



Sisteme de baze de date

Curs 5 – Limbajul SQL: prelucrarea datelor (2/3).

Sorina Preduț

sorina.predut@unibuc.ro

Universitatea din București



Regăsirea datelor din 2 sau mai multe tabele

- O **joncțiune** (operația de compunere) este o interogare care regăsește înregistrări din 2 sau mai multe tabele. Capacitatea de a realiza o joncțiune între 2 sau mai multe tabele reprezintă una dintre cele mai puternice facilități ale unui sistem relațional.
- Legătura dintre înregistrările tabelor se realizează prin **existența unor câmpuri comune** caracterizate prin domenii de definiție compatibile (**chei primare sau străine**). Realizarea unei joncțiuni se poate face precizând în clauza FROM numele tabelor utilizate, iar în clauza WHERE criteriul de compunere.



Produsul a 2 sau mai multe tabele

- În cazul în care în interogare se specifică mai multe tabele și nu este inclusă o clauză WHERE, interogarea va genera **produsul cartezian al tabelelor**. Acesta va conține toate combinațiile posibile de înregistrări din tabelele componente. Astfel, produsul cartezian a 2 tabele care conțin 100, respectiv 50 de înregistrări va avea dimensiunea de 5.000 de înregistrări.



Produsul a 2 sau mai multe tabele - cont.

- De exemplu, să considerăm tabela CATEDRA cu următoarele 4 înregistrări:

COD_CATEDRA	NUME	PROFIL
-----	-----	-----
10	INFORMATICA	TEHNIC
20	ELECTRONICA	TENHIC
30	AUTOMATICA	TENHIC
40	FINANTE	ECONOMIC

Produsul a 2 sau mai multe tabele - cont.

➤ și tabela PROFESOR cu următoarele 7 înregistrări:

COD	NUME	PRENUME	DATA_NAST	GRAD	SEF	SALARIU	PRIMA	COD_CATEDRA
----	-----	-----	-----	-----	----	-----	-----	-----
100	GHEORGHIU	STEFAN	11-AUG-46	PROF	3000	3500		10
101	MARIN	VLAD	19-APR-45	PROF	100	2500		20
102	GEORGESCU	CRISTIANA	30-OCT-51	CONF	100	2800	200	30
103	IONESCU	VERONICA		ASIST	102	1500		10
104	ALBU	GHEORGHE		LECT	100	2200	2500	20
105	VOINEA	MIRCEA	15-NOV-65	ASIST	100	1200	150	10
106	STANESCU	MARIA	05-DEC-69	ASIST	103	1200	600	20



Produsul a 2 sau mai multe tabele - cont.

- Atunci următoarea interogare va genera produsul cartezian al tabelelor, adică va avea ca rezultat $7 \times 4 = 28$ de rânduri ce vor conține toate combinațiile posibile de înregistrări din cele 2 tabele:

```
SELECT * FROM profesor, catedra;
```

[Rezolvare în varianta standard](#)



Produsul a 2 sau mai multe tabele - cont.

- Dacă în lista de SELECT sunt specificate coloanele selectate, atunci numele acestora trebuie să fie unic în cadrul tuturor tabelor.
- Dacă \exists un nume de coloană care apare în mai mult de un tabel, atunci, pentru evitarea ambiguității, trebuie specificat și tabelul din care face parte coloana în cauză.
- De exemplu, în următoarea interogare pentru coloanele `cod_catedra` și `nume` trebuie specificate tabelele din care fac parte:

```
SELECT profesor.nume, prenume, catedra.cod_catedra, catedra.nume  
FROM profesor, catedra;
```

NUME	PRENUME	COD_CATEDRA	NUME			
-----	-----	-----	-----			
GHEORGHIU	STEFAN	10	INFORMATICA			
MARIN	VLAD	10	INFORMATICA	GEORGESCU	CRISTIANA	30
GEORGESCU	CRISTIANA	10	INFORMATICA	IONESCU	VERONICA	30
IONESCU	VERONICA	10	INFORMATICA	ALBU	GHEORGHE	30
ALBU	GHEORGHE	10	INFORMATICA	VOINEA	MIRCEA	30
VOINEA	MIRCEA	10	INFORMATICA	STANESCU	MARIA	30
STANESCU	MARIA	10	INFORMATICA	GHEORGHIU	STEFAN	40
GHEORGHIU	STEFAN	20	ELECTRONICA	MARIN	VLAD	40
MARIN	VLAD	20	ELECTRONICA	GEORGESCU	CRISTIANA	40
GEORGESCU	CRISTIANA	20	ELECTRONICA	IONESCU	VERONICA	40
IONESCU	VERONICA	20	ELECTRONICA	ALBU	GHEORGHE	40
ALBU	GHEORGHE	20	ELECTRONICA	VOINEA	MIRCEA	40
VOINEA	MIRCEA	20	ELECTRONICA	STANESCU	MARIA	40
STANESCU	MARIA	20	ELECTRONICA			
GHEORGHIU	STEFAN	30	AUTOMATICA			
MARIN	VLAD	30	AUTOMATICA			



Produsul a 2 sau mai multe tabele - cont.

- În general, pentru a scurta textul comenzii, în astfel de cazuri se folosesc de obicei **aliasuri** pentru numele tabelelor, care pot fi folosite în interogare.
- Astfel interogarea de mai înainte se mai poate scrie:

```
SELECT p.num, prenume, c.cod_catedra, c.num  
FROM profesor p, catedra c;
```
- În general, produsul cartezian este rar folosit, având o utilitate practică redusă.



