Malé veľké databázy / 6.časť

V tejto časti seriálu budeme pokračovať vo vysvetľovaní príkazu **SELECT**. Minule sme si hovorili o základných agregačných funkciách ***SUM, MIN, MAX, COUNT, AVG*** a formulovali sme podmienku ***WHERE*** v tej najzákladnejšej forme.
Dnes si vysvetlíme a na príkladoch ukážeme ďalšie pokročilejšie parametre a príkazy. Budú to tieto kľúčové slová a klauzule: ***DISTINCT, BETWEEN, LIMIT, takzvané triedené výpisy a generované výpisy***. Aby sme však mali príklady ilustrovanejšie, doplníme si tabuľku **KNIHA** o ďalšie záznamy podľa priloženej tabuľky Tab.6-1:
(Samozrejme si môžete doplniť vlastné údaje)

id	nazov	autor	vydavatel	cis_odd	cena
8	Z polovnickej kapsy	Moric, Rudo	Mlade leta	4	89
9	Plebejska kosela	Mihalik, Vojtech	Slovensky spisovatel	1	15
10	Europou bez penazi	Hlubucek, Petr, Ing.	Roman Kasan	5	34

Ak sme tak vykonali (použitím príkazu INSERT INTO), pristúpime teraz k novým parametrom a kľúčovým slovám príkazu **SELECT**:

DISTINCT

Kľúčové slovo **DISTINCT** zamedzuje vypísaniu viacerých rovnakých hodnôt.
Ak nechceme, aby sa vypísali riadky, ktoré budú mať rovnaké hodnoty podľa stanovenej podmienky, použijeme toto kľúčové slovo. Potom sa takýto riadok vypíše iba raz. Predstavme si, že chceme vedieť, od akých vydavateľov máme v našej knižnici knihy. Môžeme použiť bežný príkaz **SELECT** s redukovaním stĺpcov na jeden, a to VYDAVATEL takto:

mysql> select vydavatel from kniha;

Dostaneme výsledok ako na výpise č. 6-1:

```
mysql> select vydavatel from kniha;
+----------------------+
| vydavatel            |
+----------------------+
| Slovensky spisovatel |
| EAAP                 |
| Smena                |
| Slovensky spisovatel |
| Computer Press       |
| Grada                |
| Computer Press       |
| Mlade leta           |
| Slovensky spisovatel |
| Roman Kasan          |
+----------------------+
10 rows in set (0.00 sec)
```

Vidíme, že príkaz **SELECT** vybral jeden stĺpec , ale z celej tabuľky, čím vypísal aj tie riadky, ktoré sa opakujú. Pri týchto pár riadkoch to nevadí, ale čo keď budeme mať tabuľku o tisícich záznamoch? Kto by v tom listoval? A práve na toto sa veľmi hodí **DISTINCT**. Takže použijeme :

mysql> select distinct vydavatel from kniha;

Vidíme, že výpis č. 6-2 je prehľadnejší, lebo zobrazil len sedem riadkov, čo sú všetci vydavatelia, OD ktorých knihy vlastníme.

```
mysql> select distinct vydavatel from kniha;
+----------------------+
| vydavatel            |
+----------------------+
| Computer Press       |
| EAAP                 |
| Grada                |
| Mlade leta           |
| Roman Kasan          |
| Slovensky spisovatel |
| Smena                |
+----------------------+
7 rows in set (0.05 sec)
```

BETWEEN

Parameter **BETWEEN** v podmienke ***WHERE*** určuje interval jej platnosti. Obecný zápis je :

SELECT ***meno_stĺpca*** **FROM** ***meno_tabuľky*** **WHERE** ***podmienka***
BETWEEN ***dolná_hranica*** **AND** ***horná_hranica***

Ak chceme vyhľadať knihy s minimálnou cenou napr. 200 Sk a maximálnou cenou 1000 Sk, urobíme to takto:

mysql> select id, nazov, autor, cena from kniha where cena BETWEEN 200 AND 1000;

