Comprehension Questions

NoSQL Concepts: Documents vs Tables, Collections, BSON

1. What are the three core building blocks of MongoDB's data model, and how do they relate to each other?1
2. Describe the analogy used to explain databases, collections, and documents in MongoDB.
3. How does a MongoDB collection differ from a relational database table in terms of schema requirements?
4. What is BSON, and why does MongoDB use it instead of plain JSON for storing documents?
5. What is the purpose of the `_id` field in MongoDB documents, and what happens if you do not specify it when inserting a document?
6. List two naming restrictions for MongoDB database names and two for collection names.
7. What is the maximum size allowed for a single MongoDB document, and what should you use if you need to store larger data?
8. How does MongoDB's flexible schema benefit applications with rapidly changing data requirements?
9. Compare the mapping of SQL terms (database, table, row, column) to MongoDB terms.
10. What BSON type should you use for storing high-precision numbers such as financial data, and why?

Schema Design: Embedding vs Referencing, Flexible Schema, Validation Rules

11. In what scenarios is embedding documents preferred over referencing in MongoDB schema design?
12. What are two main advantages and two challenges of using embedded documents?
13. When is referencing between documents more appropriate than embedding?
14. Explain the difference between atomic operations in embedded vs referenced document models.
15. How does MongoDB's flexible schema model differ from traditional relational databases in handling evolving data structures?
16. What is the purpose of schema validation rules in MongoDB, and how are they implemented?
17. Give an example use case where a hybrid approach (embedding and referencing) would be beneficial.
18. Why is it important to plan your schema based on access patterns and data size?
19. What is a common pitfall of over-embedding data, and what is a pitfall of over-referencing data in MongoDB?
20. How does MongoDB ensure atomicity for write operations on embedded documents?

Indexes: Single Field, Compound, TTL, Text Indexes

21. What is the main purpose of an index in MongoDB?
22. How do single field and compound indexes differ in their structure and use cases?
23. What is a TTL index, and what type of data is it commonly used for?
24. Why does the order of fields matter in a compound index?

25. How does a text index differ from other index types, and what kind of queries does it support?

26. What is the effect of creating too many indexes on a MongoDB collection?

27. How can you view all indexes on a collection, and how do you remove a specific index?

28. What is a unique index, and how does it help maintain data integrity?

29. Describe a scenario where you would use a partial TTL index.

30. What are two best practices for index creation and maintenance in MongoDB?