

Capitolul 6: Normalizarea relatiilor

- Dependentele de date si formele normale ale relatiilor
- Redundanta datelor si anomaliiile de actualizare a relatiilor
- Dependentele functionale (DF)
 - Tipuri de DF
 - Multimi de DF
 - DF si cheile relatiilor
- Descompunerea relatiilor
 - Descompunere fara pierdere de informatie la jonctiune
 - Descompunere cu conservarea DF
- Formele normale ale relatiilor determinate de DF:
 - FN1
 - FN2
 - FN3
 - FNBC
- Dependente multivalorice – forma normala FN4
- Dependente de jonctiune – forma normala FN5

Dependentele de date

- **Dependențele de date** (*data dependencies*) reprezintă constrângeri care se impun valorilor atributelor unei relații și care determină proprietățile relației în raport cu operațiile de inserare, ștergere și actualizare a tuplurilor.
- **O formă normală a unei relații** (*normal form*) presupune anumite condiții pe care le îndeplinesc valorile atributelor și dependențele de date definite pe acea relație
- Dependentele de date:
 - Dependente functionale: E.F. Codd a propus trei forme normale: FN1, FN2, FN3; apoi a fost introdusă forma normală Boyce-Codd (FNBC)
 - Dependențelor multivalorice: forma normala 4 (FN4)
 - Dependențelor de joncțiune: forma normala 5 (FN5)
- Formele normale ale relațiilor formează o colecție ordonată (FN1, FN2, FN3, FNBC, FN4, FN5), și ele impun condiții din ce în ce mai restrictive asupra dependențelor de date
- Ordonarea formelor normale de la FN1 la FN5 înseamnă că orice relație aflată în FN2 este în FN1, orice relație în FN3 este în FN1 și FN2 etc.
- **Normalizarea relațiilor** (*normalization*) constă în descompunerea lor, astfel încât relațiile să fie în forme normale cât mai avansate

Redundanta datelor si anomalii de actualizare

- In relatia AP, cu cheia primara $PK=\{IdAngajat, IdProiect\}$ sunt valori redundante ale atributelor: Nume, prenume, Adresa
 - **Anomalii de inserare:** nu se pot introduce date despre un angajat (numele, prenumele, adresa) dacă nu există cel puțin un proiect la care acesta să lucreze
 - **Anomalii de ștergere:** dacă se șterg toate tuplurile referitoare la un anumit proiect, se pot pierde toate datele referitoare la acei angajați care lucrează doar la proiectul respectiv
 - **Anomalii de actualizare:** dacă se modifică într-un tuplu valoarea unuia din attributele care au valori redundante, starea relației poate deveni inconsistentă

<u>IdAngajat</u>	Nume	Prenume	Adresa	<u>IdProiect</u>	Ore
1	Ionescu	Ion	Bucuresti	P1	100
2	Popescu	Petre	Ploiesti	P1	80
3	Marinescu	Marin	Craiova	P3	200
1	Ionescu	Ion	Bucuresti	P2	100
2	Popescu	Petre	Ploiesti	P3	120

Eliminarea anomaliilor datorate redundantei

- Eliminarea anomaliilor provocate de redundanta datelor se poate face:
 - Fie prin prevederea unor proceduri stocate (sau triggeri) care sa verifice corectitudinea fiecărei operatii de actualizare a relatiilor
 - Fie prin descompunerea relatiilor care prezinta redundante in relatii mai simple; exemplu: descompunerea relatiei AP in 2 relatii, A si P

A

<u>IdAngajat</u>	Nume	Prenume	Adresa
1	Ionescu	Ion	Bucuresti
2	Popescu	Petre	Ploiesti
3	Marinescu	Marin	Craiova

P

<u>IdAngajat</u>	<u>IdProiect</u>	Ore
1	P1	100
2	P1	80
3	P1	200
1	P2	100
2	P2	120

- Relatiile A si P au un grad de normalizare mai ridicat si nu mai au anomalii
- Dar unele interogari sunt mai ineficiente in A si P decat in AP
 - Exemplu: “Care este numărul de ore lucrate de Ionescu la proiectul P1 ?”:

$$Q1 = \Pi_{Ore} \sigma_{Nume = 'Ionescu' \text{ AND } IdProiect = 'P1'} (AP)$$

$$Q2 = \Pi_{Ore} \sigma_{Nume = 'Ionescu' \text{ AND } IdProiect = 'P1'} (A \bowtie P)$$

Dependente functionale (1)

- **Dependentele funcționale** (*functional dependencies*) sunt constrangeri în relații, prin care valoarea unui anumit set de atribute determină în mod unic valoarea altor atribute

- Fie R o schema de relație și două submultimi ale atributelor sale:

$X \subseteq R$ and $Y \subseteq R$. Dependenta funcțională (DF) $X \rightarrow Y$ există în R dacă și numai dacă pentru orice stare a relației $r(R)$, egalitatea valorilor atributelor X din două tupluri t_1 și t_2 din r ($t_1 \in r$ și $t_2 \in r$) implică egalitatea valorilor atributelor Y din acele tupluri, adică:

$$t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

- Atributul din stânga se numește *determinant*, cel din dreapta, *determinat*
- Dependentele funcționale sunt generalizarea notiunilor de chei ale relațiilor: orice cheie determină o DF în acea relație
 - Dacă $Y = R$, atunci X este o cheie a relației
 - Reciproc, dacă X este o cheie, $Y = R$ ($X \rightarrow R$)
 - În acest caz $t_1[X] = t_2[X] \Rightarrow t_1 = t_2$ dar, cum într-o relație nu pot exista două tupluri identice, rezultă că t_1 și t_2 sunt unul și același tuplu
- Cheile relațiilor:
 - se pot preciza explicit (și atunci ele implică DF corespunzătoare)
 - pot fi deduse din mulțimea DF stabilite de proiectant, folosind diferiți algoritmi

Dependente funcționale (2)

- O mulțime F de dependente funcționale în R definește constrangerile pe care trebuie să le respecte orice relație $r(R)$ pentru a fi legală (corectă)
 - Se spune că F se menține în R dacă toate relațiile legale pe R ($r(R)$) satisfac mulțimea de dependente funcționale F
 - Reciproc, o relație r este legală în raport cu o mulțime de dependente funcționale F , dacă r satisface F
- Proiectantul bazei de date specifică acele DF pe care le dorește să fie respectate și care sunt evidente din punct de vedere semantic;
- De exemplu, în relația AP se pot defini DF:
 - a) $IdAngajat \rightarrow Nume$
 - (b) $IdAngajat \rightarrow Prenume$
 - (c) $IdAngajat \rightarrow Adresa$
 - (d) $\{IdAngajat, IdProiect\} \rightarrow Ore$
- Primele 3 impun ca un identificator $IdAngajat$ să identifice un singur angajat, iar ultima impune ca un angajat efectuează un anumit număr de ore la fiecare din proiectele la care lucrează

