



Tema 4

LENGUAJE SQL



Índice

1. Creación de Tablas.
2. Formato genérico para la creación de tablas
 1. Integridad referencial
 2. Formato completo de la creación de tablas
3. Modificación de la definición de una tabla
4. Borrado de una tabla
5. Renombrado de una tabla.
6. Editar las sentencias desde el cliente en modo consola MySQL



1. Creación de Tablas

- Antes de escribir la sentencia para crear una tabla debemos pensar una serie de datos y requisitos que va a ser necesario definir en la creación.
- **Definición de la tabla**
 - Hay que definir:
 - El nombre de la tabla.
 - El nombre de cada columna.
 - El tipo de dato almacenado en cada columna.
 - El tamaño de cada columna.
- **Restricciones en las tablas**
 - Información sobre lo que pueden almacenar las filas de la tabla. Esta información serán las restricciones que almacenamos en las tablas. Son una parte muy importante del modelo relacional pues nos permiten relacionar las tablas entre sí y poner restricciones a los valores que pueden tomar los atributos (columnas) de estas tablas.



1. Creación de Tablas

- Restricciones, llamadas **CONSTRAINTS**, son condiciones que imponemos en el momento de crear una tabla para que los datos se ajusten a una serie de características predefinidas que mantengan su integridad. Se conocen con su nombre en inglés y se refieren a los siguientes conceptos:
 - **NOT NULL**. Exige la existencia de valor en la columna que lleva la restricción.
 - **DEFAULT**. Proporciona un valor por defecto cuando la columna correspondiente no se le da valor en la instrucción de inserción. Este valor por defecto debe ser una constante. No se permiten funciones ni expresiones.
 - **PRIMARY KEY**. Indica una o varias columnas como dato o datos que identifican unívocamente cada fila de la tabla. Solo existe una por tabla y en ninguna fila puede tener valor NULL, por definición. Es obligatoria su existencia en el modelo relacional.



1. Creación de Tablas

- **FOREIGN KEY.** Indica que una determinada columna de una tabla, va a servir para referenciar a otra tabla en la que está definida la misma columna (columna o clave referenciada). El valor de la clave ajena deberá coincidir con uno de los valores de esta clave referenciada o ser NULL. No existe límite en el número de claves ajenas que puede tener una tabla. Como caso particular, una clave ajena puede referenciar a la misma tabla en la que está. Para poder crear una tabla con clave ajena deberá estar previamente creada la tabla maestra en la que la misma columna es clave primaria.
- **UNIQUE.** Indica que esta columna o grupo de columnas debe tener un valor único. También admite valores nulos. Al hacer una nueva inserción se comprobará que el valor es único o NULL. Algunos sistemas gestores de bases de datos relacionales generan automáticamente índices para estas columnas
- **CHECK.** Comprueba si el valor insertado en esa columna cumple una determinada condición.



2. Formato Genérico para la Creación de Tablas

- La sentencia SQL que permite crear tablas es **CREATE TABLE**.
- Comenzaremos con un **formato básico** de creación de tabla al que iremos añadiendo posteriormente otras informaciones.
 - **CREATE TABLE [IF NOT EXISTS] NombreTabla**
(NombreColumna TipoDato [, NombreColumna TipoDato]....);
- Donde:
 - **NombreTabla** es el identificador elegido para la tabla
 - **NombreColumna** es el identificador elegido para cada columna
 - **TipoDato** indica el tipo de dato que se va a almacenar en esa columna.
- El nombre de la tabla debe ser único en la base de datos. Los nombres de columnas deben ser únicos dentro de la tabla.
- Para el nombre de la tabla y de las columnas se elegirán identificadores de acuerdo con las reglas del gestor de la base de datos. Estos identificadores no pueden coincidir con palabras reservadas.

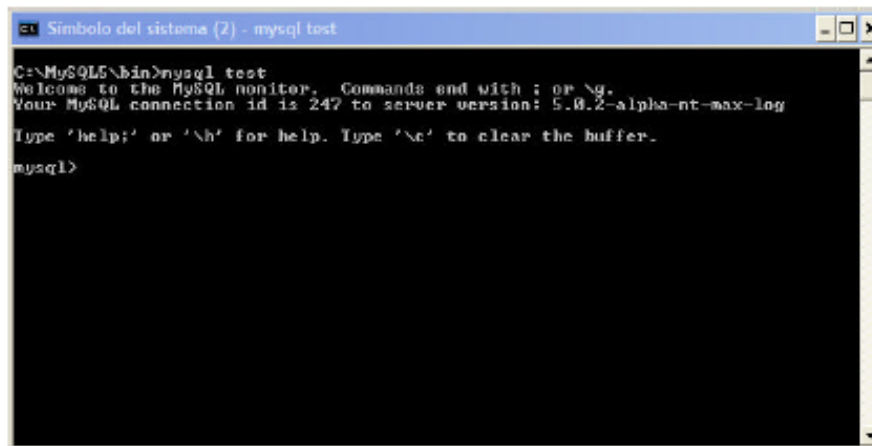


2. Formato Genérico para la Creación de Tablas

- Existirán tantas definiciones de columna como datos diferentes se vayan a almacenar en la tabla que estamos creando, todas ellas separadas por comas.
- La cláusula IF NOT EXISTS previene el posible error generado si existiese una tabla con ese nombre.

