2. INSTRUCTIUNI SQL DE ADMINISTRARE A STRUCTURII BAZELOR DE DATE

Cuprins

2.1. Operatii cu baze de date

2.1.1. Crearea unei baze de date

Un server poate gazdui una sau mai multe baze de date, fiecare continand propriul sau set de tabele. Instructiunea SQL folosita pentru crearea unei baze de date este CREATE DATABASE:

```
mysql> CREATE DATABASE magazin;
Query OK, 1 row affected (0.12 sec)
```

In MySQL exista si cuvantul cheie SCHEMA care este sinonim cu DATABASE, astfel ca instructiunea de mai sus se poate scrie si astfel:

```
CREATE SCHEMA magazin;
```

Daca baza de date dorita exista deja, instructiunea va genera o eroare; spre exemplu, daca incercam sa cream din nou baza de date magazin:

```
mysql> CREATE SCHEMA magazin;
ERROR 1007 (HY000): Can't create database 'magazin'; database exists
```

Pentru a determina crearea bazei de date numai in cazul in care aceasta nu exista deja, fara a genera o eroare, se foloseste clauza IF NOT EXISTS. Aceasta transforma eroarea intr-un warning, care poate fi vizualizat apoi cu comanda SHOW WARNINGS.

Clauza IF NOT EXISTS este utila atunci cand dorim sa ne asiguram de crearea unei baze de date fara a primi insa o eroare in cazul in care ea exista deja. Un exemplu tipic este cel al unui script SQL care creeaza o baza de date si apoi efectueaza diferite operatii in interiorul acesteia; fara clauza IF NOT EXISTS, eroarea cauzata de eventuala existenta a bazei de date ar determina neexecutarea restului instructiunilor din script, desi exista toate conditiile pentru corecta lor functionare.

```
mysql> CREATE DATABASE IF NOT EXISTS magazin;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+-----------------------------------------------------+
| Level | Code | Message                                             |
+-------+------+-----------------------------------------------------+
| Note  | 1007 | Can't create database 'desters'; database exists |
+-------+------+-----------------------------------------------------+
1 row in set (0.00 sec)
```

Exista doua proprietati ale unei baze de date care se pot seta la crearea ei: CHARACTER SET si COLLATION. Ele joaca rolul de default pentru proprietatile cu acelasi nume ale tabelelor continute. Aceste doua proprietati determina caracterele disponibile pentru valorile de tip string (sir de caractere) din tabele si felul in care se face compararea si ordonarea stringurilor. Detalii suplimentare vor fi oferite la sectiunea dedicata stringurilor din acest curs.

```
CREATE SCHEMA IF NOT EXISTS magazin CHARACTER SET latin1 COLLATE latin1_general_ci;
```

2.1.2. Stabilirea bazei de date curente

Instructiunea USE permite setarea bazei de date implicite. Toate referintele ulterioare la numele de tabele, in masura in care nu contin explicit numele bazei de date, vor fi considerate ca facand parte din baza de date implicita. In plus, unele instructiuni pentru baze de date opereaza automat pe baza de date default daca nu este specificat explicit numele altei baze de date.

Exemplu: incercarea de afisare a listei de tabele atunci cand nu a fost definita baza de date default:

```
mysql> SHOW TABLES;
ERROR 1046 (3D000): No database selected

mysql> USE magazin;
Database changed

mysql> SHOW TABLES;
+------------------+
| Tables_in_magazin |
+------------------+
| produse           |
| utilizatori       |
+------------------+
2 rows in set (0.00 sec)
```

In masura in care interactiunea cu serverul se desfasoara prin intermediul utilitarului *mysql*, baza de date implicita se poate stabili inca de la lansarea in executie a clientului mysql, prin pasarea sa ca argument la apelare:

```
user1@mycomputer:~$ mysql -u user2 magazin
mysql> SHOW TABLES;
+------------------+
| Tables_in_magazin |
+------------------+
| produse           |
| utilizatori       |
+------------------+
```

2.1.2. Stergerea unei baze de date

Stergerea unei baze de date se efectueaza folosind instructiunile DROP DATABASE sau DROP SCHEMA.

