CAPITOLUL 3

MODELUL RELAȚIONAL

O problema fundamentala a unui SGBD este modul in care datele sunt organizate in vederea stocarii si exploatarii lor de catre aplicatii. Din punct de vedere istoric, in anii '60 au existat doua modele de organizare a datelor care au fost apoi abandonate din cauza problemelor pe care le generau:

- Modelul ierarhic, folosit de IBM in sistemul IMS (care inca este unul dintre produsele furnizate de aceasta firma), in care organizarea este sub forma arborescenta: nodurile contin date si legaturi ('pointeri') catre nodurile fiu (vezi http://www-306.ibm.com/software/data/ims/
- Modelul retea. In cadrul acestuia inregistrarile sunt structurate sub forma unui graf orientat, fiecare nod putand avea mai multe inregistrari 'tata' si mai multi fii. Au existat mai multe sisteme de gestiune si limbaje de programare dezvoltate pe baza acestui model (de exemplu limbajul COBOL).

Dezavantajul principal al acestor doua modele este ca accesul la o inregistrare necesita navigarea prin arbore sau graf pentru a o localiza. Din acest motiv apar o serie de probleme mai ales legate timpul necesar scrierii de noi programme si a detectarii anomaliilor care pot sa apara in proiectarea bazei de date.

Modelul relational al datelor, folosit in acest moment de majoritatea covarsitoare a sistemelor de gestiune aflate pe piata a fost intrudus in anul 1970 prin articolul lui Edgar Frank Codd "***A relational model for large shared databanks***". Acest model, in care datele sunt stocate sub forma tabelara are o serie de avantaje care au dus la inlaturarea celorlalte modele:

- Datele sunt stocate doar ca valori; nu exista pointeri sau navigare prin date;
- Face posibila dezvoltarea de limbaje de cereri de nivel inalt in care utilizatorul specifica ***ce*** date doreste si nu ***cum*** se ajunge la rezultat, modul in care este calculat acesta fiind in sarcina sistemului de gestiune (exemplu de astfel de limbaj: SQL)
- Furnizeaza o baza solida pentru problemelor de corectitudine a datelor (redundanta, anomalii, etc).
- Permite tratarea problemelor de independenta a datelor (discutate in capitolul 1).
- Este extensibil, putand fi utilizat si pentru modelarea si manipularea de date complexe.

3.1. Elementele de baza ale modelului

3.1.1. Domeniu

Definitie: *Domeniu* (eng. Domain) = o multime de valori avand asociat un nume.

Un domeniu se poate defini fie prin enumerarea elementelor sale fie prin specificarea unor caracteristici definitorii ale acestora.
Exemple:

- Culori = {rosu, galben, albastru, violet, verde}
- Nota = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} sau Nota = $\{n \in N^* \mid n \geq 1 \text{ si } n \leq 10\}$
- Sir40 = {Multimea sirurilor de maxim 40 de caractere}
- Numar = {Multimea numerelor intregi pozitive din intervalul [0, 100000]}

Din teoria multimilor se cunoaste notiunea de ***produs cartezian*** al unor multimi: fiind date ***n*** domenii $D_1, D_2, \ldots, D_n$ produsul lor cartezian este:

$$D_1 \times D_2 \times \ldots \times D_n = \{ (v_1, v_2, \ldots, v_n) \mid vi \in D_i , i = 1, \ldots, n\}$$

Trebuie mentionat ca in sirul de domenii care participa la un produs cartezian unele se poate gasi in mod repetat:

$$PC = Numar \times Sir40 \times Numar \times Numar \times Sir40 \times Sir40$$

3.1.2. Relatie

Definitie: *Relatie* (eng. Relation) = o submultime a unui produs cartezian avand asociat un nume.

Termenul de relatie provide de asemenea din matematica. Un exemplu de relatie apartinand produsului cartezian PC din paragraful urmator este:
Produse = { (101, 'Imprimanta laser', 30, 20, 'Xerox', 'Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti',) , (105, 'Calculator PC', 20, 23, 'IBM', 'Bd. D.Cantemir nr.1, Bucuresti'),
(124, 'Copiator', 10, 20, 'Xerox', 'Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti',) }

Elementele unei relatii sunt denumite in literatura de specialitate ***tupluri*** (engl. tuple). Relatia de mai sus contine doar 3 dintre elementele produsului cartezian din care provine (3 tupluri).

O reprezentare intuitiva pentru relatia de mai sus este si urmatoarea, in care fiecare element al relatiei devine o linie a unei tabele si fiecare coloana corespunde unui domeniu din produsul cartezian de baza:

Produse

101	Imprimanta laser	30	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
105	Calculator PC	20	23	IBM	Bd. D.Cantemir nr.1, Bucuresti
124	Copiator	10	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti

In fapt deci o relatie se reprezinta o tabela care contine date, fiecare coloana avand asociat un anumit tip de date, dat de domeniul din care provine.

3.1.3. Atribut

Deoarece o relatie are o reprezentare tabelara putem vorbi de 'coloană a unei relatii'. In mod obisnuit, intr-o tabela coloanele au un nume.

Definitie: *Atribut* (eng. Attribute) = coloană a unei relatii avand asociat un nume.

Pentru relatia Produse putem fixa de exemplu urmatoarele nume de atribute:

- `IdP` – Codul produsului (nu exista doua produse avand acelasi cod)
- `NumeP` – numele produsului
- `Qty` – Cantitate
- `IdF` – Codul furnizorului (nu exista doi furnizori avand acelasi cod)
- `NumeF` – Numele furnizorului
- `AdresaF` – Adresa furnizorului

Produse

IdP	NumeP	Qty	IdF	NumeF	AdresaF
101	Imprimanta laser	30	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
105	Calculator PC	20	23	IBM	Bd. D.Cantemir nr.1, Bucuresti
124	Copiator	10	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti

3.1.4. Schema unei relatii

Continutul unei relatii (vazuta ca o tabela) poate varia in timp: se pot adauga sau sterge linii sau se pot modifica unele dintre valorile din liniile existente. Ceea ce ramane constanta este structura relatiei: numele relatiei, numarul si tipul atributelor sale. In terminologia relationala structura unei relatiei este denumita si schema relatiei.

Definitie: *Schema unei relatii* (eng. Relation scheme) = numele relatiei urmat de lista atributelor sale si (eventual) de domeniul din care acestea provin.

Exista mai multe modalitati prin care se poate specifica schema unei relatii. In exemplele urmatoare prezentam cateva dintre acestea cu referire la relatia Produse:

```
Produse(IdP, NumeP, Qty, IdF, NumeF, AdresaF)
Produse(IdP: Numar, NumeP: Sir40, Qty: Numar, IdF: Numar,
        NumeF: Sir40, AdresaF: Sir40)
Produse = IdP, NumeP, Qty, IdF, NumeF, AdresaF
```

In cazul prezentarii unora dintre elementele de teorie a bazelor de date relationale se folosesc si notatii de forma:

```
R = ABCDE
```

Cu semnificatia: schema relatiei R contine 5 atribute notate cu A, B, C, D si respectiv E.

