Bc. Vlastimil Holer, <holer@fi.muni.cz>

8. března 2004

1 Úvod

SQLite je knihovna D. Richarda Hippa poskytující malý a výkonný SQL databázový engine. Implementuje velkou část standardu SQL92, tabulky a indexy, transakce, view a triggery. Její výhody spočívají v rychlosti, rozšiřitelnosti, nezávislosti na třetích programech a možnosti běžet na různých platformách – od ARM/Linux po SPARC/Solaris. Formát databáze je nezávislý na endianitě použitého stroje a může dosáhnout velikosti až 2TB.

2 Vlastnosti

Původně bylo pro uložení databáze využito GDBM (hashování). Omezovalo však projekt natolik, že se autor rozhodl implementovat vlastní způsob ukládání založený na B-stromech. Výsledkem kompilace je proto kompaktní knihovna bez externích závislostí.

Celá databáze je uložena v jediném souboru – všechny databázové objekty (view, triggery, indexy, tabulky atd.) a vlastní data. Soubor je rozdělen na stránky, jejichž velikost je možno zvolit při vytváření databáze. Možné hodnoty jsou v rozmezí od 512B do 4GB, implicitní hodnota je 1KB.

Transakce jsou implementovány s pomocí druhého souboru (žurnálu), ve kterém je uložen stav databáze před změnami. Při `COMMIT` se žurnál ruší. Pokud databáze během transakce havaruje, první klient, který se následně připojí, obnoví databázi ze žurnálu a smaže jej.

SQLite je netypový, což znamená, že je možné do sloupce uložit jakákoliv data nezávisle na určeném typu sloupce, až na pár výjimek. Autor toto prohlašuje za vlastnost. Proto při vytváření tabulek není potřeba určit datový typ sloupců (`CREATE TABLE test(a,b,c)`) – tento způsob ale snižuje přenositelnost na jiné databázové stroje, není vhodné jej používat. Netypovost se však týká pouze ukládaných dat. Pro účely porovnávání a třídění sloupec nebo výraz může být typu **numeric** nebo **text**. Typ se určí na základě typu sloupce a uložené hodnoty.

SQLite neimplementuje sekvence. Je možné je nahradit tak, že deklarujeme sloupec jako `INTEGER PRIMARY KEY`. Ukládá-li se do sloupce hodnota `NULL`, zjistí databáze maximální hodnotiu sloupce tabulky a uloží o jednu vyšší.

Velkou předností SQLite je i rozšiřitelnost – v C je možné vytvářet vlastní funkce.

Na stránkách projektu je k dispozici porovnání rychlostí mezi staršími verzemi SQLite, PostgreSQL a MySQL. Výsledky ukazují, že SQLite je v některých operacích znatelně rych-

lejší. Např.: 25000 `INSERT`ů v rámci transakce trvalo v PG 4.9s, MSQL 2.1s a SQLite 0.9s (při volání sync za každým `INSERT`em).

3 Architektura SQLite

- **Interface** je množina funkcí, které umožňují uživatelskému programu manipulaci s databází. Většina je implementována v souboru **main.c**, něco je součástí **table.c**. Interface pro jazyk Tcl je implementován v **tclsqlite.c**. Popis základních funkcí je v části 4.
- **Tokenizer a Parser** – Interface předává Tokenizeru SQL příkaz, který má být vykonán. Tokenizer jej převezme a rozdělí na tokeny, které předá parseru. Tokenizer je psaný v C a kód je umístěný v souboru **tokenize.c**, parser je v **lemon.y**.
- **Code Generator** přebírá výstup parseru. Připraví z něj miniprogram v jazyce virtuálního stroje. Každý příkaz programu je rezrezentován instrukcí (**opcodes.h** nebo podrobný popis na adrese `http://www.sqlite.org/opcode.html`) a maximálně 3 operandy. Části generátoru kódu je možné najít v souborech **build.c**, **delete.c**, **expr.c**, **insert.c**, **select.c**, **update.c** a **where.c**.
- **Virtual Machine** – vykonává instrukce miniprogramu (implementace v souborech **vdbe.c** a **vdbe.h**).
- **B-Tree Driver** – zpřístupňuje jednotlivé části databáze (implementace v souborech **btree.c** a **btree.h**)
- **Page Cache** – část programu zodpovědná za čtení, zápis a cachování bloků databázového souboru. Zároveň se stará o provádění rollback, commit a uzamykání db. souboru pro čtení/zápis (implementace v souborech **pager.c** a **pager.h**).
- **OS Interface** – kód, který usnaňuje přenositelnost knihovny na jiné platformy. Poskytuje abstrakci pro operace, které jsou závislé na konkrétní impelementaci v operačním systému. Výsledek lze najít v **os.c** a **os.h**.

