9. Formele normale 1 si 2 (FN1 si FN2)

Aceste forme normale nu garanteaza eliminarea anomaliilor deci ele nu sunt de dorit pentru schemele de relatie ale unei baze de date a unei aplicatii. Prezentam pe scurt definitia lor.

Definitie: O relatie R este in **forma normala 1** (FN1) daca pe toate atributele sale exista doar valori atomice ale datelor.

Semnificatia termenului 'atomic' este similara cu cea de la modelul entitate asociere: valoarea respectiva este intotdeauna folosita ca un intreg si nu se utilizeaza niciodata doar portiuni din aceasta.
De exemplu, daca intr-o relatie continand date despre persoane avem atributul ***Adresa*** acesta este atomic daca niciodata nu este nevoie sa fie folosite doar anumite portiuni ale sale (strada, numar, etc).

Fiind data o relatie R si multimea de dependente asociata F putem defini inca doua concepte:
Definitie: O dependenta functionala $X \rightarrow A$ se numeste ***dependenta partiala*** daca X este strict inclusa intr-o cheie a relatiei R.

Definitie: O dependenta functionala $X \rightarrow A$ se numeste ***dependenta tranzitiva*** daca X nu este inclusa in nici o cheie a relatiei R.

Pe baza acestor notiuni putem defini forma normala 2:

Definitie: Fie R o schema de relatie si F multimea de dependente functionale asociata. Se spune ca R este in forma normala 2 (FN2) daca si numai daca F nu contine dependente partiale (dar poate contine dependente tranzitive).

Exemplu: Relatia Produse = IdP, NumeP, Qty, IdF, NumeF, AdresaF avand dependentele functionale:
F = { IdP→NumeP, IdP→Qty, IdP→ IdF, IdF→NumeF, IdF→AdresaF }
Este in FN2 pentru ca ultimele doua dependente nu sunt partiale (IdF nu apartine cheii unice IdP).

Asa cum s-a specificat anterior FN2 nu este o forma normala 'buna', ea trebuind evitata (relatia de mai sus prezinta toate anomaliile enumerate la inceputul acestui capitol).

4.4. Descompunerea schemelor de relatii

Asa cum s-a mentionat anterior, in cazul in care o relatie din baza de date nu este intr-o forma normala buna (FNBC, FN3) pot sa apara diverse anomalii. Solutia este inlocuirea relatiei respective cu doua sau mai multe relatii care sa contina aceleasi informatii dar care, fiecare in parte, este in forma normala dorita de proiectant.

4.4.1. Definitia descompunerii unei scheme de relatie

Procesul prin care se 'sparge' o relatie in mai multe relatii se numeste ***descompunerea unei scheme de relatie***.
Formal putem defini acest concept astfel:

Definitie: Fie R o schema de relatie, $R = A_1\ A_2\ \dots\ A_m$.
Se spune ca $\rho = (R_1, R_2, \dots, R_n)$ este o ***descompunere*** a lui R daca si numai daca

$$R = R_1 \cup R_2 \cup \dots \cup R_n$$

Schemele $R_1, R_2, \dots, R_n$ contin deci atribute din R, fiecare atribut A_i al schemei initiale trebuind sa se regaseasca in cel putin una dintre ele. Nu este necesar ca schemele sa fie disjuncte (in practica ele au aproape intotdeauna atribute comune).

In exemplele de mai jos sunt prezentate cateva descompuneri valide ale unor scheme de relatii (unele insa incorecte din punct de vedere al pastrarii datelor si/sau dependentelor initiale):

Exemplul 1: Fie relatia R = ABCDE avand F = { A → B, B → A , A → C, D → E }.
Putem avea descompuneri ca:

$$\rho_1 = (ABC, DE),\ \rho 2 = (ABCD, DE),\ \rho_3 = (AB, CD, DE)$$

Exemplul 2. Fie relatia Produse = IdP, NumeP, Qty, IdF, NumeF, AdresaF avand dependentele functionale:
F = { IdP→NumeP, IdP→Qty, IdP→ IdF, IdF→NumeF, IdF→AdresaF }
Putem avea o multitudine de descompuneri printre care:

ρ_1 = **(** (IdP, NumeP, Qty, IdF); (NumeF, AdresaF) **)**

ρ_2 = **(** (IdP, NumeP, Qty, IdF); (IdF, NumeF, AdresaF) **)**

ρ_3 = **(** (IdP, NumeP); (Qty, IdF); (NumeF, AdresaF) **)**

Descompunerea actioneaza deci la nivelul ***schemei*** relatiei. Ce se intampla insa cu ***continutul*** acesteia in cazul unei descompuneri?

