

Baze de date

Universitatea “Transilvania” din Brasov

Lect.dr. Costel Aldea
costel.aldea@gmail.com

Selecția (sau restricția)

Selecția (sau restricția – select, restriction) intr-o relație $r(R)$ - definita astfel:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

Unde p , *predicatul selecției*, este o formula de calcul propozițional compusa din termeni conectați prin operatorii booleani *and* (\wedge), *or* (\vee), *not* (\neg)

Fiecare termen este de forma:

- $\langle \text{atribut} \rangle \text{ op } \langle \text{atribut} \rangle$ sau
- $\langle \text{atribut} \rangle \text{ op } \langle \text{constanta} \rangle$, unde
- op este un operator de comparație: $=, \neq, >, \geq, <, \leq$

Tuplul t este selectat (introdus in rezultat) daca $p(t)=\text{TRUE}$

- In limbajul SQL: `SELECT * FROM tabel WHERE p;`

În termenii folosiți în limbajul SQL, restricția selectează o parte din liniile tabelului operand

- Exemple (bd World):

`SELECT * FROM city where CountryCode='ROM';`

`SELECT * FROM country WHERE Continent='Europe';`

`SELECT * FROM country WHERE Continent='Europe' and Population > 10000000;`

Proiectia

Proiectia (project) pe attributele A_1, A_2, \dots, A_k intr-o relatie $r(R)$ se notează:

$$\Pi_{A_1, A_2, \dots, A_k}(r), \text{ unde } A_1 \in R, A_2 \in R, \dots, A_k \in R$$

- Rezultatul este o relatie cu k attribute, cele din lista data
- Daca $\{A_1, A_2, \dots, A_k\}$ nu contine o supercheie, pot sa apara tupluri duplicat; teoretic, tuplurile duplicat se elimina, deoarece rezultatul este o multime
- In limbajul SQL proiectia se exprima astfel:

SELECT DISTINCT A_1, A_2, \dots, A_k FROM R

Daca nu se introduce parametrul DISTINCT, nu se elimina tuplurile duplicat

Exemplu (db World): SELECT DISTINCT CountryCode FROM City;

r	<table><tr><th>A</th><th>B</th><th>C</th></tr><tr><td>α</td><td>10</td><td>1</td></tr><tr><td>α</td><td>20</td><td>1</td></tr><tr><td>β</td><td>30</td><td>1</td></tr><tr><td>β</td><td>40</td><td>2</td></tr></table>	A	B	C	α	10	1	α	20	1	β	30	1	β	40	2		<table><tr><th>A</th><th>C</th></tr><tr><td>α</td><td>1</td></tr><tr><td>α</td><td>1</td></tr><tr><td>β</td><td>1</td></tr><tr><td>β</td><td>2</td></tr></table>	A	C	α	1	α	1	β	1	β	2	=	<table><tr><th>A</th><th>C</th></tr><tr><td>α</td><td>1</td></tr><tr><td>β</td><td>1</td></tr><tr><td>β</td><td>2</td></tr></table>	A	C	α	1	β	1	β	2
	A	B	C																																			
	α	10	1																																			
	α	20	1																																			
	β	30	1																																			
β	40	2																																				
A	C																																					
α	1																																					
α	1																																					
β	1																																					
β	2																																					
A	C																																					
α	1																																					
β	1																																					
β	2																																					
$\Pi_{A,C}(r)$																																						

Operația de joncțiune naturala (1)

- Joncțiunea naturala (*natural join*) combina tuplurile din doua relatii
- Fie multimile de attribute: $A = \{A1, A2, \dots, Am\}$, $B = \{B1, B2, \dots, Bn\}$, $C = \{C1, C2, \dots, Ck\}$ si doua relatii $r(R)$ si $s(S)$, unde:
 - $R = \{A, B\}$, $S = \{B, C\}$
 - deci attributele $R \cap S = B = \{B1, B2, \dots, Bn\}$ sunt comune celor două relații
- Joncțiunea naturală este o relatie $q = r \bowtie s$, care se obține în felul următor:
 - se calculează produsul cartesian al celor doua relatii: $p = r \times s$, $P = \{A, R.B, S.B, C\}$;
 - din tuplurile produsului cartesian se selecteaza acele tupluri care au valori egale pentru attributele comune ($B1, B2, \dots, Bn$): $R.B = S.B$, adică $R.B1 = S.B1$, $R.B2 = S.B2, \dots$
 - se face proiectia rezultatului pe multimea de attribute $R \cup S = \{A, B, C\}$
- Schema relatiei rezultat este $Q = R \cup S = \{A, B, C\}$
$$q = r \bowtie s = \Pi_{A,B,C} \sigma_{(r.B1=s.B1 \text{ AND } r.B2=s.B2 \dots \text{AND } r.Bn = s.Bn)}(r \times s)$$
- Attributele comune $R.B$ si $S.B$ trebuie să fie compatibile in cele doua relatii; daca sunt compatibile, ele se considera identice chiar dacă au denumiri diferite, si în reuniunea atributelor $R \cup S$ se introduc o singură dată
- Cazul cel mai frecvent de joncțiune naturala: intre doua relatii asociate (1:N), atributul comun fiind cheia straina – cheia primara (candidata) referita

