7. Subinterogari. Tipuri de subinterogari

7.1 Subinterogări

*O instrucțiune****SELECT****imbricată (inclusă) într-o altă instrucțiune****SELECT****poartă numele de****subinterogare****(****subquery****)*. Instrucțiunea **SELECT** imbricată, cea din interior mai poartă numele de **interogare secundară** sau de **subcere**.

**MySQL** permite crearea de **subinterogări**, adică interogări care sunt înglobate în alte interogări. **Subinterogările** sunt mereu prelucrate pornind de la interogarea interioară înspre exterior. Deci, prima dată este evaluată instrucţiunea **SELECT** din interior (**subinterogarea**) şi apoi interogarea exterioară.

Fiecare interogare conţine în mod obligatoriu o clauză **FROM**. *Fiecare subinterogare trebuie să fie inclusă între paranteze rotunde pentru ca serverul de baze de date să o execute mai întâi.* Aceasta înseamnă că subinterogarea este, de fapt, o interogare ale cărei rezultate sunt transmise altei interogări. Subinterogarea este o modalitate de a face legături între două sau mai multe interogări.

Există multe situații în care pentru rezolvarea unei anumite cerințe pot fi utilizate fie interogări în care sunt folosite join-uri, fie interogări care conțin subinterogări.

*O****subinterogare****(****subquery****) poate fi plasată în una din următoarele clauze ale instrucțiunii****SELECT****principale:****WHERE****,****HAVING****sau****FROM***. *În clauza****HAVING****va fi folosită subinterogarea atunci când condiția se va aplica unei****funcții de agregare****. Cele 5 funcții de agregare sau de grup, prezentate și exemplificate în modulul anterior, sunt:****COUNT****,****SUM****,****MIN****,****MAX****și****AVG****.*

De asemenea, o **subinterogare** poate fi folosită pentru obţinerea de câmpuri cu valoare calculată, cum ar fi numărul de înregistrări care îndeplinesc o anumită condiţie dintr-o altă tabelă decât cea din care se extrag celelate câmpuri în instrucţiunea **SELECT** principală. În acest caz, în instrucţiunea **SELECT** de tip **subinterogare** trebuie prefixate numele câmpurilor cu numele tabelei pentru a nu exista neclarităţi.

Rezultatul obţinut din **subinterogare** va primi un nume care poartă denumirea de **alias** care va fi folosit în instrucţiunea principală.

Întrucât pe parcursul acestui curs am făcut, în cele mai multe cazuri, referire la termenul de **interogare**, trebuie precizat că, de multe ori, se poate întâlni şi termenul în limbă engleză ce defineşte o astfel de instrucţiune, este vorba despre termenul **query**.

Evident, în mod similar, pentru o **subinterogare** se poate întâlni frecvent şi denumirea în limba engleză, şi anume, **subquery**.

În cadrul interogării principale, pentru introducerea unei subinterogări se foloseşte un operator de comparaţie. Deci, se compară rezultatul returnat de subinterogare cu un rezultat obţinut de interogarea **SELECT** principală.

Putem spune că o subinterogare reprezintă mecanismul prin care rezultatul întors de o interogare poate fi folosit mai departe, pentru a efectua o nouă interogare.

În concluzie, o subinterogare reprezintă o interogare inclusă într-o altă interogare. Rezultatul subinterogării este utilizat de **SGBD**, în cazul nostru **MySQL**, pentru a determina rezultatele interogării de nivel mai înalt (principale) care conţine subinterogarea. Subinterogarea apare, în principal, în clauzele **WHERE** sau **HAVING** ale altei interogări, dar, așa cum am precizat anterior, pot fi și subinterogări plasate în clauza **FROM** a unei interogări.

Subinterogarea este cuprinsă între paranteze rotunde, dar are forma unei instrucţiuni **SELECT**, cu o clauză **FROM**, poate avea, de asemenea, clauzele **WHERE**, **GROUP BY**, **HAVING**.

*Numărul de interogări imbricate (subinterogări) nu este limitat, singura limită vine de la timpul de execuție al instrucțiunii****SQL****, care crește cu cât sunt mai multe subinterogări în aceași frază****SQL****.* Numărul de interogări imbricate depinde și de dimeniunea memoriei buffer (temporare) folosite de subinterogare. Interogarea principală și subinterogarea pot extrage date din tabele diferite sau din aceeași tabelă.

