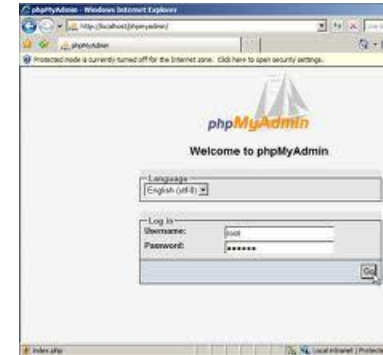
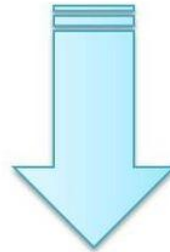


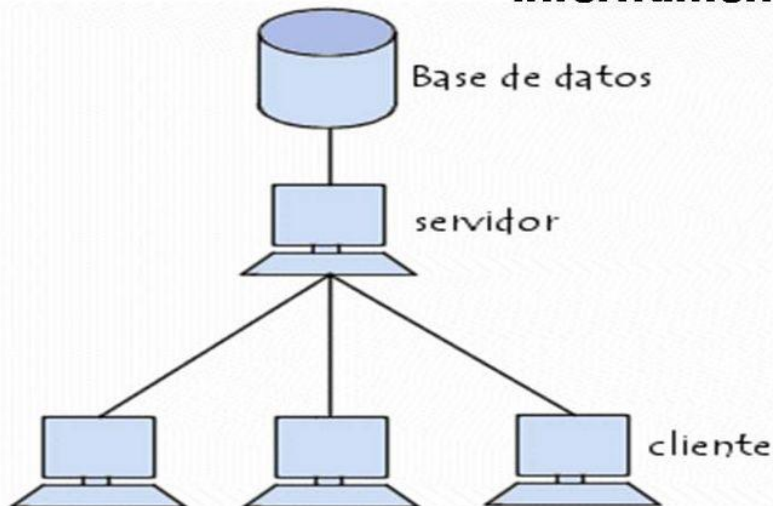
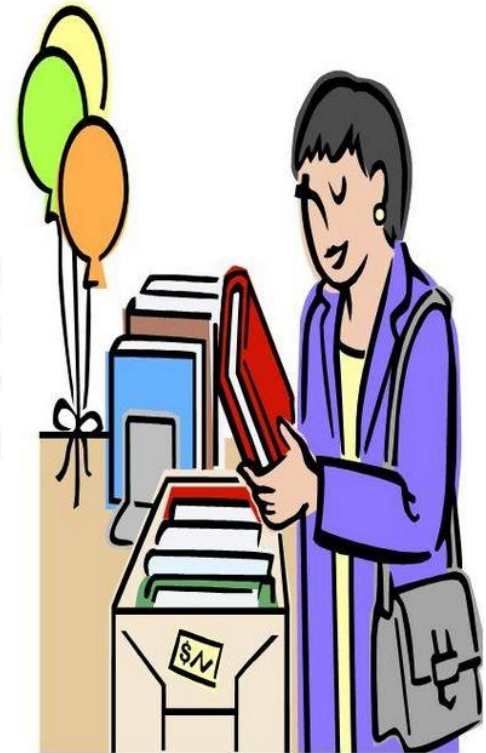
# Tema 4 : Diseño físico de Bases de Datos. DDL.



# BASES DE DATOS



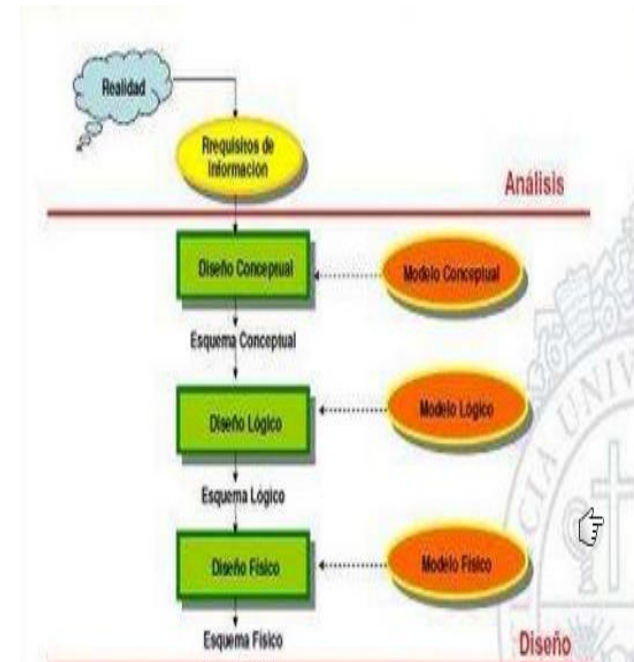
**Es el conjunto de informaciones almacenadas en un soporte legible por ordenador y organizadas internamente por registros**



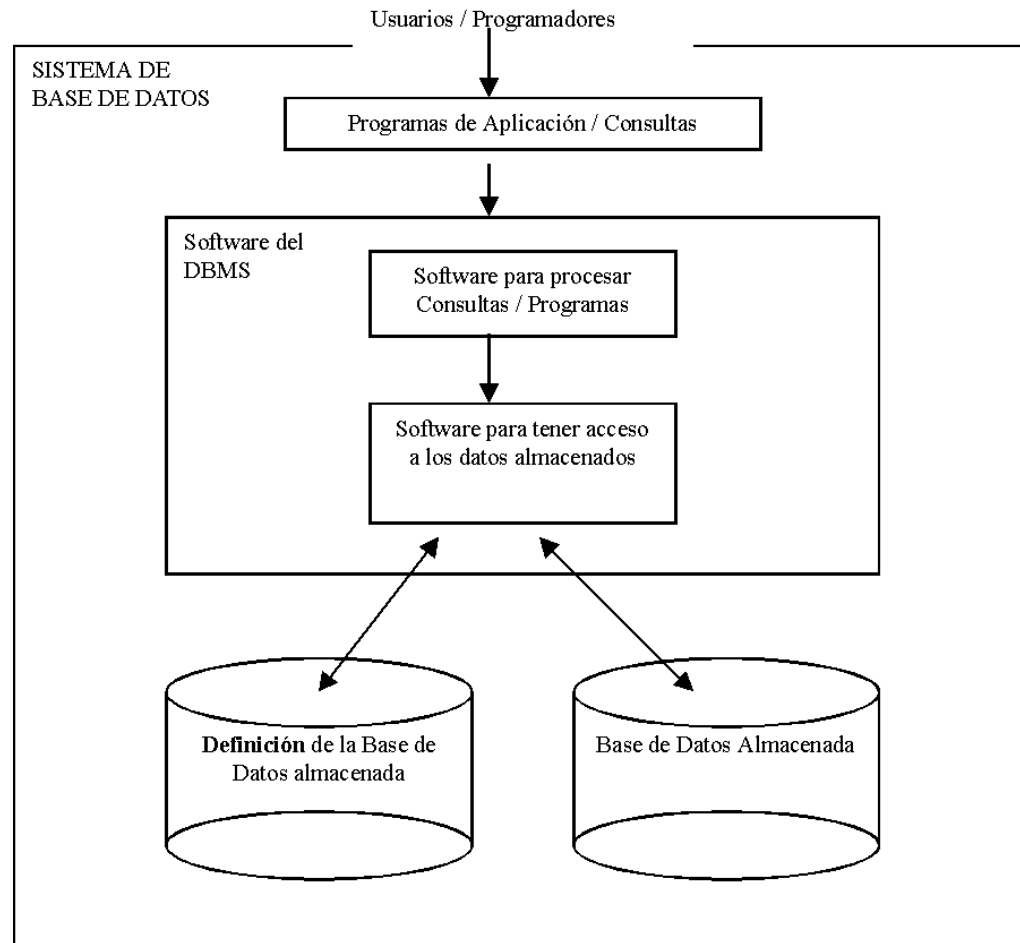
# Diseño Físico

- La fase de diseño físico es aquella en la que se transforma el esquema lógico (reflejado en un esquema relacional) en un conjunto de *estructuras propias del Sistema Gestor de Bases de Datos concreto*.
- El objetivo que se persigue es conseguir el mejor rendimiento al menor coste.
- A partir del esquema lógico y las restricciones impuestas por la organización, hay que:
  - Determinar la representación de los datos.
  - Establecer los privilegios de los usuarios.
  - Seleccionar los métodos de acceso.

Estas tareas son **responsabilidad del administrador**.



