* **Database management system(system de gestionare a bazelor de date)** sau DBMS- este o cale sau metoda de a organiza datele noastre

**Schema de la date la perspectiva**

**1.Date**

**2.Informatie –** caracteristica sau specificatie a datei. Ceea ce ne spune ce anume este data.

**3.Cunostinta** - Concluzii, idei trase din compararea diferitor informatii ce provin de la diferite date, dar date ce au o legatura comuna(model).

**4**.**Inteligenta** – se refera la decizii inteligente pe care le putem lua stiind cunostintele pe care le avem, ce pornesc de la informatii si date. Anume asta si e valoarea datelor, sa ne ajute sa luam decizii corecte si inteligente. Totodata, inteligenta ne ajuta sa prezicem.

Datele ar fi inutile fara a cunoaste lucruri despre ele si fara a le putea utiliza undeva,

Ex:

**Date** – 350$

**Informatie** - Un pret al unui bilet la un meci in 21 ianuarie 2019

**Cunostinta** – comparand cu pretul biletului din 4 ianuarie 2019, ce era de 17.25$, acesta a crescut cu 2000%.

**Inteligenta** – sa cumpar sau nu?

**Solutii pentru DataManagement(Gestionarea datelor)**

* **File system –** o colectie de fisiere unde pur si simplu scriem date si care sunt accesate de programe.(ca serializarea)
* E foarte greu sau imposibil sa cream o legatura intre ele, asa cum ele sunt separate si izolate de restul datelor din alte fisiere.
* Fiecare program ar avea propriul fisier cu date identice, si deci nu am putea Evita repetarea lor.
* Fisierele depind de aplicatii, si deci nu toate le pot accesa.
* Sunt greu de partajat.
* **DataBase(solutie perfecta) –** structura care contine date. Ea defineste cum datele sunt reprezentate, cum datele se repeta, cum sunt legate intre ele.
* **Intr-o baza de date relationara, Datele sunt organizate prin:**
* **Entitati** – model de obiecte de la care colectam date, ca clasa, care arata ce trebuie sa contina orice obiect
* **Atribute**– caracteristici ale obiectelor, arata despre ce anume vrem sa colectam date.
* **Relatii** – ne arata cum entitatile(obiectele) sunt legate(relatie=tabel).
* **DBMS** – are anume rolul de a crea o conexiune intre baza de date si user pentru ca userul sa poata utiliza baza de date mai usor. E un program prin care userul interactioneaza cu baza de date. El poate arata si date statistice despre baza de date.
* **DBMS avantaje**:
* putem obtine informatii statistice din datele noastre sau despre ele
* putem sa le impartasim usor, caci toate programele si sistemele sunt compatibile cu un DBMS, sau cel putin cu cele mai populare(MySQL etc.)
* Control redundanta – repetarea inutila a datelor
* Diferite programe daca modifica anumite date, ele se modifica chiar in baza de date, si deci se modifica pentru toate programele direct, caci toate sunt conectate la aceeasi baza de date si nu creaza doar copii ale unei baze de date pe care o modifica separat.
* Securitate mai mare
* Ofera o interfata pentru user ca sa interactioneze cu datele
* **DBMS dezavantaje:**
* Necesita cunostinte
* O greseala in baza de date duce la probleme in toate programele conectate la ea.
* Fisierele sunt mai mari.

**Continutul DBMS**

DBMS contine sau e formata din:

* **User Data** – datele cu care userul lucreaza, le modifica, le vizualizeaza. Datele sunt stocate in tabele, cu anumite relatii intre ele.

Iata de ce, se numeste **Baza de date relationara**, datorita relatiilor ce sunt reprezentate sau echivalente cu tabele.

Fiecare tabel are coloane. Un set de coloane(din tabel) **formeaza o inregistrare din baza de date.** O **structura**(schema) a bazei de date este ca o reprezentare a acesteia, aratand cum sunt create obiectele, relatiile(ce attribute au, constragerile),asocierile etc.

* **Metadate** – date despre datele noastre, si anume cum sunt datele noastre stocate in ceea ce priveste numele tabelului, numele coloanei, tipul de date, lungime, primary key…

Metadatele sunt salvate in tabele de System, numite Catalog de sistem, si sunt accesbile doar din DBMS. Doar userii cu permisiuni speciale le pot vedea sau modifica.

* **Indexi –** Pentru a prezenta diferite vederi ale datelor, indexii ofera o alternativa de a accesa datele.Cand stergem, adaugam sau modificam multe date, ar putea sa trebuiasca sa mutam informatia prin baza de date. Daca fisierul sau informatia sunt largi, pentru a le muta rapid, avem Indexi, adica biti pentru a reprezenta acea informatie. Deci nu lucram propriu zis cu fisiere, ci cu acesti biti ce le reprezenta, si asa operatiile sunt mult mai rapide.

