1. FUNDAMENTE ALE BAZELOR DE DATE RELATIONALE

Cuprins

1.1. Notiuni generale

1.1.1. Ce este o baza de date

O baza de date reprezinta o colectie structurata de informatie stocata intr-un sistem de calcul. Structura ei este specificata inaintea introducerii datelor si determina:
- tipul si compozitia datelor memorate
- felul in care datele se afla in relatie unele cu altele

Scopul memorarii datelor (fie in format electronic, fie intr-altul) este ca mai apoi ele sa poata fi consultate cu rapiditate sau modificate. Structura unei baze de date este de asa natura gandita incat sa optimizeze aceste operatii de manipulare a datelor.

1.1.2. Necesitatea existentei bazelor de date

Necesitatea memorarii datelor a existat dintotdeauna. Stocam informatie zilnic, folosind foi de hartie, stickere sau simple noduri la batista. Dezavantajul apare insa de indata ce informatia incepe sa se acumuleze si gasirea ei devine din ce in ce mai problematica. Multi dintre noi au pierdut – nu o data – timp pretios cautand o informatie scrisa intr-o agenda sau intr-un fisier, in cazul careia nu ne mai aminteam nici un criteriu care sa ne faciliteze gasirea sa.

In era sistemelor de calcul, acestea se pot ocupa in locul nostru atat de memorarea datelor, cat si de cautarea lor, deoarece ele exceleaza la operatii numerice (cautarea unei informatii presupunand, in special, operatii de comparare). A se observa insa ca acest lucru presupune structurarea informatiei de asa natura incat ea sa poate fi categorisita si parcursa in mod algoritmic de catre un computer.

1.1.3. Ce este o baza de date relationala. Modele.

O baza de date relationala este una in care organizarea informatiei respecta modelul relational.

De-a lungul timpului au fost elaborate diferite modele de organizare a informatiei intr-o baza de date, fiecare avand in spate un anumit model matematic:
- modelul flat-file – cea mai simpla (dar nu si cea mai eficienta/flexibila) modalitate de a memora date. Informatia este plasata intr-un fisier text sau binar, fiecare inregistrare ocupand cate o linie, valorile componente ale fiecarei inregistrari fiind separate prin delimitatori prestabiliti (ex: fisierele CSV)
- modelul ierarhic - informatia este structurata sub forma unui arbore in care fiecare inregistrare are un parinte si mai multe sub-inregistrari posibile (ex: cazul fisierelor XML)
- modelul retea – o extensie a modelului ierarhic, in care fiecare inregistrare poate avea mai multe sub-inregistrari dar si mai multe inregistrari parinte
- modelul relational – este un model bazat pe teoria multimilor si pe logica predicatelor, in care informatia este grupata in multimi de inregistrari numite tabele. Intre tabele pot fi definite relatii in mod flexibil, de asa natura incat datele sa poata fi stocate si manipulate eficient (vezi sectiunea dedicata structurii unei baze de date relationale)

1.1.4. Sisteme de gestiune a bazelor de date relationale

O baza de date, asa cum a fost definita pana acum, este o notiune abstracta. Pentru a putea defini structura unei astfel de baze de date si a memora efectiv informatie inauntrul sau avem nevoie de un software care sa realizeze toate functiile pe care aceste nevoi le implica. Un astfel de soft se numeste *sistem de gestionare a bazelor de date,* prescurtat in romana SGBD si in engleza DBMS (**D**ata**B**ase **M**anagement **S**ystem).

Un DBMS implementeaza functiile necesare pentru diferitele aspecte ale lucrului cu bazele de date:
- memorarea fizica a datelor, intr-un format optimizat in vederea operatiilor viitoare
- definirea si modificarea structurii bazelor de date
- introducerea informatiilor noi si manipularea celor existente
- filtrarea accesului la datele memorate (ex: un sistem de privilegii, care sa faca distinctie intre diversi utilizatori/clienti/aplicatii)
- implementarea unei modalitati de acces la toate aceste facilitati (sa nu uitam faptul ca, deseori, resursele puse la dispozitie de catre un DBMS sunt accesate de catre utilizatori sau aplicatii aflate pe alte statii decat cea pe care ruleaza software-ul de gestiune a bazelor de date)

1.1.5. Cine are nevoie sa introduca si sa manipuleze informatia?

Aparent, exista doua scenarii posibile:
- informatia este introdusa/accesata/manipulata de catre un utilizator (ex: un instructor adauga studenti noi in baza de date in care sunt memorati studentii academiei)
- informatia este introdusa/accesata/manipulata de catre o aplicatie (ex: un program care monitorizeaza activitatea unui server web memoreaza intr-o baza de date informatii despre paginile accesate, tipul de browser folosit etc)

Remarcam insa in primul caz ca, chiar daca spunem „utilizatorul introduce date”, aceasta operatie presupune interactiunea utilizatorului cu un DBMS si trimiterea de comenzi catre acesta. Utilizatorul interfateaza cu DBMS prin intermediul unui program de tip client – asadar tot o aplicatie. In concluzie, putem unifica cele doua cazuri vorbind – la modul general – despre *aplicatii client* sau, mai scurt, *clienti*.

