Sisteme de baze de date

Curs 1 – Modelul relațional

Sorina Preduț
sorina.predut@unibuc.ro
Universitatea din București

Cuprins

Modelul relațional

- reprezintă **datele** sub forma unor structuri bidimensionale, asemănătoare **tabelelor**.
- a apărut pentru a uşura **concepţia**, **accesarea** şi **procesarea datelor**.
- modelele de date existente anterior (**ierarhic** şi **reţea**) erau foarte de complexe.
  - performanţele acestor tipuri de BD (baze de date) erau dependente de proiectarea şi designul lor.
  - programatorul aplicaţiei trebuia să cunoască implementarea fizică a BD şi limbajul de manipulare a datelor corespunzător.

Modelul relațional – cont.

- simplitatea acestui model şi a limbajului său de manipulare a datelor permite utilizatorilor să execute propriile lor manipulări.
- conceptele teoretice care formează baza modelului relaţional au apărut la începutul anilor ‘70, în principal datorită lui **Edgar F. Codd**.
- acest model diferă de modelele precedente prin aceea că **nu se bazează pe dependenţele de secvenţă şi cale**.
- nu este orientat spre sistemul de calcul; **utilizatorul poate obţine date specifice folosind un limbaj ne-procedural** care îi permite să descrie datele cerute în loc de navigaţia folosită.

Modelul relațional – cont.

- modelul relaţional şi operaţiile folosite de către acesta au la bază **o riguroasă fundamentare matematică**:
  - algebra relaţională şi calculul relaţional.
- operaţiile de manipulare a datelor pot fi privite ca o serie de operaţii de
  - **proiecţie**,
  - **filtrare**,
  - **reuniune**,
  - **intersecţie**, etc.
- asupra datelor din tabele.

Modelul relațional – cont.

- **avantaje**:
  - simplitatea,
  - fundamentarea matematică riguroasă şi
  - independenţa datelor, adică separarea aspectelor logice ale acestora de cele fizice.

I. Concepte de bază

- Definirea unui model de date presupune precizarea şi identificarea elementelor:
  - **structurile de date** folosite,
  - **operatorii** care acţionează asupra structurilor de date,
  - **restricţiile** care trebuie impuse pentru menţinerea corectitudinii datelor, numite şi **restricţii de integritate**.

I. Concepte de bază

- În modelul relaţional datele sunt reprezentate ca structuri bidimensionale, numite **relații**.
- O relaţie este alcătuită dintr-un număr fix de elemente numite **atribute**, fiecare dintre acestea putând lua valori într-o mulţime finită, numită **domeniu**.
- Numărul de atribute al unei relaţii se numeşte **aritatea** relaţiei.
- **relaţie (atribut_1, atribut_2, ..., atribut_n)**
- De exemplu, datele despre angajaţii unei firme se pot reprezenta ca o relaţie de aritate 7 astfel:

```
salariat (cod_salariat, nume, prenume, adresa, dată_naştere, sex,
cod_departament)
```

I. Concepte de bază - cont.

- Elementele unei relaţii se numesc **tupluri** sau **înregistrări**.
- De obicei relaţiile sunt reprezentate sub forma unor **tabele**, în care **fiecare rând reprezintă un tuplu** şi **fiecare coloană reprezintă valorile** tuplurilor corespunzătoare **unui** anumit **atribut**.

cod_salariat	nume	prenume	Adresa	dată_naştere	sex	cod_ departament
S1	Ion	Cosmin	Str. Jiului Nr.3	01/01/1970	M	D1
S2	Popescu	Vasile		15/11/1956	M	D1
S3	Ionescu	Gina	Str. Valea Albă Nr.4	28/02/1976	F	D2
S4	Costache	Viorel	Str. Crişana Nr. 11	12/02/1975	M	D1

I. Concepte de bază - cont.

- **Relaţiile** ce constituie o BD trebuie **să satisfacă mai multe condiţii**:
  1. Fiecare atribut trebuie să poarte un nume, care este unic în cadrul relaţiei.
  2. Fiecare atribut poate avea doar valori atomice, deci care nu se pot descompune dpdv logic.
  3. Fiecare tuplu este unic. Nu sunt permise tupluri identice (duplicat). În plus, unicitatea unui tuplu nu este limitată la datele existente: fiecare tuplu trebuie să fie unic tot timpul.

