INTRODUÇÃO A MongoDB e BANCO DE DADOS NoSQL

1998 - 1º aparição do termo NoSQL
Reapareceu em 2009

NoSQL - Not Only SQL (Não somente SQL)

Os BD NoSQL surgiram para suprir as deficiências do BD's relacionais.
Hoje em dia é muito comum utilizar os dois tipos de BD.

Diferenças

Escalabilidade

Relacional - vertical (infla o hardware para ele ser capaz de suprir as necessidades)
NoSQL - horizontais (particionando os dados) - escalabilidade infinita, principalmente se estiver na nuvem

Schema

Relacional - precisa moldar a estrutura antes - esquema rígido, maior consistência
NoSQL - não precisa pré-definir quais dados o BD vai receber. Menos garantia de consistência.

Performance

Relacional - depende diretamente do subsistema de disco
NoSQL - depende do tamanho cluster e da latência da rede

Transações

Relacional - ACID: Atomicidade, Consistência, Isolamento, Durabilidade

Atomicidade- ou a transação é executada por completo ou ela não é executada

Consistência- quando a transação for concluída o BD estará exatamente em conformidade com os Schemas pré definidos.

Isolamento - uma transação nunca vai interferir em outra

Durabilidade - uma vez que a transação foi concluída ela não será perdida

NoSQL - BASE: Basically Available, Soft-State, Eventually Consistent

Características e Principais vantagens NoSQL
FLEXIBILIDADE, ESCALABILIDADE E ALTA PERFORMANCE

TIPOS DE BANCO NoSQL

Orientado a Documentos - mais utilizado entre o NoSQL
Auto contido e Auto descritivo
Todas as informações que ele precisa estão contidas dentro dele
Permite redundância e incoerência

Orientado a chave-valor (key-value) - Constituído por 2 partes: uma é a chave (única) e a outra é o valor
Dentro dos valores pode colocar listas de tipos, numérico, string, json
Tem um bom desempenho em aplicação na nuvem, mas capacidade de busca limitada.
São muito utilizados para carrinho de compra

BD mais utilizado é o Redis = exemplo de uso github, stackoverflow
Não existe definição de schema para chave-valor

Orientado a Colunas - (Cassandra utilizado por facebook, twitter) é o que menos apresenta diferenças em relação aos bancos de dados relacionais. Linguagem CQL
Key space - família de colunas - informações armazenadas em colunas

- Ter um volume muito maior de leitura do que de escrita
- Tem queda de performance quando consulta por coluna que não é a key

Keyspace = análogo ao database
Família colunas = análoga tabela
Row key = primary key
Column = nome, valor e timestamp

Registro de transações: ex - código rastreio correio

Cassandra = CREATE KEYSPACE nome WITH

Orientado a Grafos ex: Neo4j (usado muito em redes sociais)- 'nós' compõe nossos dados e os vértices compõe os nossos relacionamentos.
Neo4j - aplica ACID e concorrência
Nada precisa ser pré-definido, um cadastro pode ter informações de idade por exemplo e o outro não.
Linguagem consulta Cypher

Criando um nó - CREATE (:cliente {nome, idade, hobbies})
Consuta - MATCH (variavel) RETURN nome_variavel; MATCH (todos) RETRUN todos;
Criar nó com relacionamento - CREATE (:cliente {nome, idade, hobbies}) - [a abertura e fechamento de colchetes define o relacionamento] → seta é a direção do relacionamento
DELETE - deletar nó ou relacionamento
Atualizar dado no nó - MATCH (patrick:Client {aak} SET dados para inserir
Pode alterar a label, remover ou criar sem label (cada lable tem uma cor)

INTRODUÇÃO AO MongoBD

Ele é de código aberto, alta performance, schema-free, utiliza Json para armazenamento de dados.

- Tem um suporte a índices, auto-sharding (nativo com escalamento horizontal)
- Map-reduce (ferramenta de consulta e agregação)
- GridFS (armazenamento de arquivos)
- Tem uma rica linguagem de consulta

Estruturação do BD MongoDB

Documento (menor unidade do BD) identifica o registro, precisa ser auto-contido e auto-descritivo.
Os seus dados não podem depender de outros dados

Collection análoga a tabela, schema-free, pode armazenar vários tipos de documentos

Embedding/linking associa-se muito a questões Join. Dentro do documento você tem a subestrutura dele lá dentro mesmo.
Ex: todas as informações do usuário estarão no documento e seu endereço também

Quando usa MongoDB - indicado para grande volume de dados
dados não necessariamente estruturados

Quando não utilizar MongoDB - quando há necessidade de relacionamento/joins
não é aconselhável para ACID e transações

Grandes empresas usam o MongoDB- exemplo Linkedin

docker - o que é? Ele é plataforma de código aberto, de alto desempenho
Crie, teste e implemente em um ambiente separado

SCHEMA DESIGN

2 formas de modelar os dados: Embedding vs Referência

Embedding: tudo está contido lá dentro, sem referência para outra collection
Pros: é o mais aconselhado. Consulta informações em uma única query
Atualiza o registro em uma única operação
Contra: Limite 16MB por documento

Referência: Pros- Documentos pequenos
Não duplica informações
Deve ser utilizada quando em alguns casos você precisa da informação
Contras- Duas ou mais queries ou utilização do lookup para ver as referências

One-to-one: prefira atributos chave-valor. Não precisa criar sub documento.
One-to-few: prefira embedding
One-to-many ou many-to-many: aconselhável utilizar relacionamento por referência

No MongoDB é possível executar queries para referenciar outros documentos

BOAS PRÁTICAS

- Evite documentos muito grandes: quanto maior o documento maior o nosso overhead na consulta
- Use campos objetivos e curtos: pois além dos dados e schema tbm é armazenado
- Analise as suas queries utilizando explain() - se a query for mais complicada é importante analisá-la
- Atualize apenas os campos específicos, somente as alterações relevantes e não o Json completo
- Evite fazer queries com negações - o Mongo não indexa valores e condições negativas
- Listas/Arrays não podem crescer sem limites

JSON x BSON

Bson - forma em que o MongoDB armazena
A estrutura é parecida com o Json
Permite a representação de tipos de dados, por exemplo Date, ObjectID

Json - lida apenas com dados primitivos

Alguns comandos

db.createcollection

dbtest

inserir documentos na collection:
db.clients.insert([{“name” : “Patrick”}])

realiza atualização do doc por completo ou inserção
db.clients.save({

db.clients.find - verificar os documentos que existem
db.clients.find({}).limit - para limitar qtos resultados

Update atualiza os docs que fizerem match com a query
db.clients.update({informação existente}), {$set : informação a inserir}

Existe também o comando updateMany - para atualizar todos os docs que tenham tal informação (a mesma) para a nova informação.

OPERADORES IN, AND, OR

DELETEOne
DELETEMany

AGREGAÇÕES

Procedimento de processar dados em um ou mais etapas, com base nessas etapas o resultado da etapa anterior será usado na nova etapa com o objetivo de ter o resultado combinado de vários processos.

2 tipos de agregações:
Pipeline - É possível realizar várias customizações.
Compostos por filtros e operadores ($group (agrupar dados, max, min), $addfields(adiciona para o resultado um novo campo)

Funções - $sum, $avg..
Operadores lógicos - $and, $not, $or e de Comparação - gt (>), gte (>=), lt (<), nte (diferente), eq (=), lte(<=)

para adicionar a agregação em campos específicos (aplicar filtros nos documentos) $match

Propósito único - COUNT e DISTINCT - não aplica customizações