

## Linubaje QBE

Nu au răspundere foarte mare; se construiesc pe modelul relației

"Calcul Relational": o relație are atributele  $R(A_1, A_2, \dots, A_n)$

1) bazat pe n-upluri:  $R(t) \rightarrow$  rel  $R$  de n-uplu  $t$  (este o instanță a rel  $A_1, A_2, \dots, A_n$ )  $t \in (v_1, v_2, \dots, v_n)$

(tratează colecția datelor din rel ca fiind o mulțime de n-upluri)

$\{ \text{Angajat}(t) \mid t. D\_Nr = 2 \} \Leftrightarrow \text{SELECT * Angajat WHERE } D\_Nr = ?$

$\{ \text{Angajat}(t) \mid t. D\_Nr = 2 \text{ and } t. Salariu > 4000 \} \Leftrightarrow$

$S < D\_Nr = 2 \text{ and } Salariu > 4000 > \text{(Angajat)}$

2) bazat pe domenii:  $A_i \in \text{Dom}(A_i)$

care sunt multiniile posibile de val. ale atr.  $A_i$  (este def cînd se def tipul de date sau de constrângeri)

$\{ \text{Angajat}(\text{Name}, \text{Prenume}, \text{emp}) \}$  (echivalent cu op. Project din  
 $P < \text{listă atrbute} >$  (Relație)  $\text{Alg. Relat.})$

Operatia JOIN:

$\{ \text{Angajat}(t), \text{Departament}(u) \mid t. D\_Nr = u. D\_Nr \}$

QBE = query by example ; tabela  $T$  cîmp1  $T$  cîmp2  $T \dots T$  cîmpn

$\downarrow$   
tabela are o struct def

$\downarrow$   
la formularea unei cereri se completează un gablon dat de struct tablei (marcarea pe acest gablon a op. se dorește a fi efectuate, se utiliz. curv. cheie: CHECK - se poate as. tabelci sau ormai anumitor cîmpuri - marchează n-upluri)

sau domenile se vor fi cuplese în rez.)

Tabelă  $T_{\text{camp}1} T_{\text{camp}2} T \dots T_{\text{camp}n}$   $\Rightarrow$  se select. toată tabela  
CHECK  
CHECK ( $\checkmark$ )

Anagajat  $T_{\text{Nume}} T_{\text{Prenume}} T_{\text{cnp}} T_{\text{Salariu}} T_{D\_Nr} T \dots$   
 $\checkmark \quad \checkmark \quad \checkmark \quad >1000 \quad =2 \quad \dots$

În limbajul de manipulare a datelor apare astfel;

→ Query

Anagajat ; Nume ; Prenume ; cnp ; Salariu ; D\_Nr;  
; CHECK ; CHECK ; CHECK ; >1000 ; =2 ;

EndQuery

→ nu se specifică ordinea op Select sau Project

→ JOIN : Anagajat  $T_{\text{Nume}} T_{\text{Prenume}} T \dots T_{D\_Nr} T \dots ]$

Departament  $T_{D\_Nr} T_{D\_Nr} T \dots T \dots ]$

Def. un element exemplu: \_nume (se prefixează cu '\_')

=> elem. exemplu din Anagajat = elem. exemplu din Departament

SAU :  $T_{D\_Nr} T$

$T_{D\_Nr} T$

OBS: dacă se scriu condiții pe aceeași linie sunt automat QBE legături primă conectivă logică [ și ], atunci sănd se scriu condiții pe linii successive QBE legături condiții specifice [ SAU ].

În SQL există un mod de lucru asistat asemănător cu QBE.

