

# Introdução SQL



**PROF. ME. JOICE WOLFRANN**

# O que ?



- SQL é uma linguagem de programação usada por quase todos os bancos de dados relacionais para consultar, manipular e definir dados e fornecer controle de acesso.
- O SQL foi desenvolvido pela primeira vez na IBM nos anos 1970, com a Oracle como principal contribuinte, o que levou à implementação do padrão SQL ANSI;
- SQL estimulou muitas extensões de empresas como IBM, Oracle e Microsoft. Embora o SQL ainda seja amplamente usado hoje em dia, novas linguagens de programação estão começando a aparecer.

# Evolução

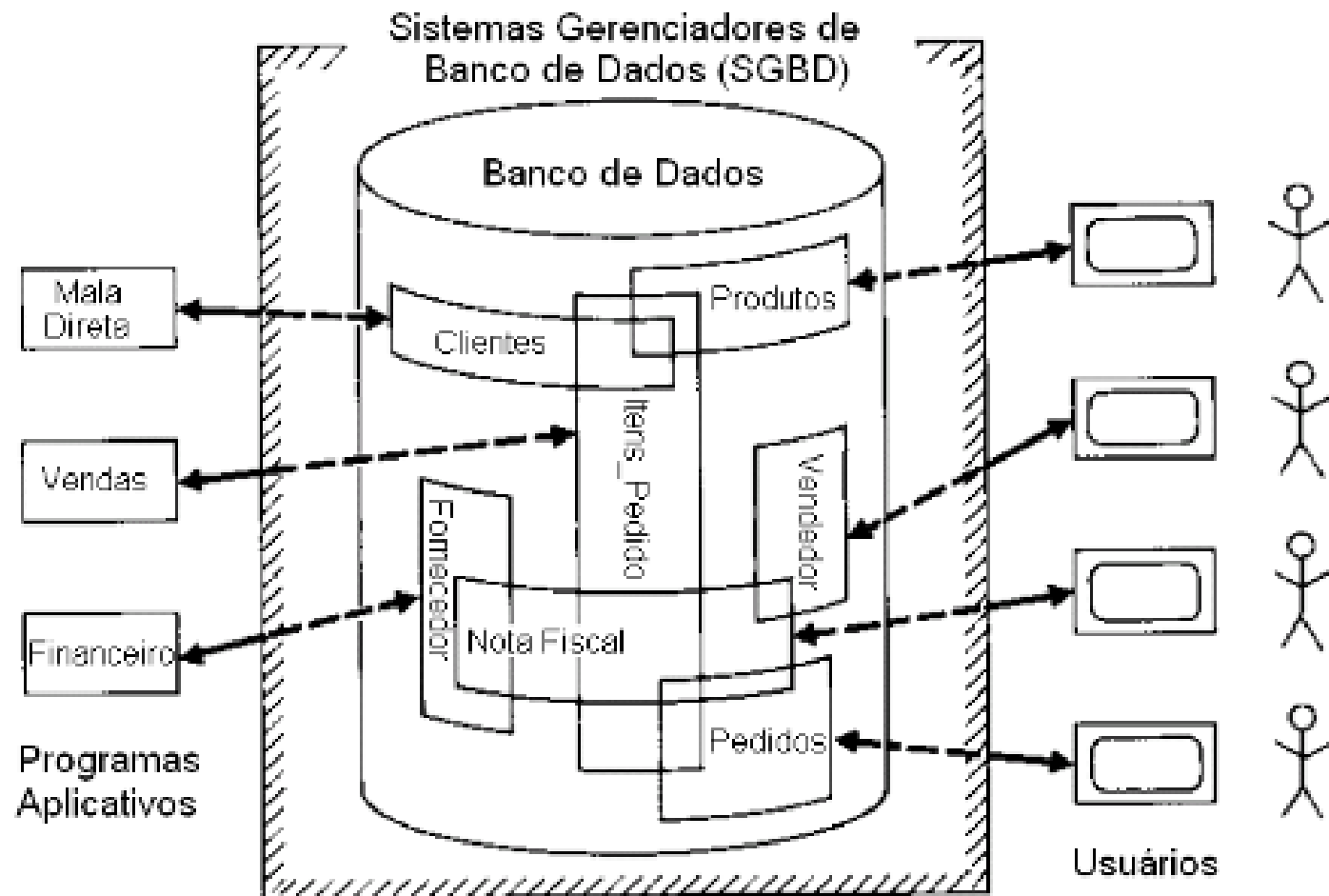


- O primeiro Sistema Gerenciador de Banco de Dados (SGBD) comercial surgiu no final de 1960 com base nos primitivos sistemas de arquivos disponíveis na época, os quais não controlavam o acesso concorrente por vários usuários ou processos.
- Os SGBDs evoluíram desses sistemas de arquivos de armazenamento em disco, criando novas estruturas de dados com o objetivo de armazenar informações.

