SISTEMAS DE BANCO DE DADOS 2

AULA 3

Revisão de SQL - *Structured Query Language*

MySQL

Vandor Roberto Vilardi Rissoli

APRESENTAÇÃO

- Linguagem SQL
- Linguagem de Definição de Dados
- Linguagem de Manipulação de Dados
- Referências

Linguagem em Banco de Dados

De forma geral, os Banco de Dados Relacionais implementam o padrão SQL e acrescentam características específicas ao padrão.

Cada SGBD procura incluir elementos que o possam diferenciar dos concorrentes e implementam também algumas possibilidades procedurais da SQL, por exemplo, a *Transact SQL* para o SQL SERVER e o *PL/SQL* para a ORACLE e outros.

Este material apresentará a SQL tão portável quanto possível, mas as instruções nele apresentadas poderão ser implementadas no SGBD **MySQL**.

Structured Query Language - SQL

A SQL surgiu no início da década de 70, por uma **iniciativa da IBM**. Ela tornou-se a linguagem mais popular para acesso aos bancos de dados, juntamente com a difusão dos SGBDs Relacionais.

Vários "dialetos" surgiram rapidamente e a ANSI padronizou o SQL, surgindo em 1986 SQL-86 e outros padrões que seguem 89, 92 e SQL-101...

Structured Query Language - SQL

CARACTERÍSTICAS DAS LINGUAGENS

A linguagem SQL é **declarativa** e os SGBDs Relacionais a têm como padrão. Suas características são originárias da **Álgebra Relacional**.

- Linguagem **Declarativa**: (ou **Não** Procedural): Linguagem que descreve O QUE se deseja realizar, SEM se preocupar em dizer COMO será feito. O sistema é que deverá determinar a forma de como fazer (não se descreve ou conhece como o sistema realizará tal processamento);
- Linguagem **Procedural**: Linguagem que fornece uma descrição detalhada de COMO um processamento será realizado, operando sobre um registro ou uma unidade de dados de cada vez.

Structured Query Language - SQL

A expressão **Programação Declarativa** é usada para discernir da programação praticada por meio das Linguagens de Programação Imperativas, sendo estas últimas sinônimos das **Linguagens Procedurais**.

Assim, em uma comparação simples, tem-se que:

"***um programa é declarativo se descreve o que ele faz e NÃO como seus procedimentos funcionam***".

ORACLE PL/SQL

procedural

MySQL

não procedural

Structured Query Language - SQL

LINGUAGEM

Um sistema de banco de dados relacional, geralmente proporciona dois tipos principais de recursos em sua linguagem SQL:

a) uma específica para manipular as estruturas do BD (DDL);

b) outra para expressar consultas e atualizações sobre o que é armazenado nessas estruturas (DML).

- **DDL** - *Data Definition Language*;
- **DML** - *Data Manipulation Language*.

Structured Query Language - SQL

- Linguagem de Definição de Dados (**DDL** - *Data Definition Language*) – uma estrutura de dados é representada por um conjunto de definições expressas por uma linguagem específica.
  - O resultado no uso da DDL constitui em um arquivo especial chamado de dicionário ou diretório de dados;
  - Um dicionário de dados é um arquivo de **metadados**.

→ **METADADOS** são dados a respeito de dados. Em um sistema de Banco de Dados, esse arquivo ou diretório é consultado antes que o dado real seja modificado (manipulado).

