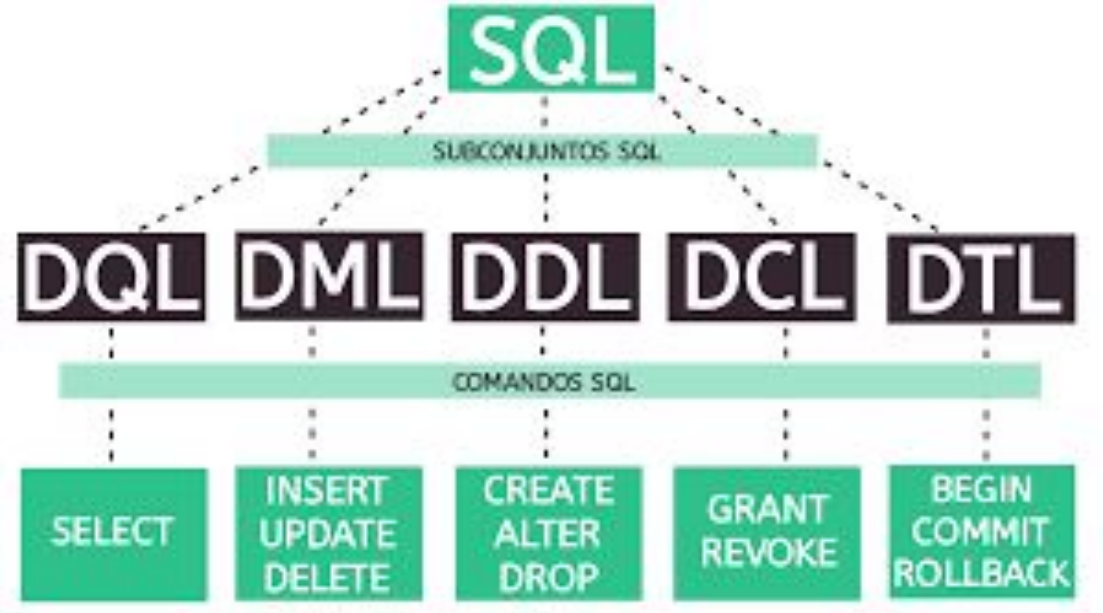
BD AULA 03
sql ansi



SQL

INTRODUÇÃO

Para se utilizar, administrar, e trabalhar com um banco de dados é utilizada uma linguagem padrão, que a maior parte dos SGBD aceitam. Essa linguagem é a SQL (Structured Query Language-Linguagem de Consulta Estruturada).

SQL

A SQL é um conjunto de declarações que são utilizadas para acessar os dados utilizando gerenciadores de banco de dados.

Apesar de nem todos os gerenciadores utilizarem a SQL, a maior parte deles aceita suas declarações.

SQL

SQL para manipulação de bancos de dados MySQL

A SQL possui comandos que são utilizados para manipular os bancos de dados, as tabelas e os registros existentes.

SQL

COMANDO CREATE

Este comando permite a criação de bancos de dados ou de tabelas num banco de dados.

Sintaxe:

CREATE DATABASE < nome_db >;

onde: nome_db: indica o nome do Banco de Dados a ser criado.

Exemplo:

CREATE DATABASE curso;

SQL

```
CREATE TABLE < nome_tabela > (
nome_atributo1 < tipo > [ NOT NULL ],
nome_atributo2 < tipo > [ NOT NULL ],
nome_atributoN < tipo > [ NOT NULL ]
PRIMARY KEY(nome_atributo)
) ;
```

SQL

Exemplo:

```
CREATE table alunos(
codigo int NOT NULL AUTO_INCREMENT
primary key,
nome VARCHAR(20) NOT NULL ,
telefone CHAR(8) NOT NULL
);
```

SQL

alunos		
nome	*cod_Aluno	cod_turma
joao santos	1001	3010
maria jose	1002	1010
clara silva	1003	2022
claudio filho	1004	2001
josé soares	1005	1011

SQL

```
CREATE TABLE alunos (
cod_aluno INT(4) NOT NULL
auto_increment,
nome_aluno VARCHAR(45),
cod_turma INT(4),
PRIMARY KEY(cod_aluno)
);
```

SQL

comando SHOW DATABASES;

SHOW TABLES;

DESC ALUNOS;

SQL

crie a tabela a seguir para podermos praticar

SQL

id_funcionario	Nome	CPF	departamento	cpf_supervisor	salario
1	Mario jose	10203040231	3	98765432198	3500,20
2	Maria Clara	32123123424	2	87654321908	3423,45
3	João Pedro	23124543234	3	76543210987	7004
4	vinicius morais	56474567865	5	45678902134	34234
5	wellington oliveira	45345367876	1	21345365476	16045
6	clarisse santos	12345678904	5	12345434567	7856
7	vilma mairia	36840793333	4	87656787654	4354
8	josé pereira	31285598333	3	23454321234	76545
9	claudemir silva	50202573040	2	32432543678	9876

SQL

COMANDO INSERT

Adiciona um ou vários registros a uma tabela.

