# Database Systems Overview Homework

1. **What database models do you know?**

   - Hierarchical model – tree-like structure.
   - Network / graph model – uses graphs
   - Relational – the most used one. Uses tables and relations between them.
   - Object-oriented – uses C# like objects.

2. **Which are the main functions performed by a Relational Database Management System (RDBMS)?**

   The main functions performed by a RDBMS are management of the data. This includes Creating / altering / deleting tables and relationships between them (database schema), adding, hanging, deleting, searching and retrieving of data stored in the tables. It also supports the SQL language and manages transactions (or most of them).

3. **Define what is "table" in database terms.**

   Database tables consist of data, arranged in rows and columns. All rows have the same structure. Columns have name and type (number, string, date, image, or other).

4. **Explain the difference between a primary and a foreign key.**

   Primary key is a column of the table that uniquely identifies its rows (usually it is a number). The foreign key is an identifier of a record located in another table (usually its primary key). In most cases a key in a table, with the same name (CourseID in a Courses table) is it's primary key and with a different name (StudentID in a Courses table) is most usually a foreign key.

5. **Explain the different kinds of relationships between tables in relational databases.**

   One-to-many (or many-to-one) - a single record in the first table has many corresponding records in the second table.

   Many-to-many - records in the first table have many corresponding records in the second one and vice versa. Implemented through additional table.

   One-to-one - a single record in a table corresponds to a single record in the other table. Used to model inheritance between tables

6. **When is a certain database schema normalized? What are the advantages of normalized databases?**

   Normalization of the relational schema removes repeating data. The advantages are that by having less data repeated the overall storage space consumed by the database is less.

7. **What are database integrity constraints and when are they used?**

Database integrity constraints are rules that ensure that the data entered in the database is valid (by setting those validations rules in advance). For example we can constrain an age to not be less than 0 and greater than 150. When we try to save incorrect data into the DB we will get an error instead of invalid data.

8. **Point out the pros and cons of using indexes in a database.**

Indexes are very helpful when we have large amount of data and we want to be able to search fast by given property. In this case we create an index on this property. This has it's cost – every index creates a B-tree which makes creating deleting and altering data connected to this property (including itself) slower – as the B-tree has to rearrange from time to time as new data is inserted.

9. **What's the main purpose of the SQL language?**

Standardized declarative language is used for manipulation of relational databases. SQL language supports creating, altering, deleting tables and other objects in the database as well as searching, retrieving, inserting, modifying and deleting table data (rows) and many others.

10. **What are transactions used for? Give an example.**

Transactions are used to execute several actions as one. We want to execute all the actions and not only half of them. If any of them fails we want to roll-back the effects of the previously executed actions – in other words if 2 of 3 actions execute and the 3rd throws an error, we don't want to save the changes done by actions 1 and 2. For example when we draw money, we want to change our balance, get the money physically and keep a log for the transactions. If any of these actions fails, we don't want to lose this money, as the new balance is saved in step 1.

11. **What is a NoSQL database?**

NoSQL DB use document-based model. It's main idea is to be non-relational. It still supports CRUD operations but holds all the relevant information for an entity in a single document, instead of joining several tables to extract this information.

12. **Explain the classical non-relational data models.**

Non-relational data models store all the information regarding an entity in a single file – imagine all the details for you (name, address, town, country etc.) in a single file. This way no joins or lookups in different files is required. In comparison in the relational data model you will have to extract information from each table (names, addresses, towns, countries etc.) and join them, to acquire all the information for you.

13. **<u>Give few examples of NoSQL databases and their pros and cons.</u>**

MongoDB, Redis, CouchDB are the most famous NoSQL Databases. The pros and cons of using NoSQL are the following:

*Pros:*

- Mostly open source.
- Horizontal scalability. There's no need for complex joins and data can be easily sharded and processed in parallel.
- Support for Map/Reduce. This is a simple paradigm that allows for scaling computation on cluster of computing nodes.
- No need to develop fine-grained data model – it saves development time.
- Easy to use.
- Very fast for adding new data and for simple operations/queries.
- No need to make significant changes in code when data structure is modified.
- Ability to store complex data types (for document based solutions) in a single item of storage.

*Cons*

- Immaturity. Still lots of rough edges.
- Possible database administration issues. NoSQL often sacrifices features that are present in SQL solutions "by default" for the sake of performance. For example, one needs to check different data durability modes and journaling in order not to be caught by surprise after a cold restart of the system. Memory consumption is one more important chapter to read up on in the database manual because memory is usually heavily used.
- No indexing support (Some solutions like MongoDB have indexing but it's not as powerful as in SQL solutions).
- No ACID (Some solutions have just atomicity support on single object level).
- Bad reporting performance.
- Complex consistency models (like eventual consistency). CAP theorem states that it's not possible to achieve consistency, availability and partitioning tolerance at the same time. NoSQL vendors are trying to make their solutions as fast as possible and consistency is most typical trade-off.
- Absence of standardization. No standard APIs or query language. It means that migration to a solution from different vendor is more costly. Also there are no standard tools (e.g. for reporting)