MySQL Procedures and Functions

Difference b/w Procedures and Function

1) You cannot mix procedures with ordinary SQL (but you can w/ functions)
2) Functions must return a value, but for procedures its optional
3) Functions can only take input parameters, whereas procedures can have input and output parameters
4) Functions can be called from procedures, but procedures cannot be called from functions.

Procedures

1) Create a simple one-line procedure, and then call it.

```
DROP PROCEDURE IF EXISTS proc_easy;

CREATE PROCEDURE proc_easy()
 SELECT COUNT(*) FROM sailors;

CALL proc_easy;
```

2) Create a multi-line procedure and call it.

```
DROP PROCEDURE IF EXISTS proc_medium;

DELIMITER //

CREATE PROCEDURE proc_medium ()
BEGIN
   SELECT COUNT(*) FROM reservations;
   SELECT COUNT(*) FROM sailors;
   SELECT COUNT(*) FROM boats;
END //

DELIMITER ;
```

```
CALL procmedium;
```

3) Write a procedure with inputs and output parameters.
(input a boat id, and output the number of reservations for that boat)

```
DROP PROCEDURE IF EXISTS numRes;

DELIMITER //

CREATE PROCEDURE numRes(
    IN inBid INT, OUT n INT)
BEGIN
    SELECT COUNT(*) INTO n
    FROM reservations
    WHERE bid = inBid;
END //

DELIMITER ;

SET @result = 0;
SELECT @result; //print the contents
                of the result variable
```

```
CALL numRes (10, @result);
SELECT @result;
```

↳ print output of the procedure

4) Write a multi-line procedure, that has input and output, and creates a temporary table for its results.
(after running this procedure, a table called "temp" will exist in our database)

```
DROP PROCEDURE IF EXISTS tempTable;
DROP TABLE IF EXISTS temp;

DELIMITER //
CREATE PROCEDURE tempTable (
    IN inBid INT)
BEGIN
  CREATE TABLE temp (bid INT, count INT);
  INSERT INTO temp
  SELECT bid, COUNT(*)
  FROM reservations r
  WHERE bid = inBid
  GROUP BY bid;
```

```
END //

DELIMTER ;
CALL tempTable(11);

SELECT * FROM temp;
```

my SQL Functions

- the following assumes the existence of the table: employees (name varchar(20),
  ssn INT,
  age INT,
  salary INT);

1) Write a function that hires a new employee, but checks if their salary is > 10k. and if not, it rejects the hire.

```
DROP FUNCTION IF EXISTS addEmployee;

DELIMTER //

CREATE FUNCTION addEmployee(
```

```
    inName varchar(20),
    inSSN INT,
    inAge INT
    inSalary INT) returns VARCHAR(50)
BEGIN
   IF inSalary < 10000 THEN
      SET @rVal = "Input salary too small.
                   Insertion rejected";
   ELSE
      SET @rVal = "Input okay, inserting...";
      INSERT INTO employees
      VALUES (inName, inSSN, inAge, inSalary);
   END IF;
   RETURN @rVal;
   END //
   DELIMITER ;
```

- to call the above function and print the return value:

```
SELECT addEmployee("Bob", 1234, 25, 5000);
```

- the above function could be expanded to accomplish other tasks as well.

Triggers

- little bits of scripting that are set up to run automatically when certain events happen (i.e. they are "triggered" by certain events)

- suppose we want a trigger that logs changes to employee salaries.

- first lets create a table for the log data

```
CREATE TABLE log (
    ssn INT,
    oldSalary INT,
    newSalary INT );

DROP TRIGGER IF EXISTS salary_update;

DELIMITER //
CREATE TRIGGER salary_update
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
    INSERT INTO log
    VALUES ( NEW.ssn, OLD.salary,
```

NEW.salary);

END //

DELIMITER ;

}

Options for triggers:

BEFORE INSERT ON table } can use NEW but not OLD
AFTER ....
BEFORE UPDATE ON table } can use NEW and OLD
AFTER ....
BEFORE DELETE ON table } can use OLD but not NEW
AFTER ....

- if you make multiple triggers for the same events, what order do they occur in?

→ they will occur in the order that they were created, unless you specify otherwise

ex: FOR EACH ROW
PRECEDES trigger_name

- what if I want a trigger that rejects an insert that doesn't meet a criteria? (eg. reject insert for employees w/ salary < 10,000)?

```
DELIMITER //
CREATE TRIGGER reject_low_salary
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    IF NEW.salary < 10000 THEN
        signal sqlstate "45000";
    END IF;
END //
```

- example of a modification trigger

```
DELIMITER //
CREATE TRIGGER minimum_salary
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    IF NEW.salary < 10000 THEN
        SET NEW.salary = 10000;
    END IF;
```

END //