CS157A: Introduction to Database Management Systems

MySQL Stored Procedures

Suneuy Kim

Stored Procedures

- Persistent, Stored Modules (PSM): part of the latest version to the SQL standard, SQL:2003
- A method to encapsulate repetitive tasks.
- It allows for useful programming techniques, including variable declarations, flow control, etc.
- It is stored in the database, as part of the schema

MySQL Stored Procedures

To create

```
DROP PROCEDURE IF EXISTS GetAllFaculty;
DELIMITER //
CREATE PROCEDURE GetAllFaculty()
BEGIN
SELECT * FROM Faculty;
END//
DELIMITER ;
```

→ Change the standard delimiter to //

→ Change it back to the standard ;

To call

```
CALL GetALLFaculty;
```

MySQL Stored Procedure Variables

```
DECLARE total_count INT DEFAULT 0
SET total_count = 10;
```

Variable Scope

- DECLARE is permitted only inside a BEGIN ... END compound statement and must be at its start, before any other statements.
- It will be out of scope if the END is reached.

MySQL Stored Procedure Parameters

- IN – A parameter whose value is unknown when the SQL statement is created. A caller passes an argument to it. Call by value.
- OUT- A parameter whose value is supplied by the SQL statement it returns.
- INOUT-A parameter that provides both input and output value.

IN parameter:Example

```
DROP PROCEDURE IF EXISTS getFacultyByName;
DELIMITER //
CREATE PROCEDURE getFacultyByName(IN
facultyName VARCHAR(50))
BEGIN
SELECT *
FROM Faculty
WHERE name=facultyName;
END//
DELIMITER ;
```

```
CALL getFacultyByName('Margaret A. Steele');
```

OUT parameter:Example

```
DROP PROCEDURE IF EXISTS countByAge;
DELIMITER //
CREATE PROCEDURE countByAge(IN retirementAge
INT, OUT total INT)
BEGIN
SELECT count(*)
INTO total
FROM Faculty
WHERE retirementAge < age;
END//
DELIMITER ;
```

```
Call countByAge(60, @result);
Select @result;
```

INOUT parameter:Example

```
DROP PROCEDURE IF EXISTS set_counter;
DELIMITER //
CREATE PROCEDURE set_counter(INOUT count
INT(4),IN inc INT(4))
BEGIN
SET count = count + inc;
END //
DELIMITER ;

SET @mycount = 1;
CALL set_counter(@mycount, 5);
SELECT @mycount;
```