SQL - 2

CERERI SELECT PE MAI MULTE TABELE

STUD

MATR	NUME	AN	GRUPA	DATAN	LOC	TUTOR	PUNCTAJ	CODS
1456	GEORGE	4	1141A	12-MAR-82	BUCURESTI		2890	11
1325	VASILE	2	1122A	05-OCT-84	PITESTI	1456	390	11
1645	MARIA	3	1131B	17-JUN-83	PLOIESTI		1400	11
3145	ION	1	2112B	24-JAN-85	PLOIESTI	3251	1670	21
2146	STANCA	4	2141A	15-MAY-82	BUCURESTI		620	21
3251	ALEX	5	2153B	07-NOV-81	BRASOV		1570	21
2215	ELENA	2	2122A	29-AUG-84	BUCURESTI	2146	890	21
4311	ADRIAN	3	2431A	31-JUL-83	BUCURESTI		450	24
3514	FLOREA	5	2452B	03-FEB-81	BRASOV		3230	24
1925	OANA	2	2421A	20-DEC-84	BUCURESTI	4311	760	24
2101	MARIUS	1	2412B	02-SEP-85	PITESTI	3514	310	24
4705	VOICU	2	2421B	19-APR-84	BRASOV	4311	1290	24

SPEC si BURSA

CODS	NUME	DOMENIU
11	MATEMATICA	STIINTE EXACTE
21	GEOGRAFIE	UMANIST
24	ISTORIE	UMANIST

TIP	PMIN	PMAX	SUMA
FARA BURSA	0	399	
BURSA SOCIALA	400	899	100
BURSA DE STUDIU	900	1799	150
BURSA DE MERIT	1800	2499	200
BURSA DE EXCEPTIE	2500	9999	300

OBIECTIV

- În foarte multe cazuri se doreşte ca un acelaşi rezultat să conţină date care sunt stocate în două sau mai multe tabele din baza de date, ca în exemplele următoare:
- Lista cu nume studenţi şi denumiri specializări. Tabelele care conţin aceste date sunt STUD (numele studentului) şi SPEC (denumirea specializării).
- Numele studenţilor (din tabela STUD) şi tipul burselor acestora (din tabela BURSA).
- Numele studentului, denumirea specializării şi cuantumul bursei. În acest caz sunt implicate toate cele trei tabele ale bazei de date de test.

JOIN

- Operaţia care permite astfel de regăsiri se numeşte ***join*** (termen preluat din limba engleză) şi este realizată prin intermediul unei cereri SELECT având următoarele caracteristici:
- În clauza FROM este specificată nu doar o singură tabelă ci o listă de tabele.
- În clauza WHERE există o condiţie care să coreleze liniile tabelelor din lista FROM (condiţie numită şi ***condiţie de join***).

SINTAXA

```
SELECT [DISTINCT] lista_de_expresii
  FROM lista_de_tabele
  WHERE conditie_de_join
              AND conditii_suplimentare
. . .
-- alte clauze: GROUP BY, HAVING, ORDER
  BY
```

OBSERVATII

- Atunci când condiţia de join lipseşte, fiecare linie a unei tabele din lista FROM este concatenată cu fiecare linie a celorlalte tabele, obţinându-se de fapt ***produsul cartezian*** al acestora.
- Dacă în condiţia de join apar numai egalităţi operaţia este numită şi ***echijoin***. În celelalte cazuri avem un ***non-echijoin***.
- În lista de tabele care participă la join o tabelă poate să apară repetat. O astfel de operaţie este numită şi ***joinul unei tabele cu ea însăşi*** sau ***self-join***.

OBSERVATII (2)

- În cazul în care o linie a unei tabele nu se corelează prin condiţia de join cu nici o linie din celelalte tabele ea nu va participa la formarea rezultatului. Se poate însă cere ca aceasta să fie luată în considerare pentru rezultat, rezultând aşa numitul ***join extern*** (în engleză ***outer join***).
- În versiunile anterioare sintaxa joinului în Oracle era diferită de standardul ANSI. Începând cu versiunea Oracle 9i au fost introduse în limbaj şi tipurile de join din standardul SQL:1999 (SQL-3) printre care ***cross-join, join natural*** şi mai multe ***variante de join extern.***

PRODUS CARTEZIAN

Cererea:

```
SELECT *
FROM STUD, SPEC;
```

va returna un rezultat având 12 coloane şi 36 de linii formate din concatenarea fiecărei linii din STUD cu fiecare linie din SPEC.

