CS-A1153 - Databases, Summer 2020
Project, part 2 (SQL)

Alberto Nieto Sandino : alberto.nietosandino@aalto.fi
Peiyi Zhao : peiyi.zhao@aalto.fi

August 28, 2020

Contents

1 UML diagram **1**

2 Relation schema converted from the UML diagram **1**

3 Explanations **2**

4 Normal form and functional dependencies **4**

5 Relation schema in SQL **5**

6 Insert data into the database **8**

7 Indexes and views **9**

7.1 Indexes 9

7.2 Views 9

8 Use cases and SQL-queries **10**

1 UML diagram

The following is the UML diagram (Figure 1) that we develop for a database given the requirements of the project. Further explanations regarding the diagram and its components are given in the next section.

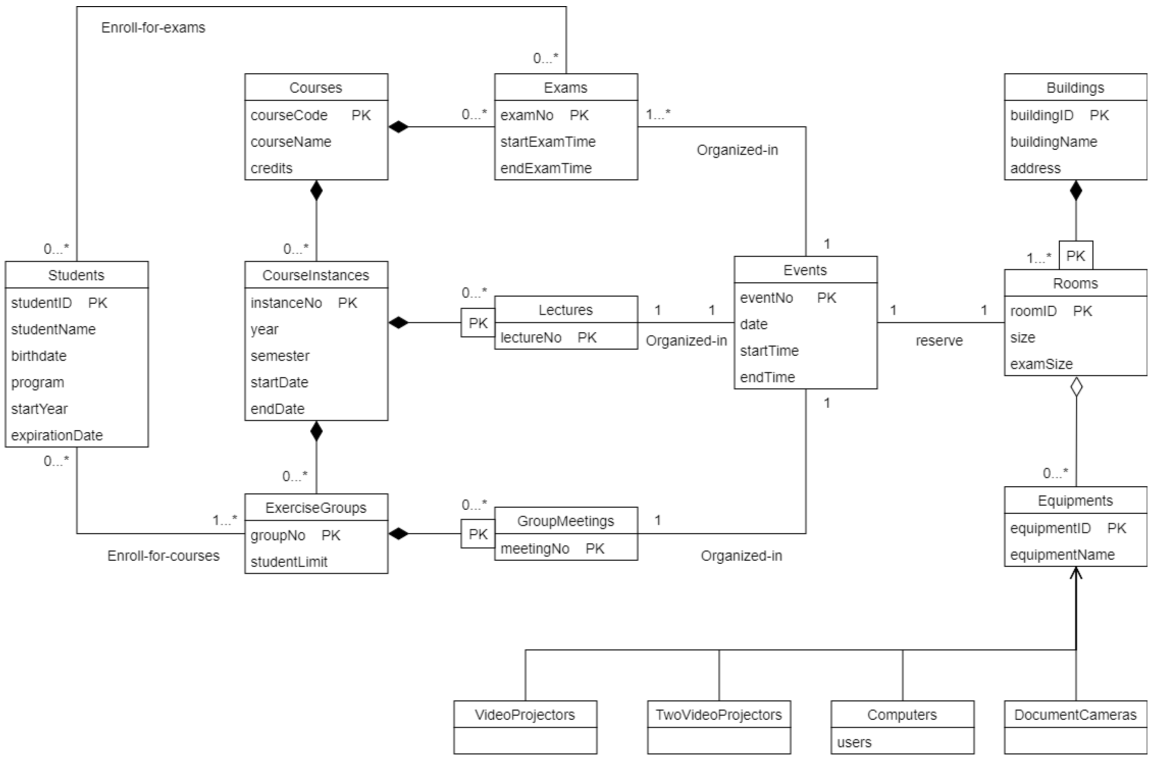

Figure 1: UML diagram for database

2 Relation schema converted from the UML diagram

Below is the relation schema for the UML diagram of the database.

