



Baze de date

Curs 2 – Modelul relațional (cont.)

Proiectarea bazelor de date relaționale

Sorina Preduț

sorina.predut@my.fmi.unibuc.ro

Universitatea din București



Cuprins (Modelul relațional)

- I. Concepte de bază
- II. Constrângeri de integritate
- III. Operatorii sistemului relațional
 - A. Algebra relațională și limbajul SQL
- IV. Tabele, rânduri, coloane
- V. Sisteme de Gestiune a Bazelor de Date Relaționale (SGBDR)



DIVISION (diviziunea)

- Diviziunea este o operație binară care se aplică asupra a două relații R și S , a.î. mulțimea atributelor lui R include mulțimea atributelor lui S .
- Dacă R este o relație cu aritatea m , iar S o relație cu aritatea n , unde $m > n$, atunci diviziunea lui R la S este **mulțimea tuplurilor de dimensiune $m - n$ la care, adăugând orice tuplu din S , se obține un tuplu din R .**
- Notățiile utilizate cel mai frecvent sunt $R \div S$, **DIVISION(R, S)**, **DIVIDE(R, S)**.



Exemplu

R

A	B	C
x1	y1	z1
x1	y2	z1
x1	y1	z2
x2	y1	z2
x2	y2	z1

S

C
z1
z2

$R \div S$

A	B
x1	y1



DIVISION (diviziunea) - cont.

- Diviziunea este o operație derivată care se exprimă cu ajutorul diferenței, produsului cartezian și proiecției:


$$R \div S = R_1 - R_2, \text{ unde}$$

$$R_1 = \Pi_X(R),$$

$$R_2 = \Pi_X((R_1 \times S) - R),$$

iar X este mulțimea atributelor lui R care nu există în S .

Pentru exemplul de mai înainte:

R_1 

A	B
x1	y1
x1	y2
x2	y1
x2	y2

 R_2

A	B
x2	y1
x2	y2
x1	y2

 $R_1 \times S$

A	B	C
x1	y1	z1
x1	y2	z1
x2	y1	z1
x2	y2	z1
x1	y1	z2
x1	y2	z2
x2	y1	z2
x2	y2	z2

 $(R_1 \times S) - R$

A	B	C
x2	y1	z1
x2	y2	z1
x1	y2	z2

 $R_1 \div R_2$

A	B
x1	y1



DIVISION (diviziunea) - cont.

- Operatorul DIVISION este legat de cuantificatorul universal (\forall) care nu există în SQL, dar care poate fi simulat cu ajutorul cuantificatorului existențial (\exists), care există în SQL, utilizând relația:
$$\forall x P(x) \equiv \text{NOT } \exists x \text{ NOT } P(x).$$
- Pentru a ilustra exprimarea operatorul DIVISION în cele două moduri (folosind cuantificatorul universal și cuantificatorul existențial), să considerăm relațiile `curs_student` și `curs_fundamental`:

 curs_student

cod_student	Curs
S1	matematica
S1	Fizica
S1	Mecanica
S2	matematica
S2	informatica
S3	Fizica
S4	matematica
S4	Fizica

curs_fundamental

curs
matematica
fizica

curs_student ÷ curs_fundamental

cod_student
S1
S4



DIVISION (diviziunea) - cont.

- Atunci relația `curs_student ÷ curs_fundamental` poate fi definită prin următoarea întrebare:
care sunt studenții care urmează toate cursurile fundamentale?
- Alternativ, această relație poate fi definită prin întrebarea:
- care sunt studenții pentru care nu există curs fundamental care să nu fie urmat de către aceștia?
- Utilizând a doua formulare, rezultă că operatorul DIVISION poate fi simulat în SQL prin doi operatori NOT EXISTS:



DIVISION (diviziunea) - cont.

```
SELECT DISTINCT cod_student
FROM curs_student cs1
WHERE NOT EXISTS
    (SELECT *
     FROM curs_fundamental cf
     WHERE NOT EXISTS
        (SELECT *
         FROM curs_student cs2
         WHERE cf.curs = cs2.curs
         AND cs1.cod_student = cs2.cod_student));
```



JOIN (compunerea, joncțiunea)

- Operatorul de compunere permite regăsirea informației din mai multe relații corelate. Compunerea este o operație binară care are ca rezultat **o nouă relație** în care **fiecare tuplu este o combinație a unui tuplu din prima relație cu un tuplu din a doua relație**.
- Operatorul JOIN este un **operator derivat**, putând fi simulat printr-o combinație de **produs cartezian, selecție și proiecție**.
- În general, se construiește un produs cartezian, se elimină tupluri prin selecție și se elimină attribute prin proiecție.
- După modalitățile în care se face selecția și proiecția, se disting mai multe tipuri de compunere: **THETA-JOIN, NATURAL-JOIN, SEMI-JOIN, OUTER-JOIN**.