Operatia de jonctiune naturala (2)

- Exemplul 1: $r \bowtie s = \Pi_{A,B,C,D,E} \sigma_{(r.D = s.D)}(r \times s)$

A	B	C	D		D	E		A	B	C	D	E
α	1	α	a		a	α		α	1	α	a	α
β	2	γ	a		b	β		β	2	γ	a	α
γ	4	β	b		c	γ		γ	4	β	b	β
α	1	γ	a		d	δ		α	1	γ	a	α
δ	2	β	b		e	ϵ		δ	2	β	b	β

- In SQL trebuie sa fie introduse explicit lista atributelor rezultatului si condițiile de egalitate ale atributelor comune: `SELECT A,B,C,R.D,E FROM R, S WHERE R.D = S.D;`
- Exemplul 2: $\text{Angajat} \bowtie \text{SECTII}$; cheia straina: `Angajat.IdSectie`
`SELECT IdAngajat, a.Nume, Prenume, DataNasterii, Adresa, Salariu, a.IdSectie, s.Nume, Buget FROM Angajat a, Sectie s WHERE a.IdSectie=s.IdSectie;`
- Exemplul 3:(bd World) $\text{city} \bowtie \text{country}$; cheia straina: `city.countryCode`
`SELECT ID, city.Name Oras, CountryCode 'Cod Tara', city.Population, country.Name Tara,Continent, from city, country where city.countryCode=country.CODE order by country.Name;`

Daca nu se afiseaza toate attributele jonctiunii, înseamnă ca s-a combinat cu o proiectie

Joncțiuni interne și externe externe

- Joncțiunea naturală se mai numește și joncțiune internă și se mai poate exprima în SQL cu cuvintele cheie **INNER JOIN**

- Exemplu de joncțiune (combinată cu o proiecție și o selecție) (bd World)

```
SELECT city.Name Oras, Code 'Cod Tara', country.Name Tara, Continent FROM city INNER JOIN country ON CountryCode=Code WHERE Continent='Antarctica' OR Continent = 'Europe' ORDER BY Continent;
```

- Joncțiunea externă introduce în plus toate liniile care există în prima relație (pentru **LEFT OUTER JOIN**) sau în cea de-a doua relație (pentru **RIGHT OUTER JOIN**) și pentru care nu există linii în cealaltă relație care să îndeplinească condiția de join; exemplu:

```
SELECT city.Name Oras, Code 'Cod Tara', country.Name Tara, Continent FROM city RIGHT OUTER JOIN country ON CountryCode=Code WHERE Continent='Antarctica' OR Continent = 'Europe' ORDER BY Continent;
```

- Se vor afișa și țările care nu au nici un oras înscris în tabelul city.

Oras	Cod Tara	Tara	Continent
Pinsk	BLR	Belarus	Europe
Moscow	RUS	Russian Federation	Europe
Ufa	RUS	Russian Federation	Europe
Nojabrsk	RUS	Russian Federation	Europe
Pușkin	RUS	Russian Federation	Europe
Tallinn	EST	Estonia	Europe
{NULL}	BVT	Bouvet Island	Antarctica
{NULL}	ATF	French Southern territories	Antarctica
{NULL}	SGS	South Georgia and the South Sandwich Islands	Antarctica

Operația de diviziune

Fie relațiile $r(R)$ și $s(S)$, unde:

- $R = \{A, B\}$ unde $A = \{A_1, A_2, \dots, A_m\}$, $B = \{B_1, B_2, \dots, B_n\}$
- $S = \{B\}$

□ Relația $q = r \div s$ are schema $Q = R - S = \{A\}$ și:

$$r \div s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

unde tu înseamnă concatenarea tuplurilor t și u

□ În limbajul SQL, diviziunea se exprimă printr-o instrucțiune SELECT, introducând explicit toate condițiile impuse valorilor atributelor

