Curso de Tecnologia da Informação

Curso de SQL e MySQL

Campina Grande, 2022

# Introdução

## O que é SQL

* A SQL é uma linguagem de banco de dados;
* É um acrônimo de Structured Query Language( linguagem estruturada de consulta)
* É uma linguagem para os bancos de dados relacionais(MySQL, PostreSQL, SQL serve);
* Trabalhamos com ela criando Queries( INSERT,UPDATE,SELECT);
* Com esta linguagem podemos criar e manipular DBs.

## SQL x MySQL

* SQL é a linguagem e MySQL é o SGBD;
* SGBD é u acrônimo de Sistema Gerenciador de Banco de Dados;
* Ou seja, MySQL é um software que gerencia os nossos bancos de dados;
* Enquanto SQL é a linguagem que vamos utilizar para manipular e criar os bancos.

## O que é um banco de dados¿

* É onde armazenamos os dados no nosso sistema;
* Vamos criar os bancos de dados através da SQL;
* E o MySQL vai ajudar-nos a gerenciar o banco de dados;
* O banco de dados possui algumas entidades fundamentais para o seu correto funcionamento, como tabelas;
* As tabelas guardam nossos dados.

## Principais elementos de um BD

* Diagrama do banco: é o projeto do banco, parte fundamental e mais avançada, que define o sucesso do projeto;
* Banco de dados: A entidade que vai guardar todos os elementos do banco;
* Tabelas: A categoria dos dados;
* Colunas: tipos de informações que precisamos salvar;
* Dados: O dado final entregue pelo usuário.

## Sintaxe do SQL

* Em SQL, todas as instruções são em maiúsculos e os nomes são em minúsculo (banco, tabela, coluna);
* CREATE DATABABE <nome>;
* Toda instrução finalizada com ponto e vírgula;
* SHOW DATABASES -> mostra os bancos de dados.

## Importação de banco

* Importação de banco é quando temos um arquivo pronto de banco e inserimos ele no nosso SGBD;
* Após a importação podemos utilizar como se o banco houvesse sido criado na nossa máquina;
* Criar um banco, pode ser empresa o nome;
* Deve-se criar o banco com o mesmo nome do arquivo;
* Pelo vs code, abra o terminal na pasta do arquivo;
* CREATE DATABASE <nome>;
* SHOW DATABASES;
* Selecionar o banco com: **USE** <nome>
* Depois utilizar o comando **source** <arquivo>.sql

## Selecionar dados de uma tabela

* **SELECT \*FROM** <nome da tabela>
* **Desta maneira receberemos os dados inseridos a tabela alvo;**

# Gerenciando banco de dados

## Criando um banco de dados

* CREATE DATABASE <nome>;

## Removendo banco de dados

* DROP DATABASE <nome>;
* Para deletar um banco de dados utilizar o comando a cima;

## Exportando bancos

* Mysqldump -u root <nome\_banco> <nome\_arquivo>.sql
* Desta forma todas as tabelas e dados ficarão salvas em um arquivo .sql

## Utilizando banco de dados

* Para qualquer ação que envolva um determinado banco, precisamos utilizar ele;
* O camando é: USE <nome>

# Gerenciamento de tabelas

## O que é uma tabela

* É a entidade responsável por guardar nossos dados;
* Uma tabela possui colunas;
* As colunas são como categorias dos nossos dados: nome, profissão, idade;
* As colunas possuem tipos de dados determinados, como: textos, números, datas e etc;
* E atributos: não nulos, chave primária, auto incremento e etc.

## Criando tabelas

* O comando é: CREATE TABLE <nome> (<coluna><tipo de dados>);
* O tipo de dado mais comum é o VARCHAR, que representa textos/strings de uma tabela;

## Removendo tabelas

* Todos os dados serão perdidos;
* Comando é: DROP TABLE <nome>;

## Tipos de dados

* Tipos de texto
  + CHAR (x) -> aceita textos com 0 a 255 caracteres;
  + VARCHAR(x): aceita textos com 0 a 65535 caracteres;
  + X é a quantidade máxima;
  + MEDIUMTEXT apenas texto com até 16777215 caracteres;
  + VARCHAR e CHAR aceitam números e caracteres especiais.
* Tipos numéricos
  + BIT(x): 1 a 64 caracteres;
  + TINYINT(x): 1 a 255;
  + BOOL: 0 é falso e outrso valores são verdadeiros;
  + INT(x): vaores entre -214783648 a 214783647;
* Tipos de data
  + DATE: aceita uma data no formato YYYY-MM-DD;
  + DATETIME Aceita uma data com horário no formato YYYY-MM-DD hh:mm:ss;
  + TIMESTAMP: aceita uma data no formato de DATETIME, porém apensas entre os anos 1970 a 2038;

## Inserir dados

* O comando é: INSERT INTO <tabela> (<colunas>) VALUES(<valores>);
* Precisamos inserir o nome das colunas e também os valores para cada uma, separados por vírgula;
* Os valores precisam corresponder a ordem das colunas;
* OBS: valores de texto são inseridos entre aspas.