THETA-JOIN

- Operatorul THETA-JOIN combină perechile de tupluri din două relații, cu condiția ca între **valorile atributelor specificate** să existe o anumită legătură, adică **să satisfacă o anumită condiție specificată explicit în cadrul operației**.
- În cadrul condiției operatorului THETA-JOIN se poate folosi orice operator de comparație (>, >=, <, <=, <>, =).
- În cazul în care este folosit operatorul de comparație =, tipul de compunere se numește **EQUI-JOIN**.
- Operatorul THETA-JOIN se reprezintă de obicei cu ajutorul simbolului \bowtie sub care se scrie condiția, $R \bowtie_{\text{condiție}} S$ sau prin **JOIN(R, S, condiție)**.



Exemplu

R

A	B	C
x1	y1	1
x2	y2	3
x3	y3	5

S

D	E
2	z1
4	z2

$R \bowtie_{C < D} S$

A	B	C	D	E
x1	y1	1	2	z1
x1	y1	1	4	z2
x2	y2	3	4	z2



THETA-JOIN - cont.

- Următorul exemplu ilustrează realizarea operatorului THETA-JOIN în SQL:

```
SELECT *  
FROM R, S  
WHERE R.C < S.D;
```



NATURAL-JOIN (Compunerea naturală)

- Compunerea naturală este o operație binară comutativă care **combină tupluri din două relații, R și S, cu condiția ca attributele comune să aibă valori identice.**
- În cazul compunerii naturale attributele specificate trebuie să aibă același nume.
- Practic diferența dintre NATURAL-JOIN și EQUI-JOIN constă în faptul că în primul caz numele atributelor sunt identice, iar în cel de al doilea caz acestea sunt diferite.
- De obicei, compunerea naturală se notează prin $R \bowtie S$ sau **JOIN(R, S).**



NATURAL-JOIN (Compunerea naturală) - cont.

- Pentru 2 relații R și S , compunerea naturală pe un set de attribute comune X constă în efectuarea succesivă a următoarelor operații:
 - Se calculează produsul cartezian $R \times S$.
 - Se selectează din $R \times S$ acele tupluri obținute pentru care valorile atributelor X din tuplul R sunt identice cu valorile atributelor X din tuplul S .
 - Deoarece în relația astfel obținută attributele X apar de două ori (o dată provenind din R și o dată din S), se elimină una dintre aparițiile acestor attribute.

lucrător

nr_lucrător	cod_secție	nr_atelier
1	S1	1
2	S1	2
3	S2	1
4	S1	2
5	S1	1

atelier

cod_secție	nr_atelier	denumire
S1	1	Proiectare
S1	2	Informatica
S2	1	Mecanica
S2	2	Electrotehnica

lucrător ✕ atelier

nr_lucrător	cod_secție	nr_atelier	denumire
1	S1	1	Proiectare
2	S1	2	Informatica
3	S2	1	Mecanica
4	S1	2	Informatica
5	S1	1	Proiectare



NATURAL-JOIN (Compunerea naturală) - cont.

- În exemplul de mai sus, {cod_secție, nr_atelier} este cheie primară în tabelul atelier și cheie străină în tabelul lucrător.
- Următorul exemplu ilustrează realizarea compunerii naturale în SQL:

```
SELECT lucrător.nr_lucrător, lucrător.cod_secție,  
       lucrător.nr_atelier, atelier.denumire  
FROM lucrător, atelier  
WHERE lucrător.cod_secție = atelier.cod_secție  
AND lucrător.nr_atelier = atelier.nr_atelier;
```



SEMI-JOIN (semi-compunerea)

- Operația de semi-compunere aplicată asupra a două relații R și S generează o relație care conține toate tuplurile din R corelate cu oricare din tuplurile din S.
- Operația nu este comutativă.
- Se notează de obicei prin **SEMI-JOIN(R, S)**.