- Courses (<u>courseCode</u>, courseName, credits)
- CourseInstances (<u>instanceNo</u>, year, semester, startDate, endDate, courseCode)
- ExerciseGroups (<u>groupNo</u>, studentLimit, instanceNo)

- Lectures (<u>instanceNo</u>, <u>lectureNo</u>, eventNo)
- GroupMeetings (<u>groupNo</u>, <u>meetingNo</u>, eventNo)
- Exams (<u>examNo</u>, startExamTime, endExamTime, courseCode, eventNo)

- Students (studentID, studentName, birthdate, program, startYear, expirationDate)
- Enroll-for-exams (studentID, examNo)
- Enroll-for-courses (studentID, groupNo)

- Buildings (buildingID, buildingName, address)
- Rooms (buildingID, roomID, size, examSize)

- Events (eventNo, date, startTime, endTime)
- Reserve (eventNo, buildingID, roomID)

- Equipments (equipmentID, equipmentName, buildingID, roomID)
- VideoProjectors (equipmentID)
- TwoVideoProjectors (equipmentID)
- Computers (equipmentID, users)
- DocumentCameras (equipmentID)

3 Explanations

Courses is a class including the common information for one course like code, name, and credits. CourseInstances has the basic information about each course implementation. It is part of the Courses, and it could not exist without Courses, so the relationship between them is composition. So does ExerciseGroup.

Exams belongs to Courses. Lectures belongs to CourseInstances. GroupMeetings belongs to ExerciseGroups. All the relationship between these three pairs is many-one relationship.

Events is a class to record the time information when some activity happens in a certain place at a certain time. So, one lecture or one GroupMeeting could be an event. But several exams may share an event because they could take place in one room at the same time.

One event could reserve one room. So, there is a one-one association between Events and Rooms.

Students is a class which stores the information of students. It has two associations. One is with Exams. The other is with ExerciseGroups. These two associations could be used to register the course or exam for students.

Rooms is a part of Buildings and could not exist without Buildings. So, the relationship between them is composition. Equipments is also a part of Rooms, but it can exist without rooms. So, the relationship between them is aggregation.

VideoProjectors, TwoVideoProjectors and so on are one type of Equipments. So, they are the subclass of Equipments. And other type of information could be stored in the superclass Equipments.

The aforementioned UML diagram was designed so as to support a series of queries that were described in the project requirements. Following is an explanation of how each of these queries is supported by our designed database.

The database supports storing information about courses, course instances, students, lectures, exercises groups and their times, exams and enrollments. Courses, course instances, students, lectures, exercise groups and exams have their own classes where information about them is stored. Information about the meeting times for exercise groups is stored in the GroupMeetings class. Enrollments, both for exams and for courses, are represented as associations:

- Enrollment for an exam is an association between the Students and Exams class.
- Enrollment for a courses is an association between the Students and ExerciseGroups.

The database supports storing information about buildings, rooms and their reservations. Buildings and rooms store their information in the Buildings classes and Rooms classes separately. The class Events contains the date and time information of every activity including exams, lectures and group meetings. Each event has a unique EventNo. If several exams take place at the same time in one room, they will be regarded as one event and get the same eventNo. By joining the class Events and the class Reserve, we could find a certain event reserves a certain room in a certain time. includes reservations for exams, reservations for lectures and reservation for group meetings.

Courses are composed of the classes Exams and CourseInstances. A CourseInstances class is a given course organized in a certain semester. Each course instance has a unique instance number so that it could be recognized even if there are several course instances for the same course in one semester. Each of these course instances is composed of ExerciseGroups and Lectures. ExerciseGroups are composed of GroupMeetings which is associated with Events. Lectures is associated with events. These two latter associations make it possible to keep track of when a certain lecture or group meeting from a given course instance of a course took place or will take place at a specific time and date.

