Limbaje de interogare - SQL

Curs 2

Interogări

- Posibile informaţii pe care dorim sa le obţinem din baza de date a facultății:
  - Care este numele studentului cu *sid= 2833*?
  - Care este salariul profesorilor care predau cursul *Alg100*?
  - Câţi studenţi sunt înscrişi la cursul *Alg100*?

- Astfel de întrebări referitoare la datele stocate într-un SGBD se numesc ***interogări.***

  → ***limbaj de interogare***

Limbaje SGBD

- *Data Definition Language (DDL)*
  - Definesc structura **conceptuală**
  - Descriu **constrângerile de integritate**
  - Influenţează **structura fizică** (în anumite SGBD-uri)

- *Data Manipulation Language (DML)*
  - Operaţii aplicate instanţelor unei baze de date
  - DML procedural (cum?) vs. DML declarative (ce?)

Limbaje de interogare pentru BD relaţionale

- **SQL (Structured Query Language)**

  SELECT *name* FROM *Students* WHERE *age*> 20

- Algebră relațională

  $\pi_{name}(\sigma_{age>20}(Students))$

- Domain Calculus

  {<X>|∃V ∃Y ∃Z ∃T: Students(V,X,Y,Z,T) ∧ Z>20}

- T-uple Calculus

  {X| ∃Y : Y∈Students ∧ Y.age>20 ∧ X.name=Y.name}

Nivele SQL

- ***Data-definition language (DDL) / Limbaj de definire a datelor**:*
  - Creare / ștergere / modificare *tabele* şi *views*.
  - Definire *constrângeri de integritate* (CI's).

- ***Data-manipulation language (DML) / Limbaj de manipulare a datelor***
  - Permit formularea de interogări
  - Inserare /ştergere / modificare înregistrări.

- ***Controlul accesului:***
  - Asignează sau elimină drepturi de acces și modificare a *tabelelor* şi a *view*-urilor.

Instrucțiuni SQL

- Instrucțiunile SQL pot fi grupate în mai multe categorii:
  - Definire componente: CREATE, ALTER, DROP
  - Gestiune & regăsire date: INSERT, UPDATE, DELETE, SELECT
  - Gestiune tranzacții: START TRANSACTION, COMMIT, ROLLBACK

Studenti

NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512

Cursuri

CursId	Titlu	Credite
BD1	Baze de date I	5
BD2	Baze de date II	6
ALG	Algebra	4

Examene

NrMatr	CursId	Nota
pa123	BD1	10
pa123	BD2	8
ma111	BD1	9

Observație. Reținem Varsta ca atribut pentru simplitatea exemplelor. În practică este de preferat să reținem data nașterii, întrucât aceasta nu se schimbă de la an la an.

Data Definition Language (DDL)
Comenzi

Crearea și ștergerea unei baze de date

- CREATE DATABASE nume_bd
- DROP DATABASE nume_bd
- ALTER DATABASE operatie

- Exemplu:

```
CREATE DATABASE Facultate

ALTER DATABASE Facultate
MODIFY Name = Universitate

DROP DATABASE Universitate
```

Crearea tabelelor

CREATE TABLE nume_tabel
(definire_coloana [,definire_coloana] ...[, restrictii_tabel])
definire_coloana: nume_coloana tip_data[(lungime)] [DEFAULT valoare] [restrictie_coloana]

```
CREATE TABLE Studenti
(nrMatr varchar(10) PRIMARY KEY,
 nume VARCHAR(30),
 prenume VARCHAR(30),
 email VARCHAR(30),
 varsta INT,
 grupa INT,
 cnp CHAR(13) UNIQUE, -- cheie candidat
 )
```

Crearea tabelelor

- În limbajul SQL fiecare coloană, variabilă locală, expresie sau parametru are un tip de date
- Un tip de date este un atribut care specifică ce fel de valori pot fi stocate în obiectul respectiv

- Exemple: int, tinyint, smallint, bigint, decimal, float, real, money, nchar, varchar, datetime, date, time

Crearea tabelelor

- **restricții associate coloanei:**
  - NOT NULL – nu poate să aibă valori nedefinite/nule
  - PRIMARY KEY – coloana se definește cheie primară **(pot fi chei primare compuse)**
  - UNIQUE – valorile pentru coloană sunt unice **(chei candidat)**
  - CHECK(condiție) – condiția pe care trebuie să o îndeplinească valorile coloanei (condiții simple cu valoarea true sau false)
  - FOREIGN KEY REFERENCES tabel_parinte[(nume_coloana)] [ON UPDATE actiune] [ON DELETE actiune]
- **restricții associate tabelului:**
  - PRIMARY KEY(lista_coloane) – definire cheie primară pentru tabel
  - UNIQUE(lista_coloane) – valorile pentru lista de coloane sunt unice
  - CHECK(condiție) – condiția pe care trebuie să o îndeplinească valorile unei linii

Exemplu constrangeri de integritate

```
CREATE TABLE Studenti
(nrMatr varchar(10) PRIMARY KEY,
 nume VARCHAR(30),
 prenume VARCHAR(30),
 email VARCHAR(30),
 varsta INT,
 grupa INT,
 cnp CHAR(13) UNIQUE,
 CONSTRAINT varstaInterval
 CHECK (varsta >= 18 AND varsta <=70))
```