Dacă *o subinterogare nu returnează nici un rezultat sau returnează* **NULL**, *atunci interogarea principală, cea care afișează rezultele, nu va returna nimic*. Este evident acest lucru, deoarece *rezultatele returnate de subinterogare sunt utilizate apoi în instrucțiunea****SELECT****principală*, ori, dacă subinterogarea nu returnează nimic sau returnează **NULL**, pe cael de consecință același lucru se va întâmpla cu instrucțiunea principală.

Clauza **ORDER BY** poate fi menţionată într-o subinterogare, dar nu va influența rezultatele afișate de interogarea principală, deoarece rezultatele subinterogării nu sunt afişate, ci sunt utilizate intern în cadrul interogării principale, deci rezultatele unei subinterogări nu sunt vizibile pentru utilizator. Clauza **ORDER BY** plasată într-o subinterogare va influența doar rezultatele întoarse de acea subinterogare, dar nu și rezultatele afișate de interogarea principlaă. Pot fi ordonate, în schimb, și rezultatul va fi vizibil la afișare, înregistrările returnate de interogarea principală **SELECT**, care au prelucrat şi rezultatele returnate de subinterogare.

Sintaxa unei subinterogări plasate în cadrul clauzei **WHERE (**precizăm că sintaxa unei subinterogări aflată în clauza **HAVING** sau în clauza **FROM** a comenzii principale este aceeași, diferă doar poziția în cadrul interogării principlae**)**, cele mai utilizate, a instrucţiunii **SELECT**principale este următoarea:

**SELECT** column\_name [, column\_name ]

**FROM**   table1 [, table2 ]

**WHERE**  column\_name **OPERATOR**

          (**SELECT** column\_name [, column\_name ]

**FROM** table1 [, table2 ]

          [**WHERE**] ...)

[…]

[**ORDER** **BY** column\_name [**ASC** | **DESC**] [, column\_name [**ASC** | **DESC**]]];

Operatorul folosit este unul de comparare, fie a egalităţii sau inegalităţii unor valori, al apartenenţei la o listă de valori (operatorii **IN** sau **NOT IN**).

# 7.2 Tipuri de subinterogări

Putem clasifica subinterogările, din punct de vedere al numărului de valori și al valorilor returnate, astfel:

·         **de tip scalar – returnează o singură valoare (un singur rând și o singură coloană);**

·         **de tip listă – returnează mai multe rânduri, dar o singură coloană;**

·         **de tip rând – returnează un singur rând, dar mai multe coloane;**

·         **de tip tabelă – returnează mai multe rânduri și mai multe coloane.**

# 7.2.1 Subinterogări de tip scalar

Subinterogările de **tip scalar** întorc un scalar, adică o valoare atomică ce poate fi folosită ca o constantă într-o expresie **SQL**. Acest tip de **subinterogare** returnează **o singură coloană** şi **un singur rând**dintr-o tabelă. Condiţia este pusă pe cheia primară.

Interogarea care întoarce valoarea se scrie inclusă între paranteze rotunde „**()**”, din acest moment ea se comportă ca şi cum, în acel loc, din punct de vedere sintactic, am avea o singură valoare.

# 7.2.2 Subinterogări de tip listă

În cazul celor de **tip listă** se va returna o **înşiruire de valori** pentru **o singură coloană**. Deci, poate să returneze **mai multe rânduri**, dar, **doar o singură coloană**. În cadrul acestor subinterogări se folosesc operatorii **IN** sau **NOT IN**, operatori care verifică apartenenţa la o listă de valori, listă care a fost returnată de subinterogare.

# 7.2.3 Subinterogări de tip rând

Interogarea subordonată (subinterogarea) de **tip rând** se foloseşte pentru a verifica dacă liniile (înregistrările) extrase din interogarea secundară sau subordonată (subinterogare) există printre liniile (rândurile) extrase din interogarea principală. În acest scop, se foloseşte cuvântul cheie **EXISTS** care poate fi şi negat, deci, se poate folosi şi sub forma **NOT EXISTS**.

