4 NORMALIZAREA RELAȚIILOR

4.1 Scopul normalizării

La proiectarea unei BD relaționale, principalul obiectiv în realizarea unui model logic este crearea unei reprezentări corecte a datelor, relațiilor dintre ele și a constrângerilor. Pentru atingerea acestui obiectiv, trebuie identificat un set adecvat de relații. Normalizarea reprezintă o tratare **de jos în sus** a proiectării bazelor de date, care începe prin examinarea relațiilor dintre atribute. Totuşi, de multe ori metodologia de proiectare abordează o tratare **de sus în jos** a BD (care începe prin identificarea principalelor entități şi relații), caz în care normalizarea este folosită ca tehnică de validare.

> **Procesul de normalizare** este o metodă formală, care identifică relațiile bazându-se pe ***cheile primare*** ale acestora şi pe ***dependențele funcționale*** dintre atributele lor. Normalizarea ajută proiectanții de BD, prin prezentarea unei serii de teste care pot fi aplicate relațiilor individuale, pentru ***a preveni apariția anomaliilor de reactualizare***.

Unul din principalele scopuri urmărite la proiectarea BD relaționale, este **gruparea atributelor în relații** în aşa fel încât să se **minimizeze redundanța datelor** şi prin aceasta să se reducă spațiul de stocare necesar relațiilor de bază implementate. Problemele asociate redundanței datelor le vom ilustra printr-un exemplu.

Exemplu: să analizăm următoarele alternative de relații:

Personal (PersID, NumeP, AdresaP, Funcție, Salariu, FilialaID)
şi
Filiala (FilialaID, AdresaF, TelF)

față de

PersonalFiliala (PersID, NumeP, AdresaP, Funcție, Salariu, FilialaID, AdresaF, TelF)

În relația **PersonalFiliala** există **date redundante**: detaliile referitoare la filială sunt repetate pentru fiecare membru al personalului aflat la filiala respectivă. Relațiile care conțin date redundante pot crea probleme, denumite anomalii de reactualizare.

Anomaliile de reactualizare se clasifică în:

- **anomalii de inserare** care pot fi de două tipuri:
  - anomalii privind identitatea datelor redundante: de ex, pentru inserarea noilor membri ai personalului, trebuie incluse detalii despre filiala la care vor lucra, detalii care trebuie să coincidă cu valorile aflate pe celelalte rânduri ale relației, altfel provocăm o incoerență a BD.
  - anomalii privind necesitatea introducerii de rânduri cu null pentru cheia primară: de ex, pentru a insera o nouă filială, care nu are nici un personal, este necesară introducerea de null-uri pentru atributele personalului; dar PersID este cheie primară şi nu e permis null-ul (deoarece se violează integritatea entităților)
- **anomalii de ştergere**: ştergerea anumitor înregistrări duce la pierderea unor detalii care nu sunt stocate în altă parte: dacă se şterge ultimul membru al personalului de la o filială, se pierd detaliile despre filială

- **anomalii de modificare**: necesiatatea modificării unei date redundante, presupune modificarea ei în toate înregistrările în care ea apare: dacă trebuie modificat unul din atributele unei filiale, este necesară reactualizarea rândurilor corespunzătoare pentru toți membrii personalului de la filiala respectivă, altfel BD devine incoerentă.

Această analiză arată că relațiile **Personal** şi **Filiala** au o structură mai bună decât **PersonalFiliala**. Procesul de normalizare furnizează o tehnică de proiectare a unor relații mai bine structurate.

4.2 Dependențe funcționale

Dependențele funcționale sunt concepte fundamentale în procesul de normalizare.

Dependența funcțională descrie **legăturile dintre atributele unei relații:** fie A şi B două atribute ale relației R; atributul B este dependent funcțional de A (notat **A→B**) dacă **fiecărei valori a atributului A îi corespunde o singură valoare a atributului B**. A şi B pot fi simple sau compuse.

Când există o dependență funcțională, ea este specificată ca o **constrângere** între atribute. Atributul din stânga săgeții se numeşte **determinant**.

