Cuprins

Model de date

Un formalism matematic cu notaţii pentru **descrierea structurilor de date** şi a mulţimii **operatorilor** folosiţi pentru manipularea şi validarea datelor.

Model de date relaţional

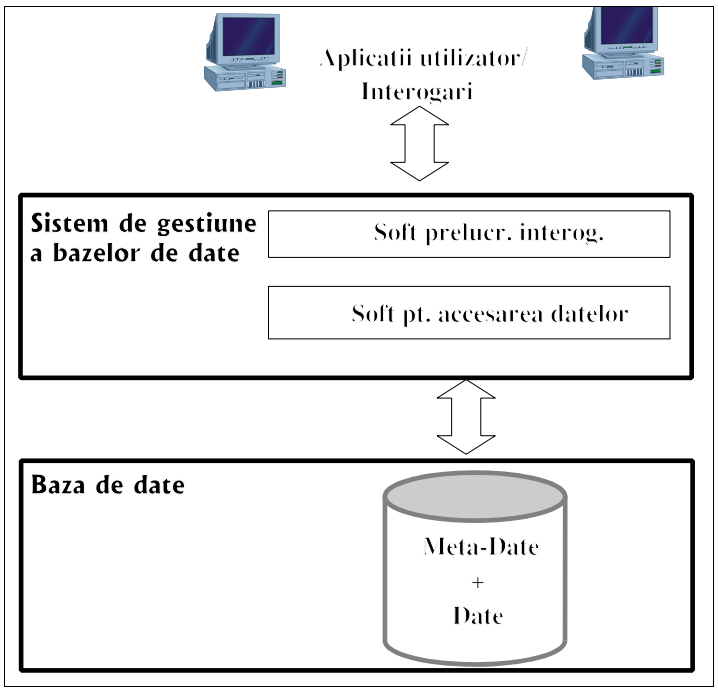
Un model de date introdus de E.F. Codd în 1970. În acest model **datele** sînt **organizate în tabele** care au nume. Mulţimea **numelor coloanelor** se numeşte **schema tabelului**. Datele se pot manipula folosind algebra relaţională. Limbajul SQL este implementarea unei astfel de algebre.

Bază de date, sistem de gestiune a bazelor de date (SGBD)

Baza de date (database): o **colecţie de date** logic interconectate, împreună cu o descriere a datelor, proiectată pentru a satisface cerinţele informaţionale ale unei organizaţii. Fişierele de date sînt conectate în scopul minimizării repetării datelor şi pot fi accesate, simultan, de mai mulţi utilizatori. Între datele specifice organizaţiei baza conţine şi date care descriu datele stocate în ea. Această descriere a datelor se numeşte catalog de sistem (dicţionare de date sau meta-date).

Sistem de gestiune a bazelor de date (database system): un **software** care permite utilizatorilor să definească, creeze, întreţină şi să controleze accesul la o bază de date. SGBD-ul este software-ul prin care utilizatorul interacţionează cu baza de date. Cîteva dintre facilităţile tipice ale SGBD-urilor sînt:

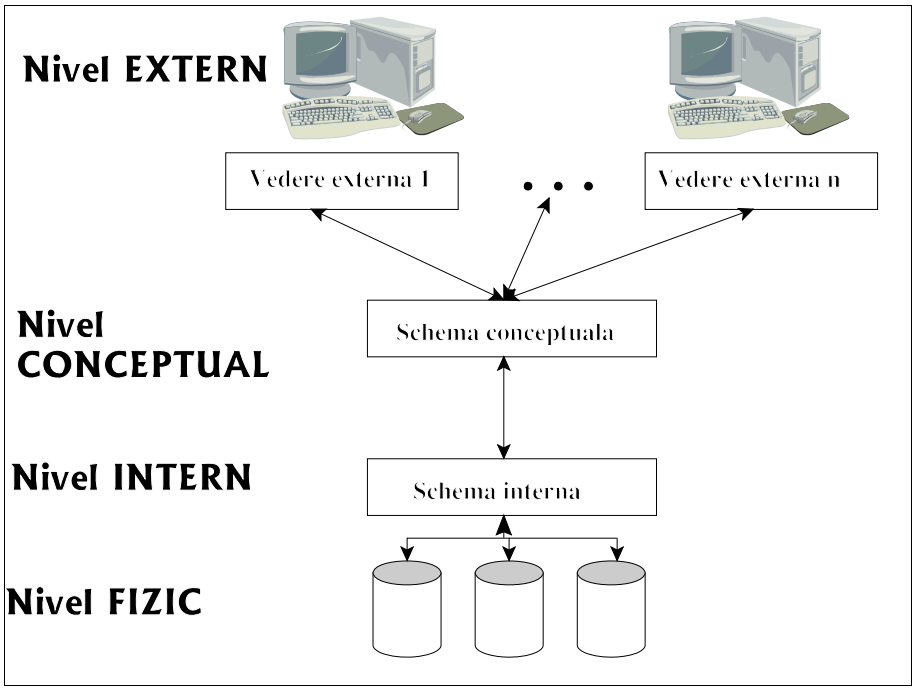
- **definirea datelor** printr-un DDL (Data Definition Language): permite ca utilizatorul să definească baza de date printr-un limbaj. DDL asigură specificarea de către utilizator a tipurilor de date, a structurilor şi a constrîngerilor referitoare la datele ce vor fi stocate în bază;
- **manipularea datelor**, deseori, printr-un DML (Data Manipulation Language): permite ca utilizatorul să insereze, actualizeze, şteargă şi să extragă datele din bază. Facilităţile de extragere a datelor au la bază un limbaj de interogare (query language). Cel mai răspîndit limbaj de interogare este SQL.
- **control al accesului** la date prin: sistem de securitate -



