Lecţia 7 – Subinterogări. Tabele virtuale (Vederi)

7.1 Subinterogări

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.

Instrucţiunile de tip **subinterogare** pot returna o singură coloană pentru ca rezultatele din această coloană să poată fi folosite în condiţiile din interogarea principală. Dacă se încearcă obţinerea mai multor coloane în instrucţiunea **SELECT** de tip **subinterogare** se va genera o eroare.

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.

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**.

Însă, clauza **ORDER BY** nu poate fi menţionată într-o subinterogare deoarece rezultatele subinterogării nu sunt afişate, ci sunt utilizate intern în cadrul interogării principale, deci nu are sens ordonarea rezultatelor unei subinterogări deoarece ele nu sunt vizibile pentru utilizator. Pot fi ordonate, în schimb, rezultatele interogării principale **SELECT**, care au prelucrat şi rezultatele returnate de subinterogare.

Sintaxa unei subinterogări plasate în cadrul clauzei **WHERE**, cele mai utilizate, a instrucţiunii **SELECT** principale este următoarea:

SELECT ***nume_coloană*$_1$*, nume_coloană*$_2$**, ... FROM** ***tabela*$_1$

WHERE (*nume_coloană*$_1$*, nume_coloană*$_2$**, ...) OPERATOR**

(SELECT ***nume_coloană*$_1$*, nume_coloană*$_2$**, ... FROM** ***tabela*$_2$ **WHERE** ***condiţie*****);**

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 astfel:

- **de tip scalar;**
- **de tip listă;**
- **de tip rând;**
- **de tip tabelă.**

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.

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.

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. Operatorul **ANY** este utilizat împreună cu unul dintre cei şase operatori de comparaţie (=, **!=,** <, <=, >, >=) pentru a compara valoarea unei coloane cu valorile furnizate de o subinterogare. 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**).

Operatorul **ALL**, ca si operatorul **ANY**, este utilizat împreună cu unul dintre cei şase operatori de comparaţie (=, **!=,** <, <=, >, >=) pentru a compara valoarea unei coloane cu valorile furnizate de o subinterogare. 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**).

Interogarea subordoantă (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 sens, se foloseşte cuvântul cheie **EXISTS** care poate fi şi negat, deci, se poate folosi şi sub forma **NOT EXISTS**.

Sintaxa unei astfel de subinterogări este următoarea:

SELECT ***coloană*$_1$** **FROM** ***tabela*$_1$**

WHERE EXISTS | NOT EXISTS

(SELECT ***nume_coloană*$_1$, *nume_coloană*$_2$, ...** **FROM** ***tabela*$_2$** **WHERE** ***condiţie*****);**

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

Interogările subordonate (subinterogările) de **tip tabelă** returnează mai multe rânduri şi mai multe coloane, deci, o tabelă. De obicei, acestea se folosesc la joncţiuni între o tabelă întreagă şi o parte dintr-o altă tabelă.

În acest caz, tabela returnată de interogarea subordonată trebuie să primească un nume (un **alias**). Acesta 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.

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.

7.3 Tabele virtuale (vederi, view-uri)

Vederile sunt, de fapt, nişte **tabele virtuale**, adică vederile conţin interogări care regăsesc în mod dinamic datele atunci când sunt utilizate. Ele nu conţin date, ci fac referire la date din tabelele bazei de date. **Vederile** (**tabelele virtuale**) sunt stocate alături de tabele în baza de date.

Vederile pot fi folosite pentru simplificarea operaţiunilor **SQL** complexe. După ce a fost scrisă o interogare aceasta poate fi refolosită cu uşurinţă dacă a fost creată o vedere pe baza acelei interogări. În plus, rezultatele din tabela virtuală se actualizează dinamic, în funcţie de ceea ce se întâmplă în tabelele din care au fost extrase datele în interogarea care a fost declarată la crearea tabelei virtuale.

