UNIDAD 3 - 2ª PARTE: INTEGRIDAD, NORMALIZACIÓN Y OPTIMIZACIÓN DE BASES RELACIONALES

INTEGRIDAD REFERENCIAL

Las reglas de integridad son:
1. Ningún componente de una clave primaria puede tener valores en blanco o nulos. (Ésta es la norma básica de integridad).
2. Para cada valor de clave ajena deberá existir un valor de clave primaria concordante en su tabla de origen

NORMALIZACIÓN DE LA BASE DE DATOS

Para asegurar su optimización e integridad, las bases de datos relacionales pasan por un proceso al que se le conoce como "Normalización", que es un proceso de chequeo para conseguir la estructura óptima de cada tabla, **evitando duplicaciones y falta de integridad referencial,** es decir su objetivo fundamental es **eliminar la redundancia.** Cuanto más alta la forma normal, más estrictos son los criterios que cumple la tabla y más fácil resulta tratarla.

El punto de partida del proceso de normalización es el **esquema relacional**. Las anomalías de datos se pueden presentar al insertar, borrar y modificar datos en una base de datos relacional, producidos por un diseño deficiente. Cuanto más alta la forma normal, más estrictos son los criterios que cumple la tabla y más fácil resulta tratarla. Para pasar a una forma normal, se deben cumplir las anteriores.

Primera Forma Normal (1FN): ATRIBUTOS ATÓMICOS.

Se dice que una relación se encuentra en 1FN cuando cada atributo sólo toma un valor. No hay campos múltiples. Cada columna debe ser atómica - indivisible-
Ej: Campo compuesto Domicilio
En el caso de un campo compuesto como el domicilio, se debe descomponer en distintos campos, o se crea una tabla aparte.
Solución-> descomposición (Calle, Número, Cod_Postal, Ciudad, País)

Segunda Forma Normal (2FN): DEPENDENCIA TOTAL.

Se dice que una relación se encuentra en 2FN si:
Se encuentra en 1FN
Todos los atributos que no son principales tiene dependencia completa respecto a la clave principal. Todo campo que no sea clave debe depender por completo de la clave.

Derivados:
- Ej: Campos Fecha de Nacimiento y Edad
- Ej: Notas y Media
- Ej: Código Postal y Ciudad
Solución: El campo derivado se elimina

Tercera Forma Normal (3FN): NO DEPENDENCIAS TRANSITIVAS.

Un campo debe depender de la clave y no de otro campo. Se dice que una relación se encuentra en 3FN si:

- Se encuentra en 2FN
- No existe ningún atributo no principal que dependa transitivamente de alguna clave de R.

Para ello debe haber al menos dos campos más a parte de la clave (sea ésta compuesta o no)
- Ej: campos Localidad_Departamento y Nombre_Departamento en:

Tabla Empleado (Cod_Emple, Nombre, Apellidos, NIF, Telefono1, Telefono2, Localidad_Departamento, Nombre_Departamento)
- Localidad_Departamento y Nombre_Departamento no dependen de Empleado.

Posible Solución:
Se crea la Tabla Departamento (Cod_Dpto, Localidad_Departamento, Nombre_Departamento)
En la Tabla Empleado se añade Cod_Dpto como FK.
Tabla Empleado (Cod_Emple, Nombre, Apellidos, NIF, Telefono1, Telefono2, FK_Cod_Dpto)

Forma Normal de Boyce-Codd (FNBC): VARIOS CANDIDATOS Y UNA SOLA CLAVE.

Se dice que una relación se encuentra en FNBC si y sólo si todo determinante es clave candidata. Todos los campos determinantes de la tabla son clave candidata pero se debe escoger solo los campos que no provoquen redundancia. Se escoge una sola clave en función de las operaciones que se van a realizar con la tabla. Sobre todo en el caso de ternarias sería el caso de tener que aumentar las claves o disminuirlas.

Ejemplo:
Servicios_Peluquería (Cod_Cliente, Cod_Empleado, Cod_Servicio, Fecha)

Por defecto en la ternaria todas las claves propagadas son determinantes. Y además le debemos añadir la fecha para que el cliente pueda ir en distintos días.

Servicios_Peluquería (Cod_Cliente, Cod_Empleado, Cod_Servicio, Fecha)

A PARTIR DE LAS SIGUIENTES FORMAS NORMALES EL DETALLE ES MUY ABSTRACTO Y SE CONSIDERA QUE SI UNA BBDD HA LLEGADO HASTA L A 3FB Y LA DE BOYCE CODD ESTA NORMALIZADA

LA CUARTA Y QUINTA FORMAL BUSCAN EVITAR REDUNDANCIAS Y CONSULTAS COMPLEJAS CREANDO SUBTABLAS, SOLO SE PRESENTAN EN CASOS MUY CONCRETOS.

