1. What database models do you know?

- A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model, which uses a table-based format. Other model which I know is the Object-Oriented-Model and the Network-Model.

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

- A relational DBMS is special system software that is used to manage the organization, storage, access, security and integrity of data. This specialized software allows application systems to focus on the user interface, data validation and screen navigation. When there is a need to add, modify, delete or display data, the application system simply makes a "call" to the RDBMS.

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

- A table is a collection of related data held in a structured format within a database. It consists of fields (columns), and rows.

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

- Primary Key:

It will not allow "Null values" and "Duplicate values"

Foreign Key:

It will allow "Null values" and "Duplicate values" and it refers to a primary key in another table.

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

- One to Many

one-to-many relationship is the most common type of relationship. In a one-to-many relationship, a record in Table A can have many matching records in Table B, but a record in Table B has only one matching record in Table A.

- One to One

In a one-to-one relationship each record in one table has at most one related record in another table.

- Many to Many

In a many-to-many relationship, a record in Table A can have many matching records in Table B, and a record in Table B can have many matching records in Table A. This type of relationship is only possible by defining a third table (called a junction table) whose primary key consists of two fields ï¿� the foreign keys from both Tables A and B. A many-to-many relationship is really two one-to-many relationships with a third table.

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

- Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships.

- Advantages of normalization

Provide indexing

When the developer creates a primary key constraint, SQL Server will automatically create a unique clustered index on the column(s) for which the constraint has been created. A clustered index is created if no other clustered index has been defined for the table. In most cases, the creation of a clustered index will speed up data access and may increase insert, update, and delete performance. I should note that, for benchmarking, vendors frequently remove indexes to increase performance; Reduce table/row size; Enforce referential integrity

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

- Intergrity means something like 'be right' and consistent. The data in a database must be right and in good condition. There are the domain integrity, the entity integrity, the referential integrity and the foreign key integrity constraints.

- Domain integrity means the definition of a valid set of values for an attribute.

- The entity integrity constraint states that primary keys can't be null. There must be a proper value in the primary key field. This is because the primary key value is used to identify individual rows in a table. If there were null values for primary keys, it would mean that we could not indentify those rows.

- The referential integrity constraint is specified between two tables and it is used to maintain the consistency among rows between the two tables.

The rules are:

1. You can't delete a record from a primary table if matching records exist in a related table.

2. You can't change a primary key value in the primary table if that record has related records.

3. You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.

4. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.

- Any time you change the primary key of a row in the primary table, the foreign key values are updated in the matching rows in the related table. This constraint overrules rule 2 in the referential integrity constraints. Any time you delete a row in the primary table, the matching rows are automatically deleted in the related table. This constraint overrules rule 1 in the referential integrity constraints.

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

- The pros are:

Low overhead (locks, speed) in sliding large amounts of data into separate table for truncation (our primary concern)

Increased performance.

The cons are:

Dealing with foreign keys to the table when sliding the partitions around.

Additional data required in indexes (and queries) to have aligned indexes

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

- SQL is a query language designed for organizing, managing, developing and querying large relational databases over computer networks

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

- Database transaction is collection of SQL queries which forms a logical one task. For transaction to be completed successfully all SQL queries has to run successfully. Database transaction executes either all or none, so for example if your database transaction contains 4 SQL queries and one of them fails then change made by other 3 queries will be rolled back.

11. What is a NoSQL database?

NoSQL, or 'Not Only SQL', represents the new class of data management technologies designed to meet the increasing volume, velocity, and variety of data that organizations are storing, processing, and analyzing.

Compared to relational databases, NoSQL databases are more scalable and provide superior performance. NoSQL databases address the opportunities that the relational model does not, including:

* Large volumes of structured, semi-structured and unstructured data
* Agile sprints, quick iteration, and frequent code pushes
* Flexible, easy to use object-oriented programmingEfficient, scale-out architecture instead of expensive, monolithic architecture

12. Explain the classical non-relational data models.

Column: Accumulo, Cassandra, Druid, HBase

Document: Clusterpoint, Apache CouchDB, Couchbase, MarkLogic, MongoDB

Key-value: Dynamo, FoundationDB, MemcacheDB, Redis, Riak, FairCom c-treeACE

Graph: Allegro, Neo4J, InfiniteGraph, OrientDB, Virtuoso, Stardog

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

NoSQL pros (not ordered by importance):

* 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.

Examples: MongoDB, Cassandra

* 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)