Introducere (sau mic ghid practic) în MySQL

Traducere de Leonte Carmen – carmen@infoiasi.ro

Acest capitol isi propune o introducere in MySQL indicand modalitatea prin care se poate crea, respectiv utiliza, o baza simpla de date prin programul mysql – client. Mysql (recunoscut uneori prin asocierea cu notiunea de monitor terminal) este, de fapt, un program interactiv care permite utilizatorului conectarea la un mysql server, rularea comenzilor si vizualizarea rezultatelor. Mysql poate fi de asemenea folosit si in varianta grupurilor de comenzi (batch): plasezi comenzile intr-un fisier initial pentru a solicita programului sa le execute ulterior.

Ambele posibilitati de acoperire a programului sunt prezentate in ceea ce urmeaza.

Pentru a vizualiza cateva optiuni asigurate de mysql apelati utilitarul cu opțiunea "--help":

```
shell> mysql --help
```

Capitolul presupune o instalare optimă a programului si existenta unui server mysql disponibil la care va puteti conecta. Daca acest lucru nu va reuseste, apelati la administratorul de program. Daca se intampla sa fiti chiar Dvs. insiva acela, atunci vi se propune si consultarea altor fragmente de specialitate din acest manual.

Tot aici este descris intregul proces al construirii si manipularii unei baze de date. Daca sunteti interesat doar in accesarea uneia deja existente, aveti posibilitatea de a omite acele subcapitole care descriu in amanunt crearea bazei de date si a tabelelor implicite.

Si, pentru ca acest capitol este totusi unul "eminamente" descriptiv multe alte detalii pot fi de asemenea ignorate. Consultati totusi secventele relevante ale manualului pentru acumulaea mai multor informatii legate de notiunile intalnite.

3.1 Conectarea si de-conectarea de la server:

Pentru a va conecta la server, trebuie, de regula, sa folositi un nume de utilizator mysql insotit de cele mai multe ori de o parola. Daca serverul ruleaza pe un alt computer decat cel de pe cae v-ati logat ca trebui specificat si numele "gazdei".

Apelati administratorul pentru a va informa care sunt parametri optimi folositi pentru conectare (care este "gazda", numele de utilizator, parola). Odata stiuti, ar trebui sa va reuseasca conectarea prin:

```
shell> mysql -h host -u user -p
Enter password: ********
```

******** reprezinta parola: aceasta o introduceti in momentul in care programul afiseaza:

```
shell> mysql -h host -u user -p
Enter password: ********
```

Daca acest lucru va reusit urmeaza sa vizualizati cateva informatii introductive urmate de o "invitatie" mysql.

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log
Type 'help' for help.

mysql>
```

Prompterul te anunta ca este gata in acest moment sa-ti primeasca comenzi. Unele instalari mysql permit utilizatorilor si o conectare anonima (fara ca numele de utilizator sa fie specificat), rularea facandu-se pe calculatorul gazda.

Daca acesta este si cazul calculatorului Dvs., ar trebui ca, insalarea sa fie posibila prin simpla apelare a mysql-lui fara alte mentiuni.

```
shell> mysql
```

Dupa ce ati reusit conectarea va puteti deconecta oricand doar tastand QUIT dupa invitatia mysql.

```
mysql> QUIT
Bye
```

Acelasi lucru il puteti obtine apasand tastele Control-D.

Exemplele ce vor urma va vor elucida teoria conectarii la server. Un posibil indiciu al acestui demers il poate constitui prompterul mysql.

3.2 Comenzi de intrare

Reasigurati-va de conectarea la server dupa intructiunile anterioare. Deocamdata nu veti selecta nici o baza de date spre a o manipula propriu-zis, dar va urma. In acest stadiu e foarte important sa aflati cate ceva despre formularea comenzilor pentru a putea face apoi un salt direct spre crearea tabelului, introducerea datelor, sau dupa caz, anularea altor date. Acest subcapitol enumera o serie de principii de

baza legate de comenzile de intrare, prin folosirea unor linii de comanda (pe care le puteti incerca pentru a va familiariza cu modul de lucru mysql). Iata o comanda simpla, care solicita serverului sa va indice numarul exemplarului si data curenta. Tastati-o dupa exemplificarile urmatoare si apasati tasta Enter:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION() | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19 |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

Toate aceste linii de comanda sugereaza citeva aspecte proprii programului mysql:

→o comanda consta de obicei din “invitatia ” mysql, urmata de “;”-dar exista si o serie de exceptii:cind folosirea acesteia nu este necesara,exemplu QUIT.Vom anexa si alte posibile circumstante , ceva mai tirziu.

→in momentul in care tastati o comanda, mysql-ul o transmite server-lui spre executie si afiseaza rezultatele, reluind apoi sirul mesajelor invitatie spre a va atentiona ca este pregatit pentru o alta comanda;

→ mysql afiseaza si comenzi sub forma tabelara (pe linii si coloane). Prima linie contine etichete pentru coloane, iar urmatoarele reprezinta rezultatele la comenzi. In ceea ce priveste coloanele, etichetele acestora indica de regula numele coloanelor alese din tabelele bazei de date. Daca urmariti sa obtineti vizualizarea doar unei expresii si nu a unei intregi coloane din tabel (cum arata exemplul urmator) mysql va eticheta coloana folosind expresia in sine.

→ programul poate indica in orice moment numarul liniilor modificate si ce anume din linia de comanda a fost executat, ceea ce, fireste confera o imagine de ansamblu asupra performatelor serverului.

Aceste valori nu sunt totusi foarte precise, pe de o parte ele ghidandu-se dupa timpul curent general si nu dupa cel real al calculatorului iar pe de alta parte sunt afectate de unii factori care tin de capacitatea serverului si latenta retelei. Cuvintele cheie pot fi introduse in oricare dintre casute. Urmatoarele linii de comenzi o demostreaza:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Iata o alta secventa. Ea demonstreaza ca puteti folosi mysql ca pe un calculator numeric:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
```

```
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
| 0.707107 | 25 |
+-------------+---------+
```

Comanda anterioara face parte din categoria celor scurte “de o singura linie”, dar pot fi introduse mai multe cerinte consecutiv pe aceeasi linie de comanda. Tot ce trebuie sa faceti este sa nu uitati sa tastati “;” dupa fiecare dinte ele.

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION() |
+--------------+
| 3.22.20a-log |
+--------------+
+---------------------+
| NOW() |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

O comanda nu terbuie introdusa neaparat pe o singura linie, astfel incat acel tip de comenzi care depasesc lungimea standard, nu sunt o problema. Programul se va ghida pentru a decide lungimea comenzilor dupa mentiunea “;” si nu dupa capacitate liniei in sine. Cu alte cuvinte, mysql-ul accepta introducerea comenzilor in sistem liber, dar nu le va si executa decat dupa recunoasterea “;”.

Iata o comanda simpla introdusa pe mai multe linii:

```
mysql> SELECT
-> USER()
-> ,
-> CURRENT_DATE;
+--------------------+--------------+
| USER() | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18 |
+--------------------+--------------+
```

In acest exemplu observati cum prompterul modifica mysql> in -> in momentul in care trece la urmatoarea linie. Prin acest procedeu mysql-ul indica faptul ca nu a recunoscut inca o comanda finalizata si asteapta restul acesteia. Considerati-l drept un prieten pentru ca el va fi acela care va va atrage atentia. Astfel veti sti care este urmatorul pas asteptat de program de la Dvs. Daca va razganditi in privinta unei comenzi tastati \c:

```
mysql> SELECT
-> USER()
-> \c
mysql>
```

Si aici observati prompterul! Revine la comanda initiala dupa ce ati tastat \c indicand ca programul este gata pentru o noua comanda.

In urmatorul tabel sunt evidentiate cateva dintre posibilele mesaje-prompter si intelesurile lor:

```
mysql> Gata pentru o noua comanda
-> Asteapta o noua linie dintr-o comanda multipla
'> Asteapta un nou rand colectand un sir inceput cu (‘’’).
"> Asteptand pentru un nou rand colectand un sir inceput cu (‘"’).
```

O comanda simpla “single line” va fi in mod accidental perceputa ca fiind una multipla daca este omisa precizarea “;”. In acest caz prompterul va anunta ->, tastati “;” si programul va executa comanda.

```
mysql> SELECT USER()
->
```

Daca va confruntati cu aceasta situatie (daca sunteti siguri ca ati introdus o comanda dar singurul raspuns pe care-l primiti este ->), mai mult ca sigur mysql asteapta ptrecizarea “;”.

```
mysql> SELECT USER()
-> ;
+--------------------+
| USER() |
+--------------------+
| joesmith@localhost |
+--------------------+
```

'> este folosita in cazul sirurilor. In mysql termenii unui sir sunt introdusi intre ‘ sau ” (exemplu ‘hello’ si “goodbye”). Cand promterul va indica ‘> sau “> inseamna ca ati inclus in comanda un sir ce incepe cu ‘ sau ” dar ati omis repetarea semnului grafic din finalul sirului “;”. In cazul in care este vorba de un sir multiplu pe mai multe randuri totul este in ordine, dar de cele mai multe ori prompterul are rolul de a va avertiza asupra caracterului omis:

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
">
```

Daca ati introdus acesta comanda tip SELECT, tastati ENTER si asteptati rezultatul, veti vedea ca nu are sa se intample nimic. In loc sa va intrebati de ce dureaza atat observati indiciul “> va sugereaza faptul ca sirul a fost incheiat. Ce faceti in aceasta situatie? Puteti anula comanda; oricum, optiunea \c nu este preferabila in acest caz, pentru ca mysql il va interpreta drept parte a sirului aflat in derulare.

Daca in schimb, introduceti caracterul grafic ce confirma incheierea sirului puteti tasta ulterior \c:

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
"> "\c
mysql>
```

Prompterul va afisa de asta data mysql> indicand ca este gata pentru o noua comanda.

