13. OPTIMIZAREA CERERILOR și ALGEBRA RELAȚIONALĂ

I.OBIECTIVE

1. Studiul operatorilor algebrei relationale
2. Operații pe date modelate cu operatorii algebrei relaționale
3. Planuri de executie si optimizarea interogarilor

II.FUNDAMENTARE TEORETICĂ

Interogarea (query) este operaţia prin care se obţin datele dorite dintr-o bază de date, selectate conform unor criterii. Deoarece această operaţie este cea mai importantă operaţie, limbajele de manipulare a datelor sunt denumite şi *limbaje de interogare.* Limbajul de interogare este un limbaj în care un utilizator solicită informaţii din baza de date (BD). De obicei, limbajele de interogare sunt de nivel mai înalt decât limbajele standard de programare. Limbajele de interogare pot fi procedurale sau ne-procedurale, astfel în limbajele procedurale, utilizatorul indică sistemului succesiunea de operaţii asupra bazei de date pentru a determina rezultatul dorit, iar în limbajele ne-procedurale, utilizatorul descrie rezultatul dorit, fără a indica procedura prin care acesta este obţinut. Cele mai multe sisteme relaţionale de baze de date folosesc un limbaj de interogare în care sunt prezente elemente ale ambelor abordări, atât procedurală, cât şi ne-procedurală (SQL, QBE, Datalog.)..

Pentru formularea conceptuală a interogărilor în bazele de date relaţionale s-au dezvoltat două limbaje abstracte de interogare: algebra relațională şi calculul relațional. Acestea sunt limbaje matematice, formale, ambele asociate cu modelul relaţional de date. Algebra relaţională este un limbaj procedural, pe când calculul relaţional pe tupluri şi calculul relaţional pe domenii sunt limbaje ne-procedurale. Ambele familii de limbaje sunt concise şi formale, fără a poseda „cadrul sintactic" al limbajelor comerciale de interogare; algebra relaţională şi calculul relaţional sunt însă limbaje care pun în evidenţă foarte bine tehnicile principale folosite în *procesul găsirii şi extragerii informaţiei din BD.*

Algebra relaţională oferă mijloace puternice de a construi relaţii noi din alte relaţii date. Atunci când relaţiile date sunt reprezentate de informaţii memorate, relaţiile construite cu mijloacele algebrei pot fi răspunsuri la fraze de interogare asupra acestor informaţii. Cunoaşterea limbajului abstract de interogare bazat pe *algebra relaţională* este obligatorie pentru înţelegerea aprofundată a modului de execuţie a interogărilor. Orice algebră permite construirea de expresii prin aplicarea unor **operatori** asupra unor **operanzi atomici** sau asupra altor expresii algebrice.

Operaţiile din algebra relaţională tradiţională (sau „clasică") se pot structura în patru clase:

a) operaţiunile specifice teoriei mulţimilor (reuniune, intersecţie, diferenţă), dar aplicate asupra relaţiilor;
b) operaţiunile care îndepărtează părţi ale unei relaţii (selecţie, proiecţie);
c) operaţiunile care combină tuplurile a două relaţii (produs cartezian, joncţiuni/joinuri)
d) operaţiunea prin care sunt atribuite nume noi unor parti din relații– redenumirea atributelor relaţiei

O proprietate fundamentală în algebra relaţională constă în faptul că fiecare operator acceptă instanţele unei (sau a două) relaţii în calitate de argumente şi întoarce ca rezultat o altă **instanţă de relaţie**. Această proprietate permite folosirea *compusă* (compunerea) operatorilor pentru a forma *fraze de interogare complexe*. O astfel de frază de interogare corespunde unei expresii algebrice relaţionale, care se defineşte **recursiv** ca fiind o relaţie, un operator algebric unar aplicat unei singure expresii, sau ca un operator algebric binar aplicat la două expresii.

2.1.OPERAȚII PE MULȚIMI APLICATE RELAȚIILOR

Reuniunea

Fie r, s relaţii. Reuniunea este t , $r \cup s$, unde t = { tupluri t_i, a. î. $t_i \in r, \forall t_i \in r$ }.
Condiţii: r, s au mulţimi identice de atribute, cu aceleaşi domenii de valori.

Intersecţia

$t = r \cap s$, unde t = { tupluri t_i, a. î. $t_i \in r \wedge t_i \in s$ }.

