



Sisteme de baze de date

Curs 3 – Proiectarea bazelor de date relaționale (cont.)
Normalizarea bazelor de date

Sorina Preduț
sorina.predut@unibuc.ro
Universitatea din București



Cuprins

1. Crearea schemei conceptuale
 - a. Modelul entitate-legătură (entitate-relație)
2. Crearea design-ului logic al bazei de date
3. Crearea design-ului fizic al bazei de date



Proiectarea bazelor de date - recap.

1. **Crearea schemei conceptuale.** Este un design de nivel înalt care descrie datele și relațiile necesare pentru execuția operațiilor necesare, fiind independent de orice model de baze de date. **Se construiește din diagrama E/R prin adăugarea tabelelor asociative și prin marcarea cheilor externe.**
2. **Crearea design-ului logic al bazei de date.** La sfârșitul acestei etape vom avea un număr de tabele care vor permite stocarea și manipularea corectă a tuturor datelor necesare sistemului.
3. **Crearea design-ului fizic al bazei de date.** În această etapă designul logic este transformat într-o structură fizică eficientă.



Modelul entitate-legătură și modelul relațional

- Modelul entitate-legătură poate fi transformat în mod natural într-o bază de date relațională. Fără a intra deocamdată în amănuntele acestei transformări, enunțăm în continuare principalele idei ale acestei transformări:
 - O entitate devine un tabel.
 - Un atribut al unei entități devine o coloană a tabelului respectiv.
 - O relație va fi reprezentată fie printr-un tabel special, fie printr-o cheie străină într-unul dintre cele două tabele entitate, care face referire la cheia primară a celuilalt tabel entitate.

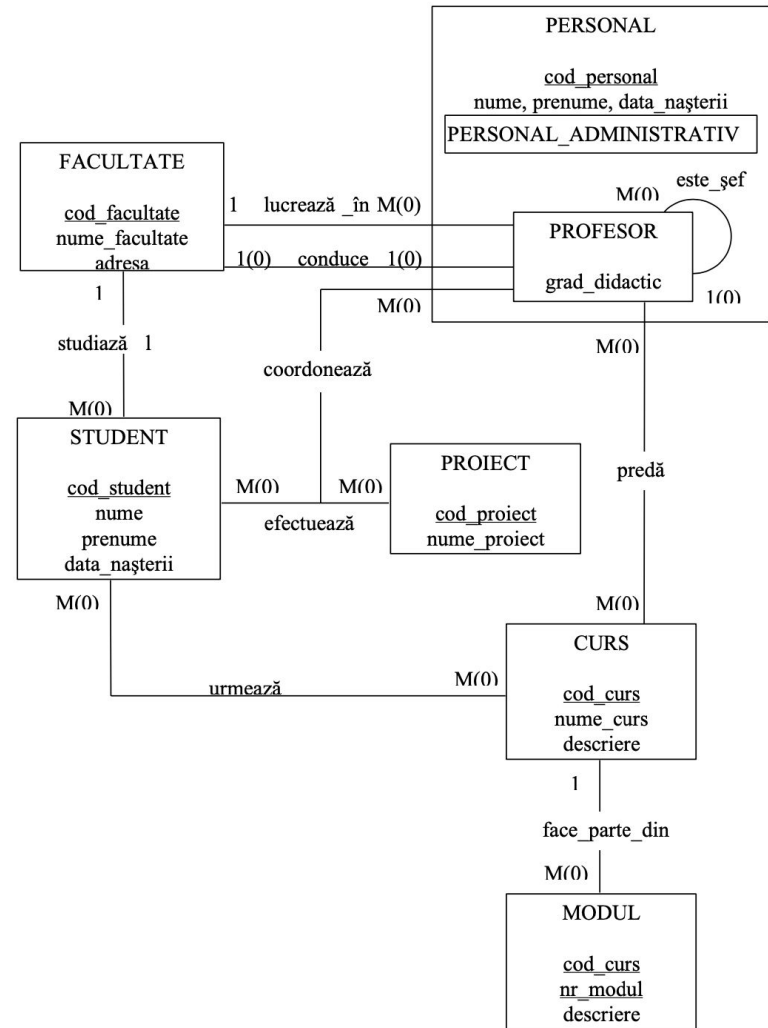


Etapele obținerii modelului entitate-legătură

- Pentru realizarea modelului entitate-legătură al sistemului analizat sunt parcurse următoarele etape:
 - Identificarea **entităților** sistemului.,
 - Identificarea **relațiilor** sistemului și stabilirea **cardinalității** acestora.
 - Identificarea **atributelor** entităților și relațiilor sistemului.
 - Stabilirea **cheilor primare** ale entităților.
 - Trasarea **diagramei entitate-legătură**.

Exemplu de ERD

Diagrama entitate-legătură a sistemului prezentat ca exemplu în acest curs, incluzând entitățile și relațiile menționate mai înainte: (mergi la [diag. conceptuală](#))





Etapele obținerii modelului entitate-legătură

- Trebuie remarcat că aceeași realitate poate fi percepută diferit de către analiști diferiți, așa că **este posibilă obținerea de modele diferite pentru același sistem**, după cum și un sistem poate să se modifice în timp, ceea ce va atrage la rândul său modificarea modelului asociat.
- În sfârșit, există și alte moduri grafice de reprezentare a diagramei entitate-legătură, cum ar fi aceea din fișierul `diagrameER.pdf`, în acest curs prezentându-se doar una dintre notațiile existente.



2. Crearea design-ului logic al bazei de date

- Pentru realizarea design-ului logic al bazei de date, **schema conceptuală este transformată într-un design al BD care va funcționa într-un SGBD specific.** Designul logic al bazei de date este o rafinare a modelului inițial furnizat de schema conceptuală.
Aceasta nu înseamnă că modelul conceptual nu este corect, dar trebuie stabilite detalii suplimentare necesare dezvoltării proiectului.



Transformarea modelului entitate legătură în modelul relațional

- Pentru obținerea design-ului logic al unei BD relaționale se pornește de la schema conceptuală, mai precis de la modelul entitate-legătură, și se încearcă **reprezentarea entităților și a legăturilor sub formă de tabele relaționale.**
- Regulile de conversie ale entităților, legăturilor și atributelor sunt următoarele:



Transformarea entităților

- Regula generală este că entitățile devin tabele, distingându-se următoarele subcazuri:
 - **Entitățile independente devin tabele independente**, adică tabele a căror CP nu conține chei străine. De exemplu, entitatea STUDENT va deveni un tabel a cărui cheie primară este „cod_student”.
 - **Entitățile dependente devin tabele dependente** (tabele detaliu) adică tabele a căror CP conține cheia străină ce face referință la CP a entității de care depinde entitatea în cauză.
De exemplu, CP a entității MODUL va fi formată din „cod_curs”, care reprezintă o cheie străină pentru entitatea CURS, plus „nr_modul”.



Transformarea entităților

- **Subentitățile** devin **subtabele** adică tabele a căror CP este cheia străină pentru tabelul superentitate. De exemplu, CP a tabelului PROFESOR este „cod_personal”, care este o cheie străină ce face referință la CP „cod_personal” din tabelul PERSONAL.
- Uneori, se preferă construirea unor supertabele, formate atât din attributele superentității - cele comune tuturor subentităților - cât și din attributele specifice fiecărei subentități.
Avantajul unor astfel de supertabele este simplificarea programelor de manipulare a datelor.

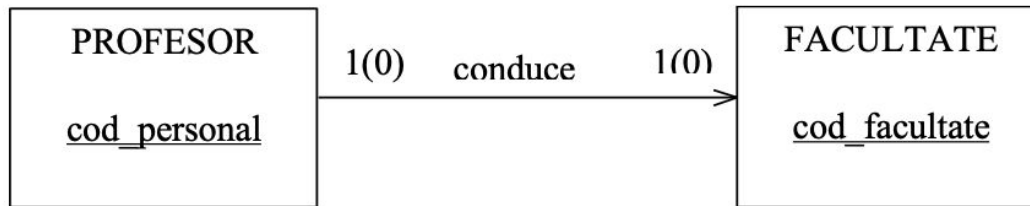


Transformarea entităților

- Pe de altă parte însă, ele creează **probleme suplimentare privind integritatea datelor**, de exemplu dacă vom avea un singur tabel pentru tot personalul din facultate, atunci când se inserează în tabel un rând corespunzător unui profesor **numai attributele specifice pot avea valori diferite de Null**.
În plus, subtabelele obținute din descompunerea unui astfel de supertabel sunt mai stabile, mai flexibile, ocupă spațiu fizic mai mic și conțin mai puține valori Null.

