Tema 6: Bases de datos distribuidas

¿Qué aprenderás?

- Identificar las diferentes formas de fragmentar los datos.
- Replicar una base de datos.
- Instalar un clúster de datos.
- Reconocer la alta disponibilidad y tolerancia a fallos.

¿Sabías que…?

- La disponibilidad se expresa en porcentaje de funcionamiento durante todo un año.
- El clúster MySQL permite montar un clúster de bases de datos a coste 0.
- La base de datos del World Data Centre for Climate es una de las más grandes del mundo con 220 terabytes de datos en la web y 6 petabytes adicionales.

6. Bases de datos distribuidas

6.1. Introducción

En un sistema de bases de datos distribuido los componentes se encuentran repartidos en diferentes máquinas conectadas mediante redes de área local o internet. Estos sistemas por tanto no comparten ni memoria ni disco, a diferencia de los modelos centralizados donde todos los componentes residen en el mismo lugar.

En un sistema de bases de datos distribuido todos los datos pertenecen al mismo sistema de forma lógica pero físicamente se almacenan en diferentes máquinas conectadas a la red. De este modo podemos hablar de dos tipos de transacciones: locales y globales.

Las transacciones locales los datos y transacción se ejecutan en una sola máquina. En las globales se puede dar el caso que la transacción se realice en una máquina y los datos residan en otra, o que acceda a datos de diferentes máquinas.

Podemos destacar las siguientes ventajas de un Sistema Gestor de bases de datos distribuido.

- Proximidad: Los datos pueden localizarse cerca de los lugares donde hay más demanda.
- Permiten un procesamiento y acceso más rápido a los datos.
- Son sistemas escalables ya que permiten añadir nuevas máquinas a la red.
- Mejoría en las comunicaciones, los clientes están más cerca de los datos.
- Reducción de costes. Es más económico trabajar con varias máquinas pequeñas que con un gran servidor centralizado.
- Seguridad: En caso de fallo o sobrecarga de una máquina su carga de trabajo se puede desviar a otras.

Como inconvenientes de los sistemas de bases de datos distribuidos podemos citar los siguientes:

- El diseño y estructura de la base de datos es más compleja.
- Coste inicial de desplegar toda la infraestructura más elevado: duplicar personal, software, licencias, sistemas operativos y máquinas.
- Gestión más compleja: Los transacciones pueden llegar de diferentes máquinas y han de ser capaces de unir datos de diversos lugares. Optimización de consultas. Control de la concurrencia, copias de seguridad, recuperación del sistema.
- Debe aumentar la seguridad, los datos se encuentran repartidos en diferentes máquinas.

Los sistemas gestores de bases de datos los podemos clasificar en función de la distribución de los datos y de los procesos. Existen tres tipos:

No hay distribución: El proceso de transacciones y los datos se encuentran ubicados en la misma máquina.

Arquitectura cliente-servidor: El proceso de transacciones se puede realizar desde múltiples lugares (clientes) y los datos residen en un solo lugar (servidor)

Totalmente distribuido: El proceso de transacciones se realiza desde diversos lugares y los datos también pueden residir en diferentes máquinas.

Un SGBD distribuido consta como mínimo de los siguientes elementos:

- Las máquinas que forman el sistema distribuido.
- El software instalado en cada máquina.
- La red de comunicaciones que permite el transporte de datos entre diferentes máquinas.
- El procesador de transacciones que recibe y procesa las transacciones y devuelve la respuesta a los diferentes clientes.
- El procesador de datos que se encarga de almacenar y recuperar los datos localizados en cada máquina.

Los sistemas gestores de bases de datos distribuidos deben garantizar la transparencia de los datos a los usuarios. Un usuario no tiene que conocer donde se ubican físicamente los datos y la forma de acceder a ellos. El sistema gestor debe garantizar tres tipos de transparencia:

- La transparencia en ubicación (en qué máquina de la red se encuentran los datos).
- Transparencia en replicación. El usuario no ha de conocer si el objeto al que accede esta replicado en diferentes máquinas para mejorar la disponibilidad o el rendimiento.
- Transparencia en fragmentación. El usuario no ha de conocer si todos los datos están en la misma máquina o se han fragmentado en máquinas diferentes para mejorar el rendimiento.

6.2. Fragmentación

La fragmentación consiste en dividir las relaciones en tablas en diferentes fragmentos almacenados en diferentes máquinas. Estos fragmentos deben tener toda la información necesaria para poder reconstruir si es necesario todas las relaciones o tablas originales.