Diferenţa

$t = r - s$, unde t = { tupluri t_i, a. î. $t_i \in r \wedge t_i \notin s$ }.

$t = s - r$, unde t = { tupluri t_i, a. î. $t_i \in s \wedge t_i \notin r$ }.

Observăm că $r - s \neq s - r$.

Proiecţia

Fie relaţia r cu atributele $A_1, A_2, ..., A_n$. Fie, de asemenea, atributele $A_1, A_2, ..., A_k$, a. î. { $A_1, A_2, ..., A_k$ } ⊂ { $A_1, A_2, ..., A_n$ }. Atunci, proiecţia relaţiei r pe atributele $A_1, A_2, ..., A_k$ (sau pe coloanele acestor atribute), notată cu $\prod_{A1, A2, ..., Ak}(r)$, este relaţia obţinută din r prin extragerea coloanelor atributelor $A_1, A_2, ..., A_k$.

Notă. Proiecţia se poate defini formal ca fiind relaţia:

$\prod_{i1,i2, ..., ik}$ = { tupluri t_i, a. î. $t_i = (a_1, a_2, ..., a_k)$, iar în r ∃ tuplul $(b_1, b_2, ..., b_n)$, $a_j = b_j$ pentru j = 1... k } Mai sus a_j, respectiv b_j, sunt valori ale atributelor corespunzătoare din mulţimile { $A_1, A_2, ..., A_k$ } şi, evident, { $A_1, A_2, ..., A_n$ }. Simbolurile i_1, i_2, etc. reprezintă coloane din r.

Selecţia

Prin definiţie, operaţia de selecţie aplicată unei relaţii r, notată cu $\sigma_F(r)$, constă în extragerea din r a acelor tupluri care îndeplinesc clauza (formula) F. Schema relaţiei obţinute este aceeaşi cu schema relaţiei r, atributele fiind aranjate – prin convenţie – în aceeaşi ordine. Operanzii conţinuţi în clauza F sunt constante sau atribute din schema relaţiei r. Operatorii sunt fie operatori aritmetici uzuali (de comparaţie), fie operatori logici.

Produsul cartezian

Fie relaţiile r şi s de arităţi R_1, R_2. Fie în r şi s tuplurile $(r_{i1}, r_{i2}, ..., r_{iK1})$, respectiv $(s_{i1}, s_{i2}, ..., s_{ik2})$. Formal, produsul cartezian $t = r \times s$ al relaţiilor r şi s se defineşte prin:

t = { tupluri t_i, a. î. $t_i = (r_{i1}r_{i2}...r_{iK1}s_{i1}s_{i2}...s_{iK2})$, cu $(r_{i1}r_{i2}...r_{iK1}) \in r$ şi $(s_{i1}s_{i2}...s_{iK2}) \in s$ }.

Situaţie particulară – cea în care relaţiile operand *au atribute comune*.

Joncţiunea „teta”

Joncţiunea „teta” este o operaţie compusă, care implică efectuarea unui *produs cartezian şi a unei selecţii*. Fie relaţiile r şi s, precum şi o clauză F care conţine ca operanzi constante şi atribute din schemele relaţiilor, iar ca operatori – operatorii aritmetici uzuali şi operatorii logici.

Notaţia folosită *r* THETAJOIN *s*
pentru reprezentarea operaţiei joncţiune „teta”
Conform definiţiei, joncţiunea „teta” este efectuată în următoarea ordine:

a) se calculează produsul cartezian r × s;
b) din relaţia rezultată sunt extrase *prin selecţie* tuplurile care satisfac clauza F.

Joncţiunea naturală

Fie relaţiile r şi s ale căror scheme conţin un număr de atribute comune. Joncţiunea naturală, se calculează în două etape:

a) mai intâi, este determinat produsul cartezian r × s;
b) apoi, asupra produsului cartezian obţinut, este efectuată o operaţiune de selecţie, prin extragerea tuplurilor care conţin acelaşi valori ale atributelor comune din schemele relaţiilor incidente r şi s;
c) în final, sunt eliminate coloanele redundante rezultate.

Redenumirea schemelor de relaţie

Fie schema de relaţie R, cu atributele { A_1, A_2, ..., A_n }. Pentru a schimba numele schemei din R în S, dar cu aceleaşi atribute, folosim operatorul ρ, cu sintaxa (în algebra relatională): $\rho_{S(A1,A2,...,An)}(R)$.
Rezultatul este exact aceeaşi relaţie, dar cu numele S.

