

Bases de datos

Unidade 5: Consultas con datos de máis dunha táboa

1. A02. Consultas con datos de máis dunha táboa

1.1 Introducción

1.1.1 Obxectivos

Os obxectivos desta actividade son:

- Realizar consultas con datos de máis dunha táboa utilizando a composición de táboas e a unión de consultas.

1.1.2 Software

Utilizarase a plataforma WAMP (Windows-Apache-MySQL-PHP) WampServer 2.5 (última versión estable en outubro 2015), que inclúe MySQL Community Edition 5.6.17 como SXBDR (Sistema Xestor de Bases de Datos Relacional). As razóns de utilización deste software son que:

- É software libre, polo que o alumnado poderá descargalo de forma gratuíta e utilízalo legalmente na súa casa.
- É unha forma sinxela de facer a instalación do software necesario para desenvolver aplicacións web.



Páxina oficial de  WampServer: <http://www.wampserver.com>



Páxina oficial de  MySQL: <https://www.mysql.com/>

Utilizarase MySQL Workbench 6.3 como ferramenta cliente gráfica xa que é a recomendada por MySQL en outubro de 2015, aínda que tamén poderían utilizarse outras como phpMyAdmin, EMS MyManager, ou MySQL Query Browser.

Normalmente, para a proba das consultas realizadas nesta actividade, mostrarase a zona de manipulación de datos de Workbench, coas filas que forman a táboa de resultados. Para completar a anterior información ou cando a consulta non pode mostrarse enteira xa que devolve moitas filas, mostrarase ademais a zona de saída (output) coa información do estado da execución da consulta e o número de filas que devolve.



En <https://www.mysql.com/products/workbench/> pode obterse información detallada sobre a ferramenta MySQL Workbench e descargar o software.



En <http://dev.mysql.com/doc/index-gui.html> pode descargarse o manual de MySQL Workbench.



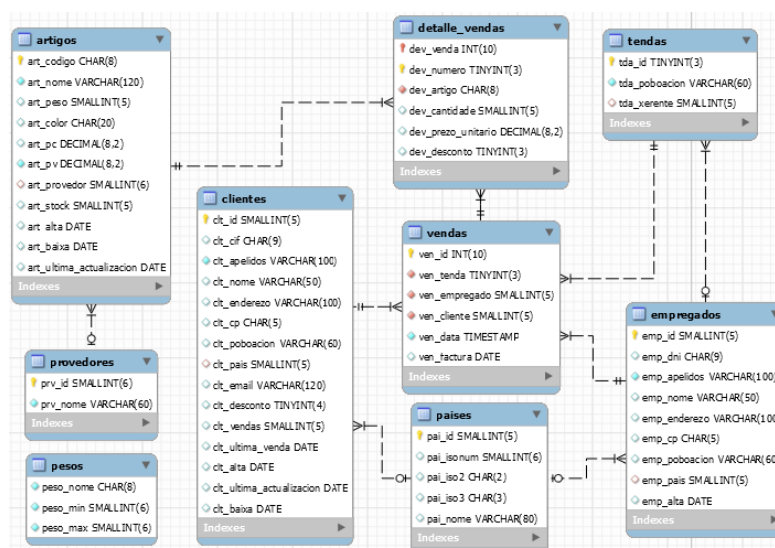
O material anexo a esta actividade inclúe unha guía básica de MySQL Workbench

1.1.3 Bases de datos de traballo

As bases de datos *tendaBD*, *traballadores* e *practicar5* utilizaranse para os exemplos e tarefas desta actividade. Antes de empezar a probar os exemplos ou realizar as tarefas, hai que executar os scripts de creación no servidor e poñer en uso a base de datos correspondente. Os scripts atópanse no cartafol anexo a esta actividade descrito no apartado '3.3 Material auxiliar'.

1.1.3.1 Base de datos tendaBD

A base de datos *tendaBD* serve para controlar as vendas dunha cadea de tendas. Gárdanse nela os datos das vendas que se realizan, das tendas nas que se fan as vendas, dos artigos vendidos, e dos clientes. As táboas desta base de datos que se van a utilizar nesta actividade móstranse no seguinte diagrama entidade relación deseñado con Workbench e descríbense a continuación.



■ Táboa *empregados*

Nome columna	Tipo	Null	Clave	Observacións
emp_id	smallint unsigned	Non	Primaria	Identificador do empregado. Numéranse de 1 en diante de forma automática.
emp_dni	char(9)			DNI do empregado.
emp_apellidos	varchar(100)	Non	Índice	Apelidos do empregado.
emp_nome	varchar(50)			Nome do empregado.
emp_enderezo	varchar(100)			Enderezo do empregado.
emp_cp	char(5)			Código postal do empregado.
emp_poboacion	varchar(60)			Poboación do empregado.
emp_pais	smallint unsigned		Foránea	Código do país segundo a táboa de países.
emp_alta	date			Data na que se deu de alta o empregado.

■ Táboa *pesos*

Nome columna	Tipo	Null	Clave	Observacións
peso_nome	char(8)	Non		Nome que describe o tipo de peso.
peso_min	smallint	Non		Peso mínimo para ese nome.
peso_max	smallint	Non		Peso máximo para ese nome.

■ Táboa *clientes*

Nome columna	Tipo	Null	Clave	Observacións
clt_id	smallint unsigned	Non	Primaria	Identificador do cliente. Numeraranse de 1 en diante de forma automática.
clt_cif	char(9)		Única	
clt_apelidos	varchar(100)	Non	Índice	Apelidos ou razón social do cliente.
clt_nome	varchar(50)			Nome ou tipo de sociedade (SL, SA, ...) do cliente.
clt_enderezo	varchar(100)			
clt_cp	char(5)			Código postal do cliente.
clt_poboacion	varchar(60)			
clt_pais	smallint unsigned		Foránea	Código do país segundo a táboa de países.
clt_email	varchar(120)			
clt_desconto	tinyint			Porcentaxe de desconto aplicable ao cliente.
clt_vendas	smallint unsigned			Número de vendas feitas ao cliente.
clt_ultima_venta	date			Data da última venda feita ao cliente.
clt_alta	date	Non		Data na que se deu de alta ao cliente.
clt_ultima_actualizacion	date			Data da última vez que se fixeron cambios nos datos do cliente.
clt_baixa	date			Data na que se deu de baixa ao cliente.

■ Táboa *artigos*

Nome columna	Tipo	Null	Clave	Observacións
art_codigo	char(8)	Non	Primaria	Toma valores entre 1 e 200.000.
art_nome	varchar(120)	Non	Índice	Nome ou descrición do artigo.
art_peso	smallint unsigned			Peso en gramos. Valor numérico enteiro.
art_color	char(20)			Cor do artigo
art_pc	decimal(8,2)			Prezo de compra do artigo.
art_pv	decimal(8,2)	Non		Prezo de venda do artigo.
art_proveedor	smallint		Foránea	Identificador do proveedor.
art_stock	smallint unsigned			Número de unidades do artigo dispoñibles no almacén.
art_alta	date	Non		Data na que se deu de alta o artigo.
art_baixa	date			Data na que se deu de baixa o artigo.
art_ultima_actualizacion	date			Data da última vez que se fixeron cambios nos datos do artigo.