If we look at a certain time interval in events, we can have multiple lectures of different course instances and multiple group meetings of different exercise groups at the same time as long as their roomID is different. This is shown as the association reserve between Events and Rooms. Thus, for a given time interval, we can look up which courses have a lecture reservation in the Events class.

Through the association between Exams and Events, we can see which exams (of a given Course) take place at a certain time interval. In the Exams class, it is specified the officialduration of the exam (start and end), while in the reservation in Events, the time is longer since there is a need for practical arrangements both before and after the exam. In order to see when a certain course will or has been arranged, one can look up the course instance class. Since course is composed of course instance, it is possible to look up a certain course code (foreign key in courseInstance which references the primary key in the class Courses) and see the year and semester when course instances of that course may be taking place.

Course instances are composed of Lectures and Exercise groups. Thus, it is straightforward to look up which lectures are associated with which course instances.

Course instances are also composed of Exercised groups, thus we can search which groups belong to which course instance. Many exercise groups are possible for one course instance.

Once the exercise group is fixed, we can get the eventNo and find the time when this group meets by looking at Group meetings and its association with events. Through checking the eventNo in the class Reserve, we can find which room (in a certain building) the group meeting is taking place in.

To select a room with a certain number of seats and free at a certain time. We can first filter the Rooms class for rooms with a certain size (i.e. the number of seats). Then, through its association with Events, we can look at which times these selected rooms are not booked.

If we have a certain roomID with buildingID and a start and end time of a reservation from Events, we can look up if that corresponds to an exam, a lecture or group meeting (i.e. an exercise group) by joining these latter 3 classes with that time slot and seeing if there are any tuples in common. That is tuples where that would be associated with that time slot and room.

Enrolling of students take place with two associations:

- Association Enroll-for-courses which enrols the Student class into the ExerciseGroup, thus automatically signing up for a certain course.
- Association Enroll-for-exams which enrols the Student class into the Exam for a certain Course.

To list all students enrolled in a course instance, we can list all of the students who are enrolled in exerciseGroups of that certain course instance. To list all students enrolled in an exercise group, we can list the students present in that certain Exercise group (with unique groupNo). To list all students enrolled in an exam, we can use the Enroll-for-exams association where each student is associated with a unique exam.

To see if a certain ExerciseGroup is full, we can inspect its studentLimit attribute and count the tuples grouped by the groupNo. By comparing these two numbers, we could see whether the limit has been reached or if more students can be enrolled into that ExerciseGroup.

4 Normal form and functional dependencies

The following are the functional dependencies from our database.

- courseCode → courseName credits
- instanceNo → year semester startDate endDate courseCode
- groupNo → studentLimit instanceNo
- instanceNo lectureNo → eventNo
- groupNo meetingNo → eventNo
- examNo → startExamTime endExamTime courseCode eventNo
- studentID → studentName birthdate program startYear expirationDate
- buildingID → buildingName address
- buildingID roomID → size examSize

- eventNo → date startTime endTime
- equipmentID → equipmentName buildingID roomID

Because all the left side is a super key of the relation, it is in BCNF.

