8. Interogări multiple

- Cum se pot prelua informații din mai multe tabele simultan
- Care sunt tipurile de join cunoscute
- Cum se realizează fiecare tip de join folosind sintaxa Oracle
- Cum se realizează fiecare tip de join folosind sintaxa ANSI
- Cum se folosesc operatorii UNION, INTERSECT, MINUS

- În capitolele anterioare am aflat cum putem afişa informaţii din baza de date, însă la fiecare rulare a unei comenzi **SELECT** am afişat date dintr-o singură tabelă.
- Unul dintre rezultatele procesului de normalizare este acela că datele sunt memorate, de cele mai multe ori, în tabele diferite.
- De aceea, la afişarea diferitelor rapoarte va trebui să puteţi prelua date din mai multe tabele printr-o singură comandă SQL.
- SQL oferă facilităţi pentru combinarea datelor din mai multe tabele şi afişarea lor într-un singur raport. O astfel de operaţie se numeşte **join**, sau **interogare multiplă**.

- Pe parcursul acestui capitol vom folosi ca exemple tabela **Persoane** a cărei cheie primară este atributul **IdPersoana**, tabela **Firme** a cărei cheie primară este atributul **IdFirm**, şi tabela **Joburi** cu cheia primară **IdJob**. Presupunem că aceste tabele conţin următoarele înregistrări:

Tabela **Persoane**

IDPERSOANA	NUME	PRENUME	LOCALITATE	IDFIRM	IDJOB
1	Ionescu	Gheorghe	Brasov	22	5
2	Vasilescu	Vasile	Cluj-Napoca	15	1
3	Popescu	Ioan	Bucuresti	10	2
4	Georgescu	Maria	Iasi	30	6
5	Marinescu	Angela	Sibiu	-	3
6	Antonescu	Elena	Sibiu	10	1
7	Bischin	Paraschin	Brasov	15	-
8	Olaru	Angela	Ploiesti	22	2

Tabela **Firme**

IdFirm	Nume	Localitate
10	SC Crisib SA	Sibiu
15	SC SoftCom	Alba Iulia
20	SC TimTip	Timisoara
22	Brasoveanca	Brasov

Tabela **Joburi**

IdJob	Nume
1	Reprezentant Vanzari
2	Manager
6	Operator IT
3	Programator
4	Administrator
5	Administrator retea

- În Oracle există două moduri diferite de a scrie joinurile:
  - Prima metodă foloseşte sintaxa specifică Oracle. În acest caz condiţiile de join sunt incluse în clauza **WHERE**. Această metodă este mai uşor de înţeles, însă are dezavantajul că în aceeaşi clauză **WHERE** se includ atât condiţiile de filtrare a înregistrărilor afişate cât şi condiţiile de join.
  - A doua variantă foloseşte sintaxa ANSI/ISO, care este puţin mai greoaie, însă comenzile scrise folosind această sintaxă sunt portabile şi în alte SGBD-uri care folosesc limbajul SQL.

- Indiferent de sintaxa folosită există mai multe moduri de legare a tabelelor şi anume:
  - **Produsul cartezian** – leagă fiecare înregistrare dintr-o tabelă cu toate înregistrările din cealaltă tabelă.
  - **Equijoin** – sunt legate două tabele cu ajutorul unei condiţii de egalitate
  - **NonEquijoin** - în acest caz condiţia de join foloseşte alt operator decât operatorul de egalitatea
  - **SelfJoin** – este legată o tabelă cu ea însăşi, e folosită de obicei în conjuncţie cu relaţiile recursive.
  - **OuterJoin** – sunt o extensie a equijoinului, când pentru unele înregistrări dintr-o tabelă nu există corespondent în cealaltă tabelă, şi dorim ca aceste înregistrări fără corespondent să fie totuşi afişate.

