# Normalizarea

Normalizarea este procesul de proiectare a structurii unei tabele pentru a se minimiza redundanţa datelor, urmărindu-se evitarea anomaliilor în cazul operaţiilor de actualizare. Astfel, schemele de relaţie nesatisfăcătoare sunt descompuse obţinându-se mai multe scheme de relaţie mai mici care să respecte proprietăţile dorite. Normalizarea nu asigură însă în mod necesar buna construcţie a bazei de date.

Din punct de vedere structural, formele normale respectă relaţia FN1 < FN2 < FN3 < FNBC, astfel că o formă normală de ordin superior este mai bună decât o formă normală de ordin inferior, cel puţin din punctul de vedere al anomaliilor ce pot apărea în cadrul procesului de actualizare. Au fost definite şi forme normale superioare, precum forma normală 4 (FN4) şi forma normală 5 (FN5) sau forma normală domeniu-cheie (DKNF - eng. domanin-key normal form), însă acestea nu sunt întâlnite în mediul de afaceri, având o imporanţă strict teoretică.

În stabilirea formei normale pe care o va respecta schema de relaţie trebuie să se aibă în vedere şi principiul vitezei de răspuns a interogărilor, care depinde de numărul de joncţiuni realizat între tabele, acesta fiind invers proporţional cu forma normală. De cele mai multe ori, forma normală 3 este suficientă spre a satisface cerinţele organizaţiilor, realizând cel mai bun compromis între evitarea anomaliilor în operaţiile de manipulare a datelor şi asigurarea unei viteze de răspuns corespunzătoare.

Procesul de normalizare asigură conformitatea unei tabele fată de conceptul de **relaţie bine formată**, caracterizată prin:

1. fiecare tabelă corespunde unei singure entităţi, conţinând exclusiv atributele specifice acesteia;
2. principiul redundanţei minime şi controlate - nici o informaţie nu va fi reţinută în mai mult de o tabelă în cazul în care nu este necesar;
3. atributele non-prime sunt dependente doar de cheia primară astfel că aceasta le identifică în mod unic;
4. principiul integrităţii şi consistenţei datelor: nici o tabelă nu conţine anomalii la adăugare, modificare sau ştergere.

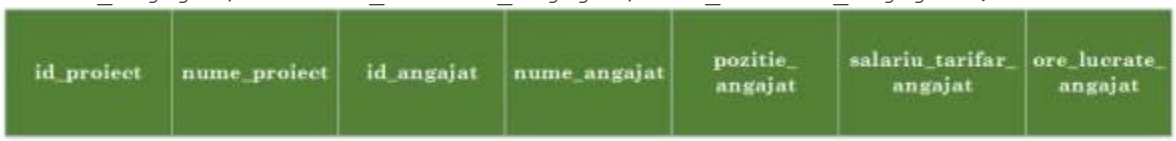
O **supercheie** $S$ într-o relaţie $R = (a_1, a_2 \dots , a_n)$ este un set de atribute din $R$ având proprietatea că nu există două n-tupluri $t_1$ şi $t_2$ în orice instanţă $r$ a lui $R$ astfel ca $t_1(S)=t_2(S)$. Diferenţa între o cheie şi o supercheie constă în faptul că întotdeauna cheia conţine un număr minim de atribute. Un atribut al relaţiei $R$ se numeşte prim dacă este membru al unei superchei din $R$ şi este nonprim dacă nu este un atribut prim. Prin urmare, o cheie primară este o supercheie minimală (ireductibilă).
O **dependenţă funcţională** este o relaţie între două atribute $X$ şi $Y$ ale unei relaţii $R$, notată $X \rightarrow Y$, cu proprietatea că fiecare valoare a lui $X$ determină o singură valoare a lui $Y$. Cu alte cuvinte, oricare ar fi două tupluri $t_1$ şi $t_2$ din $R$, astfel încât $t_1(X)=t_2(X)$, atunci $t_1(Y)=t_2(Y)$.
O dependenţă funcţională $X \rightarrow Y$ (cu $X$ atribut compus) este **completă** dacă prin eliminarea oricărui atribut $Z \in X$ dependenţa funcţională este distrusă.
O dependenţă funcţională $X \rightarrow Y$ (cu $X$ atribut compus) este **parţială** dacă există un atribut (sau set de atribute) $Z \in X$ astfel încât $X \setminus \{Z\} \rightarrow Y$.