A	B		B		A
α	1		1		α
α	2		2		β
α	3				
β	1		s		$r \div s$
β	2				
δ	1				
δ	3				
r					

Rezumat: operațiile algebrei relaționale

- Algebra relațională este o colecție de operații asupra relațiilor
- Cele opt operații propuse de E.F.Codd nu constituie o mulțime minimală de operații ale algebrei relaționale
- Mulțimea minimă de operații ale algebrei relaționale consta din cinci operații primitive, pe baza cărora se poate construi orice expresie de algebra relațională:
 - Reuniunea
 - Diferența
 - Produsul Cartesian
 - Restricția (selectia)
 - Proiecția
- Celelalte operații se pot exprima prin intermediul acestora:
 - Intersecția se poate exprima prin expresia: $R \cap S = R - (R - S)$;
 - Joncțiunea este o proiecție a unei restricții a produsului cartezian al relațiilor;
 - Diviziunea este o proiecție a unei restricții asupra relației deîmpărțit
- Si celelalte trei operații sunt deosebit de utile în formularea interogărilor, astfel încât algebra relațională a păstrat toate cele opt operații propuse de E.F.Codd, la care s-a adăugat operația de redenumire a atributelor

Formularea interogărilor

- Interogarea este operația prin care se obțin informațiile dorite (care îndeplinesc o anumită condiție) dintr-o bază de date. O interogare:
 - se formulează mai întâi în limbaj natural,
 - apoi se exprima într-un limbaj abstract de interogare (algebra relațională sau calculul relațional),
 - se transpune în limbajul de interogare al SGBD-ului folosit (ex., limbajul SQL),
 - iar aplicația client transmite SGBD-ului instrucțiunea (sau instrucțiunile) obținute
- Sistemul SGBD prelucrează programul interogării în mai multe faze:
 - analiza lexicală, sintactică și semantică
 - optimizarea interogării
 - generarea codului
 - execuția și returnarea rezultatului
- În algebra relațională o interogare se formulează printr-o expresie care definește următoarele elemente:
 - Lista atributelor relației rezultat, care se numesc atribute de proiecție;
 - Lista relațiilor din care se extrag informațiile
 - Condițiile pe care trebuie să le îndeplinească tuplurile relației rezultat.
- Sunt posibile două situații:
 - interogări care se rezolvă în cadrul unei singure relații
 - interogări care se rezolvă folosind două sau mai multe relații ale bazei de date

Interogări intr-o singură relație

- Interogare in relatia $r(R)$:
 - Expresia de algebra relationala: $q = \Pi_{\text{lista_atribute}} \sigma_p(r)$
 - Instrucțiunea SQL: `SELECT lista_atribute FROM R WHERE p = TRUE;`
- Exemplul 1: Fie relația *Angajat* și interogarea: „Care sunt numele și prenumele angajaților care au un salariu mai mare sau egal cu 2000?”.
 - Expresia de algebră relațională: $q = \Pi_{\text{Nume, Prenume}} \sigma_{\text{Salariu} \geq 2000}(\text{Angajat})$
 - Instrucțiunea SQL:
`SELECT Nume, Prenume FROM Angajat WHERE Salariu >= 2000;`
- Exemplul 2: (bd - World): “Care sunt numele si populatia oraselor din tara cu codul ‘ROM’?”
 - Expresia de algebră relațională: $q = \Pi_{\text{Name, Population}} \sigma_{\text{country_id}='ROM'}(\text{city})$
 - Instrucțiunea SQL:
`SELECT Name, Population FROM city WHERE CountryCode='ROM';`
- Exemplul 3: Fie relația *Angajat* și interogarea: „Care sunt numele, prenumele si adresa angajaților care lucrează in secția numărul 1?”.
 - Expresia de algebră relațională: $q = \Pi_{\text{Nume, Prenume, Adresa}} \sigma_{\text{IdSectie} = 1}(\text{ANGAJATI})$
 - Instrucțiunea SQL:
`SELECT Nume, Prenume, Adresa FROM ANGAJATI WHERE IdSectie=1;`

Interogări in două sau mai multe relații

- Dacă attributele de proiecție și attributele din condiția de interogare nu aparțin unei singure relații, pentru rezolvarea interogării trebuie să fie folosite toate acele relațiile care, împreună, conțin attributele și asocierile necesare
- Conceptual, o astfel de interogare se rezolvă astfel:
 - se construiește mai întâi o relație care să conțină toate attributele implicate prin combinarea relațiilor necesare, folosind operații de produs cartezian sau joncțiuni;
 - în relația obținută se aplică o selecție (restricție) (cu condiția de interogare p);
 - apoi se face proiecția (pe attributele de proiecție).