8.1. Produsul cartezian

a) Sintaxa Oracle

- După cum am precizat, acest tip de legătură între două tabele, va lega fiecare rând din prima tabelă cu fiecare rând din cea de a doua tabelă. De exemplu comanda:

SELECT p.nume, p.prenume, f.nume

FROM persoane p, firme f

adică se obţin 8x4 = 32 înregistrări (tabela persoane conţine 8 înregistrări, tabela firme 4 înregistrări)

Nume	Prenume	Nume
Ionescu	Gheorghe	SC Crisib SA
Vasilescu	Vasile	SC Crisib SA
Popescu	Ioan	SC Crisib SA
Georgescu	Maria	SC Crisib SA
Marinescu	Angela	SC Crisib SA
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC Crisib SA
Olaru	Angela	SC Crisib SA
Ionescu	Gheorghe	SC SoftCom
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC SoftCom
Georgescu	Maria	SC SoftCom
Marinescu	Angela	SC SoftCom
Antonescu	Elena	SC SoftCom
Bischin	Paraschin	SC SoftCom
Olaru	Angela	SC SoftCom
Ionescu	Gheorghe	SC TimTip
Vasilescu	Vasile	SC TimTip
Popescu	Ioan	SC TimTip
Georgescu	Maria	SC TimTip
Marinescu	Angela	SC TimTip
Antonescu	Elena	SC TimTip
Bischin	Paraschin	SC TimTip
Olaru	Angela	SC TimTip
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	Brasoveanca
Popescu	Ioan	Brasoveanca
Georgescu	Maria	Brasoveanca
Marinescu	Angela	Brasoveanca
Antonescu	Elena	Brasoveanca
Bischin	Paraschin	Brasoveanca
Olaru	Angela	Brasoveanca

a) Sintaxa Oracle

- Notaţia **p.nume**, **p.prenume**, **f.nume**, precum şi literele **p** şi **f** care urmează după numele tabelelor din clauza **FROM** definește un alias al fiecărei tabele.
- Folosim acest alias, deoarece în ambele tabele există o coloană cu numele **nume** şi dacă nu prefaţăm numele acestei coloane cu aliasul tabelei se va genera o ambiguitate pe care serverul bazei de date nu va şti să o rezolve.
- Aliasul tabelei este obligatoriu să-l folosim când două tabele conţin coloane cu acelaşi nume.

a) Sintaxa Oracle

- În exemplul anterior coloana prenume nu este obligatoriu să o prefaţăm cu aliasul coloanei, astfel comanda anterioară poate fi scrisă şi astfel:

SELECT p.nume, prenume, f.nume

FROM persoane p, firme f

- Aşadar, produsul cartezian apare atunci când nu este precizată nici o condiţie privind modul de legare al celor două tabele.

b) Sintaxa ANSI

- Pentru a obţine produsul cartezian, în sintaxa ANSI vom folosi clauza **CROSS JOIN** în cadrul clauzei **FROM** ca în exemplul următor.

```
SELECT p.nume, p.prenume, f.nume
FROM persoane p CROSS JOIN firme f
```

- Rezultatul obţinut va coincide cu cel obţinut anterior.

8.2. EQUIJOIN

- Cum procedăm dacă dorim să afişăm pentru fiecare persoană, numele firmei la care lucrează? Să vedem de exemplu cum aflăm numele firmei la care lucrează Ionescu Gheorghe. Ne uităm în tabela **persoane**, la valoarea din coloana **IdFirm**. Această valoare este **22**. Apoi, în tabela **firme** căutăm firma având codul **22**, şi preluăm numele acestei firme din coloana **nume**. Acest nume este Brasoveanca. Aşadar Ionescu Gheorghe lucrează la firma Brasoveanca.
- Deci a trebuit ca valoarea din coloana **IdFirm** din tabela **Persoane** să coincidă cu valoarea coloanei **IdFirm** din tabela Firme.

