BAZE DE DATE

CURS 9

Necesitatea normalizării

- Anomaliile care apar în lucrul cu baza de date se produc din cauza dependenţelor care există între datele din cadrul relaţiilor bazei de date.
  - Dependenţele sunt plasate greşit în tabele!!!
- O dependență poate provoca:
  - **anomalii** la inserare, modificare sau ştergere
  - **redundanţă** în date
  - probleme de **reconexiune**.

Necesitatea normalizării

- **Normalizarea** are drept **scop**:
  - suprimarea **redundanţei** logice,
  - evitarea **anomaliilor** la reactualizare,
  - rezolvarea **problemei reconexiunii**.
- Există o teorie matematică a normalizării al cărei autor este E.F. Codd.
  - Soluţia: construirea unor tabele standard (forme normale).
- Normalizarea este **procesul reversibil de transformare a unei relaţii**, în relaţii de structură mai simplă.
  - Procesul este reversibil în sensul că **nicio informaţie nu este pierdută în timpul transformării**.
  - O relaţie este într-o formă normală particulară dacă ea satisface o **mulţime specificată de constrângeri**.

Necesitatea normalizării

- Relaţie universală + mulţime de anomalii
  - Orice formă normală se obţine aplicând o schemă de descompunere. Există două tipuri de descompuneri.
- **Descompuneri ce conservă dependenţele**.
  - descompunerea relaţiei R în proiecţiile R_1, R_2, ..., R_k, a.î. dependenţele lui R sunt echivalente (au închideri pseudo-tranzitive identice) cu reuniunea dependenţelor lui R_1, R_2, ..., R_k.
- **Descompuneri fără pierderi de informaţie** (*L-join*).
  - descompunerea relaţiei R într-o mulţime de proiecţii R_1, R_2, ..., R_j, a.î. pentru orice instanță a lui R este adevărată relaţia:

  $R = \text{JOIN}(\Pi_{B1}(R), \Pi_{B2}(R), ..., \Pi_{Bj}(R))$

  - Relaţia iniţială = compunerea naturală a relaţiilor obţinute prin descompunere.