O dependenţă funcţională X → Y (cu X cheie primară) este **tranzitivă** dacă există un atribut (sau set de atribute) Z care nu fac parte din cheia primară astfel încât X → Z şi Z → Y.
Dependenţele funcţionale tranzitive pot fi identificate cu uşurinţă în schemele de relaţie unde există dependenţe funcţionale între atribute non-prime. Prin urmare, modificările în structurile tabelelor caracterizate prin această problemă vor porni de la dependenţa funcţională dintre atributele non-prime.

## Exemplu

Se consideră o schemă de relaţie pentru gestiunea proiectelor şi a resurselor umane din cadrul unei organizaţii, reţinând câte ore a lucrat fiecare angajat dintr-un departament la proiectul la care a fost asociat, precum şi salariul său tarifar:

`gestiune_organizatie = { id_proiect, nume_proiect, id_angajat, nume_angajat, pozitie_angajat, salariu_tarifar_angajat, ore_lucrate_angajat }`




Această schemă de relaţie este caracterizată prin **inconsistenţa datelor** (acelaşi departament poate fi exprimat prin valori diferite) cât şi prin **redundanţă** (pentru fiecare angajat se reţin de mai multe ori numele, departamentul şi salariul tarifar - dacă a lucrat la mai multe proiecte), ceea ce determină anomalii la principalele operaţii de manipulare a datelor:

- adăugare (pentru proiect/angajat se introduc valori null)
- modificare (datele referitoare la proiect/angajat trebuie actualizate în toate tuplurile care le referă)
- ștergere (atunci când un proiect/angajat este eliminat, se pierd şi informaţiile aferente acestora).

Pentru eliminarea acestor probleme, au fost definite forme normale:

## Forma Normală 1 (FN1)

Forma normală primară (FN1) nu permite ca pentru un atribut de tip cheie să existe mai multe valori identice sau ca schema de relaţie să conţină atribute compuse.

În acest sens, se impune **stabilirea unei chei primare** care să identifice în mod unic tuplurile din schema de relaţie, **punându-se în evidenţă totodată şi dependenţele funcţionale**.
În cazul exemplului, se observă cheia primară `(id_proiect, id_angajat)`, evidenţiindu-se totodată şi următoarele dependenţe funcţionale:

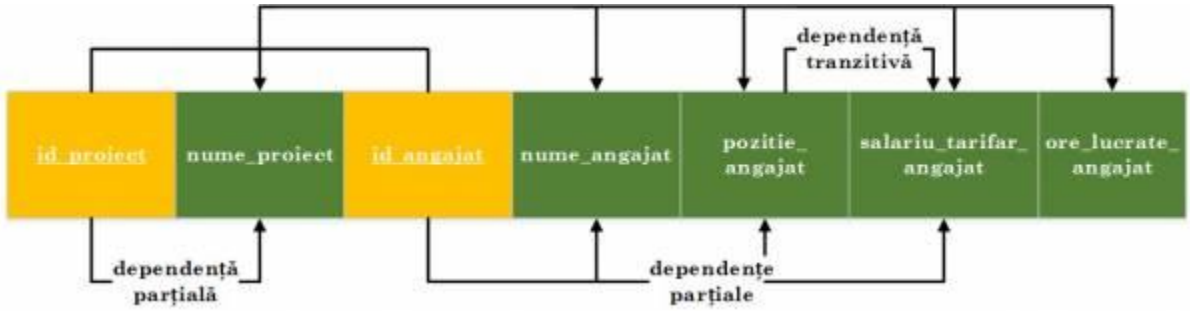
- dependenţe funcţionale parţiale

  `id_proiect → nume_proiect`

  `id_angajat → nume_angajat, pozitie_angajat, salariu_tarifar_angajat`
- dependenţe funcţionale tranzitive

  `pozitie_angajat → salariu_tarifar_angajat`

Astfel, schema de relaţie în care s-a identificat cheia primară (compusă) şi dependenţele funcţionale, respectă forma normală 1, reprezentată prin următoarea diagramă funcţională:




## Forma Normală 2 (FN2)

Forma normală secundară (FN2) este satisfăcută de schemele de relaţie care îndeplinesc condiţiile formei normale primare şi pentru care orice atribut nonprim este complet dependent funcţional de cheia primară.

