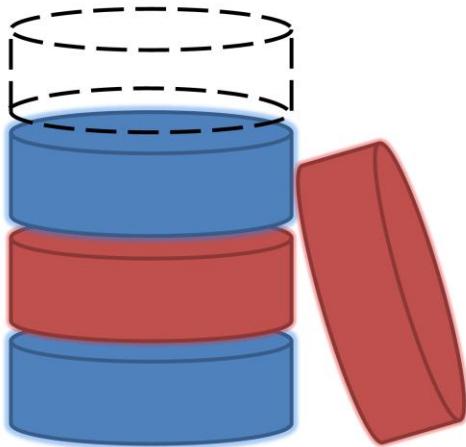


Modelul Relațional



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 relational - 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 demostrate matematic

Relația – definiție formală

- O **relație** sau **structura unei relații R** este o listă de **nume de atribută** $[A_1, A_2, \dots, A_n]$.
- **Domeniu** = multime de valori scalare (tipuri atomice - intreg, text, dată, etc)
- $D_i = \text{Dom}(A_i)$ - domeniul lui A_i , $i=1..n$
- **Instanță 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 distințte!
- **Cardinalitate** = numărul tupluri unei relații

Exemplu de relație

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

The diagram shows a table representing a relation. Above the table, two arrows point from the text 'field name' and 'field type (domain)' to the first two columns ('sid' and 'name') respectively. To the right of the table, a bracket groups the last four columns ('age', 'gr', and two empty columns) under the label 'relation schema'. Below the table, another bracket groups the same four columns under the label 'relation instance'. A third bracket groups all five columns under the label 'relation tuple'.

sid	name	email	age	gr	
2833	Jones	jones@scs.ubbcluj.ro	19	231	
2877	Smith	smith@scs.ubbcluj.ro	20	232	
2976	Jones	jones@math.ubbcluj.ro	21	233	
2765	Mary	mary@math.ubbcluj.ro	22	233	

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

Baze de date relationale

- 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: $18 \leq \text{age} \leq 70$
- TestResults(*sid:string, TotalQuestions:integer, NotAnswered:integer, CorrectAnswers:integer, WrongAnswers:integer*)
 - $\text{TotalQuestions} = \text{NotAnswered} + \text{CorrectAnswers} + \text{WrongAnswers}$ – **nu e o CI!**

Chei Primare

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

ȘI

2. Aceste lucru nu este adevărat pentru nici o submuțime a cheii

- Dacă a 2-a afirmație este falsă → **super cheie**
- Daca 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)

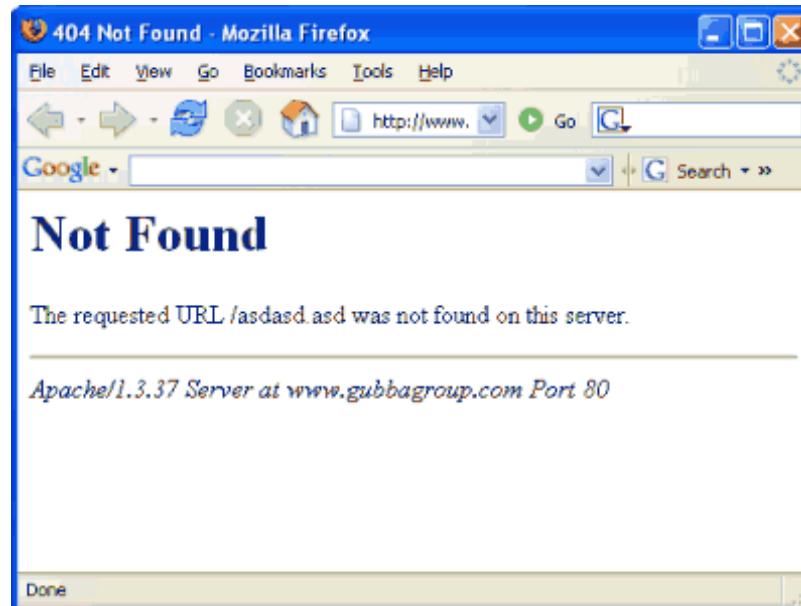
sid este cheie externă referind *Students*

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* în *Enrolled* este o cheie străină ce referă o înregistrare din *Students*.
- Adaugarea în *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
1237	DB2	9