De asemenea, cererea:

```
SELECT DATAN, DOMENIU
FROM STUD, SPEC
WHERE LOC='BUCURESTI';
```

are un rezultat conţinând 15 linii

ECHIJOIN

```
SELECT MATR, STUD.NUME, STUD.CODS,
  SPEC.NUME
FROM STUD, SPEC
WHERE STUD.CODS = SPEC.CODS AND
DOMENIU='STIINTE EXACTE`
```

- Este marcata conditia de join. Restul conditiei este suplimentar

NON-ECHIJOIN

```
SELECT NUME, AN, TIP, SUMA
FROM STUD, BURSA
WHERE PUNCTAJ BETWEEN PMIN AND PMAX
      AND CODS=11
```

- Este marcata conditia de join

ALIAS DE TABELA

```
SELECT S.NUME, S.CODS, "SP
  STUD".NUME, TIP
FROM STUD S, SPEC "SP STUD", BURSA
WHERE S.CODS = "SP STUD".CODS AND
  S.PUNCTAJ BETWEEN PMIN AND PMAX
```

- Aliasul trebuie sa indeplineasca anumite conditii (Oracle):

ALIAS DE TABELA (2)

- Nu poate fi mai lung de 30 de caractere.
- În cazul în care nu începe cu o literă sau când conţine alte caractere dacât litere, cifre, _, # şi $ trebuie pus între ghilimele (de exemplu atunci când conţine spaţii).
- În cazul unui alias între ghilimele, dimensiunea maximă de 30 de caractere nu include ghilimelele.

ALIAS DE TABELA (3)

- Între ghilimele literele mici sunt considerate diferite de literele mari.
- Nu poate fi folosit decât în cererea curentă. Sistemul nu stochează în baza de date sau altundeva aceste nume alternative.
- În cazul în care o tabelă are asociat un alias prin clauza FROM acesta poate şi trebuie să fie folosit pentru prefixare în toate celelalte clauze ale cererii.

ALIAS DE TABELA (4)

- În cazul unui alias între ghilimele literele mici sunt considerate diferite de literele mari. Din această cauză dacă prima linie a cererii anterioare este:

```
SELECT S.NUME, S.CODS, "Sp Stud".NUME,
    TIP
```

- se obţine eroare

ALIAS DE TABELA (4)

- Dacă pentru o tabelă a fost definit un alias numele tabelei nu mai poate fi folosit pentru prefixare. În cazul cererii anterioare, prima linie nu mai poate fi rescrisă astfel:

```
SELECT S.NUME, STUD.CODS,"Sp Stud".NUME,
    TIP
```

JOIN – cont.

- În cazul în care condiţia de join nu este completă rezultatul va conţine un produs cartezian între joinurile existente şi restul tabelelor.
- Exemplu: în cererea următoare lipseşte condiţia de join care leagă tabela BURSA de celelalte tabele.

```
SELECT S.NUME, S.CODS, "SP STUD".NUME,
    TIP
FROM STUD S, SPEC "SP STUD", BURSA
WHERE S.CODS = "SP STUD".CODS;
```

- În acest caz rezultatul obţinut este produsul cartezian între BURSA şi joinul lui STUD cu SPEC şi va avea 60 de linii (5 linii din BURSA * 12 linii ale joinului STUD cu SPEC).

JOIN – cont.

1. În cazul general al unui join pe ***N*** tabele, condiţia de join este compusă din ***N - 1*** subcondiţii conectate prin AND care relaţionează întreg ansamblul de tabele. Altfel spus, dacă se construieşte un graf al condiţiei în care nodurile sunt tabele şi arcele subcondiţii de join care leagă două tabele atunci acest graf trebuie să fie conex.