3.1.5. Cheia unei relatii

O relatie fiind o multime (de tupluri) nu poate contine elemente duplicat – spre deosebire de exemplu de un tabel Excel unde putem avea dubluri.
Rezulta ca tuplurile pot fi deosebite intre ele prin valorile aflate pe una sau mai multe coloane din relatie.

Definitie: ***Cheia unei relatii*** = multime minimala de atribute ale caror valori identifica in mod unic un tuplu al relatiei respective

Cheia unei relatii este o caracteristica a schemei acesteia si nu este determinata prin inspectarea valorilor aflate la un moment dat in relatie.
In tabela **Produse** cele trei linii existente pot fi identificate unic de valorile de pe 3 atribute (IdP, NumeP si Qty) dar numai IdP este cheie:

- IdP identifica (prin definitie) in mod unic un produs; rezulta ca multimea de atribute { IdP } este cheie (fiind si minimala prin natura sa}.
- Multimea de atribute {IdP, IdF} identifica de asemenea unic fiecare tuplu al relatiei dar nu este cheie nefiind minimala: prin inlaturarea lui IdF multimea ramane in continuare cheie.
- Multimea de atribute {NumeP} nu este cheie: este posibil ca in tabela **Produse** sa avem mai multe linii cu NumeP = 'Imprimanta laser', 'Copiator' sau 'Calculator PC'. Asa cum am mentionat cheia se determina din semnificatia atributelor relatiei si nu din valorile la un moment dat.
- Din acelasi motiv nici una dintre celelalte multimi de atribute ale relatiei **Produse** nu este cheie: fie nu este minimala (in cazul in care il include pe IdP) fie nu identifica unic tuplurile relatiei (pot exista valori duble)

In termeni de tabele rezulta ca nu pot exista doua linii avand aceeasi combinatie de valori pe coloanele care formeaza cheia tabelei respective (proprietate denumita in literatura de specialitate si unicitatea cheii)

O relatie poate avea mai multe chei. Sa ne imaginam o relatie **Studenti** continand date despre studentii romani ai unei facultati:

Studenti (IdStud, NrMatricol, Nume, CNP, SerieCI, NumarCI)

In acest caz avem mai multe chei:

- { IdStud } – pentru ca IdStud este un numar asignat de sistem fiecarei inregistrari, fara repetitii
- { NrMatricol } – pentru ca nu pot exista doi studenti ai unei facultati cu acelasi numar matricol
- { CNP } – pentru ca nu pot exista doi cetateni romani (deci nici doi studenti romani) cu acelasi cod numeric personal
- { SerieCI, NumarCI } – pentru ca nu pot exista doi cetateni romani (deci nici doi studenti romani) cu aceeasi combinatie serie/numar carte de identitate.

Observatie: Orice relatie are cel putin o cheie: deoarece intr-o relatie nu pot exista doua elemente identice, rezulta ca multimea tuturor atributelor relatiei este cheie sau contine cel putin o cheie.

In literatura de specialitate si in sistemele de gestiune a bazelor de date exista trei alte concepte legate de cheie si care vor fi prezentate in paragrafele urmatoare ale acestui capitol:

- Cheie primara (eng. Primary key),
- Cheie straina (eng. Foreign key),
- Supercheie (eng. Superkey).

3.1.6. Valori nule

Uneori, unele elemente ale unei relatii (celule ale tabelei) nu au nici o valoare concreta. Se spune ca in acel loc exista o ***valoare nula***.

Definitie: ***Valoare nula*** (eng. Null value) = o valoare diferita de oricare alta si care modeleaza o informatie necunoscuta sau o informatie inaplicabila.

Exemplul urmator prezinta o relatie Studenti in care exista astfel de valori nule si care au fost simbolizate (pentru a iesi in evidenta) prin <NULL>:

Studenti

IdS	NumeStud	Codfacultate	IdTutor	Medie
1001	**Ionescu Ion**	**03**	**<NULL>**	**9,10**
1002	**Popescu Vasile**	**03**	**1001**	**<NULL>**
1003	**Georgescu Ion**	**<NULL>**	**1001**	**8,40**

- ***Modelarea unei informatii necunoscute***: Codul facultatii studentului Georgescu si media lui Popescu sunt nule pentru ca in momentul incarcarii cu date informatia respectiva, desi existenta in lumea reala, nu era cunoscuta celui care a incarcat datele. La un moment ulterior aceste valori nule vor fi inlocuite cu valori nenule care specifica informatia respectiva.
- ***Modelarea unei informatii inaplicabile***: Sa presupunem ca unii dintre studenti sunt consiliati in activitatea lor de un student de an mai mare, numit si tutor. Codul tutorului unui student este inscris pe coloana IdTutor (de exemplu studentii Popescu si Georgescu il au ca tutor pe studentul Ionescu avand codul 1001). In cazul studentului Ionescu insa valoarea lui IdTutor este nula pentru ca acest student (de an mai mare) nu are la randul sau un tutor, valoarea nula fiind cea corecta in contextul respectiv.

3.1.7. Corectitudinea datelor

Schema unei relatii contine descrierea structurii acesteia dar nu da informatii privind corectitudinea datelor continute in aceasta. Exemplul urmator prezinta o incarcare cu date incorecte pentru tabela Produse, erorile fiind urmatoarele:

- Exista doua produse diferite avand acelasi IdP (101)
- Ultimul produs din tabela nu are asignata o valoare pe coloana IdP
- Aceeasi firma are doua coduri numerice diferite (20 si 22)
- Exista doua firme diferite cu acelasi cod (20)

- Aceeasi firma are doua adrese diferite

Produse

IdP	NumeP	Qty	IdF	NumeF	AdresaF
101	Imprimanta laser	30	20	Xerox	Str. Daniel Danielopolu 4-6, Sector 1, Bucureşti
101	Calculator PC	20	20	IBM	Bd. D.Cantemir nr.1, Bucuresti
	Copiator	10	22	Xerox	Str. Batistei, Bucureşti

Specificarea conditiilor de corectitudine pe care trebuie sa le verifice datele se face astfel:

- In cadrul teoriei bazelor de date relationale, o relatie contine date corecte daca acestea verifica setul de ***dependente functionale*** (sau de alt tip) atasat relatiei respective (dependentele functionale sunt prezentate in capitolul 4)
- In cazul sistemelor de gestiune a bazelor de date existente pe piata, acestea pun la dispozitie mecanisme de verificare numite ***constrangeri de integritate***. Constrangerile de integritate se definesc fie la crearea tabelei fie ulterior si sunt de obicei de cinci tipuri, descrise in continuare. Efectul definirii unor astfel de constrangeri de integritate este acela ca SGBD-ul va rejecta orice operatie care violeaza vreuna dintre constrangerile definite pe tabela respectiva.