Joncțiuni

- Pentru a realiza o joncțiune între 2 sau mai multe tabele se utilizează clauza WHERE a interogărilor pe aceste tabele.
- Pentru a realiza o compunere între n tabele, va fi nevoie de cel puțin $n - 1$ condiții de join.
- În funcție de criteriul de compunere, se disting mai multe tipuri de joncțiuni:
 - **joncțiuni echivalente (EQUI - JOIN) sau joncțiuni interne (INNER JOIN),**
 - **joncțiuni neechivalente (NON EQUI - JOIN),**
 - **joncțiuni externe (OUTER JOIN),**
 - **auto-joncțiuni (SELF JOIN).**

Joncțiuni echivalente

- O echi-joncțiune conține operatorul egalitate (=) în clauza WHERE, combinând înregistrările din tabele care au valori egale pentru coloanele specificate.
- De exemplu, pentru a afișa cadrele didactice și numele catedrei din care fac parte se combină înregistrările din cele 2 tabele pentru care codul catedrei este același.

```
SELECT p.num, p.prenume, c.num  
FROM profesor p, catedra c  
WHERE p.cod_catedra = c.cod_catedra;
```

NUME	PRENUME	NUME
-----	-----	-----
GHEORGHIU	STEFAN	INFORMATICA
IONESCU	VERONICA	INFORMATICA
VOINEA	MIRCEA	INFORMATICA
MARIN	VLAD	ELECTRONICA
STANESCU	MARIA	ELECTRONICA
ALBU	GHEORGHE	ELECTRONICA
GEORGESCU	CRISTIANA	AUTOMATICA



Joncțiuni neechivalente

- **Joncțiunile neechivalente** sunt acelea care nu folosesc în clauza WHERE operatorul egal. Operatorii cei mai utilizați în cazul joncțiunilor neechivalente sunt: <, >, <=, >=, <>, BETWEEN ... AND ...
- Pentru a exemplifica un astfel de tip de joncțiune considerăm tabela GRADSAL ce conține pragul minim și pragul maxim al salariului dintr-un anumit grad de salarizare:

GRAD_SALARIZARE	PRAG_MIN	PRAG_MAX
-----	-----	-----
1	500	1500
2	1501	2000
3	2001	2500
4	2501	3500
5	3501	10000



Joncțiuni neechivalente - cont.

- Evident, între tabelele PROFESOR și GRADSAL nu are sens definirea unei joncțiuni echivalente deoarece nu există o coloană din tabela PROFESOR căreia să-i corespundă o coloană din tabela GRADSAL.
- Exemplul următor ilustrează definirea unei joncțiuni neechivalente care evaluează gradul de salarizare a cadrelor didactice, prin încadrarea salariului acestora într-un interval stabilit de pragul minim și pragul maxim:



Joncțiuni neechivalente - cont.

```
SELECT p.num, p.grad, p.salariu, g.grad_salarizare
FROM profesor p, gradsal g
WHERE p.salariu BETWEEN g.prag_min AND g.prag_max;
```

NUME	GRAD	SALARIU	GRAD_SALARIZARE
-----	----	-----	-----
IONESCU	ASIST	1500	1
VOINEA	ASIST	1200	1
STANESCU	ASIST	1200	1
MARIN	PROF	2500	3
ALBU	LECT	2200	3
GHEORGHIU	PROF	3000	4
GEORGESCU	CONF	2800	4



Joncțiuni externe

- Dacă într-o joncțiune de tipul celor prezentate până acum una sau mai multe înregistrări nu satisfac condiția de compunere specificată în clauza WHERE, atunci ele nu vor apărea în rezultatul interogării.
Acele înregistrări pot apărea însă dacă se folosește joncțiunea externă.
- **Joncțiunea externă** returnează toate înregistrările care satisfac condiția de joncțiune plus acele înregistrări dintr-un tabel ale căror valori din coloanele după care se face legătura nu se regăsesc în coloanele corespunzătoare ale nici unei înregistrări din celălalt tabel.



Joncțiuni externe - cont.

- Pentru a realiza o joncțiune externă între tabelele A și B ce returnează **toate înregistrările din tabela A se utilizează semnul (+) în dreapta tabelului B.**
Pentru fiecare înregistrare din tabela A care nu satisface condiția de compunere pentru nici o înregistrare din tabela B se va crea în tabela B o înregistrare nulă care va fi compusă cu înregistrarea din tabela A.
- Invers, pentru a realiza o joncțiune externă între tabelele A și B ce returnează **toate înregistrările din tabela B, se utilizează semnul (+) în dreapta tabelului A.**



Joncțiuni externe - cont.

- În interogarea utilizată pentru a exemplifica joncțiunea echivalentă, se observă că au fost selectate numai catedrele în care există cadre didactice.

Pentru a afișa toate catedrele, indiferent dacă ele cuprind sau nu cadre didactice, se folosește următoarea interogare:

```
SELECT p.nume, p.prenume, c.nume  
FROM profesor p, catedra c  
WHERE p.cod_catedra(+) = c.cod_catedra;
```

- Se observă că ultima înregistrare (ce corespunde catedrei de finanțe care nu are în componență nici un cadru didactic) va avea coloanele corespunzătoare primului tabel completate cu Null.

[Rezolvare în varianta standard](#)



Joncțiuni externe - cont.

NUME -----	PRENUME -----	NUME -----
GHEORGHIU	STEFAN	INFORMATICA
IONESCU	VERONICA	INFORMATICA
VOINEA	MIRCEA	INFORMATICA
MARIN	VLAD	ELECTRONICA
STANESCU	MARIA	ELECTRONICA
ALBU	GHEORGHE	ELECTRONICA
GEORGESCU	CRISTIANA	AUTOMATICA FINANTE



Joncțiuni externe - cont.

- Pentru a afișa toate cadrele didactice, inclusiv cadrele didactice a căror catedră este necunoscută, se folosește următoarea interogare:

```
SELECT p.num, p.prenume, c.num  
FROM profesor p, catedra c  
WHERE p.cod_catedra = c.cod_catedra(+);
```

[Rezolvare în varianta standard](#)



Joncțiuni externe - cont.

