

# CASO DE ESTUDO: SQL – Introdução através de exemplos

---

**José Luís Borges**

**Mai de 2009**

**Versão 01**

Este capítulo apresenta um caso de estudo que consiste na utilização da linguagem SQL para as tarefas de gestão e manipulação da informação de uma base de dados relacional. A linguagem SQL (Structured Query Language) estabelece um standard de comunicação com sistemas de gestão de bases de dados, apresentando a característica de ser independente da ferramenta utilizada para armazenar as tabelas relacionais. Todas as instruções apresentadas neste capítulo foram testadas no Microsoft Access 2007 e no MySQL 5.1, duas populares ferramentas para gestão de bases de dados relacionais.

Será utilizado um exemplo baseado no caso de estudo dedicado à modelação conceptual de classes para ilustrar a utilização da linguagem SQL na criação de tabelas, inserção dos dados, actualização dos dados e na consulta das tabelas. De referir que não é objectivo do presente capítulo apresentar uma definição da linguagem SQL mas apenas ilustrar as suas potencialidades através de exemplos de utilização.

Na Secção 1 é apresentado o exemplo que serve de suporte ao capítulo, na Secção 2 é exemplificada a utilização das instruções de criação, actualização e consulta das tabelas com um pequeno exemplo de uma tabela. Na Secção 4 ilustra-se com maior detalhe a instrução de criação de tabelas, na Secção 5 a instrução para inserção de dados nas tabelas e na Secção 6 a instrução para consulta das tabelas. Finalmente, na Secção 7 é exemplificado o mecanismo para criação de vistas das tabelas. Em anexo é disponibilizada uma listagem de todas as instruções para criação das tabelas da base de dados e para a inserção nas tabelas de todos os registos utilizados nos exemplos.

## 1. Uma base de dados com informação sobre produção de vinhos

Neste capítulo vamos recorrer a uma base de dados baseada no exemplo utilizado para o estudo de um modelo conceptual de classes correspondente a um sistema de informação vocacionado para a gestão do negócio de um produtor de vinhos. A Figura 1 apresenta o diagrama conceptual de classes relativo à base de dados considerada. De acordo com o diagrama, a empresa possui um conjunto de propriedades agrícolas, cada uma associada a uma determinada região vinícola. São produzidas diversas marcas de vinho, sendo cada vinho caracterizado pela sua marca, teor alcoólico, preço, tara das garrafas comercializadas, número de garrafas produzidas anualmente e número de meses que o vinho passa em estágio em barricas de madeira. O tipo de vinho permite indicar se um determinado vinho é, por exemplo, um vinho tinto ou um espumante. Uma marca de vinho pode estar associada a uma, ou em alguns casos,

a mais de uma propriedade. Temos ainda o conceito de castas que permite indicar qual a percentagem de cada tipo de uva utilizadas na produção do vinho de cada marca. Uma casta é caracterizada pela sua designação corrente (por exemplo touriga nacional ou tinta roriz), uma descrição textual das suas características, a cor (tinta ou branca) e a indicação se é uma casta de baixa, média ou elevada produtividade (sabe-se por exemplo que a touriga nacional é uma casta de baixa produtividade enquanto a tinta roriz é de elevada produtividade).

De referir que à base de dados representada pelo diagrama irão corresponder sete tabelas relacionais, cinco correspondentes a cada uma das classes e duas correspondentes às relações de muitos para muitos.

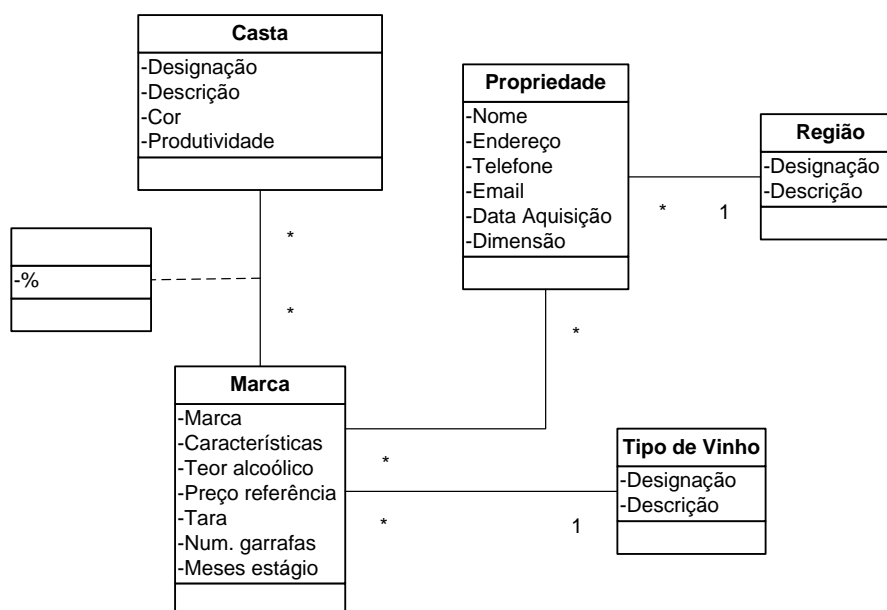


Figura 1. Diagrama conceitual de classes que representa o sistema de informação utilizado neste caso de estudo

## 2. Implementações da SQL

A linguagem SQL é reconhecida como o standard para os sistemas de gestão de base de dados. No entanto, as implementações disponibilizadas pelos sistemas comerciais suportam implementações que não são completamente compatíveis entre si, levando a que por vezes sejam necessários pequenos ajustes nas expressões. Neste capítulo vamos utilizar o MySQL para ilustrar o funcionamento das instruções apresentadas, tendo essas mesmas instruções sido também testadas no Microsoft Access 2007.

### MySQL

O MySQL é um sistema de gestão de base de dados em código aberto, sendo actualmente um dos sistemas mais populares. No endereço <http://www.mysql.com/> é possível encontrar versões para

download gratuito do sistema de gestão de base de dados assim como do pacote 'GUI tools 5.0' que disponibiliza, entre outras, a ferramenta 'MySQL Query Browser 1.2'.

A Figura 2 apresenta a janela principal do MySQL query browser sendo de realçar na sub-janela denominada 'schemata' a indicação (a bold) de qual a base de dados seleccionada. A opção para criação de uma nova base de dados surge ao clicar com o botão direito do rato na sub-janela 'shemata' e a base de dados pretendida pode ser activada através de um duplo clique no seu nome. As instruções de SQL devem ser inseridas na caixa de texto localizada por baixo da barra de menus e executadas pressionando o botão 'Execute'.

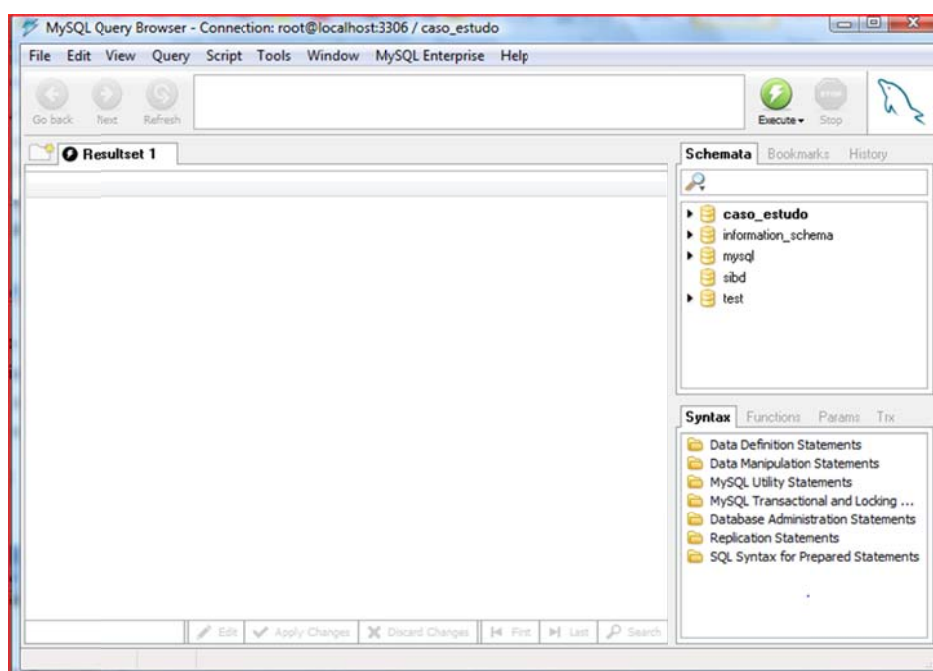
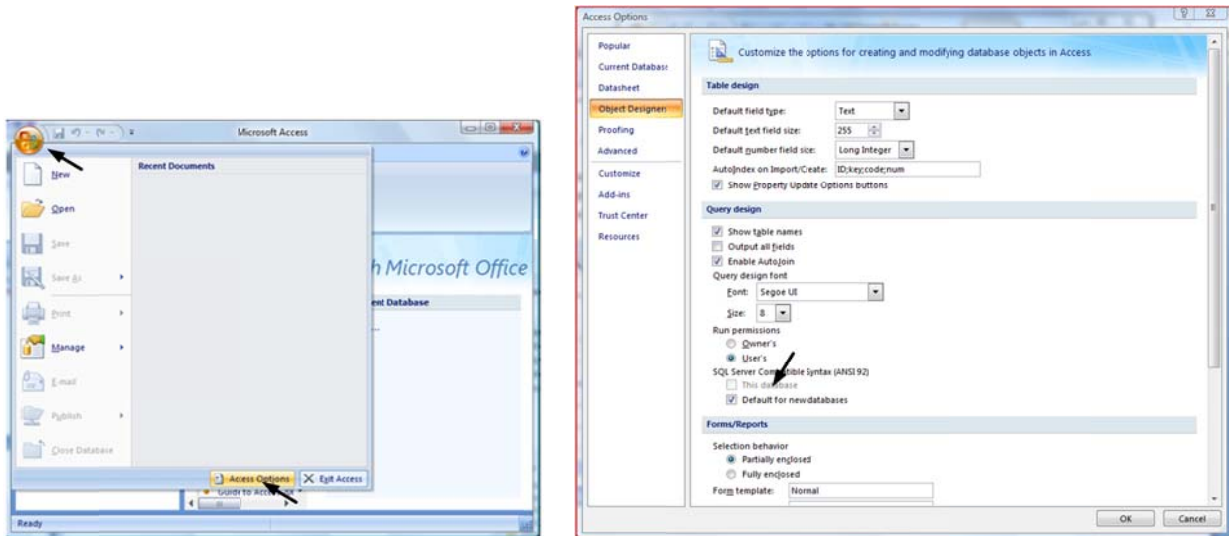


Figura 2. Interface do MySQL query browser

## 2.1. Microsoft Access

O Microsoft Access é um sistema de gestão de base de dados comercial bastante popular. Para utilizar a linguagem SQL neste sistema é importante verificar se está activa a opção de compatibilização do SQL com a versão standard. Como ilustrado na Figura 3 (a), tal pode ser verificado acedendo na janela de 'Access Options' ao conjunto de opções relativos aos 'Object Designers' e activando a opção relativa à compatibilização com o SQL versão ANSI 92, Figura 3 (b).



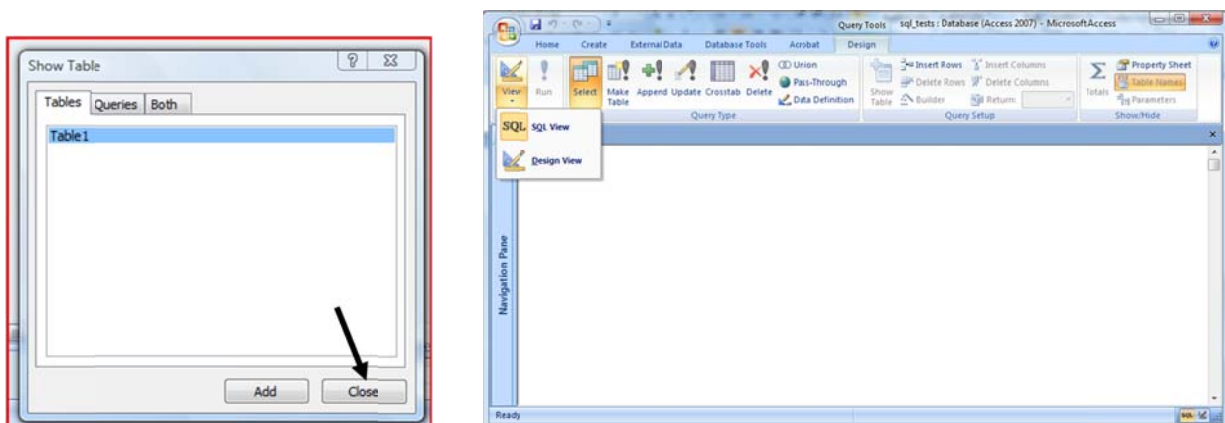
(a)

(b)

Figura 3. Configuração do MS Access para que este seja compatível com o SQL standard

Para executar no Access as instruções de SQL apresentadas neste capítulo é necessário executar os seguintes passos:

- criar uma base de dados;
- no menu de 'Create' seleccionar a opção 'Query Design';
- na janela com título 'Show table' pressionar 'close' sem seleccionar nenhuma tabela (ver Figura 4 (a));
- No canto superior esquerdo, no menu 'Design', seleccionar a opção 'SQL view' (ver Figura 4 (b));
- Escrever a instrução pretendida;
- Executar a instrução pressionando o botão 'Run'.



