3. INSTRUCTIUNI SQL DE MANIPULARE A DATELOR

Cuprins

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

3.1. Tipuri de operatii in lucrul cu date

Manipularea datelor presupune urmatoarele categorii de operatii:

- introducerea de informatie intr-o tabela. Instructiunea SQL folosita in acest scop este **INSERT**. MySQL dispune in plus de instructiunea **REPLACE**, care nu face parte din standardul SQL
- modificarea valorilor din una sau mai multe tabele. Se realizeaza cu instructiunea **UPDATE**
- stergerea de inregistrari din una sau mai multe tabele. Se efectueaza cu ajutorul instructiunii **DELETE**
- extragerea de informatie din una sau mai multe tabele. Instructiunea folosita in acest caz este **SELECT**

Exista o diferenta majora intre instructiunea SELECT si celelalte: rezultatul lui SELECT este un set de inregistrari obtinute dintr-una sau mai multe tabele, pe cand restul instructiunilor de manipulare a datelor nu intorc inregistrari, ci doar efectueaza modificari raportand eventual numarul de linii afectate.

Nota: *desi instructiunile UPDATE, DELETE si SELECT pot opera asupra mai multor tabele simultan, materialul de fata trateaza cazul lucrului cu o singura tabela. Operarea cu mai multe tabele va fi prezentata in cadrul capitolului despre join-uri.*

3.2. Instructiunea INSERT

3.2.1. Forme ale instructiunii

Aceasta instructiune SQL este folosita pentru introducerea uneia sau mai multor inregistrari intr-una sau mai multe tabele SQL. Instructiunea are mai multe forme, clientul beneficiind de flexibilitate sub urmatoarele aspecte:

- cu aceeasi instructiune INSERT pot fi introduse una sau mai multe inregistrari
- in cadrul instructiunii INSERT putem specifica setul complet de valori pentru fiecare inregistrare (cate o valoare pentru fiecare coloana din definitia tabelei) sau putem preciza valoarea doar pentru un set discret de coloane, celelalte ramanand cu valoarea lor default.
- putem introduce inregistrari noi fara a specifica nicio valoare pentru coloane, in aceste conditii toate coloanele ramanand cu valorile default
- pot fi introduse intr-o tabela inregistrarile ce constituie rezultatul unei interogari de tip SELECT

3.2.2. Introducerea unei inregistrari prin specificarea setului complet de valori

Forma generala a instructiunii INSERT in acest caz este:

```
INSERT INTO NumeTabela VALUES(val1,val2,...,valN);
```

Trebuie specificate valori pentru toate coloanele tabelei si in ordinea in care apar ele in definitia tabelei.

Aceasta forma prezinta cateva dezavantaje:

- numarul de valori specificat trebuie sa fie intotdeauna egal cu numarul de coloane ale tabelei; daca se sterge sau se adauga o coloana, instructiunea INSERT trebuie ajustata (incomod in cazul in care clientul SQL este o aplicatie deja scrisa si care in consecinta ar trebui modificata)

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

- daca se schimba ordinea coloanelor in tabela, instructiunea INSERT trebuie de asemenea modificata (mai exact, ordinea valorilor din lista)

Exemplu: crearea unei tabele Produse si introducerea a doua inregistrari; dupa modificarea structurii tabelei nu mai functioneaza acelasi format al instructiunii de introducere de inregistrare:

```
CREATE TABLE Produse (Denumire VARCHAR(100), Pret TINYINT);
INSERT INTO Produse VALUES('Mere', 5.2);
INSERT INTO Produse VALUES('Caise', 6.0);
ALTER TABLE Produse ADD COLUMN Descriere VARCHAR(500);
INSERT INTO Produse VALUES('Prune', 8); # eroare, deoarece nr de valori si cel de coloane difera
```

3.2.3. Specificarea valorilor pentru o parte a coloanelor

Deseori nu dorim ca, la introducerea unei inregistrari, sa specificam toate valorile pentru coloane. Iata cateva scenarii:

- o coloana a unei tabele contine informatie ce va fi adaugata dupa crearea inregistrarii (ex: o tabela Catalog in care introducem inregistrarea corespunzatoare unui student, insa coloana de medie generala va fi populata ulterior)
- coloane ale caror valori sunt generate automat. In cadrul acestui curs vor fi prezentate coloanele de tip AUTO_INCREMENT, gandite sa contina valori unice si care pot fi populate automat de catre DBMS, degrevand clientul de grija unicitatii valorilor

Atunci cand, in cadrul instructiunii INSERT, sunt specificate valorile numai pentru o parte a coloanelor, este necesara precizarea listei de coloane carora le corespund acele valori. Coloanele precizate nu trebuie sa fie consecutive si nici sa se afle in ordinea din tabela. Pentru toate coloanele care nu apar in lista va fi introdusa valoarea default a coloanei in cauza.