Tipuri de dependențe funcționale

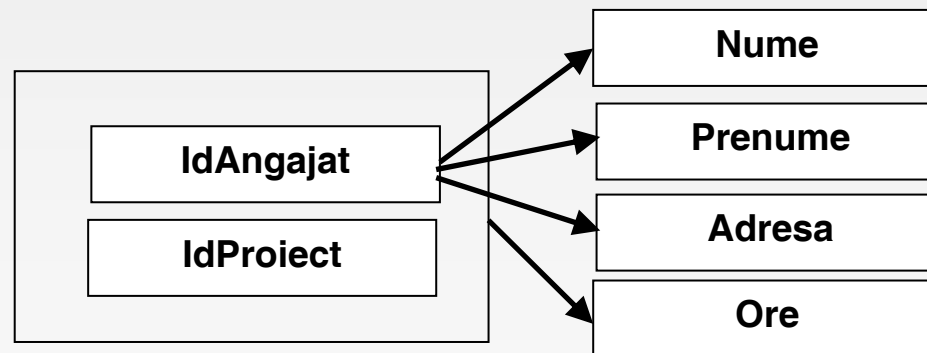
- Atributele care aparțin unei chei se numesc atribute *prime*, iar celelalte se numesc atribute *neprime*
- Dependențe funcționale *triviale si ne-triviale*
 - O dependența funcțională este trivială dacă este satisfăcută de orice stare a relației; în general $X \rightarrow Y$ este trivială dacă Y este o submulțime (proprie sau nu) a lui X , adică $Y \subseteq X$; ex. $\{IdAngajat, IdProiect\} \rightarrow IdAngajat$ sau $Nume \rightarrow Nume$
 - Celelalte dependente (în care Y nu este o submulțime a lui X) sunt ne-triviale; ex: $IdAngajat \rightarrow Nume$
- Dependențele funcționale *parțiale si totale*
 - O dependență funcțională $X \rightarrow Y$ este parțială dacă există o submulțime proprie Z a lui X ($Z \subset X$ și $Z \neq X$) care determină funcțional pe Y ($Z \rightarrow Y$); ex: $\{IdAngajat, IdProiect\} \rightarrow Nume$ este parțială deoarece $IdAngajat \rightarrow Nume$
 - O dependență funcțională $X \rightarrow Y$ este totală, dacă nu există nici o submulțime proprie Z a lui X care să determine funcțional pe Y
 - Cazuri particulare:
 - dacă atributul X este simplu, dependența funcțională $X \rightarrow Y$ este totală; ex: $IdAngajat \rightarrow Nume$
 - dacă X este o cheie candidată a lui R , atunci dependența $X \rightarrow Y$ este totală; ex: $\{IdAngajat, IdProiect\} \rightarrow Ore$

Multimi de dependente functionale (1)

- Dintr-o mulțime dată de DF se pot deduce si alte DF, folosind regulile de deducere (inferență - *inference rules*) ale lui Armstrong:
- **1. Reflexivitatea** (*reflexivity*): dacă $Y \subseteq X$, atunci $X \rightarrow Y$; prin această regulă se deduc DF *triviale*; de ex. în relația AP în care au fost definite DF:
 - (a) $\text{IdAngajat} \rightarrow \text{Nume}$
 - (b) $\text{IdAngajat} \rightarrow \text{Prenume}$
 - (c) $\text{IdAngajat} \rightarrow \text{Adresa}$
 - (d) $\{\text{IdAngajat}, \text{IdProiect}\} \rightarrow \text{Ore}$se pot deduce prin reflexivitate si DF:
 - (e) $\{\text{IdAngajat}, \text{IdProiect}\} \rightarrow \text{IdAngajat}$
 - (f) $\{\text{IdAngajat}, \text{IdProiect}\} \rightarrow \text{IdProiect}$
- **2. Augmentarea** (*augmentation*): dacă $X \rightarrow Y$, atunci $(X \cup Z) \rightarrow (Y \cup Z)$; următoarele DF (g și h) sunt deduse prin augmentare, pornind de la dependențele funcționale (a) și respectiv (b), augmentate cu $Z = \{\text{Nume}\}$, respectiv $Z = \{\text{Prenume}\}$ (se stie ca $\text{Nume} \cup \text{Nume} = \text{Nume}$ etc.)
 - (g) $\{\text{IdAngajat}, \text{Nume}\} \rightarrow \text{Nume}$
 - (h) $\{\text{IdAngajat}, \text{Prenume}\} \rightarrow \text{Prenume}$

Multimi de dependente functionale (2)

- **3. Tranzitivitatea (transitivity):** dacă $X \rightarrow Y$ și $Y \rightarrow Z$, atunci $X \rightarrow Z$. De exemplu, din DF (e) și (a) rezultă:
 $\{\text{IdAngajat}, \text{IdProiect}\} \rightarrow \text{Nume}$
- Aceste trei reguli sunt suficiente pentru calculul tuturor dependențelor funcționale pe care le implică o mulțime dată de DF
- Din aceste reguli de bază se pot deduce și alte reguli; de exemplu:
Proiecția (projection): dacă $X \rightarrow (Y \cup Z)$, atunci $X \rightarrow Y$ și $X \rightarrow Z$
Reuniunea (union): dacă $X \rightarrow Y$ și $X \rightarrow Z$, atunci $X \rightarrow (Y \cup Z)$
Pseudo-tranzitivitatea (pseudo-transitivity): dacă $X \rightarrow Y$ și $(W \cup Y) \rightarrow Z$, atunci $(W \cup X) \rightarrow Z$.
- Reprezentarea dependențelor functionale:



Inchiderea unei multimi de DF

- *Fiind dată o mulțime F de DF, mulțimea tuturor DF care sunt implicate de F se numește **închiderea mulțimii F** (closure set of F) și se notează F^+*
 - F^+ se poate deduce prin aplicarea repetată a regulilor de inferență asupra DF din F
 - Pentru a testa dacă o DF $X \rightarrow Y$ poate fi dedusă dintr-o mulțime F de DF, se afla închiderea F^+ a mulțimii F și se testează dacă $(X \rightarrow Y) \in F^+$
 - Două mulțimi de DF, E și F sunt echivalente dacă închiderile lor sunt egale ($E^+ = F^+$); adică: $\forall DF \in E \Rightarrow DF \in F^+$ și $\forall DF \in F \Rightarrow DF \in E^+$
- ***O mulțime G de DF este minimă dacă satisface următoarele condiții:***
 - membrul drept al oricărei DF din G este un atribut simplu;
 - orice dependență funcțională din G este totală;
 - mulțimea G este ireductibilă, adică, dacă se exclude o DF din G , mulțimea rezultată H nu este echivalentă cu G (adică $H^+ \neq G^+$).
- ***Acoperirea minimă a unei mulțimi F de DF*** (minimal cover of a set F of DFs) *este o mulțime minimă de dependențe funcționale G care este echivalentă cu F , adică $G^+ = F^+$*
 - Pot exista mai multe acoperiri minime ale unei mulțimi de DF