Exemplu: să considerăm atributele PersID şi Salariu din relația **Personal**

Personal (PersID, NumeP, AdresaP, Funcție, Salariu, FilialaID)

PersID → Salariu deci un membru al personalului are un singur salariu
Salariu ×→ PersID un salariu **nu** determină un singur membru al personalului

Exemplu: Să identificăm dependențele funcționale din relația **PersonalFiliala**

PersonalFiliala (PersID, NumeP, AdresaP, Funcție, Salariu, FilialaID, AdresaF, TelF)

PersID → NumeP
PersID → AdresaP
PersID → Funcție
PersID → Salariu
PersID → FilialaID
PersID → AdresaF
PersID → TelF
FilialaID→ AdresaF
FilialaID→ TelF
AdresaF→ FilialaID
AdresaF→ TelF
TelF→ FilialaID
TelF→ AdresaF

În această relație sunt 13 dependențe funcționale cu PersID, FilialaID, AdresaF şi TelF ca **determinanți**. Un format alternativ de afişare a acestor dependențe este:

PersID → NumeP, AdresaP, Funcție, Salariu, FilialaID, AdresaF, TelF
FilialaID → AdresaF, TelF
AdresaF → FilialaID, TelF
TelF → FilialaID, AdresaF

Pentru a identifica **cheia candidat** (sau cheile candidat) din relația **PersonalFiliala**, este necesar să recunoaştem **atributul** (sau grupul de atribute) care identifică în mod unic fiecare rând din relație. Dacă o relație are mai mult de o chei candidat, trebuie identificată cheia primară. Toate atributele care nu fac parte din cheia primară, trebuie să fie dependente funcțional de această cheie.

Singura cheie candidat pentru relația **PersonalFiliala** (deci şi cheie primară) este PersID, deoarece toate celelalte atribute ale relației sunt dependente funcțional de aceasta. Cu toate că atributele FilialaID, AdresaF, TelF sunt determinanți în această relație, ele nu constituie chei candidat pentru ea.

In orice relație, atributele sunt dependente funcțional de față de cheile acesteia, deoarece orice cheie are proprietatea că identifică în mod unic fiecare tuplă, deci determină în mod univoc valorile atributelor tuplei.

4.3 Procesul de normalizare - descompunerea schemelor de relație

Conceptul de dependență funcțională este elementul central în procesul de normalizare. Normalizarea este o tehnică formală de analiză a relațiilor, care se bazează pe ***cheile primare*** şi ***dependențele funcționale***. Tehnica presupune o serie de reguli care pot fi utilizate pentru testarea relațiilor individuale, astfel încât o BD poate fi normalizată până la orice grad. Atunci când o cerință nu este îndeplinită, relația care o deranjează trebuie să fie descompusă în relații care satisfac individual cerințele normalizării.

Adeseori, normalizarea este executată sub forma unei serii de paşi. Fiecare pas corespunde unei anumite **forme normale**, care are proprietăți cunoscute. Pe măsură ce se desfăşoară normalizarea, relațiile devin în mod progresiv mai restrictive (mai puternice) ca format şi mai puțin vulnerabile la anomaliile de reactualizare. Pentru modelul relațional, numai prima formă normală (**1NF**) este de importanță critică în crearea de relații adecvate. Toate formele normale următoare sunt opționale. Totuşi, pentru evitarea **anomaliilor de reactualizare**, **se recomandă efectuarea normalizării până la cel puțin forma 3NF**.

Forma nenormalizată (UNF) este un tabel care conține unul sau mai multe ***grupuri repetitive***.

Un ***grup repetitiv*** este un atribut sau grup de atribute din cadrul tabelului, care apare cu valori multiple pentru o singură instanță a atributului cheie.

4.4 Prima formă normală (1NF)

Prima formă normală (1NF) este o relație în care intersecția fiecărui rând cu fiecare coloană conține **o singură valoare şi numai una**, adică toate atributele relației iau valori unice.

Vom porni analiza de la un format de tabel nenormalizat. Pentru a-l transforma în prima formă normală (**1NF**), vom identifica şi vom elimina ***grupurile repetitive***.

