

Capitolul 4

SUBCERERI

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

- ◆ O subcerere este o cerere SELECT inclusă într-o altă cerere SQL.
- ◆ Astfel de construcții se folosesc în cazul în care rezultatul dorit nu se poate obține cu o singură parcurgere a datelor.
- ◆ Exemplu: pentru a afla cine este studentul cu cel mai mare punctaj sunt necesare două parcurgeri ale tabelului STUD:
 - ◆ prima calculează punctajul maxim iar ulterior
 - ◆ a doua afișează datele dorite despre studentul sau studenții cu acel punctaj.

PARCURGERI

- ◆ Cum o cerere SELECT specifică o singură parcurgere a datelor rezultă că pentru rezolvarea problemei sunt necesare două astfel de cereri, prima (subcererea) furnizând datele necesare pentru a doua (cererea principală).

SUBCERERILE APAR:

- ◆ în expresiile logice din clauzele WHERE și HAVING
- ◆ în clauza ORDER BY a unei cereri SELECT; valoarea returnată de subcerere pentru fiecare linie a rezultatului va determina ordinea de afișare a acestora.
- ◆ în clauza SELECT; valoarea returnată de subcerere va fi prezentă în rezultatul final.
- ◆ în clauza FROM; în acest caz ele sunt asimilate unor tabele temporare din care se calculează rezultatul cererii care le include.

SUBCERERI IN EXPRESII LOGICE

- ◆ Rezultatul unui SELECT, dacă este nevid, este întotdeauna o tabelă. Din punct de vedere al modului de folosire al unei subcereri există însă diferențe în funcție de forma rezultatului acesteia:
 1. Subcereri care întorc o singură valoare
 2. Subcereri care întorc o coloană
 3. Subcereri care întorc o tabelă.
- ◆ În continuare este prezentat modul de utilizare pentru fiecare tip în parte

O VALOARE

- ◆ În acest caz valoarea întoarsă de subcerere poate fi folosită ca oricare alta în comparații care includ operatorii obișnuiți: <, <=, >, >=, = și <>.
- ◆ Exemplu: studentul cu cel mai mare punctaj: subcererea întoarce valoarea maximă a punctajului din tabela STUD iar cererea care o include numele studentului sau studenților care au acel punctaj și valoarea acestuia:

```
SELECT NUME, PUNCTAJ
FROM STUD
WHERE PUNCTAJ =
      (SELECT MAX (PUNCTAJ) FROM STUD)
```


REZULTAT

◆ Rezultatul obținut este:

NUME	PUNCTAJ
-----	-----
FLOREA	3230

REGULI PENTRU SUBCERERI (1)

- ◆ Subcererea trebuie să fie întotdeauna în partea dreaptă a comparației, ca în exemplul de mai sus.
- ◆ Subcererea se pune obligatoriu între paranteze.
- ◆ Deoarece rezultatul este folosit pentru calculele cererii principale, ordinea valorilor returnate nu este importantă. De aceea subcererile nu pot conține clauza ORDER BY.

REGULI PENTRU SUBCERERI (2)

- ◆ Dacă subcererea întoarce mai multe linii se semnaleză eroarea *ORA-01427: single-row subquery returns more than one row*.
- ◆ Dacă subcererea nu întoarce nici o linie, comparația în care e implicată se evaluează la FALS.
- ◆ O cerere poate conține una sau mai multe subcereri, acestea putând fi pe același nivel sau incluse una în alta.

ALTI OPERATORI

- ◆ În afară de operatorii uzuali de comparație, în cazul acestui tip de subcereri se pot folosi și operatorii BETWEEN, LIKE și IS NULL.
- ◆ Exemplele următoare reprezintă cereri valide conținând și BETWEEN sau LIKE. Folosirea lui IS NULL pentru o subcerere este relevantă doar în cazul subcererilor corelate prezentate într-un alt subcapitol.

