MySQL

Professor Aguinaldo Neto

Sobre o MySQL

- O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface.

- É atualmente um dos sistemas de gerenciamento de bancos de dados mais populares da Oracle Corporation, com mais de 10 milhões de instalações pelo mundo.

História do MySQL

- O sucesso do MySQL deve-se em grande medida à fácil integração com o PHP incluído, quase que obrigatoriamente, nos pacotes de hospedagem de sites da Internet oferecidos atualmente.

- Empresas como Yahoo! Finance, MP3.com, Motorola, NASA, Silicon Graphics e Texas Instruments usam o MySQL em aplicações de missão crítica.

- A Wikipédia é um exemplo de utilização do MySQL em sites de grande audiência.

História do MySQL

- O MySQL foi criado na Suécia por suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que têm trabalhado juntos desde a década de 1980. Hoje seu desenvolvimento e manutenção empregam aproximadamente 400 profissionais no mundo inteiro, e mais de mil contribuem testando o software, integrando-o a outros produtos, e escrevendo a respeito dele.
- No dia 16 de Janeiro de 2008, a MySQLAB, desenvolvedora do MySQL foi adquirida pela Sun Microsystems, por US$ 1 bilhão, um preço jamais visto no setor de licenças livres. No dia 20 de Abril de 2009, foi anunciado que a Oracle compraria a Sun Microsystems e todos os seus produtos, incluindo o MySQL.
- Após investigações da Comissão Europeia sobre a aquisição para evitar formação de monopólios no mercado a compra foi autorizada e hoje a Sun faz parte da Oracle.

MySQL

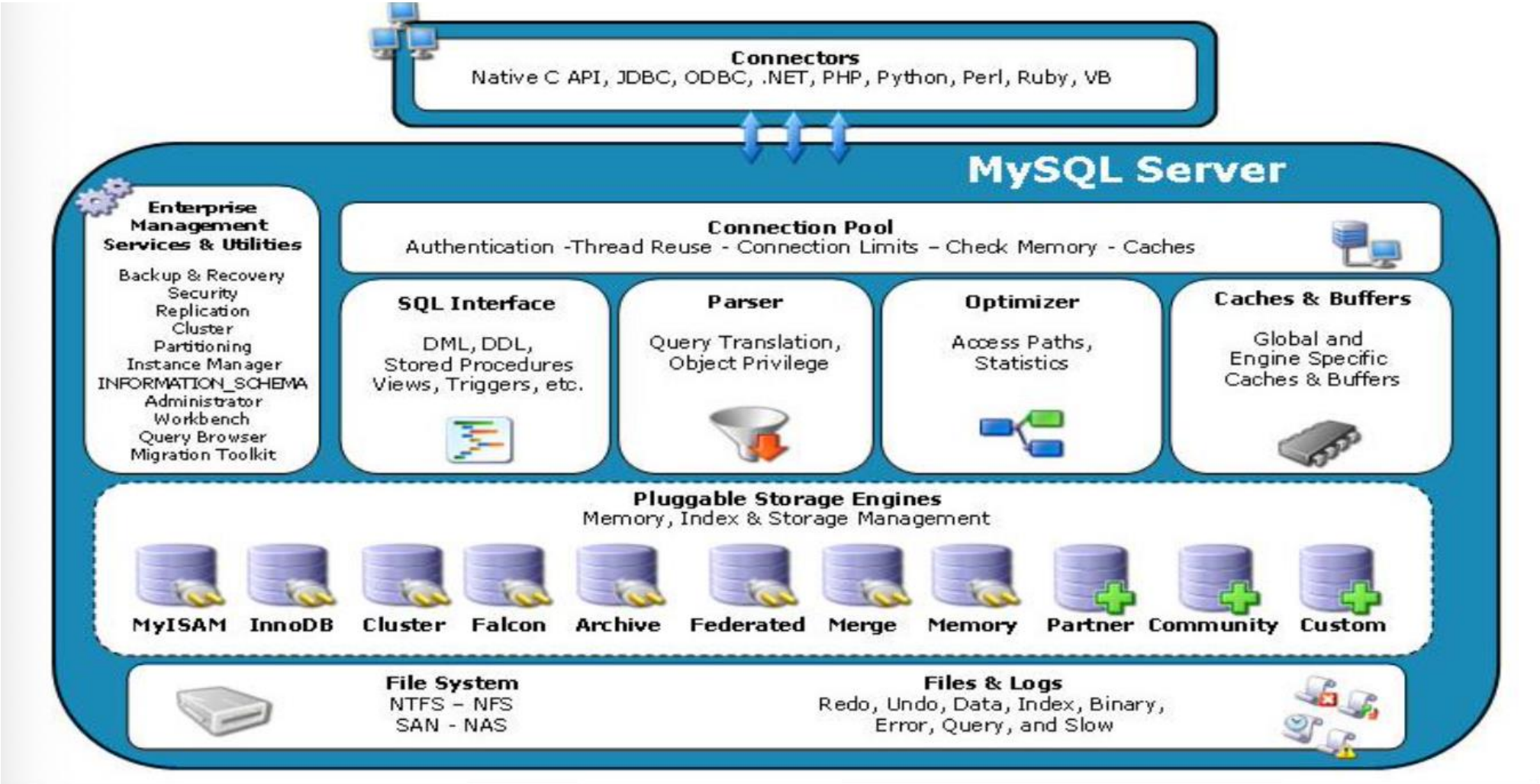
- SGBD MySQL
    - É um dos mais populares SGBDs de código aberto.
    - É executado em mais de 20 plataformas, incluindo Linux, Windows, OS/X, HP-UX, AIX, Netware.
- Licença MySQL
    - MySQL Enterprise, paga-se anualmente pelos serviços de acordo com o plano escolhido.
- https://shop.mysql.com/enterprise/ - site com os valores.

