

Bases de datos

Unidade 6: Manexo de datos

Índice

1.	A01. Manipulación de datos.....	3
1.1	Introdución.....	3
1.1.1	Obxectivos.....	3
1.1.2	Software.....	3
1.1.3	Bases de datos de traballo.....	4
1.1.3.1	Base de datos tendasbd.....	4
1.1.3.2	Base de datos practicas5.....	6
1.1.3.3	Base de datos traballadores.....	7
1.2	Actividade.....	8
1.2.1	Manipulación de datos con SQL.....	8
1.2.2	Ferramentas para modificar o contido das bases de datos.....	8
1.2.3	Sentenza INSERT.....	9
	Exemplos.....	11
	Opción VALUES.....	11
	Opción SET.....	12
	Opción SELECT.....	12
1.2.4	Sentenza Replace.....	13
1.2.5	Modo SQL do servidor MySQL e valores asignados ás columnas.....	13
1.2.6	Restricións de integridade e consistencia da información.....	14
1.2.6.1	Restricións de clave primaria (PRIMARY KEY).....	15
1.2.6.2	Restricións de unicidade (UNIQUE).....	15
1.2.6.3	Restricións de valor nulo (NOT NULL).....	15
1.2.6.4	Restricións de claves foráneas (FOREIGN KEY).....	15
1.2.6.5	Restricións DEFAULT.....	16
1.2.6.6	Restricións CHECK.....	17
1.2.6.7	Restricións dos tipos ENUM, e SET.....	17
1.2.7	Sentenza UPDATE.....	17
1.2.8	Sentenza DELETE.....	19
1.2.9	Borrado lóxico de filas dunha táboa.....	21
1.2.10	Uso de subconsultas nas sentenzas de edición de datos.....	22
1.2.11	Guións de sentenzas de edición de datos nas táboas.....	23
1.3	Tarefas.....	24
1.3.1	Tarefa 1. Escribir e probar sentenzas que fan cambios nos datos almacenados nas táboas.....	24
	Solución.....	26
1.3.2	Tarefa 2. Analizar e probar sentenzas, adoptando as medidas necesarias para manter a integridade e consistencia dos datos.....	28
	Solución.....	29

1. A01. Manipulación de datos

1.1 Introducción

1.1.1 Obxectivos

O obxectivo desta actividade é:

- Inserir, actualizar e borrar datos contidos nas bases de datos utilizando a linguaxe de manipulación de datos.

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 ilustrar as probas realizadas nesta actividade, mostraranse imaxes capturada dunha ou máis zonas do cliente gráfico Workbench. As capturas mostradas soen ser da zona de saída co número de filas afectadas pola sentenza executada ou as mensaxes de erro provocadas, e a zona de enreixado de resultados que co resultado dunha consulta.



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.



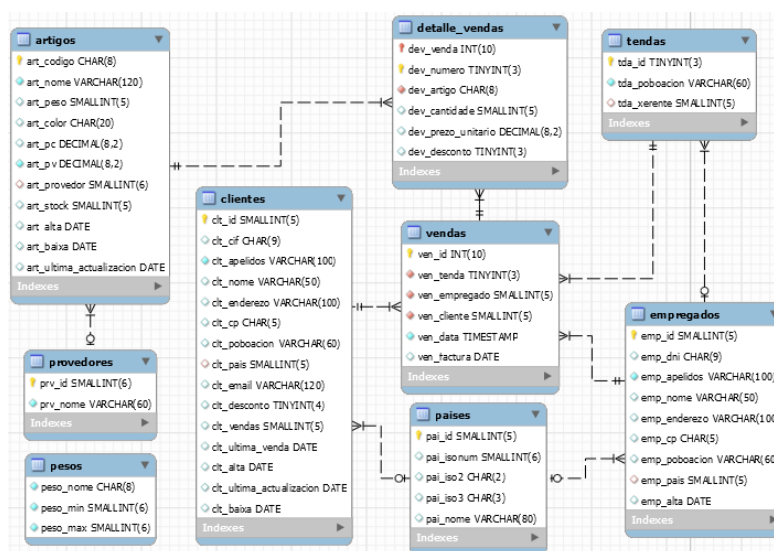
O material anexo a esta actividade inclúe unha guía básica de MySQL Workbench 6.3.

1.1.3 Bases de datos de traballo

As bases de datos *tendasBD*, *traballadores* e *practicass5* 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 *tendasbd*

A base de datos *tendasBD* 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 *empleados*

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_nombre	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_nombre	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*

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

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*

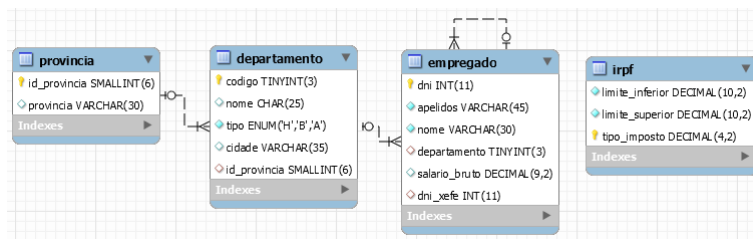
Nome columna	Tipo	Null	Clave		Observacións
dev_venta	int unsigned	Non	Primaria	Foránea	Identificador da venda á que corresponde a liña de detalle.
dev_numero	tinyint unsigned	Non			Número da liña de detalle dentro da venda.
dev_artigo	char(8)	Non	Foránea		Identificador do artigo vendido.
dev_cantidad	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.

■ Táboa *facturas*

Nome columna	Tipo	Null	Clave	Observacións
fac_numero	int unsigned	Non	Primaria	Identificador da factura. Autoincrementado.
fac_mes	tinyint	Non		Mes ao que corresponden as vendas facturadas.
fac_ano	smallint	Non		Ano ao que corresponden as vendas facturadas.
fac_data	date	Non		Data de emisión da factura.
fac_clt_cif	char(9)			Cif do cliente.
fac_clt_apelidos	varchar(100)	Non		Apelidos ou razón social do cliente.
fac_clt_nome	varchar(50)			Nome ou tipo de sociedade (SL, SA, ...) do cliente.
fac_clt_cp	char(5)			Código postal do cliente.
fac_clt_poboacion	varchar(60)			Poboación do cliente.
fac_clt_pais	smallint unsigned			Código do país segundo a táboa de países.
fac_importe	decimal(12,2)	Non		Importe total da factura.

1.1.3.2 Base de datos practicas5

A base de datos *practicas5* 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.



- Táboa *empregado*. 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_impuesto
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*, depEmpregados)

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 empregado

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 Manipulación de datos con SQL

SQL corresponde ao acrónimo de *Structured Query Language* (Linguaxe Estruturado de Consultas). Aínda que nun principio foi creado para facer consultas, utilízase para controlar todas as funcións que subministra un SXBDR aos seus usuarios, incluíndo todas as funcións propias das linguaxes deseñadas para o manexo de bases de datos: Linguaxe de Definición de Datos, Linguaxe de Manipulación de Datos, e Linguaxe de Control de Datos.

A linguaxe de manipulación de datos ou LMD (en inglés *Data Management Language* ou *DML*), permite realizar as operacións necesarias para manexar os datos almacenados nunha base de datos. Estas operacións consisten en inserir filas de datos (INSERT), modificar o contido das filas de datos (UPDATE), borrar filas de datos (DELETE), e consultar os datos contidos nas táboas da base de datos (SELECT).

1.2.2 Ferramentas para modificar o contido das bases de datos

Para modificar o contido das bases de datos, necesítase o código coas sentenzas SQL, un cliente conectado ao servidor que permita enviar o código a un servidor e un servidor que execute ese código.

A escritura de guións de sentenzas SQL pode facerse con calquera editor de texto plano.

As aplicacións cliente cunha interface gráfica (GUI) incorporan un editor especializado na escritura de guións e posibilita ademais de maneira sinxela e rápida a execución dos guións mediante un sistema de menús ou combinación de teclas.

As aplicacións cliente máis empregadas para conectase a un servidor MySQL e enviarlle as sentenzas SQL para que se executen, son as seguintes:

- **MySQL Workbench**

Cliente gráfico que de forma fácil e intuitiva permite establecer conexión cun servidor, mostrar información sobre a conexión, o servidor, bases de datos e táboas; escribir sentenzas SQL mediante un editor; enviar sentenzas ao servidor para que sexan executadas; e dispón de axuda nas tarefas de edición e execución de sentenzas. Para máis información sobre o seu funcionamento, débese utilizar o manual de referencia ou ben a guía básica de funcionamento de MySQL Workbench incluído no material auxiliar da actividade.