Structured Query Language - SQL

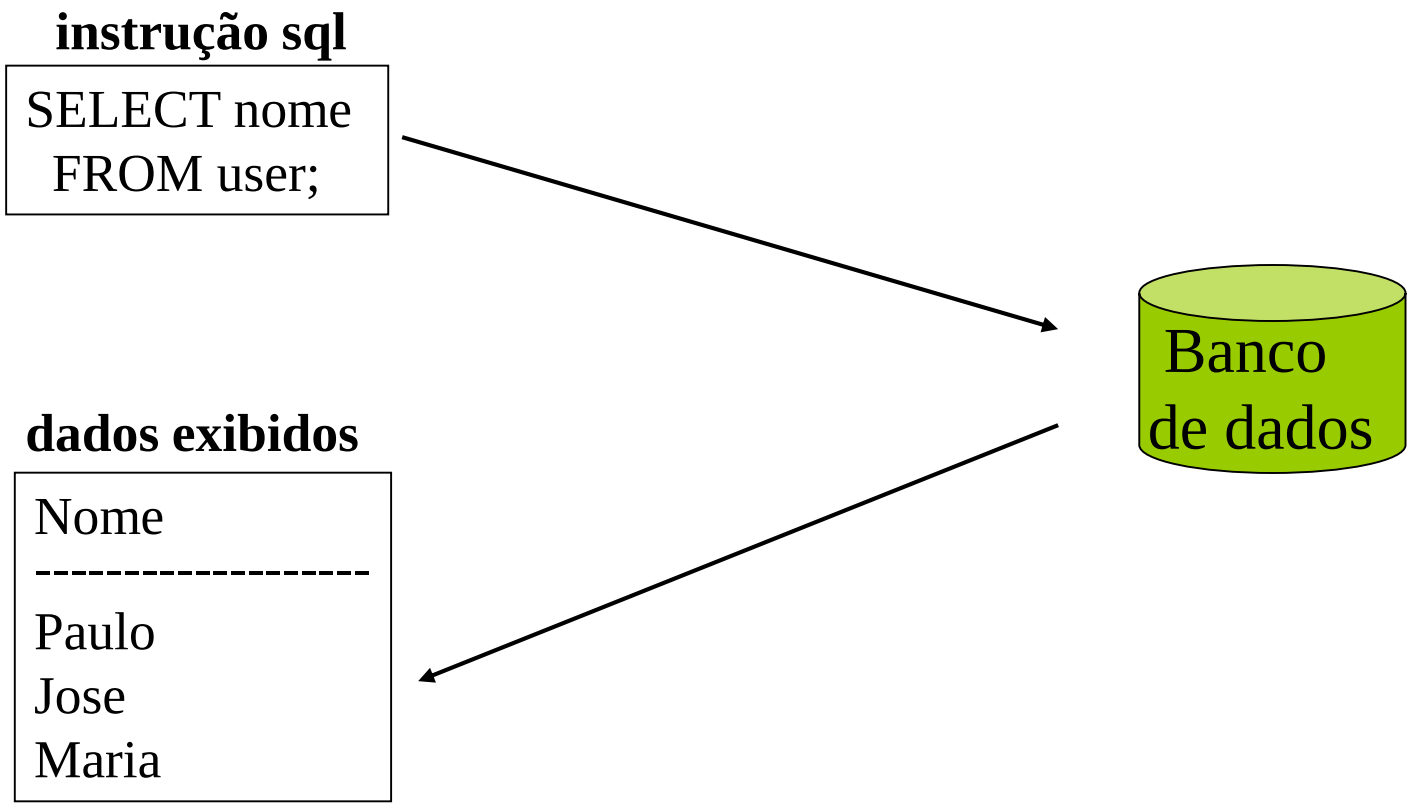
- Linguagem de Manipulação dos Dados (**DML** - *Data Manipulation Language*) – é a linguagem que viabiliza o acesso ou a manipulação dos dados de forma compatível ao modelo de dados apropriado. Por manipulação de dados entende-se:
  - Recuperação dos dados armazenados no BD;
  - Inserção de novos dados no BD;
  - Remoção e modificação de dados do BD.

- Linguagem de Consulta dos Dados (**SQL** - *Strutured Query Language*) – é parte de uma DML responsável pela recuperação de dados (pesquisa ou consulta).

→ Apesar da SQL indicar uma linguagem de consulta, ela possui recursos para definição de estruturas de dados, de modificação de dados no BD e de especificação de segurança.

Structured Query Language - SQL

A SQL consiste em uma linguagem **não procedural** (declarativa) que permite a interface básica para comunicação com o SGBD.



Structured Query Language - SQL

Um SGBD realiza alguns processos que podem ser efetuados por meio da linguagem SQL

- DEFINIÇÃO: criação descritiva do esquema(s) que atenderá as necessidades no BD;
- CONSTRUÇÃO: inserção das instâncias inicias no banco de dados;
- MANIPULAÇÃO: realização de consultas e atualizações sobre os dados decorrentes do dia a dia que usa este BD.

Structured Query Language - SQL

Será usada algumas ferramentas SQL para manipular o BD. As respectivas ferramentas são baseadas no SQL padrão, mas possuem algumas diferenças, por exemplo, recursos específicos da ORACLE que podem ser usados para se escrever relatórios e controlar o modo como a saída de tela ou papel é formatada difere da saída do MySQL.

Para começar a usar o BD é necessário estabelecer uma conexão com o SGBD que usa a SQL.

Nas características de um SGBD, lembre-se que existe uma preocupação com a restrição de **acesso NÃO autorizado**.

Structured Query Language - SQL

Criando uma Base de Dados

O estabelecimento de uma conexão com o SGBD MySQL só pode acontecer com o usuário que esteja cadastrado no BD.

Usando o SGBD MySQL será criada uma base de dados específica para a implementação do projeto. Essa base de dados receberá o nome de: ***baseDados***.

DATABASE (base de dados): Conjunto de registros de dados dispostos em estrutura regular que possibilita o seu armazenamento organizado produzindo informação.

CREATE DATABASE <nome da base de dados>;

SHOW DATABASES; -- lista as bases de dados existentes

DROP DATABASE <nome da base de dados>;

Structured Query Language - SQL

Estabelecendo a Conexão

Para se conectar ao BD MySQL disponível no computador local do laboratório use as opções do Sistema Operacional instalado para localizar o servidor MySQL disponível.

Procure a opção que indica o MySQL com acesso por linha de (*Command Prompt*) e a execute fornecendo os dados solicitados para um usuário já cadastrado nesse SGBD:

- confirme com seu professor qual o **usuário** disponível;
- solicite ao professor a **senha** individual desse usuário;
- pergunte ao professor qual será a base de dados (*database*) da atividade a ser realizada e execute a instrução abaixo:

use <nome da database>;

use baseDados;

Structured Query Language - SQL

Para iniciar a definição de um BD é necessário conhecer os seus tipos de dados válidos.

Os tipos usados na Descrição de Esquemas (DE) têm seus correspondentes em cada BD. Alguns do MySQL são:

- **int** – conjunto dos números inteiros (ver variações)
- **decimal** (*n*,*d*) – conjunto dos números reais
- **char(*n*)** – de 1 até 255 caracteres
- **varchar**(*n*) – caracteres variáveis até 4000
- **date** – data com ano, mês e dia no padrão
- **time** – horário com hora, minutos e segundos

→ *n* corresponde ao comprimento ou tamanho

→ *d* quantidade de dígitos decimais depois da vírgula

Visite o sítio virtual abaixo e confira todos os tipos do MySQL

https://dev.mysql.com/doc/refman/5.7/en/data-types.html

Structured Query Language - SQL

A primeira fase em qualquer SGBD começa com o uso de instruções **DDL** (*Linguagem de Definição de Dados*), pois serão por elas criados os recursos no BD.