Cuarta Forma Normal (4FN): Evitar SUBCONJUNTOS DE CAMPOS REDUNDANTES (SUBCONJUNTOS MULTIVALUADOS)

Se dice que una relación se encuentra en 4FN si y sólo si las únicas dependencias multivaluadas no triviales son aquellas en las que una clave multidetermina un atributo. se refiere a los multivaluados. Si vemos que una ocurrencia se repite en varias columnas, dividimos en tablas.

Ej. Animales_en_extinción (*Cod_Animal*, Animal, Continentes).

Además de que no cumple la 1FN y habría que hacer una tabla Continentes, siempre se repetiría el mismo subconjunto
- Por ejemplo Oso Panda siempre irá con Asia.
- Podríamos crear una tabla Procedencia_Animal (Cod_Animal, Continente)

Animales_en_extinción (*Cod_Animal*, Animal)

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.

— —

(AQUÍ FALTA UNA PARTE DE ALGEBRA RELACIONAL) dos folios.

— —

INTRODUCCIÓN A MySQL y a SQL,

•**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 6
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.

Después de haberse conectado, puede desconectarse en cualquier momento escribiendo QUIT (o \q) en el prompt: #mysql -h *host* -u *user* -p
Enter password: ********
Introducir Consultas
•Una vez que se ha logrado la conexión con el servidor ya se puede empezar a hacer consultas. Por ejemplo:
•Si durante la introducción de un comando se decide que no se quiere ejecutar, se cancela \c:

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:

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:

•También se podría redirigir la salida a otro fichero.

SQL: Structured Query Languag

Existe un lenguaje que permite la comunicación con el Sistema de Gestión de la Base de Datos (DBMS). **SQL: Structured Query Language** es un lenguaje estándar de comunicación con bases de datos.

El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. Determinadas bases de datos implementan tipos, operadores y funciones específicas, pero el núcleo de ordenes del lenguaje es universal.

Aparte de esta universalidad, el SQL presenta una potencia y versatilidad notables que contrasta, por otra, con su accesibilidad de aprendizaje. Además permite el uso de plataformas cliente/ servidor y es un lenguaje que puede utilizarse tanto en línea de comandos, como embebido.

HISTORIA Y ESTANDARIZACION

• El Leguaje SQL fue diseñado sobre un prototipo de IBM que incluía un lenguaje de consulta a base de datos llamado SEQUEL.

• El Instituto de estandarización ANSI y más tarde la ISO adoptan SQL como el lenguaje estándar de consulta a bases de datos. En la actualizad el último estándar definido es del año 2008.

• Aunque el ANSI/ISO SQL es el estándar “de facto”, ORACLE presentó su propia versión comercial, y más tarde lo han hecho Microsoft (SQL SERVER) y otras empresas. En la actualidad el lenguaje MySQL de libre distribución y su servidor es el más utilizado en servidores de bases de datos de Internet.

TIPOS DE SENTENCIAS

Como se ha expuesto en las unidades anteriores, hay tres tipos de lenguajes de gestión de Bases de Datos: DDL, DML y DCL.

Todas las estructuras **DDL** tienen asociadas tres operaciones:

CREATE: CREACION DE LA ESTRUCTURA
DROP: BORRADO DE LA ESTRUCTURA
ALTER: MODIFICACION DE LA ESTRUCTURA

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 Manipilation 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:

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

CREACIÓN DE BASES DE DATOS

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

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

Usar BD: USE

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.

MODIFICACIÓN DE UNA BASE DE DATOS: ALTER

ALTER {DATABASE | SCHEMA} [db_name] alter_specification [, alter_specification] ...
Nota: alter_specification:
[DEFAULT] CHARACTER SET charset_name | [DEFAULT] COLLATE collation_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:

CREACION DE UNA TABLA

CREATE TABLE [esquema.]<nombre_tabla>(campo tipo [NOT NULL], [....]);

Especifica la definición de los campos que va a contener la tabla y sus restricciones. CREATE TABLE crea una tabla con el nombre dado.
Las definiciones individuales de columnas se separan mediante comas. No se pone coma después de la última definición de columna. Hay que crear las tablas en el orden adecuado. Es decir, si en una tabla se hace referencia un campo de otra tabla, es imprescindible que se cree primero la tabla que contiene el campo original y después la tabla dependiente.
Se escribe el nombre de la columna y después el tipo de dato.

CONSULTA.

COMO SE PROCESA UNA SENTENCIA

La sentencia sintácticamente: Analiza que esta bien escrita y sigue el formato estándar. Valida la sentencia: Comprueba que existen todas las tablas y campos implicados. Optimiza y analiza las instrucciones que se han de realizar. Ejecuta la orden. Muestra por pantalla el resultado y el número de filas encontradas.