# Diseño Físico



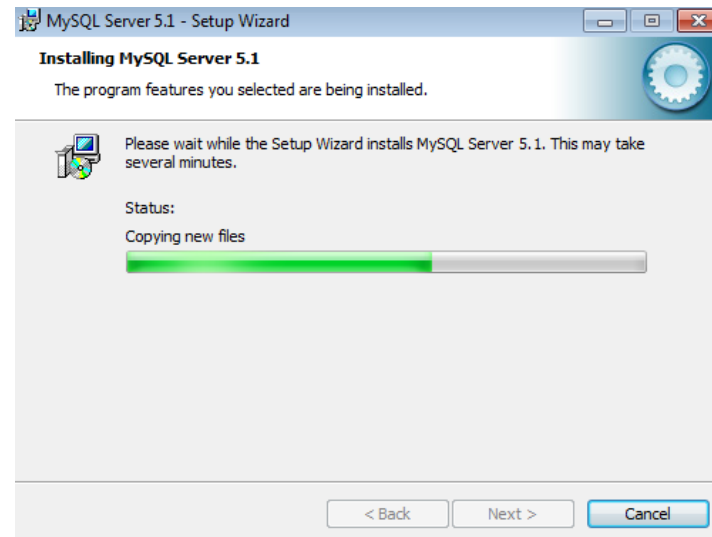
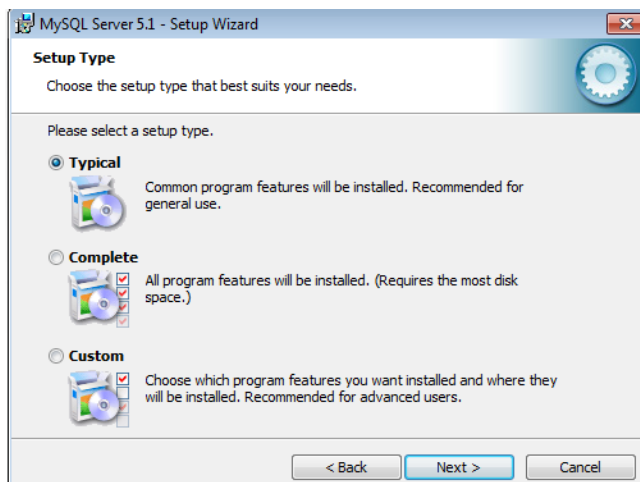


# MySQL

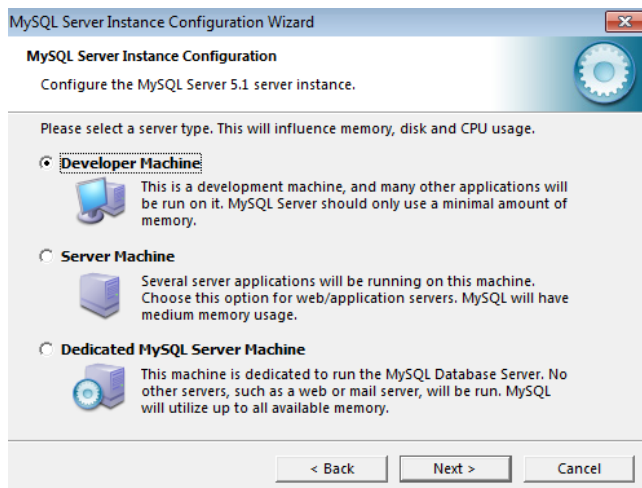
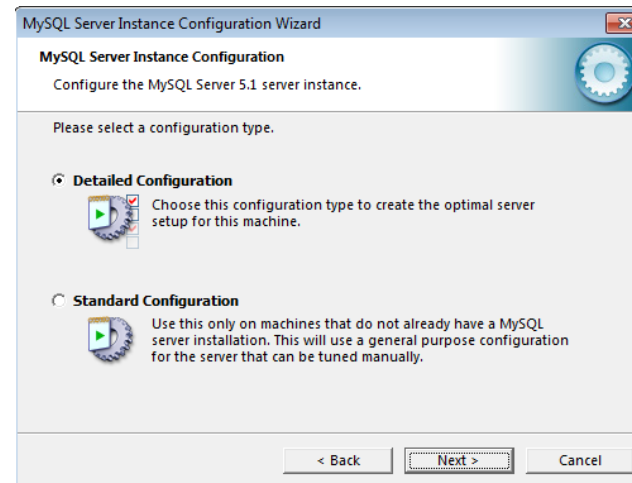
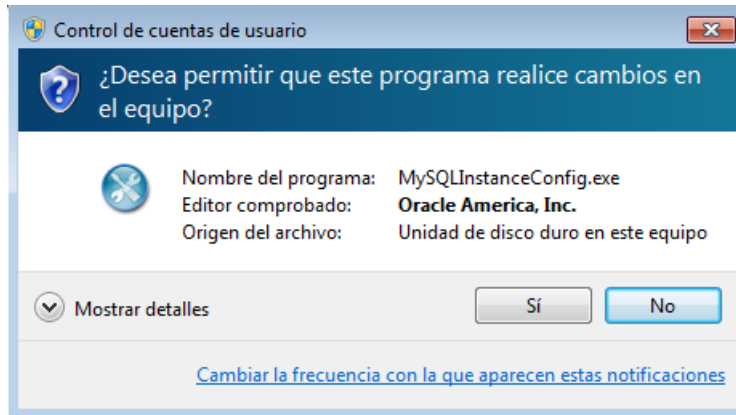


- **MySQL** es un **sistema de gestión de bases de datos relacional**, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB — desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.
- Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.
- MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante monitorizar de antemano el rendimiento para detectar y corregir errores tanto de SQL como de programación.

# Instalación mysql



# Configuración mysql



# Conexión con el servidor

- Para conectarse al servidor, generalmente se introducirá un nombre de usuario y una contraseña. Si el servidor se está ejecutando en un ordenador distinto de donde se está estableciendo la conexión, también se deberá especificar el nombre de host.

```
#mysql -h host -u user -p  
Enter password: *****
```

- Después de haberse conectado, puede desconectarse en cualquier momento escribiendo QUIT (o \q) en el prompt:

```
mysql>quit
```



# Introducir Consultas

- Una vez que se ha logrado la conexión con el servidor ya se puede empezar a hacer consultas. Por ejemplo:

```
mysql> select (2*3+9)
-> ;
+-----+
| (2*3+9) |
+-----+
|      15 |
+-----+
1 row in set (0.00 sec)
mysql>
```

```
mysql> select version(), current_date;
+-----+-----+
| version() | current_date |
+-----+-----+
| 5.1.73-community | 2014-09-30 |
+-----+-----+
1 row in set (0.00 sec)
```

- Si durante la introducción de un comando se decide que no se quiere ejecutar, se cancela \c:

Prompt	Significado
mysql>	Listo para un nuevo comando.
->	Esperando la siguiente línea en un comando de múltiples líneas.
'>	Esperando la siguiente línea, se encuentra abierta una cadena que comienza con apostrofo ('').
">	Esperando la siguiente línea, se encuentra abierta una cadena que comienza con comillas dobles ("").
`>	Esperando la siguiente línea, se encuentra abierta una cadena que comienza con tilde (`).
/*>	Esperando la siguiente línea, se encuentra abierto un comentario que comienza con /*.

# Introducir consultas desde un fichero

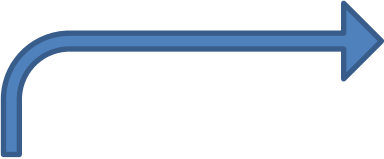
Se puede crear un fichero de texto con las órdenes sql que se quiera ejecutar y luego desde mysql ejecutar dicho fichero:

```
C:\Users\alumno>type ej.sql  
select user();  
select current_date();  
select (2*9+4);  
♦
```



```
mysql> source ej.sql  
+-----+  
| user() |  
+-----+  
| root@localhost |  
+-----+  
1 row in set (0.00 sec)  
  
+-----+  
| current_date() |  
+-----+  
| 2014-09-30 |  
+-----+  
1 row in set (0.00 sec)  
  
+-----+  
| (2*9+4) |  
+-----+  
| 22 |  
+-----+  
1 row in set (0.00 sec)  
mysql>
```

# Ejemplo Utilización source



```
mysql> source e:\AlojamientosRurales.sql
Query OK, 1 row affected, 1 warning (0.00 sec)

Database changed
Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>
```

```
Z:\>type AlojamientoRurales.sql!MORE
Create database if not exists AlojamientoRurales;
use AlojamientoRurales;
Create table if not exists ALOJAMIENTOS(
    nombreA Varchar(20) NOT NULL,
    direccion Varchar(20) NOT NULL,
    telefono Varchar(20) NOT NULL,
    contacto Char(3),
    numHabitaciones Int NOT NULL,
    Primary Key (nombreA)) ENGINE = MyISAM;

Create table if not exists PERSONAL (
    codigoP Char(3) NOT NULL,
    nombreA Varchar(20) NOT NULL,
    NIF Char(9) NOT NULL,
    nombreP Varchar(20) NOT NULL,
    dir Varchar(20),
    UNIQUE (NIF),
    Primary Key (codigoP),
    Foreign Key (nombreA) references ALOJAMIENTOS (nombreA) on delete cascade on
    update cascade) ENGINE = MyISAM;

Alter table ALOJAMIENTOS
    add Foreign Key (contacto) references PERSONAL (codigoP) on delete set NU
```

# Ejecución desde la shell

- También se pueden ejecutar los comandos de un fichero de texto desde el S.O, es decir, en modo batch desde la shell. Redirigiendo al cliente mysql un fichero de entrada:

```
C:\Users\alumno>mysql -u root -proot <ej.sql
user()
root@localhost
current_date()
2014-09-30
(2*9+4)
22
```

- También se podría redirigir la salida a otro fichero:

```
C:\Users\alumno>mysql -u root -proot <ej.sql >salida.txt
C:\Users\alumno>type salida.txt
user()
root@localhost
current_date()
2014-09-30
(2*9+4)
22
```

# SQL

- A nivel teórico, existen dos lenguajes para el manejo de bases de datos:
- **DDL** (Data Definition Language) Lenguaje de definición de datos. Es el lenguaje que se usa para crear bases de datos y tablas, y para modificar sus estructuras, así como los permisos y privilegios.
- Este lenguaje trabaja sobre unas tablas especiales llamadas *diccionario de datos*.
- **DML** (Data Manipulation Language) lenguaje de manipulación de datos. Es el que se usa para modificar y obtener datos desde las bases de datos.
- SQL engloba ambos lenguajes DDL+DML, ambos forman parte del conjunto de sentencias de SQL.

# COMANDOS INICIALES

- Podemos conocer cuántas bases de datos existen en nuestro sistema usando la sentencia SHOW DATABASES:

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| alojamientosrurales |
| caminosantiago |
| cdcol |
| maite |
| mysql |
| pepe |
| performance_schema |
| phpmyadmin |
| test |
| webauth |
+-----+
11 rows in set (0.02 sec)

mysql>
```

Para seleccionar una base de datos se usa el comando USE, que es más bien una opción de **MySQL**.