Exemplo:

→ Como seria a relação que armazenaria todas as unidades da federação brasileira (os estados)?

Supondo esta necessidade elabora-se o esquema coerente para armazenar e manipular essas Unidades da Federação.

Relação: **ESTADOS**
sigla literal (2)
nome literal (20)

Unidades da Federação

sigla	**nome**
AC	Acre
AM	Amazonas
:	:
DF	Distrito Federal

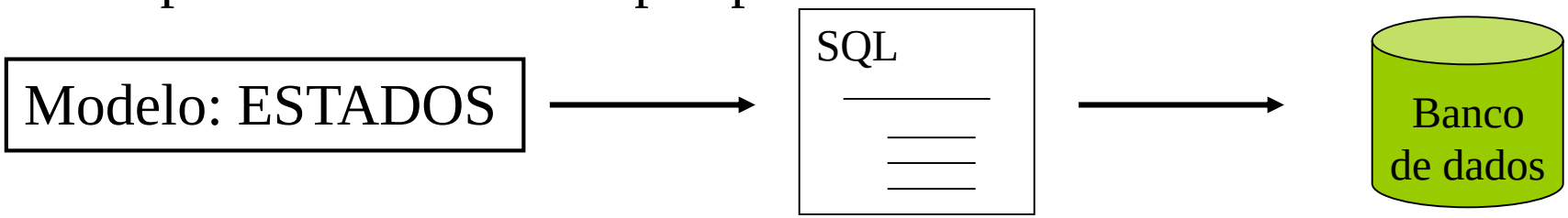
Structured Query Language - SQL

Após modelar e confirmar qual a relação que atende as necessidades, uma tabela será criada no SGBD.

Para criar esta relação (tabela) será usada a declaração SQL (DDL) que efetiva essa tabela no BD.

CREATE TABLE <nome da tabela> (
nome do atributo_1 <tipo dado>, -- tipo do atributo_1
nome do atributo_2 <tipo dado>) ;

- Na linha de comando do SQL os dois traços seguidos (--) indicam um comentário a partir deles até o final da linha;
- O ponto e vírgula encerra uma instrução deixando-a pronta para ser executada após pressionada a tecla <**ENTER**>.



Structured Query Language - SQL

Todos os ESTADOS possuem uma *sigla*, portanto sempre que se for cadastrar um novo estado ele deverá possuir uma *sigla*. Para que o SGBD controle esta restrição na tabela, não permitindo que seja informado um estado sem uma *sigla*, é necessário que o projeto da relação possua uma confirmação que a **obrigue**.

Caso nada identifique esta obrigatoriedade, será possível o armazenamento de um estado sem *sigla*.

A obrigatoriedade de um atributo é representada pela expressão **NOT NULL** (não nulo) que deve estar vinculada a todos os atributos que são obrigatórios em uma tabela.

Structured Query Language - SQL

A criação da tabela que representará as Unidades da Federação Brasileira será feita com a seguinte instrução:

```
CREATE TABLE ESTADOS  (
   sigla char(2)  NOT NULL,
   nome varchar(20) );   -- pressionando <ENTER>
```

Tabela criada. ← resultado da operação

Por meio deste comando foi criada a tabela ESTADOS que possui como domínio todas as Unidades da Federação Brasileira.

Structured Query Language - SQL

A expressão **NULL** significa que não existe valor, ou seja, não contém dados. O erro mais comum que se comete é a atribuição de um valor nulo em um atributo numérico.

O valor **nulo é diferente de zero**, pois zero é um valor, enquanto que nulo é a ausência de valor. Se ocorrer a operação 1 + NULL o resultado vai ser NULL.

"*Nunca use um **NULL** para representar um valor igual a zero em um atributo numérico. Se este atributo for usado para operações aritmé-ticas dê a ele o valor de zero e não **NULL***"

Structured Query Language - SQL

No MySQL as instruções *DESCRIBE* e *SHOW* são importantes. As duas instruções oferecem um resumo de dados sobre a tabela, também chamada de esquema, com suas colunas (atributos), tipos de dados e outras informações relevantes.

O primeiro comando pode ser usado de maneira abreviada (4 letras iniciais), obtendo-se o mesmo efeito.

```
mysql> desc estados;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| sigla | char(2)  | No   |     | NULL    |       |
| nome  | varchar(20) | No |   | NULL    |       |
+-------+----------+------+-----+---------+-------+
2 rows in set
```

Structured Query Language - SQL

A instrução que modifica o nome de uma relação é:

RENAME <nome_atual> TO <novo_nome>;

A instrução que remove uma relação (será detalhada a frente) consiste no comando a seguir:

DROP TABLE <nome_da_tabela>;

Exemplo:

RENAME TABLE ESTADOS TO UNIDADES_FEDERAIS;

Tabela renomeada. ← resultado

DROP TABLE ESTADOS;

ERRO 1017: Arquivo não encontrado. ← resultado

Código do erro

Número do erro

DROP TABLE UNIDADES_FEDERAIS;

Tabela eliminada. ← resultado

Structured Query Language - SQL

Uma importante instrução permite alterar as definições de uma tabela:

ALTER TABLE <nome_da_tabela>
<ação_ser_realizada> <dados_condizentes_ação>;

<ação_ser_realizada> => acrescentar, modificar, retirar recursos da tabela

<dados_condizentes_ação> => dados ou parâmetros coerentes com a ação que alterará a tabela

Exemplo:

ALTER TABLE ESTADOS ADD PAIS VARCHAR(100);

Tabela alterada. ← resultado

Structured Query Language - SQL

Pode-se aprender agora a manipular relações (tabelas) com DML, pois já se conhece os comandos e procedimentos básicos para a criação e remoção das relações (DDL).

