CURS 1

De ce folosesc baze de date?

- nu am cum să creez legături între date
- posibilitatea să am date duplicate, cum păstrez integratatea lor
- trebuie să implementez ACID manual
- nu pot scrie mai mulți utilizatori în același timp

Dezavantaje DBMS:

- fișiere mari
- aplicații complexe
- costuri pentru hardware
- fișierele ar putea accesate mai greu

Ce este un sistem de baze de date?

Format din:

- hardware
  - să asigure persistența datelor
  - scanare volum mare de date
- software
  - DDL(Data Definition Language)
    - construiesc tabele, campuri
    - CREATE, DROP, ALTER
  - DML(Data Manipulation Language)
    - modificare date, citire
    - INSERT, SELECT, DELETE, UPDATE
  - DCL(Data Control Language)
    - grant, revoke
  - Controlul Tranzacțiilor
    - savepoint, commit, rollback
- utilizatori
- date

Trebuie să asigure securitatea, acces controlat, stocare, suport tranzacții, control concurent, recuperare date

Ce este o tranzacție și cum se face?

Grup de comenzi de modificare de date care trebuie executate împreună

- prima comanda DML
- se termina: commit, rollback, întâlnire DDL, inchidere sesiune
- ROLLBACK: aduce baza de date la starea de dinaintea începerii tranzactiei

O tranzacție trebuie să respecte **ACID**

- A - atomicitate
  - fie execut toata tranzactia, fie nu execut nimic
- C - consistență
  - se respectă ordinea operațiilor, în sensul în care rezultatul este conform înșiruiri de operații
- I - izolare
  - tranzacțiile sunt independente una de alta
- D - durabilitate
  - după commit rezultatul rămâne salvat și nu mai poate fi modificat

O tranzacție NERELAȚIONALA este BASE:

- B A- basic availability
  - în principiu este disponibilă
- S - soft state
  - locurile nu trebuie să fie consistente
- E - Eventual Consistency
  - uneori datele vor fi consistente

Comparație Nerelațional vs Rațional?

- Teorema CAP, doar două din următoarele pot fi respectată într-un sistem distribuit, motiv pentru care bazele de date relaționale nu pot fi distribuite
  - consistentă
  - disponibilitate
  - partition
- Deci sistemele nerelaționale sunt foarte utile în momentul în care vreau distribuirea sistemului, dar renunț la consistență sau la disponibilitate(decizie de business)

Modele din trecut:

- ierarhic
- rețea
- obiect-relational

Modele Prezente:

- relațional
    - stabilesc care sunt entitățile ce trebuie să memorez în baza de date şi construiesc asociere între tabele
- relațional%&

MODEL RELAȚIONAL

- **concepte**
    - relație = tabel
    - atribute = nume coloană
    - valoare/tuplu = rând
    - domeniu atribut = mulțimea de valori pe care le poate lua un atribut
- **proprietăți**
    - fiecare celulă are info
    - atribute unice
    - valorile unui atribut sunt din acelaşi domeniu
    - nu există linii identice
    - ordine arbitrară
- **chei:**
    - **supercheie** = atribut sau mulțime de atribute care identifica unic un tuplu(COLOANA CARE IDENTIFICA UN RAND)
    - **cheie candidat** = atribut sau mulțime de atribute care identifica unic un atribut şi nici o submulțime nu joacă rolul de cheie primară
    - **cheie alternativă =** cheie candidat care nu a fost selectată
    - **cheie străină =** atribut sau submultime de atribute care face referință la o cheie a altei relații
        - poate fi NULL şi nu trebuie să fie unică

ÎNTREBĂRI POSIBILE CURSUL 1

1. Ce înseamnă ACID?

   A - atomic: tranzacția este tratată ca o unitate, fie se realizează tot, fie nu se realizează nimic
   C - consistent: rezultatul tranzacției este conform secvențelor logice care o compun
   I - izolat: fiecare tranzacție este individuală, şi se realizează ca şi cum ar fi singura care se realizează în sistem la un moment dat
   D - durabilitate: odată salvată o tranzacție aceasta rămâne în starea în care e şi nu mai poate fi adusă înapoi
2. Ce este o tranzacție şi care sunt paşii acesteia?

   O tranzacție este o unitate de operații de tipul DML executate secvențial. O tranzactie începe cand fac prima comanda de tipul DML şi se încheie fie cand apare o eroare în sistem, este închis, se face commit sau rollback (o aduce în starea de dinaintea începerii tranzactiei), sau se face o comanda de tipul DDL
3. Ce oferă partea de software a unui SGBD?

   UN SGBD trebuie să ofere DDL(Create, Alter, Drop), DML(Insert, Select, Delete), DCL(grant) şi control tranzacții
4. Concepte model relațional:
    a. relația = tabel
    b. atribute = coloane
    c. tuple = linie
    d. domeniul unui atribut = mulțime de elemente din care ia valori un atribut
    e. Fiecare atribut e unic, elementele conțin toate o informație, ordinea nu contează, nu există linii identice

CURS 2

Elemente ale unui model relational

- **U - mulțime de atribute -** coloane
- **dom(Ai) -** domeniul elementelor de pe o coloană
- **tuplu = ti =** linie, câte o asociere din domeniu pentru fiecare atribut, pentru fiecare linie asociez o funcție
- **relație = r =** mulțime de uple = tabel
    - nu modificăm structura
- **R** - numele relației
- **R[U] - schemă de relație,** definesc un nume şi atributele peste care fac relația = schema tabelei
- **Schemă bază de date** = Mulțime de scheme de relație
- **Bază de date** = mulțime de relații efective

OPERAȚII

- **Din teoria mulțimilor**
    - **REUNIUNEA (U)**
        - aceleaşi atribute(NEAPĂRAT)
        - aceeaşi semnificație ca în cazul matematicii, toate elementele unice din ambele mulțimi implicate
        - Practică: ***UNION***

    - **INTERSECȚIA**
        - aceleaşi atribute(NEAPĂRAT)
        - aceeaşi semnificația, elementele comune

        - Practică***: INTERSECT***

    - **DIFERENȚA**
        - aceleași atribute(NEAPĂRAT)
        - aceeași semnificația, apar în r1 și nu sunt în r2
        - PRACTICĂ: ***MINUS***

    - **PRODUSUL CARTEZIAN**
        - nu trebuie să am aceleași atribute, intersecția atributelor să fie vidă
        - obțin fiecare combinație a atributelor din prima relație cu cele din a doua
        - PRACTICĂ: Select * from **R1, R2,** dacă am același atribut în ambele o să apară ambele variante, sgbd-ul le interpretează diferit sunt prefixate de numele relației
- **Specifice algebrei relaționale**

    - **Proiecție (pi)**
        - proiecția se face relativ la o mulțime de atribute X, pi X[t], proiecția lui t relativ la X
        - doar valorile din tuple asociate atributelor din X
        - **PRACTICA**: ***SELECT***

