5.2 Interogări în SQL

Cererile de interogare exprimate în SQL prezintă un aspect declarativ deoarece sunt specificate proprietăţile rezultatului şi nu modul de obţinere (SQL urmăreşte principiile calculului relaţional).

Cererile SQL sunt pasate pentru execuţie optimizatorului de cereri.

Optimizatorul de cereri este o componentă a sistemului de gestiune a bazelor de date care

- analizează cererea;
- selectează o strategie de execuţie;
- formulează o cerere echivalentă în limbajul procedural intern al sistemului de gestiune a bazelor de date.

5.2.1 Interogări simple

Interogarea unei baze de date poate fi exprimată în SQL prin intermediul instrucţiunii *select*, care are sintaxa:

Select *ExprAtribut* [[as] *Alias*] {, *ExprAtribut* [[as] *Alias*]}
from *NumeTabel* [[as] *Alias*] {, *NumeTabel* [[as] *Alias*]}
[where *Conditie*]

O cerere SQL va lua în considerare doar liniile ce aparţin produsului cartezian al tabelelor listate în clauza *from* şi va stabili liniile ce satisfac condiţia exprimată în clauza *where*.

Rezultatul execuţiei unei cereri SQL este un tabel, având câte o linie pentru fiecare linie selectată de clauza *where* şi ale cărui coloane rezultă din evaluarea expresiilor *ExprAtribut* ce apar în clauza *select* (lista ţintă).

Fiecare coloană poate fi redenumită cu ajutorul unui *Alias* ce urmează imediat după expresie.

Tabelele pot fi de asemenea redenumite prin intermediul unui *Alias.*

Exemplu

Se consideră o bază de date care conţine tabelele

ANGAJATI (Nume, Prenume, Dept, Birou, Salariu),
DEPARTAMENT (Dept, Adresa, Oras),

cu precizarea ca salariul înregistrat este anual.

ANGAJATI

Nume	Prenume	Dept	Birou	Salariu
Ionescu	Maria	Administratie	10	45
Popescu	Ion	Productie	20	36
Popa	Stefan	Administratie	20	40
Dumitrescu	Vasile	Distributie	16	45
Ionescu	Ion	Planificare	14	80
Manole	Radu	Planificare	7	73
Luca	Doru	Administratie	75	40
Vasile	Alina	Productie	20	46

DEPARTAMENT

Dept	Adresa	Oras
Administratie	Independentei	Iasi
Productie	Primaverii	Bucuresti
Distributie	Central	Focsani
Planificare	Nicolina	Iasi
Cercetare	Trandafirului	Cluj

Fig. 5.1 Conţinutul tabelelor ANGAJATI şi DEPARTAMENT

Interogarea 1: Să se găsească salariile angajaţilor cu numele Ionescu.

```
select Salariu as SalariuAnual
from Angajati
where Nume = 'Ionescu'
```

Salariu
45
80

Fig. 5.2. Rezultatul interogării 1

Lista ţintă - specifică elementele schemei tabelelor rezultat.

Caracterul special * poate să apară în lista ţintă şi reprezintă selecţia tuturor atributelor tabelelor precizate în clauza *from*.

Exemplu

Interogarea 2: Să se găsească toate informaţiile referitoare la angajatul cu numele Ionescu.

```
select *
from Angajati
where Nume = 'Ionescu'
```

Nume	Prenume	Dept	Birou	Salariu
Ionescu	Maria	Administratie	10	45
Ionescu	Ion	Planificare	14	80

Fig. 5.3 Rezultatul interogării 2

Lista ţintă poate conţine expresii ce utilizează valorile atributelor din fiecare linie selectată.

Exemplu

Interogarea 3: Găsiţi salariul lunar al angajaţilor cu numele Popescu.

```
select Salariu/12 as SalariuLunar
from Angajati
where Nume = 'Popescu'
```

SalariuLunar
3.00

Fig. 5.4 Rezultatul interogării 3

Clauza from

Dacă o interogare implică înregistrări din mai multe tabele, argumentul din clauza *from* va reprezenta o listă de tabele.

Condiţiile din clauza *where* sunt aplicate în acest caz produsului cartezian al acestor tabele; se poate specifica o joncţiune prin indicarea explicită a comparaţiilor între atribute din tabele diferite.

