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

Relational model, Entity–relationship model, Object model, Document model

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

A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. In the case of a Relational DBMS, it stores data in the form of related tables.

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

A table is something that organizes the information about a single topic into rows and columns. Each row contains a specific property about that topic. Ex : A Students table might have rows like name, birthdate, etc…

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

Primary key is the unique identifier for a row (data item) in the current table. A foreign key is also a unique identifier but for a row from another table. You can’t have more than 1 unique PK in a table, but you can have many FK. (EX: Let’s say that a student has an ID of 1. This will be his PK. Now, let’s say that this student is enrolled in a course with id 16. 16 is a FK. In the STUDENTS table there can be other students also enrolled in the course 16, but none of them can have the same ID as the first student- 1)

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

**One to many** -> The primary key from one of the tables is related to none, one or many records from another table.

**One to One** -> The primary key from one of the tables is related to exactly one record from another table.

**Many to many** -> Each record in both tables can relate to any number of records (or no records) in the other table.

1. **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. We can consider a database schema normalized when this process is completed by applying the normalization rules (at least the first 3 normalization forms)

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

FK constraint - For each record in table A, which is pointing to another record in table B, this record in the second table will exist.

PK constraint – Each record in the database will be uniquely indexed by a Primary Key.

Domain constraint - all columns in relational database must be declared upon a defined domain

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

Pros: Indexes are improving the efficiency of **data retrieval** and this reduces the cost of database IO. They are also reducing the cost of **sorting** and **grouping** the indexed data.

Cons: They decrease performance on inserts, updates, and deletes. Using WHERE clauses on indexes is slower than applying those operations on non-indexed tables. Also, they take up space (this increases with the number of fields used and the length of the fields).

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

It’s a language designed for managing data held in a relational database management system. Ex – applying CRUD operations on the data.

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

Transactions are used to assure that a given task, which contains many different and independent subtasks, will be successfully completed. This means that if one of the task fails, the other won’t be completed or if they were executed, the database will roll back itself to a pre-transaction state.

A classic example of this is a back transaction. The Task is

1. Take the money from the sender.
2. Put the money to a specific bank account
3. Send the money to the receiver’s bank
4. Credit the receiver’s account with the amount that was sent

If one of those steps fails, the sender will lose his money if the database wasn’t rolling back itself to previous state.

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

**NoSQL –** not only SQL are databases that aren’t relational. They provide better scalability that the traditional relational databases and for this are preferred for IoT, big data or document oriented applications. They are less constraining than the traditional databases.

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

**Document-oriented database** is a computer program designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. In contrast to relational databases and their notions of "Relations" (or "Tables"), these systems are designed around an abstract notion of a "Document".

1. **Give few examples of NoSQL databases and their pros and cons.**

[**http://tek.io/1npqZQx**](http://tek.io/1npqZQx) **- A really great article about NoSQL databases pros and cons.**

1: Elastic scaling

For years, database administrators have relied on scale up -- buying bigger servers as database load increases -- rather than scale out -- distributing the database across multiple hosts as load increases. However, as transaction rates and availability requirements increase, and as databases move into the cloud or onto virtualized environments, the economic advantages of scaling out on commodity hardware become irresistible.

RDBMS might not scale out easily on commodity clusters, but the new breed of NoSQL databases are designed to expand transparently to take advantage of new nodes, and they're usually designed with low-cost commodity hardware in mind.

2: Big data

Just as transaction rates have grown out of recognition over the last decade, the volumes of data that are being stored also have increased massively. O'Reilly has cleverly called this the "industrial revolution of data." RDBMS capacity has been growing to match these increases, but as with transaction rates, the constraints of data volumes that can be practically managed by a single RDBMS are becoming intolerable for some enterprises. Today, the volumes of "big data" that can be handled by NoSQL systems, such as Hadoop, outstrip what can be handled by the biggest RDBMS.

3: Goodbye DBAs (see you later?)

