BAZE DE DATE

CURS 9

Partea I

Necesitatea normalizării

- În momentul proiectării și modelării unei baze de date relaționale pot să apară anomalii. Procesul de optimizare a relațiilor conceptuale poartă numele de **normalizare**. Pentru a depista eventualele redundanțe se utilizează **dependențele funcționale** (**Vezi Curs 8**)
- **X -> Y** - X determină funcțional pe Y sau Y depinde funcțional de X
- 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 (**Vezi Curs 8**)
  - **redundanţă** în date

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 mai multe relaţii cu 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.

Exemplu regula Casey-Delobel

- Se considera relatia:

AVION(A#, nume, capacitate, localitate)

α = {nume}

β = {capacitate}

γ = {A#, localitate}

Avion

A#	nume	capacitate	localitate
1	AIRBUS	250	PARIS
2	AIRBUS	250	PARIS
3	AIRBUS	250	LONDRA
4	CAR	100	PARIS
5	B707	150	LONDRA
6	B707	150	LONDRA

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:

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

α -> β

nume -> capacitate **=>** AVION1 = (nume#, capacitate)

- Avem $\Pi_{\alpha\cup\beta}(R)$, unde α ∪ β – este multimea atributelor care intervin in dependentele functionale
- $\Pi_{\alpha\cup\gamma}(R)$, unde α ∪ γ – reprezinta reuniunea determinantului cu restul atributelor relatiei AVION

In final relatia AVION se va descompune in urmatoarele scheme relationale:

AVION1 = (nume#, capacitate)
AVION2 = (A#, nume, localitate)

Exemplu regula Casey-Delobel

Avion

A#	nume	capacitate	localitate
1	AIRBUS	250	PARIS
2	AIRBUS	250	PARIS
3	AIRBUS	250	LONDRA
4	CAR	100	PARIS
5	B707	150	LONDRA
6	B707	150	LONDRA

In final relatia AVION se va descompune in urmatoarele scheme relationale:

AVION1 = (nume#, capacitate)
AVION2 = (A#, nume, localitate)

