SQL: introdução.

→ A linguagem SQL se divide em 5 subgrupos:

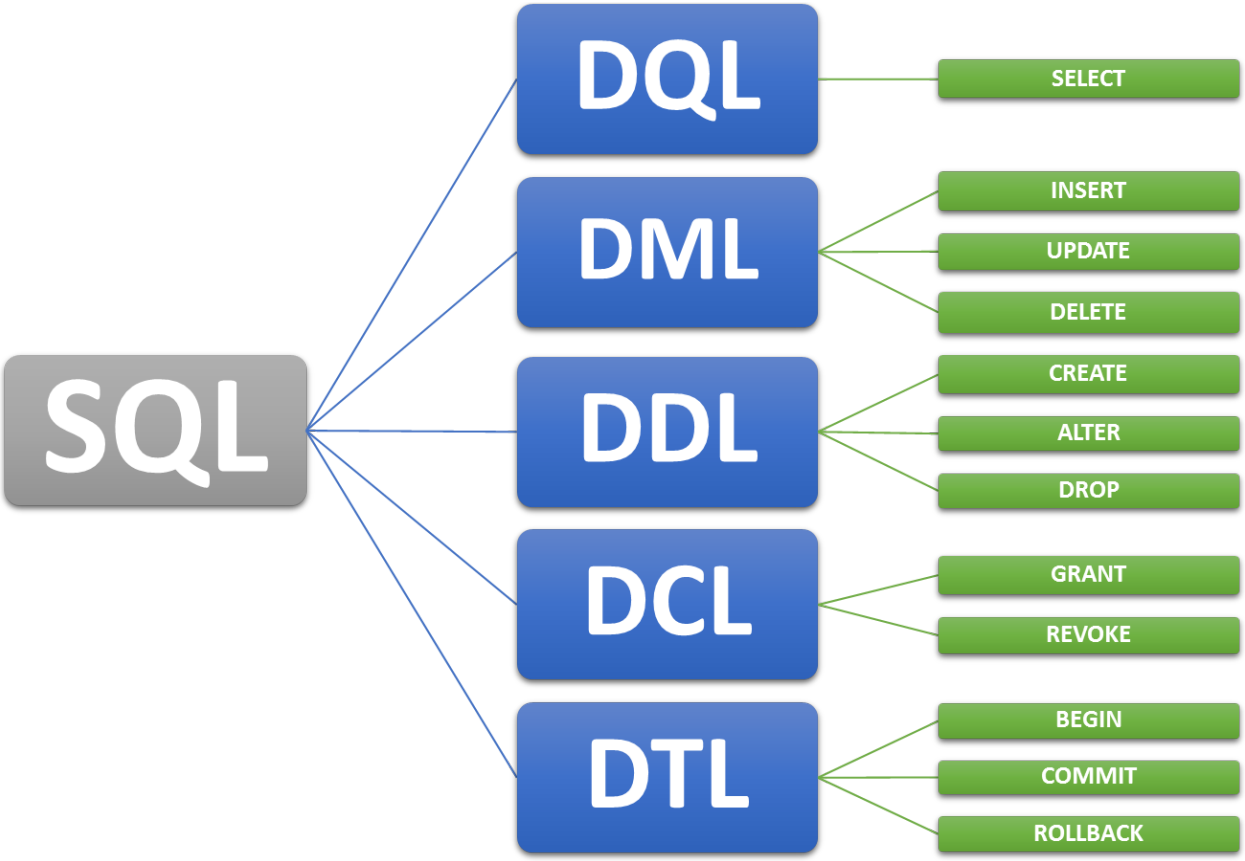
1. DQL = *Data Query Languages* = Linguagem de consulta de dados;
2. DML = *Data Manipulation Language* = Linguagem de manipulação de dados;
3. DDL = *Data Definition Language* = Linguagem de Definição de dados;
4. DCL = *Data Control Language* = Linguagem de Controle de dados;
5. DTL = *Data Transaction Language* = Linguagem de Transação de dados;

→ Cada subgrupo possui comandos próprios de execução;

→ Sempre teremos uma mensagem de execução mostrando se foi: sucesso ou erro;

OBS: todos os comandos em SQL finalizam com ponto e vírgula;

Comandos gerais;



Data Query Language → DQL;

→ No subgrupo DQL temos apenas **um** comando SQL: **SELECT**

→ Este comando é utilizado para realizar consultas no banco de dados.

→ Embora o DQL tenha apenas um comando, é a parte mais utilizada na SQL.