SUBCERERI IN BETWEEN

- ◆ Afișarea numelui și punctajului pentru studenții având un punctaj egal cu cel mediu +/- 30%. Se folosește operatorul BETWEEN având ca parametri două subcereri:

```
SELECT NUME, PUNCTAJ
FROM STUD
WHERE PUNCTAJ BETWEEN
    (SELECT AVG (PUNCTAJ) FROM STUD) *0.7 AND
    (SELECT AVG (PUNCTAJ) FROM STUD) *1.3;
```

SUBCERERI IN LIKE

- ◆ Afișarea aceluiași date ca mai sus pentru studenții având un nume care nu începe cu aceeași literă cu a studentului cu punctaj maxim:

```
SELECT NUME, PUNCTAJ
FROM STUD
WHERE NUME NOT LIKE SUBSTR(
  (SELECT NUME FROM STUD
   WHERE PUNCTAJ = (SELECT MAX(PUNCTAJ)
                     FROM STUD)
  ) , 1, 1) || '%';
```

SUBCERERI IN LIKE (2)

- ◆ Există două niveluri de imbricare:
- ◆ Subcererea de nivel 2 întoarce valoarea maximă a punctajului.
- ◆ Subcererea de nivel 1 întoarce numele studentului cu acel punctaj.
- ◆ În cererea principală este decupată prima literă a acestui nume, folosind funcția SUBSTR(rezultat, 1, 1) și este concatenată cu caracterul % pentru a forma un șablon folosit apoi de condiția NOT LIKE.

ERORI (1)

- ◆ Subcererea î ntoarce mai multe valori.
În acest caz nu vom obține un rezultat ci mesajul de eroare menționat anterior:

```
SELECT NUME, PUNCTAJ  
FROM STUD  
WHERE PUNCTAJ = (SELECT PUNCTAJ FROM  
STUD) ;
```


ERORI (2)

- ◆ Subcererea nu întoarce nici o valoare.
În acest caz vom obține un rezultat vid (fără nici o linie), condiția evaluându-se la FALS:

```
SELECT NUME, PUNCTAJ
FROM STUD
WHERE PUNCTAJ =
      (SELECT MAX(PUNCTAJ) FROM STUD WHERE
      CODS = 100)
```

O COLOANA

- ◆ În acest caz valorile coloanei întoarse de subcerere sunt asimilate unei mulțimi.
- ◆ Condiția trebuie să folosească operatorul IN sau negatul acestuia NOT IN și nu operatori de comparație.

EXEMPLU

- ◆ Cererea din exemplul următor afișează lista tuturor studenților de la specializarea cu codul 21 care sunt tutori ai altor studenți. Pentru a fi tutor, matricola studentului trebuie să aparțină mulțimii valorilor aflate pe coloana TUTOR din tabela STUD calculată cu ajutorul subcererii:

```
SELECT NUME, CODS
FROM STUD
WHERE MATR IN (SELECT TUTOR FROM STUD)
AND CODS = 21;
```

REZULTAT

- ◆ Rezultatul va conține datele pentru doi studenți:

NUME	CODS
STANCA	21
ALEX	21

- ◆ Observație: NOT IN returnează întotdeauna valoarea fals în cazul în care mulțimea conține valori nule.

NOT IN

- ◆ Din această cauză pentru a obține lista studenților de la această specializare care nu sunt tutori nu se poate folosi cererea:

```
SELECT NUME, CODS  
FROM STUD  
WHERE MATR NOT IN (SELECT TUTOR FROM  
STUD) AND CODS = 21;
```

- ◆ deoarece subcererea întoarce o coloană care conține și valori nule.

NOT IN – cont.

- ◆ În astfel de cazuri este necesară eliminarea acestor valori din rezultat prin adăugarea unei condiții suplimentare de tip IS NOT NULL:

```
SELECT NUME, CODS  
FROM STUD  
WHERE MATR NOT IN  
(SELECT TUTOR FROM STUD  
WHERE TUTOR IS NOT NULL)  
AND CODS = 21;
```

SUBCERERI CU GROUP BY

- ◆ În exemplul următor subcererea folosește o clauză GROUP BY pentru a genera punctajele maxime pentru fiecare specializare. Cererea principală afișează studenții care au un punctaj egal cu vreuna dintre valorile returnate:

```
SELECT NUME, PUNCTAJ, CODS  
FROM STUD  
WHERE PUNCTAJ IN  
  (SELECT MAX(PUNCTAJ)  
   FROM STUD  
   GROUP BY CODS) ;
```