Majoritatea limbajelor de interogare moderne se bazează pe definiţiile operaţiunilor relaţionale, precum şi pe cele privind tratarea multiset-urilor. Totodată, în practica actuală, limbaje de interogare ca SQL permit efectuarea unor operaţii suplimentare, importante în aplicaţii. Dintre acestea, menţionăm:

1. **Operatorul δ pentru *eliminarea duplicatelor*** – transformă un multiset într-o mulţime , eliminând toate copiile fiecărui tuplu, în afară de una.

2. ***Operatori de agregare*** – ca suma sau media – nu aparţin algebrei relaţionale. Aceşti operatori sunt folosiţi de operatorul de grupare. Operatorii de agregare se aplică atributelor (coloanelor) unei relaţii; de exemplu, suma unei coloane conduce la valaorea unui număr care este suma tuturor valorilor de coloană.

3. ***Gruparea tuplurilor*** se realizează tinând seama de valoarea unui atribut sau de valorile mai multor atribute din tuplurile respective. Ca urmare, mulţimea tuplurilor unei relaţii este partiţionată în „grupuri”. În continuare, coloanelor fiecărui grup li se poate aplica operaţia de agregare, ceea ce permite să fie formulate interogări care nu se pot exprima în algebra relaţională clasică. **Operatorul de grupare, notat cu γ,** combină efectele grupării şi agregării.

4. ***Operatorul de sortare* τ transformă o relaţie într-o listă de tupluri, sortate** potrivit valorii unuia sau mai multor atribute. Operatorul τ trebuie folosit numai ca un pas final al unei successiuni de operaţiuni, deoarece alţi operatori algebrici relaţionali se aplică numai mulţimilor sau multiset-urilor, dar niciodată listelor.

5. ***Proiecţia extinsă*** **amplifică funcţiile operatorului Π.** În forma s-a generalizată, operatorul Π, în afară de proiectarea unor coloane, *conduce la efectuarea de calcule asupra coloanelor din relaţia-argument*, având ca efect producerea unor coloane noi.

Operatori de agregare

SUM – produce suma unei coloane cu valori numerice.
AVG – produce valoarea medie a unei coloane cu valori numerice.
MIN, MAX – produce cea mai mică, respectiv cea mai mare valoare dintr-o coloană cu valori numerice. Dacă se aplică unei coloane cu valori reprezentate de şiruri de caractere, aceşti operatori produc prima sau ultima valoare din punct de vedere lexicografic (alfabetic).
COUNT – dă numărul de valori (nu neapărat distincte) dintr-o coloană. Aplicat oricărui atribut al unei relaţii, operatorul dă numărul de tupluri ale relaţiei, incluzând duplicatele.

2.2.OPTIMIZAREA FRAZELOR DE INTEROGARE

Obiectivul optimizarii: creşterea vitezei de access la informaţia din baza de date (reducerea timpului de access). Evident de la început: timpul cel mai lung se consumă la prelucrarea interogărilor care conţin produse carteziene, respectiv joncţiuni.

Strategii generale folosite pentru optimizare:
1. Efectuarea selecţiilor cât mai devreme posibil;
2. Înainte de joncţiuni, se prelucrează preliminar fişierele (relaţiile) prin operaţiuni de sortare şi de indexare;
3. Căutarea de sub-expresii care se repetă în expresii mai mari;
4. Aranjarea în „cascadă" a selecţiilor şi proiecţiilor;
5. Combinarea proiecţiilor cu operaţiuni binare adiacente;
6. Combinarea unora din selecţii cu produse carteziene care eventual le preced, pentru a obţine joncţiuni.

Echivalenţa expresiilor
Folosită în procesul transformării expresiilor (algebrice relaţionale). În fapt, prelucrarea frazelor de interogare este iniţiată de un procesor de cereri (interogări), a cărui acţiune începe prin construirea unui arbore de derivare destinat evaluării unei expresii algebrice (conţinută implicit în fraza de interogare).

Reguli de echivalenţă a expresiilor:

1. **Comutativitate join și produs cartezian :**

   $E_1 \times E_2 \equiv E_2 \times E_1$; $E_1 >< E_2 \equiv E_2 >< E_1$.

2. **Asociativitate join și produs cartezian:**

   $(E_1 \times E_2) \times E_3 \equiv E_1 \times (E_2 \times E_3)$; $(E_1 >< E_2) >< E_3 \equiv E_1 >< (E_2 >< E_3)$.