Identificando o domínio das unidades da federação, pode-se armazenar seus possíveis dados. Nesse armazenamento serão guardados os valores de todos os estados da Federação Brasileira.

Com isso se inicia o processo de construção do BD, pois a definição projetou o BD mais coerente para o armazenamento dos dados necessários ao problema desejado.

Structured Query Language - SQL

Instrução INSERT

Esta instrução é usada para inserir tuplas em uma relação.

INSERT INTO <tabela>[(coluna_1,...,coluna_n)] VALUES (valor_1,...,valor_n);

Exemplo:

INSERT INTO ESTADOS(sigla, nome) VALUES('GO', 'Goiás');

ERRO 1146: tabela não existe. ← resultado

A tabela desejada **não existe mais**, pois foi apagada pelo comando DROP. Portanto crie a tabela ESTADOS novamente e depois insira os estados.

INSERT INTO ESTADOS(sigla, nome) VALUES('GO', 'Goiás');

1 linha criada. ← Resultado da 1ª forma

INSERT INTO ESTADOS(nome, sigla) VALUES('ACRE', 'AC');

1 linha criada. ← Resultado da 2ª forma

INSERT INTO ESTADOS VALUES('SP', 'São Paulo');

1 linha criada. ← Resultado da forma geral

Structured Query Language - SQL

Instrução DELETE

Esta instrução é usada para remover uma ou mais tuplas da relação, possuindo duas formas básicas:

DELETE FROM <tabela>;

ou

DELETE FROM <tabela> WHERE <condição>;

A primeira forma é obrigatória e apaga todos os dados da relação, enquanto que a segunda possui uma parte opcional, a partir do **WHERE**, que apaga somente os dados da tabela que atendem a uma condição (ou condições) imposta pela cláusula **WHERE**.

Structured Query Language - SQL

Exemplo:

DELETE FROM ESTADOS WHERE SIGLA = 'SP';

1 linha deletada. ← resultado

DELETE FROM ESTADOS WHERE NOME = 'ACRE';

1 linha deletada. ← resultado

DELETE FROM ESTADOS WHERE SIGLA = 'AC';

0 linhas deletadas. ← resultado

INSERT INTO ESTADOS VALUES('DF', 'BRASÍLIA');

1 linha criada. ← resultado

DELETE FROM ESTADOS;

2 linhas deletadas. ← resultado

Structured Query Language - SQL

Instrução UPDATE

Esta instrução permite a atualização de um ou mais atributos em uma tupla, ou em um grupo de tuplas de uma relação (tabela). O conteúdo de cada atributo será ajustado com a cláusula **SET**. Como na instrução DELETE existem duas formas para UPDATE:

UPDATE <tabela> SET <atributo> = <valor>;

ou

UPDATE <tabela> SET <atributo> = <valor>
WHERE <condição>;

A primeira forma é obrigatória e atualiza todos os dados da relação, enquanto que a segunda possui uma parte **WHERE** opcional, que atualiza somente os dados da relação que atendem a uma condição (ou condições) imposta pela cláusula **WHERE**.

Structured Query Language - SQL

Exemplo:

```
UPDATE  ESTADOS SET NOME = 'ACRE';
```

0 linhas atualizadas. ← resultado

Com a remoção de todas as tuplas no último DELETE, não é possível alterar nenhum dado, pois não existem dados na tabela.

```
INSERT INTO ESTADOS VALUES('BA', 'BAIA');
```

1 linha criada. ← resultado

```
UPDATE ESTADOS SET NOME = 'Bahia'
  WHERE SIGLA = 'BA';
```

1 linha atualizada. ← resultado

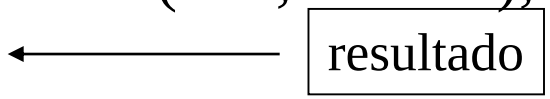
```
UPDATE ESTADOS SET NOME = 'ALAGOAS',
  SIGLA = 'AL' WHERE SIGLA = 'PA';
```

0 linhas atualizadas. ← resultado

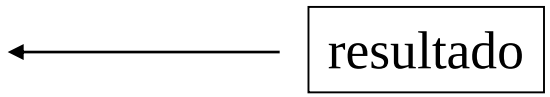
Structured Query Language - SQL

Exemplo:

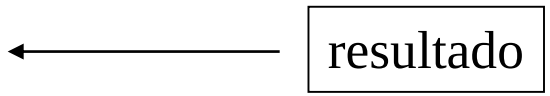
```
INSERT INTO ESTADOS VALUES('PA', 'PARA');
1 linha criada.
```