Transformarea relațiilor

- Relațiile 1:1 devin chei străine, cheia străină fiind plasată în tabelul cu mai puține linii. De exemplu, relația “conduce” dintre PROFESOR și FACULTATE se realizează prin inserarea unei chei străine în tabelul FACULTATE care face referință la CP a tabelului PROFESOR, ca în figura:





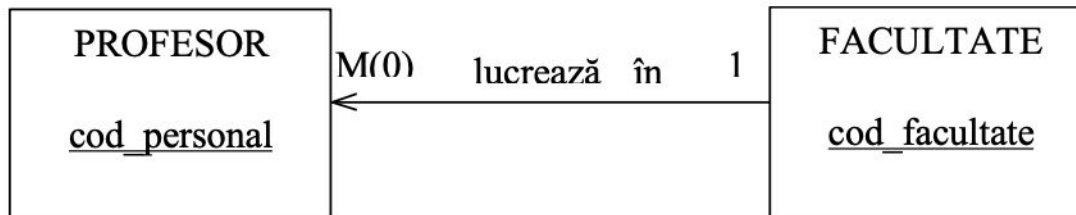
Transformarea relațiilor

- Notă: Plasamentul cheii străine va fi indicat printr-o săgeată (\rightarrow), iar când cheia străină va fi conținută în CP, atunci vârful săgeții va fi umplut (\rightarrow).
- Deci într-o relație 1:1 **poziția cheii străine depinde de cardinalitatea minimă a relației**. Dacă aceasta este tot 1:1, atunci cheia străină poate fi plasată în oricare din cele două tabele. Dacă însă această cardinalitate minimă este 1:0, atunci cheia străină este plasată în tabelul a cărui cardinalitate minimă în relație este 0.

Transformarea relațiilor

- Relațiile N:1 devin chei străine plasate în tabelul care se află de partea „mulți” a relației.

De exemplu relația „lucrează în” va fi realizată prin inserarea unei chei străine în tabelul PROFESOR care va face referință la CP a tabelului FACULTATE, ca în figura:





Transformarea relațiilor

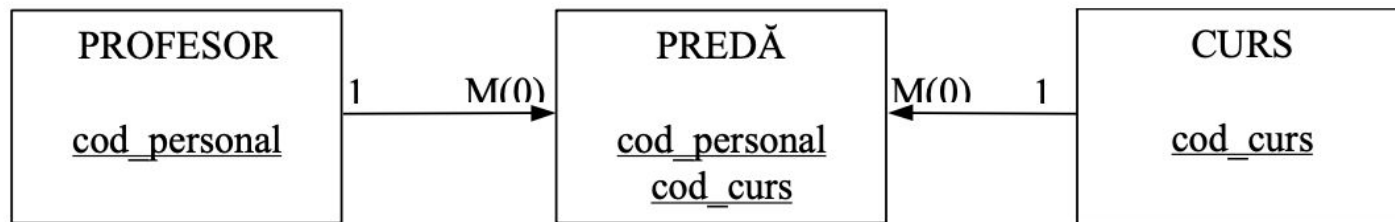
- Și în cazul relațiilor N:1 se disting 2 cazuri în funcție de cardinalitatea minimă a relației.
Dacă aceasta este 0:1, atunci cheia străină respectivă nu poate avea valoarea Null, iar în cazul entităților dependente ea va face chiar parte din CP a tabelului.
Dacă însă cardinalitatea minimă a relației este 0:0 atunci cheia străină poate avea valoarea Null și nu poate face parte din CP.



Transformarea relațiilor

- O relație mulți-la-mulți se transformă într-un tabel special, numit tabel asociativ, care are 2 chei străine pentru cele 2 tabele asociate; CP a tabelului asociativ este compusă din aceste 2 chei străine plus eventual alte coloane adiționale. În acest caz se spune că o relație mulți-la-mulți se sparge în 2 relații mulți-la-unu, tabelul asociativ fiind în relație de mulți-la-unu cu fiecare dintre cele 2 tabele asociate.
De exemplu, relația „predă” dintre PROFESOR și CURS se realizează printr-un tabel a cărui CP este combinația cheilor străine ale acestor 2 entități, ca în figura:

Tabel asociativ (relație M:M)

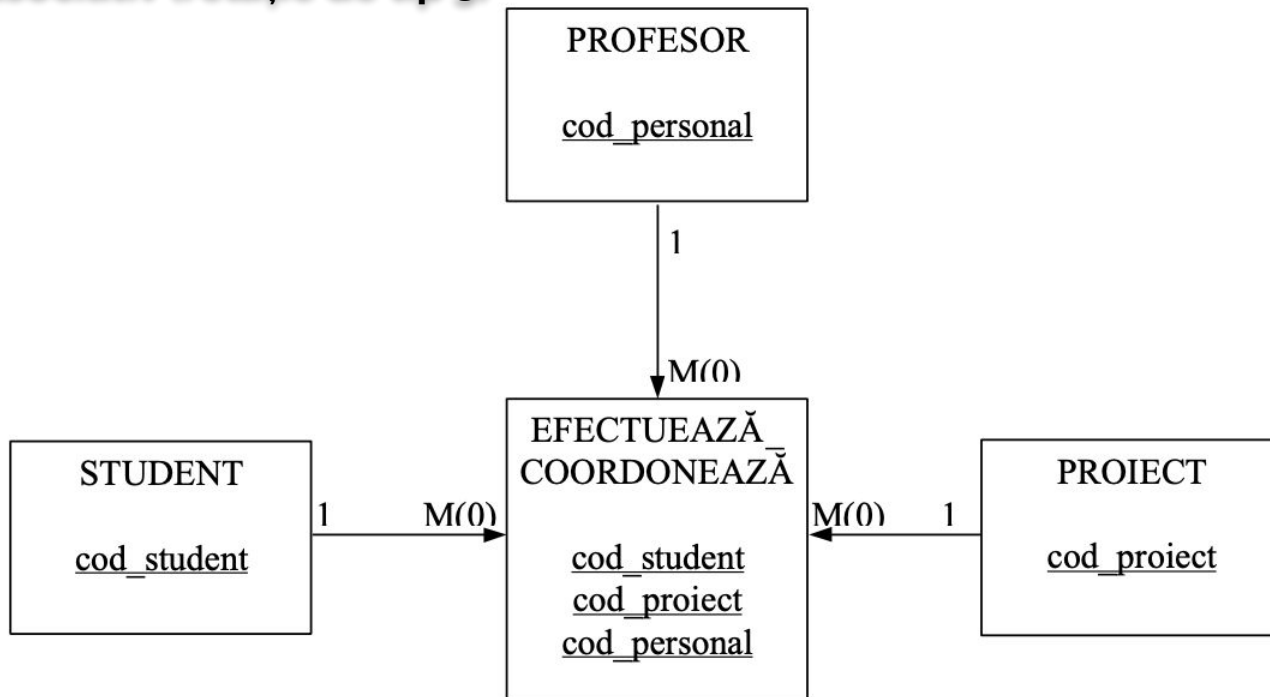




Transformarea relațiilor

- O relație de tip 3 (relație între mai mult de două entități) devine un tabel asociativ care are câte o cheie străină pentru fiecare dintre tabelele asociate; CP este compusă din aceste chei străine plus eventual alte coloane adiționale. De exemplu, tabelul reprezentat în figura următoare exprimă relația „efectuează_coordonează” dintre STUDENT, PROIECT și PROFESOR. În acest caz, CP este combinația cheilor străine corespunzătoare celor 3 entități.

Tabel asociativ (relație de tip 3)





Transformarea atributelor

- Atributele simple ale unei entități devin coloane în tabelul provenit din entitatea corespunzătoare.

De asemenea, fiecare componentă a unui atribut compus devine o coloană în tabel.

De exemplu, pentru atributul compus adresă, format din țară, oraș, stradă, număr și cod, vom avea cinci coloane, câte una pentru fiecare componentă a sa.



Transformarea atributelor

- Atributele repetitive (multivaloare) ale unei entități devin tabele dependente ce conțin o cheie străină (care face referință la CP a entității) și atributul multivaloare;

CP a acestui nou tabel este formată din cheia străină plus una sau mai multe coloane adiționale.