5 Relation schema in SQL

```sql
CREATE TABLE Courses (
    courseCode TEXT PRIMARY KEY,
    courseName TEXT NOT NULL,
    credits    REAL CHECK (credits > 0)
);

CREATE TABLE CourseInstances (
    instanceNo TEXT    PRIMARY KEY,
    year       INTEGER CHECK (year > 2000),
    semester   INTEGER CHECK (semester IN (1, 2) ),
    startDate  TEXT,
    endDate    TEXT,
    courseCode TEXT    REFERENCES Courses (courseCode)
);

CREATE TABLE ExerciseGroups (
    groupNo      TEXT    PRIMARY KEY,
    studentLimit INTEGER CHECK (studentLimit > 0),
    instanceNo   TEXT    REFERENCES CourseInstances (instanceNo)
);

CREATE TABLE Events (
    eventNo   TEXT PRIMARY KEY,
    date      TEXT,
    startTime TEXT,
    endTime   TEXT
);

CREATE TABLE Lectures (
    instanceNo TEXT,
    lectureNo  TEXT,
    eventNo    TEXT UNIQUE,
    PRIMARY KEY (
        instanceNo,
        lectureNo
    ),
    FOREIGN KEY (
        instanceNo
    )
    REFERENCES CourseInstances (instanceNo),
    FOREIGN KEY (
        eventNo
    )
    REFERENCES Events (eventNo)
);

CREATE TABLE GroupMeetings (
    groupNo   TEXT,
    meetingNo TEXT,
    eventNo   TEXT UNIQUE,
    PRIMARY KEY (
        groupNo,
        meetingNo
    ),
    FOREIGN KEY (
        groupNo
    )
    REFERENCES ExerciseGroups (groupNo),
    FOREIGN KEY (
        eventNo
    )
    REFERENCES Events (eventNo)
);

CREATE TABLE Exams (
    examNo        TEXT PRIMARY KEY,
    startExamTime TEXT,
    endExamTime   TEXT,
    courseCode    TEXT NOT NULL,
    eventNo       TEXT NOT NULL,
    FOREIGN KEY (
        courseCode
    )
    REFERENCES Courses (courseCode),
    FOREIGN KEY (
        eventNo
    )
    REFERENCES Events (eventNo)
);

CREATE TABLE Students (
    studentID      TEXT    PRIMARY KEY,
    studentName    TEXT,
    birthdate      TEXT,
    program        TEXT,
    startYear      INTEGER,
    expirationDate TEXT
);

CREATE TABLE EnrollForExams (
    studentID TEXT,
    examNo    TEXT,
    PRIMARY KEY (
        studentID,
        examNo
    ),
    FOREIGN KEY (
        studentID
    )
    REFERENCES Students (studentID),
    FOREIGN KEY (
        examNo
    )
    REFERENCES Exams (examNo)
);

CREATE TABLE EnrollForCourses (
    studentID TEXT,
    groupNo   TEXT,
    PRIMARY KEY (
        studentID,
        groupNo
    ),
    FOREIGN KEY (
        studentID
    )
    REFERENCES Students (studentID),
    FOREIGN KEY (
        groupNo
    )
    REFERENCES ExerciseGroups (groupNo)
);

CREATE TABLE Buildings (
    buildingID   TEXT PRIMARY KEY,
    buildingName TEXT,
    address      TEXT UNIQUE
);

CREATE TABLE Rooms (
    buildingID TEXT,
    roomID     TEXT,
    size       INTEGER,
    examSize   INTEGERR,
    PRIMARY KEY (
        buildingID,
        roomID
    ),
    FOREIGN KEY (
        buildingID
    )
    REFERENCES Buildings (buildingID),
    CHECK (size >= examSize)
);

CREATE TABLE Equipments (
    equipmentID   TEXT PRIMARY KEY,
    equipmentName TEXT NOT NULL,
    buildingID    TEXT,
    roomID        TEXT,
    FOREIGN KEY (
        buildingID,
        roomID
    )
    REFERENCES Rooms (buildingID,
    roomID)
);


CREATE TABLE VideoProjectors (
    equipmentID TEXT PRIMARY KEY,
    FOREIGN KEY (
        equipmentID
    )
    REFERENCES Equipments (equipmentID)
);

CREATE TABLE TwoVideoProjectors (
    equipmentID TEXT PRIMARY KEY,
    FOREIGN KEY (
        equipmentID
    )
    REFERENCES Equipments (equipmentID)
);

CREATE TABLE Computers (
    equipmentID TEXT PRIMARY KEY,
    users       TEXT NOT NULL,
    FOREIGN KEY (
        equipmentID
    )
    REFERENCES Equipments (equipmentID)
);

CREATE TABLE DocumentCameras (
    equipmentID TEXT PRIMARY KEY,
    FOREIGN KEY (
        equipmentID
    )
    REFERENCES Equipments (equipmentID)
);

CREATE TABLE Reserve (
    eventNo    TEXT UNIQUE,
    buildingID TEXT NOT NULL,
    roomID     TEXT NOT NULL,
    FOREIGN KEY (
        eventNo
    )
    REFERENCES Events (eventNo),
    FOREIGN KEY (
        buildingID,
        roomID
    )
    REFERENCES Rooms (buildingID,
    roomID)
);
```

