6. Programarea bazelor de date. Interogări simple. Sortarea datelor

- Limbajul SQL și categoriile de comenzi SQL existente
- Elemente de bază ale limbajului SQL
- Operații care se pot realiza cu comanda SELECT
- Scrierea comenzilor de interogare
- Filtrarea liniilor care vor fi afișate
- Folosirea alias-urilor coloanelor
- Eliminarea liniilor duplicate
- Sortarea datelor

6.1. Noţiuni introductive

- SQL (pronunţat fie ca un singur cuvânt “sequel” sau pe litere “S-Q-L”) se bazează pe studiile lui E.F. Codd, prima implementare a limbajului SQL fiind dezvoltată de către firma IBM la mijlocul anilor 1970. Mai târziu, compania Relational Software Inc. (cunoscută astăzi sub numele Oracle Corporation) a lansat prima versiune comercială de SQL.
- În prezent SQL este un limbaj complet standardizat, recunoscut de către Institutul Naţional American de Standarde (ANSI – American National Standards Institute).
- Puteţi folosi SQL pentru a accesa baze de date Oracle, SQL Server, DB2, sau MySQL.

SQL utilizează o sintaxă simplă, uşor de învăţat şi utilizat. Comenzile SQL pot fi grupate în cinci categori după cum urmează:

- **Limbajul de interogare** Permite regăsirea liniilor memorate în tabelele bazei de date. Vom scrie interogări folosind comanda **SELECT**.
- **Limbajul de manipulare a datelor** (**DML -** Data Manipulation Language) Permite modificarea conţinutului tabelelor. Există următoarele comenzi **DML**:
  - **INSERT** - pentru adăugarea de noi linii într-o tabelă
  - **UPDATE** - pentru modificarea valorilor memorate într-o tabelă
  - **DELETE** - pentru ştergerea liniilor dintr-o tabelă.
- **Limbajul de definire a datelor** (**DDL -** Data Definition Language) Vă permite să definiţi structura tabelelor care compun baza de date. Comenzile din această grupă sunt:
  - **CREATE** - vă permite să creaţi structurile bazei de date. De exemplu, **CREATE TABLE** este utilizată pentru crearea tabelelor, cu **CREATE USER**, puteţi crea utilizatorii bazei de date etc..
  - **ALTER** - permite modificarea structurilor bazei de date. De exemplu, cu comanda **ALTER TABLE** puteţi modifica structura unei tabele.
  - **DROP** - puteţi şterge structuri ale bazei de date. De exemplu pentru a şterge o tabelă folosiţi comanda **DROP TABLE**.
  - **RENAME** - puteţi schimba numele unei tabele.
  - **TRUNCATE** - vă permite să ştergeţi întregul conţinut al unei tabele.

- **Comenzi de control al tranzacţiilor** (**TC** - Transaction Control):
  - **COMMIT** - vă permite să faceţi ca modificările asupra bazei de date să devină permanente.
  - **ROLLBACK** - permite renunţarea la ultimele modificări asupra bazei de date.
  - **SAVEPOINT** – vă permite să definiţi un "punct de salvare" la care să puteţi reveni, renunţând la modificările făcute după acel punct asupra bazei de date.
- **Limbaj de control al datelor** (**DCL - Data Control Language**) Permite definirea şi modificarea drepturilor utilizatorilor asupra bazei de date. Există două comenzi în această categorie:
  - **GRANT** - vă permite să acordaţi drepturi altor utilizatori asupra structurilor bazei voastre de date.
  - **REVOKE** - puteţi să anulaţi anumite drepturi utilizatorilor bazei de date.

6.2. Elemente de bază ale SQL

Vom prezenta foarte pe scurt principalele elemente ce intră în componenţa unei comenzi SQL.

- **Nume**
  - Toate obiectele dintr-o bază de date, tabele, coloane, vizualizări, indexi, sinonime, etc, au un nume.
  - Numele poate fi orice şir de maxim **30** de litere, cifre şi caracterele speciale: caracterul de subliniere (underscore _), diez (#), şi dolar (**$**), primul caracter fiind obligatoriu o literă. Evident numele unui obiect din baza de date trebuie să fie unic.
- **Cuvinte rezervate**
  - Ca în orice limbaj, şi în SQL există o listă de cuvinte rezervate. Acestea sunt cuvinte pe care nu le puteţi folosi cu alt scop, ca de exemplu pentru denumirea tabelelor voastre.
- **Constante**
  - O constantă sau literal este o valoare fixă ce nu poate fi modificată. Există:
  - - constante numerice, de exemplu **2**, **3.5**, **.9** etc. Se observă că dacă un număr real are partea întreagă egală cu zero, ea nu mai trebuie precizată.
  - - constante alfanumerice (sau şir de caractere). Constantele şir de caractere sunt scrise între apostrofuri şi sunt case-sensitive. Exemple: **'abc'**, **'Numele'**.