6. Editar las sentencias desde el cliente en modo consola MySQL

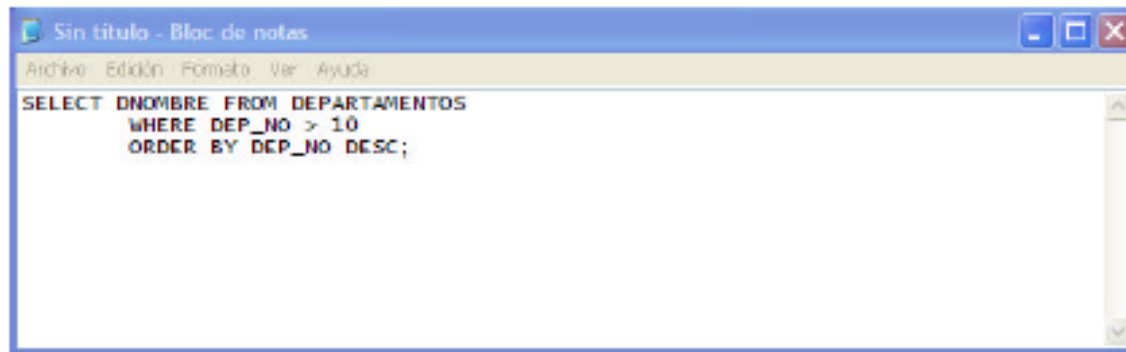
- El programa cliente *mysql* no es muy amigable a la hora de editar sentencias largas de SQL antes de lanzarlas. Es por lo tanto conveniente y aconsejable editar dichas sentencias desde un sencillo editor ASCII (*Notepad* o *Wordpad* en entorno Windows) y una vez editadas lanzar las sentencias mediante el usual procedimiento de “copiar y pegar” al cliente *mysql*.
- Veamos un ejemplo:
 - Paso 1. Se abre una ventana de comandos y se inicia una sesión con el cliente *mysql* sobre la base de datos que estemos trabajando.



```
Símbolo del sistema (2) - mysql test
C:\MySQL\bin>mysql test
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 247 to server version: 5.0.2-alpha-nt-max-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```


6. Editar las sentencias desde el cliente en modo consola MySQL

- Paso 2. Se abre en otra ventana el editor y se edita la sentencia SQL que se va a lanzar al cliente.

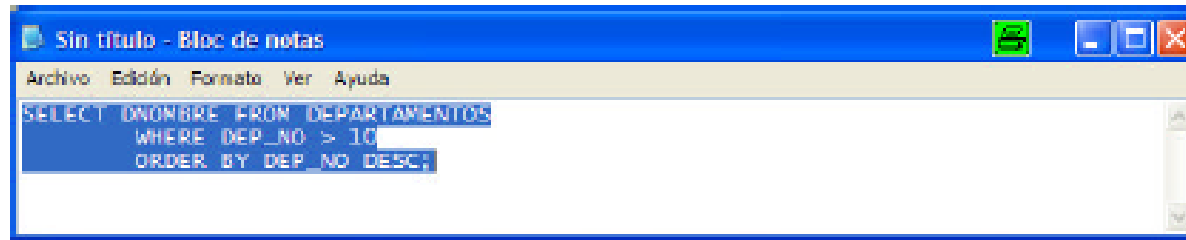


A screenshot of a Windows Notepad application window. The title bar reads 'Sin título - Bloc de notas'. The menu bar includes 'Archivo', 'Edición', 'Formato', 'Ver', and 'Ayuda'. The text area contains the following SQL query:

```
SELECT DNOMBRE FROM DEPARTAMENTOS
WHERE DEP_NO > 10
ORDER BY DEP_NO DESC;
```

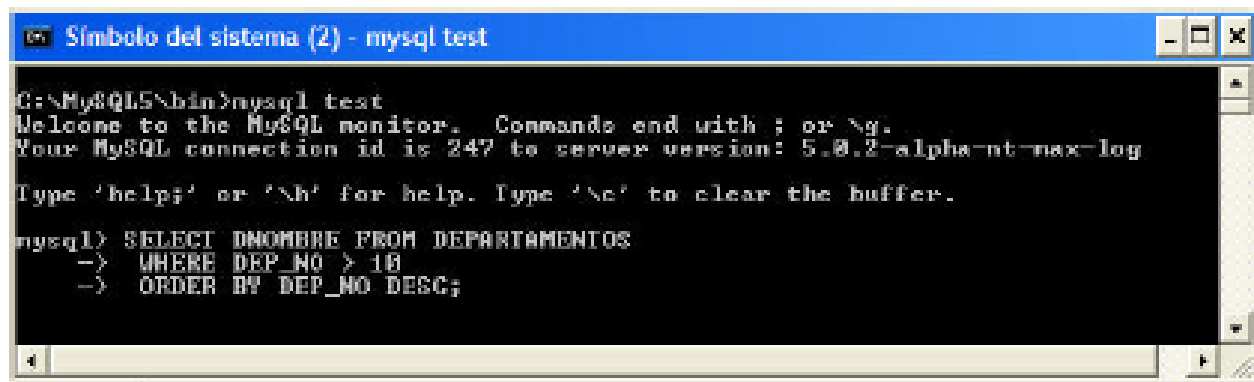
6. Editar las sentencias desde el cliente en modo consola MySQL

- Paso 3. Se marca con el ratón el párrafo con la sentencia para su copia mediante el comando *MenúàEdiciónàCopiar* o bien pulsando *Ctrl+C*