3.3 Crearea si manipularea unei baze de date

Acum ca v-ati insusit modalitatile de introducere a comenzilor este timpul sa accesati o baza de date.

Sa presupunem ca aveti citeva animale de companie (mica voastra menajerie) si ca v-ar sa tineti o mica evidenta a mai multor tipuri de date in ceea ce le priveste.

Puteti sa faceti acest lucru crind o serie de tabele pentru pastrarea (monitorizarea) si respectiv completarea informatiilor dorite. Veti putea raspunde astfel la o gama larga de intrebari despre animalele voastre de casa, doar accesind informatiile stocate in tabele.

Aceasta sectiune va invata cum sa procedati pentru:

² Crearea unei baze de date
² Crearea unui tabel
² Introducerea datelor in tabel
² Accesarea informatiilor pe mai multe cai
² Folosirea mai multor tabele consecutiv

Baza de date a manajeriei va fi una simpla (am luat-o in mod deliberat asa) , dar, va ilustra cu succes tipuri similare de tabele cu acoperire in spatiul real.

De exemplu o astfel de baza de date ar putea fi utila unui fermier pentru a tine evidenta septelului, sau, unui veterinar pentru registrele parintilor sai.

Organizarea datelor pentru mica menajerie presupune o serie de comenzi si date standard care pot fi preluate de site-ul MySQL.

Puteti folosi fie formatul tar sau zip .

Pentru a afla bazele de date existente pe server apelati optiunea SHOW:

```
+----------+
| Database |
+----------+
| mysql |
| test |
| tmp |
+----------+
```

Lista bazelor de date va fi probabil alta pe calculatorul personal; dar, mai mult ca sigur mysql si test au toate sansele sa se numere printre celelalte.

Baza de date mysql este solicitata indeosebi pentru ca descrie o serie de facilitati utilizatorului.

Baza de date -test e preferata ca suport practic, ca foaie de lucru pentru diverse incercari preliminare.

! Daca nu beneficiati de avantajul SHOW DATABASES nu puteti vizualiza toate bazele de date.

! Daca exista baza test incercati s-o accesati astfel:

```
mysql> USE test
Database changed
```

! Observati ca USE, ca dealtfel, utilitarul QUIT nu necesita “;” (Daca insa ati finalizat deja cererea, comanda cu acest semn ortografic, nu este nevoie sa-l stergeti, nu va afecta structura cu nimic).

Facilitatea USE e una speciala dintr-un punct de vedere: necesita o comanda simpla (1 linie).

Puteti folosi si baza de date test (de vreme ce ati accesat-o) pentru exemplele urmatoare, dar, exista un incovenient, puteti avea surpriza ca datele din tabele sa fie modificate, orice alt utilizator cu acces la baza de date ar putea-o face. Din acest motiv ar trebui sa cereti administratorului mysql o baza pe cont propriu. Sa presupunem ca o veti numi: menajerie. Administratorul trebuie sa execute o linie de comanda de forma:

```
mysql> GRANT ALL ON menagerie.* TO your_mysql_name;
```

→ unde your_mysql_name este chiar numele de utilizator asociat Dvs.

3.3.1 Crearea si selectarea unei baze de date

Daca administratorul a creat baza cu permisiunea Dvs. puteti incepe sa o folositi. In caz contrar puteti s-o construiti singuri:

```
mysql> CREATE DATABASE menagerie;
```

Sub UNIX denumirile bazelor de date sunt sensibile din puctul de vedere al procesarii (nu este si cazul cuvintelor cheie SQL) a.i. va trebui sa aveti grija cum selectati baza, folositi mengerie, nu Menajerie sau MENAJERIE. Acelasi lucru este valabil si petru numele tabelelor. (Sub WINDOWS aceste restrictii cad).

Crearea unei baze nu inseamna automat si selectarea ei; trebuie sa faceti asta separat. Pentru accesarea propriu-zisa a bazei menajerie folositi comanda:

```
mysql> USE menagerie
Database changed
```

Procesul crearii bazei are loc o singura data; in shimb o puteti selecta ori de cate ori incepeti o sedinta mysql. Puteti accesa printr-o comanda USE, cum ati putut vedea anterior. Altfel, puteti apela la o linie de comanda in cazul in carea apelati mysql-ul.

Specificati doar numele bazei dupa fiecare dintre parametri de comentare pe care i-ati putea furniza. Exemplu:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

Observati ca menajerie nu este si parola, in linia de comanda.

Daca vreti sa redirectionati parola direct pe linia de comanda dupa optiunea –p, puteti s-o faceti direct fara spatii, litere intermediare (ex., -pmypassword, nu –p mypassword).

Oricum situarea parolei pe aceiasi linie de comanda nu este recomandabila, pentru ca o expuneti astfel unei eventuale modificari (de catre alti utilizatori logati in acelasi calculator).

3.3.2 Crearea unui tabel

Crearea bazei de date este partea usoara, dar in acest moment selectand optiunea SHOW TABLES veti observa ca este goala:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

De aici incepe “greul”, sa decideti structura bazei Dvs. de date, de ce tabele aveti nevoie si ce coloane vor contine.

Veti dori un tabel care sa contina cate o inregistrare pentru oricare dintre animalele Dvs. Il putem numi pet (animale de companie) si va trebui sa contina, cel putin, numele fiecarui animal. Pentru ca doar numele in sine nu este foarte interesant, tabelul ar trebui sa includa si alte informatii).

De exemplu, daca nu sunteti singurul din familie ce detine animale de companie ati putea inregistra, de pilda, posesorii acestor animale. Ati putea deasemenea sa incercati sa inregistrati cateva referinte descriptive succinte asupra speciei si sexului animalului.

Dar cum ramane cu varsta? Ar putea prezenta interes dar este preferabil sa nu apelati la un tabel pentru aceasta optiune. Varsta se schimbas odata cu trecerea anilor, ceea ce inseamna ca ar trebui sa reactualizati destul de des baza de date. In schimb, ati putea inregistra, valori fixe, cum ar fi data nasterii. Astfel, oricand veti avea nevoie sa aflati varsta veti face doar diferenta dintre data curenta si ce inregistrata.

Mysql asigura si facilitati pentru calculele aritmetice astfel incat n-ar trebui sa fie prea dificil.

Stocarea datelor de nastere in locul varstei propriu zise prezinta si alte avantaje, precum:

- Puteti folosi baza pentru sarcini ca generarea unor semnale pentru a preintampina zilele onomastice ale animalelor de companie (daca aceasta optiune vi se pare una usor naiva nu uitati ca puteti folosi in mod analog baza in afaceri; pentru a va reaminti de pilda ziua unui client sau asociat si ai transmite urarile de rigoare).
- Puteti calcula varsta si relationand cu alte date decat cea curenta. De exemplu, daca inregistrati ziua decesului puteti calcula cati ani avea animalul de companie pana la acest moment.

Va puteti gindi la o multitudine de alte tipuri de informatii pe care le-ati putea introduce intr-un tabel pet , dar, cele amintite pina acum sint mai mult decit suficiente pentru exemplificare: nume, posesor, specie, sex, data de nastere si data decesului.

Folositi optiunea CREATE TABLE pentru a specifica parametrii tabelului.

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
->species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

Varchar este o alegere fericita pentru coloanele aferente Numelui, Posesorului, si Speciei, pentru ca valorile lor vor varia in lungime. Lungimea acestor coloane nu trebuie sa fie aceiasi si nu neaparat de 20.

Puteti alege orice lungime intre 1 si 125; oferta mai mult decat generoasa. Daca ati optat gresit si se dovedeste mai tarziu ca ati fi avut nevoie de un camp mai larg mysql va genera optiunea ALTER TABLE.

Iata cateva dintre valorile ce pot fi alese pentru indicatorii sexului in “registrele” fiecarui animal: “m” , “f” - “male”, “female”. Desigur cea mai simpla optiune este prima.

Datele de nastere si respectiv de deces se inregistreaza prin intermediul optiunii DATE. Acum ca ati creat tabelul, SHOW TABLES ar trebui sa reliefeze cateva optiuni de iesire:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| pet |
+---------------------+
```

Pentru a verifica daca tabelul a fost creat sau nu dupa cum v-ati dorit folositi comanda DESCRIBE (aceiasi comanda este valabila si cand ati uitat de exemplu numele unei coloane).

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name | varchar(20) | YES | | NULL | |
| owner | varchar(20) | YES | | NULL | |
| species | varchar(20) | YES | | NULL | |
| sex | char(1) | YES | | NULL | |
| birth | date | YES | | NULL | |
| death | date | YES | | NULL | |
+---------+-------------+------+-----+---------+-------+
```

3.3.3 Incarcarea datelor de pe suport fizic

Dupa ce ati creat tabelul, trebuie sa-l “populati”. Expresiile LOAD DATA si INSERT sunt tot ce va trebuie.

Sa presupunem ca inregistrarile Dvs. pot fi descrise astfel:

nume	**posesor**	**specie**	**sex**	**data nasterii**	**data decesului**
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	

Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1998-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

In momentul in care aveti de-a face cu un tabel “in alb” cel mai simplu mijloc pentru a-l popula este acela de a crea un fisier “txt” alocand cate un sir pentru fiecare dintre animalele de companie, si apoi incarcati continutul din fisier in tabel printr-o singura linie de comanda.

Puteti crea un fisier text “pet.txt” continand o inregistrare pe fiecare rand, avand diferitele valori separate prin spatii, si derulate in aceiasi ordine in care coloanele au fost listate in CREATE TABLE.

