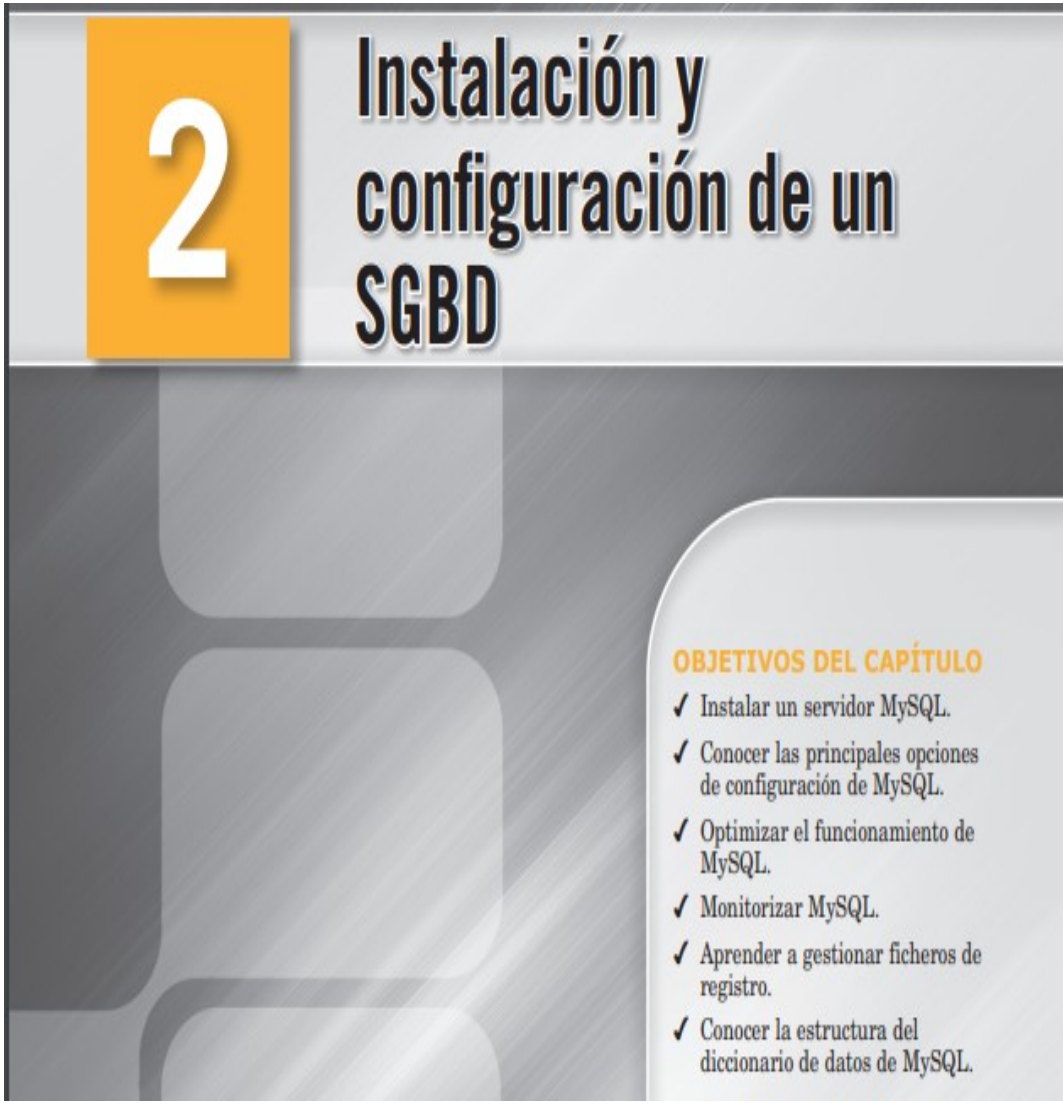
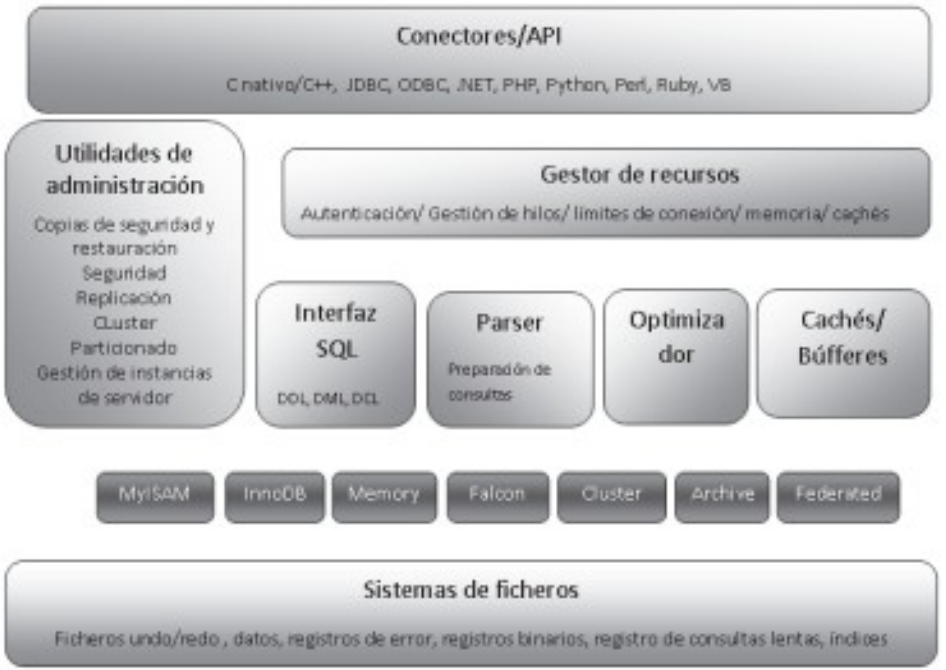
UT1 - Unidad 1
Instalación y configuración de un sistema gestor de base de datos



2.2 ARQUITECTURA MYSQL

El esquema general de organización de MySQL responde a la siguiente figura donde se representan los distintos subsistemas del sistema gestor de MySQL que permiten llevar a cabo todas las funciones del administrador de la base de datos.

Importante



2.3 INSTALACIÓN DE MYSQL

2.3.1 CUESTIONES GENERALES SOBRE LA INSTALACIÓN

Antes de instalar MySQL, debes tener en cuenta lo siguiente:

Plataforma

Que usarás en la instalación. Típicamente entre sistemas Windows o Linux. En la actualidad existen versiones disponibles para ambos.

Distribución

El desarrollo de MySQL se divide en entregas (*releases*) sucesivas, y el usuario puede decidir cuál es la que mejor satisface sus necesidades. Después de haber elegido la versión a instalar, se debe optar por un formato de distribución. Las entregas están disponibles en formato binario o código fuente.

Integridad

Descargar la distribución que se desea instalar y por seguridad verificar su integridad.

Versión

Escoger la versión de MySQL a instalar. La primera decisión a tomar es si se desea emplear una entrega "en producción" (estable) o una entrega de desarrollo. En el proceso de desarrollo de MySQL coexisten múltiples entregas, cada una con un diferente estado de madurez. En el momento de escribir este libro la versión más estable es la 5.1 aunque está en desarrollo y pruebas la 6.0. Es conveniente siempre trabajar con la estable, especialmente si nuestra intención es usarlo en entornos productivos. Para más información sobre versiones, consultar la bibliografía detallada al final del libro.

Formato

Escoger un formato de distribución. Después de haber decidido qué versión de MySQL instalar, debes elegir entre una distribución binaria o una de código fuente. Probablemente la elección más frecuente sea la distribución binaria por su facilidad de instalación, salvo que queramos configurar MySQL incluyendo o excluyendo algunas características de las distribuciones binarias estándar.

Hay disponibles tres tipos de instalación: típica, completa y personalizada.

La instalación típica instala el servidor MySQL, el cliente de línea de comandos *mysql*, y las utilidades de línea de comandos. Los clientes y utilidades incluyen *mysqldump*, *myisamchk* y otras herramientas que facilitan la administración del servidor.

La instalación completa instala todos los componentes incluidos en el paquete. El paquete completo incluye componentes como el servidor incrustado (*embedded*), el conjunto de pruebas de rendimiento *(benchmarks)*, *scripts* de mantenimiento y documentación.

La instalación personalizada otorga un control completo sobre los paquetes que se desean instalar y el directorio de instalación que se utilizará.

Dado que somos administradores usaremos la instalación personalizada de manera que podamos adaptar mejor el software a nuestras necesidades.

Una vez descargado el software de la página, comenzamos la instalación pulsando dos veces con el ratón en el ejecutable y, a partir de ahí, se nos muestra un cuadro de diálogo preguntándonos por el tipo de instalación.

Una vez que se pulsa con el ratón en el botón **Instalar**, el asistente de instalación de MySQL comienza el proceso de instalación y realiza ciertos cambios en el sistema, que se describen a continuación:

Cambios en el Registro

Leer

Se crea una clave de registro durante una situación típica de instalación, localizada en *HKEY_LOCAL_MACHINE\SOFTWARE\MySQL AB*. También crea una clave cuyo nombre es el número de versión principal (número de la serie) del servidor que se encuentra instalado, como MySQL *Server* 5.0. Contiene dos valores de cadena, *Location* y *Version*. La cadena *Location* contiene el directorio de instalación. La cadena *Version* contiene el número de entrega (*release*).

Estas claves del registro son útiles para ayudar a herramientas externas a identificar la ubicación en la que se instaló el servidor MySQL, evitando un rastreo completo del disco para descubrirla.

Cambios en el menú Inicio

El asistente de instalación crea una nueva entrada en el menú *Inicio* de Windows, bajo una opción cuyo nombre es el número de versión principal (número de la serie) del servidor que se encuentra instalado.

Se crean las siguientes entradas dentro de la nueva sección del menú *Inicio*:

- MySQL *Command Line Client*: cliente de línea de comandos.
- MySQL *Server Instance Config Wizard*: asistente de configuración.
- MySQL *Documentation*: es un vínculo a la documentación del servidor.