Cheie primară incrementată automat

- Un tabel poate avea o cheie primară de tip INT care se incrementează automat la inserarea de valori noi (valorile acestei chei sunt administrate de server)
- In MS SQL Server → IDENTITY(1,1)

```
CREATE TABLE Review
(id INT IDENTITY(1,1) PRIMARY KEY,
 descriere CHAR(500))
```

Crearea tabelelor

- **restricții associate tabelului**:
  - FOREIGN KEY (lista_coloane) REFERENCES tabel_parinte[(lista_coloane)] [ON UPDATE actiune] [ON DELETE actiune]
- **acțiuni care se pot asocia unei chei externe**:
  - NO ACTION – operația nu se poate executa daca se încalcă restricțiile de integritate
  - SET NULL – valoarea pentru cheia externă se înlocuiește cu null
  - SET DEFAULT – valoarea pentru cheia externă se înlocuiește cu valoarea implicită
  - CASCADE – se permite ștergerea sau modificarea în tabelul părinte, dar operația generează ștergeri sau modificări corespunzătoare și în tabelul fiu

Crearea tabelelor

```
CREATE TABLE Cursuri
(cursId VARCHAR(20)primary key,
 titlu VARCHAR(50),
 credite INTEGER)

CREATE TABLE Examene
(nrMatr VARCHAR(10)FOREIGN KEY REFERENCES Studenti ON DELETE CASCADE,
 cursId VARCHAR(20),
 Note REAL,
 PRIMARY KEY (nrMatr, cursId), -- cheie primara compusa
 CONSTRAINT FK_Examene_Cursuri FOREIGN KEY (cursId) REFERENCES Cursuri)
```

Modificarea tabelelor

- ALTER TABLE – modificarea structurii unui tabel definit

  ALTER TABLE nume_tabel operatie

- Câteva operații posibile (există diferențe între SGBD-uri):
  - adăugare/ modificare/ eliminarecoloane:
  - ADD definire_coloana
  - {ALTER COLUMN | MODIFY COLUMN} definire_coloana
  - DROP COLUMN nume_coloana

Modificarea tabelelor

- Exemplu adaugarea unei coloane

```
ALTER TABLE Studenti
ADD initialaTatalui VARCHAR(3)
```

- Exemplu schimbarea tipului de date a unei coloane

```
ALTER TABLE Studenti
ALTER COLUMN initialaTatalui VARCHAR(5)
```

- Exemplu stergerea unei coloane

```
ALTER TABLE Studenti
DROP COLUMN initialaTatalui
```

Modificarea tabelelor

- Adăugare/ ștergere restricție:
  - ADD [CONSTRAINT nume_constrangere] PRIMARY KEY(lista_coloane)
  - ADD [CONSTRAINT nume_constrangere] UNIQUE(lista_coloane)
  - ADD [CONSTRAINT nume_constrangere] FOREIGN KEY (lista_coloane) REFERENCES nume_tabel[(lista_coloane)] [ON UPDATE *actiune*] [ON DELETE *actiune*]
  - DROP [CONSTRAINT] nume_constrangere

- Exemplu:

```
ALTER TABLE Studenti
DROP CONSTRAINT varstaInterval
```

Ștergerea tabelelor

- DROP TABLE – ștergerea unui tabel

  DROP TABLE nume_tabel

- Exemplu:

```
DROP TABLE Studenti
```

Data Manipulation Language (DML)

Comenzi pentru inserare, modificare și ștergere date

Adăugarea datelor

- INSERT –adăugarea de înregistrări

  INSERT INTO nume_tabel [(lista_coloane)] VALUES (lista_valori)

- Bulk Insert:

  INSERT INTO nume_tabel [(lista_coloane)] subinterogare

, unde *subinterogare* se referă la o mulțime de înregistrări (generate cu ajutorul instrucțiunii SELECT)

Adăugarea datelor

- Exemplu:

```
INSERT INTO Cursuri(cursId, titlu, credite)
VALUES ('BD1', 'Baze de date', 5)


INSERT INTO Cursuri(cursId, titlu, credite)
SELECT cursId, titlu, credite
FROM PlanInvatamant
WHERE anInceput = 2019
AND anSfarsit = 2020
```

Adăugarea datelor

- Pot fi omise coloane sau valori la inserarea datelor in tabelor in următoarele cazuri:
  - Coloanele respective permit valori nule
  - Coloanele respective au o valoare DEFAULT definită la crearea tabelului
  - Coloana respectivă este o coloana în care valorile se incrementează automat (IDENTITY)

- De exemplu, sa presupunem că:
  - Titlul cursului poate fi null
  - Credite are valoarea 0 definită ca DEFAULT

```
INSERT INTO Cursuri (cursId)
VALUES ('ALG1’)
```

Modificarea datelor

- UPDATE–modificarea unor înregistrări

  UPDATE nume_tabel

  SET nume_coloana = expresie [,nume_coloana = expresie] ...