- **Variabile**
  - Variabilele sunt date care pot avea în timp valori diferite. O variabilă are întotdeauna un nume pentru a putea fi referită.
  - SQL suportă două tipuri de variabile:
    - variabilele asociate numelor coloanelor din tabele
    - variabile sistem.
- **Expresii**
  - O expresie este formată din variabile, constante, operatori şi funcţii.

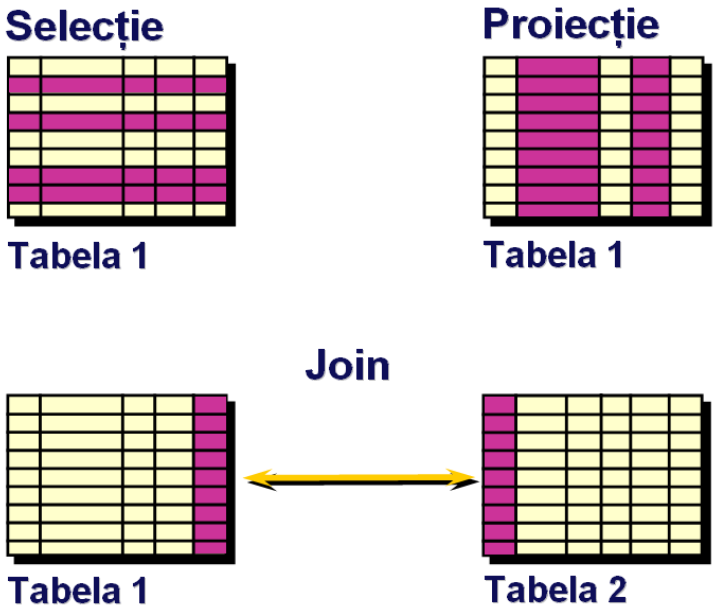
- În continuare ne vom ocupa de **operatorii ce pot fi folosiţi în expresii**:

- **Operatori aritmetici**
  - Operatorii aritmetici permişi în SQL sunt cei patru operatori din matematică: adunare **+**, scădere **-**, înmulţire *******, împărţire ***/***. Ordinea de efectuare a operaţiilor aritmetice este cea din matematică (mai întâi înmulţirea şi împărţirea şi apoi adunarea şi scăderea).
- **Operatori alfanumerici**
  - Există un singur operator alfanumeric şi anume operatorul de concatenare a două şiruri **||** (două bare verticale fără spaţii între ele).
  - De exemplu expresia **'abc'||'xyz'** are valoarea **'abcxyz'**.
- **Operatori de comparaţie**
  - Pe lângă operatorii obişnuiţi de comparaţie: **<**, **>**, **<=**, **>=**, **<>** sau **!=** (pentru diferit), **=**, SQL mai implementează următorii operatori speciali:
    - **LIKE** – despre care vom discuta puţin mai târziu în acest capitol
    - **BETWEEN** – testează dacă o valoare se găseşte într-un interval definit de două valori. Astfel expresia **x BETWEEN a AND b** este echivalentă cu expresia **(x>=a) AND (x<=b)**
    - **IN** – testează dacă o valoare aparţine unei mulţimi de valori specificate. De exemplu expresia: **x IN (a,b,c)** este echivalentă cu **(x=a) OR (x=b) OR (x=c)**
    - **IS NULL** şi **IS NOT NULL** – se folosesc pentru a testa dacă o expresie are valoarea **NULL** sau nu. Comparaţia cu **NULL** nu se poate face folosind operatorii obişnuiţi **=** şi respectiv **<>**.

- **Operatori logici** În ordinea priorităţii lor, aceştia sunt:
  - **NOT** – negaţia logică
  - **AND** – şi logic, expresia **a AND b** este adevărată dacă şi numai dacă ambii operanzi **a** şi **b** au valoarea adevărat.
  - **OR** – sau logic, expresia **a OR b** este adevărată dacă şi numai dacă cel puţin unul dintre operanzii **a** şi **b** au valoarea adevărat.