Exemplu

R

A	B	C
x1	y1	z1
x2	y1	z1
x3	y2	z1
x4	y2	z2

S

B	C	D
y1	z1	u1
y2	z2	u2
y2	z2	u3

SEMIJOIN(R,S)

A	B	C
x1	y1	Z1
x2	y1	z1
x4	y2	z2



SEMI-JOIN (semi-compunerea) - cont.

- Următorul exemplu ilustrează realizarea semi-compunerii în SQL:

```
SELECT DISTINCT R.A, R.B, R.C  
FROM R, S  
WHERE R.B = S.B  
AND R.C = S.C;
```



OUTER-JOIN (Compunerea externă)

- Operația de compunere externă este o extindere a compunerii naturale.
- În cazul aplicării operatorului NATURAL-JOIN se pot pierde tupluri atunci când există un tuplu într-una din relații care nu este corelat cu nici un tuplu din cealaltă relație.
- Operatorul OUTER-JOIN elimină acest inconvenient. Practic, la aplicarea operatorului OUTER-JOIN, se realizează **compunerea naturală a celor două relații**, la care se adaugă tuplurile din S care nu sunt conținute în compunere, completate cu valori Null pentru attributele rămase din R.



OUTER-JOIN (Compunerea externă)

- Operatorul se notează cu **OUTERJOIN(R, S)**.
- Există și alte variante ale acestui operator, de exemplu o altă variantă adaugă la tuplurile obținute din compunerea naturală a lui R și S atât tuplurile din R care nu sunt conținute în compunere cât și tuplurile din S care nu sunt conținute în compunere, completând restul cu Null.
În mod evident, această variantă a operatorului se poate obține cu ușurință din varianta prezentată de noi.
- Operatorul OUTER-JOIN, în varianta prezentată de noi, este ne-comutativ.



Exemplu

student

nr_stud	nume	prenume	cod_facult
1	Popescu	Ion	F1
2	Ionescu	Vasile	F1
3	Ionescu	Viorel	F2
4	Costache	Ion	F2
5	Matache	Mihai	F1

facultate

cod_facult	nume_facult	localitate
F1	Matematica	București
F2	Fizica	București
F3	Informatica	Pitești
F4	Mecanica	Ploiești



OUTERJOIN(student, facultate)

nr_stud	nume	prenume	cod_facult		
1	Popescu	Ion	F1	Matematica	București
2	Ionescu	Vasile	F1	Matematica	București
3	Ionescu	Viorel	F2	Fizica	București
4	Costache	Ion	F2	Fizica	București
5	Matache	Mihai	F1	Matematica	București
Null	Null	Null	F3	Informatica	Pitești
Null	Null	Null	F4	Mecanica	Ploiești



OUTER-JOIN (Compunerea externă) - cont.

- În versiunea SQL folosită de Oracle, operatorul OUTER JOIN este specificat prin sufixul (+) adăugat la câmpul după care se face compunerea, corespunzător tuplului ale cărui attribute pot fi completate cu Null:

```
SELECT student.nume_stud, student.nume, student.prenume,  
       facultate.cod_facult, facultate.nume_facult,  
       facultate.localitate  
FROM student, facultate  
WHERE student.cod_facult (+) = facultate.cod_facult;
```



IV. Tabele, rânduri, coloane

- Modelul relațional este bazat pe matematica relațională.
- Preluarea modelului relațional de către economie a implicat o transformare a terminologiei relaționale într-una care poate fi ușor înțeleasă de cei fără o pregătire specială în domeniu.
- Pentru cei care nu sunt experți în procesarea datelor, **relațiile devin tabele, tuplurile devin rânduri și attributele devin coloane.**
Acesta este și terminologia pe care o vom folosi în continuare.