- **Consola modo texto mysql.exe**

Cliente en modo texto que permite establecer a conexión co servidor e escribir e executar sentenzas SQL.

- No caso de ter instalado Wampserver, hai que abrir o menú de Wampserver, e seleccionar 'MySQL console' no submenú de MySQL. Ábrese unha consola en modo texto que solicita introducir o contrasinal do usuario *root*. Unha vez introducido o contrasinal, móstrase o prompt *mysql>* que indica que está establecida a conexión co servidor e se poden empezar a escribir sentenzas para que o servidor as execute.
- No caso de non ter instalado Wampserver, hai que abrir unha consola de ordes do sistema, e executar a utilidade cliente *mysql.exe* que está no directorio bin que se atopa no directorio de instalación de MySQL (en Windows é normalmente, *c:\Program Files\MySQL\MySQL version*).

- **Aplicación web phpMyAdmin:**

Cliente web que de forma gráfica permite establecer a conexión co servidor e enviarlle sentenzas SQL para que sexan executadas.

- No caso de ter instalado Wampserver, hai que abrir o menú e seleccionar a opción *phpMyAdmin*.
- No caso de non ter instalado Wampserver, hai que abrir un navegador e teclear a url coa identificación do servidor web no que está instalado *phpMyAdmin*. Exemplos de url: *http://localhost/phpMyAdmin* (no caso que *phpMyAdmin* estea instalado nun servidor web no equipo local), *http://proveedor_web.com/phpMyAdmin* (no caso que *phpMyAmin* estea instalado no servidor *proveedor_web.com*).

1.2.3 Sentenza INSERT

A sentenza INSERT permite inserir novas filas nas táboas existentes. Existen varias opcións para realizar a inserción: os valores poden escribirse como un conxunto de valores pechados entre parénteses para cada fila nova, unha colección de expresións de asignación, ou como unha sentenza SELECT.

Opción especificando os valores a inserir de forma explícita con VALUE:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY]
[INTO] nome_táboa [(lista_columnas)]
{VALUES | VALUE} ({expresión | DEFAULT | NULL},...), (...),...
[ ON DUPLICATE KEY UPDATE col_name=expr [, col_name=expr] ... ]
```

Opción especificando os valores a inserir de forma explícita con SET:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY]
[INTO] nome_táboa
SET col_name={expresión | DEFAULT | NULL}, ...
[ ON DUPLICATE KEY UPDATE col_name=expr [, col_name=expr] ... ]
```

Opción inserindo filas con datos contidos noutras táboas:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY]
[INTO] nome_táboa [(lista_columnas)]
SELECT ...
[ ON DUPLICATE KEY UPDATE col_name=expr [, col_name=expr] ... ]
```

Consideracións sobre a sintaxe anterior:

- As partes opcionais LOW_PRIORITY, DELAYED e HIGH_PRIORITY permiten indicar o nivel de prioridade que ten a operación de inserción.

Cando se utiliza a opción LOW_PRIORITY, o servidor agarda a que non haxa clientes lendo na táboa para facer a inserción. Cando se utiliza a opción HIGH_PRIORITY, anúlase o efecto da opción *--low-priority-updates*, se estivera habilitada. Estas opcións afectan só a táboas con motores que utilizan bloqueo a nivel de táboa (MyISAM, MEMOMRY, MERGE).

Cando se utiliza a opción DELAYED, o servidor garda a sentenza nun buffer de memoria e libera ao cliente no caso de que a táboa estea bloqueada por outros clientes; cando a táboa queda libre, o servidor fai a inserción. Esta opción está en desuso a partir de MySQL 5.6.6 e vai ser eliminada en futuras versións.

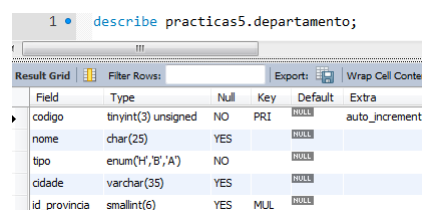
- nome_táboa* é o nome da táboa na que se van a inserir datos.
- lista_columnas* é a relación de columnas nas que se van a inserir datos, separadas por comas. Utilizarase cando se especifican os datos como unha lista de valores (VALUES), ou cando se collen os datos mediante unha consulta (SELECT).

As columnas que non figuran na lista toman o valor que teñan definido por defecto, ou o valor por defecto implícito se non teñen definido un valor por defecto. Os valores por defecto implícitos son 0 para os tipos numéricos, a cadea valeira (") para os tipos de cadeas de caracteres, e o valor "cero" para tipos de data e hora.

Este é o comportamento de MySQL cando non se está a executar no modo SQL estrito. Cando se activa o modo SQL estrito, prodúcese un erro se non se especifican valores explicitamente para todas as columnas que non teñen definido un valor por defecto.

As columnas de tipo AUTO_INCREMENT non deben aparecer na lista de columnas, e neste caso o servidor asígnalles o valor seguinte ao da última fila que se inseriu.

No caso de non poñer unha lista de columnas a continuación do nome da táboa, considérase que se van a inserir datos en todas as columnas, na orde en que se crearon. No caso de non utilizar un cliente gráfico e non recordar os nomes das columnas ou a orde en que se crearon, pódese utilizar a sentenza DESCRIBE para consultalo. Exemplo:



The screenshot shows a MySQL command window with the command `describe practicas5.departamento;` entered. The output is displayed in a table format with columns: Field, Type, Null, Key, Default, and Extra.

Field	Type	Null	Key	Default	Extra
codigo	tinyint(3) unsigned	NO	PRI	NULL	auto_increment
nome	char(25)	YES		NULL	
tipo	enum('H','B','A')	NO		NULL	
cidade	varchar(35)	YES		NULL	
id_provincia	smallint(6)	YES	MUL	NULL	

A lista de columnas utilízase no caso de que non se introduzan valores para todas as columnas, ou se escriban os valores en distinta orde á que teñen as columnas no esquema da táboa.

- Na opción 2, SET permitirá indicar explicitamente os nomes das columnas e os valores que van a tomar mediante expresións de asignación.
- Na opción 1, VALUES | VALUE permitirán indicar explicitamente a lista de valores que van a tomar as columnas da fila separados por comas e pechados entre parénteses. VALUE é un sinónimo de VALUES.

A expresión correspondente a cada columna debe devolver un valor do mesmo tipo que a columna. No caso e non ser do mesmo tipo, MySQL fai a conversión de tipos.

Nunha expresión pódese facer referencia a unha columna das que aparecen antes na lista de columnas. Tamén se poden utilizar as palabras reservadas NULL para facer referencia ao valor nulo, ou DEFAULT para facer referencia ao valor por defecto

Exemplo:

```
insert into nome_taboa (col1, col2, col3, col4) values (15, col1 * 2, null, default);
```

Unha excepción no uso de referencias a outras columnas son as columnas que teñen a propiedade AUTO_INCREMENT. Calquera referencia a esas columnas toma o valor 0.

O número de valores ten que coincidir co número de columnas relacionadas na lista de columnas.

MySQL permite inserir varias filas cunha única sentenza INSERT...VALUE, incluíndo varias listas de valores, cada unha pechada entre parénteses, e separadas por comas. Exemplo:

```
insert into nome_taboa (col1, col2, col3)
values (15, null, default), (25, 'Proba1', null), (55, 'Proba2', 'Proba3');
```

Esta sentenza insire tres filas na táboa.

- Na opción 3, as filas que se van a inserir e os valores que toman as columnas desas filas, obtéñense da execución dunha sentenza SELECT.
- Cando se utiliza a cláusula ON DUPLICATE KEY UPDATE e se fai a inserción dunha fila que pode provocar un valor duplicado nunha clave PRIMARY ou UNIQUE, faise unha actualización na fila que xa existe, modificando os valores das columnas que se indican na cláusula. Exemplo:

```
insert into taboa1 (a,b,c) values (1,2,3)
on duplicate key update c=c+1;
```

A sentenza modifica o contido da columna *c*, sumándolle unha unidade, no caso que exista unha fila co valor 1 na columna *a*, supoñendo que esta é a clave primaria.

Exemplos

Exemplos de sentenzas INSERT na base de datos *practicass5*.

Opción VALUES

- Sen utilizar lista de columnas

```
insert into empregado
values (31852963,'Varela Mendez','Luisa',4,29500,54528788);
```

38 19:11:47 insert into empregado values (31852963,'Varela Mendez','Luisa',4,29500,54528788) 1 row(s) affected

Asígnanse valores para todas as columnas da táboa. Os valores van ordenados segundo a definición do esquema da táboa.

- Utilizando lista de columnas