Cambios en el sistema de ficheros

El asistente de instalación de MySQL, por defecto instala el servidor MySQL en *C:\Program Files\MySQL\MySQL Server 5.0*, donde *Program Files* es el directorio de aplicaciones por defecto del sistema, y 5.0 es el número de versión principal del MySQL instalado. Ésta es la nueva ubicación donde se recomienda instalar MySQL, en sustitución de la antigua ubicación por defecto, *C:\mysql*.

Por defecto, todas las aplicaciones MySQL se almacenan en un directorio común localizado en *C:\Archivos de Programa\MySQL*, donde *Program Files* es el directorio de aplicaciones por defecto del sistema. Una instalación típica de MySQL en el ordenador de un desarrollador podría verse así:

Leer

```
C:\Archivos de Programa\MySQL\MySQL Server 5.0
C:\Archivos de Programa\MySQL\MySQL Administrator 1.0
C:\Archivos de Programa\MySQL\MySQL Query Browser 1.0
```

Donde los dos últimos directorios indican la ubicación de las herramientas gráficas (GUI) de administración y generación de consultas.

(En versiones superiores a la 5.0 estos dos últimos directorios corresponden al actual MySQL *Workbench* que integra el *administrator*, el *browser*, *migration toolkit* y añade una herramienta de diseño de bases de datos).

Esta proximidad entre los distintos directorios facilita la administración y el mantenimiento de todas las aplicaciones MySQL instaladas en un sistema en particular.

Es muy importante para un administrador tener clara la ubicación de directorios y archivos tras una instalación de cualquier programa, especialmente de servicios que serán accesibles por distintos clientes.

El directorio de instalación contiene los subdirectorios indicados en la siguiente tabla:

Tabla 2.1 Conformación directorios instalación MySQL

Directorio	Contenido
bin	Programas cliente y el servidor mysqld
data	Ficheros de registro y de bases de datos
Docs	Documentación
Examples	Programas y *scripts* de ejemplo
include	Ficheros de inclusión
lib	Bibliotecas
scripts	*Scripts* de utilidades
share	Ficheros con mensajes de error

2.4 CONFIGURACIÓN SERVIDOR

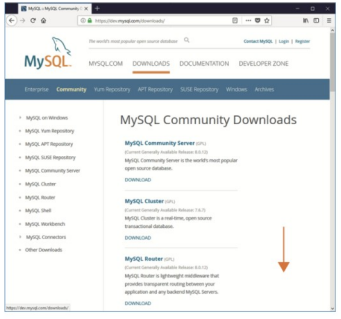
Tutorial

El asistente de configuración MySQL automatiza el proceso de configurar el servidor bajo Windows. Crea un fichero *my.ini* personalizado, realizando una serie de preguntas y aplicando las respuestas a una plantilla para generar un fichero *my.ini* apropiado para la instalación en curso.

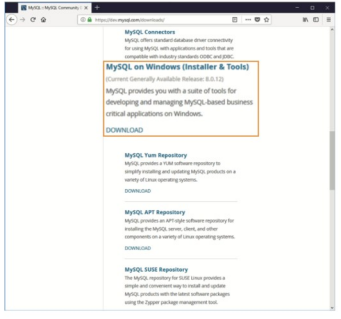
Guía Rápida:
Instalación MySQL Community Server con MySQL Workbench

MySQL

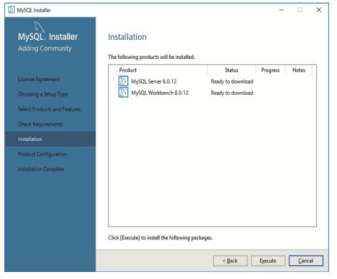
3.
Desplazar la ventana hasta hacer visible la opción **'MySQL on Windows(Installer & Tools)'**.



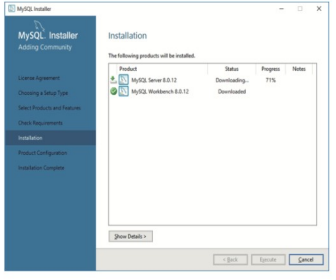
4.
Hacer clic en **'MySQL on Windows(Installer & Tools)'**.



10.
El instalador descargara e instalara los productos necesarios. Hacer clic en **'Execute'** para iniciar el proceso.



11.
Esperar a que termine el proceso de la descarga, puede tardar algunos minutos.



Diario de clase

Para visualizar las bases de datos:

EJEMPLO 2.1

```
C:\>Program Files\MySQL\MySQL Server 5.0\bin\mysqlshow

Salida:
+------------+
| Databases  |
+------------+
| mysql      |
| test       |
+------------+
```

Diario de clase

```
C:\Archivos de programa\mysql\MySQL Server 8.0\bin>mysqlshow -u root -p
Enter password: **********
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+

C:\Archivos de programa\mysql\MySQL Server 8.0\bin>
```

Diario de clase

Para visualizar la base de datos MySQL para la gestión de cuentas y permisos:

EJEMPLO 2.2

```
C:\> C:\mysql\bin\mysqlshow mysql
+---------------+
|    Tables     |
+---------------+
| columns_priv  |
| db            |
| func          |
| host          |
| tables_priv   |
| user          |
+---------------+
```

Diario de clase

Para comprobar las cuentas iniciales:

EJEMPLO 2.3

```
C:\> C:\mysql\bin\mysql -e "SELECT Host,Db,User FROM mysql.user"
C:\> C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqlshow -u root
```

Para comprobar el estado actual del servidor:

EJEMPLO 2.4

```
C:\> C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqladmin version status proc
```

Para conectarme a una base de datos directamente:

EJEMPLO 2.5

```
C:\> C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql test
```

Lo anterior son programas incluidos en el directorio *bin* de la instalación. Para un uso más detallado de estos programas podemos volcar su ayuda bien por pantalla con la opción *--help* como ya es habitual o bien enviándola a un fichero que luego podemos consultar. Es lo que hacemos con el siguiente comando para el programa *mysqladmin*:

EJEMPLO 2.6

```
C:\> C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqladmin -help>mysqladmin_
ayuda
```

2.4.2 VARIABLES Y OPCIONES EN MYSQL

El servidor dispone de un conjunto de variables que determinan su funcionamiento. Una de las tareas más importantes del administrador implica el conocimiento y ajuste óptimo de los valores de las mismas según los requerimientos de las aplicaciones.

Debemos diferenciar entre variables del servidor y las opciones que permiten modificar el valor de las variables.

Podemos ajustar los valores de las diferentes variables usando ficheros de opciones, incluyendo dichas opciones cuando arrancamos el servidor o modificándolas con el comando *SET*.

La mejor forma de conocer las variables es buscarlas cuando se necesitan. En tal caso debemos consultar el manual donde disponemos de una referencia detallada en donde para cada variable se detallan normalmente su nombre largo y corto para usar en línea de comandos, su nombre para ficheros de opciones (no siempre coincide), si son modificables con *SET*, el nombre de la variable, su alcance (global o de sesión) y si es o no dinámica (modificable en tiempo de ejecución), su dominio o valores permitidos, su tipo y su valor por defecto.

Cuando instalamos el servidor la mayor parte adquiere su valor por defecto, pero podemos modificarlas de tres modos distintos:

Opciones en línea de comandos

Cuando iniciamos el servidor (o cualquier otro programa de MySQL) podemos incluir valores para las variables directamente en la línea de comandos. La mayoría admiten una forma larga precedida por dos guiones, o corta, precedida por un guión. Algunas de ellas implican cierto valor, en cuyo caso serán seguidas por un igual y el valor correspondiente para el caso largo, o directamente por el valor (sin espacio en blanco) para el caso corto. Si los valores asignados incluyen espacios deben ir entre comillas.

El siguiente ejemplo ilustra lo dicho:

El siguiente ejemplo ilustra lo dicho:

EJEMPLO 2.7

```
#>mysql -uroot --password=root -e "show databases"
```

En este ejemplo usamos el programa cliente *mysql* para conectarnos con un usuario y contraseña especificados en la línea de comandos y ejecutar un comando con la opción *–e (execute)* para ejecutar un comando que nos mostrará las bases de datos en nuestro servidor.

EJEMPLO 2.8

```
#>mysqld -skip-grant-tables --console
```

Diario de clase

En este caso iniciamos el servidor sin tablas de permisos, de forma que cuando accede cualquier cliente lo podrá hacer sin necesidad de autenticarse. Además, los mensajes serán enviados a la consola.

Cuando se cambia una variable usando las opciones de arranque, los valores de la variable pueden darse con un sufijo K, M o G para indicar *kilobytes*, *megabytes* o *gigabytes*, respectivamente. Por ejemplo, el siguiente comando arranca el servidor con un tamaño de *key buffer* de 16 *megabytes*:

```
C:\>mysqld -key_buffer_size=16M
```

No importa que los sufijos se escriban en mayúscula o minúscula; 16 M y 16 m son equivalentes.

Leer

Ficheros de opciones

Cuando queremos que las opciones sean permanentes lo normal es hacer que los programas (*mysqld* entre ellos) de MySQL puedan leer opciones de inicio desde ficheros de opciones (también llamados ficheros de configuración). Estos proporcionan una forma conveniente de especificar opciones comúnmente usadas.

Los siguientes programas soportan ficheros de opciones: *myisamchk*, *myisampack*, *mysql*, *mysql.server*, *mysqladmin*, *mysqlbinlog*, *mysqlcc*, *mysqlcheck*, *mysqld_safe*, *mysqldump*, *mysqld*, *mysqlhotcopy*, *mysqlimport* y *mysqlshow*.

Sin embargo podemos unificar las opciones para todos ellos en un único fichero llamado *my.ini* para Windows o *my.cnf* en Linux.

Estos ficheros permiten incluir los siguientes elementos:

- Comentarios precedidos con #.
- Opciones para programas o grupos usando corchetes.
- Opciones tipo opción = valor.
- Opciones sin valor asociado.

Veamos su uso en el siguiente ejemplo:

EJEMPLO 2.9

```
[client]
port=3306
password="guara"

[mysqld]
port=3306
key_buffer_size=16M
max_allowed_packet=8M

[mysqldump]
quick

[mysqladmin]
force
```

Diario
de clase

En este caso, cuando iniciemos el servidor se pondrá a escuchar en el puerto 3306 los valores de memoria de claves y un máximo tamaño de paquete de 8 M.

Así mismo, todos los clientes deberán conectarse al puerto 3306 y su *password* será *guara*. Esta opción se aplicará a todos los clientes que se conecten al servidor.

