



Baze de date

Curs 1 – Modelul relațional

Sorina Preduț

sorina.predut@my.fmi.unibuc.ro

Universitatea din București



Cuprins

- I. Concepte de bază
- II. Constrângeri de integritate
- III. Operatorii sistemului relațional
 - A. Algebra relațională și limbajul SQL
- IV. Tabele, rânduri, coloane
- V. Sisteme de Gestiune a Bazelor de Date Relaționale (SGBDR)



Modelul relațional

- reprezintă **datele** sub forma unor structuri bidimensionale, asemănătoare **tabelelor**.
- a apărut pentru a ușura **concepția, accesarea și procesarea datelor**.
- modelele de date existente anterior (**ierarhic și rețea**) erau foarte de complexe.
 - performanțele acestor tipuri de BD (baze de date) erau dependente de proiectarea și designul lor.
 - programatorul aplicației trebuia să cunoască implementarea fizică a BD și limbajul de manipulare a datelor corespunzător.



Modelul relațional – cont.

- simplitatea acestui model și a limbajului său de manipulare a datelor permite utilizatorilor să execute propriile lor manipulări.
- conceptele teoretice care formează baza modelului relațional 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 dependențele de secvență și cale**.
- nu este orientat spre sistemul de calcul; **utilizatorul poate obține date specifice folosind un limbaj ne-procedural** care îi permite să descrie datele cerute în loc de navigația folosită.



Modelul relațional – cont.

- modelul relațional și operațiile folosite de către acesta au la bază o **riguroasă fundamentare matematică**:
 - algebra relațională și calculul relațional.
- operațiile de manipulare a datelor pot fi privite ca o serie de operații de
 - **proiecție**,
 - **filtrare**,
 - **reuniune**,
 - **intersecție**, etc.
- asupra datelor din tabele.



Modelul relațional – cont.

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



I. Concepte de bază

- Definirea unui model de date presupune precizarea și identificarea elementelor:
 - **structurile de date** folosite,
 - **operatorii** care acționează asupra structurilor de date,
 - **restricțiile** care trebuie impuse pentru menținerea corectitudinii datelor, numite și **restricții de integritate**.



I. Concepte de bază

- În modelul relațional datele sunt reprezentate ca structuri bidimensionale, numite **relații**.
- O relație este alcătuită dintr-un număr fix de elemente numite **atribute**, fiecare dintre acestea putând lua valori într-o mulțime finită, numită **domeniu**.
- Numărul de atribute al unei relații se numește **aritatea** relației.
- **relație (atribut_1, atribut_2, ..., atribut_n)**
- De exemplu, datele despre angajații unei firme se pot reprezenta ca o relație de aritate 7 astfel:
`salariat (cod_salariat, nume, prenume, adresa, dată_naștere, sex, cod_departament)`



I. Concepte de bază - cont.

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

cod_salariat	nume	prenume	Adresa	dată_naștere	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. Crișana Nr. 11	12/02/1975	M	D1



I. Concepte de bază - cont.

- **Relațiile** ce constituie o BD trebuie să satisfacă mai multe condiții:
 1. Fiecare atribut trebuie să poarte un nume, care este unic în cadrul relației.
 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 relație \exists întotdeauna unul sau mai multe attribute care asigură că **tuplul** va rămâne **unic tot timpul**.
- Un atribut sau set de attribute care identifică în mod unic un tuplu se numește **cheie candidată** sau mai simplu **cheie**.
- Pt. \forall relație se alege din mulțimea cheilor candidate o cheie care s.n. **cheie primară** a relației.
- 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, attributele care fac parte din cheie sunt alese astfel încât nici unul să nu fie în plus, adică nici un atribut nu poate fi șters 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 să fie minimală**.



I. Concepte de bază - cont.

- Uneori, în unele tupluri dintr-o relație, un **atribut poate fi necunoscut sau neaplicabil**. Pentru a reprezenta un astfel de atribut se folosește o valoare convențională, notată cu **Null**.
- Ca regulă generală, nici unul dintre attributele care alcătuiesc cheia primară nu poate avea valoarea Null pentru nici unul din tuplurile relației.



Exemplu

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



Exemplu - cont.

- În alegerea cheilor candidate s-a ținut cont de **principiul minimalității** în sensul că din cheia compusă `nume`, `prenume` și `dată_naștere` nu poate fi șters nici un atribut și nici nu mai trebuie adăugat altul, aceste trei attribute identificând în mod unic tuplurile.
De exemplu, combinația (`nume`, `prenume`, `dată_naștere`, `cod_departament`) nu este o cheie a relației `salariat` pentru că și în absența atributului `cod_departament`, un `salariat` este identificat în mod unic de primele trei attribute.