(a)

(b)

Figura 4. Janela para definição de instruções em SQL no MS Access

### 3. Introdução ao essencial do SQL

Nesta secção iremos utilizar uma das tabelas da base de dados para ilustrar os principais conceitos da linguagem SQL. Nas secções seguintes serão apresentados exemplos mais detalhados de cada um desses conceitos.

O primeiro passo na gestão de uma base de dados consiste na criação das tabelas que irão conter os dados. A sintaxe para a instrução de criação de tabelas é a seguinte:

```
CREATE TABLE nome_tabela(  
  coluna_1 TIPO_1  
  ...  
  coluna_2 TIPO_n  
  PRIMARY KEY(coluna_x, ...));
```

A expressão seguinte permite criar a tabela de regiões. O nome dado à tabela é 't\_regiao', sendo esta composta por três colunas. A coluna 'regiao\_id' permite guardar valores numéricos do tipo inteiro (INT) que correspondem ao identificador de cada região. Na definição desta coluna, a cláusula PRIMARY KEY permite indicar que o atributo 'regiao\_id' irá funcionar como chave primária da tabela. A coluna 'designacao\_regiao' permite guardar um texto de, no máximo, quinze caracteres (VARCHAR(15)) que identifique o nome das regiões. A utilização da cláusula NOT NULL obriga a que a designação da região não seja deixada em branco e a cláusula UNIQUE permite garantir que não sejam introduzidas duas regiões com a mesma designação. A coluna 'descricao' permite introduzir um texto (TEXT) com uma descrição das características de cada região. Ao definir a coluna como sendo do tipo TEXT, estamos a indicar que será permitido introduzir uma sequência de, no máximo, 65,535 caracteres. O tipo de dados VARCHAR é vocacionado para sequências de caracteres não muito longas e de tamanho variável, por outro lado, o tipo de dados TEXT é vocacionado para longas sequências de caracteres também de tamanho variável.

```
CREATE TABLE t_regiao(  
  regiao_id INT PRIMARY KEY,  
  designacao_regiao VARCHAR(15) NOT NULL UNIQUE,  
  descricao TEXT);
```

Para inserir dados nas tabelas da base de dados utiliza-se a instrução INSERT, que tem a sintaxe seguinte:

```
INSERT INTO nome_tabela [(coluna_1, coluna_2, ...)]  
VALUES (valor_coluna_1, valor_coluna_2, ...);
```

A instrução seguinte permite definir a região 'Douro' na tabela 't\_regiao', sendo atribuído o valor '1' para o atributo identificador e fornecida uma breve descrição da região:

```
INSERT INTO t_regiao (regiao_id, designacao_regiao, descricao)  
VALUES (1, 'Douro', 'A região do Douro localiza-se no Nordeste de Portugal,  
rodeada pelas serras do Marão ...');
```

No caso de pretender fornecer valores para todas as colunas da tabela e pela ordem pela qual os atributos foram definidos aquando da criação da tabela é possível omitir a indicação das colunas pretendidas. A instrução seguinte ilustra essa possibilidade:

```
INSERT INTO t_regiao  
VALUES (2, 'Alentejo', 'O Alentejo é uma das maiores regiões ...');
```

É no entanto importante referir que tal opção deve ser ponderada cuidadosamente uma vez que instruções como esta deixam de estar correctas no caso de ser acrescentada ou eliminada uma coluna da tabela. Por exemplo, se acrescentarmos um atributo à classe Região que indique a área total de vinha plantada a instrução utilizada para a definição da região do Douro continuará válida desde que o novo atributo permita valores nulos, enquanto a instrução utilizada para a caracterização da região do Alentejo resultará num erro de execução.

Para consultar a informação das tabelas utiliza-se a instrução SELECT. A sintaxe do SELECT é a seguinte:

```
SELECT coluna_1, ..., coluna_n  
FROM tabela  
[WHERE condição ]  
[GROUP BY expressão ]  
[HAVING condição ]  
[ORDER BY expressão [ ASC | DESC ] ] ;
```

Por exemplo, podemos obter uma listagem de todas as regiões com a instrução:

```
SELECT * FROM t_regiao;
```

O símbolo '\*' é um *wildcard* que corresponde a seleccionar todas as colunas da tabela indicada. A Figura 5 ilustra a execução da query no MySQL query browser.

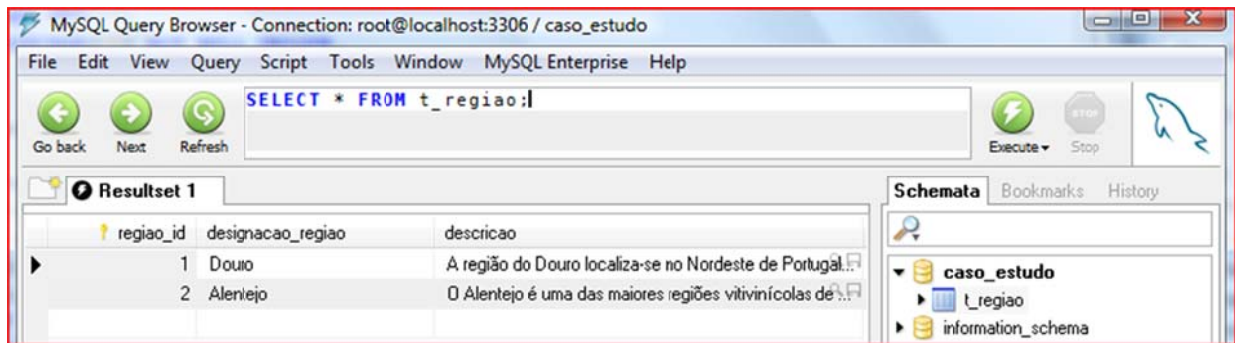


Figura 5. A execução da instrução que permite seleccionar todas as regiões no MySQL query browser

No caso de pretendermos seleccionar apenas a designação das regiões e apresentar os resultados ordenados alfabeticamente podemos utilizar a seguinte instrução:

```
SELECT designacao_regiao FROM t_regiao
ORDER BY designacao_regiao;
```



A instrução ALTER TABLE permite alterar a definição de uma tabela. Por exemplo, podemos acrescentar uma coluna que represente a dimensão em hectares de cada região através da seguinte instrução:

```
ALTER TABLE t_regiao ADD dimensao INT;
```

A mesma instrução pode ser utilizada para eliminar uma coluna utilizando DROP em vez do ADD ou para modificar uma coluna, utilizando MODIFY em vez de ADD.

Para alterar os valores dos registos já definidos podemos utilizar a instrução UPDATE. Por exemplo, para definir a dimensão da região do Douro podemos utilizar a seguinte expressão:

```
UPDATE t_regiao SET dimensao = 40000 WHERE regiao_id = 1;
```

De forma equivalente podemos inserir a informação relativa à dimensão da região vinícola do Alentejo.

```
UPDATE t_regiao SET dimensao = 22000 WHERE regiao_id = 2;
```

A Figura 6 mostra o resultado da instrução que permite a listagem de todas as colunas de todas as propriedades após a actualização da informação sobre a dimensão das regiões.

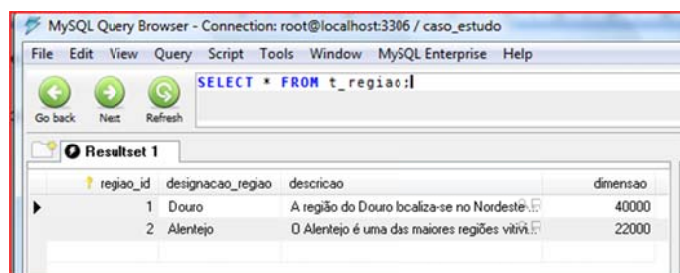


Figura 6. A execução da instrução que permite seleccionar todas as regiões já com o atributo que define a dimensão de cada região

## 4. Criação de tabelas

Nesta secção serão apresentadas as instruções necessárias para a criação das restantes tabelas correspondentes ao modelo representado pelo diagrama da Figura 1. A instrução seguinte permite criar a tabela correspondente à classe Propriedade:

```
CREATE TABLE t_propriedade(
propriedade_id INT PRIMARY KEY,
nome VARCHAR(50) NOT NULL,
endereço VARCHAR(50),
telefone VARCHAR(20),
email VARCHAR(30),
dimensao INT,
data_aquisicao DATE,
regiao_id INT,
FOREIGN KEY (regiao_id) REFERENCES t_regiao(regiao_id));
```

Na criação de uma tabela, a indicação de que um atributo tem as características de chave estrangeira pode ser efectuada através da cláusula FOREIGN KEY, uma vez que esta cláusula permite indicar, para um atributo, qual a tabela em que o mesmo é chave primária. Por exemplo, a expressão FOREIGN KEY(regiao\_id) REFERENCES t\_regiao(marca\_id) indica que o atributo 'regiao\_id' da tabela criada faz referência ao atributo com o mesmo nome que é chave primária da tabela 't\_regiao'. Desta forma, apenas serão admitidos na coluna 'regiao\_id' da tabela 't\_propriedade' valores que constem na coluna correspondente da tabela 't\_regiao' obrigando, assim, a que uma propriedade só possa ser definida para uma região já existente. De realçar ainda a utilização do tipo de dados DATE, que define o atributo 'data\_aquisicao' como sendo uma data do tipo 'ano/mês/dia' e que permite registar a data de aquisição da propriedade.

A tabela correspondente à classe Tipo de Vinho pode ser criada com uma instrução equivalente à utilizada para a criação da tabela de Região. Neste caso, é utilizada a cláusula `AUTO_INCREMENT` que permite indicar que à coluna `'tipo_vinho_id'` seja associado um gerador de números inteiros sequenciais. Desta forma, de cada vez que um tipo de vinho é inserido na tabela é-lhe atribuído de forma sequencial um código identificador único.

```
CREATE TABLE t_tipo_vinho(  
  tipo_vinho_id INT PRIMARY KEY AUTO_INCREMENT,  
  designacao_tipo_vinho VARCHAR(15) NOT NULL UNIQUE,  
  descricao TEXT);
```

A instrução seguinte é utilizada para criação da tabela de Marca. De notar a utilização do tipo de dados `FLOAT` que permite inserir números não inteiros nas colunas correspondentes. A tabela tem o atributo `'tipo_vinho_id'` que tem as características de chave estrangeira, fazendo referência à tabela de Tipo de Vinho.

```
CREATE TABLE t_marca(  
  marca_id INT PRIMARY KEY,  
  marca VARCHAR(50) NOT NULL,  
  teor FLOAT,  
  tara FLOAT,  
  preco FLOAT,  
  num_garrafas INT,  
  meses_estagio FLOAT,  
  caracteristicas TEXT,  
  tipo_vinho_id INT,  
  FOREIGN KEY (tipo_vinho_id) REFERENCES t_tipo_vinho(tipo_vinho_id));
```

A tabela correspondente à associação binária entre as classes Marca e Propriedade necessita de uma atenção especial uma vez que, dado que a chave primária é composta por mais de um atributo, temos de utilizar uma sintaxe ligeiramente diferente. Assim, na expressão seguinte temos a instrução `PRIMARY KEY(marca_id, propriedade_id)` que define que a chave primária da tabela `'t_marca_propriedade'` é composta pelo par de atributos indicado. De referir que cada um dos atributos que compõem a chave primária tem as características de chave estrangeira, sendo necessário incluir as cláusulas correspondentes à caracterização de cada um.

```
CREATE TABLE t_marca_propriedade(  
marca_id INT,  
propriedade_id INT,  
PRIMARY KEY(marca_id, propriedade_id),  
FOREIGN KEY(marca_id) REFERENCES t_marca(marca_id),  
FOREIGN KEY(propriedade_id) REFERENCES t_propriedade(propriedade_id));
```