I. Concepte de bază - cont.

- ➢ Într-o relaţie ∃ întotdeauna unul sau mai multe atribute care asigură că **tuplul** va rămâne **unic tot timpul**.
- ➢ Un atribut sau set de atribute care identifică în mod unic un tuplu se numeşte **cheie candidată** sau mai simplu **cheie**.
- ➢ Pt. ∀ relaţie se alege din mulţimea cheilor candidate o cheie care s.n. **cheie primară** a relaţiei.
- ➢ Eventualele alte chei candidate, diferite de cheia primară, se numesc **chei alternative**.

I. Concepte de bază - cont.

- Când mai mult de un singur atribut este necesar pentru a crea o cheie, se spune că avem o **cheie compusă**.
- În cazul unei chei compuse, atributele care fac parte din cheie sunt alese astfel încât nici unul să nu fie în plus, adică nici un atribut nu poate fi şters din cheia candidată a. î. partea din cheia candidată care rămâne să identifice încă în mod unic tuplurile; de aceea, se spune că **o cheie trebuie sa fie minimală**.

I. Concepte de bază - cont.

- Uneori, în unele tupluri dintr-o relaţie, u**n atribut poate fi necunoscut** sau **neaplicabil**. Pentru a reprezenta un astfel de atribut se foloseşte o valoare convenţională, notată cu **Null**.
- Ca regulă generală, nici unul dintre atributele care alcătuiesc cheia primară nu poate avea valoarea Null pentru nici unul din tuplurile relaţiei.

Exemplu

- Pentru relaţia `salariat` definită anterior, o cheie candidată este `cod_salariat`. Dacă se presupune că `nume`, `prenume` şi `dată_naştere` identifică în mod unic salariatul, atunci combinaţia acestor trei atribute este o altă cheie candidată, care, datorită faptului că este alcătuită din mai multe atribute, s. n. şi cheie compusă.
  Dacă cheia `cod_salariat` este aleasă cheie primară, atunci combinaţia celor trei atribute (`nume`, `prenume` şi `dată_naştere`) devine cheie alternativă.

Exemplu - cont.

- În alegerea cheilor candidate s-a ţinut cont de **principiul minimalităţii** în sensul că din cheia compusă `nume`, `prenume` şi `dată_naştere` nu poate fi şters nici un atribut şi nici nu mai trebuie adăugat altul, aceste trei atribute identificând în mod unic tuplurile.
  De exemplu, combinaţia (`nume`, `prenume`, `dată_naştere`, `cod_departament`) nu este o cheie a relaţiei `salariat` pentru că şi în absenţa atributului `cod_departament`, un salariat este identificat în mod unic de primele trei atribute.

Exemplu - cont.

- Observăm că tuplul identificat prin valoarea cheii primare egală cu S2 are atributul `adresa` necunoscut, adică acesta are valoarea Null.
  Prin urmare, atributul `adresa` nu poate fi cheie primară şi nici nu poate intra în componenţa unei chei compuse care este şi cheie primară.

I. Concepte de bază - cont.

- S.n. **cheie străină** un atribut sau o mulţime de atribute ale unei relaţii R1 care există şi într-o altă relaţie R2, nu neapărat distinctă de R1, şi care formează cheia primară a relaţiei R2.
  În cazul acesta, **cheia străină din R1** se spune că **face referinţă la cheia primară din R2**.
  Valorile pe care le ia cheia străină, dacă nu sunt nule, trebuie să se găsească printre valorile cheii primare la care face referinţă.

Exemplu

- Dacă mai definim o relaţie de aritate 3, pt. evidenţa departamentelor din firmă:
  `departament (cod_departament, denumire, localitate)`

cod_departament	denumire	localitate
D1	Analiză financiară	Bucureşti
D2	Depozit	Piteşti
D3	Contabilitate	Constanţa
D4	Magazin	Piteşti

Exemplu - cont.

- atributul `cod_departament` din relaţia `salariat` devine cheie străină care face referinţă la cheia primară a relaţiei departament.
- Valoarea cheii străine `cod_departament` din relaţia `salariat` trebuie ori să fie Null, ori să coincidă cu o valoare a cheii primare la care face referinţă.
  Prin urmare, dacă atributul `cod_departament` din relaţia `salariat` ar fi avut pentru un anumit tuplu valoarea D5, valoare ce nu se regăseşte printre valorile cheii primare `cod_departament` din relaţia `departament`, atunci s-ar fi încălcat o regulă de integritate.

