Lecţia 6 – Join-uri şi Reuniuni

6.1 Uniuni de tabele

Pentru a putea selecta date din două sau mai multe tabele trebuie să unim acele tabele, deci, introducem noţiunea de uniune de tabele. Întrucât ne ocupăm de baze de date relaţionale, ştim că tabelele din baza de date sunt legate între ele (sunt relaţionate). Nu este indicat să avem tabele izolate într-o bază de date. Pe parcursul acestui curs am făcut referire în mai multe rânduri la relaţionarea dintre tabelele unei baze de date. De asemenea, au fost prezentate şi tipurile de relaţii posibile a fi existente între tabele.

Aşa cum am spus, extragerea datelor din două sau mai multe tabele ale unei baze de date se realizează prin crearea unei uniuni între aceste tabele. Uniunile dintre tabele poartă şi numele de **join**. De altfel, sub această denumire, de **join**, sunt cel mai mult cunoscute.

Rezultatul returnat de o interogare **SELECT** în care se extrag date din mai multe tabele, deci un **join** sau o uniune de **tabele** va fi o singură tabelă. Deci, printr-o singură interogare extragem date din mai multe tabele, iar rezultatul returnat va fi o singură tabelă.

Există multe situaţii în care este necesară extragerea informaţiilor din mai multe tabele. Pentru acest lucru trebuie realizată o relaţie între tabele pe baza unor informaţii comune. Această interogare a bazei de date în care se realizează relaţii între tabele pe baza unor informaţii comune (câmpuri comune, coloane comune) se numeşte **joncţiune** (**JOIN**). Cu alte cuvinte, tabelele sunt unite în cadrul interogărilor de regăsire a datelor.

6.2 Tipuri de join-uri

O împărţire simplă a joncţiunilor s-ar putea face în **joncţiuni** cu clauză **WHERE** şi **joncţiuni** fără clauză **WHERE**.

Rezultatele returnate de o relaţie între tabele în absenţa unei condiţii de unire, condiţie care se pune într-o clauză **WHERE**, va returna produsul cartezian al celor două tabele. Deci, numărul înregistrărilor returnate va fi egal cu numărul înregistrărilor din prima tabelă înmulţit cu numărul de înregistrări din cea de-a doua tabelă.

6.3 CROSS JOIN

Produsul cartezian (**CROSS JOIN**) corespunde rar rezultatului dorit prin unirea a două sau mai multe tabele. Pentru a obţine rezultatul dorit, de cele mai multe ori trebuie să punem şi condiţii pe câmpurile tabelelor pe care le aducem în uniune. Pentru aceasta vom folosi clauza **WHERE** aproape de fiecare dată când utilizăm **uniuni de tabele** sau **join**-uri (**joncţiuni**). Deci, pentru a obţine un produs cartezian pe două tabele fiecare înregistrare din prima tabelă este combinată cu toate înregistrările din cea de-a doua tabelă.

La uniunile de tip produs cartezian se realizează o selecţie în care se trec toate câmpurile pe care dorim să le obţinem, din cadrul ambelor tabele, iar la clauza **FROM** se trec ambele tabele, fiecare dintre

ele putând avea definit şi un **alias**, adică un nume mai scurt prin care pot fi accesate. Tipul de uniune care returnează un produs cartezian se mai numeşte şi **uniune încrucişată**.

Nu există o limită în ceea ce priveşte numărul de tabele ce pot fi unite în cadrul unei instrucţiuni **SELECT**. Aşadar, putem avea două sau oricât de multe tabele unite printr-o **joncţiune**. Însă, cu cât avem mai multe tabele legate într-un **join**, iar tabelele sunt populate cu date numeroase, timpul de răspuns la o anumită solicitare poate fi mai mare.

Referindu-ne acum la joncţiunile cu clauză **WHERE**, acestea se pot împărţi în următoarele tipuri de **join**-uri:

- **INNER JOIN** – joncţiune naturală;
- **OUTER JOIN** – joncţiune externă;
- **SELF JOIN** – joncţiunea unei tabele cu ea însăşi.

6.4 INNER JOIN

Joncţiunea naturală (**INNER JOIN**) a două tabele reprezintă înregistrările din prima tabelă care au corespondent în cea de-a doua tabelă. Deci, **INNER JOIN** reprezintă, de fapt, **intersecţia** valorilor din cele două tabele. Dacă sunt linii (înregistrări) în prima tabelă care nu au corespondent în cealaltă, ele nu sunt extrase (selectate).

Uniunile de tip **INNER JOIN** se aplică atunci când uniunea se realizează între două tabele şi dorim ca din ambele tabele să fie afişate doar acele linii pentru care condiţia de legătură este satisfăcută. Sintaxa acestei comenzi este următoarea:

SELECT ***exp*$_1$, *exp*$_2$ ...** **FROM** ***tabela*$_1$** **[AS** ***alias*$_1$**]** **INNER JOIN** ***tabela*$_2$** **[AS** ***alias*$_2$**]**
ON ***condiţie*****;**

Evident, dacă există coloane care se numesc la fel în ambele tabele, vom prefixa acele coloane cu numele tabelei sau alias-ul tabelei din care face parte acea coloană (**tabelă.coloană**).