■ Táboa *países*

Nome columna	Tipo	Null	Clave	Observacións
pai_id	smallint unsigned	Non	Primaria	Identificador do país. Numeraranse de 1 en diante de forma automática.

pai_isonum	smallint			Número de país segundo a norma ISO 3166-1:2013. ¹
pai_iso2	char(2)			Código de país de 2 caracteres segundo a norma ISO 3166-1:2013.
pai_iso3	char(3)			Código de país de 3 caracteres segundo a norma ISO 3166-1:2013.
pai_nome	varchar(80)			Nome do país.

■ Táboa *provedores*

Nome columna	Tipo	Null	Clave	Observacións
prv_id	smallint	Non	Primaria	Identificador do provedor.
prv_nome	varchar(60)	Non		Nome do provedor.

■ Táboa *tendas*

Nome columna	Tipo	Null	Clave	Observacións
tda_id	tinyint unsigned	Non	Primaria	Identificador da tenda. Numéranse do 1 en diante de forma automática.
tda_poboacion	varchar(60)	Non		Poboación na que está situada a tenda.
tda_xerente	smallint unsigned		Foránea	Identificador do empregado que é xerente da tenda.

■ Táboa *ventas*

Nome columna	Tipo	Null	Clave	Observacións
ven_id	int unsigned	Non	Primaria	Identificador da venda. Numeraranse de 1 en diante de forma automática.
ven_tenda	tinyint unsigned	Non	Foránea	Identificador da tenda na que se fixo a venda.
ven_empregado	smallint unsigned	Non	Foránea	Identificador do empregado que fixo a venda.
ven_cliente	smallint unsigned	Non	Foránea	Identificador do cliente ao que se fixo a venda.
ven_data	date	Non		Data e hora na que se fixo a venda.
ven_factura	date			Data da factura na que se inclúe esta venda.

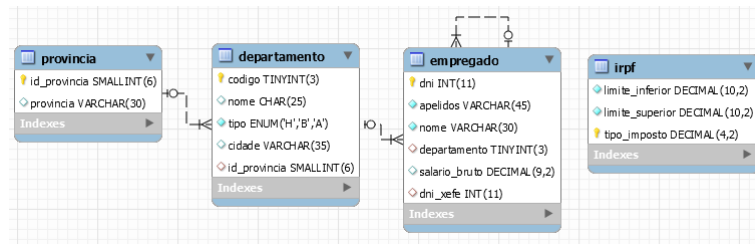
■ Táboa *detalle_ventas*

Nome columna	Tipo	Null	Clave		Observacións
dev_venta	int unsigned	Non	Primaria	Foránea	Identificador da venda á que corresponde a liña de detalle.
dev_numero	tinyint unsigned	Non			Número da liña de detalle dentro da venda.
dev_artigo	char(8)	Non	Foránea		Identificador do artigo vendido.
dev_cantidade	smallint unsigned	Non			Número de unidades vendidas.
dev_prezo_unitario	decimal(8,2) unsigned	Non			Prezo por cada unidade vendida.
dev_desconto	tinyint unsigned	Non			Porcentaxe de desconto aplicado.

1.1.3.2 Base de datos *practicass5*

A base de datos *practicass5* está creada con fins didácticos para realizar os exemplos de consultas nesta unidade. Está formada por un grupo de táboas, relacionadas entre si, tal e como se mostra no seguinte diagrama entidade relación deseñado con Workbench e se describe a continuación.

¹ Máis información sobre a norma ISO 3166-1:2013 en https://es.wikipedia.org/wiki/ISO_3166-1



- Táboa *empleado*. A columna *departamento* é unha clave foránea que contén o código do departamento no que traballa o empregado, e fai referencia á columna *codigo* da táboa *departamento*. Os valores que toma a columna *departamento* teñen que coincidir cos que toma a columna *codigo* da táboa *departamento*, ou ser NULL no caso que o empregado non teña asignado ningún departamento. A columna *dni_xefe* é outra clave foránea que contén o dni doutro empregado que sería o seu xefe, ou o valor NULL no caso que non tivera xefe.
- Táboa *departamento*. A columna *id_provincia* é unha clave foránea que fai referencia á columna *id_provincia* da táboa *provincia*.
- Táboa *irpf*. Contén a porcentaxe de imposto que hai que aplicarlle a cada empregado, en función do seu salario bruto, dependendo dos límites entre os que se atope. Esta táboa podería conter unha información similar a esta:

limite_inferior	limite_superior	tipo_imposto
0.00	17707.00	15.75
17707.00	33007.00	21.00
33007.00	53407.00	27.00
53407.00	120000.00	30.00
120000.00	175000.00	35.00
175000.00	300000.00	42.00

Salario bruto entre 0 e 17107 euros corresponde un 15.75% de imposto

1.1.3.3 Base de datos traballadores

A base de datos *traballadores* serve para levar control dos empregados, departamentos e centros dunha empresa. Está formada por un grupo de táboas, relacionadas entre si, tal e como se mostra no seguinte grafo relacional e se describe a continuación. As táboas son MyIsam (non transaccionais) e por tanto non teñen definidas claves foráneas.

EMPREGADO (*empNumero*, *empDepartamento*, empExtension, empDataNacemento, empDatIngreso, empSalario, empComision, empFillos, empNome)

DEPARTAMENTO (*depNumero*, depNome, *depDirector*, depPresuposto, *depDepende*, *depCentro*, depEmplegados)

CENTRO (*cenNumero*, cenNome, cenEnderezo)

▪ Táboa centro

Nome columna	Tipo	Null	Clave	Observacións
cenNumero	int	Non	Primaria	Número co que se identifica.
cenNome	char(30)		Índice	Nome.
cenEnderezo	char(30)			Enderezo.

▪ Táboa empleado

Nome columna	Tipo	Null	Clave	Observacións
empNumero	int	Non	Primaria	Número co que se identifica.
empDepartamento	int	Non	Índice	Número do departamento no que traballa.
empExtension	smallint	Non		Extensión telefónica para o empregado. Pode compartirse entre empregados de diferentes departamentos.
empDataNacemento	date			Data de nacemento.
empDataIngreso	date			Data de ingreso na empresa.
empSalario	decimal(6,2)			Salario mensual en euros.
empComision	decimal(6,2)			Comisión mensual.
empFillos	smallint			Número de fillos.
empNome	char(20)	Non	Índice	Nome do empregado coa forma: primeiro apelido, nome.