## Alterando tabelas

* Adição de colunas;
* Remoção de colunas
* Modificar tipo da coluna;
* Comando : ALTER TABLE <tabela> ADD COLUMN <nome><tipo>;
* Comando: ALTER TABLE <table> DROP COLUMN <nome>;
* ALTER TABLE <tabela> MODIFY COLUMN <coluna><tipo>;

# Queries do CRUD

* CRUD são as ações que mais são utilizadas em todas as aplicações;
* Create = criar/inserir dados (insert);
* Read = ler dados (select);
* Upadate = atualizar dados (update);
* Delete = deletar/remover dados (delete);

## Selecionar colunas específicas

* SELECT coluna1, coluna2 FROM <tabela>;
* Desta maneira criamos um filtro;

## WHERE

* WHERE id = 1;
* Se não inserirmos esta instrução em atualizações ou remoções, vamos aplicar a todos os dados, o que pode ser um enorme problema;
* Então lembre-se sempre de inserir WHERE em UPFATE e DELETE;
* SELECT \* FROM pessoas WHERE nome = “Pedro”;

## Atualizando dados

* Para atualizar dados utilizar a instrução UPDATE;
* UPDATE tabela SET < coluna = valor> WHERE < condição>;
* Lembre-se de utilizar o WHERE;
* Pode-se usar mais colunas utilizando a virgular;
* SET SQL\_SAFE\_UPDATES = 0;
* UPDATE <tabela> SET <coluna> <coluna2> WHERE <nome> = “string”;

## Deletando dados

* Para deleter dados vamos utiliza a instrução DELETE;
* DELETE FROM <tabela> WHERE <condição;
* Use sempre o WHERE, se não apagará todos os dados;

# Avançando em SELECT

## Importando banco de dados

* Pelo terminal, va na pasta do banco de dados, utilize o terminal do VS code;
* mysql -u root;
* CREATE DATABASE <nome>;
* USE <nome>;
* Source <nome>sql;

## A importância do SELECT

* A maioria das queries em um banco de dados são de consulta;
* E é também o comando com mais variações;
* Para receber detalhados resultados, precisamos aprender todo o poder do SELECT;
* Desta maneira criaremos filtros avançados;

## Operadores

* Comparação (< , > , >=, <=, =) estes operadores vão filtrar dados;
* BETWEEN: Seleção entre intervalo;
* LIKE: Seleção por meio de algum padrão;
* IN: seleção entre um conjunto de valores específicos;

## Cláusula WHERE

* O WHERE será utilizados junto dos operadores;
* WHERE id = 10;
* Receberá apenas o resultado onde o id é 10;

## Utilizando o DISTINCT

* O DISTINCT vai selecionar apenas as variações de valores;
* SELECT DISTINCT < coluna> FROM <tabela>;

## Operadores lógicos

* Combinar a cláusula WHERE;
* AND, OR, NOT;
* WHERE id > 10 OR salario > 5000;
  + SELECT \* FROM salaries WHERE salary > 125000 AND from\_date > "1991-12-12";
  + SELECT \* FROM salaries WHERE salary > 125000 || from\_date > "1991-12-12";
  + SELECT \* FROM salaries WHERE NOT salary > 125000 AND NOT from\_date > "1991-12-12";

## Utilizando a ORDER BY

* A ORDER BY é uma instrução par ordenação de resultados;
* Pode-se utilizar de forma ascendente (ASC);
* E descendente (DESC;
* Utilizando após o WHERE;
* SELECT \* FROM salaries ORDER BY emp\_no ASC ou DESC;

## Utilizando a LIMIT

* A instrução LIMIT é outra forma interessante de limitar os resultados e tornar a consulta mais rápida;
* Podemos especificar o número de resultados retornandos;
* LMIT 15;
* Apenas os 15 primeiros resultados serão exibidos;

## Funções no SQL

* Funções são blocos de códigos reaproveit[aveis;
* Função MAX retorna o maior valor, de uma determinada coluna;
* SELECT MIN (coluna) FROM <tabela>;
* Retorna o menor valor de uma coluna;
* Função COUNT vai retorna o numero de valores que combinam com algum critério;
* Função AVG retorna a média de uma coluna;