- Să se afișeze cadrele didactice și numele catedrei din care fac parte. Rezultatul va conține și catedrele care nu cuprind cadre didactice, precum și cadrele didactice a căror catedră este necunoscută (se folosește operatorul UNION menționat în Curs1, precum și începând cu [slide-ul 44](#) al cursului curent).

```
SELECT p.num, p.prenume, c.num  
FROM profesor p, catedra c  
WHERE p.cod_catedra = c.cod_catedra(+)  
UNION  
SELECT p.num, p.prenume, c.num  
FROM profesor p, catedra c  
WHERE p.cod_catedra(+) = c.cod_catedra;
```

[Rezolvare în varianta standard](#)



Joncțiuni externe - cont.

- Folosirea operatorului de joncțiune externă are următoarele restricții:
 - Operatorul (+) poate fi plasat în oricare parte a condiției din clauza WHERE, însă nu în ambele părți.

Tabelul de partea căruia este amplasat acest operator va crea înregistrări nule care vor fi compuse cu înregistrările din celălalt tabel care nu satisfac condiția de compunere.
 - Dacă tabelele A și B au condiții multiple de joncțiune, atunci operatorul (+) trebuie utilizat în toate aceste condiții;
 - Într-o singură interogare nu se poate realiza o joncțiune externă a unui tabel cu mai multe tabele.



Joncțiuni externe - cont.

- O condiție care conține operatorul (+) nu poate fi combinată cu o altă condiție ce utilizează operatorul IN.
- O condiție care conține operatorul (+) nu poate fi combinată cu o altă condiție prin operatorul OR.



Auto-joncțiuni

- **Auto-joncțiunea** reprezintă joncțiunea unui tabel cu el însuși.
Pentru ca rândurile dintr-un tabel să poată fi compuse cu rânduri din același tabel, în clauza FROM a interogării numele tabelului va apărea de mai multe ori, urmat de fiecare dată de un alias.

Auto-joncțiuni - cont.

- De exemplu, pentru a selecta toate cadrele didactice care au un șef direct și numele acestui șef se folosește următoarea auto-joncțiune:

```
SELECT p.num, p.prenume, s.num, s.prenume  
FROM profesor p, profesor s  
WHERE p.sef = s.cod;
```

NUME	PRENUME	NUME	PRENUME
-----	-----	-----	-----
MARIN	VLAD	GHEORGHIU	STEFAN
GEORGESCU	CRISTIANA	GHEORGHIU	STEFAN
ALBU	GHEORGHE	GHEORGHIU	STEFAN
VOINEA	MIRCEA	GHEORGHIU	STEFAN
IONESCU	VERONICA	GEORGESCU	CRISTIANA
STANESCU	MARIA	IONESCU	VERONICA



Auto-joncțiuni - cont.

- Auto-joncțiunea poate fi folosită și pentru verificarea corectitudinii interne a datelor. De exemplu, este puțin probabil să existe 2 cadre didactice care au cod diferit, dar în schimb au același nume, prenume și dată de naștere.

Pentru a verifica dacă există astfel de înregistrări se folosește interogarea:

```
SELECT a.num, a.prenume
FROM profesor a, profesor b
WHERE    a.num = b.num AND
         a.prenume = b.prenume AND
         a.data_nast = b.data_nast AND
         a.cod <> b.cod;
```



Joncțiuni - standard SQL3

- În sistemul Oracle11g, joncțiunea se poate implementa și prin intermediul unei sintaxe specifice acesteia, introdusă de **standardul SQL3**.

Față de compunerea realizată prin specificarea condiției în clauza WHERE, această sintaxă nu aduce beneficii în privința performanței.

Conform standardului SQL3, tipurile de compunere descrise anterior se pot rescrie, echivalent, prin cuvintele cheie:

- **NATURAL JOIN,**
- **FULL OUTER JOIN,**
- **JOIN** cu clauzele **USING** și **ON**.



Joncțiuni - cont.

- **Sintaxa corespunzătoare standardului SQL3** este următoarea:

```
SELECT tabel_1.nume_coloană, tabel_2.nume_coloană
FROM tabel_1
[NATURAL JOIN tabel_2]
| [JOIN tabel_2 USING (nume_coloană) ]
| [JOIN tabel_2 ON (tabel_1.nume_coloană = tabel_2.nume_coloană) ] |
[LEFT | RIGHT | FULL OUTER JOIN tabel_2
ON (tabel_1.nume_coloană = tabel_2.nume_coloană) ];
```

- **Utilizând aceste modalități de specificare a joncțiunii, clauza FROM înlocuiește clauzele FROM și WHERE din exemplele anterioare.**
- În lista SELECT sunt enumerate coloanele, calificate cu numele tabelelor din care provin, ale căror date sunt regăsite.



NATURAL JOIN

- Joncțiunea poate fi implementată cu ajutorul clauzei NATURAL JOIN în cazul în care **coloanele pe baza cărora se realizează compunerea au același nume în ambele tabele și același tip de date.**
- Clauza determină selectarea liniilor din cele 2 tabele, care au valori egale în toate coloanele al căror nume coincid.
- Dacă tipurile de date ale coloanelor cu nume identice sunt diferite, va fi returnată o eroare.
- **Coloanele având același nume în cele 2 tabele trebuie să nu conțină calificatori** (să nu fie precedate de numele sau alias-ul tabelului corespunzător).



JOIN ... USING

- Pentru a putea preciza în mod explicit coloanele pe baza cărora are loc compunerea, se poate utiliza clauza **JOIN tabel_2 USING nume_coloană**.
- Aceasta efectuează un EQUI-JOIN pe baza coloanei cu numele specificat în sintaxă.
- Clauza este utilă atunci când \exists coloane având același nume, dar care nu trebuie să intre în condiția de join.
- **Coloanele referite în clauza USING nu pot conține calificatori în nicio apariție a lor în instrucțiunea SQL.**
- Clauzele NATURAL JOIN și USING nu pot exista simultan în aceeași instrucțiune SQL.