A instrução seguinte permite criar a tabela que mantém a informação sobre as castas:

```
CREATE TABLE t_casta(  
casta_id INT PRIMARY KEY,  
designacao_casta VARCHAR(25) NOT NULL UNIQUE,  
cor VARCHAR(6),  
produtividade VARCHAR(5),  
descricao TEXT);
```

Uma vez que a cor da casta é um atributo importante que será alvo de pesquisas frequentes vamos criar um índice associado a esse atributo. Um índice, quando definido para um atributo de uma tabela, permite melhorar o desempenho das consultas que envolvem esse mesmo atributo. A pesquisa de um valor num atributo ao qual não está associado um índice obriga a percorrer todas as linhas da tabela à procura desse mesmo valor, uma vez que não há qualquer informação acerca da linha da tabela em que esse valor pode ocorrer. Um índice para um atributo numa determinada tabela não é mais do que uma estrutura paralela à tabela que mantém esse mesmo atributo ordenado. Logo um índice torna um pouco menos eficiente a inserção e a eliminação de registos, mas torna muito mais eficiente a pesquisa. Na concepção de uma base de dados devem ser analisados cuidadosamente quais os atributos que, por serem de pesquisa frequente, podem beneficiar por terem um índice associado. De referir que, a instrução de criação de tabelas associa à respectiva chave primária um índice para que a pesquisa pelos seus valores seja eficiente. A instrução seguinte permite criar um índice para o atributo cor da tabela de castas:

```
CREATE INDEX IDXcor ON t_casta (cor);
```

A base de dados fica completa com a instrução para a criação da tabela que permite definir quais as castas utilizadas na produção de uma determinada marca:

```
CREATE TABLE t_marca_casta(  
marca_id INT,  
casta_id INT,  
percentagem FLOAT,  
PRIMARY KEY(marca_id, casta_id),  
FOREIGN KEY(marca_id) REFERENCES t_marca(marca_id),  
FOREIGN KEY(casta_id) REFERENCES t_casta(casta_id));
```

Por último, importa referir que uma tabela pode ser eliminada através da instrução DROP TABLE. A instrução seguinte permite apagar a tabela de propriedades:

```
DROP TABLE t_propriedade;
```

De forma a manter a integridade referencial, uma determinada tabela só pode ser eliminada se não existir na base de dados nenhuma outra tabela que possua como chave estrangeira a chave primária da tabela em questão. Assim, a tabela 't\_propriedade' só pode ser eliminada se a tabela 't\_marca\_propriedade' for previamente eliminada ou se a restrição FOREIGN KEY que faz referência à tabela de propriedades for previamente removida.

## Exercícios

Escreva uma instrução que permita:

1. alterar a tabela de castas de forma a ter um coluna adicional que permita registar as designações alternativas associadas a cada castas.
2. criar uma tabela para guardar a informação sobre as vinhas. Cada vinha é caracterizada por uma designação, pela área de plantação, ano de plantação, tipo de solo e altitude média. Cada vinha está associada a uma propriedade.
3. criar uma tabela para registar o número de pés plantados de cada casta em cada vinha.

## 5. Inserir e alterar os dados nas tabelas

Nesta secção iremos apresentar alguns exemplos adicionais das instruções de INSERT e UPDATE. Ao inserir registos numa tabela, é importante ter em especial atenção o preenchimento de atributos que tem o papel de chave estrangeira. Como foi referido anteriormente, o preenchimento de um registo numa tabela que possui uma chave estrangeira apenas pode ser efectuado após o preenchimento do registo que define o conceito correspondente à chave primária que está associada à chave estrangeira em questão. Por exemplo, a introdução da informação relativa a uma quinta na tabela de propriedades apenas pode ser efectuada após a região correspondente à propriedade ter sido definida. A instrução

seguinte permite definir uma propriedade situada na região do Douro, sendo portanto necessário assegurar que essa mesma região foi previamente definida:

```
INSERT INTO t_propriedade VALUES (1,'Quinta de Santa Teresa', 'Peso da
Régua', '254 223 344','santateresa@vinhos.pt', 60, 1);
```

A instrução seguinte permite inserir a região da Estremadura sem fornecer mais detalhe além da sua designação:

```
INSERT INTO t_regiao VALUES (6,'Estremadura', NULL, NULL);
```

A instrução anterior é equivalente à instrução seguinte:

```
INSERT INTO t_regiao (regiao_id, desginacao_regiao)
VALUES (6,'Estremadura');
```

Para inserir registos em tabelas em que ao atributo chave primária foi associada a cláusula AUTO\_INCREMENT podemos omitir o valor para esse atributo na instrução de INSERT. Por exemplo, na instrução seguinte é inserido um tipo de vinho sendo ao atributo 'tipo\_vinho\_id' associado o próximo valor inteiro disponível:

```
INSERT INTO t_tipo_vinho (designacao_tipo_vinho, descricao)
VALUES ('Tinto', 'produzidos a partir da fermentação de uvas Tintas');
```

No entanto, tal não impede que o valor para o identificador seja incluído na instrução de insert, tal como é ilustrado na seguinte instrução:

```
INSERT INTO t_tipo_vinho
VALUES (3, 'Branco', 'Os vinhos brancos tranquilos são feitos a partir da
fermentação de uvas sem pele.');
```

A instrução UPDATE permite alterar os valores existentes numa determinada tabela. Por exemplo, podemos alterar o telefone da propriedade 1 com a seguinte instrução:

```
UPDATE t_propriedade SET telefone = '259351397'
WHERE propriedade_id = 1;
```

Podemos acrescentar o indicativo nacional a todos os números de telefones definidos através da seguinte instrução:

```
UPDATE t_propriedade SET telefone = CONCAT("00351 ", telefone)
WHERE telefone IS NOT NULL;
```

A função CONCAT permite efectuar a concatenação de strings no MySQL sendo que no MS Access a expressão equivalente toma a seguinte forma "00351 " + telefone.

Finalmente, a instrução DELETE permite apagar registos de uma tabela. Por exemplo, a instrução seguinte permite apagar todas as propriedades da região do Douro. É necessário ter em atenção que caso exista uma Marca associada a uma propriedade situada na região do Douro essa propriedade só poderá ser apagada depois de eliminada a marca. Dito de outra forma, não é possível apagar um registo de uma tabela se o atributo que o identifica ocorrer numa outra tabela com o papel de chave estrangeira.

```
DELETE FROM t_propriedade WHERE regioao_id = 1;
```

## Exercícios

Escreva uma instrução que permita:

1. inserir na tabela de castas a informação relativa à casta 'Alvarinho' de cor branca e de média produtividade.
2. inserir a propriedade 'Quinta dos Álváros', situada em Longos Vales, Monção, adquirida em '2008/02/11' e incluída na região vinícola de Monção (será também necessário inserir a informação relativa à região).
3. inserir a marca 'Alvarinho dos Álváros', produzida na 'Quinta dos Álváros', que consiste num vinho branco, graduação de 13%, com preço de referência 12€, comercializado em garrafas de 75cl, com uma produção de 50000 garrafas e produzido exclusivamente com a casta Alvarinho.
4. acrescentar 1 mês ao tempo de estágio dos vinhos que actualmente passam 12 meses em estágio.

## 6. Consulta da base de dados

### 6.1. A instrução de SELECT

A instrução SELECT permite interrogar a base de dados sendo a seguir indicada a sintaxe da mesma:

```

SELECT coluna_1, ..., coluna_n
FROM tabela
[WHERE condição ]
[GROUP BY expressão ]
[HAVING condição ]
[ORDER BY expressão [ ASC | DESC ] ] ;

```

Por exemplo, a instrução seguinte permite seleccionar todas as colunas e todos os registos da tabela de castas sendo o resultado da instrução apresentado na figura:

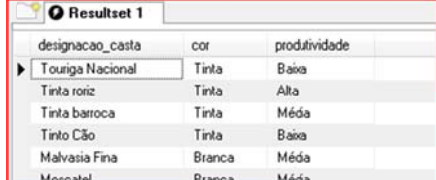
```
SELECT * FROM t_casta;
```



casta_id	designacao_casta	cor	produtividade	descricao
1	Touriga Nacional	Tinta	Baixa	Os bagos
2	Tinta roriz	Tinta	Alta	NULL
3	Tinta barroca	Tinta	Média	NULL
4	Tinto Cão	Tinta	Baixa	NULL
5	Malvasia Fina	Branca	Média	NULL
6	Moscatel	Branca	Média	NULL
7	Verdelho	Branca	Média	NULL
8	Touriga Franca	Tinta	Média	NULL
9	Tinta Amarela	Tinta	Baixa	NULL
10	Sousão	Tinta	Média	NULL
11	Maria Gomes	Branca	Alta	NULL
12	Arinto	Branca	Média	NULL
13	Baga	Tinta	Alta	NULL
14	Alicante Bouschet	Tinta	Média	NULL

Se pretendemos seleccionar apenas algumas colunas da tabela teremos de enumerar as colunas pretendidas, tal como no exemplo da instrução seguinte:

```
SELECT designacao_casta, cor, produtividade
FROM t_casta;
```



designacao_casta	cor	produtividade
Touriga Nacional	Tinta	Baixa
Tinta roriz	Tinta	Alta
Tinta barroca	Tinta	Média
Tinto Cão	Tinta	Baixa
Malvasia Fina	Branca	Média
Moscatel	Branca	Média

Para ordenar os resultados podemos fazer uso da cláusula ORDER BY. A instrução seguinte permite seleccionar as castas ordenadas pelo indicador de produtividade. De notar que a ordem das colunas também foi alterada.

```
SELECT produtividade, designacao_casta, cor,
FROM t_casta
ORDER BY produtividade;
```



produtividade	designacao_casta	cor
Alta	Tinta roriz	Tinta
Alta	Baga	Tinta
Alta	Maria Gomes	Branca
Baixa	Touriga Nacional	Tinta
Baixa	Tinto Cão	Tinta
Baixa	Tinta Amarela	Tinta
Média	Arinto	Branca

A cláusula ORDER BY, por defeito, ordena um atributo de forma ascendente (ASC) podendo a ordenação contrária ser obtida através da cláusula (DESC).

```
SELECT designacao_casta, cor, produtividade
FROM t_casta
ORDER BY produtividade DESC;
```



designacao_casta	cor	produtividade
Alicante Bouschet	Tinta	Média
Arinto	Branca	Média
Sousão	Tinta	Média
Touriga Franca	Tinta	Média
Verdelho	Branca	Média
Moscatel	Branca	Média
Malvasia Fina	Branca	Média

Podemos ainda restringir os resultados, por exemplo, às castas de elevada produtividade utilizando a cláusula WHERE. A instrução seguinte permite seleccionar as castas de produtividade alta e ordenar os resultados, primeiro por cor e, para cada cor, pela designação da casta. Uma vez que o atributo 'produtividade' é do tipo texto, o critério de selecção tem de ser colocado entre plicas ou aspas.

```
SELECT designacao_casta, cor, produtividade
FROM t_casta
WHERE produtividade = 'alta'
ORDER BY cor, designacao_casta;
```



designacao_casta	cor	produtividade
Maria Gones	Branca	Alta
Baga	Tinta	Alta
Tinta roriz	Tinta	Alta

A cláusula WHERE permite conjugar um número indeterminado de condições do tipo igual, menor, maior ou diferente utilizando os operadores lógicos AND, OR e NOT. Por exemplo, podemos seleccionar as marcas com preço superior a 10€ e com teor alcoólico inferior ou igual a 13% utilizando a seguinte expressão:

```
SELECT marca, teor, preco FROM t_marca
WHERE teor <= 13 AND preco > 10
ORDER BY preco DESC, marca;
```



marca	teor	preco
Quinta de Encosta - Vinhas Velhas	13	30

A cláusula ORDER BY, por defeito, ordena o atributo indicado de forma ascendente. Se pretendermos a ordenação inversa temos de utilizar a cláusula DESC para cada um dos atributos que se pretende ordenar dessa forma.

A cláusula BETWEEN permite especificar um intervalo de valores para um atributo numérico. A expressão seguinte ilustra a utilização do operador lógico OR juntamente com o operador BETWEEN. O resultado da expressão será o conjunto das marcas que custam mais de 15€ e possuem um teor alcoólico entre 14 e 15% ou que, alternativamente, custam menos de 10€.

