



TECNOLOGIAS E PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

# TRABALHO DE PROJETO UFCD 5085

CRIAÇÃO DE ESTRUTURA DE BASE DE DADOS EM SQL

APRESENTADO POR: ANDRÉ BARRIOS VIEIRA

FORMADOR: TIAGO LOPES

IEFP – INSTITUTO DE EMPREGO E FORMAÇÃO PROFISSIONAL

## ÍNDICE

<b>INTRODUÇÃO.....</b>	<b>2</b>
DESCRIÇÃO GERAL DO PLANO .....	2
OBJETIVO .....	2
O QUE É UMA BASE DE DADOS? .....	2
<b>ESTRUTURA DE UMA BASE DE DADOS .....</b>	<b>3</b>
BASE DE DADOS MODELO E-R (ENTIDADE RELAÇÃO).....	3
CHAVES .....	4
TIPOS DE MODELOS RELACIONAIS .....	6
<b>CRIANDO UMA ESTRUTURA DE BASE DE DADOS.....</b>	<b>8</b>
IDEALIZAÇÃO .....	8
<b>CRIANDO UMA BASE DE DADOS EM AMBIENTE SQL.....</b>	<b>10</b>
EXIBINDO A BASE DE DADOS.....	10
CRIANDO UMA BASE DE DADOS .....	11
SELECIONANDO UMA BASE DE DADOS .....	11
CRIANDO TABELAS .....	11
<b>ALTERANDO TABELAS .....</b>	<b>15</b>
<b>MANIPULANDO DADOS .....</b>	<b>16</b>
INSERINDO DADOS EM SUAS TABELAS.....	16
CONSULTANDO DADOS EM SUAS TABELAS .....	16
ALTERANDO DADOS .....	19
<b>CONSULTAS UTILIZANDO CONDIÇÕES .....</b>	<b>20</b>
WHERE .....	20
ORDER BY .....	21
<b>FUNÇÕES DE AGREGAÇÃO .....</b>	<b>22</b>
<b>OPERADORES ARITMÉTICOS .....</b>	<b>26</b>
<b>VIEWS.....</b>	<b>28</b>
VANTAGENS DA UTILIZAÇÃO DE VIEWS .....	28
CRIANDO VIEWS .....	28
MANIPULANDO VIEWS .....	30
<b>TRANSAÇÕES .....</b>	<b>32</b>
UTILIZANDO TRANSAÇÕES .....	33
<b>CONCLUSÃO .....</b>	<b>34</b>
<b>WEBGRAFIA .....</b>	<b>35</b>

## INTRODUÇÃO

No decorrer deste trabalho será apresentado de forma didática as práticas, ferramentas e métodos para criação de um banco de dados simples. Iremos passar pela idealização de um tema, elaboração de um diagrama de estruturas, criação desta base de dados em ambiente SQL bem como a manipulação de dados.

## DESCRIÇÃO GERAL DO PLANO

PASSO	DESCRIÇÃO
<b>Idealização</b>	Idealização de um cenário com tema definido para uma base de dados
<b>Elaboração de Diagrama DER</b>	Elaboração de modelo relacional, normalizado, com a identificação das chaves primárias e externas
<b>Criação</b>	Criação da base de dados em ambiente SQL
<b>Manipulação</b>	Inserção, manipulação e operações de tabelas / dados

## OBJETIVO

Introduzir através de exemplos os conceitos para criação de uma base de dados simples.

## O QUE É UMA BASE DE DADOS?

Uma base de dados é um conjunto de dados relacionados entre eles de acordo com regras estabelecidas para um objetivo específico.

Um bom exemplo de banco de dados seria uma lista telefônica. Essa lista armazena os dados referentes a um ou mais contatos, como o nome, número e e-mail.

## ESTRUTURA DE UMA BASE DE DADOS

Uma base de dados é constituída por tabelas, que por sua vez são estruturadas em linhas e colunas. Para as linhas dá-se o nome de REGISTOS e para as colunas o nome de CAMPOS.

O diagrama mostra uma tabela com o título 'Tabela CONTATOS'. A tabela possui três colunas: 'Nome', 'Numero' e 'E-mail'. Há quatro linhas de dados. À esquerda da tabela, há três legendas: 'Registos' apontando para as linhas de dados, e 'Campo' apontando para as colunas. A primeira linha de dados contém 'José Vieira', '934567234' e 'josevieira@gmail.com'. A segunda linha contém 'Maria Silva', '995676243' e 'msilva@hotmail.com'. A terceira linha contém 'Paulo Mendonça', '907654198' e 'paulo.mend@sapo.pt'. A quarta linha contém 'Ana Farias', '945236909' e 'ana.farias@msn.com'.

Tabela CONTATOS		
Nome	Numero	E-mail
José Vieira	934567234	josevieira@gmail.com
Maria Silva	995676243	msilva@hotmail.com
Paulo Mendonça	907654198	paulo.mend@sapo.pt
Ana Farias	945236909	ana.farias@msn.com

Figura 1 – Exemplo de Tabela

## BASE DE DADOS MODELO E-R (ENTIDADE RELAÇÃO)

O modelo entidade relação busca simular uma realidade onde um conjunto de **entidades** interage uma com as outras através de um conjunto de **associações** e **relações**.

Para o desenvolvimento deste modelo aplica-se o seguinte método em etapas:

### Etapa 1 - Diagrama E-R

Nesta etapa ocorre a análise das necessidades de informação e elaboração do modelo E-R.

### Etapa 2 - Tabelas não normalizadas

Etapa onde há a transformação do Diagrama E-R em um conjunto de tabelas.

### Etapa 3 - Tabelas normalizadas

Etapa responsável pela normalização das tabelas.

Sobre os termos apresentados temos que:

**Entidade** – Objetos e ou conceitos que possuem um conjunto características comuns entre eles, caracterizadas por um conjunto de atributos. Ex.: Carros distintos apresentam um conjunto de atributos com características comum, como o modelo, a cor, cilindrada, ano de

fabricação. Dados fornecidos para cada um desses atributos permitem a classificação de um ou mais carros. Importante ressaltar que a entidade corresponde a uma tabela em uma base de dados.

**Atributos** – São as características comuns entre os objetos e ou conceitos de após a definição de uma entidade. No exemplo citado acima os atributos de uma entidade “Carro” são: Modelo, cor, cilindrada e ano de fabricação. Os atributos correspondem aos campos de uma tabela.



Figura 2 – Exemplo de Tabela – Entidade / Atributo

**Domínio** – É o conjunto de todos os valores elementares que um atributo pode assumir, esse valor define o tipo que será atribuído á esse atributo. Ex.: Nome (Tipo Texto), Número (Tipo Número), E-mail (Tipo Texto).

## CHAVES

A chave é o conceito básico para identificar linhas e estabelecer relações entre linhas de uma tabela de um banco de dados, pode ser definida por um (chave simples) ou mais (chave composta) atributos.

**Chave candidata** – Pode ser definida por um ou mais atributos que por sua vez devem ser identificadores únicos de um registo.

**Chave primária (Primary Key)** – Uma chave primária corresponde a um atributo ou conjunto de atributos que distingue cada registo de todos os outros registos. Dever ser única e não pode apresentar um valor nulo.