■ Táboa departamento

Nome columna	Tipo	Null	Clave	Observacións
depNumero	int	Non	Primaria	Número co que se identifica.
depNome	char(20)		Índice	Nome.
depDirector	int	Non	Índice	Número do empregado director do departamento.
deptipoDirector	char(1)			Tipo de directo: P (en propiedade, é dicir, titular), F (en funcións).
depPresuposto	decimal(9,2)			Cantidade en euros de presuposto anual.
depDepende	int		Índice	Número do departamento do que depende.
depCentro	int		Índice	Número do centro ao que pertence.
depEmpregados	smallint unsigned			Número de empregados que traballan no departamento.

1.2 Actividade

1.2.1 Consultas con datos de máis dunha táboa

Na primeira actividade desta unidade didáctica, realizáronse consultas simples con datos dunha única táboa pero a práctica diaria de consultas sobre unha base de datos implica frecuentemente a máis dunha táboa. Este tipo de consultas pódense realizar de varias formas en función do requirimento da consulta:

- Utilizar composicións de táboas nunha sentenza SELECT escribindo unha única consulta que utiliza a cláusula JOIN dentro da cláusula FROM.
- Utilizar sentenzas SELECT aniñadas dentro doutras sentenzas SELECT. As sentenzas aniñadas chámanse subconsultas e veranse máis adiante na actividade 4.
- Unindo o conxunto de resultados dunha consulta co conxunto de resultados doutras consultas, empregando o operador UNION.

1.2.2 Composición de táboas

As primeiras normas ANSI SQL para combinar varias táboas nunha consulta permitían poñer a relación das táboas na cláusula FROM separadas por coma, e as condicións para facer os enlaces entre as táboas na cláusula WHERE. Esta sintaxe sigue estando permitida hoxe en día, aínda que non é a máis recomendable.

Exemplo: seleccionar apelidos, nome e cidade dos empregados que estean asignados a un departamento, tendo en conta que a cidade é a cidade na que está situado o departamento no que traballa o empregado.

```
/* Enlace entre dúas táboas utilizando unha clave foránea*/
select apelidos, empregado.nome, cidade
from empregado, departamento
where departamento = codigo;
```

A primeira operación que se realiza cunha sentenza como a anterior na que se fai referencia a máis dunha táboa na cláusula FROM, é o produto cartesiano entre esas táboas, é dicir, relaciona cada fila dunha táboa con todas as filas da outra táboa. Por iso despois hai que poñer a condición de enlace na cláusula WHERE para que se seleccionen só aquelas composicións de filas que nos interesan.

O número de filas que devolve a consulta do exemplo é 19 tal e como se observa na imaxe seguinte, a pesar de que hai 20 empregados. Isto é debido a que hai un empregado que ten na columna *departamento* o valor NULL, por non estar asignado a ningún departamento, e non se pode combinar con ningunha fila da táboa *departamento* porque a columna *código* é clave primaria e non pode ter o valor NULL.

apelidos	nome	cidade
García Perez	Adrian	Ourense
Canedo Tellez	Angeles	Vigo
Nuñez Bernardéz	Antonia	Monforte
Porto Novo	Begoña	Coruña
Martínez Iglesias	Benito	Lugo
Martínez Díaz	Carlos	Monforte
Ruiz Macías	Dario	Villalba
Abelleira Carrion	Dorinda	Villalba

317 23:27:37 select empregado.apelidos, empregado.nome, d... 19 row(s) returned

0.0100 sec / 0.000 sec

A sintaxe das composicións de táboas cambia a partir da norma ANSI92, para poñer as condicións de enlace na cláusula FROM xunto coa relación de táboas, coa finalidade de deixar a cláusula WHERE para as condicións que deben cumprir as filas que hai que mostrar.

Nesta actividade, utilizarase a sintaxe que recomenda a norma ANSI92 por ser máis eficiente e estruturar mellor o código das consultas.

Resumo de tipo de composición máis utilizados		
Composición interna	INNER JOIN con condición de igualdade INNER JOIN sen condición de igualdade	
Composición externa	LEFT OUTER RIGHT OUTER	Prioridade esquerda Prioridade dereita
Composición natural	NATURAL JOIN	Pode ser interna ou externa
Autocomposición	Composición dunha táboa consigo mesma	Utilizando alias de táboas

Nomes de columna cualificados

Nas combinacións de tipo interna, externa e autocomposición, hai que ter moi en conta que pode haber columnas que teñan o mesmo nome en dúas ou máis táboas. Cando sucede isto é obrigatorio cualificar os nomes das columnas, utilizando o formato *nome_táboa.nome_columna*, para evitar erros producidos polo uso de nomes ambiguos.

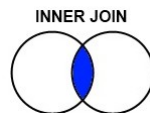
Tamén é recomendable utilizar os nomes de columna cualificados para mellorar o rendemento da consulta, porque proporcionan información a o servidor que desta maneira

non ten que buscar a que táboa pertence cada columna. Para simplificar o uso de nomes cualificados e que estes sexan máis curtos, recoméndase utilizar alias para os nomes de táboa. Exemplo:

```
select em.apelidos, em.nome, de.cidade
from empregado as em, departamento as de
where em.departamento = de.codigo;
```

1.2.2.1 Composición interna

A composición interna crea unha táboa que contén todas as columnas das táboas que forman a composición e só as filas que cumpren as condicións da composición.



Composición interna con INNER JOIN e condición de igualdade

Este tipo de composición é a máis utilizada e permite relacionar dúas táboas que teñen algunha columna con datos comúns que serve de enlace entre elas. O caso máis normal é o que establecen as claves foráneas dunha táboa, que toman valores que teñen que coincidir cos valores da clave primaria doutra táboa. Esta composición dá como resultado o conxunto formado polas parellas de filas das dúas táboas que interveñen na composición, que cumpran a condición de que o valor da clave foránea dunha sexa igual ao valor da clave primaria da outra. As columnas de enlace deben ter asociados índices para optimizar o rendemento da consulta.

A sintaxe utilizada na norma ANSI92 para a reunión interna é:

```
FROM nome_táboa_1 [ INNER ] JOIN nome_táboa_2
ON ( condición de enlace )
```

A opción INNER indica o tipo de composición interna, pero non é necesario poñela explicitamente. Cando se pon só a palabra JOIN entre os nomes de dúas táboas enténdese que é unha composición interna (INNER).

Exemplo: seleccionar *apelidos*, *nome* e *cidade* dos empregados que estean asignados a un departamento, tendo en conta que *cidade* é a cidade na que está situado o departamento no que traballa o empregado. A condición de enlace escríbese na cláusula FROM, reservando a cláusula WHERE para establecer condicións que seleccionen as filas das táboas de orixe que nos interesan.

```
/* Enlace entre dúas táboas utilizando unha clave foránea
con INNER JOIN segundo a norma ANSI-92*/
select em.apelidos, em.nome, de.cidade
from empregado as em inner join departamento as de on (em.departamento = de.codigo);
```

Cando interveñen máis de dúas táboas na composición, hai que ter en conta que para cada JOIN ten que existir unha condición de enlace. Neste caso, pódese asociar unha cláusula ON a cada JOIN coa correspondente condición de enlace, ou ben poñer unha única cláusula ON cunha condición composta formada por todas as condicións de enlace relacionadas con operadores AND.

