Orice model de date, conform unei sugestii a lui Codd, trebuie să se bazeze pe trei componente: structurile de date, constrângerile de integritate şi operatorii de manipulare a datelor.

- **Structurile de date.** Structurile sunt definite de un limbaj de definire a datelor (data definition language). Datele în modelul relaţional sunt structurate în relaţii bidimensionale. Elementele principale ale structurii relaţionale sunt relaţiile, tuplurile, atributele, domeniile.
- **Constrângerile de integritate.** Prin integritatea datelor se subînţelege că datele rămân stabile, în siguranţă şi corecte. Integritatea în modelul relaţional este menţinută de constrângeri interne care nu sunt cunoscute utilizatorului.
- **Manipularea datelor.** Relaţiile pot fi manipulate utilizând un limbaj de manipulare a datelor (data manipulation language). În modelul relaţional, limbajul foloseşte operatorii relaţionali bazaţi pe conceptul algebrei relaţionale. În afară de aceasta, există limbaje echivalente algebrei relaţionale, cum ar fi calculul relaţional orientat pe tuplu şi calculul relaţional orientat pe domeniu.

1.1.2. Tupluri

În sistemele cu fişiere o mulţime de câmpuri ce e concepută ca o unitate de salvare şi căutare se numeşte înregistrare. Înregistrarea are un format specific şi depinde de tipurile de date ale câmpurilor. O linie dintr-o relaţie tabelară corespunde înregistrării din fişiere şi în teoria relaţională se numeşte tuplu.

Din definiţiile de mai sus putem conchide următoarele:

(1) Într-o relaţie nu există coloane cu nume duplicate, fiindcă atributele A_{ij}, $1 \leq j \leq k$, sunt elemente ale mulţimii R_i.
(2) Relaţia r_i nu are tupluri identice, fiindcă r_i este o mulţime de tupluri.
(3) Ordinea tuplurilor în r_i este nesemnificativă, fiindcă r_i este o mulţime.
(4) Ordinea coloanelor e nesemnificativă.
(5) Valorile atributelor în r_i sunt atomice fiindcă domeniile sunt simple.

Relaţiile ce se stochează fizic şi formează baza de date se numesc *relaţii de bază.*

Există, însă, şi situaţii în care extensia nu se memorează în baza de date. Este cazul aşa-numitelor *relaţii virtuale*, cunoscute şi sub numele de *relaţii derivate* sau *viziuni*. Relaţia virtuală nu este definită explicit ca relaţia de bază, prin mulţimea tuplurilor componente, ci implicit pe baza altor relaţii. Relaţiile de bază sunt proiectate de administratorul bazei de date, în timp ce viziunile sunt definite de utilizatorii bazei de date.

Relaţiile asupra unei mulţimi de atribute pot avea un nume, sau pot să nu aibă, dacă ele sunt identificate în mod unic de schemele sale. Numele relaţiei, de obicei, se scrie cu minuscule, de exemplu, relaţia r.

studenţi	NUME	NOTĂ_MED	FACULTATE	DECAN
	Vasilache	7.8	Cibernetică	Popovici

Fie că în relaţia $r(A_1\ A_2\ ...A_n)$ vrem să introducem date. Operaţia de inserţie, a unui tuplu în relaţia r poate avea forma:

$$\text{Add } (r;<a_1a_2...a_n|A_1\ A_2...A_n>).$$

În cazul că ordinea atributelor în relaţie e cunoscută, e acceptabilă o formă mai scurtă a operaţiei:

$$\text{Add } (r;<a_1a_2...a_n>).$$

Scopul acestei operaţii constă în adăugarea unui tuplu într-o relaţie concretă. Rezultatul operaţiei poate să eşueze din următoarele cauze:

(1) tuplul de inserţie e definit pe o mulţime de atribute ce nu corespunde schemei relaţiei;
(2) valorile componentelor tuplului nu sunt luate din domeniile corespunzătoare;
(3) în relaţie deja se găseşte un tuplu cu asemenea componente cheie.

În toate aceste cazuri operaţia Add păstrează relaţia r intactă.

Operaţia de ştergere a datelor se utilizează pentru eliminarea conţinutului relaţiilor. Pentru relaţia r de mai sus, operaţia de ştergere se reprezintă:

$$\text{Del } (r;\ <a_1a_2...a_n|A_1\ A_2...A_n>).$$

În cazul când numele de atribute sunt sortate, poate fi utilizată următoarea notaţie scurtă:

$$\text{Del } (r;\ <a_1a_2...a_n>).$$

În realitate, o parte de date din operaţia de mai sus poate fi redundantă pentru determinarea tuplului destinat ştergerii. E suficientă definiţia valorilor atributelor cheie.

