Funcții și Proceduri Stocate în MySQL

1 Introducere

Funcțiile și procedurile stocate (stored procedures) sunt elemente esențiale ale sistemelor de baze de date relaționale, permițând reutilizarea codului și implementarea logicii de afaceri direct la nivelul bazei de date. Acestea contribuie la creșterea performanței și reducerea traficului între aplicație și baza de date.

2 Funcții în MySQL

O funcție este un set de instrucțiuni SQL care primește parametri, efectuează o operație și returnează o singură valoare.

2.1 Sintaxa Generală

```
CREATE FUNCTION nume_functie(parametru_tip)
RETURNS tip_date
DETERMINISTIC -- sau NOT DETERMINISTIC
BEGIN
    -- corpul functiei
    RETURN valoare;
END;
```

2.2 Exemplu 1: Funcție care Calculează Factorialul

Scop: Calculați factorialul unui număr dat.

```
DELIMITER $$

CREATE FUNCTION Factorial(n INT)
RETURNS BIGINT
DETERMINISTIC
BEGIN
    DECLARE rezultat BIGINT DEFAULT 1;
    WHILE n > 1 DO
        SET rezultat = rezultat * n;
        SET n = n - 1;
    END WHILE;
    RETURN rezultat;
END$$

DELIMITER ;

-- Utilizare
SELECT Factorial(5); -- Rezultat: 120
```

2.3 Exemplu 2: Funcție pentru Calculul Discountului

Scop: Returnați prețul final aplicând un discount specificat.

```
DELIMITER $$

CREATE FUNCTION CalculeazaDiscount(pret DECIMAL(10, 2), procent INT)
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    RETURN pret - (pret * procent / 100);
END$$

DELIMITER ;

-- Utilizare
SELECT CalculeazaDiscount(100, 10); -- Rezultat: 90
```

2.4 Considerații pentru Funcții

- Funcțiile trebuie să fie deterministice pentru a putea fi utilizate în expresii sau indexuri. - Acestea nu pot modifica date (de exemplu, nu pot conține comenzi `INSERT`, `UPDATE` sau `DELETE`).

3 Proceduri Stocate în MySQL

O procedură stocată este o colecție de instrucțiuni SQL care poate efectua operațiuni complexe și nu este limitată la returnarea unei singure valori.

3.1 Sintaxa Generală

```
CREATE PROCEDURE nume_procedura([parametru IN/OUT/INOUT tip])
BEGIN
    -- corpul procedurii
END;
```

3.2 Exemplu 1: Procedură pentru Inserarea unui Student

Scop: Introduceți un student în tabelul `Student`.

```
DELIMITER $$

CREATE PROCEDURE AdaugaStudent(
    IN id INT,
    IN nume VARCHAR(100),
    IN varsta INT
)
BEGIN
    INSERT INTO Student (StudentID, Nume, Varsta)
    VALUES (id, nume, varsta);
END$$

DELIMITER ;

-- Utilizare
CALL AdaugaStudent(1, 'Ion Popescu', 20);
```

3.3 Exemplu 2: Procedură pentru Actualizarea Salariilor

Scop: Creșteți salariile tuturor angajaților cu un procent specificat.

```
DELIMITER $$

CREATE PROCEDURE ActualizeazaSalarii(
    IN procent INT
)
BEGIN
    UPDATE Angajat
    SET Salariu = Salariu + (Salariu * procent / 100);
END$$

DELIMITER ;

-- Utilizare
CALL ActualizeazaSalarii(10); -- Creste salariile cu 10%
```

3.4 Exemplu 3: Procedură cu Parametri IN și OUT

Scop: Determinați media notelor unui student.

```
DELIMITER $$

CREATE PROCEDURE MediaNote(
    IN student_id INT,
    OUT media DECIMAL(5, 2)
)
BEGIN
    SELECT AVG(Nota)
    INTO media
    FROM Inscriere
    WHERE StudentID = student_id;
END$$

DELIMITER ;

-- Utilizare
CALL MediaNote(1, @media);
SELECT @media; -- Rezultatul mediei
```

4 Diferențe între Funcții și Proceduri

Caracteristică	**Funcție**	**Procedură Stocată**
Returnează valoare	Da, o singură valoare	Nu, dar poate utiliza parametri OUT
Modificare date	Nu	Da
Utilizare în expresii	Da	Nu
Complexitate operații	Simplă	Complexă

Tabela 1: Diferențe între funcții și proceduri stocate

5 Exerciții Propuse

5.1 Exerciții pentru Funcții

1. Creați o funcție `CalculeazaTVA` care primește ca parametru un preț și returnează valoarea TVA-ului (19% din preț).
2. Definiți o funcție `SumaCifre` care primește un număr întreg ca parametru și returnează suma cifrelor acestuia.
3. Scrieți o funcție `PretFinal` care primește un preț și un discount (procent) și returnează prețul final.
4. Creați o funcție `NotaMaxima` care primește un `StudentID` și returnează cea mai mare notă obținută de acel student.
5. Implementați o funcție `ZileRamase` care calculează numărul de zile rămase până la o dată specificată.

5.2 Exerciții pentru Proceduri Stocate

1. Scrieți o procedură `AdaugaCurs` care introduce un nou curs în tabela `Curs`.
2. Creați o procedură `ActualizeazaSalariu` care primește un procent și mărește salariile tuturor angajaților cu procentul specificat.
3. Implementați o procedură `StergeStudent` care șterge un student pe baza `StudentID`-ului primit ca parametru.
4. Definiți o procedură `RaportStudent` care primește un `StudentID` și returnează toate informațiile despre cursurile urmate de acel student.
5. Creați o procedură `TransferCursuri` care transferă toate cursurile unui student către alt student (doi parametri: `StudentID_sursa` și `StudentID_destinatie`).

6 Concluzie

Funcțiile și procedurile stocate sunt instrumente puternice pentru gestionarea logicii de afaceri direct în MySQL. Funcțiile sunt utilizate pentru operații simple care returnează o singură valoare, în timp ce procedurile stocate sunt mai versatile și permit operațiuni complexe, inclusiv manipularea datelor. Utilizarea acestor mecanisme poate optimiza performanța și reduce complexitatea aplicațiilor.