# Evolução



- Com o tempo, os SGBD's passaram a utilizar diferentes formas de representação, ou modelos de dados, para descrever a estrutura das informações contidas em seus bancos de dados.
- Atualmente, os seguintes modelos de dados são normalmente utilizados pelos SGBD's: **modelo hierárquico, modelo em redes, modelo relacional (amplamente usado) e o modelo orientado a objetos.**



*Representação simplificada de um SGBD*

# Introdução



## Definição

### DDL

Data Definition Language

CREATE

ALTER

DROP

RENAME

TRUNCATE

COMMENT

## Consulta

### DQL

Data Query Language

SELECT

## Manipulação

### DML

Data Manipulation Language

INSERT

UPDATE

DELETE

MERGE

## Controle

### DCL

Data Control Language

GRANT

REVOKE

## Transação

### TCL

Transaction Control Language

COMMIT

ROLLBACK

SAVEPOINT

SET TRANSACTION

@josejuniordev

# Linguagem de Definição de dados



# DDL



- DDL ou Data Definition Language (Linguagem de Definição de dados) permite ao usuário definir as novas tabelas e os elementos que serão associados a elas.
- É responsável pelos comandos de criação e alteração no banco de dados, sendo composto por três comandos: **CREATE**, **ALTER** e **DROP**.



# DDL



- Uma vez compilados, os parâmetros DDL são armazenados num conjunto de arquivos denominado dicionário de dados (ou catálogo).
- O dicionário de dados contém os metadados (dados a respeito das estruturas de armazenamento).
- O SGBD sempre consulta os metadados a cada operação sobre o banco de dados.
  - Por exemplo, um determinado programa precisa recuperar alguns campos (nome, CPF) de um arquivo de clientes. O SGBD irá verificar se os campos "nome" e "CPF" estão definidos para este arquivo. O interpretador DDL processa os comandos alimentados pelos DBAs na definição dos esquemas.

# Sql



- Um subconjunto de instruções SQL formam outra '**DDL**'. Estas declarações SQL definem a estrutura de um banco de dados, incluindo linhas, colunas, tabelas, índices e características específicas do banco de dados, tal como localizações de arquivos.
- Declarações DDL SQL fazem mais parte do [SGBD](#) e possuem grandes diferenças entre as variações da SQL.