a. Constrangeri NOT NULL (valori nenule)

Este o constrangere la nivelul unei coloane dintr-o tabela si specifica faptul ca pe coloana respectiva nu pot sa apara valori nule. In cazul tabelei Produse o astfel de constrangere se poate asocia de exemplu pentru toate coloanele sau doar o parte din acestea. Orice incercare de a adauga o linie care contine valori nule pe acea coloana sau de a modifica o valoare nenula intr-una nula va fi respinsa de sistem.

b. Constrangeri PRIMARY KEY (cheie primara)

Asa cum s-a vazut anterior, o relatie poate avea mai multe chei. In momentul creerii tabelei corespunzatoare relatiei intr-un sistem de gestiune a bazelor de date, una dintre ele poate fi aleasa ca si cheie primara (principala) a tabelei respective. O tabela nu poate avea decat o singura cheie primara, formata din una sau mai multe atribute (coloane) ale acesteia. SGBD-ul creaza automat structuri de cautare rapida (index) pentru cheia primara a tabelei.
Alegerea cheii care devine cheie primara va fi facuta in concordanta cu tipul de aplicatie in care este folosita acea tabela. Pentru exemplul tabelei de la paragraful 3.1.5:

Studenti (IdStud, NrMatricol, Nume, CNP, SerieCI, NumarCI)

avand cheile { IdStud }, { NrMatricol }, { CNP } si { SerieCI, NumarCI } alegerea cheii primare se poate face astfel:

- In cazul in care tabela este folosita intr-o aplicatie de gestiune a datelor privind scolaritatea, se poate alege cheia primara NrMatricol, avand in vedere ca o serie de date privind rezultatele unui student sunt legate de matricola sa (informatie de legatura cu alte tabele)

- In cazul in care tabela este folosita intr-o aplicatie a politiei universitare, alegerea se va face probabil intre CNP si (SerieCI, NumarCI), legatura cu bazele de date de la nivelurile superioare facandu-se dupa aceste informatii.
- In ambele cazuri se poate alege cheia primara IdStud, continand numere unice generate automat de sistem.

O caracteristica a cheii primare a unei tabele este (in majoritatea SGBD-urilor existente) cerinta ca pe coloanele componente nu pot sa apara valori nule.

c. Constrangeri UNIQUE (cheie)

Prin acest tip de constrangere se modeleaza celelalte chei ale tabelei. Pe coloanele unei chei definita cu UNIQUE pot insa sa apara valori nule, unicitatea fiind cerificata doar pentru valorile nenule de pe coloanele cheii respective.
In exemplul anterior, daca s-a ales cheia primara IdStud, pentru celelalte trei chei se pot defini trei constrangeri de acest tip.

d. Constrangeri FOREIGN KEY (cheie straina)

Sunt cazuri in care o multime de coloane ale unei tabele contin valori care exista pe cheia primara a unei alte tabele. Sa consideram o baza de date in care exista urmatoarele doua tabele (cheile lor primare sunt cele subliniate):

Studenti(IdS, NumeStud, CodFacultate, IdTutor, Medie)
Facultati(CodFacult, NumeFacultate, Adresa)

Coloana Codfacultate din tabela Studenti nu este cheie in aceasta tabela (pot exista mai multi studenti cu aceeasi valoare pe aceasta coloana, fiind studenti ai aceleiasi facultati) dar in mod normal contine valori care pot fi doar dintre cele existente pe cheia primara CodFacult din tabela Facultati.
O constrangere activa de acest tip (numita si ***constrangere referentiala***) va avea ca efect respingerea inserarilor/modificarilor in tabela Studenti care ar face ca pe coloana Codfacultate sa apara o valoare care nu este deja in tabela Facultati.
Rezulta implicit ca in momentul incarcarii cu date este necesar sa fie completata intai tabela Facultati si apoi tabela Studenti, altfel operatia de incarcare cu date va esua din cauza violarii acestei constrangeri.

In cazul multor SGBD-uri se poate specifica in constrangere si stergerea automata a inregistrarilor ‘fiu’ in cazul stergerii inregistrarii ‘tata’: la stergerea liniei corespunzatoare unei facultati (din tabela Facultati) se vor sterge automat si liniile din tabela Studenti continand studentii acelei facultati.

Constrangerile referentiale provin de obicei din transformarea asocierilor unare si binare unu-unu si multi-unu (descrisa in capitolul precedent).

e. Constrangeri CHECK (conditie)

Acest tip de constrangere specifica faptul ca valorile unei linii din tabela trebuie sa verifice o conditie (expresie logica).
Exemplu: In tabela:

Studenti(IdS, NumeStud, CodFacultate, IdTutor, Medie)

pe coloana Medie putem defini o constrangere de acest tip specificand ca valoarea (daca exista o valoare nenula) trebuie sa fie din intervalul [0, 10].

3.2. Transformarea diagramelor EA in modelul relational

In procesul de transformare vom pleca de la o diagrama EA si vom obtine trei tipuri de scheme de relatie:

a. Relatii provenite din entitati. Ele contin aceleasi informatii ca si entitatile din care au rezultat.

b. Relatii provenite din entitati si care contin chei straine.

Ele contin pe linga informatiile provenite din entitatile din care au rezultat si atribute care in alte entitati sint identificatori. Este cazul acelor entitati care au asocieri multi-unu si partial din cele care au asocieri unu-unu cu alte entitati.

c. Relatii provenite din asocieri. Este cazul celor care apar din transformarea asocierilor binare multi-multi si a asocierilor de grad mai mare ca doi. Ele contin ca atribute reuniunea identificatorilor entitatilor asociate plus atributele proprii ale asocierilor.

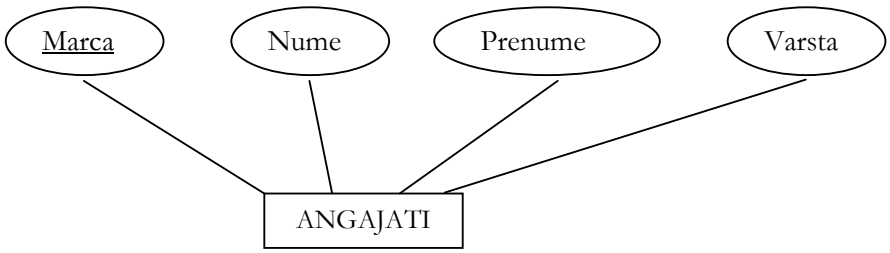
Procesul de transformare are un algoritm foarte precis si este din aceasta cauza pasul care se preteaza cel mai bine pentru crearea de instrumente software care sa-l asiste.