JOINUL UNEI TABELE CU EA INSASI

- Este obligatorie folosirea aliasurilor:

```
SELECT S.NUME "NUME STUD",
       S.AN "AN STUD",
       T.NUME "NUME TUTOR",
       T.AN "AN TUTOR"
FROM STUD S, STUD T
WHERE S.TUTOR=T.MATR
```

ALT EXEMPLU

- Studenti care au acelasi tutor:

```
SELECT S1.NUME "STUDENT 1",
       S2.NUME "STUDENT 2",
       S2.TUTOR "TUTOR"
FROM STUD S1, STUD S2
WHERE S1.TUTOR=S2.TUTOR;
```

- INCORECT! Apar si cupluri cu acelasi student

ALT EXEMPLU - cont

- Alta varianta:

```
SELECT S1.NUME "STUDENT 1",
       S2.NUME "STUDENT 2",
       S2.TUTOR "TUTOR"
FROM STUD S1, STUD S2
WHERE S1.TUTOR=S2.TUTOR
      AND S1.MATR <> S2.MATR
```

- INCORECT! Apar ambele cupluri XY si YX

ALT EXEMPLU - cont

- Varianta corecta:

```
SELECT S1.NUME "STUDENT 1",
       S2.NUME "STUDENT 2",
       S2.TUTOR "TUTOR"
FROM STUD S1, STUD S2
WHERE S1.TUTOR=S2.TUTOR
      AND S1.MATR > S2.MATR
```

JOIN EXTERN (Oracle)

- Apar si studentii fara tutor:

```
SELECT S.NUME "NUME STUD", S.AN "AN
  STUD", T.NUME "NUME TUTOR",
  T.AN "AN TUTOR"
FROM STUD S, STUD T
WHERE S.CODS = 11 AND
      S.TUTOR=T.MATR(+)
```

JOIN EXTERN (Oracle)

- Rezultat:

NUME STUD	AN STUD	NUME TUTOR	AN TUTOR
VASILE	2	GEORGE	4
GEORGE	4		
MARIA	3		

JOIN EXTERN (Oracle)

- Daca schimbam plusul apar studentii care nu sunt tutori:

```
SELECT S.NUME "NUME STUD", S.AN "AN
  STUD", T.NUME "NUME TUTOR",
  T.AN "AN TUTOR"
FROM STUD S, STUD T
WHERE S.CODS = 11 AND
      S.TUTOR(+)=T.MATR
```

JOIN EXTERN FARA =

- Joinul extern se poate folosi şi în cazul în care condiţia nu este de egalitate. Se pot folosi operatorii <, <=, >, >= sau <>. Un exemplu în acest sens este următorul: cererea

```
SELECT S1.NUME "STUDENT 1",
  S1.PUNCTAJ "PUNCTAJ 1",
   S2.NUME "STUDENT 2",
   S2.PUNCTAJ "PUNCTAJ 2"
FROM STUD S1, STUD S2
WHERE S1.PUNCTAJ > S2.PUNCTAJ;
```

JOIN EXTERN FARA = (cont.)

- Asa apar toate perechile (66) plus inca o linie cu studentul avand cel mai mare punctaj:

```
SELECT S1.NUME "STUDENT 1",
  S1.PUNCTAJ "PUNCTAJ 1",
   S2.NUME "STUDENT 2",
   S2.PUNCTAJ "PUNCTAJ 2"
FROM STUD S1, STUD S2
WHERE S1.PUNCTAJ (+) > S2.PUNCTAJ;
```

OBSERVATIE

- Marcajul de join extern se poate folosi şi atunci când condiţia de join este compusă, cu excepţia cazului în care se foloseşte sau logic (OR) sau operatorul de incluziune IN urmat de o listă care conţine mai mult de o valoare.
- Joinul extern se poate folosi şi în conjuncţie cu operatorii specifici SQL

JOIN EXTERN CU BETWEEN

- cereri valide:

```
SELECT NUME, AN, TIP, SUMA
FROM STUD, BURSA
WHERE PUNCTAJ(+)-1000 BETWEEN PMIN AND
  PMAX
```

Rezultatul conţine şase linii complete pentru studenţii cu un (PUNCTAJ-1000) care se încadrează pentru un tip de bursă din tabela BURSA şi două linii pentru cele două tipuri de bursă care nu au nici un student asociat prin condiţia de join. Aceste 2 linii conţin valori nule pe primele două coloane.

JOIN EXTERN CU BETWEEN (2)

- cereri valide:

```
SELECT NUME, AN, TIP, SUMA
FROM STUD, BURSA
WHERE PUNCTAJ-1000 BETWEEN PMIN(+) AND
   PMAX(+)
```

- Rezultat: 12 linii conţinând datele studenţilor şi tipul bursei corespunzătoare unui punctaj egal cu (PUNCTAJ-1000) sau valori nule dacă punctajul respectiv nu se încadrează în nici un tip de bursă.
- **Observaţie:** în acest caz marcajul (+) trebuie pus atât la valoarea minimă cât şi la valoarea maximă. În cazul în care lipseşte de la una dintre ele nu se semnalează eroare dar se calculează joinul normal rezultând 6 linii şi nu 12.