opreşte accesul untilizatorilor neautorizaţi la bază, sistem de integritate - păstrează consistenţa datelor etc.

Arhitectura SGBD-urilor

Datele din bază pot fi descrise la trei nivele: extern, conceptual şi intern. Modul în care utilizatorul percepe datele se numeşte **nivelul extern**. Modul în care SGDB-ul şi sistemul de operare percep datele se numeşte **nivel intern**. **Nivelul conceptual** realizează trecerea, de la nivelul extern, la cel intern asigurînd păstrarea independenţei între aceste nivele. Scopul arhitecturii pe cele trei nivele este acela de separare a vederii fiecărui utilizator individual aspura datelor de modul de reprezentare fizică a acestora în baza de date.



Nivelul extern

Corespunde modului în care utilizatorii percep baza de date. La acest nivel se descriu părţile relevante fiecărui utilizator în parte. Există entităţi care deşi sînt reprezentate în baza de date nu apar la acest nivel deoarece sînt irelevante pentru un anumit utilizator.

Nivelul conceptual

Grupează percepţiile tuturor utilizatorilor bazei de date. Descrie ce date sînt stocate în bază şi relaţiile între date. Conţine structura logică a bazei de date descrisă prin conceptele de:

- entitate, atribute şi relaţii;
- constrîngeri referitoare la date;
- informaţii de securitate şi integritate.

Nivelul conceptual conţine fiecare vedere din nivelul extern, direct sau indirect (sub o formă derivabilă), dar nu conţine detalii referitoare modul de stocare a datelor pe suportul fizic (discuri). Descrierea logică a unei entităţi va cuprinde tipurile de date ale atributelor, dar fără specificarea numărului de octeţi ocupaţi pe disc.

Nivelul intern

Descrie reprezentarea fizică a bazei de date în calculator. Aici este specificat cum anume sînt stocate datele din bază. Structurile de date, organizarea fişierelor şi spaţiul de stocare aparţin acestui nivel. Utilizează funcţii ale sistemului de operare pentru plasarea datelor pe dispozitivele de stocare, pentru

construirea indexurilor, citirea datelor etc.

Sub acest nivel se află cel fizic, acesta asigură specificarea unor metode de lucru ale sistemului de operare cu fişirele bazei şi datele din fişiere sub controlul SGBD-ului.

Sistem de gestiune a bazelor de date relaţionale (SGBDR)

O bază de date care folosește modelul dezvoltat de E. F. Cood. Permite definirea structurilor de date, stocarea, încărcarea și specificarea constrîngerilor de integritate.

Aici datele și relațiile sînt organizate în tabele. Un tabel este o colecție (mulţime) de înregistrări și fiecare înregistrare a tabelului conține aceleași cîmpuri. Înregistrări din tabele diferite pot fi legate dacă au aceeași valoare într-un cîmp particular din fiecare tabel.

Au existat mai multe prototipuri la începutul anilor ‘70, cîteva dintre acestea sînt:

- System R (IBM, San Jose);
- Ingres (UC Berkley);
- Query-by-Example(IBM, TJ Watson)

Primul SGBDR a fost Multics Relational Data Store, vîndut prima oară în 1978.

Există și SGBD-uri care nu folosesc modelul de date relațional ci folosesc alte modele cum sînt: modelul ierahic, modelul rețea sau modelul orientat pe obiecte. Aceste modele sînt foarte complexe utilizînd poantorii pentru legarea înregistrărilor. Procesele de inserare, actualizare și ștergere a înregistrărilor necesită în aceste modele sincronizarea poantorilor, o sarcină pe care aplicația trebuie să o realizeze impecabil. Întreținerea legăturilor prin poantori necesită cunoștințe de programare avansate. În modelul de date relaţional legăturile se fac pe baza valorilor de atribute.