Interogarea 4: Să se găsească numele angajaţilor şi oraşele în care aceştia lucrează.

```
select Angajati.Nume, Angajati.Prenume, Departament.Oras
from Angajati, Departament
where  Angajati.Dept = Departament.Dept
```

Nume	Prenume	Oras
Ionescu	Maria	Iasi
Popescu	Ion	Bucuresti
Popa	Stefan	Iasi
Dumitrescu	Vasile	Focsani
Ionescu	Ion	Iasi
Manole	Radu	Iasi
Luca	Doru	Iasi
Vasile	Alina	Bucuresti

Fig. 5.5 Rezultatul interogării 4

În interogarea precedentă s-a folosit operatorul *punct* (’.’) pentru identificarea tabelului din care se extrag atributele.

Folosirea acestei construcţii este necesară în cazul în care tabelele din clauza *from* au atribute cu acelaşi nume, pentru a distinge între referinţele la atribute omonime.

În cazul în care nu există posibilitatea apariţiei unei ambiguităţi se poate specifica atributul fără a preciza tabelul căruia îi aparţine.

Într-o interogare se pot utiliza alias-uri pentru tabele cu scopul de a scurta referinţa la acestea.

Exemplu

Interogarea „Să se găsească numele angajaţilor şi oraşele în care aceştia lucrează” se poate exprima astfel:

```
select a.Nume, a.Prenume, d.Oras
from Angajati a, Departament d
where a.Dept = d.Dept
```

Clauza where

Condiţia din clauza *where* este o expresie booleană formată prin combinarea predicatelor simple cu operatorii *and*, *or* şi *not*.

Fiecare predicat simplu utilizează operatorii de comparaţie (=, >, >=, <, <=, <>) şi are, într-un membru, o expresie formată din valori ale atributelor dintr-o linie şi în celălalt membru o valoarea constantă sau o altă expresie.

Prioritar este operatorul *not*, dar nu se introduce o precedenţă între *and* şi *or*. Dacă într-o expresie se folosesc ambii operatori *and* şi *or* este indicată specificarea precedenţei prin utilizarea parantezelor.

Exemplu

Interogarea 5: Să se găsească prenumele angajaţilor cu numele *Ionescu* care lucrează în departamentele *Administratie* sau *Productie*.

```
select Prenume
from Angajati
where Nume = 'Ionescu' and (Dept = 'Administratie' or Dept = 'Productie')
```

Prenume
Maria

Fig. 5.6 Rezultatul interogării 5

Operatorul *like* - pentru compararea şirurilor de caractere

Acest operator compară un şir cu alt şir, specificat parţial cu ajutorul caracterelor speciale '_' şi '%'.

- Caracterul '_' substituie un caracter oarecare
- Caracterul '%' substituie un şir oarecare de caractere, posibil vid.

Exemplu

Comparaţia like 'ab%ba_' va fi satisfăcută de toate şirurile de caractere ce încep cu secvenţa ab şi au în componenţă perechea ba înainte de ultimul caracter.

Gestiunea valorilor NULL

Predicatul is NULL este adevărat doar dacă atributul are valoarea NULL.

Predicatul is not NULL este adevărat în caz contrar celui prezentat anterior.

Sintaxa: *Atribut* is [not] NULL

Duplicate

În SQL un tabel poate avea mai multe linii ce conţin aceleaşi valori pentru toate atributele (duplicate), spre deosebire de algebra relaţională şi calculul relaţional.

Dacă se doreşte emularea comportării din algebra relaţională în SQL ar trebui eliminate toate duplicatele la fiecare execuţie a unei operaţii de proiecţie.

Deoarece operaţia de eliminare a duplicatelor este consumatoare de timp şi adesea nu este necesară, executarea acestei operaţii este lăsată la latitudinea persoanei ce implementează interogarea.

Sintaxa eliminarea duplicatelor: select [<distinct | [all]>]

Opţiunea *all* indică faptul că vor fi păstrate toate înregistrările din rezultat (deci inclusiv duplicatele) şi este opţiunea implicită.

