Databases Lab 01
Emily Antosch & Furkan Yildirim & Julian Moldenhauer
02.04.2025

Databases Lab 01

This is the first lab of Databases. This lab focuses on improving your skills in dealing with SQL queries. There are tasks for both DDL and DML statements. If you have questions or need any support, help each other, ask your tutor or use the forum in our moodle room.

Contents

1. Assignment 1: SQL-statements for the Student Information System

A schema essentially is a collection of tables and their relations to each other. Schemas are commonly implemented by SQL. Consider the following schema for the Student Information System:

- **STUDENT**(studentID, fistName, lastName, dob, programID(FK))
- **PROGRAM**(programID, name, requiredCPs)
- **COURSE**(courseID, name, description, creditPoints, programID(FK))
- **ATTEMPTS**(studentID(FK),courseID(FK), year, term, grade)
- **PREREQUISITE**(advancedCourseID(FK), prerequisiteCourseID (FK))

1. Write SQL-statements that create the corresponding tables. Come up with reasonable constraints and datatypes for the fields of the tables.

Solution

```SQL
CREATE TABLE Program
  (programID INT NOT NULL,
   name VARCHAR(32) NOT NULL,
   requiredCPs INT NOT NULL,
   PRIMARY KEY (programID) );
```

```SQL
CREATE TABLE Student
  (studentID INT NOT NULL,
   firstName VARCHAR(32) NOT NULL,
   lastName  VARCHAR(32) NOT NULL,
   dob            DATE NOT NULL,
   programID INT NOT NULL,
   PRIMARY KEY (studentID),
   FOREIGN KEY (programID) REFERENCES Program(programID) );
```

```SQL
CREATE TABLE Course
  (CourseID INT NOT NULL,
   name VARCHAR(32) NOT NULL,
   description VARCHAR(100) NOT NULL,
   creditPoints INT NOT NULL,
   programID INT NOT NULL,
   PRIMARY KEY (courseID),
   FOREIGN KEY (programID) REFERENCES Program(programID) );
```

```SQL
  CREATE TABLE Attempts
    (studentID INT NOT NULL,
     courseID INT NOT NULL,
     year INT NOT NULL,
     term INT NOT NULL,
     grade INT NOT NULL,
    PRIMARY KEY (studentID, courseID, year, term),
    FOREIGN KEY (studentID) REFERENCES Student(studentID),
    FOREIGN KEY (courseID) REFERENCES Course(courseID) );
```

```SQL
CREATE TABLE Prerequisite
  (advancedCourse INT NOT NULL,
   prerequisiteCourse INT NOT NULL,
  PRIMARY KEY (advancedCourse, prerequisiteCourse),
  FOREIGN KEY (advancedCourse) REFERENCES Course(courseID),
  FOREIGN KEY (prerequisiteCourse) REFERENCES Course(courseID) );
```

2. Write SQL-queries that insert example data into your created tables. Make sure that each table contains at least 2 rows of data. Here are some sample data.

programID	Name	requiredCPs
1	Information Engineering	120
2	Renewable Energies	110

Table 1: Table **PROGRAM**

studentID	firstName	lastName	dob	programID
123456	John	Wayne	11.05.1998	1
234567	Anna	Meyer	13.02.1999	1

Table 2: Table **STUDENT**

courseID	Name	Description	creditPoints	programID
4	MA1	Mathematics 1	8	1
9	MA2	Mathematics 2	8	1
13	SS1	Signals and Systems 1	6	1
15	DB	Databases	6	1

Table 3: Table **COURSE**

studentID	courseID	Year	Term	grade
123456	4	2021	1	7
234567	9	2021	2	9
234567	13	2022	1	3
234567	13	2022	2	6

Table 4: Table **ATTEMPTS**

advancedCourseID	prerequisiteCourseID
9	4
13	9
13	4

Table 5: Table **PREREQUISITE**

Solution

```SQL
INSERT INTO Program
VALUES ( 1, 'Information Engineering',120);

INSERT INTO Program
VALUES ( 2, 'Renewable Energies',110);
```

```SQL
INSERT INTO Student
VALUES ( 123456, 'John', 'Wayne', '1998-05-11',1);

INSERT INTO Student
VALUES ( 234567, 'Anna', 'Meyer', '1999-02-13',1);
```

```SQL
 INSERT INTO Course
 VALUES ( 4, 'MA1', 'Mathematics 1', 8,1);

 INSERT INTO Course
 VALUES ( 9, 'MA2', 'Mathematics 2', 8,1);

 INSERT INTO Course
 VALUES ( 13, 'SS1', 'Signals and Systems 1', 6,1);

INSERT INTO Course
VALUES ( 15, 'DB', 'Databases', 6,1);
```

```SQL
INSERT INTO Prerequisite
VALUES ( 9,4);

INSERT INTO Prerequisite
VALUES ( 13,4);

INSERT INTO Prerequisite
VALUES ( 13,9);
```

Solution

```SQL
INSERT INTO Attempts
VALUES (123456, 4, 2021, 1, 7);

INSERT INTO Attempts
VALUES (123456, 9, 2021, 2, 9);

INSERT INTO Attempts
VALUES (123456, 13, 2022, 1, 3);

INSERT INTO Attempts
VALUES (123456, 13, 2022, 2, 6);
```

3. Write a SQL-query for the created database that returns all students (first name and last name) that study the program “Information Engineering”.