Structuri de date relaţionale

Baza de date relaţională este o mulţime de tabele cu nume. În modelul de date relațional există o singură structură de dată "logică" numită **relaţie**, o structură de dată bidimensională cunoscută sub numele de **tabel**.Tabelul este o mulţine de rînduri. Rîndurile reprezintă relaţii între valori. Coloanele reprezintă atribute. **Atributele** reprezintă date elementare care sînt conectate prin **relaţii**. De exemplu, relația chirie(id_client, nume_client) conține atributele id_client, nume_client. Valorile actuale pentru aceste atribute ale relației se stochează în tuple sau rînduri ale tabelului.

Formal, o relaţie R este submulţime unui produsului cartezian D_1 x ... x D_n, adică R ⊆ D_1 x ... x D_n. Unde D_i reprezentă domeniile din care se aleg valorile atributelor a_i. Pentru fiecare atribut avem, a_i : R → D_i. Se numeşte schema unei relaţii o relaţie care are nume şi se defineşte printr-o mulţime de atribute ce au asociate nume de domenii. Dacă a_1, a_2, ..., a_n sînt atributele cu domeniile D_1, D_2, ..., D_n, atunci mulţimea {a_1:D_1, a_2:D_2, ..., a_n:D_n} se numeşte **schema relaţiei**. O schemă de relaţie ar putea definită prin:

```
Angajat-schema = {
        id_Angajat : string,
        NumeA : string,
        . . .
```

Salar : double,
id_filiala : integer }

Schema bazei de date relaţionale este o mulţime de scheme de relaţii, fiecare dintre ele avînd nume distincte. Dacă R_1, R_2, ..., R_n este mulţimea schemelor de relaţii, atunci schema bazei de date relaţionale sau, mai pe scurt, schema relaţională R este, $R = \{R_1, R_2, ..., R_n\}$.

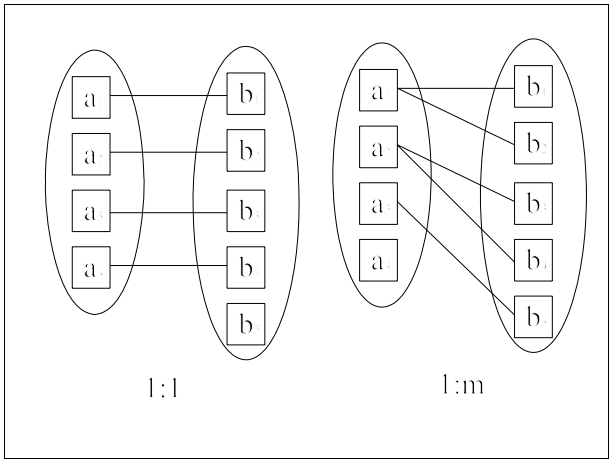
Proprietăţile relaţiilor

- numele relaţiei trebuie să fie distinct de celelelte nume din schema relaţională;
- fiecare celulă a relaţie poate conţine exact o valoare atomică (o singură valoarea);
- toate atributele trebuie să aibă nume distincte;
- valorile atributelor trebuie să fie toate din acelaşi domeniu;
- fiecare tupă trebuie să fie distinctă (nu există dubluri);

O relaţie care satisface simultan aceste condiţii se zice că este adusă la **prima formă normală**.

Chei

Deoarece nu avem dubluri de tuple într-o relaţie trebuie să existe posibilitatea identificării unei tuple din relaţie pe baza unui atribut sau a unui grup de atribute. Se numeşte cheie primară (CP) un atribut sau un grup de atribute care indentifică unic un rînd (o tuplă) din tabel (relaţie). Un tabel poate avea o sigură cheie primară. Pentru că valorile CP se folosesc pentru identificare ele nu pot fi vide (nule). Convenţia de notaţii pentru atribute CP este sublinierea, în continuare eu însă voi folosi *înclinarea* în acest scop. Dacă o CP este formată din mai multe atribute, fiecare atribut se va scrie înclinat. Legarea (asocierea - join, în enegleză) unei relaţii la o alta se face, tipic, printr-un atribut comun celor două relaţii. Atributul comun este de obicei CP într-un tabel şi cheie străină (CS) în celălalt. Prin definiţie, o CS este formată dintr-un atribut (sau un grup de atribute) care au aceleaşi valori cu cele ale unei CP. Există un grup de **reguli** numite "**de integritate**" care impun constrîngeri asupra datelor introdue în baza în vederea păstrării corectitudinii reprezentării informaţiilor în bază. **Integritatea entităţilor** constrînge ca toate CP să aibă valori completate (nu pot există valori de null, adică de nimic). **Integritatea referenţială** dictează valorile pe care o CS dintr-o asociere le poate lua în raport cu valorile CP din cealaltă relaţie. Mai precis valorile din CS trebuie să refere o tuplă existentă a unei alte relaţii. Asocierile între două relaţii, prin prisma constrîngerilor ce trebuie respectate între CP şi CS pot fi de tipul : **1:1** (unu la unu - valoarea unui atribut A este asociată cel mult cu o valoarea a unui atribut B şi invers), **1:m** (unu la mulţi - valoarea unui atribut A este asociat cu orice nr. de valori ale atributului din B, dar valorea atributului din B poate fi asociat cel mult cu o singură valorea a atributului A) şi **n:m** (mulţi la mulţi valorile atributului A sînt asociate cu orice număr ale valorilor atributului B şi invers).



