

# Modelul Relational

2

# Modele de date

- Modelul ierarhic (1965)
- Modelul rețea (1965)
- **Modelul relațional (1NF) (1970s)**
- Model relațional imbricat (1970s)
- Obiecte complexe (1980s)
- Model obiectual (1980)
- Model relațional-obiectual (1990s)
- XML (DTD), XML Schema (1990s)

# Model relațional - idei

- Utilizează o structură de date simplă: *Tabela*
  - simplu de înțeles
  - utilă în modelarea multor situații/entități din lumea reală
  - conduc la interogări de o complexitate redusă
- Utilizează matematica în descrierea/reprezentarea înregistrărilor și a colecțiilor de înregistrări: *Relația*
  - pot fi modelate formal
  - permit utilizarea de limbaje de interogare formale
  - au proprietăți ce pot fi modelate și demonstrate matematic

# Relația – definiție formală

- O **relație** sau **structura unei relații R** este o listă de **nume de attribute**  $[A_1, A_2, \dots, A_n]$ .
- **Domeniu** = mulțime de valori scalare (tipuri atomice - întreg, text, dată, etc)
- $D_i = \text{Dom}(A_i)$  - domeniul lui  $A_i$ ,  $i=1..n$
- **Instanța unei relații** ( $[R]$ ) e o submulțime a  $D_1 \times D_2 \times \dots \times D_n$

# Relația – definiție formală

- **Grad (aritate)** = numărul tuturor atributelor din structura unei relații
- **Tuplu** = un element al instanței unei relații, o înregistrare. Toate tuplurile unei relații sunt distincte!
- **Cardinalitate** = numărul tupluri unei relații

# Exemplu de relație

- Students(sid:integer; name:string; email:string; age:integer; gr:integer)

*field name*

*field type  
(domain)*

sid	name	email	age	gr
2833	Jones	<a href="mailto:jones@scs.ubbcluj.ro">jones@scs.ubbcluj.ro</a>	19	231
2877	Smith	<a href="mailto:smith@scs.ubbcluj.ro">smith@scs.ubbcluj.ro</a>	20	232
2976	Jones	<a href="mailto:jones@math.ubbcluj.ro">jones@math.ubbcluj.ro</a>	21	233
2765	Mary	<a href="mailto:mary@math.ubbcluj.ro">mary@math.ubbcluj.ro</a>	22	233

*relation  
schema*

*relation  
instance*

*relation  
tuple*

- cardinalitate = 4, grad = 5, toate tuplurile distincte !

# Baze de date relaționale

- O **bază de date** este o mulțime de relații
- **Structura** unei baze de date este mulțimea structurilor relațiilor acesteia
- **Instanța (starea)** unei baze de date este mulțimea instanțelor relațiilor acesteia

# Reprezentarea grafică a relațiilor






Students(*sid*:string, *name*:string, *email*:string, *age*:integer, *gr*:integer)




Courses(*cid*: string, *cname*: string, *credits*:integer)




Enrolled(*sid*:string, *cid*:string, *grade*:double)



Teachers(*tid*:integer; *name*: string; *sal* : integer)




Teaches(*tid*:integer; *cid*:string)

Students	
	sid
	name
	email
	age
	gr

Courses	
	cid
	cname
	credits

Teachers	
	tid
	name
	sal

Teaches	
	tid
	cid

Enrolled	
	sid
	cid
	grade



# Constrângeri de integritate (CI)

- **CI**: sunt condiții ce trebuie să fie îndeplinite de către *orice* instanță a unei baze de date
  - specificate la momentul definirii structurii relației
  - verificate la modificarea conținutului relației
- O instanță a unei relații că este *legală* dacă satisface toate CI specificate
  - SGBD nu va permite instanțe *ilegale*

# Constrângeri de integritate - exemple

- Students(*sid:string, name:string, email:string, age:integer, gr:integer*)
  - Constrângere de domeniu: *gr:integer*
  - Constrângere de interval:  $18 \leq \textit{age} \leq 70$
- TestResults(*sid:string, TotalQuestions:integer, NotAnswered:integer, CorrectAnswers:integer, WrongAnswers:integer*)
  - $\textit{TotalQuestions} = \textit{NotAnswered} + \textit{CorrectAnswers} + \textit{WrongAnswers}$  – **nu e o CI!**

# Chei Primare

- O mulțime de attribute reprezintă o **cheie** a unei relații dacă:
  1. Nu există două tuple care au aceleași valori pentru toate attributele

**ȘI**

  2. Acest lucru nu este adevărat pentru nici o submulțime a cheii
- Dacă a 2-a afirmație este falsă → **super cheie**
- Dacă există >1 cheie pentru o relație → **chei candidat**
- Una dintre cheile candidat este selectată ca **cheie primară**

## Chei străine (externe)

- O **cheie străină** (**externă**) este o mulțime de câmpuri a unei relații utilizate pentru a `referi` un tuplu al unei alte relații (un fel de `pointer logic`).
  - Aceasta trebuie să corespundă cheii primare din a doua relație.

De exemplu pentru

*Enrolled* (*sid*: string, *cid*: string, *grade*: double)

*sid* este cheie externă referind *Students*

# Integritate referențială

■ **Integritate referențială** = nu sunt permise valori pentru cheia străină care nu se regăsesc în tabela referită.