TIPOS DE CAMPOS

Es importante especificar qué tipo de valor tiene cada campo de manera que se facilite la búsqueda posteriormente y además se valide su contenido. Los tipos o dominios más comunes son los siguientes:

• Alfanuméricos: Contienen cifras y letras.

• Numéricos: Enteros (sin decimales) y reales (con decimales).

• Booleanos: Verdadero y falso (Sí o No)

• Fechas: Almacenan fechas facilitando posteriormente su explotación

• Memo: Son campos alfanuméricos de longitud ilimitada

COMANDO SELECT

En un DBMS relacional todo debe estar contenido en tablas, por lo que todo puede consultarse si se conoce el nombre de la tabla a consultar, la estructura de esta y si se tiene permiso.

Para **todas** las consultas SQL utiliza el comando **SELECT**. Su formato completo es el siguiente:

SELECT: (Obligatoria) Palabra clave que indica que la sentencia de SQL que queremos ejecutar es de selección.

ALL: Indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca. Se pede sustituir por un asterisco.

DISTINCT: Indica que queremos seleccionar sólo los valores distintos.

FROM: (Obligatoria) Indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta “consulta combinada" o "join". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR.

GROUP BY: Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas. La veremos más adelante. • HAVING :Especifica una condición que debe cumplirse para los datos. La veremos más adelante Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

ORDER BY: Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

Si un usuario quiere saber las tablas que hay en un esquema concreto que esta usando, puede ejecutar **SHOW TABLES.**

Existe el comando **DESCRIBE <tabla>** que nos muestra la estructura de una tabla.

INTEGRIDAD DE DATOS

La integridad hace referencia al hecho de que los datos de la base de datos han de ajustarse a restricciones antes de almacenarse en ella. Por ello han de quedar claros dos puntos:

1. Una restricción de integridad restringe el rango de valores para una o más columnas en la tabla. Ej. NOT NULL obliga a que un campo nunca sea nulo.

2. Tanto las sentencias de actualización como las de borrado desencadenan esas mismas operaciones en cascada para los diferentes campos o tablas afectadas por la consulta.

FORMAS DE ESPECIFICAR LAS RESTRICCIONES AL CREAR UN TABLA:Restricciones por campo y Restricciones con nombre (CONSTRAINT)

Restricciones de campo son aquellas que pueden especificarse detrás del campo afectado
NOT NULL: Obliga a que ese campo tenga un valor siempre.

EJEMPLO: NOMBRE VARCHAR(30) NOT NULL,

DEFAULT: Asigna un valor por defecto al campo

PRIMARY KEY: Permite asignar a un campo el atributo de campo clave que identificara de forma univoca al registro. Esta forma sólo permite que la clave primaria sea un solo campo.

EJEMPLO: NIF VARCHAR2(10) PRIMARY KEY,

Si hay varios campos que forman parte de la clave primaria, añadirlas al final, tras los campos de la tabla

REFERENCES : Permite especificar que ese campo es una clave ajena que hace referencia a un campo de otra tabla, para ello se especifica el nombre de la tabla y entre paréntesis el campo al que hace referencia: REFERENCES TABLA(CAMPO).

Hay dos tipos de motores fundamentales en MYSQL:
MYISAM: Muy rápido de acceso pero en los que no se controla la integridad referencial
InnoDB: Más lento pero respeta a integridad referencial
Para saber que tipo de motor (engine) usa nuestra tabla hay que consultar el diccionario de datos en la base de datos information_schema, en concreto la tabla tables.

UNIQUE : Obliga a que un campo no se repita, sin ser necesariamente el campo clave. Son las claves candidatas que no llegaron a ser primarias.

EJEMPLO
NIF VARCHAR2(10) UNIQUE,

CHECK: Realiza un chequeo o comprobación del valor introducido a través de una sencilla condición.

Restricciones especificas CONSTRAINT:

Para ello usamos tras la definición de campos, la cláusula CONSTRAINT, que puede restringir una sola columna o un grupo de columnas de una misma tabla.

Formato:
CREATE TABLE NOMBRE_TABLA(COLUMNA1 TIPO_DE_DATO,{CONSTRAINT NOMBRE_RESTRICCION RESTRICCION})

Como hemos visto anteriormente, es posible no poner constraint y de esta forma no se pone nombre a la restricción de forma que quedaría definida como restricción de campo, pero la forma más recomendada por la calidad de la programación es hacerlo con constraint especificando un nombre significativo a la restricción. Poner un nombre a la restricción nos va a permitir poder manipularla posteriormente y tenerla documentada de forma más adecuada.