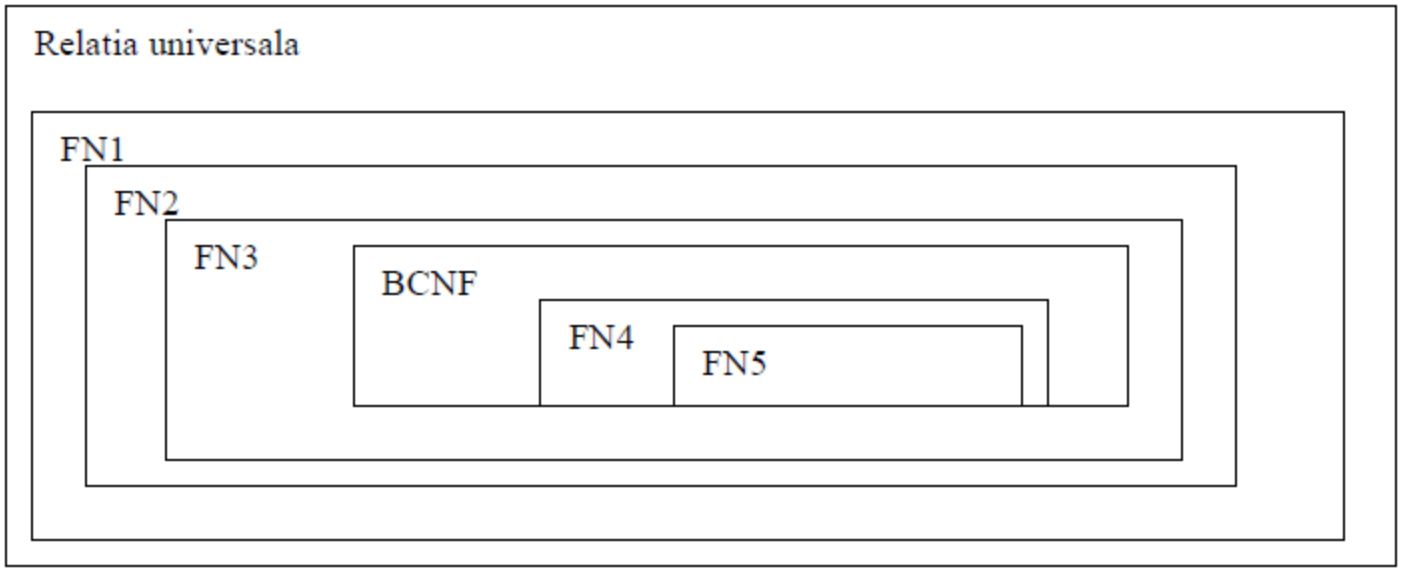
AVION1

Nume	Capacitate
AIRBUS	250
CAR	100
B707	150

AVION2

A#	Nume	Localitate
1	AIRBUS	PARIS
2	AIRBUS	PARIS
3	AIRBUS	LONDRA
4	CAR	PARIS
5	B707	LONDRA
6	B707	LONDRA

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 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*

Forma normală 2 (FN2)

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

- Un salariat are mai multe functii si poate lucra la mai multe proiecte
- O relaţie *R* este în a doua formă normală dacă şi numai dacă:
  - relaţia *R* este în FN1; -> **relatia atasat_la se afla in FN1 deoarece avem identificator unic pentru toate intrarile din tabel**
  - fiecare atribut care nu este cheie (nu participă la cheia primară) este dependent de întreaga cheie primară – in cazul nostru atributele **job_cod, functia, suma** nu sunt chei si trebuie sa depinda direct de intreaga cheie primara **cod_salariat#** si **nr_proiect#** -> aceste atribute nu depind direct de intreaga cheie primara deoarece se observa dependenta directa dintre **job_cod** si **cod_salariat**, insemnand ca job_cod depinde direct doar de o parte a cheii primare, si anume doar de cod_salariat -> **relatia atasat_la nu se afla in FN2**

Forma normală 2 (FN2)

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

- Fiecare atribut care nu este cheie (nu participă la cheia primară) este dependent de întreaga cheie primară – in cazul nostru atributele **job_cod, functia, suma** nu sunt chei si trebuie sa depinda direct de intreaga cheie primara **cod_salariat#** si **nr_proiect#** -> aceste atribute nu depind direct de intreaga cheie primara deoarece se observa dependenta directa dintre **job_cod** si **cod_salariat**, insemnand ca job_cod depinde direct doar de o parte a cheii primare, si anume doar de cod_salariat -> **relatia atasat_la nu se afla in FN2**
- Astfel avem:
  - {cod_salariat#} -> {job_cod} – cod_salariat determina functional job_cod
  - {cod_salariat#, nr_proiect#} -> {functia, suma}

Forma normală 2 (FN2)

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

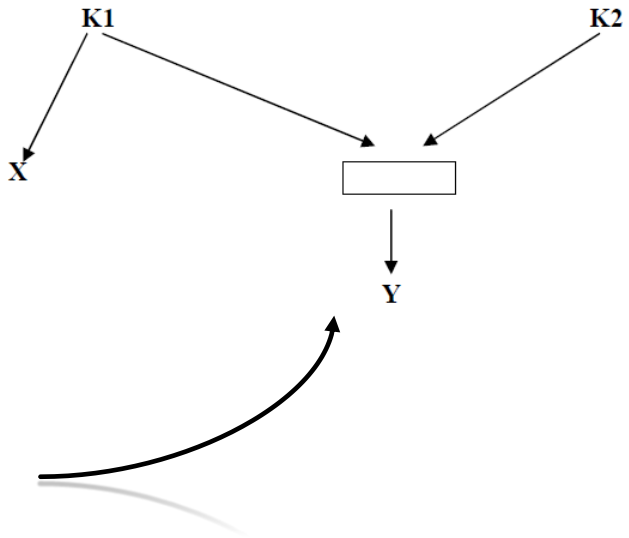
- Astfel avem:
  - {cod_salariat#} -> {job_cod} – cod_salariat determina functional job_cod
  - {cod_salariat#, nr_proiect#} -> {functia, suma}

Aplicarea regulii Casey-Delobel pentru FN2:

Fie relaţia *R*(*K*1, *K*2, *X*, *Y*), unde *K*1 şi *K*2 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) cu două proiecţii:

***R*1(*K*1, *K*2, *Y*) şi *R*2(*K*1, *X*).**



Forma normală 2 (FN2)

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

Transformarea in 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

Astfel avem:

{cod_salariat#} -> {job_cod}

– cod_salariat determina functional job_cod

{cod_salariat#, nr_proiect#} -> {functia, suma}

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**, iar **o lucrare este definita de intervalul de timp in care este executata**.