Exemplu de model de date fără integritate referențială:

*Link-uri HTML*



# Integritate referențială

- Fie *Students* și *Enrolled*; *sid* in *Enrolled* este o cheie străină ce referă o înregistrări din *Students*.
- Adaugarea in *Enrolled* a unui tuplu cu un id de student inexistent, acesta va fi respins de SGBD.

*Enrolled*

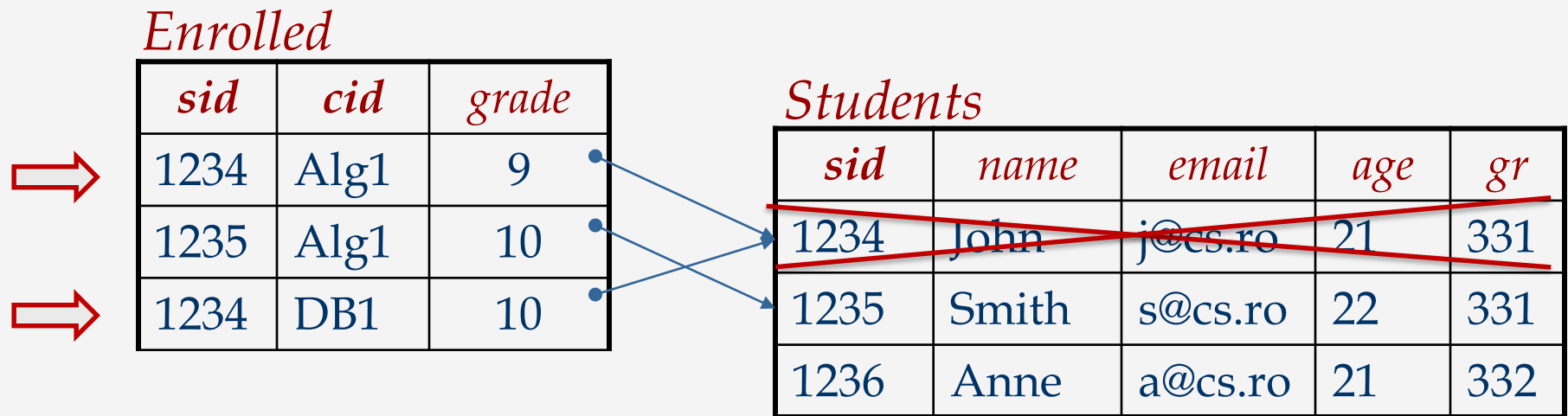
<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
<del>1237</del>	<del>DB2</del>	<del>9</del>

*Students*

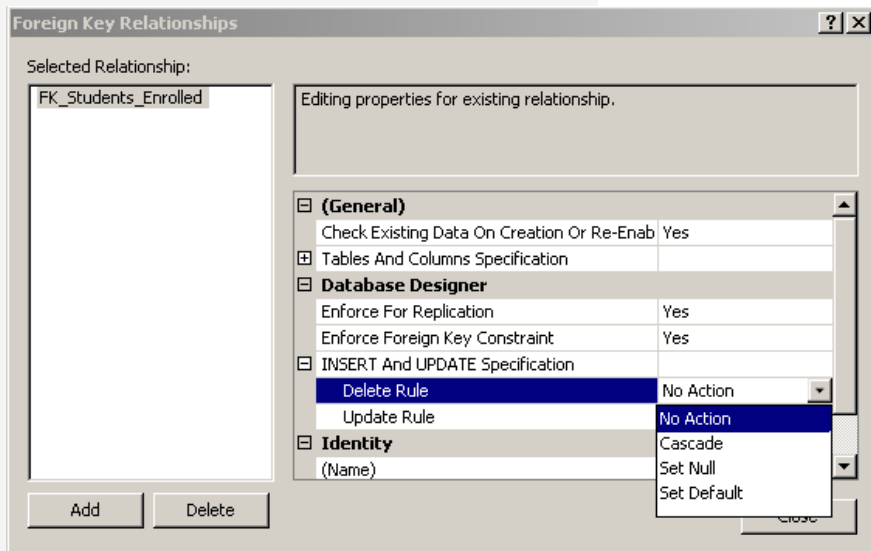
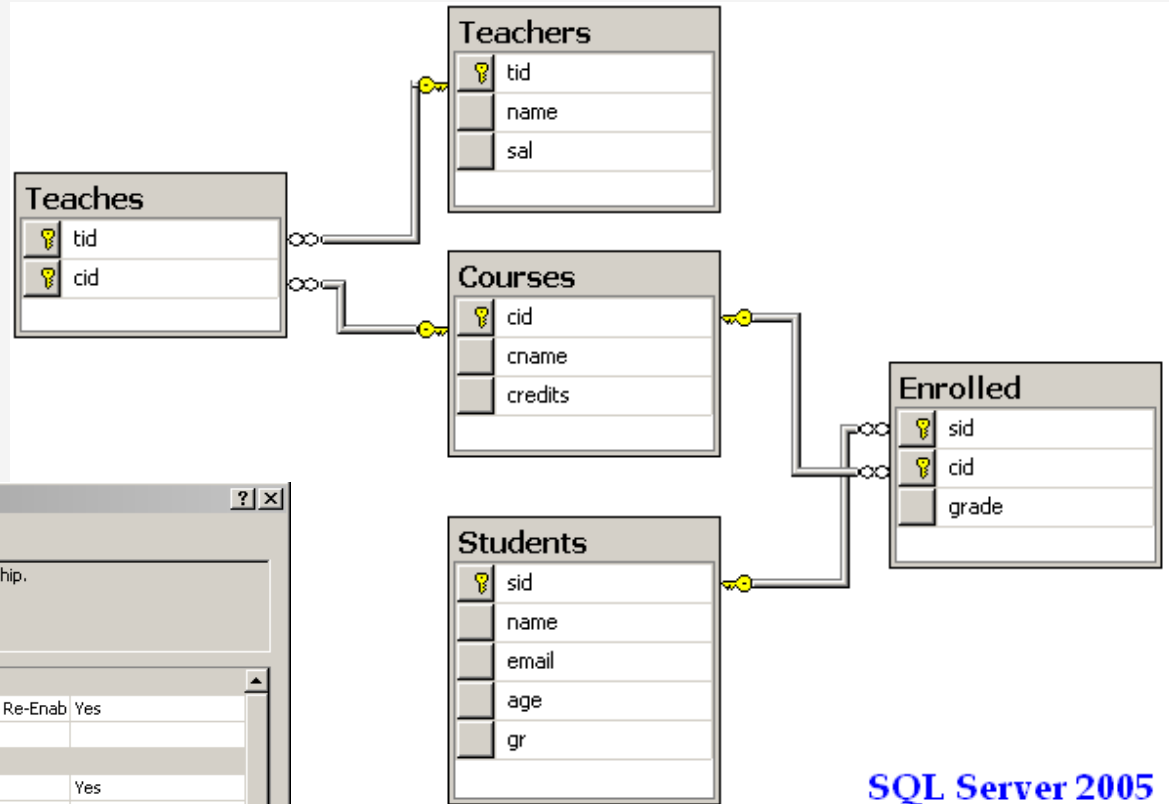
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

