



SÃO  
PAULO  
TECH  
SCHOOL



# **BD – BANCO DE DADOS**

Continuação Conceitos

# SQL – Structured Query Language

- Desenvolvido no início dos anos 1970, pelo Departamento de pesquisas da IBM
  - Interface para o sistema de banco de dados relacional System R
  - Inicialmente chamava-se SEQUEL (Structured English QUERy Language)
  - A partir de 1977, passou a ser chamada de SQL
  - Query = consulta, em inglês

# SQL – Structured Query Language

- Em 1986, o Instituto Nacional Americano de Padrões (ANSI) juntamente com a ISO (International Standards Organization) publicaram o padrão de linguagem SQL-86 ou SQL-1
  - Linguagem padrão adotada para Bancos de Dados Relacionais
  - Os vários SGBDs relacionais passaram a utilizar SQL
- A SQL-86 passou por revisões:
  - SQL-92 ou SQL-2, em 1992
  - SQL-99 ou SQL-3, em 1999
  - SQL-2003, em 2003

# SQL – Categorias de instruções

- DDL (relacionado à linguagem SQL)
  - Data Definition Language – grupo de instruções do SQL para criar tabelas, alterar a estrutura das tabelas ou eliminar tabelas.
    - Instruções CREATE, ALTER, DROP
- DML (relacionado à linguagem SQL)
  - Data Manipulation Language – grupo de instruções do SQL para criar manipular as tabelas, ou seja, para inserir dados, atualizar os dados, excluir dados, consultar dados
    - Instruções INSERT, UPDATE, DELETE, SELECT

# SQL – Categorias de instruções

- DCL (relacionado à linguagem SQL)
  - Data Control Language - lidam principalmente com os direitos, permissões e outros controles do sistema de banco de dados.
    - **GRANT** - concede privilégios de acesso do usuário ao banco de dados.
    - **REVOKE** - retira os privilégios de acesso do usuário dados usando o comando GRANT.

# Criando um banco de dados (Schema)

- Comando:  
**CREATE DATABASE** nome-do-banco;
- No MySQL Workbench, os bancos de dados aparecem na parte esquerda da tela, na janela Schemas
- As tabelas são criadas dentro de um banco de dados. Para isso, devemos selecionar o banco de dados que queremos utilizar

# Selecionando um banco de dados (Schema)

- Comando:  
**USE** nome-do-banco;
- No MySQL Workbench, é possível selecionar o banco apenas dando um duplo clique com o mouse em cima do nome do banco na janela SCHEMAS



# Criando uma tabela

- Comando:

```
CREATE TABLE nome-da-tabela (  
    nome-campo1  tipo-campo1,  
    nome-campo2  tipo-campo2,  
    .....  
    nome-campoN  tipo-campoN  
);
```

- A tabela será criada dentro do banco de dados selecionado, com os campos definidos no comando.
- No comando acima, é possível acrescentar restrições aos campos, como PRIMARY KEY, etc.

## Criando uma tabela (exemplo)

```
CREATE TABLE Aluno (  
    ra CHAR(8) PRIMARY KEY,  
    nome VARCHAR(40),  
    bairro VARCHAR(40),  
    email VARCHAR(80)  
);
```

- O comando acima criará a tabela Aluno, com 4 campos
- O campo ra terá um valor de no máximo 8 caracteres e esse campo será a chave primária da tabela.
- Os valores dos campos nome, bairro e email serão caracteres.