Transformarea entitatilor

Fiecare entitate a diagramei se transforma intr-o schema de relatie avind:

Numele relatiei	=	Numele entitatii
Atributele relatiei	=	Atributele entitatii
Cheia relatiei	=	Identificatorul entitatii

Exemplu: Fie entitatea Angajati de mai jos:



Schema rezultata este Angajati(Marca, Nume, Prenume, Varsta)

Transformarea asocierilor unare si binare unu-unu si multi-unu

Fiecare asociere din aceasta categorie va avea ca rezultat imbogatirea multimii de atribute descriptive ale uneia dintre cele doua scheme rezultate la pasul 2.1 din entitatile asociate, cu cheia celeilalte scheme. Aceste atribute care se adauga sint denumite in prezentarea de fata cheie straina deoarece ele sint cheie dar in alta schema de relatie.

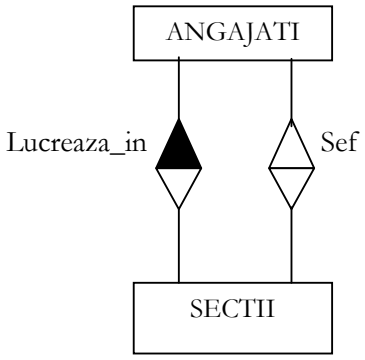
a. In cazul asocierilor multi-unu, se adauga identificatorul entitatii unu in schema rezultata din entitatea multi

b. In cazul asocierilor unu-unu, se adauga identificatorul unei entitati in schema rezultata din transformarea celeilalte. Alegerea schemei in care se face adaugarea se poate face dupa doua criterii:

- fie in acea schema care defineste relatia cu cele mai putine tupluri din cele doua,
- fie pastrindu-se, daca exista, filiatia naturala intre cele doua entitati: identificatorul tatalui se adauga la fiu.

In cazul acestui tip de asocieri, la schema de relatie care primeste cheia straina se ataseaza una sau doua dependente functionale primare (vezi tabelul 1.1.)

Exemplu:



Intre entitatile SALARIATI si SECTII exista doua asocieri: una unu-unu, SEF, care modeleaza faptul ca o sectie este condusa de un salariat (seful de sectie) si una multi-unu, LUCREAZA_IN descrisa in paragrafele anterioare.
Rezultatul transformarii este cel de mai jos. Atribulele aflate dupa punctele se suspensie sint cele adaugate (chei straine).

SALARIATI (Marca, Nume, Prenume, Virsta, ..., Cod_Sectie)
SECTII(Cod_Sectie, Profil, ..., Marca_Sef)

Din asociere LUCREAZA_IN

Din asociere SEF

Pe cimpul Cod_Sectie din relatia SALARIATI se va inregistra pentru fiecare salariat codul sectiei in care acesta lucreaza iar pe cimpul Marca din relatia SECTII se va inregistra pentru fiecare sectie marca sefului de sectie. Pentru asociere SEF s-a aplicat primul criteriu (relatia SECTII va avea mult mai putine inregistrari decit SALARIATI), dar si al doilea criteriu este indeplinit.

Transformarea asocierilor unare si binare multi-multi si a celor de grad mai mare ca doi

Fiecare asociere binara multi-multi si fiecare asociere cu grad mai mare ca doi se transforma intr-o schema de relatie astfel:

Nume relatie	=	Nume asociere
Atribute relatie	=	Reuniunea identificatorilor entitatilor asociate la care se adauga atributele proprii ale asocierii
Cheia relatiei	=	Conform tabelului 1.2.

Grad	Conectivitate	Cheia relatiei provenite din asociere
Unare	multi (E) - multi (E)	Cheie(E) + Cheie(E)
Binare	multi (E1) - multi (E2)	Cheie(E1) + Cheie(E2)
Ternare	unu (E1) - unu (E2) - unu (E3)	Cheie(E1)+Cheie(E2) sau Cheie(E1)+Cheie(E3) sau Cheie(E2)+Cheie(E3) sau
	unu (E1) - unu (E2) - multi (E3)	Cheie(E1)+Cheie(E3) sau Cheie(E2)+Cheie(E3) sau
	unu (E1) - multi (E2) - multi (E3)	Cheie(E2)+Cheie(E3)
	multi (E1) - multi (E2) - multi (E3)	Cheie(E2)+Cheie(E3)+Cheie(E1)

Cheile schemelor de relatie rezultate din asocieri

Legenda:

X + Y: Multimea de atribute X impreuna cu multimea de atribute Y

3.3. Algebra relationala

Inca din primul sau articol in care introduce modelul relational, E.F. Codd propune si un set de operatori pentru lucrul cu relatii. Cum o relatie este o multime de tupluri o parte dintre acesti operatori provin direct din teoria multimilor. Ceilalti operatori, introdusi in aceasta algebra pentru relatii (numita in literature de specialitate ***algebra relationala***) sunt specifici acesteia si au la baza operatii uzuale cu tabele – acestea fiind reprezentarea intuitive pentru relatii.

Dupa aparitia primelor sisteme de gestiune a bazelor de date relationale s-a constatat insa ca aceasta algebra nu inglobeaza o serie de situatii care apar in practica: in cazul executiei unei cereri SQL pot sa apara tabele rezultat in care exista linii duplicat. In plus, in cazul in care pe o tabela din baza de date nu a fost definita o cheie primara, putem sa avem in aceasta mai multe linii identice. Problema liniilor duplicat intra in contradictie cu definitia unei relatii in care nu putem avea doua tupluri identice. Din acest motiv in acest subcapitol prezentam urmatoarele variante de operatori:

- operatori ai algebrei relationale clasice in care pornind de la una sau mai multe relatii obtinem o relatie.
- Operatori ai algebrei pe multiseturi care lucreaza pe asa numitele multiseturi (in engleza ***bags***) care sunt asemanatoare relatiilor dar in care putem avea elemente duplicat.
- Operatori care lucreaza atat pe relatii cat si pe multiseturi. Ei sunt o extensie a algebrei relationale si pe multiseturi si provin din necesitatea de a putea rescrie orice cerere SQL in termeni al algebrei extinse.

3.3.1. Algebra relationala clasica

Exista mai multi operatori in cadrul acestei algebre, unii dintre ei fiind derivati (se pot rescrie in functie de alti operatori). Putem imparti acesti operatori in doua categorii:

a. Operatori derivati din teoria multimilor.
b. Operatori specifici algebrei relationale

Operatori derivati din teoria multimilor

Reuniunea: Fiind date doua relatii R si S, reuniunea lor, notata R ∪ S este o relatie care contine tuplurile care sunt fie in R, fie in S fie in ambele relatii. In rezultatul reuniunii nu apar tupluri duplicat.
Pentru ca aceasta operatie sa poata fi executata cele doua relatii care se reunesc trebuie sa aiba scheme compatibile (acelasi numar de coloane provenind din aceleasi domenii (deci cu acelasi tip de date).