- Expresia generală de algebra relațională a interogării este:

$$q = \Pi_{\text{lista_atribute}} \sigma_p(r \times s \times t \dots)$$

- Dacă între relațiile din produsul cartezian există attribute comune care trebuie să aibă valori egale (de regulă, perechile cheie străină - cheie candidată) atunci se pot face operații de joncțiune:

$$q = \Pi_{\text{lista_atribute}} \sigma_p \text{ AND } \text{conditii-join}(r \bowtie s \bowtie t \dots)$$

- O interogare poate conține una sau mai multe subinterogări
- În limbajul SQL, o interogare se exprimă prin instrucțiuni SELECT în care:
 - clauza WHERE combină atât condițiile impuse valorilor atributelor cât și condițiile de joncțiuni
 - Joncțiunile se pot specifica și în clauza FROM (cu INNER JOIN, OUTER JOIN)

Interogări in două relații

A. Fie interogarea: Care sunt numele, prenumele, funcția, salariul și denumireasecției în care lucrează angajații?

- Expresia de algebră relațională este: $q = \Pi_{\text{Nume, Prenume, Salariul, Denumire}}(\text{Angajat} \bowtie \text{Sectie})$
- Instrucțiunea SQL corespunzătoare acestei interogări:

SELECT Nume, Prenume, Salariul, Denumire FROM Angajat, Sectie WHERE Sectie.IdSectie = Angajat.IdSectie

- Se efectuează o “navigare” în baza de date, pe atributul comun (IdSectie)

B. Fie interogarea: Care sunt numele, prenumele, funcția și salariul angajaților care lucrează în secția cu denumirea ‘Productie’?

- $q = \Pi_{\text{Nume, Prenume, Functia, Salariul}} \sigma_{\text{Denumire} = \text{'Productie'}}(\text{Angajat} \bowtie \text{Sectie})$
- SELECT Nume, Prenume, Functia, Salariul FROM Angajat, Sectie WHERE Sectie.IdSectie = Angajat.IdSectie AND Denumire= ‘Productie’;

Interogări in trei relații (1)

- Fie următoarele relații asociate din baza de date Cinemagia:
 - FILM(film_id, title, description, release_year,)
 - ACTOR(actor_id, first_name, last_name, last_update)
 - FILM_ACTOR(film_id, actor_id, last_update)
- Interogarea: “În ce filme au jucat fiecare din actorii din baza de date Cinemagia?”
- $q = \Pi_{\text{actor_id, first_name, last_name, title}}(\text{film} \bowtie \text{film_actor} \bowtie \text{actor})$
- Instrucțiunea SQL:

```
SELECT ACTOR.actor_id, first_name, last_name, title  
FROM FILM, FILM_ACTOR, ACTOR  
WHERE FILM.film_id = FILM_ACTOR.film_id AND ACTOR.actor_id =  
FILM_ACTOR.actor_id ORDER BY ACTOR.actor_id;
```

- Se poate folosi și sintaxa INNER JOIN:

```
SELECT ACTOR.actor_id, first_name, last_name, title  
FROM (FILM INNER JOIN FILM_ACTOR ON  
FILM.film_id=FILM_ACTOR.film_id) INNER JOIN ACTOR ON  
ACTOR.actor_id = FILM_ACTOR.actor_id ORDER BY....;
```

Interogări in trei relații (2)

film_id	title	description
1	ACADEMY DINOSAUR	A Epic Drama of ...
2	ACE GOLDFINGER	A Astounding Epi...
3	ADAPTATION HOLES	A Astounding Ref...
4	AFFAIR PREJUDICE	A Fanciful Docum...
5	AFRICAN EGG	A Fast-Paced Doc...
6	AGENT TRUMAN	A Intrepid Panor...
7	AIRPLANE SIERRA	A Touching Saga ...