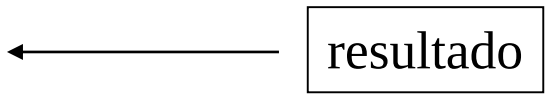

```
UPDATE ESTADOS SET NOME = 'ACRE'
  WHERE SIGLA = 'PA';
1 linha atualizada.
```




```
UPDATE ESTADOS SET NOME = 'ALAGOAS',
  SIGLA = 'AL' WHERE SIGLA = 'PA';
1 linha atualizada.
```




```
UPDATE CIDADES SET NOME = 'BRASÍLIA'
ERRO 1146: tabela não existe.
```



Structured Query Language - SQL

Instrução SELECT

Esta instrução é a essência do linguagem SQL. É por meio dela que se recupera (pesquisa) dados de um banco de dados. De modo simples, está se dizendo ao BD quais informações foram selecionadas para serem recuperadas.

Pode-se dividir esta instrução em quatro partes básicas:

- **SELECT** – seguido dos atributos que se deseja ver (obrigatório)
- **FROM** – seguido da origem dos dados no BD (obrigatório)
- **WHERE** – seguido das restrições de recuperação (opcional)
- **ORDER BY** – seguido da forma como os dados serão ordena dos (opcional)

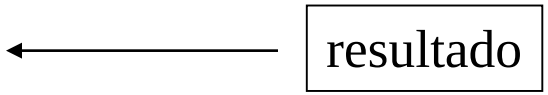
Structured Query Language - SQL

O símbolo asterisco (*) significa que todos atributos da relação informada deverão ser recuperados.

Exemplo: SELECT * FROM ESTADOS;

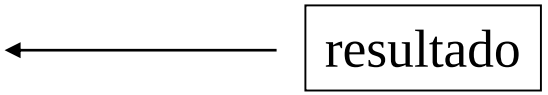
```
------------------------
| sigla |  nome        |
------------------------
| BA    | Bahia        |
| AL    | ALAGOAS |
------------------------
```

2 linhas apresentadas.



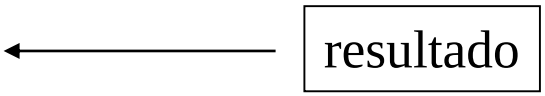
SELECT * FROM CIDADES;

ERRO 1146: Tabela não existe.



DELETE FROM ESTADOS WHERE NOME = 'ACRE';

2 linhas deletadas.



Structured Query Language - SQL

Consulta sem dados na relação e a inserção de dados:

```
SELECT * FROM ESTADOS;
não há linhas selecionadas.
```

resultado

```
INSERT INTO ESTADOS(sigla, nome) VALUES('PA', 'PARA');
1 linha criada.
```

resultado

```
INSERT INTO ESTADOS(nome, sigla) VALUES('Parana','PR');
1 linha criada.
```

resultado

```
INSERT INTO ESTADOS VALUES('AM','AMAZONAS');
1 linha criada.
```

resultado

```
SELECT NOME FROM ESTADOS;
```

resultado

```
----------------------
| nome               |
----------------------
| PARA               |
| Parana             |
| AMAZONAS           |
----------------------
3 linhas apresentadas.
```

Structured Query Language - SQL

Consulta sem dados na relação e a inserção de dados:

```
SELECT * FROM ESTADOS;
não há linhas selecionadas.
```

resultado

```
INSERT INTO ESTADOS(sigla, nome) VALUES('PA', 'PARA');
1 linha criada.
```

resultado

```
INSERT INTO ESTADOS(nome, sigla) VALUES('Parana','PR');
1 linha criada.
```

resultado

```
INSERT INTO ESTADOS VALUES('AM','AMAZONAS');
1 linha criada.
```

resultado

```
SELECT NOME FROM ESTADOS;
```

resultado

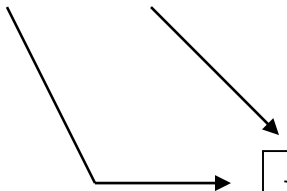
```
----------------------
| nome               |
----------------------
| PARA               |
| Parana             |
| AMAZONAS           |
----------------------
3 linhas apresentadas.
```

Structured Query Language - SQL

Pesquisando as informações atuais no banco tem-se:

```
SELECT SIGLA , NOME FROM ESTADOS;

SIGLA   NOME
--------  ----------------
AL      ALAGOAS
PR      BRASIL
AM      AMAZONAS
```



Especificando as colunas desejadas na ordem em que elas serão procuradas e apresentadas

A consulta de somente algumas tuplas pode ser conseguida por meio da cláusula **WHERE**. O uso desta cláusula é opcional e faz com que uma ou mais condições tenham que ser atendidas para que uma tupla seja recuperada.