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 $Y \rightarrow A$**
  - De exemplu, dacă K1, K2, K3 $\rightarrow$ A1 şi dacă K1, K2, A1 $\rightarrow$ A2.
  - Rezulta ca A_2 este **dependent tranzitiv** de K_1, K_2, K_3. (A_2 este dependent de K_1, K_2, K_3 prin A1)

Forma normală 3 (FN3)

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

Formal, o relaţie R este în **a treia formă normală** dacă şi numai dacă:

- relaţia R este în FN2; -> **este in FN2** deoarece este in FN1 si indeplineste conditia ca fiecare atribut care nu este cheie (nu participă la cheia primară) sa fie dependent de întreaga cheie primară
- fiecare atribut care nu este cheie (nu participă la o cheie) nu este dependent tranzitiv de nicio cheie a relatiei R. **Cu alte cuvinte 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**.

In exemplul **atasat_2a** se observa ca atributul ***suma*** depinde tranzitiv de cheia primara compusa ***cod_salariat*** si ***nr_proiect*** prin intermediul atributului ***functia.***

Forma normală 3 (FN3)

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

- In exemplul **atasat_2a** se observa ca atributul ***suma*** depinde tranzitiv de cheia primara compusa ***cod_salariat*** si ***nr_proiect*** prin intermediul atributului ***functia.***
- Astfel aveam:
  - {cod_salariat#, nr_proiect#} -> {functia} – cod_salariat si nr_proiect determina functional atributul functia
  - {cod_salariat#, nr_proiect#} -> {functia} -> {suma}

Pentru a aduce relaţia **atasat_2a** în FN3 se aplică **regula Casey-Delobel**. Relaţia se descompune, prin eliminarea dependenţelor funcţionale tranzitive, în proiecţiile:

ATASAT_3A(cod_salariat#, nr_proiect#, functia)

ATASAT_3B(functia, suma)

Forma normală 3 (FN3)

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

- Pentru a aduce relaţia **atasat_2a** în FN3 se aplică **regula Casey-Delobel**. Relaţia se descompune, prin eliminarea dependenţelor funcţionale tranzitive, în proiecţiile:

ATASAT_3A(cod_salariat#, nr_proiect#, functia)

ATASAT_3B(functia, suma)

- **Transformarea in FN3**

atasat_3a

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

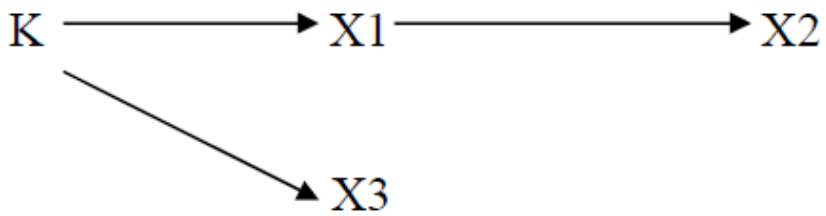
atasat_3b

Functia	Suma
Supervizor	60
Cercetator	25
Auxiliar	10

Forma normală 3 (FN3)

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:

- 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:

EXECUTA1_1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie)

EXECUTA1_2(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_{ij}**, 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_{ij}.
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

Fie relația **R(F, N, P, U, C, T)**. Se aplică **algoritmul SNF3** (aducerea unei relaţii în FN3 prin utilizarea schemei de sinteză).

1. Se determină ***F* o acoperire minimală a lui *E*** (mulţimea dependenţelor funcţionale). Aceste dependențe se aplică stiind regulile de proiectare ale modelului. Se pornește de la descrierea modelului real.

F -> N; F -> P; P, F, N -> U; P -> C; P -> T; C -> T; N -> F

Exemplu

2. Se descompune mulţimea *F* în **grupuri notate F_i**, astfel încât în cadrul fiecărui grup să existe dependenţe funcţionale **având aceeaşi parte stângă**;

F1: F -> N

F2: F -> P

F3: P, F, N -> U

F4: P -> C

F5: P -> T

F6: C -> T

F7: N -> F

Exemplu

Se elimină atributele şi dependenţele funcţionale redundante.

