Lesson 7.4: Stored Procedures & Functions

CSC430/530 – DATABASE MANAGEMENT SYSTEMS

OUTLINE

- Stored procedures.
  - Introduction.
  - Syntax.
- Functions.
  - Introduction.
  - Syntax.

STORED PROCEDURES: INTRO

- **Stored procedure -** set of **pre-compiled SQL statements** grouped under a **name** and stored in the **database**.
  - A **mini-program** that performs a **specific task** when called.
- **Benefits** of using stored procedures:
  - **Reusability**.
  - **Maintainability**.
  - **Security**.
  - **Performance gains**.
  - **Reduced network traffic**.

STORED PROCEDURES: SYNTAX

- General syntax of a stored procedure:

```
CREATE PROCEDURE procedure_name (IN input_parameter datatype, OUT output_parameter datatype)
BEGIN
  -- SQL statements
END;
```

- IN parameters.
  - Pass data from calling code into stored procedure.
- OUT parameters.
  - Return data from stored procedure back to calling code.
- INOUT parameters.
  - Procedure modifies provided data and returns updated data.
- SQL statements.
  - Any DDL or DML commands.

- **Example:**

```
DELIMITER $$
CREATE PROCEDURE GetEmployeeCount()
BEGIN
   SELECT COUNT(*) AS TotalEmployees
   FROM Employee;
END $$
DELIMITER ;
```

- **Execution**:

```
CALL GetEmployeeCount();
```

STORED PROCEDURES: EXAMPLE

- Create a stored procedure to get an employee full name given an SSN.

```
/* create the procedure */
DELIMITER $$
CREATE PROCEDURE GetEmployeeName(IN empSSN INT, OUT empName VARCHAR(50))
BEGIN
    /* "INTO" is used to put a value into our OUT parameter */
    SELECT CONCAT(fname, ' ', lname) INTO empName
    FROM Employee
    WHERE Ssn = empSSN;
END $$
DELIMITER ;

/* use the procedure */
CALL GetEmployeeName(123456789, @emp_name);
SELECT @emp_name;
```

STORED PROCEDURES: EXAMPLE

- Create a stored procedure to get the total number of hours an employee works on all projects.

```
DELIMITER $$
CREATE PROCEDURE GetEmployeeHours (IN empFname VARCHAR(20),
                                   IN empLname VARCHAR(20),
                                   OUT empHours DECIMAL(4,1))
BEGIN
    SELECT SUM(w.Hours) INTO empHours
    FROM Employee e, Works_on w
    WHERE e.Ssn = w.Essn AND
          e.Fname = empFname AND
          e.Lname = empLname;
END $$
DELIMITER ;

CALL GetEmployeeHours("John", "Smith", @emp_hrs);
SELECT @emp_hrs;
```

STORED PROCEDURES: EXAMPLE

- Create a stored procedure to raise the minimum salary level of all employees.

```
DELIMITER $$
CREATE PROCEDURE SetMinSalary(IN minSalary INT)
BEGIN
    UPDATE employee
    SET salary = minSalary
    WHERE salary < minSalary;
END$$
DELIMITER ;


CALL SetMinSalary(31000);
```

STORED PROCEDURES

- Can be used to carry out any SQL commands (INSERT, UPDATE, DELETE, and SELECT)
- Can take in parameters (using IN)
- Can give back results with out parameters (using OUT)

FUNCTIONS: INTRO

- **Function - named operations** that accept **input** and return a **single value** or a **result set** as an output.
  - Focus on **calculations** / **data transformations**.
  - **Do not** directly **modify data** within tables.
  - Encapsulate common **calculations** in a **single unit**.
- Benefits of using functions:
  - **Reusability**.
  - **Maintainability**.
  - **Readability**.
  - **Modularity**.

FUNCTIONS: SYNTAX (1)

- **General syntax of a function:**

```
CREATE FUNCTION function_name (parameter_name datatype)
RETURNS datatype
CHARACTERISTIC
BEGIN
    -- SQL statements
END;
```

- Parameters.
  - Variables that accept input values during function call.
- RETURNS.
  - A return value is mandatory.
  - Data type must be specified.
- SQL statements.
  - SELECT and data manipulation commands.