De exemplu, dacă presupunem că un student poate avea mai multe numere de telefon, atunci „nr_telefon” este un atribut multivaloare al entității STUDENT, care va da naștere unui tabel TELEFON, a cărui CP va fi combinația dintre „cod_student” și „nr_telefon”, ca în figura:

Atribute repetitive (multivaloare) ale unei entități





Transformarea atributelor

- Atributele simple ale unei relații 1:1 sau 1:N vor deveni coloane ale tabelului care conține cheia străină.

De exemplu, data angajării, care este un atribut al relației „lucrează_în” dintre PROFESOR și FACULTATE, va fi reprezentată ca o coloană în tabelul PROFESOR.

De asemenea, fiecare atribut compus al unei relații 1:1 sau 1:N va deveni o coloană în tabelul care conține cheia străină.



Transformarea atributelor

- **Atributele simple ale unei relații N:M vor deveni coloane ale tabelului asociativ.** De exemplu, nota obținută la examen, care este un atribut al relației „urmează” dintre STUDENT și CURS va fi reprezentată ca o coloană în tabelul asociativ corespunzător acestei relații.
De asemenea, fiecare componentă a unui atribut compus al unei relații N:M va deveni o coloană în tabelul asociativ.



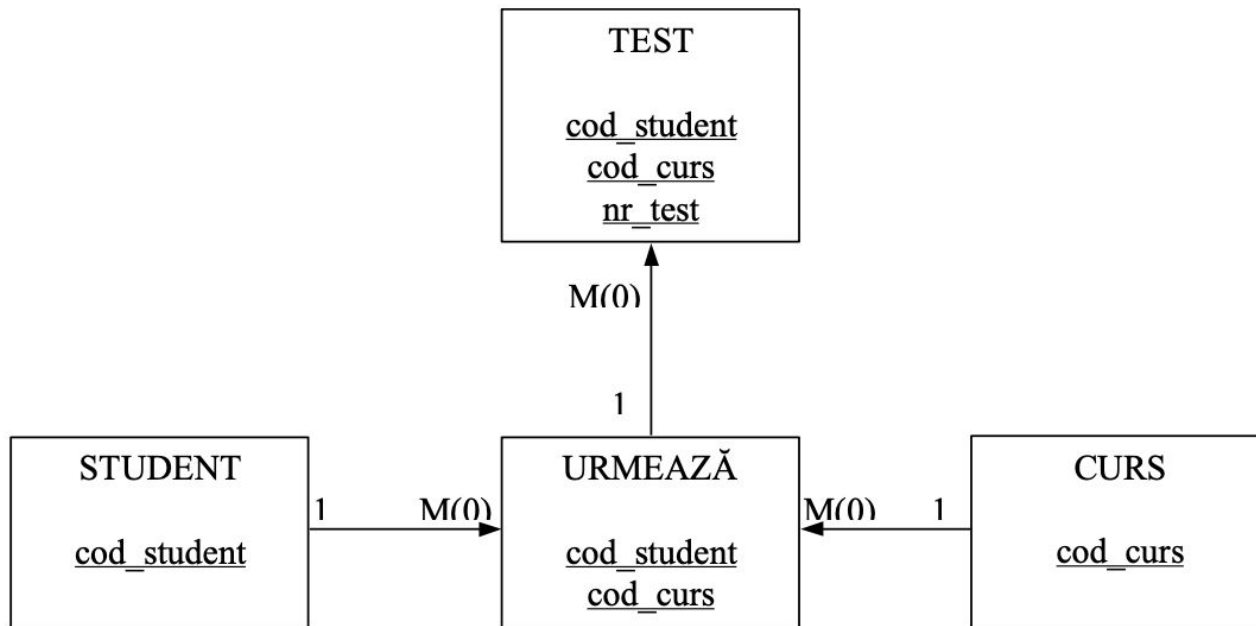
Transformarea atributelor

- Atributele repetitive (multivaloare) ale unei relații 1:1 sau 1:N vor deveni tabele dependente de tabelul care conține cheia străină, iar atributele repetitive ale unei relații N:M vor deveni tabele dependente de tabelul asociativ corespunzător relației.

Evident, CP a acestor tabele dependente va fi o combinație formată din cheia străină respectivă și una sau mai multe coloane adiționale.

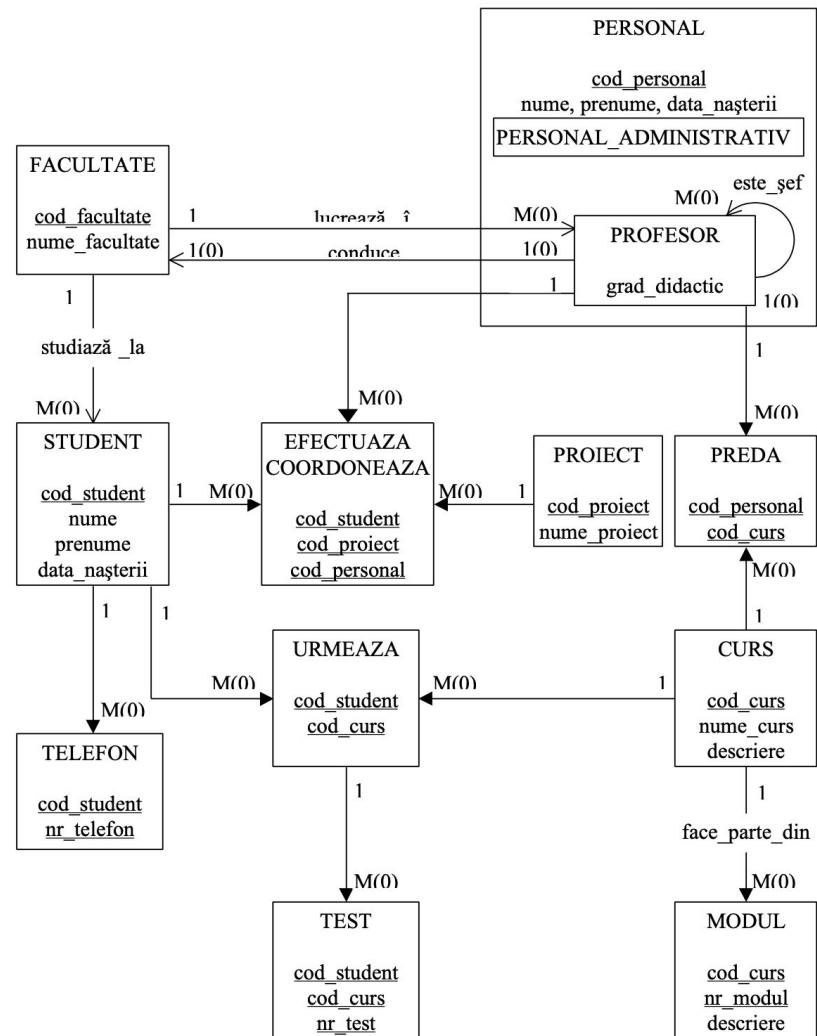
De exemplu, dacă presupunem în cadrul anumitor cursuri că studenții trebuie să dea un nr. de teste, atunci „test” va fi un atribut multivaloare al relației „urmează” dintre STUDENT și CURS și care va da naștere unui tabel dependent de tabelul asociativ al acestei relații, ca în figura:

Atribute repetitive (multivaloare) ale unei relații N:M



Exemplu de diagramă conceptuală

Diagrama conceptuală a BD pentru sistemul descris ca exemplu în acest curs care a rezultat din [ERD](#), în urma transformărilor prezentate mai înainte:





Exemplu de diagramă logică

➤ Tabelele asociate acestei diagrame sunt următoarele:

PERSONAL (cod_personal, nume, prenume, data_nastere, sex, stare_civila)

PERSONAL_ADMINISTRATIV (cod_personal, profesie, funcție)

PROFESOR (cod_personal, grad_didactic, titlu, sef, ore_predate, data_angajării, *cod_facultate*)

CURS (cod_curs, nume_curs, descriere, nr_ore)

PREDA (cod_personal, cod_curs)

MODUL (cod_curs, nr_modul, descriere)

FACULTATE (cod_facultate, nume_facultate, localitate, strada, nr, cod_postal *cod_decan*)



Exemplu de diagramă logică - cont.