Fiecare relatie rezultata va mosteni o parte dintre datele relatiei descompuse si anume proiectia acesteia pe multimea de atribute a relatiei rezultata din descompunere.

Sa consideram o instanta **r** a relatiei de schema R (instanta unei relatii este o incarcare cu date corecte a acesteia). Atunci instantele pentru relatiile din descompunerea ρ sunt:

$$r_i = \pi_{Ri}(r)$$

Exemplu: Fie relatia PRODUSE de mai jos:

IdP	NumeP	Qty	IdF	NumeF	AdresaF
101	**Imprimanta laser**	**30**	**20**	**Xerox**	**Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti**
105	**Calculator PC**	**20**	**23**	**IBM**	**Bd. D.Cantemir nr.1, Bucuresti**
124	**Copiator**	**10**	**20**	**Xerox**	**Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti**

1. In cazul descompunerii ρ_1 = **(** (IdP, NumeP, Qty, IdF); (NumeF, AdresaF) **)**
obtinem :

r_1 =

IdP	NumeP	Qty	IdF
101	Imprimanta laser	30	20
105	Calculator PC	20	23
124	Copiator	10	20

r_2 =

NumeF	AdresaF
Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
IBM	Bd. D.Cantemir nr.1, Bucuresti

2. In cazul descompunerii ρ_2 = **(** (IdP, NumeP, Qty, IdF); (IdF, NumeF, AdresaF) **)**
obtinem :

r_1 =

IdP	NumeP	Qty	IdF
101	Imprimanta laser	30	20
105	Calculator PC	20	23
124	Copiator	10	20

r2 =

IdF	NumeF	AdresaF
20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
23	IBM	Bd. D.Cantemir nr.1, Bucuresti

3. In cazul descompunerii ρ_1 = **(** (IdP, NumeP); (Qty, IdF); (NumeF, AdresaF) **)** obtinem :

r_1 =

IdP	NumeP
101	Imprimanta laser
105	Calculator PC
124	Copiator

r_2 =

Qty	IdF
30	20
20	23
10	20

r_3 =

NumeF	AdresaF
Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
IBM	Bd. D.Cantemir nr.1, Bucuresti

Observam din aceste exemple ca in cazul primei si ultimei descompuneri nu putem reconstrui prin join sau alti operatori relationali relatia initiala. In cazul in care descompunerea nu s-a facut corect putem pierde:

- Datele relatiei initiale
- Dependentele functionale ale relatiei initiale.

In paragrafele urmatoare sunt prezentati algoritmi prin care putem detecta daca prin descompunere se pierd date sau dependente.

4.4.2. Descompuneri cu join fara pierderi

Conditia pentru a nu se pierde date prin descompunere este ca relatia initiala sa poata fi reconstruita exact prin joinul natural al relatiilor rezultate, fara tupluri in minis sau in plus. Formal, definitia este urmatoarea:

Definitie: Fie R o schema de relatie, F multimea de dependente functionale asociata si o descompunere $\rho = (R_1, R_2, \ldots, R_n)$ a lui R. Se spune ca ρ este o descompunere ***cu join fara pierderi in raport cu F*** (prescurtat j.f.p.) daca si numai daca pentru orice instanta **r** a lui R care satisface dependentele F avem ca:

$$r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n = r$$

unde

$$r_i = \pi_{Ri}(r)$$

In exemplul de la paragraful 4.4.1 doar descompunerea ρ_2 = **(** (IdP, NumeP, Qty, IdF); (IdF, NumeF, AdresaF) **)** are aceasta proprietate, in cazul celorlalte, din cauza inexistentei coloanelor comune, joinul natural nu se poate efectua.

Faptul ca o descompunere are sau nu aceasta proprietate se poate testa pornind doar de la lista atributelor relatiei initiale, lista atributelor relatiilor din descompunere si a multimii de dependente functionale asociata folosind urmatorul algoritm

Algoritm de testare a proprietatii de j.f.p. pentru o descompunere

Intrare: Schema de relatie $R = A_1\ A_2\ \dots\ A_m$, multimea de dependente functionale F si o descompunere $\rho = (R_1, R_2, \dots, R_n)$
Iesire: Verdictul daca ρ are sau nu proprietatea j.f.p.
Metoda:
Se construieste o tabela avand **n** linii si **m** coloane. Liniile sunt etichetate cu elementele descompunerii ρ iar coloanele cu atributele relatiei R. Elementul (i,j) al tabelei va fi egal cu a_j daca $A_j \in R_i$ sau b_{ij} altfel.
Se parcurg dependentele $X \rightarrow Y$ din F. Daca doua (sau mai multe) linii din tabela au aceiasi simboli pe coloanele X aceste linii se egaleaza si pe coloanele din Y astfel:

- Daca pe o coloana din Y apare un a_j atunci toate elementele de pe acea coloana din liniile respective devin a_j
- Daca pe o coloana din Y nu apare nici un a_j atunci se alege unul dintre elementele de tip b_{ij} si toate elementele de pe acea coloana din liniile respective devin egale cu acel b_{ij}

Procesul se opreste:

- Fie cand s-a obtinut o linie in tabela care contine doar a-uri, caz in care descompunerea ρ **are** proprietatea j.f.p.
- Fie cand la o parcurgere a dependentelor nu mai apar schimbari in tabela si nu s-a obtinut o linie doar cu a-uri. In acest caz descompunerea ρ **nu are** proprietatea j.f.p.

In literatura de specialitate se poate gasi demonstratia faptului ca acest algoritm determina corect daca o descompunere are proprietatea j.f.p. sau nu.

Exemplul 1: Fie R = ABCDE, F = { $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $D \rightarrow E$ } si o descompunere a lui R ρ = (ABCD, DE)

Construim tabelul din algoritm:

	A	**B**	**C**	**D**	**E**
ABCD	**a1**	**a2**	**a3**	**a4**	~~**b15**~~ **a5**
DE	**b21**	**b22**	**b23**	**a4**	**a5**

La prima parcurgere, pentru dependentele $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$ nu gasim doua linii cu aceleasi valori pe coloana A dar pentru dependenta $D \rightarrow E$ cele doua linii sunt egale pe coloana D (simbolul a4). Le egalam si pe coloana E: cum pe aceasta coloana exista a5 rezulta ca b15 devine egal cu a5. S-a obtinut o linie numai cu a-uri, deci descompunerea are proprietatea de join fara pierderi.

Exemplul 2: Fie relatia R = ABCDE si F = { A → B, AC → D, D → E } si descompunerea ρ = (AB, BC, CDE). Tabelul este urmatorul:

	A	B	C	D	E
AB	**a1**	**a2**	**b13**	**b14**	**b15**
BC	**b21**	**a2**	**a3**	**b24**	**b25**
CDE	**b31**	**b32**	**a3**	**a4**	**a5**

La prima trecere nu apar modificari in tabel. Procesul se opreste si ρ nu are proprietatea j.f.p.

Exemplul 3. Fie R = ABCDE,
F = { C → E (1), A → C (2), B → D (3), D → E (4), E → B (5) } (cu dependente numerotate intre paranteze) si ρ = (BCE, AB, ACD)

	A	B	C	D	E
BCE	**b11**	**a2**	**a3**	**b14**	**a5**
AB	**a1**	**a2**	~~**b23**~~ **a3** **(2)**	~~**b24**~~ **b14** **(3)**	~~**b25**~~ **a5** **(4)**
ACD	**a1**	~~**b32**~~ **a2** **(5)**	**a3**	**a4**	~~**b35**~~ **a5** **(1)**

Prima trecere:

- Din C → E b35 devine a5
- Din A → C b23 devine a3
- Din B → D b24 devine b14
- Din D → E b25 devine a5
- Din E → B b32 devine a2

Am obtinut inca o linie doar cu a-uri. Descompunerea are proprietatea de j.f.p.

Observatie: In exemplele de mai sus a fost suficienta o singura trecere prin dependente. Exista insa situatii cand sunt necesare mai multe treceri pana procesul se opreste.

In cazul in care descompunerea are doar doua elemente se poate testa daca are proprietatea de join fara pierderi si astfel:

Fie R o schema de relatie, F multimea de dependente functionale asociata si ρ = (R1, R2) o descompunere a sa.
Atunci ρ are proprietatea de join fara pierderi daca una din dependentele urmatoare se poate deduce din F:

$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ sau

$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$

Pentru a testa daca dependenta se poate deduce din F este suficient sa calculam inchiderea lui $(R_1 \cap R_2)$. Daca ea contine fie pe $(R_1 - R_2)$ fie pe $(R_2 - R_1)$ atunci descompunerea este cu join fara pierderi.

Exemplu: In prima exemplificare a aplicarii algoritmului de testare aveam o descompunere cu doua elemente: R= ABCDE, F = { A → B, A → C, A → D, D → E } si ρ = (ABCD, DE).