    - **Selecție (sigma)**
        - operație booleană între operanzi
        - expresie de selecție
            - theta = e | theta și theta| theta
        - **PRACTICA: *WHERE***

    - **Redenumire (ro)**
        - schimbă numele unui atribut cu un alt nume
        - **PRACTICA: AS**

    - **Join Natural**
        - mulțimea tuplelor peste atributele din prima relație și atributele din cea de-a doua relație, combinând doar atributele care au aceeași valoare
        - **PRACTICA: *natural join(inner join)***
        - Tuplele Folosite pentru join = semijoinul drept
        - Tuplele Nefolosite = antijoin drept
        - asociativ

    - **theta - JOIN**
        - join pe baza unei funcții de selecție
        - arată ca o alăturare
        - **PRACTICA: JOIN ON (condiție)**
    - **equijoin**
        - theta join prin egalitate
        - echivalent cu join natural
    - **Semijoin**
        - selectează randurile din stanga care au corespondent
    - **Antijoin**
        - selectează randurile din stanga care nu sunt implicate
    - **Divizarea**
    - **Join La Stânga**
        - completez cu NULL la dreapta, pentru ce nu are corespondent
        - **PRACTIC: LEFT JOIN**
    - **Join la Dreapta**
        - completez cu NULL la stanga, pentru ce nu are corespondent
        - **PRACTIC: RIGHT JOIN**
    - **Join Exterior**
        - completez în ambele părți cu NULL
        - **PRACTIC: FULL OUTER JOIN/ FULL JOIN**

ÎNTREBĂRI POSIBILE CURS 2

1. Care sunt operatorii și corespondentul lor în SQL?

- Operatori din teoria mulțimilor
    - reuniune, mulțime de atribute trebuie să fie aceeași, UNION
    - intersecție, mulțimea de atribute trebuie să fie aceeași, INTERSECT
    - diferența, mulțimea trebuie să fie aceeași, MINUS
    - produs cartezian, mulțimile de atribute nu trebuie să fie egale, SELECT * from s1, s2
- Operatori specifici algebrei relaționale
    - selecție(sigma), WHERE
    - proiecție(pi), proiecția relativă a lui t la X, SELECT
    - redenumire, schimbă numele unui atribut, ro A1/A1'(r), AS

- join natural
    - se combină pe câmpurile comune, NATURAL JOIN sau JOIN ON
- theta - join
    - join cu o condiție, produs cartezia + condiția
- equi - join
    - dacă condiția este egal, theta join devine equijoin
- semijoin drept/stâng
    - valori din stânga/dreapta implicată în join
- antijoin
    - valorile care nu au fost implicate în join
- join drept
    - Join + tuplele din dreapta neutilizate cu valori de NULL pentru atributele din prima, RIGHT JOIN
- join stâng
    - Join + tuplele din stanga neutilizate cu valori de NULL pentru atributele din prima, LEFT JOIN
- join complet
    - JOIN + tuple și din dreapta și din stanga nefolosite, FULL JOIN

CURS 3

Egalitatea a două tuple:

- două tuple sunt egale dacă sunt egale pe fiecare componentă

DEPENDENȚĂ FUNCȚIONALĂ

De la X la Y: (x, y mulțimi de atribute), dacă atunci când sunt egale pe X sunt egale și pe Y

Dacă am egal pe X, atunci sigur este egal și pe Y, dacă nu sunt egale nu contează dacă sunt egale sau nu. Singurul caz în care nu e dependența funcțională este când am două componente egale și cele din Y nu sunt egale.

- Dacă nu am duplicate: **Dependență funcțională trivială**
- nu e o relație simetrică: X -> Y, nu implică Y -> X

PROPRIETĂȚI

- FD1: REFLEXIVITATE
    - Dacă Y inclus in X, atunci X -> Y
- FD2: EXTENSIE
    - X -> Y și Z inclus în W, atunci am XW -> YZ
- FD3: TRANZITIVITATE
    - X -> Y, Y->Z, X -> Z
- FD4: PSEUDOTRANZITIVITATE
    - X -> Y, YW -> Z atunci am XW -> Z
- FD5: UNIUNE
    - X -> Y, X-> Z atunci am și X -> YZ
- FD6: DESCOMPUNERE
    - X -> YZ, am și X->Y, X->Z
- FD7: PROIECTABILITATE
    - X -> Y și X inclus in Z, X -> Y intersectat cu Z
- FD8: PROIECTABILITATE INVERSA
    - X -> Y e satisfăcută de o proiecție atunci e satisfăcută de rඩඩඩඩඩඩඩඩඩ

CONSECINȚA din Sigma

- dacă se obține aplicând proprietățile din mulțimea SIGMA

ACOPERIRE pentru Sigma

- o mulțime de dependențe, dacă din ea pot deduce aceleași dependențe funcționale

Axiomele lui Armstrong

- **reflexivitate, descompunere, compunere, tranzitivitate**
- R1 este echivalent cu RArmstrong, generează aceeași mulțime de dependente functionale

Întrebări posibile CURS 4

1. Ce este o dependență funcțională?
    a. X -> Y, orice tuple t1, t2 cu t1[X] = t2[X] -> t1[Y] = t2[Y]
    b. ne ajută să eliminăm coloane redundante, care pot fi aflate din alte coloane
2. Care sunt proprietățile unei dependențe funcționale?
    a. Reflexivitate
    b. Tranzitivitate
    c. Uniune
    d. Descompunere

    e. Extindere
    f. Pseudo Tranzitivitate (tranzitivitate cu un atribut în plus la a doua relație)
    g. Proiectabilitate
    h. Proiectabilitate inversă
3. Ce înseamnă că o mulțime de dependent este consecința din sigma
    a. pot sa aflu dependențele aplicând proprietatea pornind de la sigma
4. Ce înseamnă acoperire?
    a. Dacă cele două sunt egale
5. Ce sunt regulile lui armstrong?
    a. 4 reguli de inferență, reflexivitate, uniune, descompunere și tranzitivitate

CURS 4

DEPENDENȚE MULTIVALUATE

- X dependenta multivaluata Y
- oricare două tuple care sunt egale pe X, există T3 și T4 astfel incat
    - t1[X] = t3[X], t1[Y] = t3[Y] și t3[Z] = t2[Z], adică e egal pe X și T cu t1, dar este egal pe Z care e mulțimea de atribute fără X și Y cu t2
    - t4[X] = t2[X], t4[Y] = t2[Y], t1[Z] = t4[z]
- t2 poate fi egal cu t3, la fel și t4 cu t1

Definiție alternativa:

- mulțimea My pentru t1(XZ) = valorile lui Y când XZ sunt egate
- My pentru t1(XZ) = My pentru t2(XZ), dacă t1[X] = t2[X]
- fac My pentru toate egale și vad dacă obțin mereu aceleași valori

Observații

- dacă am dependența funcțională atunci am și dependeta multivaluata

Proprietăți semantice:

- MVD0: Complementariere
    - X dependete multivaluată la Y și XYZ = U, atunci X dependentă la Z
- MVD1: Reflexivitate
    - ca la dependente functionale
- MVD2: Extensie
    - ca la dependente functionale
- MVD3: Tranzitivitate
    - X dependeta multivaluata Y
    - Y dependenta multivaluata Z
    - atunci am X la Z minus Y
- MVD4: Pseudo Tranzitivitate
- MVD5: Uniune
    - ca la funcționale
- MVD6: Descompunere

Proprietăți combinate funcționale și multivaluate:

- dacă am dependența funcțională atunci am și dependeta multivaluata
- dacă am X multivaluata Y și XY funcțională Z atunci am și X funcțională Z minus Y

Sistem deductiv: R fm - functional și multivaluat contine:

3. funcționale
3. multivaluate
3. combinate

Pentru o mulțime de atribute X pot construi:

- X + cu dependente functionale
- X* - cu dependete multivaluate

BAZA DE DEPENDENȚE PENTRU X

ÎNTREBĂRI CURSUL 4

1. Ce este o dependență multivaluată(ambele definiții)?
    a. Pot elimina tuple, deoarece le pot reface ulterior
    b. X multivaluat Y, oricare t1[X] = t2[X], exista t3 și t4 astfel incat

        i. t3[x] = t1[X], t3[Y] = t1[X] şi t2[Z] = t3[Z]
        ii. cu t4 şi t2
    c. oricare t1[X] = t2[X] My(t1[XZ]) = MY(t2[XZ])
        i. MY(t[XZ]) = valorile lui Y când XZ sunt egale
2. Proprietăți
    a. Reflexie
    b. Extensie
    c. Tranzitivitate
    d. Uniune
    e. Descompunere(satisface reuniunea şi diferențele)
3. Reguli de inferență

CURS 5, 6

Normalizarea Bazelor de date:

- elimină redundanțele
- oferă consistență(ex. nu o să modific în ambele locuri dacă am două valori la fel)
- se face înaintea introducerii datelor
- pa baza schemei nu pe baza datelor

CONCEPTE de CHEI

- supercheie = atribut care identifica unic un tuplu
- cheie candidat = cheie pentru care nici o submulțime proprie nu e supercheie
    - avem o dependență funcțională de la cheie la orice atribut
    - X+, unde X este cheie trebuie să fie egal cu U, şi pentru orice submulțime nu am proprietatea asta
    - din mai multe atribute = MULTIVALUATĂ
- cheie alternativă = cheie candidat care nu a fost aleasă ca şi cheie primară
- cheie primară = cheie selectată pentru a identifica tuplele
    - nu e nulă
    - este unică
- cheie străină = cheie care face referire la tuplele(altă cheie candidat) dintr-o altă tabelă

Atribute Prime

- este parte dintr-o cheie candidat

Atribute Neprime

- nu e parte din nicio cheie candidat

Cum găsesc cheile?

- fac mulțimea cu plus, adică atributele care depind funcțional de o altă mulțime de atribute, pentru toate combinațiile de atribute, în timp fac dacă găsesc o cheie candidat elimin combinațiile care o conțin
- atributele care apar doar în stânga dependențelor sigur fac parte dintr-o cheie candidat**(PRIME), verific doar cheile care conțin astea**
    - **atributele care nu apar deloc sunt în partea stângă**
- atributele care apar şi în stânga şi în dreapta, au posibilitatea de a fi
- atributele doar din dreapta: sigur nu sunt parte dintr-o cheie candidat**(NEPRIME)**

DEPENDENTĂ PLINĂ

- X -> A, atunci e plină, dacă nu există o submulțime a lui X a.i X’ -> A
    - A nu este dependent de nici o submulțime a lui X

DEPENDENȚĂ TRANZITIVĂ

- **A este dependent tranzitiv de X,** dacă există Y, pot să ajung la A cu ajutorul tranzitivității
    - A aparține U\Y
    - X -> Y
    - Y -> A
    - dacă am astea doua am şi X -> A
    - Dar nu am Y -> X

DEPENDENTĂ TRIVIALĂ

- dependența de la supramulțime la submulțime (dacă de reflexivitate)

Nume	Definiție	Cum ajung acolo
1NF	Domeniile atributelor sunt **indivizibile** şi fiecare valoare a fiecărui **atribut este**	• elimin grupurile repetitive • identific tuplele care ar putea avea

	atomică	duplica ● creez o nouă schemă
2NF	1NF + Orice **atribut neprim** este **dependent plin** de orice **cheie** **(**Un atribut neprim nu depinde de o submulțime a cheii**)** **OBS.** dacă nu am **chei multivaluate** (din mai multe atribute)obligatoriu toate dependențele sunt pline	● Atributul care nu era dependent plin îl mut într-o tabelă nou și copiez atributele de care este dependent(cele din cheie)
3NF	2NF + Orice **atribut neprim NU este dependent tranzitiv** de nicio cheie a lui R Trebuie să depindă direct de chei, nu de alte atribute Orice atribut neprim oferă info legate de cheia întreagă(2NF) și nimic înafară de cheia(3NF)	● Mut atributul neprim împreună cu cel de care depinde într-o schemă nouă
BCNF = 3.5NF	1NF + pentru orice **dependență functionala netrivială X -> A** în sigma +, X este supercheie în schemă??? Este și în 3NF	Fac descompunerea fără pierdere până ajung în forma BCNF Pentru ce nu este în BCNF(X-> A, X nu e cheie) Aleg dependința și S1 = X reunit A S2 = Ri - A
4NF	1NF + pentru orice **dependență multivaluată** de la X la A, X este cheie pentru R	Aceeași pași doar că plec de la relațiile multivaluate incorecte

DESCOMPUNEREA DE TIP JOIN FĂRĂ PIERDERE

- pentru a aduce în BCNF
- DESCOMPUNERE DE TIP JOIN
    - am ro = mai multe scheme de relație
- DESCOMPUNERE DE TIP JOIN FĂRĂ PIERDERE
    - joinul final îmi dă relația inițială
    - ro este descompunere fără pierderi dacă fie cele două mulțimi în care descompun
        - **R1 intersectat cu R2 -> R1** minus R2 aparține sigma + sau invers