Echivalent SQL: operatorul UNION prin care se pot reuni rezultatele a doua cereri SQL de tip SELECT.

Exemplu:

A	B	C
1	1	2
2	1	3
1	3	2

Relatia R

A	B	C
4	1	2
2	1	3
1	3	2
5	1	7

Relatia S

A	B	C
1	1	2
2	1	3
1	3	2
4	1	2
5	1	7

Relatia R ∪ S

Diferenta: Fiind date doua relatii R si S, diferenta lor, notata R - S este o relatie care contine tuplurile care sunt in R si nu sunt in S.
Si in cazul diferentei cele doua relatii care se reunesc trebuie sa aiba scheme compatibile.

Echivalent SQL: operatorul MINUS prin care se poate face diferenta intre rezultatele a doua cereri SQL de tip SELECT.

Exemplu:

A	B	C
1	1	2
2	1	3
1	3	2

Relatia R

A	B	C
4	1	2
2	1	3
1	3	2
5	1	7

Relatia S

A	B	C
1	1	2

Relatia R - S

Intersectia: Fiind date doua relatii R si S, intersectia lor, notata R ∩ S este o relatie care contine tuplurile care sunt si in R si in S. De asemenea cele doua relatii care se reunesc trebuie sa aiba scheme compatibile.

Echivalent SQL: operatorul INTERSECT prin care se poate calcula intersectia rezultatelor a doua cereri SQL de tip SELECT.

Exemplu:

A	B	C
1	1	2
2	1	3
1	3	2

Relatia R

A	B	C
4	1	2
2	1	3
1	3	2
5	1	7

Relatia S

A	B	C
2	1	3
1	3	2

Relatia R ∩ S

Observatie: Intersectia este un operator derivat. Putem rescrie orice intersectie astfel:

$$R \cap S = R - (R - S)$$

Produsul cartezian: Fiind date doua relatii R si S, produsul lor cartezian, notata $R \times S$ este o relatie ale carei tupluri sunt formate prin concatenarea fiecarei linii a relatiei R cu fiecare linie a relatiei S. Rezulta de aici urmatoarele:

- Numarul de atribute (coloane) ale lui $R \times S$ este egal cu suma numerelor de atribute ale lui R si S
- Numarul de tupluri (linii) ale lui $R \times S$ este egal cu produsul numerelor de tupluri ale lui R si S
- Daca in R si S avem atribute (coloane) cu acelasi nume, in produsul cartezian $R \times S$ vom avea atribute care au acelasi nume. Pentru a le deosebi se prefixeaza numele atributului cu cel al relatiei din care provine (ex.: R.A si S.A, ca in exemplul urmator)

Echivalent SQL:

- In clauza FROM a unei cereri SELECT apar doua (sau mai multe) tabele
- In cazul standardului SQL-3, se poate folosi clauza CROSS JOIN a unei cereri de regasire de date de tip SELECT prin care se poate efectua produsul cartezian a doua tabele.

Exemplu:

A	B	C
1	1	2
2	1	3
1	3	2

Relatia R

A	C	D	E
4	1	2	5
2	1	3	1

Relatia S

Rezultatul produsului cartezian este o relatie avand 7 atribute - coloane (suma dintre 3 si 4) si 6 tupluri - linii (produsul lui 3 cu 3):

R.A	R.B	R.C	S.A	S.C	S.D	S.E
1	1	2	4	1	2	5
1	1	2	2	1	3	1
2	1	3	4	1	2	5
2	1	3	2	1	3	1
1	3	2	4	1	2	5
1	3	2	2	1	3	1

Relatia $R \times S$

Operatori specifici algebrei relationale

Proiectia: Fiind data o relatie R si o multime de atribute ale acesteia $X=A_1, A_2, \ldots A_n$, proiectia lui R pe multimea de atribute X este o relatie care se obtine din R luand doar coloanele din X (in aceasta ordine) si eliminand eventualele tupluri duplicat.
Notatia pentru selectie este urmatoarea:

$$\pi_X(R) \text{ sau } \pi_{A1, A2, \ldots An}(R)$$

Echivalent SQL: Clauza SELECT a unei cereri de regasire de date in care este specificata lista de expresii care da structura de coloane a rezultatului.

Exemplu: din relatia R de mai jos dorim sa calculam $\pi_{B, C, E}(R)$

A	B	C	D	E
1	**1**	**2**	**1**	**3**
2	1	2	1	3
2	**7**	**4**	**4**	**1**
2	**3**	**9**	**2**	**1**
1	**3**	**7**	**4**	**1**
1	3	9	2	1

Relatia R

B	C	E
1	2	3
7	4	1
3	9	1
3	7	1

Rezultatul proiectiei $\pi_{B, C, E}(R)$
Observam ca s-au eliminat doua linii duplicat din rezultat (cele provenite din liniile 2 si 6).

Nota: in multimea de atribute pentru o proiectie poate sa apara toate atributele relatiei. In acest caz se obtine o relatie cu acelasi continut cu cea initiala dar in care coloanele sunt permutate:

A	B	C	D	E
1	1	2	1	3
2	1	2	1	3
2	7	4	4	1
2	3	9	2	1
1	3	7	4	1
1	3	9	2	1

Relatia R

B	C	A	E	D
1	2	1	3	1
1	2	2	3	1
7	4	2	1	4
3	9	2	1	2
3	7	1	1	4
3	9	1	1	2

Rezultatul proiectiei $\pi_{B, C, A, E, D}(R)$

Selectia (numita uneori restrictia): Fiind data o relatie R si o expresie logica F (o conditie), selectia lui R in raport cu F este o relatie care se obtine din R luand doar liniile care verifica expresia logica F.
Notatia pentru selectie este urmatoarea:

$$\sigma_F(R)$$

Echivalent SQL: Clauza WHERE a unei cereri de regasire de date de tip SELECT pe care se scrie conditia pe care trebuie sa o indeplineasca liniile pentru a trece mai departe spre rezultat.