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 și nici nu poate intra în componența unei chei compuse care este și cheie primară.



I. Concepte de bază - cont.

- S.n. **cheie străină** un atribut sau o mulțime de attribute ale unei relații R1 care există și într-o altă relație R2, nu neapărat distinctă de R1, și care formează cheia primară a relației R2.

În cazul acesta, **cheia străină din R1** se spune că **face referință 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 referință.



Exemplu

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

<code>cod_departament</code>	<code>denumire</code>	<code>localitate</code>
D1	Analiză financiară	București
D2	Depozit	Pitești
D3	Contabilitate	Constanța
D4	Magazin	Pitești



Exemplu - cont.

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



I. Concepte de bază - cont.

- Există posibilitatea ca o cheie străină să facă referință la cheia primară a propriei relații.
- Pentru a ilustra acest lucru, să adăugăm la relația `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 salariaților cu codurile S2 și 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.

- **Relațiile** ce constituie o BD trebuie **să satisfacă mai multe condiții**:
 4. Navigația în cadrul modelului relațional 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 privința ordinii tuplurilor.
 6. Atributele pot fi prezentate în orice ordine, deci utilizatorul nu trebuie să facă nici o presupunere în privința ordinii acestora.

Cu alte cuvinte, în modelul relațional nu există dependență de secvență.



Exemplu

➤ Deși relațiile

```
Salariat (cod_salariat, nume, prenume, adresa,  
data_nastere, sex, cod_departament)
```

și

```
Salariat (cod_salariat, nume, prenume, cod_departament,  
data_nastere, sex, adresa)
```

par diferite, ele sunt echivalente funcțional pentru că au attribute identice, chiar dacă sunt în altă ordine.



I. Concepte de bază - cont.

- **Relațiile** ce constituie o BD trebuie **să satisfacă mai multe condiții**:
 7. Relațiile pot fi manipulate pentru a furniza utilizatorului diferite vederi asupra datelor, rezultatul fiind noi relații.

Cu alte cuvinte, rezultatul manipulării relațiilor sunt noi relații.

În plus, relațiile produse ca rezultat al comenzilor limbajului de interogare a datelor satisfac toate regulile la care sunt supuse relațiile inițiale.



II. Constrângeri de integritate

- Pentru asigurarea integrității datelor, o BD trebuie să satisfacă un număr de constrângeri, numite **constrângeri de integritate**.
- Constrângerile de integritate ale modelului relațional se pot împărți în 2 clase:
 - **constrângeri structurale**, care trebuie satisfăcute de orice BD care folosește modelul relațional și
 - **constrângeri de comportament**, care sunt specifice fiecărei BD particulare.



II. Constrângeri de integritate - cont.

- **Constrângerile de integritate structurale** exprimă proprietăți fundamentale, inerente modelului relațional și sunt în general specificate la definirea BD, ca parte a schemei BD.
- În modelul relațional există 2 tipuri de constrângeri structurale:
 - **de entitate** și
 - **de referință.**
- Ele au fost menționate anterior, când am vorbit despre proprietățile sistemului relațional, fără însă a le menționa explicit numele.



II. Constrângeri de integritate - cont.

- **Integritatea entității:** O cheie primară nu poate conține attribute ce pot avea valoarea Null. În plus, prin însăși definiția unei chei primare, ea trebuie să fie unică și minimală.
- **Integritatea referirii:** Valoarea unei chei străine trebuie ori să fie ori Null, ori să coincidă cu o valoare a cheii primare la care face referință.
- Aceste 2 tipuri de constrângeri pot fi impuse în Oracle prin simpla lor adăugare la definiția tabelelor respective.



II. Constrângeri de integritate - cont.

- **Constrângerile de integritate de comportament** sunt specifice unei anumite BD și țin cont de semnificația valorilor atributelor din baza respectivă.
- De exemplu, constrângerile de domeniu restricționează valorile unui atribut la o anumită mulțime, iar constrângerile sintactice se pot referi la tipul datelor, lungimea atributelor, etc.
- Constrângerile de comportament pot exprima legături între valorile unor attribute diferite, de exemplu valoarea unui atribut este dependentă de valoarea altui atribut sau set de attribute sau o expresie formată din valorile mai multor attribute 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 definiția tabelelor,
 - fie prin definirea unor secvențe de program, numite **declanșatori (triggers)** care sunt atașate tabelelor și care intră în acțiune la încălcarea acestor constrângeri, împiedicând operațiile care ar duce la încălcarea integrității.