Cîmpurile care sînt folosite pentru căutare datelor pot fi indexate în vederea creşterii vitezei de regăsire a datelor.

Proiectarea unei baze de date relaţionale se face pe baza regulilor de normalizare care dictează apartenenţa atributelor la relaţii. Aceste reguli vor fi descrise în detaliu mai tîrziu în acest curs. Folosirea modelului de date relaţional normalizat asigură protecţia contra anomaliilor de inserare, ştegere şi actualizare care apar ca urmare a definirii unor relaţii incorecte.

Algebra relaţionlă

Modelul de date relaţional defineşte operaţii care pot fi aplicate unei relaţii sau asupra unui grup de relaţii. Operatorii relaţionali sînt unari şi binari şi au ca rezultat o altă relaţie. Oarecum, aceşti operatori sînt similari cu cei din teoria mulţimilor. Tabelul care urmează prezintă 7 operatori folosiţi pentru manipularea structurilor relaţionale. Operatorii de tipul binar au ca şi operanzi două relaţii în timp ce cei unari necesită o singură relaţie pe postul de operand.

Operaţie	Tip	Relaţie rezultată
reuniune: $R \sqcup S$	binar	rîndurile celor 2 relaţii se combină, fiind eliminate dublurile
intersecţie: $R \sqcap S$	binar	rîndurile comune celor două relaţii
diferenţă: R - S	binar	rîndurile care există în prima relaţie dar nu şi în a doua
proiecţie: $\prod_{a1,a2,\ldots,an}(R)$	unar	rîndurile care conţin o parte din coloanele relaţiei sursă
selecţie: $\sigma_{condiţie}(R)$	unar	rîndurile din relaţia sursă care corespund condiţiei impuse
produs cartezian: R x S	binar	concatenarea fiecărui rînd dintr-o relaţie cu toate rîndurile din cealată
legare (asociere): $R \bowtie_F S$	binar	concatenarea rîndurilor dintr-o relaţie cu cele legate din cealată relaţie (numai cele care satisfac condiţia de legare). F este un predicat (o funcţie care ia valori de adevăr şi are argumente) de forma R.ai θ S.Bi, unde θ poate fi unul dintre operatorii de comparaţie $<$, $\leq$, $>$, $\geq$, $=$ şi $\neq$.

Index

O secvenţă de perechi **(cheie, poantor)** unde fiecare poantor poantează o înregistrare din baza de date care conţine valoarea cheii stocată într-un cîmp particular. Index-ul este sortat pe baza valorilor din cheie pentru a permite căutarea rapidă a unei valori particulare de cheie (de exemplu, folosind algoritmul de căutarea binară). Index-ul este "inversat" în sensul că valoarea cheii este folosită pentru găsirea înregistrării în loc sa fie invers. Pentru bazele de date în care înregistrările pot fi sortate pe baza a mai multor cîmpuri se pot crea indexuri multiple care sînt sortate pe baza valorilor acelor chei. Un index poate conţine goluri pentru a permite adăugarea de noi intrări pentru sortarea corectă fără a realiza deplasarea intrărilor care nu sînt în poziţia corectă.

Accesul la date neprocedural (SQL)

Un SGBDR este caracterizat prin posibilitatea prelucrării unor mulţimi de date. Alte SGBD-uri realizează prelucrările la nivel de înregistrare. O metodă de comunicaţie cu SGBDR-ul este limbajul SQL (**S**tructured **Q**uery **L**anguage) care este un limbaj neprocedural proiectat special pentru accesul la date normalizate dintr-o BDR. Diferenţa esenţială dintre SQL şi alte limbaje constă în faptul că în SQL instrucţiunile specifică **ce** operaţii se vor face cu datele şi nu modul **cum** se realizează acestea.