ÎNTREBĂRI CURS 5,6

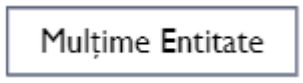
1. Tipuri de chei
    a. Supercheie
        i. atribut sau mulțime de atribute care identifica unic un tuplu
    b. Cheie candidat
        i. supercheie cu proprietatea că nici o submulțime nu este supercheie
    c. Cheie primara
        i. cheia candidat aleasă pentru a identifica tuplele din relație
    d. Cheie alternativa
        i. cheie candidat care nu a fost aleasă ca și cheie primară
    e. Cheie străină
        i. cheie dintr-o relatie care face referință la cheia candidat a altei relații
2. Ce este normalizarea și de ce am nevoie de ea?
    a. Prin normalizare eliminăm redundantele și asigurăm consistenta bazei de date. Aceasta se face înaintea introducerii datelor, asupra schemei bazei de date
3. Atribute Prime. Neprime. Dependențe pline. Dependențe tranzitive. Dependente triviale
    Un atribut este prim dacă face parte măcar dintr-o cheie candidat. Neprim dacă nu face parte din nicio cheie candidat. Atributele prime apar numai in stanga, in dreapta cele neprime și în mijloc pot fi ori prime ori neprime
    O dependenta X -> Y se numeste plina dacă nu există Z inclus in X astfel incat Z -> Y să aparțină mulții sigma(multimea de dependente)
    A este dependent tranzitiv de X dacă exista Y:
        A aparține U \ Y
        X -> Y
        Y-> A
    dar Y -> A nu e sigma
    Dependența funcțională trivială: fata de o submulțime
    Dependenta multivaluata trivială: fata de o submulțime + dacă X reunit cu Y acoperă toată mulțimea de atribute

4. 1NF
    a. Ce înseamnă? Toate domeniile atributelor să fie indivizibile și valorile să fie atomice
    b. Ca să aduc în 1NF: elimin grupurile repetitive şi creez noi scheme cu separate ca sa nu mai am duplicate
5. 2NF = 1NF + Toate atributele neprime trebuie să depindă plin de orice cheie candidat
    a. Găsesc depentețele care nu sunt pline adică cu submulțimi ale cheilor candidat şi mut atributul neprim acolo şi copii valorile submultimilor de care e dependent
6. 3NF = 2NF + Toate atributele neprime NU trebuie să depindă tranzitiv de nicio cheie din R
7. BCNF = 3.5 NF : 1 NF + X -> Y din sigma +, X este cheie în schema
    a. Ca sa aduc în BCNF folosesc operația de descompunere de tip join fără pierdere(R1 intersec R2 -> R1 \ R2 este sigma) până este in BCNF
8. 4NF = 1NF + orice dependență multivaluate X A din delta +, X este supercheie
    a. este 4NF este și BCNF
    b. :)

CURS 9

Modelarea Entitate/Asociere

Componente:

- **entitate**
    - obiect din baza de date
    - mulțime - entitate = grup de obiecte de acelaşi tip
    - 

- **asociere**
    - **c**onexiunea dintre una sau mai multe entități
    - poate avea diferite grade
        - unar, binar, ternar
    - 

- **atribut**
    - proprietate entitate
    - 

- **rol**
    - semnificația entităților în relație
- **cheie primară**
    - atribut sau submulțime generală de atribute ce identifica unic o instanță entitate
    - obligatoriu pentru entități
- **cheie străină**
    - folosită pentru asociere

CONSTRÂNGERI

_____Entitatea e implicată în minum u şi maxim v instanțe asociere

TIPURI DE CONSTRÂNGERI:

- **unu la unu**
    - 

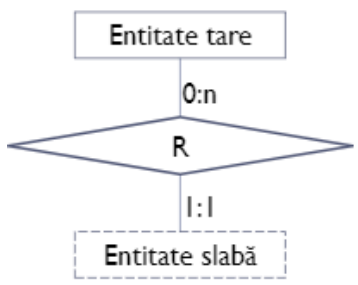
- **mulți la unu**
  - 

- **unu la multi**
  - 

- **mulți la multi**
  - 

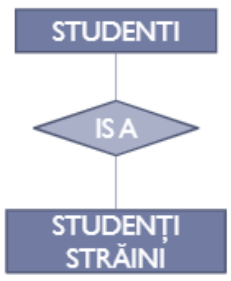
ENTITATE SLABĂ

- existența instanțelor sale depinde de existența altei clase
- nu are cheie



SPECIALIZARE

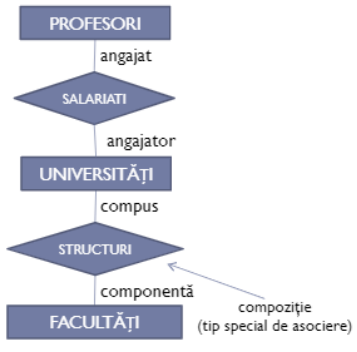
- subgrupuri distinctive de instanțe/entități
- au în plus anumite atribute



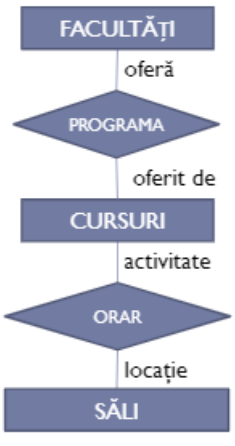
-
- relație de tipul IS-A
- poat fi
  - din punct de vedere al apartenenței la mai multe specializări
    - disjuncte
    - cu suprapunere
  - dacă toate au o specializare sau nu
    - complet
    - incomplet

CAPCANE

- **Fan traps:**

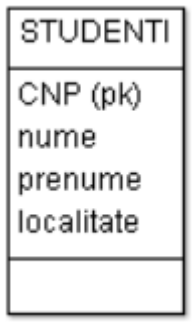
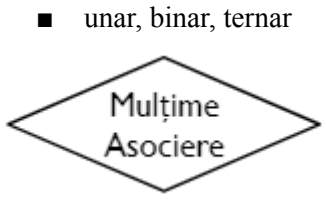
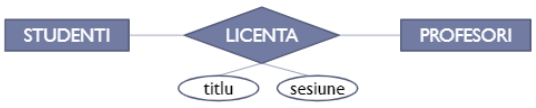
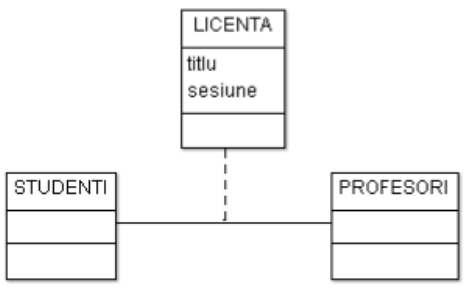
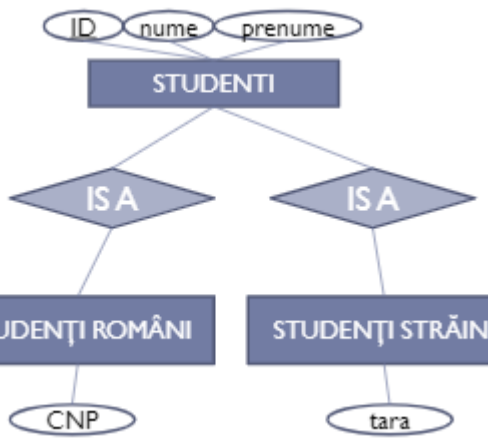
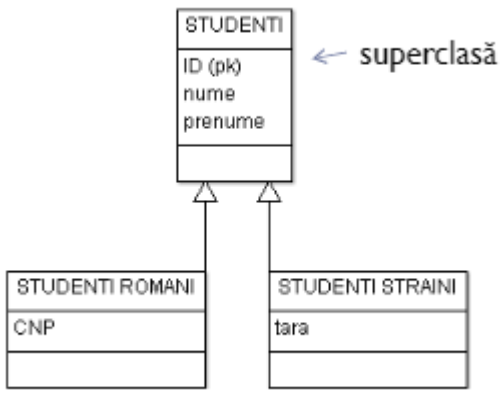