```
Sin título - Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
SELECT DNOMBRE FROM DEPARTAMENTOS
WHERE DEP_NO > 10
ORDER BY DEP_NO DESC;
```

- Paso 4. Se pega la copia anterior pulsando con el botón derecho del ratón sobre cualquier punto de la ventana de comandos y seleccionando la opción *Pegar*.



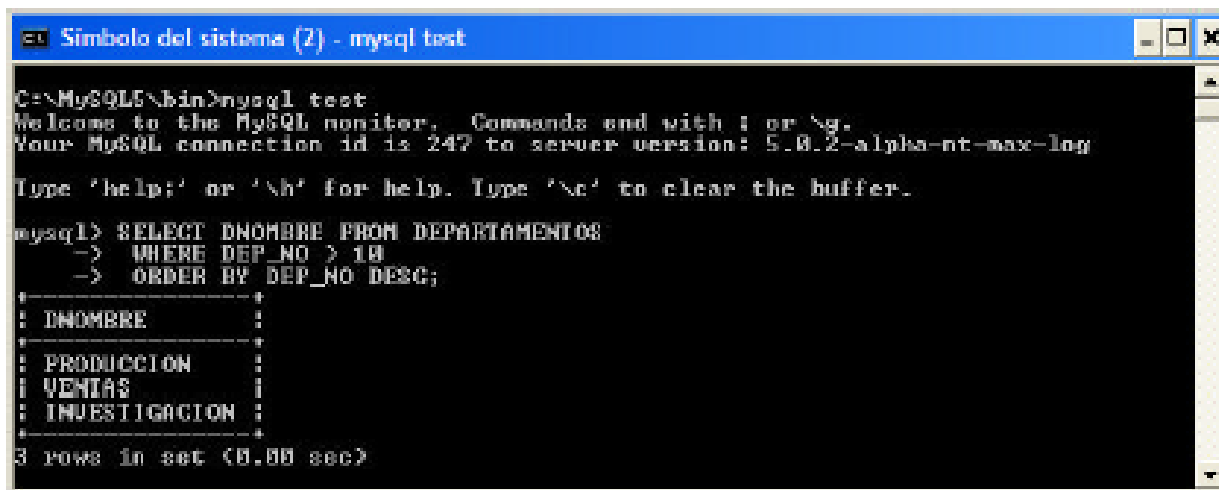
```
Símbolo del sistema (2) - mysql test
G:\MySQL5\bin>mysql test
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 247 to server version: 5.0.2-alpha-nt-max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT DNOMBRE FROM DEPARTAMENTOS
-> WHERE DEP_NO > 10
-> ORDER BY DEP_NO DESC;
```

6. Editar las sentencias desde el cliente en modo consola MySQL

- Paso 5. Se pulsa *Enter* para lanzar la sentencia.



```
Símbolo del sistema (2) - mysql test

C:\MySQL5\bin>mysql test
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 247 to server version: 5.0.2-alpha-nt-max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT DNOMBRE FROM DEPARTAMENTOS
      -> WHERE DEP_NO > 10
      -> ORDER BY DEP_NO DESC;
+-----+
| DNOMBRE |
+-----+
| PRODUCCION |
| VENIAS |
| INVESTIGACION |
+-----+
3 rows in set (0.00 sec)
```



2. Formato Genérico para la Creación de Tablas

- **Ejemplos:**

- Realizamos un ejemplo de una biblioteca en la que queremos guardar los datos de los socios en una tabla socios y los préstamos que se realizan en una tabla prestamos. Empezaremos con los formatos básicos e iremos añadiendo cláusulas.
 - Crear una tabla socios con los datos de los socios:
 - Numero de socio: Número entero de 4 dígitos
 - Apellidos del socios: Cadena de 14 caracteres máximo
 - Teléfono: Cadena de 9 caracteres
 - Fecha de alta como socio: Fecha
 - Dirección: Cadena de 20 caracteres máximo
 - Código postal: Número entero de 5 dígitos



2. Formato Genérico para la Creación de Tablas

- `mysql> CREATE TABLE SOCIOS`
 - > `(num_socio INT(4),`
 - > `apellidos VARCHAR(14),`
 - > `telefono CHAR(9),`
 - > `fecha_alta DATE,`
 - > `direccion VARCHAR(20),`
 - > `codigo_postal INT(5));`
- El campo teléfono lo creamos tipo CHAR en lugar de VARCHAR porque siempre tendrá 9 caracteres



2. Formato Genérico para la Creación de Tablas

- Crear una tabla prestamos para guardar los préstamos hechos a los socios con los datos:
 - Número del préstamo: Número entero de 2 dígitos
 - Código del socio: Número entero de 4 dígitos
 - `mysql> CREATE TABLE PRESTAMOS`
 - `-> (num_prestamo INT(2),`
 - `-> num_socio INT(4));`



2. Formato Genérico para la Creación de Tablas

- **Crear, usar y acceder a bases de datos en MySQL**

- Antes de crear la base de datos vamos a ver cuáles son las bases de datos existentes en el servidor.
 - `mysql> SHOW DATABASES;`
- Para crear estas tablas, primero debemos crear la base de datos, con la siguiente instrucción:
 - `mysql> CREATE DATABASE Biblioteca;`
- Luego habrá que abrirla (usarla), con la siguiente instrucción:
 - `mysql> USE Biblioteca;`
- Ahora ya se pueden lanzar sentencias `CREATE TABLE` para crear tablas sobre la base de datos Biblioteca que hemos creado y hemos accedido.
- Para ver un listado con las tablas dentro de una determinada base de datos usamos la siguiente instrucción:
 - `mysql> show tables;`

2. Formato Genérico para la Creación de Tablas

- **Crear, usar y acceder a bases de datos en MySQL**
 - Para ver la descripción de una tabla en sus columnas y tipos asociados, además de poder ver los atributos que aceptan nulos y las opciones por defecto debemos usar el comando DESC o DESCRIBE que realmente es un sinónimo de la sentencia SHOW COLUMNS FROM.
 - **mysql> desc socios;**