Exemplo: mostrar *apelidos* e *nome* do empregado, nome do departamento no que traballa e nome da provincia no que está o departamento.

```
select em.apelidos, em.nome, de.nome, pr.provincia
from empregado as em
join departamento as de on (em.departamento = de.codigo)
```

```
join provincia as pr on (de.id_provincia = pr.id_provincia);
```

Tamén se podería escribir utilizando unha única condición de enlace composta:

```
select em.apellidos,em.nome,de.nome,pr.provincia
from empregado as em
join departamento as de
join provincia as pr
on (em.departamento = de.codigo and de.id_provincia = pr.id_provincia);
```

Calquera destas consultas obtén o seguinte resultado:

Result Grid		Filter Rows:		Exp
	apellidos	nome	nome	provincia
▶	Martinez Iglesias	Benito	Central	Lugo
	Fernandez Lopez	Jose Luis	Central	Lugo
	Fernandez Diaz	Julian	Central	Lugo
	Nuñez Bernardes	Antonia	Oficina1	Lugo
	Martinez Diaz	Carlos	Oficina1	Lugo
	Hernandez Valin	Valentina	Oficina2	Coruña, A

191 11:22:44 select apellidos,empleado nome,departamento nome.provincia from empregado... 19 row(s) returned

Composición interna con INNER JOIN sen condición de igualdade

Cando se fai unha composición entre dúas táboas utilizando unha composición interna, é posible utilizar condicións de enlace distintas da condición de igualdade; neste caso suponse que non existe ningunha columna que conteña valores idénticos nas dúas táboas.

Exemplo: seleccionar *apellidos*, *nome*, salario bruto, tipo de imposto e salario neto (salario bruto – impostos) de todos os empregados ordenados por nome.

```
/* JOIN sen condición de igualdade */
select em.apellidos, em.nome, em.salario_bruto, ir.tipo_imposto,
round(em.salario_bruto-em.salario_bruto*ir.tipo_imposto/100,2) as salario_netos
from empregado as em join irpf as ir
on em.salario_bruto between ir.limite_inferior and ir.limite_superior
order by em.nome;
```

Neste caso, o *salario_bruto* é a columna da táboa *empleado* que se vai a utilizar como enlace e non ten que coincidir exactamente co valor almacenado en ningunha columna da táboa *irpf*, senón que ten que estar entre un *limite_inferior* e un *limite_superior*, ambos inclusive. O resultado da execución en Workbench podería ser:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	
apellidos	nome	salario_bruto	tipo_imposto	salario_neto
Iglesias Dominguez	Adolfo	52500.00	27.00	38325.00
Garcia Perez	Adrian	21500.00	21.00	16985.00
Canedo Tellez	Angeles	58500.00	30.00	40950.00
Nuñez Bernardes	Antonia	42000.00	27.00	30660.00
Porto Novo	Begoña	52000.00	27.00	37960.00
Martinez Iglesias	Benito	25000.00	21.00	19750.00

102 18:31:53 select apellidos, nome, salario_bruto, tipo_imposto... 20 row(s) returned 0.000 sec / 0.000 sec



Tarefa 1. Realizar consultas con datos de máis dunha táboa utilizando unha composición interna.

1.2.2.2 Composición externa con OUTER JOIN

A composición externa permite mostrar as filas dunha das táboas aínda que non cumpran a condición de enlace (toma o valor NULL para as columnas do resto de táboas), ademais

das parellas de filas para as que se cumpra a condición de enlace tal e como ocorrería na composición interna.



Existen dous tipos de reunión externa: pola esquerda, e pola dereita, dependendo de cal das táboas se considere táboa principal. A sintaxe é:

```
FROM nome_táboa_1 { LEFT | RIGTH } [ OUTER ] JOIN nome_táboa_2
ON ( condición de enlace )
```

- A opción LEFT mostra todas as filas da táboa da esquerda, aínda que non estean relacionas con filas do resto das táboas.
- A opción RIGHT, mostra todas as filas da táboa da dereita, aínda que non estean relacionas con filas do resto das táboas.

Exemplo: mostrar *apelidos* e *nome* dos empregados e nome da cidade na que está o departamento no que traballa, aínda que o empregado non teña departamento asignado. O resultado ten que estar ordenado polo nome.

```
/* OUTER JOIN enlace externo, neste caso, pola esquerda*/
select em.apelidos, em.nome, de.cidade
from empregado as em left join departamento as de
on (em.departamento = de.codigo)
order by em.nome;
```

Na sentenza anterior, a táboa principal é a táboa *empregado* por estar situada á esquerda (*left*), e a táboa *departamento* considérase a táboa secundaria da consulta. Na execución, móstrase unha fila para cada empregado, e para os que non cumpran a condición de enlace por non ter asignado aínda un departamento, móstrase o valor NULL na columna *cidade*. O número de filas que devolve a consulta é 20 que é o número de filas que ten a táboa *empregado*.

apelidos	nome	cidade
Iglesias Dominguez	Adolfo	NULL
Garcia Perez	Adrian	Ourense
Canedo Tellez	Angeles	Vigo
Nuñez Bernardez	Antonia	Monforte
Porto Novo	Begoña	Coruña
Martinez Iglesias	Benito	Lugo
Martinez Diaz	Carlos	Monforte
Ruiz Macias	Dario	Villalba

310 22:10:19 select empregado.apelidos, empregado.nome, d... 20 row(s) returned 0.000 sec / 0.000 sec

Exemplo de LEFT JOIN con IF NULL: mostrar *apelidos* e *nome* dos empregados que non teñen departamento asignado.

```
select em.apelidos, em.nome
from empregado as em left join departamento as de
on (em.departamento = de.codigo)
where de.cidade is null; # Vale calquera columna da táboa departamento
```

Para as filas da táboa principal (*empregado*) que non cumpran a condición de enlace, todas as columnas da táboa secundaria (*departamento*) toman o valor NULL, polo que ao establecer a condición da cláusula WHERE só se mostran as filas da táboa principal que non están relacionadas con ningunha fila da táboa secundaria.

Result Grid	Filter Rows
apellidos	nome
Iglesias Dominguez	Adolfo



Tarefa 2. Realizar consultas con datos de máis dunha táboa utilizando unha composición externa.

