MongoDB Interview Questions

- What is MongoDB?

  Explanation: MongoDB is a NoSQL database that stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

- What are Collections in MongoDB?

  Explanation: Collections in MongoDB are analogous to tables in relational databases and are used to store a set of documents.

- What is a Document in MongoDB?

  Explanation: A document is the basic unit of data in MongoDB and is similar to a JSON object, consisting of field-value pairs.

- How does MongoDB differ from SQL databases?

  Explanation: MongoDB is a NoSQL database that does not require a fixed schema, allows horizontal scaling, and uses a document-based data model, unlike structured, table-based SQL databases.

- What is the role of _id in MongoDB?

  Explanation: The _id field acts as a primary key in MongoDB documents, uniquely identifying each document in a collection.

- What are Indexes in MongoDB?

  Explanation: Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must scan every document in a collection to select those that match the query statement.

- Can you change an _id field of a document?

  Explanation: No, the _id field of a document is immutable and cannot be changed once set.

- What is a Replica Set in MongoDB?

  Explanation: A replica set in MongoDB is a group of mongod instances that maintain the same data set, providing redundancy and high availability.

- What is Sharding in MongoDB?

  Explanation: Sharding is the process of storing data records across multiple machines and is MongoDB's approach to meeting the demands of data growth.

- What are Aggregations in MongoDB?

  Explanation: Aggregations in MongoDB process data records and return computed results, similar to GROUP BY in SQL. They provide a way to perform complex data processing and transformations.

- How do you back up a MongoDB database?

  Explanation: MongoDB can be backed up using mongodump, a utility for creating binary export of the contents of a database.

- What is BSON in MongoDB?

  Explanation: BSON (Binary JSON) is a binary-encoded serialization of JSON-like documents used by MongoDB.

- What is a Namespace in MongoDB?

  Explanation: A namespace in MongoDB is the concatenation of the database name and the collection name, used to uniquely identify collections across databases.

- What is Mongoose in the context of MongoDB?

  Explanation: Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, managing relationships between data and providing schema validation.

- How does MongoDB provide concurrency?

  Explanation: MongoDB uses reader-writer locks that allow concurrent readers shared access to a resource, such as a database or collection, but give exclusive access to a single write operation.

- What are some common commands in MongoDB?

  Explanation: Common commands include find() for retrieving documents, insert() for adding new documents, update() for modifying existing documents, and delete() for removing documents.

- What is Journaling in MongoDB?

  Explanation: Journaling in MongoDB is used to safeguard data in case of a crash by recording changes before they are written to the database.

- What is GridFS and when is it used?

  Explanation: GridFS is used in MongoDB for storing and retrieving large files like images, audio files, or video files.

- How do you scale MongoDB?

  Explanation: MongoDB can be scaled horizontally by sharding, distributing data across multiple servers, or vertically by adding more resources to the existing machines.

- What is the default port for MongoDB?

  Explanation: The default port for MongoDB is 27017

- How does MongoDB handle transaction management?

  Explanation: MongoDB supports multi-document ACID transactions, similar to relational databases. Transactions in MongoDB can be used to perform a series of read and write operations atomically.

- Explain the concept of 'upsert' in MongoDB.

Explanation: 'Upsert' is a combination of 'update' and 'insert'. If the specified document exists, MongoDB updates it with the new values; if it does not exist, MongoDB inserts it as a new document.

- What are the differences between embedded documents and references in MongoDB?

  Explanation: Embedded documents are stored directly within a parent document, providing fast read access. References are links to documents stored in another collection, requiring an additional query to retrieve but are better for data normalization and avoiding data duplication.

- How do you ensure data integrity in MongoDB?

  Explanation: Data integrity in MongoDB can be ensured through proper schema design, using transactions for complex operations, and implementing validation rules in the database layer.

- Scenario: How would you design a MongoDB schema for a blogging platform?

  Explanation: A blogging platform schema might involve collections for users, posts, and comments. Posts can have embedded comments or reference them. User documents can reference their posts.

- What is MapReduce in MongoDB and when would you use it?

  Explanation: MapReduce is a data processing paradigm in MongoDB used for batch processing of data and aggregation operations. It's useful for large datasets and complex aggregations.

- How does MongoDB ensure high availability?

  Explanation: MongoDB ensures high availability through replica sets, which provide redundancy and automatic failover in case of primary node failure.

- Can you change the shard key after sharding a collection?

Explanation: No, once a shard key is chosen and sharding is implemented, you cannot change the shard key of a collection.

- What is a Covered Query in MongoDB?

  Explanation: A covered query is a query in which all the fields in the query, including the sort and projection fields, are part of an index. Covered queries can be much faster as they avoid fetching documents.

- Explain Write Concern in MongoDB.

  Explanation: Write concern in MongoDB describes the level of acknowledgment requested from MongoDB for write operations, determining the guarantee of writing data to the database.

- What is the Aggregation Pipeline in MongoDB?

  Explanation: The Aggregation Pipeline is a framework in MongoDB for data aggregation, modeled as a pipeline through which documents pass and are transformed into aggregated results.

- Scenario: How would you optimize a slow query in MongoDB?

  Explanation: To optimize a slow query, first identify the query, examine the execution plan, create appropriate indexes, and consider redesigning the schema for more efficient querying.

- What is the role of the Profiler in MongoDB?

  Explanation: The Profiler in MongoDB is used to monitor database operations, helping in identifying slow queries and performance bottlenecks.

- How can you achieve pagination in MongoDB queries?

  Explanation: Pagination can be achieved using the skip() and limit() methods in MongoDB. However, for large datasets, a range-based pagination using _id or another indexed field is more efficient.

