**Aula03 - Modelagem de Dados Não Estruturados (Parte 2)**

**Criando e Selecionando um Banco de Dados**

*Para criar e selecionar um banco de dados, use o comando use.*



**Inserindo Documentos**

Vamos inserir alguns documentos em uma coleção chamada clientes.

Texto, Carta

Descrição gerada automaticamente

**Consultando Documentos**

Vamos buscar documentos na coleção clientes.

Texto

Descrição gerada automaticamente

**Atualizando Documentos**

Vamos atualizar documentos na coleção clientes.

,Texto, Carta

Descrição gerada automaticamente

**Removendo Documentos**

Vamos remover documentos da coleção clientes.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Indexação**

Vamos criar índices para melhorar a performance das consultas.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Agregação**

Vamos usar o framework de agregação para realizar operações avançadas.

Um pipeline de agregação é composto por uma série de estágios, onde cada estágio é representado por um operador de agregação. Os documentos passam por cada estágio do pipeline, sendo processados e transformados conforme necessário.

**Exemplo de Estrutura de Pipeline:**

Texto

Descrição gerada automaticamente

**Estágios Comuns no Pipeline**

Aqui estão alguns dos estágios mais comuns usados em um pipeline de agregação:

* **$match**: Filtra documentos com base em critérios específicos, semelhante a uma cláusula WHERE em SQL.
* **$group**: Agrupa documentos com base em um ou mais campos e realiza operações de agregação, como soma, média, contagem, etc.
* **$sort**: Ordena os documentos com base em um ou mais campos.
* **$project**: Modifica a estrutura dos documentos, incluindo ou excluindo campos, ou criando novos campos.
* **$limit**: Limita o número de documentos passados para o próximo estágio.
* **$skip**: Pula um número especificado de documentos.
* **$unwind**: Desestrutura arrays, criando um documento separado para cada elemento do array.

Texto

Descrição gerada automaticamente

Vamos analisar o pipeline de agregação do exemplo acima:

**Estágio $match**

****

O estágio $match é usado para filtrar documentos com base em critérios específicos. Neste exemplo, estamos filtrando os documentos onde o campo status é igual a "ativo".

* **Objetivo**: Selecionar apenas os documentos que têm o status "ativo".
* **Resultado**: Apenas os documentos que atendem a esse critério serão passados para o próximo estágio do pipeline.

**Estágio $group**

****

O estágio $group é usado para agrupar documentos com base em um ou mais campos e realizar operações de agregação, como soma, média, contagem, etc. Neste exemplo, estamos agrupando os documentos pelo campo idade e contando o número de documentos em cada grupo.

* **Objetivo**: Agrupar os documentos pelo campo idade.
* **\_id**: O campo \_id define o campo pelo qual os documentos serão agrupados. Neste caso, estamos agrupando por idade.
* **total**: Estamos criando um novo campo chamado total que conta o número de documentos em cada grupo usando $sum: 1.

**Estágio $sort**

****

O estágio $sort é usado para ordenar os documentos com base em um ou mais campos. Neste exemplo, estamos ordenando os grupos pelo campo total em ordem decrescente.

* **Objetivo:** Ordenar os grupos pelo campo total em ordem decrescente.
* **total: -1:** O valor -1 indica que a ordenação será em ordem decrescente. Se fosse 1, seria em ordem crescente.

**Resultado Final**

O pipeline de agregação completo realiza as seguintes operações:

1. **Filtragem**: Seleciona apenas os documentos com status igual a "ativo".
2. **Agrupamento**: Agrupa os documentos pelo campo idade e conta o número de documentos em cada grupo.
3. **Ordenação**: Ordena os grupos pelo campo total em ordem decrescente.

**Exemplo de Resultado:**

**Texto

Descrição gerada automaticamente com confiança média**

Neste exemplo, o resultado mostra que há 5 clientes com 30 anos, 3 clientes com 25 anos e 2 clientes com 28 anos, todos com status "ativo".

**Vantagens do Uso de Pipeline**

* **Modularidade**: Cada estágio do pipeline realiza uma operação específica, tornando a consulta mais modular e fácil de entender.
* **Flexibilidade**: Permite combinar múltiplas operações de processamento de dados em uma única consulta.
* **Eficiência**: Processa os dados de maneira eficiente, passando os resultados de um estágio para o próximo sem a necessidade de armazenar resultados intermediários.

**Replicação**

**O que é Replicação?**

A replicação de banco de dados é o processo de criar e manter múltiplas cópias de um banco de dados em diferentes servidores ou locais. O principal objetivo da replicação é melhorar a disponibilidade dos dados, distribuir a carga de trabalho e aumentar a tolerância a falhas

1.**Tipos de Replicação**

1. **Replicação Síncrona**:
   * **Definição**: As alterações feitas no banco de dados primário são imediatamente replicadas para os servidores secundários.
   * **Vantagens**: Garante consistência dos dados entre os servidores.
   * **Desvantagens**: Pode introduzir latência, pois a operação só é confirmada após a replicação.
2. **Replicação Assíncrona**:
   * **Definição**: As alterações são replicadas para os servidores secundários em um momento posterior.
   * **Vantagens**: Menor latência, pois a operação é confirmada imediatamente.
   * **Desvantagens**: Pode haver um pequeno atraso na consistência dos dados entre os servidores.

**Benefícios da Replicação**

* **Alta Disponibilidade**: Se um servidor falhar, outro servidor com a réplica dos dados pode assumir.
* **Desempenho**: Distribui a carga de leitura entre múltiplos servidores.
* **Tolerância a Falhas**: Redundância de dados para recuperação em caso de falhas

2. **Sharding**

**O que é Sharding?**

Sharding é a técnica de dividir um grande banco de dados em partes menores, chamadas shards, que são distribuídas em múltiplos servidores. Cada shard contém uma fração dos dados totais, permitindo que o sistema gerencie grandes volumes de dados de forma mais eficiente

3.**Como Funciona o Sharding?**

* **Divisão Horizontal**: Os dados são divididos horizontalmente em shards, cada um armazenando um subconjunto das linhas do banco de dados.
* **Distribuição**: Cada shard é armazenado em um servidor diferente, permitindo a distribuição da carga de trabalho.

**Benefícios do Sharding**

* **Escalabilidade Horizontal**: Permite adicionar mais servidores para lidar com o aumento de dados.
* **Desempenho**: Melhora a performance ao distribuir a carga de trabalho entre múltiplos servidores.
* **Resiliência**: Reduz o risco de falhas catastróficas, pois a falha de um shard não afeta os outros

4.**Exemplo de Sharding**

Imagine um banco de dados de uma rede social com milhões de usuários. Em vez de armazenar todos os dados em um único servidor, os dados podem ser divididos em shards com base em critérios como a localização geográfica dos usuários. Por exemplo:

* **Shard 1**: Usuários da América do Norte.
* **Shard 2**: Usuários da Europa.
* **Shard 3**: Usuários da Ásia.