JOIN EXTERN CU LIKE

- Cerere valida:

```
SELECT S.NUME, S.CODS, F.NUME, F.CODS,
  SUBSTR(F.CODS, 2, 1)||'%' SABLON
FROM STUD S, SPEC F
WHERE S.CODS(+) LIKE SUBSTR(F.CODS, 2,
  1)||'%';
```

- Funcţia SQL SUBSTR întoarce un subşir al primului argument - în cazul nostru subşirul care începe în poziţia 2 şi are lungime egală cu 1.

REZULTAT

NUME	CODS	NUME	CODS	SABLON
GEORGE	11	MATEMATICA	11	1%
VASILE	11	MATEMATICA	11	1%
MARIA	11	MATEMATICA	11	1%
GEORGE	11	GEOGRAFIE	21	1%
VASILE	11	GEOGRAFIE	21	1%
MARIA	11	GEOGRAFIE	21	1%
		ISTORIE	24	4%

JOIN EXTERN CU LIKE – cont.

- Mutând marcajul obţinem cererea:

```
SELECT S.NUME, S.CODS, F.NUME, F.CODS,
       SUBSTR(F.CODS, 2, 1)||'%' SABLON
FROM STUD S, SPEC F
WHERE S.CODS LIKE SUBSTR(F.CODS(+), 2,
  1)||'%'
```

- Returnează 15 linii: 6 linii pentru cei 3 studenţi pentru care condiţia de join obişnuit este îndeplinită plus câte o linie pentru fiecare dintre ceilalţi 9 studenţi care nu verifică această condiţie, având valori nule pe coloanele 3 şi 4.

JOINURI SQL - 3

Exista clauzele:

- CROSS JOIN – produs cartezian
- JOIN … USING – coloane comune
- NATURAL JOIN – join natural
- JOIN … ON – join general
- OUTER JOIN … ON – join extern

Pe o clauza avem o singura tabela (nu se mai pun toate in FROM)

CROSS JOIN

- Implementeaza produsul cartezian:

```
SELECT [DISTINCT] lista_de_expresii
FROM tabela1
CROSS JOIN tabela2;
```

- Exemplu:

```
SELECT S.NUME, F.NUME
FROM STUD S
CROSS JOIN SPEC F; -- 36 de linii
```

CROSS JOIN – cont.

- Pentru a face join cu CROSS JOIN adaugam conditia de join in WHERE:

```
SELECT S.NUME, DATAN, F.NUME,
  DOMENIU
FROM STUD S
CROSS JOIN SPEC F
WHERE S.CODS=F.CODS;
```

JOIN ... USING

- E un echijoin dupa coloane cu acelasi nume specificate in USING (deci nu toate):

```
SELECT [DISTINCT] lista_de_expresii
  FROM tabela1
  JOIN tabela2 USING (nume_coloane)
```

EXEMPLU

- deoarece în STUD si SPEC avem coloane cu acelasi nume (NUME si CODS) putem face echjoinul dupa CODS astfel:

```
SELECT S.NUME, DATAN, F.NUME, DOMENIU
FROM STUD S
JOIN SPEC F USING (CODS);
```

- După cum se observă în lista USING numele coloanelor după care se efectuează joinul nu trebuie prefixat cu numele sau aliasul vreuneia dintre tabele.

EXEMPLU – cont.

- Dacă se doreşte afişarea aceloraşi date dar doar pentru studenţii născuţi în BUCUREŞTI se va adăuga condiţia suplimentară pe WHERE:

```
SELECT S.NUME, DATAN, F.NUME, DOMENIU
FROM STUD S
JOIN SPEC F USING (CODS)
WHERE LOC='BUCURESTI';
```

NATURAL JOIN

- E un echijoin dupa toate coloane cu acelasi nume:

```
SELECT [DISTINCT] lista_de_expresii
FROM tabela1
NATURAL JOIN tabela2
```

- Exemplu – fara linii rezultat, nici un student nu are acelasi nume cu o specializare:

```
SELECT NUME, DATAN, DOMENIU
FROM STUD
NATURAL JOIN SPEC; -- echijoin dupa NUME si CODS
```