  - Problema:
    La ce departament aparține profesorul X?
- **Chasm**



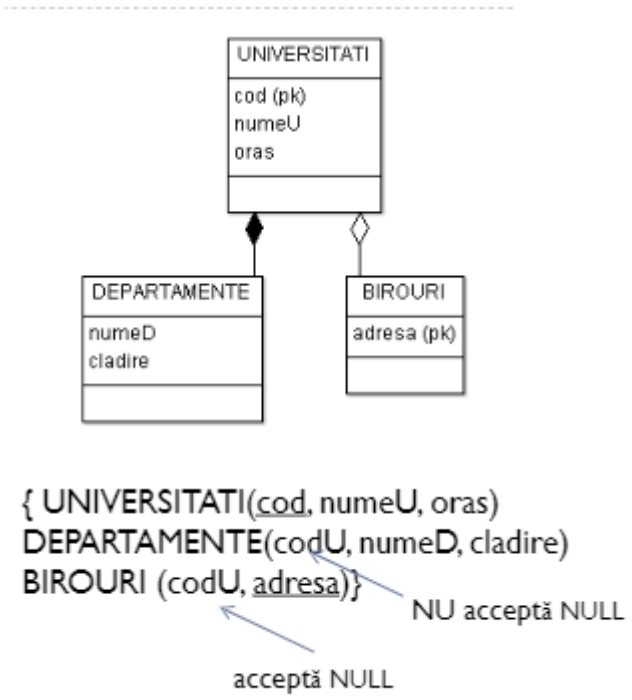
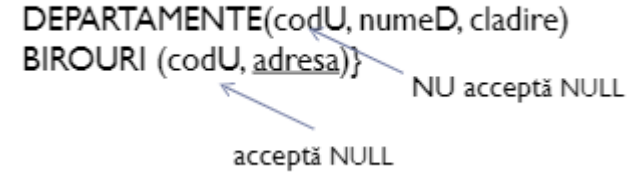
LEGĂTURA Relație/ Entitate - UML

<table>
<tr><th>E/R</th><th>UML</th><th>Schema BD relațională</th></tr>
<tr>
<td>Mulțime entitate cu atribute

● entitate
○ obiect din baza de date
○ mulțime - entitate = grup de obiecte de același tip
○  </td>
<td>Clasă
  </td>
<td>Relație cu chemă primară</td>
</tr>
<tr>
<td>Mulțime asociere fără atribute proprii
● asociere
○ conexiunea dintre una sau mai multe entități
○ poate avea diferite grade
■ unar, binar, ternar
 
○</td>
<td> 
Constrângerile sunt fix pe dos comparativ cu E/R
(Un student e îndrumat de exact un profesor și un Profesor îndrumă între 0 - 7 studenți)</td>
<td>Relație cu cheie străină</td>
</tr>
<tr>
<td>Mulțime asociere cu atribute proprii
  </td>
<td> 
Dacă multiplicitatea este 0.1 -> pot să mut atributele din clasa de asociere în clasa care participă la 0.1 instanțe de asociere</td>
<td>Relație cu cheie străină și alte atribute</td>
</tr>
<tr>
<td>Specializare
 
Poate fi:
● cu suprapunere
● fără suprapunere(disjunctă)
Sau
● completă
● incompletă</td>
<td>Subclasă
  </td>
<td>Relație cu cheie primară și atribute specializate

Mai multe posibilități:
● Relații subclasă ce conțin cheie superclasei și atributele specializate
● Relații subclasă ce conțin atributele superclasei și atributele specializare și superclasa conține doar tuple nespecializate
● O singură relație care conține și specializat și nespecializat</td>
</tr>
<tr>
<td>Cum e în general?

Clasei părți îi corespunde o entitate slabă(1..1, fără cheie primară)</td>
<td>Compoziție
Toate obiectele unei clase părți aparțin obiectelor dintr-o clasă compusă</td>
<td>Relație cu cheie străină și atribute particulare</td>
</tr>
</table>

		
Cum e în general? Multiplicitate 0..1	**Agregare** **Unele** obiecte dintr-o clasă aparţin obiectelor din clasă	**Relație cu cheie străină și atribute particulare** 

Întrebări posibile Curs 9

„Curs 10-11

Modelare fizică

- **Constrângeri**
  - restricționează stările posibile ca să elimine erori, să forțeze consistență, pentru a satisface corectitudinea
  - TIPURI
    - non - null
    - chei
    - integritate referențială
      - De la R.A la S.B, R.A este coloana care face referința(are cheie străină), S.B este cheia care este referită
        - fiecare valoare din R.A trebuie să fie o valoare din S.B
        - A cheie străină
        - B trebuie să fie cheie primară, sau măcar **unic**
      - **CUM E ÎNCĂLCATĂ?**
        - inserez în R.A
        - șterg în S
        - actualizări ori în R.A ori în S.B
      - **CUM REPAR**
        - **ștergere S.B**
          - **ON Delete Restrict (implicit) | SET NULL | CASCADE**
          - o să șterg peste tot
        - **actualizări pe S.B**
          - **ON UPDATE RESTRICT | SET NULL| CASCADA**
        - **Referințe circulare**
          - nu adaug constrângerile din prima, întâi creez o tabelă și apoi le adaug pe rând, ele sunt validate la finalul tranzacției, astfel nu o sa am probleme
    - bazate pe atribut şi pe tuplu
    - aserţiuni generale
      - create assertion key check(regula)
  - CUM?

- odată cu schema CREATE
- după ALTER
- INLINE

```
CREATE TABLE tabel (
 a1 tip not null, -- acceptă doar valori nenule
 a2 tip unique, --cheie candidat formată dintr-un singur atribut
 a3 tip primary key, -- cheie primară formată dintr-un singur atribut, implicit {not null,
   unique}
a4 tip references tabel2 (b1), --cheie străină formată dintr-un singur atribut
a5 tip check (condiție) -- condiția e o expresie booleana formulată peste atributul a5:
   (a5<11 and a5>4), (a5 between 5 and 10), (a5 in (5,6,7,8,9,10))…
)
```

- OUT - of -LINE

```
CREATE TABLE tabel (
a1 tip,
a2 tip,
a3 tip,
a4 tip,
primary key (a1,a2), --cheie primară formată din 2 (sau mai multe)
   atribute
unique(a2,a3), -- cheie candidat formată din 2 (sau mai multe) atribute
check (condiție), -- expresie booleană peste variabile declarate anterior:
   ((a1+a3)/2>=5)
foreign key (a3,a4) references tabel2(b1,b2) -- cheie străină multi-atribut
)
```