V. Sisteme de Gestiune a Bazelor de Date Relaționale (SGBDR)

- În principiu, un sistem de gestiune a bazelor de date relaționale (SGBDR) este un SGBD care utilizează drept concepție de organizare a datelor modelul relațional.
- Evident definiția este mult prea generală pentru a putea fi folosită în practică, deoarece modul de implementare a modelului relațional diferă de la un producător la altul.
- În 1985, Codd a publicat **un set de 13 reguli în raport cu care un SGBD poate fi apreciat ca relațional.**



V. SGBDR - cont.

- Nici un SGBD comercializat în prezent nu satisface în totalitate regulile lui Codd, dar aceasta nu împiedică etichetarea acestora drept relaționale.
- Regulile lui Codd nu trebuie folosite pentru a aprecia dacă un sistem este sau nu relațional, ci măsura în care acesta este relațional, adică nr. regulilor lui Codd respectate de către acesta.



Regulile lui Codd

- **Regula 1. Regula reprezentării logice a datelor:** Într-o BD relațională, toate datele sunt reprezentate la nivel logic într-un singur mod, și anume sub formă de valori atomice în tabele.
- Deci toate datele trebuie memorate sub formă de tabele, iar valoarea corespunzătoare intersecției dintre un rând și o coloană trebuie să fie atomică, adică să nu mai poată fi descompusă din punct de vedere logic.



Regulile lui Codd - cont.

- Un exemplu de încălcare a acestei reguli este stocarea ca o singură coloană a unui cod al automobilului, obținut prin concatenarea mai multor coduri, reprezentând marca automobilului, culoarea, fabrica unde este produs, seria și numărul de fabricație, etc.
- Uneori această regulă este încălcată în practică, dar acest lucru este de cele mai multe ori semnul unui design de calitate slabă, creând probleme de integritate a BD.
- **Această regulă este cea mai importantă dintre cele definite de Codd, iar un SGBD care nu respectă această regulă nu poate fi în nici un caz considerat relațional.**



Regulile lui Codd - cont.

- **Regula 2. Regula accesului la date:** Toate datele individuale din tabele trebuie să fie accesibile prin furnizarea numelui tabelului, numelui coloanei și valorii cheii primare.
- Conform modelului relațional, într-un tabel nu pot exista rânduri identice, iar fiecare rând poate fi identificat prin valoarea cheii primare.
În consecință, orice dată individuală poate fi identificată folosind numele tabelului, al coloanei și valoarea cheii primare.



Regulile lui Codd - cont.

- **Oracle nu respectă această regulă** deoarece permite existența a mai multe rânduri identice în același tabel.
- Totuși, acest lucru poate fi evitat, de exemplu prin definirea unei chei primare, care elimină implicit și posibilitatea existenței rândurilor identice.
- Pe de altă parte, această regulă este încălcată în Oracle și de existența identificatorului de rând, ROWID, care poate fi folosit pentru accesarea rândului respectiv.



Regulile lui Codd - cont.

- **Regula 3. Regula reprezentării valorilor necunoscute:** Un sistem relațional trebuie să permită declararea și manipularea sistematică a valorilor Null, cu semnificația unor valori necunoscute sau inaplicabile.
- Această regulă, implică, de exemplu, că un SGBDR trebuie să facă diferența între valoarea numerică 0 și Null sau între șirul de caractere „spațiu” și valoarea Null.
- Valoarea Null trebuie să reprezinte absența informației respective și are un rol important în implementarea restricțiilor de integritate structurală (integritatea entității și integritatea referirii).
- **Oracle respectă această regulă**, limbajul SQL permițând declararea și manipularea valorilor Null.



Regulile lui Codd - cont.

- **Regula 4. Regula dicționarului de date:** Descrierea BD (dicționarul de date) trebuie să fie reprezentată la nivel logic tot sub formă de tabele, astfel încât asupra acesteia să se poată aplica aceleași operații ca și asupra datelor propriu-zise.
- Cu alte cuvinte, dicționarul de date trebuie să fie organizat la nivel logic și accesat la fel ca orice tabel din baza de date.
- **Această regulă este respectată de către Oracle**, dicționarul de date constând din tabele și tabele virtuale (vederi) care pot fi interogate la fel ca oricare alte tabele sau vederi, folosind comanda SELECT.



Regulile lui Codd - cont.

- **Regula 5. Regula limbajului de acces:** Într-un sistem relațional trebuie să existe cel puțin un limbaj de accesare a datelor, care să asigure următoarele operații:
 - definirea tabelor de bază și a tabelor virtuale (vederilor),
 - manipularea și interogarea datelor (atât interactiv cât și prin program),
 - definirea restricțiilor de integritate,
 - autorizarea accesului la date,
 - delimitarea tranzacțiilor.