Pentru valorile lipsa (sex necunoscut, sau data decesului in cazul animalelor aflate inca in viata) puteti apela la optiunea NULL. Pentru reprezentarea acestora in fisierul text folositi \N. De exemplu, pentru Whistler o inregistrare va arata astfel:

nume	**posesor**	**specie**	**sex**	**data nasterii**	**data decesului**
Wistler	Gwen	bird	\N	1997-12-09	\N

Pentru a transcrie textul din fisierul “pet.txt” in tabel tastati comanda:

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Ori de cite ori vreti sa adaugati inregistrari optati pentru INSERT.Cel mai simplu mijloc este sa suplimentati informatiile pentru fiecare coloana, in aceeasi ordine in care acestea au fost listate in CREATE TABLE.Sa presupunem ca Diana primeste un nou hamster numit Pufball.

Noua inregistrare folosind utilitarul INSERT va arata astfel:

```
mysql> INSERT INTO pet
->VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Observati faptul ca insiruirea valorilor se face intre ghilimele.Cu acelasi utilitar INSERT
Puteti insera valorile lipsa.Nu optati pentru \N cum ati facut cu LOAD DATA.

Din acest exemplu ati putut observa ca este preferabil sa incarcati initial inregistrarea folosind utilitarul INSERT decit LOAD DATA.

3.3.4 Refacerea informatiilor dintr-un tabel

Optiunea SELECT este folosita pentru a “extrage” informatiile dintr-un tabel.In forma sa generala comanda arata astfel:

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy
```

“what-to select” indica ceea ce vreti sa vizualizati.Aceasta poate fi doar un numar de coloane sau toate.

“which-table” indica tabelul din care veti extrage informatia.

Otinea “WHERE” este facultativa.

“conditions-to satisfy” specifica conditiilre pe care liniile trebuie sa le indeplineasca pentru a putea fi “extrase”.

3.3.4.Selectarea tuturor informatiilor

Iata forma cea mai simpla de selectare a oricarui tabel:

```
mysql> SELECT * FROM pet;
+----------+--------+---------+------+------------+------------+
| name | owner | species | sex | birth | death |
+----------+--------+---------+------+------------+------------+
| Fluffy | Harold | cat | f | 1993-02-04 | NULL |
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
| Fang | Benny | dog | m | 1990-08-27 | NULL |
| Bowser | Diane | dog | m | 1998-08-31 | 1995-07-29 |
| Chirpy | Gwen | bird | f | 1998-09-11 | NULL |
| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |
| Slim | Benny | snake | m | 1996-04-29 | NULL |
| Puffball | Diane | hamster | f | 1999-03-30 | NULL |
+----------+--------+---------+------+------------+------------+
```

Aceasta modalitate este indicata in cazul in care vreti sa revedeti intregul tabel , de ex., dupa ce tocmai ati inregistrat setul initial de date.

Pot aparea , se intimpla, erori in fisierul Dvs. De exemplu, Bowser apare a fi nascut dupa data decesului sau consultati-va informatiile in original ,si descoperiti ca anul nasterii era de fapt 1989 in loc de 1998.

Exista cel putin doua solutii pentru a remedia greseala:

→Tastati EDIT pentru fisierul “pettxt” pentru a corecta eroarea, goliti tabelul si reincarcati-l folosind DELETE si LOAD DATA.

Oricum daca procedati astfel trebuie deasemenea sa reintroduceti inregistrarile pentru Puffball.

→Refaceti doar inregistrarea cu probleme printr-o comanda UPDATE.

```
mysql> SET AUTOCOMMIT=1; #
mysql> DELETE FROM pet;
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
mysql> UPDATE pet SET birth = "1989-08-31" WHERE name = "Bowser";
```

Cum am aratat este simplu sa extragem / selectam intregul tabel. Dar, in mod normal, nu veti avea nevoie sa faceti asta, si nici nu este indicat mai ales ca in timp acesta se extinde.

In schimb, va veti afla in situatia de a rezolva o singura problema, caz in care va trebui sa specificati cateva probleme implicate.

Sa urmam cateva comenzi de selectie a unor informatii despre animalele Dvs.

3.3.4.2 Selectarea unor linii din tabel

Puteti selecta doar anumite linii din intregul tabel. De exemplu, daca vreti sa verificati corectura pe care ati facut-o in privinta datei de nastere a lui Bowser, selectati inregistrarea astfel:

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
+--------+-------+---------+------+------------+------------+
| name | owner | species | sex | birth | death |
+--------+-------+---------+------+------------+------------+
| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |
+--------+-------+---------+------+------------+------------+
```

Iesirea confirma corectura facuta.

Datele sunt insensibile din punct de vederea al inregistrarii astfel incat puteti scrie numele: “bowser”, “BOWSER” etc. Rezultatul solicitarii ca fi acelasi. Puteti specifica conditiile pe oricare dintre coloane, nu doar pe aceea a numelor. De exemplu, daca vreti sa selectati animalele nascute dupa 1998 selectati coloana “data de nastere”:

```
mysql> SELECT * FROM pet WHERE birth >= "1998-1-1";
+----------+-------+---------+------+------------+-------+
```

```
| name | owner | species | sex | birth | death |
+----------+-------+---------+------+------------+-------+
| Chirpy | Gwen | bird | f | 1998-09-11 | NULL |
| Puffball | Diane | hamster | f | 1999-03-30 | NULL |
+----------+-------+---------+------+------------+-------+
```

Puteti de asemenea combina conditiile, de exemplu pentru a localiza femelele – caine:

```
mysql> SELECT * FROM pet WHERE species = "dog" AND sex = "f";
+-------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+-------+--------+---------+------+------------+-------+
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+-------+--------+---------+------+------------+-------+
```

Solicitarea anterioara foloseste operatorul logic AND. Exista si un operator OR:

```
mysql> SELECT * FROM pet WHERE species = "snake" OR species = "bird";
+----------+-------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+----------+-------+---------+------+------------+-------+
| Chirpy | Gwen | bird | f | 1998-09-11 | NULL |
| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |
| Slim | Benny | snake | m | 1996-04-29 | NULL |
+----------+-------+---------+------+------------+-------+
```

AND si OR pot fi la randul lor mixati. Daca asta doriti, e preferabil sa folositi parantezele pentru a indica modul in care vreti sa fie grupate conditiile.

```
mysql> SELECT * FROM pet WHERE (species = "cat" AND sex = "m")
-> OR (species = "dog" AND sex = "f");
+-------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+-------+--------+---------+------+------------+-------+
```

3.3.4.3 Selectarea anumitor coloane

Daca nu doriti vizualizarea tuturor liniilor din tabel precizati doar coloanele in care sunteti interesat, separate prin virgule.

De exemplu, daca vreti sa stiti datele de nastere ale animalelor detinute selectati coloanele numelui si respectiv datei de nastere.

```
mysql> SELECT name, birth FROM pet;
+----------+------------+
| name | birth |
+----------+------------+
| Fluffy | 1993-02-04 |
| Claws | 1994-03-17 |
| Buffy | 1989-05-13 |
| Fang | 1990-08-27 |
| Bowser | 1989-08-31 |
| Chirpy | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Daca vreti sa aflati posesorii fiecarui animal, tastati:

```
mysql> SELECT owner FROM pet;
+--------+
| owner |
+--------+
| Harold |
| Gwen |
| Harold |
| Benny |
| Diane |
| Gwen |
| Gwen |
| Benny |
| Diane |
+--------+
```

In orice caz, observati modalitatea incare comanda extrage pur si simplu campul posesorului din fiecare inregistrare si astfel veti vedea ca unele nume apar de mai multe ori. Pentru a mima finalul, extrageti fiecare inregistrare finala unica doar odata prin adaugarea cuvantului cheie DISTINCT:

```
mysql> SELECT DISTINCT owner FROM pet;
+--------+
| owner |
+--------+
| Benny |
| Diane |
| Gwen |
| Harold |
+--------+
```

Puteti folosi o clauza "**where"** sa realizati o combinare intre selectia randurilor si cea a coloanelor. De exemplu, selectati datele de nastere doar a cainilor si pisicilor:

```
mysql> SELECT name, species, birth FROM pet
-> WHERE species = "dog" OR species = "cat";
+--------+---------+------------+
| name | species | birth |
+--------+---------+------------+
| Fluffy | cat | 1993-02-04 |
| Claws | cat | 1994-03-17 |
| Buffy | dog | 1989-05-13 |
| Fang | dog | 1990-08-27 |
| Bowser | dog | 1989-08-31 |
+--------+---------+------------+
```

3.3.4.4 Sortarea liniilor

Trebuie sa fi observat in exemplele anterioare ca liniile rezultate nu sunt dispuse intr-o ordine anume.

Oricum, de cele mai multe ori este mai usor sa inregistrezi cand liniile sunt sortate dupa un anumit inteles. Astfel folositi clauza ORDER BY.

