MySQL

Introducere în limbajul SQL

MySQL este un sistem de gestiune a bazelor de date. O bază de date este un set de date centralizat şi structurat, stocat într-un computer şi furnizează facilități de adăugare, modificare şi ştergere a datelor la cerere, permite transformarea datelor în informații şi este adesea condusă de un Administrator al Bazei de date.

Serverul MySQL controlează accesul la date, pentru a garanta faptul că mai mulți utilizatori pot lucra simultan cu acestea. MySQL este un server multi-user şi multi-thread (mai multe fire de execuție). MySQL utilizează SQL (**S**tructured **Q**uery **L**anguage), limbaj standardizat de interogare a bazelor de date.

Principalul avantaj al MySQL este faptul că este open-source şi poate fi utilizat împreună cu limbajul de programare PHP.

Pagina de unde aflăm cele mai importante informații despre MySQL este: http://www.mysql.com.

În SQL comenzile pot fi grupate în următoarele categorii:

- Limbajul de interogare permite extragerea informațiilor din tabelele bazei de date cu ajutorul comenzii SELECT.
- Limbajul de manipulare a datelor (DML – **D**ata **M**anipulation **L**anguage) permite modificarea conținutului tabelelor. Comenzile DML sunt: INSERT, UPDATE, DELETE
- Limbajul de definire a datelor (DDL – **D**ata **D**efinition **L**anguage) permite definirea structurii tabelelor care compun baza de date. Comenzile DDL sunt: CREATE, ALTER, DROP, RENAME, TRUNCATE.
- Comenzi de control al tranzacțiilor (TC – **T**ranzaction **C**ontrol). Comenzile TC sunt: COMMIT, ROLLBACK şi SAVEPOINT.
- Limbajul de control al datelor (DCL – **D**ata **C**ontrol **L**anguage) permite definirea şi modificarea drepturilor utilizatorilor asupra bazei de date. Comenzile DCL sunt: GRANT şi REVOKE

Descrierea pe scurt a comenzilor

INSERT – permite introducerea unor noi linii cu înregistrări într-un tabel. Sintaxa este:

INSERT INTO NumeTabel (ListăColoane) VALUES (ListăValori)

O condiție pentru a funcționa este ca numărul de coloane, ordinea şi tipul lor să corespundă cu numărul, ordinea şi tipul valorilor.

Exemplu Următorul exemplu introduce un rând în tabelul elev.

```
INSERT INTO elev (id, nume, prenume, medie, localitate)
VALUES (10, 'David', 'Ana', 10, 'Targoviste')
```

UPDATE – permite modificarea uneia sau mai multor înregistrări dintr-un tabel. Sintaxa este:

```
UPDATE NumeTabel SET coloană1=valoare1,
                     coloană 2=valoare2,
                     ....
                 WHERE condiție
```

Exemplu Următorul exemplu va modifica din tabelul elev conținutul coloanei prenume, prin adăugarea prenumelui Maria înregistrării cu id=10.

```
UPDATE elev SET prenume='Ana Maria' WHERE id=10
```

DELETE – permite ştergerea uneia sau mai multor linii dintr-un tabel. Sintaxa este:

DELETE FROM NumeTabel WHERE condiție

Exemplu Următorul exemplu va şterge toate rândurile din tabel în care elevii au media mai mică decât 5

```
DELETE FROM elev WHERE medie<5
```

CREATE – permite crearea unui tabel. Sintaxa este:
CREATE TABLE NumeTabel (coloană1 tip1, coloană2 tip2, ... coloanăn tipn)
Exemplu Următorul exemplu va crea un tabel cu numele elev şi coloanele id, nume, prenume, medie şi localitate.
CREATE TABLE elev (id NUMBER(3), nume VARCHAR2(10), prenume VARCHAR2(10), medie NUMBER(5,2), localitate VARCHAR2(15))

Dacă dorim ca utilizatorii tabelei să introducă obligatoriu date în anumite coloane, utilizăm restricția NOT NULL la crearea tabelei, ca în exemplul de mai jos. În cazul în care utilizatorii nu completează un anumit câmp din tabel, se va introduce automat o valoare NULL care este echivalentă cu o valoare nedefinită.
Exemplu Următorul exemplu va crea tabelul elev cu coloanele id, nume, prenume de tip NOT NULL şi coloanele medie şi localitate.
CREATE TABLE elev (id NUMBER(3) NOT NULL, nume VARCHAR2(10) NOT NULL, prenume VARCHAR2(10) NOT NULL, medie NUMBER(5,2), localitate VARCHAR2(15))

ALTER – permite modificarea structurii unui tabel prin adăugarea (ADD), modificarea (MODIFY) sau ştergerea (DROP) unei coloane, crearea (ADD CONSTRAINT), modificarea (DISABLE/ENABLE) sau ştergerea (DROP CONSTRAINT) unor constrângere asupra unei coloane.
Exemplu Următorul exemplu va adăuga o coloană cu numele şcoala, tabelului elev şi va modifica dimensiunea coloanei prenume de la 10 la 15.
ALTER TABLE elev
ADD scoala VARCHAR2(20)
MODIFY prenume VARCHAR2(15)