JOIN ... ON

- Pentru a delimita condițiile de join și cele de căutare sau filtrare, este utilă clauza **JOIN tabel_2 ON tabel_1.nume_coloană = tabel_2.nume_coloană**, care efectuează un EQUI-JOIN pe baza condiției exprimate în opțiunea ON.
- Condițiile de căutare sau filtrare vor apărea distinct în clauza **WHERE**.



Exemple

- Să se afișeze cadrele didactice și numele catedrei din care fac parte pentru profesorii având codurile 105 și 106.

Se vor furniza 2 variante de rezolvare, utilizând sintaxa specifică standardului SQL3.

Varianta în care se folosește NATURAL JOIN

```
SELECT nume, prenume  
FROM profesor  
NATURAL JOIN catedra  
WHERE cod IN (105, 106);
```

nu poate fi folosită în acest caz deoarece avem 2 coloane cu același nume (nume și cod). Ar returna linii doar dacă valorile coincid pe ambele coloane.



Exemple - cont.

```
SELECT profesor.ume, preume, catedra.ume
FROM profesor
JOIN catedra USING (cod_catedra)
WHERE cod IN (105, 106);
```

```
SELECT p.ume, preume, c.ume
FROM profesor p
JOIN catedra c ON (p.cod_catedra = c.cod_catedra)
WHERE cod IN (105, 106);
```




Exemple - cont.

- Condițiile suplimentare ale unei interogări pot fi specificate atât în clauza WHERE, cât și în clauza ON.
- Exemplu. Să se afișeze cadrele didactice și numele catedrei din care fac parte pentru profesorii care au fost angajații în cursul lunii trecute (a se vedea Curs3, slide-urile 57, 58 pt. schema completă).

```
SELECT p.num, p.prenume, c.num  
FROM profesor p  
JOIN catedra c ON (p.cod_catedra = c.cod_catedra)  
AND TO_CHAR(p.data_angajarii, 'MM/YYYY') =  
TO_CHAR(ADD_MONTHS(SYSDATE, -1), 'MM/YYYY');
```

- Instrucțiunea anterioară este corectă și este echivalentă cu comanda obținută înlocuind operatorul AND cu clauza WHERE.



OUTER JOIN ... ON

- Prin intermediul clauzelor **LEFT**, **RIGHT** și **FULL OUTER JOIN tabel_2 ON (tabel_1.nume_coloană = tabel_2.nume_coloană)** se efectuează compunerea externă la stânga, dreapta, respectiv compunerea externă completă pe baza condiției exprimate în clauza ON.
- Joncțiunea (compunerea) externă completă (**FULL OUTER JOIN**) returnează rezultatele unei joncțiuni interne (**INNER JOIN**), dar și cele ale joncțiunilor externe la stânga și la dreapta.



Exemple

- Exemplul din [slide-ul 19](#) al cursului curent se poate rezolva în standardul SQL3 astfel:
(Pentru a afișa toate cadrele didactice, indiferent dacă aparțin unei catedre.)

```
SELECT p.nume, p.prenume, c.nume  
FROM profesor p  
LEFT OUTER JOIN catedra c ON (p.cod_catedra = c.cod_catedra);
```



Exemple - cont.

- Exemplul din [slide-ul 17](#) al cursului curent se poate rezolva în standardul SQL3 astfel:
(Pentru a afișa toate catedrele, indiferent dacă ele cuprind sau nu cadre didactice)

```
SELECT p.num, p.prenume, c.num  
FROM profesor p  
RIGHT OUTER JOIN catedra c ON (p.cod_catedra = c.cod_catedra);
```



Exemple - cont.

- Exemplul din [slide-ul 20](#) al cursului curent se poate rezolva în standardul SQL3 astfel:
(Să se afișeze cadrele didactice și numele catedrei din care fac parte.

Rezultatul va conține și catedrele care nu cuprind cadre didactice, precum și cadrele didactice a căror catedră este necunoscută)

```
SELECT p.nume, p.prenume, c.nume  
FROM profesor p  
FULL OUTER JOIN catedra c ON (p.cod_catedra = c.cod_catedra);
```



Produsul cartezian

- Joncțiunea este o submulțime a unei combinații mai generale de 2 tabele, și anume produsul cartezian.
- De obicei, produsul cartezian generează un nr. mare de linii și constituie o operație puțin utilă în practică.
- Un caz de utilizare a acestei operații apare atunci când este necesară obținerea unui volum mare de date utile pentru anumite teste de simulare.



Produsul cartezian

- Similar aplicării sale în cazul mulțimilor, produsul cartezian a 2 tabele este un alt tabel care conține toate perechile posibile de linii din cele două tabele.
Coloanele tabelului produs sunt toate coloanele din primul tabel, urmate de toate coloanele din cel de-al doilea.
- Dacă se specifică o interogare asupra a 2 tabele, fără o clauză WHERE, limbajul SQL va produce drept rezultat al interogării produsul cartezian al acestora.



Produsul cartezian

- Pentru produsul cartezian, standardul SQL3 prezintă un format special al instrucțiunii SELECT:

```
SELECT tabel_1.nume_coloană, tabel_2.nume_coloană  
FROM tabel_1 CROSS JOIN tabel_2;
```

- Exemplul din [slide-ul 6](#) al cursului curent se poate rezolva în standardul SQL3 astfel:

```
SELECT *  
FROM profesor CROSS JOIN catedra;
```




SELECT - generare rezultate joncțiune

- Procedura de generare a rezultatelor unei instrucțiuni SELECT referitoare la o joncțiune este următoarea:
 1. formarea produsului cartezian al tabelelor enumerate în clauza FROM;
 2. dacă \exists o clauză WHERE, condiția de căutare se aplică fiecărui rând al tabelului produs, fiind reținute cele care satisfac condiția;
 3. pentru fiecare linie rămasă, se determină valoarea fiecărui element din lista SELECT, pentru a produce o singură linie în tabelul rezultat;
 4. dacă a fost specificat cuvântul cheie DISTINCT, se elimină liniile duplicat din tabelul rezultat;
 5. dacă există o clauză ORDER BY, se sortează tabelul rezultat conform criteriilor specificate.