Solution

```SQL
SELECT s.firstname, s.lastname
FROM student s, program p
WHERE p.name = 'Information Engineering' and p.programID = s.programID;
```

4. Write a SQL-query that returns the name of all courses that have prerequisite courses.

Solution

```SQL
SELECT DISTINCT c.name
FROM course c, prerequisite pre
WHERE c.courseID = pre.advancedCourse;
```

5. Write a SQL-query that returns the sum of all credit points successfully achieved by student “John Wayne”. Keep in mind that the credit points only count when the student has an attempt with a grade of 5 or more points.

Solution

```SQL
SELECT SUM(c.creditPoints) AS SUM
FROM student s, course c, attempts a
WHERE s.firstname = 'John' AND s.lastname = 'Wayne' AND a.studentID =
s.studentID
AND a.courseID = c.courseID AND a.grade > 4;
```

6. A student needs to be removed from the database. Write SQL-statements to remove the student with the name “John Wayne” from the database.

Solution

```SQL
DELETE FROM attempts a
   WHERE a.studentID IN (
      SELECT s.studentID FROM Student s
      WHERE s.firstName = 'John'
      AND s.lastName = 'Wayne');

DELETE FROM Student
   WHERE firstName = 'John'
   AND lastName = 'Wayne';
```

2. Assignment 2: SQL-statements for a Shipping company

A shipping company wants to use a SQL-database to keep track of its ships and employed sailors based on the following relation schema:

- **HARBOR** (harborID, location, establishedIn)
- **SAILOR** (sailorID, lastName, dob, trainedAt(FK -> harborID))
- **SHIP** (shipID, name, grossWeight, launchDate, baseHarbor(FK -> harborID))
- **HIRE** (sailor(FK -> sailorID), ship(FK -> shipID), startOfService, annualSalary)

You can use the provided SQL-script for creating the tables and inserting some data in the tables.

1. Create a SQL-query that returns the dob (date of birth) of sailors in descending order that were hired on August 3rd, 2012.

Solution

```sql
SELECT lastname, dob
from sailor s, hire h
where s.sailorID = h.sailorID AND
h.startOfService = '2012-08-03'
ORDER BY dob DESC;
```

2. Create a SQL-query that returns all information of sailors that were hired between July 3rd, 2011, and September 3rd, 2012, and whose last name starts with a 'J'.

Solution

```sql
SELECT *
from sailor s, hire h
where s.sailorID = h.sailorID AND
h.startOfService BETWEEN '2011-07-03' AND '2012-09-03'
AND lastname like 'J%';
```

3. Create a SQL-query that returns for each ship the sum of the annual salary of every sailor who is hired for that ship.

Solution

```sql
SELECT * from hire;

SELECT s.name, SUM(h.annualSalary)
FROM ship s, hire h
WHERE s.shipID = h.shipID
GROUP BY s.shipID;
```

4. Create a SQL-query that returns the location of all harbors that are not base harbor to any ship in the database.

Solution

```SQL
Select h.location
FROM Harbour h
where h.harbourID NOT IN (SELECT s.baseharbour from Ship s);
```

5. Create a SQL-query that returns the shipId, ship name and the number of sailors who are hired on the ship and earn maximum 42.000$.

Solution

```SQL
SELECT * from hire;

select h.shipid, s.name, COUNT(h.sailorid)
FROM ship s, hire h
WHERE h.shipid = s.shipid
GROUP BY h.shipid
HAVING MAX(h.annualSalary) < 42000;
```

6. Describe in your own words the result of the following query:

```SQL
SELECT DISTINCT h1.location
FROM SHIP s1, SHIP s2, HARBOR h1, HARBOR h2
WHERE s1.baseHarbor = h1.harborID
AND s2.baseHarbor = h2.harborID
AND s1.launchDate = s2.launchDate
AND h1.location = h2.location
AND h1.harborID != h2.harborID;
```

Solution

A correct answer would include:

- This query outputs the distinct (or deduplicated) locations from the harbours
- Four tables, with all ships and harbours are selected twice each.
- Only harbours are selected that are home to a ship.
- Only ships that have the same launch date are selected.
- Only harbours that are in the same location, but are different harbours are selected.
- The result is a list of locations that have two distinct harbours, that are the home harbour of two different ship, with those two different ships that having the same launch date.