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINNESS	2006-02-15
2	NICK	WAHLBERG	2006-02-15
3	ED	CHASE	2006-02-15
4	JENNIFER	DAVIS	2006-02-15
5	JOHNNY	LOLLOBRIGID	2006-02-15
6	BETTE	NICHOLSON	2006-02-15

	actor_id	film_id	last_update
<input type="checkbox"/>	1	1	2006-02-15 05:05:03
<input type="checkbox"/>	1	23	2006-02-15 05:05:03
<input type="checkbox"/>	1	25	2006-02-15 05:05:03
<input type="checkbox"/>	1	106	2006-02-15 05:05:03
<input type="checkbox"/>	1	140	2006-02-15 05:05:03
<input type="checkbox"/>	1	166	2006-02-15 05:05:03
<input type="checkbox"/>	1	277	2006-02-15 05:05:03

```
SELECT actor_id, film.film_id, title
FROM FILM INNER JOIN FILM_ACTOR
ON FILM.film_id=FILM_ACTOR.film_id
ORDER BY actor_id;
```

1 Result 2 Profiler 3 Messages 4 Table Data

(Read Only)

actor_id	film_id	title
1	506	LADY STAGE
1	23	ANACONDA CONFESSIONS
1	939	VERTIGO NORTHWEST
1	980	WIZARD COLDBLOODED
1	277	ELEPHANT TROJAN
1	361	GLEAMING JAWBREAKER
1	25	ANGELS LIFE
1	106	BULWORTH COMMANDMENTS
1	605	MULHOLLAND BEAST
1	166	COLOR PHILADELPHIA
1	635	OKLAHOMA JUMANJI
1	832	SPLASH GUMP
1	499	KING EVOLUTION
1	749	RULES HUMAN
1	970	WESTWARD SEABISCUIT
1	140	CHEAPER CLYDE
1	509	LANGUAGE COWBOY
1	1	ACADEMY DINOSAUR
1	438	HUMAN GRAFFITI
2	314	FIGHT JAWBREAKER
2	811	SMILE EARRING

Interogări in trei relații (3)

```
SELECT ACTOR.actor_id, first_name, last_name, title
FROM FILM, FILM_ACTOR, ACTOR
WHERE ACTOR.actor_id = FILM_ACTOR.actor_id AND
      FILM.film_id = FILM_ACTOR.film_id ORDER BY ACTOR.actor_id;
SELECT ACTOR.actor_id, first_name, last_name, title
FROM (FILM INNER JOIN FILM_ACTOR ON FILM.film_id=FILM_ACTOR.film_id)
     INNER JOIN ACTOR ON ACTOR.actor_id = FILM_ACTOR.actor_id
ORDER BY ACTOR.actor_id;
```

1 Result



2 Profiler



3 Messages



4 Table Data



5 Objects



(Read Only)

actor_id	first_name	last_name	title
1	PENELOPE	GUINNESS	CHEAPER CLYDE
1	PENELOPE	GUINNESS	MULHOLLAND BEAST
1	PENELOPE	GUINNESS	SPLASH GUMP
1	PENELOPE	GUINNESS	HUMAN GRAFFITI
1	PENELOPE	GUINNESS	KING EVOLUTION
1	PENELOPE	GUINNESS	ANGELS LIFE
1	PENELOPE	GUINNESS	WIZARD COLDBLOODED
1	PENELOPE	GUINNESS	BULWORTH COMMANDMENTS
1	PENELOPE	GUINNESS	VERTIGO NORTHWEST
1	PENELOPE	GUINNESS	GLEAMING JAWBREAKER
1	PENELOPE	GUINNESS	RULES HUMAN
1	PENELOPE	GUINNESS	ELEPHANT TROJAN
1	PENELOPE	GUINNESS	ANACONDA CONFESSIONS
1	PENELOPE	GUINNESS	LADY STAGE
1	PENELOPE	GUINNESS	COLOR PHILADELPHIA
1	PENELOPE	GUINNESS	ACADEMY DINOSAUR
1	PENELOPE	GUINNESS	LANGUAGE COWBOY
1	PENELOPE	GUINNESS	OKLAHOMA JUMANJI
1	PENELOPE	GUINNESS	WESTWARD SEABISCUIT
2	NICK	WAHLBERG	FIGHT JAWBREAKER
2	NICK	WAHLBERG	MALLRATS UNITED

Subinterogări

- ❑ Subinterogările sunt operații care determină diferite date (valori scalare, tabele rezultat, număr de elemente etc.) folosite în interogarea de bază
- ❑ Subinterogările pot conține la rândul lor alte subinterogări
- ❑ Exemplul 1: Care sunt angajații (nume, prenume, adresa) care lucrează în aceeași secție cu angajatul cu numele Ionescu și prenumele Ion?
 - Se determină printr-o subinterogare în ce secție lucrează angajatul dat
 - Se selectează toți angajații din acea secție:

SELECT Nume, Prenume, Adresa FROM Angajat

WHERE IdSectie = (SELECT IdSectie FROM Angajat WHERE Nume = 'Ionescu' AND Prenume = 'Ion');

- ❑ Exemplul 2: Care sunt numele, prenumele, denumirea secției și salariul angajaților care au salariul egal cu salariul maxim pe una din secții:
 - Se determină printr-o subinterogare tabelul cu valori maxime ale salariului în fiecare secție
 - Se selectează angajații care au salariul în mulțimea salariilor maxime pe secții:

SELECT Nume, Prenume, Salariul, Denumire FROM Angajat INNER JOIN Sectie ON Angajat.IdSectie=Sectie.IdSectie

WHERE Salariul IN (SELECT MAX(Salariul) FROM Angajat Group By IdSectie);

Instrucțiuni pentru selecția datelor

- ❑ **Instrucțiunile de selecție** reprezintă una din categoriile cele mai importante ale limbajului SQL
- ❑ Indiferent dacă sunt cereri simple sau complexe, cuvântul cheie este **SELECT**
- ❑ Pentru cererile de interogare simple, sintaxa instrucțiunii este

SELECT [domeniu] listă_selecție
FROM nume_tabel1, nume_tabel2,...
[WHERE criteriu_selecție]
[ORDER BY câmpuri_criteriu **[ASC|DESC]];**

Cereri de interogare simple

□ Domeniu

- Specifică o opțiune de *includere* sau *eliminare* din rezultatul selecției, a înregistrărilor care conțin duplicate
- Opțiunile posibile sunt:
 - **ALL** cere includerea tuturor înregistrărilor care îndeplinesc condițiile impuse
 - **DISTINCT** cere eliminarea înregistrărilor care conțin duplicate în câmpurile selectate, afișând numai o apariție a acesteia.
 - **DISTINCTROW** cere eliminarea înregistrărilor care conțin duplicate în ansamblul lor, nu numai în câmpurile selectate, afișând numai o apariție a acesteia

Cereri de interogare simple

- **Listă_selecție** cuprinde câmpurile care dorim să apară în tabelul cu rezultatele interogării
- **Clauza FROM** specifică numele **tabelului** sau tabelelor pe care se face cererea de interogare. Pentru mai multe tabele, numele acestora se separă cu „,”
 - Pe lângă tabele, ca sursă de informații pot apărea și **interogări** deja create

Cereri de interogare simple

- **Clauza WHERE** cere numai înregistrările care îndeplinesc criteriul de selecție specificat
 - Criteriul de selecție este o expresie care conține obligatoriu și un **operator adecvat** tipului de dată al câmpului respectiv
 - Clauza WHERE este opțională
- **Clauza ORDER BY** cere ordonarea în mod crescător (ASC) sau descrescător (DESC) a rezultatelor interogării
 - Ordonarea este opțională și se poate face după unul sau mai multe câmpuri_criteriu

Cereri de interogare complexe

- Sunt acele interogări în care apar funcțiile
 - **de agregare**
 - **asocierile**
 - **combinările**

Funcțiile de agregare (de grup)

- Permite construirea unor interogări complexe, prin care utilizatorul cere gruparea înregistrărilor care au câmpuri cu aceeași valoare, în scopul efectuării unor calcule
- În standardul ISO sunt definite 5 funcții de agregare:
 - **COUNT** returnează numărul de valori dintr-o coloană specificată
 - **SUM** returnează suma valorilor dintr-o coloană specificată
 - **AVG** returnează media valorilor dintr-o coloană specificată
 - **MIN** returnează cea mai mică valoare dintr-o coloană specificată
 - **MAX** returnează cea mai mare valoare dintr-o coloană specificată

Funcțiile de agregare (de grup)

```
SELECT [domeniu] funcție_agregat(ume_câmp) AS alias [,listă_selecție]
FROM nume_tabel1, nume_tabel2,...
GROUP BY câmp_de_grupare
[HAVING criteriu_de_grupare]
[ORDER BY câmpuri_criteriu [ASC|DESC]];
```

- AS alias asociază un alias rezultatului funcției de agregare
- *Clauza* GROUP BY precizează câmpul sau câmpurile după care se face gruparea înregistrărilor
- *Clauza* HAVING conține criteriul care va fi aplicat câmpului argument al funcției agregat
 - Spre deosebire de *WHERE*, care acționează înainte de gruparea înregistrărilor, HAVING acționează după definirea acestora

Joncțiunea (JOIN)

