

SQL Partea a doua

Interogări cu mai multe Relații

Grupare/Agregare

Operatori „set” (UNION/INTERSECT/EXCEPT)

Adăugare/Modificare/Ștergere

Interogări asupra mai multor Relații

- Interogările combină adesea datele din mai multe relații.
- Se pot menționa mai multe relații într-o interogare prin introducerea lor în clauza FROM.
- Se face distincție între atributele cu același nume prin prefixare cu numele relației: "<relație>.<atribut>" .

Exemplu: Reunirea a două Relații

- Se folosesc relațiile `Likes(drinker, beer)` și `Frequents(drinker, bar)`, pentru a găsi băuturile preferate de cel puțin o persoană ce frecventează "Joe's Bar".

```
SELECT beer
FROM Likes, Frequents
WHERE bar = 'Joe's Bar' AND
      Frequents.drinker =
        Likes.drinker;
```

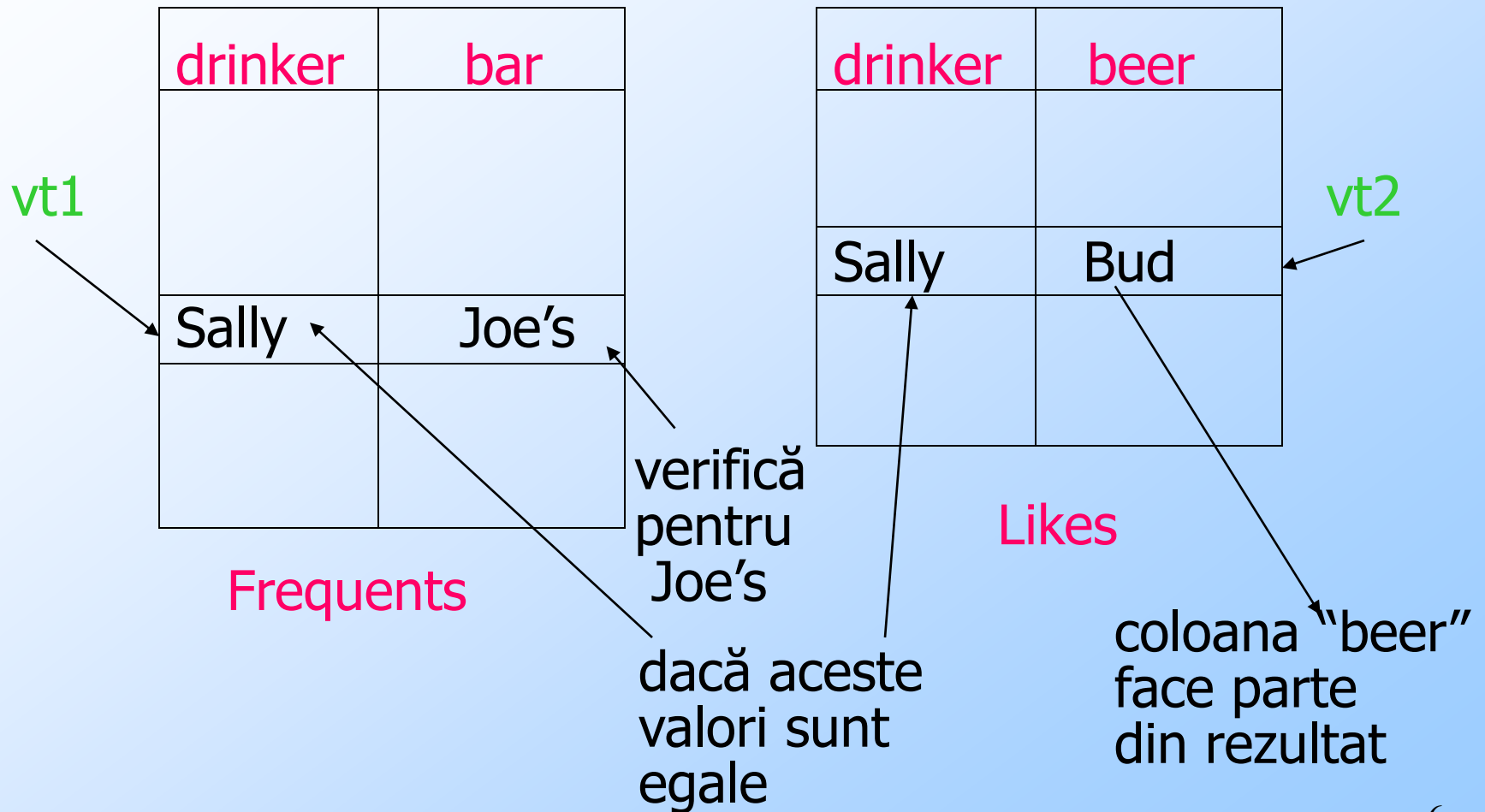
Semanticile Formale pentru o Interogare cu mai multe relații

- Sunt aproximativ aceleași ca și în cazul interogării cu o singură relație:
 1. Se începe cu produsul relațiilor din clauza FROM.
 2. Se aplică selecția impusă de condiția din clauza WHERE.
 3. Se aplică proiecția extinsă pe lista de attribute și expresii din clauza SELECT.