  [WHERE conditie]

- se modifică înregistrările din tabel pentru care condiția din clauza WHERE se evaluează la *adevărat*
- dacă se omite clauza WHERE se consideră toate înregistrările din tabel
- Valorile coloanelor precizate în SET se schimbă la valoarea expresiilor asociate

Modificarea datelor

```
UPDATE Studenti
SET varsta = varsta + 1
WHERE cnp = '123456789012'
```

Ștergerea datelor

- DELETE –ștergereade înregistrări

  DELETE FROM nume_tabel [WHERE conditie]

- se elimină din tabel înregistrările pentru care condiția din clauza WHERE se evaluează la *adevărat*\
- dacă se omite clauza WHERE, **se șterg toate înregistrările** din tabel

```
DELETE
FROM Studenti
WHERE nume = 'Popescu'
```

Condiții de filtrare a datelor

- Expresie operator_relational expresie (ex. age > 20, grupa != 711)
- Expresie [NOT] BETWEEN min AND max (ex. age BETWEEN 18 AND 30)
- Expresie[NOT] LIKE 'sablon' ("%" –orice număr de caractere, "_" – un caracter)
  - Ex. nume LIKE 'P%'
- Expresie IS [NOT] NULL (ex. email IS NOT NULL)
- Expresie[NOT] IN (valoare[, valoare] ...) (ex. age NOT IN (64, 65))
- Expresie[NOT] IN (subselectie)
- Expresie operator_relational {ALL | ANY} (subselectie)
- [NOT] EXISTS (subselectie)
- Condițiile de filtrare pot fi elementare (descrise mai sus) sau compuse cu operatorii logici NOT, AND, OR și paranteze

Data Manipulation Language (DML)
Interogări

Interogare SQL simplă

SELECT [DISTINCT] lista-select
FROM lista-from
WHERE calificare

- **lista-select** – listă de atribute care apar în relațiile din **lista-from**
- **lista-from** – listă de nume de relații; fiecare dintre acestea poate fi urmată de o variabilă range
- **calificare** – cuprinde expresii logice care conțin comparații (Atr op Const sau Atr1 op Atr2, unde op ∈{<, <=, =, >, >=, ≠} compuse cu operatorii logici AND, OR, NOT
- **DISTINCT** se folosește pentru a elimina duplicatele din rezultat

Găsiți studenții cu vârsta 21

```
SELECT *
FROM Studenti
WHERE Varsta = 21
```

Găsiți studenții cu vârsta 21, dar afișați doar numele și prenumele lor

```
SELECT Nume, Prenume
FROM Studenti
WHERE Varsta = 21


SELECT S.Nume, S.Prenume
FROM Studenti S
WHERE S.Varsta = 21
```

- Cele două interogări sunt echivalente.
- S este o variabilă range (un alias) pentru relația Studenti -> este util în primul rând când avem nevoie de **două instanțe ale aceleași relații** sau când apar **mai multe coloane cu același nume** în tabelele din interogare → soluționează o ambiguitate

Ce returnează interogarea?

```
SELECT S.Nume, E.CursId
FROM Studenti S, Examene E
WHERE S.NrMatr = E.NrMatr AND E.Nota=10
```

Ce returnează interogarea?

```
SELECT S.Nume, E.CursId
FROM Studenti S, Examene E
WHERE S.NrMatr = E.NrMatr AND E.Nota=10
```

- Returnează toți studenții care au luat o notă de 10 (indiferent la ce curs)

Ce returnează interogarea?

```
SELECT S.Nume, E.CursId
FROM Studenti S, Examene E
WHERE S.NrMatr = E.NrMatr AND E.Nota=10
```

- Observați că avem coloana *NrMatr* în ambele coloane, prin urmare trebuie specificat la ce tabel ne referim când folosim coloana.
- Care este rolul condiției `S.NrMatr = E.NrMatr` ?

Ce returnează interogarea?

```
SELECT S.Nume, E.CursId
FROM Studenti S, Examene E
WHERE S.NrMatr = E.NrMatr AND E.Nota=10
```

- Observați că avem coloana *NrMatr* în ambele coloane, prin urmare trebuie specificat la ce tabel ne referim când folosim coloana.
- Care este rolul condiției `S.NrMatr = E.NrMatr` ?
  - Leagă cele două tabele pe baza cheii străine
  - Atenție! Fără această condiție obținem produsul cartezian al celor două tabele (leagă fiecare rând din *Studenti* cu fiecare rând din *Examene*)

Strategia de evaluare conceptuală

SELECT [DISTINCT] lista-select
FROM lista-from
WHERE calificare

- Calculează produsul cartezian al tabelelor din **lista-from**
- șterge din produsul cartezian rândurile care nu îndeplinesc condițiile din **calificare**
- elimină coloanele care nu apar în **lista-select**
- dacă DISTINCT e specificat, elimină rândurile duplicate (implicit, acestea din urmă nu sunt eliminate din rezultat)

Găsiți studenții care au cel puțin o notă

```
SELECT S.NrMatr
FROM Studenti S, Examene E
WHERE S.NrMatr = E.NrMatr
```

- Dacă folosim DISTINCT se schimbă rezultatul?

