Curs 2. Modelul Relaţional

Nivelele de abstractizare ale implementării unei baze de date:

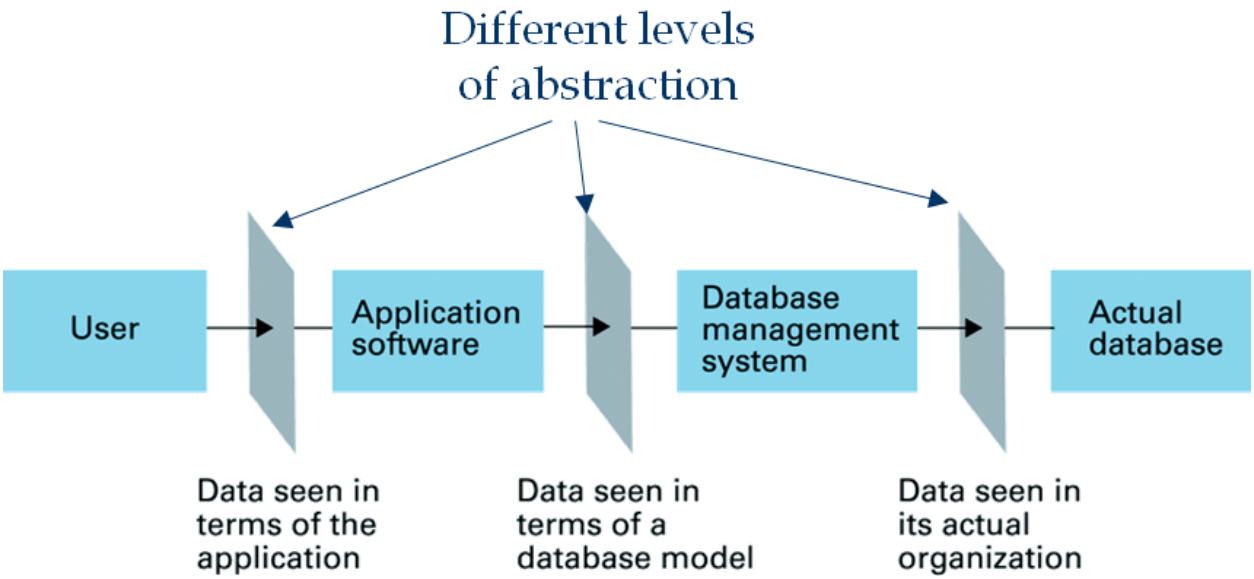


Figura 2.1.

Nivele de abstractizare:

- Strucutrile externe (*view*-urile) descriu modul în care utilizatorii văd datele
- Structura conceptuală se referă la modelul logic compus din relaţii, atribute, etc
- Structura fizică conţine fişierele de date şi indecşii utilizaţi

Exemplu: baza de date ***Faculty***
Structura conceptuală:
Students(sid:string, name:string, email:string, age:integer, gr:integer)
Courses(cid: string, cname: string, credits:integer)
Exams(sid:string, cid:string, grade:integer)

Structura fizică:
Relaţii memorate ca fişiere binare
Fişier index pentru primul atribut din ***Students***.

Structura externă (*View*):
Course_info(cid:string, enrollment:integer)

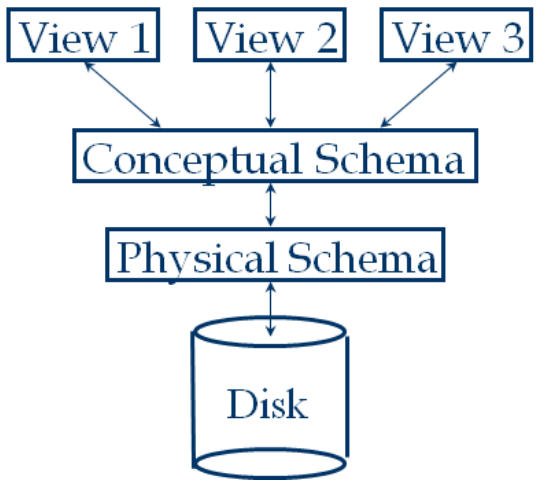


Figura 2.2.

Unul dintre beneficiile importante ale utilizării unui SGBD este separarea logicii aplicaţiilor de modul în care datele sunt structurate şi stocate.

- *Independeţă logică*: protecţie fată de modificările structurii *logice* a datelor.
- *Independenţă fizică*: protecţie fată de modificările structurii *fizice* a datelor.