3. **Proiecţii în cascadă**:

   $\Pi_{A1...An}(\Pi_{B1...Bm}(E)) \equiv \Pi_{A1...An}(E)$, când fiecare $A_j \in \{B_i\}$.

4. **Selecţii în cascadă**:

   $\sigma_{F1}(\sigma_{F2}(\mathrm{E}))\equiv\sigma_{F1}\wedge_{F2}(E)$.

5. **Comutativitatea selecţiilor cu proiecţiile:**

   $\sigma_F(\Pi_{A1}...{}_{An}(E))\equiv\Pi_{A1}...{}_{An}(\sigma_F(\mathrm{E}))$.

6. **Comutativitatea selecţiei cu produsul cartezian**:

   $\sigma_F(E_1\times E_2)\equiv\sigma_F()E_1\times E_2$, dacă toate atributele din F fac parte din E_1.
   Dacă $F=F_1\wedge F_2$, unde F_1 conţine numai atributele din E_1, iar F_2 conţine numai atribute din E_2, atunci: $\sigma_F(E_1\times E_2)\equiv\sigma_{F1}(E_1)\times\sigma_{F2}(E_2)$.
   Dacă F_1 conţine numai atribute din E_1, iar F_2 conţine atât atribute din E_1, cât şi atribute din E_2: atunci :$\sigma_F(E_1\times E_2)\equiv\sigma_{F2}(\sigma_{F1}(\mathrm{E1})\times E_2)$.

7. **Comutativitatea selecţie – reuniune, respectiv selecţie – diferenţă**:
   $\sigma_F(E_1\cup E_2)\equiv\sigma_F(E_1)\cup\sigma_F(E_2)$, $\sigma_F(E_1-E_2)\equiv\sigma_F(E_1)-\sigma_F(E_2)$.

8. **Comutativitatea proiecţie – produs cartezian**:
   $\Pi_{A1}...{}_{An}(E_1\times E_2)\equiv\Pi_{B1}...{}_{Bm}()E_1\times\Pi_{C1}...{}_{Ck}(E_2)$, unde
   $\{B_j\}$ este mulţimea atributelor care $\in\{A_i\}$, iar B_j este prezent în E_1.
   Respectiv $\{C_j\}$ este mulţimea celorlalte atribute din $\{A_i\}$, fiecare C_j fiind prezent în E_2.

9. **Comutativitatea proiecţie – reuniune:** $\Pi_{A1}...{}_{An}(E_1\cup E_2)\equiv\Pi_{A1}...{}_{An}(E_1)\cup\Pi_{A1}...{}_{An}(E_2)$.

Algoritm generic pentru optimizarea expresiilor relaţionale
Intrare: un arbore sintactic ce reprezintă o expresie relaţională.
Ieşire: program pentru evaluarea expresiei.

Metoda:
Pas 1. Fiecare selecţie $\sigma_{F1}\wedge_{...\wedge Fn}()E$ este transformată în secvenţa $\sigma_{F1}(\cdots(\sigma_{Fn}(E)))$ (regula 4).
Pas 2. Fiecare selecţie este deplasată cât mai jos posibil în arborele sintactic (regulile 4-7).
Pas 3. Fiecare proiecţie este deplasată cât mai jos posibil în arborele sintactic (regulile 3, 5, 8, 9).
Pas 4. Secvenţele de selecţii şi proiecţii sunt combinate în selecţii sau proiecţii unice, sau în selecţii urmate de proiecţii (regulile 3, 4, 5). Notă: în pasul 4, procesul recomandat poate încălca principiul potrivit căruia proiecţiile sunt efectuate cât mai curând posibil. Trebuie analizată eficienţa!

Pas 5. Nodurile interne ale arborelui rezultate prin parcurgerea paşilor anteriori sunt grupate în „blocuri”. Fiecare nod intern care corespunde unei operaţiuni binare poate face parte din acelaşi bloc ca şi predecesorii săi imediaţi cu care sunt asociate operaţiuni unare. Din bloc pot face parte şi orice lanţ de noduri succesoare asociate cu operaţiuni unare şi terminate cu o frunză. Ultima regulă nu se aplică atunci când operaţiunea binară este un produs cartezian, iar acesta nu este urmat de o selecţie care se combină cu produsul menţionat, astfel încât să formeze o joncţiune cu Θ = semnul egalităţii (echi-joncţiune!).