Găsiți studenții ai căror nume începe cu litera P

```
SELECT *
FROM Studenti
WHERE Nume LIKE 'P%’
```

- În șablon "%" reprezintă orice număr de caractere, iar "_" reprezintă un caracter.
- Ce studenți ar afișa dacă schimbăm condiția astfel:

```
WHERE Nume LIKE 'P_%N'
```

Expresii aritmetice, Coloane “noi”

```
SELECT S.varsta, varsta1 = S.varsta-5, 2*S.varsta AS varsta2
FROM Studenti S
```

- Când rezultă o coloană nouă în rezultatul interogării, de exemplu printr-o expresie aritmetică, acea coloană trebuie sa primească un nume, fie prin “**=**“, fie prin **AS** (în acest caz ambele au același rol)
- Dacă nu denumim noua coloană, în tabelul rezultat va apărea *(No colum name)* și nu o vom putea folosi mai departe pentru prelucrarea datelor.

UNION

- se poate folosi pentru a reuni rezultatul a două relații cu domeniu compatibil (același număr de coloane cu același tip de date)
- Găsiți studenții care au notă la un curs de 5 credite sau la un curs de 6 credite

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 5
UNION
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 6
```

Interogare echivalentă:

```
SELECT E.NrMatr
FROM Examene E,Cursuri C
WHERE E.CursId = C.CursId
AND (C.Credite = 5 OR
     C.Credite = 6)
```

UNION vs. UNION ALL

- UNION elimină duplicatele
- UNION ALL nu elimină duplicatele

INTERSECT

- INTERSECT = intersecția a două relații cu domeniu compatibil

- Ce se întâmplă dacă înlocuim în enunțul precedent “sau” cu “și ”?
- Găsiți studenții care au notă la un curs de 5 credite și la un curs de 6 credite (adică atât la unul cu 5 credite, cât și la unul cu 6 credite)

INTERSECT

- Găsiți studenții care au notă la un curs de 5 credite și la un curs de 6 credite

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 5
INTERSECT
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 6
```

Interogare echivalentă:

```
SELECT E1.NrMatr
FROM Cursuri C1, Examene E1,
Cursuri C2, Examene E2
WHERE E1. NrMatr = E2. NrMatr AND
    E1.CursId = C1.CursId AND
    E2.KursId = C2.KursId AND
    C1.credite = 5 AND
    C2.credite = 6
```

INTERSECT

- **Atenție pe ce coloane faceți intersecția.**
- Găsiți studenții care au notă la un curs de 5 credite și la un curs de 6 credite. E corectă intergoarea următoare?

```
SELECT S.Nume
FROM Studenti S, Examene E, Cursuri C
WHERE S.NrMatr = E.NrMatr AND E.CursId = C.CursId
AND C.Credite = 5
INTERSECT
SELECT S.Nume
FROM Studenti S, Examene E, Cursuri C
WHERE S.NrMatr = E.NrMatr AND E.CursId = C.CursId
AND C.Credite = 6
```

EXCEPT

- Găsiți studenții care s-au înscris la cursul cu titlul “Baze de date II”, dar nu s-au înscris la cursul cu titlul “Baze de date I”:

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Titlu = 'Baze de date II'
EXCEPT
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Titlu = 'Baze de date I'
```

Interogări imbricate (Nested Queries)

- O interogare poate conține o altă interogare (o subinterogare), e.g., în clauzele FROM sau WHERE.
- Operatorul **IN**: testează apartenența unei valori la o mulțime de elemente; aceasta din urmă poate fi dată explicit sau generată cu o interogare SQL.
- Operatorul **EXISTS**: testează dacă o mulțime este nevidă.
- Se pot folosi și:
  - **NOT IN**
  - **NOT EXISTS**

IN

- Găsiți toți studenții înscriși la cursul ’BD’

```
SELECT S.Nume
FROM Studenti S
WHERE S.NrMatr IN (SELECT E.NrMatr
                   FROM Examene E
                   WHERE E.CursId = 'BD’)
```

- Cum se mai poate scrie echivalent această interogare?

IN

- Atentie! Valoarea pe care o comparați cu mulțimea de valori din subinterogare trebuie sa fie de același tip atât ca și tip de date, dar și **semantic** ca să aibă sens.
- Ce afișează interogarea următoare?

```
SELECT S.Nume
FROM Studenti S
WHERE S.NrMatr IN (SELECT E.CursId
                   FROM Examene E
                   WHERE E.CursId = 'BD')
```

EXISTS

- Ce afișează următoarea interogare?

```
SELECT S.Nume
FROM Studenti S
WHERE EXISTS   (SELECT *
                FROM Examene E
                WHERE E.NrMatr = S.NrMatr
                AND E.CursId = 'BD')
```

- Atenție, de multe ori e necesar (în contextul problemei) ca subinterogarea să fie “legată” de interogarea din exterior.

ANY, ALL

- **ANY** – rezultatul expresiei este adevărat dacă respectiva condiție este adevărată pentru **cel puțin o înregistrare** din rezultatul subinterogării

- **ALL** – rezultatul expresiei este adevărat dacă respectiva condiție este adevărată pentru **toate înregistrările** din rezultatul subinterogării

Găsiți studenții care sunt mai în vârstă decât ‘Popescu’ (dacă sunt mai mulți, decât oricare dintre ei)

Găsiți studenții care sunt mai în vârstă decât ‘Popescu’ (dacă sunt mai mulți, decât oricare dintre ei)

```
SELECT *
FROM Studenti S
WHERE S.varsta > ANY (SELECT S2.varsta
                      FROM Studenti S2
                      WHERE S2.Nume='Popescu')
```

Găsiți studenții cei mai în vârstă

- E corectă interogarea?

