

Introducere în SQL

Definirea Schemei

Clauzele Select-From-Where

Sortare

Interogări imbricate

SQL

- Este un limbaj “very-high-level”.
 - Spune CE trebuie făcut și nu CUM trebuie făcut.
 - Evită detaliile legate de lucrul cu datele, caracteristice limbajelor procedurale (C++ sau Java).
- SGBD-ul se îngrijește de modul de execuție a interogării.
 - Această funcțiune se numește “query optimization.”

O Relație este o Tabelă

Atribute
(cap de
tabel)

Tuple
(rânduri)

name	mănf
Winterbrew	Pete's
Bud Lite	Anheuser-Busch

Numele
Relației

Beers

Scheme

- *Schema Relației* = numele relației și lista de atribute.
 - Opțional: tipul atributelor.
 - Exemplu: *Beers(name, manf)* sau *Beers(name: string, manf: string)*
- *BD* = colecție de relații.
- *Schema BD* = setul tuturor schemelor relațiilor din baza de date (BD).

Exemplu de BD

Beers(name, manf)

Bars(name, addr, license)

Drinkers(name, addr, phone)

Likes(drinker, beer)

Sells(bar, beer, price)

Frequents(drinker, bar)

□ Ceea ce este subliniat = "*key*" (tuplele nu pot avea aceeași valoare în toate attributele ce compun cheia relației).

□ Este un exemplu de constrângere.

Definirea schemei BD în SQL

- SQL este standardul de facto pentru modelul relațional.
- Este în primul rând un limbaj de interogare, pentru regăsirea informației din BD (LMD).
- Include de asemenea o componentă "*data-definition*" pentru descrierea schemei BD (LDD).

Crearea (Declararea) unei Relații

- Forma cea mai simplă:

```
CREATE TABLE <nume> (  
    <listă de elemente>  
);
```

- Pentru a elimina o relație:

```
DROP TABLE <nume>;
```

Elementele declarației “CREATE Table”

- Cel mai important element: un atribut (coloană) și tipul său.
- Tipurile obișnuite sunt:
 - INT sau INTEGER (sinonime).
 - REAL sau FLOAT (sinonime).
 - CHAR(n) = șir de caractere de lungime fixă, n caractere.
 - VARCHAR(n) = șir de caractere de lungime variabilă, maxim n caractere.

Exemplu: Create Table

```
CREATE TABLE Sells (  
    bar      CHAR(20) ,  
    beer     VARCHAR(20) ,  
    price    REAL  
);
```

Valori SQL

- Intregii și realii sunt reprezentați așa cum sunt.
- Șirurile de caractere se includ între două caractere apostrof.
 - Apostrof dublat = apostrof real, de ex.,
`'Joe''s Bar'`.
- Orice valoare poate fi NULL.

Date și Time

- DATE și TIME sunt tipuri în SQL.

- Forma unei valori "date" este:

DATE 'yyyy-mm-dd'

- **Exemplu:** DATE '2013-09-29' pentru 29 Septembrie 2013.

Valori Time

□ Forma pentru valoarea "time" este:

TIME 'hh:mm:ss'

cu un punct zecimal opțional și fracții de secundă în continuare.

□ **Exemplu:** TIME '15:30:02.5' = două secunde și jumătate după 3:30PM.

Declararea Cheilor

- Un atribut sau listă de attribute poate fi declarat PRIMARY KEY sau UNIQUE.
- Oricare din aceste declarații spune că nu pot exista două tuple ale relației care să corespundă întru-totul față de lista de attribute.
- Există diferențe între cele două declarații care vor fi discutate ulterior în acest curs.

Declararea Cheii formate dintr-un singur atribut

□ Se completează PRIMARY KEY sau UNIQUE după declarația de tip a atributului.