## Criando uma tabela (exemplo)

```
CREATE TABLE Aluno (  
  ra CHAR(8) PRIMARY KEY,  
  nome VARCHAR(40),  
  bairro VARCHAR(40),  
  email VARCHAR(80)  
);
```

- **EXCEÇÃO:** Regra de negócio, campo ra terá um valor de no máximo 8 caracteres e esse campo será a chave primária da tabela. Como boa prática a chave primária é do tipo inteiro (INT).
- Exemplo: id INT PRIMARY KEY

## Campos com valores caracteres

- Campo com valores caracteres como o nome, o bairro e o email podem ser definidos com o tipo CHAR ou VARCHAR
- Diferença entre CHAR e VARCHAR:
  - Quando se define que o nome é CHAR(10), então todos os campos desse tipo terão 10 caracteres, mesmo que o nome inserido tenha menos do que 10 caracteres
    - Ex: 'Bruno' será armazenado como 'Bruno '
    - O campo nome terá 5 espaços em branco para completar 10 caracteres
  - Quando se define que o nome é VARCHAR(10), então todos os campos terão no máximo 10 caracteres. Se o nome inserido tiver menos do que 10 caracteres, o campo conterá apenas os caracteres inseridos
    - Ex: 'Bruno' será armazenado como 'Bruno'

# Visualizando ou listando os dados da tabela

- Comando:

**SELECT** \* **FROM** nome-da-tabela;



\* significa “**todas as colunas**”

- O comando acima exibe todos os dados de uma tabela
- Quando a tabela acabou de ser criada e ainda não tem dados, o comando exibirá apenas os títulos das colunas

## Inserindo dados na tabela

- Comando:  
**INSERT INTO** nome-da-tabela  
**VALUES** (dado-campo1, dado-campo2,..... dado-campoN);
- O comando acima vai inserir os dados dentro dos parênteses na tabela, preenchendo um novo registro (nova “linha”) na tabela
- A ordem que os dados devem aparecer dentro do parênteses deve corresponder à ordem da criação dos campos no comando CREATE TABLE

## Inserindo dados na tabela (exemplo)

- Exemplo:

**INSERT INTO** Aluno

**VALUES** ('01212999', 'Vivian' , 'Campos Elíseos',  
'vivian.silva@sptech.school');

- O comando acima vai inserir os dados correspondentes a um aluno, de ra= '01212999', nome = 'Vivian', bairro = 'Campos Elíseos', email = 'vivian.silva@sptech.school'
- Para valores do tipo varchar ou char (caracteres), é preciso aspas
  - Tanto aspas simples quanto aspas duplas são aceitas, mas **acostume-se a utilizar aspas simples**

# Inserindo dados de mais de uma linha na tabela

- Exemplo:

**INSERT INTO** Aluno **VALUES**

```
('01212998', 'Paulo', 'Consolação', 'paulo.souza@sptech.school'),  
( '01212997', 'Marcelo', 'Paraíso', 'marcelo.rosim@sptech.school');
```

- Pode-se inserir de uma só vez os dados de vários alunos, como no exemplo acima (inserção de 2 alunos)
- É preciso tomar cuidado para colocar entre parênteses os dados de cada aluno, na ordem em que foi criada a tabela Aluno, separados por vírgula.
- Cada conjunto de parênteses deve ser separado por vírgula.



## Visualizando os dados apenas de algumas colunas

- Exemplo 1:

**SELECT** nome **FROM** Aluno;

O comando acima exibe apenas a coluna nome da tabela Aluno.

- Exemplo 2:

**SELECT** nome, bairro **FROM** Aluno;

O comando acima exibe apenas as colunas nome e bairro da tabela Aluno.

- Exemplo 3:

**SELECT** bairro, ra **FROM** Aluno;

O comando acima exibe apenas as colunas bairro e ra da tabela Aluno, nessa ordem.

# Visualizando os dados apenas de algumas linhas

- Comando:

**SELECT** \* **FROM** nome-da-tabela **WHERE** condição;



pode ser \*

ou o(s) nome(s)  
da(s) coluna(s)

desejadas

desejada(s)



utiliza-se o **WHERE**  
para apresentar a condição  
para “filtrar” as linhas

- As linhas que satisfazem a condição (colocada após o WHERE) são exibidas pelo comando.