Exemplu

Se consideră tabelul PERSOANA (Cod, Nume, Prenume, Oras)

PERSOANA

Cod	Nume	Prenume	Oras
IS001122	Ionescu	Maria	Iasi
BC012345	Popescu	Ion	Bucuresti
IS123456	Ionescu	Vasile	Iasi
SV342345	Ionescu	Radu	Suceava

Fig. 5.7 Conţinutul tabelei PERSOANA

Interogarea 6: Să se găsească oraşele în care locuiesc persoanele cu numele Ionescu.

```
select Oras
from Persoana
where Nume = 'Ionescu'
```

Oras
Iasi
Iasi
Suceava

Interogarea 7: Să se găsească oraşele în care locuiesc persoanele cu numele Ionescu, fiecare oraş apărând o singură dată.

```
select distinct Oras
from Persoana
where Nume = 'Ionescu'
```

Oras
Iasi
Suceava

Joncţiuni

SQL-2 introduce o sintaxă alternativă pentru specificarea joncţiunilor, fiind astfel posibilă realizarea unei distincţii între condiţiile ce reprezintă condiţii de joncţiune şi cele ce reprezintă selecţii de linii.

Sintaxa este:

select *ExprAtribut* [[as] *Alias*] {, *ExprAtribut* [[as] *Alias*]}
from *NumeTabel* [[as] *Alias*]
{[*TipJonctiune*] join *NumeTabel* [[as] *Alias*] on *CondJonctiune*}
[where *AlteConditii*]

În acest fel, condiţia de joncţiune este mutată din clauza *where* în clauza *from*. Parametrul *TipJonctiune* specifică tipul joncţiunii: *inner*, *left* sau *full*.

Inner join corespunde theta-joncţiunii din algebra relaţională.

Exemplu

Interogarea „Să se găsească numele angajaţilor şi oraşele în care aceştia lucrează” se poate rescrie sub forma

```
select Nume, Prenume, Oras
from Angajati a inner join Departament d
    on a.Dept = d.Dept
```

În cazul unei joncţiuni, liniile dintr-un tabel ce nu au linii corespondente în celălalt tabel vor fi eliminate din rezultat.

Pentru a forţa apariţia unor astfel de linii în rezultat se poate apela la joncţiunea externă, cu cele trei variante:

- *left join* – furnizează acelaşi rezultat ca şi *inner join*, dar include şi liniile tabelului ce apare în stânga joncţiunii pentru care nu există linii corespondente în tabelul din dreapta;
- *right join* – păstrează liniile tabelului din dreapta ce nu au corespondent în tabelul din stânga;
- *full join* – furnizează acelaşi rezultat ca şi *inner join*, suplimentat cu liniile excluse din ambele tabele.

Exemplu

Se consideră o bază de date care conţine tabelele prezentate în figură

SOFERI

Nume	Prenume	ID
Ionescu	Maria	VR 001Y
Popescu	Ion	PZ 111B
Popa	Stefan	AP 222C

AUTOVEHICULE

NrInreg	Marca	Model	ID
IS01AAA	BMW	323	VR 001Y
SV02BBB	BMW	Z3	VR 001Y
IS02CCC	Lancia	Delta	PZ 111B
IS01EFD	BMW	316	MI 222C

Fig. 5.9 Conţinutul tabelelor SOFERI şi AUTOVEHICULE

Interogarea 8: Să se găsească şoferii ce deţin autovehicule, incluzând şi şoferii fără autovehicule.

```
select Nume, Prenume, Soferi.ID, NrInreg, Marca, Model
from Soferi left join Autovehicule on
        (Soferi.ID = Autovehicule.ID)
```

Nume	Prenume	ID	NrInreg	Marca	Model
Ionescu	Maria	VR 001Y	IS01AAA	BMW	323
Ionescu	Maria	VR 001Y	SV02BBB	BMW	Z3
Popescu	Ion	PZ 111B	IS02CCC	Lancia	Delta
Popa	Stefan	AP 222C	NULL	NULL	NULL

Interogarea 9: Să se găsească toţi şoferii şi toate maşinile împreună cu posibilele relaţii între ele.

```
select Nume, Prenume, Soferi.ID, NrInreg, Marca, Model
from Soferi full join Autovehicule on
        (Soferi.ID = Autovehicule.ID)
```

Nume	Prenume	ID	NrInreg	Marca	Model
Ionescu	Maria	VR 001Y	IS01AAA	BMW	323
Ionescu	Maria	VR 001Y	SV02BBB	BMW	Z3
Popescu	Ion	PZ 111B	IS02CCC	Lancia	Delta
Popa	Stefan	AP 222C	NULL	NULL	NULL
NULL	NULL	NULL	IS01EFD	BMW	316

Unele implementări de SQL specifică joncţiunea externă prin adăugarea unui caracter special sau a unei secvenţe de caractere (* sau (+)) la atributele implicate în condiţia de joncţiune.