```
MySQL Command Line Client
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.31-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use biblioteca
Database changed
mysql> desc socios;
```

Field	Type	Null	Key	Default	Extra
num_socio	int(4)	NO	PRI	NULL	
apellidos	varchar(14)	YES	UNI	NULL	
telefono	char(9)	NO		NULL	
fecha_alta	date	YES		2000-01-01	
direccion	varchar(20)	YES		NULL	
codigo_postal	int(5)	YES		NULL	

```
6 rows in set (0.17 sec)
```

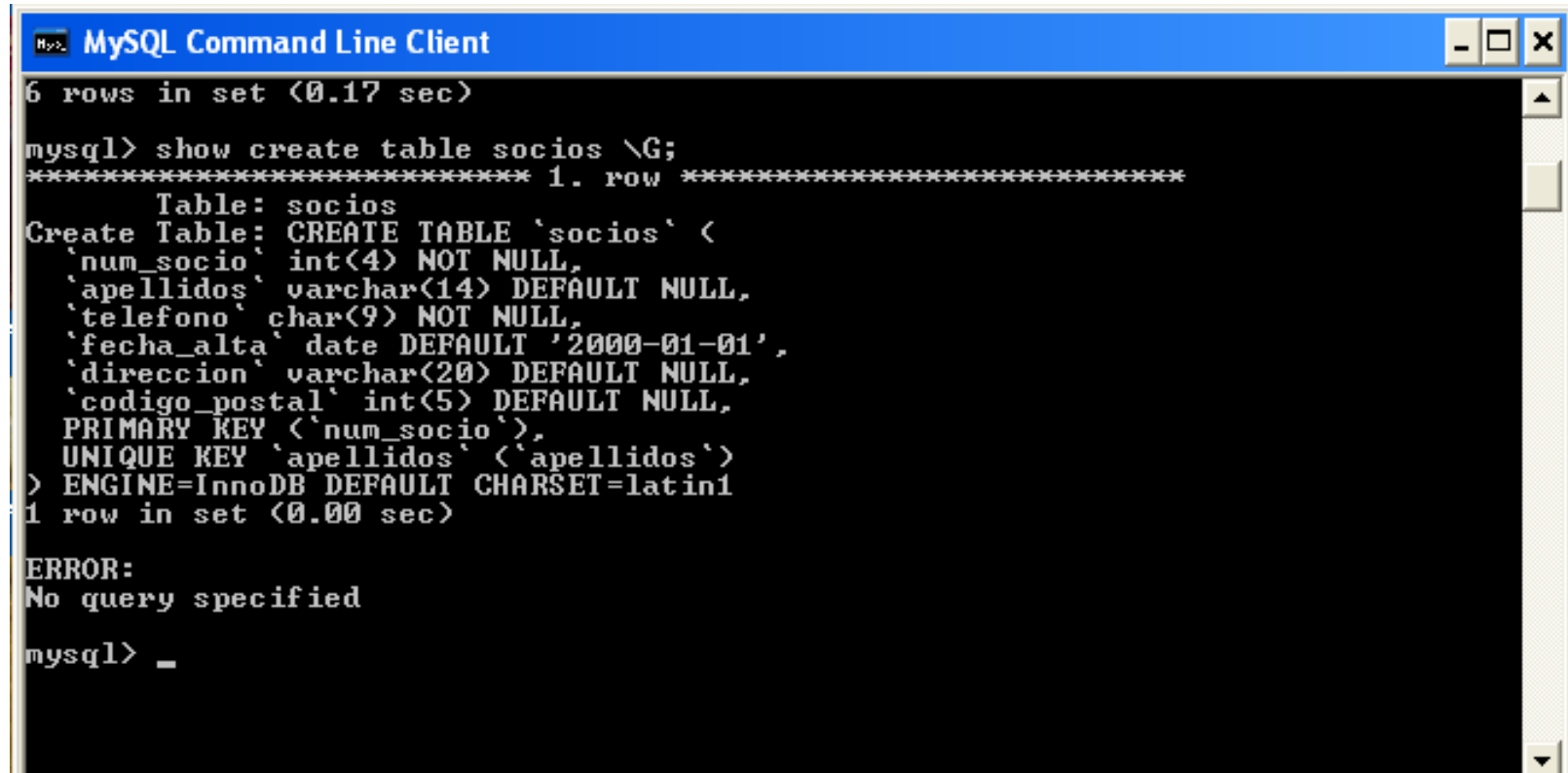



2. Formato Genérico para la Creación de Tablas

- **Crear, usar y acceder a bases de datos en MySQL**
 - Es muy interesante también poder ver la sentencia de creación de tabla asociada a una determinada tabla de la base de datos. Es una información que puede complementar la proporcionada por el comando DESC.
 - `mysql> show create table socios \G`
 - La opción `\G` permite visualizar la salida de la consulta en una sola fila.

2. Formato Genérico para la Creación de Tablas

- **Crear, usar y acceder a bases de datos en MySQL**



