10. Subinterogări

- Ce este o subinterogare
- Care sunt tipurile de probleme pe care le rezolvă subinterogările
- Care sunt tipurile de subinterogări
- Cum se scriu şi cum se folosesc subinterogările simple
- Cum se scriu şi cum se folosesc subinterogările multiple

- Sunteți patronul unei firme. În ultima perioadă unul dintre salariații firmei, pe nume Ionescu, s-a remarcat în mod deosebit prin activitatea sa. Ați decis de aceea să îi măriți salariul şi pentru a decide cu cât să-l măriți doriți să aflați care sunt persoanele cu salariu mai mare decât salariul lui Ionescu şi care sunt salariile câştigate de aceştia. Cum faceți acest lucru?
- Mai întâi veți determina salariul angajatului Ionescu:

SELECT salariul

FROM angajați

WHERE nume='Ionescu'

- Să notăm cu **S** salariul returnat de această comandă. Acum putem afişa foarte simplu angajații cu salariu mai mare decât **S**:

SELECT nume, prenume

FROM angajați

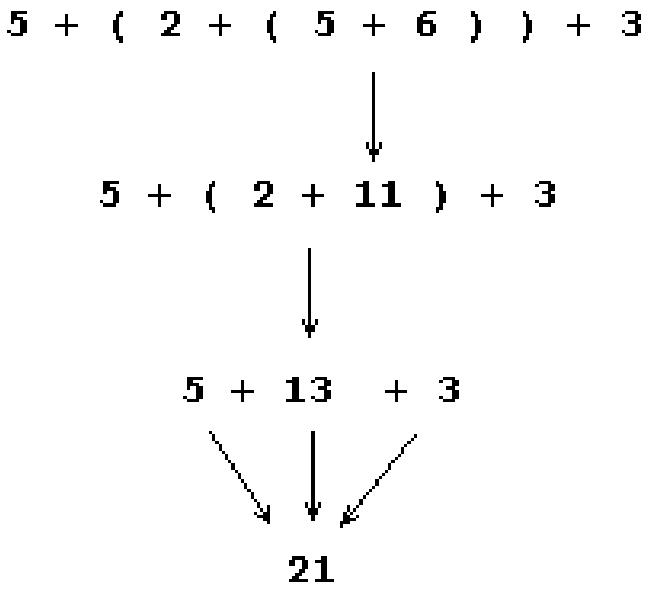
WHERE salariul>S

- Întrebarea care se pune acum este dacă nu există posibilitatea de a uni aceste două comenzi în una singură.
- Vom înlocui în a doua comandă valoarea **S** cu comanda care a generat această valoare astfel:

```
SELECT nume, prenume
FROM angajați
WHERE salariul > ( SELECT salariul
                   FROM angajați
                   WHERE nume='Ionescu' )
```

- Aşadar am inclus prima interogare în interiorul celei de a doua interogări. O astfel de interogare aflată în interiorul unei alte comenzi SQL se numeşte ***subinterogare***. Subinterogările sunt întotdeauna rulate înaintea comenzii în care sunt incluse, doar pe baza rezultatelor returnate de subinterogări putându-se obține rezultatele interogării exterioare subinterogării.

- Un proces similar cu modul de rulare al subinterogărilor este modul în care calculăm expresiile cu paranteze



- Subinterogările sunt în general folosite atunci când dorim să afişăm informaţii dintr-o tabelă pe baza unor informaţii pe care le preluăm din aceeaşi tabelă sau din alte tabele. De exemplu putem afişa angajaţii care lucrează în acelaşi departament cu angajatul **X** şi sunt mai tineri decât persoana **X**.

- Există două tipuri de subinterogări:
  - subinterogări simple care returnează o singură linie;
  - subinterogări multiple care returnează mai multe linii şi/sau mai multe coloane.
- Înainte de a prezenta fiecare din aceste tipuri de subinterogări trebuie să subliniem câteva restricții de utilizare a subinterogărilor:
  - o subinterogare va fi întotdeauna inclusă în paranteză
  - subinterogarea nu poate conține clauza **ORDER BY**.