1.2.2.3 Composición con NATURAL JOIN

A composición natural permite enlazar táboas por columnas que teñen o mesmo nome. Neste tipo de composicións non é necesario introducir a condición de enlace coa opción ON, pois sobreenténdese que a condición de enlace consiste en comprobar a coincidencia dos valores das columnas que teñen o mesmo nome. Ademais, as columnas que existen nas dúas táboas só se mostran unha vez, polo que non é necesario cualificalas empregando o formato *nome_táboa.nome columna*. Sintaxe:

```
FROM nome_táboa_1 NATURAL [{ LEFT | RIGH } [ OUTER ]] JOIN nome_táboa_2
```

A combinación con NATURAL JOIN pode dar prioridade a algunha das táboas, cando se define como enlace externo (OUTER).

Cando se utiliza NATURAL JOIN hai que ter coidado que nas dúas táboas só teñan o mesmo nome as columnas que se van a utilizar para facer o enlace, xa que en calquera outro caso os resultados obtidos poden ser imprevisibles.

Exemplo: mostrar os datos de todas as columnas da táboa *departamento*, completados co nome da provincia da táboa *provincia*.

```
/* Utilización de NATURAL JOIN */
select de.*, pr.provincia
from departamento as de natural join provincia as pr
order by de.codigo;
```

Result Grid		Filter Rows:		Export:	
	codigo	nome	cidade	id_provincia	provincia
▶	1	Central	Lugo	27	Lugo
	2	Oficina1	Monforte	27	Lugo
	3	Oficina2	Ferrol	15	Coruña, A
	4	Oficina3	Vigo	36	Pontevedra
	5	Oficina4	Ourense	32	Ourense

101 18:29:11 select departamento.*, provincia from departame... 10 row(s) returned

0.000 sec / 0.000 sec

Na mesma consulta pódense utilizar distintos tipos de composición.

Exemplo: mostrar apelidos e nome do empregado, nome do departamento no que traballa e nome da provincia no que está o departamento.

```
/* Utilización de dúas composicións, una delas NATURAL JOIN */
select em.apellidos,em.nome,de.nome,pr.provincia
from empregado as em join departamento as de on (em.departamento = de.codigo)
natural join provincia as pr;
```

Result Grid		Filter Rows:		
	apellidos	nome	nome	provincia
▶	Martinez Iglesias	Benito	Central	Lugo
	Fernandez Lopez	Jose Luis	Central	Lugo
	Fernandez Diaz	Julian	Central	Lugo
	Nuñez Bernardez	Antonia	Oficina1	Lugo
	Martinez Diaz	Carlos	Oficina1	Lugo

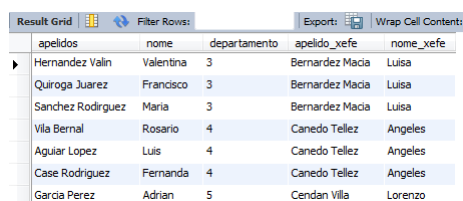
207 13:34:24 select em.apellidos,em.nome,de.nome,pr.provincia from empregado a... 19 row(s) returned

1.2.2.4 Composición dunha táboa con ela mesma. Autocomposición

A autocomposición utilízase cando hai que comparar unha fila dunha táboa con outras filas da mesma táboa. Para evitar que ao ler unha fila da táboa se perda a información da anterior fila lida, pódese poñer dúas veces o nome da táboa na cláusula FROM, asignándolle dous alias ou sinónimos diferentes; a partir dese momento, manéxanse como se foran dúas táboas distintas. As consultas feitas cunha autocomposición poden ser resoltas tamén empregando subsentenzas, como se verá en actividades posteriores.

Exemplo: mostrar apelidos e nome dos empregados, e apelidos e nome do seu xefe.

```
/* Autocomposición: Enlace dunha táboa con ela mesma */
select em1.apelidos, em1.nome, em1.departamento,
       em2.apelidos as Apelido_xefe, em2.nome as Nome_xefe
from empregado as em1 join empregado as em2
  on em1.dni_xefe = em2.dni
order by Apelido_xefe, Nome_xefe;
```



apelidos	nome	departamento	apelido_xefe	nome_xefe
Hernandez Valin	Valentina	3	Bernardez Macia	Luisa
Quiroga Juarez	Francisco	3	Bernardez Macia	Luisa
Sanchez Rodriguez	Maria	3	Bernardez Macia	Luisa
Vila Bernal	Rosario	4	Canedo Tellez	Angeles
Aguilar Lopez	Luis	4	Canedo Tellez	Angeles
Case Rodriguez	Fernanda	4	Canedo Tellez	Angeles
Garcia Perez	Adrian	5	Cendan Villa	Lorenzo

211 15:13:10 select em1.apelidos, em1.nome, em1.departamento, em2.apelidos... 19 row(s) returned



Tarefa 3. Realizar consultas dunha táboa consigo mesma.

Normas para a realización de composicións internas e externas

Non existen normas estándar para as composicións internas e externas, pero a continuación enuméranse unha serie de conclusións sobre as composicións internas e externas que axudarán a utilizalas mellor.

- As columnas que se utilizan para facer a composición deben ser do mesmo tipo. Para mellorar o rendemento da consulta deben ter asociados índices.
- Cando se fai unha composición, é como se crease unha táboa que ten as columnas de todas as táboas que se combinan, e as filas que verifican as condicións de composición. Por esta razón, na sentenza SELECT pódese utilizar calquera columna das táboas que se relacionan na cláusula FROM.
- Pódense combinar tantas táboas como desexemos, pero poñer táboas innecesarias reduce o rendemento da consulta.
- Unha mesma táboa pode aparecer máis dunha vez na composición, pero utilizando alias diferentes.
- Recoméndase utilizar nomes de columna cualificados. Ver o apartado '*Nomes de columna cualificados*' ao principio da actividade.
- Se as condicións de composición van na cláusula WHERE, hai que ter en conta que primeiro debemos poñer todas aquelas condicións que non sexan de composición. Deste xeito o produto cartesiano previo ao *join* terá menos filas que combinar.
- Se unha táboa ten unha clave primaria composta, nas condicións de composición hai que facer referencia á clave enteira. Exemplo: Temos unha táboa *t1* que ten unha clave

primaria composta formada polas columnas (*cp1, cp2*), e unha táboa *t2* que ten unha clave foránea composta polas columnas (*cf1, cf2*) que fai referencia á clave primaria da táboa *t1*. A condición de composición podería ser:

`t1.cp1 = t2.cf1 and t1.cp2 = t2.cf2`

Condición composta

`(t1.cp1, t1.cp2) = (t2.cf1, t2.cf2)`

Comparación de valores tipo fila

1.2.3 Unión de consultas (UNION)

A cláusula UNION utilízase para combinar o resultado de varias consultas nun único conxunto de resultados. A sintaxe é:

```
Sentenza SELECT
UNION [ALL | DISTINCT]
Sentenza SELECT
[UNION [ALL | DISTINCT]
Sentenza SELECT] ...
```