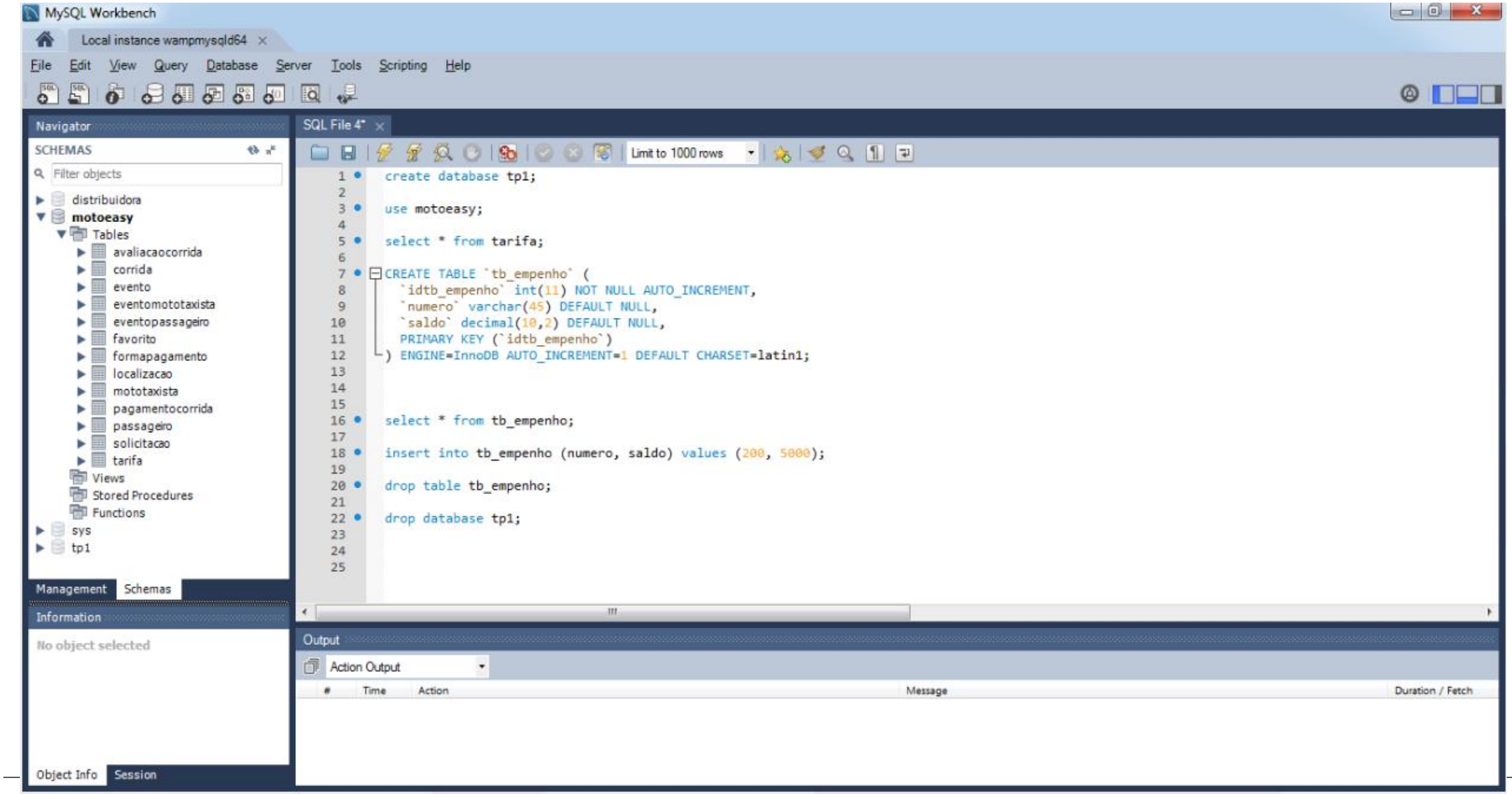
MySQL - Instalação

http://dev.mysql.com/downloads/

- MySQL Downloads
- MySQL Community Server

  (Current Generally Available Release: 8.3.0 – 16 de janeiro de 2024) a versão estável do MySQL.


Connectors
Native C API, JDBC, ODBC, .NET, PHP, Python, Perl, Ruby, VB
MySQL Server
Enterprise Management Services & Utilities
Backup & Recovery
Security
Replication
Cluster
Partitioning
Instance Manager
INFORMATION_SCHEMA
Administrator
Workbench
Query Browser
Migration Toolkit
Connection Pool
Authentication -Thread Reuse - Connection Limits – Check Memory - Caches
SQL Interface
DML, DDL,
Stored Procedures
Views, Triggers, etc.
Parser
Query Translation,
Object Privilege
Optimizer
Access Paths,
Statistics
Caches & Buffers
Global and
Engine Specific
Caches & Buffers
Pluggable Storage Engines
Memory, Index & Storage Management
MyISAM
InnoDB
Cluster
Falcon
Archive
Federated
Merge
Memory
Partner
Community
Custom
File System
NTFS – NFS
SAN - NAS
Files & Logs
Redo, Undo, Data, Index, Binary,
Error, Query, and Slow


MySQL Workbench
Local instance wampmysqld64
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
distribuidora
motoeasy
Tables
avaliacaocorrida
corrida
evento
eventomototaxista
eventopassageiro
favorito
formapagamento
localizacao
mototaxista
pagamentocorrida
passageiro
solicitacao
tarifa
Views
Stored Procedures
Functions
sys
tp1
Management Schemas
Information
No object selected
Object Info Session
SQL File 4*