## Visualizando os dados apenas de algumas linhas

- Exemplo 1:

```
SELECT * FROM Aluno WHERE ra = '01212999';
```

O comando acima exibe os dados do aluno de RA 01212999

- Exemplo 2:

```
SELECT * FROM Aluno WHERE ra <> '01212998';
```

O comando acima exibe os dados dos alunos de RA diferente de 01212998

Obs.: o MySQL e o SQL Server aceitam também != como sinal de “diferente”, mas o padrão é <>

# Visualizando os dados apenas de algumas linhas

- Exemplo 6:

**SELECT** \* **FROM** Aluno **WHERE** nome **LIKE** 'J%';

O comando acima exibe os dados dos alunos cujo nome começa com J.

Quando se especifica um padrão como 'J%', utiliza-se o LIKE.

O sinal de % representa zero ou mais caracteres.

Dessa forma, esse comando procurará os nomes que tenham a primeira letra J e depois pode vir uma quantidade qualquer de caracteres, e não importa quais caracteres.

# Visualizando os dados apenas de algumas linhas

- Exemplo 7:

**SELECT** \* **FROM** Aluno **WHERE** nome **LIKE** '%l';

O comando acima exibe os dados dos alunos cujo nome começa termina com l.

O sinal de % representa zero ou mais caracteres.

Dessa forma, esse comando procurará os nomes que tenham uma quantidade qualquer de caracteres, e não importa quais caracteres, desde que no final tenha a letra l.

# Visualizando os dados apenas de algumas linhas

- Exemplo 8:

**SELECT** \* **FROM** Aluno **WHERE** nome **LIKE** '\_a%';

O comando acima exibe os dados dos alunos cujo nome tenha a letra a como segunda letra.

O sinal de \_ representa apenas um caractere.

Dessa forma, esse comando procurará os nomes que tenham um caractere qualquer, seguido da letra a e depois do a pode vir uma quantidade qualquer de caracteres, e não importa quais caracteres

## Visualizando os dados ordenados por outra coluna

- Quando executamos o SELECT, os dados são exibidos de forma ordenada pela coluna que é a chave primária da tabela.
- No exemplo da tabela Aluno, os dados são exibidos ordenados pelo RA, que é a chave primária.

**SELECT** \* **FROM** Aluno **ORDER BY** nome-da-coluna;

OU

**SELECT** \* **FROM** Aluno **ORDER BY** nome-da-coluna **ASC**;

O comando acima exibe os dados dos alunos ordenados pela coluna especificada, em ordem ascendente (do menor para o maior, ou em ordem alfabética)

- Se quiser que a ordem seja descendente:

**SELECT** \* **FROM** Aluno **ORDER BY** nome-da-coluna **DESC**;

## Visualizando os dados ordenados por outra coluna

- Exemplo 9:

```
SELECT * FROM Aluno ORDER BY nome;
```

O comando acima exibe os dados dos alunos ordenados pelo nome (em ordem ascendente – ou seja, em ordem alfabética).

- Exemplo 10:

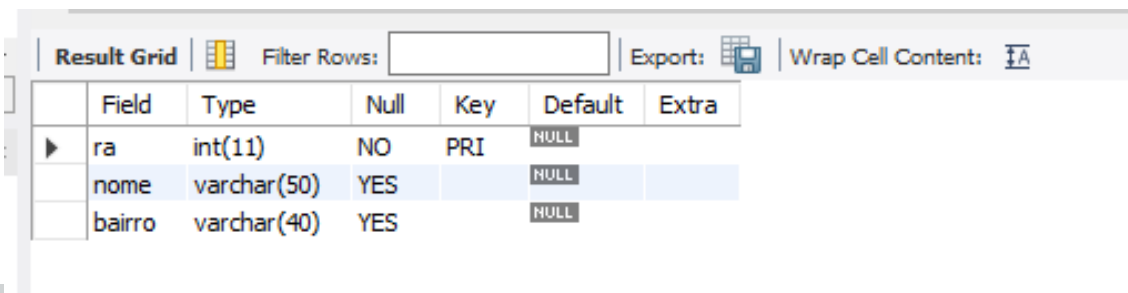
```
SELECT * FROM Aluno ORDER BY bairro DESC;
```

O comando acima exibe os dados dos alunos ordenados pelo bairro, em ordem descendente – ou seja, do 'Z' ao 'A'.