Sintaxe básica:

INSERT INTO alunos (nome, telefone) VALUES ('wellington Oliveira','34343434');

SQL

```
INSERT INTO alunos(nome_aluno, cod_turma)
VALUES ('Carlos oliveira', 3050);
INSERT INTO alunos(nome_aluno, cod_turma)
VALUES ('maria silva', 3051);
INSERT INTO alunos(nome_aluno, cod_turma)
VALUES ('cristina lima', 3053);
INSERT INTO alunos(nome_aluno, cod_turma)
VALUES ('joana santos', 3053);
INSERT INTO alunos(nome_aluno, cod_turma)
VALUES ('flavio cavalcanti', 3052);
```

SQL

COMANDO UPDATE

O comando UPDATE altera os valores de alguns campos de uma tabela especificada, com base em critérios específicos.

Sintaxe:

UPDATE tabela SET campo1 = valornovo, ...
WHERE critério;

SQL

Onde:

tabela: O nome da tabela onde você quer modificar os dados.

valornovo: Uma expressão que determina o valor a ser inserido no campo do registro que será atualizado.

critério: Uma expressão que determina quais registros devem ser atualizados. Só os registros que satisfazem a expressão são atualizados.

SQL

O comando UPDATE é bastante útil quando você quer alterar muitos registros ou quando os registros que você quer alterar estão em várias tabelas. Você pode alterar vários campos ao mesmo tempo.

Utilizando a cláusula UPDATE é possível alterar os registros

SQL

da tabela funcionários, para que os funcionarios que integram o depto 3 passem a pertencer ao depto 5:

UPDATE funcionarios SET depto=5 WHERE depto=3;

Caso fosse necessário dar um aumento de 20% de salário aos funcionários que ganham menos de 3000 reais o comando seria o seguinte:

SQL

```
UPDATE FUNCIONARIOS SET SALARIO = SALARIO*1.2 WHERE SALARIO <3000;
```

SQL

COMANDO DELETE

Remove registros de uma ou mais tabelas listadas na cláusula FROM que satisfazem a cláusula WHERE.

Sintaxe:

DELETE FROM tabela WHERE critério

SQL

O comando DELETE exclui registros inteiros e não apenas dados em campos específicos. Se você quiser excluir valores de um campo específico, use o comando UPDATE que mude os valores dos campos para NULL.

SQL

Após remover os registros usando uma consulta DELETE você não poderá desfazer a operação.
Fazendo a operação:
DELETE FROM funcionarios WHERE salario>7999;

SQL

Após remover os registros usando uma consulta DELETE você não poderá desfazer a operação.
Fazendo a operação:
DELETE FROM funcionarios WHERE salario>7999;

SQL

comando SELECT
Pesquisa dentro da tabela;
sintaxe básica
SELECT * from alunos;
SELECT nome_aluno, cod_aluno FROM alunos;

SQL

Sintaxe básica:

SELECT [DISTINCT] expressao [AS nome-atributo]
[FROM from-lista]
[WHERE condicao]
[ORDER BY attr_name1 [ASC | DESC]]

SQL

onde:

DISTINCT: Elimina linhas duplicadas na seleção.

expressao: Define os dados que queremos selecionar, normalmente uma ou mais colunas de uma tabela que está em from-lista.

AS nome-atributo: Define um alias (apelido) para o nome da coluna.

SQL

FROM: Lista das tabelas onde a pesquisa será feita.
WHERE: Condição para que um registro seja selecionado.
ORDER BY: Critério para ordenação dos registros selecionados. Utilizando **ASC** a ordem será crescente, utilizando
DESC a ordem será decrescente.

SQL

Where como base das Restrição de linhas.
A cláusula "where" restringe a seleção de dados, de acordo com seu argumento. Contém a condição que as linhas devem obedecer a fim de serem listadas. Ela pode comparar valores em colunas, literais, expressões aritméticas ou funções.

SQL

A seguir apresentamos operadores lógicos e complementares a serem utilizados nas expressões apresentadas em where.

Operadores lógicos operador significado

= igual a

> maior que

>= maior que ou igual a

< menor que

<= menor que ou igual a

SQL

Exemplos:

SELECT cidade, estado FROM brasil WHERE populacao > 100000;

Selecionará os campos cidade e estado da tabela brasil de todos os registros que tiverem valor maior que 100.000 no campo populacao.