```
SELECT *
FROM Studenti S
WHERE S.varsta > ALL (SELECT S2.varsta
                      FROM Studenti S2)
```

Găsiți studenții cei mai în vârstă

- Interogarea anterioara NU e corectă! Nu va afișa nimic ca și rezultat pentru că îl comparăm și cu el însuși si nu poate avea vârsta mai mare decât el însuși.

```
SELECT *
FROM Studenti S
WHERE S.varsta >= ALL (SELECT S2.varsta
                       FROM Studenti S2)
```

Găsiți studenții cei mai în vârstă

- Atenție la valorile pe care le comparați. Trebuie să aibă sens în contextul problemei. De exemplu nu comparăm id-uri sau chei primare, de obicei acest lucru nu are sens!

```
SELECT *
FROM Studenti S
WHERE S.NrMatr > ANY (SELECT S2.NrMatr
                      FROM Studenti S2)
```

Cum s-ar putea rescrie cu subinterogare?

- Găsiți studenții care au notă la un curs de 5 credite și la un curs de 6 credite

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 5
INTERSECT
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 6
```

Interogare echivalentă:

```
SELECT I1.NrMatr
FROM Cursuri C1, Examene E1,
Cursuri C2, Examene E2
WHERE I1. NrMatr = E2. NrMatr AND
    I1.CursId = C1.CursId AND
    I2.KursId = C2.KursId AND
    C1.credite = 5 AND
    C2.credite = 6
```

Cum s-ar putea rescrie cu subinterogare?

- Găsiți studenții care au notă la un curs de 5 credite și la un curs de 6 credite
- Interogările care folosesc INTERSECT se pot rescrie folosind IN

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Credite = 5
AND E.NrMatr IN (SELECT E.NrMatr
                 FROM Examene E, Cursuri C
                 WHERE E.CursId = C.CursId
                 AND C.Credite = 6)
```

Cum s-ar putea rescrie cu subinterogare?

- Găsiți studenții care s-au înscris la cursul cu titlul “Baze de date II”, dar nu s-au înscris la cursul cu titlul “Baze de date I”:

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Titlu = 'Baze de date II'
EXCEPT
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Titlu = 'Baze de date I'
```

Cum s-ar putea rescrie cu subinterogare?

- Găsiți studenții care s-au înscris la cursul cu titlul “Baze de date II”, dar nu s-au înscris la cursul cu titlul “Baze de date I”
- Interogările care folosesc EXCEPT se pot rescrie folosind NOT IN

```
SELECT E.NrMatr
FROM Examene E, Cursuri C
WHERE E.CursId = C.CursId
AND C.Titlu = 'Baze de date II’
AND E.NrMatr NOT IN (SELECT E.NrMatr
                     FROM Examene E, Cursuri C
                     WHERE E.CursId = C.CursId
                     AND C.Titlu = 'Baze de date I’)
```

Operatorii de Join - INNER JOIN

- Să se găsească toate notele la examene ale tuturor studenților; rezultatul va conține inclusiv numele studenților

- **join condițional**: sursa1 [alias] **[INNER] JOIN** sursa2 [alias] **ON** conditie

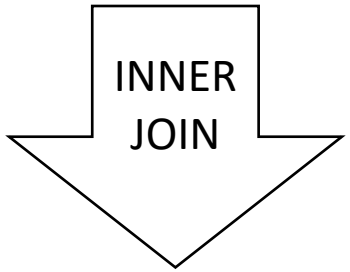
```
SELECT *
FROM Studenti S INNER JOIN
Examene E ON S.NrMatr = E.NrMatr
```

Studenti

NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512

Examene

NrMatr	CursId	Nota
pa123	BD1	10
pa123	BD2	8
ma111	BD1	9



NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP	NrMatr	CursId	Nota
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559	pa123	BD1	10
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559	pa123	BD2	8
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512	ma111	BD1	9

Operatorii de Join - LEFT JOIN

- să se găsească datele referitoare la examenele studenților; se vor include și studenții care nu au dat niciun examen; rezultatul va conține inclusiv numele studenților

- **join extern stânga**: sursa1 [alias] **LEFT [OUTER] JOIN** sursa2 [alias] **ON** conditie

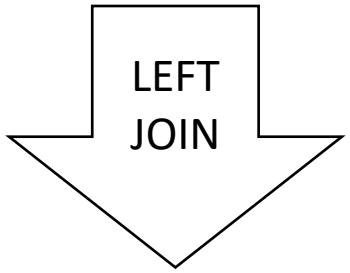
```
SELECT *
FROM Studenti S LEFT JOIN
Examene E ON S.NrMatr = E.NrMatr
```

Studenti

NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512

Examene

NrMatr	CursId	Nota
pa123	BD1	10
pa123	BD2	8
ma111	BD1	9



NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP	NrMatr	CursId	Nota
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559	pa123	BD1	10
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559	pa123	BD2	8
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512	ma111	BD1	9
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559	NULL	NULL	NULL

Operatorii de Join - RIGHT JOIN

- să se găsească datele referitoare la examenele disciplinelor; se vor include și disciplinele la care nu s-a dat niciun examen; rezultatul va conține inclusiv denumirile disciplinelor
- **join extern dreapta**: sursa1 [alias] **RIGHT [OUTER] JOIN** sursa2 [alias] **ON** conditie