Semanticile Operaționale pentru o Interogare cu mai multe Relații

- Să ne imaginăm câte o variabilă de tuplă pentru fiecare relație din clauza FROM.
 - Aceste variabile de tuplă parcurg fiecare combinație de tuple, câte o tuplă din fiecare relație.
- Dacă variabilele de tuplă indică tuple ce satisfac condiția din clauza WHERE, aceste tuple fac parte din rezultat.

Exemplu



Variabile de Tuplă Explicite

- Uneori, o interogare necesită folosirea a două copii a aceleiași relații.
- Se face distincție între cele două copii prin completarea după numele relației a numelui unei variabile de tuplă, în clauza FROM.
- Oricând se poate proceda astfel, prin redenumirea relațiilor, chiar dacă nu este o condiție obligatorie.

Revenim la întrebarea din cursul introductiv:

```
SELECT a  
FROM R, S  
WHERE R.b = S.b;
```

prin ce diferă de:

```
SELECT a  
FROM R  
WHERE b IN (SELECT b FROM S);
```


Cu ajutorul IN se construiește un predicat pentru tuplele din R

```
SELECT a  
FROM R  
WHERE b IN
```

```
(SELECT b FROM S);
```

Valoarea 2 apare de două ori

O singură
parcursere a
tupelor din R

a	b
1	2
3	4

R


b	c
2	5
2	6

S

(1,2) satisface
condiția;
1 este afișat
o singură
dată.

Tuplele din R și S fac pereche prin această interogare

```
SELECT a
FROM R, S
WHERE R.b = S.b;
```



Ciclu dublu:
primul peste
tuplele din R și
al doilea peste
tuplele din S

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) cu (2,5) și
(1,2) cu (2,6),
ambele satisfac
condiția;
1 este afișat de
două ori.

Exemplu: Self-Join

- Folosind **Beers(name, manf)**, să se găsească toate perechile de bere produse de aceeași fabrică.
 - Nu interesează perechi cum ar fi (Bud, Bud).
 - Perechile vor fi obținute în ordine alfabetică, de ex. (Bud, Miller), nu (Miller, Bud).

```
SELECT b1.name, b2.name
FROM Beers b1, Beers b2
WHERE b1.manf = b2.manf AND
      b1.name < b2.name;
```

Expresii "Join"

- SQL mai multe variante de reuniri ("bag").
- Aceste expresii pot fi interogări de sine stătătoare (stand-alone queries) sau sunt folosite în locul relațiilor în clauza FROM.

Produs și Natural Join

- Natural join:

`R NATURAL JOIN S;`

- Produs:

`R CROSS JOIN S;`

- Exemplu:

`Likes NATURAL JOIN Sells;`

- Relațiile pot fi la fel de bine interogări imbricate între paranteze.

Theta Join

□ $R \text{ JOIN } S \text{ ON } \langle \text{condiție} \rangle$

□ Exemplu: Drinkers(name, addr) și
Frequents(drinker, bar):

```
Drinkers JOIN Frequents ON  
    name = drinker;
```

generează toate cvadruplele (d, a, d, b)
a.î. persoana d locuiește la adresa a și
frecventează barul b .

Outerjoin

- R OUTER JOIN S este elementul central al unei expresii outerjoin. Expresia este modificată de:

1. Opțional NATURAL înaintea lui OUTER.
2. Opțional ON <condiție> după JOIN.
3. Opțional LEFT, RIGHT, sau FULL înainte de OUTER.

Doar una
din acestea
două

- LEFT = se completează tuplele din R care nu au echivalent în S cu atâtea NULL-uri câte attribute are S.
- RIGHT = se completează tuplele din S care nu au echivalent în R cu atâtea NULL-uri câte attribute are R.
- FULL = se completează tuplele din R care nu au echivalent în S cu atâtea NULL-uri câte attribute are S și se completează tuplele din S care nu au echivalent în R cu atâtea NULL-uri câte attribute are R.
 - Este varianta implicită.