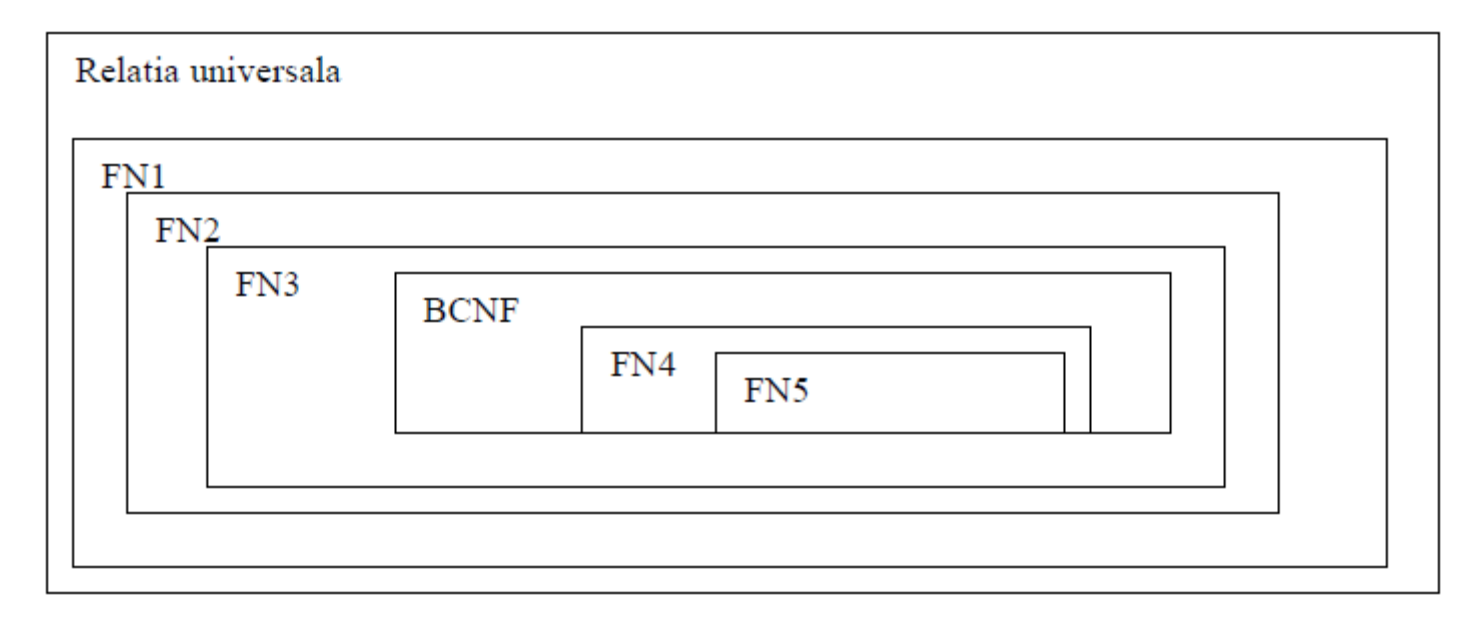
Necesitatea normalizării

- Formele normale sunt obţinute prin descompuneri fără pierderi de informaţie.
- O descompunere fără pierdere de informaţie, utilizată în procesul normalizării, este dată de **regula Casey-Delobel**:
- Fie $R(A)$ o schemă relaţională şi fie α, β, γ o partiţie a lui A. Presupunem că α determină funcţional pe β. Atunci:

$$R(A) = \text{JOIN}(\Pi_{\alpha \cup \beta}(R), \Pi_{\alpha \cup \gamma}(R)).$$

  - $\alpha \cup \beta$ → mulţimea atributelor care intervin în dependenţele funcţionale;
  - $\alpha \cup \gamma$ → reprezintă reuniunea determinantului cu restul atributelor lui A.

Necesitatea normalizării



Forma normală 1 (FN1)

- O relaţie este în prima formă normală dacă fiecărui atribut care o compune îi corespunde o valoare indivizibilă (atomică).
- Exemplu:

VEHICUL (Non FN1)

cod_persoana#	**vehicule**
P1	DL, RC, FF
P2	RM, VW
P3	DL

Forma normală 1 (FN1)

Varianta 1:

VEHICUL (FN1)

cod_persoana#	**vehicul#**
P1	DL
P1	RC
P1	FF
P2	RM
P2	VW
P3	DL

Varianta 2:

VEHICUL (FN1)

cod_persoana#	**vehicul1**	**vehicul2**	**vehicul3**
P1	DL	RC	FF
P2	RM	VW	null
P3	DL	null	null

Forma normală 1 (FN1)

Varianta 3:

VEHICUL 1

cod_persoana#	vehicul
P1	DL
P2	RM
P3	DL

VEHICUL 2

cod_persoana#	vehicul
P1	RC
P2	VW

VEHICUL 3

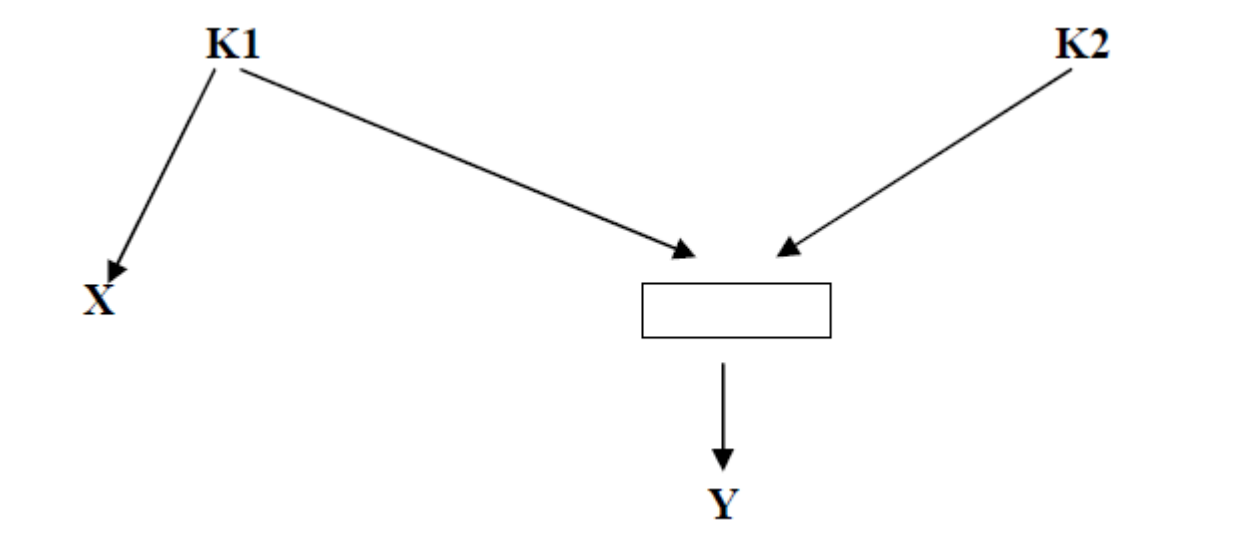
cod_persoana#	vehicul
P1	FF

Forma normală 2 (FN2)

- O relaţie *R* este în a doua formă normală dacă şi numai dacă:
  - relaţia *R* este în FN1;
  - fiecare atribut care nu este cheie (nu participă la cheia primară) este dependent de întreaga cheie primară.
- FN2 interzice manifestarea unor dependenţe funcţionale parţiale în cadrul relaţiei *R*!!!
- Ce consecință imediată se observă?

Forma normală 2 (FN2)

- Aplicarea regulii Casey-Delobel pentru FN2
  - Fie relaţia $R(K1, K2, X, Y)$, unde $K1$ şi $K2$ definesc cheia primară, iar X şi Y sunt mulţimi de atribute, astfel încât $K1 \rightarrow X$. Din cauza dependenţei funcţionale $K1 \rightarrow X$ care arată că R nu este în FN2, se înlocuieşte R (fără pierdere de informaţie) prin două proiecţii $R1(K1, K2, Y)$ şi $R2(K1, X)$.



Forma normală 2 (FN2)

- Exemplu:

atasat_la

Cod_salariat#	Job_cod	Nr_proiect#	Functia	Suma
S1	Programator	P1	Supervizor	60
S1	Programator	P2	Cercetator	25
S1	Programator	P3	Auxiliar	10
S3	Vanzator	P3	Supervizor	60
S5	Inginer	P3	Supervizor	60

De ce nu este în FN2?

Forma normală 2 (FN2)

- Transformarea în FN2:

atasat_2a

Cod_salariat#	Nr_proiect#	Functia	Suma
S1	P1	Supervizor	60
S1	P2	Cercetator	25
S1	P3	Auxiliar	10
S3	P3	Supervizor	60
S5	P3	Supervizor	60

atasat_2b

Cod_salariat#	Job_cod
S1	Programator
S3	Vanzator
S5	Inginer

Forma normală 2 (FN2)

- Exemplu:

Presupunem că un şantier poate executa mai multe lucrări de bază şi că o lucrare poate fi executată de mai multe şantiere.

LUCRARE(cod_obiectiv#, cod_lucrare#, nume);

SANTIER(nr_santier#, specialitate, sef);

EXECUTA(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator, data_inceput, data_sfarsit).

- Pentru relaţia EXECUTA sunt evidente dependenţele:

{cod_obiectiv#, cod_lucrare#} → {data_inceput, data_sfarsit},

{cod_obiectiv#, cod_lucrare#, nr_santier#} → {descriere, functie, conducator}.

- Relaţia EXECUTA este în FN1, dar nu este în FN2. Se aplică regula Casey Delobel:

EXECUTA_1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

EXECUTA_2(cod_obiectiv#, cod_lucrare#, data_inceput, data_sfarsit).

Forma normală 3 (FN3)

- **Intuitiv**, o relaţie *R* este în a treia formă normală dacă şi numai dacă:
  - relaţia *R* este în FN2;
  - fiecare atribut care nu este cheie (nu participă la o cheie) depinde direct de cheia primară.