- As opcións ALL e DISTINCT permiten indicar se hai que mostrar as filas duplicadas ou non. O valor por defecto é DISTINCT. Para que se mostren as filas duplicadas hai que poñer a opción ALL.
- As sentenzas SELECT que se utilizan na UNION teñen que ter o mesmo número de columnas na lista de selección. As columnas seleccionadas teñen que ser do mesmo tipo en todas as sentenzas SELECT. Por exemplo, se a primeira consulta selecciona tres columnas, o resto das sentenzas SELECT deben seleccionar tamén tres columnas; se a primeira columna da primeira consulta é de tipo char(50), a primeira columna do resto de sentenzas SELECT ten que ser de tipo char(50).
- Os nomes das columnas da táboa resultante da UNION son os que corresponden aos nomes das columnas da primeira sentenza SELECT.
- Restricións para as sentenzas SELECT que participan nunha unión:
 - A cláusula INTO OUTFILE só a pode levar a última sentenza.
 - Non se pode utilizar a opción HIGH_PRIORITY. Non se mostra erro no caso de poñela na primeira sentenza, pero non se ten en conta. No caso de poñela en calquera das outras sentenzas, móstrase unha mensaxe de erro de sintaxe.
- Cando se queren utilizar as cláusulas ORDER BY ou LIMIT, hai que pechar as sentenzas SELECT entre parénteses.
- Non ten efecto a cláusula ORDER BY dentro dos parénteses, xa que a UNION produce un conxunto de filas desordenadas. Só ten sentido utilizar a cláusula ORDER BY para unha sentenza SELECT individual cando se combina coa cláusula LIMIT.

Exemplo extraído do manual de referencia de MySQL 5.6:

```
(SELECT a FROM tbl_name WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM tbl_name WHERE a=11 AND B=2 ORDER BY a LIMIT 10);
```

- No caso de querer ordenar as filas que forman o conxunto de resultados, hai que poñer a cláusula ORDER BY fóra dos parénteses, despois da última sentenza SELECT. Tamén é posible limitar o número de filas que forman o conxunto de resultados poñendo a cláusula LIMIT fóra dos parénteses, despois da última sentenza SELECT.

Exemplo extraído do manual de referencia de MySQL 5.6:

```
(SELECT a FROM tbl_name WHERE a=10 AND B=1)
UNION
```

```
(SELECT a FROM tbl_name WHERE a=11 AND B=2)
ORDER BY a LIMIT 10;
```



Tarefa 4. Combinar o resultado de varias sentenzas SELECT para obter un conxunto de resultados único utilizando UNION.

1.3 Tarefas

As tarefas propostas son as seguintes:

- Tarefa 1. Realizar consultas con datos de máis dunha táboa utilizando unha composición interna.
- Tarefa 2. Realizar consultas con datos de máis dunha táboa utilizando unha composición externa.
- Tarefa 3. Realizar consultas dunha táboa consigo mesma.
- Tarefa 4. Combinar o resultado de varias sentenzas SELECT para obter un conxunto de resultados único utilizando UNION.

1.3.1 Tarefa 1. Realizar consultas con datos de máis dunha táboa utilizando unha composición interna

A tarefa consiste en realizar as seguintes consultas utilizando unha composición interna con condición de igualdade.

[Sobre a base de datos tendaBD](#)

- Tarefa 1.1. Seleccionar os artigos de cor negra e mostrar o seu número, nome e peso, así como o nome do proveedor.
- Tarefa 1.2. Seleccionar para todos os apelidos, nome e o nome da provincia na que residen. Os dous primeiros díxitos do código postal (*clt_cp*) corresponden ao código da provincia na que reside o cliente. Ordenar o resultado polo nome da provincia, e dentro da provincia, polos apelidos e nome, alfabeticamente.
- Tarefa 1.3. Mostrar para cada venda: nome e apelidos do cliente, día, mes, e ano da venda (cada un nunha columna).
- Tarefa 1.4. Mostrar unha lista que conteña: número de vendas, número de artigos vendidos, suma de unidades vendidas e a media dos prezos unitarios dos artigos vendidos.
- Tarefa 1.5. Seleccionar para cada artigo o seu número, nome, peso e o nome que corresponde ao peso (*peso_nome*), tendo en conta a información contida na táboa *pesos*, que da un nome aos pesos en función do intervalo ao que pertence. Ordenar o resultado polo peso do artigo, de maior a menor.
- Tarefa 1.6. Mostrar para cada venda: nome e apelidos do cliente, a data da venda con formato dd/mm/aa e os días transcorridos dende que se fixo a venda. Ordenar o resultado polo número de días transcorridos dende a venda.
- Tarefa 1.7. Seleccionar os nomes das provincias nas que temos clientes.
- Tarefa 1.8. Seleccionar para cada venda:
 - Datos da venda: identificador e data da venda.

- Datos do cliente: nome do cliente (nome e apelidos separados por coma).
- Datos do empregado: nome do empregado (nome e apelidos separados por coma).

Mostrar os datos ordenados polos apelidos e nome do cliente.

- Tarefa 1.9. Seleccionar información sobre os artigos vendidos. Para cada liña de detalle interesa:
 - Datos do cliente: apelidos e nome separados por coma, nunha única columna.
 - Datos do artigo: nome, cantidade, prezo unitario, desconto e o importe final para o cliente (resultado de multiplicar a cantidade polo prezo unitario e aplicar o desconto que corresponde).

Mostrar os resultados ordenados polo nome do artigo.

Sobre a base de datos traballadores

- Tarefa 1.10. Seleccionar o número e nome de departamento, xunto co nome do director, para os departamentos independentes, é dicir, que non dependen de ningún outro departamento.
- Tarefa 1.11. Mostrar nome (só nome, sen apelidos) e enderezo do *centro* ao que pertence o departamento no que traballa, dos empregados cun nome (sen ter en conta os apelidos) que empece por 'A'.
- Tarefa 1.12. Seleccionar para todos os empregados que non son directores, o nome de departamento no que traballa, o seu nome e salario, o nome e salario do director do seu departamento, e a diferenza do seu salario e o salario do director do departamento. Ordenar o resultado polo nome do departamento.

Solicítase esta información para facer un estudio da diferenza de salarios entre os directores dos departamentos e os traballadores que traballan no departamento.