## Dependente Funcționale, Normalizare în Baze de Date

OBS: Proiectarea unei structuri de BD este o activ. complexă în care intervin și implicit experiența proiectantului (nu există o "rezeta" de proiectare, depinde de cerințele aplicației; criteriu pt a decide dacă o BD este bine proiectată, având la dispoz. mai multe variante de implementare a BD pt o anumită aplicație)

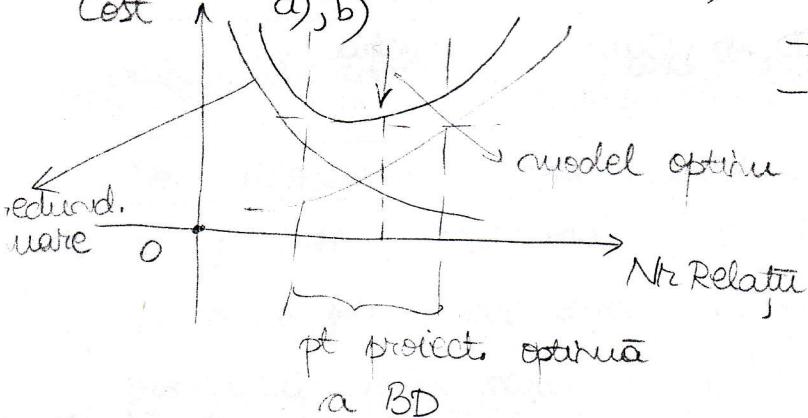
Paradoxul BD: BD  $\rightarrow$ 

- a)  $R(A_1, A_2, \dots, A_n)$  (total să se modeleze într-o singură tabelă)

- b)  $R_1, R_2, \dots, R_k$  (fiecare având un set liniu

de attribute:  $R_1(B_1, B_2, \dots, B_m)$ ,  $R_2(C_1, C_2, \dots, C_l)$ , ...

Cost  $\uparrow$       a), b)  
 $\downarrow$        $\rightarrow$  capacitate de stocare  
 redund.  $\downarrow$        $\rightarrow$  eficiență sereră  
 nere  
 uare



ex: Modelarea Companiei ca o singură tabelă:

Companie (Nume, Prez, ..., crp, ..., D-Nr, D-nume, Manager, ..., P-Nr, P-Num, etc, ...)  $\rightarrow$  redundanță maximă; o mulțime de câmpuri cu valuri nule și câmpuri care se repetă cu val

$\rightarrow$  întreținere BD: apar anomalii la modificările

structurei (modificări Manager, etc.)

### ① Semantica Atributelor:

OBS: În orice BD (oare schema relaț.) câmpurile trebuie să aibă o semantică liniu precisată, se referă la o sarcină a prop. modelatoră privind atribut, iar interpretarea valurilor atributelor este foarte usoră; ex: Angajat(Nume, Prez, crp, ...)

? Angajat(al, en, dhr, ...)

De regulă semantică atrb. conținute și o informație adică  
la domeniul Atributului ex: Name → domeniu sir de caractere

Dependențe funcționale:

Într-un model relațional se pot defini dependențe între  
valori care pot fi luate de atribut.

FD:  $X \rightarrow Y$   $X, Y =$  subșerturi ale atributelor relației și  
spunem că subșterea  $X$  restriționată subșterul  $Y \rightarrow Y$  este  
funcțional dep. de  $X$ .

Dacă într-o relație la care toate m-uplurile  $t(x) = t(x_1)$   
(subșterul  $X$  este identic cu toate m-uplurile)  $\Rightarrow t(y) = t(y_1)$

(valori subșterului  $Y$  sunt atât univoc de atât  $X$ )

ex:

Angajat (Name, Pren, cup, DNr, Salariu, Functia, ...)

$cup \rightarrow Name$

$cup \rightarrow Pren$

$cup \rightarrow \{Name, Pren\}$

$\{cup, Name\} \rightarrow \{Pren, Salariu\}$

Dacă în Angajat ( $\exists$ ) 2 m-upluri cu cup identice  $\Rightarrow$  Name să  
fie identice pt cele 2 m-upluri.

Procesul de normalizare se defineste pe baza dep. funcționale.

Reguli Armstrong (Reguli de inferență în dep. funcț.)