SELECT - generare rezultate joncțiune - cont.

- În termeni de algebră relațională, pasul 2 descris anterior creează o restricție a produsului cartezian, iar etapele 3 și 4 sunt echivalente unei proiecții a restricției asupra coloanelor menționate în lista SELECT.



Operatori pentru mulțimi

- Operatorii de mulțimi combină 2 sau mai multe interogări, efectuând operații specifice mulțimilor: reuniune, intersecție, diferență.
- Acești operatori se mai numesc și operatori verticali deoarece combinarea celor 2 interogări se face coloană cu coloană.
Din acest motiv, **nr. total de coloane și tipurile de date ale coloanelor corespondente din cele 2 interogări trebuie să coincidă.**



Operatori pentru mulțimi - cont.

- Există următorii operatori pentru mulțimi:
 1. **UNION** – Returnează rezultatele a 2 sau mai multe interogări eliminând toate înregistrările duplicat;
 2. **UNION ALL** – Returnează rezultatele a 2 sau mai multe interogări incluzând înregistrările duplicat;
 3. **INTERSECT** – Returnează toate înregistrările distincte găsite în ambele interogări;
 4. **MINUS** – Returnează toate înregistrările distincte care se găsesc în prima interogare dar nu și în a doua interogare.

[Înapoi la joncțiuni externe](#)

Operatori pentru mulțimi - cont.

- Să considerăm, de exemplu, următoarele interogări:

```
SELECT grad, salariu  
FROM profesor  
WHERE cod_catedra = 10;
```

GRAD	SALARIU
----	-----
PROF	3000
ASIST	1500
ASIST	1200

```
SELECT grad, salariu  
FROM profesor  
WHERE cod_catedra = 20;
```

GRAD	SALARIU
----	-----
PROF	2500
LECT	2200
ASIST	1200



Operatori pentru mulțimi - cont.

- În continuare exemplificăm fiecare dintre operatorii pentru mulțimi aplicați acestor interogări:

- ```
SELECT grad, salariu
FROM profesor
WHERE cod_catedra = 10
UNION
SELECT grad, salariu
FROM profesor
WHERE cod_catedra = 20;
```

| GRAD  | SALARIU |
|-------|---------|
| ----  | -----   |
| ASIST | 1200    |
| ASIST | 1500    |
| LECT  | 2200    |
| PROF  | 2500    |
| PROF  | 3000    |



## Operatori pentru mulțimi - cont.

➤ `SELECT grad, salariu`  
`FROM profesor`  
`WHERE cod_catedra = 10`  
**`UNION ALL`**  
`SELECT grad, salariu`  
`FROM profesor`  
`WHERE cod_catedra = 20;`

| GRAD  | SALARIU |
|-------|---------|
| ----  | -----   |
| PROF  | 3000    |
| ASIST | 1500    |
| ASIST | 1200    |
| PROF  | 2500    |
| LECT  | 2200    |
| ASIST | 1200    |



## Operatori pentru mulțimi - cont.

➤ `SELECT grad, salariu`  
`FROM profesor`  
`WHERE cod_catedra = 10`  
**INTERSECT**  
`SELECT grad, salariu`  
`FROM profesor`  
`WHERE cod_catedra = 20;`

| GRAD  | SALARIU |
|-------|---------|
| ----  | -----   |
| ASIST | 1200    |





## Operatori pentru mulțimi - cont.

➤ `SELECT grad, salariu`  
`FROM profesor`  
`WHERE cod_catedra = 10`  
**MINUS**  
`SELECT grad, salariu`  
`FROM profesor`  
`WHERE cod_catedra = 20;`

| GRAD  | SALARIU |
|-------|---------|
| ----  | -----   |
| ASIST | 1500    |
| PROF  | 3000    |



## Operatori pentru mulțimi - cont.

- Există următoarele reguli de folosire a operatorilor pentru mulțimi:
  - interogările trebuie să conțină același număr de coloane;
  - coloanele corespondente trebuie să aibă același tip de dată;
  - în rezultat vor apărea numele coloanelor din prima interogare, nu cele din a doua interogare chiar dacă aceasta folosește alias-uri; de exemplu:

|                           |       |
|---------------------------|-------|
| SELECT cod FROM profesor  | COD   |
| MINUS                     | ----- |
| SELECT sef FROM profesor; | 101   |
|                           | 104   |
|                           | 105   |
|                           | 106   |

## Operatori pentru mulțimi - cont.

- clauza ORDER BY poate fi folosită o singură dată într-o interogare care folosește operatori de mulțimi; atunci când se folosește, ea trebuie poziționată la sfârșitul comenzii; de exemplu:

```
SELECT grad, salariu
FROM profesor
WHERE cod_catedra = 10
UNION
SELECT grad, salariu
FROM profesor
WHERE cod_catedra = 20
ORDER BY 2;
```

| GRAD  | SALARIU |
|-------|---------|
| ----  | -----   |
| ASIST | 1200    |
| ASIST | 1500    |
| LECT  | 2200    |
| PROF  | 2500    |
| PROF  | 3000    |



## Operatori pentru mulțimi - cont.

- operatorii pentru mulțimi pot fi utilizați în subinterogări;
- pentru a modifica ordinea de execuție este posibilă utilizarea parantezelor;  
de exemplu:

```
SELECT grad FROM profesor WHERE cod_catedra = 10
INTERSECT
SELECT grad FROM profesor WHERE cod_catedra = 20
UNION
SELECT grad FROM profesor WHERE cod_catedra = 30;
```

**GRAD**  
-----  
**ASIST**  
**CONF**  
**PROF**



## Operatori pentru mulțimi - cont.

```
SELECT grad FROM profesor WHERE cod_catedra = 10
INTERSECT
(SELECT grad FROM profesor WHERE cod_catedra = 20
UNION
SELECT grad FROM profesor WHERE cod_catedra = 30);
```

GRAD  
-----  
ASIST  
PROF



## Subinterogări și operatorii ANY, ALL, EXISTS

- O **subinterogare** este o comandă SELECT inclusă în altă comandă SELECT.
- Rezultatele subinterogării sunt transmise celeilalte interogări și pot apărea în cadrul clauzelor WHERE, HAVING sau FROM.
- Subinterogările sunt utile pentru a scrie interogări bazate pe o condiție în care valoarea de comparație este necunoscută.