Normalizare

Normalizarea este o tehnică pentru producerea unei mulțimi de relații cu proprietăți convenabile, pe baza cerințelor:

- dezvoltate de E.F. Codd (1972);
- deseori se realizează pe baza unor teste aplicate relațiilor pentru a determina dacă satisfac sau violează cerințele unei anumite forme normale.

Patru forme normale sînt utilizate cel mai des în practică:

- Prima (First Normal Form - 1NF);
- A Doua (Second Normal Form - 2NF);
- A Treia (Third Normal Form - 3NF);
- Boyce-Codd (Boyce Codd Normal Form - BCNF).

Se bazează pe dependența funcțională între atributele (cîmpurile) unei relații.

Scopul major al proiectării bazelor de date relaționale este gruparea atributelor în relații în vederea minimizării redundanței datelor şi a scăderii dimensinii fişierelor folosite pentru stocarea relațiilor de bază.

Problemele asociate cu redundanţa datelor sînt ilustrate prin compararea relaţiilor Angajati şi Filiale cu relaţia Angajati_Filiale.

Relația **Angajati**

id_Angajat	NumeA	AdresaA	Post	Salar	id_Filiala
SI21	Ion Alexandru	Horea 18	Manager	13000	F5
SG37	Ana Dumiterscu	Baita 14	Director adjunct	5000	F3
SG14	Vasile David	Closca 12	Jurist	7000	F3
SA9	Maria Costin	Ciortea 22	Asistent	9000	F7
SG4	Susana Ilie	Motilor 14	Manger	11000	F3

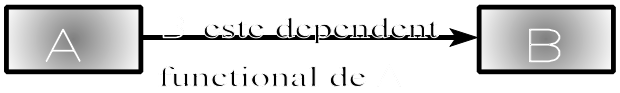
Relația **Filiala**

id_Filiala	AdresaF	Telefon
F5	Mica 12, Cluj	064-193333
F7	Principala 12, Bistrita	063-123456
F3	Maiorescu 11, Brasov	058-122233

Relaţia **Angajati_Filiale**

id_Angajat	NumeA	AdresaA	Post	Salar	id_Filiala	AdresaF	Telefon
SI21	Ion Alexandru	Horea 18	Manager	13000	F5	Mica 12, Cluj	064-193333
SG37	Ana Dumiterscu	Baita 14	Director adjunct	5000	F3	Maiorescu 11, Brasov	058-122233
SG14	Vasile David	Closca 12	Jurist	7000	F3	Maiorescu 11, Brasov	058-122233
SA9	Maria Costin	Ciortea 22	Asistent	9000	F7	Principala 12, Bistrita	063-123456
SG4	Susana Ilie	Motilor 14	Manger	11000	F3	Maiorescu 11, Brasov	058-122233

Dependenţa funcţională



Un atribut B este DEPENDENT FUNCŢIONAL de un alt atribut A, dacă o valoare a lui A determină o singură valoare a lui B în orice moment.

Notaţii

- A ➜ B
- ID_CLIENT ➜ NUME_CLIENT
- NUMAR_COMANDA ➜ DATA_COMANDA
  - NUMAR_COMANDA - variabilă independentă, uneori numită şi DETERMINANT
  - DATA_COMANDA - variabilă dependentă

Tipuri de dependenţe

Dependenţa totală (Total dependence)

- Atributul A determină atributul B ŞI atributul B determină atributul A.
- COD_PERSOANA ⟷ COD_NUMERIC_PERSONAL

Dependenţa completă (Full dependence)

- Apare cînd un atribut este întotdeauna dependent de **cel puţin două** alte atribute.
- NUMAR_COMANDA, NUME_DETALIU ➜ CANTITATE_COMANDATA
- Lipsa dependenţei complete în cazul cheilor multiple = **dependenţă parţială**.

Dependenţa tranzitivă (Transitive dependency)

- Apare atunci cînd Y depinde de X şi Z depinde de Y, astfel Z depinde şi de X.

- $X \rightarrow Y \rightarrow Z$
- NUMAR_FACTURA ➜ NUMAR_CLIENT ➜ NUME_CLIENT

Schema bazei de date ar fi bine ca:
- să aibă o semnificatie clară;
- să minimizeze redundanţa informaţiilor;
- să evite utilizarea valorilor nule (neiniţializate);
- să fie făcută numai pe baza CP (**C**heie **P**rimară) sau CS(**C**heie **S**trăină).

Prima Formă Normală

Converteşte un tabel (nenormalizat - **U**n**N**ormalized **F**orm = UNF), progresiv, în mai multe tabele cu grad şi cardinalitate mai mică pînă cînd un nivel optim al descompunerii a fost atins - redundanţa este mică sau inexistentă.

Rezultate pozitive:
- spaţiul folosit pentru stocarea datelor este mai mic;
- tabelele pot fi actualizate mai eficient;
- descrierea bazei de date va fi mai clară.