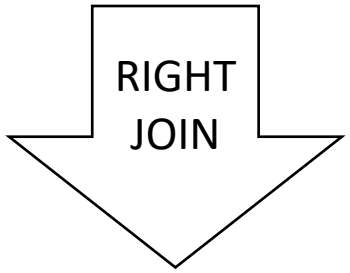
```
SELECT *
FROM Examene E RIGHT JOIN
Cursuri C ON E.cursId = C.cursId
```

Examene

NrMatr	CursId	Nota
pa123	BD1	10
pa123	BD2	8
ma111	BD1	9

Cursuri

CursId	Titlu	Credite
BD1	Baze de date I	5
BD2	Baze de date II	6
ALG	Algebra	4



NrMatr	CursId	Nota	CursId	Titlu	Credite
pa123	BD1	10	BD1	Baze de date I	5
pa123	BD2	8	BD2	Baze de date II	6
ma111	BD1	9	BD1	Baze de date I	5
NULL	NULL	NULL	ALG	Algebra	4

Operatorii de Join - FULL JOIN

- să se găsească datele referitoare la examene; se vor include studenții care nu au dat niciun examen și cursurile la care nu a dat nimeni examen; rezultatul va conține inclusiv numele studenților și numele cursurilor
- **join complet**: sursa1 [alias] **FULL [OUTER] JOIN** sursa2 [alias] **ON** conditie

```
SELECT *
FROM Studenti S FULL JOIN
Examene E ON S.NrMatr = E.NrMatr FULL JOIN
Cursuri C ON C.CursId=E.CursId
```

Studenti

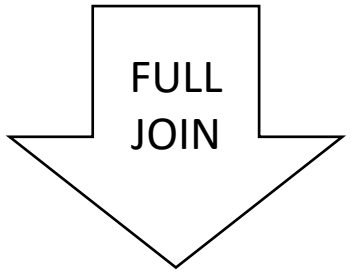
NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512

Examene

NrMatr	CursId	Nota
pa123	BD1	10
pa123	BD2	8
ma111	BD1	9

Cursuri

CursId	Titlu	Credite
BD1	Baze de date I	5
BD2	Baze de date II	6
ALG	Algebra	4



NrMatr	Nume	Prenume	[...](Email, Varsta, Grupa, CNP)	NrMatr	CursId	Nota	CursId	Titlu	Credite
pa123	Pop	Ana	...	pa123	BD1	10	BD1	Baze de date I	5
pa123	Pop	Ana	...	pa123	BD2	8	BD2	Baze de date II	6
ma111	Matei	Andrei	...	ma111	BD1	9	BD1	Baze de date I	5
ha345	Hulea	Ioana	...	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	ALG	Algebra	4

Operatorii de Join

- Observație. Următoarele două interogări produc același rezultat:

```
SELECT *
FROM Studenti S INNER JOIN
Examene E ON S.NrMatr = E.NrMatr


SELECT *
FROM Studenti S, Examene E
WHERE S.NrMatr = E.NrMatr
```

An SQL query walks into a bar.

He approaches two tables and says:

"*Mind if I join you*?"

Operatori de agregare

- se evaluează pentru o mulțime de valori, corespunzătoare unui grup de înregistrări
- **SUM (X)** → 12
- **AVG(X)** → 3
- **MAX(X)** → 6
- **MIN(X)** → 1
- **COUNT(X)** → 4
- **Eliminare de duplicate: COUNT(DISTINCT X)** → 3
- **Tratarea valorilor nule: COUNT(X)** numără doar valorile nenule

X
1
1
4
6

Operatori de agregare

- Găsiți vârsta medie a tuturor studenților

```
SELECT AVG(S.Varsta)
FROM Studenti S
```

Operatori de agregare

- Găsiți vârsta minimă și numărul studenților din grupa 711

```
SELECT MIN(S.Varsta), COUNT(*)
FROM Studenti  S
WHERE S.Grupa = 711
```

NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512

(No column name)	(No column name)
20	2

Operatori de agregare

```
SELECT S.Nume, MAX(S.Varsta)
FROM Studenti S
```

→ **Eroare**! Dacă clauza SELECT conține un operator de agregare, atunci trebuie să conțină **doar** operatori de agregare, exceptând cazul în care interogarea are și o clauză GROUP BY.

- Dacă dorim să știm numele studenților cu vârsta maximă:

```
SELECT S.Nume, S.Varsta
FROM Studenti S
WHERE S.Varsta = (SELECT MAX(S2.Varsta)
                  FROM Studenti S2)
```

GROUP BY, HAVING

- Găsiți vârsta cea mai mică din fiecare grupă de studenți

- Probleme:
  - Nu știm câte grupe sunt sau care sunt aceste grupe
  - Trebuie să funcționeze la modul general, nu doar pentru grupele existente în baza de date în acest moment

GROUP BY, HAVING

```
SELECT [DISTINCT] lista-select
FROM lista-from
WHERE calificare
GROUP BY lista-grupare
HAVING calificare-grup
```