Dependențele funcționale și cheile relațiilor

- In orice relație pot exista două categorii de DF:
 - DF determinate de cheile (candidate) ale relației; astfel de DF nu produc redundanța datelor și nici anomalii de actualizare a relației; de ex. {IdAngajat, IdProiect} → Ore
 - DF în care atributul determinant nu este o cheie a relației; astfel de DF produc redundanța datelor și anomalii de actualizare a relației; de ex. IdAngajat → Nume
- DF determinate de cheile candidate sunt *constrângeri implicite*, conținute în definiția relației și sunt verificate și impuse automat de SGBD
 - Proiectantul bazei de date nu trebuie să prevadă nimic suplimentar pentru ca aceste constrângeri să fie satisfăcute de orice stare a relației
- DF în care atributul determinant nu este o cheie sunt *constrângeri explicite*, care nu sunt verificate și nici impuse de SGBD
- Pentru DF în care atributul determinant nu este o cheie, se aplica una din următoarele soluții:
 - (a) Se accepta astfel de DF, dar se asigura verificarea și impunerea lor procedurala, prin triggere, proceduri stocate, sau funcții în programele de aplicații
 - (b) Se descompune relația în relații mai simple, în care nu mai exista astfel de DF, dar trebuie făcută o descompunere “corectă” (cel puțin fără pierdere de informație la joncțiune sau, dacă se poate, reversibilă)

Inchiderea unui atribut fata de o multime de DF (1)

- **Închiderea unui atribut față de o mulțime F de DF** (the closure of an attribute under F). Fie un atribut X al unei relații pe care este definită mulțimea F de DF; mulțimea X^+ a tuturor atributelor determinate de X ($X \rightarrow X^+$) se numește închiderea lui X în raport cu F
- Algoritmul prin care se poate deduce închiderea unui atribut:
 1. se setează $X^+ = X$ și $P_F = F$
 2. repetă
 - se extrage o DF $Y \rightarrow Z$ din P_F
(adică se alege o DF $Y \rightarrow Z$ din P_F și se setează $P_F = P_F - (Y \rightarrow Z)$)
 - dacă $Y \subseteq X^+$ atunci $X^+ = X^+ \cup Z$
 - până când $P_F = \emptyset$
- Se aplică acest algoritm pentru dependențele funcționale din relația AP:
 $F_{AP} = \{IdAngajat \rightarrow Nume, IdAngajat \rightarrow Prenume, IdAngajat \rightarrow Adresa, \{IdAngajat, IdProiect\} \rightarrow Ore\}$
- Se obțin închiderile atributelor astfel:
 $\{IdAngajat\}^+ = \{IdAngajat, Nume, Prenume, Adresa\}$
 $\{IdAngajat, IdProiect\}^+ = \{IdAngajat, IdProiect, Nume, Prenume, Adresa, Ore\}$
 $\{Nume\}^+ = \{Nume\}, \{Prenume\}^+ = \{Prenume\}, \{IdProiect\}^+ = \{IdProiect\};$
 $\{Adresa\}^+ = \{Adresa\}, \{Ore\}^+ = \{Ore\}$

Inchiderea unui atribut fata de o multime de DF (2)

- Din algoritmul de mai sus se poate deduce că închiderea unui atribut care nu determină funcțional nici un alt atribut (nu apare în partea stângă a nici unei DF) este chiar atributul respectiv:
 $\{\text{Nume}\}^+ = \{\text{Nume}\}, \{\text{Prenume}\}^+ = \{\text{Prenume}\}, \{\text{Adresa}\}^+ = \{\text{Adresa}\}, \{\text{Ore}\}^+ = \{\text{Ore}\}$
- Acest algoritm poate fi folosit pentru a verifica dacă o DF dată $X \rightarrow Y$ este consecința logică a mulțimii F (dacă $(X \rightarrow Y) \in F^+$)
 - Pentru aceasta, se calculează X^+ față de F ; dacă $Y \subseteq X^+$, atunci $(X \rightarrow Y) \in F^+$, deci $X \rightarrow Y$ este consecința logică a lui F
- Acest algoritm poate fi folosit și pentru a *verifica dacă un atribut K (simplu sau compus) este supercheie a relației R cu mulțimea F de DF*
 - Pentru aceasta se calculează K^+ în raport cu F ; dacă $K^+ = R$, atunci K este supercheie în R
- Pentru aflarea cheilor candidate ale unei relații din mulțimea DF:
 - se consideră o supercheie a relației
 - se testează condiția de ireductibilitate pentru fiecare din attributele componente ale supercheii și se elimină orice atribut care poate fi eliminat fără să se piardă proprietatea de unicitate a cheii
- Acest algoritm de aflare a cheilor candidate ale unei relații din mulțimea DF este prezentat în continuare

Aflarea cheilor unei relații din mulțimea DF

- Fiind dată o relație cu schema R și mulțimea F a DF pe această relație, cheile candidate K ale relației se pot afla cu urmatorul algoritm:
 1. Se setează supercheia $K = R$
 2. Pentru fiecare atribut X din K se testează dacă $(K - X)$ este supercheie în R
 3. Dacă $(K - X)$ este supercheie, atunci $K = (K - X)$, altfel K nu se modifica
- Prin parcurgerea repetată a pașilor 2 și 3, se găsește una din cheile candidate ale relației; dacă există mai multe chei candidate, atunci ordinea găsirii cheilor candidate depinde de atributul selectat în pasul 2 al algoritmului
- De exemplu, se aplică algoritmul de mai sus pentru găsirea unei chei candidate a relației AP cu mulțimea F_{AP} a DF (definita la începutul capitolului)
- Se pornește cu $K = R = \{IdAngajat, Nume, Prenume, Adresa, IdProiect, Ore\}$, se selectează $X = Nume$;
 - $\{K - X\} = \{IdAngajat, Prenume, Adresa, IdProiect, Ore\}$
 - $\{K - X\}^+ = \{IdAngajat, Nume, Prenume, Adresa, IdProiect, Ore\} = R$,
 - deci $K - X$ este supercheie în R , deci se repetă pașii 2 și 3 pentru alte atribute
- Se repetă pașii 2 și 3 pentru $X = Prenume$, apoi $X = Adresa$ și $X = Ore$; aceste atribute pot fi eliminate, și se obține $K = \{IdAngajat, IdProiect\}$
- Atributele $IdProiect$, $IdAngajat$ nu se pot elimina din K , deci $\{IdProiect, IdAngajat\}$ e cheie candidată (chiar primară)

