**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.9

Disciplina: *Baza de date*

Tema**:** Subinterogări

**A efectuat: Vlasitchi Stefan**

**st. gr. TI-216**

**A verificat:**

**asist. univ. Cebotar Gabriela**

**Chișinău - 2023**

**Lucrare de laborator nr. 9**

## Efectuarea sarcinilor

**Sarcina 1**

1. **Subinterogări**

O **subinterogare** reprezintă o interogare inclusă într-o altă interogare. Rezultatele subinterogării sunt utilizate de SGBD pentru a determina rezultatele interogării de nivel superior care conține subinterogarea. Subinterogările pot apărea în diferite clauze, cum ar fi WHERE, FROM, HAVING, și altele.

* 1. **Subinterogări corelate și necorelate**

Subinterogările, cunoscute și sub numele de instrucțiuni SELECT imbricate, permit exprimarea clară a expresiilor cuantificate. Există două tipuri principale de subinterogări: corelate și necorelate.

Subinterogări necorelate sunt evaluate din interior spre exterior, cu interogarea externă acționând pe baza rezultatului subinterogării. Ele pot apărea în clauza WHERE, de exemplu:

SELECT expresie1 [, expresie2 ..] FROM tabel\_1

WHERE <expresie\_conditie> <operator>

(SELECT expresie FROM tabel\_2);

Subinterogări corelate sunt evaluate invers, cu interogarea externă furnizând valorile subinterogării interne. Ele pot avea o formă generală similară în clauza WHERE:

SELECT expresie\_ext1 [,expresie\_ext2 ..] FROM tabel1 ext

WHERE <expresie\_conditie> <operator>

(SELECT expresie FROM tabel\_2

WHERE expresie = tabel1\_ext.expresie\_ext);

Pasii de execuție ai subinterogărilor corelate includ determinarea unui rând candidat de către cererea externă și execuția cererii interne utilizând valorile acestui rând. Acești pași se repetă pentru fiecare rând candidat.

* 1. **Subinterogări ce întorc un scalar sau o linie**

Există subinterogări care trebuie să returneze obligatoriu un **scalar** sau o **singură** **linie**. Sintaxa pentru acestea este:

{expresie | (expresie1 [, expresie2 ..])} {>|<|=|>=|<=|<>} (<subinterogare>)

Aceste subinterogări utilizează un operator de comparație (cum ar fi =, >, >=, <, <=, <>) și sunt concepute pentru a funcționa doar atunci când subinterogarea produce un singur rezultat.

Predicatul este evaluat ca **TRUE** atunci când comparația dintre rezultatul expresiei și cel returnat de subinterogare este adevărată. În caz contrar, se evaluează la **FALSE**. Dacă subinterogarea nu produce nicio valoare, predicatul se evaluează la **NULL**. Dacă subinterogarea returnează mai mult de o valoare, se va genera o eroare de execuție.

Pentru a ilustra acest concept, considerăm exemplul în care dorim să obținem disciplinele care au numărul maxim de ore din întregul tabel al disciplinelor:

SELECT \* FROM discipline

WHERE Nr\_ore\_plan\_disciplina =

(SELECT MAX(Nr\_ore\_plan\_disciplina) FROM discipline);

Această interogare este corectă deoarece comparația este făcută între o valoare scalară și rezultatul subinterogării care returnează o singură valoare.

Însă, în cazul în care interogarea vizează anumite valori din subinterogare, cum este prezentat în exemplu:

SELECT \* FROM discipline

WHERE Nr\_ore\_plan\_disciplina =

(SELECT Nr\_ore\_plan\_disciplina

FROM discipline

WHERE Nr\_ore\_plan\_disciplina > 60);

Se va genera o eroare, deoarece subinterogarea poate produce mai multe valori, iar predicatul este configurat să funcționeze cu un singur rezultat.

* 1. **Subinterogări cu Operatorul IN**

Operatorul **IN** verifică apartenența unei expresii sau a unei secvențe de expresii la o mulțime de valori rezultată dintr-o subinterogare. Predicatul cu operatorul IN are forma generală:

{expresie | (expresie1 [, expresie2 ..])} [NOT] IN (<subinterogare>)

Există două forme ale predicatului cu operatorul IN. Prima formă presupune că partea stângă a operatorului IN poate conține o singură expresie, iar a doua formă presupune că partea stângă este constituită dintr-o linie de expresii. Operatorul **NOT IN** verifică neapartenența la mulțime.

Predicatul este evaluat la **TRUE** dacă în subinterogare se găsește cel puțin o linie egală cu cea din stânga operatorului IN. În caz contrar, este evaluat la **FALSE**, chiar dacă subinterogarea nu întoarce nicio linie. Dacă subinterogarea întoarce unele linii nule și restul liniilor sunt distincte de linia din stânga operatorului IN, predicatul este evaluat la **NULL**.