→ O comando SELECT permite ao usuário **especificar** uma consulta (query) como uma descrição do resultado esperado.
→ Este comando é composto de várias cláusulas e opções, possibilitando eleborar consultas das mais simples às mais complexas.

Tabela exemplo;

Tipo de produto

Código	Descrição
1	Computador
2	Impressora

Produto

Código	Descrição	Preço	CódigoDoTipo
10	Desktop	1.200,00	1
20	Laptop	1.800,00	1
30	Impr. Jato Tinta	300,00	2
40	Impr. Laser	500,00	2

Lembretes;

→ **SELECT * FROM** tipos_produto; (selecionando todos os dados da tabela);

→ O * indica que queremos os dados de todos os campos da tabela;

SELECT codigo, descricao FROM tipo_produto;

→ no exemplo acima, estamos selecionando todos os dados da tabela ‘tipos_produto’, mas dessa vez, especificando os campos que queremos os dados.

SELECT * FROM produtos;

→ Selecionando todos os dados da tabela ‘produtos’

→ E teremos como resultado as 4 linhas da tabela produtos com os dados de todas as colunas, juntamente com a mensagem de sucesso;

SELECT codigo, descricao, codigo_tipo FROM produtos;

→ No exemplo acima, estamos selecionando todos os dados de apenas 3 colunas da tabela' produtos’

SELECT cod, desc, pre, ctp FROM produtos;

→ No exemplo acima, estamos selecionando dados de colunas inexistentes na tabela produtos. Isso além de não trazer como resultado nenhum dado ainda nos apresentará erro.

Colocando um alias (apelido) em nome de tabela e campos;

SELECT p.codigo AS cod, p.descricao AS desc, p.preco AS pre, p.codigo_produto AS ctp FROM produtos AS p;

-No exemplo acima, estamos realizando uma busca especificando os campos que queremos da tabela 'produtos ' e adicionando um *alias* tanto para o nome da tabela quanto para o nome dos campos.

Data Manipulation Language → DML

→ INSERT = Usado para inserir um registro à uma tabela existente;

→ UPDATE = Usado para alterar valores de dados em um ou mais registros de uma tabela;

→ DELETE = Usado para remover registros de uma tabela;

Tabela base;

Tipo de produto

Código	Descrição
1	Computador
2	Impressora

Produto

Código	Descrição	Preço	CódigoDoTipo
10	Desktop	1.200,00	1
20	Laptop	1.800,00	1
30	Impr. Jato Tinta	300,00	2
40	Impr. Laser	500,00	2

Exemplo #1 INSERT

INSERT INTO tipos_produto (descricao) VALUES ('Notebook');

→ No exemplo acima, estamos inserindo o valor 'Notebook' na tabela 'tipos_produto'. O campo 'código' do tipo de produto é **chave primária** e **auto incremento**, então será inserido automaticamente.

→ Após os comandos, a nova tabela:

Código	Descrição
1	Computador
2	Impressora
3	Notebook

Exemplo #2 INSERT

INSERT INTO produtos (descricao, preco, codigo_tipo_produto) VALUES ('Notebook', 1200, 1);

→ No exemplo acima, estamos inserindo os valores 'Notebook, 1200 e 1' na tabela 'produtos'. O campo 'codigo' do produto é chave primária e auto incremento, então será

inserido automaticamente.

Código	Descrição	Preço	CódigoDoTipo
10	Desktop	1.200,00	1
20	Laptop	1.800,00	1
30	Imp. Jato Tinta	300	2
40	Imp. Jato Laser	500	2
41	Notebook	1200	1

Exemplo #3 UPDATE ⇒ Jamais dar UPDATE sem WHERE;

UPDATE tipos_produto set descricao = 'Nobreak' WHERE codigo = 3;

Código	Descrição
1	Computador
2	Impressora
3	Nobreak

→ No UPDATE o WHERE funciona como uma clausula de FILTRO;

UPDATE produtos set descricao = 'Notebook', preco = 2.800 WHERE codigo = 20;

Código	Descrição	Preço	CódigoDoTipo
10	Desktop	1.200,00	1
20	Notebook	2.800,00	1
30	Imp. Jato Tinta	300	2
40	Imp. Jato Laser	500	2

⇒ Se utilizar UPDATE sem WHERE vai mudar todas as colunas com o respectivo comando que você passou;

UPDATE produtos set descricao = 'Notebook', preco = 2.800;

Código	Descrição	Preço	CódigoDoTipo
10	Notebook	2800	1
20	Notebook	2800	1
30	Notebook	2800	2
40	Notebok	2800	2