Na výpise č.6-3 vidíme, že tejto podmienke vyhoveli tri záznamy.

```
mysql> select id, nazov, autor, cena from kniha
    -> where cena between 200 and 1000;
+----+-------------------------------+-----------------------+--------+
| id | nazov                         | autor                 | cena   |
+----+-------------------------------+-----------------------+--------+
|  2 | KGB                           | Gordijevsky, Oleg     | 239.00 |
|  6 | Naucte se programovat v Delphi | Binzinger, Thomas     | 439.00 |
|  7 | Pouzivame linux               | Welsh, M., Kaufman, L. | 494.00 |
+----+-------------------------------+-----------------------+--------+
3 rows in set (0.06 sec)
```

(Musíme si uvedomiť, že táto podmienka sa porovnáva v stanovenom intervale od 200 do 1000, vrátane obidvoch krajných hodnôt. To len tak na okraj, lebo v matematike sa v určitých prípadoch krajné hodnoty intervalu neakceptujú.)

LIMIT

Ak sa domnievame, že prípadný výpis príkazu SELECT by bol veľmi dlhý a chceli by sme ho obmedziť na prvých n - riadkov, použijeme kľúčové slovo LIMIT na konci príkazu. Obecný zápis je :

SELECT ***mená_stĺcov*** **FROM** ***meno_tabuľky*** **WHERE** ***podmienka*** **LIMIT** ***n, m***

Ukážme si to teraz na našej tabuľke ZANER:

mysql> select * from zaner LIMIT 5;

Vidíme prvých päť riadkov výsledku príkazu SELECT, tak ako je to na výpise č. 6-4:

```
mysql> select * from zaner limit 5;
+---------+-------------+
| cis_odd | tematika    |
+---------+-------------+
|       1 | poezia      |
|       2 | roman       |
|       3 | krimi       |
|       4 | detska lit. |
|       5 | cestopis    |
+---------+-------------+
5 rows in set (0.44 sec)
```

Ak chceme vypísať ďalšie riadky z tabuľky, použijeme kľúčové slovo LIMIT s dvomi parametrami ***n*** a ***m***. Parameter **n** je *offset* a značí, od ktorého následujúceho riadku bude výpis pokračovať, a parameter **m** značí maximum vypísaných riadkov.
Takže príkaz :

mysql> select * from zaner LIMIT 5,10;

vypíše ďalších 10 riadkov od riadku, následujúcom po 5. riadku, teda riadky 6 až 15. Keďže naša tabuľka nemá toľko riadkov, výpis sa ukončí po poslednom riadku tabuľky, tak ako je to na výpise č.6-5:

```
mysql> select * from zaner limit 5,10;
+---------+--------------+
| cis_odd | tematika     |
+---------+--------------+
|       6 | lit. faktu   |
|       7 | odborna lit. |
+---------+--------------+
2 rows in set (0.00 sec)
```

Triedené výpisy

Výpisy príkazu **SELECT** môžeme formovať nielen pomocou kľúčových slov a agregačných funkcií, ale môžeme vytvárať aj tzv. ***triedené výpisy***.
Triedené výpisy tvoríme parametrami:

- ORDER BY (zoradenie vzostupne)
- ORDER BY DESC (zoradenie zostupne)
- GROUP BY (zoskupenie)
- HAVING (zoskupenie spĺňajúce podmienku)

ORDER BY

Príkazom **ORDER BY** zoradíme výpis podľa stanoveného stĺpca **vzostupne**.
Všeobecný zápis je:

SELECT ***názvy_stĺpcov*** **FROM** ***meno_tabuľky*** **ORDER BY** ***názvy_stĺpcov***