FD:  $X \rightarrow Y$  ? FD:  $Y \rightarrow X$

1. Regula de reflexivitate: dacă  $Y \subset X$ , ( $\exists$ ) FD:  $X \rightarrow Y$  =  
 $\Rightarrow Y \rightarrow X$  (dacă  $Y$  este un subset al membrului stâng)

2. Regula de argumentare:  $X \rightarrow Y$ ,  $Z \not\subset X$ ,  $Z \not\subset Y$   $\Rightarrow$   
 $\Rightarrow XZ \rightarrow YZ$  (adăugarea acelorași atribut la membr. stâng  
și la cel drept păstrează dep.)

3. Tranzitivitate:  $X \rightarrow Y$ ,  $Y \rightarrow Z \Rightarrow X \rightarrow Z$

4. Regula de deconpoziție:  $X \rightarrow YZ \Rightarrow X \rightarrow Y \wedge X \rightarrow Z$   
(concatenare ale atributelor  $Y, Z$ )

$X \rightarrow A_1, A_2, \dots, A_n \Rightarrow \begin{cases} X \rightarrow A_1 & (\text{daca } X \text{ se poate deduce } A_i, \\ & i=1:n) \\ X \rightarrow A_2 \\ \vdots \\ X \rightarrow A_n \end{cases}$

5. Regula de reunire:  $\begin{cases} X \rightarrow Z \\ X \rightarrow Y \end{cases} \Rightarrow X \rightarrow YZ$

Dep. funcțională completă: FD:  $X \rightarrow Y$  și  $A \subset X$  ( $A$  este un subset al lui  $X$ ),  $X \setminus A \not\rightarrow Y$  (daca eliminarea oricarei atrbute din subsetul stāng duce la pierderea dep. funcț)

Set minimal: se referă la subsetul stāng al FD, reprez. un minim de atrbute pt care FD se păstrează (FD este completă)

Normalizare → concept introdus în BD în 1972 de către Codd; se verifică faptul că dep. funcț. dintr-o relație respectă o serie de prop., iar dacă acestea nu sunt verif. se inițiază o procedură prin care struct. relaț. se modif. a.s. cerințele minime să fie îndepl.

În funcție de tipul de prop.: BD este normalizată pînă la un anumit ordin (ordinul de normalizare = măsură a normalizării)

FN1 (forma normală de ord 1): nu se bazează pe dep. funcț.; se spune că o relație este FN1 dacă nu conține o altă relație în terminologie de tabele: o tabelă nu poate conține o altă tabelă.

Ex: Departament (D\_Nume, D\_manager, D\_loc, ...), presup că ( $\exists$ ) dep. cu același nume în loc diferite (ex sucursale etc.)

"Proiectare", ..., (Cluj, Jaxi, București)

↓  
tabelă în cadrul D\_loc

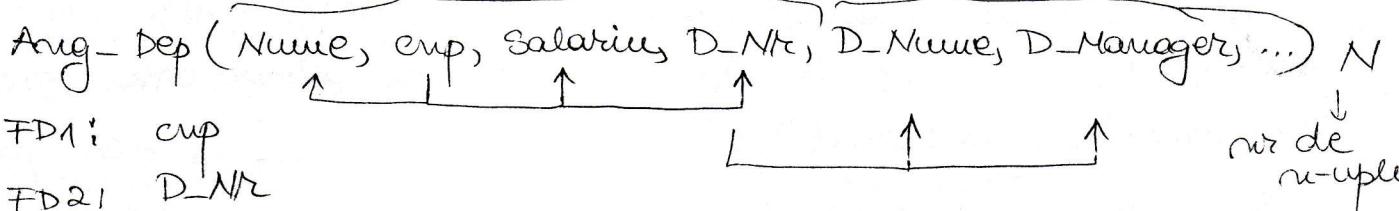
Anagajat (Nume (Fam, init, Pren), Adresa (Str, Nr, Oras), ...)