Subinterogări simple

- Subinterogările simple, aşa cum am precizat, vor returna întotdeauna o singură valoare.
- Ele pot să apară în clauza **WHERE** sau în clauza **HAVING** şi sunt folosite împreună cu operatorii **<**, **>**, **<=**, **>=**, **<>**, **=**.
- Vom prezenta câteva exemple folosind următoarele tabele:

Persoane (Id, IdFirma, Nume, Localitate, DataN)

Firme (Id, Nume, Localitate)

- Dorim să afişăm toate persoanele care lucrează la aceeaşi firmă la care lucrează şi Ionescu:

```
SELECT Nume FROM persoane
WHERE IdFirma = ( SELECT IdFirma
                  FROM angajati
                  WHERE nume = 'Ionescu')
```

- Acelaşi rezultat l-am putea obține cu ajutorul unui selfjoin astfel:

```
SELECT p.nume
FROM persoane p, persoane i
WHERE p.IdFirma = i.IdFirma AND
      i.nume = 'Ionescu'
```

- însă folosirea subinterogărilor este mult mai uşoară şi mai naturală şi în general este mai rapidă.

- Iată un exemplu de folosire a operatorului **<>** împreună cu o subinterogare:

```
SELECT nume
FROM persoane
WHERE localitatea <> (SELECT localitatea
    FROM persoane
                WHERE nume='Ionescu')
```

- Comanda afişează toate persoanele care nu locuiesc în aceeaşi localitate cu Ionescu.

- Subinterogările pot folosi funcții de grup ca în exemplul următor:

SELECT nume FROM persoane

WHERE DataN = (SELECT max(DataN) FROM persoane)

- Această comandă va afişa cea mai tânără persoană din tabela **persoane**, data sa de naştere este cea mai mare, adică este cea mai recentă dată de naştere.

- Similar putem utiliza subinterogările simple în clauza **HAVING**. Să vedem de exemplu cum putem afişa codul firmei cu cei mai mulți angajați:

```
SELECT IdFirma FROM persoane
GROUP BY IdFirma
HAVING count(*) = ( SELECT max(count(*))
 FROM persoane
 GROUP BY IdFirma )
```

- Subinterogarea determină mai întâi numărul maxim de persoane angajate la o firmă, iar apoi afişează Id-ul firmei care are numărul de angajați egal cu acest maxim.

- Am fi tentați să scriem o comandă de forma:

```
SELECT DISTINCT IdFirma
FROM persoane
WHERE count(*) = ( SELECT max(count(*))
 FROM persoane
 GROUP BY IdFirma )
```

- dar am precizat în capitolul anterior funcțiile de grup **NU** pot să apară în clauza **WHERE**.

- Subinterogările pot fi imbricate una în alta pe oricâte nivele. Numărul maxim de nivele de imbricare a interogărilor este teoretic nelimitat. Singura limitare care poate interveni este dată de dimensiunea bufferelor.
- În exemplul următor, am construit o interogare care afişează numele firmei care are numărul maxim de angajați. Această interogare foloseşte interogarea din exemplul anterior pentru a determina Id-ul firmei cu număr maxim de angajați, iar apoi caută în tabela firme numele acestei firme.

```
SELECT nume
FROM firme
WHERE Id = (SELECT IdFirma
        FROM persoane
        GROUP BY IdFirma
        HAVING count(*) = ( SELECT max(count(*))
                    FROM personae
                    GROUP BY IdFirma
                  )
      )
```

- Interesant este faptul că în cadrul unei subinterogări se poate face referire la tabelele din clauza **WHERE** a interogării părinte. Astfel dacă dorim să afişăm toate persoanele care lucrează în aceeaşi localitate în care şi locuiesc vom scrie astfel:

```
select nume
from persoane p
where localitate =  ( select localitate
                      from firme f
                      where p.idfirma=f.id
                    )
```

- Am folosit subinterogarea pentru a afla localitatea în care se găseşte firma la care lucrează fiecare angajat în parte. Acest tip de subinterogări se numesc ***subinterogări corelate***.