6 Insert data into the database

```sql
INSERT INTO Courses
VALUES ('CS-A1150', 'Databases', 5);

INSERT INTO Courses
VALUES ('CS-E4800', 'Artificial Intelligence', 5);

INSERT INTO Courses
VALUES ('CS-E4610', 'Modern Database Systems', 5);

INSERT INTO Courses
VALUES ('CS-C3100', 'Computer Graphics', 5);

INSERT INTO CourseInstances
VALUES ('CS-A1150-2020-1', 2020, 1, '2020-09-03', '2020-11-20', 'CS-A1150');

INSERT INTO CourseInstances
VALUES ('CS-E4800-2020-1', 2020, 1, '2020-09-01', '2020-12-27', 'CS-E4800');

INSERT INTO CourseInstances
VALUES ('CS-E4610-2020-1', 2020, 1, '2021-09-07', '2021-11-23', 'CS-E4610');

INSERT INTO CourseInstances
VALUES ('CS-E4610-2020-2', 2020, 2, '2021-01-05', '2021-04-05', 'CS-E4610');

INSERT INTO ExerciseGroups
VALUES ('CS-A1150-group1', 10, 'CS-A1150-2020-1');

INSERT INTO Events
VALUES ('20200903L1', '2020-09-03', '09:00', '12:00');

INSERT INTO Events
VALUES ('20200904M1', '2020-09-04', '10:00', '11:00');

INSERT INTO Events
VALUES ('20201125E1', '2020-11-25', '08:30', '09:30');

INSERT INTO Lectures
VALUES ('CS-A1150-2020-1', 'lecture1', '20200903L1');

```

```sql
INSERT INTO Lectures(instanceNo, lectureNo)
VALUES ('CS-A1150-2020-1', 'lecture2');

INSERT INTO GroupMeetings
VALUES ('CS-A1150-group1', 'meeting1', '20200904M1');

INSERT INTO GroupMeetings(groupNo, meetingNo)
VALUES ('CS-A1150-group1', 'meeting2');

INSERT INTO Exams
VALUES ('CS-A1150-exam1', '08:40', '09:30', 'CS-A1150', '20201125E1');

INSERT INTO Students
VALUES ('112233', 'Teemu Teekkari', '1990-05-11', 'TIK', '2017', '2021-08-30');

INSERT INTO Students
VALUES ('123456', 'Tiina Teekkari', '1992-09-23', 'TUO', '2019', '2023-08-30');

INSERT INTO Students
VALUES ('224411', 'Maija Virtanen', '1991-12-05', 'AUT', '2018', '2022-08-30');

INSERT INTO EnrollForExams
VALUES ('112233', 'CS-A1150-exam1');

INSERT INTO EnrollForExams
VALUES ('224411', 'CS-A1150-exam1');

INSERT INTO EnrollForCourses
VALUES ('112233', 'CS-A1150-group1');

INSERT INTO Buildings
VALUES ('building1', 'Computer Science Building', 'Tietotekniikantalo, Konemiehentie
     2, 02150 Espoo');

INSERT INTO Buildings
VALUES ('building2', 'Engineering Physics Building', 'Tietotekniikantalo,
    Konemiehentie 4, 02150 Espoo');

INSERT INTO Rooms
VALUES ('building1', 'room101', 30, 20);

INSERT INTO Rooms
VALUES ('building1', 'C206', 45, 30);

INSERT INTO Rooms
VALUES ('building2', 'C310', 50, 30);

INSERT INTO Equipments
VALUES ('equipment1', 'MacComputer', 'building1', 'room101');

INSERT INTO Equipments
VALUES ('equipment2', 'SONY VideoProject', 'building1', 'room101');

INSERT INTO VideoProjectors
VALUES ('equipment2');

INSERT INTO Computers
VALUES ('equipment1', 'teachers');

INSERT INTO Reserve
VALUES ('20200903L1', 'building1', 'room101');

INSERT INTO Reserve
VALUES ('20201125E1', 'building1', 'C206');
```