Limit to 1000 rows
1 create database tp1;
2
3 use motoeasy;
4
5 select * from tarifa;
6
7 CREATE TABLE `tb_empenho` (
8 `idtb_empenho` int(11) NOT NULL AUTO_INCREMENT,
9 `numero` varchar(45) DEFAULT NULL,
10 `saldo` decimal(10,2) DEFAULT NULL,
11 PRIMARY KEY (`idtb_empenho`)
12) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
13
14
15
16 select * from tb_empenho;
17
18 insert into tb_empenho (numero, saldo) values (200, 5000);
19
20 drop table tb_empenho;
21
22 drop database tp1;
23
24
25
Output
Action Output
Time Action Message Duration / Fetch

ACID

- ACID é um conceito que se refere às quatro propriedades de transação de um sistema de banco de dados: Atomicidade, Consistência, Isolamento e Durabilidade.
    - Atomicidade: Em uma transação envolvendo duas ou mais partes de informações discretas, ou a transação será executada totalmente ou não será executada, garantindo assim que as transações sejam atômicas.
    - Consistência: A transação cria um novo estado válido dos dados ou em caso de falha retorna todos os dados ao seu estado antes que a transação foi iniciada.
    - Isolamento: Uma transação em andamento deve permanecer isolada de qualquer outra operação, ou seja, garantimos que a transação não será interferida por nenhuma outra transação concorrente.
    - Durabilidade: Dados validados são registrados pelo sistema de tal forma que mesmo no caso de uma falha e/ou reinício do sistema, os dados estão disponíveis em seu estado correto.