Subinterogări multiple

- Am văzut cum putem utiliza subinterogările simple. Vom studia acum cum utilizăm subinterogările care returnează mai multe linii.
- Când o subinterogare returnează mai mult de o linie, nu mai este posibil să folosim operatorii de comparație **<**, **>**, **<=**, **>=**, **<>**, **=**, deoarece o valoare simplă nu poate fi comparată direct cu un set de valori.
- Va trebui să comparăm o valoare simplă cu fiecare valoare din setul de valori returnate de subinterogare.
- Pentru a realiza acest lucru vom folosi cuvintele cheie **ANY** şi **ALL** împreună cu operatorii de comparație, pentru a determina dacă o valoare este egală, mai mică sau mai mare decât orice valoare sau decât una din valorile din setul de date returnat de subinterogare.

- Pentru a exemplifica modul de folosire a subinterogărilor multiple vom utiliza tabela **jucători** cu următorul conținut:

Tabela **Jucatori**

ID	NUME	RATING	VARSTA	LOCALITATE
18	Ion	3	30	Sibiu
11	Iulian	6	18	Brasov
22	George	3	29	Bucuresti
38	Paul	2	20	Bucuresti
13	Andrei	4	19	Sibiu
24	Marian	3	26	Cluj-Napoca
48	Ilie	-	35	Sibiu
21	Alin	2	36	Brasov
17	Radu	1	22	Cluj-Napoca
63	Vasile	7	41	Iasi

Subinterogări multiple cu operatorul IN

- Cum aflăm oare numele şi localitatea jucătorilor a căror rating este egal cu al unui jucător sub **21** de ani? Vom afla mai întâi care sunt ratingurile jucătorilor sub **21** de ani:

SELECT rating

FROM jucatori

WHERE varsta<21

- Vom obține trei valori ale ratingului şi anume **2**, **4** şi respectiv **6**

RATING
6
2
4

- apoi vom afişa persoanele a căror rating este **2**, **3** sau **6**:

SELECT * FROM jucatori

WHERE rating IN (6, 2, 4)

ID	NUME	RATING	VARSTA	LOCALITATE
11	Iulian	6	18	Brasov
21	Alin	2	36	Brasov
38	Paul	2	20	Bucuresti
13	Andrei	4	19	Sibiu

- Aceste două comenzi se pot scrie împreună în una singură prin folosirea unei subinterogări multiple astfel:

SELECT * FROM jucatori

WHERE rating IN (SELECT rating FROM jucatori

WHERE varsta<21)

ID	NUME	RATING	VARSTA	LOCALITATE
11	Iulian	6	18	Brasov
21	Alin	2	36	Brasov
38	Paul	2	20	Bucuresti
13	Andrei	4	19	Sibiu

- Ce se întâmplă dacă o subinterogare multiplă returnează o valoare nulă iar operatorul folosit este **IN**? De exemplu ce va afişa comanda:

SELECT * FROM jucatori

WHERE rating IN (SELECT rating FROM jucatori

WHERE localitate='Sibiu')

- Mai întâi subinterogarea va afişa ratingurile tuturor persoanelor din Si

RATING
3
4
-

- deci interogarea anterioară este echivalentă cu

SELECT * FROM jucatori WHERE rating IN (3, 4, NULL)

- Sau

SELECT * FROM jucatori

WHERE rating=3 OR rating=4 OR rating=NULL

- însă din comparația cu **NULL** nu rezultă nimic (**NULL** nu poate fi comparat decât cu operatorii **IS NULL** sau **IS NOT NULL** în rest nu vom obține nici un rezultat), aşadar se vor afişa doar jucatorii cu ratingul egal cu **3** sau **4**:

ID	NUME	RATING	VARSTA	LOCALITATE
24	Marian	3	26	Cluj-Napoca
22	George	3	29	Bucuresti
18	Ion	3	30	Sibiu
13	Andrei	4	19	Sibiu