- ❑ Limbajul SQL oferă posibilitatea de a grupa și folosi date din tabele diferite
- ❑ Operațiile de asociere induse de ***clauza JOIN*** au ca rezultat producerea tuturor combinațiilor posibile, pentru conținutul informațional al fiecărui tabel
- ❑ Noile înregistrări care rezultă în urma joncțiunii sunt disponibile pentru selecțiile următoare
- ❑ La o asociere pot participa mai mult de 2 tabele

Joncțiunea (JOIN)

- Există mai multe categorii de joncțiuni:
 - CROSS (încrucișată) – rar folosită
 - ECHIVALENTĂ (echijoncțiune) – cea mai folosită – presupune folosirea *clauzei* **WHERE** asociată cu o egalitate dorită
 - NEECHIVALENTĂ (non echijoncțiune) - rar folosită - presupune folosirea *clauzei* **WHERE** asociată cu orice alt operator de comparare, în afară de „=”

Joncțiunea (JOIN)

```
SELECT [domeniu] listă_selecție  
FROM nume_tabel1, nume_tabel2,...  
[WHERE criteriu_de_asociere]  
[ORDER BY câmpuri_criteriu [ASC|DESC]];
```

- Deoarece în instrucțiunile SQL care descriu joncțiuni se utilizează câmpuri care fac parte din tabele diferite, trebuie specificat numele tabelului de care aparțin, folosind sintaxa
nume_tabel.nume_câmp
fără spații înainte sau după punct

Joncțiunea (JOIN)

- După modul de asociere a înregistrărilor din tabele, joncțiunile pot fi:
 - **interne** sau **INNER JOIN** determină o asociere a înregistrărilor din tabele, astfel încât să rezulte un număr total de înregistrări egal cu **produsul** numărului de înregistrări din fiecare tabel
 - **externe de stânga** sau **LEFT OUTER JOIN**
 - **externe de dreapta** sau **RIGHT OUTER JOIN**

Joncțiunea (JOIN)

```
SELECT [domeniu] listă_selecție  
FROM nume_tabel1  
{INNER|LEFT OUTER|RIGHT OUTER} JOIN nume_tabel2  
ON criteriu_de_asociere  
[{INNER|LEFT OUTER|RIGHT OUTER} JOIN nume_tabel3  
ON criteriu_de_asociere]...  
[WHERE criteriu_selecție]  
[ORDER BY câmpuri_criteriu [ASC|DESC]]
```

- ❑ **JOIN** specifică tabelul care va fi asociat (nume_tabel2, nume_tabel3) celui din clauza FROM
- ❑ **ON** arată între ce câmpuri trebuie să existe relația pe care se bazează joncțiunea. Criteriul de asociere conține obligatoriu operatorul „=”.

Combinările (UNION)

- Când utilizatorul dorește să vadă rezultatele mai multor interogări SELECT în același timp, prin combinarea ieșirilor lor, se poate utiliza facilitatea UNION

```
SELECT listă_câmpuri FROM nume_tabel1  
UNION SELECT listă_câmpuri FROM nume_tabel2  
[GROUP BY câmp_de_grupare]  
[HAVING criteriu_de_agregare]  
[UNION SELECT listă_câmpuri FROM nume_tabel3  
[GROUP BY câmp_de_grupare]  
[HAVING criteriu_de_agregare]]  
[UNION ...]  
[ORDER BY câmpuri_criteriu [ASC|DESC]];
```

Combinările (UNION)

- Există mai multe restricții pentru instrucțiunile care generează interogările UNION:
 - Numărul câmpurilor din lista de câmpuri din fiecare instrucțiune SELECT și UNION SELECT trebuie să fie aceeași
 - Secvența de nume din fiecare listă de câmpuri trebuie să corespundă unor intrări identice
 - Este permisă utilizarea doar o dată a clauzei ORDER BY, după ultima instrucțiune UNION SELECT

Combinările (UNION)

**SELECT nume, prenume, vârstă
FROM Colaboratori2001**

UNION

**SELECT nume, prenume, vârstă
FROM Colaboratori2002
ORDER BY nume;**

**SELECT nume, prenume, vârstă
FROM Colaboratori2001
GROUP BY categoria
HAVING categoria="student"**

UNION

**SELECT nume, prenume, vârstă
FROM Colaboratori2002
GROUP BY categoria
HAVING categoria="student"**

Instrucțiuni pentru manipularea datelor

- Foarte utile în exploatarea unei BD, aceste instrucțiuni se implementează prin interogările **de acțiune**
- Este necesară o mare precauție în utilizarea lor deoarece acțiunile sunt **ireversibile**, putând influența inclusiv integritatea referențială a BD
- Cele mai folosite sunt
 - CREATE
 - INSERT
 - UPDATE
 - DELETE

