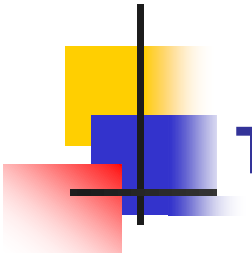
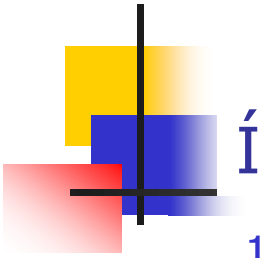


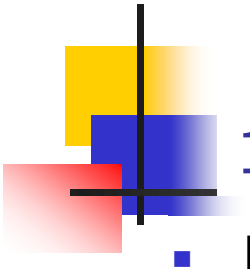
Tema 4

LENGUAJE SQL



Índice

1. Inserción de nuevas filas en la base de datos
2. Modificación de filas.
3. Eliminación de Filas
4. Restricciones de Integridad y Actualizaciones
    1. Ejemplos de borrados y modificaciones en cascada.
5. Control de transacciones: COMMIT Y ROLLBACK.



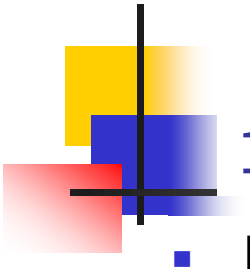
1. Inserción de Nuevas Filas en la Base de Datos

- Para añadir nuevas filas a una tabla utilizaremos la sentencia **INSERT**, cuyo formato típico es:
  - **INSERT INTO NombreTabla [(NombreColumna [,NombreColumna…])]**
  - **VALUES (Expresion [, Expresión …]);**
- ***Notación**: la lista de columnas en las que insertamos es opcional, por lo que va entre corchetes. Si no se especifica se esperan valores para todas las columnas.*
- Donde:
  - **NombreTabla:** es el nombre de la tabla en la que queremos insertar una fila.
  - **NombreColumna**: incluye las columnas en las que queremos insertar.
  - **Expresion:** indica las expresiones cuyos valores resultado se insertarán, existiendo una correspondencia con la lista de columnas anterior

1. Inserción de Nuevas Filas en la Base de Datos

- Como se ve en el formato también se pueden insertar filas en una tabla sin especificar la lista de columnas pero en este caso la lista de valores a insertar deberá coincidir en número y posición con las columnas de la tabla. El orden establecido para las columnas será el indicado al crear la tabla (el mismo que aparece al hacer una descripción de la tabla DESC NombreTabla*).*
- **Ejemplo**:
  - Insertar una nueva fila en la tabla *socios* introduciremos la siguiente instrucción:
    - mysql> INSERT INTO socios (num_socio, apellidos, telefono, direccion, codigo_postal)

      -> VALUES (1000, 'LOPEZ','913211234','C. REAL', '28804');
  - Nota: El campo fecha_alta no lo he puesto porque tiene un valor por defecto

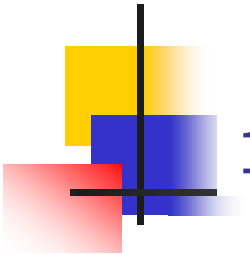


1. Inserción de Nuevas Filas en la Base de Datos

- Este formato de inserción tiene, además, las siguientes características:
  - En la lista de columnas se pueden indicar todas o algunas de las columnas de la tabla. En este último caso, aquellas columnas que no se incluyan en la lista quedarán sin ningún valor en la fila insertada, es decir, se asumirá el valor NULL o el valor por defecto para las columnas que no figuren en la lista.
  - Los valores incluidos en la lista de valores deberán corresponderse posicionalmente con las columnas indicadas en la lista de columnas, de forma que el primer valor de la lista de valores se incluirá en la columna indicada en primer lugar, el segundo en la segunda columna, y así sucesivamente.
  - Se puede utilizar cualquier expresión para indicar un valor siempre que el resultado de la expresión sea compatible con el tipo de dato de la columna correspondiente.

