STORED PROCEDURE AND TRIGGERS

EGCO321 DATABASE SYSTEMS

KANAT POOLSAWASD
DEPARTMENT OF COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

STORED PROCEDURES

- MySQL is known as the most popular open source RDBMS which is widely used by both community and enterprise. However, during the first decade of its existence, it did not support stored procedures, stored functions, triggers, and events. Since MySQL version 5.0, those features were added to MySQL database engine to make it more flexible and powerful.

STORED PROCEDURES ADVANTAGES (1)

- Typically stored procedures help increase the performance of the applications. Once created, stored procedures are compiled and stored in the database.
- However, MySQL implements the stored procedures slightly different. MySQL stored procedures are compiled on demand. After compiling a stored procedure, MySQL puts it into a cache. And MySQL maintains its own stored procedure cache for every single connection.
- If an application uses a stored procedure multiple times in a single connection, the compiled version is used, otherwise, the stored procedure works like a query.

STORED PROCEDURES ADVANTAGES (2)

- Stored procedures help reduce the traffic between application and database server because instead of sending multiple lengthy SQL statements, the application has to send only name and parameters of the stored procedure.
- Stored procedures are reusable and transparent to any applications. Stored procedures expose the database interface to all applications so that developers don't have to develop functions that are already supported in stored procedures.
- Stored procedures are secure. The database administrator can grant appropriate permissions to applications that access stored procedures in the database without giving any permissions on the underlying database tables.

STORED PROCEDURES DISADVANTAGES (1)

- If you use a lot of stored procedures, the memory usage of every connection that is using those stored procedures will increase substantially. In addition, if you overuse a large number of logical operations inside store procedures, the CPU usage will also increase because the database server is not well-designed for logical operations.
- Constructs of stored procedures make it more difficult to develop stored procedures that have complicated business logic.

STORED PROCEDURES DISADVANTAGES (2)

- It is difficult to debug stored procedures. Only a few database management systems allow you to debug stored procedures. Unfortunately, MySQL does not provide facilities for debugging stored procedures.
- It is not easy to develop and maintain stored procedures. Developing and maintaining stored procedures are often required a specialized skill set that not all application developers possess. This may lead to problems in both application development and maintenance phases.

CREATE STORED PROCEDURES

- A stored procedure contains a sequence of SQL commands stored in the database catalog so that it can be invoked later by a program
- Stored procedures are declared using the following syntax:

```
CREATE PROCEDURE <proc-name>
    (param_spec1,param_spec2,…,param_specn)
BEGIN
 -- execution code
END;
```

EXAMPLE 1 (1)

```
CREATE TABLE test (
    id CHAR(4) NOT NULL,
    fullname VARCHAR(20),
    sex CHAR(1),
    age INT,
    PRIMARY KEY(id)
);
```

And

```
INSERT INTO test VALUES('1111', 'Adam', 'M', 22);
INSERT INTO test VALUES('1234', 'David', 'M', 20);
INSERT INTO test VALUES('4321', 'John', 'M', 18);
INSERT INTO test VALUES('4444', 'Jane', 'F', 19);
```

EXAMPLE 1 (2)

```
DELIMITER $$
CREATE PROCEDURE spShowAll ()
BEGIN
   SELECT * FROM test;
END
```

Or

```
CREATE PROCEDURE spShowAll () SELECT * FROM test;
```

CALL THE PROCEDURE

- To call a procedure, you have to do is enter the word CALL and then the name of the procedure and then the parentheses.

```
Call spShowAll();
```

DISPLAY STORED PROCEDURES

- To display characteristics of a stored procedure, you use the SHOW PROCEDURE STATUS statement as follows:

```
SHOW PROCEDURE STATUS [LIKE 'pattern' |
WHERE expr];

SHOW PROCEDURE STATUS;
SHOW PROCEDURE STATUS
  WHERE Name LIKE '%spShow%'
```

STORED PROCEDURE PARAMETER TYPES

- **IN.** This is the default (if not specified). It is passed to the routine and can be changed inside the routine, but remains unchanged outside.
- **OUT.** No value is supplied to the routine (it is assumed to be NULL), but it can be modified inside the routine, and it is available outside the routine.
- **INOUT.** The characteristics of both IN and OUT parameters. A value can be passed to the routine, modified there as well as passed back again.