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);

F1: F -> N

F2: F -> P

F3: **P**, F, N -> **U**

F4: P -> C

F5: P -> T

F6: C -> T

F7: N -> F

➢**Atributele redundante** se identifică în raport cu dependențele pe care le au cu alte atribute.
În **F3** se observă dependența **P -> U**
- se poate elimina astfel atributul P deoarece avem:

- **F2: F -> P și**
- **F3: P -> U**
- rezultând **F -> U**, dependență pe care o avem deja tot în F3.

Exemplu

Se elimină atributele şi dependenţele funcţionale redundante.

F1: F -> N

F2: F -> P

F3: P, F, N -> U

F4: P -> C

F5: P -> T

F6: C -> T

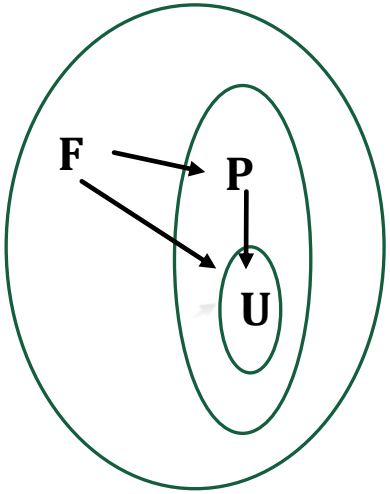
F7: N -> F

Atributul se elimină utilizand pseudo-tranzitivitatea astfel:

dacă $X \rightarrow Y$ şi $W \cup Y \rightarrow Z$, atunci $W \cup X \rightarrow Z$

F -> P si P, F -> U, atunci rezultă F -> U

SAU conform reprezentării grafice de mai jos:



Exemplu

Se elimină atributele şi dependenţele funcţionale redundante.

F1: F -> N

F2: F -> P

F3: ~~P~~, F, **N** -> **U**

F4: P -> C

F5: P -> T

F6: C -> T

F7: N -> F

La fel se procedează și în cazul atributului N

F -> N
F, N -> U
=> se elimină N și se păstrează în F3 doar dependența F -> U

Exemplu

Se elimină atributele şi dependenţele funcţionale redundante.

F1: F -> N

F2: F -> P

F3: ~~**P**~~, F, ~~**N**~~ -> U

F4: P -> C

F5: P -> T

F6: C -> T

F7: N -> F

În final F3 o să aibă dependența funcțională F -> U

Exemplu

Se elimină atributele şi dependenţele funcţionale redundante.

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);

F1: F -> N

F2: F -> P

F3: F-> U

F4: P -> C

F5: P -> T

F6: C -> T

F7: N -> F

În acest caz se identifică perechi echivalente analizând dependențe funcționale marcate cu roșu.

Din dependențele **P -> C** și **C -> T** rezultă că **P -> T**.

Deci **F5: P -> T** este o dependentă funcțională care poate fi eliminată.

Exemplu

Se elimină atributele şi dependenţele funcţionale redundante.

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);

F1: F -> N

F2: F -> P

F3: F -> U

F4: P -> C

~~**F5: P -> T**~~

F6: C -> T

F7: N -> F

Pentru eliminarea dependențelor funcționale se identifică perechi echivalente. În acest caz se analizează dependențele funcționale marcate cu roșu.

Din dependențele **P -> C** și **C -> T** rezultă că **P -> T**.

Deci **F5: P -> T** este o dependentă funcțională care poate fi eliminată.

Exemplu

Dupa eliminearea atributelor si dependentelor functionale se aplica urmatoarea regula de sinteza:

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.

Am renumerotat dependețele rămase în urma eliminărilor efectuate la pasii anteriori, după care vom aplica regula de mai sus.

G1: **F** -> N

G2: **F** -> P

G3: **F** -> U

G4: **P** -> C

G5: **C** -> T

G6: **N** -> F

Se observă dependențele care au același membru stâng. Pe acestea le vom regrupa.

Exemplu

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.