I. Concepte de bază - cont.

- Există posibilitatea ca **o cheie străină să facă referinţă la cheia primară a propriei relaţii**.
- Pentru a ilustra acest lucru, să adăugăm la relaţia `salariat` un nou atribut `cod_manager`, reprezentând codul superiorului ierarhic al salariatului (facem presupunerea că un angajat poate avea cel mult un superior ierarhic).

Exemplu

cod_salariat	nume	prenume	...	cod_manager
S1	Ion	Cosmin	...	
S2	Popescu	Vasile	...	S1
S3	Ionescu	Gina	...	S1
S4	Costache	Viorel	...	S2

- Deci salariatul cu codul S1 este superiorul ierarhic al salariaţilor cu codurile S2 şi S3, salariatul cu codul S2 este superiorul ierarhic al salariatului cu codul S4, iar S1 nu are un superior ierarhic.

I. Concepte de bază - cont.

➢ **Relaţiile** ce constituie o BD trebuie **să satisfacă mai multe condiţii**:

4. Navigaţia în cadrul modelului relaţional se face prin intermediul valorii pe care o ia un atribut.
5. Tuplurile pot fi prezentate utilizatorului în orice ordine. Deci acesta nu trebuie să facă nici o presupunere în privinţa ordinii tuplurilor.
6. Atributele pot fi prezentate în orice ordine, deci utilizatorul nu trebuie să facă nici o presupunere în privinţa ordinii acestora.
   Cu alte cuvinte, în modelul relaţional nu există dependenţă de secvenţă.

Exemplu

- Deşi relaţiile
  `Salariat (cod_salariat, nume, prenume, adresa, dată_naştere, sex, cod_departament)`
  şi
  `Salariat (cod_salariat, nume, prenume, cod_departament, dată_naştere, sex, adresa)`
  par diferite, ele sunt echivalente funcţional pentru că au atribute identice, chiar dacă sunt în altă ordine.

I. Concepte de bază - cont.

➢ **Relaţiile** ce constituie o BD trebuie **să satisfacă mai multe condiţii**:

7. Relaţiile pot fi manipulate pentru a furniza utilizatorului diferite vederi asupra datelor, rezultatul fiind noi relaţii.
   Cu alte cuvinte, rezultatul manipulării relaţiilor sunt noi relaţii.
   În plus, relaţiile produse ca rezultat al comenzilor limbajului de interogare a datelor satisfac toate regulile la care sunt supuse relaţiile iniţiale.

II. Constrângeri de integritate

- Pentru asigurarea integrităţii datelor, o BD trebuie să satisfacă un număr de constrângeri, numite **constrângeri de integritate**.
- Constrângerile de integritate ale modelului relaţional se pot împărţi în 2 clase:
  - **constrângeri structurale**, care trebuie satisfăcute de orice BD care foloseşte modelul relaţional şi
  - **constrângeri de comportament**, care sunt specifice fiecărei BD particulare.

II. Constrângeri de integritate - cont.

- **Constrângerile de integritate structurale** exprimă proprietăţi fundamentale, inerente modelului relaţional şi sunt în general specificate la definirea BD, ca parte a schemei BD.
- În modelul relaţional există 2 tipuri de constrângeri structurale:
  - **de entitate** şi
  - **de referinţă**.
- Ele au fost menţionate anterior, când am vorbit despre proprietăţile sistemului relaţional, fără însă a le menţiona explicit numele.

II. Constrângeri de integritate - cont.

- **Integritatea entităţii**: O cheie primară nu poate conţine atribute ce pot avea valoarea Null. În plus, prin însăşi definiţia unei chei primare, ea trebuie să fie unică şi minimală.
- **Integritatea referirii**: Valoarea unei chei străine trebuie ori să fie Null, ori să coincidă cu o valoare a cheii primare la care face referinţă.
- Aceste 2 tipuri de constrângeri pot fi impuse în Oracle prin simpla lor adăugare la definiţia tabelelor respective.