Por último cuando hagamos una copia de seguridad usando *mysqldump* ésta se hará en modo rápido o *quick* y *mysqladmin* se ejecutará con la opción *force*.

Comando SET

En MySQL, podemos asignar un valor a una variable de sistema dinámica del servidor usando con el comando *SET* cuyos detalles estudiaremos en la siguiente sección.

Como resumen incluimos una tabla con los tipos de variables del servidor *mysql*:

Tabla 2.2 Tipos de variables en MySQL

Tipos	Descripción
Dinámicas-Estáticas	Según si son o no modificables en tiempo de ejecución.
Globales-De sesión	Según si afectan al comportamiento de todas las conexiones o solo a un cliente en particular.
De estado-De sistema	Según si se refieren al estado del servidor en un momento dado o a su comportamiento en general.

2.4.3 VARIABLES DE SISTEMA DEL SERVIDOR

Una vez instalado debemos revisar la conformación de directorios y muy especialmente el fichero *my.ini*. En él se recogen los parámetros que determinan el funcionamiento de nuestro servidor. Existen múltiples opciones para configurarlo, todas ellas perfectamente documentadas en el manual o la web de MySQL. En esta sección nos centraremos en las más relevantes y típicamente usadas.

Para modificar el fichero *my.ini*, se debe abrir con un editor de texto (recomiendo *notepad++* de descarga y uso libre: *http://notepad-plus-plus.org/*) y realizar cualquier cambio necesario. También se puede modificar con la utilidad *mysqladmin* comentada en próximos capítulos.

Para una lista de las variables del servidor podemos ejecutar *mysqld --verbose –help*, también podemos enviar la salida a un fichero si queremos guardarla y verlas con detenimiento, *mysqld --help > fichero_ayuda*.

A continuación veremos algunas de las variables más relevantes agrupadas según su funcionalidad:

Leer

Variables generales

- *-help, -?*

  Muestra un mensaje de ayuda corto y sale. Con las opciones *--verbose* y *--help* simultáneamente podemos ver el mensaje detallado.

- *-basedir=path*, -b *path*

  El *path* al directorio de instalación de MvSQL. Todas las rutas se resuelven normalmente relativas a ésta.

- *-bind-address*=IP

  La dirección IP asociada al servidor para cuando el equipo dispone de más de una IP.

- *-console*

  Escribe los mensajes de error por *stderr* y *stdout* incluso si *--log-error* está especificado. En Windows, *mysqld* no cierra la pantalla de consola si se usa esta opción.

- *-datadir=path, -h path*

  La ruta al directorio de datos.

- *-default-storage-engine=type*

  Esta opción es un sinónimo para *--default-table-type*. Cambia el valor por defecto de tipo de tablas (normalmente *Innodb* o *MyIsam*).

- *-default-time-zone=type*

  Cambia la zona horaria del servidor. Esta opción cambia la variable global de sistema *time_zone*. Si no se da esta opción, la zona horaria por defecto es la misma que la del sistema.

- *-max_binlog_size*

  Es el tamaño del fichero de registro binario que registra los comandos SQL que modifican objetos de las bases de datos.

- *-ndbcluster*

  Si el binario incluye soporte para el motor de almacenamiento *NDBCluster*, la opción por defecto de desactivar el soporte para MySQL *Cluster* puede ignorarse usando esta opción.

- *-skip-grant-tables*

  Esta opción hace que el servidor no use el sistema de privilegios en absoluto. Da a cualquiera que tenga acceso al servidor acceso ilimitado a todas las bases de datos. Puede hacer que un servidor en ejecución empiece a usar las tablas de privilegios de nuevo ejecutando *mysqladmin flush-privileges* o el comando *mysqladmin reload* desde una consola de sistema, o utilizando el comando *FLUSH PRIVILEGES* desde la consola de *MySQL*.

- *-skip-ndbcluster*

  Deshabilita el motor de almacenamiento *NDBCluster*. Este es el comportamiento por defecto para los binarios compilados con el soporte para el motor de almacenamiento de *NDBCluster*, lo que significa que el sistema reserva memoria y otros recursos para este motor de almacenamiento solo si *--skip-ndbCluster* está habilitado explícitamente por la opción *--ndbCluster*.

- *-standalone*

  Solo para sistemas basados en Windows-*NT*; le dice al servidor MySQL que no se ejecute como servicio.

- *-version, -V*

  Muestra información de versión y termina.

Variables de registro o *log*

- *-general-log*

  Permite activar o desactivar el registro general.

- *-general-log-file, -l [file]*

  Log de conexiones y consultas en este fichero. Si no especifica un nombre de fichero, MySQL usará *host_name*. *log* como nombre de fichero.

- *-log-bin=[file]*

  El fichero de *logs* binario. Registra todas las consultas que cambian datos en este fichero. Se usa para copias de seguridad y replicación. Se recomienda especificar un nombre de fichero; en caso contrario MySQL usará *host_name-bin*.

- *- log-bin-index[=file]*

  El fichero *índice* para *log* binario. Si no especifica un nombre de fichero y si no especifica uno en *--log*-bin, MySQL, usará *host_name-bin.index* como el nombre de fichero.

- *-log-error[=file]*

  Mensajes de error y de arranque en este fichero. Si no especifica un nombre de fichero, MySQL usa *host_ name.err* como nombre de fichero. Si el nombre de fichero no tiene extensión, una extensión *.err* se añade al nombre.

- *-log-queries-not-using-indexes*

  Si usa esta opción con *--log-slow-queries*, las consultas que no usan índices también se *loguean* en el *log* de consultas lentas.

- *-log-slow-queries[=file]*

  Registra todas las consultas que han tardado más de *long_query_time* segundos en ejecutarse.

- *-log-warnings[=level], -W [level]*

  Muestra advertencias tales como *Aborted connection* en el *log* de errores. Se recomienda activar esta opción, por ejemplo, si usamos replicación (nos da más información acerca de lo que está ocurriendo, como mensajes acerca de fallos de red y *reconexiones*). Esta opción está activada por defecto, para desactivarla, debemos usar *--skip-log-warnings*. O ponerla a *level = 0*. Las conexiones abortadas no se registran en el *log* de errores a no ser que el valor *level* sea mayor que 1.

Variables relacionadas con la memoria y la caché

- *-join_buffer_size*

  Es la mínima cantidad de memoria usada para consultas que implican combinaciones (*join*) de tablas sin usar índices.

- *-read_buffer_size*

  Cada hilo (*thread*) que realiza un recorrido secuencial sobre los valores de una tabla crea un *buffer* del tamaño indicado por esta variable.

- *-read_rnd_buffer_size*

  Indica el valor de memoria requerida cada vez que se leen registros de una tabla según el orden de sus índices. Es especialmente útil en consultas de ordenación.

- *-sort_buffer_size*

  Cada vez que se necesita hacer una ordenación se crea un *buffer* del tamaño indicado por esta variable.

- *-sql_buffer_result*

  Con valor uno hace que se creen tablas temporales para almacenar los resultados de consultas. Muy útil cuando para no tener que bloquear tablas durante mucho tiempo en consultas de gran tamaño que pueden tardar tiempo en ser enviadas al cliente.

- *-binlog_cache_size*

  Tamaño de la caché para guardar cambios del registro binario durante una transacción.

- *-query_cache_size*

  La cantidad de memoria usada para cachear resultados de consultas.

- *-thread_cache_size*

  Numero de hilos que pueden ser reutilizados por nuevos clientes

  Variables relacionadas con tablas *MyISAM*.

- *-key_buffer_size*

  Es el tamaño de memoria usado para bloques de índices.

- *-bulk_insert_buffer_size*

  Tamaño de memoria usada para acelerar operaciones de inserción de datos mediante comandos *INSERT* o *LOAD DATA*.

Variables tipo have

Indican la tenencia o no de cierta característica, como por ejemplo *have_crypt* cuyo valor (ON u OFF) indica si se puede usar la función de encriptación *ENCRYPT()*.

Hemos comentado variables generales que afectan al uso de cualquier tipo de motor de almacenamiento. Existen, sin embargo, grupos de variables que afectan en particular a los motores *MyISAM* e *InnoDB*. La mayoría de ellas comienzan con el prefijo *myisam* e *innodb*, respectivamente.

Variables relacionadas con la red

Determinan parámetros relacionados con conexiones, resolución de nombres, etc.

- ✓ *-skip-networking*

  No escucha conexiones TCP/IP en absoluto. Toda interacción con *mysqld* debe hacerse vía *named pipes* o memoria compartida (en Windows) o ficheros *socket* en Unix. Esta opción se recomienda encarecidamente en sistemas donde solo se permitan clientes locales.

- ✓ *-skip-name-resolve*

  Evita que el servidor compruebe el nombre del equipo que se conecta. Es útil si no disponemos de una red con nombres.

- ✓ *-net_buffer_length*

  Es el tamaño del *buffer* asociado a cada cliente para la conexión y para los resultados de sus consultas.

Leer

2.4.4 VARIABLES DE ESTADO DEL SERVIDOR

El servidor mantiene muchas variables de estado que proveen de información sobre su estado. Pueden consultares con el comando *SHOW STATUS LIKE 'patron'*, siendo patrón una cadena dentro de la variable.

A continuación detallamos las más importantes:

Variables generales

- *-Aborted_clients*

  El número de conexiones que han sido abortadas debido a que el cliente murió sin cerrar la conexión apropiadamente.

- *-Aborted_connects*

  El número de intentos de conexión al servidor MySQL que han fallado.

- *-Bytes_received*

  El número de *bytes* recibidos desde todos los clientes.

- *-Bytes_sent*

  El número de *bytes* enviados hacia todos los clientes.

- *-Connections*

  El número de intentos de conexión (con éxito o no) al servidor MySQL.

- *-Last_query_cost*

  El coste total de la última consulta compilada tal como ha sido computada por el optimizador de consultas. Es útil para comparar el coste de diferentes planes de ejecución para la misma consulta. El valor por defecto de 0 significa que no se ha compilado ninguna consulta todavía.

- *-Max_used_connections*

  El número máximo de conexiones que han sido utilizadas simultáneamente desde que el servidor ha sido iniciado.

- *-Open_tables*

  El número de tablas que están actualmente abiertas.