Forma instructiunii este:

```
INSERT INTO NumeTabela(col1, col5, col8,..., colX) VALUES(val1, val5, val8,...,valX);
```

Numarul de coloane trebuie sa corespunda cu numarul de valori, valoarea de pe o anumita pozitie din lista de valori corespunzand coloanei aflate pe aceeasi pozitie in lista de coloane.

Aceasta forma a instructiunii INSERT are avantajul ca se poate modifica structura de coloane a tabelei (ex: adaugarea unei coloane noi sau reordonarea celor existente) fara a afecta instructiunile INSERT scrise astfel.

Exemplu:

```
CREATE TABLE Persoane(Nume VARCHAR(30) DEFAULT 'Anonim', NrCopii TINYINT NULL, Email VARCHAR(50));

# inregistrare noua pt care specificam doar NrCopii si Email; numele va ramane 'Anonim'
INSERT INTO Persoane (NrCopii, Email) VALUES(1, 'johndoe@yahoo.com');

# inregistrare noua pentru care specificam nume si NrCopii; emailul ramane NULL
# Ambele instructiuni ce urmeaza au acelasi efect, nu conteaza ordinea coloanelor
```

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

```
INSERT INTO Persoane (Nume, NrCopii) VALUES('The One',1);
INSERT INTO Persoane (NrCopii,Nume) VALUES(1,'The One'); # se schimba si ordinea valorilor!

# inregistrare pentru care „sarim” o coloana; NrCopii ramane NULL
INSERT INTO Persoane (Nume,Email) VALUES('Mihai','mihai@gmail.com');

# introducem o noua coloana la inceput; instructiunile INSERT functioneaza corect in continuare
ALTER TABLE Persoane ADD COLUMN id INT FIRST;
INSERT INTO Persoane (Nume,Email) VALUES('Paul','paul@yahoo.com');
```

Exista si o forma alternativa a instructiunii INSERT care permite specificarea valorilor pentru o parte a coloanelor sau pentru toate:

```
INSERT INTO NumeTabela SET col1=val1, col2=val2,...,colN=valN;
```

Exemplu:

```
INSERT INTO Persoane SET Nume='Andrei', NrCopii=2;
```

3.2.4. Introducerea unei inregistrari formate exclusiv din valori default

MySQL ofera posibilitatea introducerii unei inregistrari fara a specifica niciuna dintre valorile componente; in aceste conditii, noua inregistrare va fi compusa din valorile default ale tuturor coloanelor. Sintaxa instructiunii in acest caz este:

```
INSERT INTO NumeTabela VALUES();
```

3.2.5. Introducerea mai multor inregistrari cu o singura instructiune

Sintaxa lui INSERT permite introducerea mai multor inregistrari in cadrul aceleiasi instructiuni. De aceasta facilitate pot beneficia toate formele discutate mai sus:

```
INSERT INTO Persoane VALUES('Simona', 2, 'simo@yahoo.com'),('Elena',NULL, '3l3na@gmail.com');
INSERT INTO Persoane(Nume, NrCopii) VALUES('Victor',3),('George',1),('Bogdan',2);
INSERT INTO Persoane VALUES(),(),(),(); # 4 inregistrari noi cu valori default
```

Daca unul dintre seturile de valori genereaza o eroare (ex: tip de date incompatibil, NULL acolo unde nu este permis etc.), valorile de pana atunci raman introduse, insa cele urmatoare nu vor mai fi adaugate in tabela:

```
SET sql_mode='traditional';
CREATE TABLE Perechi(Nr1 INT, Nr2 INT NOT NULL);
INSERT INTO Perechi VALUES(1,2),(3,4),(5,NULL),(6,7),(8,9);
      ERROR 1048 (23000): Column 'Nr2' cannot be null
SELECT * from Perechi;
```

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

```
+-----+-----+
| Nr1 | Nr2 |
+-----+-----+
|   1 |   2 |
|   3 |   4 |
+-----+-----+
```

3.2.6. Introducerea inregistrarilor returnate de o interogare SELECT

Atunci cand dorim sa introducem intr-o tabela rezultatul unei alte interogari, avem la dispozitie sintaxa INSERT...SELECT. Ea poate fi folosita in doua moduri, asemanator cu cazurile discutate mai sus:

- fara a specifica lista de coloane. In acest caz, numarul de coloane returnate de interogarea SELECT trebuie sa fie egal cu numarul de coloane din definitia tabelei in care se introduc datele, iar tipurile de date ale coloanelor rezultate din interogare sa fie compatibile cu cele ale coloanelor corespondente:

```
/* preluarea datelor dintr-o tabela si introducerea lor in alta; cele doua tabele trebuie sa aiba acelasi numar de coloane si tipuri de date identice sau compatibile */
INSERT INTO NewTable SELECT * FROM OldTable
```