## Operador LIKE, IN , BETWEEN

* O LIKE é utilizado sempre em conjunto do WHERE;
* SELECT \* FROM employees WHERE first\_name LIKE “%ber%”;
* O IN vai fazer uma busca por um conjunto de valores;
* SELECT \* FROM dept\_emp WHERE dept\_no IN (“d004”, “d006”);
* O BETWEEN retorna uma faixa de valores;

# Criando Constrains nas tabelas

* São regras que determinam como os campos serão preenchidos;
* NOT NULL = o campo não pode ser nulo;
* As constraints são adicionadas na criação da tabela geralmente, porém podemos alterar tabelas para adiciona-las;
* Estas regras são de grane utilidade pois ajudam a organizar e padronizar nosso projeto;

## Tipos de Constraints

* NOT NULL força um valor de uma coluna especifica não ser nulo;
  + Comando: nome VARCHAR(100) NOT NULL;
* UNIQUE: fagrante que todos os valores em uma coluna serão diferentes;
  + Caso de uso em e-mail;
* PRIMARY KEY: só pode ser adicionada em uma ´nica coluna da tabela, geralmente é o id;
  + O valor deve ser único e não pode ser nulo;
* AUTO INCREMENT: serve para adicionar a quantidade de um emtoto registro adicionado;
  + Muito utilizado na coluna id, já que ela é única e produto primário;
  + Não precisa se preocupar com este valor no INSERT;
* FOREIGN KEY: ligação de uma tabela a outra;
  + Uma tabela cadastramos o usuário e em outra o endereço dele;
  + O endereço foreign key que se refere ao id do usuário;
* INDEX : faz a consulta que envlva a mesma se tornar mais rápida;
  + As consultas que são mehoradas pelo INDEX são as com WHERE

# Unindo tabelas com JOINs

* São consultas que envolvem duas ou mais tabelas;
* Tabelas geralmente possuem relação entre si;
* 3 tipos de JOIN:
  + LEFT, RIGHT JOIN, INNER JOIN

## Tipos de JOIN

* INNER JOIN: vai resultar nas colunas que fazem relação entre as tabelas;
  + Pode-se determinar qual coluna resgatar após a instrução SELECT;
  + Utilizar a instrução ON para determinar as colunas que precisam ser iguais;
* LEFT JOIN: vai retornar todos os dados da tabela da esquerda e os necessários da direita;
* RIGHT JOIN: semelhante ao LEFT, porém tras as colunas da direita;

# Agrupamentos e Subquery

## Operador UNION

* O UNION é utilizado para combinar o resultado de dois ou mais SELECTS;
* As colunas precisam ter o mesmo nome;
* Os resultados serão agregados em um coluna só, porém podemos selecionar mais de uma coluna por vez;

## UNION ALL

* O UNION ALL é utilizado para combinar o resutado de dois ou mais SELECTS;
* AS colunas precisam ter o mesmo nome;
* Os resultados serão agregados em uma coluna só, porém podemos selecionar mais de uma coluna por vez;

## GROUP BY

* O GROUP BY serve para agruparmos colunas e checar quantidades de determinados elementos;
* Agrupando as colunas somando elas e temos o resultado de grupos;

## HAVING

* O HAVING é semelhante ao WHERE;
* Porém vamos utilizar eles com aggregate functions ( SUM, AVG, GROUP BY;

## Subquery

* Subquery é uma query dentro de outra query;
* Teremos mais comumente dois SLEECTS;

## EXISTS

* Serve para checar se existe algum registro em alguma subquery;

## ANY

* O ANY vai retornar os resultados que recebrm TRUE da subquery;
* Por meio de uma subquery receberemos apenas os resultados que condizem a uma condição;

# Funções de String

## O que são Funções¿

* As funções são blocos de códigos já definidos que podem ser reutilizados;
* Assim como a maioria das linguagens de programação já possuem funções prontas;
* Elas nos ajudam a atingir resultados de forma simples, que com apensas queries seriam atingidos de forma complicada;

## CHAR\_LENGTH

* A função CHAR\_LENGTH nos retorna o número de caracteres de uma string de uma determinada coluna;
* Ela leva um argumento que é a coluna que vamos avaliar;
* CHAR\_LENGTH (<nome\_da\_coluna>);