CONSTRAINT Primary key

Constraint en español significa restricción. Se usa para identificar una columna o un conjunto de columnas que identifican unívocamente a cada fila, es decir se usa para describir la columna de la Primary Key. Debe ser única, no nula y obligatoria. Como máximo, podemos definir una clave

primaria por tabla. Esta clave se puede referenciar por una columna o por varias columnas. Cuando se crea una clave primaria, automáticamente se crea un índice que facilita el acceso a la tabla.

CREATE TABLE ALUMNO (MATRICULA VARCHAR2(10) NOT NULL,NOMBRE VARCHAR2(30), CONSTRAINT PK_ALUMNO PRIMARY KEY(MATRICULA));

En todo caso, la más recomendada es la primera, ya que al ocurrir algún error en las restricciones el error mostrado especificaría el nombre de está: PK_ALUMNO y sabríamos que se refiere a la clave primaria de ALUMNO

Si tuviéramos varios campos, por ejemplo, una tabla en la que se almacenan las practicas de los alumnos y su nota, deberíamos usar el código:

CREATE TABLE PRACTICAS (MATRICULA VARCHAR2(10) NOT NULL,PRACTICA VARCHAR2(30) NOT NULL,FECHA_ENTREGA DATE,NOTA NUMBER(3,1) CONSTRAINT PK_PRACTICAS PRIMARY KEY(MATRICULA, PRACTICA));

CONSTRAINT Foreign Key:

Se usa para definir por una o varias columnas que están asociadas a una clave primaria de otra tabla. Se pueden definir tantas claves ajenas como se precise. El valor de la columna o columnas que son claves ajenas debe ser: NULL o igual a un valor de la clave referenciada (regla de integridad referencial). Si una clave puede ser NULL, equivale a las cardinalidad (0,x) que vimos en las unidades anteriores.

Sintaxis:
CONSTRAINT NOMBRE_RESTRICCIÓN
FOREIGN KEY(CAMPO) REFERENCES NOMBRETABLA (CAMPO ORIGEN)
[ON DELETE ACCION] [ON UPDATE ACCION]);

En la cláusula REFERENCES indicamos la tabla a la cual remite la clave ajena.
Hay que crear primero una tabla y después aquella que le hace referencia.
Hay que borrar primero la tabla que hace referencia a otra tabla y después la tabla que no hace referencia.

NO ACTION: Si añado despues de la referencia no action no puedo eliminar/actualiza registros de una tabla con campos a los que hacen referencia valores de otra tabla.. Es la opción por defecto.
On CASCADE: Si añado despues de la referencia cascade se eliminaran/actualizarán en casacada los registros afectados por la referencia

SET NULL: Si añado despues de la referencia set null se pondran a NULL los campos de los registros afectados por la referencia

PRIMARIAS MULTIPLES, FORANEAS MULTIPLES

Si en una tabla hay una referencia que apunta a otra tabla con clave primaria formada por varios campos, es imprescindible que en la tabla referencia existan todos los campos referenciados y que se incluyan en la restriccion de foreign key

EJEMPLO: Imagina una tabla1 con clave primaria compuesta por t1campo1 y t1campo2
PRIMARY KEY (t1campo1, t1campo2)

Para que tabla2 referencia a tabla1 debo incluir los campos en la restricción de la clave primaria de la misma forma
FOREIGN KEY(t2campo1,t2,campo2) REFERENCES tabla1(t1campo1,t1campo2)

En Mysql encontramos esa información en information_schema.table_constraints.

BLOQUE 1_ ENTORNOS DE DESARROLLO

MYSQL Es un sistema muy flexible, portable y sencillo de instalar. De hecho es el más instalado en alojamientos web. Posee un sistema de ficheros que separa ejecutables (directorio bin) y datos (directorio data). Podemos ver su localización en el fichero my.ini de inicialización.

Es imprescindible conocer dónde se sitúan estos dos directorios, ya que en el primero encontraremos el cliente (mysql) y los ficheros ejecutables necesarios para iniciar y gestionar el servidor; y en el segundo encontraremos las bases de datos en formato de carpeta que vayamos creando. Además si hemos hecho una instalación estándar nos habrá creado un servicio que podemos parar e iniciar con la herramienta Servicios de Windows (services.msc)

Para conectarnos como root desde MSDOS: mysql -u root -p

· Para crear una Base de Datos: CREATE DATABASE nombre_basededatos;
· Para borrar una Base de Datos: DROP DATABASE nombre_basededatos;
· Para mostrar las bases de datos existentes: SHOW DATABASES;
· Para entrar/trabajar/usar una base de datos: USE nombre_basededatos;
· Para ver las tablas de la base de datos actual: SHOW TABLES;
· Para ver la descripción /estructura de una tabla: DESC nombre_tabla
Cada esquema posee sus propios objetos, y son manipulados por los usuarios con permisos sobre ellos.