Exemplu: Outerjoin

R =

A	B
1	2
4	5

S =

B	C
2	3
6	7

(1,2) se cuplează cu (2,3), dar celelalte două tuple sunt singulare (nu au pereche).

R OUTERJOIN S =

A	B	C
1	2	3
4	5	NULL
NULL	6	7

Agregare

- Funcțiile SUM, AVG, COUNT, MIN și MAX pot fi aplicate unei coloane în clauza SELECT pentru a obține un anumit nivel de agregare al coloanei.
- De asemenea există COUNT(*) ce numără tuplele.

Exemplu: Agregare

- Se foloseşte **Sells(bar, beer, price)** pentru a găsi preţul mediu de vânzare pentru berea "Bud":

```
SELECT AVG(price)
FROM Sells
WHERE beer = 'Bud';
```

Eliminarea Duplicatelor într-o Agregare

- Se folosește DISTINCT pentru o agregare.
- **Exemplu:** să se găsească numărul prețurilor *diferite* de vânzare pentru berea "Bud":

```
SELECT COUNT(DISTINCT price)
FROM Sells
WHERE beer = 'Bud' ;
```

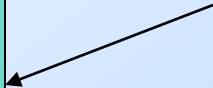
Valorile NULL sunt Ignorate într-o Agregare

- NULL nu contribuie la Sum, Avg sau Count și nu poate fi nici valoarea minimă (Min) nici valoarea maximă (Max) a unei coloane.
- Dar dacă nu există valori non-NULL pentru o coloană, atunci rezultatul agregării este NULL.
 - **Excepție:** COUNT pentru o mulțime vidă este 0.

Exemplu: Efectul NULL


```
SELECT count(*)  
FROM Sells  
WHERE beer = 'Bud';
```

Numărul barurilor
ce vând berea "Bud".



```
SELECT count(price)  
FROM Sells  
WHERE beer = 'Bud';
```

Numărul barurilor
ce vând berea "Bud"
la un preț cunoscut.



Gruparea valorilor de date

- Într-o instrucțiune SELECT-FROM-WHERE se poate folosi clauza GROUP BY cu o listă de attribute (la sfârșit).
- Tuplele din relația rezultat pentru blocul SELECT-FROM-WHERE sunt grupate conform valorilor atributelor prezente în clauza GROUP BY și se poate aplica orice agregare pe fiecare grup în parte (și numai pe grupurile formate).

Exemplu: Gruparea valorilor de date

- Se folosește **Sells(bar, beer, price)** pentru a găsi fiecare bere vândută și prețul mediu de vânzare:

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer;
```

beer	AVG(price)
Bud	2.33
...	...

Exemplu: Gruparea valorilor de date

- Se folosește **Sells(bar, beer, price)** și **Frequents(drinker, bar)** pentru a găsi prețul mediu dat pe berea "Bud" de fiecare persoană ce frecventează baruri:

```
SELECT drinker, AVG(price)
FROM Frequents, Sells
WHERE beer = 'Bud' AND
      Frequents.bar = Sells.bar
GROUP BY drinker;
```

Pasul doi:
grupează
după
drinker

Primul pas:
compune
triplete
<drinker-bar-
price>
pentru "Bud"

Restricții pentru Listele SELECT în contextul Agregării

- Dacă se folosește agregarea, atunci fiecare element din lista clauzei SELECT trebuie să fie:
 1. O Valoare Agregată, sau
 2. Un atribut al listei clauzei GROUP BY.

Exemplu de Interogare Ilegală

- Să se găsească barul ce vinde berea "Bud" la prețul cel mai ieftin:

```
SELECT bar, MIN(price)  
FROM Sells  
WHERE beer = 'Bud';
```

- Această interogare este ilegală în SQL.

Clauza HAVING

- HAVING <condiție> poate fi utilizată după clauza GROUP BY.
- Dacă apare această clauză, condiția se aplică fiecărui grup și grupurile ce nu respectă condiția sunt eliminate din rezultat.

Exemplu: HAVING

- Se folosește `Sells(bar, beer, price)` și `Beers(name, manf)` pentru a găsi prețul mediu al acelor beri ce fie sunt servite în cel puțin trei baruri, fie sunt fabricate de "Pete's".

Soluția

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer
```

```
HAVING COUNT(bar) >= 3 OR
```

```
beer IN (SELECT name
FROM Beers
WHERE manf = 'Pete's');
```

Grupurile de bere cu cel puțin 3 baruri non-NULL și de asemenea grupurile de bere au producătorul "Pete's".