1.1.6. Cum interactionam cu un DBMS

Trebuie subliniat, pentru inceput, faptul ca aplicatiile care acceseaza bazele de date o fac indirect, prin intermediul DBMS; atunci cand dorim, spre exemplu, sa cream o baza de date sau sa introducem informatie in ea, vom solicita softului DBMS sa faca acest lucru pentru noi, trimitandu-i comenzile corespunzatoare, insotite de eventualele date ce participa in operatia dorita. DBMS cunoaste formatul in care a fost memorata informatia acelei baze de date si are acces direct la datele in cauza; aplicatia client, in primul rand, nu poate avea intotdeauna acces direct la date (deseori clientii ruland pe alta masina decat DBMS), iar in al doilea rand nici nu doreste sa fie dependenta de modalitatea fizica de memorare a informatiei. **Clientul acceseaza datele independent de aspectele fizice ale stocarii lor pe HDD sau in RAM.**

Nota: *chiar daca accesul la date se face independent de formatul lor de stocare, clientul nu poate face complet abstractie de acest format, deoarece facilitatile oferite si performantele difera de la un format la altul.*

Avand in vedere ca DBMS ofera servicii clientilor in baza cererilor formulate de catre acestia, el contine un modul care actioneaza ca **server de baze de date**.

Nota: *termenul de server de baze de date este folosit uneori pentru a desemna statia pe care ruleaza DBMS sau softul DBMS. Riguros vorbind, nici unul dintre cazuri nu este corect, deoarece 1) DBMS indeplineste multe alte functii in afara de servirea clientilor, si 2) DBMS este software, statia pe care ruleaza acesta este hardware.*

Clientii unui DBMS sunt aplicatii care se pot afla:
- pe aceeasi masina cu serverul (ex: cazul companiilor de webhosting, unde interactiunea cu serverul de baze de date se efectueaza deseori prin intermediul unei aplicatii web, serverul de baze de date ruland pe aceeasi masina cu serverul web)
- pe o masina aflata in aceeasi retea sau intr-o alta retea (ex: aplicatii bancare, unde functionarul de la ghiseu ruleaza pe statia sa o suita de programe ce interfateaza cu un server de baze de date aflat undeva in reteaua bancii, pe alta masina)

Modalitatile de interfatare a clientilor cu DBMS sunt si ele diverse:
- cea mai comuna este cea care foloseste TCP/IP, ceea ce permite conectarea la DBMS atat pentru clientii aflati pe aceeasi masina cat si pentru cei din retea. In acest caz, unul dintre modulele software-ului DBMS joaca rolul de server de retea, acceptand conexiuni si implementand un protocol de comunicatie ce permite clientilor accesul la date
- atunci cand clientul si serverul se afla pe aceeasi statie, in functie de sistemul de operare gazda sunt disponibile si alte modalitati de conectare la server (socket in Linux/Unix, shared memory sau named pipe in Windows etc)

Clientii unui DBMS pot fi:
- utilitare dedicate pentru administrarea serverului sau manipularea structurii bazelor de date si a informatiilor continute – fie aplicatii grafice, fie programe care ofera utilizatorului posibilitatea de a trimite comenzi SQL catre server
- programe care se folosesc de datele memorate in bazele de date gestionate de catre DBMS (ex: un soft bancar prin intermediul caruia se creeaza noi conturi sau se vizualizeaza cele deja existente; informatiile despre conturi sunt stocate intr-o baza de date)

Cererile trimise de catre clienti serverului de baze de date poarta denumirea de **interogari** (query, in denumirea originala din limba engleza).

1.1.7. Ce este si ce rol are limbajul SQL

Unul dintre beneficiile importante pe care le ofera un DBMS este decuplarea modalitatii fizice de structurare a datelor (ex: organizarea, locatia si denumirea fisierelor) de modalitatea de accesare si manipulare a lor. Utilizatorul care beneficiaza de serviciile unui DBMS nu vede (si nici nu doreste sa stie) aspectele low-level are

stocarii informatiei, ci doreste o viziune logica a datelor memorate care sa ii permita usoara lor introducere, extragere si prelucrare.

In acest scop, DBMS pune la dispozitia utilizatorului fie o interfata grafica, fie un limbaj care sa ii permita acestuia interactiunea cu datele si facilitatile conexe acestora. Limbajul folosit in cazul bazelor de date relationale este SQL (Structured Query Language).

SQL este un limbaj ce contine instructiuni pentru definirea si modificarea structurii bazelor de date, accesarea si manipularea datelor si controlul accesului la date. El a fost creat la inceputul anilor '70 la IBM si folosit initial ca baza intr-unul dintre produsele acestei firme. Ulterior, in 1986, el este standardizat de catre ANSI (American National Standards Institute) si apoi de catre ISO (International Organization for Standardization), existand versiuni succesive ale standardului SQL, ultima dintre ele fiind SQL:2008. Toate softurile majore de tip DBMS ale zilelor noastre inglobeaza o forma a limbajului SQL.

