**SQL - Structured Query Language**

Linguagem padrão para manipular, armazenar e recuperar dados em bancos de dados.

# **Sumário**

[**Sumário**](#_heading=h.nxg0zhyvxw76) **1**

[**1. CONCEITOS INICIAIS**](#_heading=h.15k61a6bozwc) **2**

[1.0 Introdução](#_heading=h.4smis9qrgqjq) 2

[1.1 Sintaxe](#_heading=h.oqs1ayde1qod) 3

[1.2 SELECT](#_heading=h.6o7mjnffmiwd) 4

[1.3 SELECT DISTINCT](#_heading=h.a6db5byphubv) 4

[s1.4 WHERE](#_heading=h.ystq70rx224s) 5

[1.5 Operadores AND, OR e NOT](#_heading=h.46glqdy1xika) 7

[1.6 Order By](#_heading=h.j3lc748uiei4) 7

[1.7 INSERT INTO](#_heading=h.jr05nqe7d1ef) 8

[1.8 NULL](#_heading=h.eh82y8cufd54) 9

[1.9 UPDATE](#_heading=h.ubwtvmdoj05t) 9

[1.10 DELETE](#_heading=h.jqiaemlq9gpb) 10

[1.11 TOP / LIMIT / FETCH FIRST / ROWNUM](#_heading=h.bcjx3za5ybkb) 10

[1.12 MIN() and MAX()](#_heading=h.1i6leni9727o) 11

[1.13 COUNT(), AVG() e SUM()](#_heading=h.44ke8cc3suxq) 12

[1.14 LIKE](#_heading=h.ggm98sb7svxw) 12

[1.15 Wildcards](#_heading=h.ws2hgqdce43h) 13

[1.16 Operador IN](#_heading=h.s20mj5kgb9sf) 14

[1.17 Operador BETWEEN](#_heading=h.9t70ai34ja29) 15

[1.18 Aliases](#_heading=h.140kuj2zjerp) 16

[1.19 Join](#_heading=h.nio8s8hu907i) 17

[**2. Banco de dados SQL**](#_heading=h.3ihqyfx7l3g8) **17**

[2.0 CREATE DATABASE](#_heading=h.brv25tordjpi) 17

[2.1 DROP DATABASE](#_heading=h.cgyqksmw8wnn) 18

[2.2 BACKUP DATABASE (SQL SERVER)](#_heading=h.d20m1bs1htix) 18

[2.3 CREATE TABLE](#_heading=h.k1pm7sjsbr3v) 19

[2.4 DROP TABLE](#_heading=h.qul1xaq4kdar) 20

[2.5 ALTER TABLE](#_heading=h.hlad67r2bk15) 20

[2.6 CONSTRAINTS](#_heading=h.2m4gwkx7ljft) 21

[2.7 NOT NULL](#_heading=h.rdeiwk77cl1s) 22

# **1. CONCEITOS INICIAIS**

## **1.0 Introdução**

O SQL serve como uma linguagem de consulta estruturada, permitindo acessar e manipular banco de dados.

**Atribuição do SQL**:

* executa consultas.
* recupera dados.
* insere registros.
* atualiza registros.
* exclui registros.
* cria novos bancos de dados.
* cria novas tabelas de bancos de dados.
* cria procedimentos armazenados em bancos de dados.
* cria vizualizações.
* define permissões.

**Principais comandos**:

1. **SELECT**.
2. **UPDATE**.
3. **DELETE**.
4. **INSERT**.
5. **WHERE**.

**RDBMS** - gerenciamento de bancos de dados relacional.

* **armazena objetos em bancos de dados** (tabelas - coleção de entradas que consiste em colunas e linhas).

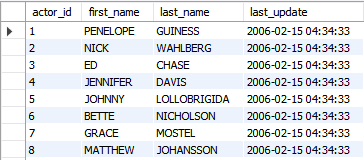
**Tabela de um banco de dados** - dividida em entidades menores chamadas de campos.

* **campo** - coluna que mantém informações específicas sobre cada registro na tabela.
* **registro (entidade horizontal)** - entrada individual que existe em cada linha da tabela.

## **1.1 Sintaxe**

Um banco de dados sempre apresenta uma ou mais tabelas, tendo cada uma um nome como identificador.