Sintaxa unei astfel de subinterogări, ce utilizează operatorul **EXISTS** sau negația acestuia, **NOT EXISTS**, este următoarea:

**SELECT** column\_name [, column\_name ]

**FROM** table1 [, table2 ]

**WHERE** **EXISTS** | **NOT EXISTS**

                   (**SELECT** column\_name [, column\_name ]

**FROM** table1 [, table2 ]

                   [**WHERE**] ...)

[...]

[**ORDER BY** column\_name [**ASC** | **DESC**]  [, column\_name [**ASC** | **DESC**]  ]];

În acest mod, se verifică existenţa unor înregistrări rezultate din subinterogare printre rezultatele interogării **SELECT** principale.

7.2.4 În cazul *subinterogărilor de tip tabelă*, adică acele instrucțiuni **SELECT** secundare care întorc mai multe rânduri și mai multe coloane, subcererea se regăsește în clauza **FROM** a instrucțiunii **SELECT** principale. Deci, această subinterogare, întrucât va returna o tabelă *este necesar să aibă asociat un nume*, adică un *alias*. Acest alias primit de subcerere reprezintă *numele tabelei care va prefixa fiecare utilizare a unei coloane extrase* în cadrul subinterogării.

*Alias*-ul se specifică tot cu ajutorul clauzei **AS**, aşa cum am expus în prezentarea **alias-**urilor descrisă în lecţiile anterioare. Trebuie, de asemenea, inclusă între paranteze rotunde „**()**”. Câmpurile tabelei întoarse de subinterogare au numele exact ca în antetul care s-ar afişa dacă ar fi executată această subinterogare.

Tabela rezultată dintr-o astfel de subinterogare poate fi utilizată pentru realizarea de uniuni cu alte tabele existente în baza de date, adică poate participa la realizarea de join-uri. Atunci când sunt folosite de către interogarea care o subordonează, câmpurile interogării subordonate trebuie adresate prin **alias**-ul tabelei, urmat de caracterul „**.**” şi apoi de numele câmpului (coloanei) din subinterogare.

Sintaxa unei subinterogări de tip tabelă este următoarea:

**SELECT** ... **FROM** (subquery) [**AS**] nume\_alias ...

# 7.3 Operatori folosiți pentru introducerea subinterogărilor

Pe lângă operatorii de comparare foarte cunoscuți și care sunt folosiți și în condițiile unei instrucțiuni **SELECT** ce nu conține subcereri (subinterogări), adică: **=, <, >, !=, <>, <=, >=, IN, NOT IN**, **BETWEEN**, etc.  mai există **o serie de operatori specifici subinterogărilor**.

Este vorba despre operatorii: **ANY**, **ALL** și **EXISTS**, respectiv, **negațiile acestora** care presupune apariția operatorului **NOT** în fața oricăruia dintre aceștia.

De asemenea, operatorii **ANY** şi **ALL** se folosesc în combinaţie cu operatorii de comparare**.** Operatorul **ANY** înseamnă oricare element din listă, în timp ce **ALL** înseamnă toate elementele din listă. Operatorii **ANY** şi **ALL** sunt operatori de cuantificare, se mai numesc şi cuantificatori. Aceştia extind, practic, operatorii de comparare.

# 7.3.1 Operatorul ANY

Operatorul **ANY** este folosit **împreună** cu operatorii **=, >, <, >=, <=, !=** sau poate fi **negat**. El este utilizat pentru compararea cu oricare dintre valorile ce urmează după el, deci, valori ce sunt returnate de subinterogare. *Vor fi returnate acele valori care îndeplinesc condiția pentru măcar una din valorile din lista generată de subinterogare.* Poate fi și negat acest operator.

Operatorul **ANY** compară o valoare cu fiecare valoare returnată de subcerere și *este suficient ca măcar o singură valoare întoarsă de subcere să îndeplinească condiția* pentru ca valoarea din instrucțiunea principală care a fost comparată cu lista de rezultate din subinterogare să fie returnată ca rezultat al comenzii **SQL**.

Dacă valoarea acestei coloane coincide cu **vreuna** dintre valorile furnizate de subinterogare, condiţia este evaluată la valoarea de adevăr **true** (**adevărat**). Altfel, este evaluată ca fiind **false**(**falsă**).

