DBS Mandatory Assignment 1

R. Brooks & T. Mortensen

Spring 2022

Introduction

The first of three mandatory assignment in DBS deals with SQL-queries, and more specifically data query language (DQL). The assignment has three sub-assignments where the first one departs from the data which you created in one of the class cases. The second assignment is a crime mystery case where you have to solve the mystery by using SQL queries on the supplied database. The last assignment is non-code related and you have to answer specific questions.

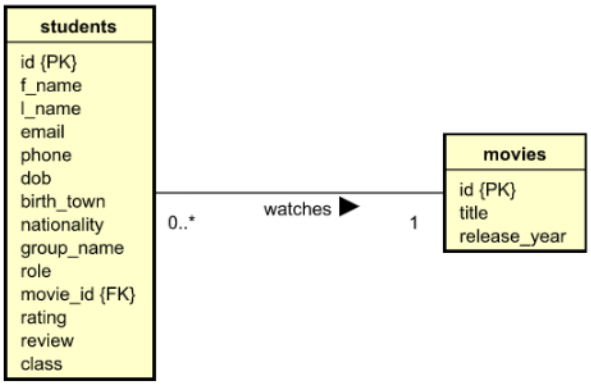
The assignments must be made in the groups that correspond to the groups which you defined in the case data for sub-assignment 1. Under each assignment it is specifically written which material you should use, what the deliveries are, and how to upload. Make sure you read each assignment carefully.

Assignment 1: Students and movies

This assignment is associated with three .sql files:

- students_and_movies_ddl.sql
- students_and_movies_data.sql
- students_and_movies_queries.sql

You create the database by running the first script and then import the data by running the second second script. Remember to run the scripts in the same way that we've done in the exercises. The third script contains all the questions/queries that you have to answer. Here is the GR of the database:



Deliveries:

i. You must fill in the template script "students_and_movies_queries.sql" so that all queries are executable. You must upload this script. The name of the script has to be "[group name]_assignment1.sql"

ii. You must create a document where you state each question/query and the output/answer/table that is obtained when you run the query. If the output table is very large, only copy the first couple of rows. You can simply take screen shots of your output and paste them into your document. The document must be converted to a .pdf and must be uploaded as such. The name of the pdf has to be "[group name]_assignment1.pdf"

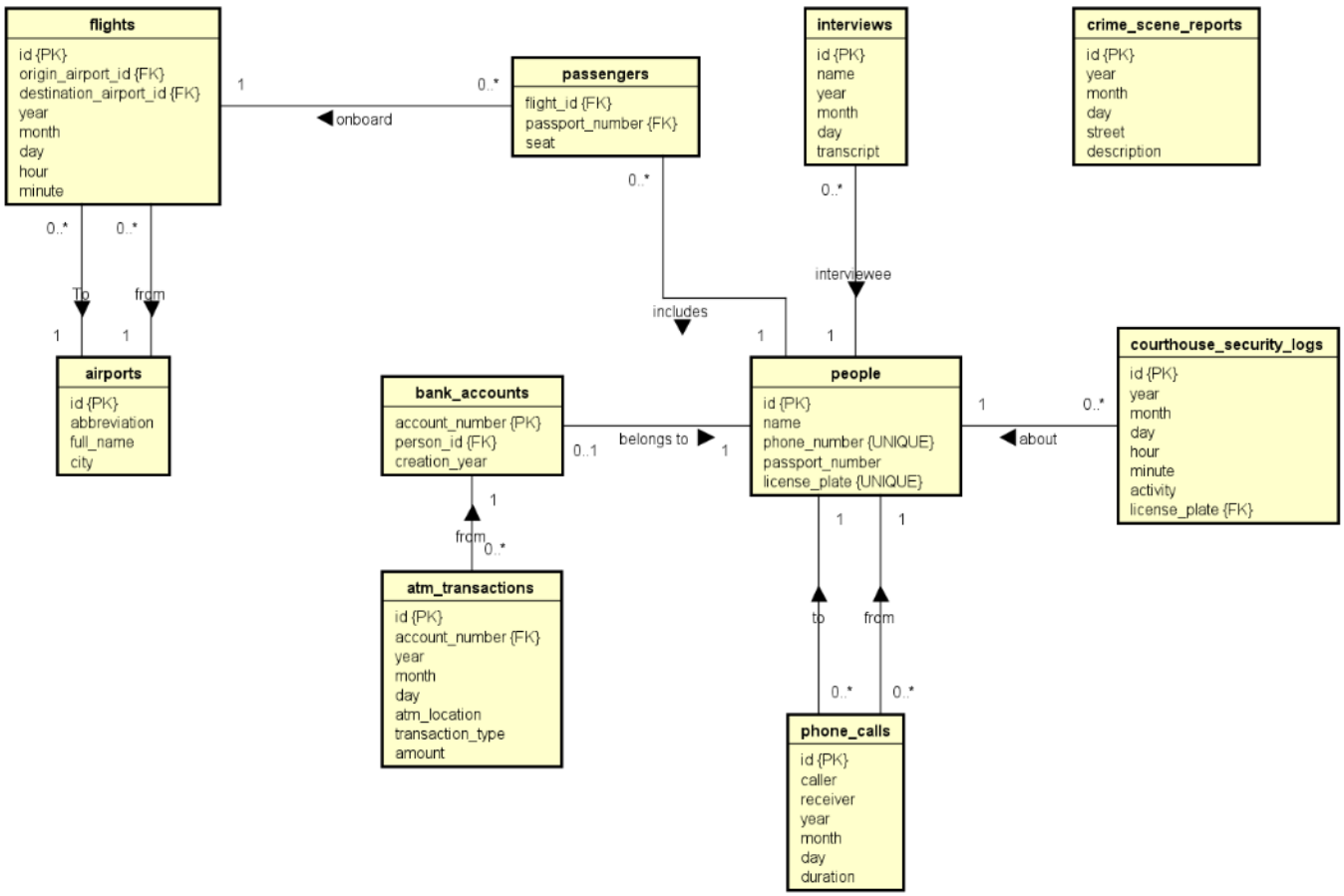
The assessment of this sub-questions will primarily be based in the correctness of your SQL queries that answer the questions.

