KAPITOLA 3

Načítání a aktualizace dat

V této kapitole:

- Instalace databáze Sakila
- Načítání dat
- Udržování dat v aktuálním stavu
- Shrnutí

V kapitole 2 jste používali příkaz `SELECT *` k zobrazování všech záznamů z tabulek `zamestnanec` a `adresa`. Měli jste tak možnost se stručně seznámit s příkazem `SELECT`. Jenomže příkaz `SELECT` je samozřejmě daleko výkonnější a flexibilnější: pomocí něho můžete načítat pouze určité sloupce, filtrovat a řadit záznamy, či dokonce díky námi definovaným závislostem načítat i záznamy z několika tabulek najednou. V této kapitole a v kapitole 4 se dozvíte, jak používat příkaz `SELECT` nejen k těmto, ale i k mnoha dalším účelům.

V závěru předcházející kapitoly jsme vám také slíbili, že vám ukážeme příkazy `UPDATE` a `DELETE`, díky nimž budete schopni provádět čtyři základní operace s daty: vytvářet nové záznamy, načítat existující záznamy a udržovat záznamy v aktuálním stavu buď jejich aktualizacemi, anebo výmazy. V tuto chvíli již víte, jak se záznamy vytvářejí; proto se v této kapitole budeme věnovat otázkám načítání, aktualizace a výmazu.

Avšak ještě než začneme psát naše první příkazy `SELECT`, `UPDATE` a `DELETE`, musíme provést nějakou údržbu. Databáze `mysql_okamzite` nám až dosud sloužila výborně, jenomže pokud bychom ji chtěli uvést do stavu vhodného pro naše experimentování i ve zbývajících částech této knihy, strávili bychom tím poměrně dost času. Nejprve si tedy nainstalujeme databázi `sakila` (ta je pojmenována po maskotu serveru MySQL).

Instalace databáze Sakila

Přejděme tedy od používání databáze `mysql_okamzite` k práci s databází `sakila`, což je standardní výuková databáze, dostupná na webu MySQL. Tato databáze představuje činnost fiktivní půjčovny DVD, a její součástí jsou proto data týkající se zákazníků, prodeje a majetku.

Poznámka: Trocha historie

Kamenné půjčovny DVD byly prakticky nahrazeny službami umožňujícími přehrávání zvoleného videa v době, kdy chcete vy sami (video na vyžádání, viz například služba Netflix), či online půjčovnami, zasílajícími vybraná DVD poštou. Věřte mi (neboť jsem natolik starý, že si tu dobu pamatuji), že v roce 2005, kdy databáze `sakila` vznikla, tomu tak nebylo. Revoluce teprve začínala a malé rodinné videopůjčovny jste mohli nalézt skoro v každé ulici.

Při stahování databáze `sakila` a její instalaci postupujte podle následujících kroků. Podobně jako v kapitole 1 i v tomto případě platí, že pokud na vašem serveru neběží grafické uživatelské rozhraní, budete muset databázi stáhnout na nějakém jiném počítači a takto získaný archív nakopírovat na server.