```
mysql> use alojamientosrurales;
Database changed
mysql>
```

```
mysql> select @@version;
+-----+
| @@version |
+-----+
| 5.6.20 |
+-----+
1 row in set (0.00 sec)
```

# CREACIÓN DE BASES DE DATOS

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS]  
db_name [create_specification [,  
create_specification] ...]
```

**NOTA:** *create\_specification*:

```
[DEFAULT] CHARACTER SET juego_caracteres |  
[DEFAULT] COLLATE nombre_collation
```

**CREATE DATABASE** crea una base de datos con el nombre dado. Para usar CREATE DATABASE, necesita el permiso CREATE en la base de datos.

Ocurre un error si la base de datos existe y no especifica **IF NOT EXISTS**.

# CREACIÓN DE BASES DE DATOS

**create\_specification** pueden utilizarse para especificar característica de la base de datos. Las características se almacenan en el fichero db.opt en el directorio de la base de datos. La cláusula CHARACTER SET especifica el conjunto de caracteres por defecto de la base de datos. La cláusula COLLATE especifica la colación por defecto de la base de datos.

- El conjunto de caracteres de la base de datos y la colación se usan como valores por defecto para una tabla si no se especifica el conjunto de caracteres y colación en el comando CREATE TABLE. El conjunto de caracteres y colación para la base de datos por defecto están disponibles como los valores de las variables character\_set\_database y collation\_database. Por defecto son: latin1 y latin1\_swedish\_ci.

Ejemplo, creación de la BD “Liga de Baloncesto”:

```
mysql> create database if not exists Baloncesto character set latin1 collate lat
in1_spanish_ci;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| Baloncesto |
| mysql |
```



# Usar BD

```
mysql> use Baloncesto;  
Database changed  
mysql> 
```

mysql> **USE test**

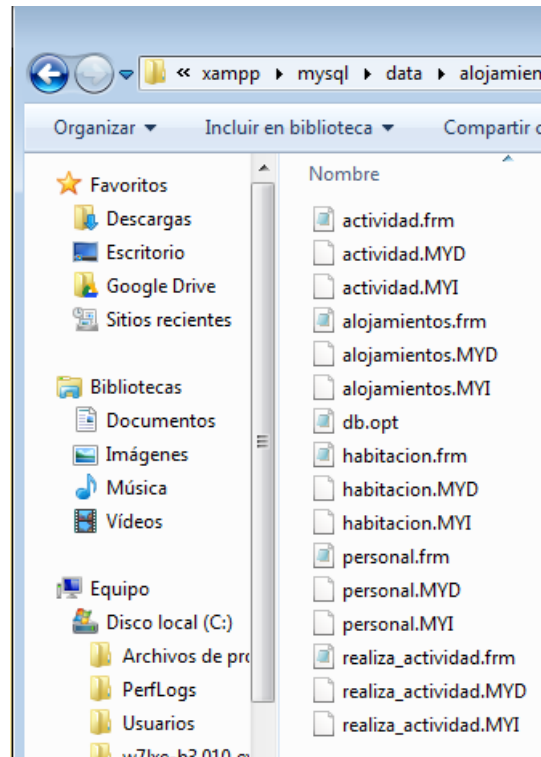
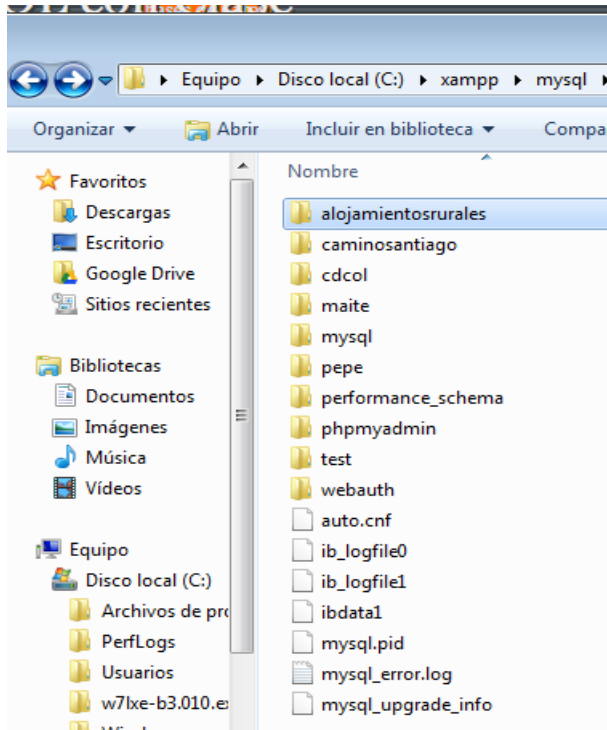
Al igual que QUIT, USE no necesita que ponga un punto y coma al final. La sentencia USE tiene otra particularidad: debe escribirse en una sola línea.

- Las bases de datos Ubuntu suele almacenarlas en el directorio: /var/lib/mysql pero para asegurarnos, debemos mirar el fichero my.cnf. Este fichero se encuentra en: /etc/mysql/my.cnf
- Ubuntu crea un nuevo directorio para cada BD, con el nombre de la misma y en él crea un fichero db.opt que contiene el juego de caracteres y collate. Y otros ficheros .frm para cada tabla.

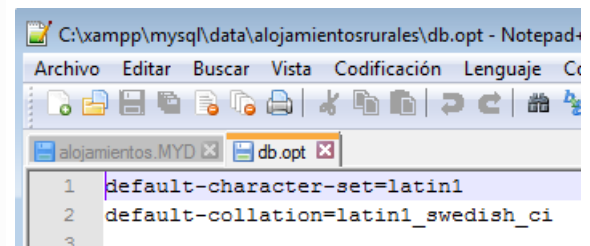
```
root@adminuser-VirtualBox:/var/lib/mysql/Baloncesto# ls -la  
total 24  
drwx----- 2 mysql mysql 4096 Oct  3 08:21 .  
drwx----- 5 mysql mysql 4096 Oct  3 06:57 ..  
-rw-rw---- 1 mysql mysql  65 Oct  2 08:09 db.opt  
-rw-rw---- 1 mysql mysql 8602 Oct  3 08:18 equipo.frm  
root@adminuser-VirtualBox:/var/lib/mysql/Baloncesto#
```

# Ver las tablas

- Podemos consultar cuántas tablas y qué nombres tienen en una base de datos, usando la sentencia SHOW TABLES:



```
mysql> show tables;
+-----+
| Tables_in_alojamientosrurales |
+-----+
| actividad                      |
| alojamientos                   |
| habitacion                     |
| personal                       |
| realiza_actividad              |
+-----+
5 rows in set (0.01 sec)
```



# MODIFICACIÓN DE UNA BASE DE DATOS

```
ALTER {DATABASE | SCHEMA} [db_name] alter_specification [,  
alter_specification] ...
```

**ALTER DATABASE** permite cambiar las características globales de una base de datos. Estas características se almacenan en el fichero `db.opt` en el directorio de la base de datos. Para usar **ALTER DATABASE**, necesita el permiso **ALTER** en la base de datos.

La cláusula **CHARACTER SET** cambia el conjunto de caracteres por defecto de la base de datos. La cláusula **COLLATE** cambia la colación por defecto de la base de datos. La colación es un conjunto de reglas (sólo una en este caso): “compara las codificaciones”.

Hay una convención para nombres de colaciones: empiezan con el nombre del conjunto de caracteres al que están asociados, normalmente incluyen el nombre del idioma, y acaban con **\_ci** (no distingue entre mayúsculas y minúsculas), **\_cs** (distingue entre mayúsculas y minúsculas), o **\_bin** (binario).

# BORRAR BASE DE DATOS

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

**DROP DATABASE** borrar todas las tablas en la base de datos y borrar la base de datos. Para usar **DROP DATABASE**, necesita el permiso **DROP** en la base de datos. **IF EXISTS** se usa para evitar un error si la base de datos no existe.

**DROP SCHEMA** puede usarse desde MySQL 5.0.2.

Si usa **DROP DATABASE** en una base de datos enlazada simbólicamente, tanto el enlace como la base de datos se borran.

**DROP DATABASE** retorna el número de tablas que se eliminan. Se corresponde con el número de ficheros **.frm** borrados.

El comando **DROP DATABASE** borrar del directorio de base de datos los ficheros y directorios que MySQL puede crear durante operaciones normales:

- Todos los ficheros con estas extensiones:

- Y el fichero **db.opt** , si existe.