1.1.7. Softuri cunoscute de tip DBMS

Prezentam mai jos lista celor mai cunoscute softuri de tip DBMS:
- **Oracle** – produs al Oracle Corporation; probabil cel mai cunoscut si folosit software proprietar de baze de date
- **DB2** – un produs al companiei IBM
- **Microsoft SQL** – produsul Microsoft pentru baze de date relationale
- **postgresql** – software DBMS de tip open-source
- **MySQL** – software DBMS open-source

Desi fiecare dintre aceste pachete software implementeaza limbajul SQL, fiecare dintre ele prezinta abateri de la standard sau extensii proprietare ale limbajului, care pe de o parte duc la performante mai bune sau o mai mare usurinta de utilizare, dar pe de alta parte introduc incompatibilitati intre diversele softuri DBMS.

1.2. Structura informatiei dintr-un DBMS relational

1.2.1. Baze de date

O baza de date relationala este o notiune logica – un ansamblu de informatie compus dintr-una sau mai multe tabele. Modalitatea fizica de stocare a datelor tine de softul DBMS folosit (diferind de la un DBMS la altul). Gratie limbajului SQL, clientul poate manipula datele independent de aspectele fizice ale stocarii datelor.

Desi este des folosita expresia „introducem date in baza de date”, informatia nu sta direct in baza de date, ci in niste sub-diviziuni ale sale numite tabele. O baza de date poate contine zero sau mai multe astfel de tabele. Deoarece datele efective stau in tabele, o baza de date care nu contine nicio tabela este o baza de date in care nu se afla informatie.

1.2.2. Tabele (relatii)

O tabela dintr-o baza de date relationala (denumita in teoria bazelor de date si „relatie”) reprezinta un ansamblu de inregistrari cu structura impusa, organizata tabelar (in randuri si coloane). Structura unei tabele este

specificata la crearea acesteia si presupune specificarea numarului de coloane si a numelui/tipului de informatie al fiecareia dintre ele. Fiecare inregistrare (rand) din tabela va fi obligata apoi sa aiba o valoare pentru fiecare dintre coloanele prezente in definitie.

Nota: *numarul de coloane este impus, numarul de randuri este variabil.*

Exemplu: dorim sa cream o tabela in care sa memoram informatii despre filme. Pentru fiecare film vrem sa avem titlul, durata si anul aparitiei. In aceste conditii, vom crea o tabela cu 3 coloane, numite – de exemplu – Titlu, Durata si An. Titlul este un sir de caractere, durata reprezinta un interval de timp, iar anul o valoare numerica; vom impune pentru fiecare coloana tipul de date corespunzator.

Imediat dupa crearea tabelei, ea nu contine date; ele trebuie introduse, inregistrare cu inregistrare. Fiecare inregistrare trebuie sa respecte structura tabelei: in exemplul nostru anterior, fiecare film va trebui sa aiba cele 3 caracteristici, iar cele 3 valori care compun fiecare inregistrare trebuie sa aiba tipurile de date corecte (ex: nu vom putea memora pe coloana An valoarea 'o mie noua sute optzeci' deoarece nu este numerica, ci string).

Nota: *a nu se confunda tabela cu tabelul! Tabela reprezinta un ansamblu de informatie cu o structura logica impusa la creare, pe cand tabelul este doar reprezentarea vizuala cea mai des intalnita pentru* datele continute *intr-o tabela. Intr-un tabel nu apar de obicei informatiile de structura (tip de date al coloanelor, impuneri suplimentare aplicate lor etc).*

1.2.3. Coloane (atribute)

O coloana a unei tabele contine ansamblul de valorilor de pe aceeasi pozitie ale tuturor inregistrarilor tabelei. Fiecare coloana are un set de caracteristici:

- numele (titlul) coloanei, folosit de catre clienti pentru a se referi la datele de pe coloana respectiva atunci cand interogheaza serverul
- tipul de date al coloanei. Toate inregistrarile (randurile) tabelei vor fi fortate sa aiba pe coloana in cauza valori de acel tip de date
- diferite proprietati suplimentare, care vor fi discutate in sectiunea dedicata crearii bazelor de date si tabelelor

Caracteristicile fiecarei coloane sunt specificate la crearea tabelei din care face parte. Desi in definitia unei tabele coloanele sunt create intr-o anumita ordine, la extragerea datelor din baza de date ele pot fi reordonate.

1.2.4. Randuri (inregistrari)

Un rand al unei tabele reprezinta un ansamblu de valori (date), cate una corespunzatoare fiecarei coloane. Putem considera ca fiecare valoare dintr-o tabela se afla la „intersectia” unei linii cu o coloana; o valoare este unic identificata odata ce am specificat baza de date, tabela, coloana si randul.

Spre deosebire de coloane, inregistrarile nu au nume, referirea la ele facandu-se prin intermediul valorilor componente (ex: studentii al caror nume de familie se termina cu „escu”). Compozitia unui rand este predeterminata de caracteristicile coloanelor: fiecare valoare a unui rand trebuie sa respecte tipul de date al coloanei de care apartine.