- *-Opened_tables*

  El número de tablas que han sido abiertas. Si *Opened_tables* es grande, probablemente el valor de *table_cache* es demasiado pequeño.

- *-Threads_created*

  El número de subprocesos creados para gestionar conexiones. Si Threads_created es grande, debería incrementar el valor de *thread_cache_size*. La tasa de éxitos de la caché puede ser calculada como *Threads_created*/*Connections*.

- *-Threads_running*

  El número de subprocesos que no están durmiendo.

- *-Uptime*

  El número de segundos que el servidor ha estado funcionando ininterrumpidamente.

Variables tipo Com

Son todas ellas variables de estado que contabilizan el número de veces que cierto comando se ha ejecutado en la sesión actual. Su formato siempre es *Com_xxx* siendo *xxx* el comando *sql* computado, por ejemplo *Com_insert* contabiliza el número de inserciones realizadas. Son todas ellas variables de estado y, por tanto, no pueden modificarse ni incluirse en ficheros de opciones.

Variables tipo Handler

Variables relacionadas con operaciones de lectura y escritura sobre las tablas, su formato es *Handler_xxx* siendo *xxx* la cadena que indica la operación realizada. Por ejemplo *Handler_read_key* indica el número de veces que se ha leído una fila de una tabla basándose en el índice.

Variables tipo InnoDB

Los motores *InndoDB* dada su mayor complejidad disponen de un conjunto específico de variables de estado que facilitan su optimización. La mayoría tienen el formato *Innodb_data_read* indica la cantidad de datos leídos desde que el servidor fue iniciado.

Variables tipo Key

Para contaje del número de operaciones relacionadas con índices. Su formato es *Key_xxx*, donde *xxx* representa la operación correspondiente. Por ejemplo *key_reads* computa el número de lecturas de disco de bloques de índices.

Variables tipo Qcache

Variables relacionadas con número de operaciones sobre la *query cache*. Su formato es *Qcache_xxx* siendo *xxx* la operación correspondiente. Por ejemplo, *Qcache_inserts* muestra el número de consultas añadidas a la caché.

Variables tipo ssl

Indican aspectos relacionados con la criptografía de clave asimétrica o SSL. Su formato es *ssl_xxx*, siendo *xxx* el nombre de la característica SSL que queremos mostrar. Por ejemplo, *ssl_cipher_list* contiene los métodos de cifrado que soporta nuestro servidor.

Variables relacionadas con Threads

Son variables relacionadas con los hilos o conexiones creadas en el servidor, por ejemplo, *threads_created* indica el número de conexiones existentes en el servidor.

2.4.5 COMANDOS PARA GESTIÓN DE VARIABLES

A continuación indicamos los comandos útiles para la gestión de variables de servidor además de para la obtención de información relevante de forma rápida y eficiente.

Comandos de consulta: *SHOW*

Para obtener los valores de las variables podemos usar distintos comandos SHOW que describimos a continuación:

- *SHOW VARIABLES*

```
mysql> SHOW VARIABLES;
```

También podemos filtrar la salida con *LIKE*.

Leer

Cuando usamos el comando *SET* para cambiar las variables de sistema los sufijos no pueden usarse, pero el valor puede tomar la forma de una expresión. En el siguiente ejemplo asignamos 10 MB al valor de la variable *sort_buffer_size*.

EJEMPLO 2.10

```
mysql> SET sort_buffer_size = 10 * 1024 * 1024;
```

Para especificar explícitamente si queremos cambiar la variable global o de sesión para este mismo caso usamos la opción *GLOBAL* o *SESSION*:

EJEMPLO 2.11

```
mysql> SET GLOBAL sort_buffer_size = 10 * 1024 * 1024;
mysql> SET SESSION sort_buffer_size = 10 * 1024 * 1024;
```

Sin dicha opción, el comando actualiza la variable de sesión por defecto.

EJEMPLO 2.12

```
mysql> SHOW VARIABLES LIKE '%cadena_busqueda%';
```

✓ *SHOW STATUS*

Para variables de estado podemos ver sus valores utilizando la sentencia *SHOW STATUS*.

```
mysql> SHOW STATUS;
+-----------------------------------+------------+
| Variable_name                     | Value      |
+-----------------------------------+------------+
| Aborted_clients                   | 0          |
| Aborted_connects                  | 0          |
| Bytes_received                    | 155372598  |
| Bytes_sent                        | 1176560426 |
...
| Connections                       | 30023      |
| Created_tmp_disk_tables           | 0          |
| Created_tmp_files                 | 3          |
| Created_tmp_tables                | 2          |
...
| Threads_created                   | 217        |
| Threads_running                   | 88         |
| Uptime                            | 1389872    |
+-----------------------------------+------------+
```

Diario de clase

Muchas variables de estado son inicializadas a 0 por la sentencia *FLUSH STATUS*.

También podemos filtrar la salida de *SHOW STATUS* con *LIKE*.

```
mysql> SHOW STATUS LIKE '%cadena_busqueda%';
```

2.5 ESTRUCTURA DEL DICCIONARIO DE DATOS

El diccionario de datos es un componente esencial en cualquier SGBD ya que contiene información (metadatos) sobre los objetos de bases de datos alojadas en nuestro servidor. *Metadatos* son datos acerca de los datos, tales como el nombre de las bases de datos o tabla, el tipo de datos de una columna o permisos de acceso. Otros términos que a veces se usan para esta información son el diccionario de datos o el catálogo del sistema.

En la mayoría de SGBD esta información se almacena en una base de datos. Para el caso de MySQL, dicha base de datos que se crea por defecto en la instalación, que se llama *information_schema*. Por ejemplo, podemos consultar la información sobre tablas de todas las bases de datos con el siguiente *select*:

EJEMPLO 2.13

```
mysql> SELECT table_name, table_type, engine
    -> FROM information_schema.tables
    -> WHERE table_schema = 'db5'
    -> ORDER BY table_name DESC;
+------------+------------+--------+
| table_name | table_type | engine |
+------------+------------+--------+
| v56        | VIEW       | NULL   |
| v3         | VIEW       | NULL   |
| v2         | VIEW       | NULL   |
| v          | VIEW       | NULL   |
| tables     | BASE TABLE | MyISAM |
|.....|
| t2         | BASE TABLE | MyISAM |
| t          | BASE TABLE | MyISAM |
| pk         | BASE TABLE | InnoDB |
|.....|
+------------+------------+--------+
17 rows in set (0.01 sec)
```

El comando pide una lista de todas las tablas en la base de datos *db5*, en orden alfabético inverso, mostrando tres informaciones: el nombre de la tabla, su tipo y su motor.

INFORMATION_SCHEMA es la base de datos de información que almacena información acerca de todas las otras bases de datos que mantiene el servidor MySQL. Dentro de *INFORMATION_SCHEMA* hay varias tablas de solo lectura. En realidad son vistas, no tablas, así que no podrás ver ningún fichero asociado con ellas.

Cada usuario MySQL tiene derecho a acceder a estas tablas, pero solo a los registros que se corresponden a los objetos a los que tiene permiso de acceso.

La única forma de acceder al contenido de sus tablas es con *SELECT*. No puede insertar, actualizar o borrar su contenido.

No hay diferencia entre el requerimiento de permisos para *SHOW* y para *SELECT*. En cada caso, debe tener algún permiso de un objeto para consultar información acerca del mismo.

2.5.1 LAS TABLAS DE *INFORMATION_SCHEMA*

A continuación, se describen algunas de las tablas más relevantes del diccionario de datos. Para más detalles se recomienda consultar la documentación oficial.

- *SCHEMADATA*

  Proporciona información acerca de las bases de datos o esquemas en nuestro servidor.

Importante

Tabla 2.3 Estructura tabla schemadata

Standard Name	SHOW name
CATALOG_NAME	
SCHEMA_NAME	
DEFAULT_CHARACTER_SET_NAME	
DEFAULT_COLLATION_NAME	
SQL_PATH	

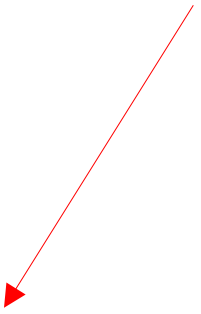
TABLES

Proporciona información acerca de las tablas en las bases de datos.

Tabla 2.4 Estructura tabla tables

Standard Name	SHOW name
TABLE_CATALOG	
TABLE_SCHEMA	Table_...
TABLE_NAME	Table_...
TABLE_TYPE	
ENGINE	Engine
VERSION	Version
ROW_FORMAT	Row_format
TABLE_ROWS	Rows
AVG_ROW_LENGTH	Avg_row_length
DATA_LENGTH	Data_length
MAX_DATA_LENGTH	Max_data_length
INDEX_LENGTH	Index_length
DATA_FREE	Data_free
AUTO_INCREMENT	Auto_increment
CREATE_TIME	Create_time
UPDATE_TIME	Update_time
CHECK_TIME	Check_time
TABLE_COLLATION	Collation
CHECKSUM	Checksum
CREATE_OPTIONS	Create_options
TABLE_COMMENT	Comment