INICIACION A LA CREACION Y MANIPULACION DE TABLAS

Crear tablas: CREATE TABLE [esquema.]<nombre_tabla>(nombre_campo1 tipo [RESTRICCION1], [RESTRICION2], ETC...,nombre_campo1 tipo [RESTRICCION1], [RESTRICION2], ETC...);

Características:

· Las definiciones individuales de columnas se separan mediante comas.
· No se pone coma después de la última definición de columna.

Borrar tablas: DROP TABLE [USUARIO].NOMBRETABLA

Insertar datos: INSERT INTO table_name(column1,column2,column3,...) VALUES (value1,value2,value3,...);

Si quiero todos los campos puedo omitir especificarlos
INSERT INTO table_name VALUES (value1,value2,value3,...);

Actualizar Datos: UPDATE table_name

SET column1=value1,column2=value2,… WHERE some_column=some_value;

Borrar datos: DELETE FROM table_name WHERE some_column=some_value;

Consultar datos: SELECT column_name,column_name FROM table_name
WHERE column_name operator value;

Tema 5 : Diseño físico de Bases de Datos. Realización de Consultas. DML 1

ÍNDICE

1.- El lenguaje DML

Las sentencias DML del lenguaje SQL son las siguientes:
La sentencia **SELECT**, que se utiliza para extraer información de la BD, ya sea de una tabla o de varias.
La sentencia **INSERT**, para insertar uno o varios registros en una tabla.
La sentencia **DELETE**, que borra registros de una tabla.
La sentencia **UPDATE**, que modifica registros de una tabla.

Cualquier ejecución de un comando en un SGBD es una consulta o Query, es decir, cualquier orden o petición que se realiza al SGBD para realizar una operación determinada ya sea de consulta, inserción, borrado o actualización es una Query o consulta.

SELECT es la sentencia más versátil de todo el SQL. El parámetro opcional DISTINCT fuerza a que solo se muestren los registros con valores distintos, es decir, que se supriman las

repeticiones. El formato básico para hacer una consulta es: **SELECT [DISTINCT]** ***select_expr*** **[,** ***select_expr*** **...]**
[FROM tabla]

2.- Filtros

Filtro es una expresión que indica la condición o condiciones que deben satisfacer los registros para ser seleccionados. Son condiciones que cualquier gestor de BD interpreta para seleccionar registros y mostrarlos como resultado de la consulta. En SQL la palabra clave es **WHERE**., que en español significa **"donde…", o "en el caso de…"**

SELECT [DISTINCT] ***select_expr*** **[,** ***select_expr*** **...]**
[FROM tabla] [WHERE filtro]

Los elementos que pueden formar parte de las expresiones son:

•Operandos (constantes o variables)
•Operadores Aritméticos (+, -. *, /, %)
•Operadores Relacionales (<, >, <=, >=, =,<>
•Operadores Lógicos (AND, OR, NOT)
•() para dar máxima prioridad a la expresión
•Funciones: date_add, concat, left, rigt …

2.1.- Filtros con operadores de pertenencia a conjuntos

2.2.- Filtros con operador de rango

El operador de rango BETWEEN permite seleccionar los registros que estén incluidos en un rango. Por ejemplo, seleccionar a los jugadores de la NBA cuyo peso está entre 275 y 300 libras:

2.3.- Filtros con test de valor nulo

Los operadores IS e IS NOT permiten verificar si un campo es o no es nulo respectivamente. Por ejemplo, determinar los jugadores cuya procedencia es desconocida:

2.4.- Filtros con test de patrón

Seleccionan los registros que cumplan una serie de características. Se pueden usar los caracteres comodines % y_ para buscar una cadena de caracteres. Por ejemplo, seleccionar los jugadores cuya procedencia incluye la palabra 'monte':

2.5.- Filtros por límite de número de registros

Este tipo de filtros no es estándar y su funcionamiento varía con el SGBD. Consiste en limitar el número de registros devuelto por una consulta. Nfilas especifica el número de filas a devolver y desplazamiento especifica a partir de qué fila se empieza a contar.

Por ejemplo, visualizar las 4 primeras filas de la tabla jugadores:

[LIMIT [desplazamiento,] nfilas]

3.- Ordenación

Para mostrar ordenados un conjunto de registros se utiliza la cláusula ORDER BY.
Esta cláusula permite ordenar el conjunto de resultados de forma ascendente (ASC) o descendente (DESC) por una o varias columnas. Si no se especifica nada, por defecto es ASC. La columna por la que se quiere ordenar puede especificarse por su nombre, posición o por una expresión numérica.