A única forma de salvar os dados perdidos com o UPDATE sem WHERE seria um backup do banco de dados;

Exemplo #4 DELETE

Tabelas base;

#1

Código	Descrição
1	Computador
2	Impressora
3	Notebook

#2

Código	Descrição	Preço	CódigoDoTipo
10	Desktop	1.200,00	1
20	Notebook	2.800,00	1
30	Imp. Jato Tinta	300	2
40	Imp. Jato Laser	500	2

DELETE FROM tipos_produto WHERE codigo = 3;

→ Estamos excluindo da tabela o registro que contenha o código 3 (Notebook), gerando a seguinte tabela:

Código	Descrição
1	Computador

2	Impressora

DELETE sem WHERE (que é o elemento que filtra irá excluir TODOS os registros da tabela);

Data Definition Language → DDL

→ CREAT:

CREATE DATABASE financeiro;

→ No exemplo acima, estamos criando um banco de dados chamado 'financeiro';

CREATE TABLE tipos_produto (codigo INT PRIMARY KEY, descricao VARCHAR(50));

→ No exemplo acima, estamos criando uma tabela chamada 'tipos_produto' contendo os campos codigo, do tipo **int e como chave primária**, e descricao do tipo **varchar** com até 50 caracteres.

Gerando:

Código	Descrição

Em resumo: Com isso uma tabela seria criada no banco de dados 'financeiro' criado anteriormente e uma tabela ficaria disponível para que possamos inserir e manipular os dados.

→ ALTER;

ALTER TABLE tipos_produto ADD peso DECIMAL(8,2);

→ No exemplo acima estamos alterando a estrutura da tabela 'tipos_produto' acrescentando um novo campo chamado 'peso' do tipo decimal com até 8 dígitos antes da vírgula e 2 dígitos após a vírgula.

Código	Descrição	Peso

→ Com isso a tabela teria mais um campo onde seria possível colocar o peso do tipo de produto.

→ DROP;

DROP TABLE tipos_produto;

→> No exemplo acima estamos apagando a tabela 'tipos_produto'. Este comando apaga toda a estrutura e os dados, desde que esta tabela não tenha relacionamentos.

DROP DATABASE financeiro;

→ No exemplo acima, excluímos o banco de dados 'financeiro' e qualquer tabela ou dado dentro dele.

Diferença entre DELETE x DROP:

Delete ⇒ Deleta os dados;

DROP ⇒ Deleta a estrutura da tabela;

Data Control Language → DCL;

→ Este subgrupo da linguagem SQL é utilizado para controlar os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para manipular dados dentro do banco de
dados.

→ GRANT: Usado para autorizar um usuário a executar ou setar operações no banco de dados;

→ REVOKE: Usado para remover ou restringir a capacidade de um usuário de executar operações;

GRANT SELECT ON tipos_produto TO geek;

→ Dando permissão na consulta da tabela *tipos_produtos para o usuário 'GEEK'*

REVOKE CREATE TABLE FROM geek;

→ Removendo permissão para o usuário ‘geek’, ele não vai mais conseguir criar tabelas;

Data Transection Language → DTL

→ Begin (ou START TRANSACTION) = É usado para marcar o começo de uma transação que pode ser completada ou não;

→ Commit = Finaliza uma transação;

→ Rollback = Faz com que as mudanças nos dados existentes deste o último commit sejam descartadas;

CREATE TABLE 'tipos_produtos' (codigo INT PRIMARY KEY, descricao VARCHAR(50));
BEGIN TRANSACTION; -- **começamos a transação**
INSERT INTO tipos_produtos VALUES ('Notebook');
INSERT INTO tipos_produtos VALUES ('Nobreak');
COMMIT; - termina a transação e grava os dados;

→ No exemplo acima, estamos iniciando a transação, inserindo os dados e salvando no banco.

ROLLBACK → EXEMPLO

CREATE TABLE 'tipos_produtos' (codigo INT PRIMARY KEY, descricao VARCHAR(50));
BEGIN TRANSACTION; -- começamos a transação
INSERT INTO tipos_produtos VALUES ('Notebook');
INSERT INTO tipos_produtos VALUES ('Nobreak');
ROLLBACK; -- as inserções das linhas acima foram desfeitas

→ No exemplo acima, estamos iniciando a transação, inserindo os dados mas note que o COMMIT não foi executado, mas sim o ROLLBACK. Desta forma, as inserções foram desfeitas e não ficarão salvas no banco.