1. Spusťte si webový prohlížeč a přejděte na stránku Other MySQL Documentation, mající adresu *http://dev.mysql.com/doc/index-other.html*.
2. Přesuňte se dolů do sekce **Example Databases** a klepněte na odkaz umožňující stažení odpovídající verze archívu databáze `sakila`. Uživatelům systému Linux doporučujeme archív TGZ, zatímco uživatelům systému Windows archív ZIP.
3. Přejděte do složky, do níž jste si archív databáze stáhli. Archív rozbalte a přejděte do složky `sakila-db`, která rozbalením vznikne.
   - Uživatelé systému Linux mohou použít následující příkazy:
     ```
     cd /tmp
     gzip –cd sakila-db.tar.gz | tar xvf –
     cd sakila-db
     ```
   - Uživatelé systému Windows mohou postupovat takto:
     a) Pomocí Průzkumníka Windows přejděte do složky, do níž jste si archív stáhli. Pravým tlačítkem klepněte na archív a zvolením **Extrahovat vše** rozbalte jeho obsah.
     b) Přejděte do nově vzniklé podsložky `sakila-db` a v ní opět do stejnojmenné podsložky. Poté byste již měli uvidět soubory s příponou SQL. Klepněte do adresního řádku Průzkumníka Windows a stisknutím klávesové kombinace **Ctrl+C** si tuto cestu uložte do schránky.
     c) Otevřete si okno příkazového řádku a přejděte do složky, v níž jsou uloženy soubory SQL. Cestu, kterou jste si uložili do schránky, můžete do okna příkazového řádku vložit jednoduše tak, že kdekoliv v tomto okně klepnete pravým tlačítkem a zvolíte **Vložit**. Nezapomeňte, že pokud se v cestě vyskytují mezery, je nutné takto vloženou cestu uzavřít uvozovkami:
     ```
     CD "C:\Users\<Jmeno_uzivatele>\Downloads\sakila-db\sakila-db"
     ```
4. Soubor `sakila-schema.sql` obsahuje všechny příkazy nezbytné pro vytvoření databáze `sakila`, jejích tabulek a veškerého dalšího příslušenství. V okně terminálu či příkazového řádku proveďte následující příkaz, jímž tento soubor naimportujete. Po

zobrazení výzvy zadejte heslo uživatele root (to jste si nastavili během studia kapitoly 1):

```
mysql –u root –p < sakila-schema.sql
```

5. Nyní musíme přidělit oprávnění našemu uživateli „cpress“, abychom mohli přistupovat k obsahu této nové databáze a pracovat s ním. Pomocí uživatelského účtu root se připojte k serveru MySQL a spusťte následující příkazy GRANT a FLUSH:

```
GRANT CREATE, DROP, ALTER, INSERT, UPDATE, SELECT, DELETE, INDEX, CREATE VIEW,
CREATE ROUTINE, ALTER ROUTINE, EXECUTE, TRIGGER, LOCK TABLES ON sakila.*
TO 'cpress'@'localhost';
GRANT SUPER, RELOAD, FILE ON *.* TO 'cpress'@'localhost';
FLUSH PRIVILEGES;
```

6. Nakonec ukončete připojení z klienta příkazového řádku a spuštěním následujícího příkazu načtěte do tabulek databáze sakila data uložená v souboru sakila-data.sql:

```
mysql –u cpress –p < sakila-data.sql
```

Poznámka: Více oprávnění

Všimněte si, že v tomto případě jsme uživateli cpress přidělili podstatně více oprávnění, než když jsme příkaz **GRANT** použili naposledy. Dodatečná oprávnění dávají tomuto uživateli přístup k funkcím, které budeme potřebovat až v dalších kapitolách této knihy. Všimněte si také, že jsme použili dva příkazy **GRANT** – první z nich se odkazuje na databázi **sakila** a druhý se odkazuje na *.*. Oprávnění přidělovaná tímto druhým příkazem jsou tudíž globálními oprávněními, platnými pro všechny databáze spravované daným serverem MySQL.

Zkuste si nyní prohlédnout tuto novou databázi. Pomocí příkazů SHOW TABLES a DESCRIBE či SHOW CREATE TABLE, s nimiž jste se seznámili v kapitole 1, si zobrazte všechny tabulky této databáze a jejich definice.

Načítání dat

Je zřejmé, že ukládání informací do nějakých databází by nemělo absolutně žádný smysl, pokud bychom nebyli schopni tyto informace znovu načíst. A tuto možnost nám dává jeden ze základních pilířů jazyka SQL, jímž je příkaz SELECT.

