Aplicații Integrate pentru Întreprinderi

Laborator 2

11.10.2011



Utilizarea sistemului de gestiune pentru baze de date MySQL

Scopul laboratorului îl reprezintă folosirea sistemului de gestiune pentru baze de date MySQL în vederea creării de tabele dependente în cadrul unei baze de date precum şi popularea lor cu informații care să respecte constrângerile specificate la proiectarea bazei de date.

În principiu, în cadrul acestul laborator veți recapitula principalele noțiuni însuşite în cadrul cursului *Baze de Date I* din semestrul trecut.

Un **sistem de gestiune pentru baze de date**[1] (SGBD) este o colecție de programe care au drept scop precizarea structurii bazei de date, popularea ei, utilizarea, protecția datelor [1,2]. Altfel spus, SGBD este aplicația informatică complexă care însoțeşte baza de date având drept funcții definirea, construcția şi manipularea ei[2]. Rolul său este de a facilita accesul la date, fapt ce este realizat prin asigurarea transparenței reprezentării lor față de utilizatori în cadrul procesului de interacțiune cu ele. Din acest punct de vedere, SGBD este interfață între nivelul fizic, cel mai scăzut, şi programele prin care sunt transmise aplicației interogările de regăsire a datelor. Împreună cu datele[3] propriu-zise, acesta formează **sistemul de baze de date**[4].

Arhitectura asociată unei baze de date este reprezentată pe trei niveluri, având ca scop separarea aplicațiilor utilizatorului de informațiile stocate [2]:

- nivelul (schema) intern(ă) specifică structura de stocare fizică a datelor, folosind un model al datelor fizice[5];
- nivelul (schema) conceptual(ă) descrie structura bazei de date prin entități şi tipul de date asociat precum şi relațiile dintre acestea, precizând şi restricțiile de integritate;
- nivelul (schema) extern(ă) prezintă moduri de vizualizare ale bazei de date din perspectiva diferiților utilizatori care o accesează

[1] *eng.* Database Management System (DBMS)

[2] Procesul de definire implică specificarea structurii şi a tipurilor de date stocate, construcția referă popularea bazei de date în timp ce manipularea cuprinde mai multe operații pornind de la implementarea interogărilor pentru regăsirea informațiilor până la generarea de rapoarte.

[3] Datele se referă nu doar la colecțiile de informații stocate de către utilizator (între care există legături de mai multe tipuri 1-1, 1-m, m-m) ci şi **dicționarul de date** care conține meta-date: structura bazei de date, definițiile tipurilor de date (restricțiile de integritate), viziunile, indecşii precum şi fişierele anexe care conțin un context curent de lucru (parametrii de configurare) [3].

[4] Pe lângă sistemele de bază de date convenționale, au fost dezvoltate sisteme de baze de date evoluate cum ar fi SBD paralele, SBD mobile, SBD spațiale, SBD multimedia, dezvoltate pentru aplicații specializate, caracterizate prin funcționalități sau tipuri de date care dau specificul sistemului de baze de date respectiv.

[5] Sunt indicate aici atât modul de stocare cât şi modul de acces.

Obiectivele unui sistem de gestiune pentru baze de date sunt [1]:

- *asigurarea independenței datelor*, încât modificarea structurii de memorare a datelor sau schimbarea modului de acces la ele (independență fizică) precum şi a schemei conceptuale (independență logică[6]) să nu presupună rescrierea programelor de aplicație;
- *asigurarea unei redundanțe minime şi controlate* a datelor;
- *asigurarea unor facilități de utilizare a datelor* (concurența utilizatorilor, uşurința regăsirii datelor prin limbaje apropiate de cel natural, optimizarea timpilor de acces);
- *oferirea unui nivel de securitate ridicat* împotriva accesului neautorizat;
- *asigurarea integrității datelor* prin proceduri de validare şi printr-un protocol de refacere a bazei de date;
- *asigurarea partajabilității datelor*.

Toate sistemele de gestiune a bazelor de date sunt proiectate spre a oferi utilizatorilor un set de **funcții** [1]:

- *funcția de definire a datelor*, implementată prin intermediul unui limbaj de definire a datelor (DDL – Data Definition Language) ce permite specificarea structurii tabelelor unei baze de date prin precizarea atributelor, a tipurilor de date, a legăturilor dintre ele, a restricțiilor sau modalitățile de acces; schema bazei de date descrisă la acest nivel va fi memorată în dicționarul de date;
- *funcția de manipulare a datelor*, realizată printr-un limbaj de manipulare a datelor (DML – Data Manipulation Language) care permite crearea bazelor de date, adăugarea, modificarea sau ştergerea unor înregistrări, regăsirea de informații potrivit unor criterii şi ordonarea lor;
- *funcția de utilizare* ce oferă interfețe de regăsire a informațiilor, printr-un limbaj de definire a vizualizărilor[7] (VDL – View Definition Language); există mai multe tipuri de utilizatori:
  - beneficiari (utilizatori conversaționali) care vizualizează informațiile fără a folosi vreun limbaj de interogare a bazei de date;
  - programatori care utilizează limbajele de manipulare, exploatând baza de date prin mecanisme complexe;
  - administratorul bazei de date care ia decizii în privința funționării corespunzătoare a întregului sistem.
- *funcția de administrare a bazei de date* care permite configurarea diferiților parametri (performanță, securitate) în funcție de scopul în care se doreşte să se utilizeze baza de date.