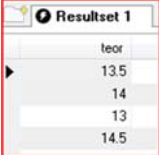
```
SELECT marca, teor, preco FROM t_marca
WHERE (teor BETWEEN 14 AND 15 AND preco > 15)
OR preco < 10
ORDER BY preco DESC, marca;
```



marca	teor	preco
Duas Casas - Reserva	14	40
Esporas - Garrafeira	14.5	30
Duas Casas	14	20
Tinto dos Freires	12.5	9
Quinta das Esporas	13	8
Branco dos Freires	11.5	7
Quinta da Encosta - Espumante	12.5	7
Duas Casas - Branco	12.5	6

Podemos eliminar repetições nos resultados ao utilizar a cláusula DISTINCT. A Expressão seguinte permite seleccionar o teor alcoólico das marcas cujo preço é superior a 12€. No caso de existirem várias marcas com o mesmo teor alcoólico que respeitem o critério de selecção iremos ter repetições desse teor nos resultados obtidos. Se utilizarmos a cláusula DISTINCT todas essas repetições serão eliminadas.

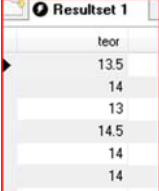
```
SELECT DISTINCT teor FROM t_marca
WHERE preco > 12;
```



teor
13.5
14
13
14.5

Como se pode ver a seguir a mesma instrução sem a cláusula DISTINCT retorna valores repetidos para o atributo seleccionado.

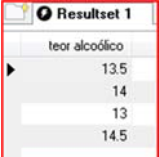
```
SELECT teor FROM t_marca
WHERE preco > 12;
```



teor
13.5
14
13
14.5
14
14

Para no resultado retornado alterar o nome de uma coluna podemos utilizar a cláusula AS, como é exemplificado na instrução seguinte. No exemplo apresentado é necessária a utilização das aspas apenas porque o nome pretendido inclui um espaço.

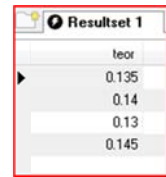
```
SELECT DISTINCT teor AS "teor alcoólico"
FROM t_marca
WHERE preco > 12;
```



teor alcoólico
13.5
14
13
14.5

É possível também incluir expressões na cláusula de SELECT. Por exemplo, para converter o teor alcoólico para uma variável entre 0 e 1 podemos utilizar a seguinte instrução:

```
SELECT DISTINCT teor/100 AS teor
FROM t_marca
WHERE preco > 12;
```

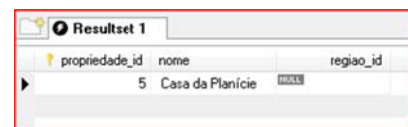


teor
0.135
0.14
0.13
0.145

Se na expressão anterior não fosse utilizado o operador AS, no MySQL a coluna retornada teria como cabeçalho a instrução definida e no Access o nome expr1000, pelo que nestes casos se recomenda a sua utilização.

Como vimos na Secção 2, a utilização da cláusula NOT NULL para um atributo na instrução de criação de tabelas determina que não sejam admitidos valores nulos para esse atributo. No entanto há atributos que pelas suas características poderão ter valores nulos. Por exemplo, o telefone de uma propriedade pode não ser conhecido na altura de introdução na base de dados da informação relativa a essa propriedade. O operador IS permite verificar a existência de valores nulos numa tabela. Por exemplo, a seguinte instrução permite identificar as propriedades para as quais não está atribuída uma região:

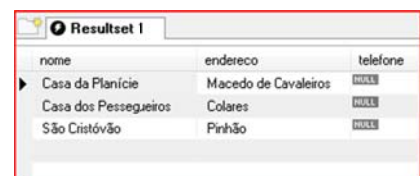
```
SELECT propriedade_id, nome, regioao_id
FROM t_propriedade
WHERE regioao_id IS NULL;
```



propriedade_id	nome	regiao_id
5	Casa da Planície	NULL

De forma semelhante, a instrução seguinte permite verificar quais as propriedades para as quais o telefone não está definido:

```
SELECT nome, endereco, telefone
FROM t_propriedade
WHERE telefone IS NULL;
```



nome	endereco	telefone
Casa da Planície	Macedo de Cavaleiros	NULL
Casa dos Pessegueiros	Colares	NULL
São Cristóvão	Pinhão	NULL

Como vimos anteriormente para a definição de uma restrição quando um atributo é do tipo VARCHAR (sequência de caracteres) temos de incluir o parâmetro de restrição entre plicas ou aspas. Dada a sua natureza, a pesquisa de variáveis de texto através da correspondência exacta é um problema delicado, uma vez que basta a existência de um espaço adicional entre duas palavras para que a selecção não identifique o registo pretendido. Por exemplo, se na tabela de propriedades utilizarmos a instrução seguinte para seleccionar o registo correspondente à Quinta dos Monges a query não retorna qualquer registo.

```
SELECT * FROM t_propriedade WHERE nome = 'Quinta dos Monges';
```

Uma análise cuidada do conteúdo da tabela das propriedades revela que na definição desta propriedade foi inadvertidamente inserido um espaço adicional entre 'dos' e 'Monges', daí a expressão não ter permitido identificar o registo pretendido. Este exemplo revela a dificuldade associada à definição de critérios de pesquisa com correspondência exacta para colunas do tipo VARCHAR.

O operador LIKE permite a pesquisa de strings através de padrões, em alternativa à correspondência exacta. Para tal são utilizados os dois seguintes wildcards:

- Qualquer sequência de zero, ou mais, caracteres '%'
- Um qualquer carácter '\_'

Para o MS Access, no caso de não ser activada a opção de compatibilidade com o ANSI 92 (ver secção 2.1) deve ser utilizado o carácter '\*' para uma sequência de caracteres e o carácter '?' para um qualquer carácter individual.

Por exemplo, podemos seleccionar todas as propriedades cujo nome começa por 'Quinta' através da seguinte instrução:

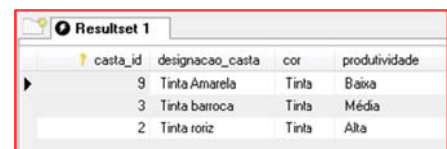
```
SELECT * FROM t_propriedade
WHERE nome LIKE 'Quinta%';
```



propriedade_id	nome	endereco	telefone
1	Quinta de Santa Teresa	Peso da Régua	254 223
2	Quinta da Encosta	Corregal do Sal	232 423
3	Quinta das Esporas	Reguengos de Monsaraz	266 587
4	Quinta dos Freires	Pinhão	254 654
6	Quinta dos Monges	Pinhão	254 777

Ou seleccionar todas as castas cuja designação começa por 'Tinta' através da instrução:

```
SELECT * FROM t_casta
WHERE designacao_casta LIKE 'Tinta%';
```



casta_id	designacao_casta	cor	produtividade
9	Tinta Amarela	Tinta	Baixa
3	Tinta barroca	Tinta	Média
2	Tinta roriz	Tinta	Alta

Para seleccionar todas as castas cuja designação tem um 'i' na segunda letra podemos usar a seguinte instrução:

```
SELECT * FROM t_casta
WHERE designacao_casta LIKE '_i%';
```



casta_id	designacao_casta	cor	produtividade
2	Tinta roriz	Tinta	Alta
3	Tinta barroca	Tinta	Média
4	Tinto Cão	Tinta	Baixa
9	Tinta Amarela	Tinta	Baixa

Ou todas as castas com um 'f' na designação através da seguinte instrução:

```
SELECT * FROM t_casta
WHERE designacao_casta LIKE '%f%';
```

casta_id	designacao_casta	cor	produtividade
5	Malvasia Fina	Branca	Média
8	Touiga Franca	Tinta	Média

Se for necessário procurar um dos wildcards dentro de uma sequência de caracteres teremos que preceder o wildcard com um caracter que irá forçar a que o wildcard seja interpretado literalmente. Vamos ilustrar esta funcionalidade através de um exemplo que pretende seleccionar as propriedades cujo email utiliza o caracter '\_', para tal temos que utilizar a seguinte sintaxe:

```
SELECT propriedade_id, nome, email
FROM t_propriedade
WHERE email LIKE '%=_%' ESCAPE '=';
```

propriedade_id	nome	email
4	Quinta dos Freires	q_freires@mail.pt

A cláusula ESCAPE '=' indica que o caracter '=' será utilizado para anular o contexto especial dos wildcards. No Access bastaria incluir o caracter reservado dentro de parêntesis rectos, como é ilustrado na instrução seguinte que permite seleccionar todas as propriedades que utilizam o 'underscore' no seu nome:

```
SELECT propriedade_id, nome, email
FROM t_propriedade WHERE nome LIKE '%[_]%';
```

## Exercícios

Escreva uma instrução que permita:

1. seleccionar as castas brancas de alta produtividade.
2. seleccionar as castas de alta produtividade, ordenadas por cor e, para cada cor, ordenadas pela sua designação.
3. seleccionar as marcas de vinho tinto com mais de 10 meses de estágio em madeira e cujo preço se situa entre os 10 e os 20€.

## 6.2. Atributos do tipo DATE

Os sistemas de gestão de base de dados oferecem, em geral, algumas funcionalidades específicas para a manipulação dos atributos do tipo DATE. A manipulação de datas é uma funcionalidade importante para a manutenção de registos históricos, merecendo por isso uma atenção especial. As funções de datas não têm um standard para os diferentes sistemas de gestão de bases de dados pelo que iremos ilustrar algumas das funcionalidades disponibilizadas no MySQL.

Por exemplo, se pretendermos identificar as propriedades adquiridas em 2002 podemos utilizar a seguinte instrução:

```
SELECT nome, data_aquisicao
FROM t_propriedade
WHERE data_aquisicao
BETWEEN '2002-01-01' AND '2002-12-31';
```

nome	data_aquisicao
Quinta das Esporas	2002-03-13

Para identificar as propriedades adquiridas nos últimos 4 anos precisamos de duas funções. A função CURDATE() que retorna a data correspondente ao dia de hoje e a função DATE\_SUB() que permite subtrair um determinado intervalo de tempo a uma data. Na instrução seguinte as funções referidas são utilizadas para obter a data correspondente ao dia de hoje menos um intervalo de 4 anos:

```
SELECT nome, data_aquisicao,
DATE_SUB(CURDATE(),INTERVAL 4 YEAR)
FROM t_propriedade
WHERE data_aquisicao >
DATE_SUB(CURDATE(),INTERVAL 4 YEAR) ;
```

nome	data_aquisicao	DATE_SUB(CU...
Casa da Planície	2007-05-11	2005-05-22
Quinta dos Monges	2006-03-14	2005-05-22
Casa dos Pessegueiros	2008-11-19	2005-05-22
São Cristóvão	2009-01-14	2005-05-22

Podemos utilizar a seguinte expressão para determinar há quantos anos foi adquirido cada uma das propriedades:

```
SELECT nome, data_aquisicao,
CURDATE() AS hoje,
DATEDIFF(CURDATE(), data_aquisicao)/365
AS idade
FROM t_propriedade;
```

nome	data_aquisicao	hoje	idade
Quinta de Santa Teresa	2000-01-12	2009-05-21	9.3616
Quinta da Encosta	2000-08-22	2009-05-21	8.7507
Quinta das Esporas	2002-03-13	2009-05-21	7.1945
Quinta dos Freires	2004-06-17	2009-05-21	4.9288
Casa da Planície	2007-05-11	2009-05-21	2.0301
Quinta dos Monges	2006-03-14	2009-05-21	3.1890
Casa dos Pessegueiros	2008-11-19	2009-05-21	0.5014
São Cristóvão	2009-01-14	2009-05-21	0.3479

A função DATADIFF() permite calcular o número de dias entre duas datas. De referir que os sistemas de gestão de base de dados disponibilizam funções semelhantes para a manipulação de atributos do tipo DATE que incluem informação sobre o tempo, isto é, tem informação do tipo hh:mm:ss.

## Exercícios

Escreva uma instrução que permita:

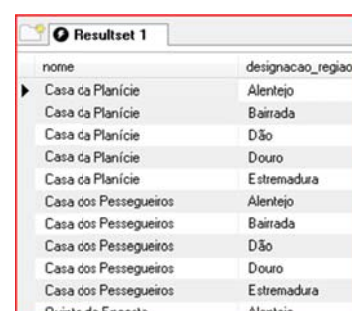
1. seleccionar as propriedades adquiridas no ano 2000.
2. seleccionar as propriedades adquiridas nos últimos dois anos na região do Douro.

### 6.3. Junção de tabelas

O comando SELECT permite juntar a informação de mais de uma tabela. Para tal, é necessário colocar na cláusula FROM a lista das tabelas que se pretende juntar. De acordo com os princípios do modelo relacional a ligação da informação entre tabelas relacionadas é realizada através das chaves primárias e estrangeiras. No entanto, a linguagem SQL por defeito não utiliza a informação associada às chaves estrangeiras definidas entre as tabelas seleccionadas na cláusula FROM sendo necessário indicar de forma explícita quais as chaves estrangeiras que se pretende utilizar.