4 Přístup k databázi

4.1 sqlite a skripty

Utilita **sqlite** umožňuje přístup do databáze z příkazového řádku. Provede vykonání příkazu předaného jako parametr nebo zpřístupní interaktivní konzolu, ve které je možné příkazy zadávat. Povinným parametrem programu je vždy název databázového souboru. Bližší popis je možné nalézt v manové stránce.

```
$ sqlite test
SQLite version 2.8.11
Enter ".help" for instructions
sqlite> CREATE TABLE fakulty(id,nazev);
sqlite> INSERT INTO fakulty VALUES (1,'FI');
sqlite> INSERT INTO fakulty VALUES (2,'FSS');
```

```
sqlite> SELECT * FROM fakulty WHERE id=1;
1|FI
$ sqlite test 'SELECT * FROM fakulty WHERE id=2'
2|FSS
```

Uvnitř shellových skriptů je možné využít program např. následujícím způsobem:

```
#!/bin/sh
cat << EOF | sqlite test
CREATE TABLE fakulty(id,nazev);
INSERT INTO fakulty VALUES (1,'FI');
INSERT INTO fakulty VALUES (2,'FSS');
EOF
```

4.2 C

Pro přístup k databázi z jazyka C si vystačíme s několika funkcemi:

```
sqlite* p_db = sqlite_open("/tmp/test", 0777, 0);
ret = sqlite_exec(p_db,"SELECT * FROM fakulty;",callback,&nrecs,&errmsg);
sqlite_close(p_db);
```

Funkce `sqlite_open` otevře databázi a vrátí handler na ní. Pro vykonání SQL příkazu slouží fce `sqlite_exec`, která kromě handleru databáze a SQL příkazu vyžaduje `callback` funkci a ukazatele na proměnné, do kterých se předá počet vrácených řádků nebo chybová zpráva. Zpracování řádků výsledku příkazu je prováděno v rámci fce `callback` – pro každý vrácený řádek je funkce zavolána a je pouze na ní, jak s výstupem naloží. Může v ní docházet ke zpracování výstupu nebo pouhé uložení do nějaké větší paměťové struktury. Není zde možnost získat celý výsledek příkazu.

4.3 Perl

V jazyce Perl je přístup k databázi jednodušší než v C. Příklad použití:

```
use DBI;
my $dbh = DBI->connect("dbi:SQLite:dbname=/tmp/test", "","");

my $cursor;
my @rec;

$cursor = $dbh->prepare("SELECT * FROM fakulty");
$cursor->execute();

while(@rec = $cursor->fetchrow_array)
{
    print "$rec[0], $rec[1]\n";
}

$cursor->finish;
$dbh->disconnect;
```

4.4 Další

Popularitu SQLite nejvíce vystihuje množství jazyků, pro než byl implementován interface přístupu k databázi. Z těch nejznámějších jmenujme Perl, Java, Python, Ruby, PHP a další. Existují ovladače pro DBI, ODBC a Borland DBExpress. Interface pro C a Tcl je součástí standardní instalace.

Podrobný výčet včetně odkazů je možné najít na adrese: `http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers`.

5 Oblasti využití

SQLite je vhodný pro webové stránky se středně velkým provozem. Díky své velikosti je ideální pro mobilních zařízení a PDA. Aplikace jej mohou využívat jako formát pro ukládání dat (po celý čas, kdy se s daty pracuje, probíhá transakce, která se ukončí při žádosti o uložení File/Save). Jednoúčelově jde databázi použít pro operace nad velkým množstvím dat. Protože výstupem komplikace je spustitelný program `sqlite` nebo `sqlite.exe`, je možné jej využít pro výukové účely – studenti dostanou jediný soubor, který si odnáší domů.

Nevhodné nasazení SQLite je v aplikacích typu Client/Server nebo na webových stránkách s vysokým vytížením.