Importante



EJEMPLO 2.14

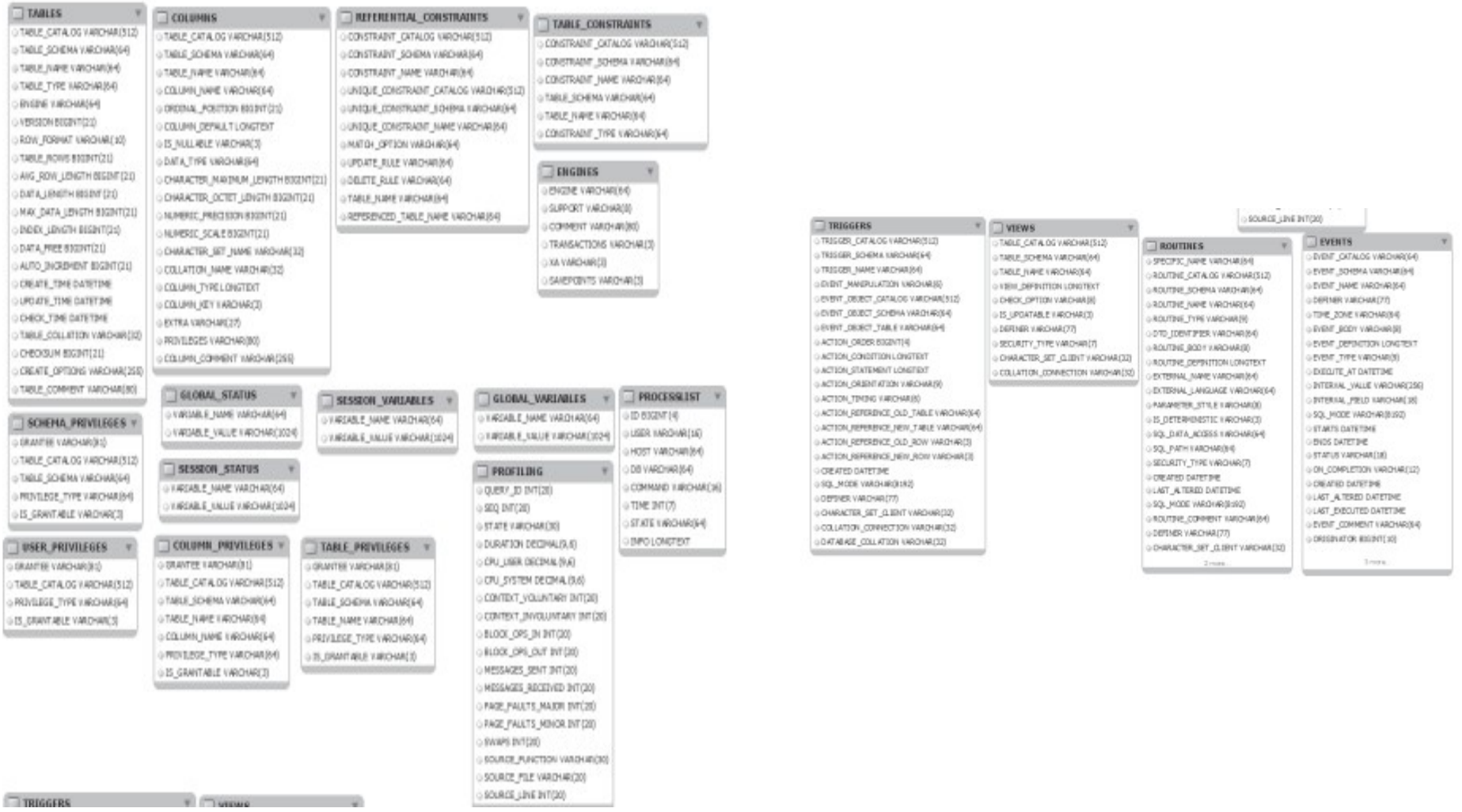
```
CREATE TABLE t1(s1 INT,s2 INT,s3 INT,PRIMARY KEY(s3))ENGINE=InnoDB;

CREATE TABLE t3(s1 INT,s2 INT,s3 INT,KEY(s1),CONSTRAINT CO FOREIGN KEY (s2)
REFERENCES t1(s3)) ENGINE=InnoDB;
```

EJEMPLO 2.15

```
CREATE VIEW v AS
SELECT s2,s1 FROM t
WHERE s1 > 5
ORDER BY s1
WITH CHECK OPTION;
```

Estas son las tablas principales de la base *information_schema obtenidas* usando *Workbench*:



2.6 FICHEROS LOG

Los ficheros de registro (*log*) de MySQL:

MySQL tiene varios archivos de registro diferentes que pueden ayudarnos a encontrar lo que está ocurriendo e servidor:

Importante

Tabla 2.5 Archivos de registro en MySQL

Archivo de registro	Tipo de información registrado en el archivo
El registro de error	Registra problemas encontrados iniciando, ejecutando o parando mysqld.
El registro de consultas	Registra las conexiones de clientes establecidas y las sentencias ejecutadas.
El registro binario	Registra todas las sentencias que cambian datos. También utilizado para replicación.
El registro de lentitud	Registra todas las sentencias que tardarón más de *long_query_time* segundos en ejecutarse,o no utilizaron índices.

Por defecto, todos los registros son creados en el directorio de datos de *mysqld*. Se puede forzar a *mysqld* a que cier y reabra los archivos de registro (o en algunos casos, cambiar a un nuevo registro) mediante el volcado de registr Este volcado de registros ocurre cuando ejecuta la sentencia *FLUSH LOGS* o el comando *mysqladmin flush-logs*.

2.6.1 EL REGISTRO DE ERRORES (*ERROR LOG*)

El archivo de registro de errores contiene información que indica cuando se ha iniciado y parado *mysqld* y también si ha ocurrido algún error crítico mientras el servidor se estaba ejecutando.

En MySQL podemos especificar dónde queremos almacenar el registro de errores con la opción *--log-error*[*=file_name*]. Si no se proporciona ningún valor para *file_name*, *mysqld* utiliza el nombre *host_name.err* y escribe el archivo en el directorio de datos. Si ejecuta *FLUSH LOGS*, el registro de errores es renombrado con el sufijo *-old* y *mysqld* crea un nuevo archivo de registro.

Si no especificas *--log-error*, o (en Windows) no utiliza la opción *--Console*, los errores se escriben en *stderr*, la salida estándar de errores.

En Windows, la salida de errores es siempre escrita al archivo *.err* si no se especifica la opción *--console*.

2.6.2 EL REGISTRO GENERAL DE CONSULTAS

Si queremos registrar los sucesos en nuestro servidor, debemos iniciarlo con la opcion *–general_log* con valor 1. Si no se da un nombre de fichero con la opción *--general_log_file*, por defecto se usa *hostname.log* siendo hostname el nombre de nuestro equipo. Esto registra todas las conexiones y sentencias a un archivo. Este registro puede ser muy útil cuando sospechemos que hay un error en un cliente y queramos saber exactamente qué hemos enviado al servidor.

El servidor *mysqld* escribe las sentencias al registro de consultas en el orden en que las recibe. Este orden puede ser diferente del de ejecución. Esto es aquí diferente que en el registro de actualizaciones o el registro binario, que son escritos tras la ejecución de la sentencia, pero antes de que se libere cualquier bloqueo (El registro de consultas también contiene todas las sentencias, mientras que el registro binario no contiene sentencias que solo seleccionen datos).

Los reinicios del servidor y volcado de registros no provocan que se genere un nuevo archivo de registro de consultas general (aunque el volcado lo cierra y lo reabre).

2.6.3 EL REGISTRO BINARIO (*BINARY LOG*)

El registro binario contiene toda la información que está disponible en el registro de actualizaciones, en un formato más eficiente y de una manera que es segura para las transacciones. Registra todas las sentencias que han actualizado datos o podrían haberlo hecho (por ejemplo, un *DELETE* que no encontró filas concordantes). Las sentenciaas se almacenan en la forma de "eventos" que describen las modificaciones.

El registro binario también contiene información sobre cuánto ha tardado cada sentencia que actualizó la base de datos. No contiene sentencias que no hayan modificado datos. Si queremos registrar todas las sentencias (por ejemplo, para identificar una sentencia problemática) deberíamos utilizar el registro general de consultas.

2.6.4 EL REGISTRO DE CONSULTAS LENTAS (*SLOW QUERY LOG*)

Cuando se inicia con la opción *--log-slow-queries[=file_name]*, *mysqld* escribe un archivo de registro que contiene todos las sentencias SQL que llevaron más de *long_query_time* segundos para ejecutarse completamente. Para ello, el registro de consultas lentas debe activarse con la opción *--slow-query-log* al valor ON ó 1. El tiempo para adquirir los bloqueos de tabla iniciales no se cuenta como tiempo de ejecución.

Si no se da ningún valor a *file_name*, el nombre por defecto es el nombre de la máquina *host* con el sufijo *-slow.log*. Si se da un nombre de archivo, pero no como ruta absoluta, el archivo se escribe en el directorio de datos.

Una sentencia se registra en el registro de consultas lentas después de que haya sido ejecutada y todos los bloqueos liberados. El orden de registro puede diferir del de ejecución.

El registro de consultas lentas se puede utilizar para encontrar consultas que tomen excesivo tiempo y sean por tanto candidatos a optimización. De cualquier modo, examinar un registro de consultas lentas puede convertirse en una tarea difícil. Para hacerlo más simple, puede procesar el registro de consultas lentas utilizando el comando *mysqldumpslow* que le ofrecerá un resumen de las sentencias que aparecen en el registro.

2.6.5 MANTENIMIENTO DE FICHEROS DE REGISTRO (*LOG*)

Se deben limpiar estos archivos regularmente para asegurarse de que no ocupan demasiado espacio.

Cuando se utiliza MySQL con el registro activado, deberíamos hacer copias de seguridad de los registros viejos de vez en cuando y eliminarlos y decirle a MySQL que comience a registrar en archivos nuevos.

Podemos forzar al servidor para que comience a utilizar archivos de registro nuevos usando *mysqladmin flush-logs* o con la sentencia *SQL FLUSH LOGS*.

Existen soluciones basadas en *Shell script* (para sistemas Linux) o lenguajes como *Perl* que combinados con el programador de tareas (*cron* para Linux) permiten automatizar este tipo de tareas, algo muy conveniente sobre todo para los registros, dada la gran cantidad de información que generan y lo rápido que pueden llegar a crecer.

No obstante no debe descartarse almacenar los *logs* de varios meses e incluso años dado que pueden ser una gran fuente de información diversa acerca del funcionamiento de nuestro servidor, el tipo de consultas, el volumen, etc., lo que es de gran ayuda para su optimización.

2.6.6 REGISTRO EN *INNODB*

Lo explicado en las anteriores secciones se aplica a todos los motores de almacenamiento sin embargo el tipo *InnoDB* incorpora además un tipo especial de registro donde almacena las operaciones sobre tablas *InnoDB*.

Para configurarlo se usan principalmente tres variables:

-*innodb_log_files_in_group*: número de ficheros a rotar. Útil para el mantenimiento. Por defecto su valor es 2.

-*innodb_log_file_size*: tamaño de los ficheros.

-*innodb_log_buffer_size*: *buffer* usado para el registro antes del volcado a disco de los mismos.

Leer

2.7 CASO BASE

Se pretende usar un servidor dedicado para la base de datos *mediaserver* que proporciona servicios de *streaming* de audio y vídeo para clientes remotos los cuales pueden ser registrados o no.