În acest tip de **join**, cuvântul cheie **INNER** poate fi omis, prin specificarea clauzei **JOIN** se subînţelege că este vorba de o joncţiune de acest tip – **INNER JOIN**.

6.5 OUTER JOIN

Joncţiunile externe (**OUTER JOIN**) se împart în:

- **LEFT OUTER JOIN;**
- **RIGHT OUTER JOIN.**

Cele două tipuri de joncţiuni externe mai sunt denumite şi în forma prescurtată **LEFT JOIN**, respectiv, **RIGHT JOIN**.

LEFT JOIN reprezintă toate înregistrările din tabela din partea stângă (prima tabelă) care au corespondent sau nu în tabela din partea dreaptă (cea de-a doua tabelă). Altfel spus, este vorba de înregistrările din prima tabelă care se regăsesc sau nu se regăsesc în cea de-a doua tabelă.

În termeni de teoria mulţimilor, considerând cele două tabele ca fiind două mulţimi **A** şi **B**, **LEFT JOIN** între cele două tabele (**A** şi **B**) reprezintă **(A \ B) U (A ∩ B)**. Deci, **LEFT JOIN** reprezintă rezultatul diferenţei dintre cele două mulţimi reunit cu rezultatul intersecţiei celor două mulţimi.

În mod similar, **RIGHT JOIN** reprezintă toate înregistrările din tabela din partea dreaptă (a doua tabelă) care au corespondent sau nu în tabela din partea stângă (prima tabelă).

Cu alte cuvinte, **RIGHT JOIN** returnează acelaşi rezultat, cu deosebirea că se vor extrage înregistrările din a doua tabelă care au sau nu înregistrări în prima. Este, practic, reciproca joncţiunii **LEFT JOIN**.

RIGHT JOIN este mult mai puţin folosită deoarece se poate transforma foarte simplu într-un **LEFT JOIN** prin schimbarea ordinii celor două tabele. De aceea, forma cea mai utilizată a joncţiunilor externe este **LEFT JOIN**.

Uniunile de tip **LEFT OUTER JOIN** se aplică atunci când uniunea se realizează între două tabele şi dorim ca din prima tabelă să fie incluse absolut toate liniile iar din a doua tabelă doar acele linii care îndeplinesc condiţia specificată la clauza **ON**. La acele linii din prima tabelă care nu au corespondent în a doua tabelă, în dreptul câmpurilor acesteia din urmă se va trece valoarea **NULL**.

Sintaxa instrucţiunii pentru **LEFT OUTER JOIN** este următoarea:

SELECT *exp$_1$, exp$_2$* ... FROM *tabela$_1$* [AS *alias$_1$*] LEFT OUTER JOIN *tabela$_2$* [AS *alias$_2$*]
ON *condiţie*;

Uniunile de tip **RIGHT OUTER JOIN** sunt similare cu cele de tip **LEFT OUTER JOIN**. După cum e de aşteptat, prin analogie cu situaţia precedentă, se aplică atunci când uniunea se realizează între două tabele şi dorim ca din a doua tabelă să fie incluse absolut toate liniile, iar din prima doar acele linii pentru care condiţia de la clauza **ON** este îndeplinită. La acele linii din a doua tabelă care nu au corespondent în prima tabelă, în dreptul câmpurilor acesteia din urmă se va trece valoarea **NULL**.

Sintaxa instrucţiunii pentru **RIGHT OUTER JOIN** este următoarea:

SELECT *exp$_1$, exp$_2$* ... FROM *tabela$_1$* [AS *alias$_1$*] RIGHT OUTER JOIN *tabela$_2$* [AS *alias$_2$*]
ON *condiţie*;

6.6 SELF JOIN

SELF JOIN sau joncţiunea unei tabele cu ea însăşi este utilizată pentru a se uni o tabelă tot cu ea în scopul extragerii anumitor date. În acest sens tabela este redenumită temporar în cadrul instrucţiunii care realizează **SELF JOIN**-ul. **SELF JOIN** mai poartă numele de auto-uniuni, pentru că o tabelă se leagă tot cu ea prin acest tip de joncţiune.

Auto-uniunile reprezintă faptul că se pot face uniuni în care ambele tabele sunt, de fapt, una şi aceaşi tabelă, însă au **alias**-uri diferite şi se consideră unite printr-o coloană.

6.7 Crearea de join-uri utilizând clauza USING

Clauza **USING** permite ca o condiţie să fie pusă pe o singură coloană. Pentru a folosi această clauză este necesar să se numească la fel coloana în ambele tabele.

Sintaxa generală cu clauza **USING** este următoarea:

SELECT ***tabela$_1$.coloana, tabela$_2$.coloana,*** **FROM** ***tabela$_1$*** **JOIN** ***tabela$_2$***

USING (*nume_coloana)*;

6.8 Crearea de join-uri utilizând clauza ON

Clauza **ON** este folosită pentru a specifica condiţii arbitrare sau coloanele care participă la **JOIN**. Condiţia de **join** este separată de celelalte condţii de căutare. De altfel, această clauză **ON** a fost introdusă în sintaxa comenzilor de **join** prezentate la punctele anterioare.