Ak chceme zoradiť výpis z tabuľky **KNIHA** podľa abecedy v stĺpci ***NAZOV***, zadáme príkaz:

mysql> select id, nazov, autor from kniha order by nazov;

tak ako je na výpise č. 6-6:

```
mysql> select id, nazov, autor from kniha
    -> order by nazov;
+----+-------------------------------+-----------------------+
| id | nazov                         | autor                 |
+----+-------------------------------+-----------------------+
|  1 | Angelika a kral               | Golon, Anne a Serge   |
|  3 | Bratia Ricovci                | Simenon, Georges      |
| 10 | Europou bez penazi            | Hlubucek, Petr, Ing.  |
|  2 | KGB                           | Gordijevsky, Oleg     |
|  5 | Linux - prakticky pruvodce    | Sobell, Mark G.       |
|  6 | Naucte se programovat v Delphi | Binzinger, Thomas     |
|  9 | Plebejska kosela              | Mihalik, Vojtech      |
|  7 | Pouzivame linux               | Welsh, M., Kaufman, L. |
|  4 | Vtaky v trni                  | McCulloughova, Collen |
|  8 | Z polovnickej kapsy           | Moric, Rudo           |
+----+-------------------------------+-----------------------+
10 rows in set (0.05 sec)
```

Vidíme, že stĺpec ***NAZOV*** sa zoradil podľa abecedy, takže stĺpec ***ID*** už nejde po poradí.

Poznámka:
Musíme spomenúť, že práve tu vznikajú problémy s československým triedením. Ak nie je server nastavený na triedenie podľa českej abecedy, tak znaky s diakritikou nezaradí správne, teda A, Á, B, C, Č, D, Ď... atď. ale na koniec súboru v zmysle ASCII tabuľky.

Ak chceme použiť triedenie podľa viacerých stĺpcov tabuľky, zadáme názvy príslušných stĺpcov za ORDER BY. Vzorovým príkladom by mohlo byť zoradenie podľa názvu, potom podľa autorov a nakoniec podľa vydavateľstva. Zoradenie sa vykoná v tom poradí, v akom zadáme názvy jednotlivých stĺpcov.

ORDER BY DESC

Tento príkaz je veľmi podobný predchádzajúcemu, len slovko **DESC** znamená, že určený stĺpec sa zoradí **zostupne**. Ak teda chceme zoradiť výpis tabuľky **KNIHA** zostupne podľa stĺpca **AUTOR**, zadáme príkaz:

mysql> select id, nazov, autor from kniha order by autor desc;

Výsledok vidíme na výpise č. 6-7:

```
mysql> select id, nazov, autor from kniha
    -> order by autor desc;
+----+--------------------------------+-------------------------+
| id | nazov                          | autor                   |
+----+--------------------------------+-------------------------+
|  7 | Pouzivame linux                | Welsh, M., Kaufman, L.  |
|  5 | Linux - prakticky pruvodce     | Sobell, Mark G.         |
|  3 | Bratia Ricovci                 | Simenon, Georges        |
|  8 | Z polovnickej kapsy            | Moric, Rudo             |
|  9 | Plebejska kosela               | Mihalik, Vojtech        |
|  4 | Vtaky v trni                   | McCulloughova, Collen   |
| 10 | Europou bez penazi             | Hlubucek, Petr, Ing.    |
|  2 | KGB                            | Gordijevsky, Oleg       |
|  1 | Angelika a kral                | Golon, Anne a Serge     |
|  6 | Naucte se programovat v Delphi | Binzinger, Thomas       |
+----+--------------------------------+-------------------------+
10 rows in set (0.05 sec)
```

GROUP BY

Parametrom GROUP BY zoskupíme výsledok príkazu SELECT k stanovenému stĺpcu.
Obecný zápis je:

SELECT ***názvy_stĺpcov, agregačná_funkcia*** **FROM** ***meno_tabuľky*** **GROUP BY** ***názov_stĺpca_pre zoskupenie***