La atributele compuse nu se face FN1; se prevede că un atribut simplu: Angajat(Nume, huit, Prez, ... , str, Nr, Dr) respectă FN1

Sau: Angajat(Nume, cup, ..., Adresa, ...) → FN1

dacă este atr. compus, dacă nu se prelucrează componentele se păstrează ca atr. atomic

**FN2 (forma normală de ord 2)**: se bazează pe dep. funcț. transițivă; spunem că o rel. îndeplinește FN2 dacă între atr. sale nu pot fi identificat dep. funcț. transițive.



} FD1: cup → {Nume, Salariu, ..., D\_Nr}

} FD2: D\_Nr → {D\_Nume, D\_Manager, ...}

→ fd: cup → {Nume, Salariu, ..., D\_Nr, D\_Nume, D\_Manager, ...} (transz.)

Pt a normaliza FN2 această rel ⇒ se va descompune în 2 sau mai multe rel a.t. în fiecare să nu aibă transițiv.

Ang - dep1 (Nume, cup, Salariu, ..., D\_Nr) (cele dep de cup)

FD1: cup → {Nume, Salariu, ..., D\_Nr} (cheia candidat. este cup)

Ang - dep2 (D\_Nr, D\_Nume, ..., D\_Manager)

FD2: D\_Nr → {D\_Nume, ..., D\_Manager}

} card (Ang - Dep) =  $N_1 + N_2$

} card (Ang - Dep1) = N

} ord (Ang - Dep) =  $N_1 + N_2$

} ord (Ang - Dep1) =  $N_1$

} card (Ang - Dep2) ≤ N

} ord (Ang - Dep2) =  $N_2 + 1$

(restrictie inițială: Dep are minim 6 angajati)

Pt cazul 2 a rezultat redundanță; dacă se face update pt Departamente se modifică numai un re-uple în Ang - Dep2, apă

descriere de cazul 1, unde se face update pt toata tabă  
iar comparările nule în cazul 1 nu sunt posibile;

Eriicare cheie candidată într-o rel. poate constitui membru  
strang al unei FD (cheie candidată = fie primary key sau unic)  
(dintre toate cheile candidate doar una va fi primary key, restul vor fi unic)

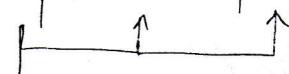
poate avea  
val NULL

**FN3** (forma normaliză de ord 3): o rel este normaliză de ord 3

dacă toate dep. funcț. sunt complete; o dată def setul  
minimal, toate alt. relației ce vor fi atașate în membrul  
deținut sunt funcț. complete.

ex: Ang-Pz (cnp, nume, prenume, ..., PNr, Prenume, Dre, ...)

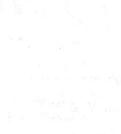
fd1:



fd2:



fd3:



{cnp, P-Nr} → {cnp, nume, prenume, ..., PNr, Prenume, Dre, ...}

Pt. Dre FD este completă, pt. celelalte nu e.

Aveam atât FD complete, cât și incomplete ⇒ nu e FN3 și  
nu trebuie normalizată:

→ Ang (cnp, nume, prenume, ...) FN3; normalizarea se face împărțind  
rel. în 3 rel FN3.

fd1: cnp → nume, prenume, ... FN3

fd2: cnp → Dre FN3

fd3: cnp → PNr FN3

→ Ang-Pz1 (cnp, PNr, Dre) FN3

OBS: Dacă avem mai multe chei candidatice avem urmă situații

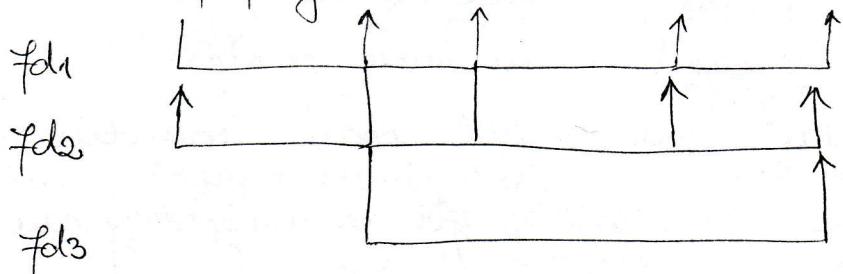
Lot (id\_prop, judet, Lot-Nr, Aria, Taxa)