Atentie! Daca baza de date nu este goala, ea va fi stearsa impreuna cu toata informatia continuta (tabele si inregistrarile lor)!

```
DROP DATABASE magazin;
DROP SCHEMA magazin;
```

Atunci cand o aplicatie client are nevoie sa se asigure de stergerea unei baze de date fara a primi eventuala eroare in cazul in care aceasta a fost deja stearsa, se poate folosi clauza suplimentara IF EXISTS, care genereaza doar un warning in cazul in care baza de date nu exista. Warning-ul poate fi vizualizat cu SHOW WARNINGS.

```
DROP DATABASE IF EXISTS magazin;
DROP SCHEMA IF EXISTS magazin;
```

2.1.3. Modificarea parametrilor unei baze de date

MySQL ofera instructiunile ALTER DATABASE sau ALTER SCHEMA ce permit modificarea parametrilor unei baze de date. De remarcat insa faptul ca ele nu pot fi folosite pentru a redenumi baza de date, ci numai pentru schimbarea setului de caractere sau al collation-ului (simultan sau pe rand):

```
ALTER DATABASE magazin CHARACTER SET utf8;
ALTER DATABASE magazin COLLATE utf8_romanian_ci;
ALTER DATABASE magazin CHARACTER SET utf8 COLLATE utf8_romanian_ci;
```

Atunci cand numele bazei de date nu este specificat, instructiunea ALTER DATABASE/SCHEMA actioneaza asupra bazei de date curente, setata cu instructiunea USE:

```
mysql> USE magazin;
Database changed

mysql> alter schema character set latin1;
Query OK, 1 row affected (0.01 sec)
```

2.1.4. Vizualizarea bazelor de date existente

Lista completa de baze de date disponibile pe un server poate fi obtinuta cu instructiunile SHOW DATABASES sau SHOW SCHEMAS:

```
mysql> SHOW DATABASES;
+----------------------------+
| Database                   |
+----------------------------+
| information_schema         |
| magazin                    |
| mysql                      |
| test                       |
+----------------------------+
```

In lista obtinuta se gasesc si baze de date care nu au fost create de catre utilizator, dar care sunt insa esentiale pentru functionarea serverului. Doua exemple notabile sunt:

- **mysql** – este baza de date in care serverul memoreaza privilegiile diferitilor clienti asupra informatiilor memorate in bazele de date
- **INFORMATION_SCHEMA** – este o baza de date cu *meta-informatie:* informatii despre celelalte baze de date aflate pe server. Prin interogarea ei se pot afla detalii despre parametrii si structura tabelelor componente

Lista poate fi restransa impunand conditii asupra numelui bazei de date. In acest scop se poate folosi clauza LIKE si metacaracterul % ce inlocuieste zero sau mai multe caractere. Exemplu: vizualizarea tuturor bazelor de date al caror nume incepe cu m:

```
mysql> SHOW DATABASES LIKE 'm%';
+----------------------------+
| Database                   |
+----------------------------+
| magazin                    |
| mysql                      |
+----------------------------+
```