□ Exemplu:

```
CREATE TABLE Beers (  
    name        CHAR(20)  UNIQUE,  
    manf        CHAR(20)  
);
```

Declararea Cheii Compuse

- O declarație de cheie poate fi făcută ca element separat în lista de elemente a instrucțiunii CREATE TABLE.
- Această formulare este esențială atunci când cheia este compusă din mai multe attribute.
- Poate fi utilizată și la declararea cheii formate dintr-un singur atribut.

Exemplu: Cheie Compusă

□ *bar* și *beer* formează împreună cheia pentru *Sells*:

```
CREATE TABLE Sells (  
    bar      CHAR(20) ,  
    beer     VARCHAR(20) ,  
    price    REAL ,  
    PRIMARY KEY (bar, beer)  
);
```


PRIMARY KEY vs. UNIQUE

1. Într-o relație poate exista doar o singură cheie declarată PRIMARY KEY, și mai multe attribute UNIQUE.
2. Nici unul din attributele ce compun PRIMARY KEY nu pot fi NULL în nici o tuplă. Spre deosebire, attributele declarate UNIQUE pot avea valori NULL, și pot exista mai multe tuple cu valoarea NULL.

Clauzele Select-From-Where

SELECT listă de attribute

FROM una sau mai multe tabele

WHERE condiție aplicată tuplelor
tabelelor

Schema BD Exemplu

- Interogările SQL prezentate în curs folosesc următoarea schemă:

Beers(name, manf)

Bars(name, addr, license)

Drinkers(name, addr, phone)

Likes(drinker, beer)

Sells(bar, beer, price)

Frequents(drinker, bar)

- Atributele subliniate indică CHEIA fiecărei relații.

Exemplu

- Folosind **Beers(name, manf)**, care sunt mărcile de bere produse de Anheuser-Busch?

```
SELECT name
```

```
FROM Beers
```

```
WHERE manf = 'Anheuser-Busch';
```

Rezultatul Interogării

name
Bud
Bud Lite
Michelob
. . .

Răspunsul este o relație cu un singur atribut, "name", și tuplele cu denumirea fiecărei mărci de bere produsă de Anheuser-Busch, cum este "Bud".

Cum funcționează interogarea asupra unei singure relații

- Relația este specificată în clauza FROM.
- Se aplică *selecția* precizată în clauza WHERE.
- Se aplică *proiecția extinsă* indicată prin lista de attribute din clauza SELECT.

Semanticele Operațiilor

name	manf
Bud	Anheuser-Busch

Variabila de tuplă t
parcure toate
tuplurile

Se include $t.name$
în rezultat, dacă e
adevărat

Verifică dacă
 $manf$ are
valoarea
'Anheuser-Busch'

Semanticile Operațiilor

- Se poate gândi la o *variabilă de tuplă* ce vizitează fiecare tuplă a relației menționate în clauza FROM.
- Se verifică dacă tupla "curentă" satisface clauza WHERE.
- Dacă se întâmplă așa, se determină attributele sau expresiile clauzei SELECT folosind componentele acestei tuple.

* în clauza SELECT

- Atunci când există o singură relație în clauza FROM, * în clauza SELECT semnifică "toate attributele acestei relații".
- Exemplu: Se folosește Beers(name, manf):

```
SELECT *  
FROM Beers  
WHERE manf = 'Anheuser-Busch';
```

Rezultatul Interogării:

name	manf
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
.

Acum, rezultatul are fiecare atribut al relației Beers.

Redenumirea Atributelor

□ Dacă se dorește ca rezultatul să aibă nume diferite de attribute, se folosește "AS <nume nou>" pentru a redenumi un atribut.

□ **Exemplu:** Se folosește **Beers(name, manf):**

```
SELECT name AS beer, manf
```

```
FROM Beers
```

```
WHERE manf = 'Anheuser-Busch'
```

Rezultatul Interogării:

beer	manf
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
.

Expresii în clauza SELECT

□ Orice expresie ce are sens poate fi un element al clauzei SELECT.