II. Constrângeri de integritate - cont.

- **Constrângerile de integritate de comportament** sunt specifice unei anumite BD şi ţin cont de semnificaţia valorilor atributelor din baza respectivă.
- De exemplu, constrângerile de domeniu restricţionează valorile unui atribut la o anumită mulţime, iar constrângerile sintactice se pot referi la tipul datelor, lungimea atributelor, etc.
- Constrângerile de comportament pot exprima legături între valorile unor atribute diferite, de exemplu valoarea unui atribut este dependentă de valoarea altui atribut sau set de atribute sau o expresie formată din valorile mai multor atribute trebuie să se încadreze în anumite limite, etc.

II. Constrângeri de integritate - cont.

- Constrângerile de comportament pot fi impuse în Oracle
  - fie prin adăugarea lor la definiţia tabelelor,
  - fie prin definirea unor secvenţe de program, numite **declanşatori (triggers)** care sunt ataşate tabelelor şi care intră în acţiune la încălcarea acestor constrângeri, împiedicând operaţiile care ar duce la încălcarea integrităţii.

III. Operatorii sistemului relaţional

- În afara relaţiilor şi a proprietăţilor acestora, modelul relaţional este definit şi prin setul de operaţii care se pot efectua asupra acestor relaţii.
- Există 2 moduri de descriere matematică a acestor operatori:
  - algebra relaţională şi
  - calculul relaţional.

III. Operatorii sistemului relaţional - cont.

- Algebra relaţională este formată dintr-o mulţime de **8 operatori,** ce acţionează asupra relaţiilor şi generează tot o relaţie.
- Operatorii algebrei relaţionale sunt
  - fie **operatorii tradiţionali pe mulţimi** (UNION, INTERSECT, DIFFERENCE, PRODUCT),
  - fie **operatori relaţionali speciali** (PROJECT, SELECT, JOIN, DIVISION).
- Cum ieşirea generată de fiecare dintre aceşti operatori este tot o relaţie, este posibilă combinarea şi compunerea lor.

III. Operatorii sistemului relaţional - cont.

- 5 dintre operatori (PROJECT, SELECT, DIFFERENCE, PRODUCT, UNION) sunt **operatorii primitivi** ai limbajului, iar ceilalţi 3 (JOIN, DIVISION, INTERSECT) sunt **operatori derivaţi**, putând fi definiţi în funcţie de primii.
- Unii dintre operatori se aplică unei singure relaţii (operatori unari), iar alţii operează asupra a două relaţii (operatori binari).

III. Operatorii sistemului relaţional - cont.

- **Calculul relaţional reprezintă o adaptare a calculului predicatelor la domeniul BDR (bazelor de date relaţionale)**. Ideea de bază este de a identifica o relaţie cu un predicat.
  Pe baza unor predicate iniţiale, prin aplicarea unor operatori ai calculului cu predicate (conjuncţia, disjuncţia, negaţia, cuantificatorul existenţial şi cel universal) se pot defini noi predicate, adică noi relaţii.
- **Algebra relaţională şi calculul relaţional sunt echivalente unul cu celălalt**, în sensul că orice relaţie care poate fi definită în algebra relaţională poate fi definită şi în calculul relaţional şi reciproc.

III. A. Algebra relaţională şi limbajul SQL

- În prezent, limbajul dominant folosit pentru interogarea BDR este SQL (Structured Query Language), care este un limbaj bazat pe operaţiile algebrei relaţionale.
- ∀ operator al algebrei relaţionale poate fi descris folosind comanda SELECT a limbajului SQL cu diverse clauze.
- În continuare vom defini operatorii algebrei relaţionale şi vom exemplifica modul de implementare a acestor operatori în SQL.
- **Comanda SELECT din limbajul SQL nu reprezintă acelaşi lucru şi nu trebuie confundată cu operatorul SELECT din algebra relaţională.**

Operatorul PROJECT (proiecția)

- Este un operator unar care are ca parametri un atribut sau mai multe atribute ale unei relaţii şi care elimină din relaţie toate celelalte atribute, producând o **submulţime „pe verticală” a relației**.
- Deoarece suprimarea unor atribute poate avea ca efect apariţia unor tupluri **duplicate**, acestea **vor fi eliminate din relaţia rezultată** deoarece, prin definiţie, o relaţie nu poate conţine tupluri cu valori identice.
- Notaţiile folosite de obicei pentru acest operator sunt $\Pi_X(R)$ şi **PROJECT(R, X)** unde R reprezintă relaţia, iar X este atributul sau mulţimea de atribute care constituie parametrii proiecţiei.