Pas 6. Se evaluează fiecare bloc, în orice ordine, astfel încât niciunul din blocuri nu este evaluat înaintea grupurilor sale succesoare. Ca rezultat al evaluării este produs un program.

Exemplu. Fie baza de date cu relaţiile:

Circuit(Cnume, Fnume, Cod),
Furnizor(Fnume, Fadr), Utilizator(Unume,Uadr, Nrdoc), Livrari(Nrdoc, Cod, Data).

Presupunem că pentru a răspunde la anumite fraze de interogare privind livrările de circuite, este mai întâi construită o vedere care conţine informaţii referitoare la circuitele livrate, cu ajutorul relaţiilor *Livrari, Utilizator, Circuit.*

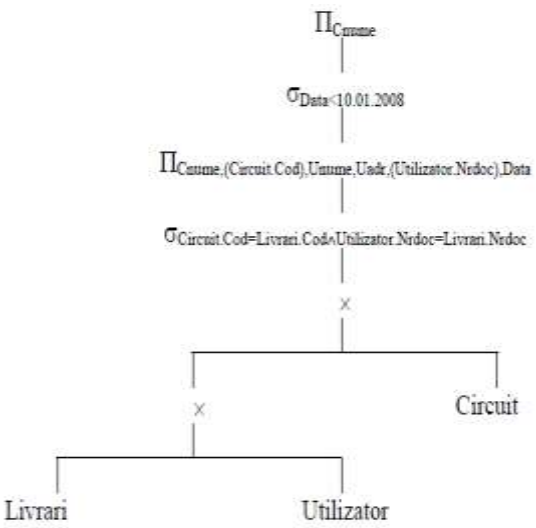
Vederea va fi definită prin expresia relaţională: $\Pi_V(\sigma_U(Livrari \times Utilizator \times Circuit))$,

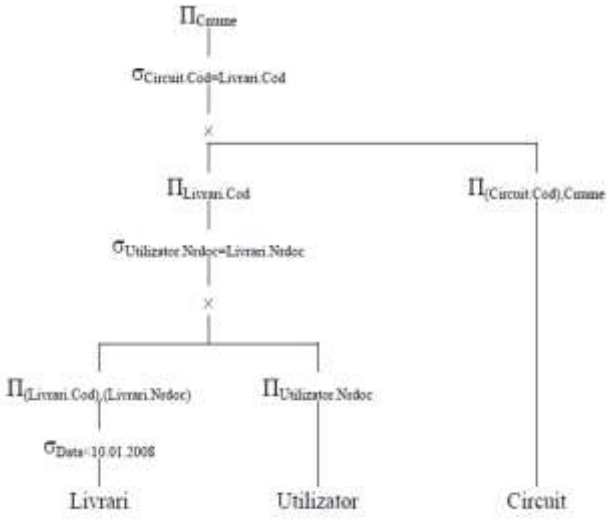
unde V = Cnume, Fnume, Cod, Unume, Uadr, Nrdoc, Data,
iar U = Utilizator.Nrdoc = Livrari.Nrdoc ∧ Circuit.Cod = Livrari.Cod.

Vom admite că fraza de interogare solicită lista numelor circuitelor livrate înainte de 10 ianuarie 2008. În termenii algebrei relaţionale, această frază poate fi formulată prin expresia:

$\Pi Cnume(\sigma Data<10.01.2008(\Pi\ U\ (\sigma V\ (Livrari \times Utilizator \times Circuit))))$.

Pentru evaluarea expresiei, este construit un prim arbore sintactic, iar ca urmare a aplicării algoritmului pentru optimizare, este obţinut un nou arbore sintactic necesar evaluării frazei de interogare.





2.3. PLANURI DE EXECUȚIE A INTEROGĂRILOR

Metode de vizualizare a modului de execuție a interogărilor și optimizări:

1. *Folosirea comenzii EXPLAIN*
   Detalii privind sintaxa acesteia https://dev.mysql.com/doc/refman/5.7/en/explain.html)

2. *Folosirea comenzii EXPLAIN cu reprezentare grafică*
   Detalii privind sintaxa :https://dev.mysql.com/doc/workbench/en/wb-tutorial-visual-explaindbt3.html)

3. *Utilizarea permisiunilor mai puţin complexe*
   Cu cât sunt mai complexe permisiunile de acces la date, cu atât exista mai multă supraîncărcare, deci timpii de răspuns sunt mai mari. Folosind permisiuni simple, atunci când se apelează funcția GRANT, MySQL permite reducerea supraîncărcarii datorată verificărilor permisiunilor atunci când clienții execută interogări.