Pentru **eliminarea grupurilor repetitive** dintr-un tabel nenormalizat, există două tratări uzuale, sau **două strategii de lucru**:

1. Se elimină grupurile repetitive ***prin introducerea de date adecvate*** în coloanele goale ale rândurilor cu date repetitive. Cu alte cuvinte, se completează spațiile libere prin dublarea datelor nerepetitive, acolo unde este necesar. Această tratare este denumită ***aplatizarea*** tabelului. Tabelul rezultant, denumit acum relație, conține valori singulare la intersecția fiecărui rând cu fiecare coloană. În cadrul acestei tratări se introduce o **redundanță** în relația rezultantă, redundanță ce va trebui ulterior eliminată în procesul de normalizare.
2. Se nominalizează un atribut (sau un grup de atribute) ca o cheie a tabelului nenormalizat, după care se elimină grupurile repetitive prin ***plasarea datelor care se repetă – împreună cu o copie a atributului cheie inițial, într-o relație separată***. Se identifică apoi cheile primare ale noilor relații. Dacă tabelul nenormalizat conține mai mult de un grup repetitiv, această tratare se aplică în mod repetat, până nu mai rămân

grupuri repetitive. Un set de relații se află în prima formă normală dacă nu conține grupuri repetitive.

Ambele tratări sunt corecte, dar a doua produce relații aflate în cel puțin forma **1NF** cu redundanță mai mică.

Exemplu: o agenție imobiliară

Pornim analiza de la un ***formular clasic***, cu date despre proprietățile închiriate de un anumit client. Pentru a simplifica exemplul, vom presupune că un client închiriază o anumită proprietate o singură dată şi nu închiriază mai multe proprietăți în acelaşi timp.

Detalii Client_Închiriere

Numele clientului: Ion Buflea **Numărul clientului:** 76

Numărul proprietății	Adresa proprietății	Începutul închirierii	Sfârşitul închirierii	Chiria	Numărul proprietarului	Numele proprietarului
4	Rozelor 25 Braşov	01.iul.94	31.aug.96	350	40	Tina Turner
16	Stejăriş 19 Braşov	01.sep.96	01.sep.98	450	93	Vali Vijelie

Detalii Client_Închiriere

Numele clientului: Alina Gospodina **Numărul clientului:** 56

Numărul proprietății	Adresa proprietății	Începutul închirierii	Sfârşitul închirierii	Chiria	Numărul proprietarului	Numele proprietarului
4	Rozelor 25 Braşov	01.sep.92	10.iun.94	350	40	Tina Turner
36	Erorilor 21 Braşov	10.oct.94	01.dec.95	375	93	Vali Vijelie
16	Stejăriş 19 Braşov	01.ian.96	10.aug.96	450	93	Vali Vijelie

Datele referitoare la proprietățile închiriate de cei doi clienți le transferăm din formularele **Detalii Client_Închiriere** în următorul format de tabel:

Tabelul **ClientÎnchiriere**

Nr Client	Nume Client	Nr Proprietate	Adresa Proprietate	Început Închir	Sfârşit Închir	Chiria	Nr Proprietar	Nume Proprietar
76	Ion Buflea	4	Rozelor 25 Braşov	01.iul.94	31.aug.96	350	40	Tina Turner
		16	Stejăriş 19 Braşov	01.sep.96	01.sep.98	450	93	Vali Vijelie
56	Alina Gospodina	4	Rozelor 25 Braşov	01.sep.92	10.iun.94	350	40	Tina Turner

		36	Erorilor 21 Braşov	10.oct.94	01.dec.95	375	93	Vali Vijelie
		16	Stejăriş 19 Braşov	01.ian.96	10.aug.96	450	93	Vali Vijelie

Acesta este un exemplu de **tabel nenormalizat**. Putem identifica atributul cheie ca fiind NrClient. Identificăm grupurile repetitive ca fiind detaliile despre proprietatea închiriată, care se repetă pentru fiecare client care a închiriat proprietatea respectivă. Structura acestui grup repetitiv este:

GrupRepetitiv (NrProprietate, AdresaProprietate, ÎnceputÎnchir, SfârşitÎnchir, Chiria, NrProprietar, NumeProprietar)

Deci există valori multiple la intersecţia dintre anumite rânduri şi coloane. De ex, există două valori ale atributului NrProprietate (4 şi 16), corespunzătoare clientului Ion Buflea.
Pentru a transforma un tabel nenormalizat în forma **1NF**, trebuie să ne asigurăm că există o singură valoare la intersecţia dintre fiecare rând şi fiecare coloană. Aceasta se realizează prin **eliminarea grupului repetitiv**. Vom analiza procesul eliminare prin ambele strategii prezentate:

1. **strategia aplatizării tabelului**

În cazul acestei tratări, se elimină grupul repetitiv prin transformarea unui rând cu valori multiple ale unui atribut în mai multe rânduri cu valori singulare pentru acel atribut (de fapt, prin introducerea datelor adecvate pe fiecare rând). Relaţia rezultată, **ClientÎnchiriere**, se află deja în prima formă normală. Identificăm cheile candidat ale relaţiei **ClientÎnchiriere** ca fiind cheile compuse (NrClient, NrProprietate), (NrClient, ÎnceputÎnchir) şi (NrProprietate, ÎnceputÎnchir). Selectăm drept **cheie primară** atributele (NrClient, NrProprietate) şi pentru claritate, le grupăm în stânga relaţiei.

Relaţia ClientÎnchiriere

Nr Client	Nr Proprietate	Nume Client	Adresa Proprietate	Început Închir	Sfârşit Închir	Chiria	Nr Proprietar	Nume Proprietar
76	4	Ion Buflea	Rozelor 25 Braşov	01.iul.94	31.aug.96	350	40	Tina Turner
76	16	Ion Buflea	Stejăriş 19 Braşov	01.sep.96	01.sep.98	450	93	Vali Vijelie
56	4	Alina Gospodina	Rozelor 25 Braşov	01.sep.92	10.iun.94	350	40	Tina Turner
56	36	Alina Gospodina	Erorilor 21 Braşov	10.oct.94	01.dec.95	375	93	Vali Vijelie
56	16	Alina Gospodina	Stejăriş 19 Braşov	01.ian.96	10.aug.96	450	93	Vali Vijelie

Relaţia **ClientÎnchiriere** este definită după cum urmează:

ClientÎnchiriere (NrClient, NrProprietate, NumeClient, AdresaProprietate, ÎnceputÎnchir, SfârşitÎnchir, Chiria, NrProprietar, NumeProprietar)

Relația este în forma **1NF** deoarece există o singură valoare la intersecția dintre fiecare coloană şi rând. Relația conține date care se repetă de mai multe ori, deci este expusă anomaliilor de reactualizare. Pentru a le elimina va trebui transformată în **2NF**.

2. **strategia creerii unei relații separate**

În cazul celei de-**a doua tratări**, se elimină grupul repetitiv prin plasarea într-o relație separată a datelor respective împreună cu o copie a atributului cheie inițial (NrClient). Apoi vom identifica o chei primară pentru noua relație. Formatele relațiilor **1NF** rezultante sunt:

Client (NrClient, NumeClient)

PropÎnchir_Proprietar (NrClient, NrProprietate, AdresaProprietate, ÎnceputÎnchir, SfârşitÎnchir, Chiria, NrProprietar, NumeProprietar)

Relația Client

NrClient	NumeClient
76	Ion Buflea
56	Alina Gospodina

Relația PropÎnchir_Proprietar

Nr Client	Nr Proprie-tate	Adresa Proprietate	Început Închir	Sfârşit Închir	Chiria	Nr Proprie-tar	Nume Proprietar
76	4	Rozelor 25 Braşov	01.iul.94	31.aug.96	350	40	Tina Turner
76	16	Stejăriş 19 Braşov	01.sep.96	01.sep.98	450	93	Vali Vijelie
56	4	Rozelor 25 Braşov	01.sep.92	10.iun.94	350	40	Tina Turner
56	36	Erorilor 21 Braşov	10.oct.94	01.dec.95	375	93	Vali Vijelie
56	16	Stejăriş 19 Braşov	01.ian.96	10.aug.96	450	93	Vali Vijelie

Ambele relații sunt în forma **1NF**, deoarece acum există o singură valoare la intersecția dintre fiecare coloană şi fiecare rând. Totuşi, relația **PropÎnchir_Proprietar** conține o oarecare redundanță şi prin urmare este vulnerabilă la anomaliile de rectualizare.

4.5 A doua formă normală (2NF)

Se bazează pe conceptul de **dependență funcțională totală**.

Dependența funcțională totală arată că, dacă A şi B sunt atribute ale unei relații R, se spune că B este total dependent funcțional de A, dacă B este dependent funcțional de A dar nu şi de orice submulțime a lui A.