Linguagens

DDL– Data Definition Language (DDL) são usadas para definir a estrutura de banco de dados ou esquema. Alguns exemplos:

CREATE- para criar objetos no banco de dados

ALTER – altera a estrutura da base de dados

TRUNCATE – remover todos os registros de uma tabela, incluindo todos os espaços alocados para os registros são removidos

DROP - Também é um comando DDL, serve para remover toda a tabela (estrutura, indices, contraints, etc…),

COMMENT – adicionar comentários ao dicionário de dados

RENAME – para renomear um objeto

DML– Data Manipulation Language (DML) são utilizados para o gerenciamento de dados dentro de objetos do banco. Alguns exemplos:

SELECT- recuperar dados do banco de dados

INSERT – inserir dados em uma tabela

UPDATE – atualiza os dados existentes em uma tabela

DELETE – exclui registros de uma tabela,

DCL– Data Control Language (DCL) declarações. Alguns exemplos:

GRANT – atribui privilégios de acesso do usuário a objetos do banco de dados

REVOKE – remove os privilégios de acesso aos objetos obtidos com o comando GRANT

Criar banco de dados

- Banco de dados também é conhecido pelo nome: Schema
- O sistema do MySQL pode suportar vários bancos de dados diferentes.
- Geralmente será criado um banco de dados para cada aplicação.
- Para cria um banco de dados no MySQL será utilizado a palavra-chave CREATE DATABASE.

- Sintaxe

```
CREATE DATABASE nome_do_banco_de_dados;

CREATE SCHEMA nome_do_banco_de_dados;
```

Nomenclatura

- Pode inserir no comando letras maiúsculas, mas o sistema interpreta e cria o banco com letras minúsculas.

Válido	Inválido
nome_banco	nome banco
3nome_banco	nome-banco
banco_pái	nome%banco
NomeBanco	

- Como boas práticas é preferível definir o nome todo em caixa baixa, sem acentos e se for necessário ter palavras compostas, separá-las com underline.

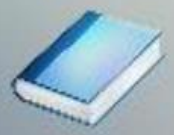
```
CREATE DATABASE nome_do_banco_de_dados;
```

Tipos Primitivos

Numérico

- **Inteiro** → `TinyInt, SmallInt, Int, MediumInt, BigInt`
- **Real** → `Decimal, Float, Double, Real`
- **Lógico** → `Bit, Boolean`

Data/Tempo → `Date, DateTime, TimeStamp, Time, Year`



Literal

- **Caractere** → `Char, VarChar`
- **Texto** → `TinyText, Text, MediumText, LongText`
- **Binário** → `TinyBlob, Blob, MediumBlob, LongBlob`
- **Coleção** → `Enum, Set`

Espacial → `Geometry, Point, Polygon, MultiPolygon`

Tipo texto

- CHAR(tamanho) : guarda um número fixo de caracteres. Pode conter letras, números e caracteres especiais. O tamanho deve ser declarado entre parênteses. Guarda até 255 caracteres.
- VARCHAR(tamanho): possui as características do tipo CHAR, com a diferença de que, se você criar com mais de 255 caracteres, ele transforma-se no tipo TEXT. Ou seja, se for criar algum campo com mais de 255, já crie como TEXT.
- TEXT: guarda uma string com o tamanho máximo de 65.535 caracteres.
- BLOB: é o tipo de dado medido pela quantidade de bytes, em vez de pela quantidade de caracteres, conforme a maioria. Pode salvar por imagens, por exemplo, com o máximo de 65.535 bytes de arquivo.