III. Operatorii sistemului relațional

- În afara relațiilor și a proprietăților acestora, modelul relațional este definit și prin setul de operații care se pot efectua asupra acestor relații.
- Există 2 moduri de descriere matematică a acestor operatori:
 - algebra relațională și
 - calculul relațional.



III. Operatorii sistemului relațional - cont.

- Algebra relațională este formată dintr-o mulțime de **8 operatori**, ce acționează asupra relațiilor și generează tot o relație.
- Operatorii algebrei relaționale sunt
 - fie **operatorii tradiționali pe mulțimi** (UNION, INTERSECT, DIFFERENCE, PRODUCT),
 - fie **operatori relaționali speciali** (PROJECT, SELECT, JOIN, DIVISION).
- Cum ieșirea generată de fiecare dintre acești operatori este tot o relație, este posibilă combinarea și compunerea lor.



III. Operatorii sistemului relațional - cont.

- 5 dintre operatori (PROJECT, SELECT, DIFFERENCE, PRODUCT, UNION) sunt **operatorii primitivi** ai limbajului, iar ceilalți 3 (JOIN, DIVISION, INTERSECT) sunt **operatori derivați**, putând fi definiți în funcție de primii.
- Unii dintre operatori se aplică unei singure relații (operatori unari), iar alții operează asupra a două relații (operatori binari).



III. Operatorii sistemului relațional - cont.

- **Calculul relațional reprezintă o adaptare a calculului predicatelor la domeniul BDR (bazelor de date relaționale).** Ideea de bază este de a identifica o relație cu un predicat.
Pe baza unor predicate inițiale, prin aplicarea unor operatori ai calculului cu predicate (conjuncția, disjuncția, negația, cuantificatorul existențial și cel universal) se pot defini noi predicate, adică noi relații.
- **Algebra relațională și calculul relațional sunt echivalente unul cu celălalt, în sensul că orice relație care poate fi definită în algebra relațională poate fi definită și în calculul relațional și reciproc.**



III. A. Algebra relațională și limbajul SQL

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



Operatorul PROJECT (proiecția)

- Este un operator unar care are ca parametri un atribut sau mai multe attribute ale unei relații și care elimină din relație toate celelalte attribute, producând o **submulțime „pe verticală” a relației**.
- Deoarece suprimarea unor attribute poate avea ca efect apariția unor tupluri **duplicate**, acestea **vor fi eliminate din relația rezultată** deoarece, prin definiție, o relație nu poate conține tupluri cu valori identice.
- Notățiile folosite de obicei pentru acest operator sunt $\Pi_X(R)$ și **PROJECT(R, X)** unde R reprezintă relația, iar X este atributul sau mulțimea de attribute care constituie parametrii proiecției.

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, proiecția fără dubluri se obține folosind comanda SELECT cu specificația DISTINCT:

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

- În cazul folosirii comenzii SELECT fără clauza DISTINCT se va obține proiecția cu dubluri:

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



Operatorul SELECT (selecția)

- Este un operator unar care este utilizat pt. extragerea tuturor tuplurilor dintr-o relație care satisfac o condiție specificată, producând astfel o **submulțime „pe orizontală” a relației**.
- Condiția este o formulă logică ce poate conține nume de attribute, constante, operatori logici (AND, NOT, OR), operatori de comparație (<, =, >, <=, >=, !=).
- Spre deosebire de alte lb. de programare, cum ar fi Pascal, limbajul SQL acordă operatorilor de comparație o prioritate mai mare decât operatorilor logici.
- Notățiile folosite de obicei pentru acest operator sunt $\sigma_C(R)$ sau **SELECT(R, C)**, unde R reprezintă relația, iar C este condiția care trebuie satisfăcută de tuplurile selectate.

Exemplu

R

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

Selecție

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

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



Operatorul SELECT (selecția) - cont.

- În SQL, selecția se obține folosind comanda SELECT cu clauza WHERE:

```
SELECT *  
FROM R  
WHERE A = 'x1' OR B = 'y1';
```
- Combinarea selecției cu proiecția 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 selecției cu proiecția 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 relații R și S este **mulțimea tuturor tuplurilor** care se obțin prin **concatenarea unui tuplu din R cu un tuplu din S**.
- Prin urmare, dacă aritatea relației R este m, iar aritatea relației S este n, atunci produsul cartezian dintre R și S va avea aritatea $m + n$.
- Notățiile folosite de obicei pentru acest operator sunt $R \times S$, **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

$R \times S$

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 relații R și S se numesc **compatibile la reuniune** dacă ele conțin același număr de atribute (au aceeași aritate) și attributele cu același număr de ordine din fiecare relație au același domeniu din care pot lua valori.
- Operatorii UNION, INTERSECT, DIFFERENCE, prezentați în continuare, sunt operatori binari ce nu pot fi aplicați decât asupra relațiilor compatibile la reuniune.