1.2.5. Conceptul de NULL

In implementarile de baze de date relationale exista conceptul de absenta a unei valori de pe o coloana a unei inregistrari. Acest lucru contravine regulilor modelului relational original, care impune ca toate inregistrarile sa aiba valori pentru toate coloanele. Pentru a indica absenta unei valori pe o anumita linie/coloana, se plaseaza in acea locatie NULL. Din acest punct de vedere, NULL nu poate fi considerat o valoare, si precum se va vedea nici nu se comporta ca atare (in comparatii, operatii aritmetice etc).

Posibilitatea de a folosi NULL se stabileste la nivel de fiecare coloana, la crearea tabelei. In aceeasi tabela putem avea coloane care permit NULL si coloane care il interzic (obligand astfel toate inregistrarile sa aiba valoare pe acea coloana).

1.3. Serverul MySQL

1.3.1. Prezentare generala

MySQL este un software open-source de tip DBMS relational, ce opereaza intr-o arhitectura client-server. Clientii MySQL se conecteaza la serverul MySQL efectuand cereri catre acesta. Programele client pot rula pe masini aflate in aceeasi retea cu serverul, in alta retea sau chiar pe masina serverului.

Pachetul MySQL contine urmatoarele categorii de software:

- **serverul MySQL**. Este cel care gestioneaza bazele de date, stocate pe hard-disk(uri) sau in memorie, si cel prin intermediul caruia clientii pot efectua operatiile dorite cu bazele de date (creare/stergere/modificare/definire structura/populare cu date/...etc...). Clientii se pot conecta fie prin retea, folosind protocolul TCP (portul din oficiu folosit de server fiind 3306), fie prin alte mijloace care depind de sistemul de operare gazda. Serverul este reprezentat de executabilul *mysqld*.
- **clienti MySQL**. Distributia MySQL include si programe ce interfateaza cu serverul, conectandu-se la el si efectuand cereri catre acesta. Distingem doua categorii de utilitare:
  - cele pentru manipulare a structurii bazelor de date si a datelor continute:
    - *mysql*, ce ofera utilizatorului o linie de comanda cu serverul SQL
    - *mysqldump*, ce realizeaza un backup al unei tabele
    - *mysqlimport*, ce importa date dintr-un fisier intr-o tabela a unei baze de date
  - cele pentru administrarea serverului
    - *mysqladmin* – utilitar de monitorizare si administrare a serverului MySQL
- **programe utilitare non-client**. In aceasta categorie intra softuri care nu se conecteaza la server, ci isi realizeaza functiile prin accesul direct la date. Un exemplu in acest sens il reprezinta programele de verificare si refacere a integritatii bazelor de date

MySQL este disponibil atat in Windows cat si Linux/Unix. Interactiunea intre server si clienti se efectueaza prin intermediul unui protocol de retea, care este acelasi indiferent de sistemul de operare al serverului si al clientilor. De aceea, este posibil ca statiile pe care ruleaza clientii sa aiba sistem de operare diferit de cel aflat pe statia serverului.

Nota: *in restul cursului, vom numi „mysql” (in aceasta ortografie) utilitarul client prezent in pachetul MySQL, iar cand vom dori sa ne referim la server vom scrie „serverul MySQL” sau „MySQL”.*

1.3.2. Aplicatii ajutatoare

In afara de aplicatiile incluse in distributia oficiala MySQL, exista alte cateva softuri care fac mai usoara interactiunea cu un server MySQL:

- **MySQL Query Browser** – utilitar grafic pentru interogarea manuala a serverului MySQL, cu facilitati multiple (creare de scripturi SQL, construire de interogare prin drag-n-drop, analiza si modificarea grafica a structurii unei baze de date, acces facil la help-ul comenzilor SQL etc)
- **MySQL Administrator** – utilitar grafic pentru administrarea serverului MySQL. Cuprinde facilitati precum urmarirea si reglarea parametrilor serverului, efectuarea de backup-uri si restaurari ale bazelor de date, gestionarea privilegiilor de acces la datele de pe server etc.
- **phpMyAdmin** – aplicatie web pentru interactiunea cu un server MySQL. Folosita foarte mult in cazul site-urilor web care interfateaza cu servere de baze de date.

Primele doua aplicatii sunt de sine statatoare, ele putand fi descarcate si instalate de pe site-ul www.mysql.com sub forma pachetului MySQL GUI Tools. Sunt disponibile atat versiuni pentru Linux/Unix, cat si pentru Windows. Cea de-a treia are nevoie, pentru a putea rula, de un server web ce integreaza si interpretorul PHP; cat timp aceasta conditie este indeplinita, nu conteaza sistemul de operare pe care ruleaza phpMyAdmin sau browserul client, fiind posibile orice combinatii.