Structured Query Language - SQL

A forma básica de uma instrução **SELECT** pode ser generalizada em:

SELECT <atributos> **FROM** <tabela> **WHERE** <condições>;

Após o **WHERE** são especificadas as cláusulas condicionais que restringirão as tuplas a serem recuperadas pela consulta desejada.

```
SELECT NOME FROM ESTADOS
        WHERE SIGLA = 'AL';
NOME
---------------
ALAGOAS
```

resultado

```
UPDATE ESTADOS SET NOME = 'BRASIL'
        WHERE NOME = 'AMAPA';
0 linhas atualizadas.
```

resultado

Structured Query Language - SQL

```
UPDATE ESTADOS SET NOME = 'BRASIL'
      WHERE SIGLA = 'AM';
1 linha atualizada.
```

← resultado

```
SELECT * FROM ESTADOS;
SI NOME
-- ---------------
AL  ALAGOAS
PR  BRASIL
AM BRASIL
```

← resultado

```
SELECT NOME, SIGLA FROM ESTADOS
      WHERE NOME = 'BRASIL';
NOME                 SIGLA
-------------------  --------
BRASIL               PR
BRASIL               AM
```

↑ resultado

```
SELECT * FROM ESTADOS
WHERE NOME='BRASIL' AND SIGLA='AM';
          SIGLA   NOME
          --------  -----------
          AM       BRASIL
```

Structured Query Language - SQL

Na cláusula **WHERE** serão utilizados alguns operadores de comparação e lógicos para que a condição seja elaborada e possa ser constatada com mais realidade.

Comparação	
Oper.	Significado
=	igual a
>	maior que
<	menor que
>=	maior ou igual que
<=	menor ou igual que

Lógico	
Oper.	Significado
AND	E lógico
OR	OU lógico
NOT	Negação

```
SELECT NOME FROM ESTADOS
    WHERE SIGLA = 'SP' OR NOME = 'ALAGOAS';
NOME
---------------
ALAGOAS
```

resultado

Structured Query Language - SQL

A cláusula das outras instruções de manipulação da informação também atendem as características dos operadores lógicos e de comparação. Então, corrija as outras duas tuplas que aparecem com o nome de ‘BRASIL’ para o correto nome, de acordo com a sua sigla já cadastrada.

```
UPDATE ESTADOS SET NOME = 'PARANA'
    WHERE SIGLA = 'PR';
1 linha atualizada.
```


resultado

```
UPDATE ESTADOS SET NOME = 'AMAZONAS'
    WHERE SIGLA = 'AM';
1 linha atualizada.
```


resultado

Agora, recupere somente as tuplas que sofreram alteração e verifique se elas estão corretas.

```
SELECT * FROM ESTADOS
    WHERE SIGLA = 'PR' OR SIGLA = 'AM';
```

resultado

```
SIGLA  NOME
-------  -----------
PR    PARANA
AM    AMAZONAS
```

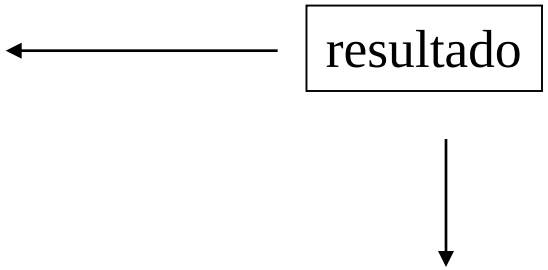
Structured Query Language - SQL

A apresentação dos dados recuperados pelo **SELECT** pode ser classificada (ordenada) de forma ascendente (crescente) ou descendente (decrescente).

Para isso se pode incluir a cláusula opcional **ORDER BY** no final da instrução **SELECT**.

```
SELECT NOME FROM ESTADOS
    ORDER BY NOME ASC;
```

```
NOME
------------------
ALAGOAS
AMAZONAS
PARANA
```




```
SELECT NOME FROM ESTADOS
    ORDER BY NOME DESC;
```

```
NOME
------------------
PARANA
AMAZONAS
ALAGOAS
```

Structured Query Language - SQL

Uma classificação com mais que um atributo pode ser feita com a especificação ordenada de quais os atributos a serem usados na classificação e a ordem de prioridade para cada um.

Os atributos devem ser especificados separados por vírgula e devem estar em ordem de prioridade, conforme sua especificação na cláusula **ORDER BY**, em que o primeiro atributo especificado será classificado primeiramente e o segundo, respeitando a classificação do primeiro, será classificado em seguida, dentro da primeira classificação.

Exemplo:

SELECT NOME FROM EMPREGADO WHERE SALARIO < 0
ORDER BY NOME, SALARIO; → Sem a especificação de ASC ou DESC a consulta é classificada de forma ASC, por definição (padrão)

↓

Primeira classificação será feita por nome e dentro da classificação por nome será feita por salário

Structured Query Language - SQL

Diante do avanço no estudo das instruções SQL, e vislumbrando a eficiência e agilidade dos profissionais envolvidos em trabalhar sobre as instruções SQL são empregadas boas práticas na elaboração dessas instruções, por exemplo:

```
SELECT idEmpregado, nome
  FROM EMPREGADO
 WHERE salario < 0
 ORDER BY nome, salario;
```

Note que a instrução acima possui cada uma de suas cláusulas em um linha, estando a primeira palavra-chave da cláusula de instrução alinha ao seu final (à direita).

Com isso a visão para compreensão da lógica envolvida é mais fácil e ágil aos profissionais da área.

Structured Query Language - SQL

Baseado no exemplo anterior, observe o comando DDL que criaria uma relação compatível com a consulta solicitada pelo **SELECT** anteriormente proposto.

```
CREATE TABLE EMPREGADO    (
   idEmpregado        int(8)   NOT NULL,    -- código funcional
    nome          varchar(40)    NOT NULL,    -- nome do empregado
    dtNascer    date    NOT NULL,    -- data de nascimento
    salario        decimal(7.2)   NOT NULL,    -- valor do seu salário
    cpf              int(11) -- número do CPF, se tiver (não é obrigatório)
);
```

Padrões no Identificador dos atributos

Identificador único => **id**Empregado (para identificadores)

Data de nascimento => **dt**Nascer (para datas)

Structured Query Language - SQL

Além dos comandos DDL e DML vistos até aqui, uma outra categoria de comandos chamada de TPL (*Transaction Proccess Language* – Linguagem de Processamento de Transação) compõe a SQL.

