1. Conceito de Índices

Índices são estruturas auxiliares criadas em colunas de tabelas para acelerar a busca e consulta de dados. Eles funcionam como um índice em um livro, permitindo acesso rápido às linhas, especialmente em tabelas grandes, melhorando muito a performance das consultas.

---

2. Tipos de Índices

- **Índice básico (CREATE INDEX):** permite acelerar buscas, podendo existir valores duplicados.
  Exemplo: índice em coluna de nome para agilizar pesquisas.

- **Índice único (CREATE UNIQUE INDEX):** garante que todos os valores da coluna sejam únicos.
  Exemplo: índice em coluna de CPF ou e-mail para garantir unicidade e busca rápida.

---

3. Índices Compostos

Índice composto é criado em mais de uma coluna simultaneamente. Ele é útil quando consultas filtram por múltiplas colunas, acelerando buscas que envolvem todas as colunas do índice. Exemplo: índice em `(cidade, estado)` para consultas que filtram ambas.

---

4. Impacto dos Índices em Operações DML

Embora índices acelerem consultas, eles podem desacelerar operações de inserção (`INSERT`), atualização (`UPDATE`) e exclusão (`DELETE`), pois o índice precisa ser mantido atualizado sempre que dados são alterados.

---

5. EXPLAIN em Consultas SQL

O comando `EXPLAIN` mostra o plano de execução da consulta, informando como o MySQL vai buscar os dados, quais índices serão usados, e o custo estimado. Isso ajuda a identificar gargalos e otimizar a consulta.

---

6. Full Table Scan

É quando o banco lê todas as linhas da tabela para responder uma consulta. Pode ser muito custoso e lento em tabelas grandes. Para evitar, deve-se usar índices adequados e filtros eficientes.

---

7. Limitação de Registros em Consultas

Usar `LIMIT` restringe a quantidade de registros retornados, reduzindo processamento e tempo de resposta.
Exemplo:

SELECT * FROM produtos ORDER BY preco DESC LIMIT 10;

Retorna apenas os 10 produtos mais caros, agilizando a consulta.

---

8. Índices e JOINs

Índices nas colunas usadas para relacionar tabelas (`JOIN ON`) melhoram a performance, pois o banco pode localizar rapidamente os registros correspondentes sem varrer toda a tabela.

---

9. Boas Práticas em Consultas SQL

- Use apenas as colunas necessárias no `SELECT` para reduzir dados transferidos.
- Evite `SELECT *`, prefira listar as colunas específicas.
- Use índices nas colunas usadas em `WHERE`, `JOIN` e ordenação (`ORDER BY`).
- Utilize filtros (`WHERE`) para limitar os dados processados.

---

10. Funções de Agregação e Performance

Funções como `COUNT`, `SUM` e `AVG` podem ser custosas em tabelas grandes, pois requerem leitura e processamento de muitos dados. Para otimizar:

- Use índices para acelerar filtros pré-agregação.
- Mantenha dados agregados em tabelas auxiliares quando possível (materialized views).
- Limite o volume de dados usando filtros e partições.