SQL

SELECT * FROM cidadao ORDER BY nome DESC;
Selecionará todos os campos da tabela cidadao e utilizará ordenação decrescente na seleção.

SQL

Seleções:

-Selecionar quantas pessoas existem cadastradas:

```
SELECT COUNT(*) FROM funcionarios;
```

-Selecionar quantos funcionários existem no departamento 3:

```
SELECT COUNT(*) FROM funcionarios WHERE depto=3;
```

SQL

Selecionar o nome e o rg dos funcionários que ganham mais que 3000 reais.

```
SELECT nome, cpf FROM funcionarios
WHERE salario>3000;
```

SQL

Exemplos:

```
SELECT NOME, departamento, salario
FROM funcionarios WHERE salario
BETWEEN 1000 AND 2500;

SELECT NOME, departamento
FROM funcionarios WHERE
departamento IN (10,30);
```

SQL

```
SELECT nome, valor FROM produtos
WHERE nome LIKE 'tv%';
```

SQL

SELECT EMPNOME, EMPSERV FROM EMP WHERE EMPCOMI IS NULL;

O símbolo "%" pode ser usado para construir a pesquisa ("%"

= qualquer sequência de nenhum até vários caracteres).

SQL

```
SELECT nome, salario
FROM funcionarios  WHERE nome LIKE
'j%';
```

O símbolo "%" pode ser usado para construir a pesquisa ("%"
= qualquer sequência de nenhum até vários caracteres).

SQL

```
select nome, salario,
if(salario>5000,'alto','baixo') as
classificação from funcionarios;
```

SQL

COMANDO DROP

Este comando elimina a definição da tabela, seus dados e referências ou um banco de dados existente:

SQL

Sintaxe:

DROP TABLE < nome_tabela > ;

DROP DATABASE <nome_banco_de_dados>;

Exemplo:

DROP TABLE alunos;

DROP DATABASE curso;

DROP TABLE estudantes;

SQL

COMANDO ALTER

Este comando permite inserir/eliminar atributos nas tabelas já existentes.

Sintaxe:

```
ALTER TABLE < nome_tabela > ADD / DROP (
nome_atributo1 < tipo > [ NOT NULL ],
nome_atributoN < tipo > [ NOT NULL ]
) ;
```

SQL

COMANDO ALTER

```
ALTER TABLE funcionarios DROP COLUMN cpf_supervisor;
```

SQL

COMANDO ALTER

ALTER TABLE alunos MODIFY nome VARCHAR (255) NOT NULL;

Tabela estoque

Field	Type	Null	Key	Default	Extra
codigo	char(8)	NO		NULL	
descrição	varchar(20)	YES		NULL	
barcode	int	NO	PRI	NULL	auto_increment
valor_unitário	int	YES		NULL	

SQL

Task do dia:

1. **corrija os nomes de atributos que possuem acentuação;**
2. **A chave primária deve ser o campo código;**
3. **corrija os tipos de dados do campo;**
4. **A chave primaria deve ser auto_increment**
5. **mude o nome da tabela para produtos**

SQL

```
CREATE TABLE matriculado(
id_matriculado INT(6) NOT NULL AUTO_INCREMENT PRIMARY KEY,
fk_matricula INT,
FOREIGN KEY (fk_matricula) REFERENCES alunos(matricula)
);
```

Desafio: Cadastro de Livros na Biblioteca
Cenário
Uma biblioteca precisa organizar seus livros e fazer um levantamento simples das informações dos livros mais lidos. Sua turma será responsável por criar a estrutura dessa base de dados e responder a algumas perguntas.

Objetivos
Criar uma base de dados chamada biblioteca.

Criar uma tabela chamada livros com os seguintes campos:

id (int, chave primária, autoincremento)
titulo (varchar(100), não nulo)
autor (varchar(50), não nulo)
ano_publicacao (int)
categoria (varchar(30))
total_emprestimos (int, inicializado como 0)

Inserir registros:

Adicione ao menos 30 livros com títulos, autores, anos de publicação e categorias variadas.

Query para listar todos os livros disponíveis na biblioteca.
Query para ordenar os livros pelo ano de publicação em ordem decrescente.
Query para selecionar apenas os livros da categoria ‘Ficção’.
Query para aumentar o número de empréstimos em 1 para o livro com id = 2.
Query para excluir um livro específico da tabela (use um id aleatório entre os que foram inseridos).
Query para mostrar os 3 livros mais emprestados, em ordem decrescente de total_emprestimos.