SUBCERERI CU GROUP BY (2)

```
SELECT NUME, PUNCTAJ, CODS  
FROM STUD  
WHERE PUNCTAJ IN  
  (SELECT MAX(PUNCTAJ)  
   FROM STUD  
   GROUP BY CODS);
```

- ◆ Întâmplător, rezultatul conține chiar studenții cu cel mai mare punctaj pentru fiecare specializare.
- ◆ În cazul general însă rezultatul poate conține și studenți care nu au punctajul maxim la specializarea lor dar egal cu maximul unei alte specializări.

SOME/ANY SI ALL

- ◆ Este posibilă folosirea operatorilor de comparație uzuali (<, >, =, etc.) în conjuncție cu o cerere care întoarce o coloană dacă aceasta este prefixată cu unul din operatorii SOME, ANY și ALL.
- ◆ Semnificația lor este următoarea:
 - SOME și ANY: condiția este adevărată dacă măcar o valoare dintre cele returnate de subcerere verifică comparația respectivă.
 - ALL: condiția este adevărată dacă toate valorile returnate de subcerere verifică comparația respectivă.

EXAMPLE (1)

- ◆ Lista studenților care au un punctaj mai mare decât al vreunui student de la specializarea cu cod 11:

```
SELECT NUME, PUNCTAJ
FROM STUD
WHERE PUNCTAJ >
      SOME (SELECT PUNCTAJ FROM STUD
            WHERE CODS = 11);
```

- ◆ Înlocuirea lui SOME cu ANY duce la obținerea aceluiași rezultat, cei doi operatori efectuând aceeași operație

EXAMPLE (2)

- ◆ Lista studenților care au un punctaj mai mare decât al tuturor studenților de la specializarea 11:

```
SELECT NUME, PUNCTAJ  
FROM STUD  
WHERE PUNCTAJ >  
      ALL(SELECT PUNCTAJ FROM STUD  
          WHERE CODS = 11);
```

O TABELA

- ◆ În cazul în care subcererea întoarce un rezultat care are mai multe coloane acesta este asimilat cu o mulțime de linii și se poate folosi operatorul IN în următorul mod:

```
WHERE (lista_de_expresii) IN (subcerere)
```

REGULI

1. Lista de expresii trebuie încadrată de paranteze rotunde.
2. Numărul de coloane din rezultatul subcererii trebuie să fie egal cu numărul de expresii din listă.
3. Corespondența între valorile expresiilor din listă și coloanele rezultatului este pozițională.

REGULI - cont

4. Tipurile elementelor corespondente trebuie să fie aceleași sau convertibile automat unul la celălalt (sistemul Oracle face conversia automată între tipurile șir de caractere și numere/date calendaristice).
5. Condiția este adevărată dacă rezultatul subcererii conține măcar o linie formată din valorile expresiilor din listă.
6. Dacă rezultatul subcererii este vid întreaga condiție este evaluată la *fals*.

EXEMPLU

- ◆ Pentru a afla care sunt studenții cu cel mai mare punctaj de la fiecare specializare, cererea este următoarea:

```
SELECT NUME, PUNCTAJ, CODS
FROM STUD
WHERE (CODS, PUNCTAJ) IN
      (SELECT CODS, MAX(PUNCTAJ)
       FROM STUD
       GROUP BY CODS);
```

- ◆ Codiția va fi adevărată dacă punctajul studentului este egal cu un punctaj maxim întors de subcerere și în același timp codul specializării este al celei pentru care s-a calculat maximul respectiv.