a) Sintaxa Oracle

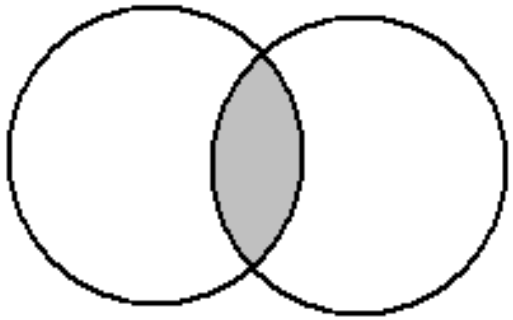
- Pentru a realiza acest lucru folosind SQL, vom preciza condiţia de egalitate dintre coloanele **IdFirm** din cele două tabele în clauza **WHERE** ca mai jos:

```
SELECT p.nume, prenume, f.nume
FROM persoane p, firme f
WHERE p.idfirm = f.idfirm
```

a) Sintaxa Oracle

Equijoin între tabelele **Persoane** şi **Firme**

Nume	Prenume	Nume
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC Crisib SA
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Olaru	Angela	Brasoveanca



Equijoin

a) Sintaxa Oracle

- În condiţia de equijoin pot fi precizate mai multe condiţii. Dacă de exemplu tabelele **elevi** şi **note** ar conţine următoarele coloane:

Elevi (#nume, #prenume, *adresa)

Note(#nume, #prenume, #disciplina, #data, *nota)

- atunci pentru a afişa toate notele unui elev vom folosi comanda:

```
SELECT a.nume, a.prenume, b.disciplina, b.data, b.nota
FROM elevi a, firme b
WHERE a.nume=b.nume AND a.prenume=b.prenume
```

b) Sintaxa ANSI

- În cazul sintaxei ANSI equijoinul se realizează folosind opţiunea **NATURAL JOIN** în cadrul clauzei **from** astfel:

```
SELECT nume, prenume, nume
FROM persoane NATURAL JOIN firme
```

- Dacă rulăm această comandă, ea nu afişează nici o linie pentru că **NATURAL JOIN**-ul leagă cele două tabele pe toate coloanele cu nume comun din cele două tabele.
- Comanda anterioară este echivalentă cu următoarea comandă scrisă folosind sintaxa Oracle:

```
SELECT p.nume, prenume, f.nume
FROM persoane p, firme f
WHERE p.idfirm = f.idfirm AND p.nume=f.nume
```

- ori nu are nici un sens să punem condiţia ca numele firmei (**f.nume**) să coincidă cu numele persoanei (**p.nume**).

b) Sintaxa ANSI

- Reguli de folosire a opţiunii **NATURAL JOIN**:
  - tabelele sunt legate pe toate coloanele cu nume comun
  - coloanele cu nume comun trebuie să aibă acelaşi tip
  - în clauza **SELECT** coloanele comune celor două tabele NU vor fi prefaţate de aliasul tabelei.
- Pentru a lega două tabele folosind sintaxa ANSI dar condiţia de egalitate să fie pusă doar pe anumite coloane (nu pe toate coloanele cu nume comun ci doar pe o parte din acestea) se va folosi în loc de **NATURAL JOIN** clauza **JOIN**, iar coloanele pe care se face joinul se precizează în opţiunea **USING**.
- Astfel comanda pentru afişarea firmelor la care lucrează fiecare angajat se scrie astfel:

```
SELECT p.nume, prenume, f.nume
FROM personae p JOIN firme f
USING (IdFirm)
```

b) Sintaxa ANSI

- Restricţii la folosirea clauzei **JOIN** cu clauza **USING**:
  - în clauza **USING** se trec în paranteză, separate prin virgulă, numele coloanelor pe care se va face joinul
  - coloanele din clauza **USING** trebuie să aibă acelaşi tip în cele două tabele
- Dacă în cele două tabele există nu există coloane cu acelaşi nume, sau coloanele cu nume comun au tipuri diferite în cele două tabele, se va folosi clauza **JOIN** în conjuncţie cu **ON**. În clauza **ON** pe poate trece orice condiţie de join între cele două tabele.

```
SELECT p.nume, prenume,  f.nume
FROM persoane p JOIN firme f
ON (p.IdFirm=f.IdFirm)
```