Na imagem abaixo podemos ver o exemplo de uma tabela chamada Livros, onde definimos como **chave primária** o atributo Título, pois o título de um livro é uma característica única, que distingue esse registo dos demais.

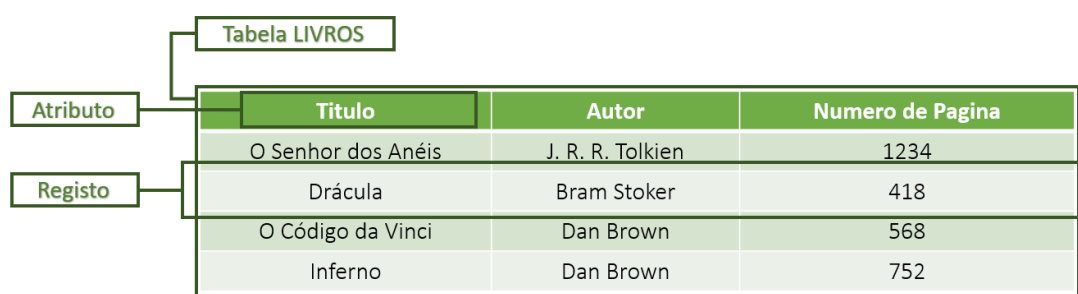


Figura 3 – Exemplo de Tabela – Chave Primária

**Chave estrangeira** – A Chave Estrangeira ou chave externa é a chave Primária de uma ou mais tabelas utilizadas em outra tabela criando deste modo um relacionamento entre elas.



Figura 4 – Exemplo de Tabela – Chave Externa

Neste exemplo os atributos Cod\_Cliente e Título são chaves primárias de suas respetivas tabelas e foram herdadas pela tabela VENDAS como chaves estrangeiras.

## TIPOS DE MODELOS RELACIONAIS

Os modelos relacionais podem ser classificados de três maneiras distintas, sendo elas: relação binária 1:1, relação binária 1:N e relação binária N:N.

**Relação binária 1:1**– O relacionamento 1:1 define que um item de uma entidade só poderá se relacionar com um item de outra entidade. Por exemplo, imaginemos que as entidades Cliente e Morada se relacionam de forma 1:1, deste modo um cliente só poderá possuir uma morada e que por sua vez só poderá estar relacionada a um cliente.



Figura 5 – Relacionamento 1:1

Neste modelo 1:1 a relação e uma das entidades deixam de existir, os atributos da entidade eliminada passam a fazer parte da entidade mantida.

**Relação binária 1:N**– No relacionamento 1:N um item de uma tabela pode se relacionar com vários itens de uma outra tabela. Como exemplo podemos imaginar um modelo relacionais entre clientes de uma concessionária e seus carros. Um cliente pode comprar vários carros, porém um carro só pode ser comprado por um cliente.



Figura 6 – Relacionamento 1:N

No modelo 1:N, a relação compra deste exemplo, deixa de existir, porém são mantidas as duas entidades Cliente e Carro.

**Relação binária N:N**– Em um relacionamento N:N um item de uma tabela pode se relacionar com vários itens de uma outra tabela e vice-versa. Por exemplo, um cliente de uma papelaria pode comprar vários produtos, e um produto pode ser vendido a vários clientes.



Figura 7 – Relacionamento N:N

O modelo N:N apresenta um comportamento particular, onde a relação passará a ser uma entidade com atributos herdados das outras entidades e possivelmente com atributos próprios.



## CRIANDO UMA ESTRUTURA DE BASE DE DADOS

Seguindo o plano descrito anteriormente, utilizaremos alguns passos para criação e demonstração de utilização e recursos de uma tabela de dados, sendo eles: Idealização, Elaboração de Diagrama Relacionamento, Criação do banco de dados em ambiente SQL e Manipulação de dados e informações.

### IDEALIZAÇÃO

Para seguirmos com nosso projeto, iremos imaginar uma solução real para uma necessidade hipotética.

Desta maneira nosso banco de dados será criado para suprir a necessidade de uma vinícola no controle entre a aquisição de uvas e a produção de vinhos.

Nosso primeiro esboço do diagrama EDR pode ser visualizado abaixo.

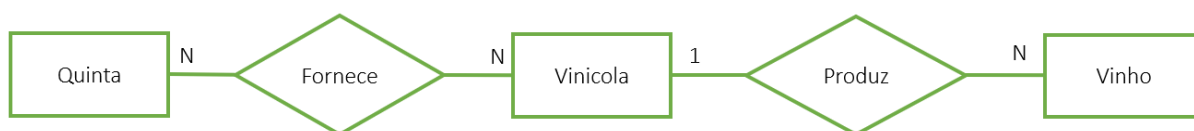


Figura 8 – EDR – Esboço

Como podemos observar as Quintas tem uma relação N:N com as Vinícolas, pois uma Quinta pode fornecer uvas para mais de uma Vinícola e esta por sua vez pode comprar uvas de mais de uma Quinta. Entre a Vinícola e o Vinho temos uma relação 1:N, logo que, uma Vinícola pode produzir diversos Vinhos, porém um Vinho só pode ser produzido por uma Vinícola.

Seguindo nosso estudo iremos realizar a normalização de nosso diagrama, onde o resultado deve apresentar apenas entidades que se relacionam entre si de modo 1:N.



Figura 9 – EDR – Normalização

Em seguida definiremos os atributos relevantes ao nosso banco de dados.

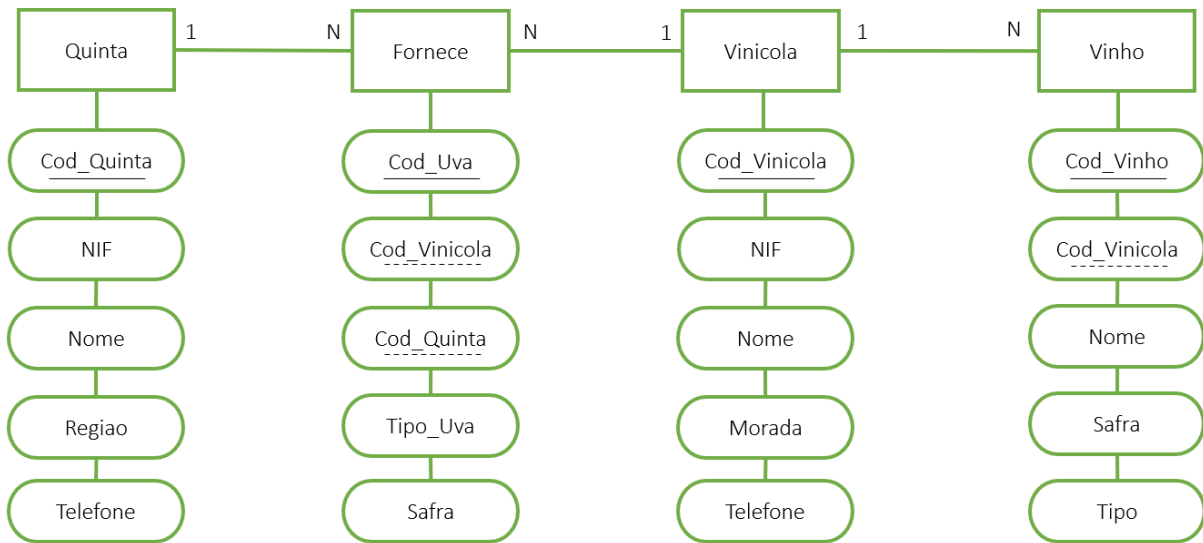


Figura 10 – EDR – Atributos

## CRIANDO UMA BASE DE DADOS EM AMBIENTE SQL

Neste ponto iremos proceder com a criação de nosso banco de dados em um ambiente SQL, em nosso caso específico utilizaremos o sistema de gerenciamento MySQL.