SELECT [DISTINCT] ***select_expr*** **[,** ***select_expr*** **...]**
[FROM *tabla*
[WHERE filtro]
[ORDER BY {*nombre_columna*l *expr* l *position*} [ASC l DESC], ...]

4.- Consultas de resumen: Funciones columna
Permiten extraer información calculada de varios conjuntos de registros. Por ejemplo, obtener el número de registros de la tabla jugadores, o el peso máximo o medio.

SUM (Expresión) # Suma los valores indicados en el argumento
AVG (Expresión) # Calcula la media de los valores
MIN (Expresión) # Calcula el mínimo
MAX (Expresión) # Calcula el máximo
COUNT (nbColumna) # Cuenta el número de valores de una columna (excepto los nulos)
COUNT (*) # Cuenta el número de valores de una fila (incluyendo los nulos)

5.- Agrupaciones de Registros

Con las consultas de resumen se pueden realizar agrupaciones de registros, es decir, conjuntos de registros que tienen una o varias columnas con el mismo valor. A este grupo de registros se le puede aplicar una función de columna para realizar determinados cálculos, por ejemplo, contarlos:

SELECT [DISTINCT] *select_expr* [, *select_expr* ...]
[FROM tabla]
[WHERE *filtro*]
[GROUP BY {expr [,expr] …}]
[ORDER BY {*nombre_columna* l *expr* l *position*} [ASC l DESC], …]

Para cada agrupación, se ha seleccionado también el nombre de la columna por la cual se agrupa. Esto no es posible si no se incluye el GROUP BY.

Cuando agrupamos, en la línea select, sólo pueden aparecer campos por los que agrupamos, o funciones de columna. En el **having** no podremos hacer referencia a campos por los que no agrupamos. Se puede agrupar por campos que no son mostrados en la línea **select**.

5.1.- Filtros de Grupos

Los filtros de grupos deben realizarse mediante el uso de la cláusula HAVING puesto que WHERE actúa antes de agrupar los registros. Es decir, si se desea filtrar resultados calculados mediante agrupaciones se debe usar la siguiente sintaxis:

SELECT [DISTINCT] ***select_expr*** **[,** ***select_expr*** **...]**
[FROM *tabla*]
[WHERE filtro]

[GROUP BY {*expr* [, expr]…}
[HAVING filtro_grupos]
[ORDER BY {*nombre_columna*l *expr* l *posición*} [ASC l DESC], ...]

6.- Subconsultas

Se utilizan para realizar filtrados con los datos de otra consulta. Estos filtros pueden aplicarse en la cláusula WHERE para seleccionar registros y en la HAVING para filtrar grupos.

Ejemplo, codificar una consulta para ver los nombres de los jugadores de la división SouthWest:

Una sentencia subordinada de otra puede tener a su vez otras sentencias subordinadas a ella. Se llama sentencia externa a la primera de todas, la que no es subordinada de ninguna. Una sentencia es antecedente de otra cuando ésta es su subordinada directa o subordinada de sus subordinadas a cualquier nivel. A las sentencias subordinadas también se les llama anidadas.

Las subordinadas pueden ser parte de los siguientes predicados:
✓Predicados básicos de comparación
✓Predicados cuantificados (ANY,SOME, ALL)
✓Predicados EXISTS
✓Predicado IN

6.1.- Test de Comparación

Consiste en utilizar los operadores de relación =, >, >=, <, >=, <> para comparar el valor producido con un valor único generado por una subconsulta. Por ejemplo, para obtener el nombre del jugador de la NBA de mayor altura:

La subconsulta produce un único resultado. La subconsulta debe estar siempre al lado derecho del operador de comparación. **Campo <= subConsulta**

6.2.- Test de pertenencia a un conjunto

Consiste en utilizar el operador **IN** para filtrar los registros cuya expresión coincida con algún valor producido por la subconsulta.

6.3.- Test de Existencia

Permite filtrar los resultados de una consulta si existen filas en la consulta asociada, e.d, si la subconsulta genera un número de filas distinto de 0.

SELECT columnas FROM *tabla* WHERE EXISTS (subconsulta)

Test de No Existencia

Es como si cada registro devuelto por la consulta principal provocara la ejecución de la subconsulta, si la consulta principal devuelve por ejemplo, 30 registros, se ejecutarían 30 subconsultas, pero en realidad el SGBD realiza sólo dos consultas y una operación de JOIN.