Berile produse de "Pete's".

Cerințe pentru Condițiile HAVING

- Într-o interogare imbricată ("subquery") se poate folosi orice.
- În exteriorul interogărilor imbricate, condițiile pot face referire la attribute doar în următoarele circumstanțe:
 1. Un atribut proprietate de grup, sau
 2. O valoare agregată(aceeași condiție ca și pentru clauza SELECT ce conține agregare).

Operatori „set”

Uniunea, Intersecția și Diferența

- Uniunea, intersecția și diferența relațiilor sunt exprimate prin formele următoare, fiecare implică interogări imbricate:
 - (`<subquery>`) UNION (`<subquery>`)
 - (`<subquery>`) INTERSECT (`<subquery>`)
 - (`<subquery>`) EXCEPT (`<subquery>`)

Exemplu: INTERSECT

- Se folosesc Likes(drinker, beer), Sells(bar, beer, price) și Frequents(drinker, bar) pentru a găsi persoanele și mărcile de bere astfel încât:
 1. Persoana preferă berea și
 2. Persoana frecventează cel puțin un bar unde se vinde berea.

De notat trucul:
interogarea
îmbricată este
o tabelă de bază.

Soluția

(SELECT * FROM Likes)

INTERSECT

(SELECT drinker, beer
FROM Sells, Frequents
WHERE Frequents.bar = Sells.bar
);

Persoana frecventează
un bar ce vinde berea.

Semanticile "Bag"

- Deși instrucțiunea SELECT-FROM-WHERE folosește semantici "bag", semanticile implicite pentru uniune, intersecție și diferență sunt "set".
- Aceasta înseamnă că duplicatele sunt eliminate implicit la aplicarea unuia din cei trei operatori (UNION, INTERSECT, EXCEPT).

Motivație: Eficiență

- Atunci când se aplică proiecția extinsă, este mai ușor să se evite eliminarea duplicatelor.
 - Deoarece se poate lucra cu o singură tuplă la un moment dat.
- Pentru intersecție sau diferență, este mai eficient să se sorteze mai întâi relațiile.
 - La acel moment dat se poate la fel de bine să se elimine duplicatele.

Eliminarea explicită a Duplicatelor

- Pentru a forța rezultatul unei interogări să fie "set":

SELECT DISTINCT . . .

- Pentru a forța rezultatul unei interogări să fie "bag" (adică să nu fie eliminate duplicatele):

se folosește ALL,

de exemplu . . . UNION ALL . . .

Exemplu: DISTINCT

- Se folosește `Sells(bar, beer, price)`, pentru a găsi toate prețurile diferite pentru mărcile de bere vândute:

```
SELECT DISTINCT price  
FROM Sells;
```

- De notat că fără `DISTINCT`, fiecare preț poate să apară de mai multe ori, multiplicat după numărul de baruri/mărci de bere.

Exemplu: ALL

Interogarea:

```
(SELECT drinker FROM Frequents)  
EXCEPT ALL
```

```
(SELECT drinker FROM Likes);
```

folosește relațiile **Frequents(drinker, bar)** și **Likes(drinker, beer)** pentru a afișa persoanele pentru care numărul de baruri frecventate este mai mare decât numărul de mărci de bere preferate și o persoană apare de atâtea ori în rezultat cât reprezintă diferența acestor numere.

Actualizarea BD

- O comandă de *actualizare* nu returnează un rezultat (așa cum face o interogare), ci modifică baza de date într-un anumit fel.
- Există trei operații de actualizare:
 1. *Insert* (Adaugă) una sau mai multe tuple.
 2. *Delete* (Șterge) una sau mai multe tuple.
 3. *Update* (Modifică) valoarea(-ile) uneia sau mai multor tuple existente .

Adăugare

- Pentru a adăuga o singură tuplă:

```
INSERT INTO <relație>
```

```
VALUES ( <listă de valori> );
```

- **Exemplu:** adaugă în **Likes(drinker, beer)** faptul că lui "Sally" îi place "Bud".

```
INSERT INTO Likes
```

```
VALUES ( 'Sally' , 'Bud' );
```


Specificarea Atributelor în INSERT

- Se poate adăuga la numele relației o listă de attribute.
- Există două motive:
 1. Nu mai ținem minte ordinea în care am definit attributele relației.
 2. Nu cunoaștem valorile pentru toate attributele și dorim ca sistemul să completeze componentele lipsă cu valori NULL sau valori "default" (implicite).