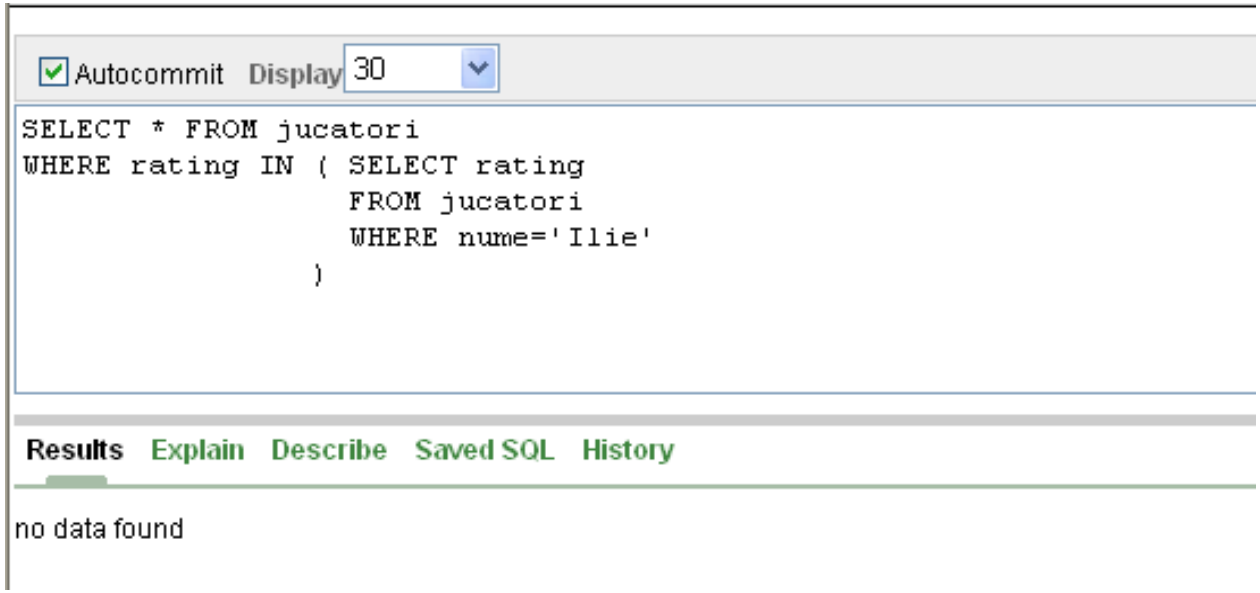
- Dacă însă subinterogarea va returna doar o singură valoare nulă ca de exemplu comanda:

SELECT rating FROM jucatori

WHERE nume='Ilie‘

RATING
-

atunci interogarea exterioară, neavând cu ce altă valoare să compare, nu va returna nici o linie:



Subinterogări multiple cu ALL

- Fie următoarea comandă:

SELECT * FROM jucatori

WHERE rating > ALL (SELECT rating FROM jucatori

WHERE varsta<21)

- Interogarea interioară returnează mulțimea valorilor ratingurilor tuturor persoanelor cu vârsta mai mică decât **21**, iar interogarea exterioară va verifica fiecare persoană din tabelă pentru a vedea dacă ratingul său este mai mare decât **fiecare** valoare returnată de către interogarea interioară.

- Interogarea interioară va returna valorile **2**, **4**, **6**, deci comanda anterioară este echivalentă cu

SELECT * FROM jucatori

WHERE rating > ALL (2, 4, 6)

sau

SELECT * FROM jucatori

WHERE rating>2 AND rating>4 AND rating>6

- În concluzie am afişat toate persoanele al căror rating este mai mare decât ratingul tuturor persoanelor mai mici de 21 de ani.