# Integritate referențială

- Dacă o înregistrare din *Students* este ștearsă dar ea este referită din *Enrolled*:
  - se șterg toate înregistrările ce o refera din *Enrolled*.
  - nu se permite ștergerea înregistrării din *Students*
  - sid din *Enrolled* va avea asignată o valoare implicită.
  - sid din *Enrolled* va avea asignată valoarea *null*.



# Reprezentarea grafică a CI



SQL Server 2005

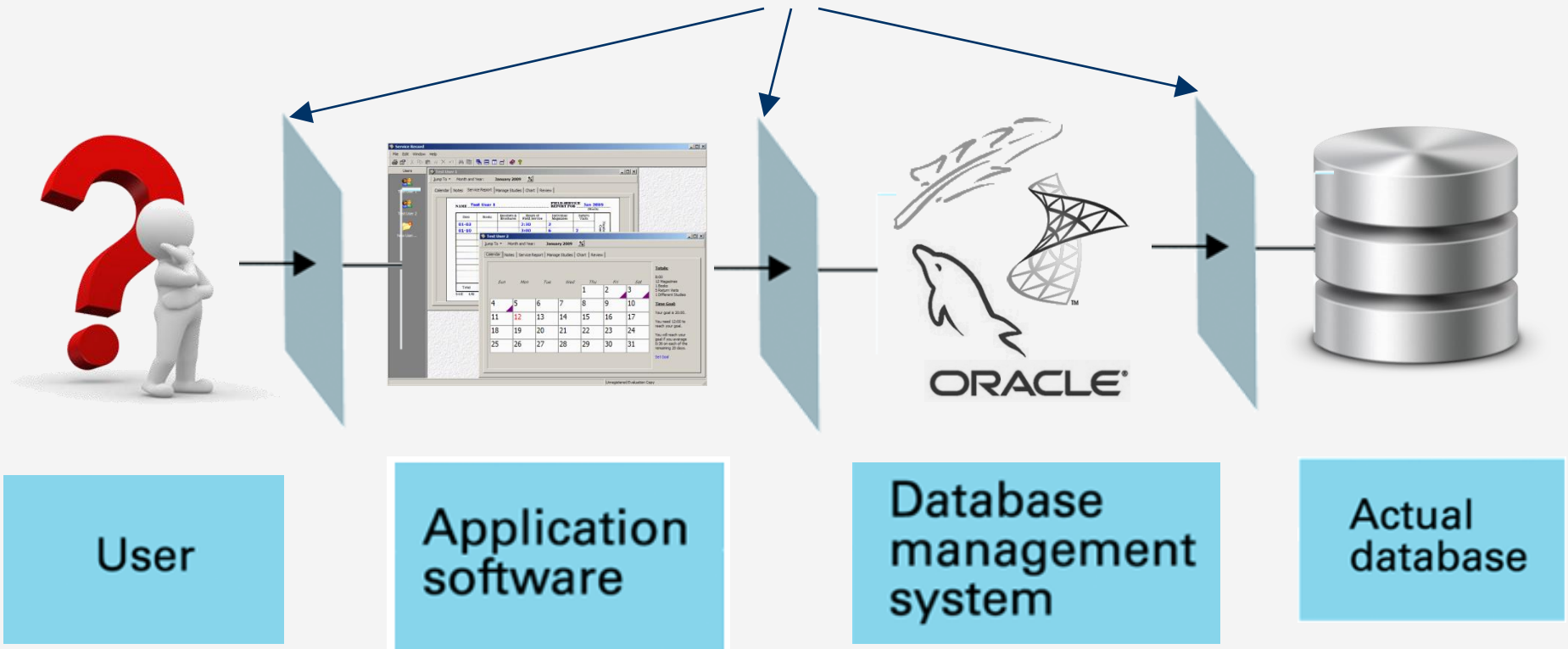


# Cum apar CI?

- CI se bazează pe semantica entităților din lumea reală / conceptuală modelate.
- Putem verifica dacă o CI este încălcată de instanța unei tabele, însă **NU** vom putea deduce dacă o CI este adevărată doar consultând o singură instanță.
  - O CI se referă la *toate instanțele* posibile ale unei tabele
- Cheile primare și externe sunt cele mai comune CI;