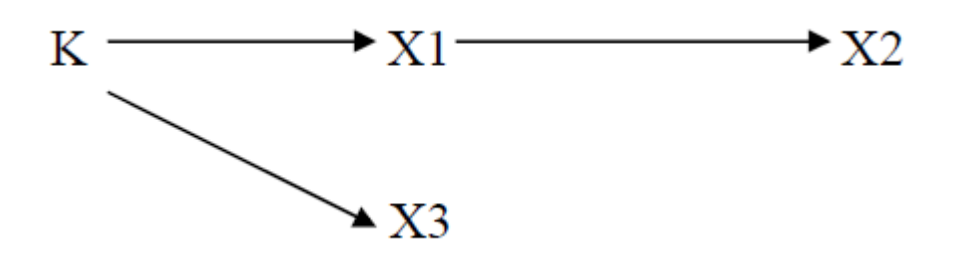
- Fie *R* o relaţie, *X* o submulţime de atribute ale lui *R* şi *A* un atribut al relaţiei *R*. *A* este **dependent tranzitiv** de *X* dacă există *Y* astfel încât $X \rightarrow Y$ şi $Y \rightarrow A$ (*A* nu aparţine lui *Y* şi *Y* nu determină pe *X*). *X* nu este dependent funcţional de *Y* sau *A*!
  - De exemplu, dacă $K_1, K_2, K_3 \rightarrow A_1$ şi dacă $K_1, K_2, A_1 \rightarrow A_2$, atunci $K_1, K_2, K_3 \rightarrow K_1, K_2, A_1$ şi $K_1, K_2, A_1 \rightarrow A_2$. Prin urmare, A_2 este dependent tranzitiv de K_1, K_2, K_3.

Forma normală 3 (FN3)

- **Formal**, o relaţie *R* este în a treia formă normală dacă şi numai dacă:
  - relaţia *R* este în FN2;
  - fiecare atribut care nu este cheie (nu participă la o cheie) nu este dependent tranzitiv de nici o cheie a lui *R*.
- **O relaţie este în FN3 dacă şi numai dacă fiecare atribut (coloană) care nu este cheie, depinde de cheie, de întreaga cheie şi numai de cheie.**

Forma normală 3 (FN3)

- Aplicarea regulii Casey-Delobel pentru FN3
  - Fie relaţia $R(K, X_1, X_2, X_3)$, unde atributul X_2 depinde tranzitiv de *K,* iar *K* este cheia primară a lui *R*. Presupunem că $K \rightarrow X_1 \rightarrow X_2$.
  - Din cauza dependenţei funcţionale $X_1 \rightarrow X_2$ care arată că *R* nu este în FN3, se înlocuieşte *R* (fără pierdere de informaţie) prin două proiecţii $R1(K, X_1, X_3)$ şi $R2(X_1, X_2)$.



Forma normală 3 (FN3)

- Exemplu:

atasat_2a

Cod_salariat#	Nr_proiect#	Functia	Suma
S1	P1	Supervizor	60
S1	P2	Cercetator	25
S1	P3	Auxiliar	10
S3	P3	Supervizor	60
S5	P3	Supervizor	60

De ce nu este în FN3?

Forma normală 3 (FN3)

- Transformarea în FN3:

atasat_3a

Cod_salariat#	Nr_proiect#	Functia
S1	P1	Supervizor
S1	P2	Cercetator
S1	P3	Auxiliar
S3	P3	Supervizor
S5	P3	Supervizor

atasat_3b

Functia	Suma
Supervizor	60
Cercetator	25
Auxiliar	10

Forma normală 3 (FN3)

- Exemplu:

În tabelul EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator) continuă să existe redundanţă în date.

- Atributul *conducator* depinde indirect de cheia primară prin intermediul atributului *functie*. Între atributele relaţiei există dependenţele:

{cod_obiectiv#, cod_lucrare#, nr_santier#} → {descriere},

{cod_obiectiv#, cod_lucrare#, nr_santier#} → {functie} → {conducator}.

- Se aplică regula Casey-Delobel. Relaţia se descompune, prin eliminarea dependenţelor funcţionale tranzitive, în proiecţiile:

EXECUTA11(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie)

EXECUTA12(functie, conducator).

Schema de sinteză pentru obţinerea lui FN3

- **Algoritmul de sinteză** construieşte o acoperire minimală *F* a dependenţelor funcţionale totale.
  - Se elimină atributele şi dependenţele funcţionale redundante.
  - Mulţimea *F* este partiţionată în grupuri F_i, astfel încât în fiecare grup F_i sunt dependenţe funcţionale care au acelaşi membru stâng şi nu există două grupuri având acelaşi membru stâng.
  - Fiecare grup F_i produce o schemă FN3.
- Algoritmul realizează o descompunere ce conservă dependenţele.