DROP – permite ştergerea unei strucuturi a bazei de date.
Exemplu Următorul exemplu va şterge tabelul elev
DROP TABLE elev

RENAME– permite redenumirea unui tabel.
Exemplu Următorul exemplu va schimba numele tabelului elev în elev1.
RENAME elev TO elev1

TRUNCATE – permite ştergerea întregului conținut al unui tabel
Exemplu Următorul exemplu va şterge tot conținutul tabelului elev.
TRUNCATE TABLE elev

SAVEPOINT – permite definirea unui punct de revenire. Sintaxa este:
SAVEPOINT NumePunctDeRevenire

ROLLBACK– permite revenirea la un punct de revenire definit anterior. Sintaxa este:
ROLLBACK TO NumePunctDeRevenire

COMMIT – permite transformarea unei tranzacții într-o tranzacție permanentă. Renunțarea la modificările efectuate în cadrul unei tranzacții se efectuează cu comanda ROLLBACK

GRANT – permite acordarea unor drepturi unui anumit utilizator.
Exemplu În următorul exemplu se va acorda dreptul de creare a unui tabel utilizatorului 'ion'.
GRANT CREATE TABLE TO ion

REVOKE– permite revocarea unor drepturi unui anumit utilizator.
Exemplu În următorul exemplu se va revoca dreptul de creare a unui tabel utilizatorului 'ion'.
REVOKE CREATE TABLE FROM ion

Interogarea tabelelor

Cu ajutorul comenzii SELECT putem realiza două tipuri de operații: de filtrare a liniilor şi de alegere a anumitor coloane ce vor fi afişate. Sintaxa comenzii SELECT este:

SELECT expresii
FROM tabela

Pentru a afişa toate datele din tabela 'utilizatori', scriem: SELECT * FROM utilizatori
SELECT nume, prenume FROM utilizatori
Exemplul de mai sus afişează doar conținutul a două coloane din tabel.

Distinct – permite eliminarea liniilor duplicat
Exemplu Acest exemplu va afişa doar județele distincte din tabela utilizatori
SELECT DISTINCT judet FROM utilizatori

Clauza **WHERE** – permite filtrarea datelor în funcție de criteriile solicitate.
Exemplu În următorul exemplu se vor afişa doar numele şi prenumele celor care locuiesc în Târgovişte sau Bucureşti din tabela utilizatori.
SELECT nume, prenume FROM utilizatori
WHERE localitate='Targoviste' OR localitate='Bucuresti'

Operatorul **LIKE** – permite verificarea unui şir de caractere şi afişează *true* dacă şirul conține un anumit caracter şi *false* în caz contrar. Verificarea se realizează cu ajutorul caracterului de subliniere (underscore _) care ține locul unui singur caracter şi a caracterului procent (%) care ține locul la zero sau mai multe caractere.
Exemplu În următorul exemplu vom afişa toate persoanele al căror nume conține litera m.
SELECT nume, prenume FROM utilizatori
WHERE lower(prenume) LIKE '%m%'
OBS *lower* - converteşte caracterele alfanumerice din şir în litere mari.

Sortarea datelor

Căutarea datelor într-un tabel se realizează mult mai uşor dacă acel tabel a fost ordonat alfabetic, crescător sau descrescător.

Clauza **ORDER BY** – permite ordonarea datelor dintr-un tabel. După această clauză se enumeră coloanele sau expresiile după care se vor ordona liniile unui tabel.
Exemplu Acest exemplu va afişa numele şi media elevilor din tabela elev în ordine alfabetică.
SELECT nume, medie FROM elev ORDER BY nume
Dacă dorim ca afişarea să fie în ordine descrescătoare a mediilor, vom adăuga DESC:
SELECT nume, medie FROM elev ORDER BY medie DESC

ROWNUM – returnează numărul de ordine al unei linii într-o tabelă.
Exemplu Următorul exemplu va afişa datele persoanelor cu mediile cele mai mari din tabela elev, în ordinea descrescătoare a mediilor.
*SELECT * from (select * FROM elev ORDER BY nume DESC)*
WHERE ROWNUM<=5

Alias-ul unei coloane – se utilizează atunci când dorim ca la afişare, capul de tabel să conțină alt text sau să nu se folosească doar majuscule. Alias-ul se introduce în clauza SELECT imediat după numele coloanei unde dorim să îl urilizăm. Pentru a introduce un alias avem două posibilități:

1. utilizăm comanda AS - pentru un alias format dintr-un singur cuvânt
2. utilizăm ghilimele - pentru un alias format din mai multe cuvinte

Exemplu Următorul exemplu va afişa numele şi prenumele elevilor din tabela elev pe o singură coloană şi media mărită cu un punct.
SELECT nume ||' '||prenume "Nume şi prenume", medie+1 AS MedieFinala
FROM elev

Funcții

În funcție de modalitatea de operare a lor, funcțiile se împart în două categorii:

1. ***funcții singulare*** – care operează asupra unei singure înregistrări la un moment dat. Ele se pot împărți în funcții care operează asupra şirurilor de caractere, funcții numerice, funcții pentru manipularea datelor calendaristice, funcții de conversie a unui tip de date în altul, funcții de uz general
2. ***funcții de grup*** – care operează asupra unui grup de înregistrări şi returnează o singură valoare pentru un grup sau un set de linii dintr-un tabel. Principalele funcții de grup sunt: MIN, MAX, SUM, AVG, COUNT.