- clauza opțională GROUP BY – coloane folosite în grupare
- clauza opțională HAVING – condiții de calificare pentru grupuri
- grup – o colecție de rânduri care au aceleași valori pentru toate coloanele din **lista-grupare**
- fiecare rând din rezultatul interogării corespunde unui grup

GROUP BY, HAVING

- **lista-select** conține:
  - coloane care trebuie să apară și în **lista-grupare (pentru alte coloane dă eroare)**;
  - termeni cu operatori de agregare, e.g.,

    opagregare(coloana) [AS NumeNou]

    MAX(S.Varsta) AS VarstaMax
- *NumeNou* dă coloanei respective un nume în tabelul-rezultat al interogării
- expresiile care apar în **calificare-grup** în clauza HAVING trebuie să aibă o singură valoare pentru fiecare grup (o coloana din **calificare-grup** apare fie în **lista-grupare**, fie ca argument pentru un operator de agregare)

GROUP BY, HAVING

- Evaluare conceptuală:
  - Calculează produsul cartezian al tabelelor din **lista-from**
  - Șterge din produsul cartezian rândurile care nu îndeplinesc condițiile din **calificare**
  - Rândurile rămase se împart în grupuri care au aceleași valori pentru atributele din **lista-grupare**
  - Se șterg grupurile care nu îndeplinesc condițiile din **calificare-grup**
  - Pentru fiecare grup rămas se va scrie o înregistrarea în rezultat (din acest motiv pot apărea în **lista-select** doar atribute care au o valoarea unică pentru un grup)

GROUP BY, HAVING, ORDER

```
SELECT [DISTINCT] lista-select
FROM lista-from
WHERE calificare
GROUP BY lista-grupare
HAVING calificare-grup
ORDER BY lista-atribute[ASC |DESC]
```

Găsiți vârsta cea mai mică din fiecare grupă de studenți

```
SELECT S.Grupa, MIN(Varsta) as VarstaMin
FROM Studenti S
GROUP BY S.Grupa
```

Găsiți cea mai mică vârsta peste 20 ani din fiecare grupă de studenți

```
SELECT S.Grupa, MIN(Varsta) as VarstaMin
FROM Studenti S
WHERE S.varsta >= 20
GROUP BY S.Grupa
```

Găsiți cea mai mică vârsta peste 20 ani din fiecare grupă de studenți cu cel puțin doi studenți

```
SELECT S.Grupa, MIN(Varsta) as VarstaMin
FROM Studenti S
WHERE S.varsta >= 20
GROUP BY S.Grupa
HAVING COUNT(*) >= 2
```

Găsiți numărul de studenți înscriși și media notelor pentru toate cursurile de 6 credite

```
SELECT C.CursId, COUNT(*) as Nr, AVG(Nota) as Medie
FROM Cursuri C INNER JOIN Examene E
ON C.CursId = E.CursId
WHERE C.Credite = 6
GROUP BY C.CursId
```

* Găsiți numărul de studenți din fiecare grupă cu mai puțini studenți decât grupa 713

```
SELECT S.Grupa, COUNT(*) as NrStudenti
FROM Studenti S
GROUP BY S.Grupa
HAVING COUNT(*) < (SELECT COUNT(*)
                   FROM Studenti S2
                   WHERE S2.Grupa = 713)
```

ORDER BY, TOP

- TOP (expresie) [PERCENT] [WITH TIES]

```
SELECT TOP(1) WITH TIES E.NrMatr
FROM Examene E
WHERE E.CursID='BD1'
ORDER BY E.Nota DESC
```

- `WITH TIES ->` dacă setați această opțiune și dacă numărul/procentul de întrerupere se încadrează în mijlocul unui set de rânduri cu valori identice în clauza ORDER BY, vizualizarea este extinsă pentru a include toate rândurile cu valori identice
- De obicei TOP se folosește împreună cu ORDER BY pentru a ști după ce criteriu au fost alese “primele” înregistrări

Găsiți primii 25% dintre studenți, ordonându-i crescător după nume

```
SELECT TOP 25 PERCENT *
FROM Studenti S
ORDER BY S.Nume ASC
```

Exemplu subinterogare în clauza FROM

Ce afișează intergoarea următoare?

```
SELECT E.Nota
FROM Examene E INNER JOIN
     (SELECT *
      FROM Cursuri C
      WHERE C.Tip = 'Optional') T
ON E.CursId = T.CursId
```

Valoarea NULL

- În anumite situații valorile particulare ale unor attribute (câmpuri) pot fi *necunoscute* sau *inaplicabile* temporar.
- SQL permite utilizarea unei valori speciale *null* pentru astfel de situații.
- Prezența valorii *null* implică unele probleme suplimentare:
  - E necesară implementarea unei logici cu 3 valori: *true*, *false*și *null*(de exemplu o condiție de tipul *rating>8* va fi intotdeauna evaluată cu *false* daca valoarea câmpului rating este *null*)
- E necesară adaugarea unui operator special IS NULL / IS NOT NULL.

Găsiți toate examenele cu note între 8 și 10

```
SELECT *
FROM Examene
WHERE Nota IS NOT NULL AND
Nota BETWEEN 8 AND 10
```

Now we are all SQL experts!