STUDENT (cod_student, nume, prenume, data_nasterii, tara, localitate, strada, nr, cod_postal, studii_anterioare)

TELEFON (cod_student, nr_telefon, tip_telefon)

PROIECT (cod_proiect, nume_proiect, domeniu)

EFFECTUEAZA_COORDONEAZA (cod_student, cod_proiect, cod_personal)

URMEAZA (cod_student, cod_curs, nota_examen, nota_restantă, observatii)

TEST (cod_student, cod_curs, nr_test, nota_test, observatii)

- **Notă:** Atributele subliniate constituie CP a tabelului, iar cele italice constituie chei străine.



Alte exemple de scheme conceptuale







Notății

- reprezentarea folosește convenția:

Cardinality and ordinality

Cardinality refers to the maximum number of times an instance in one entity can relate to instances of another entity. Ordinality, on the other hand, is the minimum number of times an instance in one entity can be associated with an instance in the related entity.

Cardinality and ordinality are shown by the styling of a line and its endpoint, according to the chosen notation style.

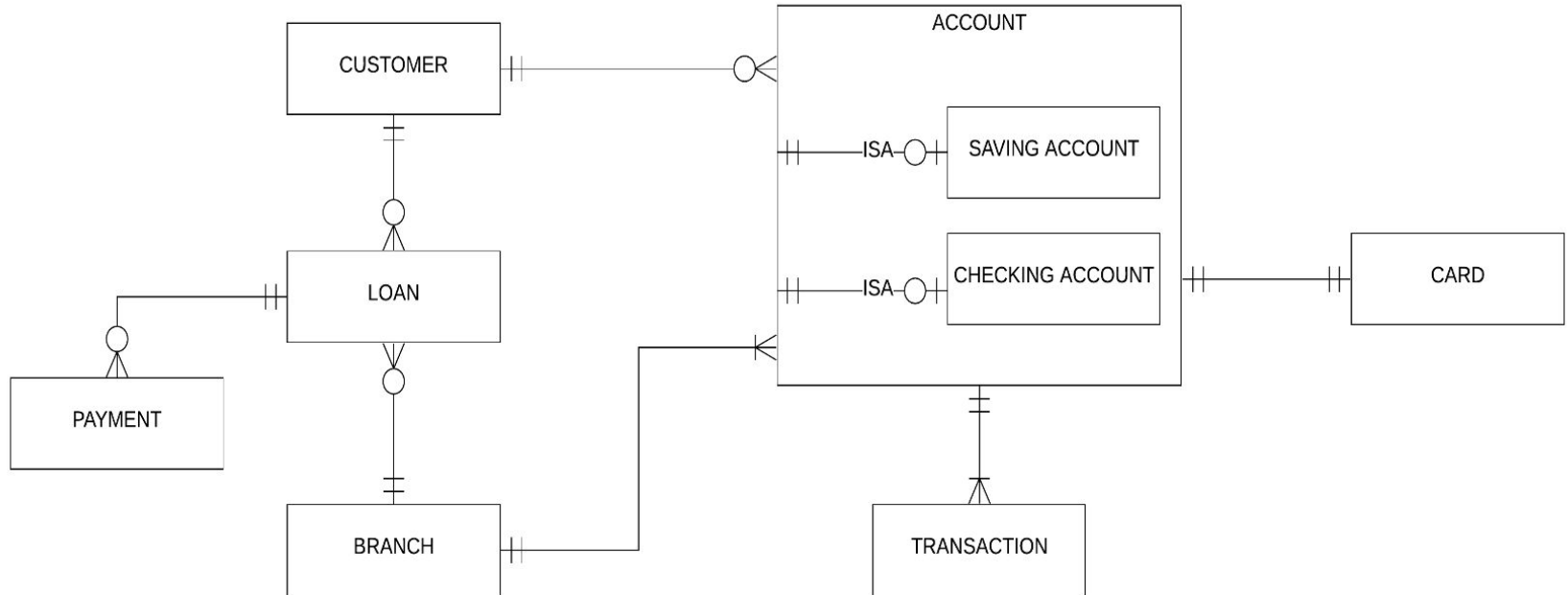
	One
	Many
	One (and only one)
	Zero or one
	One or many
	Zero or many



Banking entities

- A customer opens a saving account or a checking account, at a bank branch. He may also access loans. For each checking account he has a card. Periodically he may withdraw money from his account or partially pay his loans. He may also transfer money from one account to another.

Banking relationships

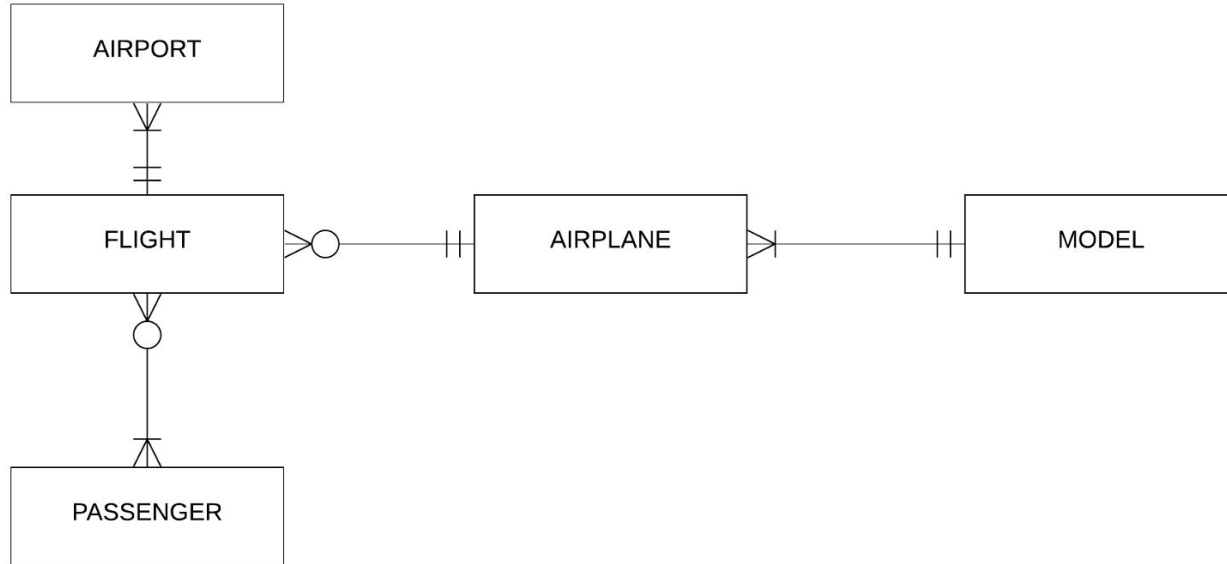




Airline relationships

- The airline has one or more airplanes. An airplane has a model number, and capacity. Each flight is carried out by airplanes. An airplane is uniquely identified by its `Registration_No` and a flight is identified by its `Flight_No`. A passenger can book a ticket for a flight.