2.1.5. Vizualizarea parametrilor unei baze de date

Instructiunea SHOW CREATE DATABASE (sau SHOW CREATE SCHEMA) este cea care afiseaza comanda de creare (cu setul complet de parametri) a unei baze de date:

```
mysql> SHOW CREATE DATABASE magazin;
+----------+-----------------------------------------------------------------------+
| Database | Create Database                                                       |
+----------+-----------------------------------------------------------------------+
| magazin  | CREATE DATABASE `magazin` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+-----------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Nota: *o modalitate alternativa de a obtine informatie despre o baza de date este interogarea bazei de date information_schema.*

2.2. Operatii cu tabele

2.2.1. Crearea unei tabele

2.2.1.1. Ce presupune crearea unei tabele

A crea o tabela SQL inseama a specifica mai multe elemente:
- numele tabelei. Acesta va fi folosit ulterior in instructiunile SQL pentru referirea la tabela in cauza sau la coloanele/inregistrarile sale.
- formatul de stocare a datelor (asa-numitul „storage engine”). Decizia folosirii unui anumit storage engine va influenta performantele ulterioare ale lucrului cu informatia continuta in tabela (vezi sectiunea despre storage engines)
- lista atributelor(coloanelor): pentru fiecare dintre ele se stabileste numele coloanei, tipul de date asociat si eventualele optiuni suplimentare
- (optional) definirea indecsilor – structuri de date redundante ce au rolul de a spori viteza de cautare a datelor si de a introduce restrictii suplimentare asupra informatiilor uneia sau mai multor coloane

2.2.1.2. Storage engines

MySQL dispune de mai multe formate de stocare a datelor („storage engines”). Acestea difera intre ele prin:
- **facilitatile oferite**. Un format de stocare poate sa dispuna sau nu de facilitati precum:
  - *tranzactii* – posibilitatea de a defini succesiuni de operarii „atomice”, care fie reusesc in integralitatea lor, fie esueaza neproducand nicio modificare
  - *integritate referentiala* – o modalitate de a impune ca, atunci cand o informatie este referita de catre o alta, ea sa nu poata fi stearsa sau modificata neconditionat

- **performante**. Fiecare motor de stocare se comporta optim in anumite scenarii de utilizare – spre exemplu, unele sunt recomandabile atunci cand operatiile sunt preponderent de introducere de informatie, pe cand altele optimizeaza cautarea si extragerea informatiei
- **felul in care serverul MySQL stocheaza datele in sistemul de fisiere sau in memoria RAM**. In functie de storage engine-ul ales, datele pot fi memorate intr-un singur fisier pentru mai multe tabele, in mai multe fisiere sau integral in memoria RAM

Iata cateva formate de stocare mai importante suportate de MySQL:
- **MyISAM** – motorul de stocare traditional MySQL, urmaș al mai vechiului ISAM. Este un motor de stocare simplu si rapid, dar care in schimb nu suporta tranzactii si referential integrity
- **InnoDB** – motorul de stocare default începând cu MySQL 5.5.5. Este un motor tranzactional si care suporta referential integrity. Se comporta bine in scenarii de acces multiplu si de operatii preponderente de introducere de date; prezinta in schimb o penalitate de viteza si un consum de memorie superior
- **MEMORY** – motor care presupune stocarea integrala a datelor in RAM. Continutul unei tabele care are un astfel de format de stocare se pierde la restartarea serverului (structura tabelei este insa pastrata). Informatiile stand in memorie, viteza manipularii lor este maxima. Lipsesc in schimb facilitatile tranzactionale si cele legate de integritatea referentiala

Engine-urile MyISAM și InnoDB sunt built-in și nu pot fi dezactivate, restul sunt dezactivabile din fisierul de configurare MySQL (precizare: InnoDB a devenit ne-dezactivabil incepand cu MySQL 5.7.5).

Pentru a vizualiza lista de engine-uri disponibile exista instructiunea SHOW ENGINES:

```
mysql> SHOW ENGINES;
+------------+----------+----------------------------------------------------------------+
| Engine     | Support  | Comment                                                        |
+------------+----------+----------------------------------------------------------------+
| MyISAM     | YES      | MyISAM storage engine                                          |
| MEMORY     | YES      | Hash based, stored in memory, useful for temporary tables      |
| InnoDB     | DEFAULT  | Supports transactions, row-level locking, and foreign keys     |
[... output omis...]
```

Decizia folosirii unui anumit storage engine trebuie luata tinand cont de modul de utilizare a datelor din tabela. In functie de operatiile preponderente si de facilitatile dorite, se alege motorul de stocare cel mai potrivit. Spre exemplu, daca avem nevoie de tranzactii, ne vom orienta probabil catre InnoDB; daca nu sunt necesare tranzactii si integritate referentiala, in schimb dorim viteza buna de extragere a datelor, vom alege probabil MyISAM (sau MEMORY, in masura in care este convenabil ca datele sa stea exclusiv in RAM).

Nota: *alegerea engine-ului nu influenteaza felul in care sunt manipulate datele – clientul va folosi tot instructiuni SQL de tip DML in acest scop – ci performantele lucrului cu datele.*

2.2.1.3. Tipuri de date MySQL

Fiecare coloana a unei tabele MySQL are asociat un anumit tip de date, care se stabileste la crearea tabelei/coloanei. Tipul de date al unei coloane determina natura informatiei, plaja de valori posibile si cantitatea de memorie ocupata pentru valorile acelei coloane. Fiecare valoare a unei inregistrari dintr-o tabela trebuie sa se supuna definitiei de coloana corespunzatoare.

Tipurile de date SQL pot fi impartite in urmatoarele categorii:

- **numerice** – folosite pentru memorarea urmatoarelor categorii de numere:

- **intregi**. Tipurile de date corespunzatoare sunt **TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT**
- **in virgula mobila**. Sunt tipuri de date folosite pentru numere rationale, numarul de zecimale fiind insa variabil. Tipuri de date MySQL: **FLOAT** si **DOUBLE**
- **in virgula fixa (numar prestabilit de zecimale)**. Tipul MySQL corespunzator este **DECIMAL**.

- **data/timp** – utilizate pentru a memora an, data calendaristica, moment exact in timp etc. Tipurile de date MySQL sunt **DATE, TIME, DATETIME, YEAR** si **TIMESTAMP**
- **sir de octeti –** pentru memorarea de informatii non-text (ex: imagini memorate in baza de date). Tipuri de date: **BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB** si **LONGBLOB**
- **sir de caractere** – pentru memorarea de siruri de caractere cu lungime fixa sau variabila. Tipuri de date: **CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT** si **LONGTEXT**
- **seturi de valori discrete** – folosite atunci cand o coloane nu poate lua decat valori dintr-o multime predefinita (ex: zilele saptamanii, notele muzicale etc). Tipurile de date corespunzatoare sunt **ENUM** si **SET**

Abordarea pe larg a tipurilor de date va fi facuta intr-o sectiune ulterioara a cursului.

Nota: *majoritatea DBMS-urilor prezinta abateri de la tipurile de date SQL standard. Chiar si atunci cand un tip de date dintr-un anumit DBMS se numeste la fel ca cel ANSI, el nu are neaparat aceleasi proprietati! Studiati manualul respectivului soft!*

2.2.1.4. Definitii de coloane

2.2.1.4.1. Componentele si sintaxa unei definitii de coloana

O definitie de coloana este formata din:

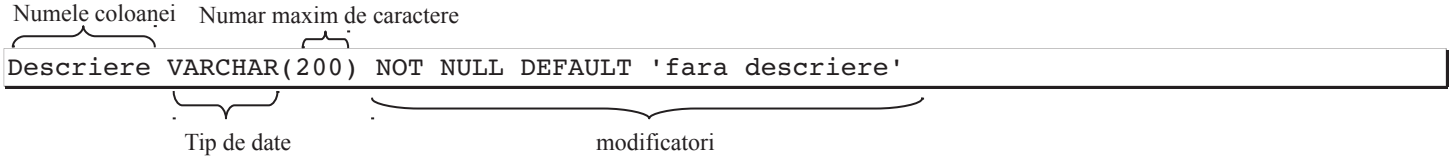
- numele coloanei. Acesta respecta regulile enuntate la capitolul dedicat identificatorilor SQL (numele de baze de date, tabele si coloane)
- tipul de date al coloanei. Poate fi unul dintre tipurile de date enumerate anterior
- eventuale detalii suplimentare asociate tipului de date al coloanei (ex: numar de zecimale pentru DECIMAL, lungime maxima pentru VARCHAR, valori posibile pentru ENUM/SET etc)
- eventuali modificatori ai coloanei – ex: daca permite sau nu NULL, daca contine numere cu semn sau fara in cazul tipurilor de date numerice etc.

Sintaxa generala a unei definitii de coloana este:

```
NumeColoana TipDeDate[(lungime/valori)] [modificatori]
```

Nota: *elementele aflate intre paranteze drepte [...] sunt optionale.*

Exemplu: definitia unei coloane pentru descrierea unui produs:



2.2.1.4.2. „Valoarea” NULL

In tabelele SQL, NULL indica absenta valorii de pe o anumita linie/coloana. NULL nu poate fi considerat propriu-zis o valoare: el nu face parte dintre valorile valide pentru tipul de date al coloanei – spre exemplu, NULL nu este totuna cu 0 (pentru o coloana cu tip de date numeric) sau cu sirul vid ”” (pentru coloane de tip sir de caractere). Totusi, el poate fi introdus explicit pe o coloana a unei inregistrari, in masura in care definitia coloanei o permite.

Nota: a nu se confunda NULL cu 'NULL' in instructiunile SQL; primul reprezinta „valoarea” NULL, iar cel de-al doilea sirul de caractere format din literele N, U, L ,L.

La crearea structurii unei tabele se stabileste, pentru fiecare coloana in parte, daca coloana respectiva permite NULL. Acest lucru se poate petrece in doua moduri:

- atunci cand nu specificam explicit aceasta proprietate a coloanei, MySQL va presupune automat ca coloana accepta NULL. Putem de asemenea permite explicit prezenta lui NULL pe coloana folosind modificatorul NULL in definitia coloanei. Urmatoarele doua definitii de coloana sunt echivalente:

```
Nume VARCHAR(100)
Nume VARCHAR(100) NULL
```

- putem specifica explicit modificatorul NOT NULL in definitia coloanei, obligand astfel toate inregistrarile sa aiba o valoare pe acea coloana:

```
Username VARCHAR(50) NOT NULL
```

2.2.1.4.3. Valori default

Precum se va vedea, la introducerea unei noi inregistrari intr-o tabela este posibil sa nu fie specificate valori pentru toate coloanele inregistrarii, ci doar pentru o parte dintre ele. In cazul coloanelor ale caror valori lipsesc vor fi folosite valorile implicite (daca exista) pentru acele coloane pentru a completa inregistrarea.

Valoarea default pentru o coloana poate fi:

- stabilita explicit, de catre creatorul tabelei, prin precizarea ei in definitia coloanei, folosind clauza DEFAULT (vezi exemplele)
- stabilita automat, de catre server, atunci cand in definitia coloanei nu este prezenta clauza DEFAULT: daca definitia coloanei permite NULL, valoarea implicita va fi automat NULL
- absenta, caz in care, la introducerea unei inregistrari la care nu specifica valoarea pentru coloana in cauza, serverul va actiona in functie de felul in care a fost configurat (vezi sectiunea din curs legata de sql_mode). In modul non-strict, serverul va insera automat o valoare din oficiu pe acea coloana, aleasa in functie de tipul de date al coloanei (coloanele de tip numeric vor primi valoarea 0, cele de tip sir de caractere – sirul vid ””, cele de tip ENUM prima valoare din multimea de valori permise etc.). In modul strict va fi generata o eroare.

***Nota:** atunci cand serverul lucreaza in modul strict, daca in definitia unei coloane ce nu permite NULL nu se specifica valoarea implicita, clientii vor fi obligati sa introduca intotdeauna valori explicite pe acea coloana.*

Exemple:

```
Culoare VARCHAR(50) NOT NULL DEFAULT 'rosu'
/* valorile pe aceasta coloana se introduc explicit; nu avem voie insa sa introducem
explicit valoarea NULL. Daca valoarea coloanei lipseste se va introduce 'rosu' */
```

```
MediaGenerala INT
MediaGenerala INT NULL
/* se permite NULL, care este in acelasi timp si valoarea default */
```

```
NumarProcesoare INT NULL DEFAULT 1
/* se permite NULL, dar el poate fi introdus doar explicit pe acea coloana; daca valoarea
coloanei lipseste, se introduce 1 */
```

2.2.1.5. Modalitati de creare a unei tabele MySQL

O tabela MySQL poate fi creata folosind instructiunea CREATE TABLE in 3 moduri, in functie de scenariul in care ne aflam:

- specificand explicit proprietatile tabelei si lista definitiilor de coloane:

```
CREATE TABLE nume_tabela(definitie_coloana_1, definitie_coloana2,...)
```

- copiind structura unei tabele deja existente:

```
CREATE TABLE nume_tabela_noua LIKE nume_tabela_veche
```

- pornind de la rezultatul unei interogari (care poate contine coloane extrase selectiv dintr-una sau mai multe tabele)

```
CREATE TABLE nume_tabela SELECT coloane FROM alte_tabele
```

In toate cazurile de mai sus, daca tabela exista deja se va genera o eroare; folosind aceeasi clauza suplimentara ca in cazul crearii de baze de date (IF NOT EXISTS), putem determina serverul sa genereze in locul erorii un warning, afisabil cu SHOW WARNINGS:

```
CREATE TABLE IF NOT EXISTS nume_tabela_noua LIKE nume_tabela_veche;
```

Nota: *la folosirea clauzei IF NOT EXISTS, serverul verifica numai existenta vechii tabele, nu si daca structura sa corespunde cu cea specificata in comanda de creare. Daca exista deja o tabela cu acelasi nume dar cu structura diferita, ea va fi pastrata ca atare, instructiunea de creare neproducand modificari asupra ei.*

2.2.1.6. Crearea unei tabele prin specificarea explicita a definitiilor coloanelor

Formatul de baza al instructiunii de creare de tabela este urmatorul:

```
CREATE TABLE nume_tabela (definitie_coloana_1, definitie_coloana_2,...)
```

Exemplu: crearea unei tabele carti ce contine coloanele titlu si nrpagini:

```
CREATE TABLE carti (Titlu VARCHAR(200), NrPagini INT);
```

Cu aceasta instructiune va fi creata o tabela cu definitia specificata si care foloseste motorul de stocare default (de obicei MyISAM). Daca dorim un alt motor de stocare il putem specifica cu clauza ENGINE dupa cum urmeaza:

```
CREATE TABLE carti (Titlu VARCHAR(200), NrPagini INT ) ENGINE InnoDB;
```

Nota: *la formatul de baza al instructiunii de creare mai pot fi adaugate clauze suplimentare ce specifica diverse alte proprietati ale tabelei (vezi MySQL Reference Manual).*

2.2.1.6. Crearea unei tabele prin copierea structurii unei tabele existente

Structura unei tabele poate fi creata pe baza uneia deja existente, folosind o alta forma a instructiunii CREATE TABLE:

```
CREATE TABLE produse2 LIKE produse;
```

Dupa executarea instructiunii de mai sus, tabela *produse2* va contine aceleasi coloane (cu definitii identice) ca si tabela produse. In schimb, chiar daca *produse* continea date, tabela *produse2* va fi goala, deoarece este copiata doar structura tabelei existente, nu si continutul.

Nota: *CREATE TABLE...LIKE copiaza structura si indecsii tabelei insa nu copiaza si cheile externe (cele care sunt folosite pentru a asigura integritatea referentiala).*

2.2.1.7. Crearea unei tabele pornind de la rezultatul unei interogari

O tabela poate fi creata pe baza rezultatului unei interogari de tip SELECT (instructiunea care realizeaza extragerea datelor din tabele). Avand in vedere ca acest rezultat contine coloane selectate din una sau mai multe alte tabele si care au, deci, nume si tip de date, exista toate informatiile necesare pentru a putea crea o tabela pe baza rezultatului:

```
CREATE TABLE nume_studenti_maturi SELECT nume, prenume FROM studenti WHERE varsta>30
```

Precum se va vedea, o interogare de tip SELECT poate extrage coloane si inregistrari din mai multe tabele, din acest punct de vedere fiind mai flexibila decat instructiunea CREATE...LIKE. Pe de alta parte, crearea unei tabele folosind CREATE...SELECT, chiar daca extrage informatie dintr-o singura alta tabela, nu copiaza toate caracteristicile acesteia (ex: nu sunt copiati indecsii si alti parametri ai tabelei).

2.2.1.8. Tabele de tip TEMPORARY

La crearea unei tabele se poate specifica clauza TEMPORARY. Rezultatul este crearea unei tabele ce difera de una obisnuita prin urmatoarele caracteristici principale:

- tabela exista atata timp cat conexiunea clientului care a creat-o este inca activa. Serverul va sterge automat tabela la inchiderea conexiunii
- tabela este vizibila numai pentru clientul care a creat-o. Mai multi clienti pot crea tabele temporare cu nume identice fara ca acestea sa intre in conflict

```
CREATE TEMPORARY TABLE produse ( Nume VARCHAR(100), Pret INT);
CREATE TEMPORARY TABLE produse2 like produse;
CREATE TEMPORARY TABLE produse2 SELECT * FROM produse;
```

O tabela temporara poate primi acelasi nume cu una obisnuita deja existenta; in acest caz, cea existenta va fi „mascata”, utilizatorul nemaiavand acces la ea pana cand cea temporara dispare. O eventuala instructiune DROP TABLE va actiona in astfel de cazuri asupra tabelei temporare, neafectand-o pe cea persistenta.

2.2.2. Vizualizarea listei de tabele dintr-o baza de date

Instructiunea SHOW TABLES afiseaza lista tabelelor non-temporare dintr-o baza de date. Executata fara parametri, ea va afisa lista tabelelor din baza de date curenta (cea setata cu USE). Adaugand clauza FROM baza_de_date, vor fi listate tabelele din baza de date vizata:

```
SHOW TABLES;
SHOW TABLES FROM magazin;
```

Atunci cand se doreste restrangerea listei de tabele afisate se poate folosi clauza LIKE pentru a specifica restrictii asupra numelui de tabela. Spre exemplu, daca dorim sa listam doar tabelele al caror nume contine fragmentul *user*, vom scrie:

```
SHOW TABLES FROM magazin LIKE '%user%';
```

2.2.3. Vizualizarea caracteristicilor unei tabele

Exista diferite instructiuni ce pot fi folosite pentru a determina structura si caracteristicile unei tabele:

- **SHOW TABLE STATUS**. Afiseaza informatii despre tabelele non-temporare din baza de date curenta. Cu clauza suplimentara LIKE se pot filtra numele tabelelor listate.
- **DESCRIBE NumeTabela** sau **SHOW COLUMNS FROM NumeTabela**. Instructiunea afiseaza lista coloanelor tabelei impreuna cu caracteristicile lor (nume, tip de date, indecsi, acceptare NULL, valoare default)
- **SHOW CREATE TABLE NumeTabela**. Afiseaza instructiunea exacta ce poate fi folosita pentru a crea o tabela identica cu cea curenta.

Exemplu: crearea unei tabele si detaliile afisate de fiecare comanda:

```
mysql> CREATE TABLE products(id INT AUTO_INCREMENT PRIMARY KEY,name VARCHAR(100) DEFAULT
'no name');
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW CREATE TABLE products\G
*************************** 1. row ***************************
       Table: products
Create Table: CREATE TABLE `products` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(100) default 'no name',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> SHOW TABLE STATUS LIKE 'products'\G
*************************** 1. row ***************************
           Name: products
         Engine: MyISAM
        Version: 10
     Row_format: Dynamic
           Rows: 18
 Avg_row_length: 59
    Data_length: 1284
Max_data_length: 281474976710655
   Index_length: 1024
      Data_free: 216
 Auto_increment: NULL
    Create_time: 2009-04-14 14:48:36
    Update_time: 2009-04-14 15:25:06
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.00 sec)

mysql> DESCRIBE products; # echivalent cu SHOW COLUMNS FROM products;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
```

```
| id     | int(11)       | NO   | PRI | NULL     | auto_increment |
| name   | varchar(100)  | YES  |     | no name  |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

2.2.4. Stergerea unei tabele

Stergerea unei tabele presupune stergerea tuturor inregistrarilor din tabela si a definitiei acesteia. Comanda folosita este DROP TABLE:

```
DROP TABLE produse;
```

Ca si in cazul operatiilor cu baze de date, incercarea de a sterge o tabela inexistenta se soldeaza cu o eroare; clauza suplimentara IF EXISTS transforma eroarea in warning, vizualizabil cu SHOW WARNINGS:

```
mysql> DROP TABLE IF EXISTS categorii;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+-------+------+-----------------------------+
| Level | Code | Message                     |
+-------+------+-----------------------------+
| Note  | 1051 | Unknown table 'catagorii'   |
+-------+------+-----------------------------+
1 row in set (0.00 sec)
```

Nota: *A nu se confunda stergerea unei tabele cu golirea ei! Golirea inseamna stergerea tuturor inregistrarilor dar pastrarea structurii.*

2.2.5. Golirea unei tabele

Golirea unei tabele presupune stergerea tuturor inregistrarilor din acea tabela. Structura tabelei va ramane insa intacta, astfel incat dupa golire se pot introduce date fara a repeta operatia de creare.

Golirea unei tabele se poate realiza in doua moduri: folosind instructiunea DELETE sau TRUNCATE.

```
DELETE FROM produse;
TRUNCATE produse;
```

Cele doua instructiuni produc acelasi efect dar nu sunt echivalente:

- DELETE este folosit pentru a sterge inregistrari din tabele (in mod selectiv, de obicei); utilizarea ca in exemplul de mai sus sterge toate randurile tabelei specificate. Uneori insa aceasta operatie determina o stergere inregistrare cu inregistrare, lucru care poate consuma mult timp in cazul unei tabele mari
- TRUNCATE este o instructiune special gandita pentru golirea unei tabele, ea incercand sa eficientizeze pe cat posibil aceasta operatie (ex: atunci cand este posibil, TRUNCATE sterge tabelul cu totul si creeaza unul nou cu structura identica)

2.2.6. Modificarea unei tabele

2.2.6.1. Schimbarea numelui

Modificarea numelui unei tabele poate fi efectuata cu instructiunile ALTER TABLE sau RENAME. Diferenta intre ele este ca RENAME poate face redenumiri multiple dintr-o singura instructiune, in schimb nu poate actiona asupra tabelelor temporare:

```
ALTER TABLE useri RENAME utilizatori
ALTER TABLE products RENAME produse
RENAME TABLE useri TO utilizatori, products TO produse
```

2.2.6.2. Schimbarea motorului de stocare

Motorul de stocare al unei tabele poate fi schimbat din mers, fara a altera datele continute:

```
ALTER TABLE produse ENGINE InnoDB;
```

Dat fiind insa faptul ca locatia si formatul datelor difera de la un motor de stocare la altul, este posibil ca pentru tabele cu multe inregistrari operatia sa fie una de durata.