O dependență funcțională **A → B** este **totală** dacă **eliminarea oricărui atribut** din A are ca rezultat **anularea dependenței**.

O dependență funcțională **A → B** este **parțială** dacă **există un atribut care poate fi eliminat** din A şi totuşi **dependența să se mențină**.
De ex, să considerăm următoarea dependență funcțională:

NrPersonal, NumePersonal → NrFilială

Este corect să afirmăm că fiecare valoare din (NrPersonal, NumePersonal) este asociată unei singure valori a atributului NrFilială. Dar asta nu e o dependență funcțională totală, deoarece atributul NrFilială este dependent funcțional şi de un subset al atributelor (NrPersonal, NumePersonal), şi anume NrPersonal.

A doua formă normală se aplică relațiilor cu **chei compuse**, adică relațiilor care au cheia primară compusă din două sau mai multe atribute. O relație a cărei cheie primară este compusă dintr-un singur atribut se află automat în cel puțin forma **2NF**. O relație care nu se află în forma **2NF** este expusă anomaliilor de reactualizare. De ex, să presupunem că trebuie să schimbăm chiria proprietății cu nr 4; va trebui să modificăm toate înregistrările (toate rândurile unde apare proprietatea nr 4), altfel BD devine incoerentă.

A doua formă normală (2NF): o relație este în a doua formă normală dacă îndeplineşte condițiile pentru prima formă normală şi **fiecare atribut care nu este cheie primară este total dependent funcțional de cheia primară**.

Normalizarea relațiilor **1NF** la forma **2NF** presupune **eliminarea dependențelor parțiale**. Asta se face prin eliminarea din relație a atributelor parțial dependente funcțional, şi plasarea lor într-o nouă relație, împreună cu o copie a determinantului acestora.
Exemplu:
Să vedem care sunt dependențele funcționale din relația **ClientÎnchiriere** a cărei cheie primară constă din atributele (NrClient, NrProprietate).

1df NrClient , NrProprietate → ÎnceputÎnchir, SfârşitÎnchir (cheie primară)
2df NrClient → NumeClient (dependență parțială)
3df NrProprietate → AdresaProprietate, Chiria, NrProprietar, NumeProprietar (dependență parțială)
4df NrProprietar → NumeProprietar (dependență tranzitivă)
5df NrClient , ÎnceputÎnchir → NrProprietate, AdresaProprietate, SfârşitÎnchir, Chiria, NrProprietar, NumeProprietar (cheie candidat)
6df NrProprietate, ÎnceputÎnchir → NrClient, NumeClient, SfârşitÎnchir (cheie candidat)

Se testează dacă relația **ClientÎnchiriere** se află în **2NF**, prin identificarea dependențelor parțiale de cheia primară. Observăm că:

- atributul (NumeClient) este parțial dependent de cheia primară – el fiind dependent numai de atributul NrClient (vezi **2df**)
- atributele proprietății (AdresaProprietate, Chiria, NrProprietar, NumeProprietar) sunt parțial dependente de cheia primară – ele fiind dependente numai de atributul NrProprietate (vezi **3df**).

Pentru a transforma relația **ClientÎnchiriere** în **2NF** este necesară crearea de noi relații, astfel încât atributele care nu sunt chei primare să fie eliminate împreună cu o copie a părții din cheia primară de care sunt total dependente funcțional. În acest fel obținem 3 relații noi:

Relația Client

NrClient	NumeClient
76	Ion Buflea
56	Alina Gospodina

Relația Închiriere

NrClient	NrProprietate	ÎnceputÎnchir	SfârşitÎnchir
76	4	01.iul.94	31.aug.96
76	16	01.sep.96	01.sep.98
56	4	01.sep.92	10.iun.94
56	36	10.oct.94	01.dec.95
56	16	01.ian.96	10.aug.96

Relația ProprietateProprietar

NrProprietate	AdresaProprietate	Chiria	NrProprietar	NumeProprietar
4	Rozelor 25 Braşov	350	40	Tina Turner
16	Stejăriş 19 Braşov	450	93	Vali Vijelie
36	Erorilor 21 Braşov	375	93	Vali Vijelie