O MySQL foi lançado em 1995 e hoje é um dos sistemas de gerenciamento mais populares, sendo utilizado por empresas como: NASA, Sony, Lufthansa, Cisco Systems, Google, entre outros.

O acesso ao sistema MySQL será feito através do software WampServer, após sua instalação, iremos trabalhar em nosso banco de dados via terminal MySQL Console, o caminho pode ser visto na imagem abaixo.

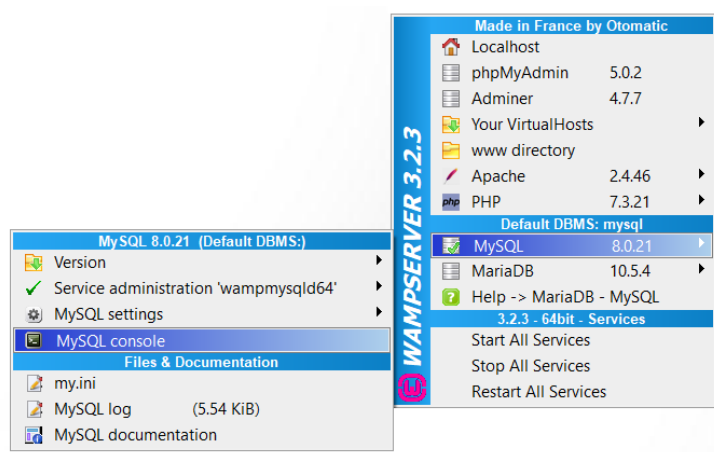


Figura 11 – Wamp – MySQL

Já no terminal a criação de nossa base de dados será feita através de linhas de comandos que serão apresentadas no decorrer desta sessão.

## EXIBINDO A BASE DE DADOS

Iniciaremos nosso caminho pelo comando responsável por exibir as bases de dados que já possuímos em nosso sistema: **SHOW DATABASES;**



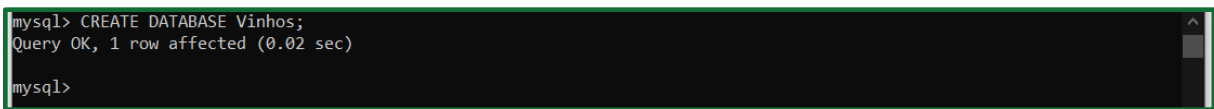
Figura 12 – MySQL – Show Databases

---

## CRIANDO UMA BASE DE DADOS

Em seguida temos o comando responsável pela criação efetiva de nossa base de dados, trata-se do comando:

**CREATE DATABASE** nome\_base\_dados;



```
mysql> CREATE DATABASE Vinhos;
Query OK, 1 row affected (0.02 sec)

mysql>
```


Figura 13 – MySQL – Create Database

---

## SELECIONANDO UMA BASE DE DADOS

Para trabalharmos em uma base de dados, devemos informar o sistema que o desejamos através do comando:

**USE** nome\_base\_dados;



```
mysql> USE Vinhos;
Database changed

mysql>
```

Figura 14 – MySQL – Use

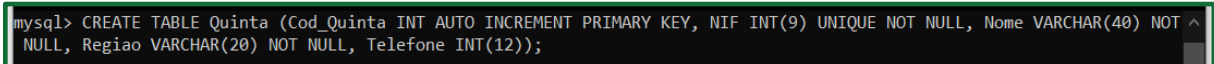
---

## CRIANDO TABELAS

Em seguida criaremos as tabelas de nosso banco de dados e nesse momento já definiremos também os atributos desta tabela, para isso teremos o comando:

**CREATE TABLE** nome\_tabela (atributos);

Neste momento também deveremos definir as propriedades individuais dos atributos, descreveremos essas propriedades a seguir mediante sua utilização.



```
mysql> CREATE TABLE Quinta (Cod_Quinta INT AUTO INCREMENT PRIMARY KEY, NIF INT(9) UNIQUE NOT NULL, Nome VARCHAR(40) NOT NULL, Regiao VARCHAR(20) NOT NULL, Telefone INT(12));
```

Figura 15 – MySQL – Create Table

Propriedades utilizadas no exemplo:

**INT** – Define que o atributo receberá como valores apenas números inteiros. O valor entre os parênteses define o número máximo de caracteres que poderá ser atribuído ao valor desse atributo.

**AUTO\_INCREMENT** – Define que o atributo receberá automaticamente um valor sequencial.

**PRIMARY KEY** – Define que o atributo como chave primária.

**UNIQUE** – Define que o valores inseridos neste atributo não poderão ser iguais.

**VARCHAR** – Define que os valores inseridos no atributo serão alfanuméricos. O valor entre parenteses também define o número máximo de caracteres.

**NOT NULL** – Não permite que um atributo tenha valor nulo.

Já o próximo exemplo nos mostra como definir chave estrangeiras em nossa tabela através do comando:

**CREATE TABLE** nome\_tabela (nome\_atributo propriedades\_atributo, **FOREIGN KEY** (nome\_atributo) **REFERENCES** nome\_tabela\_referencia(nome\_atributo\_referencia);

```
mysql> CREATE TABLE Uva (Cod_Uva INT AUTO_INCREMENT PRIMARY KEY, Tipo_Uva VARCHAR(20) NOT NULL, Safra INT(4) NOT NULL, Cod_Quinta INT NOT NULL, Cod_Vinicola INT NOT NULL, FOREIGN KEY(Cod_Quinta) REFERENCES Quinta(Cod_Quinta), FOREIGN KEY(Cod_Vinicola) REFERENCES Vinicola(Cod_Vinicola));
```

Figura 16 – MySQL – Create Table

Para visualizarmos as tabelas criadas devemos utilizar o comando:

**DESC** nome\_tabela;

```
mysql> Desc Uva;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Cod_Uva | int | NO | PRI | NULL | auto_increment |
| Tipo_Uva | varchar(20) | NO | | NULL | |
| Safra | int | NO | | NULL | |
| Cod_Quinta | int | NO | MUL | NULL | |
| Cod_Vinicola | int | NO | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
```

Figura 17 – MySQL – Desc Table

Assim para criação de nossa base de dados e tabelas foram utilizadas as seguintes linhas de comando em ordem:

**CREATE DATABASE** Vinhos;

**USE** Vinhos;

```
CREATE TABLE Quinta (Cod_Quinta INT AUTO_INCREMENT PRIMARY KEY, NIF INT(9) UNIQUE NOT NULL, Nome VARCHAR(40) NOT NULL, Regiao VARCHAR(20) NOT NULL, Telefone INT(12), CHECK(Regiao IN ('Açores', 'Alentejo', 'Algarve', 'Centro', 'Madeira', 'Metropolitana', 'Norte')));
```