# Nivelele de abstractizare

Nivele diferite  
de abstractizare



**STUDENT**

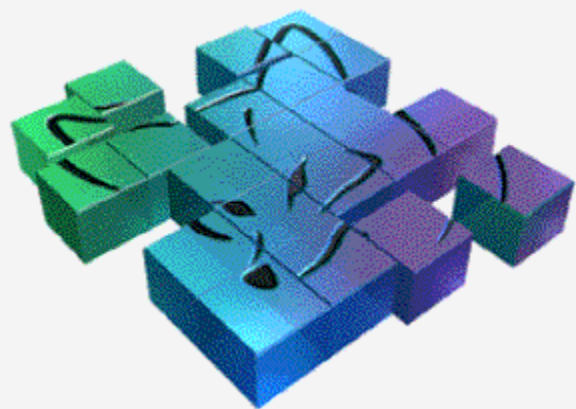
Name

Date of birth

Sex

CNP

Group



# Structura fizică

Faculty.dbc

42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

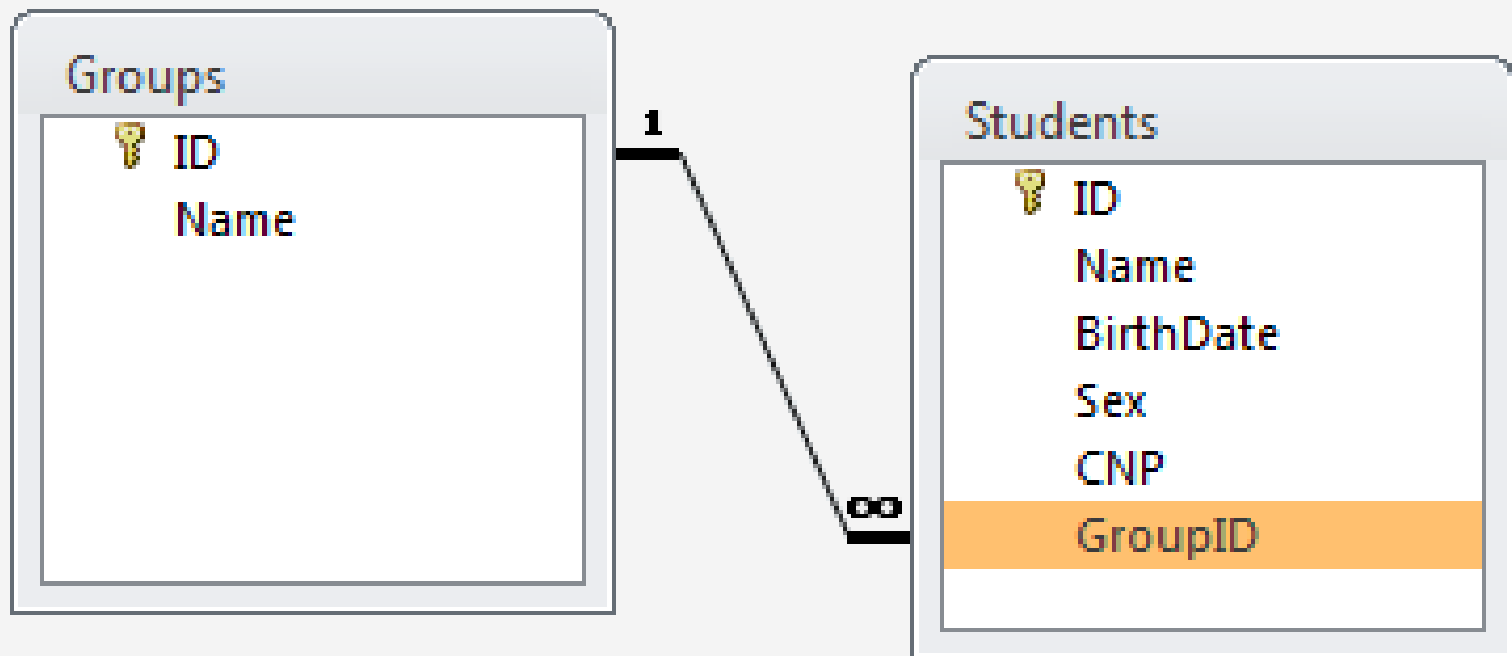
Students.dbf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