Exemplu

R

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2
x2	y2	z1

Proiecție →

C	A
z1	x1
z2	x1
z2	x1
z1	x2

Eliminare duplicate →

$\Pi_{C,A}(R)$

C	A
z1	x1
z2	x1
z1	x2

Operatorul PROJECT (proiecția) - cont.

- În SQL, proiecţia fără dubluri se obţine folosind comanda SELECT cu specificaţia DISTINCT:

```
SELECT DISTINCT C, A
FROM R;
```

- În cazul folosirii comenzii SELECT fără clauza DISTINCT se va obţine proiecţia cu dubluri:

```
SELECT C, A
FROM R;
```

Operatorul SELECT (selecția)

- Este un operator unar care este utilizat pt. extragerea tuturor tuplurilor dintr-o relaţie care satisfac o condiţie specificată, producând astfel o **submulţime „pe orizontală" a relaţiei**.
- Condiţia este o formulă logică ce poate conţine nume de atribute, constante, operatori logici (AND, NOT, OR), operatori de comparaţie (<, =, >, <=, >=, !=).
- Spre deosebire de alte lb. de programare, cum ar fi Pascal, limbajul SQL acordă operatorilor de comparaţie o prioritate mai mare decât operatorilor logici.
- Notaţiile folosite de obicei pentru acest operator sunt $\sigma_C(R)$ sau **SELECT(R, C)**, unde R reprezintă relaţia, iar C este condiţia care trebuie satisfăcută de tuplurile selectate.

Exemplu

R

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2
x2	y2	z1

Selecție

$\sigma_{A='x1' \text{ OR } B='y1'}(R)$

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2

Operatorul SELECT (selecția) - cont.

- În SQL, selecţia se obţine folosind comanda SELECT cu clauza WHERE:

```
SELECT *
FROM R
WHERE A = 'x1' OR B = 'y1';
```

- Combinarea selecţiei cu proiecţia fără dubluri se face în modul următor:

```
SELECT DISTINCT C, A
FROM R
WHERE A = 'x1' OR B = 'y1';
```

Exemplu

R

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2
x2	y2	z1

Selecție →

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2

Proiecție cu eliminare duplicate →

C	A
z1	x1
z2	x1

Operatorul SELECT (selecția) - cont.

- Combinarea selecţiei cu proiecţia cu dubluri se face în modul următor:

```
SELECT C, A
FROM R
WHERE A = 'x1' OR B = 'y1';
```

Exemplu

R

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2
x2	y2	z1

Selecție →

A	B	C
x1	y1	z1
x1	y1	z2
x1	y2	z2

Proiecție fără eliminare duplicate →

C	A
z1	x1
z2	x1
z2	x1

PRODUCT (produsul cartezian)

- Este un operator binar.
- Produsul cartezian a două relaţii R şi S este **mulţimea tuturor tuplurilo**r care se obţin prin **concatenarea unui tuplu din R cu un tuplu din S**.
- Prin urmare, dacă aritatea relaţiei R este m, iar aritatea relaţiei S este n, atunci produsul cartezian dintre R şi S va avea aritatea m + n.
- Notaţiile folosite de obicei pentru acest operator sunt **R × S**, **PRODUCT(R, S)**, **TIMES(R, S)**.

Exemplu

R

A	B	C
x1	y1	z1
x2	y1	z2
x3	y2	z1

S

D	E
z1	u1
z2	u2

R × S

A	B	C	D	E
x1	y1	z1	z1	u1
x1	y1	z1	z2	u2
x2	y1	z2	z1	u1
x2	y1	z2	z2	u2
x3	y2	z1	z1	u1
x3	y2	z1	z2	u2

PRODUCT (produsul cartezian) - cont.

- Produsul cartezian poate fi exprimat în SQL printr-o comandă SELECT pe mai multe tabele fără clauza WHERE:

```
SELECT *
FROM R, S;
```

Compatibilitate la reuniune

- Două **relaţii** R şi S se numesc **compatibile la reuniune** dacă ele conţin acelaşi număr de atribute (au aceeaşi aritate) şi atributele cu acelaşi număr de ordine din fiecare relaţie au acelaşi domeniu din care pot lua valori.
- Operatorii UNION, INTERSECT, DIFFERENCE, prezentaţi în continuare, sunt operatori binari ce nu pot fi aplicaţi decât asupra relaţiilor compatibile la reuniune.