OBSERVATIE

- ◆ Observație: Din cauza faptului că două valori nule nu sunt egale între ele un cuplu de tipul (NULL, valoare) nu va fi considerat egal cu el însuși.
- ◆ Din acest motiv, cererea următoare nu va returna date despre studenții care nu au un tutor asociat (au o valoare nulă pe această coloană) deși în aparență condiția ar trebui să fie adevărată pentru orice student al specializării având codul 11:

EXEMPLU

```
SELECT MATR, NUME, CODS, TUTOR
FROM STUD
WHERE (MATR, NUME, CODS, TUTOR) IN
      (SELECT MATR, NUME, CODS, TUTOR
       FROM STUD
       WHERE CODS = 11);
```

REZULTATE

MATR	NUME	CODS	TUTOR
1325	VASILE	11	1456

◆ deși rezultatul subcererii (executată separat) este următorul:

MATR	NUME	CODS	TUTOR
1456	GEORGE	11	
1325	VASILE	11	1456
1645	MARIA	11	

SUBCERERI PE HAVING

- ◆ Expresiile logice conținând subcereri se pot folosi și pe clauza HAVING în același mod ca mai sus.
- ◆ În acest caz însă condițiile vor conține doar elementele care pot să apară într-o astfel de clauză: constante, expresiile după care se face gruparea și funcții statistice.
- ◆ În continuare sunt prezentate câteva exemple:

EXAMPLE (1)

- ◆ Afișarea codului numeric și a punctajulelor minim și maxim doar pentru specializările care au un punctaj MEDIU peste media calculată la nivelul întregii tabele STUD:

```
SELECT CODS, MIN(PUNCTAJ), MAX(PUNCTAJ)
FROM STUD
GROUP BY CODS
HAVING AVG(PUNCTAJ) >
(SELECT AVG(PUNCTAJ) FROM STUD);
```

EXAMPLE (2)

- ◆ Afișarea codului și a punctajului mediu pentru specializarea cu cel mai mare punctaj mediu (functii statistice imbricate in subcerere):

```
SELECT CODS, AVG(PUNCTAJ)
FROM STUD
GROUP BY CODS
HAVING AVG(PUNCTAJ) =
(SELECT MAX(AVG(PUNCTAJ))
FROM STUD
GROUP BY CODS);
```

EXAMPLE (3)

- ◆ Afișarea codului și a punctajului maxim pentru toate specializările în afara celei/celor cu cel mai mic punctaj maxim. Pentru ca o specializare să apară în rezultat punctajul său maxim trebuie să fie mai mare decât al vreunei alte specializări. Subcererea întoarce lista punctajelor maxime iar comparația dorită se face folosind operatorul ANY:

```
SELECT CODS, MAX(PUNCTAJ)
FROM STUD
GROUP BY CODS
HAVING AVG(PUNCTAJ) >
    ANY (SELECT AVG(PUNCTAJ)
        FROM STUD
        GROUP BY CODS);
```

SUBCERERI PE FROM

- ◆ În cazul în care o subcerere apare pe clauza FROM ea va fi tratată ca o tabelă temporară. Cererea următoare afișează numele studenților bursieri de la specializarea cu codul 11.

```
SELECT NUME, SUMA
FROM (SELECT *
      FROM STUD, BURSA
      WHERE PUNCTAJ BETWEEN PMIN AND PMAX
      AND CODS = 11
      AND SUMA IS NOT NULL);
```

JOIN DE SUBCERERI

- ◆ În cazul în care subcererea este implicată într-un join se pot folosi aliasuri de tabelă pentru a dezambigua numele coloanelor rezultatului său. Cererea anterioară se poate rescrie și astfel:

```
SELECT NUME, SUMA
FROM (SELECT * FROM STUD WHERE CODS = 11) S11,
      (SELECT * FROM BURSA WHERE SUMA IS NOT NULL) B
WHERE S11.PUNCTAJ BETWEEN B.PMIN AND B.PMAX;
```


JOIN CU SUBCERERI

- ◆ Folosirea aliasurilor de tabelă este obligatorie în cazul în care tabelele implicate în join au coloane cu același nume. În exemplul se afișează numele studenților și numele specializării pentru specializarea cu codul 11:

```
SELECT ST.NUME, SP.NUME  
FROM STUD ST,  
(SELECT * FROM SPEC WHERE CODS = 11) SP  
WHERE ST.CODS = SP.CODS;
```

