

Term Project, COP4710 (UCF)

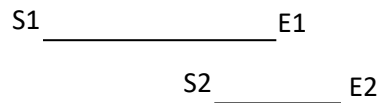
Enforcing no-overlapping-events Constraint

```
CREATE TABLE Locations(LocID          CHAR(10),
                        Name          CHAR(20),
                        Desc          CHAR(30),
                        Longitude     REAL,
                        Latitude      REAL,
                        PRIMARY KEY   (LocID))
```

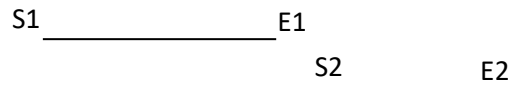
Approach 1: determining if event times are overlapped. The conditions can be verified in the app to reject overlapping events before its insertion to the Events relation, or checked in a CHECK or TRIGGER (example given below) to enforce the constraint.

- a. Time overlap: *one event starts before the other ends*, i.e.,

$$(E1 - S2) > 0 \text{ AND } (E2 - S1) > 0$$



- b. No time overlap: *one event ends by the time the other starts*, i.e.,



$$(E1 - S2) \leq 0 \text{ OR } (E2 - S1) \leq 0$$

```
CREATE TABLE Events (
    EID          Int,
    LocID        CHAR(10),
    Date         time,
    Start        time,
    End          time,
    Description   text,
    PRIMARY KEY  (EID),
    FOREIGN KEY  LocID REFERENCES Locations,
    CHECK (NOT EXIST ( SELECT *
                        FROM Events E
                        WHERE (E.LocID = LocID) AND (E.Date = Date) AND
                              ((End - E.Start) > 0) AND ((E.End - Start) > 0) )) )
```

Approach 2: All events are one hour long and start at the top of the hours.

```
CREATE TABLE Events (
    EventID      CHAR(10),
    LocID        CHAR(10),
    Event_time    time,
```

Description text,
 PRIMARY KEY (EventID),
 UNIQUE (LocID, Event_time),
 FOREIGN KEY LocID REFERENCES Locations)

Constraint: 'Active' RSOs must have at least 5 members

Enforcing via a DB design, triggers:

```
/*New Student joining
CREATE TRIGGER RSOStatusUpdateA
    AFTER INSERT ON Students_RSOs /* Event
    REFERENCING NEW AS NewMember
    WHEN ((SELECT COUNT(*)
            FROM Students_RSOs M
            WHERE M.RSO_ID = NewMember.RSO_ID) > 4)
FOR EACH ROW /* Row-level trigger
    UPDATE RSOs /* Action
        SET Status = 'active'
        WHERE RSO_ID = NewMember.RSO_ID
```

```
/*Student Leaving
CREATE TRIGGER RSOStatusUpdateP
    AFTER INSERT ON Students_RSOs /* Event
    REFERENCING OLD AS ExMember
    WHEN ((SELECT COUNT(*)
            FROM Students_RSOs M
            WHERE M.RSO_ID = ExMember.RSO_ID) < 5)
FOR EACH ROW /* Row-level trigger
    UPDATE RSOs /* Action
        SET Status = 'inactive'
        WHERE RSO_ID = ExMember.RSO_ID
```

MySQL: /*New Student joining

```
DELIMITER $$

CREATE TRIGGER RSOStatusUpdateA
  AFTER INSERT ON Students_RSOS /* Event
FOR EACH ROW BEGIN
  IF ((SELECT COUNT(*) FROM Students_RSOS M WHERE M.RSO_ID = NEW.RSO_ID) > 4)
    THEN
      UPDATE RSOS /* Action
      SET Status = 'active'
      WHERE RSO_ID = NEW.RSO_ID
    END IF;
END$$
DELIMITER ;
```

```
-----

DELIMITER //
CREATE TRIGGER test1 AFTER INSERT ON users
FOR EACH ROW
BEGIN
  DECLARE m_cnt integer;
  SET @m_cnt = (SELECT COUNT(*) FROM users M WHERE M.sid = NEW.sid);
  IF (SELECT COUNT(*) FROM users M WHERE M.sid = NEW.sid) > 4 THEN
    UPDATE users
    SET phone = 123
    WHERE sid = NEW.sid;
  END IF;
END//
```

```
DELIMITER //
CREATE TRIGGER test1 AFTER INSERT ON users
FOR EACH ROW
BEGIN
  IF (SELECT COUNT(*) FROM users M WHERE M.sid = NEW.sid) > 4 THEN
    UPDATE users
    SET phone = 123
    WHERE sid = NEW.sid;
  END IF;
END//
```