Exemplu: din relatia R de mai jos dorim sa calculam $\sigma_{B+1 > A+C}(R)$:

A	B	C	D	E
1	1	2	1	3
2	1	2	1	3
2	7	4	4	1
2	3	9	2	1
1	3	7	4	1
1	3	9	2	1

Relatia R

A	B	C	D	E
2	7	4	4	1

Rezultatul selectiei $\sigma_{B+1 > A+C}(R)$

Joinul general (numit si theta-join sau θ-join): fiind dare doua relatii R si S, joinul lor (notat $R\bowtie_F S$) se obtine din produsul cartezian al relatiilor R si S urmat de o selectie dupa conditia F (numita si ***conditie de join***). Denumirea de theta-join este folosita din motive istorice, simbolul θ fiind folosit initial pentru a desemna o conditie.
Rezulta ca:

$$R\bowtie_F S = \sigma_F(R \times S)$$

Sa luam un exemplu concret pentru exemplificarea acestui operator: Sa consideram ca avem doua relatii, STUD si SPEC avand schemele:

STUD(Matr, Nume, CodSpec, Media)
SPEC(CodS, NumeS)

Cele doua relatii au urmatorul continut:

Matr	Nume	CodSpec	Media
101	Ionescu Ion	10	8
102	Popescu Maria	11	9
302	Georgescu Vasile	10	9,50

Relatia STUD

CodS	NumeS
10	Calculatoare si Tehnologia Informatiei
11	Automatica si Informatica Industriala

Relatia SPEC

Sa consideram urmatoarele joinuri:

- STUD$\bowtie_{STUD.CodSpec=SPEC.CodS}$SPEC
- STUD$\bowtie_{STUD.CodSpec>SPEC.CodS}$SPEC

Rezultatul celor doua joinuri este urmatorul:

Pentru STUD$\bowtie_{STUD.CodSpec=SPEC.CodS}$SPEC

Matr	Nume	CodSpec	Media	CodS	NumeS
101	Ionescu Ion	10	8	10	Calculatoare si Tehnologia Informatiei
102	Popescu Maria	11	9	11	Automatica si Informatica Industriala
302	Georgescu Vasile	10	9,50	10	Calculatoare si Tehnologia Informatiei

In cazul in care conditia de join este una de egalitate, joinul se mai numeste si **echijoin.** In restul cazurilor se foloseste sintagma non-echijoin.

- Pentru STUD$\bowtie_{STUD.CodSpec>SPEC.CodS}$SPEC

Matr	Nume	CodSpec	Media	CodS	NumeS
102	Popescu Maria	11	9	10	Calculatoare si Tehnologia Informatiei

Echivalent SQL: In clauza FROM a unei cereri de regasire de tip SELECT apar tabelele care participa la join si in clauza WHERE se pune conditia de join, conectata cu AND de celelalte conditii care eventual sunt necesare in cererea respectiva.

Join natural: Joinul natural pentru doua relatii R si S (notat R⋈S)se obtine facand joinul celor doua relatii dupa conditia "coloanele cu aceeasi semnificatie au valori egale" si eliminand prin proiectie coloanele duplicat (cele dupa care s-a facut joinul).

Echivalent SQL: Clauza NATURAL JOIN din sintaxa SQL-3. Observatie: deoarece SGBD-ul nu cunoaste semnificatia coloanelor, conditia de join implicita in acest caz este "coloanele cu acelasi nume au valori egale"

Exemplu: In cazul celor doua tabele de mai sus, STUD si SPEC, joinul lor natural va fi asemanator cu echijoinul anterior, lipsind insa coloana duplicat SPEC.CodS (care are aceleasi valori ca si coloana STUD.CodSpec)

Matr	Nume	CodSpec	Media	NumeS
101	**Ionescu Ion**	**10**	**8**	**Calculatoare si Tehnologia Informatiei**
102	**Popescu Maria**	**11**	**9**	**Automatica si Informatica Industriala**
302	**Georgescu Vasile**	**10**	**9,50**	**Calculatoare si Tehnologia Informatiei**

Join extern: Asa cum se vede din nonechijoinul de mai sus (cel dupa conditia STUD.CodSpec>SPEC.CodS), in cazul in care o linie a unei tabele, oricare ar fi concatenarea ei cu o alta linie din cealalta tabela, nu indeplineste conditia de join, linia respectiva nu are corespondent in rezultat. Este cazul liniilor studentilor de la specializarea 10 si al liniei specializarii 11.
In unele cazuri se doreste insa ca aceste linii sa apara in rezultat, cu valori nule pe coloanele din cealalta tabela. Aceasta operatie poarta numele de join extern (in engleza outer join). Cum la un join participa doua tabele, pot exista trei tipuri de join extern:

- Join extern stanga (left outer join), in care in rezultat apar toate liniile tabelei din stanga operatorului. Notatia este: $R \rhd\!\!\circ\!\!\lhd_L S$.
- Join extern dreapta (right outer join), in care in rezultat apar toate liniile tabelei din dreapta operatorului. Notatia este: $R \rhd\!\!\circ\!\!\lhd_R S$.
- Join extern complet (full outer join), in care in rezultat apar toate liniile tabelei din stanga si din dreapta operatorului. Notatia este: $R \rhd\!\!\circ\!\!\lhd S$.

De notat ca in rezultatul joinului extern sunt intotdeauna continute tuplurile (liniile) din rezultatul joinului general dupa aceeasi conditie.

Exemple:

- Joinul extern stanga $STUD \rhd\!\!\circ\!\!\lhd_{L\ (STUD.CodSpec>SPEC.CodS)} SPEC$

Matr	Nume	CodSpec	Media	CodS	NumeS
102	**Popescu Maria**	**11**	**9**	**10**	**Calculatoare si Tehnologia Informatiei**
101	**Ionescu Ion**	**10**	**8**	**NULL**	**NULL**
302	**Georgescu Vasile**	**10**	**9,50**	**NULL**	**NULL**

- Joinul extern dreapta $STUD \rhd\!\!\circ\!\!\lhd_{D\ (STUD.CodSpec>SPEC.CodS)} SPEC$

Matr	Nume	CodSpec	Media	CodS	NumeS
102	**Popescu Maria**	**11**	**9**	**10**	**Calculatoare si Tehnologia Informatiei**
NULL	**NULL**	**NULL**	**NULL**	**11**	**Automatica si Informatica Industriala**

- Joinul extern complet STUD$\triangleright^{o}\triangleleft_{(STUD.CodSpec>SPEC.CodS)}$SPEC

Matr	Nume	CodSpec	Media	CodS	NumeS
102	Popescu Maria	11	9	10	Calculatoare si Tehnologia Informatiei
101	Ionescu Ion	10	8	NULL	NULL
302	Georgescu Vasile	10	9,50	NULL	NULL
NULL	NULL	NULL	NULL	11	Automatica si Informatica Industriala

Semijoin: Fie doua relatii R si S. Atunci semijoinul lui R in raport cu S (notat $R \propto S$) este o relatie care contine multimea tuplulilor lui R care participa la joinul natural cu S. Semijoinul este un operator derivat. Putem scrie ca:

$$R \propto S = \pi_R (R \bowtie S)$$

Semijoinurile pot fi folosite in optimizarea cererilor de regasire in baze de date distribuite.

3.3.2. Operatori folositi pentru multiseturi

Asa cum am spus anterior, in practica bazelor de date intr-o tabela sau un rezultat al unei cereri de regasire de date pot sa apara linii duplicat. In acest caz nu mai putem vorbi de relatii (care nu permit tupluri duplicat) ci de multiseturi (eng. ***bags***). Prezentam pe scurt efectul unora dintre operatorii de mai sus aplicati multiseturilor.

