INSTITUTO SUPERIOR DO LITORAL DO PARANÁ

Prof. Luiz Efigênio

Administração de Banco de Dados II
Visão Geral e Conceitos Avançados de SQL

INSTITUTO SUPERIOR DO LITORAL DO PARANÁ

Disciplina: Administração de Banco de Dados (ADB)
Professor: Luiz Efigênio
Data: 10/11/2025
Tema da Aula: *Performance em Bancos de Dados: Índices e Otimização de Consultas*

---

1. Objetivos da Aula

- Compreender os fatores que influenciam o desempenho em bancos de dados relacionais.
- Entender o funcionamento e os tipos de índices.
- Aprender a analisar e otimizar consultas SQL.
- Identificar boas práticas para garantir eficiência e escalabilidade em ambientes reais.

---

2. Introdução: Por que otimizar?

Em aplicações modernas, a **performance do banco de dados** impacta diretamente a experiência do usuário e o custo de infraestrutura.
Mesmo sistemas robustos podem sofrer lentidão devido a **consultas mal estruturadas**, **ausência de índices**, **joins desnecessários** ou **excesso de leituras de disco**.

A otimização de desempenho é, portanto, uma **etapa contínua** no ciclo de vida do sistema, que deve envolver:

- Planejamento de índices;
- Escrita eficiente de consultas SQL;
- Monitoramento e ajuste periódico conforme o crescimento dos dados.

---

3. Conceitos Fundamentais

3.1 O que é performance em banco de dados

Performance é a **capacidade do SGBD de responder consultas e transações em tempo aceitável**, usando o mínimo possível de recursos (CPU, memória, I/O).

Principais métricas:

- **Tempo de resposta (latência)**: quanto tempo leva para executar a consulta.
- **Throughput (vazão)**: quantidade de transações por segundo.
- **Uso de recursos**: CPU, memória e disco.
- **Planos de execução (execution plan)**: estratégias usadas pelo otimizador do SGBD para resolver a consulta.

---

4. Índices em Bancos de Dados

4.1 Definição

Um **índice** é uma estrutura de dados auxiliar que acelera a busca e filtragem de informações em uma tabela, semelhante ao índice de um livro.

Sem índices, o banco de dados precisa ler **todas as linhas da tabela** (full table scan) para localizar resultados — o que é ineficiente em grandes volumes.

4.2 Estrutura básica

A maioria dos SGBDs utiliza estruturas **árvore-B (B-tree)** ou **árvore-B+**, que organizam as chaves de forma ordenada, permitindo busca binária.

4.3 Tipos de índices

Tipo de Índice	Descrição	Exemplo

Índice Primário (PRIMARY KEY)	Criado automaticamente para a chave primária da tabela.	`PRIMARY KEY (id)`
Índice Secundário (NON-UNIQUE)	Criado em colunas usadas com frequência em filtros (`WHERE`, `JOIN`).	`CREATE INDEX idx_nome ON clientes(nome);`
Índice Único (UNIQUE)	Garante unicidade além de melhorar buscas.	`CREATE UNIQUE INDEX idx_cpf ON clientes(cpf);`
Índice Composto	Usa mais de uma coluna.	`CREATE INDEX idx_pedidos_cliente_data ON pedidos(cliente_id, data);`
Índice de Texto Completo (FULLTEXT)	Otimiza pesquisas por palavras (usado em textos longos).	`CREATE FULLTEXT INDEX ft_artigos ON artigos(conteudo);`
Índice Hash	Usa tabela hash (MySQL Memory, PostgreSQL Hash Index). Mais rápido, mas menos flexível.	—

---

5. Cuidados e Custos dos Índices

Embora melhorem consultas de leitura, **índices têm custo**:

- Aumentam o espaço em disco.

- Tornam operações de escrita (INSERT, UPDATE, DELETE) mais lentas, pois o índice também precisa ser atualizado.

- Podem confundir o otimizador caso sejam criados em excesso.

Boas práticas:

1. Criar índices apenas em colunas frequentemente usadas em filtros, ordenações e junções.

2. Evitar índices em colunas com alta cardinalidade (muitos valores repetidos).

3. Revisar periodicamente índices não utilizados (consultar o plano de execução ou o catálogo do SGBD).

4. Analisar impacto em consultas reais com `EXPLAIN` (MySQL/PostgreSQL).

---

6. Otimização de Consultas SQL

6.1 Planejamento e análise

Antes de otimizar, **é preciso medir**. Ferramentas como:

- `EXPLAIN` ou `EXPLAIN ANALYZE` (PostgreSQL, MySQL)
- `AUTOTRACE` (Oracle)
- `Query Plan Viewer` (SQL Server)

Esses comandos mostram **como o SGBD executa a consulta** — se usa índice, faz scan completo, ou cria tabelas temporárias.