Predstavme si, že chceme spočítať sumu cien kníh po jednotlivých knižných oddelenia, napr. v oddelení č.1 je suma xy korún, v oddelení č.2 je suma yz korún a podobne.
Stačí, ak spočítame sumu cien a túto zoskupíme po oddeleniach.
Zadáme:

mysql> select cis_odd, sum(cena) as 'Celkom' from kniha group by cis_odd;

```
mysql> select cis_odd, sum(cena) as 'Celkom'
    -> from kniha group by cis_odd;
+---------+---------+
| cis_odd | Celkom  |
+---------+---------+
|       1 |   15.00 |
|       2 |  122.00 |
|       3 |   18.00 |
|       4 |   89.00 |
|       5 |   34.00 |
|       6 |  239.00 |
|       7 | 2006.00 |
+---------+---------+
7 rows in set (0.16 sec)
```

Na výpise č.6-8 vidíme sumy cien kníh po jednotlivých oddeleniach knižnice. Zároveň sme využili znalosti premenovania stĺpca agregačnej funkcie **SUM** na stĺpec *Celkom* pomocou ***AS***.

HAVING

Ak chceme vyššie uvedený príklad obmedziť určitou podmienkou, použijeme klauzulu HAVING.
Obecný zápis je:

SELECT ***názvy_stĺpcov, agregačná_funkcia*** **FROM** ***meno_tabuľky*** **GROUP BY** ***názov_stĺpca***
HAVING ***podmienka***

HAVING obmedzí rozsah výpisu tabuľky tým, že z agregovaných riadkov vyradí tie, ktoré nevyhovujú uvedenej podmienke.
Ak chceme výpis cien kníh po jednotlivých oddeleniach z predchádzajúceho príkladu obmedziť iba na tie riadky, kde suma je väčšia ako 100 (korún), zadáme príkaz:

mysql> select cis_odd, sum(cena) as 'Celkom' from kniha group by cis_odd having sum(cena)>100 ;

Na výpise č.6-9 vidíme, že je to redukovaný výpis č.6-8 a riadky 1,3,4 a 5 so sumou menšou ako 100 boli ignorované.

```
mysql> select cis_odd, sum(cena) as 'Celkom'
    -> from kniha group by cis_odd
    -> having sum(cena)>100;
+---------+---------+
| cis_odd | Celkom  |
+---------+---------+
|       2 |  122.00 |
|       6 |  239.00 |
|       7 | 2006.00 |
+---------+---------+
3 rows in set (0.11 sec)
```

Generované výpisy

Niekedy je veľmi potrebné uložiť obsah niektorej tabuľky (alebo aj celej databáze) do súboru a prípadné opätovné načítanie uložených dát do servera.
Môžeme tak konať z rôznych dôvodov. Pravidelná archivácia dát je jeden z najdôležitejších . Taktiež môžeme preniesť tieto dáta na iný počítač. (Veď kto by znova zadával údaje z klávesnice!).V takom prípade využijeme tzv. generované výpisy. Sú to také výpisy, kde príslušné príkazy SQL servera vygenerujú obsah stanovených databáz a ich tabuliek do súboru, ktoré je možné v prípade potreby spätne zo súboru načítať do servera.
Pomocných programov alebo príkazov je niekoľko, my sa naučíme používať štyri, rozdelené do týchto dvoch skupín:

a) príkazy, generujúce len obsah (dáta) tabuliek:

- **select .... into outfile** - vygeneruje dáta z tabuľky do súboru
- **load data infile** - načíta dáta zo súboru do tabuľky

b) príkazy, generujúce obsah aj formu (dáta aj štruktúru) tabuliek (alebo databáz):

- **mysqldump** - vygeneruje štruktúru aj dáta z tabuľky do súboru
- **source (\.)** programu mysql - vykoná SQL skript = načíta štruktúru aj dáta do SQL servera

SELECT ... INTO OUTFILE

Ak chceme uložiť iba dáta z ľubovolnej tabuľky do súboru, použijeme tento príkaz v monitore MySQL. Obecný tvar je :

SELECT * FROM *meno_tabuľky* INTO OUTFILE *'meno_súboru'* FIELDS TERMINATED BY *'znak'*

