Consultas e Operações - DataDriven Store

Vinicius Carvalho Miranda

July 5, 2025

1 Introdução

Este documento apresenta um conjunto de consultas e operações para os bancos de dados utilizados na arquitetura da DataDriven Store. Cada banco de dados é utilizado para um propósito específico, aproveitando suas características únicas. As seções a seguir detalham cinco consultas ou operações para cada banco, incluindo PostgreSQL (relacional), MongoDB (NoSQL), Neo4j (grafos), ClickHouse (analítico) e Redis (cache).

2 Docker & Populando dados de exemplo

Utilizando o Docker, podemos abrir o arquivo docker-compose.yml para verificar os serviços que serão iniciados com o comando. Abaixo estão os comandos necessários, que também incluem o processo de popular os dados de exemplo.

2.1 Inicializando o Docker e populando tabelas

```
docker compose up -d
```

Uso: Nesta parte, ele irá popular quase todas as tabelas, exceto o Neo4j, que precisaremos executar manualmente.

2.2 Populando NEO4J

```
docker exec -it datadriven_neo4j cypher-shell -u neo4j -p admin123
    -a bolt://localhost:7687 -d neo4j -f /var/lib/neo4j/import/
   neo4j_schema.cypher
```

3 PostgreSQL - Banco Relacional (OLTP)

O PostgreSQL é usado para gerenciar dados transacionais estruturados, como clientes, pedidos e produtos.

3.1 Consulta 1: Clientes com pedidos acima de um valor

```sql
SELECT c.nome, c.email, p.id AS pedido_id, p.valor_total
FROM clientes c
JOIN pedidos p ON c.id = p.cliente_id
WHERE p.valor_total > 1000
ORDER BY p.valor_total DESC;
```

Uso: Identifica clientes que realizaram compras de alto valor.

3.2 Consulta 2: Produtos com estoque baixo

```sql
SELECT codigo_produto, nome, estoque, estoque_minimo
FROM produtos
WHERE estoque <= estoque_minimo AND ativo = TRUE
ORDER BY estoque ASC;
```

Uso: Auxilia na gestão de estoque.

3.3 Consulta 3: Vendas por cidade

```sql
SELECT c.cidade, COUNT(p.id) AS total_pedidos, SUM(p.valor_total)
   AS valor_total_vendas
FROM clientes c
JOIN pedidos p ON c.id = p.cliente_id
WHERE p.status = 'FINALIZADO'
GROUP BY c.cidade
ORDER BY valor_total_vendas DESC;
```

Uso: Fornece insights sobre desempenho de vendas por região.

3.4 Consulta 4: Itens mais vendidos

```sql
SELECT pr.nome, pr.categoria, SUM(ip.quantidade) AS
   quantidade_vendida, SUM(ip.subtotal) AS receita_total
FROM itens_pedido ip
JOIN produtos pr ON ip.produto_id = pr.id
JOIN pedidos p ON ip.pedido_id = p.id
WHERE p.data_pedido BETWEEN '2025-06-01' AND '2025-06-30'
GROUP BY pr.nome, pr.categoria
ORDER BY quantidade_vendida DESC
LIMIT 5;
```

Uso: Identifica produtos populares para planejamento.

3.5 Consulta 5: Transações por método de pagamento

```sql
SELECT metodo_pagamento, COUNT(id) AS total_transacoes, SUM(valor)
    AS valor_total
FROM transacoes_financeiras
```

```
WHERE status = 'APROVADO'
GROUP BY metodo_pagamento
ORDER BY valor_total DESC;
```

Uso: Analisa preferências de métodos de pagamento.

4 MongoDB - Banco NoSQL (Documentos)

O MongoDB armazena dados flexíveis, como detalhes de produtos e perfis de usuários.

4.1 Consulta 1: Produtos por categoria e preço

```
db.produtos.find(
    { categoria: "eletrônicos", preco: { $gte: 1000, $lte: 3000 } },
    { nome: 1, preco: 1, atributos: 1, _id: 0 }
).sort({ preco: 1 });
```

Uso: Filtra produtos para catálogos.

4.2 Consulta 2: Produtos bem avaliados

```
db.produtos.find(
    { "avaliacoes.nota": { $gte: 5 } },
    { nome: 1, avaliacoes: 1, _id: 0 }
);
```

Uso: Destaca produtos de alta qualidade.

4.3 Consulta 3: Usuários com preferências específicas

```
db.perfis_usuarios.find(
    { preferencias: "gaming" },
    { cliente_id: 1, preferencias: 1, produtos_favoritos: 1, _id: 0 }
);
```

Uso: Segmenta usuários para campanhas.

4.4 Consulta 4: Nota média de avaliações

```
db.produtos.aggregate([
    { $match: { _id: "NOTEBOOK001" } },
    { $unwind: "$avaliacoes" },
    { $group: { _id: "$_id", media_nota: { $avg: "$avaliacoes.nota" } } }
]);
```

Uso: Avalia a satisfação com um produto.

4.5 Consulta 5: Produtos visualizados recentemente

```
db.perfis_usuarios.find(
    { cliente_id: 1 },
    { historico_navegacao: 1, _id: 0 }
).sort({ "historico_navegacao.timestamp": -1 });
```

Uso: Personaliza a experiência do usuário.