- Cât de bine știți SQL?
- Testați-vă cunoștințele cu un joc: SQL Murder Mystery!
- https://mystery.knightlab.com/walkthrough.html

- Pentru mai multe exerciții interactive:
- https://sqlzoo.net/

Exerciții

Studenti

NrMatr	Nume	Prenume	Email	Varsta	Grupa	CNP
pa123	Pop	Ana	ana@ubbcluj.ro	20	711	6200513017559
ha345	Hulea	Ioana	ioana@ubbcluj.ro	23	712	6200513014559
ma111	Matei	Andrei	andrei@ubbcluj.ro	25	711	6200513017512

Cursuri

CursId	Titlu	Credite
BD1	Baze de date I	5
BD2	Baze de date II	6
ALG	Algebra	4

Examene

NrMatr	CursId	Nota
pa123	BD1	10
pa123	BD2	8
ma111	BD1	9

Găsiți toți studenții din grupa 331

```
SELECT NrMatr     -- SELECT *
FROM Studenti
WHERE Grupa = 331
```

Găsiți toți studenții din grupa 331 sau 332

```
SELECT *
FROM Studenti
WHERE Grupa = 331 OR Grupa = 332
```

Găsiți toți studenții a căror nume începe cu “Po”

```
SELECT *
FROM Studenti
WHERE Nume LIKE ‘Po%’
```

- Sa se termine cu “an”

```
SELECT *
FROM Studenti
WHERE Nume LIKE ‘%an’
```

Găsiți toți studenții care au luat atât note de 10, cât și note de 8

```
SELECT *
FROM Studenti S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
WHERE E.Nota = 10
INTERSECT
SELECT *
FROM Studenti S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
WHERE E.Nota = 8
```

Găsiți toți studenții care au luat atât note de 10, cât și note de 8

```
SELECT *
FROM Studenti S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
WHERE E.Nota = 10
AND S.NrMatr IN( SELECT S.NrMatr
                 FROM Studenti S INNER JOIN Examene E
                 ON S.NrMatr = E.NrMatr
                 WHERE E.Nota = 8)
```

Găsiți toți studenții care au luat atât note de 10, cât și note de 8

```
SELECT *
FROM Studenti S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
WHERE E.Nota = 10
AND S.NrMatr IN( SELECT E.NrMatr
                 FROM Examene E
                 WHERE E.Nota = 8)
```

Găsiți toți studenții mai în vârstă decât orice alt student din grupa 711

```
SELECT *
FROM Studenti S
WHERE S.Varsta > ( SELECT MAX(S2.Varsta)
                   FROM Studenti S2
                   WHERE S2.Grupa = 711)

SELECT *
FROM Studenti S
WHERE S.Varsta > ALL (SELECT S2.Varsta
                      FROM Studenti S2
                      WHERE S2.Grupa = 711)
```

Găsiți email-urile studenților înscriși la cursul de “BD” – 3 Metode

```
SELECT S.Email
FROM Student S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
WHERE E.CursId = ‘BD’

SELECT S.Email
FROM Student S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
INNER JOIN Cursuri C
ON C.CursId = E.CursID
WHERE C.Titlu = ‘Baze de date’
```

Găsiți email-urile studenților înscriși la cursul de “BD” – 3 Metode

```
SELECT S.Email
FROM Student S
WHERE S.NrMatr IN (SELECT E.NrMatr
                   FROM Examene E
                   WHERE E.CursId = ‘BD’)

SELECT S.Email
FROM Student S
WHERE EXISTS (SELECT E.NrMatr
              FROM Examene E
              WHERE E.CursId = ‘BD’
              AND S.NrMatr = E.NrMatr)
```

Găsiți email-urile studenților, notele și cursurile la care au luat studenții aceste note

```
SELECT S.Email, E.Nota, C.Titlu
FROM Studenti S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
INNER JOIN Cursuri C
ON E.CursId = C.CursId
```

Găsiți numărul studenților din grupa 712 a căror nume începe cu litera “A”

```
SELECT COUNT(*)AS NrStudenti
FROM Studenti S
WHERE S.Grupa = 712
AND S.Nume LIKE ‘A%’
```

Pentru fiecare student înscris la cel puțin două cursuri afișați numărul de cursuri la care el e înscris

```
SELECT E.NrMatr, S.Nume, COUNT(*) AS NrCursuri
FROM Examene E INNER JOIN Studenti S
ON E.NrMatr = S.NrMatr
--eroare: WHERE COUNT(*) >= 2
GROUP BY E.NrMatr, S.Nume
HAVING COUNT(*) >= 2
```

Găsiți studenții cu nota cea mai mare la “BD” (dacă sunt mai mulți, atunci îi afișăm pe toți)

```
SELECT TOP 1 WITH TIES *
FROM Studenti S INNER JOIN Examene E
ON S.NrMatr = E.NrMatr
ORDER BY E.Nota DESC
```

Găsiți pentru fiecare curs obligatoriu numărul de studenți înscriși la acel curs
+ doar pentru cursurile cu cel putin 10 studenti

```
SELECT [DISTINCT] E.CursId, COUNT(E.NrMatr) AS
NrStudenti
FROM Examene E INNER JOIN Cursuri C
ON E.CursId = C.CursId
WHERE C.Tip = ‘obligatoriu’
GROUP BY E.CursId
HAVING COUNT(E.NrMatr) > 10
```