Indexii permit bazei de date sa acceseze anumite instante(date) fara a trebui sa caute prin tot tabelul.

Modificarea datelor necesita si modificarea indexilor.

* **Metadate ale aplicatiei** – ofera facilitati, ca formele(introducerea usoara a datelor),sa vedem rapoarte(un sumar al bazelor de date),interogari

**Entitatile**

Pentru a crea o baza de date relationara, incepem mereu prin a crea **un model de asociere a entitatii**(Entity relationship model), numit pe scurt **ER** **Model.**

**Un ER model:**

* Descrie ce sunt entitatile pe care baza de date are de gand sa le inregistreze
* Descrie atributele/identificatorii entitatilor
* Descrie asocierile dintre entitati

**Entitatea este un obiect abstract de interes, sau in programare I se spune o clasa, care ne arata ce anume vor contine datele, din ce vor fi create**. Este abstract, deoarece nu contine nimic valori, ci doar arata ce va trebui sa contina obiectele. Entitatea nu se refera la un obiect particular din care extragem date, ci este un mode al unui obiect. De exemplu: angajati, studenti etc. Aici nu ne referim la un angajat sau student concret, ci la un model pentru fiecare. Entitatea este tabelul **gol**, ce are doar titlu si attribute, fara obiecte in el, adica fara linii.

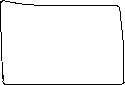
**Entitatea si relatia nu e una si aceeasi! Relatia se refera la tabelul cu date plin, entitatea la tabelul fara date!Relatia este instanta entitatii.**

**Diagrama asocierii cu entitate(Entity relationship diagram)**

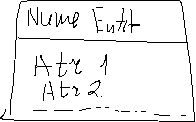
Pentru a reprezenta toate asocierele dintre tabele, nu e prea comod sa tot folosim modele(scheme) grafice, caci nu e clar care tabele cu care se unesc, si nici care e cardinalitatea.

Pentru diagrama de asociere de entitate, se folosesc diferite notatii, iar cea mai utilizata e **Crow’s Foot notation:**

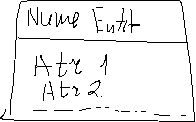
* **Entitate**



* **Enitate cu atribute**



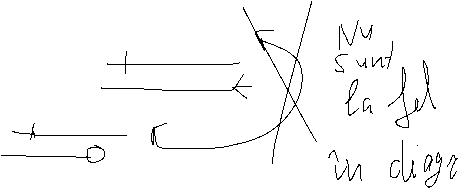
* **Entitate cu attribute si identificator/i**



* **Asocieri:**
* Nume(un verb deobicei)
* Cardinalitate(unu sau multe)
* Participare

Cardinalitate 1

Cardinalitate mai multe



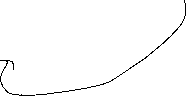
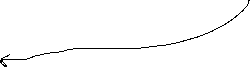
**Participare**:

- Obligatorie

- optionala

Exemplu(fara nume):

Asociere: una obligatorie (1 si numai 1) la una optionala (0 sau 1)



Asociere: una obligatorie (1 la 1) la multe obligatorii(1 sau N)



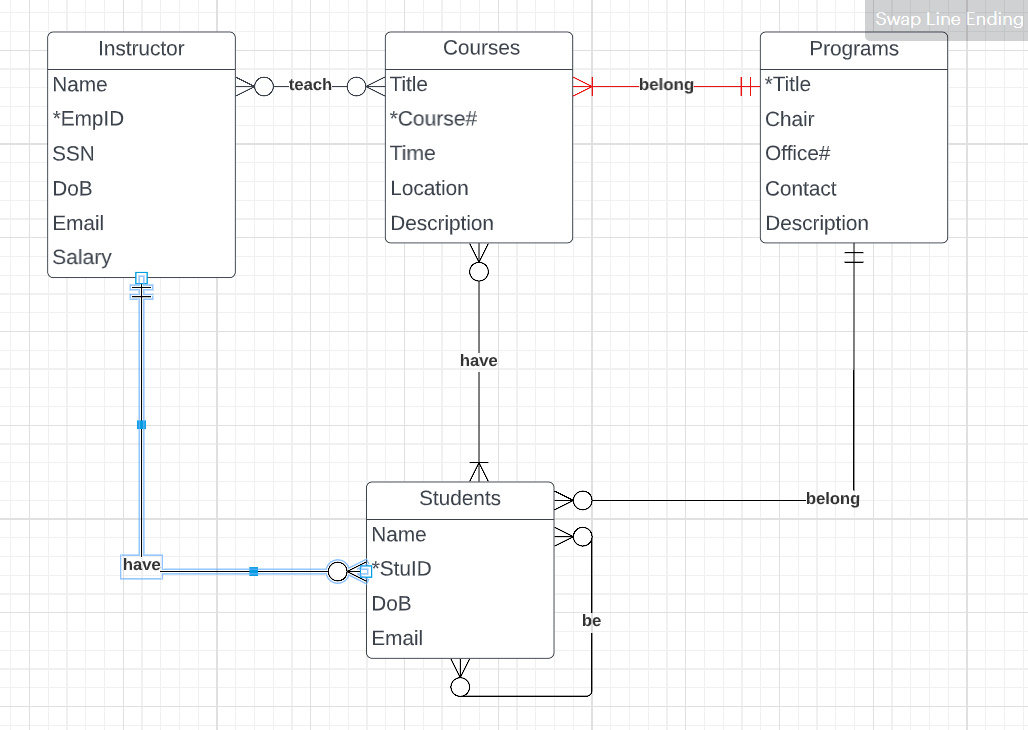
Multe optionale(0 sau N) la una optionala(0 sau 1)