Nejprve si zkusme načíst nějaká data z tabulky actor. Tabulka je tvořena čtyřmi sloupci actor_id, first_name, last_name a last_update a obsahuje celkem 200 záznamů. Již jste si možná všimli, že znak * se používá jako zástupný znak umožňující načtení obsahu všech sloupců. Nyní si tedy ukážeme, jak se dá načíst obsah pouze těch sloupců, které nás skutečně zajímají. Ujistěte se, že jste se k serveru MySQL připojili pomocí účtu „cpress“ a že jste si jako aktivní databázi nastavili databázi sakila. Poté spusťte následující příkaz:

```
SELECT last_name, first_name FROM actor;
```

Jak vidíte, požadované sloupce, v tomto případě tedy `last_name` a `first_name`, jsou příkazu `SELECT` předávány formou seznamu, v němž jsou odděleny čárkami. Odpověď serveru MySQL by měla vypadat zhruba takto:

```
+--------------+-------------+
| last_name    | first_name  |
+--------------+-------------+
| GUINESS      | PENELOPE    |
| WAHLBERG     | NICK        |
| CHASE        | ED          |
.
.
.
| FAWCETT      | JULIA       |
| TEMPLE       | THORA       |
+--------------+-------------+
200 rows in set (0.00 sec)
```

Poznámka: Zobrazena je pouze část výstupu

Představte si to množství papíru, které by bylo spotřebováno, kdybych v každém příkladu uváděl celý výstup! Ve snaze o záchranu dalších stromů jsem se rozhodl ukázat vám pouze několik řádků ze začátku a konce výstupu. Vertikální výpustka pak představuje zbývající řádky. Takto bude výstup zobrazován i na mnoha dalších místech této knihy.

S největší pravděpodobností budou záznamy zobrazeny v tom pořadí, v němž byly do tabulky `actor` přidávány; toto však není zaručeno. Server MySQL totiž neříká, v jakém pořadí vrátí načtené záznamy. Pokud tedy chcete nějaké pořadí záznamů zaručit, musíte jej nadefinovat. Je-li vraceno 200 záznamů a jsou-li zobrazovány pouze informace ze sloupců `last_name` a `first_name`, je zcela přirozené, že budete chtít vidět záznamy v jiném pořadí než v tom, které jsme vám ukázali výše.

Řazení výstupu

Je-li pořadí záznamů ve výstupu důležité, můžeme příkaz `SELECT` rozšířit o klauzuli `ORDER BY`, díky níž server MySQL data nejprve seřadí požadovaným způsobem a teprve poté je zobrazí. Znovu spusťte výše uvedený příkaz `SELECT`, tentokrát však rozšířený o klauzuli `ORDER BY`, sloužící k seřazení záznamů:

```
SELECT last_name, first_name FROM actor ORDER BY last_name;
```

Server MySQL seřadí záznamy podle hodnot ve sloupci `last_name`, a to ve vzestupném pořadí (A–Z, 1–9). Výsledek by měl vypadat zhruba takto:

```
+--------------+-------------+
| last_name    | first_name  |
+--------------+-------------+
| AKROYD       | CHRISTIAN   |
| AKROYD       | KIRSTEN     |
| AKROYD       | DEBBIE      |
.
.
.
| ZELLWEGER    | CAMERON     |
| ZELLWEGER    | JULIA       |
+--------------+-------------+
200 rows in set (0.03 sec)
```

Jak jsme naznačili již výše, standardně jsou záznamy řazeny vzestupně. Způsob řazení je však možné zadat i explicitně, k čemuž slouží klíčové slovo `ASC` (vzestupné řazení) či `DESC` (sestupné řazení neboli Z–A, 9–1). Samozřejmě je možné data ve výstupu řadit i podle hodnot ve více sloupcích.