UNF: **client_chirie**(id_client, **nume_client**(id_casa, adresa_casa, ...))

Prima Formă Normală (extras):
- un tabel este în Prima Formă Normală (1NF) dacă:
  - este un tabel valid (în particular nu există grupuri care se repetă);
  - o *cheie unică* a fost identificată pentru fiecare rînd;
  - toate atributele sînt dependente funcţional de toată cheia sau de o parte cheii.

1NF: **client**(*id_client*, *nume_client*)
chirie(*id_client*, *id_casa*, adresa_casa, ...)

Exemplu de transformare de la UNF la 1NF

UNF: **client_chirie**(id_client, **nume_client**(id_casa, adresa_casa, ...))

Relaţia **client_chirie**

id_client	clientN	id_casa	AdresaC	ChirieInceput	ChirieSfirsit	Chirie	id_proprietar	NumeP
CR75	Ion Marin	PG4 PG16	Baita 10, Cluj Bucuresti 14, Braila	1-iulie-94 1-mai-97	31-aprilie-97 1-mai-99	100 121	CO40 CO93	Antom Pan Turc Daniela
CR56	Vasile Dan	PG4 PG36 PG16	Baita 10, Cluj Crinului 12, Dej Bucuresti 14, Braila	1-august-92 10-iunie-94 1-ianuarie-96	11-septembrie-93 31-decembrie-95 10-august-99	100 120 130	CO40 CO93 CO93	Antom Pan Turc Daniela Turc Daniela

1NF: **client**(*id_client*, nume_client)

Relaţia **client**

id_client	clientN
CR75	Ion Marin
CR56	Vasile Dan

1NF: **chirie**(*id_client, id_casa,* adresa_casa, ...)

Relaţia **casa_chirie_prop**

id_client	id_casa	AdresaC	ChirieInceput	ChirieSfirsit	Chirie	id_proprietar	NumeP
CR75	PG4	Baita 10, Cluj	1-iulie-94	31-aprilie-97	100	CO40	Antom Pan
CR75	PG16	Bucuresti 14, Braila	1-mai-97	1-mai-99	121	CO93	Turc Daniela
CR56	PG4	Baita 10, Cluj	1-august-92	11-septembrie-93	100	CO40	Antom Pan
CR56	PG36	Crinului 12, Dej	10-iunie-94	31-decembrie-95	120	CO93	Turc Daniela

Transformarea UNF la 1NF

Identificaţi grupul sau grupurile care se repetă, dacă acestea există, în relaţia ne-normalizată.

Treceţi de la UNF la 1NF prin extragerea grupurilor care se repetă împreună cu CP a relaţiei principale.

Proprietate importantă a descompunerii prin normalizare:

- legăturile fără pierdere (lossless-join) permit regăsirea oricărei instanţe a relaţiei originale din instanţele corespunzătoare relaţiilor mai mici obţinute prin descompunere;
- din acest motiv trebuie extrasă CP a relaţiei principale.

Determinaţi CP a noilor relaţii create:

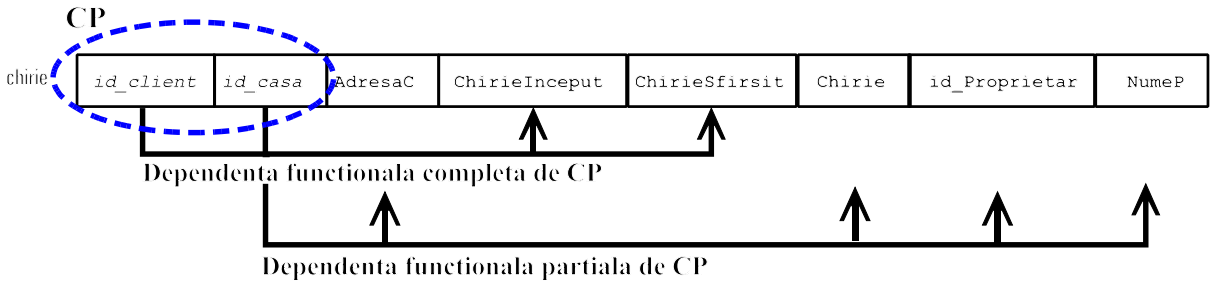
- grupurile extrase care se repetă vor avea de obicei o CP compusă incluzînd CP a relaţiei principale;
- nu întotdeauna CP a relaţiei principale va fi o simplă CS:
  - **asigurat**(id_firma, **nume_firma**(id_asigurat, nume_asigurat, ...)
    - **firma**(*id_firma*, nume_firma)
    - **asigurat**(*id_asigurat*, id_firma, nume_asigurat, ...)