- putem specifica lista numelor de coloane in cadrul instructiunii INSERT, imediat dupa numele tabelei; numarul de coloane din lista trebuie sa fie egal cu numarul de coloane al rezultatului lui SELECT

```
INSERT INTO Songs (Title, Duration) SELECT (Name, Time FROM Music)
```

3.2.7. Modul de lucru al serverului si operatiile de INSERT

Serverul MySQL dispune de o optiune de configurare denumita sql_mode. Aceasta influenteaza diverse aspecte ale functionarii serverului - modul in care acesta proceseaza interogarile, felul in care trateaza valorile invalide sau lipsa etc. Atunci cand sql_mode nu este setat, serverul MySQL este mai permisiv, ajustand valorile invalide sau alegand valori default acolo unde este posibil; daca se foloseste un mod mai strict (in ideea compatibilitatii cu standardul), serverul va genera erori acolo unde datele introduse nu sunt corespunzatoare.

Efectul lui sql_mode la introducerea de inregistrari se manifesta in doua situatii:

- in cazul coloanelor declarate ca NOT NULL si care nu au valori default. In modul non-strict, atunci cand, la introducerea unei noi inregistrari, nu se specifica valoarea pentru o astfel de coloana, MySQL alege automat o valoare implicita in functie de tipul de date (0 pentru numere, sirul vid pentru siruri de caractere etc) si o introduce pe coloana in cauza. In modul strict, serverul nu va mai alege o valoare default, ci va genera o eroare, reclamand lipsa valorii. Astfel, in modul strict clientul este obligat sa specifice o valoare pentru acea coloana la fiecare introducere de noua inregistrare.
- in cazul introducerii pe o coloana a unei valori invalide ca format sau a carei valoare este situata in afara domeniului posibil pentru tipul de date al coloanei. In modul non-strict, la introducerea unei valori invalide pe o coloana, valoarea va fi ajustata astfel incat sa fie adusa in zona valida (ex: intregii prea mari vor fi trunchiati la valoarea maxima admisibila pentru tipul de date al coloanei). In modul strict, incercarea de introducere a unei valori invalide se va solda cu o eroare

Optiunea sql_mode poate fi modificata:

- **global**, ea avand efect asupra tuturor clientilor serverului. Setarea globala poate fi efectuata:

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

- din fisierul de configurare al serverului sau pasand optiuni in linia de comanda a serverului la lansarea sa in executie
- dintr-o sesiune de client, caz in care noul mod SQL se va aplica tuturor sesiunilor de client ulterioare

– **per-sesiune**, caz in care ea are efect numai asupra conexiunii (sesiunii) clientului care a initiat modificarea

Prezentam aici numai setarea sql_mode per-sesiune, care se realizeaza prin atribuirea unei valori variabilei SQL sql_mode:

```
mysql> SET sql_mode='traditional';
```

Exemple:

```
mysql> SELECT @@sql_mode;
+------------+
| @sql_mode |
+------------+
|            |
+------------+

mysql> CREATE TABLE Persoane(Nume VARCHAR(30) DEFAULT 'Anonim', NrCopii TINYINT NOT NULL, Email
VARCHAR(50) NOT NULL);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Persoane(Nume) VALUES('Alin');
Query OK, 1 row affected, 2 warnings (0.00 sec)

Warning (Code 1364): Field 'NrCopii' doesn't have a default value
Warning (Code 1364): Field 'Email' doesn't have a default value

mysql> select * from Persoane;
+------+---------+-------+
| Nume | NrCopii | Email |
+------+---------+-------+
| Alin |       0 |       |
+------+---------+-------+
2 rows in set (0.00 sec)

mysql> SET sql_mode='traditional';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Persoane(Nume) VALUES('Viorel');
ERROR 1364 (HY000): Field 'NrCopii' doesn't have a default value
```

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

3.3. Instructiunea SELECT

3.3.1. Prezentare si sintaxa generala

Instructiunea SELECT este folosita de catre clientul SQL pentru a obtine date de la server in mod controlat. Datele pot proveni dintr-una sau mai multe tabele SQL sau pot fi scrise chiar in cadrul instructiunii SELECT. Rezultatul intors de server ca urmare a executiei acestei instructiuni consta dintr-un ansamblu de randuri si coloane, la fel ca in cazul unei tabele. Fiecare coloana are un nume ce poate fi stabilit de catre server sau ales de catre client. In cadrul instructiunii SELECT se specifica ce valori vor fi plasate pe fiecare coloana si de unde provin acestea.