.BAK	.DAT	.HSH
.MRG	.MYD	.ISD
.MYI	.db	.frm

# CREAR TABLA

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [Esquema.] NombreTabla  
[(DefiniciónCreate,...)] [OpcionesTabla] [select_statement]
```

## **DefiniciónCreate:**

### *DefiniciónColumna*

```
[CONSTRAINT [símbolo]] PRIMARY KEY [index_type] (nombreColumna,...)  
| KEY [NombreIndex] [TipoIndex] (NombreIndex,...)  
| INDEX [NombreIndex] [TipoIndex] NombreIndex,...)  
| [CONSTRAINT [símbolo]] UNIQUE [INDEX] [NombreIndex] [TipoIndex]  
(index_col_name,...)  
| [CONSTRAINT [símbolo]] FOREIGN KEY [NombreIndex] (index_col_name,...)  
[DefiniciónReferencia]  
| CHECK (expr)
```

## **DefiniciónColumna:**

```
nombreColumna TipoDato [NOT NULL | NULL] [DEFAULT valor] [AUTO_INCREMENT]  
[UNIQUE [KEY] | [PRIMARY] KEY] [COMMENT 'string'] [Definiciónreferencia]
```

# CREAR TABLA

**type:** TINYINT[(length)] [UNSIGNED] [ZEROFILL] | SMALLINT[(length)] [UNSIGNED] [ZEROFILL] | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL] | INT/INTEGER [(length)] [UNSIGNED] [ZEROFILL] | BIGINT[(length)] [UNSIGNED] [ZEROFILL] | FLOAT/REAL[(length,decimals)] [UNSIGNED] [ZEROFILL] | DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL] | DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL] | NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL] | DATE | TIME | TIMESTAMP | DATETIME | CHAR(length) [BINARY | ASCII | UNICODE] | VARCHAR(length) [BINARY] | TINYBLOB | BLOB | MEDIUMBLOB | LONGBLOB | TINYTEXT [BINARY] | TEXT [BINARY] | MEDIUMTEXT [BINARY] | LONGTEXT [BINARY] | ENUM(value1,value2,value3,...) | SET(value1,value2,value3,...)

## DefiniciónReferencia:

REFERENCES *NombreTabla* [(nombreColumna,...)]  
[ON DELETE {CASCADE | SET NULL | NO ACTION}]  
[ON UPDATE {CASCADE | SET NULL | NO ACTION}]

<http://dev.mysql.com/doc/refman/5.6/en/create-table.html>

# CREAR TABLA

- **Definición Create:** *Especifica la definición de los campos que va a contener la tabla y sus restricciones. **TEMPORARY** crea tablas temporales. La primera cláusula de la definición es la de las columnas:*
- **nombreColumna** TipoDato [NOT NULL | NULL] [DEFAULT **valor**] [AUTO\_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY] [COMMENT '**string**'] [Definición**referencia**]
- Por ejemplo:
  - ✓ Definir el código postal => cp como un tipo numérico de 5 dígitos enteros y 0 decimales, que no admite valores nulos y si no se especifica un valor que tome valor 28000.  
**cp NUMERIC(5,0) NOT NULL DEFAULT 28000**
  - ✓ El número de la seguridad social => nss como un entero que no puede repetirse.  
**nss INT UNIQUE KEY**
  - ✓ El DNI será la clave primaria  
**DNI INT PRIMARY KEY**
  - ✓ El nombre tendrá 40 caracteres => nombre VARCHAR(40)

# CREAR TABLA

```
mysql> create table if not exists empleado (cp NUMERIC(5,0) NOT
NULL DEFAULT 28000, nss int UNIQUE, dni integer PRIMARY KEY, n
ombre VARCHAR(40));
Query OK, 0 rows affected (0.24 sec)
```

```
mysql> show create table;
```

```
mysql> describe empleado;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cp     | decimal(5,0)  | NO   |     | 28000    |       |
| nss    | int(11)       | YES  | UNI | NULL     |       |
| dni    | int(11)       | NO   | PRI | NULL     |       |
| nombre | varchar(40)   | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

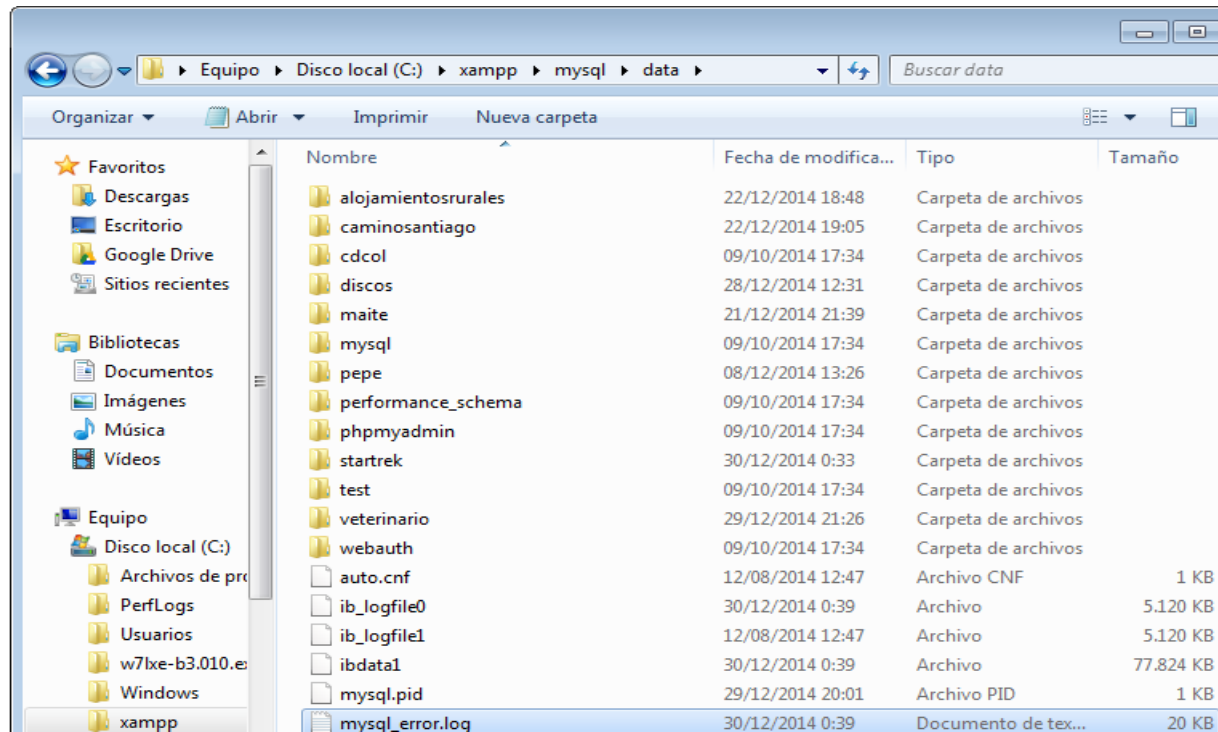
```
mysql> show create table empleado;
```

```
+-----+-----+
| Table | Create Table
+-----+-----+
```

```
empleado | CREATE TABLE `empleado` (
  `cp` decimal(5,0) NOT NULL DEFAULT '28000',
  `nss` int(11) DEFAULT NULL,
  `dni` int(11) NOT NULL,
  `nombre` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`dni`),
  UNIQUE KEY `nss` (`nss`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.00 sec)
```

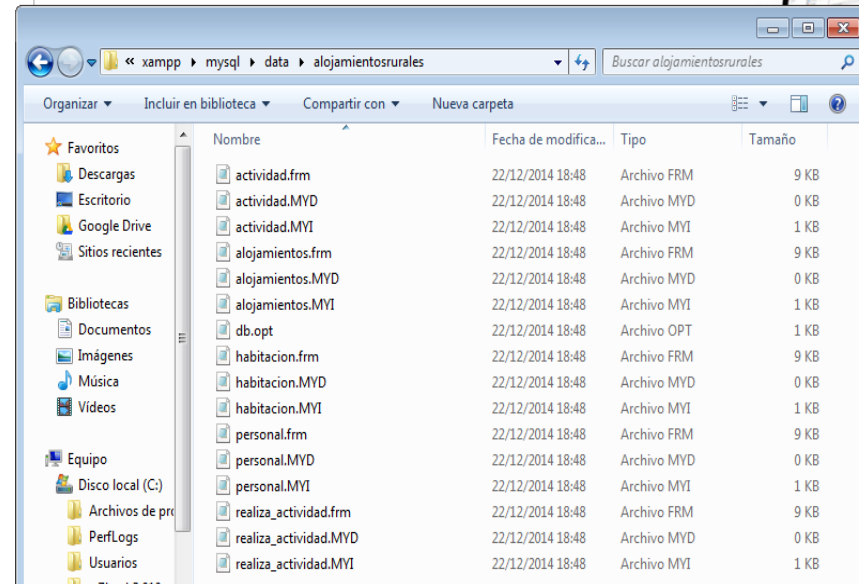


# Directorio data



# CREAR TABLA

- **CREATE TABLE** crea una tabla con el nombre dado. Debe tener el permiso CREATE para la tabla.
- MySQL representa cada tabla mediante un fichero .frm de formato de tabla (definición) en el directorio de base de datos. El motor para la tabla puede crear otros ficheros también. En el caso de tablas **MyISAM**, el motor crea ficheros índice y de datos. Por lo tanto, para cada tabla MyISAM nombreTabla, hay tres ficheros de disco:
  - **Fichero de formato de tabla** (definición) nombreTabla.frm
  - **Fichero de datos** nombreTabla.MYD
  - **Fichero índice** nombreTabla.MYI



Nombre	Fecha de modifica...	Tipo	Tamaño
actividad.frm	22/12/2014 18:48	Archivo FRM	9 KB
actividad.MYD	22/12/2014 18:48	Archivo MYD	0 KB
actividad.MYI	22/12/2014 18:48	Archivo MYI	1 KB
alojamientos.frm	22/12/2014 18:48	Archivo FRM	9 KB
alojamientos.MYD	22/12/2014 18:48	Archivo MYD	0 KB
alojamientos.MYI	22/12/2014 18:48	Archivo MYI	1 KB
db.opt	22/12/2014 18:48	Archivo OPT	1 KB
habitacion.frm	22/12/2014 18:48	Archivo FRM	9 KB
habitacion.MYD	22/12/2014 18:48	Archivo MYD	0 KB
habitacion.MYI	22/12/2014 18:48	Archivo MYI	1 KB
personal.frm	22/12/2014 18:48	Archivo FRM	9 KB
personal.MYD	22/12/2014 18:48	Archivo MYD	0 KB
personal.MYI	22/12/2014 18:48	Archivo MYI	1 KB
realiza_actividad.frm	22/12/2014 18:48	Archivo FRM	9 KB
realiza_actividad.MYD	22/12/2014 18:48	Archivo MYD	0 KB
realiza_actividad.MYI	22/12/2014 18:48	Archivo MYI	1 KB

