Tema 7

Lenguaje de manipulación de datos (DML).

Muchos de los conceptos que se verán en este tema, serán ampliados más adelante ya que por ejemplo las consultas o las transacciones son elementos entrelazados con estos.

Lenguaje de manipulación de datos[1] (DDL) sirve para insertar, modificar y eliminar datos de la BdD.

Realmente también incluye las consultas que trataremos en los temas siguientes aunque hay gente que las considera un sublenguaje propio: DQL[2].

Sentencias DDL:

- `SELECT`: consultar información de la BdD. Lo veremos en los temas siguientes.
- `INSERT`: añadir información a la BdD.
- `UPDATE`: actualizar información en la BdD.
- `DELETE`: eliminar información de la BdD.

7.1. Insertar datos.

Versión **MUY** simplificada de la sintaxis. La versión completa se puede consultar aquí.

[1] Data Definition Language.
[2] data query language.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
   [INTO] tbl_name
   [PARTITION (partition_name [, partition_name] ...)]
   [(col_name [, col_name] ...)]
   { {VALUES | VALUE} (value_list) [, (value_list)] ... }
   [AS row_alias[(col_alias [, col_alias] ...)]]
   [ON DUPLICATE KEY UPDATE assignment_list]

value:
   {expr | DEFAULT}

value_list:
   value [, value] ...

assignment:
   col_name =
      value
      | [row_alias.]col_name
      | [tbl_name.]col_name
      | [row_alias.]col_alias

assignment_list:
      assignment [, assignment] ...
```

Suponiendo la tabla:

```
CREATE TABLE empleados (
   id_empleado INT NOT NULL,
   nombre VARCHAR(20),
   apellido VARCHAR(20) NOT NULL,
   email VARCHAR(25) NOT NULL,
   telefono VARCHAR(20),
   fecha_contratacion date NOT NULL,
   id_trabajo VARCHAR(10) NOT NULL,
   salario DECIMAL(8,2),
   comision DECIMAL(2,2)
);
```

Podemos insertar datos siguiendo el orden de creación:


```
INSERT INTO empleados
   VALUES (405, 'Maya', 'Van Pobel', 'mp@scott.com', '954646331',
      '1992-08-15', 'ST_MAN', 5800, NULL);
```

Aunque es preferible indicar los nombres de columnas por varios motivos:

- Facilita la lectura.
- Si la estructura de la tabla cambia, minimizamos las posibilidades de fallo.

```
INSERT INTO empleados
   (id_empleado, nombre, apellido, email, telefono,
   fecha_contratacion, id_trabajo, salario, comision)
VALUES
   (405, 'Maya', 'Van Pobel', 'mp@scott.com', '954646331',
   '1992-08-15', 'ST_MAN', 5800, NULL);
```

Nota:

Para ver los datos de una tabla puedes usar la consulta:
`SELECT * FROM nombre_tabla;`
Por ejemplo: `SELECT * FROM empleados;`

7.1.1. Insertar datos con valores por defecto.

Si una o más columnas tienen valores por defecto:

```
CREATE TABLE usuario (
   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   nombre VARCHAR(50) NOT NULL,
   apellido1 VARCHAR(50) NOT NULL,
   apellido2 VARCHAR(50),
   sexo ENUM('H', 'M') NOT NULL DEFAULT 'M',
   fecha_nacimiento DATE NOT NULL
);
```

Podemos usar la palabra clave `DEFAULT`:


```
INSERT INTO usuario
   (id, nombre, apellido1, apellido2, sexo, fecha_nacimiento)
VALUES
   (1, 'María', 'Ramírez', 'López', DEFAULT, '1983-09-13');
```

U omitir la columna:

```
INSERT INTO usuario
   (id, nombre, apellido1, apellido2, fecha_nacimiento)
VALUES
   (2, 'Paola', 'González', 'Aguirre', '1987-12-31');
```

7.1.2. Insertar datos obtenidos con una consulta.

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
   [INTO] tbl_name
   [PARTITION (partition_name [, partition_name] ...)]
   [(col_name [, col_name] ...)]
   { SELECT ...
      | TABLE table_name
      | VALUES row_constructor_list
   }
   [ON DUPLICATE KEY UPDATE assignment_list]

row_constructor_list:
   ROW(value_list)[, ROW(value_list)][, ...]

assignment:
   col_name =
      value
      | [row_alias.]col_name
      | [tbl_name.]col_name
      | [row_alias.]col_alias

assignment_list:
   assignment [, assignment] ...
```

Veremos consultas en los temas siguientes pero vemos un ejemplo. Tenemos una tabla de aspirantes a un puesto superior:

```
CREATE TABLE aspirantes (
   id_empleado INT NOT NULL,
   nombre VARCHAR(20),
   apellido VARCHAR(20) NOT NULL,
   email VARCHAR(25) NOT NULL,
);
```