7 Indexes and views

7.1 Indexes

In order to create reasonable indexing, we need to know or have an educated guess of what the typical queries to the database may be.

7.2 Views

Based on the typical queries that we may expect, we can create views that will help simplify these queries and reduce their complexity.

```sql
CREATE VIEW LectureTimePlace AS
    SELECT instanceNo,
           lectureNo,
           date,
           startTime,
           endTime,
           buildingID,
           roomID
      FROM (
               SELECT instanceNo,
                      lectureNo,
                      Lectures.eventNo,
                      date,
                      startTime,
                      endTime
                 FROM Lectures
                      LEFT JOIN
                      Events ON Lectures.eventNo = Events.eventNo
           )
           AS LectureTime
           LEFT JOIN
           Reserve ON LectureTime.eventNo = Reserve.eventNo;

CREATE VIEW MeetingTimePlace AS
    SELECT groupNo,
           meetingNo,
           date,
           startTime,
           endTime,
           buildingID,
           roomID
      FROM (
               SELECT groupNo,
                      meetingNo,
                      GroupMeetings.eventNo,
                      date,
                      startTime,
                      endTime
                 FROM GroupMeetings
                      LEFT JOIN
                      Events ON GroupMeetings.eventNo = Events.eventNo
           )
           AS MeetingTime
           LEFT JOIN
           Reserve ON MeetingTime.eventNo = Reserve.eventNo;

CREATE VIEW ExamTimePlace AS
    SELECT courseCode,
           examNo,
            startExamTime,
            endExamTime,
            date,
            startTime,
            endTime,
            buildingID,
            roomID
      FROM (
                SELECT courseCode,
                       examNo,
                       Exams.eventNo,
                       startExamTime,
                       endExamTime,
                       date,
                       startTime,
                       endTime
                  FROM Exams
                       LEFT JOIN
                       Events ON Exams.eventNo = Events.eventNo
            )
            AS ExamTime
            LEFT JOIN
            Reserve ON ExamTime.eventNo = Reserve.eventNo;
```