Airline relationships

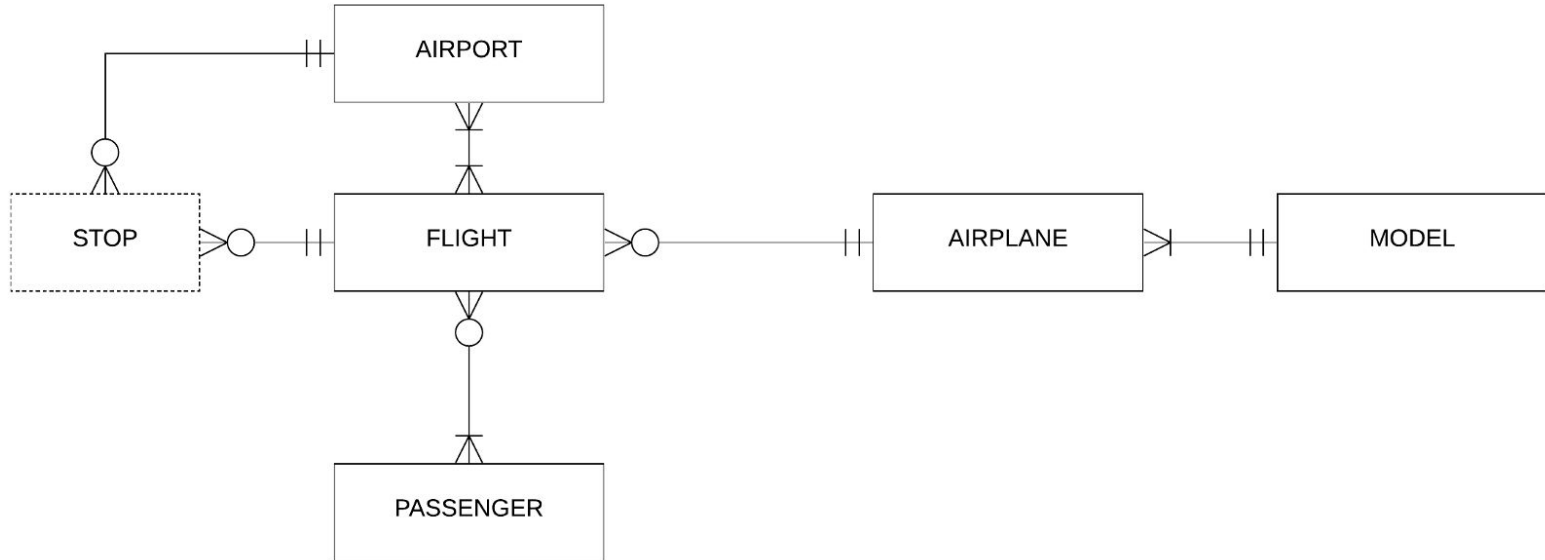




Airline relationships

- The airline has one or more airplanes. An airplane has a model number, and capacity. Each flight is carried out by airplanes. An airplane is uniquely identified by its Registration_No and a flight is identified by its Flight_No. A passenger can book a ticket for a flight. **A flight may have one or more stops.**

Airline relationships

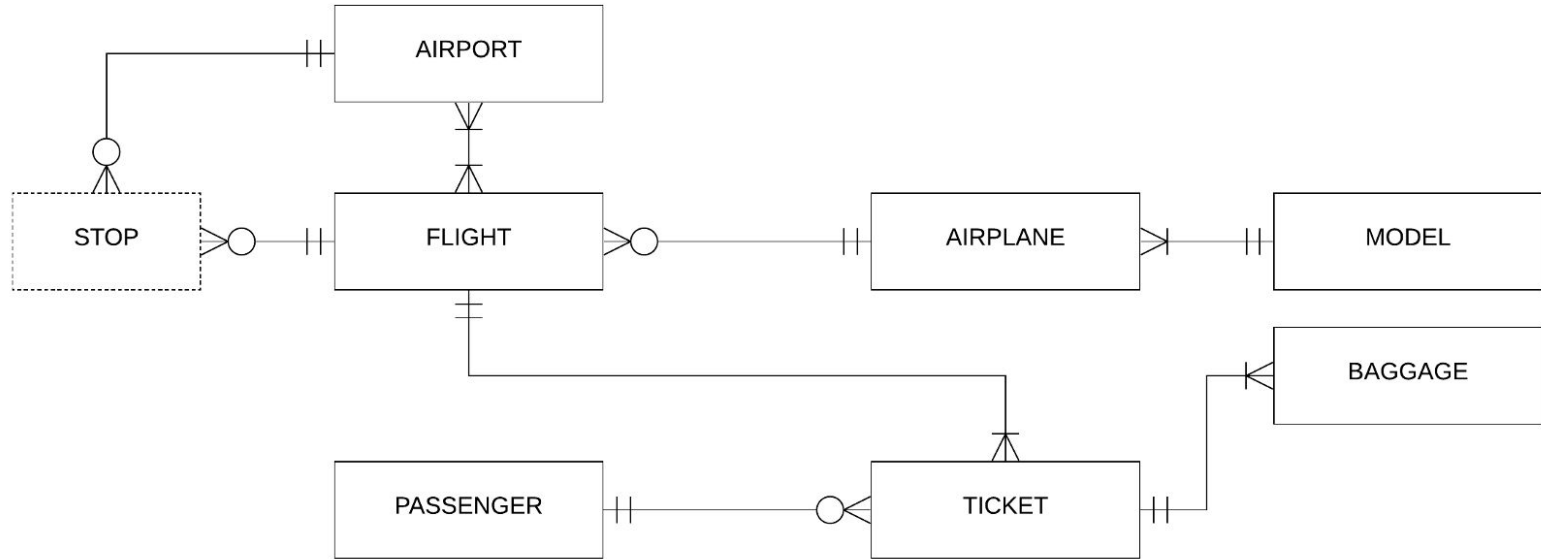




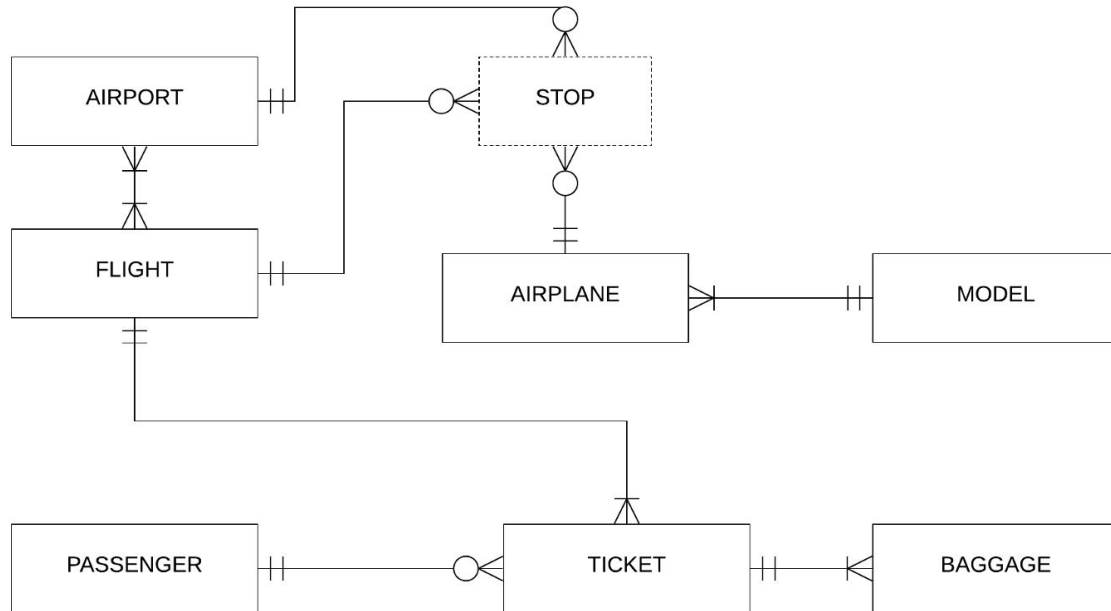
Airline relationships

- The airline has one or more airplanes. An airplane has a model number, and capacity. Each flight is carried out by airplanes. An airplane is uniquely identified by its Registration_No and a flight is identified by its Flight_No. A passenger can book a ticket for a flight. A flight may have one or more stops. **The passenger will pay for extra baggage.**

Airline relationships



Airline relationships





Alte exemple de scheme conceptuale

- în fișierul diagrameER.pdf



3. Crearea design-ului fizic al bazei de date

- aka Normalizarea bazei de date.



Normalizarea bazei de date

- În general, pt. proiectare se creează mai întâi schema conceptuală care este transformată într-un design logic.
- În trecut, în locul acestei tehnici se utiliza o altă tehnică numită normalizare.
- **Normalizarea** constă din descompunerea unui tabel relațional în mai multe tabele care satisfac anumite reguli și stochează aceleași date ca și cel inițial.
- \exists 6 forme normale: prima formă normală, a doua formă normală, a treia formă normală, forma normală Boyce-Codd, a patra formă normală și a cincea formă normală.



Dependențe (nu dependențe!) funcționale

- Fie R un tabel relațional și X și Y 2 submulțimi de coloane ale lui R .
Vom spune că **X determină funcțional pe Y** sau **Y depinde funcțional de X** dacă \nexists 2 rânduri în tabelul R care să aibă aceleași valori pentru coloanele din X și să aibă valori diferite pentru cel puțin o coloană din Y .
Cu alte cuvinte, o valoare a lui X determină în mod unic o valoare a lui Y , adică \forall 2 rânduri din R care au aceeași valoare pentru X trebuie să ia aceeași valoare pentru Y .
Notăție: **$X \rightarrow Y$** . X se va numi **determinant**, iar Y **determinat**.
Spunem că dependența $X \rightarrow Y$ este **trivială** dacă toate elementele lui Y sunt și elemente ale lui X , $Y \subseteq X$.
- **Notă:** În curs simbolul \forall poate însemna oricare, orice sau pentru orice, după caz.