În acest sens, se impune **eliminarea dependenţelor funcţionale parţiale** care implică redundanţa datelor şi anomalii la operaţiile de manipulare a acestora. În cazul în care cerinţele legate de performanţă (viteză de răspuns) implică păstrarea dependenţelor de tip parţial, este recomandată trecerea la implementarea unui depozit de date în care redundanţa reprezintă un principiu de proiectare.
**Observaţie**. Orice schemă de relaţie ce respectă FN1 şi are cheia primară formată dintr-un singur atribut respectă în mod automat şi FN2.
Se vor crea noi scheme de relaţie având drept chei primare atributele care determină dependenţele funcţionale parţiale, împreună cu atributele non-prime care se află în relaţie cu ele. Ele vor rămâne şi în schema de relaţie iniţială cu atributele non-prime faţă de care există dependenţe funcţionale complete. Menţinerea în schema de relaţie iniţială a tuturor atributelor care alcătuiesc cheia primară este determinată de faptul că acestea vor reprezenta referinţe pentru legăturile ce se vor stabili cu noile scheme de relaţie constituite.

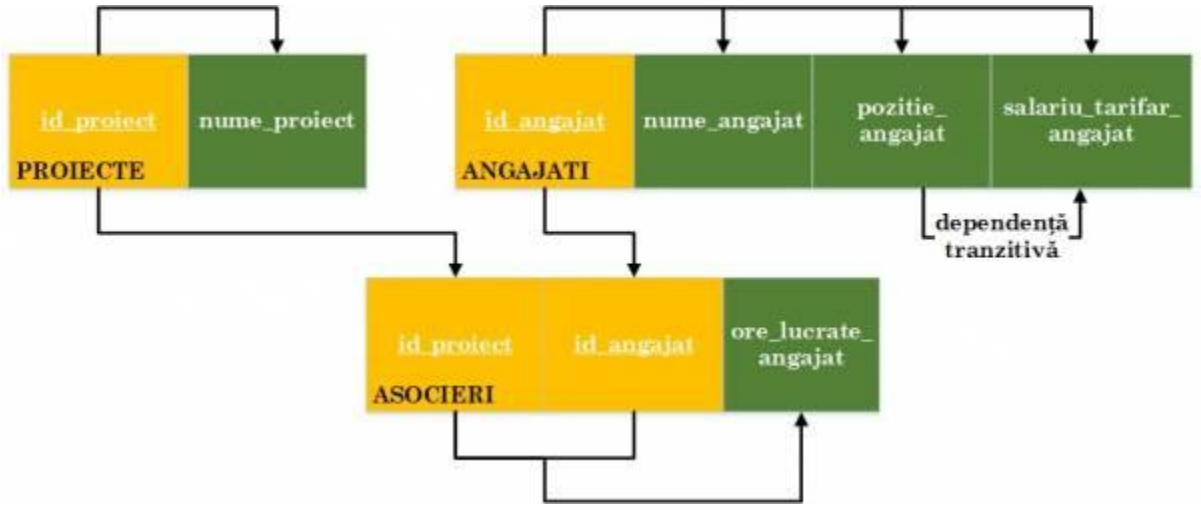
Pentru exemplul de faţă se vor constitui tabelele:

- `proiecte` (cu cheia primară `id_proiect`) din care face parte atributul non-prim determinat `nume_proiect`
- `angajati` (cu cheia primara `id_angajat`) din care fac parte atributele non-prime determinate
  `nume_angajat`, `pozitie_angajat`, `salariu_tarifar_angajat`.

În schema de relaţie iniţială (având cheia primară compusă `(id_proiect, id_angajat)`) rămâne atributul non-prim `ore_lucrate_angajat`, complet dependent funcţional de cheia primară.