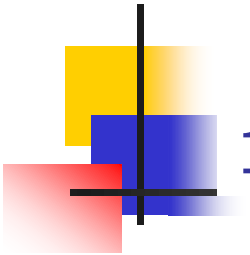
1. Inserción de Nuevas Filas en la Base de Datos

- **Realizar las siguiente inserciones la la tabla Socios**:
  - Inserción de un socio con num_socio=1001 y con fecha de alta 2008-01-12
    - mysql> INSERT INTO Socios (num_socio,apellidos,telefono,fecha_alta,direccion,codigo_postal)
      -> VALUES (1001,'PEREZ','918451256', '2005-01-12', 'C. MAYOR 31',’28400’);
  - Inserción de un socio con num_socio = 1002 y con fecha de alta 2008-01-12
    - mysql> INSERT INTO Socios (num_socio,apellidos,telefono,fecha_alta,direccion,codigo_postal)
      -> VALUES (1002,'LOPEZ PEREZ','916543267', '2005-01-18', 'C. REAL 5',’28400’);



1. Inserción de Nuevas Filas en la Base de Datos

- Inserción de un socio con num_socio=1003
  - mysql> INSERT INTO socios (num_socio,apellidos,fecha_alta,telefono,direccion,codigo_postal)
    -> VALUES (1003,'ROMA LEIVA', '2005-01-21', '912233554,'C. PANADEROS 9 ','28431');
- Inserción de un préstamo para el socio con num_socio=1000
  - mysql> INSERT INTO prestamos (num_prestamo, num_socio)
    -> VALUES (1,1000);
- Inserción de un préstamo para el socio con num_socio=1002
  - mysql> INSERT INTO prestamos (num_prestamo, num_socio)
    -> VALUES (2,1002);



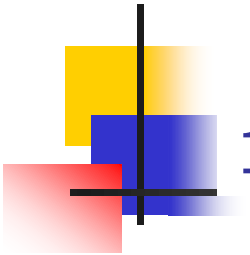
1. Inserción de Nuevas Filas en la Base de Datos

- Inserción de un socio con num_socio=1004 con una instrucción INSERT sin lista de columnas
  - mysql> INSERT INTO socios

    -> VALUES (1004,'GOMEZ DURUELO','918654329', '2005-01-31', 'C. REAL 15',’28400’);
- Inserción de un socio con num_socio=1005 con una instrucción INSERT sin valor en el campo fecha que tiene un valor por defecto.
  - mysql> INSERT INTO socios

    (num_socio,apellidos,telefono,direccion,codigo_postal)

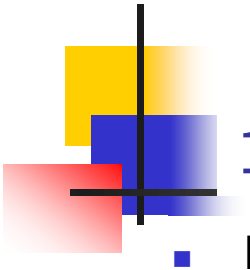
    -> VALUES (1005,'PEÑA MAYOR','918515256','C. LARGA 31',’28431’);



1. Inserción de Nuevas Filas en la Base de Datos

- Inserción de un socio con num_socio=1004 en la tabla préstamos con una instrucción INSERT sin lista de columnas
  - mysql> INSERT INTO prestamos

    -> VALUES (4,1004);
- Dará errores cuando:
  - Insertar un valor que no hemos indicado la columna.
  - Insertar un socio que ya está insertado.
  - Insertar una fila con valores duplicados para un campo que tiene la restricción UNIQUE, por ejemplo el apellido. No podríamos dar de alta a más de un socio con el mismo apellidos.
  - Insertar una fila en la tabla préstamos con un valor en la columna num_socio que no existe en la tabla socios.