Exemplu

- VÂNZĂRI (cod_client, nume_client, nr_telefon, cod_comandă, data, cod_articol, nume_articol, cost_articol, cantitate) (3NF)

VÂNZĂRI

cod_client	nume_client	nr_telefon	cod_comandă	data	cod_articol	nume_articol	cost_articol	cantitate
A1	Popescu	3215576	C1	12.05.99	P1	cămașă	100.000	2
A1	Popescu	3215576	C1	12.05.99	P3	tricou	50.000	1
A2	Ionescu	2325587	C2	13.05.99	P1	cămașă	100.000	3
A2	Ionescu	2325587	C2	13.05.99	P3	tricou	50.000	2
A2	Ionescu	2325587	C2	13.05.99	P2	pantaloni	200.000	1
A1	Popescu	3215576	C3	14.05.99	P3	tricou	50.000	3
A3	Georgescu	4555895	C4	14.05.99	P1	cămașă	100.000	1



Exemplu - cont.

- Dependențe funcționale:

$\{\text{cod_articol}\} \rightarrow \{\text{nume_articol}, \text{cost_articol}\}$

$\{\text{cod_comandă}\} \rightarrow \{\text{data}, \text{cod_client}, \text{nume_client}, \text{nr_telefon}\}$

$\{\text{cod_client}\} \rightarrow \{\text{nume_client}, \text{nr_telefon}\}$

(2NF)

- **Notă:** dependența $\{\text{cod_comandă}\} \rightarrow \{\text{nume_client}, \text{nr_telefon}\}$ poate fi dedusă din dependențele $\{\text{cod_comandă}\} \rightarrow \{\text{cod_client}\}$ și $\{\text{cod_client}\} \rightarrow \{\text{nume_client}, \text{nr_telefon}\}$. Astfel de dependențe se numesc **dependențe tranzitive** și vor fi definite mai riguros ulterior.



Regula Casey-Delobel

- Fie un tabel $R(X, Y, Z)$ care se descompune prin proiecție în tabelele $R1(X, Y)$ și $R2(X, Z)$ - unde prin X am notat coloanele comune ale tabelelor $R1$ și $R2$, iar prin Y și Z coloanele specifice lui $R1$ și respectiv $R2$; în acest caz, condiția de descompunere fără pierdere de informație presupune ca tabelul R să fie obținut prin compunerea naturală a tabelelor $R1$ și $R2$ (adică prin compunerea rândurilor celor două tabele pentru care valorile coloanelor X sunt identice).



Prima formă normală (1NF)

- Un **tabel relațional** este în **prima formă normală (1NF)** dacă fiecărei coloane îi corespunde o valoare indivizibilă (atomică), deci \forall valoare nu poate să fie o mulțime sau un tuplu de valori. În plus, nu pot să apară grupuri de attribute repetitive.
- Tabelul VÂNZĂRI se află în prima formă normală.
- **Notă:** NF reprezintă abrevierea din lb. eng. pentru normal form



Exemple

- Pentru o coloană care conține date calendaristice, sub forma **zz-ll-aa** (doi digiți pentru zi, doi pentru lună și doi pentru an), se consideră că **valoarea** respectivă **nu se poate descompune** în ziua, luna și anul corespunzătoare acestei valori.
- Dacă vom considera că **adresa** este un atribut compus format din componentele țară, oraș, stradă, număr și cod, fiecare având semnificație proprie și putând fi folosite independent la interogarea bazei de date, atunci adresa va fi reprezentată în tabel prin 5 coloane în loc de una.



Exemple - cont.

- un tabel aflat în 1NF nu poate conține atribute sau grupuri de atribute repetitive.
De exemplu, în tabelul următor, dacă un student poate avea **mai multe numere de telefon**, atunci coloanele „telefon1”, „telefon2”, „telefon3” constituie un **grup de atribute repetitive**. Deci tabelul STUDENT nu este în prima formă normală, el conținând atât un atribut compus („adresa”) cât și un grup de atribute repetitive.

Exemple - cont.

STUDENT								
cod_student	nume	prenume	adresa	telefon1	telefon2	telefon3	materia	nota
101	Ionescu	Vasile	Romania, București, Str. Polizu 5, 7355	6245981	3215678	092659019	Drept	10
101	Ionescu	Vasile	Romania, București, Str. Polizu 5, 7355	6245981	3215678	092659019	Engleza	8
102	Pop	Costică	Romania, Cluj, Str. Unirii 7, 3551	3215469			Fizică	8

- Algoritmul 1NFA permite aducerea unei relații în 1NF prin eliminarea atributelor compuse și a celor repetitive.



Algoritmul 1NFA

1. Se înlocuiesc în tabel coloanele corespunzătoare atributelor compuse cu coloane ce conțin componentele elementare ale acestora.
2. Se plasează grupurile de attribute repetitive, fiecare în câte un nou tabel.
3. Se introduce în fiecare tabel nou creat la pasul 2 CP a tabelului din care a fost extras atributul respectiv. Prin urmare, în tabelul nou creat attributele introduse vor fi chei străine ce fac referință la tabelul din care au fost extrase.
4. Se stabilește CP a fiecărui nou tabel creat la pasul 2. Aceasta va fi creată din cheia străină introdusă la pasul 3 plus una sau mai multe coloane adiționale.

Exemple - cont.

STUDENT

cod_ student	nume	prenume	adresa	telefon1	telefon2	telefon3	materia	nota
101	Ionescu	Vasile	Romania, București, Str. Polizu 5, 7355	6245981	3215678	092659019	Drept	10
101	Ionescu	Vasile	Romania, București, Str. Polizu 5, 7355	6245981			Engleza	8
102	Pop	Costică	Romania, Cluj, Str. Unirii 7, 3551	3215469			Fizică	8

TELEFON

cod_ student	telefon
101	6245981
101	3215678
101	092659019
102	3215469



Observații

- 1NF este o cerință minimală a tuturor sistemelor relaționale. Sistemele de baze de date care nu respectă nici măcar această formă nu pot fi numite relaționale.
- Tabelele aflate în 1NF permit o referire simplă a datelor prin indicarea numelui tabelului, a coloanei și a cheii rândului din care face parte informația respectivă.
- Operatorii pentru aceste tabele sunt mai simpli și permit definirea unor tehnici de proiectare și utilizare a bazelor de date.



A doua formă normală (2NF)

- Fie R un tabel relațional și fie X și Y două submulțimi de coloane ale lui R .
O **dependență funcțională** $X \rightarrow Y$ se numește **totală** dacă
 \forall subset de coloane Z al lui X , $Z \subseteq X$, dacă $Z \rightarrow Y$ atunci $Z = X$.
Cu alte cuvinte, \nexists nici un subset Z al lui X , $Z \neq X$, pentru care $Z \rightarrow Y$.
O dependență funcțională care nu este totală se numește **parțială**.



A doua formă normală (2NF) - cont.

- Un tabel relațional R este în a doua formă normală (2NF) dacă și numai dacă:
 - R este în 1NF.
 - \forall coloană care depinde parțial de o cheie a lui R este inclusă în acea cheie.
- Cu alte cuvinte, a doua formă normală **nu permite dependențe funcționale parțiale față de cheile tabelului**, cu excepția dependențelor triviale, de incluziune.

Deci tabelul VÂNZĂRI nu este în 2NF, coloanele „nume_articol” și „cost_articol” depinzând parțial de CP a tabelului (a se vedea [slide-ul anterior](#)).



Regulă de descompunere

- Pentru a obține tabele relaționale în 2NF, tabelul inițial se descompune fără pierdere de informație astfel:

Fie $R(K1, K2, X, Y)$ un tabel relațional unde $K1$, $K2$, X și Y sunt submulțimi de coloane ale lui R a.î. $K1 \cup K2$ este o cheie a lui R , iar $K1 \rightarrow X$ este o dependență funcțională totală. Dacă $X \subset K1$ atunci tabelul este deja în 2NF, altfel tabelul R poate fi descompus prin proiecție în $R1(K1, K2, Y)$ - având cheia $K1 \cup K2$ - și $R2(K1, X)$ - având cheia $K1$.



Regulă de descompunere - cont.

- Această descompunere conservă nu numai datele, ci și dependențele funcționale, atât determinantul cât și determinatul dependenței eliminate regăsindu-se în tabelul nou creat.
- Folosind această regulă, **algoritmul 2NFA permite** aducerea în 2NF a unui tabel relațional aflat în 1NF prin **eliminarea dependențelor funcționale parțiale**.