Nota: *chiar daca rezultatul returnat de o instructiune SELECT seamana pana la un punct cu o tabela (fiecare coloana are un nume si un tip de date, fiecare rand e format din valorile pentru fiecare coloana in parte) trebuie spus ca rezultatul interogarii SELECT nu are unele caracteristici proprii tabelelor (ex: storage engine - deoarece nu este memorat permanent pe server) si, in plus, chiar si in conditiile in care provine dintr-o singura tabela, rezultatul poate contine doar o parte din datele tabelei sau combinatii ale datelor tabelei cu date specificate in cadrul instructiunii SELECT. Putem privi rezultatul intors de SELECT ca pe un caraus de informatie asemanator ca structura cu o tabela.*

Pentru a exemplifica cateva cazuri tipice de folosire a instructiunii SELECT, sa presupunem ca avem o tabela cu angajati, care contine, pentru fiecare angajat, nume, prenume si salariu. Iata cateva exemple de interogari pe care le putem scrie:

- o interogare care extrage toate inregistrarile din tabela cu angajati. In acest caz, rezultatul returnat de instructiunea SELECT va coincide cu continutul tabelei de provenienta
- o interogare care extrage numele si prenumele tuturor angajatilor. Rezultatul returnat de SELECT va contine astfel doar o parte dintre coloanele tabelei din care provin datele
- o interogare care extrage datele complete ale angajatilor ce au salariul mai mare de 1000$. In acest caz, rezultatul lui SELECT va contine doar o parte dintre randurile tabelei, fiecare rand returnat avand insa datele complete ale angajatului (constituind astfel o copie a randului corespunzator din tabela)
- o interogare care returneaza doua coloane pentru fiecare angajat: una cu numele complet (rezultat din concatenarea numelui cu prenumele) si una cu contributia la asigurarile de sanatate (care se calculeaza ca procent din salariu). In acest caz, datele returnate rezulta din prelucrari aplicate informatiilor din tabela de provenienta

Fiecare dintre aceste cazuri de utilizare este posibil prin intermediul unor clauze ale instructiunii SELECT ce vor fi discutate in paragrafele urmatoare.

Sintaxa generala a instructiunii SELECT este:

```
SELECT valori_coloana_1,valori_coloana_2,...
<FROM NumeTabela>
<WHERE conditii>
<ORDER BY criterii>
<LIMIT nr_randuri>
```

Toate clauzele care urmeaza primei linii din sintaxa de mai sus sunt optionale (cele semnalizate prin <...>).

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

Nota: *instructiunea SELECT accepta si alte clauze decat cele de mai sus, care vor fi insa discutate in lectii ulterioare.*

In continuare vor fi prezentate diferitele modalitati de utilizare a instructiunii SELECT.

3.3.2. Specificarea valorilor coloanelor returnate

In cadrul instructiunii SELECT, expresiile ce dau valorile coloanelor pot contine:

1. **exclusiv constante sau expresii compuse din constante si functii predefinite** – altfel spus, informatie care nu provine dintr-o tabela aflata pe server. Spre exemplu, cea mai scurta forma a instructiunii SELECT este cea in care nu este prezenta nicio clauza:

```
mysql> SELECT 1,2+2, CURDATE();
+---+-----+------------+
| 1 | 2+2 | CURDATE()  |
+---+-----+------------+
| 1 |   4 | 2009-07-30 |
+---+-----+------------+
```

   Observam ca numele alese pentru coloane au corespuns in acest caz chiar expresiilor specificate in linia de comanda, iar valorile rezulta din evaluarea expresiilor respective (pe cea de-a doua coloana a fost efectuata adunarea). Dupa cum vom vedea, numele de coloane pot fi stabilite de catre noi tot in cadrul instructiunii SELECT.

   Remarca: *in instructiunea SELECT nu s-a specificat nicio tabela (nu exista clauza FROM).*

2. **nume de coloane de tabele**. Daca dorim sa extragem anumite coloane dintr-o tabela, putem lista numele lor pe post de expresii in instructiunea SELECT; in acest caz trebuie sa specificam si clauza FROM, care precizeaza tabela din care vor proveni datele. Spre exemplu, dintr-o tabela Angajati ce contine coloanele Nume, Prenume, Salariu si AnulAngajarii putem extrage numai numele si prenumele persoanelor:

```
mysql> SELECT Nume,Prenume FROM Angajati;
+-----------+---------+
| Nume      | Prenume |
+-----------+---------+
| Popescu   | Ion     |
| Ardeleanu | Mihai   |
+-----------+---------+
```

   Interogarea returneaza toate inregistrarile tabelei Angajati, insa pentru fiecare dintre ele sunt prezente numai coloanele solicitate.