EXAMPLE 2

```
DELIMITER $$
CREATE PROCEDURE spShowSex (IN con CHAR(1))
BEGIN
  SELECT * FROM test WHERE sex = con;
END

CALL spShowSex('F');
```

DECLARING VARIABLES

- To declare a variable inside a stored procedure, you use the DECLARE statement as follows:

```
DECLARE variable_name datatype(size)
    DEFAULT default_value;

DECLARE sum INT DEFAULT 0;
```

EXAMPLE 3

```
DROP PROCEDURE IF EXISTS spShowAge;
DELIMITER $$
CREATE PROCEDURE spShowAge (IN S INT)
BEGIN
    DECLARE X INT;
    SET X = S;
    SELECT * FROM test WHERE age >= X;
END
```

```
CALL spShowAge(20);
```

EXAMPLE 4

```
DELIMITER $$
CREATE PROCEDURE spCountAge(IN S INT, OUT n INT)
BEGIN
   SELECT COUNT(*) INTO n FROM test WHERE Age>=S;
END
```

```
DELIMITER $$
CREATE PROCEDURE spCount(IN C INT)
BEGIN
    DECLARE N INT;
    CALL spCountAge(C,N);
    SELECT N;
END
```

```
CALL spCount(20);
```

CONDITIONS

- MySQL provides both IF and CASE statements to enable you to execute a block of SQL code based on certain conditions, which is known as flow control.

CONDITION: *IF-THEN-ELSE*

```
CREATE PROCEDURE p1 (IN parameter1 INT)
BEGIN
  DECLARE variable1 INT;
  SET variable1 = parameter1 + 1;
  IF variable1 = 0 THEN
    INSERT INTO t VALUES (17);
  END IF;
  IF parameter1 = 1 THEN
    UPDATE t SET s1 = s1 + 1;
  ELSE
    UPDATE t SET s1 = s1 + 2;
  END IF;
END;
```

CONDITION: *CASE*

```
CREATE PROCEDURE p2 (IN parameter1 INT)
BEGIN
  DECLARE variable1 INT;
  SET variable1 = parameter1 + 1;
  CASE variable1
    WHEN 0 THEN INSERT INTO t VALUES (17);
    WHEN 1 THEN INSERT INTO t VALUES (18);
    ELSE INSERT INTO t VALUES (19);
  END CASE;
END;
```

LOOPS

- MySQL provides loop statements that allow you to execute a block of SQL code repeatedly based on a condition. There are three loop statements in MySQL: WHILE, REPEAT and LOOP.

LOOPS: *WHILE ... END WHILE*

```
CREATE PROCEDURE p3 ()
BEGIN
   DECLARE v INT;
   SET v = 0;
   WHILE v < 5 DO
      INSERT INTO t VALUES (v);
      SET v = v + 1;
   END WHILE;
END;
```

LOOPS: *REPEAT ... UNTIL*

```
CREATE PROCEDURE p4 ()
BEGIN
  DECLARE v INT;
  SET v = 0;
  REPEAT
    INSERT INTO t VALUES (v);
    SET v = v + 1;
    UNTIL v >= 5
  END REPEAT;
END;
```

LOOPS: *LOOP ... END LOOP*

```
CREATE PROCEDURE p5 ()
BEGIN
  DECLARE v INT;
  SET v = 0;
  loop_label: LOOP
    INSERT INTO t VALUES (v);
    SET v = v + 1;
    IF v >= 5 THEN
      LEAVE loop_label;
    END IF;
  END LOOP;
END;
```

STORED FUNCTIONS

- A stored function is a special kind stored program that returns a single value. You use stored functions to encapsulate common formulas or business rules that are reusable among SQL statements or stored programs.

STORED PROCEDURE VS. FUNCTION

Stored Procedure (SP)	User Defined Function (UDF)
SP can return zero , single or multiple values	Function must return a single value.
We can use transaction in SP.	We can't use transaction in UDF.
SP can have input/output parameter	Only input parameter.
We can call function from SP.	We can't call SP from function.
We can't use SP in SELECT/WHERE/ HAVING statement.	We can use UDF in SELECT/WHERE/ HAVING statement.
We can use exception handling using Try-Catch block in SP.	We can't use Try-Catch block in UDF.

STORED FUNCTION SYNTAX

- The following illustrates the simplest syntax for creating a new stored function:

```
CREATE FUNCTION function_name(param1,param2,…)
RETURNS datatype
DETERMINISTIC | NOT DETERMINISTIC
BEGIN
    statements
    …
END
```

EXAMPLE 5 (1)

```
DELIMITER $$
CREATE FUNCTION ShowSex(sex CHAR(1))
RETURNS VARCHAR(10)
DETERMINISTIC
BEGIN
    DECLARE s VARCHAR(10);
    IF sex = 'F' THEN
        SET s = 'Female';
    ELSE
        SET s = 'Male';
    END IF;
   RETURN (s);
END
```