Exemplu: Specificare Atribute

- O altă cale pentru a adăuga faptul că lui "Sally" îi place "Bud" în Likes(drinker, beer):

```
INSERT INTO Likes (beer, drinker)  
VALUES ('Bud', 'Sally');
```

Adăugare Valori "Default"

- La instrucțiunea CREATE TABLE, se poate menționa pentru un atribut o valoare DEFAULT.
- Atunci când tupla adăugată nu are specificată valoare pentru acel atribut, va fi utilizată valoarea default.

Exemplu: Valori "Default"

```
CREATE TABLE Drinkers (  
    name CHAR(30) PRIMARY KEY,  
    addr CHAR(50)  
        DEFAULT '123 Sesame St.',  
    phone CHAR(16)  
);
```

Exemplu: Valori "Default"

```
INSERT INTO Drinkers (name)
VALUES ('Sally');
```

Tupla rezultată :

name	address	phone
Sally	123 Sesame St	NULL

Adăugarea mai multor Tuple

- Se poate adăuga întregul rezultat al unei interogări într-o relație, utilizând forma:

```
INSERT INTO <relație>  
( <subquery> );
```

Exemplu: Adăugarea cu “Subquery”

- Se folosește `Frequents(drinker, bar)`, pentru a introduce într-o relație nouă `PotBuddies(name)` toți prietenii potențiali ai lui “Sally” (în engleză “potential buddies”), adică acele persoane ce frecventează cel puțin un bar frecventat de “Sally”.

Altă
persoană

Soluția

Perechi de tuple cu persoane în care prima tuplă corespunde lui "Sally", a doua tuplă corespunde la altcineva, astfel încât barurile sunt identice.

INSERT INTO PotBuddies

(SELECT d2.drinker

FROM Frequents d1, Frequents d2
WHERE d1.drinker = 'Sally' AND
d2.drinker <> 'Sally' AND
d1.bar = d2.bar

);

Ștergere

- Pentru a șterge tuple, ce satisfac o condiție, dintr-o anumită relație:

```
DELETE FROM <relație>  
WHERE <condiție>;
```

Exemplu: Ștergere

- Șterge din `Likes(drinker, beer)` faptul că lui "Sally" îi place "Bud":

```
DELETE FROM Likes  
WHERE drinker = 'Sally' AND  
      beer = 'Bud';
```

Exemplu: Șterge toate Tuplele

- Golește relația Likes:

```
DELETE FROM Likes;
```

- De notat lipsa clauzei WHERE.

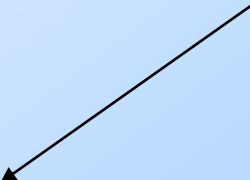
Exemplu: Șterge anumite Tuple

- Șterge din **Beers(name, manf)** toate berile pentru care există o altă bere a aceluiași producător.

```
DELETE FROM Beers b  
WHERE EXISTS (
```

```
SELECT name FROM Beers  
WHERE manf = b.manf AND  
name <> b.name);
```

Berile aceluiași producător și cu nume diferit față de numele berii reprezentate de tupla b.



Semanticile DELETE --- (1)

- Să presupunem că "Anheuser-Busch" fabrică doar "Bud" și "Bud Lite".
- Să presupunem că operația de ștergere ajunge la tupla b prima dată pentru "Bud".
- Interogarea imbricată (subquery) conține date, deoarece există berea "Bud Lite", ceea ce determină eliminarea berii "Bud".
- Atunci când operația de ștergere ajunge la tupla b pentru "Bud Lite", întrebarea este dacă se șterge această tuplă de asemenea?

Semanticile DELETE --- (2)

- Răspunsul la întrebarea de mai sus: se *șterge* "Bud Lite".
- Motivul este acela că ștergerea acționează în două etape:
 1. Se marchează toate tuplele pentru care condiția WHERE este satisfăcută.
 2. Se elimină tuplele marcate pentru ștergere.

UPDATE

- Pentru a modifica valorile anumitor attribute din anumite tuple ale unei relații:

UPDATE <relație>

SET <listă cu atribuiri de attribute>

WHERE <condiție asupra tuplelor>;

Exemplu: UPDATE

- Modifică numărul de telefon al lui "Fred" (persoană în tabela **Drinkers**) la valoarea 555-1212:

```
UPDATE Drinkers  
SET phone = '555-1212'  
WHERE name = 'Fred';
```


Exemplu: UPDATE pentru mai multe Tuple

- Modifică prețul maxim la bere la valoarea \$4:

```
UPDATE Sells  
SET price = 4.00  
WHERE price > 4.00;
```