# Tipos de Tablas

## Conceptos Básicos de MYSQL

### Tipos de tabla

- Motores de almacenamiento
- Tablas no transaccionales
  - Más rápidas (de 3 a 5 veces más)
  - No revierten los procesos, pueden crear inconsistencia
    - ISAM
    - MyISAM
    - HEAP
    - MERGE
- Tablas transaccionales
  - Transacción: grupo de sentencias SQL agrupadas de manera que deban ejecutarse todas juntas o no ejecutarse, nunca ejecutarse unas sentencias si y otras no.
  - Requieren más espacio en disco y memoria
  - Revierten el proceso si algo falla
    - Berkeley\_db (BDB)
    - InnoDB

```
BEGIN
INSERT INTO ...;
UPDATE .....;
UPDATE ....;
INSERT INTO ...;
COMMIT;
EXCEPTION
WHEN .... THEN
ROLLBACK;
END;
```

# Tipos de Tablas

## **MYISAM:**

- Tipo de tabla por defecto en MySQL desde la versión 3.23.
- Optimizada para sistemas operativos de 64 bits
- Permite ficheros de mayor tamaño que ISAM.
- Se pueden copiar tablas de una máquina a otra de distinta plataforma.
- No transaccional.
- No permite claves foráneas
- Muy rápido en lectura y escritura (excepto escrituras simultaneas en la misma tabla).
- Bajo requerimiento de espacio en disco y memoria.
- Los datos se guardan en el disco en diferentes ficheros:
  - \*.frm : definición de la tabla.
  - \*.MYD: fichero de datos.
  - \*.MYI: fichero de índices.
- Adecuadas si hay muchos SELECT.

# Tipos de Tablas

## **INNODB**

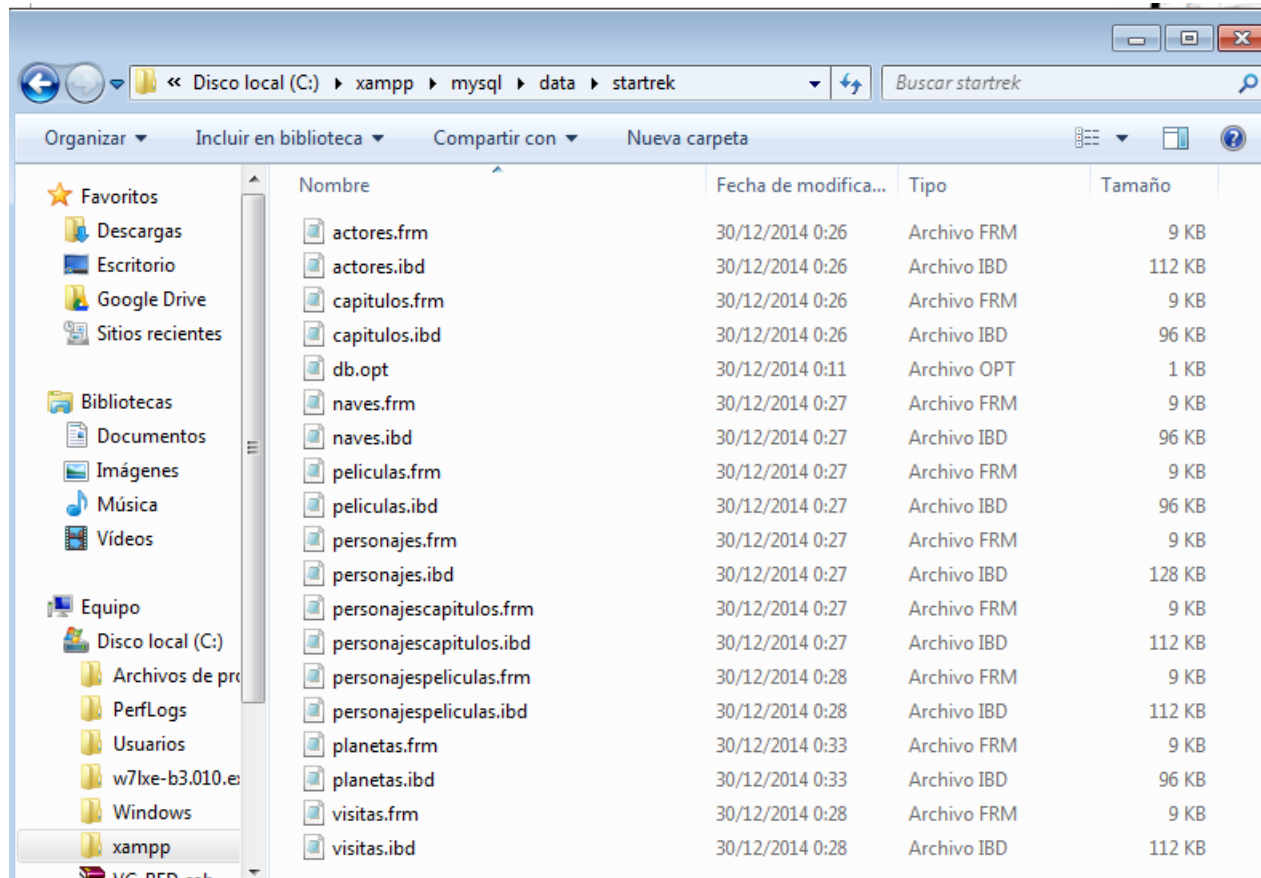
- Transaccional.
- Con posibilidad de commit, rollback
- Permite claves foráneas.
- Fácil recuperación de datos en caso de error.
- Alta concurrencia más segura en escritura.
- Deshacer transacciones a medias ("rollback").
- Necesita más espacio en disco y memoria que MyISAM para guardar los datos (unas tres veces mas de espacio en disco)
- Buena elección cuando necesitamos transacciones, restricciones de claves foráneas, o tenemos muchos INSERT y UPDATE y menos SELECT
- Los datos se guardan en disco:
  - ✓ Un fichero para la definición de la tabla: .frm
  - ✓ Un fichero llamado "tablespace" para guardar conjuntamente datos e índices.
- El tablespace puede consistir en uno o más ficheros, o incluso una partición entera en disco.

# Tipos de Tablas

- Ver los motores disponibles:  
`mysql> show engines;`
- Cambiar el motor por defecto al arrancar el servidor:  
`shell# mysqld_safe --user=mysql --default-storage-engine=InnoDB`  
`shell# mysqld_safe --user=mysql --default-table-type=InnoDB`
- Cambiar el motor de almacenamiento por defecto en my.cnf:  
`default-storage-engine = innodb`  
`default-table-type = innodb`
- Cambiar el motor de almacenamiento por defecto en la sesión actual:  
`mysql> set storage_engine=myisam;`
- Comprobar motor de almacenamiento por defecto actual:  
`mysql> show variables like '%storage%';`
- Comprobar el tipo de una tabla:  
`mysql> show create table mitabla;`

# CREAR TABLA

- Tablas innodb



# CREAR TABLA

- Si no se especifica NULL ni NOT NULL, la columna se trata como si se especificara **NULL** .
- En la sintaxis de CREATE TABLE, [Definición*referencia*] sirve para crear una clave foránea. De esta forma se enlaza el campo a su campo origen, es decir, se crea una referencia.
- Por ejemplo, podemos añadir a la tabla anterior un campo numDpto, que será una clave externa ya que hará referencia al CodDepartamento de la tabla “Departamentos”.
- Las opciones ON DELETE y ON UPDATE establecen el comportamiento del gestor en el caso de que las filas de la tabla padre (tabla referenciada) se borren o se actualicen. Los comportamientos pueden ser:
- **CASCADE** : La operación se propaga en cascada a la tabla hija.
- **SET NULL** : Se establece a NULL la clave foránea afectada.
- **NO ACTION** : La operación se impide.

Si no se especifica ON DELETE u ON UPDATE, por defecto se actúa como **NO ACTION**.



# CREAR TABLA

```
mysql> create table departamentos (nombreDpto varchar(20) UNIQUE NOT NULL, codDpto int PRIMARY KEY);
Query OK, 0 rows affected (0.45 sec)
```

```
mysql> create table if not exists empleados (nombre varchar(40), dni int PRIMARY KEY, cp numeric(5,0) NOT NULL DEFAULT 28000, nss int UNIQUE, numDpto int REFERENCES departamentos(codDpto) ON DELETE SET NULL ON UPDATE CASCADE);
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> describe empleados;
```

Field	Type	Null	Key	Default	Extra
nombre	varchar(40)	YES		NULL	
dni	int(11)	NO	PRI	NULL	
cp	decimal(5,0)	NO		28000	
nss	int(11)	YES	UNI	NULL	
numDpto	int(11)	YES		NULL	

5 rows in set (0.00 sec)

```
mysql> describe departamentos;
```

Field	Type	Null	Key	Default	Extra
nombreDpto	varchar(20)	NO	UNI	NULL	
codDpto	int(11)	NO	PRI	NULL	

2 rows in set (0.00 sec)

# CREAR TABLA

- La siguiente parte de CREATE TABLE hace referencia a las declaraciones globales sobre la tabla, restricciones, claves primarias y foráneas compuestas, etc.