6.4.- Test cuantificados ALL y ANY

•Sirven para calcular la relación entre una expresión y todos los registros de la subconsulta (ALL) o algunos de los registros de la subconsulta (ANY o SOME).
•Por ejemplo, obtener todos los jugadores de la NBA que pesan más que todos los jugadores españoles

6.5.- Subconsultas anidadas

Se puede usar una subconsulta para filtrar el resultado de otra subconsulta. Por ejemplo, obtener el nombre de la ciudad donde juega el jugador más alto de la NBA:

Los pasos serían:
1/ Obtener la altura máxima:
Select max (altura) from jugadores;
2/ Obtener el nombre del equipo , a través de la altura se localiza al jugador y por tanto el nombre de su equipo
3/ Obtener la ciudad del equipo

7.- Consultas multitablas o Join

Es aquella en la que se puede consultar información de más de una tabla. Se aprovechan los campos relacionados de las tablas para unirlas (join).

SELECT [DISTINCT] *select_expr* [, *select_expr* ...]
[FROM referencias_tablas]
[WHERE filtro]
[GROUP BY {*expr* esión [,expresión]…}
[HAVING filtro_grupos]
[ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...]

La diferencia con las consultas sencillas se halla en la cláusula FROM. En vez de una tabla se puede desarrollar el token referencias_tablas:

referencias_tablas: referencia_tabla [, referencias_tabla] … | referencia_tabla [INNER | CROSS] JOIN referencia_tabla [ON Condición] | referencia_tabla LEFT [OUTER] JOIN referencia_tabla ON Condición | referencia_tabla RIGHT [OUTER] JOIN referencia_tabla ON Condición referencia_tabla: Nombre_tabla [[AS] alias]

La primera opción, referencia_tabla[, referencia_tabla]… es típica de SQL1 para las uniones, que consiste en un producto cartesiano más un filtro por las columnas relacionadas, y el resto de opciones son propias de SQL2.

El producto cartesiano de dos tablas son todas las combinaciones de las filas de una tabla unidas a las filas de la otra tabla. Por ejemplo, en la BD de veterinarios con dos tablas mascotas y propietarios. La operación genera un conjunto de resultados con todas las combinaciones posibles entre las filas de las dos tablas, y con todas las columnas.

PRODUCTO CARTESIANO+ FILTRO

Si se aplica un filtro al producto cartesiano para obtener sólo las filas en las que el campo DNI coincida se obtendría:

SQL2 introduce las joins o composiciones internas, externas y productos cartesianos (también llamadas composiciones cruzadas):

1/ JOIN INTERNA
De Equivalencia (INNER JOIN)
Natural (NATURAL JOIN)

2/ PRODUCTO CARTESIANO (CROSS JOIN)
3/ JOIN EXTERNA
De tabla derecha (RIGHT OUTER JOIN)
De tabla izquierda (LEFT OUTER JOIN)
Completa (FULL OUTER JOIN)

JOIN INTERNA De Equivalencia (INNER JOIN)

Hay dos formas de expresar la INNER JOIN o Composiciones internas: usando la palabra reservada JOIN o separando por coma las tablas a combinar en la sentencia FROM. Con esta operación se calcula el producto cartesiano de todos los registros, después cada registro en la primera tabla es combinado con cada registro de la segunda tabla, y sólo se seleccionan aquellos registros que satisfacen las condiciones que se especifican. Los valores nulos no se combinan.
•Por ejemplo, de todos los registros de la tabla de mascotas encontrar todas las combinaciones en la tabla de propietarios en los que el DNI coincida.
JOIN INTERNA De Equivalencia (INNER JOIN)

Hay que tener en cuenta que si hay un animal sin propietario no saldrá en el conjunto de resultados puesto que no tiene coincidencia en el filtro. En el ejemplo introducimos un animal sin propietario:

Composiciones Naturales (NATURAL JOIN)

•Es una especialización de la INNER JOIN. En este caso se comparan todas las columnas que tengan el mismo nombre en ambas tablas, la tabla resultante tiene solo una columna por cada par de columnas con el mismo nombre.
•EJEMPLO: BD Jardineria

Composiciones Externas OUTER JOIN

Las tablas relacionadas no requieren que haya una equivalencia. El registro es seleccionado para ser mostrado aunque no haya otro registro que le corresponda. Outer JOIN se subdivide dependiendo de la tabla a la cual se admitirán los registros que no tienen correspondencia, ya sean de tabla izquierda, de tabla derecha o combinación completa. Si los registros que admiten no tener correspondencia son los que aparecen en la tabla de la izquierda se llama composición de tabla izquierda o LEFT JOIN (0 LEFT OUTER JOIN).