Pentru exemplificare, să luăm o interogare care afișează lista profesorilor (identificator, nume, prenume) care nu predau la grupa 'CIB171':

SELECT Id\_Profesor, Nume\_Profesor, Prenume\_Profesor

FROM profesori

WHERE Id\_Profesor NOT IN

(SELECT DISTINCT Id\_Profesor

FROM studenti\_reusita

WHERE Id\_Grupa IN (SELECT Id\_Grupa

FROM grupe

WHERE Cod\_Grupa = 'CIB171'));

Această interogare utilizează operatorul NOT IN pentru a filtra profesorii care nu predau la grupa specificată ('CIB171'). Rezultatul este o listă de profesori care nu au legături cu grupa specificată în subinterogare.

* 1. . **Subinterogări cu Operatorul IN**

Operatorul **ANY** modifică comportamentul operatorului de comparație pentru a permite interogării externe să accepte mai multe linii returnate de subinterogare. Sintaxa generală a predicatului care conține operatorul ANY este:

{expresie | (expresie1 [, expresie2 ..])} {>|<|=|>=|<=|<>} ANY (<subinterogare>)

Cuvântul-cheie **ANY**, care urmează după un operator de comparație, returnează valoarea **TRUE** dacă comparația este adevărată pentru cel puțin una dintre valorile returnate de subinterogare. Subinterogările de tip tabel sunt cele care întorc mai multe linii și coloane.

Predicatul ia valoarea **TRUE** dacă rezultatul comparației cu cel puțin o valoare construită de subinterogare este **TRUE** și returnează **FALSE** în caz contrar. Dacă subinterogarea returnează o linie cu componente nule, predicatul nu poate fi **FALSE** (va lua valoarea **TRUE** sau **NULL**).

Executăm exemplul următor, care are același efect ca și cel din secțiunea precedentă, folosind operatorul **ANY**:

SELECT Id\_Profesor, Nume\_Profesor, Prenume\_Profesor

FROM profesori

WHERE NOT Id\_Profesor = ANY

(SELECT DISTINCT Id\_Profesor

FROM studenti\_reusita AS s INNER JOIN grupe AS g

ON s.Id\_Grupa=g.Id\_Grupa

WHERE Cod\_Grupa = 'CIB171');

Această interogare utilizează operatorul **ANY** pentru a verifica dacă Id\_Profesor nu este egal cu niciunul dintre Id-urile profesorilor din subinterogare. Rezultatul este o listă de profesori care nu predau la grupa specificată ('CIB171').

* 1. **Subinterogări cu Operatorul ALL**

Operatorul **ALL** compară o valoare cu toate valorile returnate de o subinterogare. Sintaxa generală pentru utilizarea acestui operator în predicatele de filtrare este:

{expresie | (expresie1 [, expresie2 ..])} {>|<|=|>=|<=|<>} ALL (<subinterogare>)

Comparatia este adevărată dacă ea este adevărată pentru orice valoare returnată de subinterogare.

Situatia când în partea stângă este un scalar sau o expresie care este evaluată la un scalar:

În această utilizare a operatorului **ALL**, subinterogarea trebuie să returneze o singură coloană. Predicatul se evaluează la TRUE dacă comparația stabilită de operatorul de comparație este validă pentru toate valorile coloanei returnate de interogarea subordonată. Se evaluează la **TRUE** și atunci când subinterogarea nu returnează nicio linie. În caz contrar, se evaluează la **FALSE**. Dacă subinterogarea returnează vreun **NULL**, predicatul se evaluează la **NULL**.

Utilizarea în care în partea stângă este o linie:

Expresiile din partea stângă sunt evaluate și comparate cu fiecare din liniile rezultatului folosind operatorul de comparație dat. Rezultatul predicatului cu **ALL** este **TRUE** dacă comparația returnează **TRUE** pentru toate liniile subinterogării (se include și cazul special al interogării care nu returnează nicio linie). Rezultatul este **FALSE** dacă comparația are valoarea **FALSE** cel puțin pentru o linie returnată de interogarea subordonată. Rezultatul este **NULL** dacă comparația nu are valoarea **FALSE** pentru nicio linie și returnează **NULL** pentru cel puțin o linie.

Vom considera exemplul următor, care afișează numele și prenumele profesorului cu identificatorul mai mare decât al profesorilor care predau la grupa 'CIB171':

SELECT Id\_Profesor, Nume\_Profesor, Prenume\_Profesor

FROM profesori

WHERE Id\_Profesor > ALL

(SELECT DISTINCT Id\_Profesor

FROM studenti\_reusita AS s INNER JOIN grupe AS g

ON s.Id\_Grupa=g.Id\_Grupa

WHERE Cod\_Grupa = 'CIB171');

Această interogare utilizează operatorul **ALL** pentru a compara Id\_Profesor cu toate Id-urile profesorilor care predau la grupa specificată. Rezultatul este un singur profesor cu un identificator mai mare decât toți profesorii care predau la grupa 'CIB171'.