**Exemplo**:



**Instruções** - ações que podem ser executadas em um banco de dados.

**Exemplo**:



**Seleciona todos os registros da tabela address**.

**OBS: o SQL não diferencia palavras maiúsculas das minúsculas**.

**Ponto e vírgula** - forma padrão de separar instruções SQL.

* **permite que mais de uma instrução SQL seja executada** na mesma chamada do servidor.

**Comandos mais importantes**:

1. **SELECT** - extrai dados do banco.
2. **UPDATE** - atualiza dados do banco.
3. **DELETE** - exclui dados do banco.
4. **INSERTO INTO** - insere novos dados no banco.
5. **CREATE DATABASE** - cria um novo banco.
6. **ALTER DATABASE** - modifica um banco.
7. **CREATE TABLE** - cria uma nova tabela.
8. **ALTER TABLE** - modifica uma tabela.
9. **DROP TABLE** - exclui uma tabela.
10. **CREATE INDEX** - cria um índice (chave de pesquisa).
11. **DROP INDEX** - exclui um indice.

## **1.2 SELECT**

Seleciona dados que serão armazenados em uma tabela após retornados (tabela de resultados), **chamado de conjunto de resultados**.

**Exemplo**:



**Seleciona os campos da tabela onde os dados são selecionados**.

**Nota: para selecionar todos os campos utilize “ \* ”**.

## **1.3 SELECT DISTINCT**

Instrução usada para retornar apenas valores diferentes.

**Exemplo**:  


**Normalmente usado quando há muitos valores duplicados dentro da tabela**.

**Exemplo: listando número de dados diferentes**.



**outro modo: Somente para MS Access**.



## **s1.4 WHERE**

**Cláusula usada para filtrar registros**.

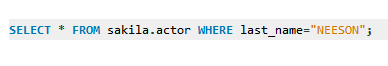
* Somente registros que atendem às condições específicas poderão ser chamados.

**Exemplo**:



**Cláusula pode ser usada em outras declarações como UPDATE, DELETE, etc**.

**Exemplo: selecionando apenas elementos específicos dentro da tabela**.

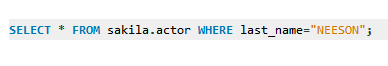


**campos de textos devem ser colocados entre aspas simples ou duplas e campos numéricos não devem ser colocados entre aspas**.

**Operadores**:

* **“ = “** → igual.

**Exemplo**:



* **“ > “** → maior que.

**Exemplo**:



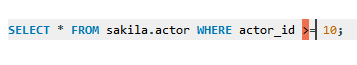
* **“ < “** → menor que.

**Exemplo**:



* **“ >= “** → maior ou igual.

**Exemplo**:



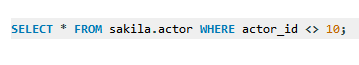
* **“ <= “** → menor ou igual.

**Exemplo**:



* **“ <> “** → diferente de (**Em algumas versões pode ser !=**).

**Exemplo**:



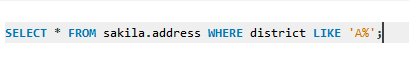
* **BETWEEN** → entre determinado intervalo.

**Exemplo**:



* **LIKE** → pesquisa um padrão.

**Exemplo**:



**Seleciona tudo que inicia com A**.

* **IN** → valores para uma coluna.

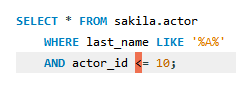
**Exemplo**:



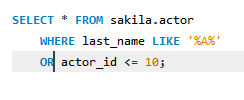
## **1.5 Operadores AND, OR e NOT**

**Operadores AND (todas condição verdadeiros - TRUE) e OR (uma condição verdadeira -TRUE)** são usados para filtrar registros com mais de uma condição.

**Exemplo**:

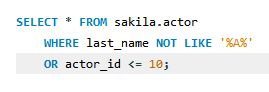


**usando AND**.



**usando OR**.

**Exemplo: utilizando o NOT**.



**Nota: os operadores podem ser combinados**.

**Exemplo**:

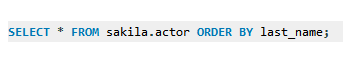
****

## **1.6 Order By**

Order by é usada para classificar um conjunto de resultados em **ordem crescente ou decrescente**.