Por exemplo, se pretendermos seleccionar o nome das propriedades e a designação da região correspondente podemos tentar a seguinte expressão:

```
SELECT nome, designacao_regiao
FROM t_propriedade, t_regiao
ORDER BY nome, designacao_regiao;
```

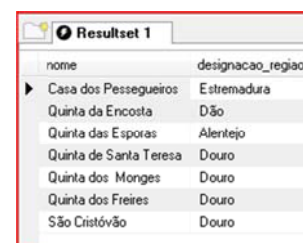


nome	designacao_regiao
Casa da Planície	Alentejo
Casa da Planície	Bairrada
Casa da Planície	Dão
Casa da Planície	Douro
Casa da Planície	Estremadura
Casa dos Pessegueiros	Alentejo
Casa dos Pessegueiros	Bairrada
Casa dos Pessegueiros	Dão
Casa dos Pessegueiros	Douro
Casa dos Pessegueiros	Estremadura
Quinta da Encosta	Alentejo

A análise do resultado obtido revela que a cada propriedade é atribuída mais de uma região. De facto, a expressão associa a cada propriedade existente na tabela 't\_propriedade' cada uma das regiões definidas na tabela 't\_regiao'. Embora o resultado obtido possa parecer desadequado é importante referir que se trata do produto cartesiano entre os dois conjuntos, sendo cada conjunto definido por o conteúdo de cada uma das tabelas. Embora esta operação não seja frequentemente pretendida, trata-se de uma operação importante da teoria dos conjuntos que permite a resolução de interrogações complexas à base de dados.

Para obter a indicação de qual a designação da região a que pertence cada uma das propriedades temos que indicar quais as chaves primária e estrangeira que ligam as duas tabelas, tal como exemplificado na seguinte expressão:

```
SELECT nome, designacao_regiao
FROM t_propriedade, t_regiao
WHERE t_propriedade.regiao_id = t_regiao.regiao_id
ORDER BY nome;
```



nome	designacao_regiao
Casa dos Pessegueiros	Estremadura
Quinta da Encosta	Dão
Quinta das Esporas	Alentejo
Quinta de Santa Teresa	Douro
Quinta dos Monges	Douro
Quinta dos Freires	Douro
São Cristóvão	Douro

A cláusula WHERE indica qual a coluna da tabela 't\_propriedade' que tem informação correspondente à chave primária da tabela 't\_regiao'. Quando se faz referência a uma coluna cujo nome consta em mais de uma das tabelas seleccionadas, é necessário indicar o nome tabela a que pertence o atributo pretendido.

A expressão 't\_propriedade.regiao\_id' indica que estamos a fazer referência à coluna 'regiao\_id' da tabela 't\_propriedade'.

Se na expressão anterior tivéssemos colocado o wildcard '\*' em vez do nome das duas colunas pretendidas, o resultado apresentaria todas as colunas das duas tabelas, dando origem à ocorrência de duas colunas de conteúdo exactamente iguais (as colunas correspondentes à ligação entre as duas tabelas). Nesse caso estaríamos perante a denominada *Equi-Join*. Quando não existe repetição das colunas envolvidas na ligação entre as tabelas diz-se que estamos perante uma *Natural Join*. Uma vez que a ligação entre as duas tabelas foi efectuada através de uma igualdade entre as respectivas colunas, os dois tipos de ligação referidos fazem parte de um tipo de ligação denominado *Inner Join*. Nas secções seguintes vamos ilustrar os tipos de ligação entre tabelas mais importantes.

### Inner join

Recapitulando, a junção de duas tabelas faz-se ligando a chave primária de uma tabela à chave estrangeira da outra tabela. Diz-se que temos um *Inner Join* entre as duas tabelas quando são retornados apenas aqueles registos em que para um dado valor do atributo de ligação há um registo correspondente nas duas tabelas. A expressão do exemplo anterior é do tipo Inner Join porque utiliza a igualdade entre os atributos de ligação.

Uma forma alternativa de especificar uma Inner Join é a seguinte:

```
SELECT propriedade_id, t_regiao.regiao_id,
nome, designacao_regiao
FROM t_regiao INNER JOIN t_propriedade
ON t_propriedade.regiao_id =
t_regiao.regiao_id;
```



propriedade_id	regiao_id	nome	designacao_regiao
3	2	Quinta das Esporas	Alentejo
2	3	Quinta da Encosta	Dão
1	1	Quinta de Santa Teresa	Douro
4	1	Quinta dos Freires	Douro
6	1	Quinta dos Monges	Douro
8	1	São Cristóvão	Douro
7	5	Casa dos Pessegueiros	Estremadura

A análise dos resultados revela que são retornados todas as propriedades para as quais foi atribuída uma região vinícola. Assim, nos resultados retornado são omitidas as propriedades sem região atribuída e as regiões sem propriedades atribuídas.

### Left join e Right Join

Diz-se que temos uma *Left Join* entre duas tabelas quando são retornados todos os registos da tabela da esquerda e apenas os registos correspondentes da tabela da direita, sendo omitidos os registos da tabela da direita que não tem um registo correspondente na tabela da esquerda. Um exemplo de uma Left Join é o da expressão abaixo:

```

SELECT propriedade_id, t_regiao.regiao_id,
nome, designacao_regiao
FROM t_regiao LEFT JOIN t_propriedade
ON t_propriedade.regiao_id =
t_regiao.regiao_id;

```

propriedade_id	regiao_id	nome	designacao_regiao
3	2	Quinta das Esporas	Alentejo
2	3	Quinta da Encosta	Dão
1	1	Quinta de Santa Teresa	Douro
4	1	Quinta dos Freires	Douro
6	1	Quinta dos Monges	Douro
8	1	São Cristóvão	Douro
7	5	Casa dos Pessegueiros	Estremadura

Como resultado desta expressão obtemos uma listagem de todas as regiões, e caso existam, as atributos relativos a cada uma das propriedade da região. Para uma região que tem associada mais do que uma propriedade a informação da tabela de regiões é repetida para cada uma das propriedades. Para uma região que não tem qualquer propriedade atribuída, é retornada uma linha com a informação da região e em que as colunas relativas às propriedades apresentam o valor NULL.

A expressão seguinte permite obter uma listagem de todas as propriedades, mesmo as que não tem uma região atribuída. Para as propriedades com região atribuída é apresentada a informação relativa a essa mesma região.

```

SELECT propriedade_id, nome,
designacao_regiao
FROM t_propriedade LEFT JOIN t_regiao
ON t_propriedade.regiao_id =
t_regiao.regiao_id;

```

propriedade_id	nome	designacao_regiao
1	Quinta de Santa Teresa	Douro
2	Quinta da Encosta	Dão
3	Quinta das Esporas	Alentejo
4	Quinta dos Freires	Douro
5	Casa da Planície	NULL
6	Quinta dos Monges	Douro
7	Casa dos Pessegueiros	Estremadura
8	São Cristóvão	Douro

Esta última expressão é equivalente à seguinte *Right Join*. Diz-se que temos uma *Right Join* quando são retornados todos os registos da tabela de direita e apenas os registos correspondentes da tabela da esquerda.

```
SELECT propriedade_id, nome,
designacao_regiao
FROM t_regiao RIGHT JOIN t_propriedade
ON t_propriedade.regiao_id =
t_regiao.regiao_id;
```

propriedade_id	nome	designacao_regiao
1	Quinta de Santa Teresa	Douro
2	Quinta da Encosta	Dão
3	Quinta das Esporas	Alentejo
4	Quinta dos Freires	Douro
5	Casa da Planície	NULL
6	Quinta dos Monges	Douro
7	Casa dos Pessegueiros	Estremadura
8	São Cristóvão	Douro

## Full join

Um *Full Join* permite obter uma listagem de todos os registos da tabela da direita e todos os registos da tabela da esquerda, combinando os registos para os casos em que os atributos de ligação satisfazem a igualdade.

Assim, como resultado da expressão seguinte obtemos uma listagem de todas as regiões e de todas as propriedades sendo apresentado com o valor NULL os atributos das regiões para as propriedades sem região atribuída e os atributos das propriedades para as regiões que não tem qualquer propriedade definida. De referir que, embora faça parte da definição da linguagem SQL, o Full Join não está disponível nas versões actuais do MySQL e do MS Access.

```
SELECT *
FROM t_regiao FULL JOIN t_propriedade
ON t_propriedade.regiao_id = t_regiao.regiao_id;
```

## SELF JOIN

No que diz respeito à junção de tabelas importa referir a situação em que se pretende juntar uma tabela com ela própria. No nosso exemplo, isto será útil para determinar quais os pares de propriedades situadas na mesma região, tal como é ilustrado na instrução seguinte.

```
SELECT p1.nome, p2.nome, p1.regiao_id
FROM t_propriedade AS p1, t_propriedade AS p2
WHERE p1.regiao_id = p2.regiao_id
AND p1.propriedade_id <> p2.propriedade_id;
```

nome	nome	regiao_id
Quinta de Santa Teresa	Quinta dos Freires	1
Quinta de Santa Teresa	Quinta dos Monges	1
Quinta de Santa Teresa	São Cristóvão	1
Quinta dos Freires	Quinta de Santa Teresa	1
Quinta dos Freires	Quinta dos Monges	1
Quinta dos Freires	São Cristóvão	1
Quinta dos Monges	Quinta de Santa Teresa	1
Quinta dos Monges	Quinta dos Freires	1
Quinta dos Monges	São Cristóvão	1

Quando na cláusula FROM se seleccionada a mesma tabela mais do que uma vez torna-se obrigatório utilizar a cláusula AS para renomear as tabelas e evitar a ambiguidade que decorre das referências aos respectivos atributos. De referir ainda a inclusão da expressão 'p1.propriedade\_id <> p2.propriedade\_id'

que permite eliminar do resultado os registos que fazem referência à mesma propriedade nas duas tabelas.

Este exemplo ilustra ainda uma limitação do SQL, uma vez que não é possível filtrar a ocorrência de um par (A,B) quando o par (B,A) já tenha ocorrido. De facto, a análise dos resultados da query revela que, por exemplo, o par (Quinta dos Freires, Quinta dos Monges) também ocorre na forma (Quinta dos Monges, Quinta dos Freires). Através da sintaxe do SQL não é possível identificar este tipo de repetições.

## UNION e UNION ALL

O operado UNION permite combinar o resultado de duas instruções de SELECT. Por exemplo, a seguinte instrução permite obter a união do conjunto das propriedades da região 1 com o conjunto das propriedades da região 2.

```
SELECT * FROM t_propriedade
WHERE regioao_id = 2
UNION
SELECT * FROM t_propriedade
WHERE regioao_id = 1;
```



propri...	nome	endereco	telefone	email
3	Quinta das Esporas	Reguengos de M...	266 587...	esporas@mail.pt
1	Quinta de Santa Teresa	Peso da Régua	254 223...	santaleresa@vir
4	Quinta dos Freires	Pinhão	254 654...	q_freires@mail.pt
6	Quinta dos Monges	Pinhão	254 777...	monjes@mail.pt
8	São Cristóvão	Pinhão		

De referir que o mesmo resultado poderia ser obtido com a expressão WHERE e o operador lógico OR. O operador UNION requer que o número de colunas seleccionado seja o mesmo nos dois selects, sendo os nomes indicados no resultado os correspondentes ao primeiro select. Além disso, o operado elimina duplicados por defeito sendo necessário utilizar o operador UNION ALL se se pretender que o resultado apresente duplicados.

## Exercícios

Escreva uma instrução que permita:

1. obter uma listagem das marcas de vinhos tintos produzidos na região do Douro. Assegure que a mesma marca não surge nos resultados mais de uma vez.
2. seleccionar as marcas juntamente com a designação das castas utilizadas e a respectiva percentagem (omita dos resultados as marcas que não tem as castas discriminadas). Apresente o resultado ordenado pelo nome da marca (de forma ascendente) e com as castas ordenadas de acordo com a sua importância.
3. seleccionar as marcas juntamente com a designação das castas utilizadas e a respectiva percentagem (inclua nos resultados as marcas que não tem as castas discriminadas). Apresente o resultado ordenado pelo nome da marca (de forma ascendente) e com as castas ordenadas de acordo com a sua importância.

4. seleccionar as castas de alta produtividade (ordenadas por cor e, para cada cor, ordenadas pela sua designação) que são utilizadas na produção de vinhos tintos da região do Douro.
5. seleccionar as marcas de vinho tinto com mais de 10 meses de estágio em madeira e cujo preço se situa entre os 10 e os 20€ e que são produzidas em propriedades situados na região do Alentejo.

