Databases
HFT Stuttgart

Exercise: Stored Procedures

Hint: You can use the manual „MySQL 5.0 Stored Procedures“ by Peter Gulutzan to help you write some stored procedures.

1. Write a stored function that computes some value (for instance to evaluate some mathematical formula).

2. Write a stored procedure in MySQL, for instance one that computes the yearly salary of a professor, when one provides as input the pName of the professor. A function would be better to do this, but MySQL does not support accessing tables from inside functions. Therefore, one needs an OUT parameter to deliver the result.

3.Use a trigger calling a stored procedure to sort the students that are newly inserted into two different tables. In particular, do the following steps:

a) Extend the schema of the table student by a column called *stipend* whose values will indicate the type of stipend or scholarship a student may have. The default value will be NULL.

b) Define a new table named *ErasmusStudents* to collect students with an Erasmus stipend. The schema of this new table is the same as the schema of the table Student, only without the attribute stipend (becasse only Erasmus students will be collected here).

c) Define a stored procedure *SortOutErasmusStudents* that takes the attributes of a newly created student as input and finds out if this new student is an Erasmus student. If it is an Erasmus student, insert this student into the table ErasmusStudents.

d) Define a BEFORE INSERT trigger for the table student, that calls the procedure SortOutErasmusStudents. The purpose is that before a new student is inserted, it will be checked whether this student is an Erasmus student and should therefore not be inserted into the normal Student table but instead into the ErasmusStudents table.

e) Insert some new students with values of a stipend into the Student table. Test some with an Erasmus stipend and some with a different one (may be NULL).

You will find out that the insert into the Student table happens even if it is an Erasmus student. An Erasmus student will in addition be inserted into the ErasmusStudents table.
To prevent this, the trigger would need to stop the insertion. This is not possible, however. The BEFORE INSERT simply means that the checking and action is performed before the insert, but after that the insert happens anyway.
To achieve that the Erasmus student is not inserted into Student, you must cause an error inside the stored procedure so that it cannot be completed, meaning that the

execution of the trigger cannot be completed successfully, thereby stopping the insertion. MySQL so far does not support the possibility of raising errors or exceptions. Therefore one can deliberately cause some unproblematic error to stop the execution of the stored procedure. For this purpose, in the procedure SortOutErasmusStudents, after the inserting of the student into ErasmusStudents, insert the student again into the Student table. Test what happens when you execute this. What causes the error? Why is it harmless?
Test whether the desired functionality (sorting students into „normal” and „Erasmus”) works now correctly.

Solution Example: Stored Procedures

Problem 1.

```
DELIMITER //

DROP FUNCTION IF EXISTS areaCircle//
CREATE FUNCTION areaCircle(radius DOUBLE(10,2))
RETURNS DOUBLE(10,2)

BEGIN
  DECLARE area DOUBLE(10,2);
  SET area = pi()*radius;
  RETURN area;
END//

DELIMITER ;
```

Problem 2.

```
DELIMITER //

DROP PROCEDURE IF EXISTS profYearSalary//
CREATE PROCEDURE profYearSalary (IN name varchar (15),
            OUT yearSalary int(11))
BEGIN
  SELECT 12 * salary INTO yearSalary
  FROM Professor
  WHERE pName = name;
END;//

CALL profYearSalary ('Langes', @x)//

SELECT @x//

DELIMITER ;
```

Problem 3.

a) Extend the schema of the table student by a column called *stipend* whose values will indicate the type of stipend or scholarship a student may have. The default value will be NULL.

```
ALTER TABLE Student
ADD stipend VARCHAR(15) DEFAULT NULL;
```

b) Define a new table named ErasmusStudents to collect students with an Erasmus stipend. The schema of this new table is the same as the schema of the table Student, only without the attribute stipend (because only Erasmus students will be collected here).

```
CREATE TABLE ErasmusStudents (
matNr INT PRIMARY KEY,
sName VARCHAR(30),
gpa DOUBLE(3,1));
```

c) Define a stored procedure SortOutErasmusStudents that takes the attributes of a newly created student as input and finds out if this new student is an Erasmus student. If it is an Erasmus student, insert this student into the table ErasmusStudents.

```
DROP PROCEDURE IF EXISTS SortOutErasmusStudents//
CREATE PROCEDURE SortOutErasmusStudents (IN matNr INT,
                                         IN sName VARCHAR(30),
                                         IN stipend VARCHAR(15),
                                         IN gpa DOUBLE(3,1) )
BEGIN
   IF stipend = 'Erasmus'
   THEN
         INSERT INTO ErasmusStudents VALUES (matNr, sName, gpa);
   END IF;
END//
```

d) Define a BEFORE INSERT trigger for the table student, that calls the procedure SortOutErasmusStudents. The purpose is that before a new student is inserted, it will be checked whether this student is an Erasmus student and should therefore not be inserted into the normal Student table but instead into the ErasmusStudents table.

```
DROP TRIGGER IF EXISTS sortStudents//
CREATE TRIGGER sortStudents
BEFORE INSERT ON Student
FOR EACH ROW
BEGIN
    CALL SortOutErasmusStudents (NEW.matNr, NEW.sName,
                                 NEW.stipend, NEW.gpa);
END//
```

e) Insert some new students with values of a stipend into the Student table. Test some with an Erasmus stipend and some with a different one (may be NULL).
You will find out that the insert into the Student table happens even if it is an Erasmus student. An Erasmus student will in addition be inserted into the ErasmusStudents table.
To prevent this, the trigger would need to stop the insertion. This is not possible, however. The BEFORE INSERT simply means that the checking and action is performed before the insert, but after that the insert happens anyway.

To achieve that the Erasmus student is not inserted into Student, you must cause an error inside the stored procedure so that it cannot be completed, meaning that the execution of the trigger cannot be completed successfully, thereby stopping the insertion. MySQL so far does not support the possibility of raising errors or exceptions. Therefore one can deliberately cause some unproblematic error to stop the execution of the stored procedure. For this purpose, in the procedure SortOutErasmusStudents, after the inserting of the student into ErasmusStudents, insert the student again into the Student table. Test what happens when you execute this. What causes the error? Why is it harmless?
Test whether the desired functionality (sorting students into „normal” and „Erasmus”) works now correctly.

```
DROP PROCEDURE IF EXISTS SortOutErasmusStudents//
CREATE PROCEDURE SortOutErasmusStudents (IN matNr INT,
                                         IN sName VARCHAR(30),
                                         IN stipend VARCHAR(15),
                                         IN gpa DOUBLE(3,1) )
BEGIN
    IF stipend = ’Erasmus’
    THEN
          INSERT INTO ErasmusStudents VALUES (matNr, sName, gpa);
         /*Double insert provokes an error and prevents insert into student*/
         INSERT INTO Student VALUES (matNr, sName, NULL);

        END IF;
   END//
```

Doing the same thing without a stored procedure, just with a trigger, can be done like this:

```
DROP TRIGGER IF EXISTS sortRecords//
CREATE TRIGGER sortRecords
BEFORE INSERT ON Student
FOR EACH ROW
BEGIN
    IF NEW.stipend = ’Erasmus’
    THEN
         INSERT INTO ErasmusStudents VALUES (NEW.matNr, NEW.sName, NULL);

         /*Double insert provokes an error and prevent insert into student*/
         INSERT INTO Student VALUES (NEW.matNr, NEW.sName,
                                       NULL, NEW.stipend);
   END IF;
END//
```