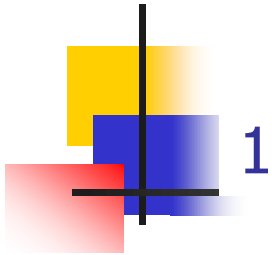


1. Inserción de Nuevas Filas en la Base de Datos

- Para comprobar los valores insertados, hacemos:
  - mysql> SELECT *
    -> FROM socios;
  - mysql> SELECT *
    -> FROM prestamos;

1. Inserción de Nuevas Filas en la Base de Datos

- **Ejemplo** de inserción con un campo autonumérico. Repasamos la creación de una tabla con una columna autonumérica.
  - mysql> CREATE TABLE inventario
    -> (num INT(2) AUTO_INCREMENT PRIMARY KEY,
    -> descripcion VARCHAR(15));
  - Inserción sin enumerar la columna autonumérica
    - mysql> INSERT INTO inventario (descripcion)
      -> VALUES ('ARMARIO BLANCO');
    - mysql> INSERT INTO inventario (descripcion)
      -> VALUES ('MESA MADERA');
  - Inserción de toda la fila (en este caso debe ponerse NULL en la columna correspondiente)
    - mysql> INSERT INTO inventario
      -> VALUES (NULL,'ORDENADOR');



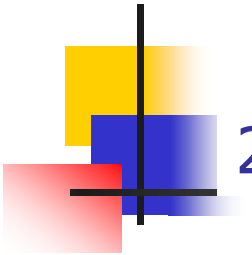
1. Inserción de Nuevas Filas en la Base de Datos

- mysql> INSERT INTO inventario
  -> VALUES (NULL,'SILLA GIRATORIA');

- Comprobación de los valores insertados:
  - mysql> SELECT *
    -> FROM inventario;

2. Modificación de Filas

- En ocasiones necesitaremos modificar alguno de los datos de las filas existentes de una tabla. En estos casos utilizaremos la sentencia UPDATE. El formato es:
  - **UPDATE NombreTabla**
  - **SET NombreColumna = Expresion [, NombreColumna = Expresion….]**
  - **[WHERE Condición];**
- **Notación**: puede actualizarse una o varias columnas, por lo que la segunda actualización va entre corchetes La cláusula WHERE aparece entre corchetes porque es opcional. En el caso de que no se utilice, la actualización afectará a toda la tabla.
- Donde:
  - **NombreTabla:** indica la tabla destino donde se encuentran las filas que queremos modificar.



2. Modificación de Filas

- **NombreColumna:** indica el nombre de la columna cuyo valor queremos modificar
- **Expresión:** es una expresión cuyo valor resultado será el nuevo valor de la columna.
- **Condición:** es la condición que deben cumplir las filas para que les afecte la modificación.

2. Modificación de Filas

- **Veamos algunos ejemplos**
  - Modificar la calle del socio 1000 por C. CUESTA 2
    - mysql> UPDATE socios
      -> SET direccion = 'C.CUESTA 2‘
      -> WHERE num_socio = 1000;
  - Modificar el teléfono del socio de número 1000. El nuevo número es 918455431
    - mysql> UPDATE socios
      -> SET telefono = '918455431‘
      -> WHERE num_socio = 1000;
  - También podíamos haber modificado las dos columnas con una sola sentencia:
    - mysql> UPDATE socios
      -> SET telefono = '918455431',direccion='C.CUESTA 2‘
      -> WHERE socio_no = 1000;

2. Modificación de Filas

- **Veamos algunos ejemplos**
  - Las actualizaciones anteriores afectan a una única fila pero podemos escribir comandos de actualización que afecten a varias filas:
    - mysql> UPDATE socios

      -> SET codigo_postal = 28401

      -> WHERE codigo_postal = 28400;
  - Si no se incluye la cláusula WHERE la actualización afectará a todas las filas. El siguiente ejemplo modifica la fecha de alta de todos los empleados al valor 1 de enero de 2005.
    - mysql> UPDATE socios

      -> SET fecha_alta = '2005-01-01';
  - Podemos comprobar las modificaciones:
    - mysql> SELECT *

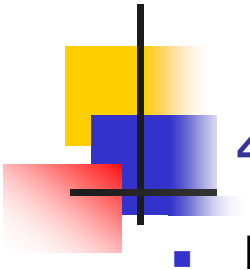
      -> FROM socios;