[CONSTRAINT [*símbolo*]] PRIMARY KEY [*index\_type*]  
(nombreColumna,...)

| [CONSTRAINT [*símbolo*]] FOREIGN KEY [Nombre*Index*]  
(nombreColumna,...) [*DefiniciónReferencia*]

En SQL, las restricciones pueden tener un nombre, para ello se utiliza CONSTRAINT [símbolo]

```
mysql> create table if not exists departamentos (codDpto int, nombre varchar(20) UNIQUE, CONSTRAINT claveDep PRIMARY KEY (codDpto));
Query OK, 0 rows affected (0.05 sec)
```

# CREAR TABLA

```
mysql> create table taba3 (id3 int, nom3 varchar(10) unique, num decimal(8,2), p  
int default 0, constraint pk primary key (id3));  
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> describe taba3;
```

Field	Type	Null	Key	Default	Extra
id3	int(11)	NO	PRI	0	
nom3	varchar(10)	YES	UNI	NULL	
num	decimal(8,2)	YES		NULL	
p	int(11)	YES		0	

4 rows in set

 XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]

```
mysql> insert into taba3 values (1,'pepe',123456.98,7);  
Query OK, 1 row affected (0.43 sec)
```

```
mysql> select * from taba3;
```

id3	nom3	num	p
1	pepe	123456.98	7

1 row in set (0.00 sec)

```
mysql> insert into taba3 (id3,nom3,num) values (7,'luid',123456.98);  
Query OK, 1 row affected (0.03 sec)
```

```
mysql>
```

```
mysql> select * from taba3;
```

id3	nom3	num	p
1	pepe	123456.98	7
7	luid	123456.98	0

2 rows in set (0.00 sec)

# CREAR TABLA

- **Opciones***Tabla* permite especificar las peculiaridades de cada gestor con respecto al almacenamiento físico de sus tablas.
- **Opciones***Tabla* :
  - {ENGINE|TYPE} = **nombreMotor**. **MyIsam** genera tablas operadas a gran velocidad, pero sin control de integridad referencial. **InnoDB** son tablas transaccionales con bloqueo de registros y clave foránea. **Memory** genera tablas en memoria sin almacenamiento en ficheros.
  - | AUTO\_INCREMENT = **valor**. Establece el valor inicial
  - | [DEFAULT] CHARACTER SET juegoCaracteres[COLLATE nombreColación]
  - | CHECKSUM = {0 | 1} Mantiene una suma de verificación para cada registro.
  - | COMMENT = '**string**'
  - | MAX\_ROWS = **valor**
  - | MIN\_ROWS = **valor**

```
mysql> checksum table empleados;
+-----+-----+
| Table                               | Checksum |
+-----+-----+
| LigaBaloncesto.empleados           |         0 |
+-----+-----+
1 row in set (0.00 sec)
```

# Ejemplo Tabla Pedidos

```
mysql> create table if not exists Pedidos(codPedido int AUTO_INCREMENT PRIMARY KEY, fecha DATETIME, estado enum('Pendiente','Entregado','Rechazado')) COMMENT = 'Tabla de pedidos a proveedores' AUTO_INCREMENT = 1000 MAX_ROWS =10000 CHECKSUM=1 ENGINE=innodb;
```

```
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> █
```

```
mysql> show tables;
```

```
+-----+
| Tables_in_LigaBaloncesto |
+-----+
| Pedidos                  |
| departamentos           |
| empleados                |
| equipos                  |
| jugadores                |
+-----+
5 rows in set (0.00 sec)
```

# Consulta de Tablas

- Para ver las tablas disponibles de una BD:  
⇒ Show tables;
- Para visualizar la estructura de una tabla:  
⇒ Describe [esquema.]NombreTabla;

```
mysql> describe empleados;
```

Field	Type	Null	Key	Default	Extra
nombre	varchar(40)	YES		NULL	
dni	int(11)	NO	PRI	NULL	
cp	decimal(5,0)	NO		28000	
nss	int(11)	YES	UNI	NULL	
numDpto	int(11)	YES		NULL	

```
5 rows in set (0.00 sec)
```

# MODIFICACIÓN DE TABLAS

**ALTER [IGNORE] TABLE *NombreTabla* EspecificaciónAlter [, EspecificaciónAlter] ...**

**EspecificaciónAlter :**

**ADD [COLUMN] *definiciónColumna* [FIRST | AFTER *nombreColumna* ]**

| **ADD [COLUMN] (*definiciónColumna*,...)**

| **ADD [CONSTRAINT [*símbolo*]] PRIMARY KEY (*nombreColumna*,...)**

| **ADD [CONSTRAINT [*símbolo*]] UNIQUE (*nombreColumna*,...)**

| **ADD [CONSTRAINT [*símbolo*]] FOREIGN KEY (*nombreColumna*,...)**

**[*definiciónReferencia*]**

| **CHANGE [COLUMN] *anteriorNombreColumna* *definiciónColumna* [FIRST | AFTER *nombreColumna*]**

| **RENAME COLUMN *anteriorNombreColumna* TO *nuevoNombreColumna***

| **MODIFY [COLUMN] *definiciónColumna* [FIRST | AFTER *nombreColumna*]**

| **DROP [COLUMN] *nombreColumna***

| **DROP PRIMARY KEY**

| **DROP FOREIGN KEY *fk\_símbolo***

| ***opcionesTabla***

# MODIFICACIÓN DE TABLAS

- La opción **ADD** permite añadir una columna, pudiendo especificar el lugar de la inserción mediante AFTER y FIRST.
- Con la opción **MODIFY** se cambia el tipo de datos de una columna y se añaden restricciones.
- Con la opción **DROP** se pueden eliminar las restricciones de claves foráneas y primarias, dejando el tipo de dato y su contenido intacto.
- *Para cambiar el nombre de una columna, MySQL utiliza **CHANGE**.*
- ***opcionesTabla** varían con cada SGBD, sirven para modificar las características del almacenamiento físico.*



# MODIFICACIÓN DE TABLAS

- Por ejemplo, para añadir el campo Ciudad en la tabla Equipos detrás del nombre:

```
mysql> alter table equipos add ciudad varchar(20) AFTER nombre;  
Query OK, 0 rows affected (0.46 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

Para eliminar la clave primaria dni de la tabla empleados y establecer como clave primaria el campo nss:

```
mysql> alter table empleados drop PRIMARY KEY, add PRIMARY KEY  
(nss);  
Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> describe empleados;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type          | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| nombre | varchar(40)   | YES  |     | NULL    |       |  
| dni    | int(11)       | NO   |     | NULL    |       |  
| cp     | decimal(5,0)  | NO   |     | 28000   |       |  
| nss    | int(11)       | NO   | PRI | 0       |       |  
| numDpto | int(11)       | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

# MODIFICACIÓN DE TABLAS

```
mysql> alter table taba2 add id integer primary key auto_increment first;  
Query OK, 0 rows affected (0.14 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> describe taba2;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(20)	YES	UNI	pepe	
descripcion	int(11)	YES		NULL	
descrip	varchar(20)	YES		hola	

```
4 rows in set (0.01 sec)
```

```
mysql> describe taba2;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(20)	YES	UNI	pepe	
descripcion	int(11)	YES		NULL	
descrip	varchar(20)	YES		hola	
id3	int(11)	YES		NULL	

```
rows in set (0.00 sec)
```

```
mysql> describe taba;
```

Field	Type	Null	Key	Default	Extra
id	tinyint(4)	NO	PRI	NULL	auto_increment
nombre	varchar(20)	YES	UNI	NULL	
casado	tinyint(1)	NO		1	

```
rows in set (0.01 sec)
```

```
mysql> alter table taba2 modify id3 tinyint;  
Query OK, 3 rows affected (0.64 sec)  
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> alter table taba2 add foreign key (id3) REFERENCES taba(id);  
Query OK, 3 rows affected (0.82 sec)  
Records: 3 Duplicates: 0 Warnings: 0
```

# MODIFICACIÓN DE TABLAS

- Para eliminar la cláusula default de una tabla:

```
mysql> alter table taba1 alter nombre1 drop default;  
Query OK, 0 rows affected (0.02 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

- Renombrar una columna:

```
mysql> alter table taba1 change dir direccion varchar(30);  
Query OK, 7 rows affected (0.18 sec)  
Records: 7 Duplicates: 0 Warnings: 0
```

- Para hacer que la clave primaria sea auto\_increment:

```
mysql> alter table taba1 modify id int auto_increment;  
Query OK, 7 rows affected (0.22 sec)  
Records: 7 Duplicates: 0 Warnings: 0
```

# Ejemplo Mascotas

- Crear una BD Veterinario y una tabla Mascotas con los siguientes campos (Nombre, especie, raza, fechaNacimiento, sexo)

```
mysql> describe mascotas;
```

Field	Type	Null	Key	Default	Extra
nombre	varchar(20)	NO		NULL	
especie	varchar(20)	NO		NULL	
raza	varchar(20)	YES		NULL	
fnac	date	YES		NULL	
sexo	char(1)	YES		NULL	

```
5 rows in set (0.03 sec)
```

Añadir una clave primaria que sea autoincrementada.

```
mysql> alter table mascotas add idMascota int auto_increment primary key first;
Query OK, 0 rows affected (0.18 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe mascotas;
```

Hacer que el campo nombre sea único.

```
mysql> alter table mascotas add UNIQUE(nombre);
```

# Ejemplo Mascotas

```
mysql> describe mascotas;
```

Field	Type	Null	Key	Default	Extra
idMascota	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(20)	NO	UNI	NULL	
especie	varchar(20)	NO		NULL	
raza	varchar(20)	YES		NULL	
fnac	date	YES		NULL	
sexo	char(1)	YES		NULL	

```
6 rows in set (0.01 sec)
```

Modificar el campo nombre, para que tenga 30 caracteres y no sea optativo.