Assignment 2: The Robbery in Shadyville

The beloved DBS Talisman has been stolen! The residents of the town of Shadyville are bewildered and need your help to solve the mystery of the stolen Talisman. Authorities believe the thief stole the Talisman and then, shortly after, took a plane out of town with the help of an accomplice. Your task is to figure out:

- Who the thief is
- Which city the thief fled to
- Who the accomplice of the thief is

The only thing you know is that the theft took place **July 28, 2020** and that it took place on **Chamberlin Street**. Fortunately, the authorities have stored a lot of information from the time around the theft and created a PostgreSQL database for you.



Data can be accessed by first creating the database using shadyville_ddl.sql and then inserting data using shadyville_data.sql. You can then query the database and obtain relevant information so you can help the residents of Shadyville.

The solution to this assignment simply consists of answering the three questions above. **But the important thing, and where you score points, is the process that led you to the solution of the mystery.** You must save this process in a file named "[group name]_shadyville_log.sql", in which you specify all the SQL queries you have run in the database. Above each query you must describe why you are running the query in question and/or what information you hope to get out of the query. You can comment in SQL code by either "- -" or "/* [insert comment] */".

The assessment of this sub-questions will primarily be based in how well you describe your process in terms of code comments.

Assignment 3: Code to Query

In this sub-assignment you have to move the other way: you have to specify the query that the SQL-statements in Q1-Q8 output. In the remaining questions, it is stated what you have to do. Reference is made to an arbitrary database, unless something else is expressly stated. You must try to describe exactly what the SQL does, both in terms of columns, rows, conditions, etc. Example:

```
SELECT *
    FROM staff
    WHERE fname NOT IN ('Conny' , 'Laila');
```

Answer: Shows all rows in the `staff` table where `fname` is not either 'Conny' or 'Laila'.

Deliveries:
Create a document where you answer all questions, convert it to pdf and upload it. The name of the pdf has to be "[group name]_assignment3.pdf"

Q1

```
SELECT DISTINCT salary
    FROM staff s1
    WHERE (SELECT COUNT(DISTINCT Salary)
        FROM staff s2
        WHERE s1.salary <= s2.salary) = 2;
```

Q2

```
SELECT DISTINCT salary
    FROM staff a
    WHERE 3 >= (SELECT COUNT(DISTINCT salary)
        FROM staff b
        WHERE a.salary <= b.salary)
    ORDER BY a.salary DESC;
```

Q3

```
SELECT *
    FROM staff
    WHERE branchNo
        NOT IN (SELECT branchNo FROM branch);
```

Q4

```sql
SELECT branchno, MAX(salary) as _max
    FROM staff
    GROUP BY branchno
    ORDER BY _max;
```

Q5

```sql
SELECT fname
    FROM staff
    WHERE DOB LIKE '196%' AND salary > 10000;
```

Q6

```sql
SELECT staffno, COUNT(*)
    FROM staff
    GROUP BY staffno
    HAVING COUNT(staffno) > 1;
```

Q7

```sql
SELECT substring(_position,1,5)
    FROM staff;
```

Q8

```sql
SELECT DISTINCT _position, LENGTH(_position)
    FROM staff
    WHERE _position IS NOT NULL;
```

Q9

Consider the two SQL-statements below. One of them finds titles that are not biographies and that do not have a price of less than 20. Determine which one and then explain what the other statement does.

```sql
SELECT title_id, type, price
    FROM titles
    WHERE NOT type = 'biography'
        AND NOT price < 20;
```

```sql
SELECT title_id, type, price
    FROM titles
    WHERE NOT type = 'biography'
        AND price < 20;
```

Q10

Consider the following two SQL-statements. Explain the differences between them, i.e. explain what each of them extract.

```sql
SELECT title_id, type, price
    FROM titles
    WHERE type = 'history'
        OR type = 'biography'
        AND price < 20;
```

```sql
SELECT title_id, type, price
    FROM titles
    WHERE (type = 'history'
        OR type = 'biography')
        AND price < 20;
```

Q11

Re-write the below SQL such that it returns the same but without using `NOT`.

```sql
SELECT * FROM mytable
    WHERE col1 = 1
    AND NOT (col1 = col2 OR col3 = 3);
```

Q12

The following SQL returns the different names of employees working in both the IT and Finance departments in the following database:

employee	(id, name, salary)
office	(number, telephone)
department	(name, address)
workfor	(employee_id, dept_name, office_num)

Rewrite the SQL below so that it returns the same, but without using subqueries. Feel free to use a join in a `WHERE` clause if it makes it easier.

```sql
SELECT DISTINCT e.name
    FROM employee e, workfor w1
    WHERE e.id=w1.employee_id
        AND w1.dept_name='IT'
        AND e.id IN (SELECT w.employee_id
                     FROM workfor w
                     WHERE w.dept_name='Finance');
```

Q13

The SQL below refers to the following database

Person	(pid, name)
ReadBook	(pid, isbn)
Book	(isbn, title, price)

Explain what the following SQL returns.

```sql
SELECT P.name
    FROM Person P
    WHERE EXISTS (SELECT *
        FROM Book B
        WHERE NOT EXISTS (SELECT *
            FROM ReadBook R
            WHERE R.isbn = B.isbn AND R.pid = P.pid));
```