8 Use cases and SQL-queries

```
/* Query all distinct courses (course code and name) taken place during the first
    semester of 2020 */
SELECT DISTINCT courseCode, courseName
  FROM CourseInstances AS CI,
       Courses AS C
 WHERE CI.courseCode = C.courseCode AND
       year = 2020 AND
       semester = 1;

/* Query all courses that contain the word 'Database' in their name */
SELECT courseName
  FROM Courses
 WHERE courseName LIKE '%Database%';

/* Find how many courses take place per semester and year*/
SELECT year,
       semester,
       COUNT(instanceNo) AS total
  FROM CourseInstances
 GROUP BY year,
          semester;

/* Find how many instances of each course are scheduled.
Those courses with no instances scheduled must also be listed.
Order the courses based on the number of instances in a descending fashion. */
SELECT courseName as Course,
       COUNT(instanceNo) AS Instances
  FROM Courses
       LEFT OUTER JOIN
       CourseInstances ON CourseInstances.courseCode = Courses.courseCode
 GROUP BY courseName
 ORDER BY instances DESC;

/* Query the exact number and the limit number of students in one group */
SELECT EnrollForCourses.groupNo,
       COUNT(studentID),
       studentLimit
```

```sql
  FROM EnrollForCourses,
       ExerciseGroups
 WHERE EnrollForCourses.groupNo = ExerciseGroups.groupNo
 GROUP BY EnrollForCourses.groupNo;

/* Query the exact number of students in one exam */
SELECT examNo,
       COUNT(studentID)
  FROM EnrollForExams
 GROUP BY examNo;

/* Query the exact number of students in a study group */
SELECT groupNo,
       COUNT(studentID)
  FROM EnrollForCourses
 GROUP BY groupNo;


/* Find the course codes and names of all the course that are organized during the
    second semester,
 but not in the second semester. */
SELECT courseCode,
       courseName
  FROM Courses
 WHERE courseCode IN (
           SELECT courseCode
             FROM courseInstances
            WHERE semester = 1
       )
AND
       courseCode NOT IN (
           SELECT courseCode
             FROM courseInstances
            WHERE semester = 2
       );

/* Query the number of computers in one room which could used by students to have a
    lecture or exam */
SELECT buildingID,
       roomID,
       COUNT(Computers.equipmentID)
  FROM Computers,
       Equipments
 WHERE users = 'students' AND
       Computers.equipmentID = Equipments.equipmentID
 GROUP BY buildingID,
          roomID;

/* Find a room which has at least 20 seats and which is free for reservation at a
    certain time (2020-09-03 at 9:00 to 10:00).
List the room ID and the address of the room where the building is located.
*/

/* In order to select those rooms without a reservation, 3 different conditions can
    occur:
    - The room has no reservations for that day
    - The room has a reservation in that day, but not within the specified times
    - The room has no reservations at all (it is not found in the Reserve relation).
   Besides this, we just have to make sure that the room has space for more than 20
    people.
In this first query, one of the rooms has an event in that day and within the given
    timeframe.
The others are available.
*/

SELECT roomID,
       buildingName,
       address
  FROM Rooms,
       Buildings
 WHERE Rooms.buildingID = Buildings.buildingID AND
       size >= 20
EXCEPT
SELECT Reserve.roomID,
       buildingName,
       address
  FROM Reserve,
       Events,
       Rooms,
        Buildings
 WHERE Rooms.buildingID = Buildings.buildingID AND
       Reserve.eventNo = Events.eventNo AND
       Reserve.roomID = Rooms.roomID AND
       date = '2020-09-03' AND
       ( (TIME('09:00') >= startTime AND
          TIME('09:00') < endTime) OR
         (TIME('10:00') > startTime AND
          TIME('10:00') <= endTime) OR
         (TIME('09:00') < startTime AND
          TIME('10:00') > endTime) );

/* In this second query, there are no reservations for any Event on 2020-09-05, so
    all the rooms that
have been inserted in the database are present. No matter if they have any scheduled
     event at some point or not.*/
SELECT roomID,
       buildingName,
       address
  FROM Rooms,
       Buildings
 WHERE Rooms.buildingID = Buildings.buildingID AND
       size >= 20
EXCEPT
SELECT Reserve.roomID,
       buildingName,
       address
  FROM Reserve,
       Events,
       Rooms,
        Buildings
 WHERE Rooms.buildingID = Buildings.buildingID AND
       Reserve.eventNo = Events.eventNo AND