   Atunci cand dorim returnarea tuturor coloanelor dintr-o tabela, putem folosi in loc de lista explicita de coloane caracterul special *:

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

```
SELECT * FROM Angajati
```

Folosirea caracterului * va determina asezarea coloanelor din rezultat in ordinea in care apar ele in definitia tabelei. Orice aplicatie client care se bazeaza pe aceasta ordine va avea probleme daca ordinea din tabela este schimbata - ex: daca se introduce o coloana in alt punct decat la sfarsitul listei de coloane). De aceea este in general mai avantajos ca interogarile SELECT sa extraga lista exacta de coloane dorite, in ordinea dorita.

Nota: *daca tabela din care se extrag datele nu face parte din baza de date implicita, atunci este necesara specificarea numelui sau complet. Exemplu: SELECT * FROM HumanResources.Angajati. (unde HumanResouces este numele bazei de date din care face parte tabela Angajati)*

3. **combinatii de constante si nume de tabele**. Limbajul SQL dispune de operatori si functii ce pot fi folosite pentru forma expresii complexe. Spre exemplu, daca dorim sa scriem o interogare SELECT care sa ne raporteze, pentru fiecare angajat, numele complet si vechimea in firma, am putea scrie:

```
SELECT CONCAT(Nume,Prenume), YEAR(CURDATE())-AnulAngajarii FROM Angajati;
+----------------------+-------------------------------+
| CONCAT(Nume,Prenume) | YEAR(CURDATE())-AnulAngajarii |
+----------------------+-------------------------------+
| PopescuIon           |                             2 |
| ArdeleanuMihai       |                             4 |
+----------------------+-------------------------------+
```

Prima coloana va consta din concatenarea numelui cu prenumele, folosind o functie MySQL predefinita, iar cea de-a doua coloana va contine numarul de ani scurs de la angajare, efectuand diferenta intre anul curent si anul angajarii. Anul curent se obtine printr-o combinatie de doua functii - functia CURDATE() returneaza data curenta, iar functia YEAR() extrage anul din aceasta.

Nota: *operatorii si functiile predefinite ce pot participa in elaborarea expresiilor vor fi discutate intr-un material ulterior.*

A se observa faptul ca numarul de coloane returnate de o interogare SELECT poate fi mai mare decat cel al tabelei din care citeste datele. Spre exemplu, daca dorim afisarea tuturor datelor din tabela Angajati si, pentru fiecare angajat, cuantumul salariului anual, putem scrie:

```
SELECT *,Salariu*12 FROM Angajati;
+-----------+---------+---------------+---------+------------+
| Nume      | Prenume | AnulAngajarii | Salariu | Salariu*12 |
+-----------+---------+---------------+---------+------------+
| Popescu   | Ion     |          2007 |    1000 |      12000 |
| Ardeleanu | Mihai   |          2005 |    1500 |      18000 |
+-----------+---------+---------------+---------+------------+
```

In instructiunea de mai sus, primul caracter * se refera la lista tuturor coloanelor tabelei, iar al doilea este operatorul de aritmetic de inmultire.

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

3.3.3. Specificarea numelor dorite pentru coloanele returnate

Precum s-a vazut si in exemplele anterioare, coloanele din rezultatul produs de o instructiune SELECT au ca nume chiar expresiile folosite in cadrul instructiunii. Acest lucru nu este insa intotdeauna convenabil, si de aceea limbajul SQL permite definirea de asa-numite **alias-uri** pentru coloane – nume atribuite coloanelor din result set dar stabilite de catre client.

Alias-urile se creeaza folosind cuvantul cheie AS urmat de numele dorit, imediat dupa expresia ce stabileste valorile unei coloane:

```
SELECT *,YEAR(CURDATE())-AnulAngajarii AS Vechime FROM Angajati
+-----------+---------+---------------+---------+---------+
| Nume      | Prenume | AnulAngajarii | Salariu | Vechime |
+-----------+---------+---------------+---------+---------+
| Popescu   | Ion     |          2007 |    1000 |       2 |
| Ardeleanu | Mihai   |          2005 |    1500 |       4 |
+-----------+---------+---------------+---------+---------+
```

Alias-urile sunt utile in cel putin doua situatii:
- cand numele implicit al coloanei ar fi altminteri prea lung (ex: YEAR(CURDATE())-AnulAngajarii ar fi fost numele implicit in exemplul de mai sus)
- cand extragem date din mai multe tabele si exista coloane cu acelasi nume in tabele diferite. Daca nu ar exista alias-uri, nu am mai putea identifica in mod unic coloanele din rezultat

Nota: *cuvantul cheie AS este optional. Este corecta sintactic si forma* SELECT YEAR(CURDATE())-AnulAngajarii Vechime*, insa este mai putin clara.*

Atentie! *Omiterea din greseala a virgulei dintre doua coloane din cadrul instructiunii SELECT duce la crearea unui alias:* SELECT Nume Prenume FROM Angajati *va produce o singura coloana cu titlul* Prenume *dar care va contine numele angajatilor!*

3.3.4. Filtrarea inregistrarilor returnate: clauza WHERE

Toate exemplele de instructiuni SELECT de pana acum, atunci cand citeau date dintr-o tabela, returnau toate inregistrarile tabelei in cauza. Prin adaugarea clauzei WHERE in cadrul instructiunii SELECT putem alege ce inregistrari din tabela dorim sa fie extrase.