De la 1NF la 2NF

O relaţie este în 2NF dacă:

- toate atributele ne-cheie sînt dependente funcţional de întrega cheie - nu există dependenţe parţile

Trecerea de la 1NF la 2NF se face prin extragerea dependenţelor parţiale.



1NF: **client**(*id_client*, nume_client) - este deja în 2NF petru că avem un singur atribut în CP, astfel nu pot exista dependenţe parţiale

1NF: **chirie**(*id_client, id_casa,* adresa_casa, ...)
devine:

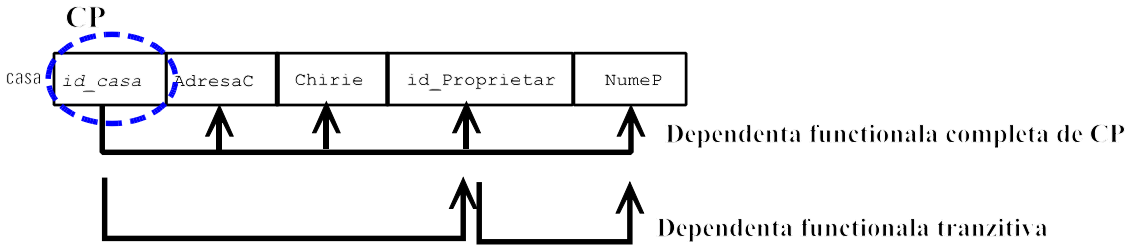
- 2NF: **chirie**(*id_client, id_casa*, ChirieInceput, ChirieSfirsit)
- 2NF: **casa**(*id_casa*, adresa_casa, chirie, id_proprietar, nume_proprietar)

De la 2NF la 3NF

O relaţie este în 3NF dacă:

- toate dependenţele tranzitive au fost extrase - se verifică pentru atributele ne-cheie dependenţa de alte atribute ne-cheie.

Trecerea de la 2NF la 3NF se face prin extragerea dependenţelor tranzitive.



2NF: **chirie**(*id_client, id_casa,* ChirieInceput, ChirieSfirsit) - este deja în 3NF, nu există dependenţe tranzitive.

2NF: **casa**(*id_casa*, adresa_casa, chirie, id_proprietar, nume_proprietar)
devine:

- 3NF: **casa**(*id_casa,* adresa_casa, chirie, id_proprietar)
- 3NF: **proprietar**(*id_proprietar,* nume_proprietar)

Forma normală BOYCE CODD

Definiția originală a lui Cood pentru 3NF are probleme cu relațiile în care sînt chei candidate multiple.

Fie relația **student-profesor** descrisă prin:

- un student poate participa la mai multe cursuri, dar numai unul pe materie;
- fiecare curs poate fi predat de mai mulţi profesori;
- fiecare curs este ţinut de un anumit profesor;
- fiecare profesor predă o singură materie;
- fiecare profesor poate preda mai multor sudenți.
  - **student_profesor**(`student, materie, id_profesor`)

Depedențe funcționale:

- `student, materie` ➜ `id_profesor`
- `id_profesor` ➜ `materie`
  - relația este în 3NF - nu există dependențe între ne-chei,
- dar **există anomalii**: de exemplu, dacă se șterge ultima linie pentru că studenții au promovat se pierd şi informațiile legate de profesor/materie.

Forma Normală BOYCE CODD: o relație R este în BCNF dacă ori de cîte ori o dependență funcțională $X \rightarrow A$ apare în R, atunci X este o cheie a lui R sau **fiecare determinant trebuie să fie cheie candidată**, în exemplul prezentat **id_profesor** era determinat fără să fie din CP.

Trecerea de la 3NF la BCNF se face divizînd relaţia în două relaţii şi extragînd într-o nouă relaţie cheia ne-candidat determinantă:

- **student_profesor**(*`student, id_profesor`*)
- **materie_profesor**(*`id_profesor,`* `materie`)

În practică majoritatea 3NF sînt şi BCNF.

4NF

Dependenţe multivalorice:

- fie 3 atribute A, B şi C;
- pentru fiecare valoare a lui A sînt mulţimi bine definite ale lui B;
- pentru fiecare valoare a lui A există mulţimi bine definite ale lui C, independent de B.
  - **oferta**(`curs,profesor,autor`)

Trecerea de la BCNF la 4NF implică etragerea dependenţelor multivalorice din relaţie prin plasarea atributului sau a atributelor într-o relaţie nouă împreună cu o copie a determinantului sau a determinanţilor.

curs	*profesor*	*autor*
Programare în C	Samuilă	Kernigham & Ritchie
Programare în C	Ion	Kernigham & Ritchie
Programare în C	Vasile	Kernigham & Ritchie
Programare în C	Samuilă	Peter Norton
Programare în C	Ion	Peter Norton
Programare în C	Vasile	Peter Norton

Descompunerea în două relaţii:

- **profesor**(*curs, profesor*)
- **autor**(*curs, autor*)

Pentru majoritatea bazelor de date practice 3NF (eventual BCNF) este suficientă.