Descompunerea relatiilor

- O descompunere $D = \{R_1, R_2, \dots, R_i, \dots, R_k\}$ a schemei de relație R (relation schema decomposition) este formată din submulțimi proprii ale lui R ($R_1 \subset R, R_2 \subset R, \dots, R_k \subset R$) a caror reuniune este egală cu R ($R = R_1 \cup R_2 \dots \cup R_k$)
- Proiecțiile relației $r(R)$ pe submulțimile $R_1, R_2, \dots, R_i, \dots, R_k$ ($r_1 = \Pi_{R_1}(r), r_2 = \Pi_{R_2}(r), \dots, r_i = \Pi_{R_i}(r), \dots, r_k = \Pi_{R_k}(r)$) reprezintă descompunerea relației $r(R)$ pe aceste submulțimi de attribute
- Fie o relație cu schema R și mulțimea F de DF ale acesteia; o descompunere a relației $r(R)$ este *reversibilă* dacă are proprietățile de:
 - **Conservarea informației** (descompunere fără pierdere de informație la joncțiune) înseamnă ca $r = r_1 \bowtie r_2 \bowtie \dots \bowtie r_i \bowtie \dots \bowtie r_k$
 - **Conservarea dependențelor funcționale**: dacă oricare din dependențele din F se regăsește sau poate fi dedusă din DF ale relațiilor cu schemele $R_1, R_2, \dots, R_i, \dots, R_k$
- **Teorema lui Ullman**. Descompunerea $D = \{R_1, R_2\}$ a unei scheme de relație R este o descompunere fără pierdere de informație la joncțiune în raport cu mulțimea F de DF, dacă și numai dacă este îndeplinită una din condițiile:
(a) $((R_1 \cap R_2) \rightarrow (R_1 - R_2)) \in F^+$, sau (b) $((R_1 \cap R_2) \rightarrow (R_2 - R_1)) \in F^+$
 - Dacă $R_1 \cap R_2$ este o supercheie a uneia dintre relațiile R_1 sau R_2 , atunci descompunerea este fără pierdere de informație la joncțiune;
 - Demonstratie: dacă $(R_1 \cap R_2)$ este o supercheie în R_1 , atunci ea determină funcțional orice submulțime de attribute din R_1 , inclusiv $(R_1 - R_2)$ adică $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$;
 - La fel se demonstrează dacă $(R_1 \cap R_2)$ este o supercheie în R_2

Descompunere fără pierdere de informație la jonctiune

- Lipsa acestei proprietăți se manifestă în mod paradoxal prin apariția unor tupluri noi (parazite) în relația obținută prin jonctiunea relațiilor $r_1, r_2, \dots, r_i, \dots, r_k$, tupluri care nu existau în relația $r(R)$
- Exemplu: descompunerea relației AP în relațiile A și P:

$$A \cap P = \{\text{IdAngajat}\},$$

$$A - P = \{\text{Nume}, \text{Prenume}, \text{Adresa}\}; \quad P - A = \{\text{IdProiect}, \text{Ore}\}$$

Din DF $\text{IdAngajat} \rightarrow \text{Nume}$, $\text{IdAngajat} \rightarrow \text{Prenume}$, $\text{IdAngajat} \rightarrow \text{Adresa}$

Se deduce: $\text{IdAngajat} \rightarrow \{\text{Nume}, \text{Prenume}, \text{Adresa}\}$ (cf. regulii de reuniune a DF)

Rezultă: $((A \cap P) \rightarrow (A - P)) \in F^+$

Deci, conform teoremei lui Ullman desc. este fără pierdere de informație la jonctiune

- Descompunerea succesivă fără pierdere de informație la jonctiune
 - Dacă o descompunere $D = \{R_1, R_2, \dots, R_i, \dots, R_k\}$ a unei scheme R este fără pierdere de informație la jonctiune în raport cu mulțimea F de DF
 - și $D_1 = \{Q_1, Q_2, \dots, Q_m\}$ este o descompunere fără pierdere de informație la jonctiune a lui R_1 în raport cu proiecția lui F pe R_1 ,
 - atunci descompunerea $D_2 = \{Q_1, Q_2, \dots, Q_m, R_2, \dots, R_k\}$ este o descompunere fără pierdere de informație a lui R în raport cu F
- Această proprietate permite descompunerea fără pierdere de informație la jonctiune a unei relații în mai multe etape (mai întâi în două relații, apoi fiecare dintre acestea se descompune în alte două relații, ș.a.m.d)

Conservarea dependențelor funcționale

- O descompunere $D = \{R_1, R_2, \dots, R_i, \dots, R_k\}$ a unei scheme de relație R prezintă proprietatea de conservare a mulțimii F de DF, dacă reuniunea proiecțiilor lui F pe schemele de relații R_i (unde $1 \leq i \leq k$) este echivalentă cu F , adică:
$$((\Pi_{R_1}(F)) \cup (\Pi_{R_2}(F)) \dots \cup (\Pi_{R_k}(F)))^+ = F^+$$
- **Proiecția unei mulțimi de dependențe funcționale.** Fiind dată mulțimea F de DF definite pe R , și o descompunere $D = \{R_1, R_2, \dots, R_k\}$ a lui R , proiecția lui F pe R_i (unde $1 \leq i \leq k$), notată $\Pi_{R_i}(F)$, este mulțimea de DF $X \rightarrow Y \in F^+$, astfel încât $X \subseteq R_i$ și $Y \subseteq R_i$
- Exemplu: descompunerea relației AP în relațiile A și P:
$$F_{AP} = \{IdAngajat \rightarrow Nume, IdAngajat \rightarrow Prenume, IdAngajat \rightarrow Adresa, \{IdAngajat, IdProiect\} \rightarrow Ore\}$$

Proiecțiile mulțimii F_{AP} pe A și P sunt:
$$\Pi_A(F_{AP}) = \{IdAngajat \rightarrow Nume, IdAngajat \rightarrow Prenume, IdAngajat \rightarrow Adresa\}$$

$$\Pi_P(F_{AP}) = \{\{IdAngajat, IdProiect\} \rightarrow Ore\}$$

Dat fiind că $\Pi_A(F_{AP}) \cup \Pi_P(F_{AP}) = F_{AP}$, rezultă că această descompunere conservă DF
- Dacă, prin descompunerea unei relații, se pierde una sau mai multe DF, pentru verificarea lor, trebuie să se efectueze mai întâi joncțiunea relațiilor obținute
- Cele două proprietăți ale unei descompuneri, conservarea informației și conservarea dependențelor funcționale, sunt independente

Formele normale ale relațiilor (FN1 si FN2)

- *O relație este normalizată în prima formă normală (FN1) dacă fiecare atribut ia numai valori atomice și scalare din domeniul său de definiție*
 - Sistemele SGBD relaționale nu admit relații care să nu fie cel puțin în FN1, dar proiectarea relațiilor normalizate în FN1 este întotdeauna posibilă
- *Fie o schema de relație R și mulțimea F de DF definite pe aceasta. O relație $r(R)$ este normalizată în FN2, dacă este în FN1 și dacă în F^+ nu există nici o DF parțială a unui atribut neprim față de o cheie a relației*
- Exemplu: relația AP cu mulțimea F_{AP} de DF este în FN1, dar nu este în FN2:
$$F_{AP} = \{IdAngajat \rightarrow Nume, IdAngajat \rightarrow Prenume, IdAngajat \rightarrow Adresa, \{IdAngajat, IdProiect\} \rightarrow Ore\}$$
 - Cheia primară este $\{IdAngajat, IdProiect\}$, deci $\{IdAngajat, IdProiect\} \rightarrow Nume$
 - Această DF se poate deduce din F_{AP} ; cf. reflexivității, $\{IdAngajat, IdProiect\} \rightarrow IdAngajat$; cf. tranzitivității, dacă $IdAngajat \rightarrow Nume$, $\{IdAngajat, IdProiect\} \rightarrow Nume$
 - DF $\{IdAngajat, IdProiect\} \rightarrow Nume$ este parțială deoarece există: $IdAngajat \rightarrow Nume$
 - Rezultă că relația AP nu este în FN2, deci prezintă redundanțe și anomalii
- Anomaliile se pot evita prin (a) normalizare sau (b) prin impunerea DF
- Normalizarea relației AP prin descompunere în relațiile A și P:
 - $A(IdAngajat, Nume, Prenume, Adresa)$
 - $P(IdAngajat, IdProiect, Ore)$,
 - S-a demonstrat că această descompunere prezintă proprietățile de joncțiune fără pierdere de informație și conservarea DF și se poate arăta că A și P sunt în FN2