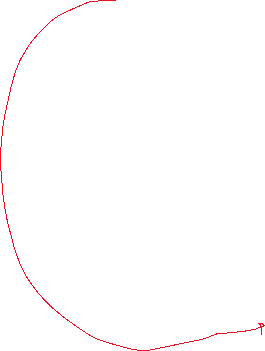
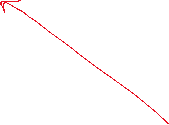


Multe optionale(0 sau N) la Multe obligatorii(1 sau N)



**In partea stanga ne arata cate elemente din tabelul 1 se conecteaza si in partea dreapta ne arata la cate cele din stanga se conecteaza l(cu cele din dreapta.)**

****



* Cardinalitatea asocierii fiecarei entitati se pune langa entitatea cu care se asociaza. Astfel, un instructor poate avea mai multi student (0 la N) si un student are doar un instructor (1 la 1). Aceasta cardinalitate se pune la **capat**,spre entitatea cu care are asociere, si la inceput, ca sa vedem cate din ele se conecteaza anume.
* Cardinalitatea se pune in partea cealalta. De ex, Instructorul poate avea 0 sau N studenti, deci asta e cerc liber langa Students. Un student trebuie sa aiba exact un instructor, deci asta e linie langa Instructor. Totodata, aceasta cadinalitate arata daca fiecare element din tabel trebuie sa fie legat de ceva sau nu. Daca in partea opusa e linie, inseamna ca fiecare element trebuie sa participe, daca e cerc, deci nu fiecare chiar.
* In mijloc punem o actiune
* Nu are importanta locul in care facem legaturile, atributele nu au nimic de a face cu asocierea.
* Un student poate fi prieten cu mai multi studenti(0 la N) si un student poate fi prietenul mai multor studenti (0 la N)
* Cand facem aceste scheme ne gandim la ce poate avea un obiect/obiecte din tabel in celelalt tabel.
* Daca avem entitate, 1 la N de exemplu, adica fiecare element din stanga se leaga de N elemente din dreapta, inseamna ca pentru tabelul din dreapta avem invers, N la 1.
* ID-urile le punem cu \*

**Este/trebuie sa** arata participare obligatorie.

**Poate** arata participare optionala.

**Mai multi** (0 la N) sau (1 la N) sau **unu sau mai multe**(1 la N)

**Atribute**

Atributele sunt caracteristici ale unei entitati de interes. Sunt lucruri pe care vrem sa le inregistram pentru entitati, adica numele coloanelor tabelului.

**Instanta a relatiei**

Este o inregistrare concreta, cu valori. Este o linie din tabel. Mai este si o instantiere a unei entitati, ca si in programare cum este o instantiere a unei clase.

**Identificatori**

Identificatorul este un atribut special(unic pentru fiecare) folosit pentru a putea gasi/identifica o specifica instanta a unei entitati. E folosit pentru a distinge toate instantele intre ele.

Identificatorul poate fi:

* Natural(nu trebuie neaprat creat) –proveni din afara bazei de date, nu il cream noi. Ele deja exista undeva, de exemplu codul SSN al unui angajat. De multe ori, un identificator natural uneste mai multe tabele(creaza o legatura intre ele). Nu e o idee prea buna sa fie setat ca identificator, deoarece deobicei contine date strict personale.
* Artificial - pe care il cream noi, ca ID

**Gradul unei relatii**

Numarul de attribute din tabel

**Cardinalitatea unei relatii**

Numarul de tuples(linii) din tabel

**Legatura/asociere(relationship)**

Reprezinta o asociere intre 2 sau mai mule tabele/entitati. Ne arata cum anume sunt legati membrii diferitor entitati/tabele.