Metodologia proiectării bazelor de date relaţionale

Reprezentaţi toate vederile utilizatorilor (formulare, rapoarte) ca şi o colecţie de relaţii UNF.

Normalizaţi aceste relaţii - un singur formular/raport la un moment dat:

- UNF ➜ 1NF
- 1NF ➜ 2NF
- 2NF ➜ 3NF
- 3NF ➜ BCNF/4NF, dacă este cazul

Reprezentaţi cheile tot timpul procesului de normalizare

Determinaţi limitările speciale - restricţiile de domeniu

Unificaţi toate rezultatele parţiale pentru a crea un unic proiect final - sinteza atributelor:

- colectaţi toate relaţiile 3NF;
- legaţi relaţiile care au CP identice.

Un exemplu de lucru:

COMANDA CLIENT			
COMANDA: 12345 DATA:28/07/1999 CLIENT: 111 NUME CLIENT: Firma MetaHot ADRESA CLIENT: str. Tutunului 19 ORAS-COD: Cluj. 3400			
PRODUS	**DESCRIERE**	**CANTITATE COMANDATA**	**PRET pe UNITATE**
P128	placa video	10	30
D123	hard disk	3	100

UNF:

- **comanda**(comanda, client, nume_client, adresa_client, oras_cod_client, data, **comanda_detalii**(produs, descriere_produs, cantitate_comandata, pret_unitar))

1NF:

- **comanda**(*comanda,* client, nume_client, adresa_client, oras_cod_client, data_comanda)
- **comanda_detalii**(*comanda, produs,* descriere_produs, cantitate_comandata, pret_unitar)

2NF:

- **comanda**(*comanda,* client, nume_client, adresa_client, oras_cod_client, data_comanda) - este deja în 2NF întrucît are un singur atribut în CP
- **comanda_detalii**(*comanda, produs,* cantitate_comandata)
- **produs**(*produs,* descriere_produs, pret_unitar)

3NF:

- **comanda**(*comanda*, client, data_comanda)
- **client**(*client,* nume_client, adresa_client, oras_cod_client)
- **comanda_detalii**(*comanda,* produs, cantitate_comandata)
- **produs**(*produs,* descriere_produs, pret_unitar)

Specificarea caracteristicilor şi a limitărilor datelor

Vizualizarea relaţiilor nu este suficientă datorită lipsei descrierii a numeroase constrîngeri. Aceste informaţii sînt esenţiale pentru o proiectare bună: chei străine, valori nule, restricţii de integritate, etc.

Marcarea cheilor alternative (**A**lternative **K**ey - **AK**) - atribute care ar fi putut fi alese CP dar nu au fost.

Selectarea cheilor secundare (**S**econday **K**ey - **SK**) - atribute folosite pentru manipularea eficientă (de exemplu, indexuri).

Cheile străine (**F**oreign **K**ey - **FK**) - atribute pentru crearea relaţiilor:
- sînt permise valorile nule pentru toate atributele?
- tipuri de actualizare (restricţionată, propagată etc.)
- ştergerea (restricţionată, propagată, etc.)

angajat(*id_angajat*, nume, adresa*, cod_num_pers, id_departament, ...)
AK cod_num_pers
SK nume
FK id_departament ➜ departament: restricţii de ştergere şi de actualizare

Listaţi tabelul **angajaţi** cu toate atributele lui.

Numai cîmpul adresa poate accepta valori nule (s-a marcat prin *).

cod_num_pers poate fi o posibilă cheie (**AK**)

nume este cheie secundară (**SK**)

id_departament este cheie străină (**FK**):
- nu se poate şterge un departament dacă mai există un angajat;
- id_departament în departament poate fi modificat, dar modificarea se propagă.

Tehnici de proiectare - reluare

TOP DOWN (de sus în jos):
- diagrama entităţilor şi a relaţiilor dintre ele;
- traduceţi această diagramă în schema bazei de date;

BOTTOM UP (de jos în sus):
- colectaţi vederile utilizatorului (rapoarte, formulare, ecrane etc.);
- reprezentaţi-le ca şi relaţii UNF;
- normalizaţi relaţiile cel puţin pînă la 3NF;
- integraţi mulţimea relaţiilor normalizate pe baza combinării relaţiilor care au aceeaşi chei - sinteza atributelor;
- aplicaţii restricţiile necesare schemei obţinute.