Databases
Exercise Sheet

Andrej Ivašković (ai294)
Compiled on: 9th October 2019

1 Introduction and relational databases

1.1 Before attempting the problems

This set of exercises covers the part of the course that gives an overview of relational databases (without any SQL).

When studying databases, remember to think about whether and how concurrent/distributed queries are supported. This is explored in more detail in *Concurrent and Distributed Systems*, but concurrency is a very important practical concern that motivates some of the design decisions.

1.2 Problems

Exercise 1.1. 'The amount of redundancy leads to fundamental tradeoffs in database design and choice of data representation'. Discuss this statement.

Exercise 1.2. What are the ACID properties in the context of databases?

Exercise 1.3. Explain how *data redundancy* and *data duplication* are distinct concepts in database design and implementation.

Exercise 1.4. State the formal definitions of the following concepts (and introduce notation when needed):

(*a*) natural join;

(*b*) superkey;

(*c*) key;

(*d*) foreign key.

Exercise 1.5. Consider two entities A and B that are in a relationship with each other. Explain how a one-to-one, one-to-many, and a many-to-many relationship between them is represented in a relational database.

Exercise 1.6. Why does the need for *weak entities* arise?

Exercise 1.7. What are *data anomalies* in the context of relational databases?

Exercise 1.8. What does it mean for a database to have *referential integrity*? Why is it an important concept?

Exercise 1.9. Let *duplication* be an operation on relations that creates a new relation with a higher dimension, where some attributes are 'duplicated' (that is, they always have the same values in tuples that are elements of this relation). Denote duplication of a set of attributes $\mathbf{A}$ with $\delta_{\mathbf{A}}$.

Show how to express $\delta_{\mathbf{A}}(R)$ in Relational Algebra.

2 SQL

2.1 Before attempting the exercises

You will have to write some SQL code as part of these exercises. Keep in mind the following points:

- give columns of a table that is the result of a `SELECT` statement meaningful names using aliases (`SELECT value AS alias`);
- use `JOIN` statements rather than nested queries;
- in general, avoid nested queries unless you really have to use them.

You should approach three-valued logic exercises with truth tables.

2.2 Problems

Exercise 2.1. The example given in Lecture 4 shows how you can encode several different relations using a single table. Discuss the pros and cons of this approach.

Exercise 2.2. Write and run a `SELECT` query for the film database from the first practical:

(*a*) that returns the names, German release dates and character names (expected format `|FILM|RELEASE DATE|CHARACTER|`) of all comedy films starring Adam Sandler, where the `RELEASE DATE` column may be blank if the release date is not known;

(*b*) that returns all film names, along with the names of their directors (expected format `|FILM|DIRECTOR|`) that were released in the United States in 2011 and received an R certificate, lexicographically ordered by the name of the film;

(*c*) that returns the names of all films directed by Ridley Scott with more than two writers.

Exercise 2.3. Suppose you are working on IT tasks for a company that provides catering services for weddings, parties and businesses. One of the requests you received from accounting and management departments is to create a handy database system that will allow them to keep track of all company finances, clients and employees. You are only told that the users of this database should be able to retrieve information about the company performance with a click of a button.

This database should contain details about all employee salaries, all events done for clients, the necessary information about these clients, which employees attended which event, as well as the costs required to organise a particular event (not counting buying all the ingredients through deliveries, which are usually bought in bulk for multiple events). The company policy is to provide a full refund in the case an event is cancelled, with the refund happening as soon as the cancellation is processed.

Note that the answers will depend on how you decide to interpret and implement the requirements. This is an open-ended question.

(*a*) Draw an ER diagram to show how you would model this scenario.

(*b*) Give a simple overview of the tables you would have in your database. State all of the constraints.

(*c*) Write SQL `SELECT` queries that will perform the following tasks:

  (*i*) list the details of all employees who attended a particular event;

  (*ii*) list the details of all employees whose average salary is greater than the average salary of all company employees;

  (*iii*) compute the total company profit between the given start and end dates;

  (*iv*) compute how much each client has paid for their events (returned as a two column result `|CLIENT_NAME|AMOUNT|`, with clients sorted by how much they have paid).