□ **Exemplu:** Se folosește `Sells(bar, beer, price)`:

```
SELECT bar, beer,  
       price*114 AS priceInYen  
FROM Sells;
```

Rezultatul Interogării

bar	beer	priceInYen
Joe's	Bud	285
Sue's	Miller	342
...

Exemplu: Constante ca Expresii

□ Se folosește `Likes(drinker, beer)`:

```
SELECT drinker,  
       'îi place Bud' AS cuiîiplaceBud  
FROM Likes  
WHERE beer = 'Bud';
```

Rezultatul Interogării

drinker	cui îi place Bud
Sally	îi place Bud
Fred	îi place Bud
...	...

Exemplu: Integrarea Informației

- Adesea se construiesc “data warehouses” din surse multiple de date.
- Să presupunem că fiecare bar are propria relație **Menu(beer, price)** .
- Pentru a contribui la **Sells(bar, beer, price)** este nevoie de interogarea fiecărui bar și de adăugarea (“insert”) denumirii barului.

Integrarea Informației

- Să spunem, că la “Joe’s Bar” emitem interogarea (query):

```
SELECT 'Joe' 's Bar', beer, price  
FROM Menu;
```

Condiții Complexe în Clauza WHERE

- Operatori logici AND, OR, NOT.
- Comparații =, <>, <, >, <=, >=.
- Și mulți alți operatori ce produc rezultate valoare-logică.

Exemplu: Condiție Complexă

- Se folosește `Sells(bar, beer, price)`, trebuie găsit prețul de la "Joe's Bar" pentru "Bud":

```
SELECT price
```

```
FROM Sells
```

```
WHERE bar = 'Joe's Bar' AND  
       beer = 'Bud';
```

Formate tipice (pattern)

- O condiție poate compara un șir de caractere cu un format tipic în felul următor:
 - <Atribut> LIKE <pattern> sau <Atribut> NOT LIKE <pattern>
- *Pattern* este un șir de caractere încadrat între ghilimele cu:
 - % = "orice șir de caractere";
 - _ = "orice caracter".

Exemplu: LIKE

- Se folosește **Drinkers(name, addr, phone)**
Să se găsească "drinkers" cu prefixul
numărului de telefon 555:

```
SELECT name  
FROM Drinkers  
WHERE phone LIKE '%555-__ __ __ __';
```

Valori NULL

- Tuplele în relații SQL pot avea valoarea NULL pentru unul sau mai multe attribute.
- Semnificația depinde de context. Două situații obișnuite:
 - *Valoare lipsă* : de exemplu, se știe că “Joe’s Bar” are o adresă, dar aceasta nu se cunoaște.
 - *Inaplicabil* : de exemplu, valoarea atributului *soț* pentru o persoană necăsătorită.

Compararea NULL cu *Valoare*

- Condițiile au de fapt o logică 3-valori în SQL: TRUE, FALSE, UNKNOWN.
- Compararea oricărei valori (incluzând NULL cu el însuși) cu NULL conduce la UNKNOWN.
- Pentru ca o tuplă să aparțină răspunsului la o interogare, condiția din clauza WHERE trebuie să aibă valoarea logică TRUE (nici FALSE și nici UNKNOWN).

Logica Trei-Valori

□ Pentru a înțelege cum lucrează AND, OR, și NOT în logica trei-valori, să ne gândim că $\text{TRUE} = 1$, $\text{FALSE} = 0$, și $\text{UNKNOWN} = \frac{1}{2}$.

□ $\text{AND} = \text{MIN}$; $\text{OR} = \text{MAX}$, $\text{NOT}(x) = 1 - x$.

□ Exemplu:

$\text{TRUE AND (FALSE OR NOT(UNKNOWN))} =$

$\text{MIN}(1, \text{MAX}(0, (1 - \frac{1}{2}))) =$

$\text{MIN}(1, \text{MAX}(0, \frac{1}{2})) = \text{MIN}(1, \frac{1}{2}) = \frac{1}{2}$.