Această valoare poate fi aflată folosind o subinterogare.

De exemplu:

```
SELECT coloane
FROM tabel
WHERE coloana = (SELECT coloane
 FROM tabel WHERE condiție)
```



## Subinterogări - cont.

- Subinterogarea, denumită și **interogare interioară (inner query)**, generează valorile pentru condiția de căutare a instrucțiunii SELECT care o conține, denumită **interogare exterioară (outer query)**.
- Instrucțiunea SELECT exterioară depinde de valorile generate de către interogarea interioară.
- În general, interogarea interioară se execută prima și rezultatul acesteia este utilizat în interogarea exterioară.
- Rezultatul interogării exterioare depinde de numărul valorilor returnate de către interogarea interioară. În acest sens, putem distinge:
  - **Subinterogări care returnează un singur rând;**
  - **Subinterogări care returnează mai multe rânduri.**



## Subinterogări - cont.

- Dpdv al ordinii de evaluare a interogărilor putem clasifica subinterogările în:
  1. **Subinterogări simple/nesincronizate** – în care interogarea interioară este evaluată prima, independent de interogarea exterioară (interogarea interioară se execută o singură dată);
  2. **Subinterogări corelate/sincronizate** – în care valorile returnate de interogarea interioară depind de valorile returnate de interogarea exterioară (interogarea interioară este evaluată pentru fiecare înregistrare a interogării exterioare).





## Subinterogări - cont.

- Subinterogările sunt îndeosebi utilizate atunci când se dorește ca o interogare să regăsească înregistrări dintr-o tabelă care îndeplinesc o condiție ce depinde la rândul ei de valori din aceeași tabelă.
- **Observație:** Clauza ORDER BY nu poate fi utilizată într-o subinterogare. Regula este că poate exista doar o singură clauză ORDER BY pentru o declarație SELECT și, dacă este specificată, trebuie să fie ultima clauză din comanda SELECT. Prin urmare, clauza ORDER BY nu poate fi specificată decât în interogarea cea mai din exterior.



## Subinterogări care returnează un singur rând

- În acest caz condiția, din clauza WHERE sau HAVING a interogării exterioare utilizează operatorii: =, <, <=, >, >=, <> care operează asupra unei subinterogări ce returnează o singură valoare.
- Interogarea interioară poate conține condiții complexe formate prin utilizarea condițiilor multiple de interogare cu ajutorul operatorilor AND și OR sau prin utilizarea funcțiilor agregat.



## Subinterogări care returnează un singur rând

- Următoarea interogare selectează cadrele didactice care au salariul minim.  
Salariul minim este determinat de o subinterogare ce returnează o singură valoare.

```
SELECT nume, prenume, salariu
FROM profesor
WHERE salariu = (SELECT MIN(salariu)
 FROM profesor);
```

| NUME     | PRENUME | SALARIU |
|----------|---------|---------|
| -----    | -----   | -----   |
| VOINEA   | MIRCEA  | 1200    |
| STANESCU | MARIA   | 1200    |



## Subinterogări care returnează un singur rând

- Procesul de evaluare al acestei interogări se desfășoară astfel:
  - se evaluează în primul rând interogarea interioară:  
Valoarea obținută este `MIN(salariu) = 1200`
  - rezultatul evaluării interogării interioare devine condiție de căutare pentru interogarea exterioară și anume:  

```
SELECT nume, prenume, salariu FROM profesor WHERE salariu = 1200;
```
- În cazul în care interogarea interioară nu întoarce nici o înregistrare, interogarea exterioară nu va selecta la rândul ei nici o înregistrare.



## Subinterogări care returnează un singur rând

- **Observație:** Dacă care se utilizează operatorii: =, <, <=, >, >=, <> în condiția interogării exterioare, atunci interogarea interioară trebuie în mod obligatoriu să returneze o singură valoare.

În caz contrar va apărea un mesaj de eroare, ca în exemplul următor:

```
SELECT nume, prenume, salariu
FROM profesor
WHERE salariu = (SELECT MIN (salariu)
 FROM profesor
 GROUP BY grad);
```

ERROR:

ORA-01427: single-row subquery returns more than one row



## Subinterogări care returnează un singur rând

- Subinterogările pot fi folosite nu numai în clauza WHERE a interogării exterioare, ci și în clauza HAVING.

Următoarea interogare afișează toate gradele didactice pentru care salariul minim este mai mare decât salariul mediu al tuturor cadrelor didactice.

```
SELECT grad
FROM profesor
GROUP BY grad
HAVING MIN(salariu) > (SELECT AVG(salariu)
 FROM profesor);
```

**GRAD**  
-----  
**CONF**  
**LECT**  
**PROF**



## Subinterog. care returnează mai multe rânduri

- În cazul când interogarea întoarce mai multe rânduri **nu mai este posibilă folosirea operatorilor de comparație.**  
În locul acestora **se folosește operatorul IN**, care așteaptă o listă de valori și nu doar una.
- Următoarea interogare selectează pentru fiecare grad didactic acele persoane care au salariul minim.  
Salariul minim pentru fiecare grad didactic este aflat printr-o subinterogare, care, evident, va întoarce mai multe rânduri:



## Subinterog. care returnează mai multe rânduri

```
SELECT nume, salariu, grad
FROM profesor
WHERE (salariu, grad) IN (SELECT MIN (salariu), grad
 FROM profesor
 GROUP BY grad)
ORDER BY salariu;
```

| NUME      | SALARIU | GRAD  |
|-----------|---------|-------|
| -----     | -----   | ----  |
| VOINEA    | 1200    | ASIST |
| STANESCU  | 1200    | ASIST |
| ALBU      | 2200    | LECT  |
| MARIN     | 2500    | PROF  |
| GEORGESCU | 2800    | CONF  |





## Subinterog. care returnează mai multe rânduri

- **Observație:** Spre deosebire de celelalte interogări de până acum, interogarea de mai înainte compară perechi de coloane.