Exemplu

Interogarea „Să se găsească şoferii ce deţin autovehicule, incluzând şi şoferii fără autovehicule” se poate exprima sub forma

```
select Nume, Prenume, Soferi.ID, NrInreg, Marca, Model
    from Soferi , Autovehicule
    where Soferi.ID * = Autovehicule.ID
```

SQL-2 oferă posibilitatea realizării joncţiunii naturale (joncţiune pe baza atributelor cu acelaşi nume) a două tabele prin utilizarea cuvântului cheie *natural* în faţa tipului joncţiunii.

Exemplu

Interogarea „Să se găsească toţi şoferii şi toate maşinile împreună cu posibilele relaţii între ele” se poate exprima sub forma

```
select Nume, Prenume, Soferi.ID, NrInreg, Marca, Model
    from Soferi natural full join Autovehicule
```

Observaţii

În mod normal, joncţiunea naturală nu este disponibilă în sistemele comerciale. Motivele acestei excluderi sunt :

- comportarea unei interogări se poate modifica în mod semnificativ ca rezultat al unei mici modificări a schemei;

- joncţiunea naturală impune analizarea completă a schemelor tabelelor implicate, cu scopul de a înţelege condiţia de joncţiune.

Utilizarea variabilelor

Prin folosirea alias-urilor se poate referi un tabel de mai multe ori, într-un mod similar operatorului de redenumire ρ din algebra relaţională.

Când este introdus un alias, se declară o variabilă tip tabel care are ca valoare conţinutul tabelului pentru care se introduce alias-ul.

Când un tabel apare doar o singură dată în interogare, nu este nici o diferenţă între a interpreta alias-ul ca pseudonim sau ca o nouă variabilă.

Când tabelul apare de mai multe ori este esenţial să privim alias-ul ca o nouă variabilă.

ANGAJATI

Nume	Prenume	Dept	Birou	Salariu
Ionescu	Maria	Administratie	10	45
Popescu	Ion	Productie	20	36
Popa	Stefan	Administratie	20	40
Dumitrescu	Vasile	Distributie	16	45
Ionescu	Ion	Planificare	14	80
Manole	Radu	Planificare	7	73
Luca	Doru	Administratie	75	40
Vasile	Alina	Productie	20	46

DEPARTAMENT

Dept	Adresa	Oras
Administratie	Independentei	Iasi
Productie	Primaverii	Bucuresti
Distributie	Central	Focsani
Planificare	Nicolina	Iasi
Cercetare	Trandafirului	Cluj

Exemplu

Interogarea 10: Se consideră baza de date de mai sus. Să se găsească toţi angajaţii ce au acelaşi nume (dar prenume diferite) cu un angajat care lucrează în departamentul Producţie.

```
select a1.Nume, a1.Prenume
from Angajati a1, Angajati a2
where  a1.Nume = a2.Nume and
          a1.Prenume <> a2.Prenume and
          a2.Dept = 'Productie'
```

Utilizarea alias-urilor de tabel are importanţă din următoarele puncte de vedere:

- se evită necesitatea scrierii întregului nume al tabelului ori de câte ori este cerut acest lucru;
- se poate face referire de mai multe ori la acelaşi tabel; introducerea unui alias are semnificaţia declarării unei variabile de tip tabel, ce are acelaşi conţinut cu tabelul al cărui alias este;
- se pot specifica cererile imbricate.

Ordonarea

În general, rezultatul unei interogări conţine linii, aranjate într-o ordine oarecare. Dacă se doreşte impunerea unei ordonări după un anumit criteriu asupra liniilor returnate de o interogare se va utiliza clauza *order by*. Sintaxa este:

order by *Atribut* [asc | desc]
{, *Atribut* [asc | desc]}

Exemplu

Interogarea 11: Extrageţi conţinutul tabelului AUTOVEHICULE în ordine descendentă după Marcă şi Model.

```
select *
from Autovehicule
order by Marca desc, Model desc
```

NrInreg	Marca	Model	ID
IS02CCC	Lancia	Delta	PZ 111B
SV02BBB	BMW	Z3	VR 001Y
IS01AAA	BMW	323	VR 001Y
IS01EFD	BMW	316	MI 222C

5.2.2 Interogări agregate

Operatorii agregaţi constituie una din cele mai importante extensii ale SQL în comparaţie cu algebra relaţională.

