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

   - **Hierarchical (tree)**
   - **Network / graph**
   - **Relational (table)**
   - **Object-oriented**
   - **Document model**

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

   - **Creating / altering / deleting tables and relationships between them (database schema)**
   - **Adding, changing, deleting, searching and retrieving of data stored in the tables**
   - **Support for SQL language**
   - **Optionally could support transactions**

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 its is a number). The foreign key is an identifier of a record located in another table (usually its primary key). The difference is that in a table the primary key is unique and the foreign key could be repeated in this table, it is usually unique in another table. It connects the two tables.

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

   - Relationship one-to-many (or many-to-one) A single record in the first table has many corresponding records in the second table

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

- Relationship 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?**

   - **It is normalized when repeated data is avoided.**

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

   - **Integrity constraints ensure data integrity in the database tables. Enforce data rules which cannot be violated**
   - **Primary key constraint .Ensures that the primary key of a table has unique value for each table row**
   - **Unique key constraint. Ensures that all values in a certain column (or a group of columns) are unique**
   - **Foreign key constraint. Ensures that the value in given column is a key from another table**
   - **Check constraint. Ensures that values in a certain column meet some predefined condition**

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

Pros: Indices speed up searching of values in a certain column or group of columns. Usually implemented as B-trees. Indices can be built-in the table (clustered) or stored externally (non-clustered).

Cons: Adding and deleting records in indexed tables is slower! Indices should be used for big tables only (e.g. 50 000 rows).

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

Standardized declarative language for manipulation of relational databases. SQL-99 is currently in use in most databases. SQL language supports: Creating, altering, deleting tables and other objects in the database. Searching, retrieving, inserting, modifying and deleting table data (rows).

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

Transactions are a sequence of database operations which are executed as a single unit: Either all of them execute successfully. Or none of them is executed at all.

Example: A bank transfer from one account into another (withdrawal + deposit).If either the withdrawal or the deposit fails the entire operation should be cancelled.

11. What is a NoSQL database?

In NoSQL databases the data is stored as documents. A single entity (document) is a single record. Documents in NoSQL databases do not have a fixed structure.

12. Explain the classical non-relational data models

- **Document model : Set of documents, e.g. JSON strings**
- **Key-value model : Set of key-value pairs**
- **Hierarchical key-value : Hierarchy of key-value pairs**
- **Wide-column model : Key-value model with schema**
- **Object model : Set of OOP - style objects**

13. Give few examples of NoSQL databases and their pros and cons.

MONGODB

PROS:

Sharding and Load-Balancing:

Sharding is the process of storing data records across multiple machines and is MongoDB's approach to meeting the demands of data growth. As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable read and write throughput. Sharding solves the problem with horizontal scaling. With sharding, you add more machines to support data growth and the demands of read and write operations.

When you have extremely large amounts of data or you need to distribute your database traffic across multiple machines for load-balancing purposes, MongoDB has heavy advantages over many classic relational databases such as MySQL.

Speed

MongoDB queries can be much faster in some cases, especially since your data is typically all in once place and can be retrieved in a single lookup. However, this

advantage only exists when your data is truly a document. When your data is essentially emulating a relational model, your code ends up performing many independent queries in order to retrieve a single document and can become much slower than a classic RDBMS.

Flexibility

MongoDB doesn't require a unified data structure across all objects, so when it is not possible to ensure that your data will all be structured consistently, MongoDB can be much simpler to use than an RDBMS. However, data consistency is a good thing, so when possible you should always attempt to ensure that a unified structure will be applied.

CONS:

No Joins

In MongoDB there exists no possibility for joins like in a relational database. This means that when you need this type of functionality, you need to make multiple queries and join the data manually within your code (which can lead to slow, ugly code, and reduced flexibility when the structure changes).

Memory Usage

MongoDB has the natural tendency to use up more memory because it has to store the key names within each document. This is due to the fact that the data structure is not necessarily consistent amongst the data objects.

Additionally you're stuck with duplicate data since there is no possibility for joins, or slow queries due to the need to perform the join within your code. To solve the problem of duplicate data in Mongo, you can store references to objects (i.e. BSON ids), however if you find yourself doing this it indicates that the data is actually "relational", and perhaps a relational database suits your needs better.

Concurrency Issues

When you perform a write operation in MongoDB, it creates a lock on the entire database, not just the affected entries, and not just for a particular connection. This lock blocks not only other write operations, but also read operations.

COUCHDB

PROS:

Simplicity. You can store any JSON data, and each document can have any number of binary attachments.

Thanks to map/reduce, querying data is somewhat separated from the data itself. This means that you can index deeply within your data, and on whether or not something exists, and across types, without paying a significant penalty. You just need to write your view functions to handle them.

CONS:

Arbitrary queries are expensive. To do a query that you haven't created a view for, you need to create a temporary view. This can be solved to some extent by using Lucene.

There's a bit of extra space overhead with CouchDB compared to most alternatives.