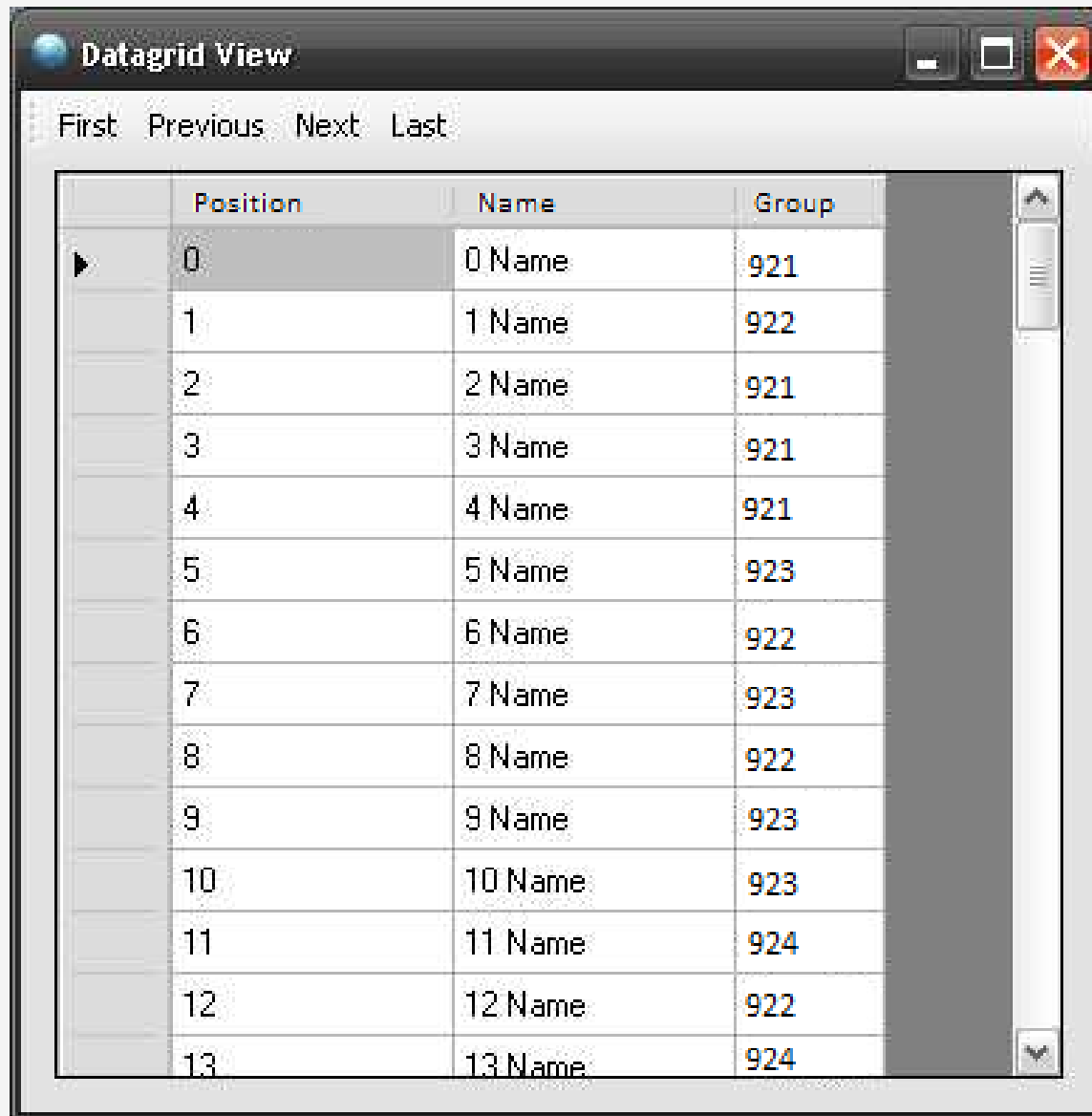
Groups.dbf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

# Structura conceptuală



# Vizualizare pentru utilizator



The image shows a software window titled "Datagrid View". Inside the window, there is a table with the following data:

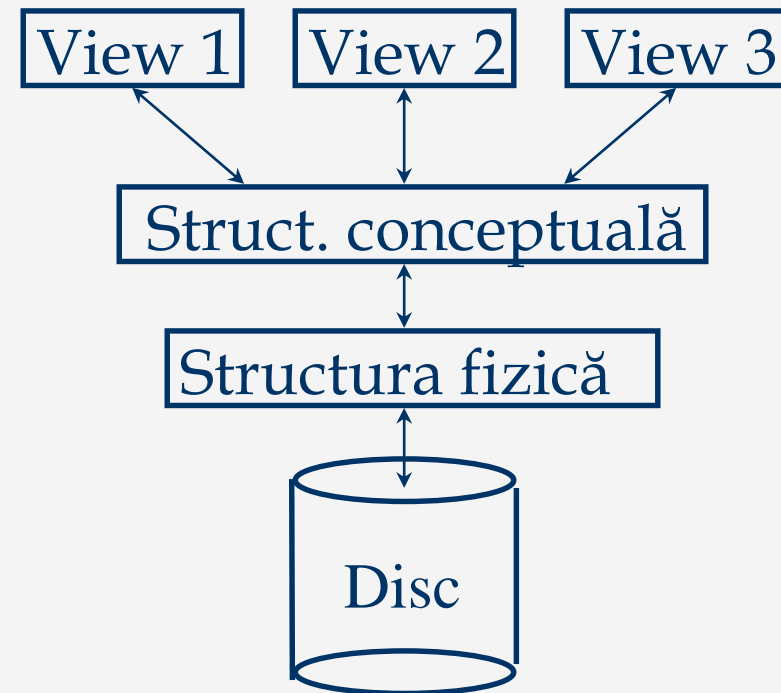
	Position	Name	Group
▶	0	0 Name	921
	1	1 Name	922
	2	2 Name	921
	3	3 Name	921
	4	4 Name	921
	5	5 Name	923
	6	6 Name	922
	7	7 Name	923
	8	8 Name	922
	9	9 Name	923
	10	10 Name	923
	11	11 Name	924
	12	12 Name	922
	13	13 Name	924

Navigation buttons: First, Previous, Next, Last.