JOIN .. ON

- Prin această clauză se implementează un join general. Condiţia de join (şi eventual şi condiţiile suplimentare) se pun la ON. Sintaxa este:

```
SELECT [DISTINCT] lista_de_expresii
  FROM tabela1
  JOIN tabela2 ON (expresie_logica)
```

EXEMPLU

- Exemplu: cererea următoare efectuează joinul dintre STUD şi SPEC după coloana CODS şi conţine şi o linie suplimentară care opreşte doar liniile corespunzătoare studenţilor născuţi în PLOIEŞTI.
- Această condiţie suplimentară se putea pune şi pe clauza WHERE.

```
SELECT S.NUME, DATAN, F.NUME, DOMENIU
FROM STUD S
JOIN SPEC F ON (S.CODS=F.CODS AND
  LOC='PLOIESTI');
```

EXEMPLU – cont.

- Rezultatul este:

```
NUME       DATAN     NUME         DOMENIU
---------- --------- ---------- ---------------
MARIA       17-JUN-83 MATEMATICA STIINTE EXACTE
ION         24-JAN-85 GEOGRAFIE  UMANIST
```

JOIN .. ON – cont.

- Cu JOIN ON se pot calcula şi non-echijoinuri. Dacă se doreşte o listă cu date despre studenţi şi bursele acestora, următoarea cerere va întoarce rezultate corecte:

```
SELECT NUME, PUNCTAJ, TIP, SUMA
FROM STUD
JOIN BURSA ON (PUNCTAJ BETWEEN PMIN AND
  PMAX)
```

OUTER JOIN ... ON

- În cazul joinului extern sintaxa este următoarea:

```
SELECT [DISTINCT] lista_de_expresii
FROM tabela1
LEFT    \
RIGHT     | OUTER JOIN tabela2
FULL    /
ON (tabela1.nume_coloana1 =
     tabela2.numecoloana2)
```

LEFT ...

- În cazul LEFT OUTER JOIN valorile nule provin din tabela2. De exemplu cererea:

```
SELECT S.NUME "NUME STUD", S.AN "AN STUD",
       T.NUME "NUME TUTOR", T.AN "AN TUTOR"
FROM STUD S
LEFT OUTER JOIN STUD T
ON (S.TUTOR=T.MATR);
```

returnează 12 linii şi este echivalentă cu:

```
SELECT S.NUME "NUME STUD", S.AN "AN STUD",
       T.NUME "NUME TUTOR", T.AN "AN TUTOR"
FROM STUD S, STUD T
WHERE S.TUTOR=T.MATR(+) -- conditia de join extern;
```

RIGHT ...

- În cazul RIGHT OUTER JOIN valorile nule provin din tabela1. Cererea:

```
SELECT S.NUME "NUME STUD", S.AN "AN STUD",
       T.NUME "NUME TUTOR", T.AN "AN TUTOR"
FROM STUD S
RIGHT OUTER JOIN STUD T
ON (S.TUTOR=T.MATR);
```

returnează 13 linii şi este echivalentă cu:

```
SELECT S.NUME "NUME STUD", S.AN "AN STUD",
       T.NUME "NUME TUTOR", T.AN "AN TUTOR"
FROM STUD S, STUD T
WHERE S.TUTOR(+)=T.MATR;
```

FULL ...

- În cazul RIGHT OUTER JOIN valorile nule provin din tabela1. Cererea:

```
SELECT S.NUME "NUME STUD", S.AN "AN STUD",
       T.NUME "NUME TUTOR", T.AN "AN TUTOR"
FROM STUD S
RIGHT OUTER JOIN STUD T
ON (S.TUTOR=T.MATR);
```

returnează 13 linii şi este echivalentă cu:

```
SELECT S.NUME "NUME STUD", S.AN "AN STUD",
       T.NUME "NUME TUTOR", T.AN "AN TUTOR"
FROM STUD S, STUD T
WHERE S.TUTOR(+)=T.MATR;
```

OBSERVATIE

- In Oracle o condiţie de tipul:

```
WHERE  S.TUTOR(+)=T.MATR(+)
```

- va produce eroarea: *ORA-01468: a predicate may reference only one outer-joined table.*

Sfarsitul capitolului
CERERI SELECT PE MAI MULTE TABELE