8.3. NONEQUIJOIN
A) SINTAXA ORACLE

- Să presupunem că în tabela **Note** avem trecute mai multe note ale elevilor unei şcoli.
- Structura tabelei este

Note(#nume, #prenume, #disciplina, #data, *nota)

- Dorim să înlocuim notele cu calificative, şi ştim de exemplu că notele de 9 şi 10 sunt transformate în calificativul **FOARTE BINE**, notele de **7** şi **8** în **BINE** etc. Aceste echivalenţe sunt memorate în tabela **CALIFICATIVE** cu structura următoare

CALIFICATIVE(#id, *nota1, *nota2, *calificativ)

- cu semnificaţia că notele cuprinse între notele **nota1** şi **nota2**, inclusiv, se vor transforma în **calificativ**.

a) Sintaxa Oracle

- Pentru a scrie calificativele corespunzătoare fiecărei note din tabela **note**, vom scrie următoarea comandă:

SELECT nume, prenume, disciplina, data, calificativ

FROM note, calificative

WHERE nota BETWEEN nota1 AND nota2

b) Sintaxa ANSI

- Echivalent vom scrie:

SELECT nume, prenume, disciplina, data, calificativ

FROM note JOIN calificative

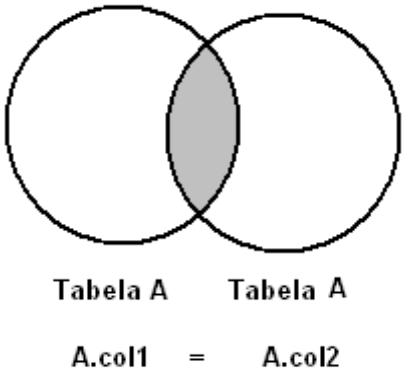
ON (nota BETWEEN nota1 AND nota2)

8.4. Self Join

- Ţinând cont de faptul că SelfJoin-ul este de fapt un equijoin dintre o tabela şi ea însăşi, lucrurile sunt mult mai simple. Considerăm de exemplu tabela **angajaţi** cu următoarea structură:

Angajaţi (#id, *nume, *prenume, *id_manager)

- în câmpul id_manager memorându-se codul şefului fiecărui angajat.



SelfJoin

Dorim să afişăm numele fiecărui angajat şi numele şefului acestuia. Vom folosi următoarele comenzi:

a) Sintaxa Oracle

```
SELECT a.nume ||' '||
    a.prenume AS "Angajat",
       b.nume ||' '|| b.prenume
    AS "Sef"
FROM angajat a, angajat b
WHERE a.id_manager = b.id
```

b) Sintaxa ANSI

```
SELECT a.nume ||' '|| a.prenume
    AS "Angajat",
       b.nume ||' '|| b.prenume AS
    "Sef"
FROM angajat a JOIN angajat b
ON (a.id_manager = b.id)
```

adică vom privi tabela **angajaţi** o dată ca tabelă de angajaţi (a) şi apoi ca tabelă de manageri.

8.5. OuterJoin

- Tabelul de mai jos este rezultatul rulării unei comenzi de equijoin. Se poate observa că lipsesc din acest tabel două persoane: **Georgescu** şi **Marinescu**. Deoarece **Georgescu** nu lucrează încă la nici o firmă, iar Marinescu este asignat unui firme care nu există (poate încă nu există sau a fost desfiinţată). Deci pentru aceşti doi angajaţi nu se poate găsi nici o înregistrare în tabela **Firme** pentru care condiţia de equijoin să fie îndeplinită, şi de aceea nu sunt afişaţi.
- Dacă dorim totuşi să afişăm toţi angajaţii din tabela persoane, indiferent dacă lucrează sau nu la o firmă, va trebui să putem suplini cumva această lipsă de informaţii.
- Pentru a indica lipsa de informaţii dintr-o tabelă, vom folosi secvenţa (+) imediat după numele coloanei din tabela respectivă din condiţia de join din clauza **WHERE**.