6.3. Interogarea tabelelor. Comanda SELECT

- Comanda **SELECT** este utilizată pentru a extrage date din baza de date. Setul de date returnate prin intermediul unei comenzi **SELECT** este compusă, ca şi tabelele bazei de date, din linii şi coloane, şi vor putea fi simplu afişate, sau vom putea popula o tabelă cu datele returnate de către comanda **SELECT**, aşa cum vom vedea într-un capitol următor.
- Cu ajutorul comenzii **SELECT** putem realiza următoarele tipuri de operaţii:
- **selecţia** – constă în filtrarea liniilor ce vor fi afişate. Vom folosi clauza **WHERE** pentru a defini criteriul sau criteriile pe care trebuie să le îndeplinească o linie pentru a fi returnată de către comanda **SELECT**.
- **proiecţia** – constă în alegerea doar a anumitor coloane pentru a fi afişate.
- **join** – constă în preluarea datelor din două sau mai multe tabele, "legate" conform unor reguli precizate.



Operaţiile realizate cu ajutorul comenzii **SELECT**

- Cea mai simplă formă a comenzii **SELECT** are sintaxa:

SELECT *Lista_expresii*

FROM *tabela*

- În clauza **SELECT** se va preciza o listă de coloane sau expresii ce se vor afişa, separate prin câte un spaţiu. În clauza **FROM** precizăm tabela din care se vor extrage coloanele ce vor fi afişate sau pe baza cărora vom realiza diverse calcule.

- Vom exemplifica modul de folosire al comenzii **SELECT** pe tabela Persoane, având următoarea structură şi conţinut:

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe	Brasov	22	5	300
4	Georgescu	Maria	Iasi	30	6	890
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840
7	Bischin	Paraschiva	Brasov	22	-	500
8	Olaru	Angela	Ploiesti	22	2	1500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
3	Popescu	Ioan	Bucuresti	10	2	1200

- Pentru a afişa toate datele (toate coloanele şi toate liniile) din tabela **persoane** vom scrie simplu:
- **SELECT * FROM persoane**
- Observaţi că în locul listei de coloane am scris un singur asterisc, ceea ce înseamnă că dorim să afişăm toate coloanele tabelei.

- Dacă însă dorim să afişăm doar informaţiile din câteva coloane ale tabelei, de exemplu dorim să afişăm numele, prenumele şi localitatea fiecărei persoane vom preciza numele coloanelor în clauza **SELECT**:

SELECT nume, prenume, localitate FROM persoane

NUME	PRENUME	LOCALITATE
Ionescu	Gheorghe	Brasov
Georgescu	Maria	Iasi
Marinescu	Angela	Sibiu
Antonescu	Elena	Sibiu
Bischin	Paraschiva	Brasov
Olaru	Angela	Ploiesti
Vasilescu	Vasile	Cluj-Napoca
Popescu	Ioan	Bucuresti

- După cum am precizat, putem realiza şi calcule cu coloanele unei tabele. De exemplu pentru a afişa pentru fiecare persoană, salariul mărit cu 10% folosim următoarea comandă:

SELECT nume, prenume, salariu, salariu * 1.10

FROM persoane

NUME	PRENUME	SALARIU	SALARIU*1.10
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
Antonescu	Elena	840	924
Bischin	Paraschiva	500	550
Olaru	Angela	1500	1650
Vasilescu	Vasile	950	1045
Popescu	Ioan	1200	1320

6.4. ALIASUL UNEI COLOANE

- În tabelul de mai jos puteţi observa că în capul de tabel afişat sunt trecute numele coloanelor cu majuscule sau expresia care a generat acea coloană, tot cu majuscule.
- Dacă dorim ca în capul de tabel să apară alt text, sau să nu se folosească doar majuscule va trebui să folosim un ALIAS pentru coloana respectivă.
- Aliasul este introdus în clauza **SELECT**, imediat după numele coloanei respective astfel:

SELECT nume, prenume, salariu AS SalariuVechi,
salariu * 1.10 AS SalariuNou FROM persoane

NUME	PRENUME	SALARIU	SALARIU*1.10
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
Antonescu	Elena	840	924
Bischin	Paraschiva	500	550
Olaru	Angela	1500	1650
Vasilescu	Vasile	950	1045
Popescu	Ioan	1200	1320