REZULTAT VID

- ◆ În cazul în care o subcerere aflată pe clauza FROM nu întoarce nici o linie nu se semnalează o eroare dar cererea care o include va returna un rezultat vid, inclusiv în cazul unui produs cartezian.
- ◆ Subcererea din exemplul următor nu returnează linii deoarece specializarea cu codul 100 nu există:

```
SELECT ST.NUME, SP.NUME  
FROM STUD ST,  
(SELECT * FROM SPEC WHERE CODS = 100) SP  
WHERE ST.CODS = SP.CODS;
```

REZULTAT VID (2)

- ◆ O astfel de subcerere poate participa însă la un join extern:

```
SELECT ST.NUME, SP.NUME
FROM STUD ST, (SELECT * FROM SPEC WHERE
               CODS = 100) SP
WHERE ST.CODS = SP.CODS (+);
```

- ◆ sau:

```
SELECT ST.NUME, SP.NUME
FROM STUD ST
FULL OUTER JOIN (SELECT * FROM SPEC
                 WHERE CODS = 100) SP
ON (ST.CODS = SP.CODS);
```

SUBCERERI CORELATE

- ◆ În exemplele anterioare subcererea se execută o singură dată după care rezultatul său este folosit pentru evaluarea care o include.
- ◆ Există însă posibilitatea ca rezultatul subcererii să fie dependent de valorile de pe linia curentă a parcurgerii definite de cerere.
- ◆ În acest caz se spune că avem o *subcerere corelată*.

SUBCERERI CORELATE (2)

- ◆ Descrierea modului de evaluare în astfel de situații va fi făcută pe baza următorului exemplu care afișează date despre studenții care au un punctaj peste media celor din specializarea lor.
- ◆ Deoarece și cererea și subcererea lucrează pe aceeași tabelă, se folosește aliasul de tabelă S pentru a specifica parcurgerea principală (cea din cerere).

```
SELECT NUME, CODS, PUNCTAJ  
FROM STUD S  
WHERE PUNCTAJ > (SELECT AVG(PUNCTAJ) FROM  
STUD WHERE CODS = S.CODS);
```

```
SELECT NUME, CODS, PUNCTAJ -- CALCUL REZULTAT
FROM STUD S
WHERE PUNCTAJ > (SELECT AVG(PUNCTAJ) FROM STUD WHERE CODS = S.CODS);
```

- ◆ Pentru fiecare linie a tabelii STUD valoarea de pe coloana CODS - codul specializării - va fi transmisă subcererii (S.CODS).
- ◆ Se execută subcererea care calculează punctajul mediu pentru studenții de la specializarea respectivă (AVG și condiția CODS = S.CODS).
- ◆ Rezultatul subcererii este folosit în cerere pentru a filtra sau nu linia curentă (condiția PUNCTAJ > rezultat subcerere).
- ◆ Teoretic subcererea nu se mai execută o singură dată ci pentru fiecare linie din parcurgerea specificată de cererea în care este inclusă.

EXEMPLU

- ◆ Numele, codul și numărul total de studenți pentru specializările la care este înmatriculat cel puțin un student de anul 1. :

```
SELECT SP.NUME, ST.CODS, COUNT(*)  
FROM STUD ST, SPEC SP  
WHERE ST.CODS = SP.CODS  
GROUP BY SP.NUME, ST.CODS  
HAVING 1 <= (SELECT COUNT(*)  
              FROM STUD  
              WHERE CODS = ST.CODS AND  
              AN = 1);
```

OBSERVATII

- ◆ În cazul în care subcererea corelată se află pe clauza WHERE ea poate primi valoarea de pe orice coloană a tabelului/produsului cartezian din cererea înconjurătoare.
- ◆ În cazul în care subcererea corelată este pe clauza HAVING poate primi doar coloanele/expresiile după care s-a făcut gruparea.
- ◆ Subcererile corelate pot returna o valoare, o coloană sau o tabelă, ca și cele necorelate. În cazul în care returnează o coloană sau o tabelă se poate folosi operatorul IN. Sunt permisi de asemenea operatorii SOME/ANY și ALL.