Podemos insertar datos como:

```
INSERT INTO aspirantes
   (id_empleado, nombre, apellido, email, telefono)
   SELECT
      id_empleado, nombre, apellido, email, telefono
   FROM
      empleados
   WHERE
      id_trabajo = 'ST_CLERK';
```

7.2. Modificar datos.

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
   SET assignment_list
   [WHERE where_condition]
   [ORDER BY ...]
   [LIMIT row_count]

value:
   {expr | DEFAULT}

assignment:
   col_name = value

assignment_list:
   assignment [, assignment] ...
```

Se pueden actualizar todos los de la tabla:

```
UPDATE empleados SET nombre=UPPER(nombre);
```

Pero es más habitual filtrar para actualizar una o más filas con `WHERE`:

```
UPDATE empleados SET id_director=601 WHERE id_empleado=503;
```

Se pueden actualizar varios campos y varias filas:

```
UPDATE empleados
SET
   salario = 1.2 * salario,
   comision = 0.1
WHERE
   id_departamento = 40;
```

Incluso con una subconsulta (las veremos más adelante):

```
UPDATE empleados
SET
   salario = 1.2 * salario,
   comision = 0.1
WHERE
   id_departamento = (SELECT
         id_departamento
      FROM
         departamentos
      WHERE
         nombre_departamento = 'Envíos');
```

Nota:

El uso de `WHERE` se puede complicar mucho más, pero lo veremos al usarlo con consultas donde es más habitual.

7.3. Eliminar datos.

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

```
DELETE FROM localizaciones WHERE id_localizacion = 3100;
```

Podemos borrar todos los datos, pero en MySQL es necesario desactivar el modo seguro.

```
DELETE FROM localizaciones;
```

7.4. Integridad referencial en operaciones de modificación y borrado.

Las opciones que vimos en el punto 6.2.1.2 tienen consecuencias.

Es necesario tenerlo en cuenta antes de realizar la operación. Puede tener consecuencias no deseadas o impedirnos llevarla a cabo.

Ejercicios 7.1

Con la configuración estándar de MySQL no puedes realizar modificaciones o borrados que afecten a más de una fila a la vez.

Si en algún caso necesitas hacerlo, deberías desactivarlo y luego volverlo a activar:

1. *Edit* > *Preferences*.
2. *SQL Editor* > Desmarca la casilla *Safe Updates* > *OK*.
3. *Query* > *Reconnect to server*.
4. Ejecuta tu consulta.
5. Vuelve a las preferencias para volver a marcar la casilla de *Safe Updates*.
6. *Query* > *Reconnect to server*.

Ahora ya puedes hacer los ejercicios:

1. Crea un script para insertar datos en la BdD creada en el ejercicio 6.2.6. Se valorará la creación de varios elementos en cada tabla de forma coherente. P.Ej: debe haber varias escuderías pero muchos más pilotos.
2. Crea un script para insertar datos en la BdD creada en el ejercicio 6.2.7. Se valorará la creación de varios elementos en cada tabla de forma coherente.
3. Crea un script para insertar datos en la BdD creada en el ejercicio 6.2.9. Se valorará la creación de varios elementos en cada tabla de forma coherente.
4. Un elemento se introdujo por error con un nombre no adecuado. Modifica el nombre de un elemento.
5. Un elemento se introdujo en una marca incorrecta. Escribe la sentencia para cambiar el elemento de marca.
6. Escribe una modificación que cambie la clave primaria de una marca. ¿Qué sucede con los elementos asociados a ella?
7. En el script de creación de la BdD *Empleados* del apéndice B, ¿qué restricciones de modificación y borrado pondrías a cada clave ajena? ¿Por qué?. Modifica el script para incluir estos cambios y prueba a ejecutarlo. Ejecuta después el de inserción, prestando atención a que no se produzcan errores al hacerlo.
8. Añade empleados de un nuevo departamento que se encuentre en una localización que no está registrada en el script.
9. Prueba a borrar un departamento que tenga empleados. ¿Qué sucede?
10. ¿Cómo puedes eliminar un departamento? Ten en cuenta que hay claves ajenas cruzadas entre *empleados* y *departamentos*. Prueba tu solución.

7.5. Bibliografía.

- Departamento de Informática, IES Luis Vélez de Guevara (2021). Gestión de Bases de Datos.
- José Juan Sánchez Hernández (2021/2022). Bases de datos / Gestión de Bases de datos
- Structured Query Language. Wikibook. Wikipedia.
- Curso de MySQL. Con Clase.
- MySQL 8.0 Reference Manual.
- Iván López Montalbán, Manuel de Castro Vázquez, John Ospino Rivas (2022). Bases de Datos (2ª Edición) . Garceta.