Sisteme cu BD: Argumente pro

- **Evitarea** redundanței și inconsistenței datelor
- Asigurarea integrității datelor
- Asigurarea accesului securizat la date
- Asigurarea **independenței** datelor
- Restricționarea **accesului neautorizat** (susținerea accesului concurent la date)
- Structuri de stocare adecvate
- Procesare **interogări eficiente** (optimizarea interogărilor)
- Asigurarea **copiilor de rezervă** (Backup & Recovery)
- Interfețe multiutilizator
- Suport pentru standardizare
- Reducerea timpului de dezvoltare a aplicațiilor
- Disponibilitatea informației

Sisteme cu BD: Dezavantaje

- **Investiție** inițiala ridicată în hardware, software , training
- **Complexitate indusă de securizare**, controlul concurenței, recovery și funcții de integritate
- Generalitatea oferită de SGBD pentru definirea și procesarea datelor este peste **nivelul cerințelor**
- Se poate întâmpla sa lucreze "lent" din cauza unui număr mare de **verificări**
- Dimensiunea
- Susceptibilă la eșecuri (*Discutabil*)
- Complexitatea de recuperarea a eșecurilor (*Discutabil*)

- Astfel, SGBD-ul trebuie:
  - **să posede** un limbaj de definire a datelor care ar permite uşor crearea bazei de date, dar şi modificarea structurii ei;
  - **să posede** un limbaj de manipulare a datelor care ar permite inserarea, suprimarea, modificarea şi interogarea bazei de date într-o formă eficientă şi potrivită;
  - **să permită** stocarea unor cantităţi enorme de date (mii de milioane de caractere) fără ca utilizatorul să perceapă vre-o degradare în ce priveşte randamentul total al sistemului;
  - **să permită** o gestiune sigură a datelor, cu respectarea accesului neautorizat şi să controleze afecţiunile produse de dispozitivele mecanice şi electronice asupra datelor stocate;
  - **să permită** accesul simultan cel puţin pentru o parte din utilizatori, adică să asigure accesul concurent la date;

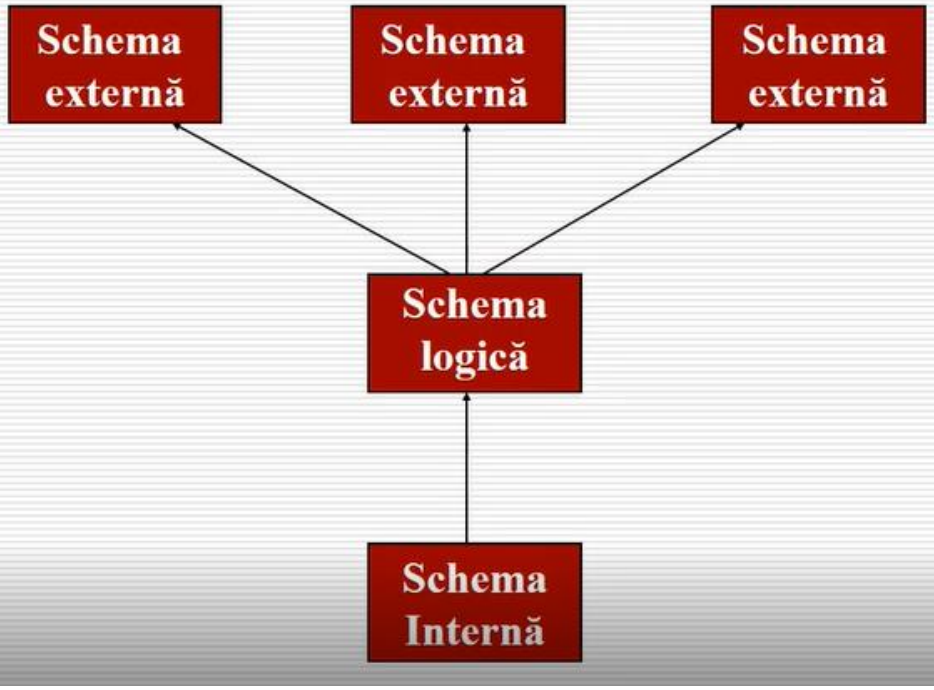
- **Datele**
  - Reprezentare stocată a ***obiectelor*** sau ***evenimentelor*** din domeniul de interes.
  - Pot fi structurate: ***numere***, ***text***, ***date***.
  - Nestructurate: ***imagini***, ***video***, ***documente***.
- **Informația**
  - Datele procesate care măresc cunoașterea persoanei care le utilizează.