Tabela PERSOANE

IDPERSOANA	NUME	PRENUME	LOCALITATE	IDFIRM	IDJOB
1	Ionescu	Gheorghe	Brasov	22	5
2	Vasilescu	Vasile	Cluj-Napoca	15	1
3	Popescu	Ioan	Bucuresti	10	2
4	Georgescu	Maria	Iasi	30	6
5	Marinescu	Angela	Sibiu	-	3
6	Antonescu	Elena	Sibiu	10	1
7	Bischin	Paraschin	Brasov	15	-
8	Olaru	Angela	Ploiesti	22	2

Tabela FIRME

IdFirm	Nume	Localitate
10	SC Crisib SA	Sibiu
15	SC SoftCom	Alba Iulia
20	SC TimTip	Timisoara
22	Brasoveanca	Brasov

Tabela EQUIJOIN dintre tabelele PERSOANE și FIRME

Nume	Prenume	Nume
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC Crisib SA
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Olaru	Angela	Brasoveanca

- De exemplu următoarea comandă va afişa toate persoanele cu sau fără firmă corespunzătoare vom scrie în sintaxa Oracle:

SELECT a.nume, a.prenume, b.nume

FROM persoane a, firme b

WHERE a.IdFirm = b.IdFirm (+)

Nume	Prenume	NumeFirma
Antonescu	Elena	SC Crisib SA
Popescu	Ioan	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Vasilescu	Vasile	SC SoftCom
Olaru	Angela	Brasoveanca
Ionescu	Gheorghe	Brasoveanca
Marinescu	Angela	-
Georgescu	Maria	-

- Se observă că semnul **(+)** se găseşte după coloana **IdFirm** din tabela **firme** (**b**). Această tabelă fiind a doua tabelă din clauza **FROM**, vom spune că este vorba de un **LEFT OUTER JOIN**, adică sunt afişate toate înregistrările din tabela din stânga din clauza **FROM** *cu sau fără înregistrări corespunzătoare în tabela a doua*. Sintaxa ANSI foloseşte clauza **LEFT OUTER JOIN** împreună cu **ON**. Comanda anterioară este echivalentă cu următoarea comandă în sintaxa ANSI:

SELECT a.nume, a.prenume, b.nume

FROM persoane a LEFT OUTER JOIN firme b

ON (a.IdFirm = b.IdFirm)

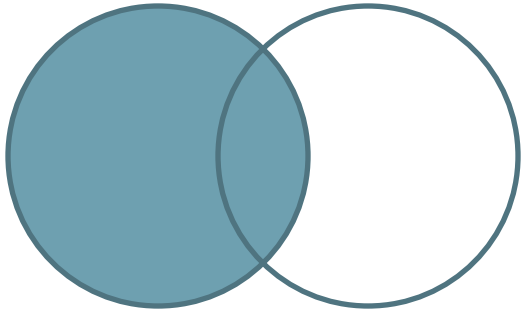


Tabela A	Tabela B
A.col1	B.col1(+)
LEFT OUTER JOIN	
	null

Left Outer Join

- Dacă vom pune semnul **(+)** în dreptul celeilalte tabele, adică vom scrie:

SELECT a.nume, a.prenume, b.nume

FROM persoane a, firme b

WHERE a.IdFirm (+) = b.IdFirm

- se vor afişa toate firmele, cu sau fără angajaţi, adică toate înregistrările din tabela aflată în dreapta în clauza **FROM** (firme), cu sau fără înregistrări corespunzătoare în cealaltă tabelă, adică cu sau fără angajaţi. Este aşadar vorba despre un **RIGHT OUTER JOIN**. Astfel în sintaxa ANSI vom scrie:

SELECT a.nume, a.prenume, b.nume

FROM persoane a RIGHT OUTER JOIN firme b

ON (a.IdFirm = b.IdFirm)