UNION (reuniunea)

- Reuniunea a două relaţii R şi S este **mulţimea tuplurilor aparţinând fie lui R, fie lui S.**
- Reuniunea celor 2 mulţimi va cuprinde fiecare tuplu o singură dată, chiar dacă el face parte din ambele mulţimi.
- Reuniunea se poate aplica doar relaţiilor compatibile la reuniune.
- Notaţiile folosite de obicei pentru acest operator sunt **R ∪ S** sau **UNION(R, S)**.
- Reuniunea este o operaţie binară comutativă, adică R ∪ S = S ∪ R.

Exemplu

R

A	B
x1	y1
x2	y1
x3	y1

S

C	D
x1	y1
x1	y2

R ∪ S

A	B
x1	y1
x2	y1
x3	y1
x1	y2

UNION (reuniunea) - cont.

- În SQL reuniunea se poate exprima folosind operatorul UNION:

```
SELECT A, B
FROM R
UNION
SELECT C, D
FROM S;
```

DIFFERENCE (diferența)

- Diferenţa a două relaţii R şi S este **mulţimea tuplurilor care aparţin lui R, dar nu aparţin lui S**.
- Diferenţa este o operaţie binară ne-comutativă, adică R – S ≠ S – R, care se poate aplica doar relaţiilor compatibile la reuniune.
- Notaţiile folosite de obicei pentru acest operator sunt **R - S, DIFFERENCE(R, S), MINUS(R, S)**.

Exemplu

R

A	B
x1	y1
x2	y1
x3	y1

S

C	D
x1	y1
x1	y2

R - S

A	B
x2	y1
x3	y1

DIFFERENCE (diferența) - cont.

- În SQL diferența se poate exprima folosind operatorul MINUS:

```
SELECT A, B
FROM R
MINUS
SELECT C, D
FROM S;
```

DIFFERENCE (diferența) - cont.

- În plus, diferenţa poate fi simulată şi prin operatorul NOT EXISTS. De exemplu, comanda SQL de mai înainte este echivalentă cu următoarea:

```
SELECT A, B
FROM R
WHERE NOT EXISTS(SELECT *
                 FROM S
                 WHERE R.A = S.C AND R.B = S.D);
```

DIFFERENCE (diferența) - cont.

- Pentru simplitate, în comanda SQL anterioară am presupus că nici un atribut din relaţiile R sau S nu poate avea valoarea Null.
- Dacă, de exemplu, atributul A din relaţia S şi atributul C din relaţia S pot avea valoarea Null, atunci expresia `R.A = S.C` trebuie înlocuită cu `R.A = S.C OR (R.A IS NULL AND S.C IS NULL)`.
  Acest lucru se datorează faptului că în SQL, expresia NULL = NULL are valoarea False.

INTERSECT (intersecţia)

- Intersecţia a două relaţii R şi S este **mulţimea tuplurilor care aparţin atât lui R cât şi lui S**.
- Intersecţia este o operaţie binară comutativă care se poate aplica doar relaţiilor compatibile la reuniune.
- Notaţiile folosite de obicei pentru acest operator sunt **R ∩ S, INTERSECT(R, S), AND(R, S)**.
- Intersecţia este un operator derivat, putând fi exprimat cu ajutorul reuniunii şi diferenţei:
  R ∩ S = R − (R − S) sau
  R ∩ S = S − (S − R).

Exemplu

R

A	B
x1	y1
x2	y1
x3	y1

S

C	D
x1	y1
x1	y2

R ∩ S

A	B
x1	y1

INTERSECT (intersecţia) - cont.

- În SQL intersecția se poate exprima folosind operatorul INTERSECT:

```
SELECT A, B
FROM R
INTERSECT
SELECT C, D
FROM S;
```

INTERSECT (intersecţia) - cont.