1.4. Folosirea clientului mysql

1.4.1. Moduri de lucru

Utilitarul *mysql* ofera utilizatorului o modalitate simpla de a trimite comenzi SQL catre serverul MySQL. Programul *mysql* poate fi folosit in doua moduri:

- **modul interactiv**. In acest mod de lucru, dupa pornirea *mysql* utilizatorul este plasat intr-o linie de comanda din care poate trimite comenzi serverului si poate vizualiza raspunsurile acestuia
- **modul non-interactiv** („batch mode”). Rulat in acest mod, *mysql* preia si executa comenzi deja scrise intr-un fisier si le trimite serverului de baze de date. Modul non-interactiv este convenabil atunci cand dorim sa rulam cu mai multe ocazii o aceeasi succesiune de comenzi SQL, sau daca dorim rularea unei serii de comenzi fara a le introduce una cate una (ex: copierea structurii unei baze de date pe un alt server, care presupune exportare de pe serverul original si importare pe cel nou)

Modul de lucru poate fi ales la lansarea in executie a utilitarului, sau, odata lansat *mysql* in mod interactiv, exista comenzi care au ca efect prelucrarea instructiunilor SQL dintr-un fisier.

1.4.3. Lansarea in executie a programului mysql

Pornirea *mysql* se realizeaza in general din linia de comanda oferita de sistemul de operare (Command Prompt in cazul Windows, consola sau emulator de terminal in Linux/Unix). La invocare se pot pasa executabilului optiuni ce configureaza diferitele aspecte ale functionarii programului si ale interactiunii sale cu serverul MySQL:

- optiuni legate de conectarea la server (modalitatea de conectare, user si parola folosite, adresa/portul/socket-ul pe care asculta serverul etc)
- optiuni ce specifica tipul de output produs (text, HTML, XML)

- optiuni ce personalizeaza interactiunea *mysql* cu utilizatorul (afisarea automata a warning-urilor, afisarea automata a numelor de coloane, etc)

--host -h	Numele sau adresa serverului statiei pe care ruleaza serverul MySQL
--user -u	Username-ul folosind pentru autentificarea clientului la serverul MySQL
--password=parola -p parola	Parola folosita pentru conectarea la serverul MySQL
--socket=cale/catre/fisier.socket	In cazul in care clientul se afla pe aceeasi statie cu serverul si conexiunea se face prin intermediul unui fisier local de tip socket (Linux/Unix), calea catre fisierul in cauza
--protocol={TCP|SOCKET|PIPE|MEMORY}	Modalitatea de conectare folosita (TCP/IP, socket etc)
--safe-updates --i-am-a-dummy	Interzice instructiunile SQL care ar produce output prea bogat sau modificari masive (utila pentru incepatori)

Exemplul 1: conectarea la serverul aflat pe statia cu adresa 1.2.3.4 si care asculta pe portul 4000, folosind username user1 si parola pass1:

```
user1@mycomputer:~$ mysql -h 1.2.3.4 -P 4000 -u user1 -p pass1
```

Exemplul 2: aceeasi comanda, insa fara a specifica parola explicit in linia de comanda, pentru a nu ramane ca atare in istoria comenzilor interpretorului:

```
user1@mycomputer:~$ mysql -h 1.2.3.4 -u user1 -p
Enter password:
```

In general, atunci cand nu specificam la pornire valoarea pentru o anumita optiune, ea va primi automat valoarea din oficiu. Spre exemplu, atunci cand apelam comanda mysql fara optiuni sau argumente, ea va incerca automat sa se conecteze la serverul MySQL care ruleaza pe aceeasi statie folosind username-ul utilizatorului curent, fara parola.

Comanda mysql poate primi si argumente. Primul argument este cosiderat automat a fi numele bazei de date pe care va opera interpretorul (ea se poate schimba ulterior).

Exemplu: conectarea la serverul local folosind username-ul user2 si parola pass2, setand ca baza de date implicita *studenti*:

```
user1@mycomputer:~$ mysql -u user2 -p pass2 studenti
```

1.4.4. Folosirea mysql in modul interactiv

1.4.4.1. Comenzi si instructiuni

Odata pornit in modul interactiv, *mysql* afiseaza un prompt si asteapta utilizatorul sa introduca linii de comanda, una cate una (prelucrarea fiecarei comenzi se efectueaza dupa ce utilizatorul apasa ENTER). Fiecare linie de comanda trebuie incheiata cu unul dintre terminatorii descrisi in sectiunea urmatoare, cel mai intalnit fiind ; (punct si virgula). Pot fi scrise mai multe comenzi pe aceeasi linie:

```
user1@mycomputer:~$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 120
Server version: 5.0.51a-24+lenny1-log (Debian)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+----------------------------+
| Database                   |
+----------------------------+
| information_schema         |
| mysql                      |
| test                       |
+----------------------------+
3 rows in set (0.00 sec)

mysql> select version(); select now();
+-----------------------+
| version()             |
+-----------------------+
| 5.0.51a-24+lenny1-log |
+-----------------------+
1 row in set (0.00 sec)

+---------------------+
| now()               |
+---------------------+
| 2009-06-14 17:39:18 |
+---------------------+
1 row in set (0.00 sec)
```

Comenzile disponibile in linia de comanda mysql se impart in doua categorii:

- intructiuni SQL, care sunt trimise catre server si prelucrate de catre acesta
- comenzi ale clientului mysql, care nu fac parte din standardul SQL. Unele dintre ele actioneaza doar local, fara a contacta serverul (ex: afisarea automata a eventualelor warning-uri, salvarea output-ului intr-un fisier), altele „imbraca” comenzi trimise serverului furnizand date utile (ex: afisarea parametrilor serverului si conexiunii)