Declanșatoare

- monitorizează schimbările și inițiază acțiuni

```
Create Trigger nume
Before|After|Instead Of evenimente
[ variabile-referenţiate ]
[ For Each Row ]   -- actiune se execută pt fiecare linie modificată (tip row vs. statement)
[ When ( conditie ) ] -- ca o condiţie WHERE din SQL
actiune -- în standardul SQL e o comandă SQL, în SGBD-uri poate fi bloc procedural
```

-
- Evenimente:
  - BEFORE, AFTER, INSTEAD OF
- Variabile Referențiate
  - OLD TABLE
  - NEW TABLE
  - OLD ROW
  - NEW ROW

VIEW-uri - TABELE VIRTUALE

- **De ce?**
  - **vrea** să am acces modular la date
  - vreau să ascund informații
  - ușurez formulare de interogări
- CE ESTE?
  - interogare stocată formulate peste tabele sau alte view-uri
- CUM?
  -
    ```
    Create View numeView [a1,a2,...] As <frază_select>
    ```
- PROBLEME
  - Cum fac ca modificări asupra view-ului să se reflecte și în tabela inițială
    - rescriu comenzile cu INSTEAD OF astfel în loc să modific view-ul, modific tabela inițială
    - **VIEW-uri inerent actualizabile**
      - view creat cu Select fără DISTINCT pe o singură tabelă
      - cele care nu fac parte din view să poată fi NULL sau default
      - subinterogările nu referă T
      - nu există GROUP BY

VIEW MATERIALIZAT

