

## 1 Creating Tables

Consider the following relational schema, which obviously does not describe the standard situation at Aalborg University.

We assume that tutors are responsible for one or multiple study groups, students individually (not per group) hand in solutions for exercise sheets and receive individual grades in terms of the number of achieved points per sheet. Some of the tutors are more experienced (senior) than others.

student: {[ sid: int, firstname: string, lastname: string, semester: int, birthdate: date ]}

tutor: {[ tid: int, firstname: string, lastname: string, issenior: boolean ]}

studygroup: {[ gid: int, tid → tutor, weekday: string, room: string, starttime: time ]}

exercisesheet: {[ eid: int, maxpoints: int ]}

handsin: {[ sid → student, eid → exercisesheet, achievedpoints: int ]}

member: {[ sid → student, gid → studygroup ]}

Please formulate appropriate SQL statements to create these 6 tables using:

- Sequence number generators whose values automatically increase when data is inserted
- Appropriate data types
- Primary and foreign keys

Hint:

*There might be more than one possible data type for some of the attributes.*

### Solution.

*Sequence number generators*

```
CREATE SEQUENCE studentseq START 1;
CREATE SEQUENCE tutorseq START 1;
CREATE SEQUENCE studygroupseq START 1;
CREATE SEQUENCE exercisesheetseq START 1;
```

*Tables*

```
CREATE TABLE student (
  sid INT PRIMARY KEY DEFAULT nextval('studentseq'),
  firstname VARCHAR(32) NOT NULL,
  lastname VARCHAR(32) NOT NULL,
```



```
semester INTEGER,  
birthdate DATE  
);
```

```
CREATE TABLE tutor (  
tid INT PRIMARY KEY DEFAULT nextval('tutorseq'),  
firstname VARCHAR(32) NOT NULL,  
lastname VARCHAR(32) NOT NULL,  
issenior BOOLEAN  
);
```

```
CREATE TABLE studygroup (  
gid INT PRIMARY KEY DEFAULT nextval('studygroupseq'),  
tid INT,  
weekday VARCHAR(12) NOT NULL,  
room VARCHAR(12) NOT NULL,  
starttime TIME,  
FOREIGN KEY (tid) REFERENCES tutor (tid)  
);
```

```
CREATE TABLE exercisesheet (  
eid INT PRIMARY KEY DEFAULT nextval('exercisesheetseq')  
,  
maxpoints INTEGER  
);
```



```
CREATE TABLE handsin (  
  sid INT,  
  eid INT,  
  achievedpoints INTEGER,  
  PRIMARY KEY (sid,eid),  
  FOREIGN KEY (sid) REFERENCES student (sid),  
  FOREIGN KEY (eid) REFERENCES exercisesheet (eid)  
);
```

```
CREATE TABLE member (  
  sid INT,  
  gid INT,  
  PRIMARY KEY (sid,gid),  
  FOREIGN KEY (sid) REFERENCES student (sid),  
  FOREIGN KEY (gid) REFERENCES studygroup (gid)  
);
```

## 2 Querying Tables I

Translate the following queries into equivalent SQL statements that run on the tables created above.

1. Find the different last names of the students whose first name is “Helle”.

**Solution.**

```
SELECT S.lastname
FROM student S
WHERE S.firstname = 'Helle';
```

2. Find all the different last names of students that end with 'sen'.  
*Hint: you can test if attribute test ends with 'xyz' by using “test LIKE '%xyz’”*

**Solution.**

```
SELECT DISTINCT lastname FROM student WHERE
    lastname LIKE '%sen';
```

3. List the first and last names of the tutors that are senior.

**Solution.**

```
SELECT tutor.firstname, tutor.lastname
FROM tutor
WHERE tutor.issenior;
```

4. Find the first and last names of all students who have study group on Wednesday or Friday.

**Solution.**

```
SELECT S.firstname, S.lastname
FROM student S, studygroup SG, member M
WHERE S.sid = M.sid AND M.gid = SG.gid
    AND (SG.weekday = 'Wednesday'
        OR SG.weekday = 'Friday');
```

5. Output the IDs of all students and their achieved points for exercise sheet 1 in descending order by the number of achieved points.

**Solution.**

```
SELECT sid, achievedpoints  
FROM handsin  
WHERE eid = 1  
ORDER BY achievedpoints DESC;
```

### 3 Querying Tables II

Considering the tables created above, identify the missing information that should go into the boxes to make the queries compute the requested information.

