Q1. What is MongoDB and explain its basic concepts like document, collection, and database?

Ans: MongoDB is a NoSQL, document-oriented database that uses JSON-like documents to store data. The basic concepts of MongoDB include:

1. Document: A document in MongoDB is a single record in a collection and is roughly equivalent to a row in a relational database.
2. Collection: A collection is a group of related documents and functions as a table in a relational database.
3. Database: A database in MongoDB is a container for collections, similar to a schema in a relational database. A single MongoDB server can contain multiple databases.

Q2. How does MongoDB differ from traditional relational databases?

Ans: MongoDB differs from traditional relational databases in several ways:

1. Data Model: MongoDB uses a document-based data model, which allows for flexible and scalable data representation, while relational databases use a fixed table structure.
2. Scalability: MongoDB uses a horizontal scale-out approach, allowing for the easy and efficient addition of more capacity by adding more nodes to a cluster, while relational databases typically scale vertically by adding more powerful hardware.
3. Structure: MongoDB uses dynamic schemas, which means that documents in the same collection do not have to have the same structure. Relational databases require a fixed schema, which can make it difficult to modify the structure of the data over time.
4. Performance: MongoDB is designed for high performance, with built-in indexing, caching, and support for in-memory processing. Relational databases can also perform well, but the performance can depend on the specific implementation and configuration.

Q3. Can you explain the MongoDB document structure and the different data types it supports?

Ans: A MongoDB document is a collection of key-value pairs, similar to a JSON object. The structure of a MongoDB document is flexible and does not enforce a fixed schema, allowing for fields to be added or removed as needed.

MongoDB supports several data types, including:

1. String: For storing text data.
2. Number: For storing numeric data, including integers and floating-point numbers.
3. Boolean: For storing true/false values.
4. Array: For storing multiple values in a single field.
5. Date: For storing date and time values.
6. Object ID: For unique identification of documents within a collection.
7. Null: For representing missing or non-existent values.
8. Object: For embedding one document within another.
9. BinData: For storing binary data, such as images or other multimedia.

10. Timestamp: For storing a timestamp in a format that can be used to determine the creation and modification time of a document.

Q4. How does MongoDB handle indexing and explain the different types of indexes available?

Ans: Indexing in MongoDB is used to improve the performance of query operations by allowing the database to quickly locate the documents that match a query. MongoDB supports several types of indexes, including:

1. Single Field Indexes: Indexes on a single field, such as the _id field or a specific field in a document.
2. Compound Indexes: Indexes on multiple fields, allowing for efficient queries that match on multiple criteria.
3. Unique Indexes: Indexes that enforce uniqueness for a specific field, ensuring that there are no duplicate values.
4. Text Indexes: Indexes for searching text data, allowing for efficient searches for specific words or phrases.
5. Sparse Indexes: Indexes that only include documents that have the indexed field, excluding documents where the field is missing or has a null value.

Q5. Can you discuss MongoDB's scalability features and how they can be achieved using sharding?

Ans: MongoDB provides several features for scaling out to handle large amounts of data and high traffic. One of the primary methods for scaling MongoDB is sharding.

Sharding is the process of distributing data across multiple servers, allowing for horizontal scaling. In a sharded MongoDB cluster, data is split into smaller chunks, called shards, and each shard is stored on a separate server. When a query is executed, the query coordinator determines which shard(s) contain the required data and forwards the query to those shards. The results are then combined and returned to the client.

Sharding in MongoDB can be achieved by sharding collections, which means that the data in a collection is split across multiple shards based on the shard key. The shard key can be any field or combination of fields in a document, and it is used to determine which shard a document should belong to.

Scaling MongoDB with sharding provides several benefits, including:

1. Increased storage capacity: By adding more nodes to the cluster, more data can be stored.
2. Improved query performance: Queries are distributed across multiple servers, reducing the load on any single server and improving overall performance.