- Explain the role of the WiredTiger storage engine in MongoDB.

Explanation: WiredTiger, the default storage engine in MongoDB, offers advantages like support for transactions, compression, and cache management, leading to improved performance and storage efficiency.

- Scenario: How would you handle a scenario where your MongoDB database is hitting memory limits?

  Explanation: Addressing memory limits involves optimizing indexes, queries, and schema design; considering sharding for horizontal scaling; and potentially increasing the server's memory capacity.

- What are TTL Indexes in MongoDB?

  Explanation: Time-To-Live (TTL) indexes are used to automatically remove documents from a collection after a certain amount of time, useful for data that only needs to be stored temporarily.

- What is the difference between $lookup and DBRef in MongoDB?

  Explanation: $lookup is an aggregation pipeline stage that lets you join documents from different collections, while DBRef is a convention for representing a reference to another document.

- Scenario: Describe how you would migrate data from a SQL database to MongoDB.

  Explanation: Migrating data involves exporting data from the SQL database, transforming it into a format suitable for MongoDB (like JSON), and then importing it using tools like mongoimport.

- What is a Capped Collection in MongoDB?

  Explanation: Capped collections are fixed-size collections that automatically overwrite their oldest entries when they reach their maximum size. They are ideal for logging and caching purposes.

- Explain the use of the $facet stage in the aggregation pipeline.

Explanation: The $facet stage allows for performing multiple aggregation pipelines within a single stage. It is useful for creating multi-faceted aggregations that categorize data into different metrics in a single query.

- How does MongoDB handle large-scale join operations?

  Explanation: MongoDB isn't designed for large-scale join operations like traditional RDBMS. However, it can perform left outer joins using the $lookup stage in the aggregation pipeline.

- What are the limitations of using transactions in MongoDB?

  Explanation: Transactions in MongoDB can affect performance due to increased latency and have limitations like a 60-second transaction timeout and increased storage space requirements for the WiredTiger engine.

- Scenario: Design a MongoDB schema for a real-time analytics dashboard.

  Explanation: For a real-time analytics dashboard, a schema that supports fast reads is crucial. This might involve denormalizing data, using pre-aggregated metrics, and optimizing indexes for common queries.

- Explain the role of oplog in MongoDB Replica Sets.

  Explanation: The oplog (operations log) is a special capped collection that keeps a rolling record of all operations that modify the data stored in databases. It's used in replica sets for replication purposes.

- How does MongoDB ensure data durability?

  Explanation: MongoDB ensures data durability through journaling and replication. Journaling writes operations to disk to prevent data loss, and replication ensures data is copied across multiple servers.

- What are the strategies for handling hotspots in sharded clusters?

Explanation: Handling hotspots involves choosing an effective shard key, ensuring even data distribution, using compound shard keys if necessary, and monitoring and redistributing chunks when needed.

- Scenario: How would you optimize a sharded MongoDB cluster with uneven shard loads?

  Explanation: To optimize an unevenly loaded sharded cluster, analyze the shard key and chunk distribution, use zone sharding for better control, and possibly reshard the data with a more appropriate shard key.

- Discuss the impact of document size on MongoDB's performance.

  Explanation: Larger documents consume more memory and can lead to slower read/write operations. Optimizing document size by avoiding large, complex documents or unnecessary fields can improve performance.

- How does MongoDB handle network partitions in a sharded environment?

  Explanation: In the event of a network partition, MongoDB maintains consistency by ensuring that each shard remains independently consistent. However, the cluster might not be able to achieve overall consistency until the partition is resolved.

- Explain how the WiredTiger cache works in MongoDB.

  Explanation: The WiredTiger cache holds recently read and written data in memory. When the cache becomes full, older or less frequently accessed data is written to disk. Proper cache management is crucial for performance.

- What are the best practices for securing a MongoDB database?

  Explanation: Best practices include enabling authentication, using role-based access control, encrypting data at rest and in transit, regularly updating MongoDB, and securing the underlying server.

- Scenario: Implement a strategy to handle time-series data in MongoDB.

  Explanation: For time-series data, use a schema that groups data into buckets (documents) based on time intervals. This approach optimizes storage and query efficiency for time-based data.

- Discuss the use of Change Streams in MongoDB.

  Explanation: Change streams allow applications to access real-time data changes without polling the database. They are useful for triggering actions, updating caches, and synchronizing data with external systems.

- Explain how MongoDB’s query planner selects indexes for executing queries.

  Explanation: MongoDB's query planner evaluates various query plans using available indexes and chooses the one with the lowest estimated cost based on factors like index selectivity and document counts.

- What are the considerations for selecting a shard key in MongoDB?

  Explanation: A good shard key should provide even distribution of data, support the query patterns of your application, and minimize the need for resharding.

- Scenario: How do you migrate a large collection in MongoDB with minimal downtime?

  Explanation: For minimal downtime, perform the migration in stages: start by syncing data to the new collection or database, redirect read/write traffic, and then finalize the migration.

- How does MongoDB handle write operations in sharded clusters?

  Explanation: Write operations in sharded clusters are directed to the primary shard responsible for the data's shard key. The primary shard then applies the write locally and replicates it to secondary members.

- What is a Write Amplification in MongoDB and how can it be minimized?

Explanation: Write amplification occurs when more writes are performed than necessary, often due to updates that cause document relocations. It can be minimized by schema design that reduces document growth and by using update operators efficiently.

- How do you monitor and tune the performance of a MongoDB cluster?

  Explanation: Performance can be monitored using tools like MongoDB Atlas, mongostat, and mongotop. Tuning involves analyzing query patterns, optimizing indexes, adjusting server configurations, and ensuring adequate resources.