Algoritmul 2NFA

1. \forall coloană X care depinde funcțional parțial de o cheie K , $K \rightarrow X$, și care nu este inclusă în K , se determină $K1 \subset K$ un subset al lui K , astfel încât **dependența** $K1 \rightarrow X$ este **totală** și se creează un nou tabel **$R1(K1, X)$** , adică un tabel format din determinantul ($K1$) și determinatul (X) acestei relații.
2. Dacă în tabelul $R \exists$ mai multe dependențe totale ca mai sus cu același determinant, atunci pentru acestea se creează un singur tabel format din determinant - luat o singură dată - și din determinații dependențelor considerate.



Algoritmul 2NFA - cont.

3. Se elimină din tabelul inițial R toate coloanele, X , care formează determinatul dependenței considerate. Determinantul rămâne în tabelul inițial și va fi cheie străină ce face referință la tabelul nou creat.
4. Se determină CP a fiecărui tabel nou creat, $R1$. Aceasta va fi $K1$, determinantul dependenței considerate.
5. Dacă noile tabele create conțin alte dependențe parțiale, atunci se merge la pasul 1, altfel algoritmul se termină.



Exemplu

- Pentru tabelul VÂNZĂRI CP este {cod_comandă, cod_articol}.
Atributul „nume_articol” și „cost_articol” depind funcțional parțial de această CP și depind funcțional total de atributul „cod_articol” conținut în CP.
Același lucru se întâmplă și cu attributele „data”, „cod_client”, „nume_client” și „nr_telefon” care depind funcțional total numai de atributul „cod_comandă” conținut în CP.

Prin urmare vom avea următoarele **dependențe totale**:

{cod_articol} → {nume_articol, cost_articol}

{cod_comandă} → {data, cod_client, nume_client, nr_telefon}



Exemplu - cont.

- Tabelul VÂNZĂRI (cod_client, nume_client, nr_telefon, cod_comandă, data, cod_articol, nume_articol, cost_articol, cantitate) va fi descompus în

VÂNZĂRI_1 (cod_client, nume_client, nr_telefon, cod_comandă, data, cod_articol, cantitate)
și

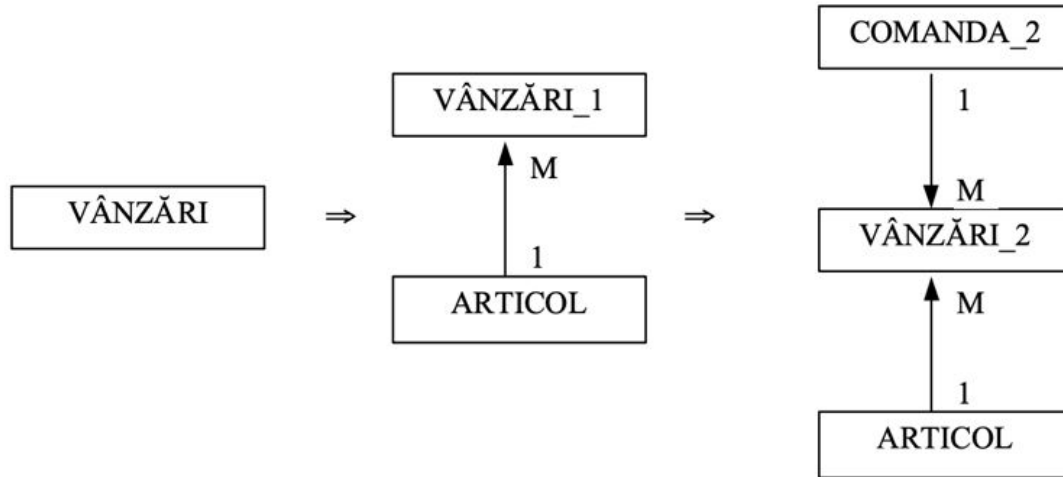
ARTICOL (cod_articol, nume_articol, cost_articol).

Tabelul VÂNZĂRI_1 poate fi descompus în

VÂNZĂRI_2 (cod_comandă, cod_articol, cantitate) și

COMANDA_2 (cod_comandă, data, cod_client, nume_client, nr_telefon).

Exemplu - cont.



- Un tabel care are CP formată dintr-un singur atribut este automat în 2NF. Prin urmare, algoritmul 2NFA nu se poate aplica decât în cazul în care CP a unui tabel este o cheie compusă.



A treia formă normală (3NF)

- Deși tabelul COMANDA_2 este în 2NF, se observă că încă mai există redundanță în date - tuplul (A1, Popescu, 3215576) apare de două ori (a se vedea [slide-ul anterior](#)).
Intuitiv, aceasta se explică prin faptul că attributele „nume_client” și „nr_telefon” depind indirect de CP a tabelului, dependența făcându-se prin intermediul atributului „cod_client”. Aceste dependențe indirecte vor fi îndepărtate în 3NF.



A treia formă normală (3NF) - cont.

- Un tabel relațional **R** este în a treia formă normală (3NF) dacă și numai dacă:
 - R este în 2NF.
 - \forall coloană **A** neconținută în nici o cheie a lui R, dacă \exists un set de coloane **X** a.î. $X \rightarrow A$, atunci fie **X** conține o cheie a lui R, fie **A** este inclusă în **X**.
- A doua condiție din definiție **interzice dependențele funcționale totale față de alte coloane în afara celor care constituie chei ale tabelului.**
Prin urmare, un tabel este în 3NF dacă \forall coloană care nu este conținută într-o cheie depinde de cheie, de întreaga cheie și numai de cheie.



A treia formă normală (3NF) - cont.

- În mod evident tabelul COMANDA_2 nu este în 3NF existând dependențele
 $\{\text{cod_client}\} \rightarrow \{\text{nume_client}\}$ și
 $\{\text{cod_client}\} \rightarrow \{\text{nr_telefon}\}.$

Pe de altă parte, tabelele VÂNZĂRI_2 și ARTICOL sunt în 3NF.



A treia formă normală (3NF) - cont.

- Cea de-a doua condiție din definiția de mai înainte se formulează folosind noțiunea de **dependență tranzitivă**.
Fie R un tabel relațional, X o submulțime de coloane a lui R și A o coloană a lui R .
Spunem că **A este dependentă tranzitiv de X** dacă \exists o submulțime de coloane Y care nu include A și nu determină funcțional pe X astfel încât $X \rightarrow Y$ și $Y \rightarrow A$.
Dacă în această definiție se dorește să se evidențieze și Y atunci se spune că A depinde funcțional de X prin intermediul lui Y și se scrie $X \rightarrow Y \rightarrow A$.



A treia formă normală (3NF) - cont.

- De exemplu, în tabelul COMANDA_2 coloanele „nume_client” și „nr_telefon” depind tranzitiv de CP „cod_comandă” prin intermediul coloanei „cod_client”.
Folosind această definiție, condiția ca un tabel să fie în 3NF se poate reformula astfel:
- Un **tabel relațional R** este în a treia formă normală (3NF) dacă și numai dacă:
 - R este în 2NF.
 - \forall coloană neconținută în nici o cheie a lui R nu este dependentă tranzitiv de nici o cheie a lui R.



Regulă de descompunere

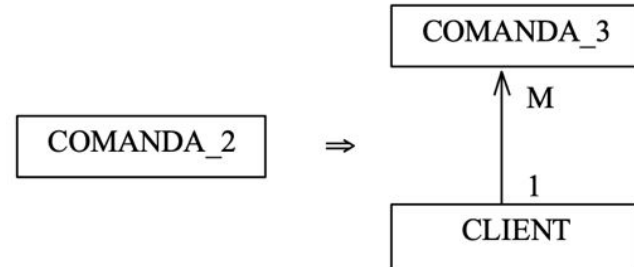
- Pentru a obține tabele relaționale în 3NF, tabelul inițial se descompune fără pierdere de informație după următoarele reguli.

Fie $R(K, X, Y, Z)$ un tabel relațional unde K este o cheie a lui R , iar X, Y și Z sunt submulțimi de coloane ale lui R .

- Dacă există **dependența tranzitivă** $K \rightarrow X \rightarrow Y$, atunci R se poate descompune în $R1(K, X, Z)$ - având cheia K - și $R2(X, Y)$ - având cheia X .
- Dependența tranzitivă poate fi mai complexă. Fie $K1 \subset K$ o parte a cheii K astfel încât există dependența tranzitivă $K \rightarrow K1 \cup X \rightarrow Y$. În acest caz, R poate fi descompus în $R1(K, X, Z)$ - având cheia K - și $R2(K1, X, Y)$ - având cheia $K1 \cup X$.