Os comandos desta categoria são responsáveis pela gravação definitiva no banco de dados das instruções DDL e DML, além da recuperação do banco de dados, caso algo aconteça de errado.

Várias das instruções DDL são de salvamento automático (***autocommit***), portanto, uma vez executadas elas já estão implantadas no banco de dados. Por exemplo a instrução CREATE TABLE.

Structured Query Language - SQL

Salvando as Realizações

Após realizar algumas instruções sobre o banco de dados é necessário solicitar que ele armazene efetivamente (grave) o resultado de suas solicitações feitas por meio das instruções executadas (transações).

A efetiva realização solicitada pelas instruções **INSERT**, **DELETE** e **UPDATE** precisam ser salvas no banco de dados. Para que as suas ações (ou realizações) sejam salvas no banco de dados, é necessária a execução do comando ***COMMIT***.

Por meio do comando ***COMMIT*** essas instruções DML são efetivadas (salvas) no banco de dados.

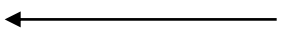
Structured Query Language - SQL

Salvando as suas realizações no BD será liberado todos os recursos exigidos até concluir a sua transação. Existe uma redução nos "**custos**" do acesso de outros usuários e a diminuição do seu tempo de espera.

Outra característica importante para estar salvando sempre o que você estiver realizando, principalmente as DML, diminui os riscos de perder algo que terminou de fazer com tanto cuidado no BD.

Exemplo: (após uma instrução DML correta executar o salvamento)

```
COMMIT;
Efetivação completada.
```



resultado

Structured Query Language - SQL

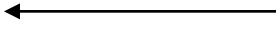
Durante a utilização do BD, pode acontecer a execução de uma instrução equivocada, ou problemas como a falta de energia no meio de uma transação.

A restauração da situação do BD, antes de um salvamento ser executado, é possível por meio do comando ***ROLLBACK***. Ele retrocede o BD para a situação que o BD estava após o último comando de salvamento (*COMMIT*) ser executado.

As instruções de **AUTOCOMMIT** não serão retornadas, pois assim que elas são executadas elas são salvas.

Exemplo:

```
ROLLBACK;
Restabelecimento completado.
```



resultado

Structured Query Language - SQL

Remover uma Tabela

Para se remover uma relação (tabela) do BD deve ser usada a instrução **DROP TABLE**.

DROP TABLE <nome da tabela>;

Por meio desta instrução a tabela deixará de existir no BD, sendo todos os dados contidos nessa tabela apagados. Similar a instrução **DROP DATABASE**, que apaga a base de dados com todas as tabelas e recursos existentes no BD.

Exemplo:

DROP TABLE ESTADOS;

Tabela eliminada. ← resultado

DROP TABLE CIDADES;

ERRO 1051: tabela desconhecida. ← resultado

Structured Query Language - SQL

Atributo de Auto Incremento na Tabela

A implementação de um atributo (coluna da tabela) de **auto incremento** permite que um número inteiro único seja gerado quando um novo registro for inserido no MySQL, nunca sendo repetido este número.

A palavra ou expressão chave **AUTO_INCREMENT** deve estar indicada em um único atributo da tabela. Seu valor inicial padrão é 1 e o incremento será de 1 em 1.

Porém, é possível definir um valor inicial diferente do padrão, assim como alterar o valor do auto incremento (o passo) para seguir a partir de um valor arbitrário que será indicado por uma instrução DDL (***alter table***).

Structured Query Language - SQL

Características do Atributo de Auto Incremento

- Ao inserir novos valores na tabela, não é necessário especificar o valor do atributo *auto-incremento;*
- Só é permitido usar um atributo de auto incremento por tabela, geralmente do tipo inteiro;
- Necessita também da restrição (*constraint*) NOT NULL, que será configurado automaticamente, mas deve ser incluída na documentação e no *script* que cria a tabela corretamente;
- Exige a criação de chave primária no atributo.

Structured Query Language - SQL

Exemplo:

```
CREATE TABLE EQUIPE    (
    idEquipe int(5)   NOT NULL AUTO_INCREMENT,
    nome     varchar(100)   NOT NULL,
 constraint equipe_PK PRIMARY KEY(idEquipe)
) ENGINE = InnoDB  AUTO_INCREMENT = 1 ;

INSERT INTO EQUIPE (nome) VALUES ('Flamengo');
    1 linha criada.
INSERT INTO EQUIPE VALUES (null, 'Vasco');
    1 linha criada.

SELECT * FROM EQUIPE;
```

resultado

resultado

conferindo resultado

Structured Query Language - SQL

Exemplo:

```
INSERT INTO EQUIPE VALUES('Fluminense');
    1 linha criada.
INSERT INTO EQUIPE VALUES(7, 'Botafogo');
    1 linha criada.

SELECT * FROM EQUIPE;

INSERT INTO EQUIPE (nome) VALUES('Palmeiras');
    1 linha criada.
```

resultado

resultado

confirindo resultado

Qual resultado (valor) será apresentado no atributo **idEquipe** dessa última inserção?