Field	Type	Null	Key	Default	Extra
Cod_Quinta	int	NO	PRI	NULL	auto_increment
NIF	int	NO	UNI	NULL	
Nome	varchar(40)	NO		NULL	
regiao	varchar(20)	NO		NULL	
Telefone	int	YES		NULL	

Figura 18 – MySQL – Tabela Quinta

**NOTA:** Nesta linha de criação do exemplo acima apresentamos a constraint **CHECK**. Utilizamos o **CHECK** para definirmos quais os valores dos dados que podem ser inseridos em um atributo. Para o atributo **regiao** da tabela **Quinta**, definimos quais os nomes de regiões que poderão ser valores atribuídos a este atributo.

```
CREATE TABLE Vinicola (Cod_Vinicola INT AUTO_INCREMENT PRIMARY KEY, NIF INT(9) UNIQUE NOT NULL, Nome VARCHAR(40) NOT NULL, Morada VARCHAR(100), Telefone INT(12));
```

Field	Type	Null	Key	Default	Extra
Cod_vinicola	int	NO	PRI	NULL	auto_increment
NIF	int	NO	UNI	NULL	
nome	varchar(40)	NO		NULL	
Morada	varchar(100)	YES		NULL	
Telefone	int	YES		NULL	

Figura 19 – MySQL – Tabela Vinicola

```
CREATE TABLE Fornece (Cod_Fornece INT AUTO_INCREMENT PRIMARY KEY, Tipo_Uva VARCHAR(20) NOT NULL, Safra INT(4) NOT NULL, Cod_Quinta INT NOT NULL, Cod_Vinicola INT NOT NULL, FOREIGN KEY(Cod_Quinta) REFERENCES Quinta(Cod_Quinta), FOREIGN KEY(Cod_Vinicola) REFERENCES Vinicola(Cod_Vinicola));
```

Field	Type	Null	Key	Default	Extra
Cod_Fornece	int	NO	PRI	NULL	auto_increment
Tipo_Uva	varchar(20)	NO		NULL	
Safra	int	NO		NULL	
Cod_Quinta	int	NO	MUL	NULL	
Cod_Vinicola	int	NO	MUL	NULL	

Figura 20 – MySQL – Tabela Fornece

```
CREATE TABLE Vinho (Cod_Vinho INT AUTO_INCREMENT PRIMARY KEY, Nome VARCHAR(20)
UNIQUE NOT NULL, Safra INT(4) NOT NULL, Tipo VARCHAR(10) NOT NULL, Cod_Vinicola INT
NOT NULL, FOREIGN KEY(Cod_Vinicola) REFERENCES Vinicola (Cod_Vinicola), CHECK(tipo IN
('Tinto', 'Branco', 'Verde', 'Rose', 'Licoroso', 'Espumante')));
```

Field	Type	Null	Key	Default	Extra
Cod_Vinho	int	NO	PRI	NULL	auto_increment
Nome	varchar(20)	NO	UNI	NULL	
Safra	int	NO		NULL	
Tipo	varchar(20)	NO		NULL	
Cod_Vinicola	int	NO	MUL	NULL	

Figura 21 – MySQL – Tabela Vinho

---

## ALTERANDO TABELAS

Provavelmente durante a criação de suas tabelas, haverá momentos em que seja necessário que ajustes sejam realizados, para isso temos o comando ALTER TABLE e suas variáveis como descrito a seguir.

**ADD** – Adiciona um novo atributo a tabela.

**ALTER TABLE** nome\_tabela **ADD** nome\_atributo propriedades\_atributo;

**CHANGE** – Altera o nome e propriedades de um atributo.

**ALTER TABLE** nome\_tabela **CHANGE** nome\_atributo novo\_nome\_atributo  
propriedades\_atributo;

**MODIFY** – Altera as propriedades de um atributo.

**ALTER TABLE** nome\_tabela **MODIFY** nome\_atributo propriedades\_atributo;

**DROP** – Elimina um atributo da tabela.

**ALTER TABLE** nome\_tabela **DROP COLUMN** nome\_atributo;

**RENAME** – Altera o nome de uma tabela.

**ALTER TABLE** nome\_tabela **RENAME TO** novo\_nome;



## MANIPULANDO DADOS

Neste capítulo iniciaremos a abordagem ao tratamento de dados de nosso banco. Serão apresentados aqui as linhas de comando e técnicas para inserção, manipulação e visualização dos dados referentes as nossas tabelas.

### INSERINDO DADOS EM SUAS TABELAS

Agora que temos nossas tabelas criadas, iniciaremos a inserção dos dados referentes. Os dados inseridos devem sempre respeitar a propriedade do atributo a qual se relaciona, por exemplo, se meu atributo Telefone tem a propriedade INT(12), os dados inseridos deverão ser numéricos e ter no máximo 12 caracteres. O comando utilizado para inserção de dados em uma tabela é:

**INSERT INTO** nome\_tabela **VALUES** (dados)

```
mysql> INSERT INTO Quinta VALUES ("",265712957, "Quinta do Olivardo", "Algarve", 914345678);
```

Figura 22 – MySQL – Insert Into

Nota 1.: Observe que o primeiro dado referente a Cod\_Quinta, foi definido como “ ”, isso porque nosso atributo tem a propriedade **AUTO INCREMENT**, logo caberá ao próprio sistema o preenchimento deste campo.

Nota 2.: Os dados do tipo texto como VARCHAR deverão ser inseridos entre plicas. Por exemplo, no atributo **Nome** inserirmos o dado “Quinta do Olivardo”.

### CONSULTANDO DADOS EM SUAS TABELAS

Você pode visualizar os dados inserido em uma tabela através do comando:

**SELECT \* FROM** nome\_tabela;

```
mysql> SELECT * FROM Quinta;
+-----+-----+-----+-----+-----+
| Cod_Quinta | NIF      | Nome           | Regiao  | Telefone |
+-----+-----+-----+-----+-----+
|          1 | 265712957 | Quinta do Olivardo | Algarve | 914345678 |
+-----+-----+-----+-----+-----+
```

Figura 23 – MySQL – Select From

Para continuidade de nosso exemplo os seguintes dados foram inseridos em nossas tabelas:

#### Tabela Quinta

```
INSERT INTO Quinta VALUES ('', 265712957, "Quinta do Olivardo", "Algarve", 914345678);
```

```
INSERT INTO Quinta VALUES ('', 145783290, "Folha D'Ouro", "Norte", 235987345);
```

```
INSERT INTO Quinta VALUES ('', 135896234, "Quinta o Alentejano", "Alentejo", NULL);
```

```
INSERT INTO Quinta VALUES ('', 276980672, "Castelo do Norte", "Norte", 276243987);
```

Cod_Quinta	NIF	Nome	regiao	Telefone
1	265712957	Quinta do Olivardo	Algarve	914345678
2	145783290	Folha D'Ouro	Norte	235987345
3	135896234	Quinta o Alentejano	Alentejo	NULL
4	276980672	Castelo do Norte	Norte	276243987