Regulă de descompunere - cont.

- Descompunerile corespunzătoare regulilor de mai înainte conservă nu numai datele, ci și dependențele funcționale, determinantul și determinatul dependențelor eliminate regăsindu-se în tabelele nou create.
- Un exemplu de aplicare a primei reguli este descompunerea tabelului COMANDA_2 în COMANDA_3 (cod_comandă, data, cod_client) și CLIENT (cod_client, nume_client, nr_telefon)



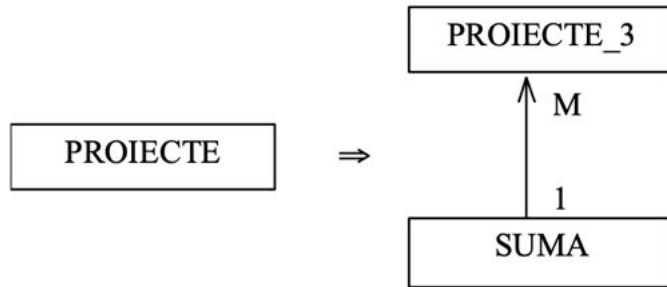


Exemplu

- Pentru a exemplifica cea de-a doua regulă, considerăm PROIECTE (cod_angajat, cod_proiect, rol_în_proiect, suma_obținută), care stochează date privind repartizarea pe proiecte a angajaților unei firme. Presupunem că suma obținută de un angajat depinde de proiectul respectiv și de rolul angajatului în acel proiect, deci avem dependența $\{\text{cod_proiect}, \text{rol_în_proiect}\} \rightarrow \{\text{suma_obținută}\}$.

Exemplu - cont.

- Aplicând regula a doua, tabelul PROIECTE se descompune în:
PROIECTE_3 (cod_angajat, cod_proiect, rol_în_proiect) și
SUMA (cod_proiect, rol_în_proiect, suma_obținută).



- Folosind această regulă, algoritmul 3NFA permite aducerea în 3NF a unui tabel relațional aflat în 2NF prin **eliminarea dependențelor funcționale tranzitive**.

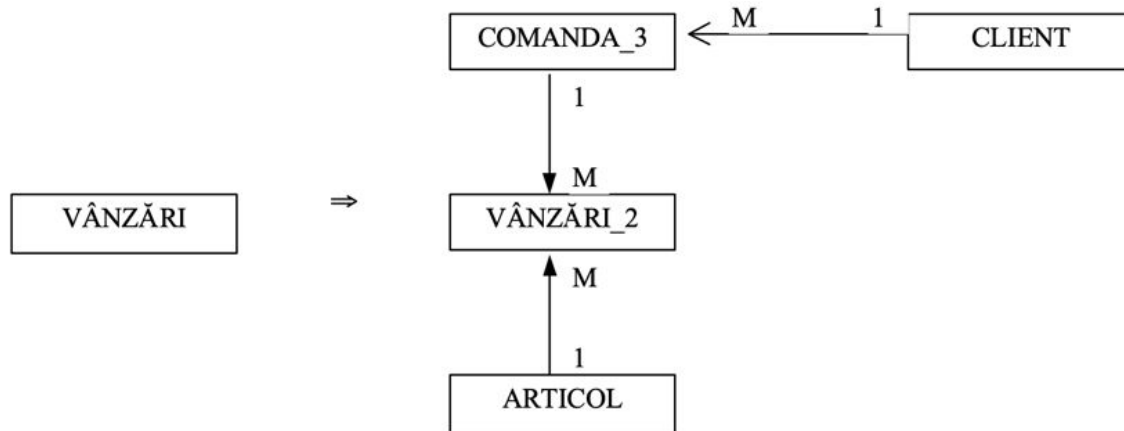


Algoritmul 3NFA

1. Pentru fiecare dependență funcțională tranzitivă $K \rightarrow X \rightarrow Y$ se transferă coloanele din X și Y într-o nouă relație.
2. Se determină CP a fiecărei noi relații create la pasul 1, aceasta fiind formată din coloanele din X.
3. Se elimină din relația principală coloanele din Y. Coloanele din X sunt cheie străină ce face referință la tabelul nou creat.
4. Dacă tabelele rezultate conțin alte dependențe tranzitive, atunci se merge la pasul 1, altfel algoritmul se termină.

Exemplu

- Aplicând algoritmi 2NFA și 3NFA, tabelul VÂNZĂRI a fost descompus în tabelele VÂNZĂRI_2, ARTICOL, COMANDA_3 și CLIENT, care sunt toate în 3NF. În aceste tabele nu mai există nici redundanță în date și nici anomalii de actualizare.





Forma normală Boyce-Codd (BCNF)

- Nu toate tabelele aflate în 3NF sunt lipsite de redundanță în date și anomalii de actualizare. Pentru a ilustra această situație să considerăm următorul exemplu.

O companie de transporturi efectuează curse, în care poate folosi unul sau mai mulți șoferi - de exemplu prima jumătate a cursei conduce un șofer, cea de-a doua alt șofer - și mai multe dintre autobuzele aflate în dotare, cu condiția ca într-o cursă un șofer să conducă un singur autobuz.

Pe de altă parte însă, un autobuz este repartizat unui șofer și deci nu poate fi condus decât de acesta.



Forma normală Boyce-Codd (BCNF) - cont.

- Această situație poate fi modelată printr-un tabel TRANSPORTURI (cod_cursă, cod_șofer, cod_autobuz, loc_plecare, loc_sosire).
- În acest tabel avem dependențele:
 $\{\text{cod_cursă}, \text{cod_șofer}\} \rightarrow \{\text{cod_autobuz}, \text{loc_plecare}, \text{loc_sosire}\}$
 $\{\text{cod_autobuz}\} \rightarrow \{\text{cod_șofer}\},$
iar cheile tabelului sunt $\{\text{cod_cursă}, \text{cod_șofer}\}$ și $\{\text{cod_cursă}, \text{cod_autobuz}\}$.
- Tabelul TRANSPORTURI este în 3NF și totuși în acest tabel există redundanță în date datorită dependenței $\{\text{cod_autobuz}\} \rightarrow \{\text{cod_șofer}\}.$



Forma normală Boyce-Codd (BCNF) - cont.

- Forma Boyce-Codd elimină acest tip de redundanțe.
Intuitiv, un tabel R este în BCNF dacă fiecare determinant al unei dependențe funcționale este cheie candidată a lui R.
Tabelul TRANSPORTURI de mai înainte nu este în BCNF, „cod_autobuz” nefiind o cheie a lui R.
- Un **tabel relațional este în forma normală Boyce-Codd (BCNF)** $\Leftrightarrow \forall$ dependență funcțională totală $X \rightarrow A$, unde X este un subset de coloane iar A o coloană neconținută în X, X este o cheie a lui R.



Forma normală Boyce-Codd (BCNF) - cont.

- \forall tabel care are cel mult 2 coloane este în BCNF.
- \forall tabel relațional se poate descompune fără pierdere de informație în tabele aflate în BCNF, dar nu același lucru se poate spune despre descompunerea cu păstrarea dependențelor funcționale, după cum vom vedea în continuare.



Algoritmul BCNFA

1. \forall dependență **non-cheie** $X \rightarrow Y$, unde X și Y sunt subseturi de coloane ale lui R , se creează 2 tabele. Una dintre ele va fi formată din coloanele $\{X, Y\}$, iar cealaltă va fi formată din toate coloanele inițiale, mai puțin coloanele Y .
2. Dacă tabelele rezultate conțin alte dependențe non-cheie, atunci se merge la pasul 1, altfel algoritmul se termină.



Exemplu

- Aplicând acest algoritm, tabelul TRANSPORTURI se va descompune în tabelele TRANSPORTURI_BC (cod_cursă, cod_autobuz, loc_plecare, loc_sosire) și AUTOBUZ (cod_autobuz, cod_șofer).
- Descompunerea s-a făcut fără pierdere de informație, dar a fost pierdută dependența funcțională $\{\text{cod_cursă}, \text{cod_șofer}\} \rightarrow \{\text{cod_autobuz}\}$.



Bibliografie

F. Ipate, M. Popescu, *Dezvoltarea aplicațiilor de baze de date în Oracle 8 și Oracle Forms 6*, Editura ALL, 2000.