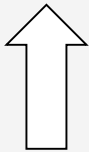
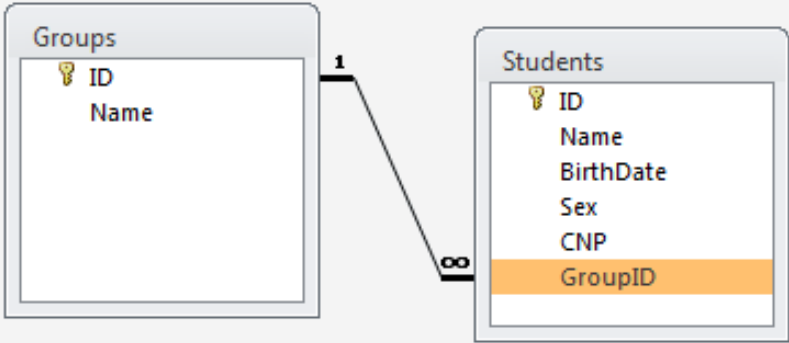
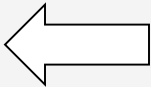
# Nivelele de abstractizare

- Mai multe structuri externe (views), câte o singură structură conceptuală (logică) și o structură fizică (internă).

- *Views* – cum văd utilizatorii datele.
- *Conceptual* - modelul logic compus din relații, attribute, etc
- *Fizic* - fișierele de date și indecși



Datagrid View			
First Previous Next Last			
	Position	Name	Group
0		0 Name	921
1		1 Name	922
2		2 Name	921
3		3 Name	921
4		4 Name	921
5		5 Name	923
6		6 Name	922
7		7 Name	923
8		8 Name	922
9		9 Name	923
10		10 Name	923
11		11 Name	924
12		12 Name	922
13		13 Name	924



Faculty.dbc

42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

Students.dbf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

Groups.dbf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

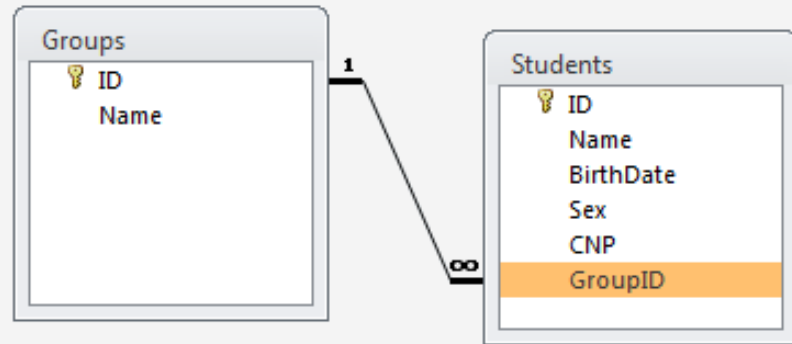
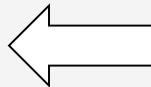


# Independența fizică a datelor

Datagrid View

First Previous Next Last

	Position	Name	Group
0		0 Name	921
1		1 Name	922
2		2 Name	921
3		3 Name	921
4		4 Name	921
5		5 Name	923
6		6 Name	922
7		7 Name	923
8		8 Name	922
9		9 Name	923
10		10 Name	923
11		11 Name	924
12		12 Name	922
13		13 Name	924



Faculty.mdf

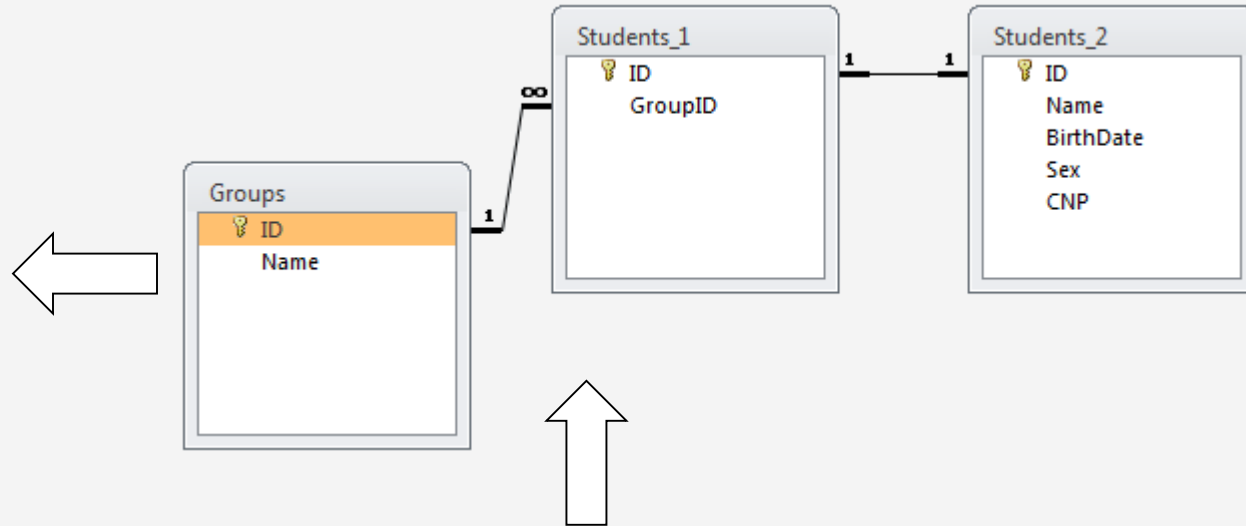
20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