- În această comandă am stabilit două alias-uri: **SalariuVechi** şi respectiv **SalariuNou**. Trebuie subliniat că nu este obligatorie folosirea cuvântului **AS** pentru a defini un alias, însă este de preferat să îl utilizăm pentru o mai mare claritate. Comanda anterioară va afişa:

NUME	PRENUME	SALARIUVECHI	SALARIUNOU
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
….			
Popescu	Ioan	1200	1320

- Puteţi observa că deşi în comanda **SELECT** am scris aliasele folosind atât litere mici cât şi litere mari, la afişare acestea sunt scrise tot cu majuscule. Pentru a evita acest lucru, trebuie să introducem aliasul între ghilimele:

SELECT nume, prenume, salariu AS "SalariuVechi", salariu * 1.10 AS "SalariuNou" FROM persoane

NUME	PRENUME	SalariuVechi	SalariuNou
Ionescu	Gheorghe	300	330
Georgescu	Maria	890	979
Marinescu	Angela	2100	2310
….			
Popescu	Ioan	1200	1320

- De asemenea dacă dorim ca aliasul să conţină mai multe cuvinte de exemplu **Salariul Nou** respectiv **Salariul Vechi**, va trebui să folosim şi de această dată ghilimele, în caz contrar generându-se o eroare. De exemplu comanda următoare va afişa tabelul II.1.6:

SELECT nume||' '||prenume "Numele si prenumele",

salariu AS "Salariu Vechi",

salariu * 1.10 AS "Salariu Nou" FROM persoane

Numele si prenumele	Salariu Vechi	Salariu Nou
Ionescu Gheorghe	300	330
Georgescu Maria	890	979
Marinescu Angela	2100	2310
…		
Popescu Ioan	1200	1320

- În cadrul clauzei **SELECT**, se pot folosi orice fel expresii în care se folosesc nume de coloane, constante, operatori, funcţii etc. De exemplu, comanda următoare va afişa tabelul II.1.7.

SELECT nume||' '||prenume||' are salariul egal cu '|| salariu AS "Informatii persoane" FROM persoane

Informatii persoane
Ionescu Gheorghe are salariul egal cu 300
Georgescu Maria are salariul egal cu 890
Marinescu Angela are salariul egal cu 2100

6.5. Eliminarea liniilor duplicate

- Să analizăm rezultatul rulării următoarei comenzi:

SELECT localitate, firma

FROM persoane

- În tabelul alăturat se poate observa că în localitatea Braşov există două persoane care lucrează la aceeaşi firma având codul 22.

LOCALITATE	FIRMA
Brasov	**22**
Iasi	30
Sibiu	-
Sibiu	10
Brasov	**22**
Ploiesti	22
Cluj-Napoca	15
Bucuresti	10

- Dacă dorim să vedem la ce firme lucrează persoanele din fiecare localitate, însă o firmă să fie afişată o singură dată pentru o localitate anume, deci combinaţia valorilor localitate şi firmă să fie unică, vom folosi clauza **DISTINCT** în cadrul clauzei **SELECT** astfel:

SELECT DISTINCT localitate, firma FROM persoane

- Combinaţia (Braşov, 22) este afişată acum o singură dată:

LOCALITATE	FIRMA
Brasov	22
Bucuresti	10
Cluj-Napoca	15
Iasi	30
Ploiesti	22
Sibiu	10
Sibiu	-

- Dar dacă dorim să afişăm doar localităţile ce apar în tabela Persoane, fiecare localitate să fie afişată o singură dată? Vom scrie:

SELECT DISTINCT localitate FROM persoane

LOCALITATE
Brasov
Bucuresti
Cluj-Napoca
Iasi
Ploiesti
Sibiu

6.6. Filtrarea liniilor. Clauza WHERE

- Spre exemplu, tabela persoane conţine date despre mii de persoane şi că la un moment dat vă interesează doar informaţiile despre persoanele dintr-o anumită localitate.
- Pentru a putea selecta doar acele linii care ne interesează, trebuie să adăugăm clauza **WHERE** la comanda **SELECT**. În această clauză vom preciza condiţiile pe care trebuie să le îndeplinească o linie pentru a fi afişată. Aşadar clauza **WHERE** permite realizarea operaţiei de selecţie.
- De exemplu pentru a afişa toate persoanele care provin din **Bucureşti** sau **Braşov** vom scrie:

SELECT * FROM persoane

WHERE localitate='Brasov' OR localitate='Bucuresti’

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500
3	Popescu	Ioan	Bucuresti	10	2	1200

- Operatorul **LIKE** este utilizat pentru a verifica dacă un şir de caractere respectă un anumit "model". Dacă valoarea se potriveşte modelului, operatorul va returna valoarea **true** (adevărat) în caz contrar va returna valoarea **False** (fals).
- În model se pot utiliza următoarele caractere speciale:
  - caracterul de subliniere (underscore _) ţine locul unui singur caracter, oricare ar fi acesta.
  - caracterul procent (**%**) ţine locul la zero sau mai multe caractere, oricare ar fi acestea.
- De exemplu, dacă dorim să afişăm toate persoanele al căror prenume conţine litera **a** pe orice poziţie, vom scrie:

```
SELECT * FROM persoane
WHERE lower(prenume) LIKE '%a%'
```

- Modelul **'%a%'** precizează că în faţa caracterului **a**, în prenume, se pot găsi oricâte caractere, inclusiv zero caractere, iar după caracterul **a** se găsesc de asemenea oricâte caractere, inclusiv zero. Am folosit funcţia **LOWER** pentru a transforma toate caracterele în litere mici, altfel numele care încep cu litera **A**, întrucât acesta e scris cu majuscule, nu ar fi fost afişat.

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
4	Georgescu	Maria	Iasi	30	6	890
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840
7	Bischin	Paraschiva	Brasov	22	-	500
8	Olaru	Angela	Ploiesti	22	2	1500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
3	Popescu	Ioan	Bucuresti	10	2	1200

- Dacă însă dorim să afişăm persoanele al căror prenume conţine litera **a** pe a doua poziţie vom folosi caracterul underscore în model:

SELECT * FROM persoane

WHERE prenume LIKE '_a%'

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
4	Georgescu	Maria	Iasi	30	6	890
7	Bischin	Paraschiva	Brasov	22	-	500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950

- În cazul în care trebuie să verificăm dacă un şir conţine unul dintre caracterele speciale underscore (_), backslash (****), procent (**%**) vom scrie în model caracterul respectiv precedat de orice caracter special (de exemplu **** sau **&**), iar după model vom preciza cu ajutorul clauzei **ESCAPE** care este caracterul special care introduce secvenţa corespunzătoare caracterelor ****, _, **%**.
- Pentru a afişa persoanele din tabela **employees** al căror **job_id** conţine caracterul underscore (_) pe a treia poziţie de la sfârşit folosim comanda:

```
SELECT first_name, job_id FROM employees
WHERE job_id LIKE '%&_ _ _' ESCAPE '&'
```

sau

```
SELECT first_name, job_id FROM employees
WHERE job_id LIKE '%\_ _ _' ESCAPE '\'
```

- Dacă dorim să afişăm persoanele al căror job_id conţine un caracter underscore oriunde în şir vom utiliza comanda:

SELECT first_name, job_id FROM employees

WHERE job_id LIKE '%&_%' ESCAPE '&'

sau

SELECT first_name, job_id FROM employees

WHERE job_id LIKE '%_%' ESCAPE '\'

6.7. Sortarea datelor. Clauza ORDER BY

- În situaţia de a trebui să ordonaţi anumite date pe baza unor criterii oarecare.
- Pentru a preciza criteriile după care se ordonează datele folosim clauza **ORDER BY**. În această clauză se vor preciza coloanele sau expresiile după care se vor ordona liniile unei tabele înainte de a fi afişate.
- De exemplu, afişarea datelor din tabela **persoane** în ordine alfabetică (crescătoare) a localităţii se face folosind comanda:

SELECT * FROM persoane

ORDER BY localitate

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	**Ionescu**	**Gheorghe**	**Brasov**	**22**	**5**	**300**
7	**Bischin**	**Paraschiva**	**Brasov**	**22**	**-**	**500**
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
4	Georgescu	Maria	Iasi	30	6	890
8	Olaru	Angela	Ploiesti	22	2	1500
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840

Se observă că există mai multe persoane din aceeaşi localitate.

- Dacă vrem ca persoanele din aceeaşi localitate să fie ordonate descrescător după salariu scriem:

SELECT * FROM persoane

ORDER BY localitate, salariu DESC

- opţiunea **DESC** precizează că sortarea se face descrescător. Pentru a sorta crescător se poate preciza acest lucru cu opţiunea **ASC**, dar aceasta este opţională deoarece implicit datele sunt sortate crescător.