- **MetaDatele** sau **Dicționarul de Date**
  - Conține **schema** BD, **permisele** de acces, etc.
  - Sunt **date despre date**
  - Păstrează informații care permite **traducerea** între cele 3 nivele ale Arhitecturii **ANSI/SPARC**

Funcţiile SGBD (1)

- **Gestiunea dicţionarului de date**
  - Programele aplicative accesează datele prin intermediul SGBD, care caută în dicţionarul de date structura datelor şi a legăturilor necesare rezolvării problemei utilizatorului
- **Gestiunea fişierelor de date**
  - Colecţia de date (BD) se materializează printr-un fişier sau colecţie de fişiere de date, fişiere index, etc.

Arhitectura ANSI/SPARC



- Arhitectura în 3 niveluri
  - **Nivelul Intern**: pentru proiectanții de sistem
  - **Nivelul Logic**: pentru proiectanții bazei de date și administratori
  - **Nivelul Extern**: pentru utilizatorii bazei de date

Nivelul intern

- Descrie **modul în care sunt stocate datele** (fișiere și indecși)
- Schema fizică este **dependentă de SGBD-ul** utilizat. Există elemente comune ce trebuie menționate:
  - Strategia de memorare
  - Căile de acces
  - Tehnicile de comprimare a datelor
  - Tehnicile de criptografieze
  - Corelația logic/fizic
  - Tehnici de arhivare și optimizare
  - Structura memoriei
  - Organizarea fizică
  - Controlul accesului

Nivelul logic

- **Descrie datele** stocate în BD **şi relaţiile** dintre acestea
  - Propune **schema logică** a bazei de date care descrie structura bazei de date şi constrângerile asociate acesteia
  - Schema logică **ascunde detaliile** de stocare fizică a datelor
  - Pentru construirea schemei logice sistemul de gestiune oferă un limbaj de definire a datelor (**DDL**)

Independenţa datelor

- ***Independentă*** **-** abilitatea de a modifica un nivel în arhitectură fără a afecta celelalte niveluri ale schemei
  - Independenta logică
  - Independenta fizică
- Cel mai ***important avantaj*** al utilizării SGBD
  - Aplicaţiile sunt izolate fată de modificările la nivel logic sau la nivel fizic prin cele trei nivele de abstractizare

Independenţa logică

- **Independenta logica** – capacitatea de a modifica schema logică fără a afecta vederea externa (programele de aplicație)
- Vederile (**view** în modelul relațional, tabelă virtuală, schema externă) asigură posibilitatea modificării structurii datelor (schema logică), acest lucru fiind ascuns aplicațiilor

Independenţa fizică

- **Independenta fizica –** abilitatea de a modifica *schema interna* fără a modifica *vederea externă*
- Schema logică asigură posibilitatea modificării aranjării datelor pe suport secundar sau a indecşilor, acest lucru fiind, de asemenea, ascuns aplicaţiilor

Nivelul de abstracţie fizic

- **Schema fizica :**
  - Specifică detalii suplimentare legate de *stocarea datelor*
  - Menționează **modul în care tabelele** (la modelul relațional) descrise prin schema logică **sunt stocate** pe dispozitive suport secundar, discuri sau benzi magnetice
  - Descrie tipul fișierelor pentru stocare pe suport secundar și crearea unor structuri auxiliare de date numite **indecși** în scopul regăsirii mai rapide a datelor

Nivelul de abstracţie fizic

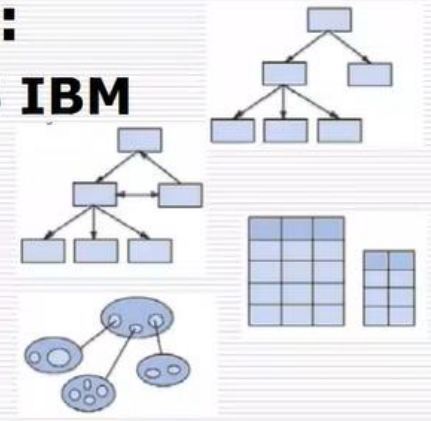
- **Fişiere:**
  - Secvenţiale
  - Indexat-secvenţiale
  - B-arbori
  - B^+-arbori
  - Tabele cu dispersie (Hash tables)