```
MySQL Command Line Client
6 rows in set (0.17 sec)

mysql> show create table socios \G;
***** 1. row *****
      Table: socios
Create Table: CREATE TABLE `socios` (
  `num_socio` int(4) NOT NULL,
  `apellidos` varchar(14) DEFAULT NULL,
  `telefono` char(9) NOT NULL,
  `fecha_alta` date DEFAULT '2000-01-01',
  `direccion` varchar(20) DEFAULT NULL,
  `codigo_postal` int(5) DEFAULT NULL,
  PRIMARY KEY (`num_socio`),
  UNIQUE KEY `apellidos` (`apellidos`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)

ERROR:
No query specified

mysql> _
```



2. Formato Genérico para la Creación de Tablas

- **Formatos para la creación de tablas con la definición de restricciones**
- Los valores por defecto pueden ser: constantes, funciones SQL o las variables USER o SYSDATE. Las restricciones pueden definirse de dos formas, que llamaremos
 - **Restriccion1.** Definición de la restricción a nivel de columna
 - **Restriccion2.** Definición de la restricción a nivel de tabla
- Estas restricciones , llamadas CONSTRAINTS se pueden almacenar con o sin nombre. Si no se lo damos nosotros lo hará el sistema siguiendo una numeración correlativa, que es poco representativa.
- Es conveniente darle un nombre, para después poder referirnos a ellas si las queremos borrar o modificar. Estos nombres que les damos a las CONSTRAINTS deben ser significativos para hacer mas fácil las referencias.
Por ejemplo:



2. Formato Genérico para la Creación de Tablas

- *pk_NombreTabla* para PRIMARY KEY
- *fk_NombreTabla1_NombreTabla2* FOREIGN KEY donde NombreTabla1 es la tabla donde se crea y NombreTabla2 es la tabla a la que referencia.
- *uq_NombreTabla_NombreColumna* para UNIQUE
- Definimos cada restricción al mismo tiempo que definimos la columna correspondiente.
 - CREATE TABLE [IF NOT EXISTS] NombreTabla
(NombreColumna TipoDato [**Restriccion1**]
[, NombreColumna TipoDato [**Restriccion1**]]);
- donde ***Restriccion1.....*** es la definición de la restricción a nivel de columna
- Las restricciones solo se pueden definir de esta forma si afectan a una sola columna, la que estamos definiendo es ese momento.



2. Formato Genérico para la Creación de Tablas

- **Definición de los diferentes tipos de CONSTRAINTS a nivel de columna (Restriccion1)**
 - **CLAVE PRIMARIA:** PRIMARY KEY
 - **POSIBILIDAD DE NULO:** NULL | NOT NULL
 - **VALOR POR DEFECTO:** DEFAULT ValorDefecto
 - **UNICIDAD:** UNIQUE
 - **COMPROBACION DE VALORES:** CHECK (Expresion)
 - **Nota:** Esta cláusula de SQL estándar, en MySQL en la versión 5 está permitida pero no implementada



2. Formato Genérico para la Creación de Tablas

- **CLAVE AJENA:** REFERENCES NombreTabla [(NombreColumna)]
 - **Notación:** el nombre de tabla referenciada es el nombre de la tabla a la que se va a acceder con la clave ajena. Si la columna que forma la clave referenciada en dicha tabla no tiene el mismo nombre que en la clave ajena, debe indicarse su nombre detrás del de la tabla referenciada y dentro del paréntesis. Si los nombres de columnas coinciden en la clave ajena y en la primaria, no es necesario realizar esta indicación.



2. Formato Genérico para la Creación de Tablas

- **Ejemplo:**

- **PRIMARY KEY.** El numero de socio en la *tabla socios*

- mysql> CREATE TABLE SOCIOS

- > (num_socio INT(4) PRIMARY KEY,

- > apellidos VARCHAR(14),

- > telefono CHAR(9),

- > fecha_alta DATE,

- > direccion VARCHAR(20),

- > codigo_postal INT(5));



2. Formato Genérico para la Creación de Tablas

- **NOT NULL.** La columna teléfono es obligatoria en la tabla socios, nunca irá sin información
 - `mysql> CREATE TABLE SOCIOS`
 - > `(num_socio INT(4) PRIMARY KEY,`
 - > `apellidos VARCHAR(14),`
 - > `telefono CHAR(9) NOT NULL,`
 - > `fecha_alta DATE,`
 - > `direccion VARCHAR(20),`
 - > `codigo_postal INT(5));`



2. Formato Genérico para la Creación de Tablas

- **DEFAULT.** En ausencia de valor el campo fecha _ alta tomará el valor de 1 de enero de 2000.
 - mysql> CREATE TABLE SOCIOS
 - > (num_socio INT(4) PRIMARY KEY,
 - > apellidos VARCHAR(14),
 - > telefono CHAR(9) NOT NULL,
 - > fecha_alta DATE DEFAULT '2000-01-01',
 - > direccion VARCHAR(20),
 - > codigo_postal INT(5));



2. Formato Genérico para la Creación de Tablas

- **UNIQUE.** La columna apellidos será única en la tabla socios.

- `mysql> CREATE TABLE SOCIOS`

- > `(num_socio INT(4) PRIMARY KEY,`
 - > `apellidos VARCHAR(14) UNIQUE,`
 - > `telefono CHAR(9) NOT NULL,`
 - > `fecha_alta DATE DEFAULT '2000-01-01',`
 - > `direccion VARCHAR(20),`
 - > `codigo_postal INT(5));`

.



2. Formato Genérico para la Creación de Tablas

- **CHECK.** Se comprobará que la columna código_postal corresponde a Madrid (valores entre 28000 y 28999)
 - mysql> CREATE TABLE SOCIOS
 - > (num_socio INT(4) PRIMARY KEY,
 - > apellidos VARCHAR(14) UNIQUE,
 - > telefono CHAR(9) NOT NULL,
 - > fecha_alta DATE DEFAULT '2000-01-01',
 - > direccion VARCHAR(20),
 - > codigo_postal INT(5) CHECK (codigo_postal BETWEEN 28000 AND 28999));



2. Formato Genérico para la Creación de Tablas

- **AUTO INCREMENTO.** Crearemos una tabla con una columna AUTO_INCREMENT y posteriormente (siguiente tema) insertaremos valores.