# Independența logică a datelor

Datagrid View

First Previous Next Last

	Position	Name	Group
▶	0	0 Name	921
	1	1 Name	922
	2	2 Name	921
	3	3 Name	921
	4	4 Name	921
	5	5 Name	923
	6	6 Name	922
	7	7 Name	923
	8	8 Name	922
	9	9 Name	923
	10	10 Name	923
	11	11 Name	924
	12	12 Name	922
	13	13 Name	924



Faculty.mdf

20	20	20	31	56	31	2e	30
38	31	39	32	44	65	66	61
61	67	65	20	53	65	74	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	00	ff	01	00	7c	80	00
45	41	44	45	52	34	0f	53
4e	55	4d	42	45	52	14	34
34	21	0a	20	20	20	20	20
42	53	54	42	20	2e	30	37
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20
03	50	81	01	f0	06	4f	52
4c	45	2d	49	44	01	14	3c
54	2d	4e	41	4d	45	01	13
45	54	2d	4e	55	4d	42	45

# Interogări

- Posibile informații pe care dorim să le obținem din baza de date anterioară (*Faculty Database*) :
  - Care este numele studentului cu *sid* = 2833?
  - Care este salariul profesorilor care predau cursul *Alg100*?
  - Câți studenți sunt înscriși 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**
  - Influențează **structura fizică** (în anumite SGBD-uri)
- *Data Manipulation Language (DML)*
  - Operații aplicate instanțelor unei baze de date
  - DML procedural (**cum?**) vs. DML declarative (**ce?**)
- *Limbaaj gazdă*
  - Limbaj de programare obișnuit ce permite utilizatorilor să includă comenzi DML în propriul cod

# Limbaje de interogare pentru BD relaționale

SQL (Structured Query Language)

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

Algebra

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

Domain Calculus

$\{ \langle X \rangle \mid \exists V \exists Y \exists Z \exists T : Students(V, X, Y, Z, T) \wedge Z > 20 \}$

T-uple Calculus

$\{ X \mid \exists Y : Y \in Students \wedge Y.age > 20 \wedge X.name = Y.name \}$

# Structured Query Language (SQL)

- Dezvoltat de IBM (*system R*) în anii 1970
- Ulterior a apărut nevoia de standardizare
- Standarde (ANSI):
  - SQL-86
  - SQL-89 (minor revision)
  - SQL-92 (major revision) - *1,120 pagini*
  - SQL-99 (major extensions) - *2,084 pagini*
  - SQL-2003 (secțiuni SQL/XML) - *3,606 pagini*
  - SQL-2006
  - SQL-2008
  - SQL-2011






# Nivele SQL




- Data-definition language (DDL):
  - Creare / stergere / modificare *tabele* și *views*.
  - Definire *constrangeri de integritate* (CI's).
- Data-manipulation language (DML)
  - Permit formularea de interogari
  - Inserare / ștergere / modificare înregistrări.
- *Controlul accesului:*
  - Asignează sau elimină drepturi de acces si modificare a *tabelelor* și a *view-urilor*.

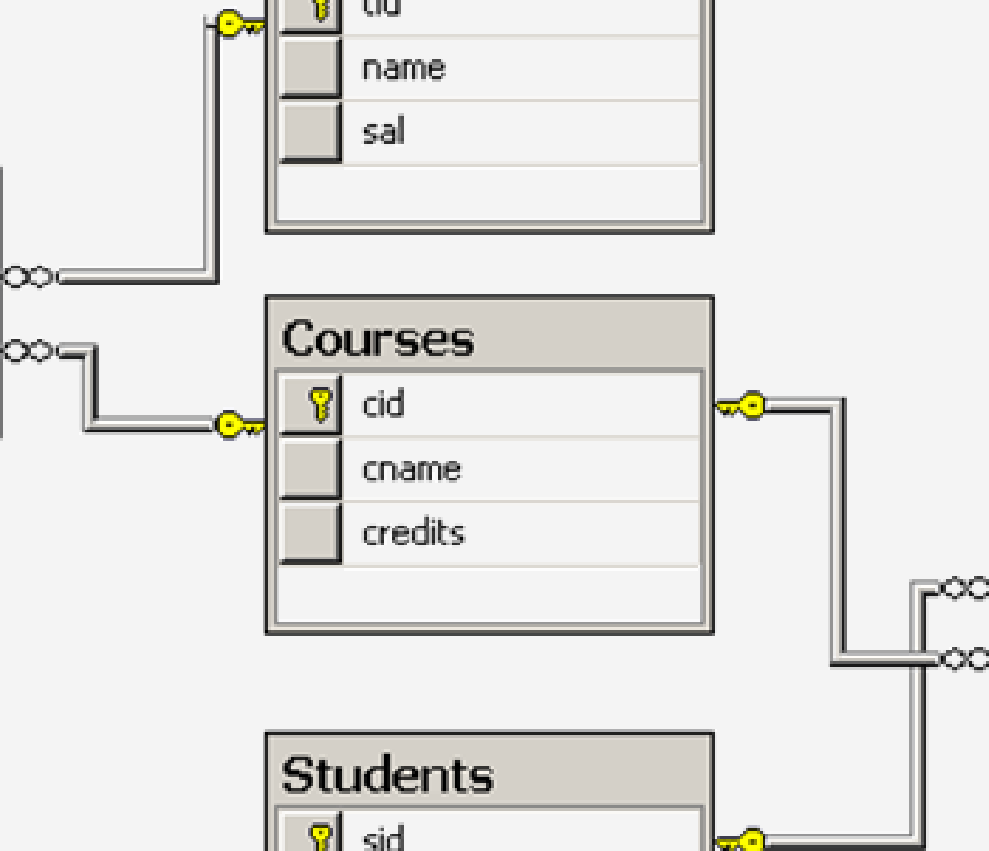
Teaches	
 tid	
 cid	

Teachers	
 tid	
 name	
 sal	

Courses	
 cid	
 cname	
 credits	

Students	
 sid	
 name	
 email	
 age	
 gr	

Enrolled	
 sid	
 cid	
 grade	





## *Students*

<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

## *Enrolled*

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9

## *Courses*

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

# SELECT

Studentii cu vârsta de 21 de ani:

```
SELECT *  
FROM Students S  
WHERE S.age = 21
```