În algebra relaţională toate condiţiile sunt evaluate pentru un singur tuplu la un moment dat. Adesea apare necesitatea evaluării unor proprietăţi ce depind de o mulţime de tupluri (de exemplu aflarea numărului de angajaţi ce lucrează în departamentul Producţie).

Operatorii agregaţi sunt:

- *count*

  Sintaxă: count (< * | [distinct | all] *ListaAtribute*>)

  - Opţiunea * returnează numărul de linii din rezultat.
  - Opţiunea *distinct* returnează numărul valorilor distincte pentru lista de atribute din rezultat.
  - Opţiunea *all* returnează numărul liniilor ce conţin valori diferite de NULL pentru lista de atribute.

Dacă se specifică un atribut fără un *distinct* sau *all* se consideră opţiunea implicită *all*.

Exemplu

Se consideră baza de date cu relaţiile

ANGAJATI (Nume, Prenume, Dept, Birou, Salariu),
DEPARTAMENT (Dept, Adresa, Oras).

Interogarea 12: Să se găsească numărul angajaţilor din departamentul Productie.

```
select count (*)
from Angajati
where Dept = 'Productie'
```

Interogarea 13: Să se găsească numărul valorilor distincte pentru atributul Salariu pentru toţi angajaţii din tabela ANGAJAŢI.

```
select count (distinct Salariu)
from Angajati
```

Interogarea 14: Să se găsească numărul de linii din tabelul ANGAJATI care au valori diferite de NULL pentru atributul Salariu.

```
select count (all Salariu)
from Angajati
```

- *sum, max, min, avg*

Sintaxă: <sum | max | min | avg> ([distinct | all] *ExprAtribut*)

Aceşti operatori se aplică liniilor selectate de clauza *where* a interogării şi au următoarele semnificaţii:

- *sum* returnează suma valorilor deţinute de expresia atribut;
- *max* şi *min* returnează valoarea maximă, respectiv minimă;
- *avg* returnează media valorilor expresiei atribut.

Expresia atribut *ExprAtribut* poate fi un atribut sau o expresie.

Operatorii *sum* şi *avg* acceptă ca argument expresii ce reprezintă valori numerice sau intervale de timp.

Funcţiile *min* si *max* necesită definirea unei ordini în expresia atribut, fiind aplicabile şi asupra şirurilor de caractere şi momentelor de timp.

Cuvintele cheie *distinct* şi *all* au semnificaţiile discutate deja la operatorul *count*.

Exemplu

Se consideră baza de date cu relaţiile

ANGAJATI (Nume, Prenume, Dept, Birou, Salariu),
DEPARTAMENT (Dept, Adresa, Oras).

Interogarea 15: Să se găsească salariul maxim, mediu şi minim pentru toţi angajaţii din tabela ANGAJAŢI.

```
select max(Salariu), avg(Salariu), min(Salariu)
from Angajati
```

Interogarea 16: Să se găsească salariul maxim pentru angajaţii care lucrează într-un departament din Iasi.

```
select max(Salariu)
from Angajati a, Departament d
where a.Dept = d.Dept  and Oras = 'Iasi'
```

5.2.3 Interogări group by

Funcţiile agregat prezentate operează pe toate liniile returnate de interogare.

În cazul în care se doreşte utilizarea funcţiilor agregat pe o submulţime a liniilor selectate SQL pune la dispoziţie clauza *group by*.

Această clauză specifică modul în care va fi împărţit tabelul în submulţimi de linii.

Clauza acceptă ca argument o mulţime *X* de atribute, iar interogarea va opera separat pe fiecare mulţime de linii ce posedă aceleaşi valori pentru *X*.