# Declarações Create

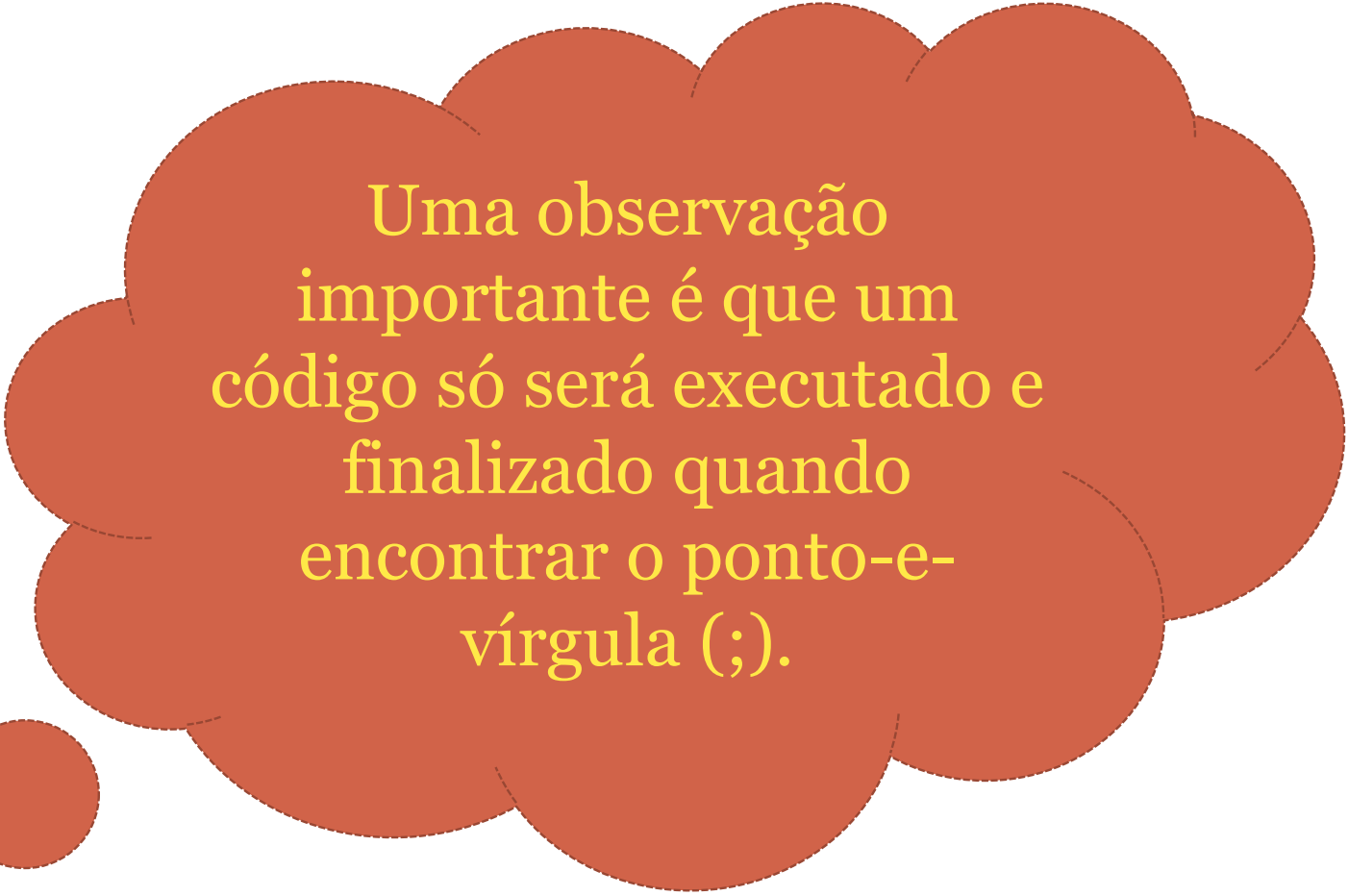


- Create - utilizada para construir um novo banco de dados, tabela, índice ou consulta armazenada. Uma declaração CREATE, em SQL, cria um objeto dentro do Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR).
- Os **tipos de objetos** que podem ser criados dependem de qual SGBDR está sendo utilizado, porém a maioria suporta a criação de tabelas, índices, usuários e banco de dados.
- Alguns sistemas (tais como PostgreSQL) suportam o comando CREATE, e outros comandos DDL, dentro de uma transação e portanto suportam rollback.



- O comando **CREATE DATABASE** é responsável pela criação de um novo banco de dados vazio:

```
CREATE DATABASE banco_teste;
```



Uma observação  
importante é que um  
código só será executado e  
finalizado quando  
encontrar o ponto-e-  
vírgula (;).

# USE



- Com o comando USE, estamos selecionando o Banco de Dados que iremos criar nossas tabelas. Após criar o banco de dados, ele não terá nenhuma tabela com atributos, portanto, o próximo passo é criar as tabelas e suas relações.
- **USE banco\_teste;**



- O comando **DROP** é utilizado para remoção de uma tabela ou do banco de dados por completo. Desta forma para remover um banco de dados por completo, basta inserirmos o seguinte comando:

```
DROP DATABASE banco_teste;
```



# Tipos de dados



Os tipos de dados existentes na linguagem SQL variam de acordo com a versão e fabricante.

A primeira versão, surgida por volta de 1970, não possuía tipos de dados para armazenamento de informações multimídia como som, imagem, vídeo; tão comuns nos dias de hoje.

A maioria dos Sistemas Gerenciadoras de Banco de Dados incorporou esses novos tipos de dados às suas versões da linguagem SQL. Os tipos apresentados abaixo fazem parte do conjunto de tipos do padrão ANSI 92 da linguagem SQL.