- Deci operatorul **>ALL** se poate interpreta ca **mai mare decât valoarea maximă** din mulțimea de valori returnată de către subinterogare.
- Similar operatorul **<ALL** se poate interpreta ca **mai mic decât valoarea minimă** din mulțimea valorilor returnate de către subinterogare

- Dacă una dintre valorile returnate de către interogarea interioară este nulă atunci interogarea exterioară nu va afişa nici o linie dacă este folosită opțiunea **ALL**.
- Dorim să afişăm toate persoanele cu rating mai mare decât ratingurile tuturor persoanelor din Sibiu:

```
select * from jucatori
where rating >ALL ( select rating
                    from jucatori
                    where localitate='Sibiu' )
```

- Interogarea interioară returnează următoarele valorile **3**, **4** şi **NULL** şi interogarea exterioară se poate scrie echivalent:

```
select * from jucatori
where rating>3 AND rating>6 AND rating>NULL
```

- Condiția din clauza **where** are valoarea true doar dacă toate cele trei condiții sunt adevărate. Însă expresia "**rating>NULL**" are valoarea **NULL**, adică nu este nici adevărată nici falsă.
- Aşadar condiția din clauza **WHERE** nu este adevărată niciodată şi comanda nu afişează nici o linie.

Subinterogări multiple cu ANY

- Dacă folosirea opțiunii **ALL** se putea traduce printr-o condiție compusă cu operatorul **AND**, în cazul opțiunii **ANY** se va putea traduce condiția în altă condiție care foloseşte operatorul **OR**.
- Fie următoarea comandă:

```
select * from jucatori
where rating >any ( select rating from jucatori
                    where vârsta<21 )
```

- Interogarea interioară returnează valorile **2**, **4** şi **6**.
- Comanda exterioară va afişa toți jucătorii care au un rating mai mare decât a oricărui jucător sub 21 de ani, sau altfel spus, se afişează persoanele cu rating mai mare decât a **cel puțin** unei persoane cu vârsta sub **21** de ani.

- Putem spune că operatorul **>ANY** poate fi interpretat ca **mai mare decât valoarea minimă** din mulțimea de valori returnată de către subinterogare.
- Similar operatorul **<ANY** se poate interpreta ca **mai mic decât valoarea maximă** din mulțimea valorilor returnate către subinterogare

- Dacă una din valorile returnate de către interogarea interioară este nulă, interogarea exterioară poate afişa totuşi ceva. De exemplu comanda

SELECT * FROM jucatori

WHERE rating >ANY (SELECT rating FROM jucatori

WHERE localitate='Sibiu')

- va afişa

ID	NUME	RATING	VARSTA	LOCALITATE
63	Vasile	7	41	Iasi
11	Iulian	6	18	Brasov
13	Andrei	4	19	Sibiu

- Acest lucru se întâmplă deoarece comanda dată se poate scrie echivalent

SELECT * FROM jucatori

WHERE rating >ANY (3, 4, NULL)

- deoarece subinterogarea returnează valorile **3**, **4** şi **NULL** şi această comandă se poate scrie şi

SELECT * FROM jucatori

WHERE rating>3 OR rating>4 OR rating>NULL

- Condiția din **WHERE** este adevărată dacă cel puțin una din cele trei condiții este adevărată.
- Cum ultima condiție, **rating>NULL**, nu va fi niciodată adevărată, este suficient ca ratingul jucătorului să fie mai mare decât **3** sau mai mare decât **4**, pentru ca el să fie afişat.