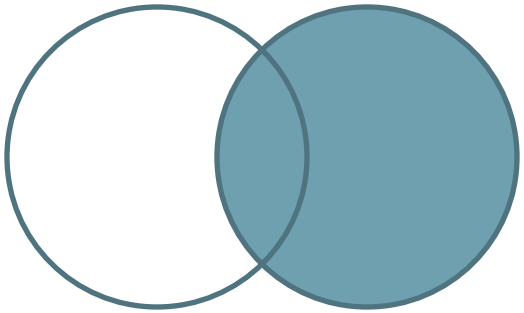


Tabela A	Tabela B
A.col1(+)	B.col1
null	RIGHT OUTER JOIN

Este importantă ordinea tabelelor în clauza **FROM** nu ordinea în care sunt scrise cele două părţi ale egalităţii din clauza **WHERE** respectiv **ON**.

Right Outer Join

Nume	Prenume	NumeFirma
Ionescu	Gheorghe	Brasoveanca
Vasilescu	Vasile	SC SoftCom
Popescu	Ioan	SC Crisib SA
Antonescu	Elena	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Olaru	Angela	Brasoveanca
-	-	SC TimTip

- Astfel comenile:

SELECT a.nume, a.prenume, b.nume

FROM persoane a, firme b

WHERE a.IdFirm = b.IdFirm (+)

- şi

SELECT a.nume, a.prenume, b.nume

FROM persoane a, firme b

WHERE b.IdFirm (+) = a.IdFirm

- sunt echivalente şi reprezintă un **LEFT OUTER JOIN**, chiar dacă semnul **(+)** apare o dată în stânga semnului de egalitate şi o dată în dreapta semnului de egalitate.

- De asemenea, deşi următoarele două comenzi sunt echivalente:

SELECT a.nume, a.prenume, b.nume

FROM persoane a, firme b

WHERE a.IdFirm = b.IdFirm (+)

- şi

SELECT a.nume, a.prenume, b.nume

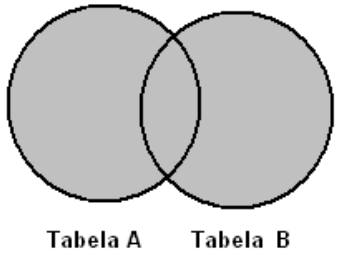
FROM firme b, persoane a

WHERE a.IdFirm = b.IdFirm (+)

- prima este un **LEFT OUTER JOIN** iar a doua este un **RIGHT OUTER JOIN**, pentru că se afişează toate înregistrările din tabela a (cea care nu are + în dreptul ei), tabelă care în prima comandă se găseşte în stânga în clauza **FROM**, iar în a doua comandă se găseşte în dreapta în clauza **FROM**.

- Cum am putea să afişăm toate înregistrările din ambele tabele, indiferent dacă ele au sau nu corespondent în cealaltă tabelă. Am dori deci să obţinem tabelul următor:

Nume	Prenume	NumeFirma
Antonescu	Elena	SC Crisib SA
Popescu	Ioan	SC Crisib SA
Bischin	Paraschin	SC SoftCom
Vasilescu	Vasile	SC SoftCom
Olaru	Angela	Brasoveanca
Ionescu	Gheorghe	Brasoveanca
Marinescu	Angela	-
Georgescu	Maria	-
-	-	SC TimTip



Full Outer Join

Full Outer Join

- Apar atât persoanele care nu sunt încă angajate, sau a căror firmă nu mai există în baza de date, dar şi firmele pentru care nu avem nici un angajat memorat în baza de date. Am fi tentaţi să scriem:

SELECT a.nume, a.prenume, b.nume

FROM firme b, persoane a

WHERE a.IdFirm (+) = b.IdFirm (+)

- adică să punem (+) în ambele părţi ale semnului de egalitate pentru că avem de suplinit lipsa de informaţii din ambele tabele.
- Însă ***sintaxa Oracle nu permite acest lucru! Singura modalitate de a obţine un FULL OUTER JOIN este de a folosi sintaxa ANSI:***

SELECT a.nume, a.prenume, b.nume

FROM persoane a FULL OUTER JOIN firme b

ON (a.IdFirm = b.IdFirm)

- Tabelul următor face o sinteză a comenzilor **JOIN** din acest capitol, punând faţă în faţă comenzile echivalente folosind cele două sintaxe.