* 1. **Subinterogări cu Operatorul EXIST si NOT EXIST**

Predicatul **EXISTS** este un cuantificator existențial, folosit într-o expresie logică pentru a determina dacă o interogare subordonată poate să returneze vreo înregistrare. În Transact SQL, un predicat cu un cuantificator de existență este reprezentat de expresia **EXISTS** **(SELECT** **\*** **FROM** **...)**. Operatorul EXISTS este frecvent proiectat pentru a fi utilizat numai în contextul subinterogărilor. El este folosit cu subinterogările corelate și testează dacă ceva există sau nu există:

[NOT] EXISTS (<subinterogare>)

Cuvintele-cheie **EXISTS** și **NOT** **EXISTS** sunt destinate exclusiv folosirii subinterogărilor. Așadar, operatorul produce un rezultat simplu, de tip boolean, reprezentând valoarea logică TRUE sau FALSE. Pentru cuvântul-cheie **EXISTS**, rezultatul este TRUE dacă și numai dacă cel puțin un rând este prezent în tabelul de rezultate returnat de subinterogare. Dacă tabelul de rezultate al subinterogării este vid, rezultatul procesării operației EXISTS este FALSE.

Pentru operatorul **NOT** **EXISTS**, sunt utilizate reguli de procesare inversă a cuvântului-cheie **EXISTS**. Deoarece clauzele **EXISTS** și **NOT** **EXISTS** verifică doar prezența rândurilor în tabelul subinterogării, acesta poate conține un număr arbitrar de coloane. Acest operator este util pentru a testa dacă valoarea recuperată de cererea externă există în mulțimea valorilor regăsite de o cerere internă corelată a sa.

Un exemplu de utilizare este afișarea tuturor studenților care nu au note la examen:

SELECT Nume\_Student, Prenume\_Student

FROM studenti

WHERE EXISTS

(SELECT \* FROM studenti\_reusita

WHERE studenti\_reusita.Id\_Student = studenti.Id\_Student

AND Tip\_Evaluare = 'Examen'

AND Nota IS NULL);

Această interogare utilizează operatorul **EXISTS** pentru a verifica dacă există cel puțin o înregistrare în tabelul studenti\_reusita pentru fiecare student cu o notă la examen care este **NULL**.

Operatorul **EXISTS** poate fi negat folosind forma **NOT** **EXISTS**. Este important de menționat că pentru subinterogări, **NOT** **EXISTS** este mai eficient decât **EXISTS**. Utilizând **NOT** **EXISTS**, interogarea returnează o listă a profesorilor care nu predau la grupa **'CIB171'**:

SELECT DISTINCT Nume\_Profesor, Prenume\_Profesor

FROM profesori AS p

WHERE NOT EXISTS

(SELECT \* FROM studenti\_reusita WHERE Id\_Grupa =

(SELECT Id\_Grupa FROM grupe

WHERE Cod\_Grupa = 'CIB171' )

AND Id\_profesor= p.Id\_profesor);

Această interogare utilizează **NOT** **EXISTS** pentru a selecta profesorii care nu au nicio înregistrare în tabelul **studenti\_reusita** asociată grupei 'CIB171'.

* 1. **Subinterogări din clauza FORM**

Până acum, în clauza , a fost specificată o listă de tabele. Cu toate acestea, rezultatul unei instrucțiuni SELECT poate fi un tabel, iar o întrebare naturală este dacă o interogare poate participa în calitate de tabel în clauza FROM. Răspunsul este afirmativ. Formatul unei subinterogări în clauza FROM arată astfel:

(<instrucțiune SELECT>) AS <alias> [(<coloană>[,<coloană2> ..])]

În acest format, **<alias>** este aliasul atribuit tabelului construit de subinterogare, ale cărui coloane pot fi, de asemenea, atribuite opțional cu nume noi. Numărul de coloane ale aliasului coincide cu numărul de coloane returnate de subinterogare. La rândul său, subinterogarea poate conține subinterogări (subinterogări simple fără operații pe mulțimi).

Aceste expresii pot fi specificate într-o listă a clauzei FROM exact ca un tabel. Interogarea principală poate face referire la coloanele subinterogării ca la coloanele oricărui tabel obișnuit. Următoarea interogare cere afișarea tuturor coloanelor din tabelul **discipline** (alias 1) cu numărul minim de ore planificate. Un SELECT imbricat stă după clauza FROM și se utilizează de SELECT-ul principal ca un tabel virtual numit **t2**:

SELECT t1.\*

FROM discipline AS t1,

(SELECT MIN(Nr\_ore\_plan\_disciplina) AS ore

FROM discipline) AS t2

WHERE t1.Nr\_ore\_plan\_disciplina = t2.ore;

Această interogare selectează toate coloanele din tabelul **discipline** unde numărul de ore planificate este egal cu numărul minim de ore planificate din același tabel. Subinterogarea (**t2**) calculează minimul numărului de ore planificate din tabelul **discipline**, iar rezultatul este folosit în interogarea principală pentru a filtra rezultatele.

**Concluzie**

În final, acest studiu detaliat asupra subinterogărilor în limbajul Transact-SQL oferă cunoștințe profunde și practice.Prin explorarea subinterogărilor corelate și necorelate, operatorilor de comparație și utilizării operatorilor IN, ALL, ANY, [NOT] EXISTS, acest ghid furnizează instrumente esențiale pentru manipularea datelor în bazele de date. Integrarea subinterogărilor în clauza FROM, prezentată într-un mod coerent, permite utilizatorului să creeze și să utilizeze tabele virtuale temporare în interogări complexe..