Interogări în SGBD-urile relaţionale

Se pot enunţa mai multe posibile întrebări legate de conţinutul bazei de date ***Faculty***:

- "*Care este numele stui sdentului având sid egal cu 2833*"?
- "*Care este salarul profesorului care predă cursul cu codul Alg100*"?
- "*Câţi studenţi sunt înscrişi la cursul Alg100*"?

Astfel de întrebări, ce implică date stocate într-un SGBD poartă numele de ***interogări.*** Un SGBD furnizează un limbaj specializat, numit ***limbaj de interogare***, ce permite definirea de interogări. Fiecare limbaj de interogare este compus din:

- *Limbaj de definire a datelor (Data Definition Language - DDL)*
  - Utilizat în definirea structurilor conceptuale;
  - Include un limbaj de definire a constrângerilor (*constraint definition language* - CDL) pentru descrierea condiţiilor pe care trebuie să le satisfacă instanţele bazei de date;
  - Includes un limbaj de definire a modului de stocare (*storage definition language* - SDL) ce permite influenţarea arhitecturii structurii fizice (în anumite SGBD-uri);
- *Limbaj de manipulare a datelor (Data Manipulation Language - DML)*
  - Utilizat pentru descrierea operaţiilor aplicate asupra instanţei unei baze de date
  - Exista atât limbaje procedurale (orientate pe *cum*) cât şi declarative (*ce*).

Limbaje de interogare pentru BD-uri relaţionale

SQL (Structured Query Language):

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

Algebră relaţională:

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

Domain Calculus:

$\{<X> | \exists V \exists Y \exists Z \exists T : Students(V, X, Y, Z, T) \wedge Z>20\}$

T-uple Calculus:

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

Actori ce interacţionează cu bazele de date:

- Proiectant/Arhitect baze de date: Proiectează structurile logice/fizice
- Programator
- Administrator baze de date
    - Gestionează securitatea şi autorizarea accesului
    - Asigură disponibilitatea si recuperarea datelor
    - Responsabil cu întreţinerea bazei de date (ex. actualizarea indecşilor)
- Administrator de sistem
- Utilizatori finali (mai mult sau mai puţin experimentaţi)

Modelul relaţional

Modelul relaţional este un model de date utilizat pe scară largă în bazele de date datorită a două caracteristici esenţiale: este simplu şi elegant.

Simplitatea constă în structurile de date în formă tabelară.

Tabelele

- sunt simplu de înţeles
- sunt utile în modelarea multor situaţii/entităţi din lumea reală
- conduc la construirea de interogări şi limbaje de interogare de o complexitate redusă.

Eleganţa este dată de uşurinţa utilizării matematicii în descrierea şi reprezentarea înregistrărilor şi a colecţiilor de înregistrări în sub formă de *relaţii*.

Relaţiile

- pot fi modelate formal
- permit utilizarea de limbaje de interogare formale
- au proprietăţi ce pot fi modelate şi demostrate matematic.

Relaţia – definiţie formală

Un **domeniu** este o mulţime de valori scalare (adică limitate la tipuri atomice - intreg, string, boolean, data calendaristică). O **relaţie** sau **o structură de relaţie R** este o listă de **nume de atribute** [A_1, A_2, …, A_n].

$D_i = Dom(A_i)$ – domeniul lui A_i, i=1..n

Instanţa unei relaţii ([R]) este o submulţime a

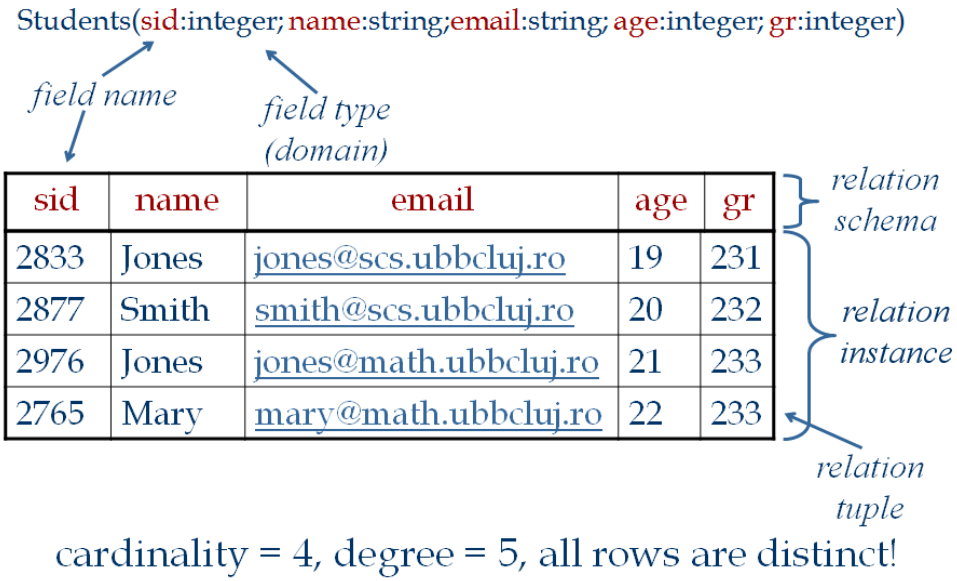
$$D_1 \times D_2 \times \ldots \times D_n$$