În acest caz trebuie respectate următoarele reguli:

- coloanele din stânga condiției de căutare sunt în paranteze și fiecare coloană este separată prin virgulă;
- coloanele returnate de interogarea interioară trebuie să se potrivească ca număr și tip cu coloanele cu care sunt comparate în interogarea exterioară; în plus, ele trebuie să fie în aceeași ordine cu coloanele cu care sunt comparate.



## Subinterog. care returnează mai multe rânduri

- Alături de operatorul IN, o subinterogare care returnează mai multe rânduri poate folosi **operatorii ANY, ALL sau EXISTS**.  
Operatorii ANY și ALL sunt prezentați în continuare, iar operatorul EXISTS va fi prezentat în secțiunea “Subinterogări corelate”.
- Operatorii ANY și ALL sunt folosiți în mod obligatoriu în combinație cu operatorii relaționali =, !=, <, >, <=, >;
- Operatorii IN și EXISTS nu pot fi folosiți în combinație cu operatorii relaționali, dar pot fi utilizați cu operatorul NOT, pentru negarea expresiei.

# Operatorul ANY

- Operatorul ANY (sau sinonimul său **SOME**) este folosit pentru a compara o valoare cu oricare dintre valorile returnate de o subinterogare.
- Pentru a înțelege modul de folosire al acestui operator să considerăm următorul exemplu ce afișează cadrele didactice ce câștigă mai mult decât profesorii care au cel mai mic salariu:

```
SELECT nume, salariu, grad
FROM profesor
WHERE salariu > ANY (SELECT DISTINCT salariu
 FROM profesor
 WHERE grad = 'PROF');
```

| NUME      | SALARIU | GRAD |
|-----------|---------|------|
| -----     | -----   | ---- |
| GHEORGHIU | 3000    | PROF |
| GEORGESCU | 2800    | CONF |



## Operatorul ANY - cont.

- Interogarea anterioară este evaluată astfel: dacă salariul unui cadru didactic este mai mare decât cel puțin unul din salariile returnate de interogarea interioară, acea înregistrare este inclusă în rezultat.  
Cu alte cuvinte,  
>ANY înseamnă mai mare decât minimul dintre valorile returnate de interogarea interioară,  
<ANY înseamnă mai mic ca maximumul, iar  
=ANY este echivalent cu operatorul IN.
- **Observație:** opțiunea DISTINCT este folosită frecvent atunci când se folosește operatorul ANY pentru a preveni selectarea de mai multe ori a unor înregistrări.



# Operatorul ALL

- Operatorul ALL este folosit pentru a compara o valoare cu toate valorile returnate de o subinterogare.
- Considerăm următorul exemplu ce afișează cadrele didactice care câștigă mai mult decât asistenții cu salariul cel mai mare:

```
SELECT nume, salariu, grad
FROM profesor
WHERE salariu > ALL (SELECT DISTINCT salariu
 FROM profesor
 WHERE grad = 'ASIST');
```

| NUME      | SALARIU | GRAD  |
|-----------|---------|-------|
| -----     | -----   | ----- |
| GHEORGHIU | 3000    | PROF  |
| MARIN     | 2500    | PROF  |
| GEORGESCU | 2800    | CONF  |
| ALBU      | 2200    | LECT  |



## Operatorul ALL - cont.

- Interogarea de mai sus este evaluată astfel: dacă salariul unui cadru didactic este mai mare decât toate valorile returnate de interogarea interioară, acea înregistrare este inclusă în rezultat.  
Cu alte cuvinte,  
>ALL înseamnă mai mare ca maximum dintre valorile returnate de interogarea interioară, iar  
<ALL înseamnă mai mic ca minimum dintre acestea.
- **Observație:** operatorul ALL nu poate fi utilizat cu operatorul = deoarece interogarea nu va întoarce nici un rezultat cu excepția cazului în care toate valorile sunt egale, situație care nu ar avea sens.

## Operatorul ALL - cont.

- Subinterogările pot fi imbricate (utilizate cu alte subinterogări) până la 255 de niveluri, indiferent de nr. de valori returnate de fiecare subinterogare.
- Pentru a selecta cadrele didactice care au salariul mai mare decât cel mai mare salariu al cadrelor didactice care aparțin catedrei de Electronică, vom folosi următoarea interogare:

```
SELECT nume, prenume, salariu
FROM profesor
WHERE salariu > (SELECT MAX(salariu)
 FROM profesor
```

```
 WHERE cod_catedra = (SELECT cod_catedra
 FROM catedra
```

```
 WHERE nume='ELECTRONICĂ')) ;

NUME PRENUME SALARIU

GHEORGHIU STEFAN 3000
GEORGESCU CRISTIANA 2800
```



## Subinterogări corelate

- În exemplele considerate până acum interogarea interioară era evaluată prima, după care valoarea sau valorile rezultate erau utilizate de către interogarea exterioară. Subinterogările de acest tip sunt numite **subinterogări simple**.
- O altă formă de subinterogare o reprezintă **interogarea corelată**, caz în care interogarea exterioară transmite repetat câte o înregistrare pentru interogarea interioară.
- Interogarea interioară este evaluată de fiecare dată când este transmisă o înregistrare din interogarea exterioară, care se mai numește și înregistrare candidată.
- Subinterogarea corelată poate fi identificată prin faptul că interogarea interioară nu se poate executa independent ci depinde de valoarea transmisă de către interogarea exterioară.
- Dacă ambele interogări accesează aceeași tabelă, trebuie asigurate alias-uri pentru fiecare referire la tabela respectivă.