Spre deosebire de coloanele unei tabele, care pot fi identificate dupa nume, inregistrarile nu mai au nume propriu care le identifica in mod unic, ci pot fi filtrate numai dupa datele pe care le contin. De aceea, clauza WHERE specifica randurile ce se doresc extrase prin intermediul unui ansamblu de conditii ce filtreaza randurile tabelei. Acest lucru corespunde modului in care formulam aceeasi cerinta in limbaj natural: spunem „care sunt angajatii cu salariu mai mare de 1000$ care s-au alaturat firmei inainte de 2004?”. Descompunand intrebarea, constatam ca ea contine doua conditii impuse angajatilor si care actioneaza ca un filtru: salariu mai mare decat 1000$ si anul angajarii mai mic decat 2004. Exact pe acest principiu functioneaza WHERE: el filtreaza inregistrarile tabelei pastrandu-le numai pe cele care indeplinesc setul de criterii specificat in clauza WHERE.

Sintaxa unei instructiuni SELECT ce contine si clauza WHERE este:

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

```
SELECT lista_coloane FROM tabela WHERE criterii
```

Criteriile consta din expresii construite folosind constante, operatori, functii si nume de coloane. Rezultatul unei astfel de expresii poate fi adevarat sau fals, evaluarea ei facandu-se pentru fiecare rand al tabelei. Randurile pentru care criteriile din WHERE se evalueaza true sunt pastrate in rezultatul final, restul nu.

Cea mai simpla expresie WHERE consta dintr-un singur criteriu. Iata cateva exemple:

```
# persoanele angajate dupa anul 2000
SELECT * FROM Angajati WHERE AnulAngajarii > 2000


# persoanele cu salariu mai mic decat 500$
SELECT * FROM Angajati WHERE Salariu < 500


# persoanele al caror venit anual depaseste 10000$
SELECT * FROM Angajati WHERE Salariu*12 > 10000


# persoanele angajate in alt an decat 2009
SELECT * FROM Angajati WHERE AnulAngajarii != 2009
```

Atunci cand dorim aplicarea mai multor criterii, putem crea expresii complexe in care criteriile elementare sunt combinate folosind operatorii logici AND (SI logic) si OR (SAU logic). Operatorul AND are prioritate mai mare decat OR.

```
# angajatii numiti Popescu sau Ionescu
SELECT * FROM Angajati WHERE Nume='Popescu' OR Nume='Ionescu'


# persoanele angajate dupa anul 2000 si al caror salatiu depaseste 1000$
SELECT * FROM Angajati WHERE AnulAngajarii>2000 AND Salariu >1000
```

Data fiind diferenta de prioritate intre operatori, in cadrul unei expresii mai complexe in care ordinea de evaluare implicita nu este convenabila, putem prioritiza prin paranteze anumite portiuni:

```
# angajatii cu nume de familie Popescu sau Ionescu si care au salariu > 1000
SELECT * FROM Angajati WHERE (Nume='Popescu' OR Nume='Ionescu') AND Salariu>1000


/* fara paranteze, ar fi fost returnati toti salariatii numiti Popescu (indiferent de salariu),
impreuna cu cei numiti Ionescu si care au salariu > 1000 */
SELECT * FROM Angajati WHERE Nume='Popescu' OR Nume='Ionescu' AND Salariu>1000
# ...echivalent cu a scrie:
SELECT * FROM Angajati WHERE Nume='Popescu' OR (Nume='Ionescu' AND Salariu>1000)
```

Un caz special in expresiile WHERE il reprezinta „valoarea” NULL. Atunci cand dorim sa punem conditia ca o anumita coloana sau expresie sa fie NULL (sau dimpotriva), nu putem folosi operatorul =, deoarece NULL nu

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

este egal cu nicio alta valoare – nici macar cu el insusi! Solutia este sa folosim IS NULL sau IS NOT NULL, ca in exemplul de mai jos:

```
# angajatii in cazul carora nu a fost completat inca campul de salariu
SELECT * FROM Angajati WHERE Salariu IS NULL
```

Nota: *in MySQL exista si operatorul <=> („NULL-safe equal”), care, in cazul in care ambii operanzi sunt NULL, returneaza true.*