UNION (reuniunea)

- Reuniunea a două relații R și S este **mulțimea tuplurilor aparținând fie lui R , fie lui S .**
- Reuniunea celor 2 mulțimi va cuprinde fiecare tuplu o singură dată, chiar dacă el face parte din amândouă mulțimile.
- Reuniunea se poate aplica doar relațiilor compatibile la reuniune.
- Notățiile folosite de obicei pentru acest operator sunt **$R \cup S$** sau **$\text{UNION}(R, S)$.**
- Reuniunea este o operație binară comutativă, adică $R \cup S = S \cup R$.



Exemplu

R

A	B
x1	y1
x2	y1
x3	y1

S

C	D
x1	y1
x1	y2

R U 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)

- Diferența a două relații R și S este mulțimea tuplurilor care aparțin lui R , dar nu aparțin lui S .
- Diferența este o operație binară ne-comutativă, adică $R - S \neq S - R$, care se poate aplica doar relațiilor compatibile la reuniune.
- Notățiile 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, diferența poate fi simulată și 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 relațiile R sau S nu poate avea valoarea Null.
- Dacă, de exemplu, atributul A din relația S și atributul C din relația S pot avea valoarea Null, atunci expresia $R.A = S.C$ trebuie înlocuită cu $R.A = S.C \text{ OR } (R.A \text{ IS NULL AND } S.C \text{ IS NULL})$.

Acest lucru se datorează faptului că în SQL, expresia $\text{NULL} = \text{NULL}$ are valoarea False.



INTERSECT (intersecția)

- Intersecția a două relații R și S este **mulțimea tuplurilor care aparțin atât lui R cât și lui S .**
- Intersecția este o operație binară comutativă care se poate aplica doar relațiilor compatibile la reuniune.
- Notățiile folosite de obicei pentru acest operator sunt $R \cap S$, **INTERSECT(R, S)**, **AND(R, S)**.
- Intersecția este un operator derivat, putând fi exprimat cu ajutorul reuniunii și diferenței:
 $R \cap S = R - (R - S)$ sau
 $R \cap S = S - (S - R)$.



Exemplu

R

A	B
x1	y1
x2	y1
x3	y1

S

C	D
x1	y1
x1	y2

$R \cap S$

A	B
x1	y1



INTERSECT (intersecția) - cont.

- În SQL intersecția se poate exprima folosind operatorul INTERSECT:

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



INTERSECT (intersecția) - cont.

- În plus, intersecția poate fi simulată și 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 relații care sunt obținute prin selecție din aceeași relație, atunci aceștia pot fi simulați prin aplicarea operatorilor logici corespunzători (OR, AND NOT, AND) asupra condițiilor de selecție. 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 operație binară care se aplică asupra a două relații R și S , a.î. mulțimea atributelor lui R include mulțimea atributelor lui S .
- Dacă R este o relație cu aritatea m , iar S o relație cu aritatea n , unde $m > n$, atunci diviziunea lui R la S este **mulțimea tuplurilor de dimensiune $m - n$ la care, adăugând orice tuplu din S , se obține un tuplu din R .**
- Notățiile utilizate cel mai frecvent sunt $R \div S$, **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

$R \div S$

A	B
x1	y1



DIVISION (diviziunea) - cont.

- Diviziunea este o operație derivată care se exprimă cu ajutorul diferenței, produsului cartezian și proiecției:

$$R \div S = R_1 - R_2, \text{ unde}$$

$$R_1 = \Pi_X(R),$$

$$R_2 = \Pi_X((R_1 \times S) - R),$$

iar X este mulțimea 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 (\forall) care nu există în SQL, dar care poate fi simulat cu ajutorul cuantificatorului existențial (\exists), care există în SQL, utilizând relația:
$$\forall x P(x) \equiv \text{NOT } \exists x \text{ NOT } P(x).$$
- Pentru a ilustra exprimarea operatorul DIVISION în cele două moduri (folosind cuantificatorul universal și cuantificatorul existențial), să considerăm relațiile `curs_student` și `curs_fundamental`:



Bibliografie

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



Întrebări & Răspunsuri

Mulțumesc!

Mulțumesc!