## 6.4. Funções de agregação

As funções de agregação disponibilizadas na linguagem SQL permitem obter um valor calculado para um conjunto de linhas de uma tabela. Vamos ilustrar a sua utilidade através de alguns exemplos. Começemos por analisar o conteúdo da tabela de marcas através de uma instrução que permite obter uma listagem de todos os preços das marcas produzidas:

```
SELECT marca, preco FROM t_marca
ORDER BY marca;
```



marca	preco
Branco dos Freires	7
Duas Casas	20
Duas Casas - Branco	6
Duas Casas - Reserva	40
Esporas - Garrafeira	30
Esporas - Reserva	12
Esporas - Trincadeira	10
Quinta da Encosta - Espumante	7
Quinta da Encosta - Vinhas Velhas	30
Quinta das Esporas	8
Quinta das Esporas Reserva	15
Quinta das Esporas Touriga Nacional	14
Tinto dos Freires	9

Utilizando as funções de agregação, podemos obter o preço máximo de um vinho da seguinte forma:

```
SELECT MAX(preco) FROM t_marca;
```



MAX(preco)
40

De forma semelhante podemos calcular o preço médio das marcas produzidas através da instrução:

```
SELECT AVG(preco) FROM t_marca;
```



AVG(preco)
16

Podemos também consultar a variabilidade do preço das marcas produzidas na região do Douro com a seguinte instrução:

```

SELECT MAX(preco), MIN(preco),
AVG(preco), STDDEV(preco)
FROM t_marca, t_marca_propriedade,
t_propriedade, t_regiao
WHERE t_marca.marca_id =
t_marca_propriedade.marca_id
AND t_marca_propriedade.propriedade_id =
t_propriedade.propriedade_id
AND t_propriedade.regiao_id = t_regiao.regiao_id
AND designacao_regiao = 'Douro';

```

MAX[preco]	MIN[preco]	AVG[preco]	STDDEV[preco]
40	6	18.5	13.527749...

A expressão retorna o preço máximo, mínimo, médio e o desvio padrão dos preços das marcas produzidas na região do Douro. De referir que é necessário consultar quatro tabelas para poder efectuar a restrição pela designação da região. Outras funções úteis são a função COUNT, que permite contar o número de registos seleccionados, a função SUM, que permite somar os valores de um atributo numérico, e a função VARIANCE, que permite calcular a variância de um atributo numérico.

A função COUNT, pela sua importância, merece uma atenção especial. Vamos começar por analisar o conteúdo da tabela de propriedades no que diz respeito aos atributos 'propriedade\_id' e 'regiao\_id'.

```

SELECT propriedade_id, regiao_id
FROM t_propriedade;

```

propriedade_id	regiao_id
5	1
1	1
4	1
6	1
8	1
3	2
2	3
7	5

Se quisermos contar o número de propriedades existentes podemos utilizar a seguinte instrução:

```

SELECT COUNT(*) FROM t_propriedade;

```

COUNT(*)
8

A cláusula COUNT(\*) conta o número de linhas que resultam da selecção especificada. Por outro lado, a cláusula COUNT(nome\_atributo) permite contar o número de linhas que resultam da selecção mas que não tem um valor nulo no atributo indicado. Por exemplo, para contar o número de propriedades que tem uma região atribuída podemos utilizar a seguinte instrução:

```
SELECT COUNT(regiao_id) FROM t_propriedade;
```

Resultset 1
COUNT(regiao...)
7

Finalmente, a função COUNT admite ainda a utilização da cláusula DISTINCT. Por exemplo, para determinar em quantas regiões são possuídas propriedades podemos utilizar a seguinte instrução:

```
SELECT COUNT(DISTINCT regiao_id)
FROM t_propriedade;
```

Resultset 1
COUNT(DISTI...)
4

## Exercícios

Escreva uma instrução que permita:

1. seleccionar a data de aquisição da propriedade mais antiga.
2. calcular o número de marcas produzidas em propriedades situadas na região do Alentejo.
3. calcular o número de castas utilizadas na produção de espumantes.

## 6.5. Agrupamento de registos

Na instrução de SELECT, a cláusula GROUP BY permite agrupar os registos de acordo com o valor de uma, ou mais de uma, coluna. Esta cláusula é, em geral, utilizada conjuntamente com as funções de agregação.

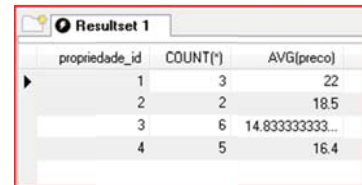
Anteriormente vimos como determinar o preço máximo dos vinhos produzidos. Vamos agora supor que pretendemos analisar os preços por propriedade. Se pretendemos saber os preços dos vinhos produzidos em cada uma das propriedades podemos utilizar a seguinte instrução:

```
SELECT propriedade_id, marca, preco
FROM t_marca, t_marca_propriedade
WHERE t_marca.marca_id =
t_marca_propriedade.marca_id
ORDER BY propriedade_id;
```

Resultset 1		
propriedade_id	marca	preco
1	Duas Casas - Reserva	40
1	Duas Casas	20
1	Duas Casas - Branco	6
2	Quinta da Encosta - Espumante	7
2	Quinta da Encosta - Vinhas Velhas	30
3	Quinta das Esporas Touiga Nacional	14
3	Esporas - Trincadeira	10
3	Esporas - Garrafeira	30
3	Quinta das Esporas	8
3	Quinta das Esporas Reserva	15
3	Esporas - Reserva	12
4	Branco dos Freires	7
4	Duas Casas - Reserva	40
4	Tinto dos Freires	9

Como resultado obtemos, para cada propriedade, uma listagem das marcas produzidas e do preço correspondente. Agora, para calcular o número de marcas produzidas em cada propriedade e a média dos seus preços podemos utilizar a seguinte instrução:

```
SELECT propriedade_id, COUNT(*), AVG(preco)
FROM t_marca, t_marca_propriedade
WHERE t_marca.marca_id =
t_marca_propriedade.marca_id
GROUP BY propriedade_id;
```



propriedade_id	COUNT(*)	AVG(preco)
1	3	22
2	2	18.5
3	6	14.8333333333...
4	5	16.4

A análise dos resultados obtidos com a última expressão permite verificar que o identificador de cada propriedade ocorre apenas uma vez, juntamente com o número de vinhos nela produzidos e a média dos preços desses mesmos vinhos.

A cláusula GROUP BY actua no resultado de um SELECT agrupando as linhas do seu resultado pelos atributos indicados na cláusula de agrupamento. Actuando em conjunto com a clausula CROUP BY as funções de agregação permitem calcular o resultado de operações efectuadas para cada um dos grupos formados. Vejamos um outro exemplo que permite calcular o número de casta de cada:

```
SELECT cor, COUNT(casta_id)
FROM t_casta
GROUP BY cor;
```



cor	COUNT(casta_id)
Branca	5
Tinta	9

Por outro lado, para calcular o número total de castas definidas da tabela podemos utilizar a seguinte instrução:

```
SELECT COUNT(casta_id) FROM t_casta;
```

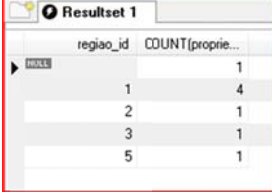


COUNT(casta_id)
14

Importante referir que nesta última instrução não podemos seleccionar nenhum atributo das castas para o resultado uma vez que, sendo retornado uma única linha de resultados, seria impossível definir qual o valor pretendido para, por exemplo, a cor. Da mesma forma, na penúltima instrução só é possível seleccionar para o resultado o valor de atributos que tenham um único valor para cada um dos grupos retornados. A regra definida é, apenas podemos seleccionar para o resultado de uma instrução que utilize a cláusula GROUP BY atributos que constem nessa mesma cláusula ou o resultado de funções de agregação.

Por exemplo, podemos calcular o número de propriedades por região através da seguinte instrução:

```
SELECT regioao_id, COUNT(propriedade_id)
FROM t_propriedade
GROUP BY regioao_id;
```



regiao_id	COUNT(propriedade_id)
1	4
2	1
3	1
5	1

Se pretendermos incluir no resultado a designação de cada uma das regiões temos que seleccionar também a tabela de regiões. No entanto, apenas podemos seleccionar o atributo 'designacao\_regiao' para o resultado se o mesmo for colocado na cláusula de GROUP BY.

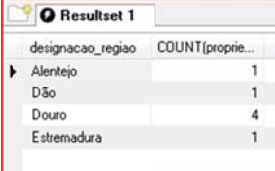
```
SELECT t_propriedade.regiao_id, designacao_regiao,
COUNT(propriedade_id)
FROM t_propriedade, t_regiao
WHERE t_propriedade.regiao_id = t_regiao.regiao_id
GROUP BY t_propriedade.regiao_id, designacao_regiao;
```



regiao_id	designacao_regiao	COUNT(propriedade_id)
1	Douro	4
2	Alentejo	1
3	Dão	1
5	Estremadura	1

De referir que na definição da tabela 't\_regiao' foi utilizada a cláusula UNIQUE para o atributo 'designacao\_regiao' que obriga a que não possam ser inseridas na tabela duas regiões com a mesma designação. Como tal, se não se pretendesse incluir o identificador das regiões no resultado poderíamos definir a expressão da seguinte forma:

```
SELECT designacao_regiao, COUNT(propriedade_id)
FROM t_propriedade, t_regiao
WHERE t_propriedade.regiao_id = t_regiao.regiao_id
GROUP BY designacao_regiao;
```



designacao_regiao	COUNT(propriedade_id)
Alentejo	1
Dão	1
Douro	4
Estremadura	1

Em geral, é recomendável agrupar os resultados em primeiro lugar pelo identificador e, caso se pretenda, incluir como segundo critério de agrupamento a designação. No caso de não serem admitidas repetições na designação o segundo critério não irá alterar os grupos formados mas permite a inclusão desse atributo nos resultados da expressão.

Quando se utiliza a cláusula GROUP BY podemos utilizar a cláusula HAVING para definir restrições relativas aos grupos formados. Enquanto a cláusula WHERE é utilizada para restringir as linhas seleccionadas antes de serem processadas pela cláusula de agrupamento, a cláusula HAVING é utilizada para definir restrições para os grupos formados.

Vamos ilustrar a diferença através de um exemplo. A instrução anterior permitiu ver quantas propriedades temos por região, enquanto a instrução seguinte permite ver o nome, a região e a dimensão de cada propriedade.

```
SELECT nome, designacao_regiao,
t_propriedade.dimensao
FROM t_propriedade, t_regiao
WHERE t_propriedade.regiao_id =
t_regiao.regiao_id;
```



nome	designacao_regiao	dimensao
Quinta das Esporas	Alentejo	200
Quinta da Encosta	Dão	120
Quinta de Santa Teresa	Douro	60
Quinta dos Freires	Douro	100
Quinta dos Monges	Douro	133
São Cristóvão	Douro	38
Casa dos Pessegueiros	Estremadura	75

Se quisermos contar o número de propriedades que tem mais de 70 ha em cada região, temos que antes de formar os grupos indicar quais as propriedades que devem ser seleccionadas. Para tal, temos que definir o critério relativo à selecção das propriedades com o tamanho pretendido na cláusula WHERE, tal como indicado na seguinte instrução:

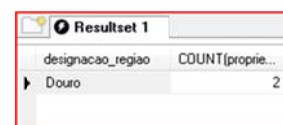
```
SELECT designacao_regiao, COUNT(propriedade_id)
FROM t_propriedade, t_regiao
WHERE t_propriedade.regiao_id = t_regiao.regiao_id
AND t_propriedade.dimensao > 70
GROUP BY designacao_regiao;
```



designacao_regiao	COUNT(proprie...
Alentejo	1
Dão	1
Douro	2
Estremadura	1

Se quisermos agora apenas seleccionar as regiões que tem duas, ou mais, propriedades com mais de 70 ha temos que utilizar a seguinte instrução:

```
SELECT designacao_regiao, COUNT(propriedade_id)
FROM t_propriedade, t_regiao
WHERE t_propriedade.regiao_id = t_regiao.regiao_id
AND t_propriedade.dimensao > 70
GROUP BY designacao_regiao
HAVING COUNT(propriedade_id) >= 2;
```



designacao_regiao	COUNT(proprie...
Douro	2

Nesta instrução, a cláusula 't\_propriedade.dimensao > 70' faz com que apenas as propriedades com mais de 70 ha sejam agrupadas e a cláusula 'HAVING COUNT(propriedade\_id) >= 2' faz com que apenas as regiões com duas, ou mais, propriedades com o tamanho pretendido sejam retornadas nos resultados.

