**MySQL:**

**ORIGEM BANCO DE DADOS –** Origem década 1940 antes do surgimento dos computadores, dados eram guardados em papel e pastas. Analogia: Fichas (**registros**) > Pastas (**tabelas**) > Armários (**arquivos**). Originalmente os arquivos digitalizados eram **arquivos sequenciais** para serem achados era necessário percorrer desde o início o que gerava lentidão. Depois os registros passaram a ter **índices** e foram chamados de **arquivos de acesso direto.** Já na década de 1960 o departamento de defesa dos EUA no codasyl criou o COBOL.

**BANCO DE DADOS –** É um modelo de estrutura composta por:

1. **BASE DE DADOS:** Dados e estrutura de dados;
2. **SGBD:** Sistema Gerenciador de Banco de Dados;
3. **LINGUAGEM DE EXPLORAÇÃO:** É lima linguagem que permite acessar/consultar os dados;
4. **PROGRAMAS ADICIONAIS:** Gerencia de usuários, otimizados de dados;

**IBM –** Propôs criação dos modelos: (1) **modelo hierárquico >** (2) **modelo em rede.** Mas só mais tarde foi criado um modelo que permitia um bom **relacionamento** de dados, permitindo os dados de um registro em uma tabela acessar/consultar os dados de outro registro de outra tabela.

**MODELO RELACIONAL –** Permite percorrer os relacionamentos que compõem a estrutura de dados que estão relacionados por meio da linguagem de exploração.

**SQL (Structured Query Language) –** Linguagem de Consulta Estruturada. Com ela é possível dar comandos ao meio ambiente do BD e ele retornará uma **Query** uma resposta a uma solicitação. A SQL foi **padronizada** pela **ANSI** e ISSO**.**

**Exemplos comerciais (pagos):** Oracle, IBM, dBase, SQLServer.

**Exemplos comerciais (gratuitos):** MySQL, MariaDB (fork do MySQL), FireBird, PostgreSQL.

**MySQL 1994 (Suécia) –** Linguagem de exploração simples gratuita (e livre open source), compatível com SQL e baseada no modelo relacional. Em 2007 San comprou MySQL e 2009 foi comprada pela Oracle.

Empresas que usam MySQL: Nasa | Google | Wikipedia | Adobe | Cisco | Ebay | Claro, Tim, OI e Vivo | Bradesco | Forças Armadas.

**MySQL – Structured Query Language –** Instruções.

* **DDL (Data Definition Language):** Linguagem de Definição

(criar bd, tabela, alterar) comandos de **definição** da **estrutura de banco de dados**;

* **DML (Data Manipulation Language):** Linguagem de Manipulação

(incluir, exclui, alterar) comandos de **manipulação dos dados;**

* **DQL:** Linguagem de solicitação;
* **DCL:** Linguagem de Controle

Quais usuários tem acesso, quais comandos eles podem executar;

* **DTL:** Linguagem de transações;

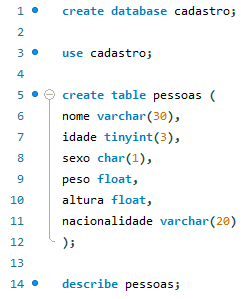
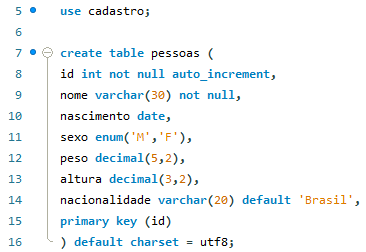
**FERRAMENTAS: Wampserver (para Windows) / XAMPP** (**Cross-plataforma**) usa MySQL junto com PHP. E **MySQL Workbanch:** ferramenta para acessar oambiente para desenvolvimento MySQL. **W**indows **/ X**(qualquer) **A**pache **M**ySQL **P**HP

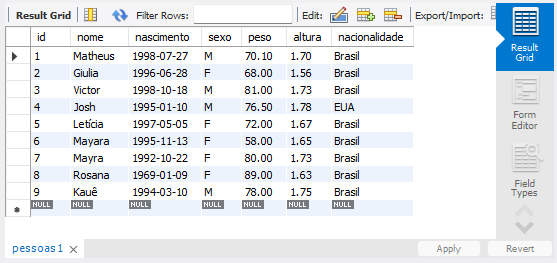
**OBJETIVO DO BANCO DE DADOS –** Registrar instâncias separadas de coisas que tem características semelhantes (valores individuais).

* BANCO DE DADOS: **Conjuntos de Tabelas**. são **coleções de tabelas** de dados de características separadas, mas que estão localizados em locais específicos (tabelas).
* TABELAS: **Conjuntos de Registros**. são locais específicos dentro do banco de dados que guardam características semelhantes. As tabelas possuem **campos.**
* REGISTRO: são os dados agrupados dentro de uma tabela.