Client (NrClient, NumeClient)
Închiriere (NrClient, NrProprietate, ÎnceputÎnchir, SfârşitÎnchir)
ProprietateProprietar(NrProprietate, AdresaProprietate, Chiria, NrProprietar, NumeProprietar)

4.6 A treia formă normală (3NF)

Cu toate că relațiile **2NF** conțin mai puțină redundanță decât cele **1NF**, ele tot mai sunt vulnerabile la anomaliile de reactualizare. De exemplu dacă dorim reactualizarea numelui unui proprietar, trebuie ca în relația **ProprietateProprietar** să modificăm toate rândurile în care apare acel nume. Această anomalie de reactualizare este cauzată de dependența tranzitivă. Astfel de dependențe trebuie eliminate prin trecerea la **3NF**.

Pentru a descrie această trecere, să explicăm întâi noțiunea de dependență tranzitivă.

Dependența tranzitivă: dacă A, B, C sunt atribute ale relației R şi există dependențele $A \rightarrow B$ şi $B \rightarrow C$, se spune că **C este dependent tranzitiv de A prin intermediul lui B** (cu condiția ca A să nu fie dependent funcțional de B sau C).

De **exemplu**, să considerăm următoarele dependențe funcționale:

NrPersonal → NrFilială şi NrFilială → AdresăFilială

Atunci dependența funcțională NrPersonal → AdresăFilială are loc prin intermediul atributului NrFilială. Iar condiția cerută, ca NrPersonal să nu fie dependent funcțional de NrFilială şi AdresăFilială este adevărată.

A treia formă normală (3NF): o relație este în a treia formă normală dacă îndeplineşte condițiile pentru prima şi a doua formă normală şi în plus, **nici un atribut (care nu este cheie primară) nu este dependent tranzitiv de cheia primară**.

Normalizarea relațiilor de la **2NF** la **3NF** presupune **eliminarea dependențelor tranzitive**. Asta se face prin eliminarea din relație a atributelor dependente tranzitiv, şi plasarea lor într-o nouă relație, împreună cu o copie a determinantului acestora.
Exemplu:
Să vedem care sunt dependențele funcționale din relațiile **Client**, **Închiriere** şi **ProprietateProprietar**.

Relația Client
2df NrClient → NumeClient

Relația Închiriere
1df NrClient , NrProprietate → ÎnceputÎnchir, SfârşitÎnchir
5df1 NrClient , ÎnceputÎnchir → NrProprietate, SfârşitÎnchir
6df1 NrProprietate, ÎnceputÎnchir → NrClient, SfârşitÎnchir

Relația ProprietateProprietar
3df NrProprietate → AdresaProprietate, Chiria, NrProprietar, NumeProprietar (dependență parțială)
4df NrProprietar → NumeProprietar (dependență tranzitivă)

Toate atributele relațiilor **Client** şi **Închiriere**, care nu sunt chei primare, sunt **dependente funcțional** numai de cheile primare. Aceste relații nu au dependențe tranzitorii, deci se află deja în **3NF**.
În relația **ProprietateProprietar**, toate atributele care nu sunt chei primare, sunt dependente funcțional numai de cheia primară, cu excepția atributului NumeProprietar, care este dependent şi de NrProprietar (vezi **4df**). Acesta este un exemplu de **dependență tranzitivă**.
Pentru a transforma relația **ProprietateProprietar** în **3NF**, trebuie eliminată această dependență tranzitivă prin crearea a 2 relații noi, **Proprietate** şi **Proprietar**, de forma:

Proprietate (NrProprietate, AdresaProprietate, Chiria, NrProprietar)
Proprietar (NrProprietar, NumeProprietar)

Relația Proprietate

NrProprietate	AdresaProprietate	Chiria	NrProprietar
4	Rozelor 25 Braşov	350	40
16	Stejăriş 19 Braşov	450	93
36	Erorilor 21 Braşov	375	93

Relația Proprietar

NrProprietar	NumeProprietar

40	Tina Turner
93	Vali Vijelie

Observați că relația inițială **ClientÎnchiriere** a fost transformată prin procesul de normalizare în 4 relații aflate în forma **3NF**. Acestea au forma:

Client (NrClient, NumeClient)
Închiriere (NrClient, NrProprietate, ÎnceputÎnchir, SfârşitÎnchir)
Proprietate (NrProprietate, AdresaProprietate, Chiria, NrProprietar)
Proprietar (NrProprietar, NumeProprietar)

4.7 Forma normală Boyce-Codd (BCNF)

Relațiile din baza de date trebuie proiectate astfel încât *să nu aibă nici dependențe parțiale, nici dependențe tranzitive*, deoarece acestea duc la apariția anomaliilor de reactualizare. Formele **2NF** şi **3NF** elimină dependențele parțiale şi tranzitive de cheia primară, dar **nu tratează situațiile în care rămân astfel de dependențe față de cheile candidat** ale unei relații.