EXAMPLE 5 (2)

```
SELECT id, fullname, ShowSex(sex), age
  FROM test
```

EXAMPLE 6 (1)

```
DELIMITER $$
CREATE FUNCTION avgAge(x INT)
RETURNS FLOAT
DETERMINISTIC
BEGIN
    DECLARE a FLOAT;
    DECLARE b FLOAT;
    SELECT AVG(age) INTO a FROM test;
    SET b = x - a;
    RETURN(b);
END
```

EXAMPLE 6 (2)

```
SELECT id, fullname, ShowSex(sex), avgAge(age)
     FROM test
```

SQL TRIGGERS

- A SQL trigger is a set of SQL statements stored in the database catalog.
- A SQL trigger is executed or fired whenever an event associated with a table occurs e.g., insert, update or delete.
- A SQL trigger is a special type of stored procedure. It is special because it is not called directly like a stored procedure. The main difference between a trigger and a stored procedure is that a trigger is called automatically when a data modification event is made against a table whereas a stored procedure must be called explicitly.

ADVANTAGES OF USING SQL TRIGGERS

- SQL triggers provide an alternative way to check the integrity of data.
- SQL triggers can catch errors in business logic in the database layer.
- SQL triggers provide an alternative way to run scheduled tasks. By using SQL triggers, you don't have to wait to run the scheduled tasks because the triggers are invoked automatically before or after a change is made to the data in the tables.
- SQL triggers are very useful to audit the changes of data in tables.

DISADVANTAGES OF USING SQL TRIGGERS

- SQL triggers only can provide an extended validation and they cannot replace all the validations. Some simple validations have to be done in the application layer.
  - For example, you can validate user's inputs in the client side by using JavaScript or in the server side using server-side scripting languages such as JSP, PHP, ASP.NET, Perl, etc.
- SQL triggers are invoked and executed invisible from the client applications, therefore, it is difficult to figure out what happen in the database layer.
- SQL triggers may increase the overhead of the database server.

CREATE SQL TRIGGERS

- To monitor a database and take a corrective action when a condition occurs
- Examples:
  - Charge $10 overdraft fee if the balance of an account after a withdrawal transaction is less than $500
  - Limit the salary increase of an employee to no more than 5% raise

```
CREATE TRIGGER trigger-name
trigger-time trigger-event
ON table-name
FOR EACH ROW
   trigger-action;
```

- trigger-time ∈ {BEFORE, AFTER}
- trigger-event ∈ {INSERT, DELETE, UPDATE}

EXAMPLE 7 (1)

```
CREATE TABLE test_log (
  ts DATETIME      NOT NULL,
  id CHAR(4)       NOT NULL,
  act VARCHAR(50) NOT NULL,
  PRIMARY KEY (ts, id)
);
```

EXAMPLE 7 (2)

```
DELIMITER $$
CREATE TRIGGER before_test_update
BEFORE UPDATE ON test FOR EACH ROW
BEGIN
  INSERT INTO test_log
  VALUES(NOW(), OLD.id, 'update');
END
```

EXAMPLE 7 (3)

```
UPDATE test SET age = 24 WHERE id = '4444'
```

```
UPDATE test SET sex = 'F'
   WHERE id LIKE '1%'
```

DISPLAY TRIGGERS

- To display characteristics of a trigger, you use the SHOW TRIGGERS statement as follows:

```
SHOW TRIGGERS [FROM|IN] database_name
[LIKE expr | WHERE expr];


SHOW TRIGGERS;
SHOW TRIGGERS FROM db_name;


USE db_name;
SHOW TRIGGERS LIKE '%test%';
```

ASSIGNMENT 4 (1)

- จงใช้ข้อมูลจาก Database: company เพื่อใช้ในการสร้าง Stored Function และ Trigger ต่อไปนี้
- Stored Function สำหรับคำนวณส่วนลด ตามเงื่อนไขดังนี้
  - ตั้งแต่ 3,000 ขึ้นไป ลด 30% (purch_amt >= 3000)
  - ตั้งแต่ 1,000 ขึ้นไป ลด 20% (purch_amt >= 1000)
  - ตั้งแต่ 100 ขึ้นไป ลด 10% (purch_amt >= 100)
- โดยมี Syntax ดังนี้
  - discount(purch_amt)
- ทดสอบการทำงานโดยใช้ SQL Command เพื่อแสดงข้อมูล ord_no, ord_date, purch_amt, **discount(purch_amt)** AS discount, **purch_amt-discount(purch_amt)** AS total

ASSIGNMENT 4 (2)

- สร้าง Trigger ชื่อ orders_insert ที่ทำงานทุกครั้ง **หลังการเพิ่มข้อมูล** ในตาราง **orders** โดยเพิ่มข้อมูลใหม่ลงในตาราง **orders_audit** ดังนี้
  - log_date (ord_date)
  - ord_no
  - customer_id
  - salesman_id
  - action = ‘INSERT’