3. Assignment 3: SQL-statements for the COMPANY example from Elmasri also used in the lecture

Let's have a look on the COPMPANY example from the book „Fundamentals of Database Systems" from Elmasri which is also used in the lecture. Given is the database schema in Figure 1 and the database state in Figure 2.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Write SQL statement for the following tasks:

1. Retrieve the names of all employees in department 5 who work more than 10 hours per week on a project.

Solution

```sql
SELECT first_name, last_name
FROM employee e
LEFT JOIN works_on wo ON e.ssn = wo.essn
WHERE e.dno = 5
AND wo.hours >= 10;
```

2. List the names of all employees who have a dependent with the same first name as themselves.

Solution

```sql
SELECT first_name, last_name
FROM employee e
LEFT JOIN DEPENDENT d ON e.ssn = d.essn
WHERE d.dependent_name = e.first_name;
```

3. Find the names of all employees who are directly supervised by ‘Franklin Wong’.

Solution

```sql
SELECT first_name, last_name
FROM employee e
WHERE super_ssn = 333445555;
```

4. Suppose that the EMPLOYEE table’s constraint EMPSUPERFK as specified below is changed to read as follows:

```sql
CONSTRAINT EMPSUPERFK  FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
ON DELETE CASCADE ON UPDATE CASCADE;
```

Answer the following questions:

- What happens when the following command is run on the database state?

```sql
DELETE FROM EMPLOYEE WHERE Lname = ‘Borg’ ;
```

Solution

The entire database entries of employee would get deleted, since Borg does not have a supervisor and everyone else does (provided there is no other constraint to the table). However, since there is a foreign key constraint on the department table that references employee, the operation will return an error.

- Is it better to `CASCADE` or `SET NULL` in case of `EMPSUPERFK` constraint `ON DELETE`?

Solution

In this case, it would be better to `SET NULL`, since when only the head of the company changes, the res of the employees will probably still remain.

5. For each project, list the project name and the total hours per week (by all employees) spent on that project.

Solution

```sql
SELECT
  pname,
  SUM(hours)
FROM project p
LEFT JOIN works_on wo ON wo.pno = p.pnumber
GROUP BY pname;
```

6. Retrieve the average salary of all female employees.

Solution

```sql
SELECT
  AVG(Salary)
FROM employee e
WHERE SEX = 'F'
GROUP BY SEX;
```

7. Write SQL statements to create a table EMPLOYEE_BACKUP to back up the EMPLOYEE table shown.

Solution

```sql
CREATE TABLE employee_backup SELECT * FROM employee;
```

8. For each department, whose average employee salary is more than $30,000, retrieve the department name and the number of employees working for that department.

Solution

```SQL
SELECT d.dname, d.dno, COUNT(e.lname), AVG(salary)
FROM department d
LEFT JOIN employee e ON e.dno = d.dnumber
GROUP BY dno, dname
HAVING AVG(salary) > 30000;
```

4. Assignment 4: Relational Algebra vs. SQL query for a Cinema Database

The following excerpt from a database schema models a database about a cinema. The following assignments are to be answered in the form of an SQL query.

- **Movies** (FilmID (PK), Title, Director, Release Year, Genre)
- **MovieHasActor** (MovieHasActorID (PK), FilmID (FK), ActorID (FK), RoleName)
- **Actors** (ActorID (PK), First Name, Last Name, Birthdate)
- **Screenings** (ScreeningID (PK), FilmID (FK), Cinema Hall, Date, Time)
- **Reservations** (ReservationID (PK), ScreeningID (FK), Seat, Customer Name, Booking Date)

1. Display a list of actors (first name, last name) and their roles in a specific movie (e.g., “FilmXYZ”).

Solution

```sql
SELECT a.first_name, a.last_name, ma.role_name, m.title
FROM actors a
LEFT JOIN movie_has_actor ma ON ma.actor_id = a.actor_id
LEFT JOIN movies m ON m.film_id = ma.film_id
WHERE m.title = 'FilmXYZ';
```

2. Find all movies that will be shown in the screenings (ScreeningID) for the movie theater “HallA” on 2024-01-30 at 19:00.

Solution

```sql
SELECT ScreeningID
FROM Screenings s
WHERE cinema_hall = 'HallA'
AND date = '2024-01-30'
AND time = '19:00';
```

3. Create a table with information about all reservations made by customers with the last name “Schmidt”, including the movie title and seat number.

Solution

```SQL
CREATE TABLE CUSTOMER_SCHMIDT
SELECT
  r.reservation_id,
  r.seat,
  r.customer_name,
  r.booking_date,
  m.title
FROM RESERVATIONS r
LEFT JOIN screenings s ON s.screening_id = r.screening_id
LEFT JOIN movies m ON m.film_id = s.film_id
WHERE r.customer_name like '%Schmidt';
```