Iata datele de nastere dispuse cronologic:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
| name | birth |
+----------+------------+
| Buffy | 1989-05-13 |
| Bowser | 1989-08-31 |
| Fang | 1990-08-27 |
| Fluffy | 1993-02-04 |
| Claws | 1994-03-17 |
| Slim | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Pentru sortarea inversa (descrescaroare) adugati cuvantul cheie DESC la numele coloanei pe care o selectati:

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
+----------+------------+
| name | birth |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim | 1996-04-29 |
| Claws | 1994-03-17 |
| Fluffy | 1993-02-04 |
| Fang | 1990-08-27 |
| Bowser | 1989-08-31 |
| Buffy | 1989-05-13 |
+----------+------------+
```

Puteti sorta mai multe coloane consecutive. Exemplul dupa specie, apoi dupa data nasterii facand anumite precizari precum numele celui mai tanar animal:

```
mysql> SELECT name, species, birth FROM pet ORDER BY species, birth
DESC;
```

```
+----------+---------+------------+
| name | species | birth |
+----------+---------+------------+
| Chirpy | bird | 1998-09-11 |
| Whistler | bird | 1997-12-09 |
| Claws | cat | 1994-03-17 |
| Fluffy | cat | 1993-02-04 |
| Fang | dog | 1990-08-27 |
| Bowser | dog | 1989-08-31 |
| Buffy | dog | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim | snake | 1996-04-29 |
+----------+---------+------------+
```

Observati ca DESC se aplica doar coloanelor al calor nume a fost specificat. Celelalte raman neschimbate. Ex: valorile speciei sunt inca enumerate in ordine crescatoare.

3.3.4.5 Calculul datelor

Mysql asigura cateva facilitati pe care le puteti folosi pentru diverse calcule asupra datelor:

Ex: calculul varstelor sau extragerea unor parti eronate a datelor. Pentru a determina ce varsta are fiecare animal, calculati diferenta dintre data curenta si cea inregistrata apoi scadeti 1, daca data curenta este situata in calendar inaintea datelor de nastere.

```
mysql> SELECT name, birth, CURRENT_DATE,
-> (YEAR(CURRENT_DATE)-YEAR(birth))
-> - (RIGHT(CURRENT_DATE,5)<RIGHT(birth,5))
-> AS age
-> FROM pet;
+----------+------------+--------------+------+
| name | birth | CURRENT_DATE | age |
+----------+------------+--------------+------+
| Fluffy | 1993-02-04 | 2001-08-29 | 8 |
| Claws | 1994-03-17 | 2001-08-29 | 7 |
| Buffy | 1989-05-13 | 2001-08-29 | 12 |
| Fang | 1990-08-27 | 2001-08-29 | 11 |
| Bowser | 1989-08-31 | 2001-08-29 | 11 |
| Chirpy | 1998-09-11 | 2001-08-29 | 2 |
```

```
| Whistler | 1997-12-09 | 2001-08-29 | 3 |
| Slim | 1996-04-29 | 2001-08-29 | 5 |
| Puffball | 1999-03-30 | 2001-08-29 | 2 |
+----------+------------+-------------+------+
```

Aici YEAR() inregistreaza anul din data de nastere, iar RIGHT celelalte cinci caractere reprezentand luna, respectiv ziua. Rezultatul poate fi scanat cu mai multa usurinta daca liniile au fost inregistrate intr-o ordine prestabilita. Acest lucru se obtine adaugand clauza ORDER BY spre a sorta inregistrarile finale dupa nume. Pentru o inregistrare finala dupa varsta este de ajuns sa optati pentru o alta clauza ORDER BY.

```
mysql> SELECT name, birth, CURRENT_DATE,
-> (YEAR(CURRENT_DATE)-YEAR(birth))
-> - (RIGHT(CURRENT_DATE,5)<RIGHT(birth,5))
-> AS age
-> FROM pet ORDER BY name;
+----------+------------+-------------+------+
| name | birth | CURRENT_DATE | age |
+----------+------------+-------------+------+
| Bowser | 1989-08-31 | 2001-08-29 | 11 |.170 MySQL Technical Reference
for Version 4.1.0-alpha
| Buffy | 1989-05-13 | 2001-08-29 | 12 |
| Chirpy | 1998-09-11 | 2001-08-29 | 2 |
| Claws | 1994-03-17 | 2001-08-29 | 7 |
| Fang | 1990-08-27 | 2001-08-29 | 11 |
| Fluffy | 1993-02-04 | 2001-08-29 | 8 |
| Puffball | 1999-03-30 | 2001-08-29 | 2 |
| Slim | 1996-04-29 | 2001-08-29 | 5 |
| Whistler | 1997-12-09 | 2001-08-29 | 3 |
+----------+------------+-------------+------+

mysql> SELECT name, birth, CURRENT_DATE,
-> (YEAR(CURRENT_DATE)-YEAR(birth))
-> - (RIGHT(CURRENT_DATE,5)<RIGHT(birth,5))
-> AS age
-> FROM pet ORDER BY age;
+----------+------------+-------------+------+
| name | birth | CURRENT_DATE | age |
```

```
+----------+------------+-------------+------+
| Chirpy | 1998-09-11 | 2001-08-29 | 2 |
| Puffball | 1999-03-30 | 2001-08-29 | 2 |
| Whistler | 1997-12-09 | 2001-08-29 | 3 |
| Slim | 1996-04-29 | 2001-08-29 | 5 |
| Claws | 1994-03-17 | 2001-08-29 | 7 |
| Fluffy | 1993-02-04 | 2001-08-29 | 8 |
| Fang | 1990-08-27 | 2001-08-29 | 11 |
| Bowser | 1989-08-31 | 2001-08-29 | 11 |
| Buffy | 1989-05-13 | 2001-08-29 | 12 |
+----------+------------+-------------+------+
```

O formulare similara poate fi folosita pentru a determina varsta in momentul decesului. Care anume sunt animalele aflate in situatia data se observa simplu verificand daca valoarea “death” este sau nu NULL. Pentru cele care nu sunt faceti diferenta dintre data decesului si cea a nasterii.

```
mysql> SELECT name, birth, death,
-> (YEAR(death)-YEAR(birth)) - (RIGHT(death,5)<RIGHT(birth,5))
-> AS age
-> FROM pet WHERE death IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| name | birth | death | age |
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 | 5 |
+--------+------------+------------+------+
```

Dar, daca vreti sa aflati, de exemplu ce animale “isi sarbatoresc ziua” luna viitoare? Pentru astfel de tipuri de calcule, parametri ca anul sau ziua devin irelevanti; tot ce trebuie sa faceti este sa extrageti valorile referitoare la luna din coloana datei de nastere.

Mysql furnizeaza cateva posibilitati de astfel de selectari “partiale”: YEAR(), MONTH(), DAYOFMONTH(). In exemplul nostru luna MONTH() este optiunea potrivita.

Spre a veda concret cum functioneaza rulati o comanda simpla care include deopotriva valorile datelor de nastere (birth) respectiv ale lunii (month).

```
mysql> SELECT name, birth, MONTH(birth) FROM pet;
+----------+------------+--------------+
| name | birth | MONTH(birth) |
+----------+------------+--------------+
```

```
| Fluffy | 1993-02-04 | 2 |
| Claws | 1994-03-17 | 3 |
| Buffy | 1989-05-13 | 5 |
| Fang | 1990-08-27 | 8 |
| Bowser | 1989-08-31 | 8 |
| Chirpy | 1998-09-11 | 9 |
| Whistler | 1997-12-09 | 12 |
| Slim | 1996-04-29 | 4 |
| Puffball | 1999-03-30 | 3 |
+----------+------------+--------------+
```

Deasemenea este usoara cautarea animalelor care au ziua de nastere in luna ce urmeaza. Presupunem ca luna curenta este Aprilie. Atunci valoarea lunii este 4 si Dvs. cautati animale nascute in Mai (luna 5) astfel:

```
mysql> SELECT name, birth FROM pet WHERE MONTH(birth) = 5;
+-------+------------+
| name | birth |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

Desigur exista o mica complicatie daca luna curenta este Decembrie. Nu puteti adauga 1 la numarul lunii (12) si apoi sa cautati animale nascute in luna 13, deoarece nu exista o asemenea luna. In loc veti cauta animale nascute in Ianuarie (luna 1).
Puteti chiar sa scrieti comanda astfel incat ea va functiona indiferent de luna curenta. In acest fel nu trebuie sa folositi numarul aferent unei luni oarecare in comanda. DATE_ADD() va permite sa adaugati o luna la valoarea lui NOW(), apoi extrageti luna cu ajutorul functiei MONTH(), apoi rezultatul va fi luna in care sa cautati zilele de nastere:

```
mysql> SELECT name, birth FROM pet
-> WHERE MONTH(birth) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
```

O alta cale de a realiza aceiasi tema este sa adaugati 1 pentru a ajunge la luna urmatoare dupa cea curenta (folosind functia MOD() pentru a rotunji valoarea lunii la 0 daca ea este 12):

```
mysql> SELECT name, birth FROM pet
-> WHERE MONTH(birth) = MOD(MONTH(NOW()), 12) + 1;
```

Trebuie notat faptul ca functia MONTH() intoarce un numar intre 1 si 12. Iar MOD(ceva,12) intoarce un numar intre 0 si 11. Asadar incrementarea trebuie sa fie functia MOD(), altfel am trece de la Noiembrie (11) la Ianuarie (1).

Lucrul cu valori NULL

Valoarea NULL poate fi surprinzatoare pana va obisnuiti cu ea. Conceptual, NULL inseamna valoare lipsa sau necunoscuta si este tratata oarecum diferit decat alte valori. Pentru a testa valoarea NULL nu puteti folosi operatori aritmetici de comparatie cum ar fi =,< sau <>. Pentru a demonstra aceasta pentru Dvs. insiva, incercati urmatoarea comanda:

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
| NULL | NULL | NULL | NULL |
+----------+-----------+----------+----------+
```

Evident nu obtineti rezultate satisfacatoare din aceasta comparatie. Folositi operatorii IS NULL si IS NOT NULL in loc:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
| 0 | 1 |
+-----------+---------------+
```

Observati ca in Mysql, 0 sau NULL inseamna fals si orice altceva insemna adevarat. Valoarea standard pentru adevar dintr-o operatie booleana este 1.