3. Eliminación de Filas

- Para eliminar o suprimir filas de una tabla utilizaremos la sentencia DELETE. Su formato genérico es el siguiente:
  - **DELETE FROM NombreTabla**
  - **[WHERE Condicion];**
- ***Nota**: la cláusula WHERE es opcional por eso aparece entre corchetes. Si no se especifica se borraran todas las filas de la tabla*
  - donde **NombreTabla**: indica la tabla destino donde se encuentran las filas que queremos borrar
  - **Condición**: es la condición que deben cumplir las filas a borrar

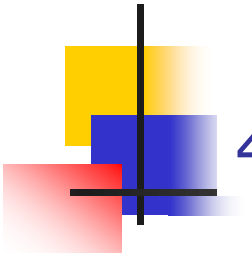
3. Eliminación de Filas

- **Veamos algunos ejemplos**
  - A continuación se muestran algunos ejemplos de instrucciones DELETE
    - mysql> DELETE
      -> FROM socios
      -> WHERE num_socio = 1001;
  - Hay que tener cuidado con las claves ajenas. Como prestamos tiene una clave ajena que referencia a socios si pretendemos borrar un socio que tiene préstamos nos encontraremos con un error.
    - mysql> DELETE
      -> FROM socios
      -> WHERE num_ socio = 1002;
  - Podemos comprobar las modificaciones:
    - mysql> SELECT *
      -> FROM socios;



4. Restricciones de Integridad y Actualizaciones

- Hay que tener en cuenta las restricciones que podemos tener en nuestras tablas. Con restricciones me refiero a las siguientes:
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
  - NOT NULL
  - UNIQUE
- Estas restricciones dan lugar a que no podamos hacer cualquier modificación en los datos de las tablas. No nos estará permitido hacer inserciones ni modificaciones con valores no permitidos en las columnas ni borrar filas a las que referencien otras filas de la misma o de otra tabla.
  - **PRIMARY KEY**: No podemos hacer inserciones en una tabla con valores repetidos de las claves primarias.



4. Restricciones de Integridad y Actualizaciones

- **FOREIGN KEY**: Se utilizan para hacer referencia a columnas de otras tablas. Cualquier valor que se inserte en esas columnas tendrá su equivalente en la tabla referida. Esto nos limita los valores de las claves ajenas no podrán tomar valores que no existan en las columnas referenciadas

4.1.- Ejemplos de Borrados y Modificaciones

- Vamos a ver con un ejemplo como funciona el borrado en cascada.
  - Creamos la base de datos EMPRESA que se encuentra en el documento **empresa.sql** colgado en la carpeta compartida.
- Una vez creada la base de datos vamos a realizar los siguientes apartados
  - Borramos una fila en la tabla departamentos. Si no existiese borrado en cascada, debido a la integridad referencial, no podríamos borrar ningún departamento que tuviese empleados. De esta forma al borrar un departamento se borrarán todos los empleados de ese departamento.
    - mysql> DELETE FROM departamentos

      -> WHERE dept_no=10;
- Hacemos ahora un select * from departamentos y select * from empleados para comprobar el resultado.

5. Control de Transacciones: COMMIT y ROLLBACK

- Los gestores de bases de datos disponen de dos comandos que permiten confirmar o deshacer los cambios realizados en la base de datos:
  - **COMMIT**: confirma los cambios realizados haciéndolos permanentes.
  - **ROLLBACK**: deshace los cambios realizados.
- Cuando hacemos modificaciones en las tablas, estas no se hacen efectivas (llevan a disco) hasta que no ejecutamos la sentencia COMMIT. Cuando ejecutamos comandos o cerramos la sesión se ejecuta un COMMIT automático.
- El comando ROLLBACK nos permite deshacer estos cambios sin que lleguen a validarse. Cuando ejecutamos este comando se deshacen todos los cambios hasta el último COMMIT ejecutado.