Regulile lui Codd - cont.

- **Limbajul SQL folosit de către Oracle permite**
 - definirea tabelor (comenzile CREATE TABLE, ALTER TABLE, DROP TABLE), a vederilor (comenzile CREATE VIEW, ALTER VIEW, DROP VIEW),
 - manipularea (comenzile INSERT, UPDATE, DELETE) și interogarea acestora (comanda SELECT),
 - definirea restricțiilor de integritate (clauza CONSTRAINT folosită la definirea tabelor),
 - autorizarea accesului la date (printr-un set de privilegii de sistem și la nivel de obiect),
 - delimitarea tranzacțiilor (operațiile COMMIT și ROLLBACK).



Regulile lui Codd - cont.

- **Regula 6. Regula de actualizare a tabelelor virtuale (vederilor):** Un SGBD trebuie să poată determina dacă o vedere poate să fie actualizată sau nu.
- **Un tabel virtual (vedere)** este un tabel logic, în sensul că el organizează datele sub forma unor rânduri și coloane, ca orice alt tabel, dar în schimb el nu stochează datele, fiind construit pe baza unor interogări asupra unuia sau mai multor tabele de bază.



Regulile lui Codd - cont.

- De exemplu, să considerăm tabelul :
`salariu(cod_salariat, salariu_brut, zile_totale, zile_lucrate).`
- Pe baza acestui tabel se poate defini vederea
`salariu_r(cod_salariat, salariu_brut, salariu_realizat)`
unde `salariu_realizat` este definit ca
`salariu_realizat = salariu_brut*zile_totale/zile_lucrate`
- Să presupunem că se dorește actualizarea coloanei `salariu_brut` din vedere.



Regulile lui Codd - cont.

- Acest lucru este posibil, datorită faptului că actualizarea se propagă înapoi la coloana din tabelul de bază, producându-se și actualizarea acesteia.
- Pe de altă parte, nu este posibilă actualizarea coloanei `salariu_realizat`, datorită faptului că schimbarea valorii acesteia s-ar putea produce datorită schimbării valorilor mai multor coloane (`salariu_brut`, `zile_totale` sau `zile_lucrate`), SGBD-ul neștiind care din aceste coloane trebuie actualizată în tabelul de bază.
- **Oracle respectă această regulă**, existând un set de reguli care determină dacă o coloană a unei vederi poate sau nu să fie actualizată.



Regulile lui Codd - cont.

- **Regula 7. Regula manipulării datelor:** Un sistem relațional trebuie să ofere posibilitatea procesării tabelelor (de bază sau virtuale) nu numai în operațiile de interogare a datelor cât și în cele de inserare, actualizare și ștergere.
- Aceasta înseamnă că operațiile de manipulare a datelor (inserare, actualizare și ștergere) trebuie să se poată efectua asupra oricărei mulțimi de rânduri dintr-un tabel, pornind de la întregul tabel și terminând cu un singur rând sau cu nici unul.



Regulile lui Codd - cont.

- Deci, un SGBD relațional nu obligă utilizatorul să caute într-un tabel rând cu rând pentru a regăsi, modifica sau șterge informația dorită, deoarece operațiile prin care se manipulează conținutul BD lucrează la nivel de mulțime de rânduri.
- **Limbajul SQL asigură această facilitate** prin instrucțiunile: INSERT cu subinterogare, UPDATE și DELETE.



Regulile lui Codd - cont.

- **Regula 8. Regula independenței fizice a datelor:** Programele de aplicație nu trebuie să depindă de modul de stocare și accesare fizică a datelor.
- Deci un SGBD relațional trebuie să separe complet aspectele de ordin fizic ale datelor (modul de stocare și modul de acces la date) de cele de ordin logic.



Regulile lui Codd - cont.

- De exemplu, dacă un fișier care conține un tabel de date este mutat pe o altă unitate de disc sau îi este schimbat numele, aceasta nu trebuie să aibă vreun efect asupra aplicațiilor care folosesc acel tabel, utilizatorilor fiindu-le transparentă această schimbare.
- În mare, **Oracle respectă această regulă**, deși stocarea fizică a datelor trebuie luată în considerație la proiectarea bazei de date.