## Exercícios

Escreva uma instrução em SQL que permita obter:

1. uma listagem das marcas juntamente com o número de castas utilizadas em cada uma das marcas.
2. uma listagem das marcas produzidas em propriedades situadas na região do Douro juntamente com o número de castas utilizadas em cada uma das marcas.
3. uma listagem das marcas produzidas em propriedades situadas na região do Douro juntamente com o número de castas utilizadas em cada uma das marcas, limitando os resultados às marcas que utilizam duas ou mais castas.
4. uma listagem das regiões em há pelo menos uma marca associada a mais de uma propriedade.
5. uma listagem do número total de garrafas produzidas para cada tipo de vinho (identificado pela sua designação).
6. uma listagem do número total de garrafas produzidas para cada tipo de vinho (identificado pela sua designação) limitando os resultados aos tipos de vinho dos quais são produzidas mais de 150 mil garrafas.

## 6.6. Subqueries

Nesta secção vamos ilustrar um conceito importante no SQL, as subqueries. Diz-se que temos uma subquery quando temos uma instrução SELECT dentro de outra instrução SELECT. De facto, uma vez que um SELECT retorna um conjunto de linhas e colunas em formato de tabela, podemos substituir qualquer expressão que faça referência a uma tabela ou a um valor por uma instrução de SELECT. Importante ter em atenção que numa subquery a instrução interior tem obrigatoriamente de ser colocada entre parêntesis.

Devemos utilizar uma subquery quando a query a escrever está dependente de um valor desconhecido à partida. A forma de utilização mais frequente de subqueries é na cláusula WHERE. Por exemplo, para determinar qual a área de vinha da região vinícola com maior área de plantação de vinha podemos usar a seguinte instrução:

```
SELECT MAX(dimensao) FROM t_regiao;
```

The screenshot shows a window titled 'Resultset 1' with a single column header 'MAX(dimensao)' and one row containing the value '40000'.

MAX(dimensao)
40000

Por outro lado, podemos determinar qual a dimensão das diferentes regiões vinícolas com a seguinte instrução:

```
SELECT designacao_regiao, dimensao FROM t_regiao;
```

The screenshot shows a window titled 'Resultset: 1' with two columns: 'designacao\_regiao' and 'dimensao'. The data rows are: Douro (40000), Alentejo (22000), Dão (20000), Bairrada (10000), and Estremadura (10000).

designacao_regiao	dimensao
Douro	40000
Alentejo	22000
Dão	20000
Bairrada	10000
Estremadura	10000

Agora, para determinar a designação da maior região vinícola podemos encadear as duas queries anteriores da seguinte forma:

```
SELECT designacao_regiao, dimensao
FROM t_regiao WHERE dimensao =
(SELECT MAX(dimensao) FROM t_regiao);
```



designacao_regiao	dimensao
Douro	40000

Note que, o atributo 'dimensao' é do tipo numérico e a query interior, além de retornar uma única coluna, retorna um valor também numérico, através da função MAX. Como estamos a utilizar o operador '=' a query interior tem obrigatoriamente de retornar um valor unico para que a comparação possa ser efectuada.

Neste tipo de queries há um grupo de operadores que obrigam a que a query interior retorne um valor único (=, >, <, <=, >=, <>) e um grupo de operadores que permite a comparação com um conjunto de valores (IN, ANY, ALL, EXISTS). Qualquer dos operadores referidos pode ser precedido do operador NOT.

Vamos ilustrar a utilização de alguns destes operadores.

Podemos seleccionar as marcas cujo preço é superior ao preço médio de todas as marcas através da seguinte instrução:

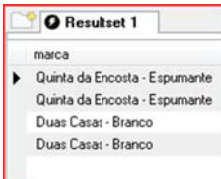
```
SELECT marca FROM t_marca WHERE preco >
(SELECT AVG(preco) FROM t_marca);
```



marca
Quinta da Encosta - Vinhas Velhas
Esporas - Garrafeira
Duas Casas
Duas Casas - Reserva

Se quisermos seleccionar as marcas que usam pelo menos uma casta branca podemos utilizar a instrução seguinte:

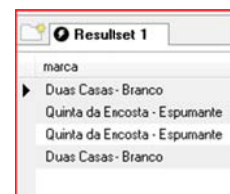
```
SELECT marca FROM t_marca, t_marca_casta
WHERE t_marca.marca_id = t_marca_casta.marca_id
AND casta_id IN
(SELECT casta_id FROM t_casta WHERE cor = 'branca');
```



marca
Quinta da Encosta - Espumante
Quinta da Encosta - Espumante
Duas Casas - Branco
Duas Casas - Branco

Esta última questão podia ter sido resolvida sem recorrer às sub-queries através da seguinte instrução:

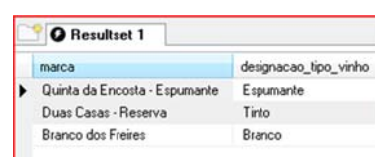
```
SELECT marca FROM t_marca, t_marca_casta, t_casta
WHERE t_marca.marca_id = t_marca_casta.marca_id
AND t_marca_casta.casta_id = t_casta.casta_id
AND cor = 'branca';
```



marca
Duas Casas - Branco
Quinta da Encosta - Espumante
Quinta da Encosta - Espumante
Duas Casas - Branco

Para determinar a marca de vinho mais cara de cada tipo podemos utilizar a seguinte instrução, que corresponde a uma subquery do tipo multi-coluna, uma vez que temos na query exterior um par de valores que é comparado com o par de valores correspondente retornado pela query interior (este tipo de subqueries não estão disponíveis no MS ACCESS).

```
SELECT marca, designacao_tipo_vinho
FROM t_marca, t_tipo_vinho
WHERE t_marca.tipo_vinho_id =
t_tipo_vinho.tipo_vinho_id
AND (t_marca.tipo_vinho_id, preco) IN
(SELECT tipo_vinho_id, MAX(preco)
FROM t_marca GROUP BY tipo_vinho_id);
```



marca	designacao_tipo_vinho
Quinta da Encosta - Espumante	Espumante
Duas Casas - Reserva	Tinto
Branco dos Freires	Branco

A seguinte instrução selecciona as marcas para as quais não estão identificadas as castas. A cláusula EXISTS verifica se a query interior retorna pelo menos um registo, sendo que neste caso a utilização do NOT implica que a expressão seja válida quando a subquery não retorna qualquer registo.

```
SELECT marca FROM t_marca
WHERE NOT EXISTS
(SELECT marca_id FROM t_marca_casta
WHERE t_marca_casta.marca_id = t_marca.marca_id);
```



marca
Quinta das Esporas: Touiga Nacional
Branco dos Freires
Tinto dos Freires

De referir que a instrução anterior é do tipo query correlacionada, dado que a query interior depende da query exterior. Note que na expressão WHERE da query interior se faz referência à marca da query exterior. Neste tipo de queries a query interior é executada para cada linha da query exterior, sendo nesta caso executada tantas vezes quantas as marcas definidas na tabela 't\_marca'.

Quando a query interior não depende da query exterior diz-se que temos uma query não-correlacionada, sendo nestes casos a query interior executada uma única vez. Um exemplo de uma query não correlacionada é a query seguinte, que permite seleccionar a marca mais cara:

```
SELECT marca FROM t_marca
WHERE preco =(SELECT MAX(preco) FROM t_marca);
```

marca
Duas Casas - Reserva

Importante também referir que não existe limite para o número de queries encadeadas. Por exemplo, se pretendermos seleccionar a segunda marca mais cara podemos recorrer à seguinte instrução:

```
SELECT marca FROM t_marca WHERE preco =
(SELECT MAX(preco) FROM t_marca WHERE preco <
(SELECT MAX(preco) FROM t_marca));
```

marca
Quinta da Encosta - Vinhas Velhas
Esporas - Garrafeira

Vamos agora ilustrar o operado ALL com uma instrução que permite seleccionar as marcas cujo preço é inferior ao de todas as marcas com um teor alcoólico de 14%:

```
SELECT marca, preco FROM t_marca
WHERE preco < ALL
(SELECT preco FROM t_marca WHERE teor = 14);
```

marca	preco
Quinta das Esporas	8
Quinta da Encosta - Espunante	7
Esporas - Reserva	12
Esporas - Trincadeira	10
Duas Casas - Branco	6
Branco dos Freires	7
Tinto dos Freires	9

O operador '< ALL' é equivalente a definir '< (SELECT MIN(x)...)'

É ainda possível colocar uma subquery na cláusula HAVING. Por exemplo, para seleccionar a propriedade cujos vinhos tem a mais elevada graduação média podemos utilizar a seguinte expressão:

```
SELECT propriedade_id FROM t_marca, t_marca_propriedade
WHERE t_marca.marca_id = t_marca_propriedade.marca_id
GROUP BY t_marca_propriedade.propriedade_id
HAVING AVG(teor) =
(SELECT MAX(media) FROM
(SELECT AVG(teor) AS media
FROM t_marca, t_marca_propriedade
WHERE t_marca.marca_id = t_marca_propriedade.marca_id
GROUP BY t_marca_propriedade.propriedade_id) AS marcas);
```

propriedade_id
3

Na instrução anterior é necessário utilizar o alias 'AS marcas' dado que sempre que se utiliza uma instrução de SELECT a substituir uma referência a uma tabela é necessário atribuir um nome á tabela resultante desse SELECT.

## Exercícios

Escreva uma instrução que permita:

1. seleccionar o nome da propriedade mais antiga.
2. obter a marca com teor alcoólico superior a 13.5% que tem maior tempo de estágio em barricas de madeira.
3. seleccionar a casta utilizada no maior número de vinhos.
4. seleccionar o preço médio das marcas que não tem as castas discriminadas.
5. seleccionar o preço médio das marcas que são produzidas com uvas de apenas uma casta.

## 7. Criação de vista (views)

No SQL as vistas (views) permitem a criação de uma janela sobre uma tabela ou um conjunto de tabelas. Esta funcionalidade é importante por permitir filtrar a informação a que cada utilizador pode ter acesso. Um exemplo da utilidade das vistas será a criação de uma vista com as marcas de uma determinada propriedade uma vez que possibilita disponibilizar essa informação sem revelar as marcas de outras propriedades. A instrução seguinte permite criar uma vista com as marcas da Quinta de Santa Teresa:

```
CREATE VIEW marcas_santa_teresa AS
SELECT t_marca.* FROM t_marca, t_marca_propriedade
WHERE t_marca.marca_id = t_marca_propriedade.marca_id
AND propriedade_id = 1;
```

É assim possível utilizar a vista criada como uma tabela normal, mas ao mesmo tempo reservando registos que se consideram confidenciais. As vistas poderão também ser úteis para simplificar o acesso às tabelas através da omissão de atributos menos importantes ou da criação de uma vista para mascarar o acesso a várias tabelas através de uma query complexa.

Uma vista pode ser eliminada através de uma instrução como a seguinte:

```
DROP VIEW marcas_santa_teresa;
```

## 8. ANEXO – script para criação da base de dados

### tabela t\_regiao

```
CREATE TABLE t_regiao (
regiao_id INT PRIMARY KEY,
designacao_regiao VARCHAR(15) NOT NULL UNIQUE,
descricao TEXT);
```

```
ALTER TABLE t_regiao ADD dimensao INT;
```

```
INSERT INTO t_regiao (regiao_id,designacao_regiao,descricao) VALUES (1,'Douro', 'A região do Douro localiza-se no Nordeste de Portugal, rodeada pelas serras do Marão e Montemuro. A área vitícola ocupa cerca de 40000 hectares, apesar da região se prolongar por cerca de 250000 hectares. O rio Douro e os seus afluentes, como por exemplo o Tua e o Corgo, correm em vales profundos e a maior parte das plantações são encaixadas nas bacias hidrográficas dos rios.');
```

```
UPDATE t_regiao SET dimensao= 40000 WHERE regiao_id = 1;
```

```
INSERT INTO t_regiao (regiao_id,designacao_regiao,descricao,dimensao) VALUES (2, 'Alentejo','O Alentejo é uma das maiores regiões vitivinícolas de Portugal, onde a vista se perde em extensas planícies que apenas são interrompidas por pequenos montes. Esta região quente e seca beneficiou de inúmeros investimentos no sector vitivinícola que se traduziu na produção de alguns dos melhores vinhos portugueses e consequentemente, no reconhecimento internacional dos vinhos alentejanos.', 22000);
```

```
INSERT INTO t_regiao (regiao_id,designacao_regiao,dimensao) VALUES (3,'Dão',20000);
```

```
INSERT INTO t_regiao (regiao_id,designacao_regiao,dimensao) VALUES (4,'Bairrada',10000);
```

```
INSERT INTO t_regiao (regiao_id,designacao_regiao, dimensao) VALUES (5,'Estremadura',null);
(5,'Estremadura',null);
```