6.2 Práticas recomendadas

Problema	Solução

Consulta lenta com `LIKE '%texto%'`	Prefira `LIKE 'texto%'` ou use índice `FULLTEXT`.
`SELECT *` em tabelas grandes	Especifique colunas necessárias.
Filtros em funções (`WHERE LOWER(nome) = 'joao'`)	Evite funções na cláusula `WHERE`; normalize dados e use índices apropriados.
JOIN sem índice	Crie índices nas chaves de junção.
Subconsultas aninhadas	Considere `JOIN` ou `WITH` (CTE) para clareza e performance.
Ordenação frequente (`ORDER BY`)	Adicione índice para colunas ordenadas.

---

7. Casos práticos

7.1 Exemplo 1 — Uso de índice

```
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    cidade VARCHAR(80)
);
```

```
CREATE INDEX idx_cidade ON clientes(cidade);


-- Consulta otimizada
SELECT nome FROM clientes WHERE cidade = 'Paranaguá';
```

Sem o índice, o banco faria leitura linha a linha; com o índice, localiza rapidamente as chaves correspondentes.

7.2 Exemplo 2 — Otimização de JOIN

```
CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    cliente_id INT,
    valor DECIMAL(10,2),
    FOREIGN KEY (cliente_id) REFERENCES clientes(id)
);


CREATE INDEX idx_cliente_id ON pedidos(cliente_id);


-- Consulta otimizada
SELECT c.nome, SUM(p.valor)
FROM clientes c
JOIN pedidos p ON c.id = p.cliente_id
GROUP BY c.nome;
```

O índice em `cliente_id` permite que o JOIN use busca indexada em vez de varredura completa.

7.3 Exemplo 3 — Uso de EXPLAIN

```
EXPLAIN ANALYZE

SELECT * FROM pedidos WHERE valor > 500;
```

Mostra se o SGBD utilizou o índice ou fez *full scan*.
Se o plano indicar “Seq Scan”, talvez um índice seja necessário.

---

8. Manutenção e Monitoramento de Performance

- **Reindexação periódica:** índices fragmentam-se com o tempo (`REINDEX`, `OPTIMIZE TABLE`).
- **Estatísticas atualizadas:** os otimizadores precisam de informações atualizadas (`ANALYZE` ou `VACUUM ANALYZE` no PostgreSQL).
- **Monitoramento:** use ferramentas nativas (PgAdmin, MySQL Workbench, SQL Profiler).
- **Caching:** resultados frequentemente consultados podem ser armazenados em cache de aplicação (Redis, Memcached).

---

9. Boas Práticas Gerais

1. Planeje índices no **design do banco**.
2. Analise consultas reais (logs, slow queries).
3. Monitore uso de índices com o tempo.
4. Evite redundância — um índice composto pode substituir vários simples.
5. Documente alterações e valide o impacto em produção.
6. Considere *sharding* e *partitioning* para grandes volumes.

10. Atividade de Pesquisa (individual ou dupla)

Tema: *Análise de Índices e Otimização de Consultas em um SGBD real*

Instruções:

1. Escolha um SGBD (MySQL, PostgreSQL, Oracle ou SQL Server).
2. Crie um pequeno conjunto de tabelas com no mínimo 10.000 registros.
3. Execute uma consulta sem índice e registre o tempo e o plano de execução.
4. Crie índices adequados e repita o teste.
5. Compare e descreva os resultados em um breve relatório (2 páginas).
6. Inclua o script SQL e a captura do `EXPLAIN`.
7. Conclua com uma reflexão: "Quando o excesso de índices pode se tornar prejudicial?".

---

11. Referências Bibliográficas

- DATE, C. J. *Introdução a Sistemas de Banco de Dados.* 8ª ed. Rio de Janeiro: Elsevier, 2004.
- ELMASRI, R.; NAVATHE, S. *Sistemas de Banco de Dados.* 7ª ed. Pearson, 2019.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Database System Concepts.* 7th Edition. McGraw-Hill, 2020.
- CORONEL, C.; MORRIS, S.; ROB, P. *Database Systems: Design, Implementation, and Management.* 14th Edition, Cengage, 2023.
- PostgreSQL Documentation — *Indexes and Query Optimization.* (https://www.postgresql.org/docs/current/indexes.html)
- MySQL Reference Manual — *Optimization and Indexing.* (https://dev.mysql.com/doc/refman/8.0/en/optimization.html)

Mensagem Final

> "Mais índices não significam mais performance — significam mais responsabilidade."
> O domínio da otimização em banco de dados exige **equilíbrio**, **análise constante** e **entendimento profundo das consultas**.

Otimizar é uma arte: o desafio é garantir rapidez sem comprometer a integridade nem a escalabilidade.