Impunerea DF in relatia nenormalizata AP (1)

- Dacă relația AP nu se normalizează, atunci trebuie să se prevadă proceduri speciale care să impună DF care nu sunt determinate de chei
- De exemplu, dacă la inserare nu se admit doi sau mai mulți angajați cu același identificator (IdAngajat) dar cu nume, prenume, adresă diferite se definește următorul trigger (trigger_AP)

- Trigger-ul trigger_AP arata astfel:

```
DELIMITER $$ DROP TRIGGER IF EXISTS `normalizare`.`trigger_AP` $$
CREATE TRIGGER `trigger_AP` BEFORE INSERT ON `normalizare`.`AP` FOR EACH ROW
BEGIN DECLARE done INT DEFAULT 0;
    DECLARE error INT DEFAULT 0;
    DECLARE l_nume, l_prenume, l_adresa VARCHAR(20);
    DECLARE cursor_AP CURSOR FOR SELECT Nume, Prenume, Adresa FROM AP
        WHERE IdAngajat = NEW.IdAngajat;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN cursor_AP;
    REPEAT FETCH cursor_AP INTO l_nume, l_prenume, l_adresa;
        IF NEW.Nume <> l_nume OR NEW.Prenume != l_prenume OR NEW.Adresa <> l_adresa
            THEN SET error = 1; END IF;
    UNTIL done = 1 OR error = 1
    END REPEAT; CLOSE cursor_AP;
    IF error = 1 THEN SET NEW.IdAngajat = NULL;
    END IF;
END $$ DELIMITER ;
```

Impunerea DF in relatia nenormalizata AP(2)

- Trigger-ul este de tip BEFORE INSERT, astfel ca linia dorita se insereaza in tabel numai daca verificarile facute de trigger permit inserarea
- In trigger se citesc intr-un cursor toate liniile care au valoarea atributului IdAngajat egală cu valoarea de inserat (NEW.IdAngajat)
- Se parcurg liniile cursorului si daca valorile de inserat ale atributelor Nume, Prenume, Adresa (NEW.Nume, NEW.Prenume, NEW.Adresa) difera de cele existente in tabel, se seteaza error = 1. Daca dupa parcurgerea cursorului:
 - error = 0, se termină triggerul (instructiunea END), apoi se introduce linia noua
 - error=1, se seteaza SET NEW.IdAngajat = NULL, apoi se termină triggerul (instructiunea END); după aceasta SGBD-ul nu va introduce linia noua deoarece un atribut al cheii primare este NULL (IdAngajat)
- Exemplu:
 - Fie tabelul AP cu liniile (1,'Ionescu','Ion', 'Bucuresti',1,50), (1,'Ionescu','Ion', 'Bucuresti',2,100) si (1,'Ionescu','Ion','Bucuresti',3,80)
 - Se observa redundanta datelor: valorile 'Ionescu', 'Ion', 'Bucuresti' sunt memorate pentru fiecare proiect la care lucreaza angajatul cu IdAngajat = 1
 - Anomalia de inserare: daca nu se activeaza trigger-ul, se poate insera si linia (1,'Marinescu','Mihai','Bucuresti',4,60), ceea ce inseamna ca angajatul cu IdAngajat =1 este nedeterminat (este Ionescu Ion sau Marinescu Mihai?)
 - Aceasta linie nu se poate insera daca s-a activat trigger-ul trigger_AP

A treia forma normala (FN3)

- Fie o schema de relatie R si multimea F de DF definite pe aceasta. O relatie $r(R)$ este normalizată în FN3, dacă este în FN2 și dacă oricare DF din F^+ a unui atribut neprim este determinata de o cheie a relatiei
- Exemplu: AFS(IdAngajat, Nume, Prenume, Adresa, Functie, Salariu)
 $F_{AFS} = \{IdAngajat \rightarrow Nume, IdAngajat \rightarrow Prenume, IdAngajat \rightarrow Adresa, IdAngajat \rightarrow Functie, Functie \rightarrow Salariu\}$.
 - Cheia primară a relației este IdAngajat, și poate fi dedusă din F_{AFS}
 - DF față de cheia primară (primele patru DF) sunt totale, deci relația este în FN2
 - Relația nu este în FN3 datorita DF $Functie \rightarrow Salariu$; prezinta redundante si anomalii
- Normalizare prin descompunere in:
AF(IdAngajat, Nume, Prenume, Adresa, Functie)
FS(Functie, Salariu)
- Se demonstreaza ca AF si FS sunt in FN3 si ca descompunerea este reversibila
 - Proiectiile multimii F_{AFS} :
 $F_{AF} = \{IdAngajat \rightarrow Nume, IdAngajat \rightarrow Prenume, IdAngajat \rightarrow Adresa, IdAngajat \rightarrow Functie\}$
 $F_{FS} = \{Functie \rightarrow Salariu\}$ si se deduc (sau se verifica) cheile relatiilor
 - $PK_{AF} = \{IdAngajat\}$, $PK_{FS} = \{Functie\}$, deci AF si FS sunt in FN3
 - $F_{AF} \cup F_{FS} = F_{AFS}$ deci descompunerea conserva DF
 - $AF \cap FS = \{Functie\}$ si $(Functie \rightarrow Salariu) \in F_{AFS}$, deci, cf. regulii lui Ullman, descompunerea este fara pierdere de informatie la jonctiune

Impunerea DF in relatia nenormalizata AFS (1)

- Dacă relația AFS nu se normalizează, atunci trebuie să se prevadă proceduri speciale care să verifice și să impună DF care nu sunt determinate de chei
- Se poate înlocui operația de INSERT cu apelul unei proceduri stocate care verifică mai întâi valorile și execută INSERT numai dacă acestea respectă DF
- Procedura stocată pentru relația AFS arată astfel:

```
DELIMITER $$ DROP PROCEDURE IF EXISTS sp_AFS $$
CREATE PROCEDURE `sp_AFS`(OUT error INT, s_id INT, s_nume VARCHAR(20), s_prenume
    VARCHAR(20), s_adresa VARCHAR(20), s_functia VARCHAR(20), s_salariu DECIMAL)
BEGIN DECLARE done INT DEFAULT 0;
    DECLARE l_salariu DECIMAL;
    DECLARE cursor_AFS CURSOR FOR
        SELECT Salariu FROM AFS WHERE Functia = s_functia;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    SET error = 0; OPEN cursor_AFS;
    REPEAT FETCH cursor_AFS INTO l_salariu;
        IF s_salariu <> l_salariu THEN SET error = 1; END IF;
    UNTIL done = 1 OR error = 1
    END REPEAT;
    CLOSE cursor_AFS;
    IF error = 0 THEN
        INSERT INTO AFS VALUES (s_id, s_nume, s_prenume, s_adresa, s_functia, s_salariu);END IF;
END$$ DELIMITER ;
```