Vederile sau **tabelele virtuale** sunt adesea denumite cu termenul în limba engleză, este vorba de termenul **view**. Mai trebuie precizat şi faptul că, **vederile** (**view**-urile) pot fi folosite pentru a folosi doar părţi din anumite tabele, iar nu tabele complete.

După ce au fost create **vederile**, se pot efectua operaţii de interogare (instrucţiuni **SELECT**) şi pe acestea. Ca şi tabelele, vederile trebuie să aibă un nume unic în cadrul unei baze de date. O **vedere nu** poate primi **numele unei tabele** din acea bază de date. O **tabelă virtuală** în **SQL** este creată utilizând comanda **CREATE VIEW**. De asemnea, trebuie specificat un nume pentru tabela virtuală imediat după comanda **CREATE VIEW**.

Putem spune că o **vedere** reprezintă o instrucţiune **SELECT** stocată în baza de date. **Vederile** reprezintă un mod de accesare a datelor, ele nu stochează date efectiv. Sunt legate de tabela sau tabelele pe baza cărora au fost create. Ele pot fi folosite din motive de securitate. De exemplu, se pot crea părţi din tabelă vizibile anumitor utilizatori.

Vederile pot fi actualizabile sau neactualizabile. Cele actualizabile permit instrucţiunile **INSERT**, **UPDATE**, **DELETE**. Pe cele neactualizabile nu se pot efectua aceste operaţii. Pentru a fi actualizabile trebuie să fie o corespondenţă linie cu linie între o tabelă virtuală şi tabela pe baza căreia a fost creată. Este evident acest lucru, dacă nu există această corespondenţă între view şi tabelă nu se pot face actualizări în tabela virtuală.

De asemenea, este important de menţionat că nu trebuie să avem în instrucţiunea care creează tabela virtuală actualizabilă **funcţii de grup**, clauzele **DISTINCT**, **GROUP BY** şi **HAVING**. Nu putem avea nici **subinterogări**, **reuniuni** sau **OUTER JOIN**. În schimb, dacă avem **INNER JOIN** în instrucţiunea de creare, atunci este permisă actualizarea.

Sintaxa folosită pentru a crea o vedere (view) este următoarea:

CREATE VIEW *nume_vedere* AS

SELECT ...;

O vedere (view) poate fi redefinită. În acest sens, avem o instrucţiune specială, similară cu cea de creare a tabelei virtuale.

Sintaxa folosită pentru redefinirea unui view este următoarea:

ALTER VIEW *nume_vedere*$_2$ AS

SELECT ...;

Evident, avem şi posibilitatea de a şterge o tabelă virtuală din baza de date. Instrucţiunea care permite ştergerea unei vederi din baza de date este următoarea:

DROP VIEW *nume_vedere*;

Alte menţiuni pe care le vom face în legătură cu vederile se referă la faptul că o vedere poate fi creată şi pe baza unei alte vederi, deci, nu este obligatoriu să fie realizată pe baza unei tabele. Tabelele virtuale şi tabelele din baza de date sunt stocate în acelaşi loc pe disc, din acest motiv, este necesar ca ele să aibă nume diferite.

În această lecţie am discutat despre noţiunile de subinterogare şi tabelă virtuală în **MySQL**. Sunt noţiuni complexe în lucrul cu baze de date, însă utile în foarte multe situaţii.

Ultima lecţie a acestui curs va face o scurtă prezentare a **rutinelor MySQL**, **proceduri** şi **funcţii**, cunoscute şi sub numele de proceduri stocate. Vor fi explicate diferenţele dintre proceduri şi funcţii, modul în care se creează, vor fi prezentate tipurile de **variabile** disponibile, precum şi instrucţiunile de **decizie** şi **repetitive** existente în **rutinele MySQL**. De asemenea, se va face o scurtă trecere în revistă a **trigger**-ilor (**declanşatorilor**) şi a **tranzacţiilor**.