1. `proiecte = {id_proiect, nume_proiect}`
   `id_proiect → nume_proiect`
2. `angajati = {id_angajat, nume_angajat, pozitie_angajat, salariu_tarifar_angajat}`
   `id_angajat → nume_angajat, pozitie_angajat, salariu_tarifar_angajat`
   `pozitie_angajat → salariu_tarifar_angajat (dependența tranzitivă)`

3. asocieri = {id_proiect, id_angajat, ore_lucrate_angajat}
   (id_proiect, id_angajat) → ore_lucrate_angajat



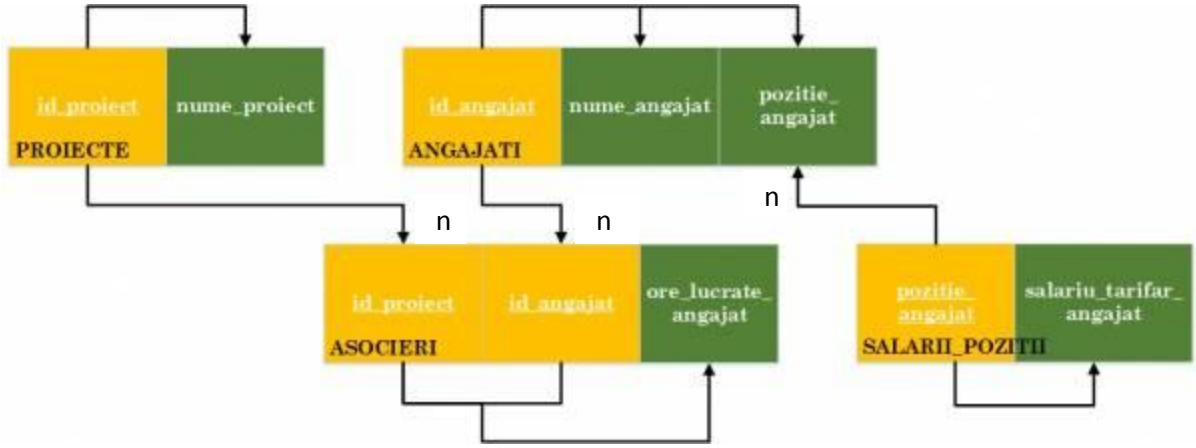

## Forma Normală 3 (FN3)

Forma normală terţiară (FN3) este îndeplinită de schemele de relaţie care satisfac condiţiile formei normale secundare şi în care nu există dependenţe tranzitive.

În acest sens, se impune **eliminarea dependenţelor funcţionale tranzitive** care implică redundanţa datelor şi anomalii la operaţiile de manipulare a acestora.
Se vor crea noi scheme de relaţie având drept chei primare atributele care determină dependenţele funcţionale tranzitive, împreună cu atributele ce se află în relaţie cu ele. Ca şi în situaţia descompunerii din cazul formei normale 2, ele vor rămâne şi în schema de relaţie iniţială.

Pentru exemplul de faţă se va constitui tabela `salarii_pozitii`, având cheia primară `pozitie_angajat` şi atributul `salariu_tarifar_angajat` ce va fi eliminat din schema de relatie `angajaţi`.

1. `proiecte = {id_proiect, nume_proiect}`
   `id_proiect → nume_proiect`
2. `angajati = {id_angajat, nume_angajat, pozitie_angajat}`
   `id_angajat → nume_angajat, pozitie_angajat`
3. `salarii_pozitii = {pozitie_angajat, salariu_tarifar_angajat}`
   `pozitie_angajat → salariu_tarifar_angajat`
4. `asocieri = {id_proiect, id_angajat, ore_lucrate_angajat}`
   `(id_proiect, id_angajat) → ore_lucrate_angajat`




În procesul de normalizare a unei scheme de relaţie trebuie asigurate iniţial formele normale inferioare trecându-se apoi la formele normale superioare.

În cazul FN2 se elimină dependenţele funcţionale parţiale, iar în cazul FN3 dependenţele funcţionale tranzitive, mecanismul fiind acelaşi: se creează noi scheme de relaţie conţinând atributele implicate în dependenţa funcţională respectivă, păstrând atributele determinante în schema de relaţie iniţială (pentru a servi drept referinţe) şi scoţând atributele determinate.