Tema 5, Cursul 10

**Răspundeți la un set de întrebări date, specifice unui interviu de angajare pentru un post de analist de date, cu accent pe optimizarea interogărilor SQL și prezentarea unor instrumente avansate.**

### **Întrebare 1:** Cum folosești diferite tipuri de join-uri în SQL și care sunt cazurile de utilizare pentru fiecare?

**INNER JOIN**: Returnează doar rândurile care au corespondență în ambele tabele implicate în join. Se folosește atunci când dorești să obții doar datele care există în ambele tabele. De exemplu, pentru a extrage toate comenzile plasate de clienții care există în tabelul Customers.

**LEFT (OUTER) JOIN**: Returnează toate rândurile din tabelul din stânga și rândurile corespunzătoare din tabelul din dreapta. Dacă nu există corespondență în tabelul din dreapta, vor fi returnate NULL pentru coloanele din dreapta. Este util atunci când vrei să vezi toate datele din tabelul principal, chiar dacă nu există corespondență în al doilea tabel, cum ar fi toate comenzile, inclusiv cele care nu au o confirmare de livrare.

**RIGHT (OUTER) JOIN**: Similar cu LEFT JOIN, dar returnează toate rândurile din tabelul din dreapta și rândurile corespunzătoare din stânga. Este mai puțin utilizat, dar necesar când ordinea tabelelor este inversă față de cazul unui LEFT JOIN.

**FULL (OUTER) JOIN**: Returnează toate rândurile atunci când există o corespondență fie în tabelul din stânga, fie în cel din dreapta. Dacă nu există corespondență, NULL va fi prezent pentru valorile lipsă. Este utilizat pentru a combina toate datele din ambele tabele și a evidenția toate discrepanțele.

**CROSS JOIN**: Returnează produsul cartezian al tabelelor implicate, adică fiecare rând din primul tabel este combinat cu fiecare rând din al doilea tabel. Este rar utilizat, dar poate fi util pentru a genera combinații de date pentru testare sau simulare.

### **Întrebare 2:** Ce sunt subinterogările și cum le-ai folosi într-o analiză?

**Subinterogări simple**: Returnează un singur rând și o singură coloană. De exemplu, pentru a afla salariul mediu și a selecta angajații care au un salariu peste medie.

**Subinterogări corelate**: Interogarea internă depinde de rândul curent al interogării principale. De exemplu, pentru a găsi toți angajații care au salariul maxim din fiecare departament.

**Subinterogări multiple**: Returnează mai multe rânduri și/sau coloane. De exemplu, pentru a selecta toți clienții care au făcut comenzi în ultimele 30 de zile.

### **Întrebare 3:** Explică funcțiile analitice în SQL și dă un exemplu.

**Funcțiile analitice** în SQL sunt folosite pentru a efectua calcule pe un set de rânduri care sunt legate de interogarea curentă, adesea pentru a obține rezultate cumulative, de rang sau de fereastră. Spre deosebire de funcțiile de agregare, funcțiile analitice nu agregă rezultatele în rânduri unice, ci păstrează rândurile de bază și adaugă o coloană nouă cu rezultatele.

**EXEMPLU:**

SELECT EmployeeID, Salary,

ROW\_NUMBER() OVER (PARTITION BY Department ORDER BY Salary DESC) AS Rank

FROM Employees;

### **Întrebare 4:** Cum poți optimiza o interogare SQL?

**Utilizarea indexilor**: Crearea de indecși pe coloanele utilizate frecvent în condițiile WHERE, JOIN sau ORDER BY poate accelera semnificativ interogările.

**Evitarea subinterogărilor complexe**: Înlocuirea subinterogărilor corelate sau a celor redundante cu JOIN-uri simple poate reduce timpul de execuție.

**Filtrarea timpurie**: Aplicarea condițiilor de filtrare cât mai devreme în interogare (WHERE sau HAVING) reduce numărul de rânduri procesate ulterior.

**Reducerea utilizării de funcții în WHERE**: Funcțiile aplicate pe coloane în condiția WHERE pot împiedica utilizarea indexilor, deci e preferabil să fie evitate.

**Explorarea planului de execuție**: Utilizarea comenzilor precum EXPLAIN sau EXPLAIN ANALYZE pentru a examina planul de execuție al interogării și a identifica eventualele puncte slabe sau ineficiențe.

### **Întrebare 5:** Poți oferi un exemplu de când ai folosit un trigger în SQL?

Un trigger este o procedură SQL care se execută automat ca răspuns la un anumit eveniment într-un tabel, cum ar fi inserarea, actualizarea sau ștergerea datelor.

Am folosit triggere în Oracle SQL, acesta a fost folosit pentru auto-incrementarea valorilor introduse în tabele și crearea coloanelor de ID pentru nu fi repetitive. Aplicația pe care am dezvoltat-o avea la baza 3 tabele fiecare având nevoie de ID-uri unice pentru a putea îndeplinii condițiile impuse în timpul manipulării datelor.

**EXEMPLU:**

–crearea tabelului Masina fara a avea auto-increment

CREATE TABLE Masina (

idMasina NUMBER(3) PRIMARY KEY,

marca VARCHAR2(20) NOT NULL,

model VARCHAR2(20) NOT NULL,

anFabricatie NUMBER(4) NOT NULL,

capacitateCilindrica NUMBER(4) NOT NULL,

tip\_combustibil VARCHAR2(20) NOT NULL,

CONSTRAINT chk\_tip\_combustibil CHECK (tip\_combustibil IN ('benzina', 'GPL', 'diesel', 'electric', 'hibrid')),

kilometraj NUMBER default 0 not null,

pretInchiriere NUMBER(4) NOT NULL,

culoare VARCHAR2(13) NOT NULL

);

–crearea secventei ce va fi folosita de catre trigger

CREATE SEQUENCE seq\_masina\_id START WITH 1 INCREMENT BY 1;

– cream trigger pentru autoincrement

CREATE OR REPLACE TRIGGER trg\_masina\_id

BEFORE INSERT ON Masina

FOR EACH ROW

BEGIN

–atribuim idMasina urmatoarea valoare din secventă

:new.idMasina := seq\_masina\_id.NEXTVAL;

END;