Despite the many manageability improvements claimed by RDBMS vendors over the years, high-end RDBMS systems can be maintained only with the assistance of expensive, highly trained DBAs. DBAs are intimately involved in the design, installation, and ongoing tuning of high-end RDBMS systems.

NoSQL databases are generally designed from the ground up to require less management: automatic repair, data distribution, and simpler data models lead to lower administration and tuning requirements -- in theory. In practice, it's likely that rumors of the DBA's death have been slightly exaggerated. Someone will always be accountable for the performance and availability of any mission-critical data store.

4: Economics

NoSQL databases typically use clusters of cheap commodity servers to manage the exploding data and transaction volumes, while RDBMS tends to rely on expensive proprietary servers and storage systems. The result is that the cost per gigabyte or transaction/second for NoSQL can be many times less than the cost for RDBMS, allowing you to store and process more data at a much lower price point.

5: Flexible data models

Change management is a big headache for large production RDBMS. Even minor changes to the data model of an RDBMS have to be carefully managed and may necessitate downtime or reduced service levels.

NoSQL databases have far more relaxed -- or even nonexistent -- data model restrictions. NoSQL Key Value stores and document databases allow the application to store virtually any structure it wants in a data element. Even the more rigidly defined BigTable-based NoSQL databases (Cassandra, HBase) typically allow new columns to be created without too much fuss.

The result is that application changes and database schema changes do not have to be managed as one complicated change unit. In theory, this will allow applications to iterate faster, though,clearly, there can be undesirable side effects if the application fails to manage data integrity.

Five challenges of NoSQL

The promise of the NoSQL database has generated a lot of enthusiasm, but there are many obstacles to overcome before they can appeal to mainstream enterprises. Here are a few of the top challenges.

1: Maturity

RDBMS systems have been around for a long time. NoSQL advocates will argue that their advancing age is a sign of their obsolescence, but for most CIOs, the maturity of the RDBMS is reassuring. For the most part, RDBMS systems are stable and richly functional. In comparison, most NoSQL alternatives are in pre-production versions with many key features yet to be implemented.

Living on the technological leading edge is an exciting prospect for many developers, but enterprises should approach it with extreme caution.

2: Support

Enterprises want the reassurance that if a key system fails, they will be able to get timely and competent support. All RDBMS vendors go to great lengths to provide a high level of enterprise support.

In contrast, most NoSQL systems are open source projects, and although there are usually one or more firms offering support for each NoSQL database, these companies often are small start-ups without the global reach, support resources, or credibility of an Oracle, Microsoft, or IBM.

3: Analytics and business intelligence

NoSQL databases have evolved to meet the scaling demands of modern Web 2.0 applications. Consequently, most of their feature set is oriented toward the demands of these applications. However, data in an application has value to the business that goes beyond the insert-read-update-delete cycle of a typical Web application. Businesses mine information in corporate databases to improve their efficiency and competitiveness, and business intelligence (BI) is a key IT issue for all medium to large companies.

NoSQL databases offer few facilities for ad-hoc query and analysis. Even a simple query requires significant programming expertise, and commonly used BI tools do not provide connectivity to NoSQL.

Some relief is provided by the emergence of solutions such as HIVE or PIG, which can provide easier access to data held in Hadoop clusters and perhaps eventually, other NoSQL databases. Quest Software has developed a product -- Toad for Cloud Databases -- that can provide ad-hoc query capabilities to a variety of NoSQL databases.

4: Administration

The design goals for NoSQL may be to provide a zero-admin solution, but the current reality falls well short of that goal. NoSQL today requires a lot of skill to install and a lot of effort to maintain.

5: Expertise

There are literally millions of developers throughout the world, and in every business segment, who are familiar with RDBMS concepts and programming. In contrast, almost every NoSQL developer is in a learning mode. This situation will address naturally over time, but for now, it's far easier to find experienced RDBMS programmers or administrators than a NoSQL expert.

Conclusion

NoSQL databases are becoming an increasingly important part of the database landscape, and when used appropriately, can offer real benefits. However, enterprises should proceed with caution with full awareness of the legitimate limitations and issues that are associated with these databases.