Atentie! In cadrul clauzei WHERE nu pot fi folosite alias-uri de coloane, deoarece coloanele in cauza apar abia dupa evaluarea conditiilor din WHERE (ele sunt un rezultat al aplicarii acelor conditii):

```
# instructiunea de mai jos nu va functiona
mysql> SELECT CONCAT(Nume, ' ', Prenume) AS NumeComplet FROM Oameni WHERE NumeComplet LIKE '%escu'
ERROR 1054 (42S22): Unknown column 'NumeComplet' in 'where clause'
```

3.3.5. Limitarea numarului de inregistrari returnate: clauza LIMIT

Sintaxa instructiunii SELECT permite clientului sa limiteze superior numarul de inregistrari returnate folosind clauza LIMIT. Aceasta are doua forme:

- una in care dupa LIMIT se specifica un singur parametru: numarul maxim de inregistrari returnate
- alta in care dupa LIMIT se specifica doi parametri separati prin virgula: primul este numarul de inregistrari „sarite” (ignorate), iar cel de-al doilea numarul maxim de inregistrari returnate

```
SELECT * FROM Angajati WHERE Salariu > 1000 LIMIT 3
```

Efectul instructiunii de mai sus este acela ca, din setul complet de inregistrari ce corespund conditiei din WHERE, vor fi returnate numai primele trei.

```
SELECT * FROM Angajati WHERE AnulAngajarii = 2005 LIMIT 2,4
```

Instructiunea anterioara va ignora primii doi angajati din setul complet de persoane angajate in anul 2005 dar ii va retuna in schimb pe urmatorii 4.

Aceasta ultima forma a clauzei LIMIT este utila atunci cand se doreste extragerea pe portiuni a informatiei (ex: extragerea angajatilor 10 cate 10, in ideea afisarii lor pagina cu pagina intr-o aplicatie).

Trebuie inteles faptul ca LIMIT N returneaza intr-adevar primele N inregistrari cerute, insa care sunt acelea depinde de ordinea in care sunt extrase. Clauza suplimentara ORDER BY permite specificarea criteriilor de ordonare a inregistrarilor, si in aceste conditii este foarte probabil, spre exemplu, ca primii 3 angajati ordonati dupa alfabetic nume sa difere de primii 3 angajati ordonati numeric dupa salariu.

3.3.6. Ordonarea inregistrarilor returnate: clauza ORDER BY

Clauza ORDER BY permite clientului sa stabileasca ordinea in care ii sunt livrate inregistrarile produse de o interogare SELECT. ORDER BY poate fi urmat de unul sau mai multe criterii de ordonare. Pentru fiecare criteriu, ordonarea poate fi crescatoare (folosind cuvantul cheie ASC), descrescatoare (folosind cuvantul cheie

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

DESC) sau aleatoare (folosind RAND()). Daca nu se specifica tipul de ordonare, ordonarea crescatoare este cea implicita.

Exemple:

```
# primii 10 angajati cu cel mai mare salariu (top ten)
SELECT * FROM Angajati ORDER BY Salariu DESC LIMIT 10


# primii 3 angajatii cu cel mai mic salariu (ordonare crescatoare implicita)
SELECT * FROM Angajati ORDER BY Salariu LIMIT 3


/* criterii de ordonare multiple: ordonarea inregistrarilor se face dupa nume, crescator
(implicit); atunci cand mai multe inregistrari au acelasi nume, ele vor fi ordonate dupa prenume,
crescator (implicit) */
SELECT * FROM Angajati WHERE Salariu > 1000 ORDER BY Nume,Prenume


/* criterii de ordonare multiple, sens de ordonare diferit: lista angajatilor grupati pe an,
crescator, in cadrul fiecarui an ei fiind ordonati descrescator dupa salariu, iar pentru
inregistrarile cu acelasi an si salariu se face ordonare crescatoare dupa numele de familie si apoi
dupa prenume */
SELECT * FROM Angajati ORDER BY AnulAngajarii, Salariu DESC, Nume, Prenume


# tombola: extragem un angajat in mod aleator pentru a-l premia
SELECT * FROM Angajati ORDER BY RAND() LIMIT 1
```

Nota: *ordonarea sirurilor de caractere depinde de proprietatile character set si collation ale sirurilor in cauza (vezi capitolul dedicat tipurilor de date)*

Clauza ORDER BY poate fi folosita in combinatie cu LIMIT, inregistrarile produse putand diferi in functie de criteriile de ordonare alese:

```
# primele 3 persoane angajate in firma
SELECT * FROM Angajati ORDER BY AnulAngajarii LIMIT 3


# primii 3 angajatii cu cel mai mare salariu
SELECT * FROM Angajati ORDER BY Salariu DESC LIMIT 3
```

3.3.7. Combinarea rezultatelor mai multor interogari SELECT

MySQL permite concatenarea seturilor de rezultate provenite din mai multe interogari folosind sintaxa:

```
interogare_select_1  UNION interogare_select_2  UNION ....
```

Rezultatele vor fi concatenate in ordinea in care apar in interogarea compusa ce foloseste UNION. Toate interogarile componente trebuie sa produca acelasi numar de coloane. Daca exista randuri duplicat, UNION le va elimina, pastrand numai exemplarele unice. Atunci cand dorim pastrarea duplicatelor putem folosi UNION ALL pentru separarea interogarilor componente.