Structured Query Language - SQL

Outros bancos de dados (SGBD) empregam outras formas de controle sobre valores únicos e sequenciais relevantes as implementações relacionais em SGBD's.

Por exemplo:

ORACLE utiliza os objetos de banco de dados conhecidos como SEQUENCE;

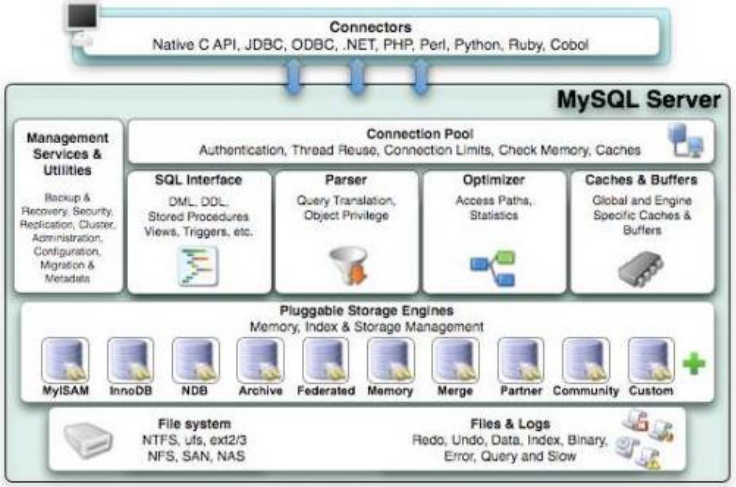
PostgreSQL usa também a SEQUENCE, mas com alguns outros recursos e sintaxes diferentes do ORACLE.

Structured Query Language - SQL

Opção ENGINE no MySQL

A **Engenharia de Armazenamento** (*Storage Engine*) são características modulares que podem ser atribuídas as relações de uma base de dados no MySQL.

Essas características, também chamadas de Motores de Armazenamento, são nativas do MySQL (MyISAM, Archive, InnoDB, Federated, Memory e CSV), mas outras personalizadas podem ser integradas a este SGBD



Structured Query Language - SQL

Exemplo:

Suponha que as tabelas criadas em outros SGBD sejam sempre ***Ferraris***, ou seja, tenham potente motor, grande velocidade e outros apetrechos de luxo.

Imagine também que exista a necessidade de cruzar uma avenida bem congestionada e você deverá escolher entre os veículos disponíveis qual será o mais rápido:

uma Ferrari ou **uma (excelente) Bicicleta**

No MySQL é possível escolher qual motor (*storage engine*) será usado de maneira coerente com a necessidade da atividade que será realizada. Assim, pode-se dirigir uma Ferrari (tabelas transacionais) ou pedalar uma bicicleta (não transacionais), conforme seja mais conveniente ao BD.

Structured Query Language - SQL

Quais são as principais características, recursos ou funcionalidades inerentes as **Engenharias de Armazenamento** (*Storage Engines*) do MySQL:

a) Capacidade Transacional: capacidade da tabela aceitar múltiplos acessos (de múltiplos usuários/aplicações), com colisão e travamento mínimos, sem que um usuário interfira com a operação do outro, além das propriedades ACID;

b) Meio de armazenamento: no MySQL a forma de armazenamento e uso das tabelas depende do motor escolhido;

c) Índices: de acordo com o motor têm-se diferentes tipos de índices;

d) Integridade Referencial: há motores que usam e outros que não usam e/ou respeitam a chave estrangeira (*foreign key*);

Structured Query Language - SQL

e) Tipo de bloqueio: capacidade de poder bloquear o acesso a um único registro (linha), vários registros ou a tabela inteira;

f) Cópia de segurança *on line* (BOL – *Backup On Line*): realizar *backup* sem parar o trabalho do BD e de seus usuários;

g) Reparação Automática (*Auto Recovery*): havendo falha no banco de dados (BD) alguns motores registrarão no LOG do erro que foi encontrado e consertado automaticamente.

Em resumo, *storage engine* (motor) não é simplesmente um tipo de tabela, mas características únicas e exclusivas do MySQL que permite escolher um conjunto de recursos pré-definidos para melhor atender aos requisitos e as necessidades de cada tabela em sua base de dados, respeitando também o hardware disponível.

Exercícios de Fixação

1) A secretaria de saúde resolveu contratar uma equipe de desenvolvedores de software para solucionar um problema de gerenciamento de escalas de plantonistas em um hospital. O hospital é composto por vários setores e cada setor possui um nome (ou descrição).

   O plantonista está alocado para um setor em um horário determinado. Ele possui nome e matrícula que precisam ser armazenados. Um plantonista pode estar alocado para mais de um setor, porém não no mesmo horário. Cada setor pode ter vários plantonistas em um mesmo horário.

   Elabore um projeto lógico de banco de dados que atenda a essa situação e forneça acompanhamento adequado ao hospital sobre seus plantonistas.

Referência de Criação e Apoio ao Estudo

Material para Consulta e Apoio ao Conteúdo

- ELMASRI, R. e Navathe, S. B., Fundamentals of Database Systems, Addison-Wesley, 3rd edition, 2000
  - Capítulo 8

- SILBERSCHATZ, A. & Korth, H. F., Sistemas de Banco de Dados - livro
  - Capítulo 4

- Universidade de Brasília (UnB Gama)
  - https://cae.ucb.br/conteudo/unbfga
    (escolha a disciplina **Sistemas de Banco de Dados 1**)