Pentru a înţelege mai bine semnificaţia clauzei *group by* să analizăm următorul exemplu:

Exemplu

Se consideră tabela ANGAJATI (Nume, Prenume, Dept, Birou, Salariu)

Interogarea 17: Să se găsească suma salariilor angajaţilor din acelaşi departament.

```
select Dept, sum(Salariu)
from Angajati
group by Dept
```

Nume	Prenume	Dept	Birou	Salariu
Ionescu	Maria	Administratie	10	45
Popescu	Ion	Productie	20	36
Popa	Stefan	Administratie	20	40
Dumitrescu	Vasile	Distributie	16	45
Ionescu	Ion	Planificare	14	80
Manole	Radu	Planificare	7	73
Luca	Doru	Administratie	75	40
Vasile	Alina	Productie	20	46

Într-o primă fază interogarea este executată fără a ţine cont de clauza *group by*. De fapt se execută interogarea

```
select Dept, Salariu
from Angajati
```

Dept	Salariu
Administratie	45
Productie	36
Administratie	40
Distributie	45
Planificare	80
Planificare	73
Administratie	40
Productie	46

Fig. 5.13 Proiecţia pe atributele Dept şi Salariu a tabelei ANGAJAŢI

Tabelul rezultat este apoi împărţit în mulţimi ce au aceeaşi valoare pentru atributele listate în clauza *group by*.

Dept	**Salariu**
Administratie	45
Administratie	40
Administratie	40
Productie	36
Productie	46
Distributie	45
Planificare	80
Planificare	73

Fig. 5.14 Regrupare în acord cu valorile atributului Dept

Odată stabilite grupurile de linii, funcţia agregat se aplică fiecărui grup în parte. Rezultatul final al interogării este tabelul din figura 5.15.

Dept	**Salariu**
Administratie	125
Productie	82
Distributie	45
Planificare	153

Fig. 5.15 Rezultatul final al interogării

Restricţie - atributele ce pot apărea în clauza *select* să fie o submulţime a atributelor din clauza *group by*.

Motivul acestei restricţii va fi prezentat prin următorul exemplu.

Exemplu

Se consideră tabelul ANGAJATI (Nume, Prenume, Dept, Birou, Salariu).
Fie interogarea

```
select Birou
from Angajati
group by Dept
```

Această interogare este incorectă, deoarece pentru aceeaşi valoare a atributului Dept, atributul Birou deţine mai multe valori.

După ce liniile au fost grupate după atributul Dept, fiecare grup trebuie să corespundă unei singure linii în tabelul returnat de interogare.

Interogarea corectă este

```
select Birou
from Angajati
group by Dept, Birou
```

Pentru a lua în considerare doar grupurile de linii ce satisfac anumite condiţii trebuie utilizată clauza *having*.

Dacă aceste condiţii pot fi verificate la nivel de linie, atunci este suficientă utilizarea predicatelor corespunzătoare ca argument al clauzei *where*.

Clauza *having* conţine condiţii ce trebuie aplicate la terminarea execuţiei interogării ce utilizează clauza *group by*. Fiecare submulţime de linii va participa la formarea rezultatului doar dacă satisface condiţia din clauza *having*.

Sintaxa permite ca în clauza having să apară expresii booleene, formate din predicate simple şi operatori booleeni. Predicatele simple pot fi:

- comparaţii între rezultatul evaluării unei funcţii agregat şi o expresie generică
- comparaţii între atribute ce formează clauza *group by* şi o expresie generică.

Se recomandă ca în clauza *having* să apară doar predicatele ce implică o funcţie agregat şi restul predicatelor să fie incluse în contextul clauzei *where*.

Exemplu

Se consideră tabelul ANGAJATI (Nume, Prenume, Dept, Birou, Salariu)

Interogarea 18: Să se găsească departamentele în care salariul mediu al angajaţilor din biroul 20 este mai mare ca 25.

```
select Dept
from Angajati
where Birou = ’20’
group by Dept
having avg(Salariu) > 25
```

Forma completă a unei instrucţiuni *select* devine

InterogareSQL ::= select *ListaTinta*
from *ListaTabele*
[where *Conditie*]
[group by *ListaAtributeGrupare*]
[having *ConditieAgregata*]
[order by *ListaAtributeOrdonare*]

5.2.4 Interogări cu operatori din teoria mulţimilor

SQL pune la dispoziţie operatori din teoria mulţimilor, cum ar fi operatorii de reuniune (*union*), intersecţie (*intersect*) şi diferenţă (*except* sau *minus*).