Acest tratament special al lui NULL este motivul pentru are, in sectiunea precedenta, a fost necesara determinarea animalelor ce nu mai sunt in viata folosind death IS NOT NULL in loc de death <> NULL.

Doua valori NULL sunt considerate egale intr-un GROUP BY.

Atunci cand utilizati un GROUP BY, valorile NULL sunt prezentate primele cand utilizati GROUP BY … ASC si ultimele daca folositi GROUP BY … DESC.

Notati ca intre Mysql 4.0.2 – 4.0.10, valorile NULL erau sortate incorect, ele fiind primele indiferent de directia de sortare.

Cautare dupa caz (forma)

Mysql asigura stabdard sql dupa caz (forma) cat si o forma de cautare dupa caz bazate pe expresii similare celor folosite de catre utilitatile UNIX cum ar fi vi, grep si sed.

Cautare dupa caz sql va permite sa folositi '_' pentru a asocia un singur caracter oarecare si '%' pentru a asocia un numar arbitrar de caractere (inclusiv caracterele 0).

In Mysql, tipurile sql sunt insensibile la caz in modul standard. Cateva exemple sunt aratate aici. Observati ca Dvs. nu folositi = sau <> atunci cand utilizati tipuri sql; utilizati operatorii de comparatie LIKE sau NOT LIKE in loc.

Pentru a gasi nume care incep cu 'b':

```
mysql> SELECT * FROM pet WHERE name LIKE "b%";
+--------+--------+---------+------+------------+------------+
| name | owner | species | sex | birth | death |
+--------+--------+---------+------+------------+------------+
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

Pentru a gasi nume ce se termina cu 'fy':

```
mysql> SELECT * FROM pet WHERE name LIKE "%fy";
+--------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat | f | 1993-02-04 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+--------+--------+---------+------+------------+-------+
```

Pentru a gasi nume ce contin litera 'w':

```
mysql> SELECT * FROM pet WHERE name LIKE "%w%";
+----------+-------+---------+------+------------+------------+
| name | owner | species | sex | birth | death |
+----------+-------+---------+------+------------+------------+
```

```
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |
+----------+-------+---------+------+------------+------------+
```

Pentru a gasi nume ce contin exact cinci caractere, folositi caracterul tip '_':

```
mysql> SELECT * FROM pet WHERE name LIKE "_____";
+-------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+-------+--------+---------+------+------------+-------+
```

Alte tipuri de cautari dupa caz asigurate de Mysql folosesc expresii extinse (regulate). Aunci cand testati pentru acest tip de caz o asociere, folositi operatorii REGEXP so NOT REGEXP (sau RLIKE so NOT RLIKE, care sunt sinonimi).

Unele caracteristici ale expresiilor extinse sunt:

- '.' Asociaza un singur caracter oarecare.
- O clasa de caractere '[…]' asociaza orice caracter din interiorul parantezelor. De exemplu, '[abc]' asociaza caracterele 'a','b' sau 'c'. Pentru a asocia un sir de caractere, folositi semnul "-" (dash). '[a-z]' asociaza orice litera mica , iar '[0..9]' orice cifra.
- '*' asociaza zero sau mai multe instante ale lucruluicel preceda. De exemplu, 'x*' asociaza orice numar de 'x' caractere, '[0..9]' asociaza orice numar de cifre, iar '*' asociaza orice numar de oricare fel.
- Tipul este asociat daca apare oriunde in valoarea care este testata. Tipurile sql asociaza doar daca ele asociaza inteaga valoare.
- Pentru a ancora un tip astfel incat sa tebuiasca asociat cu inceputul sau finalul valorii testate, folositi '^' la inceput sau '$' la sfarsitul tipului.

Pentru a demonstra cum functioneaza expresiile extinse, comenzile LIKE aratate anterior sunt rescrise aici pentru a folosi REGEXP.

Pentru a gasi nume ce incep cu 'b', folositi '^' pentru a asocia inceputul numelui:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^b";
+--------+--------+---------+------+------------+------------+
| name | owner | species | sex | birth | death |
```

```
+--------+--------+---------+------+------------+------------+
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

Inainte de Mysql versiunea 3.23.4, REGEXP este sensibil la caz, si comanda de mai inainte nu va intoarce nici un rand. Pentru a asocia atat litere mici cat si mari, exemplu ‘b’, folositi aceasta comanda in loc:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^[bB]";
```

De la Mysql 3.23.4 incolo, pentru a forta o comparatie REGEXP sa fie sensibila la caz, folositi cuvantul cheie BINARY pentru a converti un sir intr-un sir binar. Urmatoarea comanda va asocia doar litera mica ‘b’ la inceputul unui nume:

```
mysql> SELECT * FROM pet WHERE name REGEXP BINARY "^b";
```

Pentru a gasi nume ce se termina cu ‘fy’, folositi ‘$’ pentru a asocia finalul unui nume:

```
mysql> SELECT * FROM pet WHERE name REGEXP "fy$";
+--------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat | f | 1993-02-04 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+--------+--------+---------+------+------------+-------+
```

Pentru a gasi nume ce contin litera mica sau mare ‘w’, folositi acesta comanda:

```
mysql> SELECT * FROM pet WHERE name REGEXP "w";
+----------+-------+---------+------+------------+------------+
| name | owner | species | sex | birth | death |
+----------+-------+---------+------+------------+------------+
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Bowser | Diane | dog | m | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen | bird | NULL | 1997-12-09 | NULL |
+----------+-------+---------+------+------------+------------+
```

Deoarece o expresie regulara tip asociaza daca ea apare oriunde in valoare, nu este necesar ca in comanda precedenta sa fie introdusi specificatori de nici o parte a tipului pentru a obtine o asociere cu intreaga valoare asa cum ar fi fost daca ati fi folosit un tip sql.

Pentru a gasi un nume ce contine exact cinci caractere, folositi '^' si '$' ca sa asociati inceputul si sfarsitul numelui, ci cinci instante de '.' intre:

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.....$";
+-------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+-------+--------+---------+------+------------+-------+
```

Ati putea deasemenea scrie comanda anterioara folosind '{n}', operatorul "repetat-n-ori".

```
mysql> SELECT * FROM pet WHERE name REGEXP "^.{5}$";
+-------+--------+---------+------+------------+-------+
| name | owner | species | sex | birth | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen | cat | m | 1994-03-17 | NULL |
| Buffy | Harold | dog | f | 1989-05-13 | NULL |
+-------+--------+---------+------+------------+-------+
```

Numararea randurilor

Bazele de date sunt adesea folosite pentrua raspunde intrebarii: "Cat de des apare un anumit tip de data intr-un tabel?" De exemplu, daca doriti sa aflati cate animale aveti, sau cate animale are fiecare proprietar, sau poate doriti sa efectuati diferite tipuri de recensamant la animalele Dvs.

Calcularea numarului total de animale pe care le aveti este acelasi lucru ca si cum ati intreba: "Cate randuri sunt intr-un tabel de animale?" deoarece exista o singura inregistrare pentru fiecare animal. Functia COUNT() numara numarul de rezultate care nu sunt NULL, astfel incat comanda de a numara animalele Dvs. Va arata dupa cum urmeaza:

```
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
```

```
| 9 |
+----------+
```

Mai devreme ati obtinut numele persoanelor ce detin animale. Puteti folosi COUNT() daca doriti sa aflati cate animale are fiecare proprietar in parte:

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
+--------+----------+
| owner | COUNT(*) |
+--------+----------+
| Benny | 2 |
| Diane | 2 |
| Gwen | 3 |
| Harold | 2 |
+--------+----------+
```

Observati cum este utilizat GROUP BY pentru a grupa la un loc toate inregistrarile pentru fiecare proprietar. Fara el ati obtine doar un mesaj de eroare:

```
mysql> SELECT owner, COUNT(owner) FROM pet;
ERROR 1140 at line 1: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

COUNT() si GROUP BY sunt folositoare pentru a va caracteriza datele in diferite moduri. Urmatoarele exemple arata diferite cai de a efectua operatii de recensamant la animalele Dvs.

Numarul de animale in functiede specie:

```
mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
+---------+----------+
| species | COUNT(*) |
+---------+----------+
| bird | 2 |
| cat | 2 |
| dog | 3 |
| hamster | 1 |
| snake | 1 |
+---------+----------+
```

Numarul de animale in functie de sex:

```
mysql> SELECT sex, COUNT(*) FROM pet GROUP BY sex;
+------+----------+
| sex | COUNT(*) |
+------+----------+
| NULL | 1 |
| f | 4 |
| m | 4 |
+------+----------+
```

(La acest rezultat, NULL indica sex necunoscut.)

Numarul de animale in combinatie de specie si sex:

```
mysql> SELECT species, sex, COUNT(*) FROM pet GROUP BY species, sex;
+---------+------+----------+
| species | sex | COUNT(*) |
+---------+------+----------+
| bird | NULL | 1 |
| bird | f | 1 |
| cat | f | 1 |
| cat | m | 1 |
| dog | f | 1 |
| dog | m | 2 |
| hamster | f | 1 |
| snake | m | 1 |
+---------+------+----------+
```

Nu este nevoie sa extrageti intregul tabel atunci cand folositi COUNT(). De exemplu, comanda anterioara, cand este efectuata pe caini si pisici, arata astfel:

```
mysql> SELECT species, sex, COUNT(*) FROM pet
-> WHERE species = "dog" OR species = "cat"
-> GROUP BY species, sex;
+---------+------+----------+
| species | sex | COUNT(*) |
+---------+------+----------+
| cat | f | 1 |
```

```
| cat | m | 1 |
| dog | f | 1 |
| dog | m | 2 |
+---------+------+----------+
```

Sau, daca doriti doar numarul de animale cu sexul cunoscut:

```
mysql> SELECT species, sex, COUNT(*) FROM pet
-> WHERE sex IS NOT NULL
-> GROUP BY species, sex;
+---------+------+----------+
| species | sex | COUNT(*) |
+---------+------+----------+
| bird | f | 1 |
| cat | f | 1 |
| cat | m | 1 |
| dog | f | 1 |
| dog | m | 2 |
| hamster | f | 1 |
| snake | m | 1 |
+---------+------+----------+
```

3.3.4.9 Folosirea mai multor tabele

Tabelul “pct” va tine evidenta animalelor de companie pe care le aveti. Daca vreti sa inregistrati si alte ‘informatii’ despre ele, evenimente importante precum nasterea puilor sau vizitele la veterinar, veti avea nevoie de un alt tabel.

Cum ar trebui sa arate? Ar trebui sa includa:

- Numele animalului spre a sti unde plasati evenimentul.
- Data, spre a sti cand s-a petrecut.
- Un pasaj - camp in care sa-l descrieti.
- Un camp pe tipuri de evenimente, spre a le ordona.

Toate acestea fiind opuse, optiunea CREATE TABLE pentru tabelul evenimentelor ar trebui sa arate ca asa:

```
mysql> CREATE TABLE event (name VARCHAR(20), date DATE,
-> type VARCHAR(15), remark VARCHAR(255));
```

Dat fiind tabelul initial “pet” va fi mai usor sa introduceti datele initiale in cel de-al doilea creaind un text a carui date vor fi delimitate prin cate un tab:

```
name date type remark
Fluffy 1995-05-15 litter 4 kittens, 3 female, 1 male
Buffy 1993-06-23 litter 5 puppies, 2 female, 3 male
Buffy 1994-06-19 litter 3 puppies, 3 female
Chirpy 1999-03-21 vet ciocul indreptat
Slim 1997-08-03 vet coasta rupta
Bowser 1991-10-12 kennel
Fang 1991-10-12 kennel
Fang 1998-08-28 birthday o noua jucarie de plastic
Claws 1998-03-17 birthday o noua zgarda
Whistler 1998-12-09 birthday zi de nastere
```

Incarcati inregistrarile astfel:

```
mysql> LOAD DATA LOCAL INFILE "event.txt" INTO TABLE event;
```

Orientandu-va dupa cunostintele acumulate la tabelul anterior veti reusi sa efectuati diverse operatiuni si modificari in tabelul evenimentelor. Principiile sunt aceleasi.

Dar in cazul in care datele introduse sunt insuficiente pentru a raspunde anumitor intrebari, ce ar trebui sa faceti?

Sa presupunem, ce varsta avea fiecare animal in momentul in care a avut nou-nascuti Tabelul evenimentelor indica datele in care acest lucru sa petrecut, dar pentru a calcula varsta “mamei”, aveti, nevoie si de data de nastere.

Pentru ca aceasta e stocata in tabelul “pet” aveti nevoie de ambele tabele in acelasi timp, pentru urmatoarea comanda:

```
mysql> SELECT pet.name,
-> (TO_DAYS(date) - TO_DAYS(birth))/365 AS age,
-> remark
-> FROM pet, event
-> WHERE pet.name = event.name AND type = "litter";.178 MySQL Technical
Reference for Version 4.1.0-alpha
+--------+------+-----------------------------+
| name | age | remark |
+--------+------+-----------------------------+
```

```
| Fluffy | 2.27 | 4 kittens, 3 female, 1 male |
| Buffy | 4.12 | 5 puppies, 2 female, 3 male |
| Buffy | 5.10 | 3 puppies, 3 female |
+--------+------+-----------------------------+
```

Trebuie subliniate cateva lucruri in legatura cu aceasta comanda:

- Clauza FROM ruleaza ambele tabele pentru ca cererea va extrage date din amandoua;
- Cand sunt combinate in formatii din mai multe tabele, trebuie specificat modul in care inregistrarile dintr-unul pot fi cuplate cu cele din al doilea. Acest lucru ar trebui sa fie usor , pentru ca , nu-i asa, ambele tabele poseda o coloana a numelui. Si in cazul nostru, se vor cupla datele din doua tabele pe baza valorilor numelui, prin caluza WHERE.
- Tocmai pentru ca coloana numelui se afla in ambele tabele, cand o folositi trebuie sa specificati tabelul.

Nu intotdeauna aveti nevoie de doua tabele diferite pentru a face combinari. Orice tabel poate fi combinat cu el insusi.

De exemplu, pentru a gasi posibile perechi printre animalele dumneavoastra puteti combina orientativ femelele si masculii inscrisi in acelasi tabel, de exemplu dupa specie:

```
mysql> SELECT p1.name, p1.sex, p2.name, p2.sex, p1.species
-> FROM pet AS p1, pet AS p2
-> WHERE p1.species = p2.species AND p1.sex = "f" AND p2.sex = "m";
+--------+------+--------+------+---------+
| name | sex | name | sex | species |
+--------+------+--------+------+---------+
| Fluffy | f | Claws | m | cat |
| Buffy | f | Fang | m | dog |
| Buffy | f | Bowser | m | dog |
+--------+------+--------+------+---------+
```

3.4 Obtinerea informatiilor despre bazele de date si tabele

Ce faceti daca uitati numele unei baze de date sau unui tabel, sau care e structura acestuia?

MySQL rezolva aceasta problema prin cateva tipuri de comenzi avand drept scop obtinerea informatiilor despre baza si tabelele componente.

V-ati familiarizat deja cu SHOW DATABASES, care sint bazele de date manipulate pe server.

Pentru a afla ce baza de date ruleaza intr-un anume moment, optati pentru functia DATABASE():

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie |
+------------+
```

Daca n-ati selectat nici o baza de date inca, rezultatul va fi nul.

Pentru a afla ce tabele contine baza de date selectata folositi comanda:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| event |
| pet |
+---------------------+
```

Daca vreti informatii despre structura tabelului, e absolut necesara comanda DESCRIBE, va va furniza informatii de spre fiecare dintre coloanele tabelului:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name | varchar(20) | YES | | NULL | |
| owner | varchar(20) | YES | | NULL | |
| species | varchar(20) | YES | | NULL | |
| sex | char(1) | YES | | NULL | |
| birth | date | YES | | NULL | |
| death | date | YES | | NULL | |
+---------+-------------+------+-----+---------+-------+
```

Field - indica numele coloanei; Type - tipul datelor;

Null - daca coloana include si valori nule, Key - daca coloana este indexata si Default valorile prestabilite in sistemul de operare.

Daca exista indici in tabel, SHOW INDEX FROM tbl-name va procura informatii despre ele.

3.5 Exemple de comenzi

Iata cateva exemple despre cum puteti rezolva cateva probleme curente prin MySQL.

Cateva dintre exemple folosesc tabelul shop pentru a inregistra pretul fiecarui articol (item) in cazul anumitor comerciant (dealers). Sa presupunem ca fiecare comerciant are cate un singur pret fix pentru fiecare produs. Atunci vom considera ca (article, dealers) este cheie primara pentru inregistrari:

Lansati linia de comanda mysql si selectati o baza de date:

Puteti crea un tabel exemplu astfel:

```
CREATE TABLE shop (
article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
dealer CHAR(20) DEFAULT '' NOT NULL,
price DOUBLE(16,2) DEFAULT '0.00' NOT NULL,
PRIMARY KEY(article, dealer));
INSERT INTO shop VALUES
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
(3,'D',1.25),(4,'D',19.95);
```

Acum totul este OK. Baza exemple arata astfel:

```
mysql> SELECT * FROM shop;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
| 0001 | A | 3.45 |
| 0001 | B | 3.99 |
| 0002 | A | 10.99 |
| 0003 | B | 1.45 |
| 0003 | C | 1.69 |
| 0003 | D | 1.25 |
| 0004 | D | 19.95 |
+---------+--------+-------+
```

3.5.1 Valoarea maxima pentru o coloana

"Care este cel mai mare numar?"

```
SELECT MAX(article) AS article FROM shop
+---------+
| article |
+---------+
| 4 |
+---------+
```

3.5.2. Linia continand maxima unei anume coloane.

"Gasiti numarul, dealerul, si pretul celui mai scump articol."

In SQL - 99 (si myqsl vers. 41) se rezolva repede printr-o subcomanda:

```
SELECT article, dealer, price
FROM shop
WHERE price=(SELECT MAX(price) FROM shop)
```

In versiunea MySQL de pana la 4.1 veti obtine rezultatul in 2 etape:

1. Obtineti valoarea maxima a pretului cu SELECT.
2. Folosind valoarea rezultata o includeti in:

```
SELECT article, dealer, price
FROM shop
WHERE price=19.95
```

O alta posibila solutie consta in sortarea tuturor randurilor (liniilor) descrescator dupa valoarea pretului si selectati doar prima folosind clauza LIMIT:

```
SELECT article, dealer, price
FROM shop
ORDER BY price DESC
LIMIT 1
```

NOTA: Daca exista mai multe valori maxime (acelasi pret) programul va indica doar una dintre ele.

3.5.3 Maxima coloanei per grup

"Care este valoarea maxima a pretului dupa articol?"

```
SELECT article, MAX(price) AS price
FROM shop
GROUP BY article
+---------+-------+
| article | price |
+---------+-------+
| 0001 | 3.99 |
| 0002 | 10.99 |
| 0003 | 1.69 |
| 0004 | 19.95 |
+---------+-------+
```

3.5.4 Un rand cu un camp care cauta automat maxinmul dintr-un grup anume

"Petru fiecare articol, gasiti distribuitorul cu pretul cel mai ridicat."

```
SELECT article, dealer, price
FROM shop s1
WHERE price=(SELECT MAX(s2.price)
FROM shop s2
WHERE s1.article = s2.article);
```

In versiunea MySQL dinaintea lui 4.1 este bine sa urmati citiva pasi:

1. Obtineti listele (pret max / articol).
2. Pentru oricare articol preluati linia corespunzatoare care are stocata pretul maxim.