Impunerea DF in relatia nenormalizata AFS (2)

- Fie tabelul AFS care contine liniile: (1, 'Ionescu', 'Ion', 'Bucuresti', 'inginer', 2000), (2, 'Popescu', 'Petre', 'Craiova', 'inginer', 2000)
- Se observa redundanta datelor: valoarea salariului este memorata de fiecare data pentru o functie anume, desi trebuie sa fie aceeasi (conform DF Functie → Salariu)
- Datorita redundantei pot apare anomalii: se poate insera linia (3, 'Mateescu', 'Viorel', 'Bucuresti', 'inginer', 2100), care este admisă de SGBD, deși nu respectă DF Functie → Salariu: pentru functia 'inginer' se admite si salariul 2000 si salariul 2100; (se sterge apoi linia inserata eronat, pentru ca ulterior sa functioneze bine procedura stocata)
- Inserarea liniilor in tabelul AFS prin apelul procedurii stocate sp_AFS inlatura aceasta anomalii
- De exemplu, la apelul procedurii stocate cu aceleasi valori:

```
call sp_AFS(@error, 7, 'Mateescu', 'Irinel', 'Bucuresti', 'inginer', 2100);  
select @error;
```

se obtine @error=1 si linia respectiva nu va fi introdusa in tabelul AFS

Forma normala Boyce-Codd (FNBC)

- Fie schema de relatie R si multimea F de DF definite pe aceasta. O relatie $r(R)$ este în forma normală Boyce-Codd (FNBC) în raport cu F dacă este în FN1 și dacă orice DF netrivială din F^+ este determinata de o cheie a relației
- Este evident că o relație în FNBC este în FN3, dar o relație în FN3 poate să fie sau nu în FNBC
- Exemplu: relația $EDP(\underline{IdElev}, \underline{IdDisciplina}, IdProfesor)$, cu cheia $PK = \{IdElev, IdDisciplina\}$ și cu mulțimea F_{EDP} de DF:
$$F_{EDP} = \{\{IdElev, IdDisciplina\} \rightarrow IdProfesor, IdProfesor \rightarrow IdDisciplina\}$$
- Se consideră că relația este în FN1; din F_{EDP} se observă că nu există DF parțiale față de cheia relației (deci relația este în FN2) și nu există nici o DF a unui atribut neprim față de un alt atribut neprim, deci relația este în FN3
- Relația EDP nu este în FNBC, datorită DF a atributului prim $IdDisciplina$ față de atributul neprim $IdProfesor$; aceasta relatie prezintă redundante de date și anomalii de actualizare
- Exemplu: fie starea în care tabelul EDP conține liniile $(1,1,1)$ și $(2,1,1)$
 - redundanta: disciplina (1) predată de profesorul 1 este memorată de două ori în tuplurile $(1,1,1)$ și $(2,1,1)$
 - anomalie de inserare: se poate insera și tuplul $(1,2,1)$ care nu respectă DF $IdProfesor \rightarrow IdDisciplina$ (profesorul 1 predă și disciplina 1 și disciplina 2)

Impunerea DF in relatia nenormalizata EDP

- Dacă relația EDP nu se normalizează, atunci trebuie să se prevadă o procedură care să verifice și să impună DF (IdProfesor→IdDisciplina) care nu este determinată de cheia relației, pentru operațiile de INSERT și UPDATE
- De ex: se înlocuiește operația INSERT cu apelul unei proceduri stocate care verifică mai întâi valorile și execută INSERT numai dacă acestea respectă DF

DELIMITER \$\$

```
DROP PROCEDURE IF EXISTS `sp_EDP`$$
```

```
CREATE PROCEDURE `sp_EDP` (INOUT error INT, s_Elev INT, s_Disciplina INT, s_Profesor INT)
```

```
BEGIN DECLARE done INT DEFAULT 0; DECLARE l_Disciplina INT;
```

```
DECLARE cursor_EDP CURSOR FOR
```

```
SELECT IdDisciplina FROM EDP WHERE IdProfesor = s_Profesor;
```

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
```

```
SET error = 0; OPEN cursor_EDP;
```

```
REPEAT FETCH cursor_EDP INTO l_Disciplina;
```

```
IF s_Disciplina <> l_Disciplina THEN SET error = 1; END IF;
```

```
UNTIL done = 1 OR error = 1
```

```
END REPEAT;
```

```
CLOSE cursor_EDP;
```

```
IF error = 0 THEN INSERT INTO EDP VALUES (s_Elev, s_Disciplina, s_Profesor);
```

```
END IF; END$$ DELIMITER ;
```

- Pentru verificare: se șterge linia(1,2,1) (dacă există) și se execută:
 - call sp_EDP(@error,1,2,1); select @error;
 - se obține @error = 1 și nu s-a inserat acest tuplu

Normalizarea relatiei EDP

- Normalizarea relației EDP astfel încât relațiile obținute să fie în FNBC se poate face prin descompunerea acesteia. Se pot încerca trei descompuneri:
D1 = {EP,PD}, unde EP = {IdElev, IdProfesor},
PD = {IdProfesor, IdDisciplina};
D2 = {ED,PD}, unde ED = {IdElev, IdDisciplină},
PD = {IdProfesor, IdDisciplina};
D3 = {EP,ED}, unde EP = {IdElev, IdProfesor},
ED = {IdElev, IdDisciplina}.
- Se poate observa că relațiile rezultate în oricare din aceste descompuneri sunt relații în FNBC (fiind relații formate din două attribute).
- Dintre cele trei descompuneri, descompunerea D1 prezintă proprietatea de joncțiune fără pierdere de informație. Într-adevăr, $EP \cap PD = \{IdProfesor\}$, $PD - EP = \{IdDisciplina\}$ și $IdProfesor \rightarrow IdDisciplina$, deci este îndeplinită condiția lui Ullman de conservare a informației.
- Celelalte descompuneri, D2 și D3, nu îndeplinesc această condiție.
- În oricare din aceste descompuneri se pierde dependența funcțională $\{IdElev, IdDisciplina\} \rightarrow IdProfesor$, deci relația EDP nu poate fi descompusă în mod reversibil în relații FNBC.

Impunerea constrangerilor pierdute prin descompunere (1)

- Dacă relația EDP se normalizează prin descompunerea (EP, PD) atunci trebuie să se prevadă o procedură care să verifice și să impună constrangerea pierdută
- Se pot înlocui operațiile de INSERT în tabelele EP, PD cu apelul unei proceduri stocate care verifică mai întâi valorile și execută INSERT numai dacă acestea respectă constrangerea: {IdElev, IdDisciplina} → IdProfesor