       Reserve.roomID = Rooms.roomID AND
       date = '2020-09-05' AND
       ( (TIME('09:00') >= startTime AND
          TIME('09:00') < endTime) OR
         (TIME('10:00') > startTime AND
          TIME('10:00') <= endTime) OR
         (TIME('09:00') < startTime AND
          TIME('10:00') > endTime) );

/* Find out for which purpose a certain room is reserved at a certain time.
For example, is there some event in room C206 from the Computer Science Building
on 2020-11-25 between 08:30 and 10:30?*/

/* The query should return empty if no events are scheduled, or it will return the
    EventNo
that identifies which kind of event is scheduled and its schedule.
If there is an event that overlaps the time that is given, it will be shown as well.
No matter how small the time overlap is.*/
SELECT DISTINCT eventNo,
                startTime,
                endTime
  FROM Events,
       Reserve,
       Rooms,
       Buildings
 WHERE Events.eventNo = Reserve.eventNo AND
       Reserve.roomID = Rooms.roomID AND
       Rooms.buildingID = Buildings.buildingID AND
       Rooms.roomID = 'C206' AND
       Buildings.buildingName = 'Computer Science Building' AND
       date = '2020-11-25' AND
       (
           (TIME(startTime) <= TIME('08:30') AND TIME(endTime) > TIME('08:30') )
           OR
           (TIME(startTime) > TIME('08:30') AND TIME(startTime) < TIME('10:30') )
        );

/* Now, we will check another room (e.g. room101) of the same building for the same
    time schedule.
Since the previous one had an Event (Exam because of the code).
*/
SELECT DISTINCT eventNo,
                startTime,
                endTime
  FROM Events,
       Reserve,
       Rooms,
       Buildings
 WHERE Events.eventNo = Reserve.eventNo AND
       Reserve.roomID = Rooms.roomID AND
       Rooms.buildingID = Buildings.buildingID AND
       Rooms.roomID = 'room101' AND
       Buildings.buildingName = 'Computer Science Building' AND
       date = '2020-11-25' AND
       (
           (TIME(startTime) <= TIME('08:30') AND TIME(endTime) > TIME('08:30') )
           OR
           (TIME(startTime) > TIME('08:30') AND TIME(startTime) < TIME('10:30') )
        );

/* As we can see, no tuples are returned and, thus, we can book this room for this
    time schedule freely.*/


/* List all exams that are scheduled, the course name and where and when it is
    scheduled to take place.
Now, we can use the view that we created earlier 'ExamTimePlace' to make the query
    much simpler.*/
SELECT examNo,
       courseName,
       date,
       startExamTime AS start,
       endExamTime AS end,
       roomId,
       buildingName
  FROM ExamTimePlace,
       Courses,
       Buildings
 WHERE ExamTimePlace.courseCode = Courses.courseCode AND
       ExamTimePlace.buildingID = Buildings.buildingID;


/* List all students who have enrolled for a exam in the Databases course.
List the student name, the program they are in, which exam they are enrolled in (
    multiple exams
per year in a course are possible) and the date of the exam.*/
```

```sql
SELECT studentName,
       program,
       examNo,
       date
  FROM EnrollForExams,
       Exams,
       Courses,
       Students,
       Events
 WHERE EnrollForExams.studentID = Students.studentID AND
       EnrollForExams.examNo = Exams.examNo AND
       Exams.courseCode = Courses.courseCode AND
       Exams.eventNo = Events.eventNo AND
       Courses.courseName = 'Databases';

/* For a particular course instance (e.g. 'CS-A1150-2020-1'), list the exercise
    groups that are not full yet.
List how many students are enrolled in those groups that are not full.*/
SELECT EnrollForCourses.groupNo,
       COUNT(studentID) AS totalStudents
  FROM EnrollForCourses,
       ExerciseGroups
 WHERE EnrollForCourses.groupNo = ExerciseGroups.groupNo
 GROUP BY EnrollForCourses.groupNo
HAVING COUNT(studentID) < ExerciseGroups.studentLimit;

/* Find out, when a certain course has been arranged or when it will be arranged */
SELECT courseName,
       year,
       semester,
       startDate,
       endDate
  FROM CourseInstances,
       Courses
 WHERE CourseInstances.courseCode = Courses.courseCode
 ORDER BY startDate, endDate;
/* As we can see the course Modern Database Systems is organized twice in 2020.
The course instances that are listed are ordered by their start date (starting the
    earliest).
If they would have the same start date, the order would be determined by the end
    date (earliest first). */
```