En la web encontrarás el código para recrear la base de datos además de la descripción de la aplicación y sus requisitos en cuanto a datos. Este es un ejemplo real y cercano al alumno. No obstante es perfectamente aplicable a otras aplicaciones conocidas como Youtube o Twiter, siempre desde luego usando versiones simplificadas.

Diseño y cálculos iniciales

En esta sección debemos determinar lo siguiente:

- -Hardware necesario (detalles de procesador, memoria RAM y almacenamiento).
- -Diseño particionado físico.
- -Sistemas operativos que instalaremos. Ubicación y espacio necesario.
- -Versión de MySQL a instalar.
- -Estimación inicial del tamaño de la base de datos y el ritmo previsto de crecimiento.
- -Estima la frecuencia de consultas, inserciones y modificaciones.
- -Según las operaciones de datos que preveas, calcula el espacio necesario para los ficheros *log* de cada tipo (general, errores, binario, consultas lentas y de *innodb*, en caso de haberlo).

Configuración

- Configura las variables más importantes comentadas en el capítulo acorde con la previsión realizada.

RESUMEN DEL CAPÍTULO

En este capítulo se ha tratado de introducir al alumno en los conceptos principales de la administración de un SGBD, desde su instalación y configuración básica inicial hasta su mantenimiento y monitorización con el uso de variables de configuración y ficheros de registro pasando por la descripción del diccionario de datos propio de MySQL.

En los siguientes capítulos profundizaremos en otras tareas de administración más complejas, las cuales hacen uso de lo explicado aquí, especialmente en la parte de variables del servidor.

Leer

EJERCICIOS PROPUESTOS

Tema 1. Ejercicios propuestos.

- **1.** Vuelca en un fichero la ayuda del comando *mysqld*.

- **2.** Dentro del fichero *my.ini*, ¿para qué sirven las secciones *mysqld* y *mysql*?

- **3.** ¿De qué manera piensas que podemos hacer que el servidor funcione sin que nadie pueda acceder a él de manera remota? ¿Y sin usar resolución de nombres de dominio?

- **4.** Configura los registros de errores binarios y com- prueba qué ocurre en el registro de errores apagando e iniciando el servidor de manera incorrecta, por ejemplo, usando *taskkill* o apagándolo desde el administrador de tareas de Windows.

- **5.** Crea una tabla *t1* en la base test con un campo numérico y de tipo *InnoDB*. Modifícala para que sea *MyISAM*. ¿Qué cambios observas en el sistema de ficheros en el directorio de datos de MySQL?

- **6.** Averigua el tamaño máximo de los archivos de registro binario y en qué variable se configura. Configura dicha variable para un tamaño máximo de 5 KB y comprueba su funcionamiento después de algunas inserciones en la tabla *test.t1*.

- **7.** ¿Qué dos aspectos debemos tener en cuenta en el servidor para permitir el acceso remoto? Haz que uno o más compañeros se conecten a tu servidor. Indica el comando para visualizarlas usando el pro- grama *mysqladmin*.

- **8.** Usa tu máquina virtual con Linux para comprobar el contenido del directorio *bin* de MySQL. Indica al menos tres programas adicionales que incluya, con respecto a la instalación de Windows y coméntalos brevemente usando la ayuda de *man*.

- **9.** Averigua el significado del concepto de replicación en el contexto de bases de datos. ¿Qué relación crees que tiene la replicación con los registros binarios?

- **10.** Has perdido todos tus datos aunque posees una copia de seguridad completa (un fichero *sql* genera- do con el programa *mysqldump*) de hace 48 horas. Dispones también de un registro binario de esas 48 horas. ¿Cómo retornas el servidor a su estado exacto en el momento de la pérdida de datos?

Entregar Moodle

TEST DE CONOCIMIENTOS

Test de conocimientos UT1 - Tema 1- Instalación y configuración de un sistema gestor de base de datos.
Marca la casilla que creas correcta con una X.

1. ¿Qué se entiende por SQL?
a) Un lenguaje para la gestión de bases de datos
b) Un estándar de SGBD
c) Un gestor de bases de datos
d) Un lenguaje de programación

A	B	C	D

2. El diccionario de datos en un SGBD es:
a) Una base de datos donde se guardan los datos de las bases de datos del sistema
b) La información sobre los datos almacenados en una base de datos
c) Lo que hay en la base de datos information_schema
d) Lo que hay en la base de datos mysql

A	B	C	D

3. ¿En qué directorio se almacenan los programas de MySQL?
a) data
b) scripts
c) bin
d) tables

A	B	C	D

4. ¿Qué hace la opción console cuando arrancamos el servidor?
a) Mostrar la versión
b) Volcar por pantalla la salida del comando del servidor
c) Reiniciar el servidor
d) Activar los permisos en las bases de datos

A	B	C	D

5. Si tenemos varias tarjetas de red en nuestro servidor, ¿cómo asociamos cada una de ellas?
a) Poniendo en el fichero my.ini ip=ip-interfaz
b) Usando la opción bind
c) Usando la opción bind-address
d) Usando la opción ipconfig

A	B	C	D

6. La palabra engine en MySQL designa:
a) Un tipo de tabla
b) Un motor de almacenamiento
c) Un tipo de estructura en la que almacenar datos
d) Todas las anteriores

A	B	C	D

7. ¿Cuál de los siguientes no es un campo de la tabla columns del diccionario de datos?
a) Type
b) Numeric_scale
c) Extra
d) Table_schema

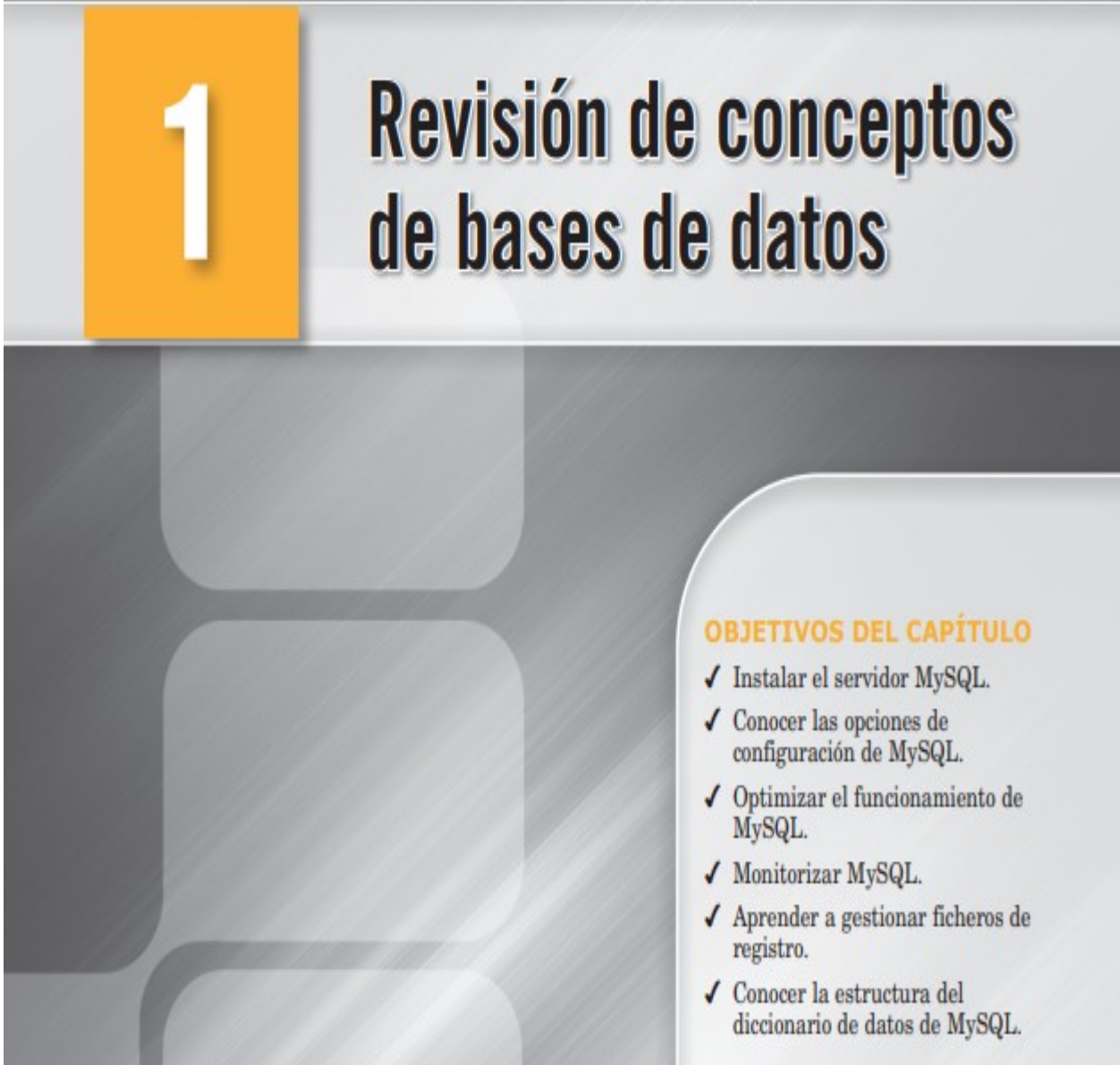
A	B	C	D

8. El comando purge:
a) Es como flush pero más rápido
b) Permite reparar tablas
c) Elimina datos espurios
d) Vacía el registro de actualización de consultas

A	B	C	D

Entregar Moodle

UT2 - Unidad 2
Acceso a la Información



1.1 INTRODUCCIÓN. DEFINICIÓN DE BASES DE DATOS Y SGBD

Conviene, antes de comenzar con el tema principal del libro, recordar los conceptos más relevantes relacionados con sistemas gestores de bases de datos, así como las herramientas relacionadas.

En primer lugar, y aunque probablemente ya se ha visto en otros módulos, es importante diferenciar entre el concepto de base de datos y el de sistema gestor, ya que es habitual confundirlos y sin embargo son cosas muy distintas.

Definición 1:

Una base de datos es un conjunto de datos relacionados y organizados con cierta estructura. Según dicha organización distinguimos entre diferentes modelos de bases de datos como el relacional, jerárquico o en red.

El modelo de bases de datos más extendido es el **relacional** y es el que trabajaremos en este libro.