- În plus, intersecția poate fi simulată şi prin operatorul EXISTS. De exemplu, comanda SQL de mai înainte este echivalentă cu următoarea:

```
SELECT A, B
FROM R
WHERE EXISTS(SELECT *
             FROM S
             WHERE R.A = S.C AND R.B = S.D);
```

Remarcă

- În cazul când operatorii UNION, DIFFERENCE sau INTERSECT se aplică unor relaţii care sunt obţinute prin selecţie din aceeaşi relaţie, atunci aceştia pot fi simulaţi prin aplicarea operatorilor logici corespunzători (OR, AND NOT, AND) asupra condiţiilor de selecţie. De exemplu, următoarele comenzi sunt echivalente:

Remarcă - cont.

```
SELECT A, B
FROM R
WHERE A = 'x1'
MINUS
SELECT A, B
FROM R
WHERE B = 'y1';

SELECT A, B
FROM R
WHERE A = 'x1' AND NOT B = 'y1';
```

DIVISION (diviziunea)

- Diviziunea este o operaţie binară care se aplică asupra a două relaţii R şi S, a.î. mulţimea atributelor lui R include mulţimea atributelor lui S.
- Dacă R este o relaţie cu aritatea m, iar S o relaţie cu aritatea n, unde m > n, atunci diviziunea lui R la S este **mulţimea tuplurilor de dimensiune m – n la care, adăugând orice tuplu din S, se obţine un tuplu din R.**
- Notaţiile utilizate cel mai frecvent sunt **R ÷ S, DIVISION(R, S), DIVIDE(R, S)**.

Exemplu

R

A	B	C
x1	y1	z1
x1	y2	z1
x1	y1	z2
x2	y1	z2
x2	y2	z1

S

C
z1
z2

R ÷ S

A	B
x1	y1

DIVISION (diviziunea) - cont.

- Diviziunea este o operaţie derivată care se exprimă cu ajutorul diferenţei, produsului cartezian şi proiecţiei:
  $R \div S = R_1 - R_2$, unde
  $R_1 = \Pi_X(R)$,
  $R_2 = \Pi_X((R_1 \times S) - R)$,
  iar X este mulţimea atributelor lui R care nu există în S.
  Pentru exemplul de mai înainte:

R_1

A	B
x1	y1
x1	y2
x2	y1
x2	y2

R_2

A	B
x2	y1
x2	y2
x1	y2

$R_1 \times S$

A	B	C
x1	y1	z1
x1	y2	z1
x2	y1	z1
x2	y2	z1
x1	y1	z2
x1	y2	z2
x2	y1	z2
x2	y2	z2

$(R_1 \times S) - R$

A	B	C
x2	y1	z1
x2	y2	z1
x1	y2	z2

$R_1 \div R_2$

A	B
x1	y1

DIVISION (diviziunea) - cont.

- Operatorul DIVISION este **legat de cuantificatorul universal (∀) care nu există în SQL, dar care poate fi simulat cu ajutorul cuantificatorului existenţial (∃)**, care există în SQL, utilizând relaţia:
  ∀ x P(x) ≡ NOT ∃ x NOT P(x).
- Pentru a ilustra exprimarea operatorul DIVISION în cele două moduri (folosind cuantificatorul universal şi cuantificatorul existenţial), să considerăm relaţiile `curs_student` şi `curs_fundamental`:

curs_student

cod_student	Curs
S1	matematica
S1	Fizica
S1	Mecanica
S2	matematica
S2	informatica
S3	Fizica
S4	matematica
S4	Fizica

curs_fundamental

curs
matematica
fizica

curs_student ÷ curs_fundamental

cod_student
S1
S4

DIVISION (diviziunea) - cont.

- Atunci relaţia `curs_student ÷ curs_fundamental` poate fi definită prin următoarea întrebare:
  care sunt studenţii care urmează toate cursurile fundamentale?
- Alternativ, această relaţie poate fi definită prin întrebarea:
- care sunt studenţii pentru care nu există curs fundamental care să nu fie urmat de către aceştia?
- Utilizând a doua formulare, rezultă că operatorul DIVISION poate fi simulat în SQL prin doi operatori NOT EXISTS:

DIVISION (diviziunea) - cont.

```
SELECT DISTINCT cod_student
FROM curs_student cs1
WHERE NOT EXISTS
    (SELECT *
     FROM curs_fundamental cf
     WHERE NOT EXISTS
        (SELECT *
         FROM curs_student cs2
         WHERE cf.curs = cs2.curs
         AND cs1.cod_student = cs2.cod_student));
```

Bibliografie

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

Întrebări & Răspunsuri

Mulțumesc!