Exemplu Surprizător

□ Pentru următoarea relație Sells :

bar	beer	price
Joe's Bar	Bud	NULL

SELECT bar

FROM Sells

WHERE price < 2.00 OR price >= 2.00;

← UNKNOWN → ← UNKNOWN →

← UNKNOWN →

Motivație: Legile 2-Valori! = Legile 3-Valori

- Anumite legi obișnuite, cum este comutativitatea lui AND, rămân valabile în logica 3-valori.
- Dar altele nu, cum este *legea excluderii mijlocului* : $p \text{ OR NOT } p = \text{TRUE}$.
 - Când $p = \text{UNKNOWN}$, membrul stâng este $\text{MAX}(\frac{1}{2}, (1 - \frac{1}{2})) = \frac{1}{2} \neq 1$.

Sortare

□ ***ORDER BY nume_atribut
[ASC/DESC],...***

□ specifică ordinea tuplelor în relația rezultat

□ ordinea poate fi:

- crescătoare **ASC** ("ASCending") sau
- descrescătoare **DESC** ("DESCending")

Exemplu: ORDER BY

A	B
1	2
3	4
5	2

R

A	B
1	2
5	2
3	4

```
SELECT *  
FROM R  
ORDER BY B
```

Interogări imbricate (Subqueries)

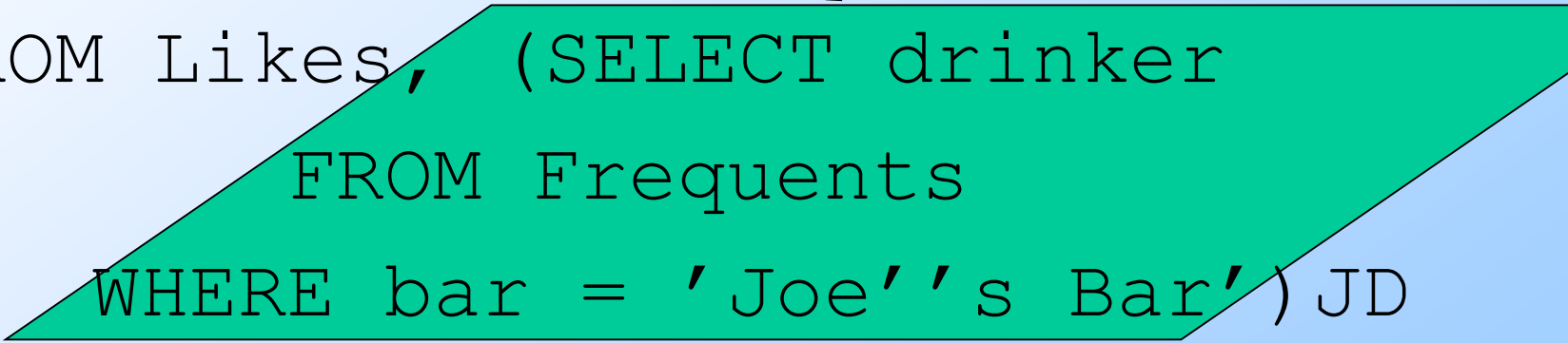
- O instrucțiune SELECT-FROM-WHERE amplasată între paranteze (*subquery*) poate fi folosită ca valoare în mai multe locuri, inclusiv în clauzele FROM și WHERE.
- **Exemplu:** în locul unei relații în clauza FROM, se poate folosi o interogare imbricată (subquery).
 - Este obligatoriu să fie utilizată o variabilă de tuplă pentru a numi tuplele rezultatului subinterogării.

Exemplu: Subquery în FROM

- Să se găsească berea preferată de cel puțin o persoană ce frecventează "Joe's Bar".

```
SELECT beer
FROM Likes, (SELECT drinker
              FROM Frequenters
              WHERE bar = 'Joe's Bar') JD
WHERE Likes.drinker = JD.drinker;
```

Persoane ce
frecventează "Joe's Bar"



Interogări imbricate ce Returnează 1 Tuplă

- Dacă o interogare imbricată returnează 1 tuplă și numai una, atunci interogarea imbricată poate fi utilizată ca valoare.
- De obicei tupla are o singură componentă.
- Dacă interogarea imbricată nu returnează nici o tuplă sau mai mult de una, se semnalează o eroare run-time.