Forma normală Boyce-Codd se bazează pe **dependențele funcționale care iau în considerație toate cheile candidat dintr-o relație**.

Pentru o relație cu o singură cheie candidat, formele **3NF** şi **BCNF** sunt echivalente.

Forma normală Boyce-Codd: o relație se află în BCNF dacă şi numai dacă **fiecare determinant este o cheie candidat**.

Pentru a testa dacă o relație este în **BCNF**, se identifică toți determinanții şi se verifică dacă sunt chei candidat. Amintim că **un determinant este un atribut sau un grup de atribute, de care alte atribute sunt total dependente funcțional**.

Diferența între formele **3NF** şi **BCNF** constă în faptul că, pentru o dependență funcțională A → B, forma **3NF** permite această dependență în cadrul unei relații dacă B este atribut cheie primară şi A nu este cheie candidat. Prin urmare **BCNF** este o variantă mai strictă a **3NF**, astfel că orice relație aflată în **BCNF** este şi în forma **3NF**. Reciproca nu e neapărat adevărată.

Relațiile Client, Închiriere, Proprietate, Proprietar se află deja în forma **BCNF** întrucât fiecare din ele are un singur determinant care este cheie canidat. Aşa că pentru prezentarea formei **BCNF**, vom imagina alt exemplu.

Exemplu:

RezultateSesiune (NumeProf, Disciplina, NumeStud, Nota)

Cu ipotezele că fiecare profesor predă numai o disciplină, acordă numai o notă fiecărui student, există următoarele dependențe:

Disciplina, NumeStud → Nota
Disciplina, NumeStud → NumeProf

Această relație se poate descompune în:

ProfDisc (NumeProf, Disciplina)
ProfStudNota (NumeProf, NumeStud, Nota)

4.8 A patra formă normală (4NF)

Cu toate că **BCNF** elimină toate anomaliile datorate dependențelor funcționale, mai există un tip de dependență, numită **dependență multivalorică**, ce poate cauza probleme de redundanță a datelor. Posibila existență a dependențelor multivalorice în cadrul unei relații se datorează primei forme normale **1NF** care nu permite unui atribut dintr-un rând să aibă o mulțime de valori. De exemplu, dacă într-o relație există două atribute multivalorice, trebuie să repetăm fiecare valoare a unuia din atribute împreună cu fiecare valoare a celuilalt, dar efectul este apariția redundanței datelor.

Dependența multivalorică: se spune că între atributele A, B, C ale unei relații există o dependență multivalorică, dacă pentru fiecare valoare a lui A există o mulțime de valori ale lui B şi o mulțime de valori ale lui C, dar mulțimile valorilor lui B şi C sunt independente unele de altele.
Notația simbolică: A –>> B

Nu se iau în considerație cazurile în care

- B este o submulțime a lui A
- $A \cup B = R$

pentru că acestea nu specifică o constrângere asupra relației.

Să considerăm **Relația Aproviz**:

Aproviz (NrMagazie, Furnizor, Produs, Preț)

În această relație dependența multivalorică este:
NrMagazie –>> Furnizor
NrMagazie –>> Produs
Cu toate că relația este în **BCNF** (deoarece există o singură cheie candidat), ea este prost structurată datorită **redundanței datelor**, cauzată de prezența **dependenței multivalorice**.

A patra formă normală (4NF): o relație care se află în BCNF şi **nu conține dependențe multivalorice**.

Este o formă mai strictă decât **BCNF** pentru că împiedică relațiile să conțină dependențe multivalorice, prevenind astfel redundanța datelor. Normalizarea de la **BDNF** la **4NF** presupune **eliminarea dependențelor multivalorice prin plasarea atributului (atributelor) într-o nouă relație împreună cu o copie a determinantului**.