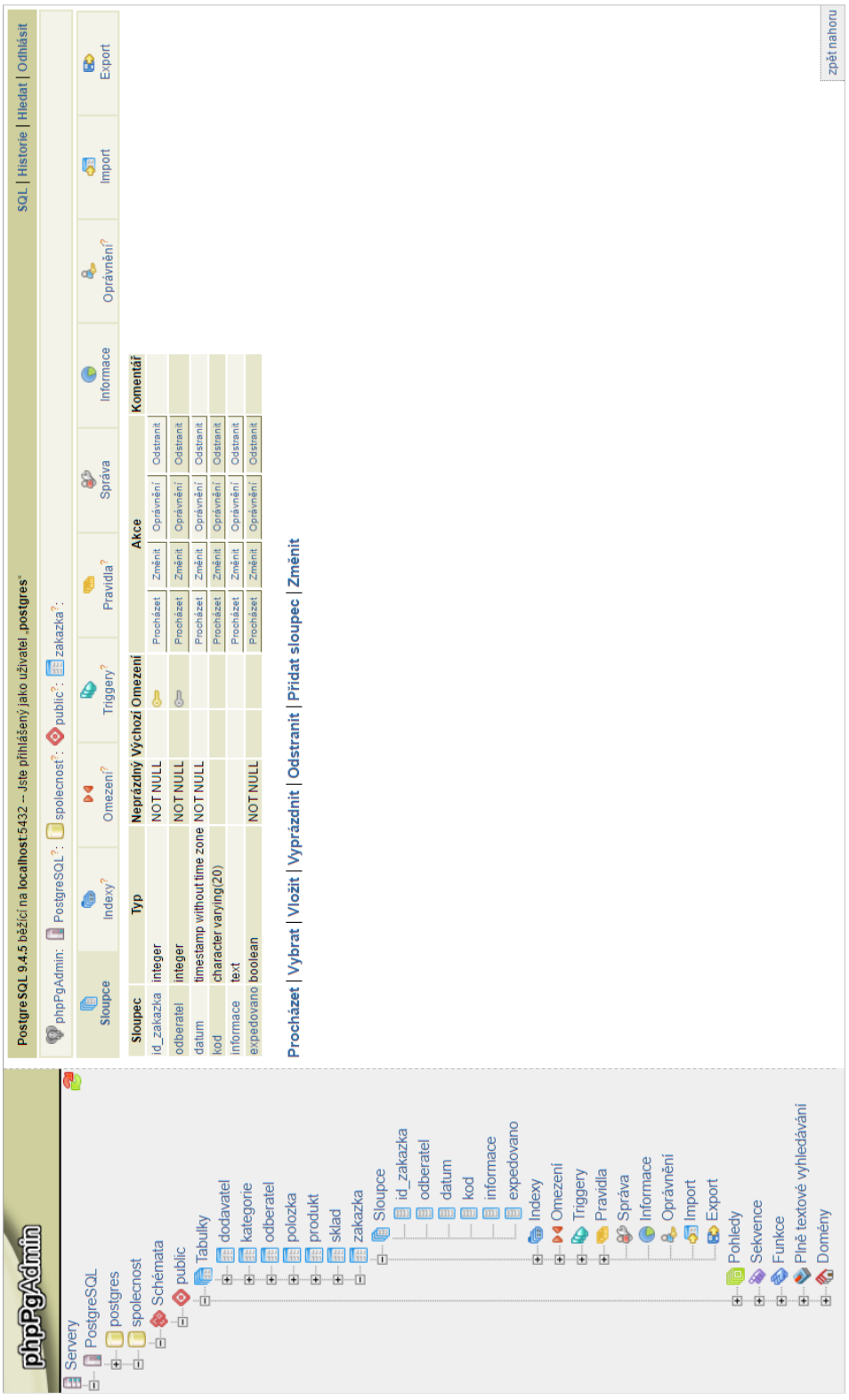


A.4 Prostředí programu SQLiteBrowser

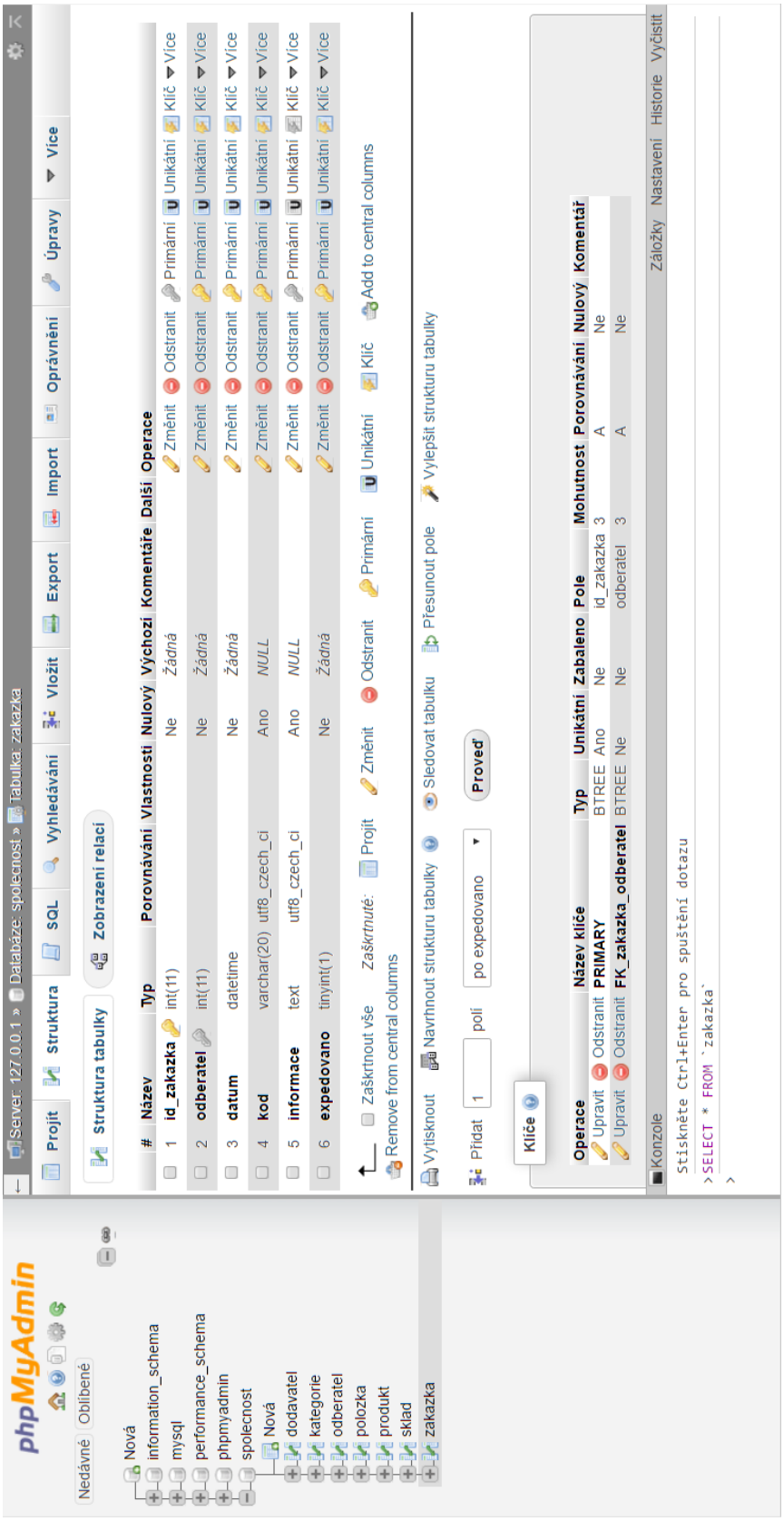


B WEBOVÁ ROZHRANÍ

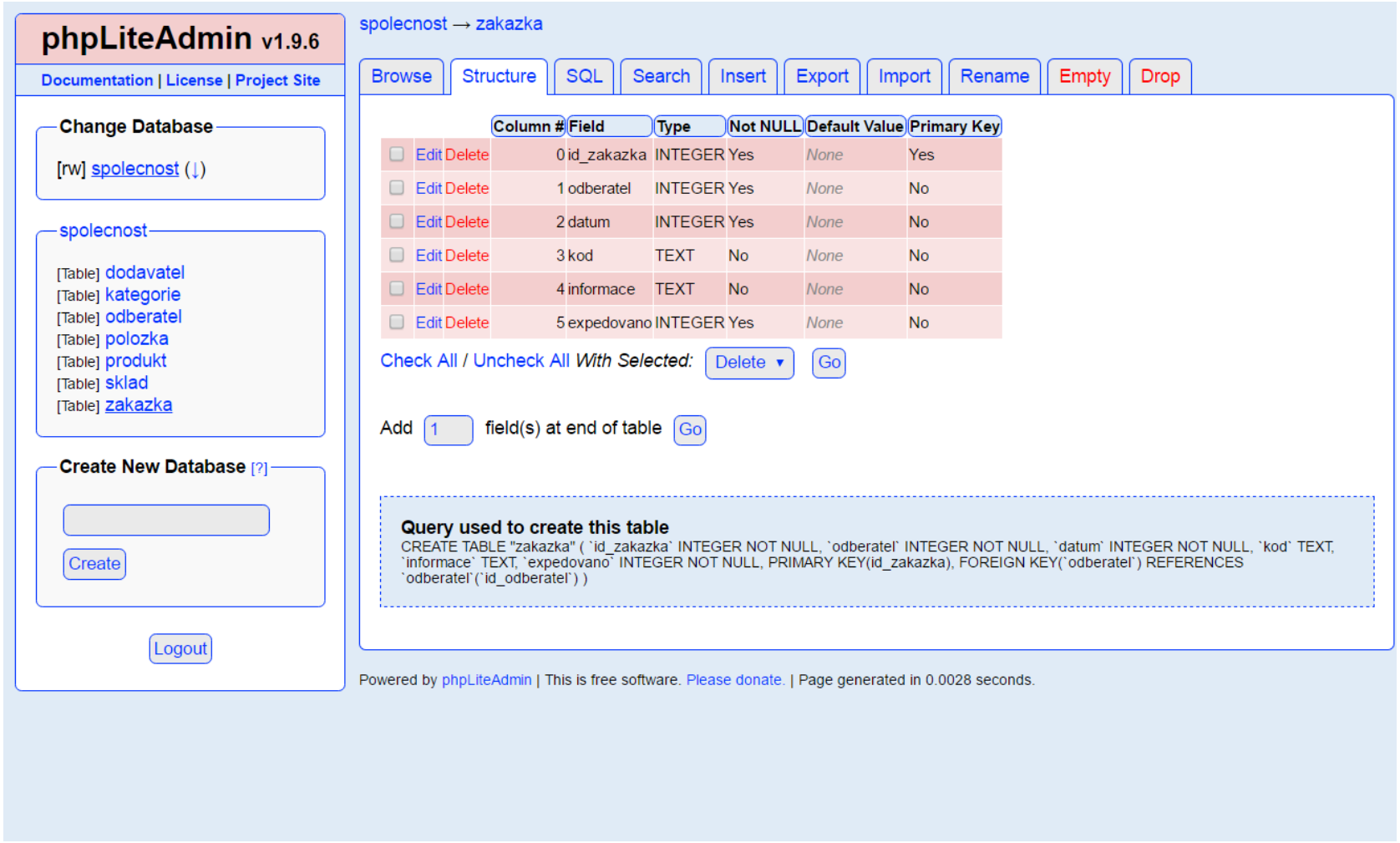
B.1 Prostředí webového rozhraní phpPgAdmin



B.2 Webové rozhraní phpMyAdmin



B.3 Webové prostředí FirebirdWebAdmin


phpLiteAdmin v1.9.6
Documentation | License | Project Site
Change Database
[rw] spolecnost (↓)
spolecnost
[Table] dodavatel
[Table] kategorie
[Table] odberatel
[Table] polozka
[Table] produkt
[Table] sklad
[Table] zakazka
Create New Database [?]
Create
Logout
spolecnost → zakazka
Browse
Structure
SQL
Search
Insert
Export
Import
Rename
Empty
Drop
Column # Field Type Not NULL Default Value Primary Key
Edit Delete 0 id_zakazka INTEGER Yes None Yes
Edit Delete 1 odberatel INTEGER Yes None No
Edit Delete 2 datum INTEGER Yes None No
Edit Delete 3 kod TEXT No None No
Edit Delete 4 informace TEXT No None No
Edit Delete 5 expedovano INTEGER Yes None No
Check All / Uncheck All With Selected: Delete Go
Add 1 field(s) at end of table Go
Query used to create this table
CREATE TABLE "zakazka" (`id_zakazka` INTEGER NOT NULL, `odberatel` INTEGER NOT NULL, `datum` INTEGER NOT NULL, `kod` TEXT, `informace` TEXT, `expedovano` INTEGER NOT NULL, PRIMARY KEY(id_zakazka), FOREIGN KEY(`odberatel`) REFERENCES `odberatel`(`id_odberatel`))
Powered by phpLiteAdmin | This is free software. Please donate. | Page generated in 0.0028 seconds.