Sintaxa Oracle	Sintaxa ANSI/ISO
Produsul Cartezian	

`SELECT p.nume, p.prenume,`
`  f.nume`
`FROM persoane p, firme f`
	`SELECT p.nume, p.prenume,`
	`  f.nume`
	`FROM persoane p` **`CROSS JOIN`**
	`  firme f`

Equijoin	

`SELECT p.nume, prenume,`
`  f.nume`
`FROM persoane p, firme f`
`WHERE p.idfirm = f.idfirm`
	`SELECT p.nume, prenume,`
	`  f.nume`
	`FROM personae p` **`JOIN`** `firme f`
	`USING` `(IdFirm)`

`SELECT p.nume, prenume,`
`  f.nume`
`FROM persoane p, firme f`
`WHERE p.idfirm = f.idfirm AND`
`  p.nume=f.nume`
	`SELECT nume, prenume,`
	`FROM persoane p` **`NATURAL JOIN`**
	`  firme f`
	`NU AFIŞEAZĂ NIMIC !!!`

`SELECT a.nume, a.prenume,`
`  b.disciplina, b.data,`
`  b.nota`
`FROM elevi a, firme b`
`WHERE a.nume=b.nume AND`
`  a.prenume=b.prenume`
	`SELECT nume, prenume,`
	`  disciplina, data, nota`
	`FROM elevi` **`NATURAL JOIN`** `note`

`SELECT p.nume, prenume,`
`  f.nume`
`FROM persoane p, firme f`
`WHERE p.IdFirm=f.IdFirm`
	`SELECT p.nume, prenume,`
	`  f.nume`
	`FROM persoane p` **`JOIN`** `firme f`
	`USING` `(IdFirm)`

Nonequijoin	
SELECT nume, prenume, disciplina, data, calificativ FROM note, calificative WHERE nota BETWEEN nota1 AND nota2	SELECT nume, prenume, disciplina, data, calificativ FROM note **JOIN** calificative **ON** (nota BETWEEN nota1 AND nota2)
Selfjoin	
SELECT a.nume ||' '|| a.prenume AS "Angajat", b.nume ||' '|| b.prenume AS "Sef" FROM angajat a, angajat b WHERE a.id_manager = b.id	SELECT a.nume ||' '|| a.prenume AS "Angajat", b.nume ||' '|| b.prenume AS "Sef" FROM angajat a **JOIN** angajat b **ON** (a.id_manager = b.id)
Outer Join	
SELECT a.nume, a.prenume, b.nume FROM persoane a, firme b WHERE a.IdFirm = b.IdFirm **(+)**	SELECT a.nume, a.prenume, b.nume FROM persoane a **LEFT OUTER JOIN** firme b on(a.IdFirm = b.IdFirm)
SELECT a.nume, a.prenume, b.nume FROM persoane a, firme b WHERE a.IdFirm **(+)** = b.IdFirm	SELECT a.nume, a.prenume, b.nume FROM persoane a **RIGHT OUTER JOIN** firme b **ON** (a.IdFirm = b.IdFirm)
NU EXSITA ECHIVALENT !	SELECT a.nume, a.prenume, b.nume FROM persoane a **FULL OUTER JOIN** firme b **ON** (a.IdFirm = b.IdFirm)

8.6. Operatorii UNION, INTERSECT, MINUS

- Un caz mai special de interogare a mai multor tabele este acela în care combinăm rezultatele a două sau mai multe interogări independente una de cealaltă.
- Operatorii folosiţi în acest scop sunt:
  - **UNION ALL** – returnează toate liniile returnate de de interogările pe care le leagă, inclusiv duplicatele (dacă cele două subinterogări returnează amânduua o aceeaşi linie, acest operator le va include pe ambele în rezultat)
  - **UNION** – asemănător cu operatorul anterior însă sunt eliminate duplicatele
  - **INTERSECT** – afişează liniile returnate de ambele interogări
  - **MINUS** – returnează liniile care sunt returnate de prima interogare dar nu sunt returnate şi de a doua interogare.
- **Atenţie!** Numărul de coloane şi tipul coloanelor returnate de cele două nterogări trebuie să fie acelaţi, chiar dacă au alt nume.

