5. Advanced SQL

5.2.1 Create a Function

Create a function named *BuildingCapacityFct* which takes as input a *Building* of the *Classroom* table in the *University* database and returns the total capacity of the building.

Test the function (i.e. execute an SQL command containing a function call of that function).

Solution:

```
DELIMITER //
CREATE FUNCTION
    BuildingCapacityFct(vBuilding VARCHAR(20))
    RETURNS INT
BEGIN
  DECLARE vMaxCapacity INT;
  SELECT SUM(Capacity) INTO vMaxCapacity
  FROM Classroom WHERE Building = vBuilding;
  RETURN vMaxCapacity;
END //
DELIMITER ;
```

```
SELECT BuildingCapacityFct('Watson');
```

```
SELECT * FROM Classroom
WHERE Capacity > BuildingCapacityFct('Watson');
```

```
DROP FUNCTION BuildingCapacityFct;
```

5.2.2 Procedure with Error Signalling

For the *TimeSlot* table of the *University* database, state informally constraints (besides the key constraint) that should hold between the values of attributes in a single row and between values of attributes of two rows. Check your suggestion with a teaching assistant before continuing.

Create a procedure *InsertTimeSlot* for inserting a row into the *TimeSlot* table. It should signal an error in case the insertion of the new tuple leads to a violation of the constraints. (The procedure should not check whether the constraints already hold before the insertion.) To make the procedure more readable, it is advisable to define one or several auxiliary/helper functions to express the error conditions.

Test systematically the auxiliary functions and procedure.

Solution:

Constraints (desired properties) for *TimeSlot*:

- For all rows it must hold that the StartTime is strictly less than the EndTime.
- If two rows have the same TimeSlotID, their time intervals must not overlap.

In the procedure, one should raise a signal, if the insertion of a new row leads to a violation of these constraints.

```
CREATE FUNCTION timeoverlap
(vDayCode1 ENUM('M','T','W','R','F','S','U'),
vStartTime1 TIME,  vEndTime1 TIME,
vDayCode2 ENUM('M','T','W','R','F','S','U'),
vStartTime2 TIME,  vEndTime2 TIME)
RETURNS BOOLEAN
RETURN  vDayCode1 = vDayCode2 AND
((vstartTime1 <= vStartTime2 AND
  vStartTime2 <= vEndTime1) OR
 (vstartTime2 <= vStartTime1 AND
  vStartTime1 <= vEndTime2));
# assumes vStartTimei <= vEndTimei

#testing timeoverlap function:
#different start
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'T', '08:00:00', '08:50:00'); #should return 0
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'M', '09:00:00', '09:50:00'); #should return 0
#same start
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'M', '08:00:00', '08:40:00'); #should return 1
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'M', '08:00:00', '08:40:00'); #should return 1
#first starts before second on the same day
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'M', '08:10:00', '08:40:00'); #should return 1
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'M', '08:10:00', '08:50:00'); #should return 1
SELECT timeoverlap('M', '08:00:00', '08:50:00',
'M', '08:10:00', '09:00:00'); #should return 1
#second starts before first on the same day
SELECT timeoverlap('M', '08:10:00', '08:40:00',
'M', '08:00:00', '08:50:00'); #should return 1
SELECT timeoverlap('M', '08:10:00', '08:50:00',
'M', '08:00:00', '08:50:00'); #should return 1
SELECT timeoverlap('M', '08:10:00', '09:00:00',
'M', '08:00:00', '08:50:00'); #should return 1
```

```
CREATE FUNCTION timeoverlapWithTable
(vTimeSlotID VARCHAR(4),
vDayCode ENUM('M','T','W','R','F','S','U'),
vStartTime TIME, vEndTime TIME)
RETURNS BOOLEAN
RETURN  EXISTS
(SELECT * FROM  TimeSlot
WHERE TimeSlotID = vTimeSlotID AND
timeoverlap(vDayCode, vStartTime, vEndTime,
DayCode, StartTime, EndTime));

#testing timeoverlapWithTable function:
SELECT timeoverlapWithTable('A', 'M',
'08:10:00', '08:40:00'); #should return 1
SELECT timeoverlapWithTable('A', 'M',
'09:00:00', '09:50:00'); #should return 0
SELECT timeoverlapWithTable('A', 'T', '08:00:00',
'08:50:00'); #should return 0


DELIMITER //
CREATE PROCEDURE InsertTimeSlot
(IN vTimeSlotID VARCHAR(4),
 IN vDayCode ENUM('M','T','W','R','F','S','U'),
 IN vStartTime TIME, IN vEndTime TIME)
BEGIN
  IF  vEndTime <= vStartTime  #bad time interval
  THEN SIGNAL SQLSTATE 'HY000'
               SET MYSQL_ERRNO = 1525,
               MESSAGE_TEXT = 'EndTime is
               equal to or after StartTime';
  END IF;
  IF timeoverlapWithTable(vTimeSlotID,
vDayCode, vStartTime, vEndTime)
  THEN SIGNAL SQLSTATE 'HY000'
               SET MYSQL_ERRNO = 1525,
               MESSAGE_TEXT =
               'time interval overlaps with
               existing timeinterval for the
               same TimeSlotID';
  END IF;
  INSERT TimeSlot
          VALUES (vTimeSlotID,
                  vDayCode,
                  vStartTime,
```

```
                    vEndTime);
END // #END BEGIN
DELIMITER ;

#testing procedure
SELECT * FROM TimeSlot; #
CALL InsertTimeSlot('A', 'T', '08:50:00',
'08:00:00'); # should give error message
'EndTime is equal to or after StartTime'
CALL InsertTimeSlot('A', 'M', '08:50:00',
'08:00:00'); # should give error message
'EndTime is equal to or after StartTime'
CALL InsertTimeSlot('A', 'M', '08:10:00',
'08:40:00'); # should give error message 'time
interval overlaps with existing timeinterval for
the same TimeSlotID'
SELECT * FROM TimeSlot; #no changes in
TimeSlot
CALL InsertTimeSlot('A', 'T', '08:00:00',
'08:50:00'); # is succesfull
SELECT * FROM TimeSlot; #new timeslot is
inserted
```