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
7	**Bischin**	**Paraschiva**	**Brasov**	**22**	**-**	**500**
1	**Ionescu**	**Gheorghe**	**Brasov**	**22**	**5**	**300**
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
4	Georgescu	Maria	Iasi	30	6	890
8	Olaru	Angela	Ploiesti	22	2	1500
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840

- Dacă sortăm tabela **Persoane** după codul firmei, vom scrie:

SELECT * FROM persoane ORDER BY firma

- Rularea acestei comenzi duce la afişarea tabelului de mai jos. Observăm că Marinescu Angela, deoarece nu are completat codul firmei (valoarea codului firmei este **null**) a fost afişată ultima.
- La ordonarea crescătoare (implicită) valorile nule se trec la sfârşit, în timp ce la sortarea descrescătoare valorile nule apar la început.

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
6	Antonescu	Elena	Sibiu	10	1	840
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500
8	Olaru	Angela	Ploiesti	22	2	1500
4	Georgescu	Maria	Iasi	30	6	890
5	Marinescu	Angela	Sibiu	-	3	2100

- Comanda

SELECT * FROM persoane

ORDER BY firma DESC

- va face ca Marinescu Angela să fie afişată prima:

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
5	Marinescu	Angela	Sibiu	-	3	2100
4	Georgescu	Maria	Iasi	30	6	890
1	Ionescu	Gheorghe	Brasov	22	5	300
8	Olaru	Angela	Ploiesti	22	2	1500
7	Bischin	Paraschiva	Brasov	22	-	500
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950
6	Antonescu	Elena	Sibiu	10	1	840
3	Popescu	Ioan	Bucuresti	10	2	1200

- În criteriile de ordonare pot să apară şi expresii nu doar coloane din tabela interogată. Astfel putem scrie:

SELECT * FROM persoane ORDER BY prenume || nume

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
6	Antonescu	Elena	Sibiu	10	1	840
7	Bischin	Paraschiva	Brasov	22	-	500
4	Georgescu	Maria	Iasi	30	6	890
1	Ionescu	Gheorghe	Brasov	22	5	300
5	Marinescu	Angela	Sibiu	-	3	2100
8	Olaru	Angela	Ploiesti	22	2	1500
3	Popescu	Ioan	Bucuresti	10	2	1200
2	Vasilescu	Vasile	Cluj-Napoca	15	1	950

- De asemenea putem preciza ca sortarea să se facă după o expresie care apare în clauza **SELECT** prin indicarea poziţiei expresiei respective în lista de expresii din clauza **SELECT**.

- Astfel comanda

SELECT nume, prenume, salariu

FROM persoane ORDER BY 3 DESC

va sorta descrescător liniile după salariu, deoarece în caluza **SELECT**, salariu este a treia expresie (deși în tabela persoane salariul este coloana a 7-a):

NUME	PRENUME	SALARIU
Marinescu	Angela	2100
Olaru	Angela	1500
Popescu	Ioan	1200
Vasilescu	Vasile	950
Georgescu	Maria	890
Antonescu	Elena	840
Ionescu	Gheorghe	300
Bischin	Paraschiva	500

- În clauza **ORDER BY** putem folosi aliasul unei coloane ca în exemplul următor:

```
SELECT nume||' '||prenume AS "Nume si prenume", salariu
FROM persoane
ORDER BY "Nume si prenume"
```

Nume si prenume	SALARIU
Antonescu Elena	840
Bischin Paraschiva	500
Georgescu Maria	890
Ionescu Gheorghe	300
Marinescu Angela	2100
Olaru Angela	1500
Popescu Ioan	1200
Vasilescu Vasile	950

- Desigur clauzele **WHERE** şi **ORDER BY** pot apărea împreună în aceeaşi comandă, ordinea în care acestea apar fiind **WHERE** şi apoi **ORDER BY**, aceasta fiind şi ordinea în care sunt executate: mai întâi sunt selectate liniile care trebuie să fie afişate şi abia apoi sunt sortate conform criteriului stabilit prin clauza **ORDER BY**. De exemplu, pentru a afişa în ordine descrescătoare a salariilor doar persoanele din Braşov şi Sibiu scriem:

```
SELECT * FROM persoane
WHERE localitate IN ('Sibiu', 'Brasov')
ORDER BY salariu DESC
```

