



Boas práticas em SQL



**Certified
Developer**
The Ultimate Tech Degree

DigitalHouse >
Coding School



Índice

1

**Boas práticas em
SQL**

2

CREATE

3

SELECT

4

WHERE

5

UNION

6

CRUD



1 | Boas práticas em SQL



Boas práticas em SQL

As recomendações a seguir têm como objetivo **otimizar os tempos de execução da consulta e escrever consultas SQL robustas**.

Quando não implementamos essas recomendações, SQL executa a análise em todas as tabelas que se está consultando. Se a mesma tiver vários registros, o tempo para retornar o resultado pode prejudicar o software.

Vamos começar!



2

CREATE



Boas práticas CREATE

Tente usar **VARCHAR** em vez de **TEXT**



Se você precisar armazenar grandes volumes de texto, mas eles tiverem menos de 8.000 caracteres, use o tipo de dados **VARCHAR** em vez de **TEXT**.





Boas práticas CREATE

Avalie cuidadosamente o
uso de **CHAR** e **VARCHAR**



Avalie cuidadosamente o uso de **CHAR** e **VARCHAR** dependendo se o tamanho do campo a ser usado varia muito. Isso serve para pesar o desempenho da velocidade em relação ao desempenho do armazenamento. As colunas de tamanho fixo são processadas mais rapidamente pelo mecanismo SQL. Use **CHAR** para colunas de pouca variação de tamanho e **VARCHAR** para aquelas que não têm tamanho estável ou médio.



Boas práticas CREATE

**Tipos de dados em
FOREIGN KEY.**



Não use colunas com tipos de dados FLOAT, REAL ou DATETIME como FOREIGN KEY.

**Use CONSTRAINT para
manter a integridade dos
dados.**





Boas práticas CREATE

**Evite chaves primárias
COMPOSTAS.**



Observe que, se você espera que sua tabela com uma chave primária composta tenha milhões de linhas, o desempenho da operação CRUD será seriamente prejudicado.

Nesse caso, é muito melhor usar uma chave primária de ID simples que tenha um índice compacto o suficiente e defina as restrições necessárias do Mecanismo de Banco de Dados para manter a exclusividade.

3 | SELECT



Boas práticas SELECT

**Evite usar `SELECT *` FROM
tabela.**



Embora seja fácil e conveniente usar o caractere asterisco (*) para buscar todos os campos, ele deve ser omitido e, em vez disso, especifique os campos que precisam ser buscados.

O uso do asterisco também impede o uso eficaz dos índices de forma eficiente.





Boas práticas SELECT

Anexe o **ALIAS** da tabela a cada coluna.



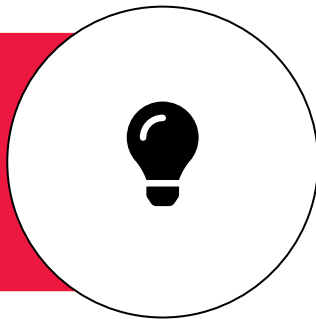
Especifique o **ALIAS** da tabela na frente de cada campo definido no **SELECT**, isso economiza o tempo do mecanismo de ter que encontrar a qual tabela o campo especificado pertence.





Boas práticas SELECT

**Evite usar GROUP BY,
DISTINCT e ORDER BY o
máximo possível.**



Evite sempre que possível o uso de GROUP BY, DISTINCT e ORDER BY, pois consomem uma grande quantidade de recursos.

Considere se realmente é necessário utilizá-lo ou, se por outro lado, a ordenação dos resultados pode ser deixada para o aplicativo que receberá os dados.



4 | WHERE



Boas práticas WHERE

**Evite usar wilcards em
LIKE como "% value%"**



Caso a instrução LIKE seja usada, evite usar o wildcard "%" no início da string a ser pesquisada. Isso porque, se aplicada, a busca teria que ler todos os dados da tabela ou tabelas envolvidas para responder à consulta. Recomenda-se que haja pelo menos três caracteres antes do wildcard.





Boas práticas WHERE

**Evite usar IN, é melhor
EXISTS.**



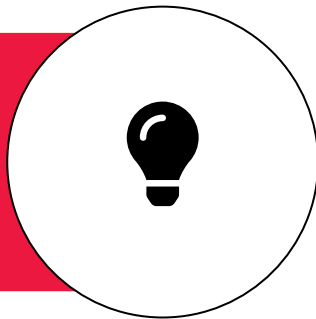
Prefira o uso de EXISTS e NOT EXISTS, ao invés de IN e NOT IN.





Boas práticas WHERE

**Tente não usar funções
nas condições WHERE.**



O SQL não pode pesquisar registros de maneira eficiente quando você usa funções, por exemplo de conversão, dentro de uma coluna. Nessas condições, tente usar o formato de coluna original.

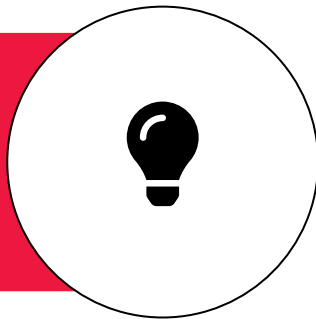


5 | UNION



Boas práticas UNION

Use UNION ALL para evitar uma distinção implícita.



No caso de utilizar a instrução UNION e houver certeza de que registros duplicados não serão obtidos nos SELECTs, é aconselhável neste cenário substituir UNION por UNION ALL para evitar o uso implícito da instrução DISTINCT, já que este consumo de recursos pode aumentar.

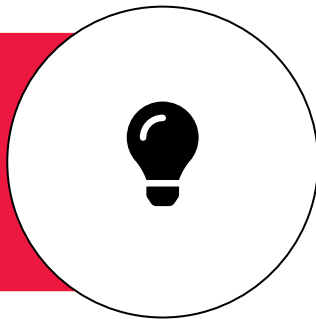


6 | CRUD



Boas práticas CRUD

Use **SET NOCOUNT ON**
com operações CRUD



Use **SET NOCOUNT ON** com operações CRUD para evitar a contagem do número de linhas afetadas e obter desempenho, especialmente em tabelas com muitos registros.



DigitalHouse>
Coding School