Avem R_1 = ABCD, R_2 = DE,

$(R_1 - R_2)$ = ABCD – DE = ABC

$(R_2 - R_1)$ = DE – ABCD = E

$(R_1 \cap R_2)$ = D

Cele doua dependente sunt:

$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ devine D → ABC

$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ devine D → E

Ultima este chiar o dependenta din F deci se poate deduce din F deci ρ are proprietatea de join fara pierderi.

4.4.2. Descompuneri care pastreaza dependentele

O a doua problema in cazul descompunerii unei scheme de relatie R avand dependentele F in mai multe relatii R_1, R_2, …, R_n este aceea a pastrarii corelatiilor intre date, corelatii date de dependentele functionale din F.

Fiecare relatie R_i va mosteni o multime de dependente data de proiectia multimii de dependente functionale F pe R_i

$$F_i = \pi_{Ri}(F)$$

Exemplu: Fie relatia Produse = IdP, NumeP, Qty, IdF, NumeF, AdresaF avand dependentele functionale:

F = { IdP→NumeP, IdP→Qty, IdP→ IdF, IdF→NumeF, IdF→AdresaF }

In cazul descompunerii ρ_2 = (R1, R2) unde:

R1 = (IdP, NumeP, Qty, IdF)

R2 = (IdF, NumeF, AdresaF)

cele doua relatii mostenesc urmatoarele dependente:

$F_{R1} = \pi_{R1}(F)$ = { IdP→NumeP, IdP→Qty, IdP→ IdF}

$F_{R2} = \pi_{R2}(F)$ = { IdF→NumeF, IdF→AdresaF }

Dupa cum se observa toate dependentele relatiei initiale sunt pastrate fie in F_{R1} fie in F_{R2}. Exista insa si cazuri in care unele dependente din F nu mai pot fi regasite in multimile de dependente asociate schemelor din descompunere si nu se pot deduce din acestea. In primul caz se spune ca descompunerea pastreaza dependentele iar in al doilea ca descompunerea nu pastreaza dependentele.

O definitie formala a acestui concept este urmatoarea:

Definitie: Fie R o schema de relatie, F multimea de dependente functionale asociata, o descompunere ρ = (R_1, R_2, …, R_n) a lui R si $F_i = \pi_{Ri}(F)$ multimile de dependente functionale ale elementelor descompunerii. Se spune ca ρ ***pastreaza dependentele*** din F daca si numai daca orice dependenta din F poate fi dedusa din $\cup_{i=1..n}(F_i)$.

Rezulta ca o descompunere pastreaza dependentele daca si numai daca:

$$F \subseteq (\cup_{i=1..n} (F_i))^+$$

Din pacate atat proiectia unei multimi de dependente cat si incluziunea de mai sus implica un calcul de inchidere a unei multimi de dependente (F si respectiv reuniunea multimilor F_i). Exista si in acest caz un algoritm pentru a testa daca o dependenta este sau nu pastrata dupa descompunere fara a fi necesar efectiv calculul multimilor F_i

Algoritm de testare a pastrarii dependentelor

Intrare: o schema de relatie R, multimea de dependente functionale asociata F si o descompunere $\rho = (R_1, R_2, \ldots, R_n)$
Iesire: verdictul daca ρ pastreaza sau nu dependentele
Metoda: Pentru fiecare dependenta $X \rightarrow Y$ din F se procedeaza astfel:

- Se porneste cu o multime de atribute $Z = X$
- Se parcurg repetat elementele descompunerii ρ. Pentru fiecare R_i se calculeaza o noua valoare a lui Z astfel:
$$Z = Z \cup ((Z \cap R_i)^+ \cap R_i)$$
- Procesul se opreste in momentul cand Z ramane neschimbat la o parcurgere a elementelor R_i. Daca $Y \subseteq Z$ atunci dependenta $X \rightarrow Y$ este pastrata, altfel nu e pastrata

Daca toate dependentele din F sunt pastrate inseamna ca ρ pastreaza dependentele din F.