**TIPOS PRIMITIVOS MYSQL –** Em MySQL os dados são armazenados em DISCO e por isso precisam ser MUITO BEM DIMENSIONADOS (Tipo e tamanho).

1. **Numéricos: Inteiro** (TinyInt, SamllInt, Int, MediumInt, BigInt. Eles dependem da quantidade de bytes “tamanho”) | **Real** (Decimal, Float, Double, Real) | **Lógico** (Bit, Boolean)
2. **Data/Tempo:** Date | **Date** (data) **| DateTime** (datas, horas, infos) | **TimeStamp** | **Time** (somente hora) | **Year** (somente ano)
3. **Literais: Caractere** (Char é fixo e armazena excedente com “espaço”, Varchar variável “não usa excedente com espaço”) | **Texto** (TinyText, Text, MediumText, LongText são usados textos mais longos) | **Binário** (TinyBlob, Blob, MeidumBlob, LongBlob)| **Coleção** (Enum, Set
4. **Espaciais: Geometry | Point | Polygon | MultiPolygon** são tipos novos do MySQL que permitem guardar dados volumétricos.

**EXEMPLO DE CRIAÇÃO DE BD E TABELA (Simples x Complexa)**

****

**DIFERENÇA LÓGICA ENTRE PK E UNIQUE –** UNIQUE KEY representa um dado único e que não pode ser repetido, por exemplo, os cursos de uma graduação (não existe duas vezes o mesmo curso/matéria ainda que ele seja ministrado por professores diferentes). Já as PRIMARY KEY além de serem UNIQUE (únicas) ele também serve para identificar os registros.

**MANIPULANDO LINHAS / REGISTROS (TUPLAS) –** Tudo oque aparecer nas linhas são registros, tudo o que aparece em colunas são campos.

\***Um comando** manipula uma linha por vez, mesmo manipulando colunas diferentes

**EXPORTAÇÃO (BACKUP DUMP) DE BANCO DE DADOS –** Menu > Server > Data Export (Schema/Table) > Schema Objects. O **dump** é um arquivo de scripts MySQL que contém todos os dumps, comandos de criação do Banco de Dados

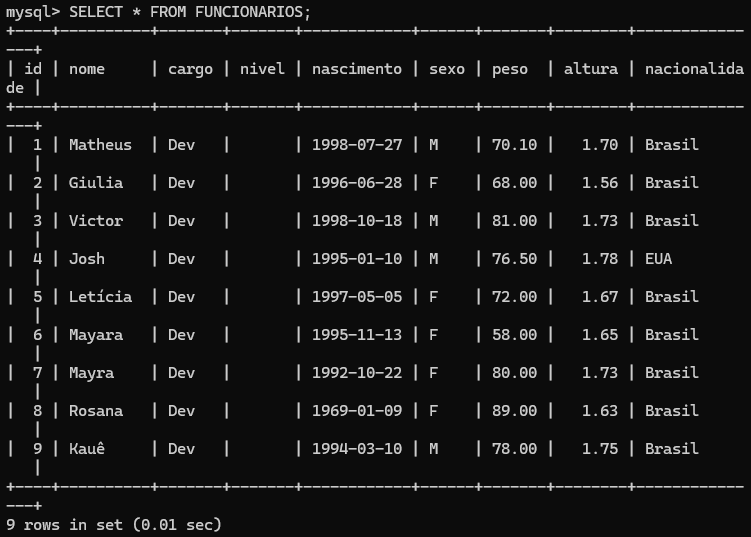
**IMPORTAÇÃO DE (BACKUP DUMP) DE BANCO DE DADOS -** Menu > Server > Data Import (Schema/Table) > Schema Objects.

**PHPMyAdmin –** Workbanch e PHPMyAdqmin são ferramentas para usar banco de Dados, porém não são banco de dados.

* **Banco** **de** **dados**: conjunto de dados organizados (tabelas, registros, índices e relacionamentos);
* **SGBD**: Software que permite criar, consultar e manter banco de dados;
* **Servidos**: é a máquina (física/virtual)( onde o SGBD roda e armazena os bancos de dados;

**MySQL Comand Line (terminal preto) comandos (password):** muitos dos comandos que funcionam no workbanch também funcionam no terminal MySQL. Todos os **dados cadastrados** estarão em um **servidor** MySQL, portanto independente da ferramenta usada (Workbanch, Comand Line ‘terminal’, PHPMyAdmin) você conseguirá acessar.