FIELDS TERMINATED BY znamená *polia ukončené* (znakom), tzv. oddelovačom.
Parameter ***znak*** je typ oddelovača jednotlivých stĺpcov. Spravidla to býva *čiarka* (,) alebo *pipe = rúra* (|).
Predstavme si, že chceme uložiť dáta z tabuľky **KNIHA** do súboru ***kniha.dat*** , kde oddelovačom stĺpcov bude pipe. Názov súboru nie je záväzný, prípona je ľubovoľná, ale **.dat** symbolizuje, že bude obsahovať dáta.
Vtedy zadáme:

mysql> select * from kniha into outfile 'kniha.dat' fields terminated by '|';

Keďže v našich dátach (v stĺpci AUTOR) už používame čiarku na oddelenie mena od priezviska, ako oddelovač použijeme iný znak, najvhodnejšie pipe, aby nedošlo k chybe.
Výpis č. 6-10 potvrdzuje, že všetko prebehlo v poriadku a bolo uložených 10 riadkov:

```
mysql> select * from kniha into outfile 'kniha.dat'
    -> fields terminated by '|';
Query OK, 10 rows affected (0.06 sec)
```

Vzhľadom na relatívnu cestu sa súbor ***kniha.dat*** nachádza v adresári *MYSQL\DATA\KNIZNICA*. Teraz sa môžeme pozrieť, aký je obsah súboru (výpis č.6-11) :

```
1|Angelika a kral|Golon, Anne a Serge|Slovensky spisovatel|2|56.00|
2|KGB|Gordijevsky, Oleg|EAAP|6|239.00|
3|Bratia Ricovci|Simenon, Georges|Smena|3|18.00|
4|Vtaky v trni|McCulloughova, Collen|Slovensky spisovatel|2|66.00|
5|Linux - prakticky pruvodce|Sobell, Mark G.|Computer Press|7|1073.00|
6|Naucte se programovat v Delphi|Binzinger, Thomas|Grada|7|439.00|
7|Pouzivame linux|Welsh, M., Kaufman, L.|Computer Press|7|494.00|
8|Z polovnickej kapsy|Moric, Rudo|Mlade leta|4|89.00|
9|Plebejska kosela|Mihalik, Vojtech|Slovensky spisovatel|1|15.00|
10|Europou bez penazi|Hlubucek, Petr, Ing.|Roman Kasan|5|34.00|
```

Je zrejmé, že oddelovač pipe (|) oddeľuje jednotlivé stĺpce tabuľky.

Teraz si môžeme tento súbor odložiť na bezpečné miesto, aby sme ho použili, keď to bude potrebné.

LOAD DATA INFILE

Predstavme si situáciu, že z určitého dôvodu je zrazu tabuľka **KNIHA** prázdna. To sa môže stať pri nechcenom výmaze (vyprázdnení) celej tabuľky (*delete from kniha* - spomínate si?), alebo je nekorektná, chýbaju jej niektoré záznamy a tak sme ju vyprázdnili. Ak máme odložený aktuálny! súbor ***kniha.dat***, nemusíme byť mrzutí. Stačí, ak použijeme príkaz **LOAD DATA INFILE**.
Jeho obecný tvar je:

LOAD DATA INFILE *meno_súboru* INTO TABLE *meno_tabuľky* FIELDS TERMINATED BY *'znak'*

Takže v našom prípade skopírujeme súbor do príslušného adresára (s názvom databáze) a zadáme príkaz:

mysql> load data infile 'kniha.dat' into table kniha fields terminated by '|';

Znak, ktorým sú oddelené jednotlivé stĺpce, zistíme nahliadnutím do súboru.