## Subinterogări corelate - cont.

- Subinterogările corelate reprezintă o cale de a accesa fiecare înregistrare din tabel și de a compara anumite valori ale acesteia cu valori ce depind tot de ea.
- **Evaluarea unei subinterogări corelate** se execută în următorii pași:
  1. Interogarea exterioară trimite o înregistrare candidată către interogarea interioară;
  2. Interogarea interioară se execută în funcție de valorile înregistrării candidate;
  3. Valorile rezultate din interogarea interioară sunt utilizate pentru a determina dacă înregistrarea candidată va fi sau nu inclusă în rezultat;
  4. Se repetă procedeul începând cu pasul 1 până când nu mai există înregistrări candidate.

## Subinterogări corelate - cont.

- De exemplu, pentru a regăsi cadrele didactice care câștigă mai mult decât salariul mediu din propria catedră, putem folosi următoarea interogare corelată:

```
SELECT nume, prenume, salariu
FROM profesor p
WHERE salariu > (SELECT AVG(salariu)
 FROM profesor
 WHERE cod_catedra = p.cod_catedra);
```

| NUME      | PRENUME  | SALARIU |
|-----------|----------|---------|
| -----     | -----    | -----   |
| GHEORGHIU | STEFAN   | 3000    |
| MARIN     | VLAD     | 2500    |
| ALBU      | GHEORGHE | 2200    |



## Subinterogări corelate - cont.

- În exemplul anterior coloana interogării exterioare care se folosește în interogarea interioară este `p.cod_catedra`.  
Deoarece `p.cod_catedra` poate avea valoare diferită pentru fiecare înregistrare, interogarea interioară se execută pentru fiecare înregistrare candidată transmisă de interogarea exterioară.



## Subinterogări corelate - cont.

- Atunci când folosim subinterogări corelate împreună cu clauza HAVING, coloanele utilizate în această clauză trebuie să se regăsească în clauza GROUP BY.  
În caz contrar, va fi generat un mesaj de eroare datorat faptului că nu se poate face comparație decât cu o expresie de grup.
- De exemplu, următoarea interogare este corectă, ea selectând gradele didactice pentru care media salariului este mai mare decât maximul primei pentru același grad:

```
SELECT grad
FROM profesor p
GROUP BY grad
HAVING AVG(salariu) > (SELECT MAX(prima)
 FROM profesor
 WHERE grad = p.grad);
```



# Operatorul EXISTS

- Operatorul EXISTS verifică dacă, pentru fiecare înregistrare transmisă de interogarea exterioară, există sau nu înregistrări care satisfac condiția interogării interioare, returnând interogării exterioare valoarea TRUE sau FALSE.
- Cu alte cuvinte, operatorul EXISTS cere în mod obligatoriu corelarea interogării interioare cu interogarea exterioară.
- Datorită faptului că operatorul EXISTS verifică doar existența rândurilor selectate și nu ia în considerare nr. sau valorile atributelor selectate, **în subinterogare** poate fi specificate orice număr sau oricâte attribute (deși acest lucru nu este recomandabil din punct de vedere al eficienței).
- În particular, **poate fi folosită o constantă și chiar simbolul '\*'**.  
De altfel EXISTS este singurul operator care permite acest lucru.

## Operatorul EXISTS - cont.

- Următoarea interogare selectează toate cadrele didactice care au **măcar** un subordonat:

```
SELECT cod, nume, prenume, grad
FROM profesor p
WHERE EXISTS (SELECT '1'
 FROM profesor
 WHERE profesor.sef = p.cod)
ORDER BY cod;
```

| COD  | NUME      | PRENUME   | GRAD  |
|------|-----------|-----------|-------|
| ---- | -----     | -----     | ----  |
| 100  | GHEORGHIU | STEFAN    | PROF  |
| 102  | GEORGESCU | CRISTIANA | CONF  |
| 103  | IONESCU   | VERONICA  | ASIST |



## Operatorul EXISTS - cont.

- La fel ca și operatorul IN, operatorul EXISTS poate fi negat, luând forma **NOT EXISTS**.  
Totuși, o remarcă foarte importantă este faptul că pentru subinterogări **NOT IN nu este la fel de eficient ca NOT EXISTS**.

Astfel dacă în lista de valori transmisă operatorului NOT IN există una sau mai multe valori NULL, atunci condiția va lua valoarea de adevăr FALSE, indiferent de celelalte valori din listă.

- De exemplu, următoarea interogare încearcă să returneze toate cadrele didactice care nu au nici un subaltern:

```
SELECT nume, grad
FROM profesor
WHERE cod NOT IN (SELECT sef FROM profesor);
```

## Operatorul EXISTS - cont.

- Interogarea anterioară nu va întoarce nici o înregistrare deoarece coloana sef conține și valoarea NULL.

Pentru a obține rezultatul corect trebuie să folosim următoarea interogare:

```
SELECT nume, grad
FROM profesor p
WHERE NOT EXISTS (SELECT sef
 FROM profesor
 WHERE sef = p.cod);
```

| NUME     | GRAD  |
|----------|-------|
| -----    | ----- |
| MARIN    | PROF  |
| ALBU     | LECT  |
| VOINEA   | ASIST |
| STANESCU | ASIST |





## Operatorul EXISTS - cont.

- În general, operatorul EXISTS se folosește în cazul subinterogărilor corelate și este câteodată cel mai eficient mod de a realiza anumite interogări.
- Performanța interogărilor depinde de folosirea indecșilor, de numărul rândurilor returnate, de dimensiunea tabelului și de necesitatea creării tabelului temporar pentru evaluarea rezultatelor intermediare.

Tabelele temporare generate de Oracle nu sunt indexate, iar acest lucru poate degrada performanța subinterogărilor dacă se folosesc operatorii IN, ANY sau ALL.



# Bibliografie

F. Ipate, M. Popescu, *Dezvoltarea aplicațiilor de baze de date în Oracle 8 și Oracle Forms 6*, Editura ALL, 2000.