```
mysql> alter table mascotas modify column nombre varchar(30) NOT NULL;
Query OK, 0 rows affected (0.12 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Añadir el campo pedigree después de raza.

```
mysql> alter table mascotas add pedigree bool after raza;
Query OK, 0 rows affected (0.21 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Cambia el campo fnac por fechaNacimiento.

```
mysql> alter table mascotas change column fnac fechaNacimiento date;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

# Ejemplo Mascotas

- Modificar el campo especie, para que por defecto sea 'canina':

```
mysql> alter table mascotas modify especie varchar(20) not null default 'canina';
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe mascotas;
```

Field	Type	Null	Key	Default	Extra
idMascota	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(30)	NO	UNI	NULL	
especie	varchar(20)	NO		canina	
raza	varchar(20)	YES		NULL	
pedigree	tinyint(1)	YES		NULL	
fechaNacimiento	date	YES		NULL	
sexo	char(1)	YES		NULL	

```
7 rows in set (0.01 sec)
```

# BORRADO DE TABLAS

```
DROP [TEMPORARY] TABLE [IF EXISTS] nombreTabla [, nombreTabla] ...
```

**DROP TABLE** borra una o más tablas. Debe tener el permiso **DROP** para cada tabla. **IF EXISTS** para evitar un error si la tabla no existe. La palabra **TEMPORARY** tiene el siguiente efecto:

- ✓ El comando sólo borra tablas **TEMPORARY**.
- ✓ El comando no acaba una transacción en marcha.
- ✓ No se chequean derechos de acceso. (Una tabla **TEMPORARY** es visible sólo para el cliente que la ha creado, así que no es necesario.)
- ✓ Usar **TEMPORARY** es una buena forma de asegurar que no borra accidentalmente una tabla no **TEMPORARY**.

# RENOMBRAR TABLAS

```
RENAME TABLE nombreTabla TO nuevoNombreTabla [, nombreTabla 2 TO  
nuevoNombreTabla2] ..
```

Este comando renombra una o más tablas.

La operación de renombrar se hace automáticamente, lo que significa que ningún otro flujo puede acceder a ninguna de las tablas mientras se ejecuta el renombrado. Por ejemplo, si tenemos una tabla existente *tablaMaeAct*, y se quiere renombrar como *tablaMae*, pero conservando ésta última como copia de Backup:

```
RENAME TABLE tablaMae TO backupTablaMae;  
RENAME table tablaMaeAct TO tablaMae;
```



# COPIAR TABLAS

- Existen dos procedimientos:
- 1/ En el caso de querer hacer la copia sin los datos de la tabla origen, en **MySQL**, lo podemos hacer mediante **CREATE TABLE LIKE**.

```
mysql> show tables;
+-----+
| Tables_in_veterinario |
+-----+
| mascotas               |
| propietarios           |
+-----+
2 rows in set (0.00 sec)

mysql> create table copiaMascotas like mascotas;
Query OK, 0 rows affected (0.03 sec)

mysql> describe copiaMascotas;
+-----+-----+-----+-----+-----+-----+
| Field                | Type                | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| idMascota             | int(11)             | NO   | PRI | NULL    | auto_increment |
| nombre               | varchar(30)         | NO   | UNI | NULL    |                |
| especie              | varchar(20)         | NO   |     | NULL    |                |
| raza                 | varchar(20)         | YES  |     | NULL    |                |
| pedigree             | tinyint(1)          | YES  |     | NULL    |                |
| fechaNacimiento      | date               | YES  |     | NULL    |                |
| sexo                 | char(1)            | YES  |     | NULL    |                |
| propietario          | varchar(10)         | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql>
mysql> select * from copiamascotas;
Empty set (0.00 sec)

mysql>
```

# COPIAR TABLAS

2/ En el caso que se quieran copiar los datos o un subconjunto de ellos lo haríamos mediante **CREATE TABLE SELECT**.

```
mysql> create table copiaMascotas2 select * from mascotas;
Query OK, 7 rows affected (0.07 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> select count(*) from copiamascotas2;
+-----+
| count(*) |
+-----+
|        7 |
+-----+
1 row in set (0.02 sec)

mysql> describe copiaMascotas2;
+-----+-----+-----+-----+-----+-----+
| Field           | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idMascota       | int(11)       | NO   |     | 0        |       |
| nombre         | varchar(30)   | NO   |     | NULL     |       |
| especie        | varchar(20)   | NO   |     | canina    |       |
| raza           | varchar(20)   | YES  |     | NULL     |       |
| pedigree       | tinyint(1)    | YES  |     | NULL     |       |
| fechaNacimiento | date          | YES  |     | NULL     |       |
| sexo           | char(1)       | YES  |     | NULL     |       |
| propietario    | varchar(10)   | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

# COPIAR TABLAS

También se puede copiar la estructura de la tabla y sus datos en dos pasos, primero creando la estructura y luego copiando los datos:

```
mysql>
mysql> create table copiaMascotas3 like mascotas;
Query OK, 0 rows affected (0.16 sec)
```

```
mysql>
mysql> insert into copiaMascotas3 select * from mascotas;
Query OK, 7 rows affected (0.05 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> describe copiaMascotas3;
```

Field	Type	Null	Key	Default	Extra
idMascota	int(11)	NO	PRI	NULL	auto_increment
nombre	varchar(30)	NO	UNI	NULL	
especie	varchar(20)	NO		canina	
raza	varchar(20)	YES		NULL	
pedigree	tinyint(1)	YES		NULL	
fechaNacimiento	date	YES		NULL	
sexo	char(1)	YES		NULL	
propietario	varchar(10)	YES		NULL	

```
rows in set (0.01 sec)

mysql> select * from copiaMascotas3;
```

idMascota	nombre	especie	raza	pedigree	fechaNacimiento	sexo	propietario
11	kira	canina	dalmata	1	1970-01-01	h	
12	terry	canina	pastor aleman	1	1973-01-01	m	
14	lorato	ave	loro	1	1973-01-01	m	
15	donna	canina	fox terrier	0	1980-01-01	h	
17	dante	canina	cairn terrier	0	1980-01-01	m	
18	boris	canina	bichon maltes	0	1998-01-01	m	
19	laica	canina	boxer	0	0000-00-00	h	
NULL							

```
rows in set (0.00 sec)
```

# COPIA DE SEGURIDAD

- Mysql dispone de diversas "caches" en las que se almacenan datos temporalmente con el objetivo de mejorar en rendimiento, de forma que por ejemplo, una vez hecha una modificación en una tabla, puede ser que los datos no se guarden inmediatamente en disco, hasta que termine, por ejemplo, una consulta que se estaba ejecutando. Por esto, es necesario "forzar" a Mysql a escribir todos los datos en el disco, mediante la sentencia "**Flush Tables**".
- Además es necesario que no se escriba en las tablas mientras se esta haciendo la copia de seguridad de la base de datos, que se consigue con el comando "lock tables", seguido del nombre de la tabla.
- Se puede realizar una copia de seguridad a través de la sentencia sql "**backup table**".
- También es posible realizar copias de seguridad a través de las herramientas que nos proporciona el propio gestor de base de datos, como pueden ser **mysqldump** ó **mysqlhotcopy**.

# Comando mysqldump

**El comando mysqldump del sistema gestor de base de datos MySQL sirve para hacer copias de seguridad.** Este comando permite hacer la copia de seguridad de una o múltiples bases de datos. Además permite que estas copias de seguridad se puedan restaurar en distintos tipos de gestores de bases de datos, sin la necesidad de que se trate de un gestor de mysql. Esto lo consigue creando unos ficheros, que contienen todas las sentencias sql necesarias para poder restaurar la tabla, que incluyen desde la sentencia de creación de la tabla, hasta una sentencia insert por cada uno de los registros que forman parte de la misma.

Las limitaciones de la restauración dependerán de las opciones que se han especificado a la hora de hacer la copia de seguridad, por ejemplo, si se incluye la opción `--add-drop-table` al hacer la copia de seguridad, se podrán restaurar tablas que existen actualmente en el servidor (borrándolas primero). Por lo que es necesario estudiar primero los procedimientos que se utilizarán tanto en la copia como en la restauración, para que todo salga correcto!

# Opciones de mysqldump:

- **-add-locks**  
Añade LOCK TABLES antes, y UNLOCK TABLE después de la copia de cada tabla.
- **--add-drop-table**  
Añade un drop table antes de cada sentencia create
- **-A, --all-databases**  
Copia todas las bases de datos. Es lo mismo que utilizar --databases seleccionando todas.
- **-a, --all**  
Incluye todas las opciones de creación específicas de Mysql.
- **--allow-keywords**  
Permite la creación de nombres de columnas que son palabras clave, esto se realiza poniendo de prefijo a cada nombre de columna, el nombre de la tabla
- **-c, --complete-insert**  
Utiliza inserts incluyendo los nombres de columna en cada sentencia (incrementa bastante el tamaño del fichero)
- **-C, --compress**  
Comprime la información entre el cliente y el servidor, si ambos soportan compresión.

# Opciones de mysqldump:

- **-B, --databases**  
Para copiar varias bases de datos. En este caso, no se especifican tablas. El nombre de los argumentos se refiere a los nombres de las bases de datos. Se incluirá USE db\_name en la salida antes de cada base de datos.
- **--delayed**  
Inserta las filas con el comando INSERT DELAYED.
- **-e, --extended-insert**  
Utiliza la sintaxis de INSERT multilínea. (Proporciona sentencias de insert más compactas y rápidas.)
- **-#, --debug[=option\_string]**  
Utilización de la traza del programa (para depuración).
- **--help**  
Muestra mensaje de ayuda y termina.

# Opciones de mysqldump:

- **-l, --lock-tables.**  
Bloquea todas las tablas antes de comenzar con la copia. Las tablas se bloquean con READ LOCAL para permitir inserts concurrentes en caso de las tablas MyISAM. Cuando se realiza la copia de múltiples bases de datos, --lock-tables bloqueará la copia de cada base de datos por separado.
- **-n, --no-create-db**  
No se incluirá en la salida CREATE DATABASE /\*!32312 IF NOT EXISTS\*/ db\_name; Esta línea se incluye si la opción --databases o --all-databases fue seleccionada.
- **-t, --no-create-info**  
No incluirá la información de creación de la tabla (sentencia CREATE TABLE).
- **-d, --no-data**  
No incluirá ninguna información sobre los registros de la tabla. Esta opción sirve para crear una copia de sólo la estructura de la base de datos.
- **--opt**  
Lo mismo que --quick --add-drop-table --add-locks --extended-insert --lock-tables. Esta opción le debería permitir realizar la copia de seguridad de la base de datos de la forma más rápida y efectiva.



# mysqldump

- **-v, --verbose**  
Va mostrando información sobre las acciones que se van realizando (más lento)
- **-w, --where='cláusula where'**  
Sirve para realizar la copia de determinados registros
- **-X, --xml**  
Realiza la copia de seguridad en un documento xml
- **-x, --first-slave**  
Bloquea todas las tablas de todas las bases de datos
- Para realizar la copia de seguridad de la base de datos mibase al fichero copia\_seguridad.sql
- `mysqldump --opt --password=miclave --user=miuser mibasededatos > archivo.sql`