G1: **F** -> N

G2: **F** -> P

G3: **F** -> U

G4: **P** -> C

G5: **C** -> T

G6: **N** -> F

F1 = {G1, G2, G3}
F2 = {G4}
F3 = {G5}
F4 = {G6}

Exemplu

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_{ij}**, 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_{ij}.

G1: F -> N

G2: F -> P

G3: F -> U

G4: P -> C

G5: C -> T

G6: N -> F

F1 = {G1, G2, G3}
F2 = {G4}
F3 = {G5}
F4 = {G6}

F1 = {G1, G2, G3, G6}
F2 = {G4}
F3 = {G5}
~~**F4 = {G6}**~~

G1 si G6 sunt echivalente

Exemplu

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).

F1 = {G1, G2, G3, G6}
F2 = {G4}
F3 = {G5}

R1 = {F#, N, P, U}
R2 = {P#, C}
R3 = {C#, T}

G1: F -> N
G2: F -> P
G3: F -> U
G4: P -> C
G5: C -> T
G6: N -> F

Exemplul 2

Se dă următoarea schemă relațională. Stiind că ***descrierea*** se realizeaă în funcție de ***obiectiv***, de ***lucrarea executată*** și de ***șantierul*** pe care aceasta are loc, iar ***conducătorul*** depinde de ***funcția*** pe care o are, aplicați **algoritmul de sinteză FN3** (SNF3) si aduceți relația în forma normală 3.

EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

Se noteaza atributele relatiei **EXECUTA1**:

cod_obiectiv = A

cod_lucrare = B

nr_santier = C

descriere = D

functie = E

conducator = F

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.

Exemplul 2

Se dă următoarea schemă relațională. Stiind că ***descrierea*** se realizeaă în funcție de ***obiectiv***, de ***lucrarea executată*** și de ***șantierul*** pe care aceasta are loc, iar ***conducătorul*** depinde de ***funcția*** pe care o are, aplicați **algoritmul de sinteză FN3** (SNF3) si aduceți relația în forma normală 3.

EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

1. Se determină ***F* o acoperire minimală a lui *E*** (mulţimea dependenţelor funcţionale). Aceste dependențe se aplică stiind regulile de proiectare ale modelului. Se pornește de la descrierea modelului real.

2. Se descompune mulţimea *F* în **grupuri notate F_i**, astfel încât în cadrul fiecărui grup să existe dependenţe funcţionale **având aceeaşi parte stângă**;

Exemplul 2

Se dă următoarea schemă relațională. Stiind că ***descrierea*** se realizeaă în funcție de ***obiectiv***, de ***lucrarea executată*** și de ***șantierul*** pe care aceasta are loc, iar ***conducătorul*** depinde de ***funcția*** pe care o are, aplicați **algoritmul de sinteză FN3** (SNF3) si aduceți relația în forma normală 3.

EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

F1: A, B, C -> D

F2: A, B, C -> E

F3: E -> F

F1: cod_obiectiv, cod_lucrare, nr_santier -> descriere

F2: cod_obiectiv, cod_lucrare, nr_santier -> functie

F3: functie -> conducator

Exemplul 2

Se elimină atributele şi dependenţele funcţionale redundante.

În exemplul curent nu există atribute sau dependențe funcționale redundante. Așadar, se trece la pașii următori.

EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

F1: A, B, C -> D

F2: A, B, C -> E

F3: E -> F

F1: cod_obiectiv, cod_lucrare, nr_santier -> descriere

F2: cod_obiectiv, cod_lucrare, nr_santier -> functie

F3: functie -> conducator

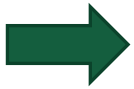
Exemplul 2

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.

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_{ij}**, 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_{ij}.

EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

F1: A, B, C -> D
F2: A, B, C -> E
F3: E -> F



$F_{1,2}$= {A, B, C, D, E}

F3 = {E, F}

$F_{1,2}$= {cod_obiectiv, cod_lucrare, nr_santier, descriere, functie}

F3 = {functie, conducator}

Exemplul 2

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).

EXECUTA1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie, conducator)

$F_{1,2}$= {cod_obiectiv, cod_lucrare, nr_santier, descriere, functie}

F3 = {functie, conducator}

EXECUTA1_1(cod_obiectiv#, cod_lucrare#, nr_santier#, descriere, functie)

EXECUTA1_2(functie#, conducator).