Orice interogare ce utilizează operatorii de intersecţie şi diferenţă poate fi exprimată cu ajutorul interogărilor imbricate.

Sintaxa pentru utilizarea operatorilor din teoria mulţimilor este:

InterogareSQL {<union | intersect | except> [all] *InterogareSQL*}

Operatorii din teoria mulţimilor presupun eliminarea duplicatelor ca opţiunea implicită.

Dacă se doreşte utilizarea acestor operatori cu menţinerea duplicatelor este suficientă specificarea opţiunii *all*.

Observaţii. SQL nu impune ca schemele pe care se execută operaţiile să fie identice (spre deosebire de algebra relaţională), ci doar ca atributele să aibă domenii compatibile.

Corespondenţa între atribute nu se bazează pe nume, ci pe poziţia atributelor. Dacă atributele au nume diferite, rezultatul va prelua numele de atribute din primul operand.

Exemple

Se consideră tabelul ANGAJATI (Nume, Prenume, Dept, Birou, Salariu)

a) Să se găsească numele şi prenumele tuturor angajaţilor.

```
select Prenume as NumeAngajat
from Angajati
union
select Nume
from Angajati
```

Nume	Prenume	Dept	Birou	Salariu
Ionescu	Maria	Administratie	10	45
Popescu	Ion	Productie	20	36
Popa	Stefan	Administratie	20	40
Dumitrescu	Vasile	Distributie	16	45
Ionescu	Ion	Planificare	14	80
Manole	Radu	Planificare	7	73
Luca	Doru	Administratie	75	40
Vasile	Alina	Productie	20	46

NumeAngajat
Ionescu
Popescu
Popa
Dumitrescu
Manole
Luca
Vasile
Maria
Ion
Stefan
Radu
Doru
Alina

b) Să se găsească numele de angajaţi care sunt şi prenume

```
select Prenume as NumeAngajat
from Angajati
intersect
select Nume
from Angajati
```

NumeAngajat
Vasile

c) Să se găsească numele de angajaţi care nu sunt şi prenume

```
select Nume as NumeAngajat
from Angajati
except
select Prenume
from Angajati
```

NumeAngajat
Ionescu
Popescu
Popa
Dumitrescu
Manole
Luca

5.2.5 Interogări imbricate

Până acum toate interogările formulate conţineau în clauza *where* o condiţie compusă, în care fiecare predicat reprezintă o comparaţie între două valori.

Se pot defini predicate cu structură complexă, în care o expresie valorică poate fi comparată cu rezultatul execuţiei unei interogări SQL. Interogarea utilizată în comparaţie se defineşte în interiorul predicatului din clauza *where* şi se numeşte *interogare imbricată*.

În general, primul operand al unei comparaţii de genul celei amintite anterior este un atribut, în timp ce în celălalt membru avem o mulţime de valori (rezultatul interogării).

Pentru a rezolva această problemă a eterogenităţii termenilor comparaţiei, SQL pune la dispoziţie cuvintele cheie *any* şi *all* pentru a extinde operatorii de comparaţie.

any - specifică faptul că linia este validă dacă valoarea atributului se află în relaţie cu cel puţin o valoare returnată de interogarea imbricată.

all - specifică faptul că linia este validă dacă valoarea atributului se află în relaţie cu toate valorile returnate de interogare.

Sintaxa cere ca domeniul elementelor returnate de interogarea imbricată să fie compatibil cu atributul cu care se face comparaţia.

Exemplu

Se consideră tabelele

ANGAJATI (Nume, Prenume, Dept, Birou, Salariu),
DEPARTAMENT (Dept, Adresa, Oras).

Interogarea 19: Să se găsească angajaţii ce lucrează într-un departament din Iaşi.

```
select Nume, Prenume
from Angajati
where Dept = any (select Dept from Departament where Oras = 'Iasi')
```

Observaţie. Această interogare poate fi rezolvată prin realizarea unei joncţiuni între cele două tabele.

Interogarea 20: Să se găsească departamentele în care nu lucrează nici un angajat cu numele Ionescu.

```
Select Dept
from Departament
where Dept <> all  (select Dept
                    from Angajati
                    where Nume = 'Ionescu')
```

```
select Dept
from Departament
except
select Dept
from Angajati
where Nume = 'Ionescu'
```

Operatorii *in* şi *not in* - reprezintă apartenenţa la o mulţime. Aceşti operatori sunt echivalenţi cu = *any*, respectiv <> *all*.