Exemplu: Interogare imbricată cu 1 tuplă

- Se folosește `Sells(bar, beer, price)`, pentru a găsi barurile ce servesc “Miller” la prețul cu care “Joe’s Bar” vinde “Bud”.
- Răspunsul poate fi dat sub forma a două interogări:
 1. Găsește prețul cu care “Joe’s Bar” vinde “Bud”.
 2. Găsește barurile ce servesc “Miller” la acel preț.

Soluția cu interogare și interogare imbricată

SELECT bar

FROM Sells

WHERE beer = 'Miller' AND

price = (SELECT price

FROM Sells

WHERE bar = 'Joe's Bar'

AND beer = 'Bud');

Prețul cu
care "Joe's Bar"
vinde "Bud"



Operatorul IN

□ $\langle \text{tuplă} \rangle \text{ IN } (\langle \text{subquery} \rangle)$ are valoarea logică **True** dacă și numai dacă tupla este membră a relației produse de interogarea imbricată.

□ La modul opus este: $\langle \text{tuplă} \rangle \text{ NOT IN } (\langle \text{subquery} \rangle)$.

□ Expresiile "IN" pot apare în clauze WHERE.

Exemplu: IN

- Se folosesc **Beers(name, manf)** și **Likes(drinker, beer)**, pentru a găsi numele și producătorul fiecărei beri preferată de Fred.

SELECT *

FROM Beers

WHERE name IN (SELECT beer
FROM Likes
WHERE drinker = 'Fred');

Mulțimea (set)
berilor preferate
de Fred



Operatorul EXISTS

- EXISTS(<subquery>) are valoarea logică **True** dacă și numai dacă rezultatul interogării imbricate este nevid.
- **Exemplu:** Se folosește **Beers(name, manf)** , pentru a găsi acele beri ce reprezintă unica bere a producătorului ei.

Exemplu: EXISTS

```
SELECT name  
FROM Beers b1  
WHERE NOT EXISTS (
```

De notat regula: manf se referă
la o relație din cel mai apropiat
FROM, ce conține acel atribut.

Mulțimea
berilor
ce sunt
produse
de același
producător
(cu b1), dar
au altă
denumire

```
SELECT *  
FROM Beers  
WHERE manf = b1.manf AND  
      name <> b1.name);
```

De notat
operatorul
SQL "not
equals"

Operatorul "ANY"

- $x = \text{ANY}(<\text{subquery}>)$ este o condiție booleană ce are valoarea logică **True** dacă x este "egal" cu cel puțin una din tuplele rezultatului interogării imbricate.
 - "=" poate fi oricare din operatorii de comparație.
- **Exemplu:** $x \geq \text{ANY}(<\text{subquery}>)$ semnifică x nu este singura tuplă cea mai mică produsă de interogarea imbricată.
 - De notat că tuplele trebuie să aibă doar o componentă.

Operatorul "ALL"

- $x <> \text{ALL}(<\text{subquery}>)$ are valoarea logică **True** dacă pentru fiecare tuplă t din relația "subquery", x este diferit de t .
 - Cu alte cuvinte, x nu se regăsește în rezultatul interogării imbricate.
- " $<>$ " poate fi orice operator de comparație.
- **Exemplu:** $x \geq \text{ALL}(<\text{subquery}>)$ semnifică nu există tuplă mai mare ca x în rezultatul interogării imbricate.

Exemplu: ALL

- Se folosește **Sells(bar, beer, price)**, pentru a găsi berea(-ile) vândute la cel mai mare preț.

SELECT beer

FROM Sells

WHERE price >= ALL(

SELECT price
FROM Sells);

prețul "Sells" din exterior să nu fie mai mic decât nici un preț.