Puteti face asta simplu cu un tabel provizoriu:

```
CREATE TEMPORARY TABLE tmp (
article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
price DOUBLE(16,2) DEFAULT '0.00' NOT NULL);
LOCK TABLES shop read;
INSERT INTO tmp SELECT article, MAX(price) FROM shop GROUP BY article;
SELECT shop.article, dealer, shop.price FROM shop, tmp
WHERE shop.article=tmp.article AND shop.price=tmp.price.
```

```
UNLOCK TABLES;
DROP TABLE tmp;
```

Daca nu folositi un tabel TEMPORARY, trebuie sa blocati tabelul ‘tmp’.

“Poate fi realizat cu o singura comanda?”

Da, dar doar folosind un truc eficient pe care I-l numesc trucul “MAX-CONCAT”:

```
SELECT article,
SUBSTRING( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 7) AS dealer,
0.00+LEFT( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 6) AS price
FROM shop
GROUP BY article;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
| 0001 | B | 3.99 |
| 0002 | A | 10.99 |
| 0003 | C | 1.69 |
| 0004 | D | 19.95 |
+---------+--------+-------+
```

Ultimul exemplu poate fi, dsigur, facut, inca si mai eficient prin divizarea coloanei in client.

3.5.5 Folosirea variabilelor utilizatorului

Puteti folosi variabilele MySQL spre a va reaminti rezultatele fara a trebui sa le stocati in variabile temporare in client.

De exemplu, pentru a gasi articolele cu cel mai mare sau cel mai mic pret puteti face astfel:

```
mysql> SELECT @min_price:=MIN(price),@max_price:=MAX(price) FROM shop;
mysql> SELECT * FROM shop WHERE price=@min_price OR price=@max_price;
+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
| 0003 | D | 1.25 |
| 0004 | D | 19.95 |
+---------+--------+-------+
```

3.5.6 Folosirea cheilor straine

In cazul MySQL 3.23.44 si mai departe, tabelele InnDB suporta verificarea contrangerilor cheilor straine. De fapt nu aveti nevoie de chei straine pentru a uni doua tabele. Singurul lucru pe care MySQL nu il face in mod curent (in alte tipuri de tabele inafara de InnoDB), este acela de a verifica, “CHECK”, daca cheile pe care le folositi exista in tabel si nu sterge automat randuri din tabel cu chei straine predefinite. Daca folositi cheile Dvs. in mod normal atunci totul va funtiona perfect:

```
CREATE TABLE person (
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
name CHAR(60) NOT NULL,
PRIMARY KEY (id)
);
CREATE TABLE shirt (
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
PRIMARY KEY (id)
);
INSERT INTO person VALUES (NULL, 'Antonio Paz');
INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());
INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');
INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'red', LAST_INSERT_ID()),
(NULL, 'dress', 'blue', LAST_INSERT_ID()),
(NULL, 't-shirt', 'white', LAST_INSERT_ID());
SELECT * FROM person;
+----+---------------------+
| id | name |
+----+---------------------+
| 1 | Antonio Paz |
| 2 | Lilliana Angelovska |
+----+---------------------+
```

```
SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style | color | owner |
+----+---------+--------+-------+
| 1 | polo | blue | 1 |
| 2 | dress | white | 1 |
| 3 | t-shirt | blue | 1 |
| 4 | dress | orange | 2 |
| 5 | polo | red | 2 |
| 6 | dress | blue | 2 |
| 7 | t-shirt | white | 2 |
+----+---------+--------+-------+
SELECT s.* FROM person p, shirt s
WHERE p.name LIKE 'Lilliana%'
AND s.owner = p.id
AND s.color <> 'white';
+----+-------+--------+-------+
| id | style | color | owner |
+----+-------+--------+-------+
| 4 | dress | orange | 2 |
| 5 | polo | red | 2 |
| 6 | dress | blue | 2 |
+----+-------+--------+-------+
```

3.5.7 Cautarea dupa doua chei

MySQL nu este inca optimizat pentru cautarea dupa doua chei combinate in functie de OR (cautarea dupa o singura cheie cu diferite parti OR este in schimb funtionala):

```
SELECT field1_index, field2_index FROM test_table WHERE field1_index =
'1' OR field2_index = '1'
```

Explicatia ar fi aceea ca nu am avut inca timp inca timp sa punem la punct o cale plauzibila pentru a realiza acest lucru (varianta AND in schimb a fost finalizata si funtioneaza excelent).

Pentru moment puteti rezolva problemele de acest tip folosind un tabel TEMPORARY. Acest gen de optimizare este preferabila si in cazul in care ati folosit comenzi complicate unde serverul SQL a reusit optimizarile dar nu in ordinea lor initiala.

```
CREATE TEMPORARY TABLE tmp
SELECT field1_index, field2_index FROM test_table WHERE field1_index =
'1';
INSERT INTO tmp
SELECT field1_index, field2_index FROM test_table WHERE field2_index =
'1';
SELECT * from tmp;
DROP TABLE tmp;
```

Urmatorul mijloc prin care puteti rezolva probleme similare este de fapt o reuniune a doua comenzi.

3.5.8 Numerotarea accesarilor zilnice

In cele ce urmeaza se creioneaza despre cum puteti folosi grupul principal de functii pentru a calcula frecventa zilelor din luna in care un utilizator acceseaza o pagina Web:

```
CREATE TABLE t1 (year YEAR(4), month INT(2) UNSIGNED ZEROFILL,
day INT(2) UNSIGNED ZEROFILL);
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),
(2000,2,23),(2000,2,23);.Chapter 3: Tutorial Introduction 185
SELECT year,month,BIT_COUNT(BIT_OR(1<<day)) AS days FROM t1
GROUP BY year,month;
```

Care intoarce:

```
+------+-------+------+
| year | month | days |
+------+-------+------+
| 2000 | 01 | 3 |
| 2000 | 02 | 2 |
+------+-------+------+
```

3.5.9 Folosirea AUTO_INCREMENT

Atributul AUTO_INCREMENT poate fi folosit pentru a genera o identitate unica noilor linii:

```
CREATE TABLE animals (
id MEDIUMINT NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (id)
);
INSERT INTO animals (name) VALUES ("dog"),("cat"),("penguin"),
("lax"),("whale");
SELECT * FROM animals;
```

Ce intoarce:

```
+----+---------+
| id | name |
+----+---------+
| 1 | dog |
| 2 | cat |
| 3 | penguin |
| 4 | lax |
| 5 | whale |
+----+---------+
```

Puteti recupera cheia AUTO_INCREMENT pe care ati folosit-o prin functia LAST_INSERT_ID() SQL sau prin functia mysql_insert_id() API. Nota: pentru inserarea multipla (mai mult de doua linii functia LAST_INSERT_ID()/mysql_insert_id() va recupera cheia autoincrement din prima linie inserata. Acest lucru va permite inserarilor multiple sa fie reproduse si pe alte servere.

Pentru MyISAM si DB puteti specifica AUTO_INCREMENT pe coloana secundara printr-o cheie pe mai multe chei.

In acest caz valoarea generata pentru auto incrementarea coloanei este calculata ca MAX(auto_increment_column)+1) WHERE prefix=given-prefix. Acest calcul var putea fi de folos in caz nca doriti introducerea unor date in grupuri prestabilite:

```
CREATE TABLE animals (
grp ENUM('fish','mammal','bird') NOT NULL,
id MEDIUMINT NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (grp,id)
);
INSERT INTO animals (grp,name) VALUES("mammal","dog"),("mammal","cat"),
```

```
("bird","penguin"),("fish","lax"),("mammal","whale");
SELECT * FROM animals ORDER BY grp,id;

+--------+----+---------+
| grp | id | name |
+--------+----+---------+
| fish | 1 | lax |
| mammal | 1 | dog |
| mammal | 2 | cat |
| mammal | 3 | whale |
| bird | 1 | penguin |
+--------+----+---------+
```

Observati ca in acest caz valoarea AUTO_INCREMENT va fi reactivata daca renuntati la linia cu valoarea maxima AUTO_INCREMENT din grup.

3.6 Folosirea MySQL in Batch mode (grupat)

In sectiunile anterioare ati folosit programul MySQL interactiv pentru a introduce comenzi si a vizualiza rezultatele.

Puteti deasemenea rula MySQL in batch mode. Pentru a face asta trebuie sa introduceti comenzile pe care le doriti intr-un fisier, si comandati programului sa le citeasca.

```
shell> mysql < batch-file
```

Daca rulati MySQL-ul folosind ferestrele si aveti cateva caractere speciale in fisier, care va cauzeaza probleme puteti tasta:

```
dos> mysql -e "source batch-file"
```

Daca trebuie sa speficati parametri de conectare la linia de comanda, aceasta din urma ar trebui sa arate cam asa:

```
shell> mysql -h host -u user -p < batch-file
Enter password: ********
```

Cand folositi MySQL-ul sub aceasta forma creati un fisier document si finalizati ulterior textul, daca vreti ca acesta sa continue chiar daca exista erori trebuie sa folositi optiunea linie de comanda --force

De ce sa folositi un fisier text? Iata cateva motive.