Grad (**aritate**) = numărul tuturor atributelor din structura unei relaţii

Tuplu = un element al instanţei unei relaţii, o înregistrare. Toate tuplele unei relaţii sunt distincte!

Cardinalitate = numărul tuplelor unei relaţii

Exemplu de relaţie:



Deseori, în vorbirea curentă, dăm acelaşi înţeles unor concepte diferite ca

- *relaţie*, *structură* şi *instanţă*
- *instanţă* şi *tabelă*
- *atribut*, *câmp* şi *coloană*
- *tuplu*, *înregistrare* şi *linie*

O **bază de date** este o mulţime de relaţii. **Structura** unei baze de date este mulţimea structurilor relaţiilor bazei de date respective. **Instanţa** unei baze de date este mulţimea instanţelor relaţiilor bazei de date.

Constrângeri de integritate

Constrângerile de integritate (CI) sunt condiţii ce trebuie să fie îndeplinite de către *orice* instanţă a bazei de date. CI sunt specificate la momentul definirii structurii unei relaţii şi sunt verificate la modificarea conţinutului relaţiei.

Spunem despre o instanţă a unei relaţii că este *legală* dacă satisface toate CI specificate.

Exemple de constrângeri de integritate:

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

Constrângere de domeniu (atributul *gr* poate lua doar valori întregi): *gr:integer*

Constrângere de interval: $18 \leq age \leq 70$

TestResults*(sid:string, TotalQuestions:integer, NotAnswered:integer, CorrectAnswers:integer, WrongAnswers:integer)*

TotalQuestions = NotAnswered+CorrectAnswers+WrongAnswers – **nu reprezintă o CI!**

Chei Primare

O mulţime de atribute reprezintă o **cheie** pentru o relaţie dacă:

1. Nu există două tuple care au aceleaşi valori pentru toate atributele

ŞI

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

Dacă a 2-a afirmaţie este falsă → **super cheie**.

Daca există >1 cheie pentru o relaţie → **chei candidat**

Una dintre cheile candidat este selectată ca **cheie primară**

Chei străine. Integritate referenţială

O **cheie străină** este o mulţime de câmpuri a unei relaţii utilizate pentru a `*referi*' un tuplu al unei alte relaţii (un fel de `*pointer logic*'). Aceasta trebuie să corespundă cheii primare din a doua relaţie.

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

În cazul tabelelor ***Students*** şi ***Enrolled***: *sid* din ***Enrolled*** este o cheie străină ce referă o înregistrări din tabela *Students*. Pentru a se respecta integritatea referenţială, dacă se incearcă inserarea unei înregistrări în ***Enrolled*** ce referă un student inexistent, SGBD-ul va respinge această înregistrare.

În acelaşi timp, dacă o înregistrare din ***Students*** este ştearsă dar ea este referită din ***Enrolled***, se poate opta pentru una dintre următoarele abordări:

- se şterg toate înregistrările corespunzătoare din ***Enrolled***.

- nu se permite ştergerea înregistrării referite din ***Students***
- *sid* din ***Enrolled*** va avea asignată o valoare implicită.
- *sid* din ***Enrolled*** va avea asignată valoarea *null*.

Aceleaşi 4 abordări sunt valabile şi in cazul în care se modifică valorile cheii primare ale înregistrărilor din ***Students***.