Regulile lui Codd - cont.

- **Regula 9. Regula independenței logice a datelor:** Programele de aplicație nu trebuie să fie afectate de nici o restructurare logică a tabelelor BD care conservă datele.
- Deci orice modificare efectuată asupra unui tabel care conservă datele din acesta (de exemplu, dacă un tabel trebuie divizat în două, din rațiuni de creștere a performanțelor) nu trebuie să afecteze funcționarea programelor de aplicație.



Regulile lui Codd - cont.

- **Această regulă este respectată de către Oracle** prin posibilitatea definirii vederilor: dacă un tabel este divizat în 2, atunci se poate crea o vedere care alătură cele 2 tabele, astfel încât această împărțire nu va avea nici un efect asupra aplicației.



Regulile lui Codd - cont.

- **Regula 10. Regula independenței datelor din punctul de vedere al integrității:**
Regulile de integritate a BD trebuie să fie definite în limbajul utilizat de sistem pentru definirea datelor și nu în cadrul aplicațiilor individuale; în plus, aceste reguli de integritate trebuie stocate în dicționarul de date.
- Cu alte cuvinte, restricțiile de integritate trebuie impuse la definirea tabelelor BD și nu în cadrul aplicațiilor care folosesc aceste tabele.



Regulile lui Codd - cont.

- În general, Oracle respectă această regulă, la definirea tabelului (în cadrul comenzii CREATE TABLE) putându-se defini atât restricțiile de integritate structurală (NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY) cât și unele restricții de comportament (CHECK).
Informații despre aceste restricții sunt stocate în dicționarul bazei de date.



Regulile lui Codd - cont.

- **Regula 11. Regula independenței datelor din punctul de vedere al distribuirii:**
Programele de aplicație nu trebuie să fie afectate de distribuirea pe mai multe calculatoare a BD.
- Cu alte cuvinte, BD trebuie să meargă corect indiferent dacă se găsește pe un singur calculator sau este distribuită în mai multe noduri ale unei rețele.
- Această regulă este o extensie a regulii 8, privind independența programelor de aplicație față de modul de stocare fizică a datelor.



Regulile lui Codd - cont.

- Această regulă este în general respectată de Oracle, existând totuși restricții privind accesarea unor obiecte aflate în alt nod al rețelei.
- În plus, în Oracle există posibilitatea replicării locale a tabelelor aflate în alte noduri ale rețelei, evitându-se astfel transmiterea în mod repetat a informațiilor prin rețea.



Regulile lui Codd - cont.

- **Regula 12. Regula privind prelucrarea datelor de către un limbaj de nivel inferior:** Orice limbaj nerelațional folosit pentru accesarea datelor trebuie să respecte aceleași condiții de integritate ca și limbajul relațional de acces.
- De exemplu, dacă sistemul posedă un limbaj de nivel inferior, prin care se accesează datele la nivel de rând și nu, potrivit sistemului relațional, la nivelul mulțimilor de rânduri, acest limbaj nu poate fi folosit pentru evitarea restricțiilor de integritate (integritatea entității, integritatea referențială, restricții de comportament) pe care trebuie să le respecte limbajul procedural de acces la date.



Regulile lui Codd - cont.

- Această regulă este respectată de către Oracle prin faptul că singurul limbaj de accesare a datelor este SQL, care este un limbaj relațional.
- Dacă un SGBD îndeplinește principiile sistemului relațional (folosește ca structuri de date tabele conforme cu modelul relațional, suportă cele două reguli de integritate structurală și operațiile relaționale) și respectă aceste 12 reguli, atunci el poate fi numit relațional.
- Codd rezumă aceste lucruri prin regula zero:



Regulile lui Codd - cont.

- **Regula 0. Regula de bază:** Un SGBD Relațional trebuie să fie capabil să gestioneze BD exclusiv pe baza caracteristicilor sale relaționale.
- Aceasta înseamnă că sistemul trebuie să-și îndeplinească toate funcțiile prin manipulări în care unitatea de procesare să fie tabelul (mulțimea de rânduri), asupra căruia să se efectueze operațiile specifice modelului relațional.
- **Regula 0 nu este respectată în totalitate de nici un SGBD existent pe piață, inclusiv Oracle**, implementarea acestora folosind atât caracteristici relaționale cât și nerelaționale.