5.2.3 Trigger with Error Signalling

Make a trigger TimeSlot_Before_Insert, which automatically raises a signal when inserting a row into TimeSlot (directly with an INSERT without using the InsertTimeSlot procedure), if the insertion of the new row leads to a violation of the constraints identified in the previous exercise. Test the trigger by making INSERTs into TimeSLOT.

Solution:

```
DELIMITER //
CREATE TRIGGER TimeSlot_BEFORE_INSERT
BEFORE INSERT ON TimeSlot FOR EACH ROW
BEGIN
  IF  NEW.EndTime <= NEW.StartTime
      #bad time interval
  THEN SIGNAL SQLSTATE 'HY000'
                SET MYSQL_ERRNO = 1525,
                MESSAGE_TEXT = 'EndTime is
                equal to or after StartTime';
  END IF;
  IF timeoverlapWithTable(NEW.TimeSlotID,
NEW.DayCode, NEW.StartTime, NEW.EndTime)
  THEN SIGNAL SQLSTATE 'HY000'
                SET MYSQL_ERRNO = 1525,
                MESSAGE_TEXT =
                'time interval overlaps with
                existing timeinterval for the
                same TimeSlotID';
  END IF;
END // #END BEGIN
DELIMITER ;

INSERT TimeSlot VALUES ('A', 'R', '08:00:00',
'08:50:00'); #ok
INSERT TimeSlot VALUES ('A', 'T', '08:50:00',
'08:00:00'); # should give error message 'EndTime is
equal to or after StartTime'
INSERT TimeSlot VALUES ('A', 'M', '08:50:00',
'08:00:00'); # should give error message 'EndTime is
equal to or after StartTime'
INSERT TimeSlot VALUES ('A', 'M', '08:10:00',
'08:40:00'); #should give error message 'time
interval overlaps with existing timeinterval for the
same TimeSlotID'
```

5.2.4 Create an Event of European Roulette

Design a Gambling Machine which rolls a ball on a European Roulette every 10 seconds and stores the Lucky number.

Create a table called BallRolls with attributes RollNo and LuckyNo.

Create an event RollBall that executes every 10 seconds and inserts RollNo (automatically counting from 1) and LuckyNo (i.e. random number between 0 and 36) into the table BallRolls.

Solution:

```
SET GLOBAL event_scheduler = 1;

CREATE TABLE BallRolls (
RollNo INTEGER AUTO_INCREMENT PRIMARY KEY,
LuckyNo INTEGER);

CREATE EVENT RollBall
ON SCHEDULE EVERY 10 SECOND
DO
INSERT BallRolls (LuckyNo) VALUES
(FLOOR(37*RAND()));

SELECT * FROM BallRolls;
```

RollNo	LuckyNo
1	12
2	28
3	2
4	26
NULL	NULL

4. Advanced SQL