Reuniunea: Efectul este asemanator cu al reuniunii din algebra relationala dar din rezultatul final nu se elimina duplicatele.

A	B	C
1	1	2
1	1	2
1	3	2

Multiset R

A	B	C
1	3	2
2	1	3

Multiset S

A	B	C
1	1	2
1	1	2
1	3	2
1	3	2
2	1	3

Multiset R $\cup$ S

Intersectia, diferenta, produsul cartezian, selectia, joinul, joinul natural, joinul extern: acelasi mod de calcul ca si in cazul relatiilor dar:

- Multiseturile operand pot sa contina linii duplicat
- Din rezultat nu se elimina liniile duplicat

Observatie: in cazul acestor operatii nu pot aparea linii duplicat decat daca operanzii contin linii duplicat.

Proiectia: Acelasi mod de calcul ca si in cazul relatiilor dar la final nu eliminam liniile duplicat.

A	B	C	D	E
1	1	2	1	3
2	1	2	1	3
2	7	4	4	1
2	3	9	2	1
1	3	7	4	1
1	3	9	2	1

Multiset R

B	C	E
1	2	3
1	2	3
7	4	1
3	9	1
3	7	1
3	9	1

Rezultatul proiectiei $\pi_{B, C, E}$ (R) pentru multisetul R
Observam ca nu s-au eliminat liniile duplicat

3.3.3. Operatori pentru relatii si multiseturi

(Nota: acest subcapitol este redactat doar sumar. In scurt timp fiecare operator va fi prezentat cu mai multe detalii)

Redenumirea: Exista doua modalitati de a face redenumirea tabelelor si/sau coloanelor:

a. **Operatorul de redenumire ρ** permite atat redenumirea relatiilor/multiseturilor cat si a atributelor acestora:

Fiind data o relatie R, putem obtine o alta relatie $S = \rho_{S(A1, A2, \ldots, An)}$ care are acelasi continut ca si R dar atributele se numesc A1, A2, … , An.

Echivalent SQL: Aliasurile de coloana si de tabela folosite in clauzele SELECT, respectiv FROM dintr-o cerere de regasire de tip SELECT

b. **Constructorul →** care permite redenumirea atributelor in rezultatul unei expresii relationale sau pe multiseturi:

Putem redenumi intr-un rezultat un atribut prin constructia:

Nume_vechi → Nume_nou

Exemplu: Fie o relatie R=ABCDE.

In rezultatul proiectiei $\pi_{B\to Nume, C\to Prenume, E\to DataN}$ (R) atributele nu sunt B, C si E ci Nume, Prenume si DataN

Echivalent SQL: aliasul de coloana folosit in clauza SELECT a unei cereri de regasire.

Eliminare duplicate: Acest operator se poate aplica doar pe multiseturi (relatiile nu contin tupluri duplicat). Efectul este eliminarea duplicatelor din multiset. Notatia operatorului este urmatoarea:

Fiind dat un multiset R, δ(R) este un multiset fara duplicate (deci o relatie)

Echivalent SQL: SELECT DISTINCT dintr-o cerere de regasire de tip SELECT
Exemplu:

A	B	C
1	2	3
1	2	3
7	4	1
3	9	1
3	7	1
3	9	1

Multisetul R

A	B	C
1	2	3
7	4	1
3	9	1
3	7	1

Multisetul (relatia) δ(R)

Grupare: Forma operatorului de grupare este urmatoarea:

$$\gamma_{\text{Lista\_atribute\_si\_functii\_statistice}}(R)$$

- Atributele din lista sunt criterii de grupare. Ele apar in rezultatul returnat de operator
- Functiile statistice din lista (ex.: MIN, MAX, SUM, AVG, COUNT) se calculeaza la nivelul fiecarui grup si de asemenea apar in rezultatul operatorului

Acest operator se poate aplica atat relatiilor cat si multiseturilor.
Echivalent SQL: Functii statistice si grupare cu GROUP BY

Un exemplu in acest sens este edificator: In cazul relatiei STUD anterioare

$$\gamma_{\text{CodSpec, Count(*)}\rightarrow\text{NrStud, AVG(Medie)}\rightarrow\text{Medie}}(\text{STUD})$$

va returna o relatie avand urmatorul continut:

CodSpec	NrStud	Medie
10	2	8,75
11	1	9,00

Sortare: Forma operatorului de sortare este urmatoarea:

$$\tau_{\text{Lista\_atribute}}(R)$$

Efectul este sortarea relatiei sau multisetului R in functie de atributele din lista. Cum atat in cazul relatiilor cat si a multiseturilor nu este presupusa o relatie de ordine, acest operator practic nu modifica argumentul (doar rearanjeaza elementele). El are sens doar atunci cand este ultimul aplicat unei expresii .
Echivalent SQL: Clauza ORDER BY dintr-o cerere de regasire de tip SELECT

Proiectie extinsa: Acest operator este analog proiectiei obisnuite dar permite atribute (coloane) calculate pentru rezultatul unei expresii pe relatii sau multiseturi. Forma sa este urmatoarea:

$$\pi_{\text{Expresie1, Expresie2, … Expresien}}(R)$$

Observatie: in functie de rezultatul dorit (relatie sau multiset) dupa ce se calculeaza rezultatele duplicatele se elimina sau nu se elimina.
Echivalent SQL: Ca si in cazul proiectiei obisnuite, acest operator este implementat prin clauza SELECT a unei cereri de regasire a informatiei

Exemplu: Pentru expresia:

$$\pi \text{ Nume, } _{\text{Medie*2}\rightarrow\text{Dublu}} (\text{STUD})$$

Rezultatul este:

Nume	**Dublu**
Ionescu Ion	**16**
Popescu Maria	**18**
Georgescu Vasile	**19**

3.4. Calcul relational

Pe langa algebra relationala, cererile de regasire a informatiei intr-o baza de date relationala pot fi exprimate si prin calcul relational pe tupluri (CRT) sau calcul relational pe domenii (CRD). In acest paragraf sunt prezentate pe scurt aceste doua modalitati de exprimare a cererilor.

3.4.1. Calcul relational pe tupluri

In calcului relational pe tupluri o cerere se exprima printr-o ***expresie*** de forma:

$$\{ t \mid \psi(t) \}$$

Unde t este o ***variabila tuplu*** iar ψ o ***formula***. Semnificatia expresiei este "multimea tuturor tuplurilor t care verifica formula ψ".
Formula este compusa din elemente (numite si atomi) care pot fi de trei tipuri:

- Elemente de tip R(s) unde R este un nume de relatie iar s o variabila tuplu. Semnificatia este "s este un tuplu din R"
- Elemente de tip s[i] θ v[j] unde s si v sunt variabile tuplu iar θ un operator prin care se poate compara componenta i a variabilei tuplu s cu componenta j a variabilei tuplu v
- s[i] θ a sau a θ s[i] prin care componenta i a variabilei tuplu s se compara cu constanta a.