# 7.3.2 Operatorul ALL

Operatorul **ALL** este, la fel, utilizat **împreună** cu operatorii de comparare **specificați la ANY**.  Utilizarea lui înseamnă *compararea cu toate rezultatele întoarse de subinterogare*. Vor fi returnate acele valori care *îndeplinesc condiția, fiind comparate cu toate valorile din lista întoarsă din subinterogare*. Poate fi negat și acest operator.

Operatorul **ALL** *compară o valoare cu fiecare valoare returnată de subinterogare și trebuie ca toate valorile întoarse de subcerere să îndeplinească condiția de comparare* pentru ca valoarea din instrucțiunea principală care a fost comparată cu lista de valori întoarsă din subinterogare să fie returnată ca rezultat al comenzii **SQL**.

Dacă valoarea  acestei coloane coincide cu **fiecare** dintre valorile furnizate de subinterogare, condiţia este evaluată la valoarea de adevăr **true** (**adevărat**). Altfel, este evaluată ca fiind **false**(**falsă**).

Operatorii **ANY**, respectiv, **ALL** se pot regăsi în clauzele **WHERE** sau **HAVING** ale unei instrucțiuni **SELECT**.

# 7.3.3Operatorul EXISTS

Operatorul **EXISTS** *verifică dacă o valoare se regăsește în mulțimea de valori întoarsă de o subinterogare*. Dacă **valoarea există**, comanda va întoarce **TRUE**, iar **dacă valoarea nu există** în lista de valori returnate de subinterogare, atunci comanda va întoarce **FALSE**. Și acest operator poate fi negat.

Cu alte cuvinte, operatorul **EXISTS** *verifică dacă subinterogarea returnează vreo linie*. O subinterogare ce folosește operatorul **EXISTS** *poate fi implementată și cu ajutorul operatorului IN*, însă, prin aplicarea operatorului **EXISTS** *performanța comenzii****SQL****este mai mare* deoarece operatorul **IN** *compară fiecare valoare întoarsă de subcerere*, pe când **EXISTS** *verifică doar existența a cel puțin unei linii* (înregistrări) întoarse de subinterogare, *fără a face nici o comparație*.

# 7.4 Concluzii

În această lecţie am discutat despre noţiunea de **subinterogare**și despre **tipurile de subinterogări**. Sunt noţiuni complexe în lucrul cu baze de date, însă utile în foarte multe situaţii.

Următoarea lecţie a acestui curs va conține prezentări ale noțiunilor de **tabelă virtuală** (**vedere** sau **view**), **tabelă temporară** și **index**. De asemenea, vor fi prezentate **tipurile de index**, precum și avantajele și dezavantajele pe care le oferă indexarea unor coloane din tabelele unei baze de date

## Tema Sedinta 7

În fișierul de la link-ul **Baza de Date classicmodels de la Ședința 5**sunt instrucțiunile pentru crearea bazei de date **classicmodels** și a tabelelor din această bază de date.

**Cerințe**

1. Rulați instrucțiunile din fișier pentru crearea bazei de date și a tabelelor.
2. Scrieți o instrucțiune **SQL** prin care să **afișați clienții care au interacționat cu angajați de la birourile din Australia** (**câmpuri afișate:** **contactLastName, contactFirstName, city, country**). **Utilizați subinterogări.**
3. Scrieți o instrucțiune **SQL** prin care să **afișați angajații de la birouri din afara USA** (**câmpuri afișate:** **lastName**, **firstName, officeCode**). **Utilizați subinterogări.**
4. Scrieți o instrucțiune **SQL** prin care să **afișați produsele comandate de clienți din Boston, Madrid și Los Angeles**(**câmpuri afișate:** **productCode**, **productName**).

**Realizați 2 variante: una cu JOIN si una cu subinterogări.**

1. Scrieți o instrucțiune **SQL** prin care să **afișați comenzile cu valoarea mai mare decât valoarea medie a tuturor comenzilor în care au fost comandate produse Ferrari** (**câmpuri afișate:orderNumber**, **productCode**, **valoare comandă (priceEach\*quantityOrdered)** și **valoarea medie a tuturor comenzilor**). **Utilizați subinterogări.**

**Valoarea comenzii și media vor fi afișate cu 2 zecimale ia rezultatele vor fi ordonate crescător după valoarea comenzii.**