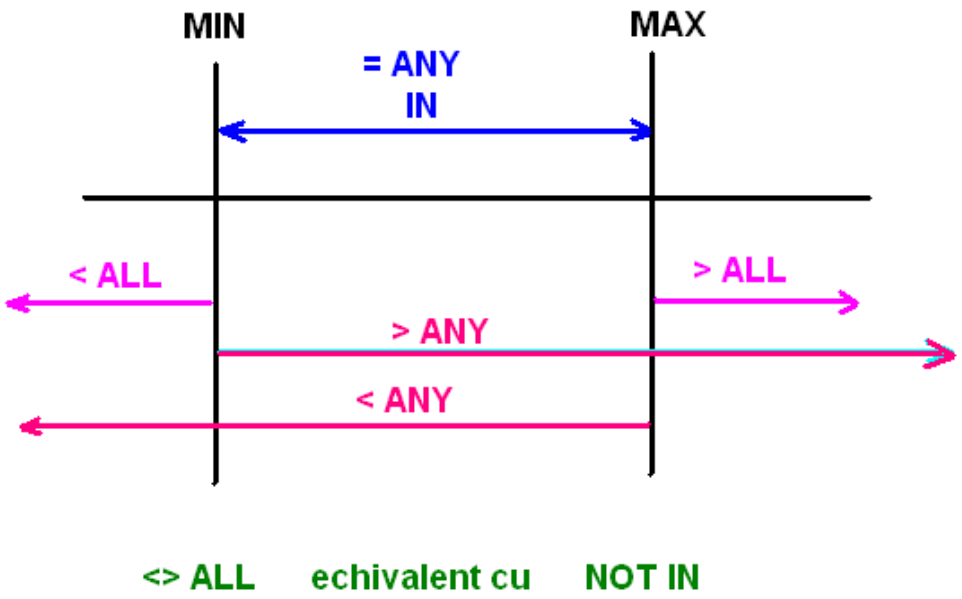
- Dacă însă subinterogarea va returna o singură valoare nenulă, şi nimic altceva, atunci comanda exterioară nu va afişa nimic:

```
SELECT * FROM jucatori
WHERE rating >ANY ( SELECT rating FROM jucatori
                    WHERE nume='Ilie' )
```

Results Explain Describe Saved SQL History

no data found

- Modul în care se pot folosi opțiunile **ANY**. **IN** şi **ALL** se pot rezuma în figura de mai jos



- Echivalențele ce se pot folosi cu aceste opțiuni sunt rezumate în tabelul următor:

IN	**=ANY**
NOT IN	**<> ALL**
< ANY	**< maxim**
> ANY	**> minim**
< ALL	**< minim**
> ALL	**> maxim**

Subinterogări multiple cu EXISTS

- Putem folosi operatorul **EXISTS** pentru a verifica dacă o subinterogare returnează vreo linie. De obicei se foloseşte acest operator împreună cu subinterogări corelate.
- Comanda următoare afişează toţi angajaţii care sunt managerii altor angajaţi:

```
SELECT employee_id, first_name, last_name
FROM employees a
WHERE EXISTS (SELECT employee_id FROM
                    employees b
                    WHERE b.manager_id=a.employee_id)
```

- În subinterogare am determinat angajaţii coordonaţi de către un angajat afişat de către interogarea exterioară.

- Această comandă o putem transcrie cu ajutorul operatorului IN astfel:

```
SELECT employee_id, first_name, last_name
FROM employees a
WHERE employee_id IN
    (SELECT employee_id FROM employees b
     WHERE a.employee_id=b.employee_id)
```

- Folosirea operatorului **EXISTS** oferă performanțe mai mari întrucât **IN** compară fiecare valoare returnată de către interogarea exterioară cu fiecare valoare returnată de subinterogare, pe când operatorul **EXISTS** verifică doar existența a cel puțin unei linii returnată de subinterogare, fără a face nici o comparație.

Subinterogări multiple în clauza FROM

- O subinterogare multiplă poate fi folosită şi în clauza **FROM** a unei interogări ca în exemplul următor:

```
SELECT a.employee_id, first_name,
   last_name, nrang
FROM employees a, (SELECT manager_id,
   count(*) nrang
                FROM employees GROUP BY
   manager_id
                HAVING count(*)>0) b
WHERE a.employee_id=b.manager_id
```

- care afişează id-ul, numele, prenumele şi numărul de subalterni ai tuturor managerilor

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	NRANG
100	Steven	King	5
101	Neena	Kochhar	2
102	Lex	De Haan	1
103	Alexander	Hunold	2
124	Kevin	Mourgos	4
149	Eleni	Zlotkey	3
201	Michael	Hartstein	1
205	Shelley	Higgins	1

În această lecție am învățat despre:

- Ce este o subinterogare
- Care sunt tipurile de probleme pe care le rezolvă subinterogările
- Care sunt tipurile de subinterogări
- Cum se scriu și cum se folosesc subinterogările simple
- Cum se scriu și cum se folosesc subinterogările multiple

SFÂRȘIT