### tabela t\_propriedade

```
CREATE TABLE t_propriedade(
propriedade_id INT PRIMARY KEY,
nome VARCHAR(50) NOT NULL,
endereço VARCHAR(50),
telefone VARCHAR(20),
email VARCHAR(30),
dimensao INT,
data_aquisicao DATE,
regiao_id INT,
FOREIGN KEY (regiao_id) REFERENCES t_regiao(regiao_id));
```

```
INSERT INTO t_propriedade VALUES(1,'Quinta de Santa Teresa', 'Peso da Régua', '254 223
```

```

344','santateresa@vinhos.pt', 60,'2000/01/12', 1);
INSERT INTO t_propriedade VALUES(2,'Quinta da Encosta', 'Carregal do Sal','232 423
454','encosta@vinhos.pt',120,'2000/08/22', 3);
INSERT INTO t_propriedade VALUES(3,'Quinta das Esporas', 'Reguengos de Monsaraz', '266 587 876',
'esporas@mail.pt', 200,'2002/03/13', 2);
INSERT INTO t_propriedade VALUES(4,'Quinta dos Freires', 'Pinhão','254 654
777','q_freires@mail.pt',100,'2004/06/17',1);
INSERT INTO t_propriedade VALUES(5,'Casa da Planície', 'Macedo de
Cavaleiros',null,'planicie@mail.pt',40,'2007/05/11',null);
INSERT INTO t_propriedade VALUES(6,'Quinta dos Monges', 'Pinhão','254 777
654','monjes@mail.pt',133,'2006/03/14',1);
INSERT INTO t_propriedade (propriedade_id,nome,endereco,dimensao,data_aquisicao,regiao_id)
VALUES (7,'Casa dos Pessegueiros','Colares',75,'2008/11/19',5);
INSERT INTO t_propriedade (propriedade_id,nome,endereco,dimensao,data_aquisicao,regiao_id)
VALUES (8,'São Cristóvão','Pinhão',38, '2009/01/14',1);

```

### tabela t\_tipo\_vinho

```

CREATE TABLE t_tipo_vinho(
tipo_vinho_id INT PRIMARY KEY AUTO_INCREMENT,
designacao_tipo_vinho VARCHAR(15) NOT NULL UNIQUE,
descricao TEXT);

INSERT INTO t_tipo_vinho (designacao_tipo_vinho,descricao) VALUES ('Tinto', 'Produzidos a partir da
fermentação de uvas Tintas');
INSERT INTO t_tipo_vinho (designacao_tipo_vinho,descricao) VALUES ('Branco', 'Os vinhos brancos
tranquilos são feitos a partir da fermentação de uvas sem pele. ');
INSERT INTO t_tipo_vinho (designacao_tipo_vinho,descricao) VALUES ( 'Rosé', ' Indicação Os vinhos
rosés são elaborados a partir de castas Tintas e através de um processo especial de fermentação. Após
um curto período de tempo retiram-se as peles das uvas, pois já foi transferida alguma coloração rosada
ao vinho. Depois segue-se um processo de fermentação semelhante ao do vinho branco (fermentação
sem peles). ');
INSERT INTO t_tipo_vinho (designacao_tipo_vinho,descricao) VALUES ( 'Espumante', ' Os vinhos
espumantes distinguem-se pela presença de dióxido de carbono proveniente da fermentação secundária,
que lhes atribui a típica “bolha” e espuma. Normalmente os vinhos espumantes têm a sua fase final de
fermentação em garrafa (método clássico ou champanês). Existe ainda o método contínuo onde a
fermentação se efectua através da passagem do vinho por diferentes tanques (onde o vinho fermenta e
envelhece) e o método charmat onde a fermentação se realiza numa cuba fechada. ');
INSERT INTO t_tipo_vinho (designacao_tipo_vinho,descricao) VALUES ( 'Generoso', ' Os vinhos
generosos ou licorosos resultam da adição de álcool (álcool puro, aguardente ou brandy) durante o
processo de fermentação, de modo a suspender o processo de transformação dos açúcares em álcool.
Deste modo, o vinho fica mais doce e alcoólico do que qualquer vinho de mesa');

```

## tabela t\_marca

```
CREATE TABLE t_marca(  
marca_id INT PRIMARY KEY,  
marca VARCHAR(50) NOT NULL,  
teor FLOAT,  
tara FLOAT,  
preco FLOAT,  
num_garrafas INT,  
meses_estagio FLOAT,  
caracteristicas TEXT,  
tipo_vinho_id INT,  
FOREIGN KEY (tipo_vinho_id) REFERENCES t_tipo_vinho(tipo_vinho_id));
```

```
INSERT INTO t_marca VALUES(1,'Quinta das Esporas',13.0, 0.75,8,60000,10, 'Aroma concentrado,  
sobressaindo os frutos vermelhos maduros. Boa estrutura com taninos redondos e final persistente', 1);
```

```
INSERT INTO t_marca VALUES(2,'Quinta das Esporas Reserva', 13.5,0.75,15,15000,17, 'Aroma  
concentrado, nota balsâmicas de madeira de carvalho, fruto de figo maduro e amora. Muito encorpado,  
carnudo com taninos maduros e redondos.', 1);
```

```
INSERT INTO t_marca VALUES(3,'Quinta das Esporas Touriga Nacinal', 14.0,0.75,14,12500,15, 'Muito  
concentrado, nota balsâmicas de madeira de carvalho, frutos vermelhos bem maduros e violeta.  
Equilibrado com taninos maduros e redondos.', 1);
```

```
INSERT INTO t_marca VALUES(4,'Quinta da Encosta - Espumante',null,0.75,7,5000,0,'Castas  
fermentadas separadamente em cubas de inox com controlo de temperatura. A segunda fermentação foi  
efectuada em garrafa para conservar o aroma floral e frutado.',4);
```

```
INSERT INTO t_marca VALUES(5,'Quinta da Encosta - Vinhas Velhas',13.0,0.75,30,5000,15,'Castas  
fermentadas separadamente em cubas de inox com controlo de temperatura. A segunda fermentação foi  
efectuada em garrafa para conservar o aroma floral e frutado.',1);
```

```
INSERT INTO t_marca VALUES(6,'Esporas - Reserva',14.5,0.75,12,30000,12,'perfumado, sugerindo fruta  
vermelha delicada e ligeiro toque floral, bem integrado com a madeira nova. A boca ostenta textura  
sedosa, com várias camadas de fruta sobrepostas, que lhe conferem grande riqueza e  
complexidade.',1);
```

```
INSERT INTO t_marca VALUES(7,'Esporas - Trincadeira',14.5,0.75,10,10000,6,'vinho macio, com aroma  
complexo da casta que lhe deu origem e um sabor rico a frutos vermelhos. Paladar aveludado e  
equilibrado com uma estrutura de taninos suaves.',1);
```

```
INSERT INTO t_marca VALUES(8,'Esporas - Garrafeira',14.5,0.75,30,7000,12,'Vinho de cor intensa,  
aroma fino com sugestões a fruta madura, e notas de tabaco. Na boca é denso com um toque  
balsâmico.',1);
```

```
INSERT INTO t_marca VALUES(9,'Duas Casas',14.0,0.75,20,17500,18,'Cor intensa e limpa, com reflexos  
vermelhos. Aroma fresco e frutado, com notas de cereja e de ameixa. Macio e volumoso na boca  
apresentando uma estrutura aveludada.',1);
```

```
INSERT INTO t_marca VALUES(10,'Duas Casas - Reserva',14.0,0.75,40,35000,14,'Cor densa e  
brilhante, com reflexos vermelho escuro. Aroma floral e de frutos silvestres. Boa estrutura, macio e com
```

```
final de boca persistente.',1);
INSERT INTO t_marca VALUES(11,'Duas Casas - Branco',12.5,0.75,6,75000,0,' ',2);
INSERT INTO t_marca VALUES(12,'Branco dos Freires',11.5,0.75,7,30000,2,' ',2);
INSERT INTO t_marca VALUES(13,'Tinto dos Freires',12.5,0.75,9,50000,6,' ',1);
```

### tabela t\_casta

```
CREATE TABLE t_casta(
casta_id INT PRIMARY KEY,
designacao_casta VARCHAR(25) NOT NULL UNIQUE,
cor VARCHAR(6),
produtividade VARCHAR(5),
descricao TEXT);
```

```
CREATE INDEX IDXcor ON t_casta (cor);
```

```
INSERT INTO t_casta VALUES(1,'Touriga Nacional', 'Tinta','Baixa',' Os bagos têm uma elevada
concentração de açúcar, cor e aromas. Os vinhos produzidos ou misturados com a casta Touriga
Nacional são bastante equilibrados, alcoólicos e com boa capacidade de envelhecimento.');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(2,'Tinta roriz',
'Tinta','Alta');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(3,'Tinta barroca',
'Tinta','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(4,'Tinto Cão',
'Tinta','Baixa');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(5,'Malvasia Fina',
'Branca','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(6,'Moscatel',
'Branca','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(7,'Verdelho',
'Branca','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(8,'Touriga Franca',
'Tinta','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(9,'Tinta Amarela',
'Tinta','Baixa');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(10,'Sousão',
'Tinta','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(11,'Maria Gomes',
'Branca','Alta');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(12,'Arinto',
'Branca','Média');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(13,'Baga', 'Tinta','Alta');
```

```
INSERT INTO t_casta (casta_id,designacao_casta,cor,produtividade) VALUES(14,'Alicante Bouschet',  
'Tinta','Média');
```

### tabela t\_marca\_propriedade

```
CREATE TABLE t_marca_propriedade(  
marca_id INT,  
propriedade_ID INT,  
PRIMARY KEY(marca_id, propriedade_id),  
FOREIGN KEY(marca_id) REFERENCES t_marca(marca_id),  
FOREIGN KEY(propriedade_id) REFERENCES t_propriedade(propriedade_id));
```

```
INSERT INTO t_marca_propriedade VALUES(1,3);  
INSERT INTO t_marca_propriedade VALUES(2,3);  
INSERT INTO t_marca_propriedade VALUES(3,3);  
INSERT INTO t_marca_propriedade VALUES(4,2);  
INSERT INTO t_marca_propriedade VALUES(5,2);  
INSERT INTO t_marca_propriedade VALUES(6,3);  
INSERT INTO t_marca_propriedade VALUES(7,3);  
INSERT INTO t_marca_propriedade VALUES(8,3);  
INSERT INTO t_marca_propriedade VALUES(9,1);  
INSERT INTO t_marca_propriedade VALUES(9,4);  
INSERT INTO t_marca_propriedade VALUES(10,1);  
INSERT INTO t_marca_propriedade VALUES(10,4);  
INSERT INTO t_marca_propriedade VALUES(11,1);  
INSERT INTO t_marca_propriedade VALUES(11,4);  
INSERT INTO t_marca_propriedade VALUES(12,4);  
INSERT INTO t_marca_propriedade VALUES(13,4);
```

### tabela t\_marca\_casta

```
CREATE TABLE t_marca_casta (  
marca_id INT,  
casta_id INT,  
percentagem FLOAT,  
PRIMARY KEY(marca_id, casta_id),  
FOREIGN KEY(marca_id) REFERENCES t_marca(marca_id),  
FOREIGN KEY(casta_id) REFERENCES t_casta(casta_id));
```

```
INSERT INTO t_marca_casta VALUES (1,8,0.2);  
INSERT INTO t_marca_casta VALUES (1,1,0.3);
```

```
INSERT INTO t_marca_casta VALUES (1,2,0.5);
INSERT INTO t_marca_casta VALUES (2,8,0.6);
INSERT INTO t_marca_casta VALUES (2,1,0.2);
INSERT INTO t_marca_casta VALUES (2,2,0.2);
INSERT INTO t_marca_casta VALUES (4,11,0.95);
INSERT INTO t_marca_casta VALUES (4,12,0.05);
INSERT INTO t_marca_casta VALUES (5,13,1);
INSERT INTO t_marca_casta VALUES (6,2,0.7);
INSERT INTO t_marca_casta VALUES (6,9,0.3);
INSERT INTO t_marca_casta VALUES (7,9,1);
INSERT INTO t_marca_casta VALUES (8,2,null);
INSERT INTO t_marca_casta VALUES (8,14,null);
INSERT INTO t_marca_casta VALUES (9,1,0.2);
INSERT INTO t_marca_casta VALUES (9,2,0.4);
INSERT INTO t_marca_casta VALUES (9,8,0.4);
INSERT INTO t_marca_casta VALUES (10,1,0.8);
INSERT INTO t_marca_casta VALUES (10,3,0.05);
INSERT INTO t_marca_casta VALUES (10,8,0.15);
INSERT INTO t_marca_casta VALUES (11,7,null);
INSERT INTO t_marca_casta VALUES (11,12,null);
```