**Nota: padrão está na ordem crescente**.

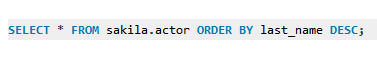
**Exemplo**:



**ordenando a tabela actor a partir do last\_name**.

**Desc** - classifica os registros em ordem decrescente.

**Exemplo**:



**ordenando a tabela actor a partir do last\_name de forma decrescente**.

**Exemplo: ordenando por várias colunas**.



**ordenar por last\_name e actor\_id. Se outras linhas tiverem o mesmo last\_name, ordena por actor\_id**.

**Exemplo: ordenando de forma crescente e decrescente**.



## **1.7 INSERT INTO**

Instrução usada para inserir novos registros em uma tabela.

**SINTAXE**:

* **INSERT INTO table\_name ( columnN ) VALUES (value1, …, valueN);**.
* inserindo valores para todas as colunas.
  + **INSERT INTO table\_name VALUES (value1, …, valueN);**.

**OBS: ordem dos valores devem está na mesma ordem das colunas**.

**Exemplo: inserindo registros a tabela**.



## **1.8 NULL**

**Campo de valor nul**l - sem valor.

* Se a **tabela for opcional**, é possível inserir um novo registro ou atualizar registro sem adicionar valor.

**OBS: campo com valor null é diferente de valores em zero ou campo que contém espaços**.

**Exemplo: testando valores nulos**.



**verifica se a valores nulos**.



**verifica quais tabelas não têm valor nulo**.

## **1.9 UPDATE**

Instrução usada para modificar registros existentes na tabela.

**Exemplo**:



**A condição WHERE é obrigatória. Set o valor de personID se lastName se iniciar com D**.

**OBS: se WHERE for omitida, todos os registros serão atualizados**.

**cláusula WHERE** - determina quantos registros serão atualizados.



**Atualiza valor somente se o PersonID for 1**.

## **1.10 DELETE**

Instrução usada para excluir registros existentes em uma tabela.

**Exemplo**:



**deleta somente aqueles com PersonID igual a 1. Cláusula WHERE é obrigatória para que não sejam excluídos todos os registros**.

**Exemplo: excluindo linhas de tabela sem perder sua estrutura, atributos e índices**.



## **1.11 TOP / LIMIT / FETCH FIRST / ROWNUM**

**SELECT TOP** - especifica o número de registros a serem retornados.

* **retornar um grande número de registros pode afetar o desempenho**.

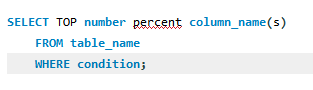
**OBS: nem todos os sistemas de banco suportam SELECT TOP**.

* **MySQL** suporta LIMIT que seleciona número limitado de registros.
* **Oracle** usa FEITCH FIRST e ROWS ONLY ROWNUM.

**Sintaxe em alguns bancos de dados**:

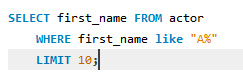
* **SQL server / MS Access syntax**

**Exemplo**:



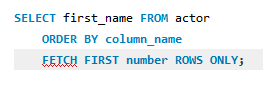
* **MySQL**

**Exemplo**:



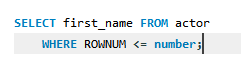
* **Oracle 12**

**Exemplo**:



* **Older oracle**

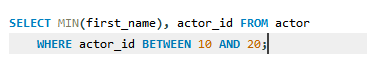
**Exemplo**:



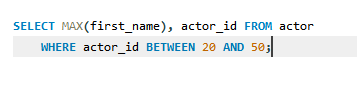
## **1.12 MIN() and MAX()**

funções retornam o menor e maior valor respectivamente.

**Exemplo**:



**retornando o menor nome dentro da coluna entre os valores 10 e 20**.

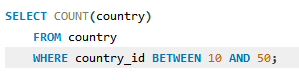


**retornando o maior nome dentro da coluna entre os valores 20 e 50**.

## **1.13 COUNT(), AVG() e SUM()**

**count** - retorna um número de linhas que correspondem ao critério especificado.

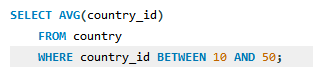
**Exemplo**:



**retorna a quantidade de linhas que se encontram entre o intervalo**.

**avg** - retorna valor médio de uma **coluna numérica**.

**Exemplo**:



**sum** - retorna o soma total de uma **coluna numérica**.

## **1.14 LIKE**

Operador usado em condicional WHERE para **pesquisar um padrão presente em uma coluna**.