Pe baza acestor atomi se poate defini recursiv ce este o formula si ce sunt aparitii ***libere*** sau ***legate*** ale variabilelor tuplu:

- Orice atom este in acelasi timp formula. Toate aparitiile unei variabile tuplu intr-un atom sunt aparitii libere
- Daca ψ si ϕ sunt doua formule, atunci $\psi \vee \phi$, $\psi \wedge \phi$ si $\neg\psi$ sunt formule cu semnificatia ψ sau-logic ϕ, ψ si-logic ϕ" si respectiv "not ψ". Aparitiile de variabile tuplu sunt libere sau legate in aceste formule dupa cum ele sunt libere

sau legate in componentele acestora. Este permis ca o aceeasi variabila tuplu sa aiba o aparitie libera in ψ si o alta legata in ϕ

- Daca ψ este o formula atunci si $(\exists s)(\psi)$ este formula. Aparitiile variabilei tuplu s care sunt libere in ψ sunt legate in $(\exists s)(\psi)$. Celelalte aparitii de variabile tuplu din ψ raman la fel (libere sau legate) in $(\exists s)(\psi)$. Semnificatia acestei formule este urmatoarea: exista o valoare concreta a lui s care inlocuita in toate aparitiile libere din ψ face ca aceasta sa fie adevarata.
- Daca ψ este o formula atunci si $(\forall s)(\psi)$ este formula. Aparitiile variabilei tuplu s care sunt libere in ψ sunt legate in $(\forall s)(\psi)$. Celelalte aparitii de variabile tuplu din ψ raman la fel (libere sau legate) in $(\forall s)(\psi)$. Semnificatia acestei formule este urmatoarea: orice valoare concreta a lui s pusa in locul aparitiilor libere ale acestuia din ψ face ca ψ sa fie adevarata.
- Parantezele pot fi folosite in formule dupa necesitati. Precedenta este: intai comparatiile, apoi $\exists$ si $\forall$ si in final $\neg$, $\wedge$, $\vee$ (in aceasta ordine)

Exemple de expresii si formule:

- Expresia $\{t \mid R(t) \vee S(t)\}$ este echivalenta reuniunii a doua relatii din algebra relationala. Analog $\{t \mid R(t) \wedge S(t)\}$ reprezinta intersectia a doua relatii.
- Expresia pentru proiectia lui R pe atributele $i_1, i_2, \ldots, i_k$ se poate scrie astfel:
  $$\{ t^{(k)} \mid (\exists u)(R(u) \wedge t[1] = u[i_1] \wedge t[2] = u[i_2] \wedge \ldots \wedge t[k] = u[i_k]) \}$$
- Formula $(\exists s)(R(s))$ spune ca relatia R este nevida

Din pacate unele din expresiile scrise in calcul relational pe tupluri duc la rezultate infinite. De exemplu: daca R este o relatie finita expresia $\{t \mid R(t)\}$ este de asemenea finita dar expresia $\{t \mid \neg R(t)\}$ este infinita (exista o infinitate de tupluri care nu apartin lui R).
Pentru a evita astfel de rezultate s-au introdus asa numitele ***expresii sigure***. Pentru definirea lor este necesara definirea unui alt concept si anume ***domeniul*** unei formule:

Definitie: Daca ψ este o formula atunci domeniul sau, notat cu DOM(ψ) este multimea tuturor valorilor care fie apar explicit in ψ sau sunt componente ale tuplurilor relatiilor prezente in ψ. Cum orice relatie este finita rezulta ca si domeniul oricarei formule este finit.

Exemplu: Fie formula $\psi = R(t) \wedge t[1] > 100$ care reprezinta conditia pentru o selectie din R dupa conditia "valoarea pe prima coloana este mai mare decat 100". Atunci:

DOM(ψ) = { 100 } $\cup$ {multimea valorilor care apar in tuplurile lui R }

Definitie: O expresie ψ se zice ca este sigura daca rezultatul sau este compus doar din valori apartinand lui DOM(ψ).

Conform acestei definitii:

- expresiile $\{t \mid R(t)\}$, $\{t \mid R(t) \wedge t[1] > 100\}$, $\{t \mid R(t) \vee S(t)\}$, $\{t \mid R(t) \wedge S(t)\}$ sau $\{ t^{(k)} \mid (\exists u)(R(u) \wedge t[1] = u[i_1] \wedge t[2] = u[i_2] \wedge \ldots \wedge t[k] = u[i_k]) \}$ sunt sigure
- expresiile $\{t \mid \neg R(t)\}$ sau $\{t \mid \neg R(t) \wedge \neg S(t)\}$ nu sunt sigure.

In literatura de specialitate se poate gasi demonstratia faptului ca expresiile sigure din CRT sunt echivalente cu expresii din algebra relationala si reciproc.

3.4.2. Calcul relational pe domenii

In calculul relational pe domenii nu avem variabile tuplu ci variabile de domeniu, ele constituind elementele care formeaza tuplurile. In acest caz rescriem regulile de formare pentru o formula astfel:

- Un atom poate fi:
  - $R(x_1, x_2, \ldots, x_n)$ unde R este o relatie iar x_i sunt variabile de domeniu sau constante
  - $x\ \theta\ y$ unde x si y sunt variabile de domeniu sau constante iar θ este in continuare un operator de comparatie.
- Formulele din CRD sunt construite analog cu cele din CRT utilizand de asemenea $\neg, \wedge, \vee$ precum si $\exists, \forall$.
- Notiunile de aparitie libera sau legata a unei variabile de domeniu sunt analoge cu cele din CRT
- Analog cu CRT se definesc: domeniul unei variabile de domeniu DOM(x) si expresii sigure in CRD.

Exemple de expresii:

- Reuniunea a doua relatii R si S: $\{x_1x_2\ldots x_n \mid R(x_1x_2\ldots x_n) \vee S(x_1x_2\ldots x_n)\}$
- Intersectia a doua relatii R si S: $\{x_1x_2\ldots x_n \mid R(x_1x_2\ldots x_n) \wedge S(x_1x_2\ldots x_n)\}$
- Selectia dupa conditia "valoarea pe prima coloana este mai mare decat 100:

$$\{x_1x_2\ldots x_n \mid R(x_1x_2\ldots x_n) \wedge x_1 > 100\}$$

Toate expresiile de mai sus sunt sigure. Analog cu CRT, expresiile:

$\{x_1x_2\ldots x_n \mid \neg R(x_1x_2\ldots x_n)\}$ si $\{x_1x_2\ldots x_n \mid \neg R(x_1x_2\ldots x_n) \wedge \neg S(x_1x_2\ldots x_n)\}$ nu sunt sigure.

In literatura de specialitate se poate gasi demonstratia faptului ca expresiile sigure din CRD sunt echivalente cu expresii din algebra relationala si reciproc.