## CONCAT

* Concatena duas ou mais strings;
* Unir strings;
* CONCAT ("MYSQL”, “é”, ”bom”);
* CONCAT\_WS
  + Concatena duas ou mais strings, porem com um separador comum;
  + CONCAT\_WS(“-”, “Olá”, “-“);

## Format

* Vai formatar um numero com um numero de casas determinado por argumentos.
* FORMAT (número, 2);

## INSTR

* Retorna a posição do caracter ou strings que estamos buscando;
* INSTR(<strings>,”a”);

## LCASE

* Transforma todo o texto retornado com lower case, ou seja, caixa baixa;
  + LEFT: extrai quantidade de caracteres de uma strings, da esquerda para a direita;
  + LEFT(<strings>,5);
  + RIGHT estrai da direita para a esquerda;
  + RIGHT(<strings>,5);

## REPLACE

* Troca alguma parte de uma strings por outra enviada por parâmetro;
* REPLACE(<texto>,<antigo>,<novo>);
  + SUBSTR extrai uma strings a partir de dois índices;

# Funções NUMBERS

* CEIL: arredonda numero com casas decimais pra cima;
  + CEIL(<coluna>)
* FLOOR: arredonda para baixo;
  + FLOOR(<coluna>)
* MAX: retorna o maior valor de uma coluna;
  + Max(<coluna>)
* MIN: retorna o menor valor da coluna
  + MIN(<coluna>)
* SUM: retorna a soma de valores de uma coluna
  + SUM(<coluna>)

# Funções DATE

* ADDDATE: adicionar ou remove uma quantidade horas, dias, meses ou anos a uma data;
  + ADDDATE(<coluna>, <data para adicionar>)
* DATEDIFF: calcula a diferença de duas datas;
  + DATEDIFF(<data1>,<data2>)
* ADDTIME: adiciona ou remove um tempo a uma data;
  + ADDTIME(<coluna>,<tempo>)
* DATE\_FORMAT: formata uma data com um padrão indicado;
  + DATE\_FORMAT(<data>,<formato>)
* DAY: retorna o dia da data utilizada na função;
  + DAY(<data>)
* DAYOFWEEK: retorna o dia da semana de uma determinada data;
  + DAYOFWEEK(<data>)
* MONTH: extrai o mês de uma data;
  + MONTH(<data>)

# Relacionamento de tabelas

## O que são relacionamento¿

* SQL é uma linguagem de bancos de dados relacionais, ou seja, possuem relações;
* Estas relações servem para a separação de responsabilidades de tabelas;
* Hpa um link entre as tabelas que são o FOREIGN KEYS (FKs);

## Tipos de relacionamentos

* One to One: quando uma tabela possui uma conexão com outra e vice-versa;
  + No máximo um registro ligado a outro;
  + Ex: Estudante x Informações de contato;
  + Cada estudante pode ter apenas uma informação de contato e a informação de contato é apenas daquele estudante;
* One to Many: Quando uma tabela possui diversos registros em outra, porém a segunda só pode possuir uma conexão;
  + Uma tabela que possui vários relacionamentos com outra, mas o inverso não ocorre;
  + Ex: Cliente x Pedido;
  + Um cliente pode ter diversos pedidos na loja, porem um pedido é de um único cliente;
* Many to Many: quando duas tabelas podem ter conexões com diversos registros entre elas;

# Planejamento de banco de dados

## A importância do DB design

* Aplicações que tem desenho do banco já tem iniciam com uma documentação;
* E ainda uma organização prévia, que dita as regras do sistema e como ele funciona;
* O relacionamento entre entidades também é previsto;
* Erros podem ser resolvidos antes de acontecerem no código;

## Análise de requisitos

* É o planejamento e a definição do sistema;
* Onde, como o sistema deve funcionar é apresentado ou uma conversa com quem precisa do sistema é feita;
* Com base nestes dados e descrições vamos planejar o banco de dados;
* Temos um ponto de contato com a parte não técnica e técnica;
* Aqui também podem ser relatadas dificuldades técnicas e criam-se alternativas para possíveis problemas;

## Normalização

* É dividida em diversos níveis, algumas das premissas são:
  + Colocar chave primaria na tabela;
  + 1FN – colunas guardam um único valor;
  + 2FN – colunas que não pertecem ao tópico central da tabela
  + 3FN – deicar no banco de dados apenas valores que não são dependentes de outros, devem ir para outras camadas;

## Diagrama de entidade relacionais

* É um quadro onde se definem as tabelas e as relações entre si;
* Deixando o banco de dados visual a todos da equipe;
* Utilizado também para dar nome as tabelas e colunas;
* Além de também os tipos de dados da mesma;