```
C:\Users\alumno>mysqldump --password=usuario --user=usuario empresa > miempresa.
sql
Warning: Using a password on the command line interface can be insecure.
C:\Users\alumno>
```

# Restaurar la base de datos

- Si deseamos recuperar la información de un fichero para restaurar una copia de seguridad de la base de datos lo haremos con el comando mysql. Utilizaremos una sintaxis como esta:
- `mysql mibase < archivo.sql`

```
C:\Users\maite>mysql --user=usuario --password=usuario empresa < miempresa.sql
Warning: Using a password on the command line interface can be insecure.
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| cdcol      |
| empresa   |
| ligabaloncesto |
| mysql     |
+-----+
```

```
mysql> use empresa;
Database changed
mysql>
mysql> show tables;
+-----+
| Tables_in_empresa |
+-----+
| departamentos      |
| empleados          |
+-----+
2 rows in set (0.00 sec)
```

# Algunos comandos de Administración

- Cambiar la contraseña del usuario root:  
`mysqladmin -u root -ppuestoXX password 'puestoYY'`
- Chequear si el servidor MySQL está corriendo  
`mysqladmin -u root -ppuestoXX ping`
- Saber que versión de MySQL está corriendo  
`mysqladmin -u root -ppuestoXX version`
- Cuál es el estado actual del servidor MySQL  
`mysqladmin -u root -ppuestoXX status`
- El comando status muestra la siguiente información:
  - Uptime:** segundos desde los que se inició el servidor
  - Threads:** número total de clientes conectados
  - Questions:** número total de consultas que el servidor ha ejecutado desde su inicio.
  - Slow queries:** número total de consulta que el servidor ha ejecutado y que tienen un tiempo mayor de ejecución a la variable `long_query_time`.
  - Opens:** número total de tablas abiertas por el servidor.
  - Flush tables:** cuántas tablas han sido "volcadas".
  - Open tables:** número total de tablas abiertas en la base de datos.

# ACTIVIDAD 4.1

- En un fichero texto con nombre startrek1.sql, escriba las sentencias SQL para crear el modelo físico de la práctica 3.2. Después crea una BD llamada startrek en un servidor MySQL y ejecuta las sentencias mediante el comando source.
- **Notas:**
  - ✓ Por defecto la nacionalidad de los personajes será 'Española'
  - ✓ Todos los códigos serán autoincrementados y los campos, en general, serán obligatorios, excepto galaxia, año de la película y numTripulantes
  - ✓ La tabla Personajes tendrá dos claves ajenas:CodigoActor y codigoSuperior.

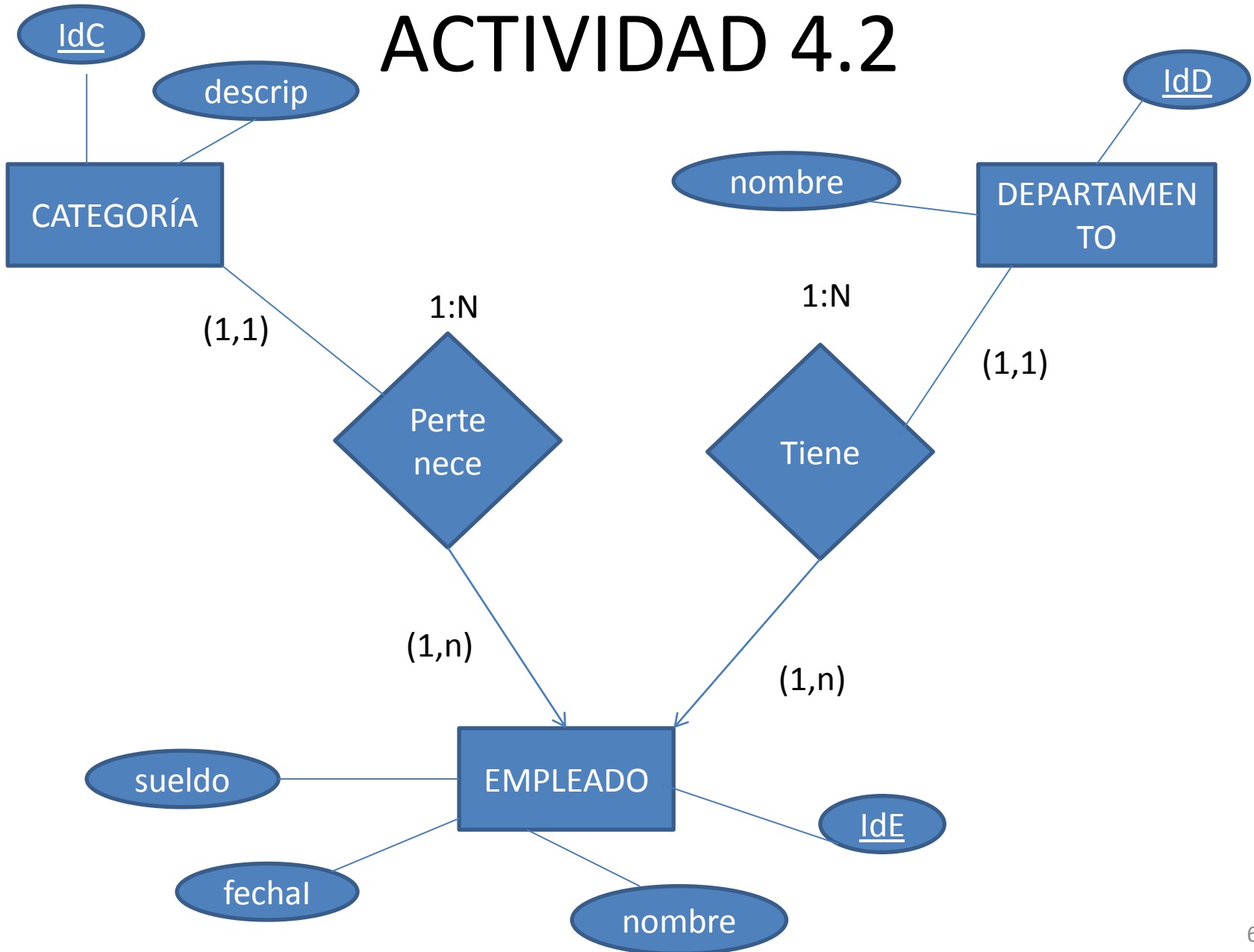
# ACTIVIDAD 4.1

- RESTRICCIONES:
- Que todas las tablas tengan el motor de almacenamiento innodb para que las claves foráneas no sean ignoradas.
- Que el campo galaxia de la tabla planetas sea una enumeración de los valores ('Vía Láctea', 'Andrómeda', 'Sombrero')

```
mysql> alter table Peliculas ENGINE=innodb;  
Query OK, 0 rows affected (0.15 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> alter table Personajes ENGINE=innodb;  
Query OK, 0 rows affected (0.63 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> alter table PersonajesCapitulos ENGINE=innodb;  
Query OK, 0 rows affected (0.26 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> alter table PersonajesPeliculas ENGINE=innodb;  
Query OK, 0 rows affected (0.09 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> alter table Planetas ENGINE=innodb;  
Query OK, 0 rows affected (0.34 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> alter table Visitas ENGINE=innodb;  
Query OK, 0 rows affected (0.27 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> alter table Planetas modify galaxia enum ("Vía Láctea", "Andrómeda", "Sombrero");  
Query OK, 0 rows affected (0.56 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

# ACTIVIDAD 4.2



# BIBLIOGRAFÍA

- <http://dev.mysql.com/doc/refman/5.0/es/>
- <http://mysql.conclase.net/curso/?cap=005#inicio>  
<http://blog.evidaliahost.com/2012/09/03/copiar-la-estructura-o-los-datos-de-tablas-de-mysql/>  
[http://www.ite.educacion.es/formacion/materiales/93/cd/pdf/M6\\_1\\_SQL.pdf](http://www.ite.educacion.es/formacion/materiales/93/cd/pdf/M6_1_SQL.pdf)
- <http://www.prograweb.com.mx/tallerBD/0203IntegridadReferencial.html>