```
insert into departamento (nome, tipo, cidade, id_provincia)
```

```
values ('OficinaProba1','H','Ares',15);
```

39 19:11:51 insert into departamento (nome, tipo, cidade, id_provincia) values ('OficinaProba1','H','Ares',15) 1 row(s) affected

Asígnanse valores poñendo de forma explícita as columnas e os valores que se van a asignar. O primeiro valor da lista de valores ('OficinaProba1') asígnase á primeira columna da lista de columnas (*nome*), e así co resto. O número de valores ten que coincidir co número de columnas, e os tipos de datos teñen coincidir cos tipos de columnas.

Nesta sentenza non se incluíu na lista de columnas a columna *codigo* xa que é de tipo AUTO_INCREMENT, e por tanto asígnaselle automaticamente o valor seguinte ao que ten esa columna para a última fila que se inseriu. Consulta de datos:

codigo	nome	tipo	cidade	id_provincia
11	OficinaProba1	H	Ares	15

- Utilizando a sintaxe estendida de MySQL para poder inserir máis dunha fila cunha sentenza INSERT

```
insert into departamento (nome, tipo, cidade, id_provincia)
values ('OficinaProba2','H','Ferrol',15),
       ('OficinaProba3','A','Ourense',32),
       ('OficinaProba4','H','Allariz',32);
```

40 19:11:55 insert into departamento (nome, tipo, cidade, id_provincia) values ('OficinaProba2','H','Ferrol',15), ('OficinaProba3','A','Ourense',32), ('OficinaProba4','H','Allariz',32); 3 row(s) affected Registros: 3 Duplicados: 0 Peligros: 0

Fai a inserción de tres filas na táboa, e mostra a mensaxe indicando o número de filas inseridas. Consulta de datos:

codigo	nome	tipo	cidade	id_provincia
12	OficinaProba2	H	Ferrol	15
13	OficinaProba3	A	Ourense	32
14	OficinaProba4	H	Allariz	32

Opción SET

```
insert into departamento
set nome = 'OficinaProba5',
    tipo = 'A',
    cidade = 'Lugo',
    id_provincia = 27;
```

41 19:11:59 insert into departamento set nome = 'OficinaProba5', tipo = 'A', cidade = 'Lugo', id_provincia = 27 1 row(s) affected

Insire unha fila na táboa cos valores que se deron a cada columna. Consulta de datos:

codigo	nome	tipo	cidade	id_provincia
15	OficinaProba5	A	Lugo	27

A función LAST_INSERT_ID() permite obter o último valor que se almacenou na columna con propiedade AUTO_INCREMENT da táboa sobre a que se fixo a última operación de inserción. Por exemplo, se a última sentenza INSERT que se executou referíase á táboa *departamento* que ten a columna *codigo* coa propiedade AUTO_INCREMENT:

```
select last_insert_id();
```

Result Grid	Filter Rows:
LAST_INSERT_ID()	
15	

Opción SELECT

Esta inserción correspóndese coa terceira opción das expostas anteriormente e permite inserir novas filas nunha táboa copiando os datos que xa están gravados noutras táboas, tomando como entrada o resultado dunha consulta feita coa sentenza SELECT.

A sentenza SELECT debe ter na lista de selección o mesmo número de columnas, e o mesmo tipo de datos, que hai na lista de columnas, ou ben, o mesmo número de columnas que ten o esquema da táboa, se non se pon a lista de columnas. Exemplo:

```
/* Créase unha táboa temporal para gardar datos dos departamentos da provincia de Lugo
co mesmo esquema que a táboa departamento */
create temporary table departamento_lugo like departamento;
/* Faise a inserción dos datos dos departamentos de Lugo (contenén o valor 27 na
columna id_provincia). */
insert into departamento_lugo
select * from departamento where id_provincia = 27;
-- Consúltanse os datos da táboa temporal
select * from departamento_lugo;
```

Result Grid				Filter Rows:	Export
	codigo	nome	tipo	cidade	id_provincia
1		Central	H	Lugo	27
2		Oficina1	H	Monforte	27
6		Oficina5	A	Villalba	27
8		Oficina7	H	Lugo	27
15		OficinaProba5	A	Lugo	27

1.2.4 Senteza Replace

A sentenza REPLACE é unha alternativa para INSERT que só se diferencia en que se existe algunha fila anterior co mesmo valor para unha clave primaria (PRIMARY KEY) ou única (UNIQUE), elimínase a fila que xa existía con ese valor na clave, e insírese no seu lugar a fila cos novos valores. Exemplo:

```
insert into departamento (codigo,nome,tipo,cidade,id_provincia)
values (16,'OficinaProba10','H','Carballo',15);
```

52 20:45:40 insert into departamento (codigo,nome,tipo,cidade,id_provincia) values (16,'OficinaProba10',... Error Code: 1062. Entrada duplicada '16' para la clave 'PRIMARY'

Xa existe un departamento co valor 16 na clave primaria (*codigo*), o que fai que se mostre unha mensaxe de erro e a fila non pode ser inserida. Execútase a sentenza REPLACE:

```
replace into departamento (codigo,nome,tipo,cidade,id_provincia)
values (16,'OficinaProba10','H','Carballo',15);
```

50 20:37:54 replace into departamento (codigo,nome,tipo,cidade,id_provincia) values (16,'OficinaProba10','H','Carballo',15) 2 row(s) affected

A mensaxe informa que foron afectadas dúas filas. Unha é a que existía co mesmo valor na clave primaria e outra é a nova fila que a substitúe. Consulta de datos:

Result Grid			Filter Rows:		Edit:
	codigo	nome	tipo	cidade	id_provincia
	16	OficinaProba10	H	Carballo	15

1.2.5 Modo SQL do servidor MySQL e valores asignados ás columnas

O comportamento do servidor MySQL cando se producen erros asociados aos tipos de datos dependerá de se está traballando en modo estrito ou non. No caso de non estar

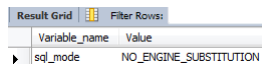
activado o modo estrito, os datos que non se axustan á definición da columna son engadidos á táboa pero se mostra unha mensaxe de alerta (*warning*). No caso de estar activado o modo estrito, xa non se permite que os datos sexan engadidos.

'MySQL server puede operar en distintos modos SQL, y puede aplicar estos modos de forma distinta a diferentes clientes. Esto permite que cada aplicación ajuste el modo de operación del servidor a sus propios requerimientos.'

*Los modos definen qué sintaxis SQL debe soportar MySQL y que clase de chequeos de validación de datos debe realizar. Esto hace más fácil de usar MySQL en distintos entornos y usar MySQL junto con otros servidores de bases de datos.'*²

Para ver o modo SQL activo no servidor, pódese ver o contido da variable de sistema `sql_mode`, executando unha das sentenzas:

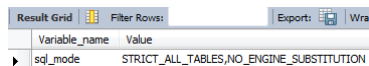
```
select @@sql_mode;    # ou ben:
show variables like 'sql_mode';
```



Variable_name	Value
sql_mode	NO_ENGINE_SUBSTITUTION

Para cambiarlle o valor á variable `sql_mode`, pódese executar unha sentenza de asignación SET ou cambiar o valor da variable no ficheiro de configuración de MySQL (`mi.ini` ou `my.cnf`). Exemplo:

```
set sql_mode = concat('STRICT_ALL_TABLES, ', @@sql_mode);
show variables like 'sql_mode';
```



Variable_name	Value
sql_mode	STRICT_ALL_TABLES,NO_ENGINE_SUBSTITUTION

Cando está activado o modo SQL estrito e se asignan a unha columna valores fóra do rango permitido, móstrase unha mensaxe de erro e abórtase a inserción. Exemplo de inserción cando está activado o modo SQL estrito:

```
insert into departamento (nome,tipo,cidade,id_provincia)
values ('ProbaErro','Z','Santiago',15);
```

8 19:02:22 insert into departamento (nome,tipo,cidade,id_provincia) values ('ProbaErro','Z','Santiago',15) Error Code: 1265. Datos truncados para columna tipo' en la línea 1

Algunhas consideracións sobre os valores asignados ás columnas que poden provocar mensaxes de advertencia (*warning*) cando non está activado o modo SQL estrito:

- Cando se asigna un valor NULL a unha columna declarada NOT NULL, prodúcese unha mensaxe de advertencia (*warning*) e gárdase na columna o valor por defecto implícito (0 para columnas de tipo numérico, unha cadea baleira " para columnas de tipo cadea, ou o valor 0 para columnas tipo data e hora).
- Cando se asigna un valor fora de rango a unha columna de tipo numérico, o valor trúncase.
- Cando se asigna un valor tipo cadea a unha columna de tipo numérico, cópanse os díxitos numéricos se os tivera e non se teñen en conta o resto de caracteres, no caso de non ter ningún dígito numérico gárdase o valor 0. Exemplo: o valor '25.23 euros' almacénase como 25.23 nunha columna de tipo numérico e non se ten en conta o texto euros.
- Cando se asigna unha cadea que excede do tamaño dunha columna tipo cadea, trúncase ata a lonxitude máxima.

- Cando se asigna un valor non válido a unha columna relacionada co tempo (data, hora ou data e hora), gárdase o valor cero.
- Cando se asigna un valor diferente dos permitidos a unha columna tipo ENUM, gárdase na columna o valor por defecto implícito. Exemplo:

```
insert into departamento (nome, tipo, cidade, id_provincia)
values ('ProbaErro', 'Z', 'Santiago', 15);
```

⚠ 48 20:18:34 insert into departamento (nome, tipo, cidade, id_provincia) values ('ProbaErro', 'Z', 'Santiago', 15) 1 row(s) affected, 1 warning(s): 1265 Datos truncados para colu...

A columna *codigo* foi definida como ENUM('H','B','A'). O valor 'Z' non é un valor dos permitidos, pero como o servidor non está en modo SQL estrito, a fila insírese na táboa e nesa columna gárdase o valor por defecto implícito (''), unha cadea baleira. Consulta de datos:

codigo	nome	tipo	cidade	id_provincia
16	ProbaErro		Santiago	15

1.2.6 Restricións de integridade e consistencia da información

Os datos almacenados nas columnas dunha táboa dunha base de datos relacional deberían cumprir coas:

- Restricións de clave primaria (PRIMARY KEY).
- Restricións de unicidade (UNIQUE)
- Restricións de valor nulo (NOT NULL).
- Restricións de claves foráneas (FOREIGN KEY).
- Restricións DEFAULT.
- Restricións CHECK.
- Restricións dos tipos ENUM, e SET.

Cando se executan sentenzas que fan modificacións nos datos das bases de datos hai que asegurarse que cumpren estas restricións, para poder seguir mantendo a integridade dos datos. Se as restricións establecéronse correctamente no momento do deseño e na creación das bases de datos, o servidor fará ese traballo por nós, avisando cando algunha sentenza vulnere algunha das restricións impostas. De aí a importancia do deseño correcto nunha BD relacional.

1.2.6.1 Restricións de clave primaria (PRIMARY KEY)

As restricións que debe cumprir a clave primaria son:

- As columnas que forman parte da clave primaria non poden tomar o valor NULL.
- Os valores que toman as columnas que forman a clave primaria teñen que ser únicos nunha táboa. Isto significa que para cada fila, a clave primaria ten que tomar un valor diferente ao resto das filas.

1.2.6.2 Restricións de unicidade (UNIQUE)

As restricións de unicidade refírense a que as columnas definidas como de tipo UNIQUE teñen que tomar valores únicos para cada fila. Non pode haber dúas filas que teñan o mesmo valor na columna.

1.2.6.3 Restricións de valor nulo (NOT NULL)

As restricións de valor nulo indican que as columnas ás que se lle asigna a propiedade NOT NULL, teñen que ter sempre un valor válido non admitindo o valor descoñecido (NULL).

1.2.6.4 Restricións de claves foráneas (FOREIGN KEY)

Unha columna definida como clave foránea só pode ter como valor algún dos da clave primaria á que fai referencia ou o valor NULL se é que ese valor está permitido.

As táboas transaccionais de MySQL (por exemplo, as que utilizan o motor de almacenamento InnoDB) son as únicas táboas de MySQL que soportan restricións de claves foráneas. Tanto a táboa que contén a restrición de clave foránea como a táboa á que se fai referencia, teñen que ser táboas transaccionais e non poden ser táboas temporais.

Sintaxe para definir unha restrición de clave foránea en táboas transaccionais:

```
[CONSTRAINT nome_restrición] FOREIGN KEY (nome_columna [,nome_columna] ...)
REFERENCES nome_de_táboa (nome_índice [,nome_índice] ...)
[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
```

- Se existe *nome_restrición*, debe ser único na base de datos; se non se subministra, InnoDB crea o nome automaticamente.
 - O servidor non permitirá realizar ningunha INSERT ou UPDATE que intente asignar un valor a unha columna definida como clave foránea que non coincida con algún valor da clave primaria da táboa á que fai referencia, ou tome o valor NULL no caso de que estea permitido este valor para esa columna.
 - Cando se intenta realizar unha operación UPDATE ou DELETE sobre unha fila da táboa pai (a que figura en REFERENCES) e existen táboas que conteñen claves foráneas que sinalan a esa fila, terase en conta a acción especificada nos apartados ON DELETE e ON UPDATE da cláusula FOREIGN KEY. As accións permitidas en MySQL son:
 - CASCADE.
Borra ou actualiza a fila na táboa pai e automaticamente borra ou actualiza as filas con claves foráneas que fan referencia a esa fila.
 - SET NULL
Borra ou actualiza a fila na táboa pai e garda o valor NULL na columna que é clave foránea e fai referencia a esa fila. Para poder empregar esta opción a columna que é clave foránea debe permitir almacenar o valor NULL, é dicir, non ter NOT NULL na súa definición.
 - NO ACTION
Non é estándar ANSI SQL-92 e significa que non está permitido borrar ou modificar unha fila na táboa pai, se hai algunha fila que teña unha clave foránea que faga referencia a ela.
 - RESTRICT
En MySQL, NO ACTION e RESTRICT son equivalentes.
 - SET DEFAULT
Pon na clave foránea o valor definido por defecto para esa columna. Non está implementada en MySQL.
- Cando se intenta facer unha operación non permitida polas restricións de integridade referencial, prodúcese un erro e móstrase unha mensaxe de erro do tipo:


```
Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`practicas5`.`empleado`, CONSTRAINT
`fk_empleado_empleado1` FOREIGN KEY (`dni_xefe`) REFERENCES `empleado` (`dni`) ON DELETE NO ACTION ON UPDATE NO
ACTION)
```

Na mensaxe de erro infórmase de que se intenta modificar ou borrar una fila pai (que hai filas doutras táboas que fan referencia a ela) e mostra información da definición de clave foránea que provoca o erro. Na mensaxe anterior, a fila que se intenta borrar ou modificar corresponde a un empregado que ten un dni que coincide co valor que toma en algunha fila a columna *dni_xefe*, definida como clave foránea e que ten asociada a operación NO ACTION para o caso de borrado e modificación.

- Para mellorar o rendemento do funcionamento das claves foráneas e das restricións de integridade asociado a elas, as columnas definidas como clave foránea e a clave primaria á que fai referencia deben ter asociado un índice. A partir da versión 5.0 de MySQL, cando se crea unha clave foránea créase o índice de forma automática.
- Pódese desactivar a verificación de restricións de clave foránea de forma temporal, mediante a variable `FOREIGN_KEY_CHECKS`, por exemplo para facer operacións de mantemento de táboas, como pode ser a recuperación de copias de seguridade. É posible establecer o valor desta variable dende a liña de comandos do cliente MySQL ou engadila nun script de sentenzas SQL. Exemplo:

```
set foreign_key_checks = 0 # desactiva a verificación de restricións de clave foráneas
set foreign_key_checks = 1 # activa a verificación de restricións de clave foráneas
```

1.2.6.5 Restricións DEFAULT

Pódese asignar un valor por defecto a unha columna coa cláusula DEFAULT. Ese será o valor que toma a columna no caso de non asignarlle un valor de maneira explícita.

1.2.6.6 Restricións CHECK

A restrición CHECK asocia unha restrición de valor a unha columna, poñendo entre parénteses unha expresión coas condicións que teñen que cumprir os datos almacenados nesa columna. Cando se insiren ou modifican datos nunha columna co atributo CHECK, compróbase que se verifiquen as condicións establecidas para esa columna.

MySQL non implementa esta restrición. Permite escribir a instrución sen dar erro de sintaxe, pero non fai as comprobacións de valor.

1.2.6.7 Restricións dos tipos ENUM, e SET

Estes tipos de datos son propios de MySQL, e permiten asociar unha restrición de valor a unha columna poñendo entre parénteses o conxunto de valores válidos para a columna. As columnas de tipo ENUM só poden tomar un valor do conxunto de valores permitidos, e as de tipo SET poden tomar un ou máis valores do conxunto de valores permitidos.

Cando se insiren ou modifican datos nunha columna do tipo ENUM ou SET, hai que comprobar que os valores que toma a columna estean na lista de valores permitidos.

1.2.7 Sentenza UPDATE

Permite modificar os datos contidos nas táboas. Existen varias opcións de sintaxe.

Opción para modificar datos dunha soa táboa:

```
UPDATE [LOW_PRIORITY] nome_táboa
SET nome_columna = {expresión | DEFAULT | NULL}
[, nome_columna = {expresión | DEFAULT | NULL}][, ...]
[WHERE condición]
[ORDER BY expresión]
```

```
[LIMIT número_filas]
```

Opción para modificar datos de varias táboas:

```
UPDATE [LOW_PRIORITY] referencia_táboas
SET nome_columna = {expresión | DEFAULT | NULL}
    [, nome_columna = {expresión | DEFAULT | NULL}] [, ...]
[WHERE condición]
```

- A opción `LOW_PRIORITY` permite indicar o nivel de prioridade que ten a operación de inserción. Con esta opción, o servidor espera que non haxa clientes lendo na táboa para facer a modificación. Só afecta a táboas con motores que utilizan bloqueo a nivel de táboa (MyISAM, MEMOMRY, MERGE).
- Cando se utiliza a opción para actualizar varias táboas, a sentenza `UPDATE` actualiza as filas das táboas nomeadas en *referencia_táboas* que cumpren as condicións establecidas na cláusula `WHERE`. Cada xogo de filas actualízase só unha vez, aínda que cumpran varias veces as condicións.
- A cláusula `SET` indica as columnas que se van a modificar e os valores novos que van a recibir. Para os novos valores que se asignan ás columnas nunha operación de modificación aplícase o visto no apartado '2.2.3.2. Modo SQL do servidor MySQL e valores asignados ás columnas'.
- A cláusula `WHERE` é opcional e especifica as filas que deben actualizarse. Moi importante: se non se escribe a cláusula `WHERE`, actualízanse todas as filas.
- A cláusula `ORDER BY` é opcional e indica a orde na que se actualizan as filas. Non se pode utilizar na actualización de múltiples táboas.

Pode ser útil en certas situacións como por exemplo para actualizar unha columna que non admite valores duplicados, engadíndolle unha unidade ao valor que ten a columna.

```
update táboa
set columna = columna + 1;
```

Supoñendo que a columna ten os valores 1,2,3,4 e 5. Cando se actualiza a primeira fila asígnaselle á columna o valor 2, que é o valor que ten a segunda fila nesa columna, polo que se produce un erro de '*clave duplicada*' e non se pode facer a actualización. Utilizando a cláusula `ORDER BY`:

```
update táboa
set columna = columna + 1
order by columna desc;
```

Neste caso a primeira fila que se actualiza é a que ten o valor maior, neste caso o 5. Despois da actualización a columna toma o valor 6 que non existe en ningunha outra fila, e desta maneira, soluciónase o problema de claves duplicadas.

- A cláusula `LIMIT` establece o número de filas a actualizar. Non se pode utilizar na actualización de múltiples táboas.

Exemplos de sentenzas UPDATE na base de datos practicas5:

- Actualizar todas as filas dunha táboa

Aumentar nun 5% os valores da columna *tipo_imposto* da táboa *irpf*.

```
update irpf
set tipo_imposto=round(tipo_imposto*1.05,2);
```

86 19:31:31 update irpf set tipo_imposto=round(tipo_imposto*1.05,2)

7 row(s) affected Líneas correspondientes: 7 Cambiadas: 7 Avisos: 0

A mensaxe informa que foron modificadas as 7 filas que ten a táboa.

- Actualizar as filas que cumpren unha condición

Aumentar nun 3% os valores das columnas *limite_inferior* e *limite_superior* da

táboa *irpf*, para todas as filas excepto a primeira fila na que *limite_inferior* ten o valor 0.

```
update irpf
set     limite_inferior=round(limite_inferior*1.03,2),
        limite_superior=round(limite_superior *1.03,2)
where  limite_inferior > 0 ;
```

88 20:01:48 update irpf set limite_inferior=round(limite_inferior*1.03,2), limite_superior=round(limite... 6 row(s) affected, 1 warning(s): 1264 Out of range value for column 'limite...

A mensaxe informa que foron modificadas 6 filas e mostra un aviso advertindo que o valor para a columna *limite_superior* da fila 7 da táboa ten un valor fora de rango, e como non está activado o modo SQL estrito, trúncase o valor. O texto completo da mensaxe é o seguinte:

6 row(s) affected, 1 warning(s): 1264 Out of range value for column 'limite_superior' at row 7 Líneas correspondientes: 6 Cambiadas: 6 Avisos: 1

– Actualizar un número de filas limitado

Reducir un 5% o salario bruto dos empregados que teñen os 2 salarios máis altos, do departamento número 4.

```
update empleado
set salario_bruto=round(salario_bruto*0.95,2)
where departamento=4
order by salario_bruto desc
limit 2 ;
```

91 20:27:16 update empleado set salario_bruto=round(salario_bruto*0.95,2) where departamen... 2 row(s) affected Líneas correspondientes: 2 Cambiadas: 2 Avisos: 0

A mensaxe informa que foron modificadas 2 filas.

– Actualizar columnas de máis dunha táboa

Cambiar o tipo de departamento ao departamento número 1 que pasa a ser de tipo 'A', e ao mesmo tempo, aumentarlle un 5% ao salario bruto aos empregados dese departamento.

```
update empleado as em, departamento as de
set     de.tipo='A',
        em.salario_bruto=round(salario_bruto*1.05,2)
where  em.departamento=de.codigo
        and em.departamento=1;
```

110 20:55:19 update empleado as em, departamento as de set de.tipo='A', em.salario_bruto=rou... 4 row(s) affected Líneas correspondientes: 4 Cambiadas: 4 Avisos: 0

Na cláusula WHERE faise a combinación das táboas relacionando cada fila da táboa *empleado* coa fila correspondente da táboa *departamento*. Se non se pon esta condición, a modificación faríase sobre a táboa formada polo produto cartesiano das dúas táboas. A condición inclúe tamén o resto de condicións que teñen que cumprir as filas que hai que modificar; neste caso, selecciónanse só os empregados que corresponden ao departamento número 1.

A mensaxe informa que foron modificadas 4 filas, 3 delas corresponden ás filas da táboa *empleado* correspondentes aos tres empregados do departamento número 1, e a outra corresponde á fila da táboa *departamento* correspondente ao departamento número 1.

1.2.8 Sentenza DELETE

Permite eliminar filas das táboas. Existen varias opcións de sintaxe.

Opción 1 para borrar filas dunha soa táboa:

```
DELETE [LOW_PRIORITY] FROM nome_táboa
[WHERE condición]
[ORDER BY expresión]
[LIMIT número_filas]
```

Opción 2 para borrar filas de múltiples táboas:

```
DELETE [LOW_PRIORITY] [nome_táboa[.*]] [,nome_táboa[.*]] ...
{FROM | USING} referencia_táboas
[WHERE condición]
```

Opción 3 para borrar filas de múltiples táboas:

```
DELETE [LOW_PRIORITY]
FROM nome_táboa[.*]] [,nome_táboa[.*]] ...
USING referencia_táboas
[WHERE condición]
```

- A opción `LOW_PRIORITY` permite indicar o nivel de prioridade que ten a operación de borrado. Con esta opción o servidor espera que non haxa clientes lendo na táboa para facer a operación de borrado. Só afecta a táboas con motores que utilizan bloqueo a nivel de táboa (MyISAM, MEMOMRY, MERGE).
- Cando se utilizan as opcións 2 ou 3 da sintaxe, para borrar filas de varias táboas, a sentenza `DELETE` borra as filas das táboas nomeadas en *referencia_táboas* que cumpren as condicións establecidas na cláusula `WHERE`.
- Na cláusula `FROM` noméanse as táboas nas que se van a borrar filas.
 - Cando se van a borrar filas de varias táboas utilizando a opción 2, establécense as relacións entre as táboas mediante a sentenza `JOIN`. Exemplo:

```
DELETE t1, t2 FROM t1 INNER JOIN t2 INNER JOIN t3
WHERE t1.id=t2.id AND t2.id=t3.id;
```
 - Cando se van a borrar filas de varias táboas utilizando a opción 3, na cláusula `USING` establécense as relacións entre as táboas mediante a sentenza `JOIN`. Exemplo:

```
DELETE FROM t1, t2 USING t1 INNER JOIN t2 INNER JOIN t3
WHERE t1.id=t2.id AND t2.id=t3.id;
```
- A cláusula `WHERE` é opcional e especifica que filas deben borrarse.

Cando se utilizan as opcións 2 ou 3 para borrar filas en varias táboas, teñen que poñerse en primeiro lugar as condicións de enlace entre as táboas e despois o resto de condicións.

Moi importante: se non se escribe a cláusula `WHERE`, bórranse todas as filas da táboas nomeadas.
- A cláusula `ORDER BY` é opcional e indica a orde na que se borran as filas. Non se pode utilizar no borrado en múltiples táboas.
- A cláusula `LIMIT` establece o número de filas a borrar. Non se pode utilizar na actualización de múltiples táboas.

Exemplos de sentenzas `DELETE` na base de datos *practicass5*:

- Borrar as filas dunha táboa que cumpren unha condición.

Borrar os datos do empregado que ten o DNI número 12549563.