Clasificarea sistemelor de gestiune a bazelor de date se poate face după mai multe criterii [2]: în funcție de modelul de date utilizat (relațional, rețea, ierarhizat, obiectual), numărul de utilizatori ce pot accesa baza de date simultan (monouser / multiuser), modul de stocare al datelor (centralizat / distribuit), costurile pentru licențiere (medii gratuite şi cu cost de licențiere scăzut / ridicat), limbajul pentru scrierea de aplicații (medii cu limbaj nativ şi cu limbaj gazdă).

[6] În cazul SBD distribuite, se vorbeşte şi despre o independență distributivă, care se referă la nodurile în care este memorată baza de date.
[7] Unele sisteme de baze de date implementează acest lucru într-un DDL care acceptă declarații specifice nivelului extern [2].

1. Instalare
1.1. Instalarea pe Unix
1.2. Instalarea pe Windows
2. Utilizare: conectare şi deconectare la baza de date
3. Tipuri de date
4. Proiectarea tabelelor şi operaţii de manipulare în bazele de date
5. Interogări
6. Folosirea de script-uri specifice

1. Instalare

MySQL reprezintă cea mai populară soluție *open-source* la nivel mondial pentru gestiunea bazelor de date. Vom lucra cu MySQL Community Server[8], disponibil la http://mysql.com/downloads/mysql/ atât pentru sisteme de operare Unix, cât şi Windows. Versiunea curentă este 5.5.16.

1.1 Instalarea pe Unix

Pentru sistemele de operare Unix există posibilitatea compilării surselor, însă sunt puse la dispoziție şi distribuții binare.

Vom selecta platforma Linux – Generic şi vom instala *Linux - Generic 2.6 (x86, 32/64-bit), Compressed TAR Archive*.

Trebuie să fiți conectați cu utilizatorul de sistem root

```
andrei@ubuntu:/$ su - root
```

sau în cazul în care nu aveți aceste drepturi, trebuie ca înaintea fiecărei comenzi să scrieți folosiți `sudo`.

Presupunem că vom extrage conținutul arhivei MySQL în `/usr/local`:

```
andrei@ubuntu:/$ sudo cd /usr/local
andrei@ubuntu:/$ sudo tar -xzvf mysql-5.5.16-linux2.6-i686.tar.gz
andrei@ubuntu:/$ sudo ln -s mysql-5.5.16-linux2.6-i686.tar.gz mysql
```

Se crează un grup `mysql` şi un utilizator `mysql` în cadrul grupului care vor fi proprietarii directorului de instalare MySQL:

```
andrei@ubuntu:/$ sudo groupadd mysql
andrei@ubuntu:/$ sudo useradd -g mysql mysql
andrei@ubuntu:/$ sudo chown -R mysql /usr/local/mysql
andrei@ubuntu:/$ sudo chgrp -R mysql /usr/local/mysql
```

Se instalează baza de date MySQL prin scriptul `mysql_install_db`:

```
andrei@ubuntu:/$ sudo /usr/local/mysql/scripts/mysql_install_db -
user=mysql [--basedir=/usr/local/mysql]

Installing MySQL system tables...
OK
Filling help tables...
OK
```

În cazul în care se obțin erori, verificați dacă aveți instalată altă versiune de MySQL şi dezinstalați-o, eventual instalați pachetul `libaio-dev`, dacă eroarea indică acest pachet ca fiind neinstalat:

```
andrei@ubuntu:/$ sudo apt-get install libaio1
```

[8] Pentru sistemele de operare Linux, trebuie instalat separat serverul şi clientul, în timp ce pentru sistemele de operare Windows, serverul şi clientul sunt incluse în acelaşi pachet.

Se modifică proprietarul fişierelor binare MySQL (`root`), asigurându-se ca utilizatorul `mysql/mysql` să aibă acces la informațiile bazei de date:

```
andrei@ubuntu:/$ sudo chown –R root /usr/local/mysql
andrei@ubuntu:/$ sudo chown –R mysql /usr/local/mysql/data
```

Serverul MySQL poate fi pornit prin scriptul `mysqld_safe`:

```
andrei@ubuntu:/$ sudo /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Nu uitați să precizați parola pentru utilizatorul `root` al bazei de date:

```
andrei@ubuntu:/$ sudo /usr/local/mysql/bin/mysqladmin –u root
password ******
```

Mai multe detalii cu privire la instalarea pe Unix se găsesc la adresa: http://onaxer.com/blog/blog/2011/03/08/install-mysql-5-5-on-ubuntu/.

De asemenea, sunt disponibile pentru instalare şi pachete tip .rpm, care pot fi instalate astfel:

```
rpm –install [options] filename.rpm
```

iar pentru distribuții de tipul Debian sau Ubuntu care necesită fişiere .deb, este necesară mai înainte convertirea între cele două tipuri de fişiere (în acest sens, poate fi folosit un utilitar de tip Alien[9]):

```
alien –scripts filename.rpm
dkpg –i filename.deb
```

În cazul că se obțin erori la convertirea între cele două tipuri de fişiere, programele pot fi instalate şi astfel:

```
andrei@ubuntu:/$ sudo apt-get install mysql-server
andrei@ubuntu:/$ sudo apt-get install mysql-client
```

Verificarea versiunii de MySQL instalată prin intermediul acestor comenzi poate fi verificată prin comanda:

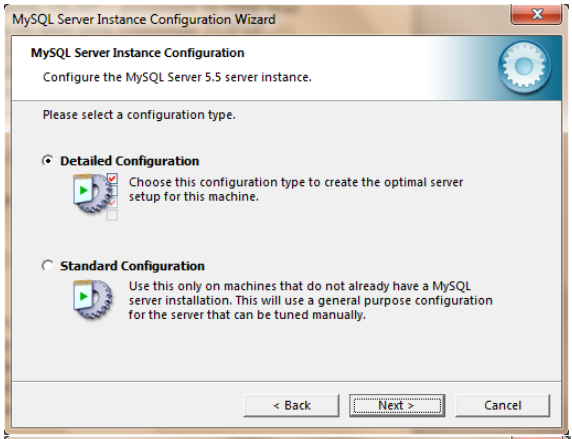
```
andrei@ubuntu:/$ mysql --version
mysql  Ver 14.14 Distrib 5.1.41, for debian-linux-gnu (i486) using
readline 6.1
```

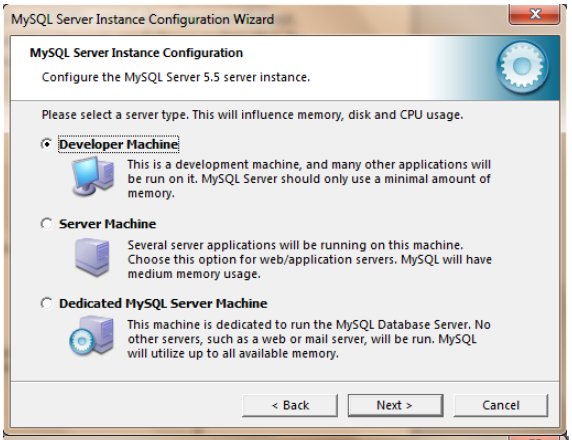
1.2 Instalarea pe Windows

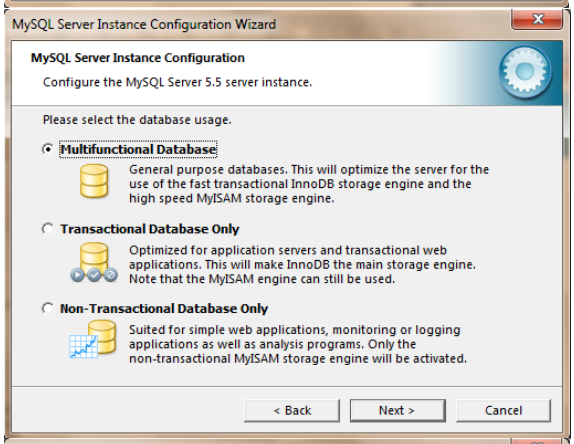
Pentru sistemele de operare Windows sunt puse la dispoziție atât installere .msi, cât şi arhive .zip.

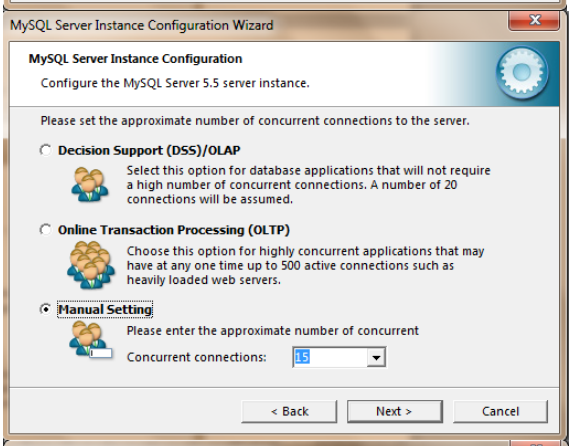
Instalarea oferă posibilitatea configurării serverului, specificându-se destinația aplicației (client, server [dedicat]), utilizarea bazei de date (multifuncțională, tranzacțională sau nu), conexiuni simultane în funcție de destinația bazei de date (sistem de suport decizie, OLAP – procesare în timp real a tranzacțiilor), permiterea conectării la baza de date prin conexiuni TCP/IP (dacă e necesară accesarea bazei de date din Internet, în afara maşinii locale, specificându-se şi portul de comunicație, implicit 3306), setul de caractere folosit, instalarea serverului ca serviciu Windows pornit automat, stabilirea politicilor de securitate prin specificare de utilizatori cu parole asociate.

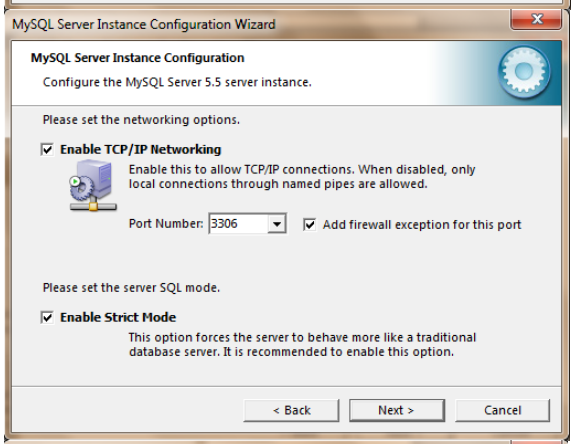
[9] În caz că un astfel de utilitar nu este încă instalat, acest lucru poate fi realizat prin comanda `apt-get install alien`

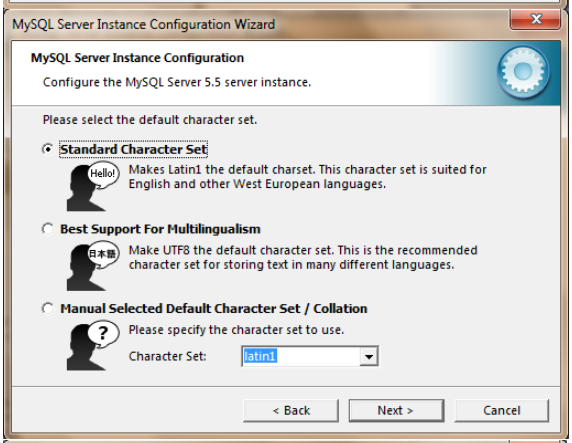

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select a configuration type.
Detailed Configuration
Choose this configuration type to create the optimal server setup for this machine.
Standard Configuration
Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.
< Back
Next >
Cancel

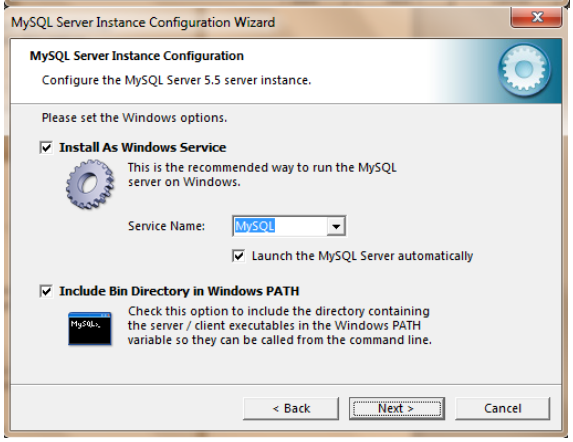

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select a server type. This will influence memory, disk and CPU usage.
Developer Machine
This is a development machine, and many other applications will be run on it. MySQL Server should only use a minimal amount of memory.
Server Machine
Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage.
Dedicated MySQL Server Machine
This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select the database usage.
Multifunctional Database
General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.
Transactional Database Only
Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.
Non-Transactional Database Only
Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the approximate number of concurrent connections to the server.
Decision Support (DSS)/OLAP
Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.
Online Transaction Processing (OLTP)
Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.
Manual Setting
Please enter the approximate number of concurrent
Concurrent connections:
< Back
Next >
Cancel

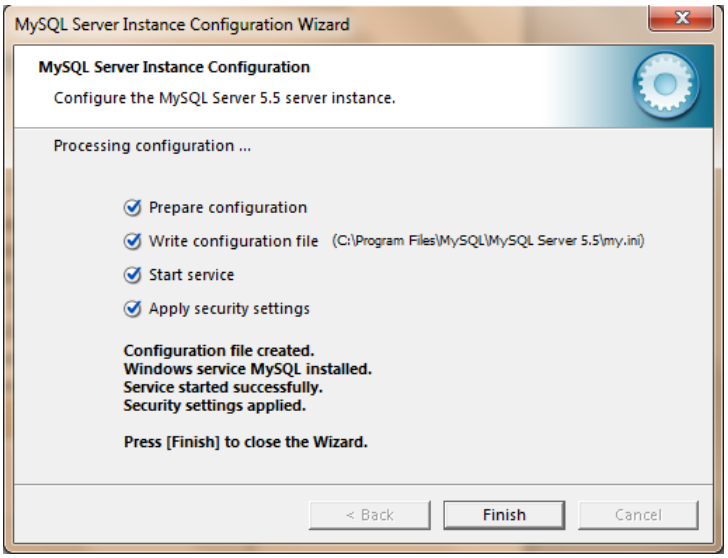

MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the networking options.
Enable TCP/IP Networking
Enable this to allow TCP/IP connections. When disabled, only local connections through named pipes are allowed.
Port Number: 3306
Add firewall exception for this port
Please set the server SQL mode.
Enable Strict Mode
This option forces the server to behave more like a traditional database server. It is recommended to enable this option.
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please select the default character set.
Standard Character Set
Hello!
Makes Latin1 the default charset. This character set is suited for English and other West European languages.
Best Support For Multilingualism
日本語
Make UTF8 the default character set. This is the recommended character set for storing text in many different languages.
Manual Selected Default Character Set / Collation
Please specify the character set to use.
Character Set: latin1
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the Windows options.
Install As Windows Service
This is the recommended way to run the MySQL server on Windows.
Service Name: MySQL
Launch the MySQL Server automatically
Include Bin Directory in Windows PATH
Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.
< Back
Next >
Cancel


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.
Please set the security options.
Modify Security Settings
root
New root password:
Enter the root password.
Confirm:
Retype the password.
Enable root access from remote machines
Create An Anonymous Account
This option will create an anonymous account on this server. Please note that this can lead to an insecure system.
< Back
Next >
Cancel

După rularea script-ului de configurare, este afişat rezultatul instalării.



2. Utilizare: conectare şi deconectare la baza de date

Conectarea la un server de baze de date se face prin comanda `mysql`, avand sintaxa:

```
mysql -h host -u username -p
```

unde

host – reprezintă maşina pe care rulează serverul de MySQL (poate fi pornit prin comanda `mysqld`, în caz că nu a fost instalat ca serviciu care să pornească automat); nu este necesară specificarea acestui parametru în condițiile în care serverul rulează pe aceeaşi maşină cu clientul;

`username` – reprezintă numele utilizatorului care se conectează la server;

În continuare, va fi solicitată parola asociată utilizatorului şi, în caz că aceasta este corectă, va apărea prompt-ul `mysql>` la care pot fi realizate interogări ale bazei de date.

Deconectarea de la serverul MySQL se face prin intermediul comenzii quit:

```
C:\Users\Andrei>mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates.
All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates.
Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql> quit
Bye

C:\Users\Andrei>
```

3. Tipuri de date

În MySQL pot fi folosite trei tipuri de date:

- numerice;
- dată calendaristică.
- şiruri de caractere;

Tipurile de date numerice sunt redate în tabelul de mai jos:

Tipuri de Date Numerice
BIT[(*M*)]
TINYINT[(*M*)] [UNSIGNED] [ZEROFILL]
BOOL, BOOLEAN
SMALLINT[(*M*)] [UNSIGNED] [ZEROFILL]
MEDIUMINT[(*M*)] [UNSIGNED] [ZEROFILL]
INT[(*M*)] [UNSIGNED] [ZEROFILL]
BIGINT[(*M*)] [UNSIGNED] [ZEROFILL]
FLOAT[(*M*,*D*)] [UNSIGNED] [ZEROFILL]
DOUBLE[(*M*,*D*)] [UNSIGNED] [ZEROFILL]
DOUBLE PRECISION[(*M*,*D*)] [UNSIGNED] [ZEROFILL] REAL[(*M*,*D*)] [UNSIGNED] [ZEROFILL]
FLOAT(*p*) [UNSIGNED] [ZEROFILL]
DECIMAL[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]
DEC[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]
NUMERIC[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]
FIXED[(*M*[,*D*])] [UNSIGNED] [ZEROFILL]

Observație.
`SERIAL` este un alias pentru `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE`

În schema de mai sus, M reprezintă numărul total de cifre pe care poate fi reprezentată o valoare de tipul numeric specificat, iar D numărul de cifre care urmează după virgulă în cazul numerelor reale.

În cazul în care se specifică valoarea `ZEROFILL` în cazul unor coloane conținând valori numerice, acestora li se adaugă automat valoarea `UNSIGNED`.

Informațiile de tip dată calendaristică pot avea următoarele tipuri:

Tipuri de Date Calendaristice
DATE
DATETIME
TIMESTAMP
TIME
YEAR[(2|4)]

Tipurile de date şir de caractere sunt redate în tabelul de mai jos:

Tipuri de Date Şir de Caractere
[NATIONAL] CHAR[(*M*)] [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
[NATIONAL] VARCHAR(*M*) [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
BINARY(*M*)
VARBINARY(*M*)
TINYBLOB
TINYTEXT [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
BLOB[(*M*)]
TEXT[(*M*)] [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
MEDIUMBLOB
MEDIUMTEXT [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
LONGBLOB
LONGTEXT [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
ENUM('*value1*','*value2*',...) [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
SET('*value1*','*value2*',...) [CHARACTER SET *charset_name*] [COLLATE *collation_name*]

4. Proiectarea tabelelor şi operații de manipulare în bazele de date

În proiectarea tabelelor trebuie să se aibă în vedere cateva reguli: structura tabelei trebuie să fie **normalizată**[10], deci să respecte o formă normală, astfel încât să se evite anomaliile la operațiile de actualizare în baza de date; pentru fiecare tabelă trebuie să se definească o **cheie primară** (formată din unul sau mai multe atribute) care să identifice în mod unic o înregistrare; tipul de date trebuie ales astfel încât să corespundă necesităților, evitând irosirea de memorie pe discul fizic, dar evitând situația în care anumite informații nu pot fi reținute datorită unei alocări insuficiente.

Normalizarea nu asigură buna construcție a bazei de date, deci nu este suficient să se verifice că fiecare tabelă aparține unei forme normale (de regulă, FN3).

O **supercheie** într-o relație $R=(a_1,...,a_n)$ este un set de atribute din R având proprietatea că nu există două n-tupluri t_1 şi t_2 în orice instanță r a lui R astfel ca $t_1(S)=t_2(S)$. Diferența dintre o cheie şi o supercheie este constă în faptul că întotdeauna cheia conține un număr minim de atribute. Un atribut al relației R se numeşte **prim** dacă este membru al unei chei primare din R şi este **nonprim** dacă nu este un atribut prim.

Au fost definite următoarele forme normale:

- *forma normală primară* (1NF) nu permite ca un atribut să aibă mai multe valori, atribute compuse sau combinații ale acesteia.
  Exemplu
  Fie schema de relație `clădire_şcolara = { cod, denumire, adresă, clasă }`. Aceasta nu face parte din 1NF, întrucât atributul clasă poate avea mai multe valori

[10] Normalizarea este procesul prin care schemele de relație nesatisfăcătoare sunt descompuse astfel încât să se obțină scheme de relație mai mici care să respecte proprietățile dorite. Codd propune trei forme normale, urmând ca cea mai bună definire să fie dată de Boyce-Codd.

Vom descompune această schemă de relație în altele două:

```
clădire_școlară = { cod, denumire, adresă } şi
clasă = { număr, etaj, dimensiune, cod_clădire_școlară }
```

- *forma normală secundară* (2NF) e satisfăcută dacă orice atribut nonprim este complet dependent funcțional de cheia primară din schema de relație în cauză. Spunem că o dependență funcțională X→Y este completă dacă oricare ar fi A un atribut din X prin eliminarea lui A din X se distruge dependența funcțională[11].

  Exemplu

  Fie schema de relație `angajat_proiect = { cod angajat, cod proiect, număr_ore, nume_angajat, denumire_proiect, locaţie_proiect }`, având cheia formată din `cod_angajat` şi `cod_proiect`. Se observă faptul că `nume_angajat` este complet dependent funcțional de `nume_angajat`, în timp ce `denumire_proiect` şi `locaţie_proiect` complet funcțional de `cod_proiect`.

  Vom descompune această schemă de relație în:

```
angajat_proiect = { cod_angajat, cod_proiect, număr_ore }
angajat = { cod_angajat, nume_angajat }
proiect = { cod_proiect, denumire_proiect, locatie_proiect }
```

- *forma normală 3* este îndeplinită de schemele de relație care satisfac condițiile formei normale secundare şi în plus nu există atribute noncheie dependente tranzitiv de o cheie primară. O dependență X→Y este tranzitivă dacă există un set de atribute Z care nu sunt un subset de chei din schema de relație astfel încât există relatiile X→Z şi Z→Y.

  Exemplu

  Fie schema de relație `elev = { cod, nume, prenume, strada, număr, oraş, judeţ }`. Se observă că județ, un atribut noncheie, este dependent tranzitiv față de cheia primară cod.

  Vom descompune această schemă de relație în altele două:

```
elev = { cod, nume, prenume, strada, număr, oraş }
localitate = { oraş, judeţ }
```

- *forma normală Boyce-Codd* respectă forma normală 3; aşadar orice schemă de relație care este în FNBC este şi în FN3, fără ca reciproca să fie valabilă în mod necesar. Condiția pe care trebuie să o îndeplinească tabela este că pentru orice dependență X→A, X este o supercheie în A, diferența față de FN3 fiind că aceasta permite ca A să fie non-prim dacă X nu este supercheie.

  Exemplu

  Fie schema de relație `magazin_apropiat = { nume, tip_magazin, magazin_apropiat }`. Candidate pentru chei sunt `(nume, tip_magazin)` sau `(nume, magazin_apropiat)`. Toate cele trei atribute sunt prime, deci schema face parte din FN3. Totuşi, `tip_magazin` depinde de `magazin_apropiat` care nu este supercheie, deci tabela nu respectă FNBC, ceea ce înseamnă că este susceptibilă de producerea anomaliilor la actualizare.

  Vom descompune această schemă de relație în altele două:

```
magazin_apropiat = { nume, magazin_apropiat }
specializare = { magazin apropiat, tip_magazin }
```

  Totuşi, această schemă permite ca pentru un `nume` să se specifice mai multe `magazine_apropiate` de acelaşi tip.

[11] Similar, spunem că o dependență funcțională X→Y este parțială dacă pot fi eliminate atribute din X fără ca dependența funcțională să fie afectată.

În MySQL, se pot vizualiza bazele de date existente prin comanda:

```
SHOW DATABASES;
```

În cadrul unei baze de date pot fi realizate operații imediat ce e selectată:

```
USE databasename;
```

Operația de acordare sau revocare de drepuri pentru un utilizator referitor la o tabelă dintr-o bază de date se face prin comanda:

```
GRANT ALL ON databasename.* TO 'username'@'host';
REVOKE INSERT ON *.* FROM 'username'@'host';
```

Tabelele din cadrul unei baze de date (care a fost selectată anterior) se pot vizualiza prin comanda:

```
SHOW TABLES;
```

Informații suplimentare despre structura unei tabele pot fi aflate astfel:

```
DESCRIBE tablename;
```

O tabelă în baza de date este creată prin `CREATE TABLE`:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [partition_options]
```

un exemplu de folosire fiind descris mai jos:

```
mysql> CREATE TABLE pet (
    ->                    name VARCHAR(20) NOT NULL,
    ->                    owner VARCHAR(20) NOT NULL,
    ->                    species VARCHAR(20) NOT NULL,
    ->                    sex CHAR(1),
    ->                    birth DATE NOT NULL,
    ->                    death DATE,
    ->                    PRIMARY KEY (name),
    ->                    FOREIGN KEY owner REFERENCES customer (id)
                          ON UPDATE CASCADE
    ->                   );
```

Au fost definite ca restricții de integritate o cheie primară (`name`) precum şi o cheie străină (`owner`) ce va fi actualizată în momentul în care valorile din tabela referită sunt modificate. La fel se poate proceda în cazul ştergerii. Alte acțiuni posibile în aceste situații, pe lânga `CASCADE` sunt şi `SET NULL`, `RESTRICT` (respingerea cererii), `NO ACTION`. Tot restricții de integritate sunt `INDEX`[12] şi `UNIQUE`

Structura unei tabele poate fi modificată folosind comanda `ALTER TABLE`:

```
mysql> ALTER TABLE pet RENAME TO animals;
mysql> ALTER TABLE pet CHANGE owner petowner VARCHAR(30);
mysql> ALTER TABLE pet ADD preferences VARCHAR(50) NOT NULL;
mysql> ALTER TABLE pet DROP sex;
mysql> ALTER TABLE pet DROP PRIMARY KEY;
```

Ştergerea tabelei (ca de altfel şi a bazei de date) se face prin comanda `DROP`:

```
DROP TABLE tablename;
DROP DATABASE databasename;
```

Adăugarea de informații poate fi realizată prin comanda `INSERT` sau prin comanda `LOAD DATA` (în acest caz sunt introduse date direct din fişier, specificându-se formatul datelor din fişier):

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE col_name=expr [, col_name=expr] ... ]
```

[12] Un index poate fi create şi prin comanda `CREATE INDEX` şi şters prin comanda `DROP INDEX`. Un tip special de index este `FULLTEXT`, proiectat special pentru căutarea în interiorul câmpurilor de tip text, conducând la operații mai rapide în comparație cu operatorul `LIKE`.

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [CHARACTER SET charset name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']]
    [IGNORE number LINES]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Exemple de populare a tabelei creată anterior prin comanda `CREATE TABLE` ar putea fi:

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet
    -> LINES TERMINATED BY '\r\n';
```

Modificarea înregistrărilor din tabelă se face prin comanda `UPDATE`:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Clauza `WHERE` poate folosi operatori aritmetici (`+, -, *, /, %`), operatori pentru comparație (`=, <> sau !=, <, <=, >, >=, BETWEEN, IN, IS NULL, IS NOT NULL, LIKE`[13], `REGEXP` sau `RLIKE`), operatori logici (`NOT` sau `!`, `AND` sau `&&`, `OR` sau `||`, `XOR`).

Clauza `ORDER BY` are ca efect ordonarea înregistrărilor şi poate primi drept parametri atributele `ASC` sau `DESC`.

Clauza `LIMIT` stabileşte `row_count` drept număr maxim de înregistrări întoarse ca rezultat al interogării.