- Sintaxa folosirii acestor operatori este:

interogare operator interogare

- Vom exemplifica utilizarea acestor operatori pe două tabele formale

Tabela **A**

ColA	ColB
A	10
A	15
B	7
C	20
C	30
D	40

Tabela **B**

ColC	ColD
A	8
B	6
B	7
C	15
C	30
C	60
D	8

Tabela **C**

ColE	ColF
A	10
B	6
C	20
D	8
E	10

- Interogarea

  SELECT ColA, ColB FROM A

  UNION ALL

  SELECT ColC, ColD FROM B
- va afişa tabela alăturată:

COLA	COLB
A	10
A	15
B	7
C	20
C	30
D	40
A	8
B	6
B	7
C	30
C	15
C	60
D	8

Utilizarea operatorului **UNION ALL**

- . Comanda următoare va elimina duplicatele, rezultatul fiind cel din tabela alăturată:

  SELECT ColA, ColB FROM A

  UNION

  SELECT ColC, ColD FROM B

COLA	COLB
A	8
A	10
A	15
B	6
B	7
C	15
C	20
C	30
C	60
D	8
D	40

Utilizarea operatorului **UNION**

SELECT ColA, ColB FROM A

INTERSECT

SELECT ColC, ColD FROM B

COLA	COLB
B	7
C	30

Utilizarea operatorului **INTERSECT**

SELECT ColA, ColB FROM A

MINUS

SELECT ColC, ColD FROM B

COLA	COLB
A	10
A	15
C	20
D	40

Utilizarea operatorului **MINUS**

- Un exemplu practic de folosire a acestor operatori poate fi dat dacă ne imaginăm că pentru cercul de informatică de la liceul vostru, profesorul coordonator de cerc a întocmit un tabel **info** conţinând **numele**, **prenumele** şi **clasa** elevilor înscrişi la acest cerc. Similar, profesorul de la cercul de matematică a realizat un tabel **mate** cu aceeleaşi coloane, memorând elevii de la cercul de matematică.
- Directorul şcolii doreşte, de exemplu, o listă cu elevii înscrişi la ambele cercuri. Nu aveţi altceva de făcut decât să scrieţi următoarea comandă:

SELECT nume, prenume, clasa FROM info

INTERSECT

SELECT nume, prenume, clasa FROM mate

- Desigur puteţi combina mai mult de două interogări folosind operatorii **UNION**, **INTERSECT** şi **MINUS**. Implicit operatorii sunt evaluaţi de jos în sus, însă puteţi indica ordinea de efectuare a acestor operaţii prin folosirea parantezelor. De exemplu comanda

SELECT colA, colB FROM A

UNION

SELECT colC, colD FROM B

INTERSECT

SELECT colE, colF FROM C

COLA	COLB
A	10
B	6
C	20
D	8

```
SELECT colA, colB FROM A
UNION
(SELECT colC, colD FROM B
 INTERSECT
 SELECT colE, colF FROM C)
```

COLA	COLB
A	10
A	15
B	6
B	7
C	20
C	30
D	8
D	40

În această lecție am învățat despre:

- Cum se pot prelua informații din mai multe tabele simultan
- Care sunt tipurile de join cunoscute
- Cum se realizează fiecare tip de join folosind sintaxa Oracle
- Cum se realizează fiecare tip de join folosind sintaxa ANSI
- Cum se folosesc operatorii UNION, INTERSECT, MINUS

SFÂRȘIT