Ejemplo RIGHT OUTER JOIN
•Si los registros que admiten no tener correspondencia son los que aparecen en la tabla de la derecha, se llama composición de tabla derecha o RIGHT JOIN

Ejemplo FULL OUTER JOIN Composición externa completa

Esta operación admite registros sin correspondencia tanto para la tabla izquierda como para la derecha, e.d, animales sin propietarios y propietarios sin animales. Presenta valores nulos para los registros sin pareja.

Como se observa, MySql no implementa FULL OUTER JOIN. En SQL existe el operador UNION, que añade al conjunto de resultados producidos por una SELECT, los resultados de otra SELECT SELECT … FROM … UNION [ALL] SELECT … FROM … El parámetro ALL incluye todos los registros de las dos DELECT, incluyendo los que son iguales. Si no se indica ALL, se excluyen los duplicados.

Composición externa completa, mediante UNION
MySql simula FULL OUTER JOIN, haciendo una UNION de los resultados de un LEFT OUTER JOIN y los resultados de un RIGHT OUTER JOIN, ya que UNION, sin la opción ALL, elimina los registros duplicados, por tanto, se podría codificar:

8.- Consultas Reflexivas

A veces, es necesario obtener información de relaciones reflexivas, por ejemplo un informe de empleados donde junto a su nombre y apellidos apareciera el nombre y apellidos de su jefe. Para ello hay que hacer un JOIN entre registros de la misma tabla:

Se ha usado la tabla Empleados dos veces, una con un alias emp que representa los Empleados como Subordinados y otra con alias jefe que representa los Empleados como Jefes. Ambas tablas (aunque en realidad es una sóla) se unen en una JOIN a través de la relación CodigoEmpleado y CodigoJefe.
Además el primer campo seleccionado es la concatenación del nombre y apellido del Empleado al que se le da un alias (NEMPLEADO) y lo mismo con (NJEFE).
Se observa que en esta query no aparecen empleados sin jefe, ya que se ha utilizado un INNER JOIN.

9- Consultas con Tablas Derivadas

Las consultas con Tablas Derivadas o inline views, son aquellas que utilizan sentencias select en la cláusula FROM en lugar de nombres de tablas:

La tabla derivada (select CodigoEmpleado, nombre FROM Empleados) tiene un alias 'tabla_derivada'. Es como una especie de tabla temporal resultado de ejecutar la consulta.

Por ejemplo, en la BD jardinería, si se quiere obtener el importe del pedido de menor coste de todos los pedidos. Primero se obtiene el total de todos los pedidos y luego el pedido de menor coste con la función de columna MIN:

Para determinar cuál es el código de pedido que le corresponde al importe menor de todos los pedidos:

```
mysql> select codigopedido from detallepedidos group by codigopedido having
sum(cantidad*preciounidad)=
(select min(total)from(select sum(cantidad*preciounidad)as total,codigopedido from detallepedidos
group by codigopedido) as tablad);
```

SOPA DE CONCEPTOS (repetidos en su mayoría)

•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

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.

•Ver los motores disponibles:

mysql> show engines;
•Cambiar el motor por defecto al arrancar el servidor:

shell# mysqld_safe --user=mysl –-default-storage-engine=Innodb
shell# mysqld_safe --user=mysl –-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

•Si no se especifica NULL ni NOT NULL, la columna se trata como si se especificara NULL .
•En la sintaxis de CREATE TABLE, [Definiciónreferencia] 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
•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 [NombreIndex] (nombreColumna,...) [DefiniciónReferencia]
En SQL, las restricciones pueden tener un nombre, para ello se utiliza CONSTRAINT [símbolo]

CREAR TABLA

•OpcionesTabla permite especificar las peculiaridades de cada gestor con respecto al almacenamiento físico de sus tablas.
•OpcionesTabla :
– {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

•Para visualizar la estructura de una tabla:
Describe [esquema.]NombreTabla;

•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.

•varían con cada SGBD, sirven para modificar las características del almacenamiento físico.

•Renombrar una columna:

•Para hacer que la clave primaria sea auto_increment:

Ejemplo Mascotas

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

Añadir una clave primaria que sea autoincremetada. Hacer que el campo nombre sea único.

Ejemplo Mascotas

Modificar el campo nombre, para que tenga 30 caracteres y no sea optativo.
Añadir el campo pedigree después de raza.
Cambia el campo fnac por fechaNacimiento.

•Modificar el campo especie, para que por defecto sea ‘canina’:

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 tablaMaeTO backupTablaMae;
DROP table tablaMae , 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.

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

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

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 restauran 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.

•-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.

•-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.

-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 se seguridad de la base de datos mibase al fichero copia_seguridad.sql

mysqldump --opt --password=miclave --user=miuser mibasededatos > archivo.sql

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

RESTRICCIONES:

Que todas las tablas tengan el motor de almacenamiento innodb para que las claves foráneas no sean ignoradas.