Výpis č.6-12 potvrdzuje, že bolo nahratých desať riadkov do tabuľky:

```
mysql> load data infile 'kniha.dat' into table kniha
    -> fields terminated by '|';
Query OK, 10 rows affected (0.33 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 0
```

Obsah súboru ***kniha.dat*** môžeme nahrať aj do prázdnej tabuľky s iným názvom ako je **KNIHA**. Podmienkou je, že táto prázdna tabuľka musí existovať a musí mať presne tú istú štruktúru ako tabuľka **KNIHA!**

MySQLDump

Ak chceme uložiť do súboru nielen dáta, ale aj štruktúru určitej tabuľky alebo aj celej databáze, je lepšie použiť program **MySQLDump**. Pozor! Nie je to príkaz monitoru MySQL, ale je to samostatný pomocný program MySQL servera. Preto sa spustí v príkazovom riadku príslušného operačného systému.
Obecný zápis je:

mysqldump *meno_databáze meno_tabuľky* > *meno_súboru*

Zobáčik „**>**" znamená, že výstup programu *mysqldump* sa presmeruje do súboru ***meno_súboru.*** Ak by sme ho nezadali, výstup programu by sa smeroval štandardne na obrazovku. Ak by sme zadali za zobáčikom namiesto mena súboru slovko **prn**, výstup by sa presmeroval na pripojenú tlačiareň.
Pozrime sa bližšie na obsah súboru ***zaner.sql*** (výpis č. 6-13):

```
# MySQL dump 8.2
#
# Host: localhost    Database: kniznica
#--------------------------------------------------------
# Server version  3.22.34-shareware-debug

#
# Table structure for table 'zaner'
#

CREATE TABLE zaner (
  cis_odd int(11) DEFAULT '0' NOT NULL auto_increment,
  tematika varchar(20),
  PRIMARY KEY (cis_odd)
);

#
# Dumping data for table 'zaner'
#

INSERT INTO zaner VALUES (1,'poezia');
INSERT INTO zaner VALUES (2,'roman');
INSERT INTO zaner VALUES (3,'krimi');
INSERT INTO zaner VALUES (4,'detska lit.');
INSERT INTO zaner VALUES (5,'cestopis');
INSERT INTO zaner VALUES (6,'lit. faktu');
INSERT INTO zaner VALUES (7,'odborna lit.');
```

Z výpisu vidíme, že súbor ***zaner.sql*** obsahuje SQL príkazy pre server. Na začiatku sú SQL príkazy pre vytvorenie tabuľky **ZANER** (*create table*), následujú SQL príkazy pre naplnenie tabuľky príslušnými dátami (*insert into*).
Takto vygenerovaný výpis je veľmi užitočný. Je zrejmé, že ho prípadne môžeme upraviť, doplniť dáta alebo prepracovať štruktúru tabuľky. A hlavne ho môžeme použiť aj tam, kde nie je nadefinovaná príslušná tabuľka. Takto generovaný výpis je veľmi vhodný na prenášanie celých databáz na iný počítač so serverom MySQL. A teraz už chápete, prečo som mu dal príponu **.sql**, aj keď som ho mohol pomenovať ľubovoľne.

Source (\.) programu MySQL

Môžeme povedať, že súbor ***zaner.sql*** je akýsi dávkový súbor SQL príkazov. Hovoríme mu aj **SQL skript** (z angl. script). No dobre, ale ako najjednoduchšie tieto príkazy vykonať bez toho, aby sme ich pracne preťukávali zo súboru do klávesnice? No predsa jednoduchým spustením tohto skriptu. Náš starý známy monitor MySQL má jeden parameter, ktorý umožňuje vykonať súbor s SQL skriptom automaticky tak, ako keby boli jednotlivé príkazy zadávané z klávesnice. Tento parameter sa nazýva **source** (zdroj), a označuje sa aj skrátene „**\.**" (pozri help). Ak teda zadáme:

mysql> source zaner.sql;

dostaneme požadovaný výsledok. Výpis č.6-14 ukazuje, že boli vykonané jednotlivé príkazy zo súboru ***zaner.sql***:

```
mysql> source zaner.sql
Query OK, 0 rows affected (0.22 sec)

Query OK, 1 row affected (0.22 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.00 sec)
```

Že sa tak skutočne stalo, môžeme sa presvedčiť už známym príkazom ***select * from zaner***.

Nabudúce si vysvetlíme vzájomné spojovanie tabuliek a manipuláciu s časovými údajmi SQL servera.

Miroslav Oravec