2.2.6.3. Modificarea definitiilor de coloane

2.2.6.3.1. Stergerea unei coloane

MySQL permite stergerea de coloane dintr-o tabela SQL existenta fara a fi afectat restul informatiei din tabela. Ulterior stergerii, fiecare inregistrare va avea cu o valoare mai putin:

```
ALTER TABLE produse DROP COLUMN denumire;
```

2.2.6.3.2. Adaugarea unei coloane

Ca si stergerea, adaugarea poate fi facuta fara a afecta restul datelor. Fiecare inregistrare din tabela va avea pe pozitia noii coloane valoarea default a coloanei. Sintaxa generala este:

```
ALTER TABLE produse ADD COLUMN definitie_coloana [pozitie];
```

Nota: *cuvântul COLUMN este optional.*

Noua coloana este adaugata:
- la sfarsitul listei de coloane, in cazul in care nu se specifica pozitia de adaugare
- la inceput, daca pozitia este FIRST
- imediat dupa coloana specificata, daca pozitia este de forma AFTER nume_coloana

Exemple:

```
# adaugare la sfarsitul listei de coloane; valoare default NULL
ALTER TABLE produse ADD COLUMN Pret DECIMAL(5,2)
# introducere pe prima pozitie; valoare default 0
ALTER TABLE produse ADD id INT NOT NULL DEFAULT 0 FIRST;
/* introducere pe pozitia imediat urmatoare unei coloane specificate; valoare default
sirul vid "" */
ALTER TABLE produse ADD COLUMN Descriere VARCHAR(200) NOT NULL DEFAULT '' AFTER Nume;
```

2.2.6.3.3. Modificarea unei coloane

Standardul ANSI SQL permite doar o manipulare limitata a definitiei unei coloane: posibilitatea de a schimba sau elimina valoarea default. Softurile DBMS majore ofera insa extensii ale limbajului care permit modificarea „din mers" a definitiei de coloana, chiar si in conditiile in care tabela a fost deja populata si deci exista valori corespunzatoare acelei coloane. In acest ultim caz, se incearca ajustarea sau convertirea valorilor de pe coloana respectiva daca este nevoie pentru a se potrivi cu noua definitie a coloanei.

Sintaxa generala a instructiunii ANSI SQL pentru modificarea unei coloane este:

```
# setarea/schimbarea valorii default a coloanei
ALTER TABLE NumeTabela ALTER COLUMN NumeColoana SET DEFAULT ValoareDefault

# eliminarea valorii default a coloanei
ALTER TABLE NumeTabela ALTER COLUMN NumeColoana DROP DEFAULT
```

MySQL suporta aceasta sintaxa, insa adauga si propriul cuvant cheie – CHANGE – ca actiune posibila aplicabila unei coloane, rezultand instructiunea ALTER TABLE...CHANGE COLUMN. Sintaxa ei este:

```
ALTER TABLE NumeTabela CHANGE NumeColoana DefinitieNouaColoana
```

In acest fel se poate modifica atat numele coloanei, cat si tipul de date sau eventualii modificatori (valoare default, acceptarea de NULL etc)

Exemplu: schimbarea titlului coloanei Nume in Denumire, cu modificarea tipului de date si a modificatorilor:

```
ALTER TABLE produse ADD COLUMN Nume CHAR(100);
ALTER TABLE produse CHANGE COLUMN Nume Denumire VARCHAR(255) NOT NULL;
```

Nota: *asa cum MySQL a introdus CHANGE COLUMN, Oracle adauga MODIFY COLUMN (suportat si de MySQL!)*

2.3. BIBLIOGRAFIE

- MySQL 5 Certification Study Guide – Cap. 7, Databases
- MySQL 5 Certification Study Guide – Cap. 8, fara portiunile referitoare la indecsi (8.6, 8.7 si partial 8.8)
- MySQL Reference Manual – http://dev.mysql.com/doc/refman/5.0/en/sql-syntax-data-definition.html, portiunile referitoare la operatii cu baze de date si tabele (12.1.1, 12.1.4, 12.1.6, 12.1.10, 12.1.13, 12.1.17, 12.1.20)
- InnoDB vs MyISAM: http://www.mikebernat.com/blog/MySQL_-_InnoDB_vs_MyISAM
- MySQL 5 Certification Study Guide – Cap. 29.1, MySQL Storage Engines. Pentru detalii suplimentare, optional poate fi consultat si restul capitolului 29
- MySQL 5 Certification Study Guide – Cap. 5.1, Data Type Overview