```
delete from empregado
where dni = 12549563;
```

A mensaxe informa que foi borrada unha fila.

- Borrar un número limitado de filas dunha táboa

Borrar os datos de empregado do departamento 1 que ten o salario máis baixo.

```
delete from empregado
where departamento=1
order by salario_bruto
limit 1;
```

- Borrar filas de máis dunha táboa

Borrar os departamentos que están na provincia de Madrid (*id_provincia* 28) e todos os empregados que pertencen a el.

```
delete em, de
from empregado as em straight_join departamento as de
where em.departamento=de.codigo
and de.id_provincia=28;
```

A mensaxe informa que foron borradas dúas filas. Unha corresponde ao departamento número 10 que tiña o valor 28 na columna *id_provincia*, e a outra corresponde ao empregado que traballaba nese departamento.

Na cláusula FROM noméanse as táboas que interveñen na operación de borrado e as relacións que hai entre elas. Utilízase a combinación STRAIGHT_JOIN para forzar ao optimizador a que use as táboas na orde en que están escritas, desta maneira faise primeiro o borrado das filas na táboa *empregado* e despois as da táboa *departamento*. Se non se utiliza STRAIGHT_JOIN, pode que o optimizador empece a borrar pola táboa departamento, e nese caso prodúcese un erro debido ás restricións de clave foránea xa que non está permitido borrar un departamento se hai empregados relacionados con el.

Na cláusula WHERE establécese a condición de enlace entre as táboas e o resto de condicións. Neste caso o resto das condicións son que o departamento pertenza á provincia de Madrid (o valor da columna *id_provincia* tome o valor 28).

Pódese escribir esta mesma sentenza de borrado utilizando a opción.

```
delete
from em, de
using empregado as em straight_join departamento as de
where em.departamento=de.codigo
and de.id_provincia=28;
```

No caso de non utilizar *straight_join*, o optimizador pode escoller empezar a borrar pola táboa departamento, e nese caso produciríase un erro debido ás restricións de clave foránea. Código para facer a proba:

```
delete
from em, de
using empregado as em join departamento as de
where em.departamento=de.codigo
and de.id_provincia=28;
```

A mensaxe de erro advirte que non se pode borrar unha fila pai da táboa *departamento* porque hai algunha fila na táboa *empregado* que fai referencia a ela, e a operación DELETE ten asociada a acción NO ACTION para esa clave foránea. O texto completo da mensaxe é o seguinte:

Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('practicas5`.`empregado`, CONSTRAINT `fk_empregado_departamento` FOREIGN KEY (`departamento`) REFERENCES `departamento` (`codigo`) ON DELETE NO ACTION ON UPDATE NO ACTION)

1.2.9 Borrado lóxico de filas dunha táboa

A sentenza DELETE borra fisicamente as filas das táboas. Nas bases de datos, hai outra alternativa para 'eliminar' filas das táboas que se coñece como 'borrado lóxico' e consiste en marcar a fila poñendo nunha columna un dato que a identifica como non dispoñible. Por exemplo, pode ser unha columna que conteña unha data de baixa, ou unha columna que tome un valor lóxico (verdadeiro ou falso, 0 ou 1) que indique se a fila está eliminada ou non. Exemplos:

- Borrado físico:

```
delete from cliente
where id = 9563;
```

Esta sentenza elimina fisicamente a fila coa información do cliente que ten o identificador 9563. Se a táboa utiliza un motor transaccional, e hai filas doutras táboas que fan referencia a esa fila, aplicaranse as restricións de comportamento asociadas ás claves foráneas, restrinxindo a operación ou facendo un borrado en cascada.

- Borrado lóxico:

```
update cliente
set data_baixa=curdate()
where id = 9563;
```

Ou ben:

```
update cliente
set eliminada=1
where id = 9563;
```

Este tipo de borrado pode ser moi útil en certas ocasións, por exemplo, se o cliente volve a facer unha compra, evitamos borrar fisicamente os datos do cliente e ter que volver a inserilos cando se quere dar de alta de novo, ademais de poder conservar as referencias a ese cliente no resto de táboas.

O problema que produce este tipo de borrado é que en cada consulta que se fai na que se utilice esa táboa hai que comprobar o estado desa columna e comprobar as restricións de integridade referencial. De non facelo, pódese producir unha inconsistencia de datos, por exemplo porque pode haber filas en *vendas* que fan referencia a clientes 'eliminados'.³

Tamén hai que ter en conta que certas lexislacións, como é o caso da Lei Orgánica de Protección de datos (LOPD) en España, poden obrigar a facer borrado físico de certos tipos de datos.

Para xestionar as baixas lóxicas é recomendable utilizar disparadores (*triggers*), vistas ou procedementos almacenados, que se verán máis adiante.


```

select @salario_minimo:=min(salario_bruto)
from empregado
where departamento = 2
/* operación de borrado*/
delete from empregado
where salario_bruto = @salario_minimo
and departamento=2;

```

```

261 13:24:57 set @salario_minimo = (select min(salario_bruto) from empregado where departamento = 2) 0 row(s) affected
262 13:24:57 delete from empregado where salario_bruto = @salario_minimo and departamento=2 1 row(s) affected

```

Esta solución é a máis completa e borraría todos os empregados que teñan un salario igual ao salario máis baixo.

1.2.11 Guións de sentenzas de edición de datos nas táboas

As sentenzas de edición pódense executar de forma directa escribindo a sentenza e enviándoa ao servidor para que a execute, ou escribindo e gardando nun ficheiro de ordenes (*script*) para que se executen todas xuntas no momento que se envía o *script* ao servidor.

Un exemplo de utilización de guións de sentenzas de edición pode ser para borrar fisicamente as filas marcadas para eliminar en todas as táboas ao finalizar o ano. As sentenzas poderían ser as seguintes:

```

delete from cliente
where eliminado = 1;
delete from artigo
where eliminado = 1;
delete from empregado
where eliminado = 1;
.... # aquí irían as sentenzas para borrar as fila eliminadas no resto das táboas

```

A práctica habitual para este exemplo consiste en gardar todas esas sentenzas nun ficheiro de texto coa extensión *.sql* e cando chega o final de cada ano, enviar o ficheiro ao servidor para que execute as sentenzas que contén. En Workbench, a execución faise abrindo o contido do *script* e dando a orde de executar. Dende o cliente en modo texto de MySQL, faise utilizando o comando *source* (ou *\.*):