Students

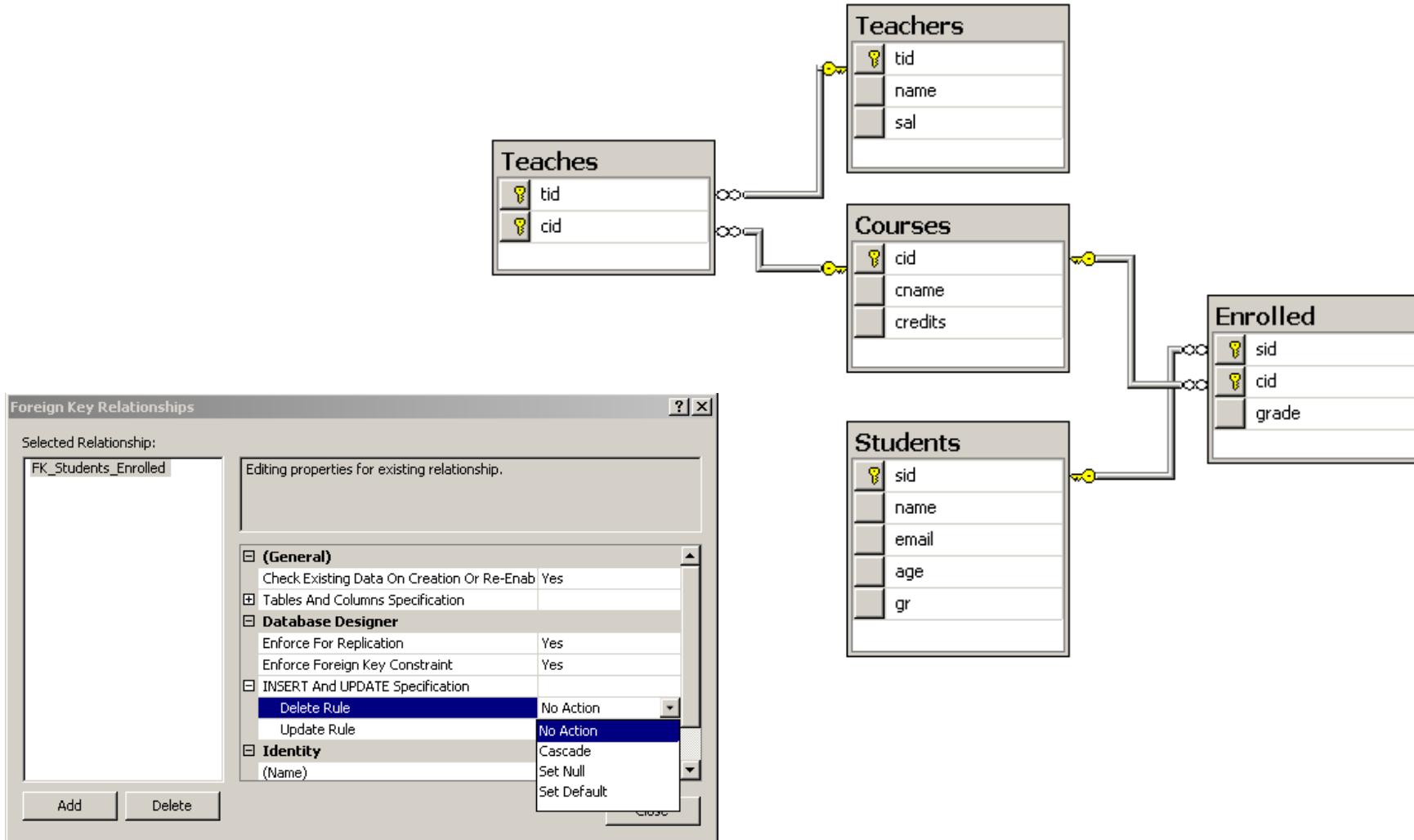
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Integritate referențială

- Dacă o înregistrare din *Students* este ştersă dar ea este referită din *Enrolled*:
 - se sterg toate înregistrările ce o referă din *Enrolled*.
 - nu se permite stergerea înregistrării din *Students*
 - sid din *Enrolled* va avea asignată o valoare implicită.
 - sid din *Enrolled* va avea asignată valoarea *null*.

Enrolled			Students				
<i>sid</i>	<i>cid</i>	<i>grade</i>	<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	Alg1	9	1234	John	j@cs.ro	21	331
1235	Alg1	10	1235	Smith	s@cs.ro	22	331
1234	DB1	10	1236	Anne	a@cs.ro	21	332

Reprezentarea grafică a CI



Cum apar CI?

- CI se bazează pe semantica entităților din lumea reală / conceptuală modelate.
- Putem verifica dacă o CI este incălcată 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;

Interogări

- Posibile informații pe care dorim sa le obținem din baza de date anterioară (*Faculty Database*) :
 - Care este numele studentului cu *sid* = 2833?
 - Care este salarul 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?**)
- *Limbaj 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