(*d*) Suppose that, after a few months of using the database, you are asked to optionally provide a way to associate some deliveries with a particular event. How would you change your schemas? What are the difficulties you might run into?

Exercise 2.4. Recall that database indexes are used for improving efficiency of certain kinds of queries. In relational databases, these are usually defined on a column (or non-empty set of columns). Suppose you decided to use an *inverted index* on a particular attribute, which is represented by a map from all values present in the column to references to rows of the table. What are the kinds of queries for which a query optimiser would consider using this kind of index?

Exercise 2.5. Other than indexing, one strategy used by database management systems for improved benchmarks is *caching*: you can store views behind the scenes that contain the results of recent or common `SELECT` queries. Describe a real-life scenario in which caching significantly improves performance and suggest a simple caching strategy.

Exercise 2.6. Can you represent `GROUP BY` in Relational Algebra (as presented in the lectures)?

Exercise 2.7. State why `NULL` is used in practice, as well as the problems caused by it.

Exercise 2.8. *(Supervision 2, Question 1 from the suggested exercises)* On our movie database, will the SQL query

```
SELECT person_id, movie_id, position FROM credits
WHERE NOT((NOT (position IS NULL)) OR position = 17);
```

return the same results as the query

```
SELECT person_id, movie_id, position FROM credits
WHERE position <> 17;
```

Exercise 2.9. *(Supervision 2, Question 3 from the suggested exercises)* In boolean logic, the expression `not(a and b)` always has the same value as `(not a) or (not b)`. Is that true in 3-valued logic? What about the expression `not(a or b)` and the expression `(not a) and (not b)`? Is the expression `a or (not a)` always true in 3-valued logic? If not, can you extend this axiom to make it always true?

Exercise 2.10. Look up the keyword `RECURSIVE` in SQL. How can you compute Bacon numbers using it?

3 Beyond SQL

3.1 Before attempting the problems

These exercises cover the part of the course that explores approaches to databases other than relational. The key takeaway is understanding why they exist and when they are applicable.

For amusement (somewhat NSFW), watch *Mongo DB Is Web Scale* on YouTube (`https://www.youtube.com/watch?v=b2F-DItXtZs`) or on its original website (`http://www.mongodb-is-web-scale.com/`).

3.2 Problems

Exercise 3.1. What are the use cases for *relational*, *graph* and *document* databases?

Exercise 3.2. Write a Cypher query that is equivalent to your solution of Exercise 1.b that you did for the practicals.

Exercise 3.3. *(Supervision 2, Question 5 from the suggested exercises)* Our SQL tutorial presented several examples of `LEFT JOIN` such as

```
SELECT title, person_id
FROM movies
LEFT JOIN credits ON movie_id = id AND type = 'costume_designer';
```

In Cypher, that query can be written with an "optional match" as

```
match (m:Movie)
optional match (m)<-[:COSTUME_DESIGNER_FOR]-(p:Person)
return m.title as title, p.person_id as person_id;
```

As with the `LEFT JOIN`, this returns a null value for `person_id` when there is no match in the `optional match` clause. How might we modify this query to only return the titles of movies that have no costume designer?

Exercise 3.4. Consider the following set of tables, representing two entities in a many-to-many relation:

$S(\underline{sid}, A)$ $R(\underline{sid}, \underline{tid}, B)$ $T(\underline{tid}, C)$

sid	A
s1	a1
s2	a2
s3	a3

sid	tid	B
s1	t1	b1
s1	t2	b2
s2	t1	b4
s2	t3	b5

tid	C
t1	c1
t2	c2
t3	c3
t4	c4

(*a*) Write an SQL query that returns all values a (found in table S, in column A) that are associated via R with an entry in T that have the value c in the C column.

(*b*) Represent this set of tables as three JSON objects, each associated with one of the values of the *sid* key. Is any information lost?

(*c*) How would you implement the query from (*a*) with the document database from (*b*)?

Exercise 3.5. How do the tasks of designing OLTP and OLAP systems differ?

Exercise 3.6. Explain the main operations of the *OLAP cube* model and how a *star schema* is used in that model.