Schema de sinteză pentru obţinerea lui FN3

- **Algoritm SNF3** (aducerea unei relaţii în FN3 prin utilizarea unei scheme de sinteză):

1. Se determină *F* o acoperire minimală a lui *E* (mulţimea dependenţelor funcţionale).
2. Se descompune mulţimea *F* în grupuri notate F_i, astfel încât în cadrul fiecărui grup să existe dependenţe funcţionale având aceeaşi parte stângă.
3. Se determină perechile de chei echivalente (*X*, *Y*) în raport cu *F* (două mulţimi de atribute X, Y sunt chei echivalente dacă în mulţimea de dependenţe E există atât dependenţa X →Y, cât şi dependența Y → X).

Schema de sinteză pentru obţinerea lui FN3

4. Pentru fiecare pereche de chei echivalente: se identifică grupurile F_i şi F_j care conţin dependenţele funcţionale cu partea stângă X şi respectiv Y; se formează un nou grup de dependenţe $F_{i\,j}$, care va conţine dependenţele funcţionale având membrul stâng (X, Y); se elimină grupurile F_i şi F_j, iar locul lor va fi luat de grupul $F_{i\,j}$.
5. Se determină o acoperire minimală a lui F, care va include toate dependenţele $X \rightarrow Y$, unde X şi Y sunt chei echivalente (celelalte dependenţe sunt redundante).
6. Se construiesc relaţii FN3 (câte o relaţie pentru fiecare grup de dependenţe funcţionale).

Schema de sinteză pentru obţinerea lui FN3

- Se observă că **algoritmul solicită**:
  - determinarea unei **acoperiri minimale** (algoritmii EAR şi EDF);
  - determinarea **închiderii (A^+) unei mulţimi de atribute A** în raport cu mulţimea de dependenţe funcţionale E (algoritm AIDF).
- Determinarea acoperirii minimale presupune eliminarea atributelor şi dependenţelor redundante.
- Acoperirea minimală **nu este unică** şi depinde de ordinea în care sunt eliminate aceste atribute şi dependenţe redundante.

Schema de sinteză pentru obţinerea lui FN3

- **Algoritm EAR** (elimină atributele redundante din determinantul dependenţelor funcţionale)
- Pentru fiecare dependenţă funcţională din E şi pentru fiecare atribut din partea stângă a unei dependenţe funcţionale:
  - **Pas1**. Se elimină atributul considerat.
  - **Pas2**. Se calculează închiderea părţii stângi reduse.
  - **Pas3**. Dacă închiderea conţine toate atributele din determinantul dependenţei, atunci atributul eliminat la pasul 1 este redundant şi rămâne eliminat. În caz contrar, atributul nu este redundant şi se reintroduce în partea stângă a dependenţei funcţionale.

Schema de sinteză pentru obţinerea lui FN3

- **Algoritm EDF** (elimină dependenţele funcţionale redundante din E)
- Pentru fiecare dependenţă funcţională X →Y din E:
  - **Pas1**. Se elimină dependenţa din E.
  - **Pas2**. Se calculează închiderea X^+ , în raport cu mulţimea redusă de dependenţe.
  - **Pas3**. Dacă Y este inclus în X^+, atunci dependenţa X →Y este redundantă şi rămâne eliminată. În caz contrar, se reintroduce în E.

Schema de sinteză pentru obţinerea lui FN3

- **Algoritm AIDF** (determină închiderea lui A)
  - **Pas1.** Se caută dacă există în E dependenţe X →Y pentru care determinantul X este o submulţime a lui A, iar determinatul Y nu este inclus în A.
  - **Pas2**. Pentru fiecare astfel de dependenţă funcţională se adaugă mulţimii A atributele care constituie determinatul dependenţei.
  - **Pas3**. Dacă nu mai există dependenţe funcţionale de tipul de la pasul 1, atunci $A^+ = A$.

- Exemplu!