Regulile lui Codd - cont.

- Se obișnuiește ca, în conformitate cu tipul de cerințe pe care le exprimă, regulile să fie grupate în 5 categorii, și anume:
 1. reguli de bază: Regula 0 și Regula 12;
 2. reguli structurale: Regula 1 și Regula 6;
 3. reguli privind integritatea datelor: Regula 3 și Regula 10;
 4. reguli privind manipularea datelor: Regula 2, Regula 4, Regula 5 și Regula 7;
 5. reguli privind independența datelor: Regula 8, Regula 9 și Regula 11.
- Trebuie spus că **nici unul dintre SGBD-urile existente în prezent nu satisface în totalitate toate cele 13 reguli ale lui Codd.**



Regulile lui Codd - cont.

- De aceea, în practică nu sunt utilizate regulile lui Codd, fiind formulate în schimb un set de cerințe minimale pe care trebuie să le satisfacă un sistem SGBD pentru a putea fi considerat relațional.



Regulile lui Codd - cont.

- Un SGBD este denumit **minimal relațional**, dacă satisface următoarele condiții:
 - Toate datele din cadrul BD sunt reprezentate prin valori în tabele.
 - Nu există pointeri între tabele observabile de către utilizator.
 - Sistemul suportă operatorii relaționali de proiecție, selecție și compunere naturală, fără limitări impuse de considerente interne.



Regulile lui Codd - cont.

- Un SGBD este denumit **complet relațional** dacă este minimal relațional și satisface în plus următoarele condiții:
 - Sistemul suportă toate operațiile de bază ale algebrei relaționale, fără limitări impuse de considerente interne.
 - Sistemul suportă restricțiile de integritate de bază ale modelului relațional (integritatea entității și integritatea referențială).
- SGDB-ul Oracle este **complet relațional** și chiar se apropie destul de mult de un SGBD relațional ideal, definit prin regulile lui Codd.



Cuprins (Proiectarea bazelor de date)

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

- După mai bine de 2 decenii de folosire a modelului relațional, **proiectarea (designul) BD rămâne încă mai degrabă artă decât știință.**
- Au fost sugerate un număr de metode, dar până în prezent nici una nu este dominantă.
- Pe de altă parte proiectarea bazelor de date trebuie să fie bazată pe considerații practice care stau la baza oricărei activități de procesare a datelor.
- Pentru a crea un design adecvat este necesară o cunoaștere aprofundată a funcționării întreprizei, a modului în care aceasta folosește datele și a sistemului de management al bazelor de date folosit.



Proiectarea bazelor de date - cont.

- Metodele curente de proiectare a BD sunt în general divizate în 3 etape separate:
 - crearea schemei conceptuale,
 - crearea design-ului logic al bazei de date și
 - crearea design-ului fizic al bazei de date.



Proiectarea bazelor de date - cont.

- **Crearea schemei conceptuale.**
 - Aceasta este un design de nivel înalt (incluzând relațiile dintre datele întregului sistem), care descrie datele și relațiile necesare pentru execuția operațiilor necesare, fiind independent de orice model de baze de date.
 - Designul de la acest nivel este foarte general, se realizează într-o perioadă scurtă de timp și prezintă modul în care grupările de date sunt integrate în sistemul de ansamblu.



Proiectarea bazelor de date - cont.

- **Crearea design-ului logic al bazei de date.**
 - În această fază, schema conceptuală este transformată în structuri specifice unui anumit sistem de management al bazei de date.
 - La acest nivel designul este rafinat, sunt definite elemente de date specifice și sunt grupate în înregistrări.
 - În cazul modelului relațional, la sfârșitul acestei etape vom avea un număr de tabele care vor permite stocarea și manipularea corectă a tuturor datelor necesare sistemului.



Proiectarea bazelor de date - cont.

- **Crearea design-ului fizic al bazei de date.**
 - În această etapă designul logic este transformat într- o structură fizică eficientă.
- Primele 2 etape vor fi prezentate pe larg în acest curs. Ce de-a treia va fi abordată într-un curs ulterior, când va fi prezentată organizarea logică a bazei de date Oracle.



Bibliografie

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