1. Find all the ids of the study groups without any members.

```
SELECT SG.gid
FROM studygroup SG
WHERE NOT EXISTS(SELECT 
                  FROM 
                  WHERE );
```

**Solution.**

```
SELECT SG.gid
FROM studygroup SG
WHERE NOT EXISTS(SELECT M.sid
                    FROM member M
                    WHERE SG.gid = M.gid);
```

2. List the first and last names of the students that obtained the highest number of points for exercise sheet 1.

```
SELECT firstname, lastname
FROM  NATURAL JOIN handsin
WHERE eid = 1 AND
      achievedPoints >=  (SELECT 
                                           FROM handsin
                                           WHERE  = 1);
```

**Solution.**

```
SELECT firstname, lastname
FROM student NATURAL JOIN handsin
WHERE eid = 1 AND
      achievedPoints >= ALL (SELECT achievedPoints
                             FROM handsin
                             WHERE eid = 1);
```

3. Find the first and last names of all students that have “Helle” as tutor.

```
SELECT S.firstname, S.lastname
FROM student S, [box 1], tutor T, [box 2]
WHERE T.firstname = 'Helle' AND T.tid = [box 3]
      AND S.sid= [box 4] AND [box 5];
```

**Solution.**

```
SELECT S.firstname, S.lastname
FROM student S, studygroup SG, tutor T, member M
WHERE T.firstname = 'Helle' AND T.tid = SG.tid
      AND S.sid= M.sid AND SG.gid = M.gid;
```

4. Find the first and last names of all students who have achieved at least 7 points for all the exercise sheets they have handed in.

```
SELECT S.firstname, S.lastname
FROM student S
WHERE [box 1] (SELECT 42
               FROM [box 2]
               WHERE S.sid = [box 3]
                  AND [box 4] < 7);
```

**Solution.**

```
SELECT S.firstname, S.lastname
FROM student S
WHERE NOT EXISTS (SELECT 42
                  FROM handsin H
                  WHERE S.sid = H.sid
                     AND H.achievedPoints < 7);
```

5. Find the IDs of all students born before 01.03.1998 that have the same first name as one of the tutors.

```
SELECT sid
FROM student
WHERE [box 1] < '1998-03-01'
      AND firstname [box 2] (SELECT [box 3] FROM [box 4]);
```



**Solution.**

```
SELECT sid
FROM student
WHERE birthdate < '1998-03-01'
AND firstname IN ( SELECT firstname FROM tutor );
```



## 4 Manipulating Tables

1. Populate the above created tables by inserting at least one valid tuple per table.

### Solution.

*Because of the foreign key constraints you need to pay attention to the order in which the statements are issued.*

```
INSERT INTO student (firstname, lastname, semester,  
    birthdate)  
VALUES ('John', 'Doe', 4, '1986-01-01');
```

```
INSERT INTO tutor (firstname, lastname, issenior)  
VALUES ('Jane', 'Doe', false);
```

```
INSERT INTO studygroup (tid, weekday, room,  
    starttime)  
VALUES (1, 'Tuesday', '1.2.3', '08:15:00');
```

```
INSERT INTO exercisesheet (maxpoints)  
VALUES (18);
```

```
INSERT INTO handsin (sid, eid, achievedpoints)  
VALUES (1, 1, 17);
```

```
INSERT INTO member (sid, gid)  
VALUES (1, 1);
```

2. Formulate an SQL statement that sets the first names of all students to 'Jens' whose last name is 'Doe'.

### Solution.

```
UPDATE student SET firstname='Jens'  
WHERE lastname='Doe';
```

3. Delete all student tuples from table student for which the first name is 'Tom'.

**Solution.**

```
DELETE FROM student  
WHERE firstname = 'Tom';
```

## 5 Using PostgreSQL

Test your solutions to the previous exercises using PostgreSQL.

Hint:

*you might want to have a look at <https://www.moodle.aau.dk/mod/page/view.php?id=1463706> (“Suggestions for Installation and usage of Postgres for the Self-Study” in Moodle) and <https://www.postgresql.org/docs/13/datatype.html> (data types supported by PostgreSQL).*

**Exploration and Experimentation:** Test what happens in PostgreSQL when you insert 2 students without specifying their “sid” value, then insert a new student specifying as “sid” the value 3, then try insert other 2 students without specifying their “sid”.