-
  ```
  Create Materialized View V [a1,a2,...] As <frază_select>
  ```

- se creează un tabel nou V pe baza schemei
- +
    - crește viteza interogărilor
- -
    - poate fi prea mare
    - rămâne problema actualizării bazelor de date inițiale

ÎNTREBĂRI POSIBILE CURS 10-11

1. Tipuri de constrângeri de integritate
    a. NOT NULL
    b. Aserțiuni generale
    c. cheie primară
        i. nu poate fi null sau duplicat
    d. integritate referențială
        i. de la R.A la S.B
            1. S.b trebuie să fie sau măcar să fie declarată unic
            2. A este cheie străină
            3. Probeleme
                a. inserare în R
                b. ștergere S
                c. update R.A sau S.b
            4. pot sa rezolv cu declanșatori pentru ștergerea și update-ul din S(Set NULL | CASCADE)
    e. bazate pe atribut și pe tuplu
2. Ce este un View?
    a. Interogare stocată pentru a avea acces modular la date, ca sa ascund unele chestii și ca sa usurez formularea interogarilor
    b. Problema apare cand vreau sa modific un view, trebuie să se reflecte și în tabela de bază
        i. declanșatoare cu INSTEAD OF ca sa fac modificările direct pe tabela de bază
        ii. view inerent actualizabil
    c. View Materializat
        i. se creează efectiv o tabelă
        ii. ramane problema cu modificările

Curs 12-13

Calibrare/optimizare

1. **Cum se face accesul la date?**
    - timpul citirii unui bloc de date depinde de:
        - timpul de localizare(să poziționez capul de citire)
        - latență rotațională
        - timpul de transfer

    Select * From Student WHERE ID = 40, id = 40 este cheie de căutare, dacă datele sunt sortate după ea atunci pot să fac căutare binară
    - O(logN)
    -
2. **CONCEPTE DE BAZĂ**
    - **fișier de date**
        - secvență de blocuri de conține înregistrările
    - **cheie de căutare**
        - atributul care constituie criteriul de selecție
    - **cheie de sortare**
        - atributul după care e sortat fișierul de date
    - **fișier index**
        - asociat unei chei de căutare și conține înregistrări index:
            - valoare cheie : pointer
    - **index rar**
        - **nu stochează toate cheile**
    - **index des**
        - stochează câte o intrare pentru fiecare cheie

3. **Tipuri de indecși:**

- **ORDONAȚI -** cheile sunt ordonate
    - **secvențiali**
        - **INDEX PRIMAR:** cheia de căutare = cheia de sortare a fișierului de date(poste fi chiar cheie primară)
            - **DENS:** Un pointer pentru toate înregistrările cu aceeași valoare (am index primar deci fișierul este sortat după valorile pentru care fac index, deci ele sunt în ordine, valorile egale vor fi una după alta)
            - **RAR:** pot avea index rar, un index trimite un un bloc de date
        - **INDEX SECUNDAR:** cheia de căutare dă altă ordine față de ordinea din tabelul inițial
            - **DENS:** aș putea avea mai multe înregistrări pentru aceeași valoare
            - **RAR: nu pot deaorece fișierul nu e sortat pe baza chei de căutare**
        - **MULTI-LEVEL:** index pentru index
            - intern : peste date
            - extern: peste indexul intern

- **B+ arbori -** structuri secvențiale, sunt foarte utilizați în SGBD-uri

    - Structură
    - NOD
        - 

        - constanta m - specifică câte valori pot fi salvate într-un nod, poate fi calculat(dimensiune nod = dimensiune bloc de date)
        - valorile în subarborele către care trimite Pi sunt mai mici decât Ki, iar valorile către care trimite Pi+1 sunt mai mari
    - **PROPRIETĂȚI**
        - Rădăcina: 2 <= pointeri <= m+1
        - Intern: [(m+1)/2] <= pointeri <= m+1
        - Frunză: [m/2] <= pointeri < m
    - Frunzele reprezintă un index secvențial dens pentr fișierul de date (pot să fac căutări în interval)
    - **CĂUTARE**
        - log((m +1)/2)(k)
    - **INSERARE**
        - **2** (log((m +1)/2)(k))
        - găsesc nodul care ar trebui să conțină K
        - dacă valoarea există
            - adaug pointerul
        - dacă nu există
            - dacă am loc: inserez perechea
            - altfel, divid
    - **ȘTERGEREA**
        - **2** (log((m + 1)/2)(k))
        - găsesc valoare care conține k
        - dacă e parte din bucket, șterg direct de acolo (bucket gol = șterg valoare)
        - dacă am prea puține înregistrări în nod
            - mut în frunza alăturată dacă am loc
            - dacă nu am loc în frunza alăturată, împart nodurile între nodul curent și o frunză vecină
- **B-arbori**
    - ca B+ arbori, dar permit o singură apariție a unei chei
    - nu mai pot să fac căutări în interval
    - are adâncimea mai mare
    - posibil uneori să găsesc pe parcurs, înainte de frunză
- **HASH -** valorile cheii sunt uniform distribuite în grupuri numite buckets cu o funcție

    - valorile sunt grupate în bucketuri cu ajutorul unei funcții hash (bucket aproximativ egal cu bloc de memorie)
    - ineficienți pentru căutare în interval
    - **bucket =** unitate de stocare cu una sau mai mult înregistrări
    - **funcție hash = mapează valorile dintr-o dimensiune variabilă la una fixă**
        - **distribuție uniformă**
        - **asignări aleatorii**
    - **CĂUTARE**
        - face hash pentru valoare și scanează bucketul corespunzător = O(1)
    - **INSERARE**
        - face hash și îl înregistrează în bucketul corespunzător
    - **ȘTERGERE**
        - calculez hash, scanez, șterg
- **HASH DINAMIC:**
    - modific după necesitate numărul de bucketuri
    - generez valori într-o mulțime mare, 32 biți
    - utilizăm doar un prefix, a cărui dimensiune crește sau descrește după nevoi
    - +
        - nu scade performanța când creste memoria
        - minimizează alocarea de spațiu
    - -
        - tabela devine foarte mare(utilizez un B+ arbore)
        - modificarea tabele de adrese a costisitoare

- **BITMAP -** asociați atributelor categoriale/discrete, codifică ca o matrice binară
    - aplicabil dacă am puține valori

nume	prenume	str	loc	judet
Alexa	Marian	Strada Florilor	Cluj Napoca	Cluj
Popescu	Valentin	Strada Unirii	Dej	Cluj
Andrici	Ioana	Bulevardul Republicii	Vaslui	Vaslui
Acatrinei	Marcel		Putna	Suceava
Popescu	Vasile	Bulevardul Independentei	Oradea	Bihor
Costache	Ioan	Strada Teiului	Nucet	Bihor
Ungureanu	Daniel	Aleea Amara	Arad	Arad
Sandu	Maria	Strada Victoriei	Barlad	Vaslui

Arad	0 0 0 0 0 0 1 0
Bihor	0 0 0 0 1 1 0 0
Cluj	1 1 0 0 0 0 0 0
Suceava	0 0 0 1 0 0 0 0
Vaslui	0 0 1 0 0 0 0 1

    -
    - pentru fiecare înregistrarea pun 1 în dreptul valorii dacă apare în aceasta
    - dacă vreau să fac mai multe selecții folosesc operatorii pe biți între tabele

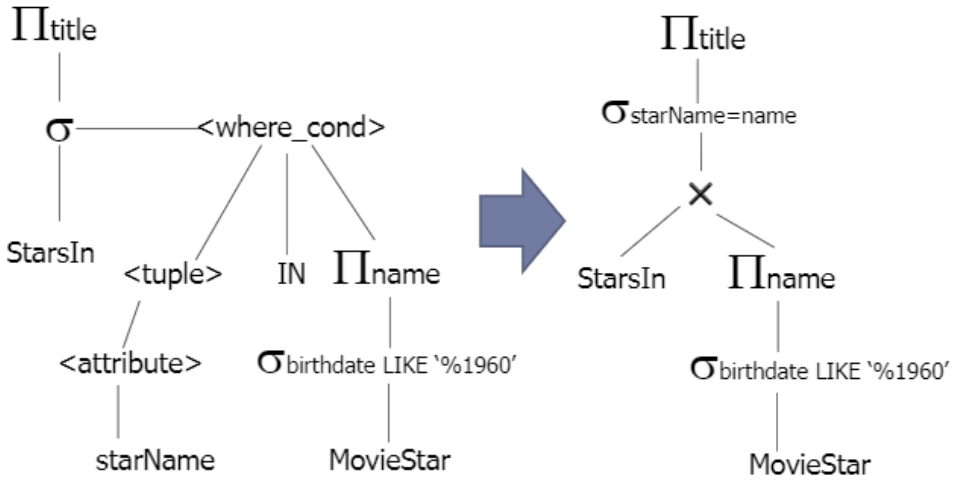
DEFINIRE

create index <index-name> **on** <relation-name>
(<attribute-list>)
E.g.: **create index** *c-index* **on** *student(judet)*

- Ștergere:
-
- **ORACLE: s**uport implicit B+ arbori
- recomandat după inserarea datelor în tabel
- CUM ALEG?
    - Coloane cu număr redus de valori distincte -> bitmap
    - Tip interval -> B+
    - Interogări punctuale -> hash
    - selecții cu funcții -> indecși definiți peste funcții

CURS 14

- Etapele Procesării Interogărilor
  - Analiza sintactică
    - Parsare
      - Se realizează Arbore de parsare
  - Analiza semantică
    - Preprocesare și rescriere în AR
      - ex. de Rescriere în AR



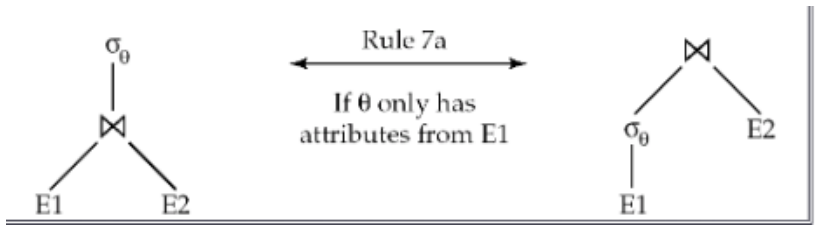
      - Rescriere apeluri la view-uri
      - Verificare existența relații
      - Verificare existența atribute și ambiguități
      - verificare tipuri
    - Selecție reprezentări algebrice
      - Plan logic
        - Operatori de bază
          - Selecție
          - Proiecție
          - Redenumire
          - Diferența
          - Produs Cartezian
          - Reuniune
        - Operatori adiționali
          - Intersecția pe mulțimi
          - Joinul natural
          - Agregarea(nu poate fi exprimată cu ajutorul operațiilor de bază)