Figura 24 – MySQL – Insert Quinta

#### Tabela Vinicola

```
INSERT INTO Vinicola VALUES ('', 234768465, "Monte Mor", "Coimbra - Norte", NULL);
```

```
INSERT INTO Vinicola VALUES ('', 324678452, "Alto da Torre", "Beja - Alentejo", 234876345);
```

```
INSERT INTO Vinicola VALUES ('', 234678123, "Arco do Castelo", "Mira- Centro", 978265254);
```

```
INSERT INTO Vinicola VALUES ('', 287450786, "Vale do Rio", "Aljustrel - Alentejo, ", NULL);
```

Cod_vinicola	NIF	nome	Morada	Telefone
1	234768465	Monte Mor	Coimbra - Norte	NULL
2	324678452	Alto da Torre	Beja - Alentejo	234876345
3	234678123	Arcos do Castelo	Mira-Centro	978265254
4	287450786	Vale do Rio	Aljustrel - Alentejo	NULL

Figura 25 – MySQL – Insert Vinicola

### Tabela Fornece

```
INSERT INTO Fornece VALUES ( "", "Malbec" , 2012, 2, 1);
```

```
INSERT INTO Fornece VALUES ( "", "Merlot", 2014, 4, 3);
```

```
INSERT INTO Fornece VALUES ( "", "Chardonnay", 2014, 1, 4);
```

```
INSERT INTO Fornece VALUES ( "", "Malbec" , 2012, 2, 2);
```

Cod_Fornece	Tipo_Uva	Safra	Cod_Quinta	Cod_Vinicola
1	Malbec	2012	2	1
2	Merlot	2014	4	3
3	Chardonnay	2014	1	4
4	Malbec	2012	2	2

Figura 26 – MySQL – Insert Fornece

### Tabela Vinho

```
INSERT INTO Vinho VALUES ( "", "Bela Aurora", 2012, "Tinto" 3);
```

```
INSERT INTO Vinho VALUES ( "", "Herdade dos 4", 1990, "Verde", 4);
```

```
INSERT INTO Vinho VALUES ( "", "Belo Traje", 2012, "Verde", 1);
```

```
INSERT INTO Vinho VALUES ( "", "Ouro Rubi", 2019, "Rose", 3);
```

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola
1	Bela Aurora	2012	Tinto	3
2	Herdade dos 4	1990	Verde	4
3	Belo Traje	2012	Verde	1
4	Ouro Rubi	2019	Rose	3

Figura 27 – MySQL – Insert Vinho

---

## ALTERANDO DADOS

Os dados inseridos através do comando **INSERT INTO**, podem ser manipulados por meio de comandos que serão apresentados a seguir:

**UPDATE** – Altera o valor existente no campo de uma tabela.

**UPDATE** nome\_tabela **SET** nome\_atributo = novo\_dado

Para que esse comando se torne mais preciso podemos utilizar a condição **WHERE**.

**UPDATE** nome\_tabela **SET** nome\_atributo = novo\_dado **WHERE** condição.

Exemplo.:

**UPDATE** Quinta **SET** Telefone = 916354876 **WHERE** Cod\_Quinta = 3;

Neste exemplo estaríamos alterando os dados previamente inserido no campo Telefone da Vinicola 3 para o novo valor de 916354876.

**DELETE** – Apaga os dados inseridos em uma tabela.

**DELETE FROM** nome\_tabela;

Caso queira apagar apenas um registo, pode-se utilizar o comando **DELETE** em conjunto com **WHERE**.

**DELETE FROM** nome\_tabela **WHERE** condição;

## CONSULTAS UTILIZANDO CONDIÇÕES

Existem algumas cláusulas de seleção que permitem decidirmos quais termos serão exibidos em nossas consultas, sendo extremamente úteis no refinamento de informações.

### WHERE

A cláusula **WHERE** é utilizada fornecendo uma condição específica de quais informações desejamos invocar em nossa consulta, podemos definir que só queremos apresentar as informações dos clientes que sejam maiores de idade, ou apenas dos clientes que vivem em Lisboa, a seguir apresentaremos exemplos relacionados a nossa Base de Dados em desenvolvimento.

**SELECT \* FROM** nome\_tabela **WHERE** condição;

Criaremos a seguir uma consulta de nossa tabela **Quinta** invocando apenas as informações onde o atributo **Regiao** contenha o valor “Norte”, para isso utilizaremos a seguinte linha de comando:

**SELECT \* FROM** Quinta **WHERE** Regiao = “Norte”;

```
mysql> SELECT * FROM Quinta WHERE Regiao = "Norte";
```

Cod_Quinta	NIF	Nome	regiao	Telefone
2	145783290	Folha D'Ouro	Norte	235987345
4	276980672	Castelo do Norte	Norte	276243987

Figura 28 – MySQL – Where Quinta

Em outro exemplo realizamos uma consulta na tabela **Vinho** onde queremos o retorno das safras antes do ano 2000.

**SELECT \* FROM** Vinho **WHERE** Safra < 2000;

```
mysql> SELECT * FROM Vinho WHERE Safra < 2000;
```

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola
2	Herdade dos 4	1990	Verde	4

Figura 29 – MySQL – Where Vinho

---

## ORDER BY

A clausula **ORDER BY** permite exibir de maneira ordenada ascendente ou decrescente as informações invocadas em nossa consulta.

Para ordenação ascendente utilizamos a linha de comando:

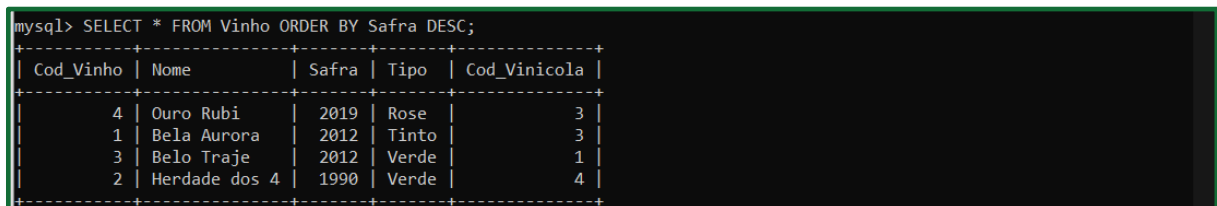
```
SELECT * FROM nome_tabela ORDER BY nome_atributo ASC
```

Para ordenação decrescente utilizamos a linha de comando:

```
SELECT * FROM nome_tabela ORDER BY nome_atributo DESC
```

Utilizaremos a tabela vinho para exemplificar uma consulta ordenada pela safra, iniciando pela mais recente e terminando com a mais antiga:

```
SELECT * FROM Vinho ORDER BY Safra DESC;
```



```
mysql> SELECT * FROM Vinho ORDER BY Safra DESC;
```

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola
4	Ouro Rubi	2019	Rose	3
1	Bela Aurora	2012	Tinto	3
3	Belo Traje	2012	Verde	1
2	Herdade dos 4	1990	Verde	4

Figura 30 – MySQL – Order By Vinho

## FUNÇÕES DE AGREGAÇÃO

Antes de darmos início a este capítulo, iremos implementar um novo atributo em nossa tabela **Vinho** utilizando recursos já abordados anteriormente, neste caso o comando **ALTER TABLE**, para inserirmos na tabela já existente o atributo preço e o comando **UPDATE** para inserirmos dados nos campos deste novo atributo.