5 Neo4j - Banco de Grafos

O Neo4j modela relacionamentos para sistemas de recomendação.

5.1 Consulta 1: Recomendações baseadas em similaridade

```
MATCH (c1:Cliente {id: 1})-[:SIMILAR_A]->(c2:Cliente)-[:COMPROU
   ]->(p:Produto)
WHERE NOT (c1)-[:COMPROU]->(p)
RETURN p.id, p.nome, COUNT(*) AS recomendacao_score
ORDER BY recomendacao_score DESC
LIMIT 5;
```

Uso: Gera recomendações personalizadas.

5.2 Consulta 2: Produtos mais comprados por categoria

```
MATCH (c:Cliente)-[:COMPROU]->(p:Produto)-[:PERTENCE_A]->(cat:
   Categoria {nome: "eletrônicos"})
RETURN p.nome, COUNT(*) AS total_compras
ORDER BY total_compras DESC
LIMIT 5;
```

Uso: Identifica produtos populares.

5.3 Consulta 3: Clientes que visualizaram sem comprar

```
MATCH (c:Cliente)-[:VISUALIZOU]->(p:Produto {id: "NOTEBOOK001"})
WHERE NOT (c)-[:COMPROU]->(p)
RETURN c.nome, c.email
ORDER BY c.nome;
```

Uso: Remarketing para clientes.

5.4 Consulta 4: Marcas mais populares

```
MATCH (p:Produto)-[:DA_MARCA]->(m:Marca)<-[:DA_MARCA]-(p2:Produto)
   <-[:COMPROU]-(c:Cliente)
RETURN m.nome, COUNT(DISTINCT c) AS total_clientes
ORDER BY total_clientes DESC;
```

Uso: Analisa popularidade de marcas.

5.5 Consulta 5: Caminhos de recomendação

```
MATCH path = (c:Cliente)-[:COMPROU]->(:Produto)-[:PERTENCE_A]->(:
   Categoria)-[:PERTENCE_A*0..1]->(:Categoria)
WHERE c.id = 1
RETURN path
LIMIT 10;
```

Uso: Explora conexões entre categorias.

6 ClickHouse - Banco Analítico (OLAP)

O ClickHouse processa eventos para análises rápidas.

6.1 Consulta 1: Taxa de conversão por produto

```
SELECT
    produto_id,
    COUNT(*) AS total_visualizacoes,
    SUM(adicionou_carrinho) AS total_carrinho,
    SUM(comprou) AS total_compras,
    (SUM(comprou) * 100.0 / COUNT(*)) AS taxa_conversao
FROM funil_conversao
WHERE data >= '2025-06-01'
GROUP BY produto_id
ORDER BY taxa_conversao DESC;
```

Uso: Avalia eficiência de conversão.

6.2 Consulta 2: Eventos por origem de tráfego

```
SELECT
    utm_source,
    COUNT(*) AS total_eventos,
    SUM(CASE WHEN evento_tipo = 'purchase' THEN 1 ELSE 0 END) AS
       total_compras,
    SUM(valor_total) AS receita_total
FROM eventos
WHERE data >= '2025-06-01'
GROUP BY utm_source
ORDER BY receita_total DESC;
```

Uso: Otimiza campanhas de marketing.

6.3 Consulta 3: Produtos mais visualizados

```
SELECT
    produto_id,
    categoria,
```

```
    COUNT(*) AS total_visualizacoes
FROM eventos
WHERE evento_tipo = 'view' AND data BETWEEN '2025-06-01' AND
   '2025-06-30'
GROUP BY produto_id, categoria
ORDER BY total_visualizacoes DESC
LIMIT 10;
```

Uso: Ajusta catálogo com base em popularidade.

6.4 Consulta 4: Comportamento por dispositivo

```
SELECT
    device_type,
    COUNT(DISTINCT session_id) AS sessoes_unicas,
    SUM(CASE WHEN evento_tipo = 'purchase' THEN valor_total ELSE 0
        END) AS receita_total
FROM eventos
WHERE data >= '2025-06-01'
GROUP BY device_type
ORDER BY receita_total DESC;
```

Uso: Otimiza experiência por dispositivo.

6.5 Consulta 5: Buscas populares

```
SELECT
    termo_busca,
    COUNT(*) AS total_buscas,
    AVG(resultados_busca) AS media_resultados
FROM eventos
WHERE evento_tipo = 'search' AND termo_busca != ''
GROUP BY termo_busca
ORDER BY total_buscas DESC
LIMIT 5;
```

Uso: Otimiza motor de busca.

7 Redis - Cache e Sessões

O Redis gerencia sessões, carrinhos e rankings em tempo real.

7.1 Operação 1: Recuperar carrinho

```
HGETALL cart:user_001
```

Uso: Exibe itens no carrinho.

7.2 Operação 2: Produtos mais visualizados

```
ZREVRANGE most_viewed 0 4 WITHSCORES
```

Uso: Destaca produtos populares.

7.3 Operação 3: Verificar sessão

```
GET session:user_001
```

Uso: Valida sessões ativas.

7.4 Operação 4: Resultados de busca em cache

```
GET search:gaming
```

Uso: Reduz latência em buscas.

7.5 Operação 5: Contagem de visualizações

```
INCR views:NOTEBOOK001:2025-06-28
```

Uso: Rastreia popularidade em tempo real.