Exemplul 1: Fie R = ABCDE, ρ = (BCE, AB, ACD) si multimea de dependente functionale $F = \{ C \rightarrow E , A \rightarrow C , B \rightarrow D , D \rightarrow E , E \rightarrow B \}$

Se observa ca dependentele $C \rightarrow E$, $A \rightarrow C$ si $E \rightarrow B$ sunt pastrate: ele apartin proiectiei lui F pe BCE (prima si ultima) si ACD (a doua). Raman de testat dependentele $B \rightarrow D$ si $D \rightarrow E$. Sa aplicam algoritmul pentru $B \rightarrow D$:

Initial Z = B
Trecerea 1 prin elementele lui ρ:

Pentru BCE: $Z = B \cup ((B \cap BCE)^+ \cap BCE) = B \cup (BDE \cap BCE) = BE$

Pentru AB: $Z = BE \cup ((BE \cap AB)^+ \cap AB) = BE \cup (BDE \cap AB) = BE$

Pentru ACD: $Z = BE \cup ((BE \cap ACD)^+ \cap AB) = BE \cup \varnothing = BE$

La urmatoarea trecere Z ramane neschimbat si procesul se opreste. Cum $\{D\} \not\subset BE$ rezulta ca dependenta $B \rightarrow D$ nu este pastrata deci ρ nu pastreaza dependentele.

Exemplul 2: Fie schema de relatie R = ABCD, $F = \{ A \rightarrow B, A \rightarrow C, C \rightarrow D, D \rightarrow A\}$ si o descompunere ρ = (ABC, CD). Trebuie sa testam daca $D \rightarrow A$ este pastrata (celelalte dependente se regasesc direct in proiectiile lui F pe elementele descompunerii).

Initial Z = D

Prima trecere prin elementele lui ρ:

Pentru ABC: $Z = D \cup ((D \cap ABC)^+ \cap ABC) = D \cup \varnothing = D$

Pentru CD: $Z = D \cup ((D \cap CD)^+ \cap CD) = D \cup (ABCD \cap CD) = CD$

A doua trecere prin elementele lui ρ:

Pentru ABC: $Z = CD \cup ((CD \cap ABC)^+ \cap ABC) = CD \cup (ABCD \cap ABC) =$ ABCD. Stop. Am obtinut ca $A \subseteq Z$, deci dependenta $D \rightarrow A$ este pastrata, deci ρ pastreaza dependentele.

4.4.3. Algoritmi de descompunere

Algoritmii de testare al pastrarii dependentelor si a joinului fara pierderi pot fi aplicati atunci cand descompunerea unei scheme de relatie se face 'de mana', pe baza experientei pe care o are proiectantul bazei de date. Exista insa si niste algoritmi simpli care, pornind de la o schema de relatie si multimea de dependente functionale asociata ne duc direct la o descompunere care este in FN3 sau FNBC si in plus au proprietatea de join fara pierderi (deci nu se pierd date prin descompunere) si/sau de pastrare a dependentelor.

Algoritm de descompunere in FN3 cu pastrarea dependentelor

Fie R o schema de relatie si F multimea de dependente functionale asociata, cu

$F = \{ X_1 \rightarrow Y_1, X_2 \rightarrow Y_2, \ldots X_n \rightarrow Y_n \}$

Atunci descompunerea $\rho = (X_1Y_1, X_2Y_2, \ldots X_nY_n)$ este o descompunere in FN3 cu pastrarea dependentelor.

Se observa din definitia de mai sus a descompunerii ρ ca:

- Toate dependentele sunt pastrate: dependenta $X_i \rightarrow Y_i$ este in proiectia lui F pe X_iY_i
- Pentru a minimiza numarul de elemente din descompunere se aplica regula reuniunii: daca avem mai multe dependente care au aceeasi parte stanga le reunim intr-una singura.
- Daca in descompunere exista doua elemente X_iY_i si X_jY_j astfel incat $X_iY_i \subseteq X_jY_j$ atunci X_iY_i se elimina.

In literatura de specialitate exista demonstratia faptului ca fiecare schema din descompunere este in FN3.

Exemplul 1: R = ABCDE, $F = \{ A \rightarrow B, A \rightarrow C, A \rightarrow D, D \rightarrow E \}$. Rescriem prin reuniune multimea de dependente functionale: $F = \{ A \rightarrow BCD, D \rightarrow E \}$. Rezulta din algoritm descompunerea ρ = (ABCD, DE)

Exemplul 2: Fie relatia Produse = IdP, NumeP, Qty, IdF, NumeF, AdresaF avand dependentele functionale:

$F = \{ IdP \rightarrow NumeP, IdP \rightarrow Qty, IdP \rightarrow IdF, IdF \rightarrow NumeF, IdF \rightarrow AdresaF \}$

Rescriem multimea de dependente. Raman doar doua dependente:

$F = \{ IdP \rightarrow NumeP, Qty, IdF; IdF \rightarrow NumeF, AdresaF \}$

Descompunerea in FN3 cu pastrarea dependentelor va fi:

ρ = ((IdP, NumeP, Qty, IdF), (IdF, NumeF, AdresaF))