5. Control de Transacciones: COMMIT y ROLLBACK

- Hay dos formas de trabajar con AUTO_COMMIT: activado (validación automática de los cambios) o desactivado. Si está activado se hace COMMIT automáticamente de cada sentencia y no es posible hacer ROLLBACK. Si no lo está, tenemos la posibilidad de hacer ROLLBACK después de las sentencias INSERT, UPDATE y DELETE dejando sin escribir en disco los cambios. Es útil para hacer pruebas.
- Existe una variable, AUTO_COMMIT, que indica la forma de trabajo y tiene el valor 1 si está en modo AUTO_COMMIT y 0 si no lo está. Por defecto el MySQL está en modo AUTO_COMMIT A 1. Su valor se puede cambiar con la sentencia:
  - mysql> SET AUTOCOMMIT = 0; o mysql> SET AUTOCOMMIT = OFF;
- Para ver el valor de autocommit, se puede hacer una de las dos opciones:
  - select @@autocommit;
  - show variables like 'autocommit';

5. Control de Transacciones: COMMIT y ROLLBACK

- Vamos a ver un ejemplo
  - mysql> SELECT * FROM inventario;

```
+-----+-----------------+
| num | descripcion     |
+-----+-----------------+
|   1 | ARMARIO BLANCO  |
|   2 | MESA MADERA     |
|   3 | ORDENADOR       |
|   4 | SILLA GIRATORIA|
+-----+-----------------+
```

  - mysql> SET AUTOCOMMIT = 0;
  - mysql> SELECT * FROM inventario;

```
+-----+-----------------+
| num | descripcion     |
+-----+-----------------+
|   1 | ARMARIO BLANCO  |
|   2 | MESA MADERA     |
|   3 | ORDENADOR       |
|   4 | SILLA GIRATORIA|
+-----+-----------------+
```

5. Control de Transacciones: COMMIT y ROLLBACK

- mysql> INSERT INTO inventario
  -> VALUES (NULL,'IMPRESORA');
- mysql> SELECT * FROM inventario;

```
+-----+----------------+
| num | descripcion    |
+-----+----------------+
|   1 | ARMARIO BLANCO |
|   2 | MESA MADERA    |
|   3 | ORDENADOR      |
|   4 | SILLA GIRATORIA|
|   5 | IMPRSORA       |
+-----+----------------+
```

- mysql> ROLLBACK;
- mysql> SELECT * FROM inventario;

```
+-----+----------------+
| num | descripcion    |
+-----+----------------+
|   1 | ARMARIO BLANCO |
|   2 | MESA MADERA    |
|   3 | ORDENADOR      |
|   4 | SILLA GIRATORIA|
+-----+----------------+
```

5. Control de Transacciones: COMMIT y ROLLBACK

- Para hacer los ejercicios y los ejemplos puede ser interesante modificar esta variable y así disponer de la posibilidad de hacer ROLLBACK. Cada vez que se inicie una sesión estará en modo AUTO_COMMIT que es el valor por defecto y el ROLLBACk no será aplicable.
  - mysql> SET AUTOCOMMIT = 1;
  - mysql> INSERT INTO inventario
    -> VALUES (NULL,'ARCHIVADOR');
  - mysql> SELECT * FROM inventario;

```
+-----+----------------+
| num | descripcion    |
+-----+----------------+
|   1 | ARMARIO BLANCO |
|   2 | MESA MADERA    |
|   3 | ORDENADOR      |
|   4 | SILLA GIRATORIA|
|   6 | ARCHIVADOR     |
+-----+----------------+
```

5. Control de Transacciones: COMMIT y ROLLBACK

- mysql> ROLLBACK;
- mysql> SELECT * FROM inventario;

```
+-----+----------------+
| num | descripcion    |
+-----+----------------+
|   1 | ARMARIO BLANCO |
|   2 | MESA MADERA    |
|   3 | ORDENADOR      |
|   4 | SILLA GIRATORIA|
|   6 | ARCHIVADOR     |
+-----+----------------+
```