TIPO	DESCRIÇÃO
BIT	o mesmo que TINYINT
BOOL	o mesmo que TINYINT
SMALLINT[(M)]	inteiros pequenos
INT[(M)]	inteiros regulares
INTEGER[(M)]	o mesmo que INT
BIGINT[(M)]	inteiros grandes
FLOAT(precisão)	números de ponto flutuante de precisão simples ou dupla
FLOAT[(M,D)]	números de ponto flutuante de precisão simples. O mesmo que FLOAT(4)
DOUBLE[(M,D)]	números de ponto flutuante de precisão dupla. O mesmo que FLOAT(8)
DOUBLE	O mesmo que DOUBLE[(M,D)]
PRECISION[(M,D)]	O mesmo que DOUBLE[(M,D)]
REAL[(M,D)]	O mesmo que DOUBLE[(M,D)]
DECIMAL[(M,D)]	número de ponto flutuante armazenado como char
NUMERIC[(M,D)]	O mesmo que DECIMAL
DEC[(M,D)]	O mesmo que DECIMAL

Tipo de dados

Dados Temporais		
Tipo	Formato padrão	Valores permitidos
Date	AAAA-MM-DD	1000-01-01 a 9999-12-31
Datetime	AAAA-MM-DD HH:MI:SS	1000-01-01 00:00:00 a 9999-12-31 23:59:00
Timestamp	AAAA-MM-DD HH:MI:SS	1970-01-01 00:00:00 a 2037-12-31 23:59:00
Year	AAAA	1901 a 2155
Time	HHH:MI:SS	-838:59:59 a 838:59:59

Dados de Texto Não-Binário	
Tipo de texto	Numero máximo de bytes
Tinytext	255
Text	65.535
MediumText	16.777.215
LongText	4.294.967.295
Varchar	65.535
Char	255

Atributos SQL

- Not null
- Auto-incremente
- Primary key
- Constraint

Criar tabela

- Após criar um banco de dados, é necessário criar tabelas para atender certa demanda.
- Não é possível criar mais de uma tabela com mesmo nome em um banco de dados.
- Para cria uma tabela em um banco de dados no MySQL será utilizado a palavra-chave CREATE TABLE.
- Sintaxe

```
CREATE TABLE nome_tabela
(
        <nome_da_coluna1> <tipo_da_coluna1> [<atributos_da_coluna1>],
        ...
        <nome_da_coluna> <tipo_da_coluna> [<atributos_da_coluna>]
);
```

Exemplo

- Crie a tabela carro no banco de dados banco_1

```
CREATE TABLE carro
(
id_carro int not null auto_increment,
modelo varchar(50),
cor varchar(15),
ano smallint(4),
primary key (id_carro)
)
```

Inserir dados

- Após criar uma tabela no banco de dados, é possível inserir um ou vários dados.
- Para inserir um registro na tabela serão utilizadas as palavras-chave INSERT INTO e VALUES.
- As strings devem ser incluídas em pares de aspas simples ou dupla.
- Números Inteiros ou Flutuantes não necessitam de aspas.
- Sintaxe

```
INSERT INTO <nome_tabela>
( <campo1> , ... , <campoN>)
VALUES
( <valorCampo1> , ... , < valorCampoN> );
```

- Sintaxe alternativa

```
INSERT INTO <nome_tabela> VALUES ( <valorCampo1> , ... , < valorCampoN> );
```

Exemplo

- Insira os seguintes registros na tabela carro.

```
INSERT INTO carro ( modelo , cor , ano ) VALUES ( 'Corsa' , 'Vermelho' , 2003 );
INSERT INTO carro ( modelo , cor , ano ) VALUES ( 'Fusca' , 'Branco' , 1966 );
INSERT INTO carro ( modelo , cor , ano ) VALUES ( 'Palio' , 'Prata' , 2009 );
INSERT INTO carro ( modelo , cor , ano ) VALUES ( 'Gol' , 'Branco' , 2008 );
```

Recuperar dados

- Após inserir registros em uma tabela no banco de dados, é possível recuperá-los de várias formas.
- Para recuperar registros de uma tabela serão utilizadas as palavras-chave SELECT e FROM.
- Sintaxe

```
SELECT * FROM nome_tabela
```

- Sintaxe alternativa

```
SELECT <coluna1> , ... , <colunaN> FROM nome_tabela
```

Exemplo

- Recuperar todas as colunas da tabela carro.

```
SELECT * FROM carro
```

- Recuperar apenas as colunas modelo e ano da tabela carro.

```
SELECT modelo , ano FROM carro
```

Recuperar dados específicos

- Para recuperar registros específicos em uma tabela serão utilizadas as palavras-chave WHERE, OR e AND.
- Sintaxe

  SELECT * FROM nome_tabela WHERE <condição>

- Sintaxe utilizando AND
  - Todas as condições envolvidas devem ser verdadeiras.