1.4.4.2. Terminatori ai liniei de comanda

O linie de comanda poate fi incheiata in 4 feluri, dintre care doua sunt echivalente:
- cu ; sau \g (echivalente), ce determina executarea comenzii scrise pana in acel moment
- cu \G – efectul este ca valorile coloanelor fiecarei inregistrari sunt listate una sub alta. Acest terminator de linie este util atunci cand rezultatul contine prea multe coloane pentru afisarea in linia de comanda (prea „lat" pentru a incapea pe ecran),
- \c – determina abandonarea liniei de comanda curente si revenirea la promptul mysql>

Iata un exemplu de output cu doua coloane (data si ora curenta):

```
mysql> select curdate(), curtime()\g
+------------+-----------+
| curdate()  | curtime() |
+------------+-----------+
| 2009-06-14 | 17:46:40  |
+------------+-----------+
1 row in set (0.00 sec)
```

...si acelasi rezultat afisat in formatul alternativ oferit de terminatorul \G:

```
mysql> select curdate(), curtime()\G

*************************** 1. row ***************************
curdate(): 2009-06-14
curtime(): 17:46:42
1 row in set (0.00 sec)
```

Iesirea din utilitarul *mysql* se poate face cu CTRL-C, sau scriind in linia de comanda mysql *quit* sau *\q*.

1.4.4.3. Help in linia de comanda mysql

Exista doua categorii de help disponibile in linia de comanda mysql:
- help local, care afiseaza comenzile built-in ale interpretorului mysql (nu si cele SQL).
- help obtinut de la server, care ofera informatii despre instructiunile SQL si parametrii acestora

Help-ul local se obtine cu comanda (locala) help sau cu ? :

```
mysql> help

List of all MySQL commands:

Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear command.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter. NOTE: Takes the rest of the line as new
delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
```

```
help        (\h) Display this help.
nopager     (\n) Disable pager, print to stdout.
notee       (\t) Don't write into outfile.
pager       (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print       (\p) Print current command.
prompt      (\R) Change your mysql prompt.
quit        (\q) Quit mysql.
rehash      (\#) Rebuild completion hash.
source      (\.) Execute an SQL script file. Takes a file name as an argument.
status      (\s) Get status information from the server.
system      (\!) Execute a system shell command.
tee         (\T) Set outfile [to_outfile]. Append everything into given outfile.
use         (\u) Use another database. Takes database name as argument.
charset     (\C) Switch to another charset. Might be needed for processing binlog with
multi-byte charsets.
warnings    (\W) Show warnings after every statement.
nowarning   (\w) Don't show warnings after every statement.

For server side help, type 'help contents'
```

Help-ul de la server se poate obtine cu aceeasi comanda, insa urmata de numele categoriei sau comenzii despre care se doresc informatii ajutatoare:

```
mysql> help contents;

You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the following
categories:
   Account Management
   Administration
[...output omis...]

mysql> help Administration;

You asked for help about help category: "Administration"
For more information, type 'help <item>', where <item> is one of the following
topics:

   DESCRIBE
   FLUSH QUERY CACHE
   HELP COMMAND
   HELP STATEMENT
```

In output-ul de mai sus sunt prezente comenzile disponibile in acea categorie, pentru fiecare dintre ele existand cate o pagina ajutatoare. Exista help chiar si pentru comanda help:

```
mysql> help help command;

Name: 'HELP COMMAND'
Description:
Syntax:

mysql> help search_string
```

```
If you provide an argument to the help command, mysql uses it as a

search string to access server-side help from the contents of the MySQL
Reference Manual. The proper operation of this command requires that

[...output omis...]
```

1.4.5. Folosirea mysql in modul non-interactiv

1.4.5.1. Script-uri SQL

Clientul *mysql* este folosit in modul non-interactiv („batch mode") pentru a prelucra fisiere ce contin instructiuni SQL. Un astfel de fisier este denumit „script" sau „batch file" si reprezinta un fisier text ce contine o succesiune de instructiuni (respectand aceeasi sintaxa ca in modul interactiv). Instructiunile vor fi executate una cate una in ordinea in care apar in fisier, pana la epuizarea acestora sau pana la aparitia primei erori. Rezultatul executiei va fi afisat pe ecran.

Din acest punct de vedere, un script SQL seamana cu un program scris intr-un limbaj de programare, diferenta fiind ca, fata de un limbaj de programare, in care existau structuri de control al executiei, aici instructiunile se executa secvential.

