# Conceitos de Bancos de Dados e Sistemas Distribuídos

1. De acordo com o Teorema CAP de Brewer, um sistema distribuído não pode suportar simultaneamente as três características: Consistência, Disponibilidade e Tolerância à Partição.

2. As bases de dados NoSQL são preferencialmente utilizadas em relações de entidades 'muitos para muitos'.

3. A normalização em sistemas de bases de dados visa reduzir a redundância dos dados.

4. Uma estrutura de dados imutável favorece as escritas aleatórias, pois os dados podem ser gravados em qualquer posição no arquivo.

5. As bases de dados NoSQL distribuídas são 'não relacionais' e não oferecem propriedades 'ACID'.

6. As Data Warehouses estão geralmente associadas ao padrão OLTP.

7. Quando criamos índices, aumentamos a eficiência das consultas, embora isso possa reduzir o espaço geral de armazenamento.

8. 'Append-only log' é uma estrutura de dados imutável (ou seja, não substituímos as entradas anteriores) que armazena dados em disco em ordem sequencial.

9. No armazenamento tipo 'Append-only log', os pares chave-valor são armazenados de forma ordenada pela chave.

10. Write-Ahead Logs (WAL) são arquivos do tipo 'append only' que ajudam a restaurar uma B-Tree para um estado consistente em caso de perda de dados em memória.

11. Qual frase melhor caracteriza um sistema OLTP (Online Transactional Processing)?

12. Qual frase melhor caracteriza um sistema OLAP (Online Analytical Processing)?

13. Em comparação com a codificação binária, os formatos de texto (por exemplo, JSON, TXT) são menos compactos, mas mais fáceis de processar.

14. Protocol Buffers, RDF e BSON são exemplos de formatos binários.

15. O formato BSON é mais apropriado para representar objetos do tipo imagem do que o formato JSON.

16. No Redis, um comando ZRANGEBYSCORE retorna os elementos na estrutura Hash que estão dentro de um determinado intervalo de pontuação (score).

17. Em Redis, podemos definir um tempo de expiração de uma entrada chave-valor, após o qual ela será removida automaticamente da base de dados.

18. O Redis permite pesquisa por chave e pelo valor da chave.

19. As bases de dados do tipo chave-valor fornecem altas taxas de inserção de dados, favorecendo aplicações de escrita única e leitura múltipla.

20. Em Redis, o comando 'KEYS' permite obter todas as entradas cuja chave começa por 'cbd'.

21. Em Redis, o comando 'SET key value' permite inserir um novo par chave-valor ou substituir o valor de uma chave já existente.

22. As bases de dados do tipo chave-valor são eficazes para representar relacionamentos entre entidades.

23. Uma base de dados orientada a documentos é recomendada quando a estrutura do(s) documento(s) muda constantemente.

24. Em Redis, podemos definir um tempo de expiração para uma entrada chave-valor, após o qual será removida automaticamente da base de dados.

25. Numa base de dados orientada a documentos, é necessário ler sempre um documento para acessar um de seus atributos.

26. Em MongoDB, o comando de shell:

$ mongo -username user -password pass -host host -port 28015

executa o servidor MongoDB, permitindo definir um usuário, senha, hostname e porta.

27. Em MongoDB, o comando de shell:

db.inventory.update({"item":"journal"}, {$set: {"status": "B"}})

permite modificar um documento ou criá-lo caso este não exista.

28. Numa base de dados orientada a documentos, é necessário ler sempre um documento para acessar um de seus atributos.

29. 'Single Field', 'Compound Index' e 'Multikey' são exemplos de tipos de índice no MongoDB.

30. O MongoDB oferece suporte a documentos embutidos em arrays (ou seja, um documento pode conter um array de documentos).

31. A atualização de um atributo num documento em MongoDB implica a reescrita de todo o documento.

32. MongoDB, Couchbase e Riak são exemplos de bases de dados orientadas a documentos.

33. Uma base de dados orientada a documentos está otimizada para alterações simultâneas de múltiplos documentos.

34. As bases de dados orientadas a colunas são adequadas para consultas de agregação em grandes volumes de dados.

35. As bases de dados orientadas a colunas não garantem propriedades ACID em todas as operações.

36. As bases de dados orientadas a colunas são adequadas para carregamento incremental de dados.

37. Em Cassandra, os índices são locais (ou seja, por partição).

38. Em Cassandra, na cláusula WHERE, aplicam-se diferentes restrições na chave de partição, nos atributos de clustering e nas demais colunas.

39. Em Cassandra, a organização dos dados dentro de cada partição segue um mapa multidimensional ordenado.

40. Em uma consulta no Cassandra, é possível aplicar restrições nas 'clustering columns' utilizando 'range queries' (<, ≤, >, ≥) e 'exact match' (=).

41. Em Cassandra, os dados estão particionados em um anel de nós, cada um suportando um intervalo de tokens (ou seja, hash da PartitionKey).