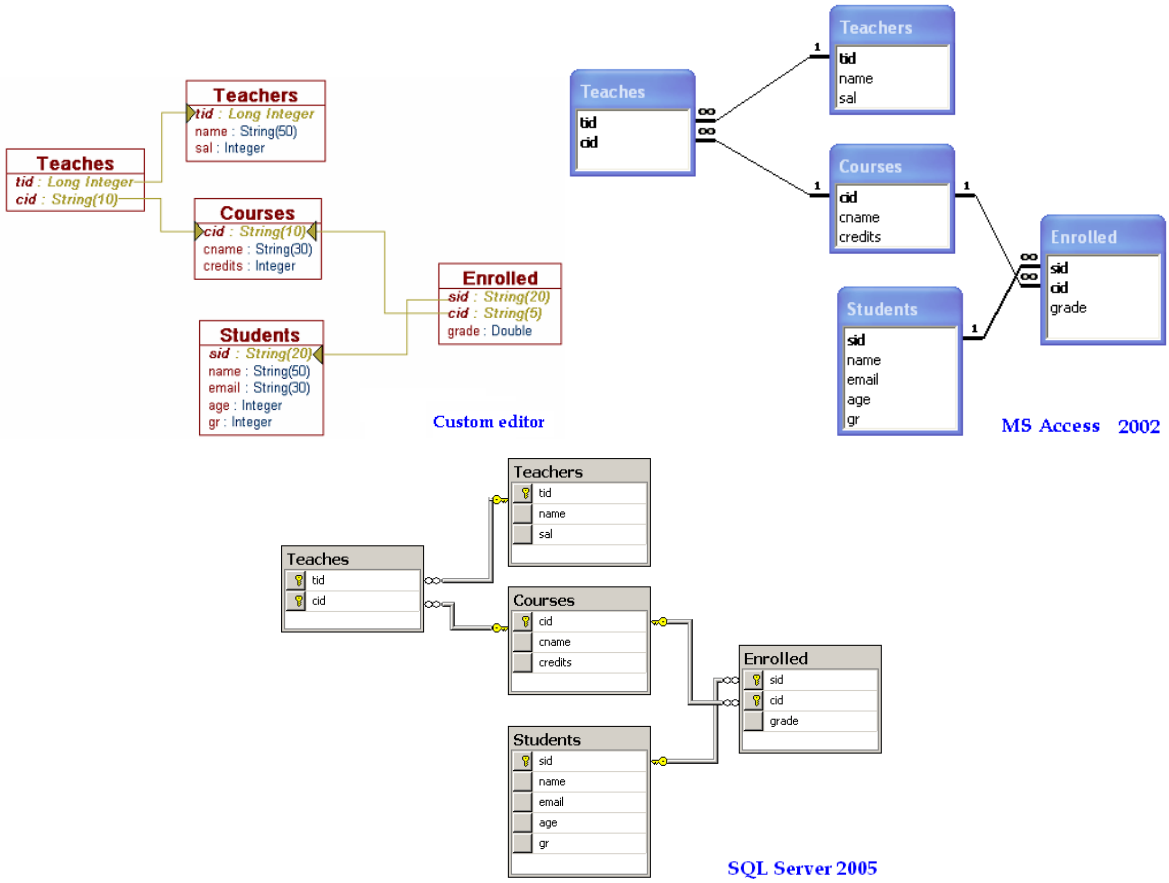


Figura 2.3. Exemple de reprezentări grafice ale tabelelor, câmpurilor, cheilor primare şi cheilor străine în diverse medii

Cheile primare şi străine sunt cele mai comune CI. CI se bazează pe semantica entităţilor din lumea reală / conceptuală ce sunt modelate cu ajutorul relaţiilor din baza de date. Putem verifica dacă o CI este violată de instanţa unei tabele, însă nu vom putea deduce dacă o CI este adevărată doar consultând o instanţă a unei tabele. O CI se referă la *toate instanţele posibile* ale unei tabele particulare! \

Limbaje de interogare relaţionale

Unul dintre avantajele modelului relaţional este acela că oferă suport pentru interogări simple dar puternice ale datelor. Interogările pot fi scrise intuitiv, într-un limbaj asemănător cu limbajul

natural, iar DBMS-ul este responsabil de evaluare eficientă a acestora.

Structured Query Language (SQL) a devenit limbajul de interogare standard de-facto pentru interogarea bazelor de date relaţionale. A fost dezvoltat de IBM (*System R*) în 1970, şi a cunoscut diverse extensii ulterioare, trecând prin mai multe etape de standardizare.

Standarde SQL:

- SQL-86
- SQL-89 (minor revision)
- SQL-92 (major revision) - *1,120 pagini*
- SQL-99 (major extensions) - *2,084 pagini*
- SQL-2003 (s-a introdus XML ca secţiune nouă) - *3,606 pagini*
- SQL-2008
- SQL- 2011

Sub-limbaje SQL

Data-definition language (DDL):

- Creare / eliminare / modificare *relaţii* şi *view*-uri.
- Definire *constrângeri de integritate* (CI).

Data-manipulation language (DML)

- Permite utilizatorilor să definească interogări
- Inserare / ştergere / modificare înregistrări.

Access Control:

- Permite gestionarea drepturilor de acces şi manipulare a tabelelor şi a conţinutului lor.