**ALTER TABLE** Vinho **ADD** Preço **DECIMAL(6,2)** **NOT NULL**;

```
mysql> ALTER TABLE Vinho ADD Preço DECIMAL(6,2) NOT NULL;
```

Figura 31 – MySQL – Alter Vinho Preço

**IMPORTANTE:** Nesta linha foi nos apresentado mais um tipo de dado, neste caso o **DECIMAL**. Este dado é utilizado para valores decimais com casas separadas por virgula, o número entre os parenteses determina o tamanho do número e a quantidade de casas: **DECIMAL(tamanho, casas)**.

Desta maneira nossa tabela **Vinho** apresentaria agora o atributo **Preço** em sua estrutura.

```
mysql> desc vinho;
```

Field	Type	Null	Key	Default	Extra
Cod_Vinho	int	NO	PRI	NULL	auto_increment
Nome	varchar(20)	NO	UNI	NULL	
Safra	int	NO		NULL	
Tipo	varchar(20)	NO		NULL	
Cod_Vinicola	int	NO	MUL	NULL	
Preço	decimal(6,2)	NO		NULL	

Figura 32 – MySQL – Tabela Vinho Preço

Porém se listarmos os dados inseridos em nossa tabela, os campos referentes ao atributo Preço ainda não possuirão valores, pois esses valores ainda não foram inseridos.

```
mysql> select * from vinho;
```

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola	Preço
1	Bela Aurora	2012	Tinto	3	0.00
2	Herdade dos 4	1990	Verde	4	0.00
3	Belo Traje	2012	Verde	1	0.00
4	Ouro Rubi	2019	Rose	3	0.00

Figura 33 – MySQL – Dados Vinho

Logo precisamos inserir os dados que desejamos nos campos deste atributo e para isso utilizaremos o comando **UPDATE**.

**UPDATE** Vinho **SET** Preco = 85 **WHERE** Cod\_Vinho = 1;

**UPDATE** Vinho **SET** Preco = 120 **WHERE** Cod\_Vinho = 2;

**UPDATE** Vinho **SET** Preco = 30 **WHERE** Cod\_Vinho = 3;

**UPDATE** Vinho **SET** Preco = 12.5 **WHERE** Cod\_Vinho = 4;

```
mysql> select * from vinho;
```

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola	Preco
1	Bela Aurora	2012	Tinto	3	85.00
2	Herdade dos 4	1990	Verde	4	120.00
3	Belo Traje	2012	Verde	1	30.00
4	Ouro Rubi	2019	Rose	3	12.50

Figura 34 – MySQL – Dados Vinho Preco

Agora já com os dados de preço inserido, podemos dar continuidade a introdução dos conceitos de **FUNÇÕES DE AGREGAÇÃO**.

As **FUNÇÕES DE AGREGAÇÃO** são funções SQL que permitem executar uma operação aritmética nos valores de uma coluna em todos os registos de uma tabela e são elas:

**MIN** - Retorna o registo com menor valor da coluna seleccionada.

**SELECT MIN(nome\_atributo) FROM nome\_tabela;**

Ex.: **SELECT MIN(Preco) FROM Vinho;**

```
mysql> SELECT MIN(Preco) FROM Vinho;
```

MIN(Preco)
12.50

1 row in set (0.00 sec)

Figura 35 – MySQL – Função MIN



**MAX** - Retorna o registo com maior valor da coluna seleccionada.

**SELECT MAX(nome\_atributo) FROM nome\_tabela;**

**Ex.: SELECT MAX(Preco) FROM Vinho;**

```
mysql> SELECT MAX(Preco) FROM Vinho;
+-----+
| MAX(Preco) |
+-----+
|      120.00 |
+-----+
```

Figura 36 – MySQL – Função MAX

**SUM** - Soma os valores dos campos de uma determinada coluna.

**SELECT SUM(nome\_atributo) FROM nome\_tabela;**

**Ex.: SELECT SUM(Preco) FROM Vinho;**

```
mysql> SELECT SUM(Preco) FROM Vinho;
+-----+
| SUM(Preco) |
+-----+
|      247.50 |
+-----+
```

Figura 37 – MySQL – Função SUM

**AVG** – Informa a média aritmética dos valores de uma determinada coluna.

**SELECT AVG(nome\_atributo) FROM nome\_tabela;**

**Ex.: SELECT AVG(Preco) FROM Vinho;**

```
mysql> SELECT AVG(Preco) FROM Vinho;
+-----+
| AVG(Preco) |
+-----+
|  61.875000 |
+-----+
```

Figura 38 – MySQL – Função AVG

**COUNT** - Informa o número de registos de uma tabela e ou coluna.

**SELECT COUNT(\*) FROM nome\_tabela;** (Conta o número de elemento de uma tabela)

Ex.: **SELECT COUNT(\*) FROM Vinho;**

```
mysql> SELECT COUNT(*) FROM Vinho;
+-----+
| COUNT(*) |
+-----+
|         4 |
+-----+
```

Figura 39 – MySQL – Função COUNT

**SELECT COUNT (DISTINCT nome\_atributo) FROM nome\_tabela;** (Conta o número de registos de uma coluna sem contar repetições)

Ex. **SELECT COUNT(DISTINCT Safra) FROM Vinho;**

```
mysql> select count(distinct safra) from vinho;
+-----+
| count(distinct safra) |
+-----+
|                 3 |
+-----+
```

Figura 40 – MySQL – Função COUNT DISTINCT

## OPERADORES ARITMÉTICOS

Podemos utilizar operações para realizar cálculos com os dados de uma tabela. É possível usarmos essas ferramentas para atualizarmos campos ou mesmo para criação de novos, a seguir uma lista com os principais operadores.

### OPERADORES ARITIMÉTICOS

+ Soma

– Subtração

/ Divisão

\* Multiplicação

% ou **MOD** Módulo (Seleciona o resto de uma divisão)

**DIV** Divisão inteira

Exemplificaremos a seguir o uso de alguns operadores:

**UPDATE** vinho **SET** Preco = Preco \* 1.1;

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola	Preco
1	Bela Aurora	2012	Tinto	3	93.50
2	Herdade dos 4	1990	Verde	4	132.00
3	Belo Traje	2012	Verde	1	33.00
4	Ouro Rubi	2019	Rose	3	13.75

Figura 41 – MySQL – Operadores

Podemos também especificar um dado específico a qual desejamos realizar a atualização através do **WHERE**.