Solución

- Tarefa 1.1

```

/*****
Seleccionar os artigos de cor negra e mostrar o seu número, nome e peso, así como o
nome do provedor.
*****/

select ar.art_codigo as Numero,
       ar.art_nome as Articulo,
       ar.art_peso as Peso,
       pr.prv_nome as Proveedor
from artigos as ar join provedores as pr on ar.art_proveedor=pr.prv_id
where ar.art_color='negro';

```

- Tarefa 1.2

```

/*****
Seleccionar para todos os apelidos, nome e o nome da provincia na que residen. Os dous
primeiros díxitos do código postal (clt_cp) corresponden ao código da provincia na que
reside o cliente. Ordenar o resultado polo nome da provincia, e dentro da provincia,
polos apelidos e nome, alfabeticamente.
*****/

select cl.clt_apelidos, cl.clt_nome, pr.pro_nome
from clientes as cl join provincias as pr on left(trim(cl.clt_cp),2)=pr.pro_id
order by pr.pro_nome, cl.clt_apelidos, cl.clt_nome;

```

- Tarefa 1.3

```

/*****
Mostrar para cada venda: nome e apelidos do cliente, día, mes, e ano da venda (cada
un nunha columna).
*****/

select cl.clt_nome as Nome_cliente,

```



```

cl.clt_apellidos as Apellidos,
day(ve.ven_data) as Dia_venta,
month(ve.ven_data) as Mes_venta,
year(ve.ven_data) as Ano_venta
from clientes as cl join vendas as ve on cl.clt_id=ve.ven_cliente;

```

■ Tarefa 1.4

```

/*****
Mostrar unha lista que conteña: número de vendas, número de artigos vendidos, suma de
unidades vendidas e a media dos prezos unitarios dos artigos vendidos. Facer os
cálculos coa información contida nas táboas vendas e detalle:vendas
*****/

select count(distinct ve.ven_id) as Numero_vendas,
       count(distinct dv.dev_artigo) as Numero_artigos,
       sum(dv.dev_cantidad) as Suma_unidades,
       round(avg(dv.dev_prezo_unitario),2) as Media_prezo
from vendas as ve join detalle_vendas as dv on dv.dev_venta = ve.ven_id;

```

■ Tarefa 1.5

```

/*****
Seleccionar para cada artigo o seu número, nome, peso e o nome que corresponde ao peso
(peso_nome), tendo en conta a información contida na táboa pesos, que da un nome aos
pesos en función do intervalo ao que pertence. Ordenar o resultado polo peso do artigo,
de maior a menor.
*****/

select ar.art_codigo,
       ar.art_nome,
       ar.art_peso,
       pe.peso_nome
from artigos as ar join pesos as pe on ar.art_peso between peso_min and peso_max
order by ar.art_peso;

```

■ Tarefa 1.6

```

/*****
Mostrar para cada venda: nome e apelidos do cliente, a data da venda con formato
dd/mm/aa e os días transcorridos dende que se fixo a venda. Ordenar o resultado polo
número de días transcorridos dende a venda.
*****/

select distinct cl.clt_apellidos as Apellidos,
               cl.clt_nome as Nome,
               date_format(ve.ven_data, '%d/%m/%Y') as Data_venta,
               datediff(curdate(),ve.ven_data) as Dias_diferencia
from vendas as ve join clientes as cl on ve.ven_cliente=cl.clt_id
order by Dias_diferencia;

```

■ Tarefa 1.7

```

/*****
Seleccionar os nomes das provincias nas que temos clientes.
*****/

select distinct pro_nome
from provincias as pr
join clientes as cl on pr.pro_id = left(trim(cl.clt_cp),2);

```

■ Tarefa 1.8

```

/*****
Seleccionar para cada venda:
Datos da venda: identificador e data da venda.
Datos do cliente: nome do cliente (nome e apelidos separados por coma).
Datos do empregado: nome do empregado (nome e apelidos separados por coma).
Mostrar os datos ordenados polos apelidos e nome do cliente
*****/

select ve.ven_id as Numero_venta,
       ve.ven_data as Data_da_venta,
       concat(cl.clt_apellidos,', ',cl.clt_nome) as Apellidos_nome_cliente,
       concat(em.emp_apelidos,', ',em.emp_nome) as Apellidos_nome_empleado
from vendas as ve
join clientes as cl on ve.ven_cliente = cl.clt_id

```

```

    join empregados as em on ve.ven_empleado = em.emp_id
order by Apellidos_nombre_cliente;

```

■ Tarefa 1.9

```

/*****
Selección información sobre os artigos vendidos. Para cada liña de detalle interesa:
- Datos do cliente: apellidos e nome separados por coma, nunha única columna.
- Datos do artigo: nome, cantidade, prezo unitario, desconto e o importe final para
o cliente (resultado de multiplicar a cantidade polo prezo unitario e aplicar o
desconto que corresponde).
Mostrar os resultados ordenados polo nome do artigo.
*****/

select concat(cl.clt_apellidos,', ' ,cl.clt_nombre) as Apellidos_nombre_cliente,
       ar.art_nombre as Artigo,
       dv.dev_cantidad as Cantidad,
       dv.dev_prezo_unitario as Prezo_unitario,
       dv.dev_desconto as '% Desconto',
       round((dv.dev_prezo_unitario*dv.dev_cantidad)*(1-dv.dev_desconto/100),2) as Total
from vendas as ve
     join clientes as cl on ve.ven_cliente = cl.clt_id
     join detalle_vendas as dv on ve.ven_id = dv.dev_venta
     join artigos as ar on dv.dev_artigo = ar.art_codigo
order by Artigo;

```

■ Tarefa 1.10

```

/* Selección o número e nome de departamento, xunto co nome do director, para os
departamentos independentes, é dicir, que non dependen de ningún outro departamento.*/

select de.depNumero as Numero,
       de.depNombre as Nome,
       em.empNombre as Director
from departamento as de join empleado as em on de.depDirector=em.empNumero
where de.depDepende is null;

```

■ Tarefa 1.11

```

/* Mostrar nome (só nome, sen apellidos) e enderezo do centro ao que pertence o
departamento no que traballa, dos empregados cun nome (sen ter en conta os apellidos)
que empece por 'A'.*/

select trim(right(em.empNombre, length(em.empNombre)-locate(',', em.empNombre ))) as Nome,
       ce.cenEnderezo as Enderezo
from empleado as em
     join departamento as de on em.empDepartamento = de.depNumero
     join centro as ce on de.depCentro=ce.cenNumero
where trim(right(em.empNombre, length(em.empNombre)-locate(',', em.empNombre))) like 'A%';
/* Ollo: A función TRIM elimina espazos en branco ao principio e ao final da cadea.
Non é o mesmo utilizar right(em.empNombre, length(em.empNombre)-locate(',', em.empNombre)-1)
xa que hai un empleado que non ten espazo en branco despois da coma e ten A como
segunda letra do nome: SANTOS,SANCHO */

```

■ Tarefa 1.12

```

/* Selección para todos os empregados que non son directores, o nome de departamento
no que traballa, o seu nome e salario, o nome e salario do director do seu
departamento, e a diferenza do seu salario e o salario do director do departamento.
Ordenar o resultado polo nome do departamento.*/

select de.depNombre as Departamento,
       em1.empNombre as Empleado,
       em1.empSalario as Salario_empleado,
       em2.empNombre as Nombre_Director,
       em2.empSalario as Salario_director,
       em2.empSalario-em1.empSalario as Diferenza
from empleado as em1 join departamento as de on em1.empDepartamento = de.depNumero
     join empleado as em2 on de.depDirector=em2.empNumero
where em1.empNumero <> em2.empNumero
order by de.depNombre;

```

1.3.2 Tarefa 2. Realizar consultas con datos de máis dunha táboa utilizando unha composición externa

A tarefa consiste en realizar as seguintes consultas utilizando unha composición externa.