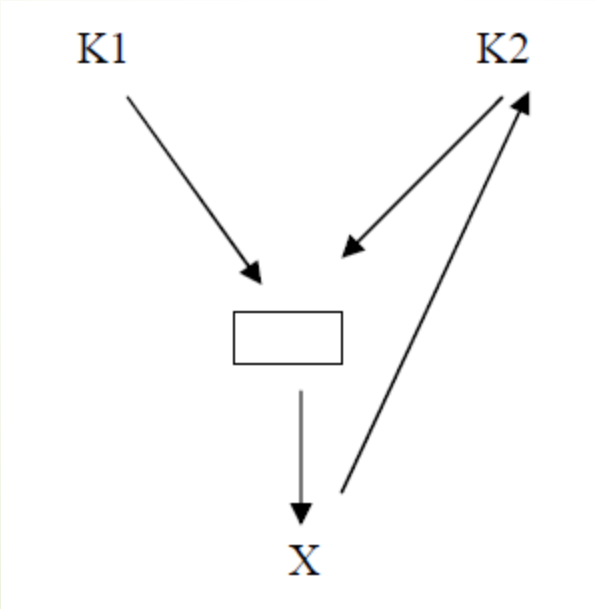
Forma normală Boyce-Codd (BCNF)

- Determinantul este un atribut sau o mulţime de atribute neredundante, care constituie un identificator unic pentru alt atribut sau altă mulţime de atribute ale unei relaţii date.
- **Intuitiv**, o relaţie *R* este în forma normală Boyce-Codd dacă şi numai dacă **fiecare determinant este o cheie candidat**.
- **Formal**, o relaţie *R* este în forma normală Boyce-Codd dacă şi numai dacă pentru orice dependenţă funcţională totală $X \rightarrow A$, *X* este o cheie (candidat) a lui *R*.

Forma normală Boyce-Codd (BCNF)

- Regula Casey Delobel pentru R(K1#, K2#, X) presupunând că există dependenţa: X → K2.

  ➔ R1(K1#, X) şi R2(X#, K2)



Exemplu!

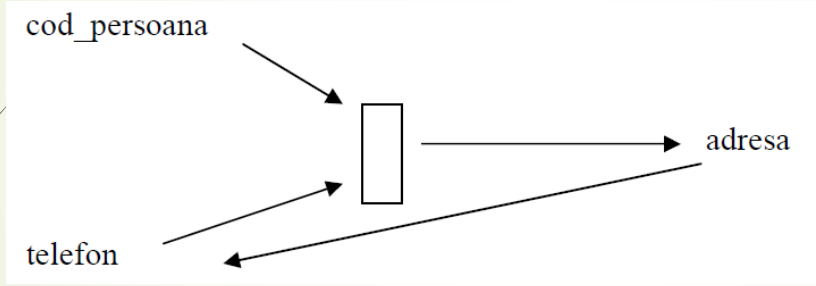
Forma normală Boyce-Codd (BCNF)

- Pentru ca o relaţie să fie adusă în BCNF nu trebuie în mod obligatoriu să fie în FN3.
- Se pot aduce în BCNF şi relaţii aflate în FN1 sau FN2.
  - Acest lucru este posibil întrucât dependenţele funcţionale parţiale şi cele tranzitive sunt tot dependenţe noncheie, adică dependenţe ai căror determinanţi nu sunt chei candidat.

Forma normală Boyce-Codd (BCNF)

- Exemplu:

ADRESA(cod_persoana#, telefon#, adresa)



- În dependenţa *adresa _ Telefon* se observă că determinantul nu este o cheie candidat. Relaţia ADRESA se descompune în:

ADRESA_1(cod_persoana#, adresa);

ADRESA_2(adresa#, telefon).

- Relaţiile sunt în BCNF, se conservă datele, dar nu se conserve dependenţele (s-a pierdut *cod_persoana, telefon → adresa*).

Forma normală Boyce-Codd (BCNF)

- Exemplu:
- Relaţia INVESTESTE_IN leagă entităţile INVESTITOR şi OBIECTIV_INVESTITIE. Ea are schema relaţională:

INVESTESTE_IN(cod_contractant#, cod_obiectiv#, nr_contract, cota_parte).

- Între atributele relaţiei există dependenţele:

{cod_contractant#, cod_obiectiv#} → {nr_contract, cota_parte},

{nr_contract} → {cod_obiectiv}.

- Se aplică regula Casey-Delobel şi se aduce relaţia în BCNF.

INVESTESTE_IN_1(cod_obiectiv, nr_contract#);

INVESTESTE_IN_2(cod_contractant#, nr_contract, cota_parte).