**Wildcards (curingas) usados frequentemente**:

* **porcentagem (%)** - representa zero, um ou vários caracteres.
* **sinal sublinhado (\_)** - representa um único caractere.

**Nota: no MS Access é usado (\*) no lugar de (%) e (?) no lugar de (\_)**.

**Nota: pode ser usado AND e OR para qualquer número de condições**.

**Exemplo: Selecionando nomes que possuem o mesmo padrão**.



**seleciona todos os elementos que iniciam com A**.



**Seleciona elementos que possuem A no meio da palavra**.



## **1.15 Wildcards**

Usado para substituir um ou mais caracteres em uma string.

* **Os caracteres wildcards são usados como operador** - pesquisa um padrão especificado em uma coluna.

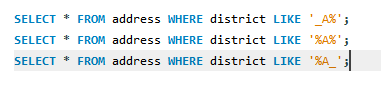
**Caracteres no MS Access**:

* **( \* )** - representa zero ou mais caracteres.
* **( ? )** - representa um único caractere.
* **[ ]** - representa qualquer caractere único dentro de colchetes.
* **( ! )** - representa qualquer caractere que não esteja entre colchetes.
* **( - )** - representa qualquer caractere único dentro de um intervalo especificado.
* **( # )** - representa qualquer caractere numérico.

**Caracteres no SQL Server**:

* **( % )** - representa zero ou vários caracteres.
* **( \_ )** - representa um único caractere.
* **[ ]** - representa qualquer caractere único dentro de colchetes.
* **( ^ )** - representa qualquer caractere que não esteja entre colchetes.
* **( - )** - representa qualquer caractere único dentro de um intervalo especificado.

**Exemplo**:



**Exemplo: selecionando endereços que iniciam com B, S ou C**.



## **1.16 Operador IN**

Permite especificar vários valores em um WHERE cláusula.

* atalho para vários OR condições.

**Exemplo**:



**seleciona as linhas que possuem city\_id 300**.



**valores que não possuem 300 serão selecionados**.



**seleciona todos os endereços que tenham o mesmo city\_id de city**.

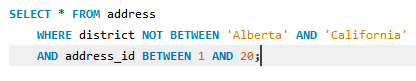
## **1.17 Operador BETWEEN**

**Operador seleciona valores de um determinado intervalo**, podendo ser números, textos ou datas.

* valores final e inicial são incluídos.

**Exemplo**:



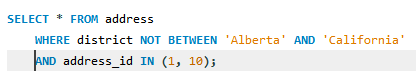


**endereços estão entre Alberta e Califórnia**.

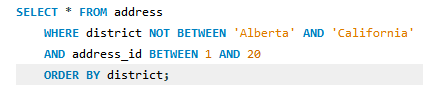
**Exemplo: selecionando valores que estão fora do intervalo**.



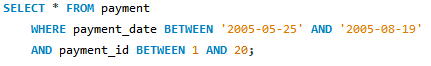
**Exemplo: between com in**.



**Exemplo: ordenando valores dentro de um intervalo**.



**Exemplo: intervalo de datas**.



**em outros bancos de dados**,



## **1.18 Aliases**

São usados para dar a uma tabela ou colunas um nome temporário, **tornando os nomes mais legíveis**.

* só existe durante uma consulta.

**Exemplo**:



**dando nome a coluna**.



**dando nome a tabela**.

**OBS: se nome temporário contiver espaços deve estar entre aspas duplas ou colchetes**.

**Exemplo: nome para colunas combinadas**.



**em outros bancos de dados**,



**No Oracle**,



**Exemplo: selecionando duas tabelas**.



**Alias podem ser úteis quando há mais de uma tabela sendo selecionada**.

**QUANDO USAR**:

* mais de uma tabela.
* funções usadas nas consultas.
* nomes de colunas grandes e não legíveis.
* mais de uma coluna combinada.