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
5	Marinescu	Angela	Sibiu	-	3	2100
6	Antonescu	Elena	Sibiu	10	1	840
1	Ionescu	Gheorghe	Brasov	22	5	300
7	Bischin	Paraschiva	Brasov	22	-	500

6.8. AFIŞAREA PRIMELOR N LINII

- La sfârşitul anului şcolar, dirigintele clasei vă roagă să-l ajutaţi să afle care sunt primii trei elevi din clasă, în ordinea descrescătoare a mediei generale, pentru a şti cui să dea premiile. Aşadar se pune problema ca la afişarea datelor dintr-o tabelă să afişaţi doar primele **n** linii.
- Pentru aceasta veţi avea nevoie de pseudocoloana **ROWNUM** care returnează numărul de ordine al unei linii într-o tabelă. De exemplu comanda următoare va afişa codul, numele şi prenumele persoanelor împreună cu numărul de ordine al acestora în tabela **persoane**:

SELECT cod, nume, prenume, rownum

FROM persoane

COD	NUME	PRENUME	ROWNUM
1	Ionescu	Gheorghe	1
4	Georgescu	Maria	2
5	Marinescu	Angela	3
6	Antonescu	Elena	4
7	Bischin	Paraschiva	5
8	Olaru	Angela	6
2	Vasilescu	Vasile	7
3	Popescu	Ioan	8

- Deşi ne-am aştepta ca într-o comandă **SELECT** care foloseşte clauza **ORDER BY**, **ROWNUM** să ne afişeze numărul de ordine al înregistrărilor în ordinea dată de **ORDER BY**, acest lucru nu se întâmplă, numărul de ordine fiind cel din tabela iniţială. Observaţi în acest sens tabelul II.1.25 afişat la rularea comenzii următoare

select rownum, cod, nume, prenume, localitate, firma, job, salariu

from persoane

order by salariu desc

ROWNUM	COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
3	5	Marinescu	Angela		-	3	2100
6	8	Olaru	Angela		22	2	1500
8	3	Popescu	Ioan	Bucuresti	10	2	1200
7	2	Vasilescu	Vasile		15	1	950
2	4	Georgescu	Maria		30	6	890
4	6	Antonescu	Elena		10	1	840
5	7	Bischin	Paraschiva		22	-	500
1	1	Ionescu	Gheorghe		22	5	300

- Aşadar dacă dorim să afişăm primele 3 înregistrări din tabela iniţială vom putea scrie simplu:

```
SELECT cod, nume, prenume, rownum
FROM persoane
WHERE ROWNUM<=3
```

COD	NUME	PRENUME	ROWNUM
1	Ionescu	Gheorghe	1
4	Georgescu	Maria	2
5	Marinescu	Angela	3

- Pentru a afişa persoanele cu cele mai mici trei salarii, comanda următoare nu afişează ceea ce am dori, deaorece Oracle prima dată va returna primele trei înregistrări din tabela persoane şi abia apoi le va sorta:

```
select rownum, cod, nume, prenume, localitate, firma, job, salariu
from persoane
where rownum<=3
order by salariu desc
```

ROWNUM	COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
3	5	Marinescu	Angela	Sibiu	-	3	2100
2	4	Georgescu	Maria	Iași	30	6	890
1	1	Ionescu	Gheorghe	Brașov	22	5	300

- Pentru a obţine rezultatul dorit de noi vom folosi o subinterogare astfel:

```
select *
from (select * from persoane
order by salariu)
where rownum<=3
```

- În acest fel am forţat Oracle să sorteze mai întâi liniile şi apoi să afişeze primele trei linii din tabela obţinută.

COD	NUME	PRENUME	LOCALITATE	FIRMA	JOB	SALARIU
1	Ionescu	Gheorghe		22	5	300
7	Bischin	Paraschiva		22	-	500
6	Antonescu	Elena		10	1	840

În această lecție am învățat despre:

- Limbajul SQL și categoriile de comenzi SQL existente
- Elemente de bază ale limbajului SQL
- Operații care se pot realiza cu comanda SELECT
- Scrierea comenzilor de interogare
- Filtrarea liniilor care vor fi afișate
- Folosirea alias-urilor coloanelor
- Eliminarea liniilor duplicate
- Sortarea datelor

SFÂRȘIT