Nota: *in SQL exista si alte elemente proprii limbajelor de programare: variabile, functii etc.*

ATENTIE! *A nu se confunda un script SQL care construieste o tabela si introduce date in ea cu un fisier ce contine datele respective. Spre exemplu, un fisier CSV sau text poate contine o lista de studenti (fiecare cu nume, prenume, mail etc), insa aceasta nu inseamna ca fisierul in cauza este un script SQL, deoarece el nu contine instructiuni SQL si deci nu poate fi executat ca atare de catre server.*

1.4.5.2. Comentarii in scripturi si instructiuni SQL

Intr-un script SQL (sau chiar in cadrul unei linii de comanda) pot fi incluse comentarii, pentru a oferi indicii/clarificari in privinta operatiilor efectuate. Comentariile incep cu un delimitator format dintr-o combinatie de caractere prestabilita si se incheie fie la sfarsitul liniei, fie la intalnirea delimitatorului de final, in functie de caz (vezi mai jos). MySQL suporta urmatoarele tipuri de comentarii:

- comentarii care se intind pe o linie sau o portiune a ei. Astfel de comentarii incep cu # sau cu -- (doua minus-uri urmate de un spatiu) si se incheie la sfarsitul liniei. Tot ceea ce se afla intre delimitatorul de inceput si finalul liniei este ignorat de catre interpretorul de comenzi
- comentarii care se pot intinde pe mai multe linii. Comentariile de acest tip debuteaza cu /* si se incheie cu */, cele doua delimitatoare putandu-se afla pe linii diferite. Tot ceea ce se afla intre delimitatoare este ignorat la executia comenzilor
- comentarii de gestionare a compatibilitatii. Astfel de comentarii incep cu /*! si se incheie cu */ . Continutul lor este executat de catre serverul MySQL in anumite conditii, insa la executia aceluiasi script pe un alt soft de baze de date portiunea respectiva este vazuta ca comentariu obisnuit si ignorata. Acest lucru permite scrierea de scripturi care contin extensii MySQL ale limbajului SQL dar care se pot executa fara erori pe alte softuri de tip DBMS. In plus, daca delimitatorul de inceput este urmat de o versiune de server, continutul „comentariului" nu va fi executat decat de catre serverele MySQL cu versiune mai mare sau

egala cu cea specificata, ceea ce permite rularea scripturilor scrise pentru versiuni mai noi de server MySQL pe variante mai vechi

Exemplu (zonele gri sunt comentarii):

```
-- crearea bazei de date principale
CREATE SCHEMA academie; # baza de date cu clase si studenti
/* creare tabele:
-  studenti
-  clase
*/
CREATE TABLE studenti /* conturile de utilizator sunt in alta tabela */ (Nume CHAR(50))
# portiunea gri a instructiunii se executa numai daca ruleaza pe srv MySQL > 5.0.21
SHOW /*!50021 FULL */ TABLES;
```

1.4.5.3. Prelucrarea unui script in modul non-interactiv

Exista doua modalitati de a rula un script SQL prin intermediul clientului mysql:

- se porneste clientul in modul interactiv si se foloseste comanda SOURCE pentru a citi si executa instructiunile din fisierul primit ca argument. Fisierul trebuie sa se afle in sistemul de fisiere al clientului:

```
mysql> SOURCE c:/sql/import.sql;
[...output omis...]
```

- se porneste clientul mysql in modul non-interactiv, adaugand in linia de comanda caracterul < si numele fisierului in care se afla instructiunile de executat:

```
/* conectarea la serverul local folosind username si parola, setarea bazei
de date test ca implicita si rularea scriptului import.sql */
mysql -ustudent -p test < import.sql
```

1.5. Elemente generale de limbaj SQL

1.5.1. Categorii de instructiuni/operatii

Distingem 3 categorii de instructiuni ale limbajului SQL, ce formeaza 3 portiuni ale limbajului:

- **DDL (Data Definition Language)**. Sunt instructiunile care permit manipularea structurii bazei de date (creare/stergere/vizualizare de baze de date, creare/modificare de tabele cu specificarea structurii acestora etc)
- **DML (Data Manipulation Language)**. Reprezinta setul de instructiuni pentru lucrul cu datele (introducere/extragere/modificare/stergere de informatie)
- **DCL (Data Control Language)**. Este format din acele instructiuni care permit gestionarea accesului la date (definirea de conturi, stabilirea de privilegii ale conturilor pe diversele baze de date/tabele/coloane)

Dat fiind faptul ca intr-o baza de date nu se pot introduce informatii pana nu se creeaza baza de date si tabelele componente, studiul SQL va incepe cu instructiunile ce compun DDL si abia apoi va trece la manipularea datelor.

1.5.2. Standarde si dialecte SQL

Precum s-a spus anterior, limbajul SQL a fost standardizat de catre ANSI in 1986. De atunci au mai existat alte cateva versiuni ale standardului SQL, fiecare ingloband noi facilitati ale limbajului (vezi http://en.wikipedia.org/wiki/SQL#Standardization). Desi exista un standard, produsele de tip DBMS prezente pe piata contin in general abateri sau adaugiri fata de el, implementand practic „dialecte" ale limbajului standard. Desigur ca portiunea comuna intre dialecte si standard este consistenta – conceptele fiind in orice caz comune - astfel incat cineva care stapaneste un dialect va putea folosi cu un minim efort limbajul standard, si invers. Extensiile proprietare ale limbajului sunt cele care fac deseori diferenta in cazul optimizarilor, si in acelasi timp fac delicata migrarea de la un tip de DBMS la altul (ex: de la MySQL la Oracle), caci, desi principiile sunt aceleasi, „the devil is in the details".