## **1.19 Join**

**Cláusula usada para combinar linhas de duas ou mais tabelas**, com base em coluna relacionada entre elas.

**INNER JOIN** - seleciona os registros que possuem valores correspondentes em ambas as tabelas.

CONTINUAR DAQUI → seleciona os registros que possuem valores correspondentes em ambas as tabelas:...<https://www.w3schools.com/sql/sql_join.asp>

# **2. Banco de dados SQL**

## **2.0 CREATE DATABASE**

Usada para criar uma novo banco de dados.

**Exemplo: criando um banco de dados**.



**OBS: é necessário ter permissão de adm para criar um banco de dados**.

**Exemplo: mostrando todos os bancos de dados**.



## **2.1 DROP DATABASE**

Instrução usada para descartar um banco de dados existente.

**Exemplo**:



**OBS: excluir um banco de dados (adm) resultará na perda de informação completa armazenada**.

## **2.2 BACKUP DATABASE (SQL SERVER)**

Usada no SQL SERVER para criar um backup completo de um banco de dados existente.

**sintaxe**: **BACKUP DATABASE dataBaseName TO DISK = “ filepath ”**;

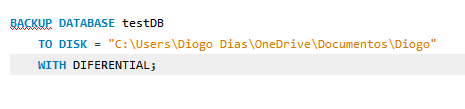
**sintaxe backup com diferencial**:

**BACKUP DATABASE dataBaseName TO DISK = “ filepath ” WITH DIFERENTIAL**;

**Exemplo: backup completo para o disco C:**.



**Exemplo: backup diferencial**.



## **2.3 CREATE TABLE**

Instrução usada para criar uma nova tabela em um banco de dados.

**sintaxe**:

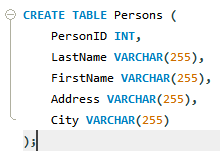
**CREATE TABLE table\_name (**

**column1 datatype, … , columnN datatype**

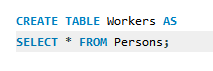
**)**.

* **column** → especifica o nome das colunas.
* **datatype** → define o tipo de dado da coluna.

**Exemplo**:



**Exemplo: criando cópia de tabela existente**.



**Todas as colunas de Persons foram copiadas**.

A nova tabela existente terá as **mesmas definições de coluna**.

* todas as colunas ou colunas existentes poderiam ser copiadas.

## **2.4 DROP TABLE**

Instrução usada para descartar uma tabela existente em um banco de dados.

**Sintaxe: DROP TABLE table\_name**;

**OBS: excluir uma tabela resultará na perda de informações completas armazenadas na tabela**.

**Exemplo**:



**TRUNCATE TABLE** - usado para excluir os dados dentro de uma tabela, mas não a tabela.

* **sintaxe: TRUNCATE TABLE table\_name**;

**Exemplo**:



## **2.5 ALTER TABLE**

Instrução usada para **adicionar, excluir ou modificar colunas** em uma tabela existente.

* pode ser usada para remover / adicionar várias restrições.

**Exemplo: adicionando uma coluna**.



**Exemplo: excluindo uma coluna**.



**Exemplo: renomeando uma coluna**.



**Exemplo: modificando o tipo de dado da tabela**.



**Modificação realizada no MySQL / Oracle**.



**Modificação realizada no Servidor SQL / MS Access**.



**Modificação realizada no Oracle ou superior**.

## **2.6 CONSTRAINTS**

Usada para especificar regras para dados em uma tabela.

* restrições criadas com **CREATE TABLE (na criação)** ou **ALTER TABLE (após a criação)**.
* limita os tipos de dados que entrarão na tabela.

Sintaxe: **CREATE TABLE table\_name ( column1 datatype CONSTRAINT, … );**.

**Níveis de restrições**:

* **coluna** → se aplicam a uma coluna.
* **tabela** → se aplicam a toda tabela.

**Tipos de restrições**:

* **NOT NULL** - sem valor null.
* **UNIQUE** - valores na coluna devem ser diferentes.
* **PRIMARY KEY** - identifica exclusivamente cada linha em uma tabela.
* **FOREIGN KEY** - impede ações que destruiriam links em tabelas.
* **CHECK** - valores devem satisfazer condições específicas.
* **DEFAULT** - define valor padrão.
* **CREATE INDEX** - cria e recupera dados do banco rapidamente.

**Exemplo**:



## **2.7 NOT NULL**

CONTINUAR DAQUI…<https://www.w3schools.com/sql/sql_notnull.asp>