4. *Funcţiile specifice MySQL pot fi testate folosind comanda "benchmark"*

   Se pot efectua teste de sincronizare prin invocarea funcției BENCHMARK (), folosind programul client al mysql-ului. Sintaxa sa este *BENCHMARK (loop_count, expresie).* Valoarea returnată este întotdeauna zero, dar MySQL afișează aproximativ cât de mult timp a luat funcției să se execute.

5. *Optimizări în clauzele where*

   1. Se folosește declaraţia INSERT pentru a stoca mai multe rânduri cu o instrucţiune SQL.
   2. Comanda EXPLAIN poate să ofere informatii despre ce indecsi sunt folositi in interogarea specificata şi multe alte informaţii utile care ajută la alegerea unui index mai bun sau a unei interogări mai bune(mai rapide).
   - Se scot parantezele inutile.
   - COUNT (*) pe un singur tabel fără WHERE este preluată direct de la tabela de informații pentru MyISAM şi tabele de memorie. Această actiune se realizează, de asemenea, pentru orice expresie NOT NULL atunci când este utilizat un singur tabel.

Întrebarea firească ce apare la execuția de către motorul bazei de date a unei interogări este aceea a modalităților prin care *datele necesare interogării* sunt accesate în cea mai eficientă manieră. Componenta sistemului de gestiune a bazei de date responsabilă cu acest proces se numește ***Optimizatorul de interogări.(engl. Query optimizer).*** Rolul acestuia este acela de a considera dintr-o *varietate de posibile strategii de execuție* a unei interogări având drept scop interogarea datelor ,**pe cea mai eficientă strategie** selectată ce poartă numele de **plan de execuție.** Optimizatorul ia decizia în funcție de un set de criterii și anume: dimensiunea tabelelor, indecși existenti, ce operatori booleeni,(AND, OR, NOT) sunt utilizați în clauza WHERE.

Taskul optimizatorului este acela de a alege *planul de execuție optim* pentru o anumită interogare, task ce presupune următoarele etape : parsare interogare, compilare interogare, optimizare interogare și execuție interogare. Parsarea presupune validarea sintactică și transformarea interogării într-un arbore (cu operanzi şi operatori), urmând validarea obiectelor bazei de date utilizate în interogare cu formarea ulterioară a arborelui final, iar arborele este apoi compilat de către optimizator

Rezultatul compilării devine intrare pentru optimizator și investighează potențiale strategii de acces la date înainte de a decide modul de procesare a interogării.Pentru a identifica cea mai potrivită strategie, optimizatorul va analiza arborele de interogare și va căuta argumente de căutare și operații de tip join. Ulterior va selecta ce indecși va utiliza, va alege apoi ordinea optimă pentru execuția join ,dacă acesta există și tehnica de procesare a acestuia

optimă sub aspectul timpului de răspuns. După crearea planului de execuție, acesta va fi *stocat permanent și executat.*

Modul de lucru al optimizatorului. Optimizatorul lucrează în 4 etape , după cum urmează analiza interogării,selecția indecșilor ,selecția ordinii de execuție join şi alegerea tehnicii de procesare pentru operatorul SQL join.

Aceste etape vor fi descrise succint în continuare.

1.Analiza interogării – interogarea este analizată pentru a identifica argumentele de căutare, utilizare OR și apoi existența operatorului JOIN.Argumente de căutare este parte a interogării ce restricționează setul de rezultate intermediare generate de query.Scopul este acela de a permite utilizarea indecșilor existenți corelat cu expresiile/părți de expresie existente în query.Expresiile ce conțin operatorul NOT nu se constituie ca argumente de căutare.

2.Alegerea indecșilor – Identificarea argumentelor de căutare permite optimizatorului să decidă dacă și care indecși vor fi utilizați.Optimizatorul va verifica printr-un proces complex , selectivitatea unei expresii (cât de multe tuple respectă condiția respectivă din numărul total de tuple), raportat la coloana index utilizând *statisticile* create raportat la la distribuția valorilor unei anumite coloane.

