Maulik Bhatt

Dynamic SQL

Contents

[Dynamic SQL in MySQL and PostgreSQL 2](#_Toc191376577)

[1. Dynamic SQL in MySQL 2](#_Toc191376578)

[Example of Dynamic SQL in MySQL Stored Procedure: 2](#_Toc191376579)

[2. Dynamic SQL in PostgreSQL 2](#_Toc191376580)

[Example of Dynamic SQL in PostgreSQL Function: 2](#_Toc191376581)

[3. Comparison between MySQL and PostgreSQL 3](#_Toc191376582)

[1. Dynamic SQL in Stored Functions: 3](#_Toc191376583)

[2. Dynamic SQL in Stored Procedures: 3](#_Toc191376584)

[3. Flexibility: 3](#_Toc191376585)

# Dynamic SQL in MySQL and PostgreSQL

## 1. Dynamic SQL in MySQL

In MySQL, dynamic SQL can be executed within stored procedures using the PREPARE, EXECUTE, and DEALLOCATE statements. These statements allow you to build a SQL query dynamically as a string and then execute it.

### Example of Dynamic SQL in MySQL Stored Procedure:

DELIMITER $$  
  
CREATE PROCEDURE dynamic\_procedure()  
BEGIN  
 DECLARE @sql\_query VARCHAR(1000);  
 SET @sql\_query = 'SELECT \* FROM your\_table WHERE your\_column = ''some\_value''';  
  
 PREPARE stmt FROM @sql\_query;  
 EXECUTE stmt;  
 DEALLOCATE PREPARE stmt;  
END$$  
DELIMITER ;

MySQL does not support using dynamic SQL in stored functions that return values. Functions in MySQL are expected to return deterministic values without side effects. Therefore, dynamic SQL is not recommended for functions.

## 2. Dynamic SQL in PostgreSQL

In PostgreSQL, dynamic SQL can be executed in both stored functions and stored procedures using the EXECUTE command. This command allows you to run dynamically constructed SQL queries within PL/pgSQL blocks.

### Example of Dynamic SQL in PostgreSQL Function:

CREATE OR REPLACE FUNCTION dynamic\_function()  
RETURNS INT AS $$  
DECLARE  
 result INT;  
 sql\_query TEXT;  
BEGIN  
 sql\_query := 'SELECT COUNT(\*) FROM your\_table';  
  
 EXECUTE sql\_query INTO result;  
 RETURN result;  
END;  
$$ LANGUAGE plpgsql;

Dynamic SQL in PostgreSQL is more flexible compared to MySQL since it allows the use of the EXECUTE command in both functions and procedures. You can also use the USING clause for parameterized queries and return result sets using RETURN QUERY.

## 3. Comparison between MySQL and PostgreSQL

### 1. Dynamic SQL in Stored Functions:

* **MySQL**: Does not support dynamic SQL in stored functions that return a value.
* **PostgreSQL**: Allows dynamic SQL in functions using the EXECUTE command.

### 2. Dynamic SQL in Stored Procedures:

* **Both MySQL and PostgreSQL**: Support dynamic SQL in stored procedures using the PREPARE and EXECUTE commands.
* **MySQL**: Requires the use of PREPARE, EXECUTE, and DEALLOCATE.
* **PostgreSQL**: Uses the simpler EXECUTE command.

### 3. Flexibility:

* **PostgreSQL**: Provides more flexibility with dynamic SQL in functions and allows result sets to be returned with RETURN QUERY.
* **MySQL**: Dynamic SQL is more limited, and you cannot use it in functions that return values.