- ```
mysql> CREATE TABLE inventario
-> (num INT(2) AUTO_INCREMENT PRIMARY KEY,
-> descripcion VARCHAR(15));
```

- Para borrar las tablas que ya hemos creado para crear las nuevas con las restricciones usaremos la siguiente instrucción:

- ```
DROP TABLE IF EXISTS SOCIOS;
```



2. Formato Genérico para la Creación de Tablas

- Veamos el caso del **formato para la creación de tablas con restricciones definidas a nivel de tabla.**
- En este caso definimos todas las restricciones al final de la sentencia, una vez terminada la definición de las columnas.
 - `CREATE TABLE [IF NOT EXISTS] NombreTabla
 (NombreColumna TipoDato [, NombreColumna TipoDato.....]
 [Restriccion2 [, Restriccion2].....];`
- donde ***Restriccion2.....*** es la definición de la restricción a nivel de tabla
- Las restricciones siempre se pueden definir de esta forma tanto si afectan a una sola columna como a varias columnas y puede darse un nombre a cada una de las restricciones.



2. Formato Genérico para la Creación de Tablas

- **Definición de los diferentes tipos de CONSTRAINTS a nivel de tabla (Restriccion2)**
 - **PRIMARY KEY:**
 - [CONSTRAINT [NombreConstraint]]
PRIMARY KEY (Nombrecolumna [,NombreColumna....])
 - **UNIQUE:**
 - [CONSTRAINT [NombreConstraint]] UNIQUE (NombreColumna
[,NombreColumna...])
 - **CHECK:**
 - [CONSTRAINT [NombreConstraint]] CHECK (Expresion)



2. Formato Genérico para la Creación de Tablas

■ **FOREIGN KEY:**

- [CONSTRAINT [NombreConstraint]]
- FOREIGN KEY (NombreColumna[, NombreColumna...])
- REFERENCES (NombreTabla [NombreColumna [, NombreColumna.....]])

- **Notación:** los nombres de columna o columnas que siguen a la cláusula FOREIGN KEY es aquella o aquellas que están formando la clave ajena. Si hay más de una se separan por comas. El nombre de tabla referenciada es el nombre de la tabla a la que se va a acceder con la clave ajena. Si la columna o columnas que forman la clave referenciada en dicha tabla no tienen el mismo nombre que en la clave ajena, debe indicarse su nombre detrás del de la tabla referenciada y dentro de paréntesis. Si son más de una columna se separan por comas. Si los nombres de columnas coinciden en la clave ajena y en la primaria, no es necesario realizar esta indicación.



2. Formato Genérico para la Creación de Tablas

- Veamos algunos ejemplos:
 - **PRIMARY KEY.** El número de socio será la clave primaria en la *tabla socios*. Será obligatoriamente no nulo y único.
 - `mysql> CREATE TABLE Socios`
 - > `(num_socio INT(4),`
 - > `apellidos VARCHAR(14),`
 - > `telefono CHAR(9),`
 - > `fecha_alta DATE,`
 - > `direccion VARCHAR(20),`
 - > `codigo_postal INT(5),`
 - > `CONSTRAINT PK_SOCIOS PRIMARY KEY (num_socio));`



2. Formato Genérico para la Creación de Tablas

- **UNIQUE.** El campo apellidos es único. Tendrá valores diferentes en cada fila o el valor nulo
 - mysql> CREATE TABLE Socios
 - > (num_socio INT(4),
 - > apellidos VARCHAR(14),
 - > telefono CHAR(9),
 - > fecha_alta DATE,
 - > direccion VARCHAR(20),
 - > codigo_postal INT(5),
 - > CONSTRAINT PK_SOCIOS PRIMARY KEY (num_socio),
 - > CONSTRAINT UQ_APELLIDOS UNIQUE (apellidos));



2. Formato Genérico para la Creación de Tablas

- **CHECK.** La columna `codigo_postal` no admitirá como válidas aquellas filas en las que el código postal no tenga valores entre 28.000 y 28.999
 - `mysql> CREATE TABLE Socios`
 - `-> (num_socio INT(4),`
 - `-> apellidos VARCHAR(14),`
 - `-> telefono CHAR(9),`
 - `-> fecha_alta DATE,`
 - `-> direccion VARCHAR(20),`
 - `-> codigo_postal INT(5),`
 - `-> CONSTRAINT PK_SOCIOS PRIMARY KEY (num_socio),`
 - `-> CONSTRAINT UQ_APELLIDOS UNIQUE (apellidos),`
 - `-> CONSTRAINT CK_CODIGO`
 - `-> CHECK (codigo_postal BETWEEN 28000 AND 28999));`



2. Formato Genérico para la Creación de Tablas

- **FOREIGN KEY.** El número de socio en la *tabla prestamos* será clave ajena referenciando a la columna correspondiente de la tabla socios
 - mysql> CREATE TABLE Prestamos
 - > (num_prestamo INT(2),
 - > num_socio INT(4) ,
 - > CONSTRAINT PK_PRESTAMOS PRIMARY KEY (num_prestamo),
 - > CONSTRAINT FK_PRESTAMOS FOREIGN KEY (num_socio) REFERENCES Socios (num_socio));
- Borrar las tablas que ya hemos creado para volverlas a crear usando las restricciones a nivel de tablas.