La fragmentación horizontal consiste en dividir las filas de las tablas o relaciones en dos o más subconjuntos en función de los valores almacenados en uno o varios atributos.

codigo	nombre	ciudad	antiguedad	salario	departamento
1	Marta Rojo	Barcelona	12	20000.00	comercial
2	Iván Pérez	Madrid	5	48000.00	facturación
3	Fernado Torres	Madrid	11	55000.00	comercial
4	María Rubio	Barcelona	4	36000.00	recursos
5	Isabel Llamas	Barcelona	13	28000.00	comercial
6	Manuel Gómez	Madrid	15	22000.00	facturación

Por ejemplo disponemos de los siguientes datos relativos a una serie de trabajadores. Un ejemplo de fragmentación horizontal podría consistir en dividir la tabla en función de la ciudad. De esta forma se pueden ubicar los datos en máquinas de cada ciudad.

Primer fragmento, ciudad de Barcelona.

codigo	nombre	ciudad	antiguedad	salario	departamento
1	Marta Rojo	Barcelona	12	20000.00	comercial
4	María Rubio	Barcelona	4	36000.00	recursos
5	Isabel Llamas	Barcelona	13	28000.00	comercial

Segundo fragmento, ciudad de Madrid.

codigo	nombre	ciudad	antiguedad	salario	departamento
2	Iván Pérez	Madrid	5	48000.00	facturación
3	Fernado Torres	Madrid	11	55000.00	comercial
6	Manuel Gómez	Madrid	15	22000.00	facturación

La fragmentación vertical consiste en dividir los atributos de la relación (los campos o columnas) en diferentes fragmentos. De tal forma que deben contener los datos que utilizaran más frecuentemente los usuarios de las máquinas donde serán almacenados.

Podríamos dividir la tabla de trabajadores en dos fragmentos verticales uno con los datos postales para la administración, y otro con los datos laborales para los de recursos humanos y nóminas. En los dos casos debe haber como mínimo la clave principal que enlaza todos los datos.

Primer fragmento vertical, datos postales.

codigo	nombre	ciudad
1	Marta Rojo	Barcelona
2	Iván Pérez	Madrid
3	Fernado Torres	Madrid
4	María Rubio	Barcelona
5	Isabel Llamas	Barcelona
6	Manuel Gómez	Madrid

Segundo fragmento vertical datos laborales.

codigo	antiguedad	salario	departamento
1	12	20000.00	comercial
2	5	48000.00	facturación
3	11	55000.00	comercial
4	4	36000.00	recursos
5	13	28000.00	comercial
6	15	22000.00	facturación

El tercer tipo de fragmentación es la fragmentación mixta que consiste en aplicar tanto la fragmentación vertical como la horizontal. Se puede aplicar primero la vertical y después la horizontal o al revés.

El siguiente ejemplo muestra una fragmentación primero horizontal por ciudad y luego vertical dividiendo los datos postales de los datos laborales.

codigo	antiguedad	salario	departamento
2	5	48000.00	facturación
3	11	55000.00	comercial
6	15	22000.00	facturación

codigo	nombre	ciudad
2	Iván Pérez	Madrid
3	Fernado Torres	Madrid
6	Manuel Gómez	Madrid

Ahora con la fragmentación horizontal de Barcelona.

codigo	antiguedad	salario	departamento
1	12	20000.00	comercial
4	4	36000.00	recursos
5	13	28000.00	comercial

codigo	nombre	ciudad
1	Marta Rojo	Barcelona
4	María Rubio	Barcelona
5	Isabel Llamas	Barcelona

El grado de fragmentación de la base de datos es un parámetro que debe ser valorado, ya que una base de datos muy fragmentada puede hacer disminuir notablemente el rendimiento en la ejecución de las consultas.

6.3. Replicación de una base de datos MySQL

Mysql permite replicar una base de datos maestra en varios servidores esclavos. La finalidad es balancear la carga de trabajo entre diferentes servidores especialmente útil en aplicaciones de consulta. Se trata de una replicación unidireccional asíncrona. Para implementar el sistema MySQL utiliza un fichero de log donde anota todas las modificaciones que realiza en el sistema. Los esclavos leen este fichero para actualizar las bases de datos replicadas.

Este sistema aporta los siguientes beneficios:

- Disponibilidad: Sistema más robusto, en caso de fallo en el servidor maestro se puede utilizar o substituir por un esclavo
- Aumento del paralelismo: Se pueden dividir los selects entre los diferentes servidores de esta manera mejoramos el tiempo de respuesta.
- Aumento de la sobrecarga en las actualizaciones: El sistema debe asegurar los cambios deben propagarse a todas a lo largo del sistema distribuido.
- Rendimiento: Los servidores trabajan en paralelo, lo cual permite balancear la carga se pueden hacer todas las consultas SELECT a un servidor esclavo, mientras las actualizaciones se realizan en el maestro.
- Modularidad: se pueden modificar, agregar o quitar servidores de la base de datos distribuida sin afectar a los demás sistemas.