# Tipos de dados



Para dados do tipo	Tipo definido	Tamanho
Caracteres	Char(n), varchar (n)	Armazena até n bytes
Numérico exato	Decimal (p,e) ou numeric (p, e)	Depende
Número aproximado	Float, real	8 bytes e 4 bytes
Número inteiro	Int, smallint, tinyint	4 bytes, 2 bytes e 1 byte
Data e hora	Date, Time, smalldatetime	8 bytes, 4 bytes
Texto e imagens	Text, image, ntext	Variável
Monetário	Money, smallmoney	8 bytes, 4 bytes
Monetário	Money, smallmoney	8 bytes, 4 bytes

# Numéricos (inteiros e exatos):



- Smallint – Guarde os valores numéricos, em 2(dois) bytes, compreendidos entre o intervalo de -32768 a +32767.
- Int- (Integer) - Armazena valores numéricos, em quatro bytes binários, compreendidos entre o intervalo -2.147.483.648 a +2.147.483.647.
- Tinyint- usa 8 bits (1 byte) números não negativos de 0 a 255.

# Alfanuméricos ou caracteres:



- Os tipos Varchar (n) e Char (n) permitem o armazenamento em um campo alfanumérico de n caracteres, onde n deve ser menor ou igual a 254 caracteres.
- Por exemplo: o tipo char (30) reserva na memória 30 espaços fixo e o tipo varchar (30) permite uma variação no tamanho, até o limite de 30 caracteres.



- **Campo data**

- O tipo Date define um campo que irá guardar datas.

- **Campo Hora**

- Para guardar a hora o use o tipo Time.



- **Números aproximados**

- Para valores decimais sem exatidão é possível utilizar o tipo float, mas ele não tem precisão suficiente para vários dígitos.

- **Texto e imagem**

- Os tipos text, guarda texto. É possível armazenar imagens com o tipo image.

# Criando Tabelas



Nome da coluna		Coluna	Linha ou (registro)				
EMP	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Campos

# Criando tabelas



- A CREATE TABLE é usada para criar uma nova tabela em um banco de dados.

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
);
```



**CREATE TABLE** <nome-tabela>(<nome-coluna>  
<tipo-dado> [NOT NULL],  
**PRIMARY KEY** (<nome-coluna-chave>),  
**FOREIGN KEY** (<nome-coluna-chave-estrangeira>)  
REFERENCES  
<nome-tabela-origem> **ON DELETE** [RESTRICT]  
[CASCADE]  
[SET NULL]);



# Onde (ELMASRI e NAVATHE, 2005).:



- **< nome-tabela >** - Representa o nome da tabela que será criada.
- **<nome-coluna>** - será representado o nome dos campos que serão criados e deve ser colocado um campo após o outro, separado por vírgula.
- **<tipo-do-dado>** - Define o tipo e tamanho dos campos definidos para a tabela.
- **NOT NULL** – para chave primária, essa cláusula é obrigatória, pois indica que esse atributo deve ter um conteúdo.



**NOT NULL WITH DEFAULT** - Preenche o campo com valores pré-definidos, de acordo com o tipo do campo, caso não seja especificado o seu conteúdo no momento da inclusão do registro (ELMASRI e NAVATHE, 2011). Os valores pré-definidos são:

- Campos numéricos - Valor zero.
- Campos alfanuméricos - Caractere branco.
- Campo formato Date - Data corrente.
- Campo formato Time - Horário no momento da operação.



- **PRIMARY KEY**
- (nome-coluna-chave-primária): Define qual atributo será a chave primária da tabela. Caso ela tenha mais de uma coluna como chave, elas deverão ser relacionadas entre os parênteses.



- **FOREIGN KEY** (nome-coluna-chave-estrangeira)  
**REFERENCES** (nome-tabela origem):
- serve para definição de chaves estrangeiras, ou seja, os campos que são chaves primárias de outras tabelas. Na opção REFERENCES devemos escrever o nome da tabela de onde veio a chave estrangeira, ou seja, a tabela de origem onde o campo era a chave primária (HEUSER,2004).



- **ON DELETE**
- caso haja exclusão de um registro na tabela de origem e existe um registro correspondente nas tabelas filhas essa opção diz os procedimentos que devem ser feitos pelo SGBD (SILBERSCHATZ e SUDARSHAN, 2012).



As opções disponíveis são:

- **RESTRICT** - Opção default (padrão). Não permite a exclusão na tabela de pai (tabela de origem) de um registro, cuja chave primária exista em alguma tabela filha.
- **CASCADE** – Realiza a exclusão em todas as tabelas filhas que possuam o valor da chave que será excluída na tabela pai.
- **SET NULL** - Atribui o valor NULO nas colunas das tabelas filha que contenham o valor da chave que será excluída na tabela pai.

# Evitando valores nulos



- É muito comum definirmos campos que não podem conter valores nulos. Isto é, o seu preenchimento do campo é obrigatório para que se mantenha a integridade dos dados no sistema. Para evitar que em algum momento um campo de uma tabela possa conter valor nulo (null) deve-se utilizar a cláusula NOT NULL após a definição do campo.
- nome char(30) NOT NULL

# Evitando valores duplicados



- Podem existir situações onde o valor armazenado em um campo de um registro deve ser único em relação a todos os registros da tabela. Isto é, não pode haver dois registros com o mesmo valor para um determinado campo.
- Para implementar esta restrição de integridade deve-se utilizar a cláusula **UNIQUE** após a especificação de uma coluna.
- `rg decimal(9) NOT NULL UNIQUE`



# Evitando valores inválidos



- Existem situações onde um campo pode receber apenas alguns determinados valores. Para que o valor de um campo fique restrito a um determinado conjunto de valores, utilizase a cláusula CHECK.
- `sexo char(1) CHECK( sexo in ('M', 'F') )`
- `salario decimal(10,2) CHECK(salario>350)`



- Para facilitar a visualização de cada uma das nomenclaturas vejamos o exemplo para criar uma tabela de um funcionário:



```
CREATE TABLE Funcionario (  
  
    CODIGO_func INTEGER NOT NULL  
    AUTO_INCREMENT,  
  
    NOME_FUNC VARCHAR(20) ,  
  
    PRIMARY KEY(CODIGO_func)  
  
);
```



**create table funcionario**

```
(  
matricula decimal(5) NOT NULL,  
nome char(30),  
rg decimal(9),  
sexo char(1),  
depto decimal(5),  
endereço varchar(40),  
cidade varchar(20),  
salario decimal(10,2),  
PRIMARY KEY (matricula)  
);
```



- As restrições de integridade servem para garantir as regras inerentes ao sistema que está sendo implementado, prevenindo a entrada de informações inválidas pelos usuários desse sistema. Para isso, o Sistema de Banco de Dados deve possibilitar a definição de regras de integridade a fim de evitar a inconsistência dos dados que nele serão armazenados.

# Tarefa 01



- Criar o Banco de dados BD\_BANDA

<b>Banda</b>
<u>Codigo_banda</u>
nome_banda
02

<b>Musico</b>
<u>Codigo_musico</u>
Codigo_banda
Nome_musico
tempo_experiencia

# TAREFA 02



**Tabela de Fornecedores**

Nº Cadastro	Nome	Tel /Fax	Rua / Av.	nº	Bairro	Cidade	UF	CEP
1	5 de Agosto Distribuidor de Peças	12 3957-1256	R. Miguel Francisco	345	Jd. Emilia	São Paulo	SP	12321-390
2	Alternative Peças	11 3967- 0549	Av. Pensylvania	1500	Jd. Florida	São Paulo	SP	12578-555
3	Compel Distribuidora	11 2127-1297	R. Regina Lemes	49	Santa Maria	São Paulo	SP	12890-123
4	Dispemec Auto Peças	12 3632-3805	Av. Paraibuna	457	Centro	São José dos Campos	SP	12356-967

**Tabela de Peças**

Nº da Peça	Descrição	Carro	Preço Unitário	Nº do Fornecedor
137	Para-choque Dianteiro	Uno	R\$ 22,00	6490
138	Para-choque Traseiro	Uno	R\$ 38,00	1093
139	Para-choque Dianteiro	Gol G3	R\$ 45,00	7594
140	Para-choque Dianteiro	Gol G4	R\$ 32,00	88463
141	Para-choque Traseiro	Fiesta	R\$ 28,00	84732

# Tarefa 03



cliente

id_cliente	nome	data_nasc	sexo
1	José	1978-04-21	m
2	Maria	1980-10-17	f
3	João	1995-08-12	m
4	Pedro	1990-03-18	m

dvd

id_dvd	titulo	valor	id_genero
1	O Grito	8,00	1
2	Velozes e Furiosos 16	8,50	3
3	O Berro	6,00	1
4	Deja ir	6,50	2
5	Transformers 8	10,00	3

aluguel

id_cliente	id_dvd	data_aluguel	hora_aluguel	valor
2	4	2015-04-24	09:11	6,00
2	3	2015-04-24	09:11	9,00
1	5	2015-04-26	15:50	9,00
4	2	2015-04-27	13:45	9,50
3	3	2015-04-28	10:25	6,00

genero

id_genero	descricao
1	Terror
2	Suspense
3	Ação
4	Romance