Un exemplu de utilizare a comenzii poate fi consultat mai jos:

```
mysql> UPDATE pet SET species='rotweiller' WHERE name LIKE 'P%b%l';
```

Ştergerea unor tupluri din tabelă se face apelând comanda `DELETE`:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
    [PARTITION (partition_name,...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

În cazul tabelei oferite ca exemplu, este validă interogarea:

```
mysql> DELETE FROM pet WHERE species='hamster' AND sex='M';
```

În MySQL, variabilele sunt precedate de caracterul @.

```
mysql> SET @species='hamster';
mysql> DELETE FROM pet WHERE species=@species AND sex='M';
```

Nu se face distincția între minuscule şi majuscule în stabilirea identificatorului pentru o variabilă.

[13] Prin intermediul directivei `LIKE` adăugată la comanda `SELECT`, pot fi folosite diferite şabloane în scopul identificării unor şiruri de caractere care se potrivesc cu modele diverse, sintaxa fiind asemănătoare ca în cazul expresiilor regulate.

5. Interogări

Interogarea bazei de date se face prin comanda `SELECT`, ce are sintaxa:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name' [CHARACTER SET charset_name]
        export_options
        | INTO DUMPFILE 'file_name' | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

O interogare poate fi făcută pe mai multe tabele din baza de date între care există o relație (operație de join):

```
mysql> SELECT pet.name,
    -> (YEAR(date)-YEAR(birth)) - (RIGHT(date,5)<RIGHT(birth,5)) AS age,
    -> remark
    -> FROM pet INNER JOIN event
    ->   ON pet.name = event.name
    -> WHERE event.type = 'litter';
```

Există mai multe tipuri de legături (joins) ce se pot stabili între tabelele unei baze de date [4]:

- cross-join, rezultatul fiind produsul cartezian al tuplurilor din cadrul tabelelor implicate în operație; fie t_1 şi t_2 tabelele pentru care se realizeaza cross-join, având m şi respectiv n înregistrări; rezultatul interogării conține m x n înregistrări;
- inner-join, care include doar rezultatele care au corespondență în toate tabelele;
- outer-join, care include înregistrările comune unei laturi a legăturii, completând câmpurile necompletate cu `NULL`;
  În acest caz, se pot folosi cuvintele cheie `LEFT | RIGHT JOIN`, specificând partea relației care va include toate rezultatele.

```
mysql> SELECT pet.name, event.remark
    -> FROM pet LEFT JOIN event
    -> ON pet.name = event.name
    -> WHERE event.type = 'litter';
```

Rezultatul poate fi acesta

```
+------------+------------+
| Name       | Remark     |
+------------+------------+
| Tom        | clean      |
| Jerry      | NULL       |
| Croco      | dirty      |
+------------+------------+
```

În cazul în care numele câmpului de referință este acelaşi în ambele tabele se poate folosi clauza `USING` în loc de `ON`.

- self-join, care implică duplicarea tabelei prin utilizarea de alias-uri;

- unions care presupune adăugarea tuturor înregistrărilor din tabele pentru a determina suma compozită a acestora.

```
mysql> CREATE TEMPORARY TABLE cats
    -> SELECT * FROM pets WHERE species='cat';
mysql> CREATE TEMPORARY TABLE dogs
    -> SELECT * FROM pets WHERE species='dog';
mysql> SELECT name, owner FROM cats
    -> UNION
    -> SELECT name, owner FROM dogs;
```

Se poate identifica numărul de înregistrări dintr-o bază de date care corespund unui anumit criteriu:

```
mysql> SELECT owner, COUNT(*) AS numberofpets
    -> FROM pet GROUP BY owner HAVING numberofpets >= 2 ;
```

Clauza `GROUP BY` permite gruparea înregistrărilor pe baza valorii unui anumit câmp, fiecare grup creat fiind considerat ca fiind o singură înregistrare.

Clauza `HAVING` funcționează similar cu clauza `WHERE` cu excepția faptului că se aplică doar pentru funcții de grup.

Pot fi utilizate şi sub-interogări, în interiorul unei clauze `WHERE` sau `HAVING` (cu operatori de comparare sau operatori logici, cu testul de apartenența `IN` sau cu testul boolean `EXISTS`) sau în cadrul unei clauze `FROM` în cazul `UPDATE` şi `DELETE`.

```
mysql> SELECT name FROM pets
    -> WHERE name = (SELECT name FROM events WHERE type = 'litter');
```

O **rutină stocată** (fie funcție, fie procedură) are trei componente:

- parametrii de intrare (argumentele);
- parametrii de ieşire (valorile returnate);
- corpul rutinei care conține instrucțiunile ce trebuie executate.

Pentru a putea crea o rutină, un utilizator trebuie să aibă privilegiul `CREATE ROUTINE`, iar pentru a o executa, privilegiul `EXECUTE`.