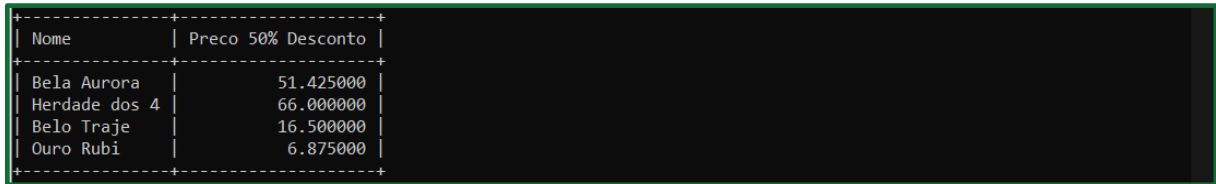
**UPDATE** vinho **SET** Preco = Preco \* 1.1 **WHERE** Nome = “Bela Aurora”;

Cod_Vinho	Nome	Safra	Tipo	Cod_Vinicola	Preco
1	Bela Aurora	2012	Tinto	3	102.85
2	Herdade dos 4	1990	Verde	4	132.00
3	Belo Traje	2012	Verde	1	33.00
4	Ouro Rubi	2019	Rose	3	13.75

Figura 42 – MySQL – Operadores

É possível também gerar consultas utilizando os operadores;

`SELECT Nome, Preço * 1.2 AS 'Preço Atualizado' FROM vinho;`



Nome	Preço 50% Desconto
Bela Aurora	51.425000
Herdade dos 4	66.000000
Belo Traje	16.500000
Ouro Rubi	6.875000

Figura 43 – MySQL – Operadores Consulta

**NOTA.:** Uma **CONSULTA** não altera o valor dos dados da tabela consultada.

---

## VIEWS

View é uma tabela “virtual” originada a partir de uma tabela existente através de uma consulta pré-definida. Como o próprio nome diz, ela representa uma visão de dados e não contém dados. Com ela você tem a ilusão que está vendo uma tabela que não existe. Claro que o que você vê nesta tabela existe de outra forma no banco de dados. Uma das maiores utilizações das Views, é a criação de “Tabelas Virtuais” específicas para consulta de determinados perfis de usuários. Qualquer alteração nos dados de uma Tabela que foi associada para a criação de uma Views, refletirá também nos dados desta View.

---

## VANTAGENS DA UTILIZAÇÃO DE VIEWS

**Segurança** - Evita que alguns campos ou linhas sejam acedidos por alguns tipos de utilizadores.

**Confidencialidade** - Evita que os utilizadores possam consultar dados de acesso reservado.

**Simplicidade** - A forma como se acede a uma View é exatamente igual à forma como se acede a uma tabela.

**Integração** - Possibilidade de consultar numa única View dados de várias tabelas.

**Facilidade** - Um comando SELECT simples realizado a uma View pode ser equivalente a um comando SELECT bastante complexo.

**Independência** - Reforço da independência dos dados das aplicações a que se destinam.

---

## CRIANDO VIEWS

Para criação de Views utilizamos o comando **CREATE VIEW**. É possível criar uma View de uma tabela completa, de atributos previamente selecionado ou até mesmo uma view com dados de mais de uma tabela, a seguir mostraremos exemplos baseados em nossa base de dados Vinhos. As views podem ser consultadas utilizando o comando **SELECT FROM**.

**CREATE VIEW** nome\_view **AS SELECT \* FROM** nome\_tabela,

**EX.: CREATE VIEW** Quinta\_View **AS SELECT \* FROM** Quinta;

```
mysql> select * from Quinta_View;
```

	Cod_Quinta	NIF	Nome	regiao	Telefone
1	265712957		Quinta do Olivardo	Algarve	914345678
2	145783290		Folha D'Ouro	Norte	235987345
3	135896234		Quinta o Alentejano	Alentejo	NULL
4	276980672		Castelo do Norte	Norte	276243987

Figura 44 – MySQL – VIEW

No exemplo a seguir criaremos uma View que nos mostre apenas as quintas de uma determinada região, para isso utilizaremos o comando:

**CREATE VIEW** nome\_view **AS SELECT \* FROM** nome\_tabela **WHERE** condição;

**EX.: CREATE VIEW** Quinta\_Norte **AS SELECT \* FROM** Quinta **WHERE** Regiao = “Norte”;

	Cod_Quinta	NIF	Nome	regiao	Telefone
2	145783290		Folha D'Ouro	Norte	235987345
4	276980672		Castelo do Norte	Norte	276243987

Figura 45 – MySQL – VIEW NORTE

Podemos criar Views com apenas os atributos que são relevantes ao perfil que ira consultá-las. No exemplo a seguir criaremos uma View chamada lista de preços que contenha apenas o nome e o preço de cada vinho.

**CREATE VIEW** nome\_view **AS SELECT** nomes\_atributos **FROM** nome\_tabela;

**EX.: CREATE VIEW** Lista\_Preco **AS SELECT** Nome, Preco **FROM** Vinho;

	Nome	Preco
	Bela Aurora	102.85
	Herdade dos 4	132.00
	Belo Traje	33.00
	Ouro Rubi	13.75

Figura 46 – MySQL – VIEW LISTA PREÇO

Como já dito, é possível também a criação de views com dados provenientes de mais de uma tabela. A seguir criaremos uma view que nos mostre o nome do vinho e o nome da Vinicola que o produziu.

```
CREATE VIEW nome_view AS SELECT nome_tabela1.nome_atributo,
nome_tabela2.nome_atributo FROM nome_tabela1, nome_tabela2;
```

Ex.: `CREATE VIEW Vinho_Origem AS SELECT Vinho.Nome AS Nome_Vinho, Vinicola.Nome AS Nome_Vinicola FROM Vinho, Vinicola WHERE Vinho.Cod_Vinicola = Vinicola.Cod_Vinicola;`



```
mysql> select * from Vinho_Origem;
```

Nome_Vinho	Nome_Vinicola
Bela Aurora	Arcos do Castelo
Herdade dos 4	Vale do Rio
Belo Traje	Monte Mor
Ouro Rubi	Arcos do Castelo

Figura 47 – MySQL – VIEW VINHO ORIGEM

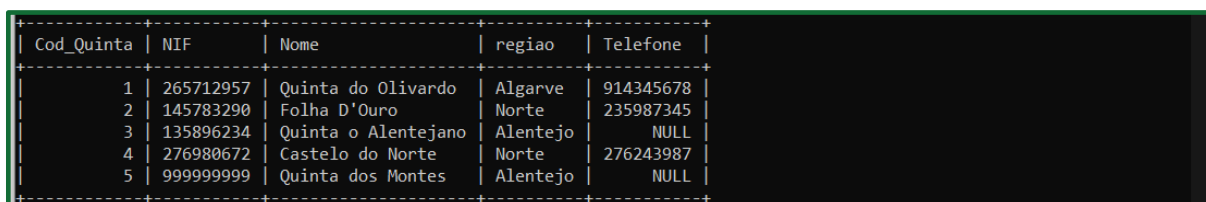
**IMPORTANTE:** Devemos observar algumas particularidades desta criação, a primeira delas é que foi necessário alterar o nome de nossas colunas pois ambas chamavam “Nome” e a criação não aceita nomes iguais, logo alteramos para Nome.Vinho e Nome.Vinicola através do comando `AS`. A segunda particularidade é a condição `WHERE` utilizada, criamos um vínculo entre os atributos `Cod_Vinicola` de ambas as tabelas para associarmos o vinho que foi produzido pela específica vinícola.