Para su manipulación y gestión surgieron los sistemas gestores de bases de datos (**SGBD** en lo sucesivo).

Definición 2:

El sistema de gestión de la base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener bases de datos, proporcionando acceso controlado a las mismas. Es una herramienta que sirve de interfaz entre el usuario y las bases de datos.

Es decir, por un lado tenemos los datos organizados según ciertos criterios y, por otro, un software que nos permite o facilita su gestión con distintas herramientas y funcionalidades que describimos a continuación.

1.2 ARQUITECTURA DE SISTEMAS DE BASES DE DATOS

Hay tres características importantes inherentes a los sistemas de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos. En 1975, el comité ANSI-SPARC (*American National Standard Institute - Standards Planning and Requirements Committee*) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. En esta arquitectura, el esquema de una base de datos se define en tres niveles de abstracción distintos como se aprecia en la siguiente imagen:

Importante

Figura 1.1. *Arquitectura de Sistemas de Bases de Datos*

En el nivel interno se describe la estructura física de la base de datos mediante un esquema interno. Este esquema se especifica mediante un modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso.

En el nivel conceptual se describe la estructura de toda la base de datos para una comunidad de usuarios (todos los de una empresa u organización), mediante un esquema conceptual. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar el esquema.

En el nivel externo se describen varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinado y oculta a ese grupo el resto de la base de datos. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo, como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo. Por lo tanto, el SGBD debe transformar cualquier petición expresada en términos de un esquema externo a una petición expresada en términos del esquema conceptual, y luego, a una petición en el esquema interno, que se procesará sobre la base de datos almacenada. Si la petición es de una obtención (consulta) de datos, será preciso modificar el formato de la información extraída de la base de datos almacenada para que coincida con la vista externa del usuario. En definitiva, tenemos la vista del usuario, la del sistema gestor y la vista física o de almacenamiento.

Leer

La arquitectura de tres niveles es útil para explicar el concepto de **independencia de datos,** que podemos definir como *la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior*. Se pueden definir dos tipos de independencia de datos:

La **independencia lógica** es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.

La **independencia física** es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere solo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica.

En los SGBD que tienen la arquitectura de varios niveles es necesario ampliar el catálogo o diccionario, de modo que incluya información sobre cómo establecer la correspondencia entre las peticiones de los usuarios y los datos entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el catálogo. La independencia de datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios, solo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

Por lo tanto, la arquitectura de tres niveles puede facilitar la obtención de la verdadera independencia de datos, tanto física como lógica. Sin embargo, los dos niveles de correspondencia implican un gasto extra durante la ejecución de una consulta o de un programa, lo cual reduce la eficiencia del SGBD. Es por esto que muy pocos SGBD han implementado esta arquitectura completa.

1.3 FUNCIONES DEL SISTEMA GESTOR DE BASE DE DATOS (SGBD)

La función principal de un **SGBD** es permitir a los usuarios realizar las cuatro operaciones fundamentales posibles, tanto sobre las estructuras de datos como sobre los datos que albergan, es decir operaciones de inserción o creación, consulta, actualización y borrado, de una manera eficiente y coherente.

Para tal fin, la mayoría de SGBD incorporan las siguientes características y funciones:

Un catálogo

Donde se almacenen las descripciones de los datos y sea accesible por los usuarios. Este **catálogo** es lo que se denomina *diccionario de datos* y contiene información que describe los datos de la base de datos (metadatos). Normalmente, un diccionario de datos describe entre otras cosas:

- Nombre, tipo y tamaño de los datos.
- Relaciones entre los datos.
- Restricciones de integridad sobre los datos.
- Usuarios autorizados a acceder a los objetos de base de datos.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Garantizar la integridad

Disponer de un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen, o que no se realice ninguna. Una transacción es un conjunto de acciones que cambian el contenido de la base de datos. Una **transacción** en el sistema informático de la empresa inmobiliaria sería dar de alta a un empleado o eliminar un inmueble. Una transacción un poco más complicada sería eliminar un empleado y reasignar sus inmuebles a otro empleado. En este caso hay que realizar varios cambios sobre la base de datos. Si la transacción falla durante su realización, por ejemplo porque falla el hardware, la base de datos quedará en un estado inconsistente. Algunos de los cambios se habrán hecho y otros no, por lo tanto, los cambios realizados deberán ser deshechos para devolver la base de datos a un estado consistente.

Permitir actualizaciones

Asegurar que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente. Uno de los principales objetivos de los SGBD es el permitir que varios usuarios tengan acceso concurrente a los datos que comparten. El acceso concurrente es relativamente fácil de gestionar si todos los usuarios se dedican a leer datos, ya que no pueden interferir unos con otros. Sin embargo, cuando dos o más usuarios están accediendo a la base de datos y al menos uno de ellos está actualizando datos, pueden interferir de modo que se produzcan inconsistencias en la base de datos. El SGBD se debe encargar de que estas interferencias no se produzcan en el acceso simultáneo.

Recuperación de datos

Permitir recuperar las bases de datos en caso de que ocurra algún suceso que la dañe. Como se ha comentado antes, cuando el sistema falla en medio de una transacción, la base de datos se debe devolver a un estado consistente. Esta falta puede ser a causa de un fallo en algún dispositivo hardware o un error del software, que hagan que el SGBD aborte, o puede ser a causa de que el usuario detecte un error durante la transacción y la aborte antes de que finalice. En todos estos casos, el SGBD debe proporcionar un mecanismo capaz de recuperar la base de datos llevándola a un estado consistente.

Integración

Ser capaz de integrarse con algún software de comunicación. Muchos usuarios acceden a la base de datos desde terminales. En ocasiones estos terminales se encuentran conectados directamente a la máquina sobre la que funciona el SGBD. En otras ocasiones los terminales están en lugares remotos, por lo que la comunicación con la máquina que alberga al SGBD se debe hacer a través de una red. En cualquiera de los dos casos, el SGBD recibe peticiones en forma de mensajes y responde de modo similar. Todas estas transmisiones de mensajes las maneja el gestor de comunicaciones de datos. Aunque este gestor no forma parte del SGBD, es necesario que el SGBD se pueda integrar con él para que el sistema sea comercialmente viable.

Cumplir restricciones

Proporcionar los medios necesarios para garantizar que tanto los datos de la base de datos, como los cambios que se realizan sobre estos datos, sigan ciertas reglas. La integridad de la base de datos requiere la validez y consistencia de los datos almacenados. Se puede considerar como otro modo de proteger la base de datos, pero además de tener que ver con la seguridad, tiene otras implicaciones. La integridad se ocupa de la calidad de los datos. Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar. Por ejemplo, se puede establecer la restricción de que cada empleado no puede tener asignados más de diez inmuebles. En este caso sería deseable que el SGBD controlara que no se sobrepase este límite cada vez que se asigne un inmueble a un empleado.

Herramientas de administración

Proporcionar herramientas que permitan administrar la base de datos de modo efectivo, lo que implica un diseño óptimo de las mismas, garantizar la disponibilidad e integridad de los datos, controlar el acceso al servidor y a los datos, monitorizar el funcionamiento del servidor y optimizar su funcionamiento. Muchas de ellas van integradas en el sistema gestor, otras son creadas por terceros o por el propio administrador según sus requerimientos.

1.4 COMPONENTES

Son los elementos que deben proporcionar los servicios comentados en la sección anterior. No se puede generalizar ya que varían mucho según la tecnología. Sin embargo, es muy útil conocer sus componentes y cómo se relacionan cuando se trata de comprender lo que es un sistema de bases de datos.

El **SGBD** es la aplicación que interacciona con los usuarios de los programas de aplicación y la base de datos. En general, un SGBD suele incluir los siguientes componentes:

- **Lenguaje de definición de datos** (**DDL**: *Data Definition Language*)

  Sencillo lenguaje artificial para definir y describir los objetos de la base de datos, su estructura, relaciones y restricciones.

- **Lenguaje de control de datos** (**DCL**: *Data Control Language*)

  Encargado del control y seguridad de los datos (privilegios y modos de acceso, etc.). Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos. Todo esto se almacenará en la base de datos.

- **Lenguaje de manipulación de datos** (**DML**: *Data Manipulation Language*)

  Para la inserción, actualización, eliminación y consulta de datos. Para tal fin el lenguaje por excelencia es el conocido SQL (*Structured Query Language*). Incluye instrucciones para los tres tipos de lenguajes comentados y por su sencillez y potencia se ha convertido en el lenguaje estándar de los **SGBD relacionales**.

- **Diccionario de datos**

  Esquemas que describen el contenido del **SGBD** incluyendo los distintos objetos con sus propiedades.

- **Objetos**: **Tablas base y vistas (tablas derivadas)**
  - Consultas.
  - Dominios y tipos definidos de datos.
  - Restricciones de tabla y dominio y aserciones.
  - Funciones y procedimientos almacenados.
  - Disparadores o *triggers*.

- **Distintas herramientas para**:
  - Seguridad: de modo que los usuarios no autorizados no puedan acceder a la base de datos.
  - Integridad: que mantiene la integridad y la consistencia de los datos.
  - El control de concurrencia: que permite el acceso compartido a la base de datos.
  - El control de recuperación: que restablece la base de datos después de que se produzca un fallo del hardware o del software.
  - Gestión del diccionario de datos (o catálogo): accesible por el usuario que contiene la descripción de los datos de la base de datos.
  - Programación de aplicaciones.
  - Importación/exportación de datos (migraciones).
  - Distribución de datos.
  - Replicación (arquitectura maestro-esclavo).
  - Sincronización (de equipos replicados).
- **Optimizador de consultas**

  Para determinar la estrategia óptima para la ejecución de las consultas.
- **Gestion de transacciones**

  Este módulo realiza el procesamiento de las transacciones.
- **Planificador** (*scheduler*)

  Para programar y automatizar la realización de ciertas operaciones y procesos.
- **Copias de seguridad**

  Para garantizar que la base de datos se puede devolver a un estado consistente en caso de que se produzca algún fallo.