OPERATORUL EXISTS

- ◆ Acest operator testează dacă subcererea primită ca argument întoarce un rezultat nevid. Sintaxa sa este:

EXISTS (subcerere)

- ◆ Returnează valoarea logică ADEVARAT dacă subcererea are un rezultat nevid și FALS dacă rezultatul e vid (nici o linie).
- ◆ În cazul acestui operator nu este important conținutul rezultatului subcererii ci doar existența sau absența sa.

EXEMPLU

- ◆ Lista specializărilor unde există cel puțin un student de anul 1:

```
SELECT CODS, NUME
FROM SPEC
WHERE EXISTS (SELECT 1
              FROM STUD
              WHERE SPEC.CODS = CODS
              AND AN = 1);
```

SUBCERERI IN ORDER BY

- ◆ În actualele versiuni ale sistemului Oracle clauza ORDER BY poate conține o subcerere Aceasta trebuie să returneze o singură valoare și să fie o subcerere corelată.
- ◆ Prezența unei cereri necorelate pe această clauză nu duce la ordonarea rezultatului deoarece rezultatul fiind o constantă are aceeași valoare pentru fiecare dintre liniile supuse sortării.

EXEMPLU

- ◆ Ordonarea se face după numărul de studenți îndrumați de fiecare tutor:

```
SELECT NUME, CODS, AN
FROM STUD S
WHERE CODS = 24
ORDER BY (SELECT COUNT(*) FROM STUD
WHERE TUTOR = S.MATR) DESC;
```

SUBCERERI PE SELECT

- ◆ Ca și în cazul celor din ORDER BY aceste subcereri trebuie să întoarcă o singură valoare.
- ◆ Subcererea poate fi necorelată (în acest caz pe coloana respectivă din rezultat vom avea aceeași valoare) sau corelată.

EXEMPLU

```
SELECT NUME, (SELECT COUNT(*)  
              FROM STUD  
              WHERE TUTOR = S.MATR) NUMAR  
FROM STUD S  
WHERE CODS = 24  
      AND 1 <= (SELECT COUNT(*)  
                FROM STUD  
                WHERE TUTOR = S.MATR)  
ORDER BY (SELECT COUNT(*)  
          FROM STUD  
          WHERE TUTOR = S.MATR) DESC
```

UNION INTERSET si MINUS

◆ Sintaxa:

Subcerere

UNION | INTERSECT | MINUS

Subcerere

REGULI

- ◆ Ambele cereri întorc același număr de coloane.
- ◆ Coloanele corespondente au valori de același tip sau de tipuri pentru care sistemul poate face automat conversia.
- ◆ Operatorii pot fi folosiți repetat și succesiv pentru a forma expresii. În acest caz, dacă nu se folosesc paranteze pentru a schimba ordinea de evaluare, ei se execută în ordinea în care apar în expresie.
- ◆ Capul de tabel al rezultatului este cel dat de prima cerere din expresie.
- ◆ Cererile implicate în aceste operații nu pot conține clauza ORDER BY. Aceasta poate să fie pusă doar la sfârșitul expresiei și poate conține nume de coloane din rezultat sau numerele de ordine ale acestora.
- ◆ Rezultatul nu conține linii duplicate. Acestea sunt eliminate chiar dacă provin din aceeași cerere - operand al expresiei.

EXEMPLU

```
SELECT NUME, CODS, LOC, PUNCTAJ  
FROM STUD  
WHERE CODS = 11 AND PUNCTAJ > 2000
```

UNION

```
SELECT NUME, CODS, LOC, PUNCTAJ  
FROM STUD  
WHERE CODS = 21 AND LOC = 'BUCURESTI'
```

UNION

```
SELECT NUME, CODS, LOC, PUNCTAJ  
FROM STUD  
WHERE CODS = 24 AND PUNCTAJ >1500
```

INTERSECT

```
SELECT NUME, CODS, LOC, PUNCTAJ  
FROM STUD  
WHERE PUNCTAJ >= 700
```

ORDER BY LOC DESC, 4;

Sfarsitul capitolului

SUBCERERI