Nivelul de abstracţie logic

- **Schemă logică:**
  - **Schemă logică**, descrie datele stocate în BD în termeni ai **modelului de date al SGBD**
  - Pentru un SGBD relaţional, schema logică **descrie toate tabelele** (relaţiile) stocate în BD
  - Limbajul folosit se numeşte **Limbaj de Descriere a Datelor** (LDD)

Nivelul de abstracţie logic

- **Modele de date:**
  - Ierarhic, ex. **IMS IBM**
  - De tip rețea
  - Relațional
  - Obiectual
  - Obiect-relațional

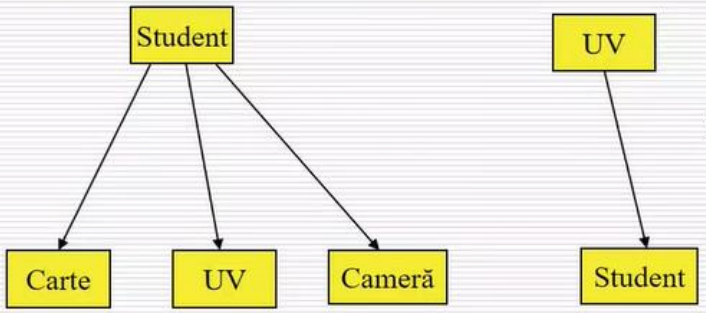


Nivelul de abstracţie logic

- **Modelul ierarhic:**
  - Folosit de IBM în sistemul **IMS** (care încă este unul dintre produsele furnizate de această firmă), în care organizarea este sub formă arborescentă
  - Nodurile conțin date și legături (‘**pointeri**’) către nodurile fiu

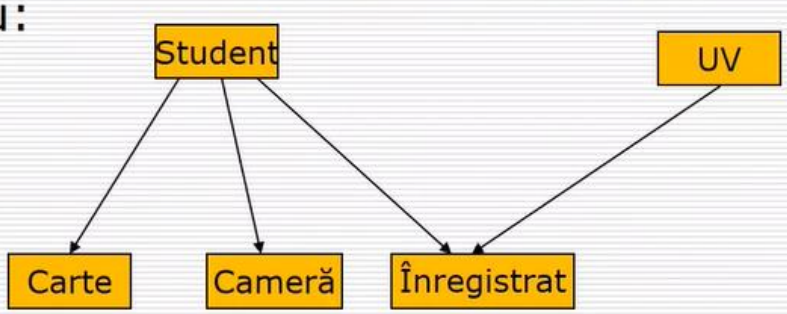
Modelul ierarhic: Nivelul de abstracţie logic

- Sistemul IBM IMS proiectat la sfârșitul anilor 1960 pentru programul Appolo (NASA)
- Exemple



Modelul rețea:
Nivelul de abstracție logic

- Model definit de grupul DBTG al Comitetului CODASYL în 1971 (revizuit în 1978)
- Exemplu:



Modelul relațional (1):
Nivelul de abstracţie logic

- Istoric (Codd (1970), bazat pe teoria matematică a relațiilor – introduce relațiile ca structuri de date și algebra/calcul relațional pentru *specificarea interogărilor*
- **Relațiile = STRUCTURI de DATE**, încorporează constrângeri de integritate, permit interogări și capabilități de actualizare

π = proiectie

σ = selectie

v = sau

& = si

~ = toate combinatiile inexistente (adica faci tabel cu toate combinatiile si imparti la tabelul initial ca sa aflii acelea inexistente)

U = unesti doua tabele (uniune)

∩ = doar acelea in comun

**** = din tabelul stang stergi tuplurile care sunt in comun din tabelul drept

|><| = jonctiunea naturala (datele care sunt in comun din tabelul drept si stang)

|><|LEFT = jonctiunea de stanga (datele din stanga se atribuie cu datele din dreapta care sunt in comun, daca nu exista se scrie null)
|><|RIGHT = ca si LEFT, insa pentru tabelul din dreapta
|><|FULL = LEFT si RIGHT la un loc

- Cu **litere majuscule se scriu denumirea coloanei**, iar **cu litere minuscule se scriu datele din coloane**:
  Exemplu: **R\S**, adica se sterg **coloanele** din R care sunt in comun cu coloanele din S
  Exemplu**: r\s**, se sterg **datele** din r care sunt in comun cu datele din s