Todos los SGBD no presentan la misma funcionalidad, depende de cada producto. En general, los grandes SGBD multiusuario ofrecen todas las funciones que se acaban de citar y muchas más. Los sistemas modernos son conjuntos de programas extremadamente complejos y sofisticados, con millones de líneas de código y con una documentación consistente en varios volúmenes. Lo que se pretende es proporcionar un sistema que permita gestionar cualquier tipo de requisitos y que tenga un 100% de fiabilidad ante cualquier fallo hardware o software. Los SGBD están en continua evolución, tratando de satisfacer los requerimientos de todo tipo de usuarios. Por ejemplo, muchas aplicaciones de hoy en día necesitan almacenar imágenes, vídeo, sonido, etc. Para satisfacer a este mercado, los SGBD deben cambiar. Conforme vaya pasando el tiempo irán surgiendo nuevos requisitos que serán incorporados paulatinamente.

1.5 USUARIOS DE LOS SGBD

Generalmente distinguimos cuatro grupos de usuarios de sistemas gestores de bases de datos: los usuarios administradores, los diseñadores de la base de datos, los programadores y los usuarios de aplicaciones que interactúan con las bases de datos.

Administrador de la base de datos

Se encarga del diseño físico de la base de datos y de su implementación, realiza el control de la seguridad y de la concurrencia, mantiene el sistema para que siempre se encuentre operativo y se encarga de que los usuarios y las aplicaciones obtengan buenas prestaciones. El administrador debe conocer muy bien el SGBD que se esté utilizando, así como el equipo informático sobre el que esté funcionando.

Diseñadores de la base de datos

Realizan el diseño lógico de la base de datos, debiendo identificar los datos, las relaciones entre datos y las restricciones sobre los datos y sus relaciones. El diseñador de la base de datos debe tener un profundo conocimiento de los datos de la empresa y también debe conocer sus reglas de negocio. Las reglas de negocio describen las características principales de los datos tal y como los ve la empresa. Para obtener un buen resultado, el diseñador de la base de datos debe implicar en el desarrollo del modelo de datos a todos los usuarios de la base de datos, tan pronto como sea posible. El diseño lógico de la base de datos es independiente del SGBD concreto que se vaya a utilizar, es independiente de los programas de aplicación, de los lenguajes de programación y de cualquier otra consideración física.

Programadores de aplicaciones

Se encargan de implementar los programas de aplicación que servirán a los usuarios finales. Estos programas de aplicación son los que permiten consultar datos, insertarlos, actualizarlos y eliminarlos. Estos programas se escriben mediante lenguajes de tercera generación o de cuarta generación.

Usuarios finales

Clientes de la base de datos que hacen uso de ella sin conocer en absoluto su funcionamiento y organización. Son personas con pocos o nulos conocimientos de informática.

1.6 TIPOS DE SGBD

Existen numerosos SGBD en el mercado que podemos clasificar según lo siguiente:

- **Modelo lógico en el que se basan**
  - Modelo Jerárquico.
  - Modelo de Red.
  - Modelo Relacional.
  - Modelo Orientado a Objetos.
- **Número de usuarios**
  - Monousuario.
  - Multiusuario.
- **Número de sitios**
  - Centralizados.
  - Distribuidos: homogéneos y heterogéneos.
- **Ámbito de aplicación**
  - Propósito General.
  - Propósito Específico.

Basados en el modelo **relacional**, los datos se describen como relaciones que se suelen representar como tablas bidimensionales consistentes en filas y columnas. Cada fila (*tupla*, en terminología relacional) representa una ocurrencia. Las columnas (atributos) representan propiedades de las filas. Cada *tupla* o fila se identifica por una clave primaria o identificador.

Esta organización de la información permite recuperar de forma flexible los datos de una o varias tablas, así como combinar registros de diferentes tablas para formar otras nuevas. No todas las definiciones posibles de tablas son válidas según el modelo relacional. En él, deben emplearse diseños normalizados que garantizan que no se producirán anomalías en la actualización de la base de datos.

Los SGBD relacionales se han impuesto hasta llegar a dominar casi totalmente el mercado actual. Ello se ha debido principalmente a su flexibilidad y sencillez de manejo. Igualmente conviene destacar la amplia implantación del lenguaje SQL, que se ha convertido en un estándar para el manejo de datos en el modelo relacional, lo que ha supuesto una ventaja adicional para su desarrollo.

1.7 SISTEMAS GESTORES DE BASE DE DATOS COMERCIALES Y LIBRES

Con el advenimiento de Internet, el software libre se ha consolidado como alternativa, técnicamente viable y económicamente sostenible al software comercial, contrariamente a lo que a menudo se piensa, convirtiéndose el software libre como otra alternativa para ofrecer los mismos servicios a un coste cada vez más reducido.

Estas alternativas se encuentran tanto para herramientas de ofimática como Openoffice o Microsoft Office. También disponemos de herramientas mucho más avanzadas a un nivel de propósito general como MySQL, SQL Server y si hablamos de software con más potencia y funcionalidad vale la pena señalar a Postgresql u Oracle, entre otros.

No conviene decantarse por uno en concreto, debemos usar en primer lugar lo que mejor nos funciona dadas nuestras restricciones particulares. Así, si solamente quiero una aplicación de agenda puede ser perfectamente válido un Access o incluso un Openoffice calc (y en según qué casos incluso un *Bloc de notas*).

Si por el contrario, mi base de datos requiere cierta cantidad de accesos de usuarios diversos, control de integridad y otras funcionalidades, debería plantearme algo como MySQL.

Finalmente, si hablamos de una gran corporación que requiere herramientas avanzadas para grandes bases de datos quizás debamos plantearnos sistemas más potentes como Oracle o Postgresql.

En cuanto a si debe ser libre o no la decisión dependerá de si disponemos de personal cualificado en cuyo caso un sistema libre es más barato y potente. En caso contrario el sistema de pago es la elección más adecuada. No obstante todas las tecnologías disponen de servicios de soporte de gran calidad.

Por otro lado, los sistemas libres cada vez proveen una mejor y más eficiente documentación tanto a nivel oficial como a través de múltiples foros y *blogs*, lo que hace que cada vez lo use más gente y mejore continuamente. Sin embargo siempre existe el riesgo de que sea comercializado y deje de estar disponible de forma abierta.

En resumen, para la toma de esta decisión se tendrán en cuenta factores como:

- Documentación.
- Seguridad, control de acceso a los recursos.
- Volúmenes de información que soportará y número de accesos esperable.
- Complejidad en la migración de los datos.
- Soporte ofrecido.

Como siempre, al final la solución estará determinada por las características de la organización en cuanto a requerimientos de su sistema de información y al personal de que dispone, recursos económicos, etc.

En todo caso una solución general de compromiso podría ser aquella que involucre software libre y un contrato de soporte.

RESUMEN DEL CAPÍTULO

En este capítulo introductorio hemos repasado someramente las principales características de los sistemas gestores de bases de datos actuales.

De los mismos, hemos visto sus componentes, funcionalidades principales y usuarios que trabajan con los denominados SGBD. Las posibilidades han crecido enormemente de un tiempo a esta parte tanto en lo que respecta a sistemas de pago como de código libre y hoy en día disponemos de un repertorio bastante amplio y potente de herramientas que hacen más fácil nuestra tarea, tanto como administración de bases de datos como de diseño y uso de las mismas.

EJERCICIOS PROPUESTOS

Responde a las siguientes preguntas.

1. Comenta qué se entiende por software libre considerando aspectos como:

- Gratuidad
- Código fuente
- Uso comercial

2. Lista al menos 3 ventajas e inconvenientes de los productos de pago respecto a los libres.

3. ¿Qué tiene que ver la administración de un SGBD con el diseño de bases de datos?

4. Cita al menos 3 ventajas de usar bases de datos frente a los tradicionales sistemas de ficheros.

5. Enumera al menos tres objetos típicos de una base de datos indicando su función.

6. ¿Qué es una base de datos distribuida?

7. Indica resumidamente las fases involucradas en el desarrollo de una base de datos desde su concepción hasta su puesta en marcha.

8. ¿Para qué sirve un disparador en un SGBD?

9. Explica con tus palabras qué es el diccionario de datos en un SGBD.

10. Eres administrador de la base de datos. Indica un problema y su posible solución que te pueda surgir considerando dos casos:

- **Caso 1:** Una base de datos con miles de usuarios y centrada en consultas, como un buscador.
- **Caso 2:** Una base de datos de venta online con miles de usuarios y operaciones por segundo.

Entregar Moodle

TEST DE CONOCIMIENTOS

TEST DE CONOCIMIENTOS UT 2 – ACCESO A LA INFORMACIÓN:

1. ¿Qué es una base de datos?
a) Un programa para organizar datos
b) Un software que facilita la gestión de datos
c) Un conjunto de datos organizados
d) Todo lo anterior

2. ¿Cuál es el significado de GPL en el contexto?
a) General Public Library
b) Great Politic Licence
c) General Public Licence

3. ¿Cuáles de los siguientes objetos no son equivalentes a una tabla?
a) Vista
b) Consulta
c) Trigger
d) Procedimiento

4. ¿Qué se quiere decir cuando se habla de nivel conceptual?
a) La que percibe el usuario
b) La imagen de base de datos vista por el ordenador
c) El código para crear la base de datos
d) Una imagen de la base de datos independiente de la implementación física

5. Las bases de datos son:
a) Relacionales
b) Relacionales o jerárquicas
c) Primero eran en red y ahora son relacionales
d) La mayoría son relacionales

6. Un modelo es:
a) Una forma de representar información
b) Un programa para dibujar cajas y flechas
c) Una forma de representar un sistema
d) Una representación de un conjunto de datos

7. Un sistema de información:
a) Describe los datos de un sistema
b) Permite controlar la información de una empresa
c) Es el conjunto de elementos para gestionar la información de un sistema
d) Todo lo anterior

8. ¿Qué es cierto respecto a los SGBD y bases de datos?
a) No hay diferencia
b) Uno hace referencia a un software y una base es conceptual
c) Las bases de datos se crean necesariamente con un SGBD
d) Un SGBD es una herramienta CASE

9. Los sistemas libres:
a) Son más potentes y mejores que los comerciales
b) Son más baratos
c) Son más difíciles
d) Ninguno de los anteriores necesariamente

10. La independencia física:
a) Permite acceder a los datos desde diferentes ubicaciones
b) Permite modificar los modelos independientemente de su almacenamiento
c) Evita problemas de redundancia
d) Hace que podamos usar las bases de datos independientemente del sistema operativo

Entregar Moodle