É possível **exportar e importar** usando a ferramenta PHPMyQdmin



* show databases;
* use noma\_tabela;
* status; (mostra qual o bd corrente)
* show tables;

**SELECT (IMPORTANTE)-**

**COMANDOS MYSQL:**

* **CREATE DATABASE NOME\_TABELA;** Cria um bd.
* **USE NOME\_TABELA;** Seleciona tabela para edições
* **CREATE TABLE IF NOT EXISTS NOME\_TABELA;** Cria uma tabela dentro do BD selecionado
* **DESCRIBE NOME\_TABELA;** Mostra/Descreve em detalhes uma tabela
* **DROP DATABASE NOME\_BD;** Exclui um BD
* **DEFAULT CHARACTER SET UTF8;** Definição de caracteres
* **DEFAULT COLLATE UTF8\_GENERAL\_CI;** Definição de caracteres
* **CONSTRAINTS;** São parâmetros / regras em SQL.
* **NOT NULL;** Torna obrigatório oi preenchimento do campo.
* **DEFAULT;** Define um padrão caso não seja preenchido, exemplo: default ‘Brasileiro’.
* **AUTO\_INCREMENT;** Deixa automático o cadastro de um campo de forma crescente e unitária.
* **ENUM (‘M’, ‘F’);** Define uma coleção de uma forma mais rígida, exemplo: F (feminino) e M (masculino).
* **PRIMARY KEY( );** Define um campo como chave primária (garantindo a ‘tupla’ não repetição).
* **DECIMAL (5,2);** Define como número decimal sendo o primeiro número o total de casas e o segundo número a quantidade de casas depois da vírgula.
* **INSER INTO NOME\_TABELA ( ) VALUES ( ‘ ‘);** Insere na tabela especificada os valores entre aspas simples.
* **INSERT INTO NOME\_TABELA VALUES ( ‘ ‘);** Insere na tabela especificada os valores entre aspas simples, na mesma ordem em que os campos forem definidos
* **SELECT \* FROM NOME\_TABELA;**
* **ALTER TABLE NOME\_TABELA;** Seleciona uma tabela para ser alterada

**ADD COLUMN NOME\_CAMPO VARCHAR(10);** Adiciona o campo na tabela na última posição

**ADD COLUMN NOME\_CAMPO VARCHAR(10) AFTER NOME\_CAMPO;** Adiciona o campo na tabela na posição imediatamente posterior ao campo mencionado

**ADD COLUMN NOME\_CAMPO VARCHAR(10) FIRST;** Adiciona o campo na tabela exatamente na primeira posição

**DROP COLUMN NOME\_CAMPO;** Exclui o campo (coluna)

**MODIFY COLUMN NOME\_CAMPO VARCHAR(20);** Capaz de alterar o TIPO, TAMANHO do tipo e as CONSTRAINTS do campo, mas não é capaz de mudar o nome do campo

**CHANGE COLUMN NOME\_CAMPO\_ANTIGO NOME\_CAMPO\_NOVO TIPO;** Muda o nome do campo (antigo 🡪 novo).

**RENAME TO NOME\_TABELA;** Renomeia a tabela inteira

**ADD PRIMARY KEY (NOME\_PK);** Adiciona uma PK escolhida dentro dos parênteses ().

* **UPDATE NOME\_TABELA**

**SET NOME\_CAMPO = ‘NOVO VALOR’**

**WHERE IDCURSO = ‘1’;** Muda o valor de uma linha (campo especificado) onde especifiquei

* **UPDATE NOME\_TABELA**

**SET NOME\_CAMPO = ‘NOVO\_VALOR’**

**WHERE NOME\_COLUNA = ‘VALOR’**

**LIMIT 1;** Trava todas as mudanças para que ela só aconteça em uma linha (SEGURANÇA PARA NÃO ALTERAR OUTROS DADOS SEM QUERER)

* **DELETE FROM CURSOS**

**WHERE NOME\_CAMPO = ‘VALOR\_CAMPO’;** Deleta uma linha inteira da tabela

COMANDOS SELECT

* **SELECT \* FROM NOME\_TABELA;** Seleciona todas as tuplas (registros) e exibe a tabela inteira
* **SELECT \* FROM NOME\_TABELA ORDER BY NOME\_CAMPO;** Coloca a tabela inteira de acordo com ordem alfabética do campo nome
* **SELECT \* FROM NOME\_TABELA ORDER BY NOME\_CAMPO DESC;** Coloca a tabela inteira de acordo com ordem alfabética (AO CONTRÁRIO) do campo nome
* **SELEC CAMPO\_1, CAMPO\_2, CAMPO\_3 FROM NOME\_TABELA**
* **SELEC \* FROM NOME\_TABELA**

**WHERE CAMPO\_3 = ‘VALUE’**

**ORDER BY CAMPO\_2;** Seleciona tudo, onde o campo 3 tiver determinado valor ele exibirá os registros do campo 2.

* **SELECT \* FROM NOME\_TABELA WHERE CAMPO BETWEEN X AND Y;**
* **SELECT \* FROM NOME\_TABELA WHERE CAMPO IN (VALOR\_1, VALOR\_2, VALOR\_3);** seleciona tudo que está compreendido em valores específicos

**DDL (Data Definition Language):** create database | craate table | alter table | drop table (apaga dados e estrutura da tabela)

**DML (Data Manipulation Language):** insert into | update | delete | truncate table (apaga apenas os dados da tabela)

**DQL (Data Query Language):** select