<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1236	Anne	a@cs.ro	21	332

Returnează doar numele și adresele de e-mail:

```
SELECT S.name, S.email  
FROM Students S  
WHERE S.age = 21
```

<i>name</i>	<i>email</i>
John	j@cs.ro
Anne	a@cs.ro

# Interogare SQL simplă

```
SELECT [DISTINCT] target-list  
FROM relation-list  
WHERE qualification
```

- *relation-list* - lista de nume de relații/tabele.
- *target-list* - listă de attribute ale relațiilor din  
*relation-list*
- *qualification* - comparații logice (*Attr op const* sau *Attr1 op Attr2*, unde *op* is one of  $<$ ,  $>$ ,  $=$ ,  $\leq$ ,  $\geq$ ,  $\neq$ ) combinate cu AND, OR sau NOT.
- *DISTINCT* (optional) - indică faptul că rezultatul final nu conține duplicate.

# Evaluare conceptuală

```
SELECT [DISTINCT] target-list  
FROM relation-list  
WHERE qualification
```

- Calcul produs cartezian al tabelelor din *relation-list*.
- Filtrare înregistrări ce nu verifică *qualifications*.
- Ștergere attribute ce nu aparțin *target-list*.
- Dacă **DISTINCT** e prezent, se elimină înregistrările duplicate.

1. PRODUS CARTEZIAN

2. ELIMINA LINII

3. ELIMINA COLOANE

4. ELIMINA DUPLICATE

Această strategie e **doar**  
la nivel *conceptual*!

Modul actual de evaluare  
a unei interogări  
e **mult** optimizat

# Range Variables

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid AND E.grade=10
```



```
SELECT name, cid  
FROM Students, Enrolled  
WHERE Students.sid=Enrolled.sid  
AND grade=10
```

Interogare: *Studentii care au cel puțin o notă*

```
SELECT S.sid  
FROM    Students S, Enrolled E  
WHERE    S.sid=E.sid
```

- Rezultatul e diferit cu **DISTINCT**?
- Ce efect are înlocuirea *S.sid* cu *S.sname* în clauza **SELECT**?  
Rezultatul e diferit cu **DISTINCT** în acest caz?



# Expresii și string-uri

- Obține triplete (cu vârsta studenților + alte două expresii) pentru studenții al căror nume începe și se termină cu B și conține cel puțin trei caractere.

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2  
FROM Students S  
WHERE S.name LIKE 'B_%B'
```

- **AS** și **=** sunt două moduri de redenumire a câmpurilor în rezultat.
- **LIKE** e folosit pentru comparații pe siruri de caractere. **'\_'** reprezintă orice caracter și **'%'** stands reprezintă 0 sau mai multe caractere arbitrare.

# INNER JOIN

```
SELECT S.name, C.cname
FROM Students S,
Enrolled E, Courses C
WHERE S.sid = E.sid
AND E.cid = C.cid
```



```
SELECT S.name, C.cname
FROM Students S
INNER JOIN Enrolled E ON
S.sid = E.sid,
INNER JOIN Courses C ON
E.cid = C.cid
```

*Students*

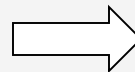
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

*Courses*

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

*Enrolled*

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1

# LEFT OUTER JOIN

- Dacă dorim să regăsim și studenții fără nici o notă la vreun curs:

```
SELECT S.name, C.cname
FROM Students S
LEFT OUTER JOIN Enrolled E
ON S.sid = E.sid,
LEFT OUTER JOIN Courses C
ON E.cid = C.cid
```

*Students*

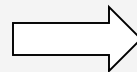
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

*Courses*

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

*Enrolled*

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
Anne	<b>NULL</b>

# RIGHT OUTER JOIN

- Pentru a gasi notele asiguate unor studenti inexistenti:

```
SELECT S.name, C.cname  
FROM Students S
```

```
RIGHT OUTER JOIN Enrolled E  
ON S.sid = E.sid,  
INNER JOIN Courses C ON  
E.cid = C.cid
```

*Students*

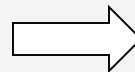
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

*Courses*

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

*Enrolled*

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
<b>NULL</b>	Databases2

# FULL OUTER JOIN

- LEFT+RIGHT OUTER JOIN
- In majoritatea SGBD OUTER e optional

```
SELECT S.name, C.cname
FROM Students S
FULL OUTER JOIN Enrolled E
ON S.sid = E.sid,
FULL OUTER JOIN Courses C
ON E.cid = C.cid
```

*Students*

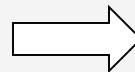
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

*Courses*

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

*Enrolled*

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
NULL	Databases2
NULL	Databases1
Anne	NULL



A **SQL** query walks into a bar.

He approaches two **tables** and says:

*"Mind if I **join** you?"*