6. 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 $R \bowtie S$)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: $R \rhd^{o} \lhd_{L} S$.
- Join extern dreapta (right outer join), in care in rezultat apar toate liniile tabelei din dreapta operatorului. Notatia este: $R \rhd^{o} \lhd_{R} S$.
- Join extern complet (full outer join), in care in rezultat apar toate liniile tabelei din stanga si din dreapta operatorului. Notatia este: $R \rhd^{o} \lhd S$.

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

Exemple:

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

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 $STUD \rhd^{o} \lhd_{D\ (STUD.CodSpec>SPEC.CodS)} SPEC$

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 $\pi_{B\rightarrow Nume, C\rightarrow Prenume, E\rightarrow DataN}$ (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:

$$\pi_{\text{Expresie1, Expresie2, \ldots Expresie}_n}(R)$$

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 θ 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.