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE get_owner(
    ->     IN event_type CHAR,
    ->     OUT owner_name VARCHAR(20)
    -> )
    -> BEGIN
    -> SELECT p.owner INTO owner_name FROM pets p, events e
    -> WHERE p.name = e.name and e.type=event_type;
    -> END //
mysql> CALL get_owner ('litter', @var);
mysql> SELECT @var;
```

O funcție similară ar fi avut antetul:

```
mysql> DELIMITER //
mysql> CREATE FUNCTION get_owner(event_type CHAR)
    ->     RETURNS VARCHAR(20)
```

6. Folosirea de script-uri specifice

Câteodată este util, în special în caz de testare, ca toate comenzile să fie grupate în cadrul unui fişier şi executate secvențial.

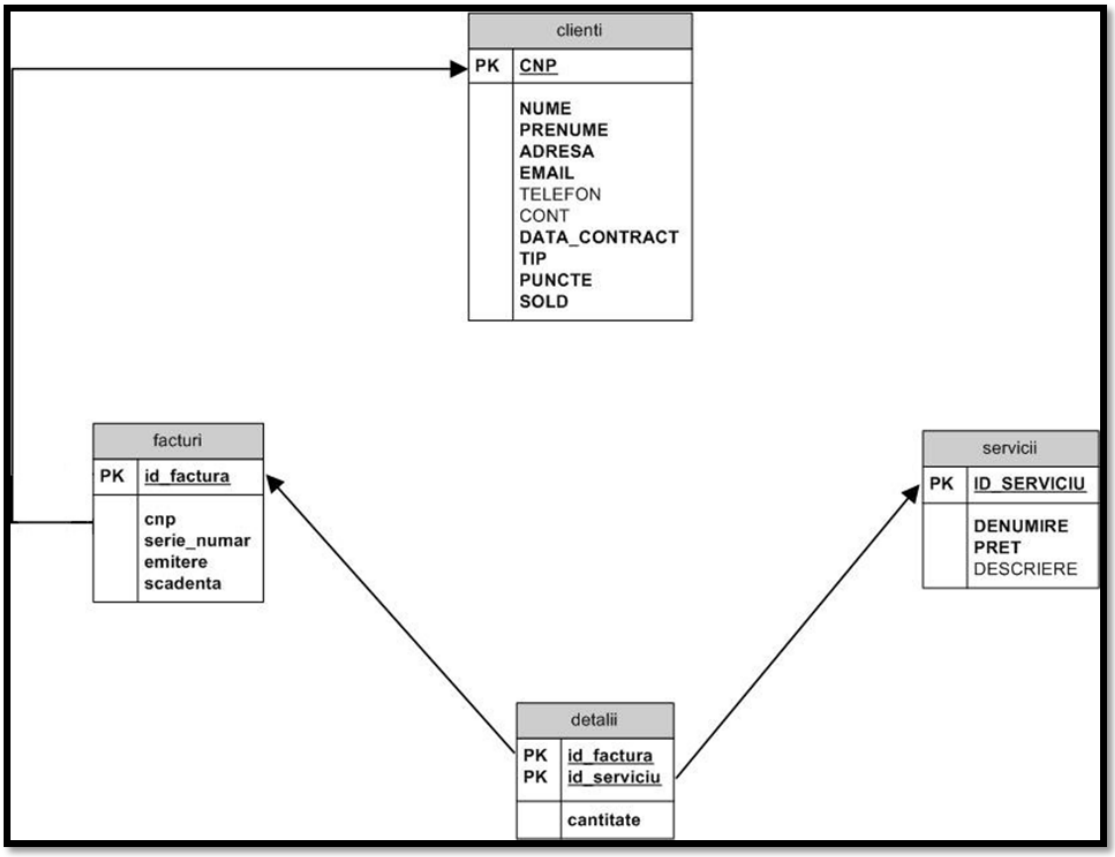
Se poate folosi fie operatorul de indirectare sau se pot folosi comenzile `source` sau \.:

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

```
mysql> source filename;
mysql> \. filename
```

Aplicație de Laborator

Se doreşte dezvoltarea unei aplicații integrate pentru întreprinderi destinată implementării unui serviciu de facturare la o companie, folosind o bază de date având următoarea schemă conceptuală:



(4p) 1. Să se creeze o bază de date cu tabelele `clienti`, `facturi`, `detalii`, `servicii` folosind schema conceptuală de mai sus.

Se vor specifica următoarele restricții:

- câmpurile nu pot fi vide, cu excepția atributelor `clienti.telefon, clienti.cont, servicii.descriere`;
- `clienti.email` are forma `'%@%.%'` unde `'%'` are semnificația unui şir de caractere oarecare;
- `clienti (tip)` poate avea doar valorile `('BZ', 'AR', 'AU', 'PL')`;
- `servicii (pret)` este strict pozitiv;
- `detalii (cantitate)` este strict pozitiv.

(1p) 2. Să se populeze corespunzător tabelele.
(1p) 3. Să se majoreze cu 20% toate prețurile serviciilor mai mici decât 350 ron.
(2p) 4. Să se creeze o procedură stocată pe server `valoare_factura` care primeşte un parametru de tip numeric `serie_numar` şi care calculeaza valoarea facturii.
(1p) 5. Folosind procedura stocată pe server `valoare_factura`, să se afişeze valorile tuturor facturilor înregistrate în sistem, grupate pe clienti.
(1p) 6. Să se afişeze suma tuturor facturilor pentru clientul cu cnp 1220713379378.
(1p) 7. Să se identifice lista celor mai profitabile servicii pe baza înregistrărilor din baza de date, ordonate în funcție de profitul adus.

Bibliografie

[1] Gheorghe Sabău, Vasile Avram, Ramona Bologa, Mihaela Muntean, Marian Dârdală, Răzvan Bologa – *Baze de Date*, Editura Matrix Rom, Bucureşti, 2008

[2] Dorin Cârstoiu – *Baze de Date*, Editura Matrix Rom, Bucureşti, 2009

[3] Manole Velicanu, Ion Lungu, Iuliana Botha, Adela Bâra, Anda Velicanu, Emanuil Rednic – *Sisteme de Baze de Date Evoluate*, Editura ASE, Bucureşti, 2009

[4] Vikram Vaswani – *MySQL. Utilizarea şi administrarea bazelor de date MySQL*, traducere de Cristian Alexe Dumitrescu, Editura Rosetti Educational, Bucureşti, 2010