INNER JOIN – Consultar dados em duas ou mais Tabelas no MySQL

JOINS

A cláusula JOIN é usada para combinar dados provenientes de duas ou mais tabelas do banco de dados, baseado em um relacionamento entre colunas destas tabelas. há duas categorias principais de joins:

INNER JOIN – Consultar dados em duas ou mais Tabelas no MySQL

INNER JOIN: *Retorna linhas (registros) quando houver pelo menos uma correspondência em ambas as tabelas.*
OUTER JOIN: *Retorna linhas (registros) mesmo quando não houver ao menos uma correspondência em uma das tabelas (ou ambas).* ***O OUTER JOIN divide-se em LEFT JOIN, RIGHT JOIN e FULL JOIN.***

INNER JOIN – Consultar dados em duas ou mais Tabelas no MySQL

>

```
SELECT  alunos.matricula, alunos.nome,
        disciplinas.id_disc, disciplinas.nome
        FROM matriculados
        INNER JOIN alunos ON matriculados.matricula =
alunos.matricula
        INNER JOIN disciplinas ON matriculados.id_disc=
disciplinas.id_disc;
```

INNER JOIN – Consultar dados em duas ou mais Tabelas no MySQL

Onde tabela1.matricula é o nome da primeira tabela concatenado com um ponto e com o nome da coluna chave primária ou estrangeira da tabela, e tabela2.matricula é o nome da segunda tabela concatenado com um ponto e com a chave estrangeira ou primária dessa tabela que se relaciona com a chave da primeira tabela

INNER JOIN – Consultar dados em duas ou mais Tabelas no MySQL

```
SELECT * FROM tbl_Livro
INNER JOIN tbl_autores
ON tbl_Livro.ID_Autor = tbl_autores.ID_Autor;
```

INNER JOIN – Consultar dados em duas ou mais Tabelas no MySQL

ID_Livro	Nome_Livro	ISBN	ID_Autor	Data_Pub	Preco_Livro	ID_editora	ID_Autor	Nome_Autor	Sobrenome_Autor
1	Linux Command Line and Shell Scripting	143856969	5	2009-12-21	68.35	4	5	Richard	Blum
2	SSH, the Secure Shell	127658789	1	2009-12-21	58.30	2	1	Daniel	Barret
3	Using Samba	123856789	2	2000-12-21	61.45	2	2	Gerald	Carter
4	Fedora and Red Hat Linux	123346789	3	2010-11-01	62.24	1	3	Mark	Sobell
5	Windows Server 2012 Inside Out	123356789	4	2004-05-17	66.80	3	4	William	Stanek
6	Microsoft Exchange Server 2010	123366789	4	2000-12-21	45.30	3	4	William	Stanek
7	Enciclopédia de Componentes Eletrônicos vol. 03	153642397	13	2016-05-05	63.39	5	13	Charles	Platt

SQL View

Uma View é um objeto que pertence a um banco de dados, definida baseada em declarações SELECT's, retornando uma determinada visualização de dados de uma ou mais tabelas. Esses objetos são chamados por vezes de "virtual tables", formada a partir de outras tabelas que por sua vez são chamadas de "based tables" ou ainda outras Views.

SQL View

Em alguns casos, as Views são atualizáveis e podem ser alvos de declaração INSERT, UPDATE e DELETE, que na verdade modificam sua “based tables”.

SQL View

Os benefícios da utilização de Views, além dos já salientados, são:

Uma View pode ser utilizada, por exemplo, para retornar um valor baseado em um identificador de registro;
Pode ser utilizada para promover restrições em dados para aumentar a segurança dos mesmos e definir políticas de acesso em nível de tabela e coluna. Podem ser configurados para mostrar colunas diferentes para diferentes usuários do banco de dados;

SQL View

```
CREATE VIEW vw_viewCity AS
SELECT ID, Name
FROM City;
```

```
SELECT *
FROM vw_viewCity
LIMIT 3;
```

SQL View

alter table alunos add constraint foreign key (fk_turmas) references turmas(id_turma);

SQL View

```
select alunos.nome, turmas.nome from alunos inner join turmas on alunos.fk_turma = turmas.id_turma;
```

"cep": "01001-000",
"logradouro": "Praça da Sé",
"complemento": "lado ímpar",
"unidade": "",
"bairro": "Sé",
"localidade": "São Paulo",
"uf": "SP",
"estado": "São Paulo",
"regiao": "Sudeste",
"ibge": "3550308",
"gia": "1004",
"ddd": "11",
"siafi": "7107"

```
import requests
cep = input("Qual o cep?")
if len(cep) == 8:
    link = f'https://viacep.com.br/ws/{cep}/json/'
    requisicao = requests.get(link)
    print(requisicao)
    dic_requisicao = requisicao.json()

    uf = dic_requisicao['uf']
    cidade = dic_requisicao['localidade']
    bairro = dic_requisicao['bairro']
    print(dic_requisicao)
else:
    print("CEP Inválido")
```