↓  
(id de proprietate  
unic pe țară)

cheie candidată  
pt că Lot-Nr  
unic pe judet

↓  
taxa de  
impozitare  
stabilită pe  
judet (pe unitate de suprafață)

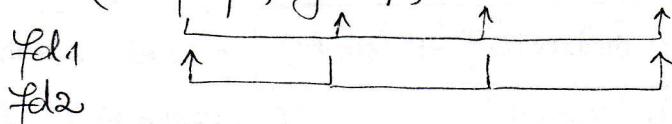
Lot (ID-prop, judet, Lot\_Nr, Aria, Taxa)



(pt că ID-prop este un pe ţară  $\Rightarrow$  toate celelalte depind de ea)

Judepl. FN2 sau FN3?

Decompozitie: Lot1 (ID-prop, Judet, Lot\_Nr, Aria)



Lot2 (Judet, Taxa) (contine un nr de inreg = nr de judepte)

Dacă Lot1 este FN2, nu este strict FN3, minimul  
 $ID\_prop \rightarrow \{Judet, Lot\_Nr, Aria\}$  Tabela are 2 chei candidate  
 $\{Judet, Lot\_Nr\} \rightarrow \{Id\_Prop, Aria\}$

**Def:** Spunem că o FN este FN strictă de ord 3 dacă nu conține attribute prime ale foreign key, de alte attribute primă se poate face normalizare strictă?  $\rightarrow$  Abău se referă la o cheie candidată  $\Rightarrow$  Lot1-1 (ID-prop, Judet, Aria) (Lot\_Nr nu mai este necesar, nu mai are sens)

FN Boyce Codd = FN strictă (2 sau 3); nu se recurge la forma strictă pt că pt prelucrare este suficient FN ce nu e strictă ex: este mai util să se manipuleze Lot\_Nr în cadrul județului și nu ID-prop la nivel de ţară  $\Rightarrow$  rezultat în prelucrare;

În general o BD este bine proiectată dacă este normalizată la ord 3, FN BC se aplică doar dacă prelucrările sunt ingreunțate de ea.

## Reguli de proiectare BD:

- 1) se modelează în aceeași rel. attributele unei entități tip (nu se mixează atr. specifice mai multor entități tip)
- 2) pt a surprinde rel. dintre entități tip se aduce la una dintre entit. tip în relație, dacă este posibil, un atribut chei candidată (de regulă primară) și entit. cu care este în relație
- 3) toate attributele ce nu se găsesc în relațiile aferente entit. tip se constituie într-o relație suplimentară, care poate fi normalizată sau nu; normaliz. acesteia este făcută în conjuncție cu cererile de prelucrare.
- 4) toate rel. aferente entit. tip se analiză d.p.v. al normalizării și se normalizează de regulă până la 3NF, în anumite situații 2NF.
- 5) dacă sunt FN sau neapărat înseanță că BD este bine proiectată; se poate decide să se lăsa BD nnormalizată dacă tipul de prelucrări nu afectează performanța.

→ Proiectarea BD

└ Aplicația se manipulează BD

## Programe de aplicații

De regulă acestea contin 3 componente specifice:

- 1) de tip meniu (tratat de un modul sau mai multe)
- 2) FORMS (screen - componentă ce se referă la toate interfețele utilizator; o interfață care are regulă și specifică: popularea BD, căutare a înreg ce îndeplinește anumite cond., pot include liste de selecție, butoane, spațiu pt text, ferestre de afișare în struct. specific de tabele = grid, se deschid în alte linșe, fie cu utilizarea modului BD)
- 3) REPORT (îndispensabilă, generează rapoarte având fie criterii predef. fie def dinamic; ieșire → screen, printer)