```
DELIMITER $$ DROP PROCEDURE IF EXISTS `sp_EP_PD`$$
CREATE PROCEDURE `sp_EP_PD` (OUT error INT, s_Elev INT, s_Disciplina INT, s_Profesor INT)
BEGIN DECLARE done INT DEFAULT 0;
    DECLARE l_Elev, l_Profesor, l_Disciplina INT;
    DECLARE cursor_EP_PD CURSOR FOR
        SELECT IdElev, EP.IdProfesor, IdDisciplina FROM EP, PD
            WHERE EP.IdProfesor = PD.IdProfesor ;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    SET error = 0; OPEN cursor_EP_PD;
    REPEAT FETCH cursor_EP_PD INTO l_Elev, l_Profesor, l_Disciplina;
        IF s_Elev = l_Elev AND s_Disciplina = l_Disciplina AND s_Profesor <> l_Profesor THEN
            SET error = 1; END IF;
    UNTIL done = 1 OR error = 1
    END REPEAT;
    CLOSE cursor_EP_PD;
    IF error = 0 THEN
        INSERT INTO EP VALUES (s_Elev, s_Profesor);
        INSERT INTO PD VALUES (s_Profesor, s_Disciplina);
    END IF; END$$ DELIMITER ;
```

Impunerea constrangerilor pierdute prin descompunere (2)

- Procedura `sp_EP_PD` primește ca argumente un flag de eroare și valorile celor trei atribute `IdElev`, `IdDisciplina`, `IdProfesor` care trebuie să respecte constrângerea $\{IdElev, IdDisciplina\} \rightarrow IdProfesor$
- În această procedură se creează un cursor în care se încarcă rezultatul joncțiunii naturale între relațiile EP și PD pe atributul comun `IdProfesor`
- Pentru fiecare linie a rezultatului joncțiunii EP \bowtie PD se verifică respectarea constrângerii dorite, testând valorile existente în linia respectivă cu noile valori care urmează să fie introduse:
 - dacă această constrângere este respectată, atunci se execută două instrucțiuni `INSERT`, câte una în fiecare din tabelele EP și PD
 - dacă această constrângere nu este respectată, nu se introduce nici o linie
- Exemplu: dacă tabelul EP conține linia (1,1) și tabelul (PD) conține linia (1,1), prin `INSERT` se pot introduce și valorile (1,1,2) pentru (`IdElev`, `IdDisciplina`, `IdProfesor`) adică liniile: (1,2) în EP și (2,1) în PD; dar aceste valori nu respectă constrângerea $\{IdElev, IdDisciplina\} \rightarrow IdProfesor$ deoarece:
 - în liniile existente (`IdElev`, `IdDisciplina`) = (1,1), iar `IdProfesor` = 1
 - în valorile de inserat (`IdElev`, `IdDisciplina`) = (1,1), iar `IdProfesor` = 2
- În schimb apelul procedurii: `call sp_EP_PD(@error, 1,1,2); select @error;` produce `@error=1` și nu se introduce nici o linie

Dependente multivalorice

- O dependență multivalorică - DMV- (multivalued dependency) $X \twoheadrightarrow Y$ specificată pe schema de relație $R = \{X, Y, Z\}$ stabilește următoarele constrângeri pe care trebuie să le respecte orice relație $r(R)$: dacă există două tupluri t_1 și t_2 în r astfel ca $t_1[X] = t_2[X] = x$, atunci vor exista și tuplurile t_3 și t_4 cu următoarele proprietăți:
 $t_3[X] = t_4[X] = t_1[X] = t_2[X] = x$;
 $t_3[Y] = t_1[Y] = y_1$ și $t_4[Y] = t_2[Y] = y_2$;
 $t_3[Z] = t_2[Z] = z_2$ și $t_4[Z] = t_1[Z] = z_1$.
- Datorită simetriei egalităților de mai sus, rezulta că, dacă într-o relație cu schema R există DMV $X \twoheadrightarrow Y$, atunci există și $X \twoheadrightarrow Z$, unde $Z = R - (X \cup Y)$
- O DF este un caz particular al unei DMV: DF $X \rightarrow Y$ este o DMV $X \twoheadrightarrow Y$ cu restricția că unei valori a lui X îi corespunde o singură valoare a lui Y
- O relație cu schema R este în a patra formă normală (FN4) în raport cu o mulțime F de DF și DMV dacă este în FN1 și dacă, pentru orice DMV netrivială $X \twoheadrightarrow Y$ din F^+ , X este cheie a relației $r(R)$.
- Asemănarea acestei definiții cu definiția FNBC: pentru FNBC se impun restricții DF, iar pentru FN4 se impun restricții DMV
- Dacă o schemă de relație respectă condiția de FN4, atunci înseamnă că ea respectă și condiția de FNBC

Dependente de jonctiune

- Fie o schema de relație R și R_1, R_2, \dots, R_k submulțimi de attribute ale lui R , unde $R_1 \cup R_2 \cup \dots \cup R_k = R$. Se spune că există o dependență de jonctiune (DJ) pe R , notată $*(R_1, R_2, \dots, R_k)$, dacă și numai dacă orice relație $r(R)$ este egală cu jonctiunea proiecțiilor relației pe submulțimile R_1, R_2, \dots, R_k , adică $r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \dots \bowtie \Pi_{R_k}(r)$.
- Rezulta ca:
 - $*(R_1, R_2, \dots, R_k)$ este o DJ pe R dacă și numai dacă descompunerea lui R în proiecțiile pe R_1, R_2, \dots, R_k este fără pierdere de informație la jonctiune
 - Relația $r(R)$ este k -decompozabilă fără pierdere de informație dacă există o DJ $*(R_1, R_2, \dots, R_k)$
- Tipuri de DJ (dupa valoarea lui k):
 - Cazul $k = 1$ reprezintă o DJ trivială
 - Cazul $k = 2$ al unei DJ este o DMV; o DMV $X \twoheadrightarrow Y$ în relația $r(R)$ reprezintă o DJ $*(XY, XZ)$, unde $Z = R - (X \cup Y)$.
 - În cazul $k > 2$, DJ nu mai sunt echivalente cu DMV
- DJ sunt dificil de identificat și nu există reguli de deducere (inferență) care să genereze toate DJ pornind de la o mulțime dată
- Datorită acestor aspecte, analiza DJ are un pronunțat caracter intuitiv

Forma normala FN5

- *O relație cu schema R este în a cincea formă normală (FN5) în raport cu o mulțime F de dependențe funcționale, multivalorice sau de joncțiune dacă este în FN1 și dacă, pentru orice dependență de joncțiune $*(R1, R2, \dots Ri, \dots Rk)$ din F^+ , Ri (unde $1 \leq i \leq k$) este o cheie a relației $r(R)$*
- Având în vedere faptul că o dependență multivalorică este un caz special de dependență de joncțiune, iar o dependență funcțională este un caz special de dependență multivalorică, se poate afirma că o relație care este în FN5, este implicit în FN4, deci și în FNBC, ș.a.m.d.
- S-a demonstrat că orice relație poate fi transformată în relații FN5 (sau FN4, sau FNBC) printr-o descompunere fără pierdere de informație la joncțiune, dar nu se asigura conservarea tuturor DF
- Condițiile de normalizare în FNBC, FN4 și FN5 se pot rezuma la faptul că într-o relație normalizată nu există decât dependențe determinate de chei:
 - O relație este în FNBC dacă orice DF este determinată de o cheie a relației
 - O relație este în FN4 dacă orice DF sau DMV este determinată de o cheie a relației
 - O relație este în FN5 dacă orice DF, DMV sau DJ este determinată de o cheie a relației

Proiectarea relațiilor normalizate

- Normalizarea relațiilor asigură un proiect al bazei de date mai concis și de aceea se consideră că a normaliza este avantajos, chiar dacă normalizarea nu este o garanție că s-a realizat cel mai bun model
- Proiectarea bazelor de date pornind de la diagrama Entitate-Asociere conduce, în general, la relații normalizate, deoarece:
 - Relațiile corespunzătoare mulțimilor de entități sunt, de regulă, relații normalizate, dat fiind că toate atributele descriu tipul de entitate respectiv.
 - Relațiile de asociere binară, care conțin două chei străine care referă cheile primare din cele două relații pe care le asociază, rezultă tot ca relații normalizate
 - Relațiile care modelează asocierile multiple pot să rezulte nenormalizate și necesită operații de normalizare suplimentare
- Dar, cu cât nivelul de normalizare crește, cu atât se obțin mai multe relații cu grad mai mic și pentru fiecare interogare sunt necesare mai multe operații de joncțiune, ceea ce face ca timpul de execuție a interogărilor să crească; în general, se recomandă ca:
 - relațiile asupra cărora se efectuează operații de actualizare frecvente să fie normalizate într-o formă normală cât mai avansată
 - relațiile asupra cărora se efectuează interogări frecvente pot fi păstrate într-o formă de normalizare mai redusă
- Menținerea unei relații într-o formă de normalizare mai redusă se numește “denormalizare”, și are scopul de a obține performanțe ridicate la interogări

Algoritmi de normalizare

- Analiza normalizării relațiilor trebuie să fie făcută pentru orice proiect de baze de date, pentru a asigura funcționarea corectă a acestora:
 - Dacă o relație se păstrează într-o formă de normalizare mai redusă, atunci trebuie să se prevadă proceduri de verificare și impunere a dependențelor de date care nu sunt determinate de cheile relației (ca și constrângeri explicite)
 - Dacă se normalizează o relație, dar prin descompunere se pierde unele DF, acestea pot fi impuse explicit prin proceduri stocate sau funcții în programele de aplicație, care execută joncțiunea între relațiile rezultate și impun constrângerea respectivă
- S-a demonstrat și există algoritmi prin care orice relație poate fi descompusă reversibil (cu conservarea informației și conservarea DF) în relații în formele normale FN2 sau FN3
- S-a demonstrat și există algoritmi prin care orice relație poate fi descompusă în relații FN2, FN3, FNBC, FN4 sau FN5 fără pierdere de informație la joncțiune, dar se pot pierde unele dependențe

Descompunerea fara pierdere de informatie la jonctiune (1)

- Fiind dată o relație cu schema R și mulțimea F de DF, o descompunere D fără pierdere de informație la jonctiune în relații într-una din formele normale FN2, FN3 sau FNBC se poate obține aplicând algoritmul următor:
 1. se setează $D = \{R\}$;
 2. atât timp cât în D există o relație Q (cu mulțimea F_Q a DF) care nu se află în forma normală dorită:
 - se alege o DF $X \rightarrow W$ din F_Q care nu respecta forma normala dorita și se construiește închiderea X^+ a atributului X și mulțimea $Y = X^+ - X$;
 - în D se înlocuiește relația Q cu două relații: $Q1 = X \cup Y$ și $Q2 = X \cup Z$, unde $Z = Q - (X \cup Y)$;
- Demonstrație:
 - La fiecare pas de execuție a algoritmului, o relație Q se descompune în două relații $Q1$ și $Q2$, astfel încât $Q1 \cap Q2 = X$, și $Q1 - Q2 = Y$
 - Din definiția închiderii unui atribut rezultă că $X \rightarrow Y$, deci, conform teoremei lui Ullman, această descompunere este fără pierdere de informație la jonctiune
 - Astfel de descompuneri succesive păstrează caracterul de descompuneri fără pierdere de informație la jonctiune
- Există algoritmi similari pentru descompunerea fără pierdere de informatie la jonctiune a unei relații în relații în forme normale FN4 sau FN5

Descompunerea fara pierdere de informatie la jonctiune (2)

- Exemplu de aplicare a algoritmului pentru descompunerea in relatii FNBC a relatiei $R = \{\underline{\text{IdAngajat}}, \text{Nume}, \text{Prenume}, \text{Adresa}, \text{Functie}, \text{Salariu}, \underline{\text{IdProiect}}, \text{Ore}\}$ cu $\text{PK} = \{\text{IdAngajat}, \text{IdProiect}\}$, aflata în FN1 și cu mulțimea F de DF
 $F = \{\text{IdAngajat} \rightarrow \text{Nume}, \text{IdAngajat} \rightarrow \text{Prenume}, \text{IdAngajat} \rightarrow \text{Adresa}, \text{IdAngajat} \rightarrow \text{Functie}, \text{Functie} \rightarrow \text{Salariu}, \{\text{IdAngajat}, \text{IdProiect}\} \rightarrow \text{Ore}\}$
- Executia algoritmului:
 - Se setează $D = \{R\}$
 - Din F se alege DF $\text{IdAngajat} \rightarrow \text{Nume}$ care nu respectă condiția impusă de FNBC
 - Închiderea atributului $X = \text{IdAngajat}$ este $[\text{IdAngajat}]^+ = \{\text{IdAngajat}, \text{Nume}, \text{Prenume}, \text{Adresa}, \text{Functie}, \text{Salariu}\}$ și rezultă $Y = X^+ - X = \{\text{Nume}, \text{Prenume}, \text{Adresa}, \text{Functie}, \text{Salariu}\}$ și $Z = R - (X \cup Y) = R - X^+ = \{\text{IdProiect}, \text{Ore}\}$.
 - Se obtine descompunerea D a relației R : $D = \{R_{11}, R_{12}\}$, unde
 $R_{11} = X \cup Y = \{\underline{\text{IdAngajat}}, \text{Nume}, \text{Prenume}, \text{Adresa}, \text{Functie}, \text{Salariu}\}$; in FN2
 $R_{12} = X \cup Z = \{\underline{\text{IdAngajat}}, \underline{\text{IdProiect}}, \text{Ore}\}$; in FNBC
 - In acelasi mod se descompune relația R_{11} , in relatiile R_{111} si R_{112} :
 $R_{111} = \{\underline{\text{Functie}}, \text{Salariu}\}$; este in FNBC
 $R_{112} = \{\underline{\text{IdAngajat}}, \text{Nume}, \text{Prenume}, \text{Adresa}, \text{Functie}\}$; este in FNBC
- Se poate demonstra usor ca descompunerea obtinuta $D = (R_{111}, R_{112}, R_{12})$ conserva si dependentele functionale