  SELECT * FROM nome_tabela WHERE <condição> AND <condição>

- Sintaxe utilizando OR
  - Pelo menos uma condição envolvida deve ser verdadeira.

  SELECT * FROM nome_tabela WHERE <condição> OR <condição>

Operadores de comparação para where

Nome	Operador	Exemplo	Descrição
Igualdade	=	valor_coluna = 5	Verificar se os dois valores são iguais.
Maior que	>	valor_coluna > 30	Verificar se o valor da esquerda é maior que o da direita.
Menor que	<	valor_coluna < 45	Verificar se o valor da esquerda é menor que o da direita.
Maior ou igual	>=	valor_coluna >= 12	Verificar se o valor da esquerda é maior ou igual ao da direita.
Menor ou igual	<=	valor_coluna <= 94	Verificar se o valor da esquerda é menor ou igual ao da direita.
Desigualdade	!= ou <>	valor_coluna != 2009	Verificar se os dois valores são diferentes.

Exemplo

- Recuperar apenas os registros de carro brancos.

```
SELECT * FROM carro WHERE cor ='Branco’;
```

- Recuperar apenas os registros de carro brancos do ano 1966.

```
SELECT * FROM carro WHERE cor ='Branco' AND ano = 1966;
```

- Recuperar apenas os registros de carro brancos ou pratas.

```
SELECT * FROM carro WHERE cor ='Branco' OR cor =‘Prata’;
```

- Recuperar apenas carros produzidos a partir do ano 2000.

```
SELECT * FROM carro WHERE ano >= 2000;
```

- Recuperar apenas carros que não foram produzidos em 2009.

```
SELECT * FROM carro WHERE ano != 2009;
```

Alterar dados

- Após inserir dados na tabela, podemos alterar os mesmos, caso necessário.
- Para altera registros em uma tabela serão utilizadas as palavras-chave UPDATE e SET.
- Sintaxe

```
UPDATE <Nome da Tabela>
SET <Coluna 1> = ‘Valor Coluna 1‘ , <Coluna 1> = ‘Valor Coluna 1‘
WHERE <Condição>;
```

- Sintaxe alternativa

```
UPDATE <Nome da Tabela>
SET <Coluna 1> = ‘Valor Coluna 1‘ , <Coluna 1> = ‘Valor Coluna 1‘
```

Exemplo

- Alterar o modelo do Gol para Gol 2008.

```
UPDATE carro SET modelo = 'Gol 2008‘ WHERE id_carro = 4;
```

- Alterar a cor de todos os carros brancos para branco gelo.

```
UPDATE carro SET cor = 'Branco Gelo' WHERE cor = 'branco‘;
```

- Alterar a cor de branco gelo para branco neve nos carros fabricados a partir do ano 2000.

```
UPDATE carro SET cor = 'Branco Neve’
WHERE cor = 'branco gelo' AND ano >= 2000;
```

- Alterar a cor de todos os carros para Cinza.

```
UPDATE carro SET cor = ‘Cinza‘;
```

Excluir registro

- Após inserir dados na tabela, podemos excluí-los se for necessário.
- Para excluir registro(s) em uma tabela será utilizada a palavra-chave DELETE.
- Sintaxe

```
DELETE FROM <Nome da Tabela> WHERE <Condição>

DELETE FROM <Nome da Tabela>
```

Exemplo

- Excluir apenas o carro que tenha o código 4.

```
DELETE FROM carro WHERE id_carro = 4;
```

- Excluir todos os registros de carro.

```
DELETE FROM carro
```

■ Referência

- MYSQL. *Manual de Referência do MySQL 4.1*.
- https://dev.mysql.com/doc/workbench/en/. Acesso em: 12 fev. 2025.