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

3.4. Instructiunea DELETE

Instructiunea DELETE permite stergerea controlata a inregistrarilor dintr-una sau mai multe tabele SQL. Sintaxa sa generala este:

```
DELETE FROM tabela <WHERE conditii> <ORDER BY criterii> <LIMIT limite>
```

Clauzele WHERE, ORDER BY si LIMIT sunt optionale. Atunci cand nu este specificata niciuna dintre ele, instructiunea DELETE sterge toate inregistrarile tabelei; dupa cum se stie insa, acest lucru poate fi insa realizat mai eficient folosint TRUNCATE. Clauza WHERE functioneaza la fel ca in cazul instructiunii SELECT, permitand specificarea inregistrarilor ce se doresc sterse.

Atunci cand se foloseste LIMIT, conteaza din nou ordinea inregistrarilor:

```
# stergerea angajatului cu cel mai mare salariu
DELETE FROM Angajati ORDER BY Salariu DESC LIMIT 1


# stergerea celor mai noi trei angajati
DELETE FROM Angajati ORDER BY AnulAngajarii DESC LIMIT 3
```

Instructiunea DELETE nu intoarce inregistrari, ci doar raporteaza numarul de randuri sterse:

```
mysql> DELETE FROM Angajati WHERE AnulAngajarii=2009;
Query OK, 2 rows affected (0.03 sec)
```

3.5. Instructiunea UPDATE

Instructiunea UPDATE permite modificarea valorilor inregistrarilor deja continute in tabele. Sintaxa sa generala este:

```
UPDATE NumeTabela SET coloana1 = expresie1, <coloana2 = expresie2, ....> <WHERE conditii>
```

Cu o singura instructiune pot fi operate modificari pe una sau mai multe coloane. Clauza WHERE poate lipsi insa, in aceste conditii, vor fi modificate toate inregistrarile tabelei. Expresiile ce dau valorile coloanelor respecta aceleasi reguli ca in cazul lui SELECT – ele pot contine constante, functii predefinite, nume de coloane, operatori.

```
# acordam tuturor angajatilor o crestere salariala de 10%
UPDATE Angajati SET Salariu=Salariu*1.1


# corectam numele de familie al persoanelor numite Popescu care din greseala nu incep cu majuscula
UPDATE Angajati SET Nume='Popescu' WHERE Nume='popescu'


# corectam inregistrarea lui Ion Popescu astfel incat numele si prenumele sa inceapa cu majuscule
UPDATE Angajati SET Nume='Popescu',Prenume='Ion' WHERE Nume='popescu' AND Prenume='ion'
```

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.

Instructiunea UPDATE poate fi combinata cu LIMIT si ORDER BY, la fel ca si celelalte prezentate pana acum:

```
# primii 3 angajati cu cel mai mic salariu primesc o marire de 20%
UPDATE Angajati SET Salariu = Salariu*1.2 ORDER BY Salariu LIMIT 3
```

Instructiunea UPDATE raporteaza numarul de inregistrari ce fac subiectul modificarii si cate dintre ele au fost modificate. Este posibil ca cele doua sa fie diferite sau numarul de inregistrari modificate de fie 0 – iata cateva situatii posibile:

- atunci cand nu exista inregistrari in tabela
- atunci cand valorile setate sunt aceleasi cu cele existente:

```
mysql> UPDATE Angajati SET AnulAngajarii=2009 WHERE AnulAngajarii>2008 AND AnulAngajarii<2010;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

- atunci cand nicio inregistrare nu corespunde criteriilor specificate:

```
mysql> UPDATE Angajati SET Salariu=Salariu/2 WHERE Salariu>100000;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

3.6. BIBLIOGRAFIE

- MySQL 5 Certification Study Guide: Cap. 9.1-9.3, Cap. 11
- Instructiunea SELECT: http://dev.mysql.com/doc/refman/5.1/en/select.html
- Instructiunea INSERT: http://dev.mysql.com/doc/refman/5.1/en/insert.html
- Instructiunea DELETE: http://dev.mysql.com/doc/refman/5.1/en/delete.html
- Instructiunea UPDATE: http://dev.mysql.com/doc/refman/5.1/en/update.html

Studentul poate utiliza prezentul material si informatiile continute in el exclusiv in scopul asimilarii cunostintelor pe care le include, fara a afecta dreptul de proprietate intelectuala detinut de InfoAcademy.