Instrucțiuni pentru manipularea datelor CREATE

- ❑ Duc la generarea unui nou tabel pornind de la structura și conținutul unor tabele deja existente
- ❑ Se folosește instrucțiunea **SELECT ... INTO**

```
SELECT [domeniu] (câmp1,câmp2...)  
INTO tabel_nou  
FROM tabel_sursa  
[WHERE criteriu_de_adăugare];
```

Instrucțiuni pentru manipularea datelor INSERT

- Sunt folosite pentru adăugarea de înregistrări dintr-un tabel în altul
- Există două forme ale instrucțiunii:
 - **INSERT ... VALUES**
 - **INSERT ... SELECT**

Instrucțiuni pentru manipularea datelor INSERT

- ❑ In primul caz se adaugă *o singură înregistrare într-un tabel*, menționându-se câmpurile și valorile acestora
- ❑ Se utilizează pentru operații simple, care presupun lucrul cu un număr redus de înregistrări

INSERT INTO nume_tabel (câmp1, câmp2...)

VALUES (valoare1, valoare2...)

Instrucțiuni pentru manipularea datelor INSERT

□ Reguli

- Valorile din clauza VALUES trebuie să aibă aceeași natură cu câmpurile din clauza INTO
- Mărimea valorii trebuie să fie mai mică decât dimensiunea câmpului
- Este obligatorie corespondență între câmp1 și valoare1, etc.
- Dacă un câmp are specificația NOT NULL, este obligatorie introducerea unei valori pentru aceasta

Instrucțiuni pentru manipularea datelor INSERT

- În al doilea caz, este posibil să se copieze *mai multe înregistrări* dintr-un tabel în unul sau mai multe tabele

INSERT INTO tabel_destinație (câmp1, câmp2...)

SELECT [domeniu] câmp1, câmp2...

FROM tabel_sursă

WHERE criteriu_de_adăugare;

Instrucțiuni pentru manipularea datelor INSERT

- În plus față de regulile enunțate mai devreme, trebuie respectate și acestea:
 - numărul și natura câmpurilor din clauza **INTO** să fie aceleași cu cele returnate de instrucțiunea **SELECT**
 - dacă nu se introduce **WHERE**, toate înregistrările din **tabel_sursă** vor fi adăugate în **tabel_destinație**

Instrucțiuni pentru manipularea datelor DELETE

- ❑ Șterg parțial sau total înregistrările dintr-un tabel
- ❑ Nu se folosește pentru ștergerea de valori din câmpuri individuale, ci acționează asupra înregistrării în totalitatea ei
- ❑ Dacă se șterg toate înregistrările, structura de tabel rămâne, ea putând fi eliminată numai cu DROP TABLE

Instrucțiuni pentru manipularea datelor DELETE

DELETE FROM nume_tabel
[WHERE criteriu_de_ștergere];

- Ca și instrucțiunea INSERT, operația de ștergere a înregistrărilor dintr-o tabelă poate duce la probleme de integritate referențială în alte tabele
 - Exemple

DELETE *
FROM Vânzări

DELETE *
FROM Angajați
WHERE Vârsta>60

Instrucțiuni pentru manipularea datelor UPDATE

- Permite introduce înregistrări noi și permite modificarea valorilor câmpurilor din înregistrări existente

UPDATE nume_tabel

SET nume_câmp1=valoare1 [,nume_câmp2=valoare2]...
[WHERE criteriu_de_actualizare];

Instrucțiuni pentru manipularea datelor UPDATE

- Ca și în celelalte locuri unde apare clauza WHERE, restricționarea se poate accentua folosind și operatori logici

Exemplu

UPDATE Comunicații

SET Rețea="Orange"

WHERE Rețea="Dialog" AND Data>#12.12.2001;

Cereri de interogare imbricate

- ❑ Scrierea unei interogări în cadrul alteia duce la apariția unei subinterogări, setul de rezultate obținute de la aceasta constituind argument pentru prima interogare
- ❑ Cele două tabele trebuie să aibă un câmp comun (nume_câmp) care va reprezenta câmpul de legătură ce stă la baza construirii subinterogării

```
SELECT lista_câmpuri  
FROM tabel1  
WHERE tabel1.nume_câmp=  
    (SELECT nume_câmp  
     FROM tabel2  
     WHERE criteriu_de_selecție);
```