Nota: *atunci cand se doreste a se face referire la limbajul SQL standard se foloseste sintagma ANSI SQL.*

1.5.3. Nume de baze de date, tabele si coloane

1.5.3.1. Tipuri de nume si conditii de validitate

Fiecare entitate informationala de pe un server de baze de date (o baza de date, tabela, coloana etc) este identificata printr-un nume. In cadrul instructiunilor SQL, numele poate fi scris ca atare sau plasat intre caractere delimitatoare (cel mai intalnit fiind caracterul ` numit in engleza *back quote*). Plasarea intre delimitatori se efectueaza atunci cand numele contine caractere speciale care ar duce la o interpretare gresita a instructiunii SQL.

Nota: *a nu se confunda caracterul ` (back quote) cu ' (apostroful)! Back quote se afla pe tasta cu ~, aflata de obicei in stanga tastei 1 pe tastaturile cu layout English US.*

Pentru numele ce nu folosesc delimitatori, regulile sunt:
- pot contine litere, cifre, _ sau $
- nu pot fi formate numai din cifre (deoarece pot fi confundate de catre interpretor cu o reprezentare valida a unei valori intregi)
- pot incepe cu orice caracter legal, inclusiv cifra, insa unele nume care incep cu cifre pot fi gresit interpretate ca valori intregi (ex: 2e3 (care inseamna 2000), 0x1a (care inseamna 26) etc)

Atunci cand dorim ca numele sa contina exclusiv cifre sau si alte caractere decat cele de mai sus (ex: spatiu), va trebui sa includem numele intre delimitatori:

```
SELECT * FROM `tabela produse`
```

Nota: *cand serverul lucreaza in modul ANSI_SQL (compatibilitate cu standardul SQL), delimitatorii pot fi si ghilimele (").*

Includerea intre delimitatori elimina aproape toate restrictiile asupra numelui, cu urmatoarele exceptii:
- nu sunt permise caracterele cu codul 0 sau 255

- numele de baze de date si tabele nu pot contine punct (.), slash (/) sau backslash (\). Punctul este folosit in construirea numelor absolute (vezi mai jos), iar / si \ sunt folosite ca separator de directoare pentru caile in sistemul de fisiere

1.5.3.2. Conteaza daca folosim litere mici sau mari?

Serverul MySQL stocheaza pe hard disk datele sub forma de fisiere si directoare. De obicei, fiecarei baze de date ii va corespunde un director, iar unei tabele unul sau mai multe fisiere. In aceste conditii, va conta daca folosim litere mici sau mari doar in masura in care sistemul de fisiere in care sunt memorate datele este case sensitive. De obicei, in sistemul de operare Windows sistemele de fisiere sunt case insensitive, iar in Linux/Unix sunt case sensitive.

Aceasta proprietate a sistemului de fisiere afecteaza doar numele de baze de date si tabele. Numele coloanelor si ale altor elemente din bazele de date MySQL (indecsi, proceduri stocate, trigger-e etc) sunt case-insensitive.

Exemplu: odata ce am creat o tabela Produse, daca serverul ruleaza sub Windows ne vom putea referi la ea folosind numele PRODUSE sau produse, pe cand in Linux va trebui sa folosim numele exact atribuit la creare.

1.5.3.3. Nume absolute si relative

Pentru a ne referi la informatia dintr-o baza de date, este nevoie sa specificam toate elementele necesare pentru a identifica fara echivoc datele dorite: baza de date, tabela, coloana sau coloanele dorite si caracteristicile setului de inregistrari dorit. A specifica de fiecare data toate aceste informatii ar duce la interogari lungi si greu de urmarit, motiv pentru care au fost create modalitati de a referi date cu nume scurte.

Numele de tabele sau coloane functioneaza pe principiul cailor din sistemul de fisiere: reprezinta succesiuni de nume despartite printr-un separator specific, si care identifica in mod unic resursa dorita (spre exemplu, C:\Windows\System32 este o cale in sistemul de operare Windows, in care numele directoarelor ce compun calea sunt separate prin \). In sistemele de fisiere exista notiunea de cale absoluta si relativa; cea absoluta este o cale completa, ce porneste din radacina sistemului de fisiere, iar cea relativa are ca reper directorul curent.

In acelasi fel, intr-o baza de date relationala ne putem referi la tabele sau coloane folosind nume complete sau relative:

- un nume complet este de forma **bazadedate.tabela** (in cazul referirii la o tabela) sau **bazadedate.tabela.coloana** daca dorim sa ne referim la o coloana.
- un nume relativ reprezinta numele simplu al tabelei sau coloanei. El va fi automat considerat relativ la baza de date sau tabela curenta. (setarea bazei de date curente se realizeaza cu instructiunea USE – vezi instructiuni pentru operatii cu baze de date)

Exemplu: extragerea tuturor valorilor dintr-o tabela:

```
# specificand numele complet al tabelei (baza de date academie, tabela clase)
SELECT * FROM academie.clase

# folosind un nume relativ la baza de date curenta
USE academie
SELECT * FROM clase
```

1.6. BIBLIOGRAFIE

- MySQL Certification Study Guide
- MySQL Reference Manual: http://dev.mysql.com/doc/refman/5.1/en/