1. Funcțiile singulare se pot clasifica în:

a) funcții care acționează asupra ***şiruri****lor* ***de caractere***: LOWER (transformă un şir de caractere din litere mici în litere mari), UPPER (transformă un şir de caractere din litere mari în litere mici), INITCAP (transformă în majusculă prima literă din fiecare cuvânt al unui şir), CONCAT (concatenează două şiruri de caractere), SUBSTR (extrage un şir începând de la o anumită poziție);

b) funcții care acționează asupra ***numere***lor: ABS (returnează valoarea absolută a unui număr), SQRT (returnează rădăcina pătrată a unui număr), MOD (returnează restul împărțirii unui număr la alt număr), ROUND (rotunjeşte valoarea unui număr la un număr specificat de cifre);

c) funcții care acționează asupra ***date****lor* ***calendaristice***: SYSDATE (returnează data şi ora curentă a serverului bazei de date), ADD_MONTHS (adaugă un număr de luni la data curentă), MONTHS_BETWEEN (returnează numărul de luni dintre două date calendaristice);

d) funcții de conversie: TO_CHAR (transformă o dată calendaristică sau un număr în şir de caractere), TO_DATE (transformă un şir de caractere în dată calendaristică), TO_NUMBER (transformă un şir de caractere într-un număr).

2. Funcțiile de grup

MIN – permite aflarea celei mai mici valori a unei expresii, ignorând valorile nule.

Exemplu Următorul exemplu va afişa media cea mai mică a elevilor din tabela elev.

SELECT min(medie) FROM elev

MAX – permite aflarea celei mai mari valori a unei expresii, ignorând valorile nule.

Exemplu Următorul exemplu va afişa media cea mai mare a elevilor din tabela elev.

SELECT max(medie) FROM elev

SUM – permite aflarea sumei valorilor unei expresii şi se aplică numai coloanelor care conțin valori numerice.

Exemplu Următorul exemplu va afişa numărul de absențe ale elevilor navetişti din tabela elev.

SELECT sum(absente) FROM elev WHERE navetist='DA'

AVG – permite aflarea mediei aritmetice a valorilor numerice dintr-o coloană, ignorând valorile nule.

Exemplu Următorul exemplu va afişa media clasei de elevi promovați din tabela elev.

SELECT avg(medie) FROM elev WHERE medie>=5

COUNT – permite aflarea numărului de valori, ignorând valorile nule.

Exemplu Următorul exemplu va afişa numărul de elevi din tabela elev.

SELECT count(id) FROM elev

Dacă dorim afişarea numărului de rânduri completate parțial din tabelul elev, utilizăm *SELECT count(*) FROM elev*

Crearea şi utilizarea unui tabel

CREATE TABLE este declarația utilizată pentru a crea un tabel în baza de date.
Sintaxa este:
CREATE TABLE nume_tabel (nume_coloana1 tip_de_date, nume_coloana2 tip_de_date, …,
nume_coloanan tip_de_date)

Un exemplu de utilizare al acestei declarații ar putea fi dat de crearea tabelului utilizatori utilizat în cadrul portalului pentru e-learning prezentat în capitolele anterioare.

```
CREATE TABLE `utilizatori` (
  `id` int(11) NOT NULL auto_increment,
  `utilizator` char(60) collate latin1_general_ci NOT NULL,
  `parola` char(60) collate latin1_general_ci NOT NULL,
  `nume` char(30) collate latin1_general_ci NOT NULL,
  `prenume` char(30) collate latin1_general_ci NOT NULL,
  `sex` char(10) collate latin1_general_ci NOT NULL,
  `initiala` char(1) collate latin1_general_ci NOT NULL,
  `telefon` char(10) collate latin1_general_ci NOT NULL,
  `email` char(30) collate latin1_general_ci NOT NULL,
  `cursuri` char(30) collate latin1_general_ci NOT NULL,
  `localitate` char(30) collate latin1_general_ci NOT NULL,
  `adresa` char(60) collate latin1_general_ci NOT NULL,
  `cod` char(10) collate latin1_general_ci NOT NULL,
  `judet` char(15) collate latin1_general_ci NOT NULL,
  `tara` char(60) collate latin1_general_ci NOT NULL,
  `scoala` char(30) collate latin1_general_ci NOT NULL,
  UNIQUE KEY `id` (`id`),
  UNIQUE KEY `utilizator` (`utilizator`)
)    ENGINE=MyISAM    DEFAULT    CHARSET=latin1    COLLATE=latin1_general_ci
AUTO_INCREMENT=16 ;
```

Setarea parolei pentru baza de date:

```
SET PASSWORD FOR 'root'@'localhost' = PASSWORD( '******' )
```