# Exibindo a descrição da tabela

- Comando:  
    **DESC** nome-da-tabela;  
    OU  
    **DESCRIBE** nome-da-tabela;
- Esse comando exibe uma descrição da tabela
- Exemplo:  
    **DESC** Aluno;
- A execução desse comando produz esse resultado:



	Field	Type	Null	Key	Default	Extra
▶	ra	int(11)	NO	PRI	NULL	
	nome	varchar(50)	YES		NULL	
	bairro	varchar(40)	YES		NULL	

## Excluindo a tabela

- Comando:  
**DROP TABLE** nome-da-tabela;
- Exemplo:  
**DROP TABLE** Aluno;
- A execução desse comando excluirá a tabela Aluno do banco de dados.

# Excluindo o banco de dados

- Comando:  
**DROP DATABASE** nome-do-banco;
- Exemplo:  
**DROP DATABASE** BancoAluno;
- A execução desse comando excluirá o banco de dados (Schema) BancoAluno.

## Alterando o valor de algum dado já inserido

- Comando:

**UPDATE** nome-da-tabela **SET** coluna-a-ser-alterada = novo-valor  
**WHERE** condição;

- As linhas que satisfazem a condição (colocada após o WHERE) terão o valor alterado na coluna-a-ser-alterada, especificada no comando.

- Exemplo:

**UPDATE** Aluno **SET** bairro = 'Tatuapé' **WHERE** ra = '01212999';

O comando acima altera o valor do bairro do aluno de RA 01212999 para 'Tatuapé'

## Alterando o valor de algum dado já inserido

- Pode-se alterar o valor de mais de uma coluna numa mesma linha em um só comando.
- Exemplo:

```
UPDATE Aluno SET nome = 'Ana Maria', bairro = 'Tatuapé'  
                WHERE ra = '01212999';
```

O comando acima altera o valor do nome e do bairro do aluno de RA 01212999.

- **ATENÇÃO:** Se não for colocado a cláusula **WHERE**, todas as linhas da tabela serão afetadas por este comando.

(o MySQL tem uma proteção contra isso, mas ela pode ser desabilitada, e outros SGBDs não têm essa proteção)

## Excluindo uma ou mais linhas da tabela

- Comando:

**DELETE FROM** nome-da-tabela **WHERE** condição;

- As linhas que satisfazem a condição (colocada após o WHERE) serão excluídas da tabela

- Exemplo: **DELETE FROM** Aluno **WHERE** ra = '01212999';

O comando acima exclui da tabela a linha referente ao aluno de RA 01212999.

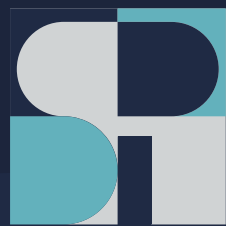
- **ATENÇÃO:** Se não for colocado a cláusula WHERE, todas as linhas da tabela serão afetadas por este comando.

(o MySQL tem uma proteção contra isso, mas ela pode ser desabilitada, e outros SGBDs não têm essa proteção)

# MySQL – Comandos UPDATE e DELETE

- O MySQL tem uma proteção que **não permite** que o **UPDATE** e o **DELETE FROM** sejam executados **sem que se coloque a cláusula WHERE**.
- O MySQL também obriga que se utilize na condição do **WHERE** **a coluna que é a chave primária da tabela**. Isso vale para os comandos **UPDATE** e **DELETE FROM**
- Isso já não ocorre no SQL Server (que será usado no Azure)
- No MySQL, essa proteção pode ser desabilitada, caso o usuário queira.
- Por isso, é bom tomar cuidado em **não esquecer** de colocar **WHERE** nos comandos **UPDATE** e **DELETE FROM**.

# Obrigada!



SÃO  
PAULO  
TECH  
SCHOOL



[vivian.silva@sptech.school](mailto:vivian.silva@sptech.school)