```
mysql> source c:\administrador\scripts\eliminar_marcados.sql
```

Outra posibilidade para este exemplo consistiría en automatizar a tarefa anterior, creando un evento no servidor que se execute todos os anos o día 1 de xaneiro. Na unidade 7 'Programación de bases de datos' verase a maneira de automatizar tarefas mediante eventos.

Outro exemplo de utilización de guións de sentenzas de edición pode ser para facer unha carga masiva de datos, xa que é posible crear un ficheiro de ordenes SQL (*script*), que conteña un grupo de instrucións INSERT para engadir filas. Este tipo de scripts son os que crean algunhas utilidades de *backups*, como por exemplo *mysqldump* de MySQL (que se verá na unidade 8 de Administración de bases de datos), ou están incorporados na maioría dos clientes gráficos.



Tarefa 1. Escribir e probar sentenzas que fan cambios nos datos almacenados nas táboas.



Tarefa 2. Analizar e probar sentenzas, adoptando as medidas necesarias para manter a integridade e consistencia dos datos.

1.3 Tarefas

As tarefas propostas son as seguintes:

- Tarefa 1. Escribir e probar sentenzas que fan cambios nos datos almacenados nas táboas.
- Tarefa 2. Analizar e probar sentenzas, adoptando as medidas necesarias para manter a integridade e consistencia dos datos.

1.3.1 Tarefa 1. Escribir e probar sentenzas que fan cambios nos datos almacenados nas táboas

A tarefa consiste en realizar as seguintes operacións de manipulación de datos utilizando sentenzas INSERT, REPLACE, UPDATE e DELETE.

Sobre a base de datos *traballadores*

- Tarefa 1.1. Inserir unha fila na táboa *centros* cos seguintes datos:

Nome columna	Valor ou observacións
cenNumero	40
cenNome	FRANQUICIA LUGO
cenEnderezo	C/ PROGRESO, 8 - LUGO

- Tarefa 1.2. Inserir na táboa *empregado*, nunha única sentenza INSERT, as filas cos datos do seguintes empregados.
 - Primeiro empregado:

Nome columna	Valor ou observacións
empNome	BARCIA, ANGELES
empNumero	750
empDepartamento	110
empExtension	25
empDataNacemento	12 de febreiro de 1990
empDataIngreso	Hoxe
empSalario	825
empComision	50
empFillos	1

- Segundo empregado:

Nome columna	Valor ou observacións
empNome	MENDEZ, RICARDO
empNumero	751
empDepartamento	110
empExtension	25
empDataNacemento	22 outubro de 1985
empDataIngreso	Fai 15 días
empSalario	900
empComision	Descoñecida polo momento
empFillos	0

- Terceiro empregado:

Nome columna	Valor ou observacións
empNome	BERNARDEZ, LUCIA
empNumero	752
empDepartamento	120
empExtension	45
empDataNacemento	9 de maio de 1992
empDataIngreso	Descoñecida. Pendente de facer contrato
empSalario	1200
empComision	150
empFillos	2

– Cuarto empregado:

Nome columna	Valor ou observacións
empNome	VALIN, EVA
empNumero	753
empDepartamento	100
empExtension	200
empDataNacemento	5 de novembro de 1980
empDataIngreso	Hoxe
empSalario	1000
empComision	300
empFillos	1

- Tarefa 1.3. Acórdase aumentar o salario a todos os empregados un 5% e a comisión un 6,5% como consecuencia da revisión do convenio. Facer as modificacións correspondentes na base de datos.
- Tarefa 1.4. Cambiarlle a data de ingreso na empresa do empregado número 752, asignándolle a data que corresponde ao día 1 do mes seguinte ao mes actual.
- Tarefa 1.5. Aumentar un 2% o salario a todos os empregados do departamento 120.
- Tarefa 1.6. Aumentarlle 50 euros á comisión de todos os empregados que traballen nun departamento que dependa do centro de traballo que ten por nome 'SEDE CENTRAL'.
- Tarefa 1.7. Reducir nun 10% o presuposto anual do departamento que teña o presuposto máis alto na actualidade.
- Tarefa 1.8. Escribir un *script* para facer todos os seguintes cambios nos presupostos dos departamentos pero sen modificar o presuposto total:
 - Traspasar 20000 do presuposto do departamento de 'PERSOAL' ao departamento de PROCESO DE DATOS.
 - Reducir en 10000 o presuposto do departamento de 'SECTOR INDUSTRIAL', dos que 4000 se traspasan ao departamento de 'ORGANIZACION' e 6000 ao departamento de 'DIRECCION COMERCIAL'.
- Tarefa 1.9. Borra o empregado co número 380.
- Tarefa 1.10. Borrar da táboa dos empregados aos que teñan cumpridos os 60 anos.
- Tarefa 1.11. Escribir unha única sentenza que permita borrar da táboa *departamento* o departamento número 121 e da táboa *empregado* todos os empregados que traballan nese departamento.

- Tarefa 1.12. Executar o seguinte script para poder crear unha táboa temporal co nome *empregado_120*:

```
create temporary table empregado_120 like empregado;
```

Inserir na táboa *empregado_120* os datos de todas as filas da táboa *empregado* que teñan o valor 120 na columna *empDepartamento*;

Sobre a base de datos *tendaBD*

- Tarefa 1.13. Inserir filas na táboa *facturas* collendo os datos de todos os clientes que teñan vendas no mes 5 de 2015 sen facturar (*ven_factura* toma o valor null). As columnas que non se obteñen da táboa *clientes*, teñen os seguintes valores:

Nome columna	Valor ou observacións
fac_numero	Valor autoincrementado
fac_mes	5
fac_ano	2015
fac_data	A data actual do sistema
fac_importe	0

- Tarefa 1.14. Inserir na táboa *vendas* unha fila cos seguintes datos; se existe unha venda co mesmo id, debe ser substituída por esta:

Nome columna	Observacións
ven_id	151
ven_tenda	8
ven_empregado	25
ven_cliente	12
ven_data	10 de xuño de 2015 ás 12:25:00
ven_factura	Descoñecida. Aínda non se facturou a venda

Solución

- Tarefa 1.1

```
insert into centro
values( 40,'FRANQUICIA LUGO', 'C/PROGRESO, 8 - LUGO' ) ;
```

Tarefa 1.2

```
insert into empregado (empNumero, empDepartamento, empExtension, empDataNacemento,
empDataIngreso, empSalario, empComision, empFillos, empNome)
values (750,110,25,'1990-02-12',curdate(),825,50,1,'BARCIA, ANGELES'),
(751,110,25,'1985-10-22',subdate(curdate(),15),900,null,0,'MENDEZ, RICARDO'),
(752,120,45,'1992-05-09',null,1200,150,2,'BERNARDEZ, LUCIA'),
(753,100,200,'1980-11-05',curdate(),1000,300,1,'VALIN, EVA');
```

- Tarefa 1.3

```
update empregado
set empSalario = round(empSalario*1.05,2),
empComision = ifnull(round(empComision*1.065,2),null);
```

- Tarefa 1.4

```
update empregado
set empDataIngreso = adddate(last_day(curdate()),1);
where empNumero = 752 ;
```

- Tarefa 1.5

```
update empregado
set empSalario = round(empSalario*1.02,2)
where empDepartamento = 120 ;
```

■ Tarefa 1.6

```
update empregado
set empComision=ifnull(empComision,0)+50
where empDepartamento in (select depNumero
                           from departamento
                           where depCentro in (select cenNumero
                                                from centro
                                                where cenNome='SEDE CENTRAL'));
```

■ Tarefa 1.7

```
update departamento
set depPresuposto = depPresuposto-depPresuposto*0.10
order by depPresuposto desc
limit 1 ;
-- Outra forma de facelo, aínda que en MySQL dá un erro:
update departamento
set depPresuposto = depPresuposto-depPresuposto * 0.10
where depPresuposto = (select max(depPresuposto) from departamento);
-- Unha alternativa en MySQL, empregando variables de usuario:
select @mayor:=max(depPresuposto) from departamento;
update departamento
set depPresuposto=depPresuposto-depPresuposto*0.10
where depPresuposto = @mayor;
```

■ Tarefa 1.8

En Workbench, escríbense as seguintes sentenzas e coa opción de 'gardar script como', gárdase o texto no arquivo *cambio_presuposto.sql*

```
update departamento
set depPresuposto = depPresuposto-20000
where depNome='personal';
update departamento
set depPresuposto = depPresuposto + 20000
where depNome='proceso de datos';
update departamento
set depPresuposto = depPresuposto - 10000
where depNome='sector industrial';
update departamento
set depPresuposto = depPresuposto + 4000
where depNome='organizacion';
update departamento
set depPresuposto = depPresuposto + 6000
where depNome='direccion comercial';
```

■ Tarefa 1.9

```
delete from empregado
where empNumero = 380 ;
```

■ Tarefa 1.10

```
delete from empregado
where timestampdiff(year, empDataNacemento, curdate())>=60;
-- Outra forma de facelo:
delete from empregado
where date_add(empDataNacemento, interval 60 year)<curdate();
```

■ Tarefa 1.11

```
delete em, de
from departamento as de join empregado as em
where de.depNumero=em.empDepartamento
and de.depNumero=121;
```

■ Tarefa 1.12

```
-- creación da táboa temporal
```

```
create temporary table empregado_120 like empregado;
-- inserción de datos
insert into empregado_120
select *
from empregado
where empDepartamento = 120;
```

■ Tarefa 1.13

```
set @mes=5;
set @ano=2015;
set @data_factura=curdate();
set @importe=0;
insert into facturas (fac_mes, fac_ano,fac_data,fac_clt_cif,fac_clt_apelidos,
    fac_clt_nome,fac_clt_enderezo,fac_clt_cp,fac_clt_poboacion,fac_clt_pais,fac_importe)
select @mes,@ano,@data_factura,clt_cif,clt_apelidos,clt_nome,clt_enderezo,clt_cp,
    clt_poboacion,clt_pais,@importe
from clientes
where clt_id in (select ven_cliente
    from vendas
    where month(ven_data)=@mes and year(ven_data) =@ano
    and ven_factura is null);
```

■ Tarefa 1.14

```
replace into vendas values (151,8,25, 12, '2015-06-10 12:25:00',null);
```

1.3.2 Tarefa 2. Analizar e probar sentenzas, adoptando as medidas necesarias para manter a integridade e consistencia dos datos

A tarefa consiste en analizar e probar sentenzas INSERT, REPLACE, UPDATE e DELETE, adoptando as medidas necesarias para manter a integridade e consistencia dos datos.

Sobre a base de datos tendabd

- Tarefa 2.1. Executar a seguinte sentenza, analizar os erros que provocan, e indicar as medidas que se deberían de adoptar para manter a integridade e consistencia dos datos.