FUNCTIONS: SYNTAX (1)

- **General syntax of a function:**

```
CREATE FUNCTION function_name (parameter_name datatype)
RETURNS datatype
CHARACTERISTIC
BEGIN
    -- SQL statements
END;
```

- CHARACTERISTIC
  - DETERMINISTIC – given the same input and database state, the same result(s) will be produced
  - NOT DETERMINISTIC – results may be different each time this function is executed even if input and database state haven't changed
  - READS SQL DATA – function may contain queries to retrieve data from database (such as SELECT)
  - MODIFIES SQL DATA – function may *indirectly* modify data in database (such as calling a stored procedure that performs modifications). Note however that a function cannot directly use UPDATE, INSERT, nor DELETE commands.
  - CONTAINS SQL – function contains SQL code but does not attempt to read nor modify data from database
  - NO SQL – function does not contain any SQL code (some versions of SQL allow other languages to be used here)

FUNCTIONS: EXAMPLE

```
/* Create */
DELIMITER $$
CREATE FUNCTION NameBySSN(empSSN VARCHAR(9))
RETURNS VARCHAR(20)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE empName VARCHAR(20);

    SELECT Fname INTO empName
    FROM Employee
    WHERE Ssn = empSSN;

    RETURN empName;
END $$
DELIMITER ;

/* Use */
SELECT NameBySSN(123456789);
```

FUNCTIONS: EXAMPLE

- Create a function to calculate the age of an employee given the date of birth.

```
DELIMITER $$
CREATE FUNCTION GetAge(DOB DATE)
RETURNS INT
NOT DETERMINISTIC
CONTAINS SQL
BEGIN
    DECLARE today DATE;
    SET today = CURDATE();
    RETURN YEAR(today) - YEAR(DOB);
END$$
DELIMITER;

SELECT GetAge( (SELECT Bdate FROM Employee WHERE Ssn = 123456789) );
```

FUNCTIONS: EXAMPLE

- Create a function to calculate the number of the employees for a given department.

```
/* Create */
DELIMITER $$
CREATE FUNCTION GetEmployeesByDepartment(deptNo INT)
RETURNS INT
DETERMINISTIC
READS SQL DATA
BEGIN
    RETURN (SELECT COUNT(*)
            FROM Employee
            WHERE Dno = deptNo);
END$$
DELIMITER ;

/* Use */
SELECT GetEmployeesByDepartment(5);
```

FUNCTIONS: BUILT IN

- MySQL has many built in functions. Here is a quick reference of some of them:

FUNCTION	PURPOSE
SUM, COUNT, MAX, MIN, AVG	Aggregates collections of values by group
MONTH, DAY, YEAR	Extracts the numeric month, day or year from a date
HOUR, MINUTE, SECOND	Extracts the numeric hour, minute or second from a time
NOW	Returns the current date and time
CONCAT, CONCAT_WS	Concatenates multiple strings, or concatenates them with a custom separator (CONCAT_WS)
LCASE / LOWER, UCASE / UPPER	Changes case of string to lower or upper case
LTRIM, RTRIM	Remove the leading/left space (LTRIM) or removes the trailing/right space (RTRIM)
ROUND, TRUNCATE	Mathematically rounds or truncates a float
LEFT, RIGHT, MID / SUBSTR	Grabs the left, right, or middle of a string using specified starting/stopping points
FORMAT	Returns a string version of the given number with thousands separator and rounded to custom number of decimal places

Reference: https://www.techonthenet.com/mysql/functions/index.php

What's The Difference?

Triggers	Views	Stored Procedures	Functions
Invoked automatically when the trigger event happens	Invoked when you use them in FROM clause	Invoked when you CALL them	Invoked when you use them in SELECT or WHERE clause
Can execute any SQL commands	Can only SELECT (acts as a virtual table)	Can execute any SQL commands	Can SELECT and calculate
Can stop an operation from occurring		Can take parameters	Can take parameters
Can alter what an operation does		Can produce results (as OUT parameter)	Can produce a single result (as a returned value)

SUMMARY

- Purpose of stored procedures.
- Stored procedures syntax.
- Purpose of functions.
- Functions syntax.