2.1. Integridad Referencial

- La definición de claves ajenas nos permiten mantener la integridad referencial en una base de datos relacional. Hemos dicho que la columna o columnas definidas como clave ajena deben tomar valores que se correspondan con un valor existente de la clave referenciada. La pregunta es: ¿qué sucede si queremos borrar o modificar un valor de la clave primaria referenciada? Pues que el sistema debe impedirnos realizar estas acciones pues dejaría de existir la integridad referencial.
- Por ejemplo si tenemos
 - `CREATE TABLE departamentos`
 - `(num_dep INT(4)`
 - `CONSTRAINT pk_departamentos PRIMARY KEY.....);`
 - `CREATE TABLE empleados`
 - `(...num_dep INT(4) CONSTRAINT fk_empleados_departamentos`
`REFERENCES departamentos(num_dep)....);`
 - En este caso `empleados.num_dep` solo puede tomar valores que existan en `departamentos.num_dep` pero ¿que sucede si queremos borrar o modificar un valor de `departamentos.num_dep`? El sistema no nos lo permitirá si existen filas con ese valor en la tabla de la clave ajena.



2.1. Integridad Referencial

- Sin embargo, en ocasiones, será necesario hacer estas operaciones. Para mantener la integridad de los datos, al borrar (DELETE), modificar (UPDATE) una fila de la tabla referenciada, el sistema no nos permitirá llevarlo a cabo si existe una fila con el valor referenciado. También se conoce como RESTRICT. Es la opción por defecto, pero existen las siguientes opciones en la definición de la clave ajena:
 - **CASCADE.** El borrado o modificación de una fila de la tabla referenciada lleva consigo el borrado o modificación en cascada de las filas de la tabla que contiene la clave ajena. Es la más utilizada.
 - **SET NULL.** El borrado o modificación de una fila de la tabla referenciada lleva consigo poner a NULL los valores de las claves ajenas en las filas de la tabla que referencia.
 - **SET DEFAULT.** El borrado o modificación de una fila de la tabla referenciada lleva consigo poner un valor por defecto en las claves ajenas de la tabla que referencia.
 - **NO ACTION.** El borrado o modificación de una fila de la tabla referenciada solo se produce si no existe ese valor en la tabla que contiene la clave ajena. Tiene el mismo efecto que RESTRICT.



2.1. Integridad Referencial

- **Formato de la definición de clave ajena con opciones de referencia**
- REFERENCES NombreTabla [(NombreColumna [, NombreColumna])]
[ON DELETE {CASCADE | SET NULL | NO ACTION | SET DEFAULT | RESTRICT}]
[ON UPDATE {CASCADE | SET NULL | NO ACTION | SET DEFAULT | RESTRICT}]
- por **ejemplo:**
 - CREATE TABLE empleados
 - (... dept_no NUMBER(4) CONSTRAINT FK_EMPLEADOS_DEPARTAMENTOS
REFERENCES departamentos(dept_no)
ON DELETE SET NULL
ON UPDATE CASCADE);
- Esto quiere decir que el sistema pondrá nulos en dept_no de la tabla empleados si se **borra** el valor correspondiente en departamentos y **modificará** el valor de dept_no en la tabla empleados con el nuevo valor si se modifica la columna dept_no en la tabla departamentos.
- Veremos ejemplos más adelante



2.2. Formato completo de la creación de tablas

- **CREATE TABLE [IF NOT EXISTS] NombreTabla**
 (NombreColumna TipoDato [Restriccion1]
 [, NombreColumna TipoDato [Restriccion1.....]
 [Restriccion2 [, Restriccion2....]);
- ***Notación:*** *los diferentes formatos de definición de restricciones, restricción1 y restricción2, están entre corchetes porque son opcionales, pudiéndose elegir entre ambos sólo si la restricción afecta a una sola columna.*



2.2. Formato completo de la creación de tablas

- Vamos a crear la tabla socios y prestamos con el formato completo. Algunas CONSTRAINTS las creamos a nivel de columna y otras de tabla:
 - `mysql> CREATE TABLE Socios`
 - `-> (num_socio INT(4),`
 - `-> apellidos VARCHAR(14),`
 - `-> telefono CHAR(9) NOT NULL,`
 - `-> fecha_alta DATE DEFAULT '2000-01-01',`
 - `-> direccion VARCHAR(20),`
 - `-> codigo_postal INT(5),`
 - `-> CONSTRAINT PK_SOCIOS PRIMARY KEY(num_socio),`
 - `-> CONSTRAINT UQ_UNIQUE UNIQUE(apellidos),`
 - `-> CONSTRAINT CK_CODIGO`
 - `-> CHECK (codigo_postal BETWEEN 28000 AND 28999));`



2.2. Formato completo de la creación de tablas

- `mysql> CREATE TABLE prestamos`
 - `-> (num_prestamo INT(2) PRIMARY KEY,`
 - `-> num_socio INT(4) ,`