Forma normală Boyce-Codd (BCNF)

- **Algoritm TFBCNF** (aducerea unei relaţii *R* din FN1 în BCNF)

1. Dacă relaţia conţine cel mult două atribute, atunci *R* este în BCNF şi algoritmul s-a terminat.
2. Dacă relaţia conţine mai mult de două atribute, se consideră toate perechile (*X*, *Y*) de atribute distincte din *A*.
3. Se determină A_1^+, închiderea mulţimii $A_1 = A - \{X, Y\}$.
4. Dacă pentru orice pereche (*X*, *Y*), $X \notin A_1^+$ atunci relaţia *R* este în BCNF şi algoritmul s-a terminat.
5. În caz contrar (pentru cel puţin o pereche (*X*, *Y*), *X* aparţine lui A_1^+), relaţia *R* nu este în BCNF.
6. Se reduce progresiv schema relaţiei şi se reia algoritmul, exploatând relaţia redusă. Orice relaţie obţinută prin reducerea lui *R* şi care este în BCNF se consideră ca făcând parte din descompunerea lui *R* în procesul aducerii sale în BCNF.

Forma normală 4 (FN4)

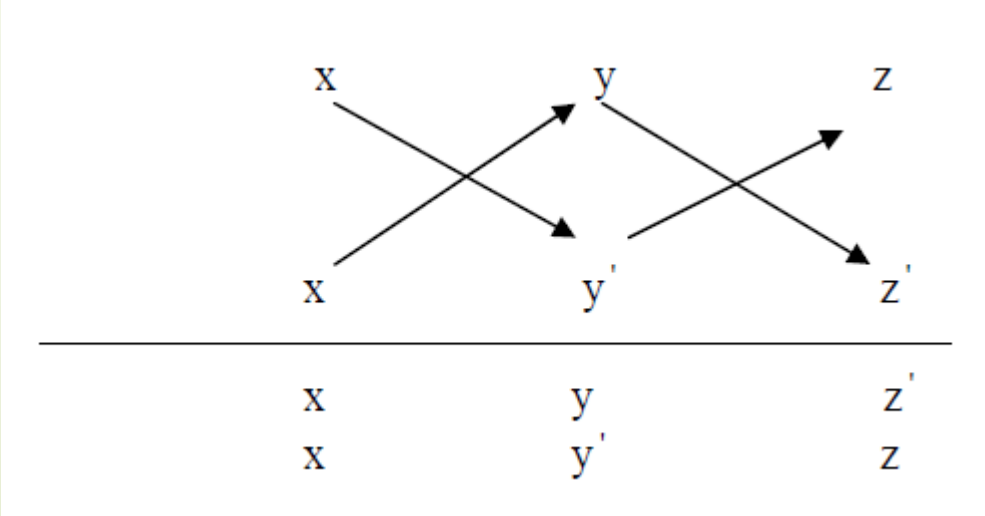
- FN4 elimină redundanţele datorate relaţiilor *m*:*n*, adică datorate **dependenţei multiple**.
- **Intuitiv**, o relaţie *R* este în a patra formă normală dacă şi numai dacă relaţia este în BCNF şi nu conţine relaţii *m*:*n* independente.

Forma normală 4 (FN4)

- Fie R o relaţie definită pe o mulţime de atribute $A = \{A_1, A_2, ..., A_n\}$ şi fie $X, Y, Z \subset A$. Se spune că X **multideterminină** pe Z sau că Z este multidependent de X :
  - dacă pentru fiecare valoare a lui Z în R există numai o valoare pentru perechea (X, Y);
  - dacă valoarea lui Z depinde numai de valoarea lui X.
- Acest tip de dependenţă, numită şi multivaloare sau multidependenţă (MVD) se notează prin $\boldsymbol{X \rightarrow\!\!\!\rightarrow Z}$.
- **Intuitiv, multidependenţa reprezintă situaţia în care valoarea unui atribut (sau a unei mulţimi de atribute) determină o mulţime de valori a altui atribut (sau mulţimi de atribute)!!!**

Forma normală 4 (FN4)

- Multidependenţa $X \rightarrow\rightarrow Y$ poate fi gândită ca o regulă de deducţie:
  - dacă tuplurile <*x*, *y*, *z*> şi <*x*, *y*′, *z*′> sunt în relaţie la un moment *r*, atunci la momentul *r* sunt în relaţie şi tuplurile <*x*, *y*, *z*′> şi <*x*, *y*′, *z*>.