```
insert into clientes ( clt_id, clt_cif, clt_apelidos, clt_nome, clt_enderezo,
    clt_cp, clt_poboacion, clt_pais, clt_alta)
values (92, '33956665D','Varela Montero','Luisa','Rua Vella, 5-2º', '15006',
    'Coruña',73,curdate());
```

- Tarefa 2.2. Executar a seguinte sentenza, analizar os erros que provocan, e indicar as medidas que se deberían de adoptar para manter a integridade e consistencia dos datos.

```
insert into clientes ( clt_cif, clt_apelidos, clt_nome, clt_enderezo, clt_cp,
    clt_poboacion, clt_pais, clt_alta)
values ('16137107P','Nuñez Castro','Maria','Rua Nova, 22 - 5º','27001',
    'Lugo',73,curdate());
```

- Tarefa 2.3. Executar a seguinte sentenza, analizar os erros que provocan, e indicar as medidas que se deberían de adoptar para manter a integridade e consistencia dos datos..

```
insert into vendas (ven_tenda, ven_empregado, ven_cliente, ven_data, ven_factura)
values (24,null,55,now(),null);
```

- Tarefa 2.4. Executar a seguinte sentenza, analizar os erros que provocan, e indicar as medidas que se deberían de adoptar para manter a integridade e consistencia dos datos.

```
insert into vendas (ven_tenda, ven_empregado, ven_cliente, ven_data, ven_factura)
values (24,10,155,now(),null);
```

- Tarefa 2.5. Executar a seguinte sentenza, analizar os erros que provocan, e indicar as medidas que se deberían de adoptar para manter a integridade e consistencia dos datos.

```
insert into vendas (ven_tenda, ven_empregado, ven_cliente)
values (24,10,55);
```

Solución

■ Tarefa 2.1

Ao executar a sentenza prodúcese un erro debido a que non se cumpre a restrición de clave primaria (PRIMARY KEY) asociada á columna *clt_id* porque xa existe un cliente que ten o valor 92 na columna *clt_id*. A mensaxe de erro é a seguinte:

```
10 12:57:59 insert into clientes (clt_id, clt_cif, clt_apellidos, clt_nombre, clt_enderezo, clt_cp, clt_poboa... Error Code: 1062. Entrada duplicada '92' para la clave 'PRIMARY'
```

- Unha medida que se podería adoptar é eliminar da lista de columnas e da lista de valores, a columna *clt_id* que ten a propiedade AUTO_INCREMENT, desta maneira o sistema asígnalle á columna un valor que non se vai a repetir e que será o seguinte ao que se lle asignou ao último cliente. Código alternativo:

```
insert into clientes (clt_cif, clt_apellidos, clt_nombre, clt_enderezo, clt_cp,
                    clt_poboacion, clt_pais, clt_alta)
values ('33956665D', 'Varela Montero', 'Luisa', 'Rua Vella, 5-2º', '15006',
       'Coruña', 73, curdate());
```

```
23 13:47:33 insert into clientes (clt_cif, clt_apellidos, clt_nombre, clt_enderezo, clt_cp, clt_poboacion, clt_pais, clt_alta) values ('33956665D', 'Varela... 1 row(s) affected
```

- Outra medida a adoptar se o que se quere é asignarlle o *id* número 92 a ese cliente é borrar o que había antes con ese *id*, podería ser substituír a sentenza INSERT por unha sentenza REPLACE mediante o código:

```
replace into clientes ( clt_id, clt_cif, clt_apellidos, clt_nombre, clt_enderezo,
                    clt_cp, clt_poboacion, clt_pais, clt_alta)
values (92, '15202002D', 'Varela Montero', 'Luisa', 'Rua Vella, 5-2º', '15006',
       'Coruña', 73, curdate());
```

```
11 12:58:49 replace into clientes (clt_id, clt_cif, clt_apellidos, clt_nombre, clt_enderezo, clt_cp,... 2 row(s) affected
```

Esta segunda medida non sería válida se o cliente tivera algunha venda asociada, porque a clave foránea *ven_cliente* ten asociada a acción RESTRICT para a operación DELETE.

■ Tarefa 2.2

Ao executar a sentenza prodúcese un erro debido a que non se cumpre a restrición de unicidade (UNIQUE) asociada á columna *clt_cif* xa que existe un cliente que ten o mesmo valor na columna *clt_cif*. A mensaxe de erro é a seguinte:

```
13 13:05:39 insert into clientes (clt_cif, clt_apellidos, clt_nombre, clt_enderezo, clt_cp, clt_poboacio... Error Code: 1062. Entrada duplicada '16137107P' para la clave 'clt_cif'
```

A mensaxe informa que hai unha entrada duplicada para a columna *clt_cif* que ten asociado un índice tipo UNIQUE. A única medida que se pode adoptar é cambiar o valor da columna. Non debería haber dous clientes co mesmo CIF, polo que un deles ten mal o seu CIF.

■ Tarefa 2.3

Ao executar a sentenza prodúcese un erro debido a que non se cumpre a restrición de valor nulo (NOT NULL) asociada á columna *ven_empregado* xa que non admite o valor NULL para esa columna. A mensaxe de erro é a seguinte:

15 13:27:27 insert into vendas (ven_tienda, ven_empleado, ven_cliente, ven_data, ven_factura) ... Error Code: 1048. La columna 'ven_empleado' no puede ser nula

A mensaxe informa que se asignou o valor NULL á columna *ven_empleado* que está definida coa propiedade NOT NULL. A única medida que se pode adoptar é asignarlle á columna un valor distinto de NULL e que coincida con algún dos valores que toma a columna *emp_id*, xa que está definida como unha clave foránea que fai referencia a esa columna da táboa *empleado*.

■ Tarefa 2.4

Ao executar a sentenza prodúcese un erro debido a que non se cumpre a restrición de clave foránea (FOREIGN KEY) asociada á columna *ven_cliente* porque o valor que toma esa columna non coincide con ningún valor dos que toma a columna *clt_id* da táboa *clientes* á que fai referencia. A mensaxe de erro é a seguinte:

16 13:34:09 insert into vendas (ven_tienda, ven_empleado, ven_cliente, ven_data, ven_factura) ... Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (tendabd`.`vendas`, CONSTRAINT `fk_vendas_cliente` FOREIGN KEY (`ven_cliente`) REFERENCES `clientes` (`clt_id`) ON UPDATE CASCADE)

A mensaxe informa de que non se pode inserir unha fila que ten na clave foránea (*cen_cliente*) un valor que non coincide con ningún dos valores que toma a columna *clt_id* da táboa *clientes* á que fai referencia. O texto completo da mensaxe é a seguinte:

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (tendabd`.`vendas`, CONSTRAINT `fk_vendas_cliente` FOREIGN KEY (`ven_cliente`) REFERENCES `clientes` (`clt_id`) ON UPDATE CASCADE)

A única medida que se pode adoptar é asignarlle á columna *ven_cliente* un valor que coincida con algún valor dos que toma a columna *clt_id* da táboa *clientes* á que fai referencia.

■ Tarefa 2.5

Ao executar a sentenza non se produce ningún erro a pesar de non asignarlles valores ás columnas *ven_data* e *ven_factura* e que a columna *ven_data* non admite valores nulos (NOT NULL). Isto é debido a que esas columnas teñen asignados valores por defecto de forma explícita (DEFAULT). A mensaxe que se mostra ao executar a sentenza é a seguinte:

18 13:38:23 insert into vendas (ven_tienda, ven_empleado, ven_cliente, ven_data, ven_factura) ... 1 row(s) affected

A mensaxe informa que foi inserida unha fila na táboa de *vendas*. As columnas *ven_data* e *ven_factura* toman os valores por defecto que son a data e hora actual (CURRENT_TIMESTAMP) para a primeira e o valor nulo (NULL) para a segunda.

ven_id	ven_tienda	ven_empleado	ven_cliente	ven_data	ven_factura
153	24	10	55	2015-12-27 13:38:23	NULL