1 Before attempting the problems

This exercise sheet covers the part of the course that explores database models other than the relational. The key takeaway is understanding why the different approaches exists and what they are good for. Another thing considered are the limitations of SQL. This corresponds to lectures 5, 6, 7, 8.

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

2 Problems

1. What are the use cases for relational, graph and document databases?

2. 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?

3. 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.

4. How can you represent GROUP BY in Relational Algebra?

5. State why NULL is used in practice, as well as the problems caused by it.
6. (Supervision 2, Question 1 from the suggested exercises) On our movie database, will the SQL query

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

return the same results as the query

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

7. (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?

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

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

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

```sql
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

```cypher
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?

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

12. Explain the notion of eventual consistency and why it is preferred to strong consistency.