Zkusme náš výstup ještě vylepšit další úpravou klauzule `ORDER BY`:

```
SELECT last_name, first_name FROM actor ORDER BY last_name ASC, first_name DESC;
```

Tímto způsobem server MySQL požádáme o to, aby vybraná data seřadil nejprve podle hodnoty ve sloupci `last_name`, přičemž jsme navíc explicitně řekli, že chceme řazení vzestupné (tj. nechceme spoléhat na výchozí chování serveru), a v případě záznamů se stejnou hodnotou ve sloupci `last_name` by mělo být použito řazení podle hodnoty ve sloupci `first_name`, a to sestupné. Spustíte-li tento příkaz, uvidíte následující výstup:

```
+--------------+-------------+
| last_name    | first_name  |
+--------------+-------------+
| AKROYD       | KIRSTEN     |
| AKROYD       | DEBBIE      |
| AKROYD       | CHRISTIAN   |
.
.
.
| ZELLWEGER    | JULIA       |
| ZELLWEGER    | CAMERON     |
+--------------+-------------+
200 rows in set (0.00 sec)
```

Tabulka `actor` obsahuje pouhých 200 záznamů, díky čemuž je server MySQL schopen všechny tyto záznamy načíst, seřadit a nakonec i zobrazit ve zlomku vteřiny. Avšak se vzrůstajícím počtem záznamů či se vzrůstající složitostí příkazu `SELECT` se proces načítání dat a jejich řazení bude postupně zpomalovat. Z tohoto důvodu vám doporučujeme, abyste

se důkladně zamýšleli nad příkazy, které píšete, a abyste se vždy snažili načítat pouze ta data, která skutečně potřebujete.

Řazení výstupu je ovlivňováno nastavením, kterému se říká **collation**. Tímto pojmem se označuje sada pravidel říkajících, který znak bude při řazení vybrán jako první, který jako druhý apod. Například při použití collation odpovídajícího angličtině jsou veškeré znaky s diakritickými znaménky ignorovány, díky čemuž bude ve výstupu nejprve uvedeno příjmení „Böhm" a teprve poté „Brown". Naopak ve švédštině je znak „ö" samostatným znakem, následujícím v abecedě až za písmenem „z", a proto bude ve výstupu nejprve uvedeno příjmení „Brown" a teprve poté „Böhm". Kromě toho ale collation říká i to, zda jsou či nejsou při řazení rozlišována velká a malá písmena.

Příkaz `SHOW COLLATION` zobrazí seznam všech těch collation, které vaše instalace serveru MySQL „zná". Přitom první část názvu collation tvoří označení znakové sady, pro niž je dané collation vytvořeno. Collation, jejichž název končí na `_ci`, pak při řazení nerozlišují velká a malá písmena. Collation, jejichž název končí na `_cs`, při řazení velká a malá písmena rozlišují. A collation, jejichž název končí na `_bin`, vycházejí při řazení z binární hodnoty každého znaku.

```
SHOW COLLATION;
+--------------------------+----------+-----+---------+----------+---------+
| Collation                | Charset  | Id  | Default | Compiled | Sortlen |
+--------------------------+----------+-----+---------+----------+---------+
| big5_chinese_ci          | big5     |   1 | Yes     | Yes      |       1 |
| big5_bin                 | big5     |  84 |         | Yes      |       1 |
| dec8_swedish_ci          | dec8     |   3 | Yes     | Yes      |       1 |
.
.
.
| gb18030_bin              | gb18030  | 249 |         | Yes      |       1 |
| gb18030_unicode_520_ci   | gb18030  | 250 |         | Yes      |       8 |
+--------------------------+----------+-----+---------+----------+---------+
222 rows in set (0.02 sec)
```

Hodnota `yes` ve sloupci `Default` tohoto výstupu znamená, že dané collation je výchozím pro příslušnou znakovou sadu. Výchozí znakovou sadou serveru MySQL je `latin1` a výchozím collation je `latin1_swedish_ci` (koneckonců společnost MySQL AB sídlila ve Švédsku). Avšak tabulky databáze `sakila` byly vytvářeny tak, aby pracovaly se znakovou sadou `utf8`, jíž odpovídá výchozí collation `utf8_general_ci`. Pokud však při řazení záznamů chcete použít jiné collation než výchozí, můžete jej do klauzule ORDER BY zadat.

```
SELECT last_name, first_name FROM actor
  ORDER BY last_name COLLATE utf8_bin ASC;
```