3.Selectarea ordinii tabelelor în operația JOIN – în general această ordine a tabelelor implicate în operația JOIN este specificată în interogare, însă o serie de factori determină decizia optimizatorului de a alege o anumită tabelă *prima accesată* în evaluarea întru execuție a unei clauze JOIN.Se va alege acea care oferă cel mai bun timp de răspuns (minima parcurgere a tuplelor din tabele pentru returnarea rezultatului interogării).

4.Alegerea tehnicii de procesare JOIN.Deoarece cel mai consumator de timp operator este operatorul JOIN, *există mai multe tehnici de procesare a operatorului JOIN* , astfel încât optimizatorul o poate alege pe cea mai adecvată unei anumite interogări, și anume nested loops, merge join și hash join.

- **Nested loop** - este tehnică numită și forța brută astfel pentru fiecare linie din tabela exterioară - outer se regăsește și compară fiecare linie din tabela interioară - inner și se compară.

- **Merge join** - oferă o soluție cost eficientă, astfel liniile tabelei interne trebui să fie sortate fizic după valorile coloanei de JOIN, apoi ambele tabele sunt scanate în ordinea specificată de coloana de join, identificând tuplele pentru care există potrivire (ca valoare a coloanei de join)

- **Hash join** – este soluția abordată atunci cînd NU există indecși pentru coloanele join, astfel ambele tabele vor fi considerate ca intrări tip build și probe, cea mică fiind aleasă de tip build. Procesul se desfășoară într-un mod relativ complex, fiind determinat de algoritmul de hash.Ea este aplicabilă pentru *interogări ad-hoc, fără indecși*.

Detalii suplimentare privind diverse soluții practice utilizate în optimizarea interogărilor MySQL pot fi consultate la adresa https://dzone.com/articles/how-to-optimize-mysql-queries-for-speed-and-perfor

III.ACTIVITĂȚI PRACTICE

3.1. Pentru interogările propuse în http://www.ntu.edu.sg/home/ehchua/programming/sql/sampledatabases.html, se va realiza analiza modului de execuție conform planului de execuție realizat cu ajutorul comenzii *Explain* aplicată interogărilor, în mod text și manieră grafică.

Detalii suplimentare :https://dev.mysql.com/doc/workbench/en/wb-tutorial-visual-explaindbt3.html

Modificați ordinea tabelelor în interogările realizate cu ajutorul operatorilor JOIN și reluați execuția interogării prin intermediul comenzii Explain. Cum se modifică rezultatul ca timp de răspuns prin inversarea ordinii tabelelor asociate operatorului? Argumentați răspunsul.

IV.Probleme propuse

4.1.Se dau schemele de relaţie, cu cheile relaţiilor. subliniate:

Studenţi (ids:integer, numes:string, medie:integer, varsta:real)
Cursuri (idc:integer, numec:string, nprofesor:string)
Examene (ids:integer, idc:integer, ziua:date).

Se cere, prin formularea de expresii *algebrice relaţionale*:

a. să fie găsite numele studenţilor care au programat examenul la disciplina cu numărul de identificare idc=120.
b. Să se determine numele studenţilor care au planifricat examenul cu profesorul Paul.
c. Să se găsească numele profesorilor care predau cursurile la care va fi examinat Radu.
d. să fie găsite numele şi vârstele tuturor studenţilor cu o medie egală cu cel puţin 9.
e. Să se determine numele studenţilor, numerele de identificare ale cursurilor şi datele stabilit pentru examene, pentru fiecare examen.

Pentru rezolvare se poate utiliza calculatorul online pentru expresii relaționale Relax, de la adresa https://dbis-uibk.github.io/relax/landing sau https://github.com/ltworf/relational

4.2. Fie urmatoarea baza de date :

Student (s_id, nume, nivel, varsta)
Curs (c_nume, l_id , loc, ora)
Inscris (s_id, c_nume)
Lector(l_id, l_nume, depart)

Realizati urmatoarele interogari in SQL , respectiv algebra relationala, in conditiile in care continutul campului nivel poate fi (baza:B, master:M, doctorat:D) :

a. Numele studentilor din ciclul de baza inscrisi la cursul de Sisteme_de_operare
b. Lista studentilor inscrisi la cele mai multe cursuri
c. Pentru fiecare nivel de studiu se solicita varsta medie a studentilor inscrisi

4.3. Se vor implementa și analiza cu ajutorului optimizatorului de interogari din MySQL Workbench diverse interogări din tutorial , observând modul de execuție și translatare în planuri de execuție echivalente. Pentru acest scop se va utiliza un instrument specializat online https://www.eversql.com/