- Daca rulati o comanda in mod repetat (zilnic, saptamanal) construiti un fisier document si veti fi scutiti de retastarea comenzii de fiecare data.
- Puteti crea noi comenzi similare pornind de la cele deja existente copiind si editand fisiere text.
- Modul batch poste servi deasemenea momentul derularii unei comenzi, in special a celor multiple sau a secventelor de comenzi. Daca faceti vreo greseala, nu va trebui sa retastati totul. Doar prelucrati textul sa corecteze eroarea si comandati apoi programului sa o execute.
- Daca o comanda conduce la mai multe rezultate le puteti rula, folosind un program de paginare.

```
shell> mysql < batch-file | more
```

Puteti consemna rezultatul intr-un fisier pentru a putea fi procesat ulterior:

```
shell> mysql < batch-file > mysql.out
```

Puteti distribui documentele si altor persoane.

- Unele situatii nu permit modul interactiv. De exemplu cand rulati o comnda dintr-un **cron job**.

Rezultatul este diferit cand rulati MySQL in batch mode decat in cel interactiv, de exemplu, rezultatul din SELECT DISTINCT species FROM pet arata astfel in mediul interactiv:

```
+---------+
| species |
+---------+
| bird |
| cat |
| dog |
| hamster |
| snake |
+---------+
```

Dar, astfel cand rulati in batch mode:

```
species
bird
cat
dog
hamster
snake
```

Daca vreti sa treceti la modul interactiv in cadrul batch mode folositi **mysql –t.**

Pentru ca, comenzile pe care le executati sa trimita un ecou folositi **mysql –vvv.**

Puteti de asemenea folosi fisierul text pentru linia de comanda pornind de la comnda initiala:

```
mysql> source filename;
```

3.7 Comenzi din proiecte gemene

La Analytikerna si Lentus, am activat sistemele si campurile pentru un proiect amplu de cercetare.

Acest proiect este de fapt o colaborare intre Institute of Environmental Medicine at Karolinska Institutet Stockholm si Section on Clinical Research in Aging
and Psychology at the University din Southern California.

Proiectele implica un proces de monitorizare in cadrul caruia gemenii din Suedia sunt intervievati telefonic. Cei care indeplinersc anumite criterii trec intr-o faza superioara a programului. Aici cei care vor sa participe sunt pusi sub observatia unei echipe de medici. Consultatiile include examene neuro-fiziologice, teste de laborator, encefalograme, evidente psiholigice, culegandu-se totodata despre istoricul familiei.

Mai multe informatii despre studiile asuprea gemenilor le puteti gasi la: http://www.imm.ki.se/TWIN/TWINUKW.HTM

Ultima parte a proiectului este organizata cu ajutorul unei interfete web scrisa in Perl si Mysql.

In fiecare noapte toate datele sunt stocate intr-o baza de date mysql.

3.7.1 Localizarea gemenilor nedistribuiti

Urmatoarea comanda este folosita spre a-I selecta pe cei ce urmeaza a intra in faza superioara a proiectului:

```
SELECT
CONCAT(p1.id, p1.tvab) + 0 AS tvid,
```

```
CONCAT(p1.christian_name, " ", p1.surname) AS Name,
p1.postal_code AS Code,
p1.city AS City,
pg.abrev AS Area,
IF(td.participation = "Aborted", "A", " ") AS A,
p1.dead AS dead1,
l.event AS event1,
td.suspect AS tsuspect1,
id.suspect AS isuspect1,
td.severe AS tsevere1,
id.severe AS isevere1,
p2.dead AS dead2,
l2.event AS event2,
h2.nurse AS nurse2,
h2.doctor AS doctor2,
td2.suspect AS tsuspect2,
id2.suspect AS isuspect2,
td2.severe AS tsevere2,
id2.severe AS isevere2,
l.finish_date
FROM
twin_project AS tp
/* For Twin 1 */
LEFT JOIN twin_data AS td ON tp.id = td.id
AND tp.tvab = td.tvab
LEFT JOIN informant_data AS id ON tp.id = id.id
AND tp.tvab = id.tvab
LEFT JOIN harmony AS h ON tp.id = h.id
AND tp.tvab = h.tvab
LEFT JOIN lentus AS l ON tp.id = l.id
AND tp.tvab = l.tvab
/* For Twin 2 */
LEFT JOIN twin_data AS td2 ON p2.id = td2.id
AND p2.tvab = td2.tvab
LEFT JOIN informant_data AS id2 ON p2.id = id2.id
AND p2.tvab = id2.tvab
LEFT JOIN harmony AS h2 ON p2.id = h2.id
AND p2.tvab = h2.tvab
```

```
LEFT JOIN lentus AS l2 ON p2.id = l2.id
AND p2.tvab = l2.tvab,
person_data AS p1,
person_data AS p2,
postal_groups AS pg
WHERE
/* p1 gets main twin and p2 gets his/her twin. */
/* ptvab is a field inverted from tvab */
p1.id = tp.id AND p1.tvab = tp.tvab AND
p2.id = p1.id AND p2.ptvab = p1.tvab AND
/* Just the sceening survey */
tp.survey_no = 5 AND
/* Skip if partner died before 65 but allow emigration (dead=9) */
(p2.dead = 0 OR p2.dead = 9 OR
(p2.dead = 1 AND
(p2.death_date = 0 OR
(((TO_DAYS(p2.death_date) - TO_DAYS(p2.birthday)) / 365)
>= 65))))
AND
(
/* Twin is suspect */
(td.future_contact = 'Yes' AND td.suspect = 2) OR
/* Twin is suspect - Informant is Blessed */
(td.future_contact = 'Yes' AND td.suspect = 1
AND id.suspect = 1) OR
/* No twin - Informant is Blessed */
(ISNULL(td.suspect) AND id.suspect = 1
AND id.future_contact = 'Yes') OR
/* Twin broken off - Informant is Blessed */
(td.participation = 'Aborted'
AND id.suspect = 1 AND id.future_contact = 'Yes') OR
/* Twin broken off - No inform - Have partner */
(td.participation = 'Aborted' AND ISNULL(id.suspect)
AND p2.dead = 0))
AND
l.event = 'Finished'
/* Get at area code */
AND SUBSTRING(p1.postal_code, 1, 2) = pg.code
```

```
/* Not already distributed */
AND (h.nurse IS NULL OR h.nurse=00 OR h.doctor=00)
/* Has not refused or been aborted */
AND NOT (h.status = 'Refused' OR h.status = 'Aborted'
OR h.status = 'Died' OR h.status = 'Other')
ORDER BY
tvid;
```

Cateva explicatii:

```
CONCAT(p1.id, p1.tvab) + 0 AS tvid
```

Dorim sortarea cheilor id si tvab in ordine numerica. Adugand 0 rezultatului determina Mysql-ul sa recunoasca rezultatul drept numar.

coloana id	Aceasta identifica o pereche de gemeni. Aceasta este o cheie in toate tabelele.
coloana tvab	Aceasta este un geaman intr-o pereche. Are o valoare de 1 sau 2.
column ptvab	Aceasta este inversa lui tvab. Atunci cand tvab este 1 aceasta este2, si viceversa. Exista pentru a economisi timpul de redactare si pentru a conferi Mysql-ului posibilitatea de a optimiza aceasta comanda.

Aceast comanda demonstreaza, printre altele, cum sa cautati un tabel din acelasi tabel folosind o combinare (p1 sau p2). In exemplu, aceasta est folosita pentru a verifica atunci cand partenerul unui geaman na murit inaintea varstei de 65. Daca este asa, atunci randul nu este indicat in rezultat.

Tot ce este mai exista toate tabele cu o informatie de tip gemeni. Avem o cheie pentru id, tvab (toate tabelurile) cat si id, ptvab (person_data) pentre a face comenzile mai rapide.

Pe calculatorul nostru (UN UltraSPARC 200Mhz) aceasta comanda intoarce 150-200 randuri si dureaza mai putin de o secunda.

Numarul curent de inregistrari din tabelele folosite mai sus:

```
Table Rows
person_data 71074
lentus 5291
twin_project 5286
twin_data 2012
informant_data 663
harmony 381
postal_groups 100
```

3.7.2 Vizualizarea unui tabel dupa “perechi de gemeni”

Fiecare intervievare se termina cu un cod ‘event’.

Comanda aratata aici este folosita pentru a afisa un tabel cu privire la perechile de gemeni combinate in funtie de ‘event’.

Aceasta indica din cate perechi ambii gemeni au finalizat programul si in cate perechi un geaman a terminat iar celalalt a fost refuzat, s.a.m.d.

```
SELECT
t1.event,
t2.event,
COUNT(*)
FROM
lentus AS t1,
lentus AS t2,
twin_project AS tp
WHERE
/* We are looking at one pair at a time */
t1.id = tp.id
AND t1.tvab=tp.tvab
AND t1.id = t2.id
/* Just the sceening survey */
AND tp.survey_no = 5
/* This makes each pair only appear once */
AND t1.tvab='1' AND t2.tvab='2'
GROUP BY
t1.event, t2.event;
```

3.8 Utilizarea MySQL cu Apache

Exista programe ce va permit sa autentificati utilizatorii dintr-o baza de date mysql si sa va permita logarea fisierelor Dvs. intr-un tabel mysql.

Puteti modifica formatul de logare Apache pentru a fi citit cu usurinta de mysql modificand urmatoarea linie in fisierul de configuratie Apache:

```
LogFormat \
"\"%h\",%{%Y%m%d%H%M%S}t,%>s,\"%b\",\"%{Content-Type}o\", \
\"%U\",\"%{Referer}i\",\"%{User-Agent}i\""
```

In MySQL puteti face acelasi lucru dupa cum urmeaza:

```
LOAD DATA INFILE '/local/access_log' INTO TABLE table_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```