Să mai facem observaţia că funcţiile agregat *max* şi *min* pot fi utilizate în interogările imbricate.

Exemplu

Se consideră tabelele ANGAJATI şi DEPARTAMENT prezentate în figura 5.1.

Interogarea 21: Să se găsească departamentele în care lucrează angajaţii ce câştigă cel mai mare salariu.

```
select Dept
from Angajati
where Salariu = any (select max(Salariu)
                     from Angajati)
```

```
select Dept
from Angajati
where Salariu >= all  (select Salariu
                       from Angajati)
```

Observaţii.

- Deşi cele două interogări sunt echivalente, este indicată folosirea funcţiilor agregat deoarece sunt mai concludente şi se execută mai eficient.
- În cazul primei interogări nu există nici o diferenţa dacă în loc de operatorul *any* se foloseşte operatorul *all* (deoarece interogarea internă are ca rezultat o singură linie).

Pentru a înţelege mecanismul rezolvării interogărilor ce conţin interogări imbricate se pleacă de la presupunerea că interogarea imbricată se execută înaintea analizei liniilor din interogarea externă.

Rezultatul interogării poate fi salvat într-o variabilă temporară şi predicatul interogării externe poate fi evaluat cu ajutorul rezultatului temporar.

Uneori interogarea imbricată face referire la contextul interogării în care este imbricată; acest lucru are loc prin intermediul unei variabile definită în interogarea externă şi utilizată în cea internă.

Un astfel de mecanism este cunoscut sub numele de *transferul legăturilor* dintr-un context în altul. În acest caz, noua interpretare pentru interogările imbricate este următoarea: pentru fiecare linie din interogarea externă se evaluează mai întâi interogarea imbricată şi apoi se evaluează predicatul din interogarea externă.

Vizibilitatea variabilelor SQL

O variabilă poate fi utilizată doar în interiorul interogării în care este definită sau în interogarea imbricată din interogarea în care este definită.

Dacă o interogare conţine interogări imbricate pe acelaşi nivel, variabilele declarate în clauza *from* a unei interogări nu pot fi utilizate în contextul celeilalte interogări.

Operatorului logic *exists*

Acest operator acceptă ca parametru o interogare imbricată şi returnează valoarea *adevărat* doar dacă interogarea nu produce un rezultat vid.

Exemplu

Se consideră relaţia PERSOANA (Cod, Nume, Prenume, Oras).

Interogarea 22: Să se găsească persoanele care au acelaşi nume şi prenume, dar coduri diferite.

```
select *
from Persoana P where exists  (select * from Persoana P1
                                where P1.Nume = P.Nume
                                 and P1.Prenume = P.Prenume
                                 and P1.Cod <> P.Cod )
```

În acest caz nu se poate executa interogarea imbricată înaintea evaluării interogării externe, dat fiind că interogarea imbricată nu este definită până când nu se asignează o valoare variabilei P.

Este necesar în schimb să se evalueze interogarea imbricată pentru fiecare linie produsă în cadrul interogării externe.

În exemplul prezentat vor fi examinate mai întâi liniile variabilei P una câte una. Pentru fiecare din aceste linii va fi executată interogarea imbricată. Această interogare poate fi formulată realizând o joncţiune a tabelei PERSOANA cu ea însăşi.

O altă cale de a formula interogarea din exemplul anterior este prin folosirea constructorului de tuplu, reprezentat de o pereche de paranteze rotunde care marchează lista de atribute.

Exemplu

Se consideră relaţia PERSOANA (Cod, Nume, Prenume, Oras).

Interogarea 23: Să se găsească toate persoanele ce nu au omonime.

```
select *
from Persoana P
where (Nume, Prenume) not in (select Nume, Prenume
                              from Persoana P1
                              where P1.Cod <> P.Cod )
```

Sistemele comerciale nu rezolvă întotdeauna interogările imbricate prin scanarea tabelului extern şi producerea unei interogări pentru fiecare linie din relaţie.

În schimb se încearcă procesarea cât mai multor interogări într-o manieră orientată pe mulţimi, cu scopul de a manevra cantităţi mari de date prin cât mai puţine operaţii posibile.