6. Edición dos datos

6.1 Introdución

SQL corresponde ao acrónimo de Structured Query Language (Linguaxe Estruturada 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).

6.2 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, MEMORY, 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 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 carácteres, 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 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:

```
1 •  describe practicas5.departamento;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

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 tomar mediante expresións de asignación.

Na opción 1, VALUES | VALUE permitirán indicar explicitamente a lista de valores que van 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 de 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 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.

6.3 Sentenza REPLACE

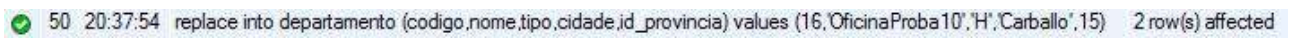
A sentenza REPLACE é unha alternativa para INSERT que só se diferenza en que se existe algunha fila anterior co mesmo valor para unha clave primaria (PRIMARY KEY) ou única (UNIQUE), elimínase a fila que xa existía con ese valor na clave, e insírese no seu lugar a fila cos novos valores.

Exemplo:

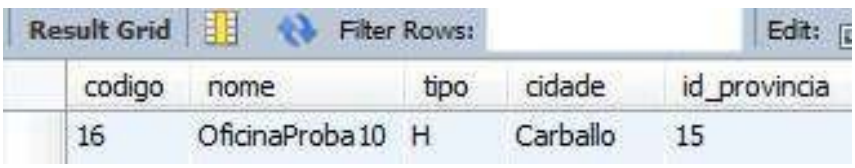
```
insert into departamento (codigo,nome,tipo,cidade,id_provincia)
values (16,'OficinaProba10','H','Carballo',15);
```

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);
```



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:



6.4 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, MEMORY, 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 columna 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.

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

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

6.4.1 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, MEMORY, 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 borrar filas.

Cando se van 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 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:
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;
```

6.5 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'.

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.

6.6 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.
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 ordes 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 de Administración de bases de datos), ou están incorporados na maioría dos clientes gráficos.