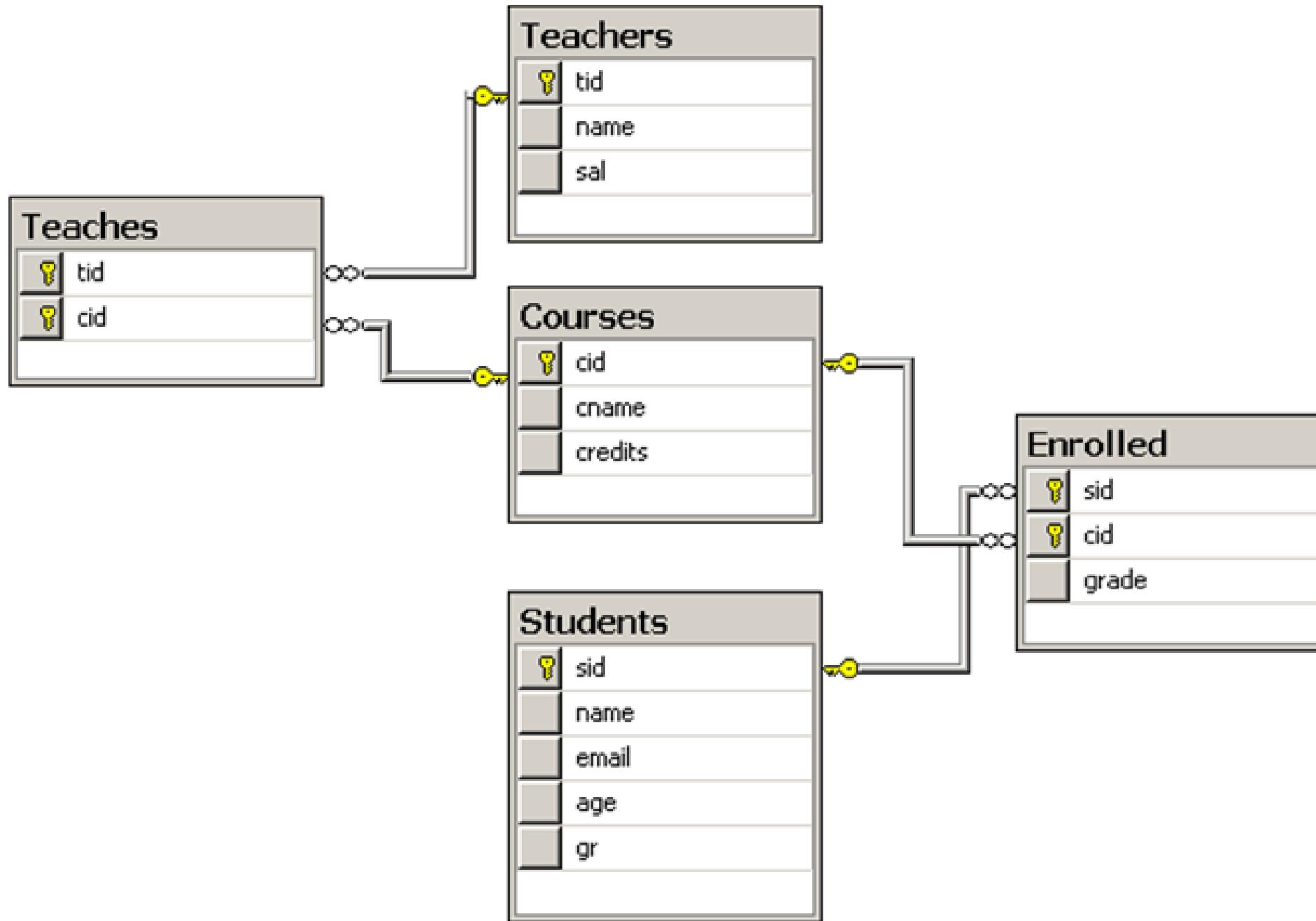
{ $X \mid \exists Y : Y \in Students \wedge Y.age > 20 \wedge X.name = Y.name$ }

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 (sectiuni 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*.



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

Studenții 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 <, >, =, ≤, ≥, ≠) 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

Utilizare *alias* pentru

```
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: Studenții 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 studentilor + 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 comparatii pe siruri de caractere. **'_'** reprezinta orice caracter si **'%'** stands reprezinta 0 sau mai multe caractere arbitrar.

INNER JOIN

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



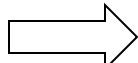
```
SELECT S.name, C cname  
FROM Students S  
INNER JOIN Enrolled E ON  
S.sid = E.sid,  
INNER JOIN Courses C ON  
E.cid = C.cid
```

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

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

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

LEFT OUTER JOIN

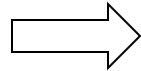
- Daca dorim sa regasim si studentii fară nici o notă la vreun curs:

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

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

```
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
```

RIGHT OUTER JOIN

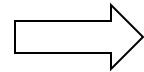
- Pentru a gasi notele asignate unor studenți inexistenti:

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



```
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
```

Courses

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

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

FULL OUTER JOIN

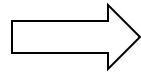
- LEFT+RIGHT OUTER JOIN
- In majoritatea SGBD OUTER e optional

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

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

```
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
```