## MANIPULANDO VIEWS

Se houver necessidade de inserirmos novos dados em uma view, podemos utilizar o comando `INSERT INTO`.

```
INSERT INTO nome_view VALUES(dados);
```

Ex.: `INSERT INTO Quinta_View VALUES('999999999', 'Quinta dos Montes', 'Alentejo', NULL);`



Cod_Quinta	NIF	Nome	regiao	Telefone
1	265712957	Quinta do Olivardo	Algarve	914345678
2	145783290	Folha D'Ouro	Norte	235987345
3	135896234	Quinta o Alentejano	Alentejo	NULL
4	276980672	Castelo do Norte	Norte	276243987
5	999999999	Quinta dos Montes	Alentejo	NULL

Figura 48 – MySQL – VIEW INSERT INTO

**NOTA:** Importante lembrarmos que a tabela Quinta não será alterada, apenas haverá alteração na View Quinta\_View.

Podemos também atualizar os dados de uma view através do comando UPDATE.

**UPDATE** nome\_view **SET** atributo = novo\_valor **WHERE** condição;

EX.: **UPDATE** Quinta\_View **SET** NIF = 324872674 **WHERE** Cod\_Quinta = 5;

Cod_Quinta	NIF	Nome	regiao	Telefone
1	265712957	Quinta do Olivardo	Algarve	914345678
2	145783290	Folha D'Ouro	Norte	235987345
3	135896234	Quinta o Alentejano	Alentejo	NULL
4	276980672	Castelo do Norte	Norte	276243987
5	324872674	Quinta dos Montes	Alentejo	NULL

Figura 49 – MySQL – VIEW UPDATE

Através do comando DELETE podemos apagar os dados de uma view.

**DELETE FROM** nome\_view **WHERE** condição;

EX.: **DELETE FROM** Quinta\_View **WHERE** Cod\_Quinta = 5;

Cod_Quinta	NIF	Nome	regiao	Telefone
1	265712957	Quinta do Olivardo	Algarve	914345678
2	145783290	Folha D'Ouro	Norte	235987345
3	135896234	Quinta o Alentejano	Alentejo	NULL
4	276980672	Castelo do Norte	Norte	276243987

Figura 50 – MySQL – VIEW DELETE

E finalmente para se excluir um view utilizamos o comando DROP VIEW.

**DROP VIEW** nome\_view;

EX.: **DROP VIEW** Quinta\_View;



---

## TRANSAÇÕES

Transações são utilizadas quando há necessidade de alterar dois atributos em comum onde um exerce influência sobre o outro. O exemplo mais comum para explicar esse processo é através do funcionamento de uma transação bancária. Um cliente quando realiza uma transferência bancária retira um valor de sua conta e a transfere para uma outra conta. Imagine que temos uma Tabela Chama CLIENTE1 e outra chama CLIENTE2, ambas possuem um atributo chamado SALDO. Seguindo com a suposição vamos dizer que o valor do atributo saldo no CLIENTE1 é 1.000,00, enquanto no CLIENTE2 o atributo saldo tem o valor 700,00. Para atualizarmos as duas tabelas após o CLIENTE1 transferir 100,00 para o CLIENTE2, poderíamos utilizar o comando UPDATE, assim teríamos de executar o comando duas vezes, uma para atualizar o valor do saldo CLIENTE1 para 900,00 e outro para atualizar o valor do saldo do CLIENTE2 para 800,00. Porém o que aconteceria se o UPDATE da tabela CLIENTE2 falhasse ou fosse esquecido? Neste caso a tabela CLIENTE1 seria atualizada para 900,00 porém o CLIENTE2 não teria de 100,00 acrescido em seu saldo.

Para evitar este tipo de problema é que utilizamos as transações, pois através dela o processo só é concluído se as duas tabelas forem atualizadas. Alguns comandos utilizados nas transações são:

**BEGIN ou START TRANSACTION** - Indica onde ela deve começar, então os comandos SQL a seguir estarão dentro desta transação.

**COMMIT** - Indica o fim de uma transação, neste momento tudo o que foi manipulado passa a fazer parte do banco de dados normalmente e operações diversas passam a enxergar o que foi feito.

**ROLLBACK** - Também fecha o bloco da transação e é a indicação que a transação deve ser terminada, mas tudo que tentou ser feito deve ser descartado porque alguma coisa errada aconteceu e ela não pode terminar normalmente. Nada realizado dentro dela será perdurado no banco de dados.

O conceito de transação só se aplica aos comandos que fazem a manipulação de dados, ou seja, aos comandos INSERT, UPDATE e DELETE e para utilizar uma transação com o MYSQL,

as tabelas têm de ser criadas através de um tipo que suporte de transações, como é o caso mais popular, o InnoDB.

Por defeito o PHPMyAdmin cria suas tabelas com o tipo MyISAM, logo para que as transações funcionem, devemos alterar o esse tipo para InnoDB através do comando:

**ALTER TABLE** nome\_tabela **ENGINE** = InnoDB;

---

## UTILIZANDO TRANSAÇÕES

As transações são criadas através das estruturas:

<b>BEGIN;</b>  Comandos de manipulação;  <b>COMMIT;</b>	<b>EX.:</b>  <b>BEGIN;</b>  <b>UPDATE</b> CONTA1 <b>SET</b> saldo = saldo - 100;  <b>UPDATE</b> CONTA2 <b>SET</b> saldo = saldo + 100;  <b>COMMIT;</b>
---	--

Neste caso as duas alterações seriam implementadas após o COMMIT.

<b>BEGIN;</b>  Comandos de manipulação;  <b>ROLLBACK;</b>	<b>EX.:</b>  <b>BEGIN;</b>  <b>UPDATE</b> CONTA1 <b>SET</b> saldo = saldo - 100;  <b>UPDATE</b> CONTA2 <b>SET</b> saldo = saldo + 100;  <b>ROLLBACK;</b>
---	--

Neste Segundo exemplo o ROLLBACK anularia as modificações de ambas as tabelas.

---

## CONCLUSÃO

O objetivo deste trabalho foi compartilhar uma introdução a criação de bases dados utilizando de exemplos segundo uma necessidade imaginária. Abordamos desde o conceito básico sobre o que é uma base de dados até sua criação em um ambiente SQL. Passamos por tópicos como a criação e normalização de um diagrama EDR baseado em uma necessidade específica, criamos e manipulamos Tabelas, Atributos, Dados e Views, utilizamos operadores aritméticos, funções de agregação entregamos ao fim uma base de dados simples, porém funcional ao exercício proposto.

É aconselhado caso o leitor queira se aprofundar ainda mais neste assunto, que utilize os meios de pesquisas on-line onde haverá uma enorme quantidade de informações extremamente úteis sobre esta matéria, deixamos como sugestão fontes como “[stackoverflow.com](https://stackoverflow.com)”, “[www.w3schools.com](http://www.w3schools.com)” e “[www.bosontreinamentos.com.br](http://www.bosontreinamentos.com.br)”.

Como menção final, porém não menos importante, agradeço o empenho e disposição do formador Tiago Lopes que através do compartilhamento de seus conhecimentos, tornou possível a realização deste trabalho.

---

## WEBGRAFIA

Stack Overflow

*Stack Overflow - Where Developers Learn, Share, & Build Careers*

W3Schools

*W3Schools Online Web Tutorials*

Bosón Treinamentos em Ciência e Tecnologia

*Bóson Treinamentos - Cursos de Tecnologia (bosontreinamentos.com.br)*