            - avg
            - max
            - min
            - sum

$$\mathcal{G}_{\mathbf{sum(c)}}(\mathrm{r})$$

            - count
            - var
          - Joinul Extern
          - Teta-Joinul
    - Reguli de echivalență
      1. selecția e comutativă
      2. selecția de conjunctii = o secvență de selecții
      3. doar ultima proiecție e necesară într-un șir
      4. pot să combin o selecție cu un produs cartezian și teta join
         a. $\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$
         b. $\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$
         -
      5. Teta-join și join natural sunt comutative
      6. a. Joinul natural e asociativ

- $(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$

6. b. Theta join sunt asociative astfel

- $(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$
- 

7. Distribuția selecției asupra operatorului de teta-join
   a) când θ_0 implică atribute doar din una dintre expresiile (E_1) din join:

$$\sigma_{\theta 0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_\theta E_2$$

   b) când θ_1 implică numai atribute din E_1 și θ_2 implică numai atribute din E_2:

$$\sigma_{\theta 1 \wedge \theta 2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta 1}(E_1)) \bowtie_\theta (\sigma_{\theta 2}(E_2))$$

-
- În funcție de ce implică fiecare pot să mut astfel încât să fac mai întâi selecția

8. Distribuție **proiecție** asupra teta-join
   a) dacă θ implică numai atribute din $L_1 \cup L_2$:

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\Pi_{L_1}(E_1)) \bowtie_\theta (\Pi_{L_2}(E_2))$$

   b) Fie joinul $E_1 \bowtie_\theta E_2$

Fie L_1 și L_2 mulțimi de atribute din E_1 și respectiv E_2

Fie L_3 atribute din E_1 care sunt implicate în condiția de join θ, dar nu sunt în $L_1 \cup L_2$,

Fie L_4 atribute dinE_2 care sunt implicate în condiția de join θ, dar nu sunt în $L_1 \cup L_2$

$$\Pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \Pi_{L_1 \cup L_2}((\Pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\Pi_{L_2 \cup L_4}(E_2)))$$

9. **Reuniune și intersecție** comutative
10. **Reuniunea și intersecția** asociative
11. **Selecția** se distribuie peste **reuniune, intersectie și diferenta**
12. **Proiecția** se distribuie **peste reuniune**

METODE DE OPTIMIZARE:

- **Impingerea selecțiilor**, fac selecția cât mai devreme posibil
- **IMPINGEREA PROIECȚIILOR:** reduce dimensiunea joinului
- **Ordonare JOIN-URI**
  - fac mai întâi joinul cu dimensiuni mai mici

- Selecție algoritmilor și a ordinii
  - Plan fizic
    - Expresii în algebra relațională

- Estimare costuri interogări
- Algoritmi pentru Evaluare operatori/ expresii în algebra relațională
  - **ALGORITMI PENTRU SELECȚIE**
    - căutare liniară:
      - cost: br(număr blocuri) * tT(timp transfer) + ts(timp pentru localizare)
    - căutare binară
      - log(bt) * (tT + ts)
    - scanare index

- **ALGORITM PENTRU SELECȚII COMPLEXE**
  - Conjunctii
    - index + verificare condiții pe masura ce aduc tuplele în memorie
    - index multi-cheie
  - Disjuncție
    - reuniune identificatori
- **ALGORITMI PENTRU JOIN**
  - **Cu bucle imbricate**
    - două for-uri
  - **Indexat cu bucle imbricate**
    - căutarea index înlocuiește scanarea fișierelor dacă:
      - echi-join
      - exista un index pe atributul de join al relației interioare
  - **Cu Fuziune**
    - sortez relațiile după atributul de join și le unesc
    - doar echi - join
    - CU FUZIUNE HIBRID
      - o relație sortată și a doua are un index secundar pe atributul de join de tip B+ și fac
  - **Join HASH**
    - equijoin
    - o funcție hash care ia atributele de join partiționează tuplele ambelor relații în blocuri ce încap în memorie

ALTERNATIVE

- **Materializare**: subsexpresiile sunt materializate sub forma unor relații stocate pe disc
- **Pipelining :** tuplele sunt date ca intrare operațiilor de pe nivelul superior imediat ce sunt returnare

Întrebări de laborator?

Funcții de agregare: returnează o singură linie pe baza unui grup de linii
Pot fi utilizate in SELECT și în clauzele Order By și HAVING
Ex. COUNT, AVG, MAX, VARIANCE, SUM

SUBINTEROGARI NECORELATE

Fraze SELECT incorporate în clauzele WHERE, HAVING, FROM. Se execută odată înaintea interogării exterioare. Extrage date pe baza cărora se filtrează alte date.

SUBINTEROGARI CORELATE

În cazul subinterogarilor corelate, subinterogarea este evaluată pentru fiecare rând generat de interogarea principală. De această dată, la fiecare rulare, subinterogarea utilizează o valoare generată de interogarea principală. De exemplu vreau să compar media studentilor cu media anului în care se află. Devine corelată cand folosesc un atribut exterior

VARIABILE DE SUBSTITUȚIE

Suplinesc constante, condiții , nume de campuri, tabele

ÎNTREBĂRI

1. Algoritmi pentru Join-uri
   a. JOIN cu bucla imbricata, Join indexat cu bucla imbricata, Join cu Fuziune, Join cu Hash
2. Multivalente de la X -> Y
   a. dacă oricare t1 = t2 pe mulțime de atribute X exista t3 și t4 astfel incat
      i. t3[X] = t1[X], t3[Y] = t1[Y], t2[Z] = t3[Z], unde Z = U \ XY, etc
3. interogari corelate vs interogari necorelate

   Interogări corelate

   Efectuez subinterogarea pentru fiecare rând generat de interogarea principală. De fiecare data când utilizez valoare unei coloane returnată de interogarea principală

   Interogări necorelate:

   Fraze select incorporate in WHERE, HAVING, FROM

   Se execută o singură dată, înaintea interogării principale
1. View-uri: rol, clasificare, comportament la dml

   View: interogare stocată

ROL:

- limităm accesul utilizatorului la anumite date
- acces modular
- pot scrie mai ușor interogări

CLASIFICARE

- normal
- inerent modificabil
- materializat

Problema e ca nu modific și datele din tabele originală în cazul unor operații de actualizare

2. Relații și operații cu relații în modelul relational și corespondența în SQL (cel puțin 5 operații).
    a. Selecție(sigma) = WHERE
    b. PROIECȚIE(pi) = Select
    c. Redenumire(ro) = AS
    d. Pe mulțimi
        i. reuniune: union
        ii. intersecție: intersect
        iii. diferență: minus
        iv. produs cartezian
    e. JOIN natural
    f. Equijoin
    g. theta join
    h. semijoin
    i. join la dreapta(RIGHT JOIN)
    j. join la stanga

3. Agregarea înregistrărilor (aici trebuia explicat despre Group By și dat un exemplu și despre min, max, count)

    Funcții de agregare iau mai multe linii și rezultă un singur rezultat din acesta

    Se face cu group BY(nu pot utiliza valori individuale decât dacă sunt în clauza GROUP BY) pot fi utilizate in select, order by, **having - doar cu group by**, dar nu cu WHERE

    Count = numărul de linii

    MAX = Cea mai mare valoare de pe linie
4. Forme normale și rolul lor, discuție despre bcnf sau 4nf la alegere
    a. Rolul principal al formelor normale este acela de a elimina, posibile redundante. Normalizarea se face asupra schemei bazei de date
    b. BCNF este numit și forma 3.5NF și spune să dacă îl mulțimea sigma plus avem o dependența funcțională X -> A, atunci X este cheie pentru relație.
    c. Pentru a aduce în forma BCNF - apelez la operația de descompunere de tip join fără pierdere succesiv până îndeplinesc condițiile
5. Suportul SQL al tranzacțiilor (cand incepe o tranzactie, cand se termina⇒ referire la commit, rollback savepoint)

    O tranzacție începe la prima comanda DML sau explicit dacă scriu BEGIN TRANSACTION și se încheie când fac commit, rollback(mă întorc în ultimul stadiu salvat al bazei de date), pe parcurs pot să adaug un save point
6. Constrangeri posibile pe o baza de data SQL (chei și tipuri) -> vrea sa ii zici ce anume se intampla daca faci inserare pe o cheie primare (ceva de genul)

    cheie primare

    NOT-Null, Distinct

    cheie referențială

    asertiuni generale

    Not NULL
7. Dependențe funcționale și multivaluate. Definiție și exemple.