* Exemplue fie ca avem Angajati si Companii. Ne intereseaza la cate companii poate un angajat lucra sau cati de multi angajati o companie poate avea. Aici ne trebuie o relatie intre tabele.
* Fie ca avem Companii si Produse. Ne intereseaza de cate companii poate un produs sa fie creat sau la cate companii apartine.

**Gradul asocierii**

**Gradul** unei asocieri este numarul de entitati(tabele) care participa in asociere. Ne arata cate tabele sun legate intre ele. Asocierele pot fi:

* de gradul 1 si sunt numite **Unare** sau **recursive**.(um tabel e legat tot de el)
* De gradul 2 – binare.(cel mai des utilizate)
* De gradul n

**Cardinalitatea asocierii**

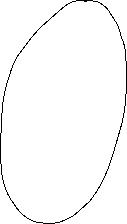
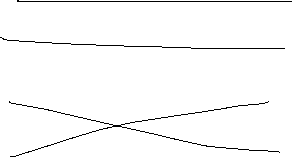
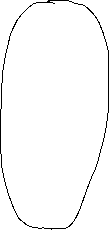
Ne arata cate obiecte din primul tabel se leaga de un anumit numar de obiecte din alt 2 tabel. E formata din 2 elemente:

**Nr1(primul elemen) la Nr2(al 2 element)**

Numarul 1 arata de cate elemente din primul tabel e vorba, si Nr2 de cate elemente din al 2 tabel e vorba. Asocierea scrisa arata cum se leaga elementele din primul tabel cu al doilea, iar al doilea tabel se asociaza cu primul in mod invers.

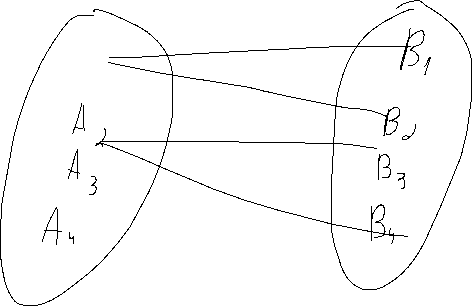
Tipuri:

* 1 la 1 – o instanta a unei entitati din A e conectata cel mult cu o instanta a altei entitati din B libera si o instanta a unei entitati din B e conectata cel mult cu una din A libera



* 1 la N – o instanta a unei entitati din A se conecteaza cu una sau mai multe instante libere din B, iar o instanta din B se poate conectata la cel mult una din A, chiar daca alta e deja conectata la ea,sau mai spunem ca mai multe entitati din B se pot conecta la una si aceeasi din A. Nu putem conecta o instanta din B la mai multe din A, caci asa se strica asocierea.

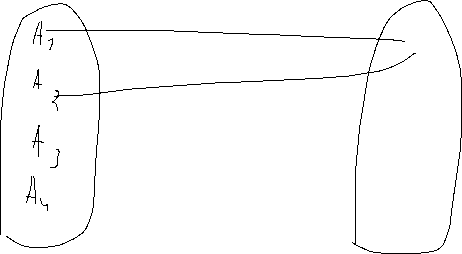
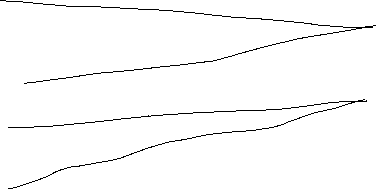
**Pe scurt: una din A la mai multe din B**



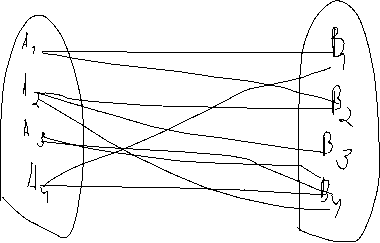
* N la 1 – fiecare instanta din A pote fi conectata cel mult la una din B, chiar de alta e deja conectata la ea,sau putem spune ca mai multe instante din A se pot conecta la 1 din B, si una din B poate fi conectata la mai multe libere din A



**Pe scurt: mai multe din A pot la una din B**



* N: M – multe la multe inseamna ca mai multe instante ale unei entitati pot fi conectate la mai multe instante ale altei entitati



**Pe scurt: Mai multe din A pot la mai multe din B si invers**

**Daca A are o cardinalitate (1 la N) fata de B, atunci B are invers, adica o cardinalitate (N la 1) fata de B. La fel e si cand A are cardinalitate (N la 1) fata de B, B are (1 la N) fata de A**

**Participarea asocierii**

Intr-o asociere, participarea poate fi de 2 tipuri:

* Obligatory – o instanta dintr-o entitate trebuie neaparat sa fie legata de o alta instanta din alta entitate(verbul **trebuie** e folosit pentru a o arata)
* Optionala - o instanta dintr-o entitate nu trebuie neaparat sa fie legata de o alta instanta din alta entitate(verbul **poate** e folosti)