Forma normală 4 (FN4)

- Orice dependenţă funcţională este o multidependenţă.
- Afirmaţia inversă nu este adevărată.
- Dacă $X \rightarrow Y$ (FD), atunci pentru oricare două tupluri $<x, y, z>$ şi $<x, y', z'>$, se obţine $y = y'$. Prin urmare în relaţie apar tuplurile $<x, y', z>$ şi $<x, y, z'>$ şi deci $X \rightarrow\rightarrow Y$ (MVD).

Forma normală 4 (FN4)

- Fie *W*, *V*, *X*, *Y* şi *Z* submulţimi de atribute ale unei scheme relaţionale *R*. Fiind dată o mulţime *T* de multidependenţe există o mulţime completă de axiome (Ax1–Ax8) care permit obţinerea tuturor multidependenţelor ce se pot deduce din mulţimea *T*. *-> Vezi curs!*
- O **multidependenţă elementară** este o multidependenţă care are părţi stângi şi drepte minimale (nu există $X' \subset X$ şi $Y' \subset Y$ a.i. $X' \rightarrow\rightarrow Y'$).
- **Formal**, relaţia *R* este în a patra formă normală dacă şi numai dacă:
  - *R* este în BCNF;
  - orice dependenţă multivaloare este o dependenţă funcţională.

Forma normală 4 (FN4)

- **Regula de descompunere în relaţii FN4**.
  - Fie $R(X, Y, Z)$ o schemă relaţională care nu este în FN4 şi fie $X \rightarrow\rightarrow Y$ o multidependenţă elementară care nu este de forma „CHEIE $\rightarrow\rightarrow$ atribut“.
  - Această relaţie este descompusă prin proiecţie în două relaţii:

    $$R = \text{JOIN}(\Pi_{X \cup Y}(R), \Pi_{X \cup Z}(R)).$$

Forma normală 4 (FN4)

- **Exemplu:**
- Fie relaţia INVESTITIE(cod_contractant#, denumire, telefon) şi presupunem că un investitor poate avea mai multe numere de telefon şi că poate investi în mai multe obiective.
- Între atributele relaţiei există multidependenţele:

cod_contractant# $\rightarrow\rightarrow$ denumire;

cod_contractant# $\rightarrow\rightarrow$ telefon.

- Relaţia INVESTITIE este în BCNF. Pentru a aduce relaţia în FN4 o

vom descompune prin proiecţie în două relaţii:

INVESTITIE_1(cod_contractant#, denumire),

INVESTITIE_2(cod_contractant#, telefon).

- INVESTITIE = JOIN(INVESTITIE_1, INVESTITIE_2).

Forma normală 5 (FN5)

- FN5 îşi propune eliminarea redundanţelor care apar în relaţii *m:n* dependente.
  - În general, aceste relaţii nu pot fi descompuse.
  - S-a arătat că o relaţie de tip 3 este diferită de trei relaţii de tip 2. Există totuşi o excepţie, şi anume, dacă relaţia este ciclică
- **Intuitiv**, o relaţie *R* este în forma normală 5 dacă şi numai dacă:
  - relaţia este în FN4;
  - nu conţine dependenţe ciclice.

Concluzii

1. FN1 → FN2 elimină redundanţele datorate dependenţei netotale a atributelor care nu participă la o cheie, faţă de cheile lui *R*. Se suprimă dependenţele funcţionale care nu sunt totale.
2. FN2 → FN3 elimină redundanţele datorate dependenţei tranzitive. Se suprimă dependenţele funcţionale tranzitive.
3. FN3 → BCNF elimină redundanţele datorate dependenţei funcţionale. Se suprimă dependenţele în care partea stângă nu este o supercheie.
4. BCNF → FN4 elimină redundanţele datorate multidependenţei. Se suprimă toate multidependenţele care nu sunt şi dependenţe funcţionale.
5. FN4 → FN5 elimină redundanţele datorate dependentei ciclice. Se suprimă toate *join*-dependenţele care nu sunt implicate de o cheie.
6. BCNF, FN4 şi FN5 corespund la regula că orice determinant este o cheie, dar de fiecare dată dependenţa cu care se defineşte determinantul este alta şi anume dependenţa funcţională, multidependenţa sau *join*-dependenţa).
7. Descompunerea unei relaţii FN2 în FN3 conservă datele şi dependenţele, pe când descompunerea unei relaţii FN3 în BCNF şi, respectiv, a unei relaţii BCNF în FN4 conservă doar datele.