 - `-> CONSTRAINT FK_SOCIO_PRESTAMOS FOREIGN KEY (num_socio)`
`REFERENCES Socios(num_socio)`
`ON UPDATE CASCADE`
`ON DELETE SET NULL);`



3. Modificación de la definición de una tabla

- Una vez que hemos creado una tabla, a menudo se presenta la necesidad de tener que modificarla. La sentencia SQL que realiza esta función es **ALTER TABLE**.
- La especificación de la modificación es parecida a la de la sentencia CREATE pero varía según el objeto SQL del que se trate.
 - **ALTER TABLE NombreTabla**
EspecificacionModificacion [, EspecificacionModificacion.....]
- donde **NombreTabla:** nombre de la tabla se desea modificar. Y **EspecificacionModificacion:** las modificaciones que se quieren realizarse sobre la tabla



3. Modificación de la definición de una tabla

- Las modificaciones que se pueden realizar sobre una tabla son:
 - Añadir una nueva columna
 - Añadir un nueva restricción
 - Borrar una columna
 - Borrar una restricción
 - Modificar una columna sin cambiar su nombre
 - Modificar una columna y cambiar su nombre
 - Renombrar la tabla



3. Modificación de la definición de una tabla

- **Añadir una nueva columna**

- `ADD [COLUMN] NombreColumna TipoDato [Restriccion1]`
- Puede añadirse una nueva columna y todas las restricciones, salvo NOT NULL. La razón es que esta nueva columna tendrá los valores NULL al ser creada.

- **Añadir una constraint**

- `ADD [CONSTRAINT [NombreConstraint]]
PRIMARY KEY (NombreColumna [, NombreColumna...])`
- `ADD [CONSTRAINT [NombreConstraint]]
FOREIGN KEY (NombreColumna [, NombreColumna...])
REFERENCES NombreTabla[(NombreColumna[,NombreColumna...])]`
- `ADD [CONSTRAINT [NombreConstraint]]
UNIQUE (NombreColumna [, NombreColumna...])`



3. Modificación de la definición de una tabla

- **Borrar una columna**

- DROP [COLUMN] NombreColumna

- **Borrar una constraint**

- DROP PRIMARY KEY
- DROP FOREIGN KEY NombreConstraint

- **Modificar una columna sin cambiar su nombre**

- MODIFY [COLUMN] NombreColumna TipoDato [Restriccion1]

- **Modificar la definición de una columna y su nombre**

- CHANGE [COLUMN] NombreColumnaAntiguo
NombreColumnaNuevo TipoDatos [Restriccion1]

- **Renombrar la tabla**

- RENAME [TO] NombreTablaNuevo



3. Modificación de la definición de una tabla

■ Ejemplos

- Añadir la columna para la dirección de correo electrónico, `direccion_correo`, a la tabla de Socios con un tipo de dato alfanumérico de 20 caracteres.
 - `mysql> ALTER TABLE Socios ADD (direccion_correo varchar(20));`
- Añadir en la tabla Socios la restricción UNIQUE para la columna telefono.
 - `mysql> ALTER TABLE Socios`
 - `-> ADD CONSTRAINT uq_socios_telefono UNIQUE(telefono);`
- Borrar la columna `fecha_alta` de la tabla Socios
 - `mysql> ALTER TABLE Socios`
 - `-> DROP COLUMN fecha_alta;`



3. Modificación de la definición de una tabla

- Borrar la clave primaria de la tabla Socios
 - `mysql> ALTER TABLE Socios`
 `-> DROP PRIMARY KEY;`
- Modificar la tabla Socios para poner NOT NULL en la columna *apellidos*.
 - `mysql> ALTER TABLE Socios`
 `-> MODIFY apellidos VARCHAR(14) NOT NULL;`
- Modificar la tabla Socios cambiándole el nombre a la columna apellidos por apellidos_a incrementar de 14 a 20 caracteres el tamaño.
 - `mysql> ALTER TABLE Socios`
 `-> CHANGE apellidos apellidos_a VARCHAR(20);`
- Cambiar el nombre de la tabla Socios por Sociosdos
 - `mysql> ALTER TABLE Socios`
 `-> RENAME TO Sociosdos`



4. Borrado de una tabla

- Como se ha dicho anteriormente, para borrar una tabla el formato utilizado es:
 - **DROP TABLE [IF EXISTS] NombreTabla**
- La cláusula IF EXISTS previene los errores que puedan producirse si no existe la tabla que queremos borrar



5. Renombrado de una tabla

- Permite cambiar de nombre una tabla, asignándole un nombre nuevo y el nombre antiguo. El formato será:
 - **RENAME TABLE NombreTablaAntiguo TO NombreTablaNuevo**
[, NombreTablaAntiguo TO NombreTablaNuevo]
- **Notación:** se pueden renombrar varias tablas en una sentencia por eso van entre corchetes las siguientes tablas a renombrar.
- Donde:
 - **NombreTablaAntiguo:** es el nombre antiguo de la tabla, que debe existir
 - **NombreTablaNuevo:** es el nombre nuevo de la tabla, que no debe existir.
- Permite renombrar en la misma instrucción una o varias tablas.