Sobre a base de datos tendaBD

- Tarefa 2.1. Para todos os clientes con identificador inferior ou igual a 10, seleccionar os datos das vendas que se lle fixeron. Hai que mostrar para cada venda, o identificador do cliente, apelidos, nome e data de venda. Se a algún deses clientes non se lle fixo ningunha venda, deberá aparecer na lista co seu identificador, nome, apelidos, e o texto 'SEN COMPRAS' na columna da data da venda.
- Tarefa 2.2. Seleccionar os nomes das provincias nas que non temos ningún cliente.
- Tarefa 2.3. Seleccionar o código (*emp_id*), apelidos e nome de todos os empregados. Engadir unha columna na lista de selección, co alias Vendas, na que se mostre o literal 'Si' se o empregado fixo algunha venda, ou o literal 'Non' no caso de que aínda non fixera ningunha venda.

Solución

■ Tarefa 2.1

```
/* *****  
Obter os datos: identificador de cliente, apelidos e data de venda para todas as  
compras dos clientes con identificador inferior ou igual a 10. Se non existen compras,  
deberá aparecer na lista co texto 'SEN COMPRAS'.  
***** */  
  
select distinct cl.clt_id as Codigo_cliente,  
               cl.clt_apelidos as Apelidos,  
               cl.clt_nome as Nome,  
               ifnull(ve.ven_data,'SEN COMPRAS') as Data_compra  
from   clientes as cl left join vendas as ve on cl.clt_id=ve.ven_cliente  
where  cl.clt_id<=10;
```

■ Tarefa 2.2

```
/* *****  
Seleccionar os nomes das provincias nas que non temos ningún cliente.  
***** */  
  
select pro_nome  
from   provincias as pr  
       left join clientes as cl on pr.pro_id = left(trim(cl.clt_cp),2)  
where  clt_nome is null;  /* Vale calquera columna da táboa clientes
```

■ Tarefa 2.3

```
/* *****  
Tarefa 2.3. Seleccionar o código (emp_id), apelidos e nome de todos os empregados.  
Engadir unha columna na lista de selección, co alias Vendas, na que se mostre o literal  
'Si' se o empregado fixo algunha venda, e o literal 'Non' no caso de que aínda non  
fixera ningunha venda.  
***** */  
  
select distinct emp_id, emp_dni, emp_apelidos, emp_nome,  
               if(ven_id is null,'Non','Si') as Vendas  
from   empregados left join vendas on emp_id = ven_empregado;
```

1.3.3 Tarefa 3. Realizar consultas dunha táboa consigo mesma

A tarefa consiste en realizar as seguintes consultas compoñendo unha táboa consigo mesma.

Sobre a base de datos tendaBD

- Tarefa 3.1. Obter unha lista de todos os artigos que teñan un prezo de compra superior ao prezo de compra do artigo con código '0713242'.

Sobre a base de datos traballadores

- Tarefa 3.2. Mostrar o número, nome e salario de todos os empregados que teñen un salario maior que o salario do empregado número 180. Engadir na lista de selección unha columna para mostrar o salario do empregado número 180.

Solución

- Tarefa 3.1

```
/* *****  
Obter unha lista de todos os artigos que teñan un prezo de compra superior ao prezo  
de compra do artigo con código '0713242'.  
***** */
```

```
select ar1.art_codigo as Numero,  
       ar1.art_nome as Nome,  
       ar1.art_pc as Prezo_compra,  
       ar2.art_pc as Prezo_artigo_8  
from artigos as ar1 join artigos as ar2 on ar1.art_pc>ar2.art_pc  
where ar2.art_codigo='0713242';
```

- Tarefa 3.2

```
/* Mostrar o número, nome e salario de todos os empregados que teñen un salario maior  
que o do empregado número 180. Engadir na lista de selección unha columna para mostrar  
o salario do empregado número 180.*/
```

```
select em1.empNumero, em1.empNome, em1.empSalario, em2.empSalario as 'Empregado 180'  
from empregado as em1 join empregado as em2 on em1.empNumero != em2.empNumero  
where em2.empNumero=180  
       and em1.empSalario > em2.empSalario
```

1.3.4 Tarefa 4. Combinar o resultado de varias sentenzas SELECT para obter un conxunto de resultados único utilizando UNION

A tarefa consiste en realizar as seguintes consultas utilizando UNION.

Sobre a base de datos tendaBD

- Tarefa 4.1. Seleccionar todos os artigos negros, xunto cos artigos que pesan máis de 5000 gramos, escribindo dúas consultas, e empregando o operador de unión de consultas.
- Tarefa 4.2. Para facer un envío de cartas con información dunha nova campaña por correo postal, seleccionar apelidos, nome, enderezo, código postal e poboación de todos os clientes e de todos os empregados. Na lista hai que diferenciar se a persoa é cliente ou empregado. Ordenar o resultado por orden alfabético de apelidos e nome.

Solución

- Tarefa 4.1

```
/* *****  
Seleccionar todos os artigos negros, xunto cos artigos que pesan máis de 5000 gramos,  
escribindo dúas consultas, e empregando o operador de unión de consultas  
***** */
```

```
select art_codigo as Codigo,  
       art_nome as Nome,  
       art_peso as Peso,
```

```

    art_color as Color
from artigos
where art_peso>5000
union
select art_codigo as Codigo,
    art_nome as Nome,
    art_peso as Peso,
    art_color as Color
from artigos
where art_color="negro"
order by Codigo;

```

■ Tarefa 4.2

```

/*****
Para facer un envío de cartas con información dunha nova campaña por correo postal,
seleccionar apelidos, nome, enderezo, código postal e poboación de todos os clientes
e de todos os empregados. Na lista hai que diferenciar se a persoa é cliente ou
empregado. Ordenar o resultado por orden alfabético de apelidos e nome.
*****/

(select 'Cliente' as Persoa,
    cl.clt_apelidos as Apelidos,
    cl.clt_nome as Nome,
    cl.clt_enderezo as Enderezo,
    cl.clt_cp as CP,
    cl.clt_poboacion as Poboacion,
    pr.pro_nome
from clientes as cl
    join provincias as pr on left(trim(cl.clt_cp),2)=pr.pro_id
where clt_baixa is null)
union
(select 'Empregado' as Persoa,
    em.emp_apelidos as Apelidos,
    em.emp_nome as Nome,
    em.emp_enderezo as Enderezo,
    em.emp_cp as CP,
    em.emp_poboacion as Poboacion,
    pr.pro_nome as Provincia
from empregados as em
    join provincias as pr on left(trim(em.emp_cp),2)=pr.pro_id )
order by Apelidos, Nome;

```