6.9 Alias-uri

Pentru tabelele care alcătuiesc uniunea se pot stabili **alias**-uri de nume, sub forma **nume_tabelă as alias_nume**, introduse în clauza **FROM** a instrucţiunii **SELECT**. **Alias**-urile pot fi utilizate în orice parte a instrucţiunii **SELECT**. Adresarea unei coloane a unei tabele se va face, atunci când tabela are un **alias**, sub forma **nume_alias.nume_coloană**. Este recomandat să asigurăm nume scurte pentru **alias**-urile tabelelor. Un alt avantaj al **alias**-urilor este că putem realiza auto-uniuni, în cadrul cărora aceeaşi tabelă poate avea **alias**-uri diferite. Acest aspect l-am explicat şi când am prezentat **SELF JOIN**. Utilizarea unui **alias** duce şi la o scurtare a sintaxei **SQL** într-o comandă.

Pe lângă **alias**-urile pentru tabele se mai pot folosi **alias**-uri şi pentru coloanele unei tabele, dar şi pentru câmpuri cu valoare calculată sau pentru câmpuri rezultate prin aplicarea unei funcţii de agregare.
Sintaxa unei instrucţiuni de selecţie din mai multe tabele este următoarea:

SELECT ***exp$_1$, exp$_2$*** **...** **FROM** ***tabelă$_1$*** **[AS** ***alias$_1$*****],** ***tabelă$_2$*** **[AS** ***alias$_2$*****]**

[WHERE ***condiţii*****];**

Expresiile din interogarea **SELECT**, ***exp$_1$, exp$_2$,*** **...** vor trebui să conţină numele coloanelor din tabele, dacă nu există confuzii (coloane care au în ambele tabele acelaşi nume) sau numele **alias**-urilor acelor coloane urmate de caracterul „**.**” (**punct**) şi de numele coloanelor, în caz contrar, adică în caz că există confuzii. O astfel de comandă, evident, poate fi generalizată şi pentru mai mult de două tabele.

6.10 Reuniuni

Majoritatea interogărilor conţin o singură instrucţiune **SELECT**, care returnează date din una sau mai multe tabele. Dar este permisă şi efectuarea mai multor interogări şi returnarea rezultatelor sub forma unui singur set de rezultate ale interogării. Aceste interogări combinate se numesc reuniuni sau interogări compuse.

Interogările **SQL** se combină folosind operatorul **UNION**. Prin utilizarea acestui operator se pot specifica instrucţiuni **SELECT** multiple iar rezultatele lor pot fi combinate într-un singur set de rezultate.

O reuniune trebuie să fie alcătuită din două sau mai multe instrucţiuni **SELECT**, separate prin cuvântul cheie **UNION**.

Fiecare instrucţiune **SELECT** trebuie să conţină acelaşi număr de coloane, expresii sau funcţii agregat, dar nu contează ordinea în care sunt specificate coloanele. Coloanele din instrucţiunile **SELECT** trebuie să abiă tipuri de date compatibile.

Operatorul **UNION** elimină în mod automat toate rândurile duplicate din setul de rezultate al interogării. Deci, dacă sunt înregistrări returnate de mai multe instrucţiuni **SELECT** dintr-o reuniune, este afişată o singură dată această înregistrare.

Sintaxa unei instrucţiuni în care se realizează o reuniune a datelor din două tabele prin utilizarea operatorului **UNION** este următoarea:

SELECT *nume_coloană$_1$, nume_coloană$_2$,* ... FROM *tabela$_1$* [WHERE *condiţii*]

UNION

SELECT *nume_coloană$_1$, nume_coloană$_2$,* ... FROM *tabela$_2$* [WHERE *condiţii*];

Pentru ordonarea rezultatelor obţinute în urma efectuării unei reuniuni, clauza **ORDER BY** se trece la final, după ultima instrucţiune **SELECT**.

Reluăm sintaxa instrucţiunii anterioare, adăugând şi clauza **ORDER BY**:

SELECT *nume_coloană$_1$, nume_coloană$_2$,* ... FROM *tabela$_1$* [WHERE *condiţii*]

UNION

SELECT *nume_coloană$_1$, nume_coloană$_2$,* ... FROM *tabela$_2$* [WHERE *condiţii*];

ORDER BY *nume_coloană;*

În această lecţie am abordat noţiunile de **joncţiune** sau **uniune de tabele** şi **reuniune**. Am prezentat tipurile de joncţiuni (**join**-uri) care se pot întâlni în **MySQL**, şi, de asemenea am discutat despre unirea interogărilor folosind operatorul **UNION**.

Aşadar, începând cu această lecţie am început să expunem lucruri mai complexe din domeniul bazelor de date. Lecţia următoare continuă să abordeze teme cu o complexitate mai mare, şi anume, vom prezenta **subinterogările** în **MySQL**, şi vom introduce termenul de **tabelă virtuală** (**vedere**, **view**) şi vom prezenta modul în care sunt definite aceste **tabele virtuale**.