Como desventajas podemos indicar:

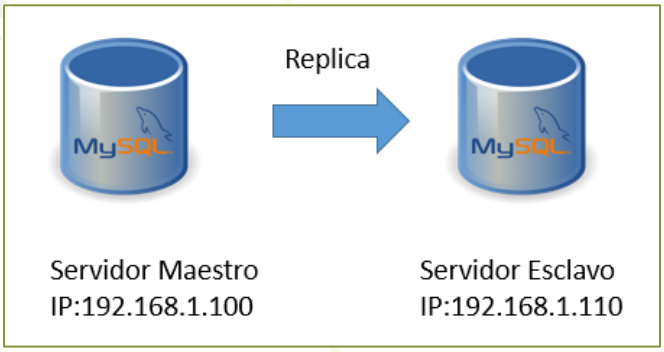
- Complejidad: Los servidores pueden estar implementados en infraestructuras diferentes.
- Seguridad: se debe trabajar en la seguridad de la infraestructura así como cada uno de los sistemas.
- Mayor probabilidad de errores ante fallos del sistema.

El funcionamiento del sistema es el siguiente:

- Se ejecutan las consultas en el servidor maestro y éstas son registradas en un log-binario.
- El master manda un evento al proceso esclavo para avisarle que algo ha cambiado en el maestro
- El servidor esclavo lee la siguiente posición desde la última leída del log binario hasta el final y obtiene las consultas nuevas del servidor maestro.
- El esclavo ejecuta las consultas leídas en su sistema.

6.3.1. Implementación de la replicación en MySQL.

El modelo más básico consiste en la implementación con un maestro y un solo esclavo, aunque lo más habitual sería un maestro y varios esclavos. Vamos a implementar el primer caso con dos máquinas virtuales de Windows 10 y MySQL 5.7 instalado.



Las máquinas se han clonado. En este caso debemos tener en cuenta que mysql tiene el mismo identificador en las dos instalaciones. Accedemos al archivo auto.cnf del esclavo y cambiamos el valor del UUID (solo cambiando una letra es suficiente). Y reiniciamos el servidor.

```
[auto]
server-uuid=61d803b1-e3c1-11ea-b8a1-0800274b3b0f
```

Los servidores deberán estar identificados con una variable Server-id, el número debe ser diferente en el maestro y en el esclavo y deberemos dar permisos de replicación a un usuario del servidor esclavo.

Acciones a realizar en el servidor maestro.

- Modificar el archivo my.cnf que se encuentra en el directorio \ProgramData\ MySQL\MySQL Server 5.7. Le añadimos al final del fichero el identificador del servidor y el archivo log que almacena los cambios a replicar.

```
server-id=1
log-bin=mysql-bin
```

- Reiniciamos el servidor Mysql.
- Damos permiso de replicación sobre todo a un usuario del servidor esclavo, en mi caso utilizo el usuario.

```
mysql> grant replication slave on *.* to 'root'@'192.168.1.110' identified by '1234';
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

- Actualizamos los privilegios (flush privileges) y mostramos el estado del master (show master privileges). Obtenemos el nombre del fichero donde el maestro va escribiendo las transacciones realizadas y la posición.

```
mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)

mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000001 |      601 |              |                  |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

Acciones a realizar en el servidor esclavo.

- Modificar el archivo my.cnf que se encuentra en el directorio \ProgramData\ MySQL\MySQL Server 5.7. Le añadimos al final del fichero el identificador del servidor debe ser diferente al del maestro.

```
server-id=2
```

- Reiniciamos el servidor Mysql.

- Tras el cambio y una vez que hayamos reiniciado el servidor MySQL Esclavo, nos conectamos a su interfaz de línea de comando para ejecutar la instrucción CHANGE MASTER TO para indicar:
    - MASTER_HOST: El nombre del host o dirección IP del servidor Maestro.
    - Master_user: Es el usuario con privilegios de replicación
    - Master_password: La clave que le hemos asignado al usuario.
    - Master_log_file: El nombre del log donde el Master va escribiendo las transacciones realizadas.
    - Master_log_pos: La posición del Maestro.

```
mysql> change master to
    -> master_host='192.168.1.100',
    -> master_user='root',
    -> master_password='1234',
    -> master_log_file='mysql-bin.000001',
    -> master_log_pos=601;
Query OK, 0 rows affected, 1 warning (0.03 sec)
```

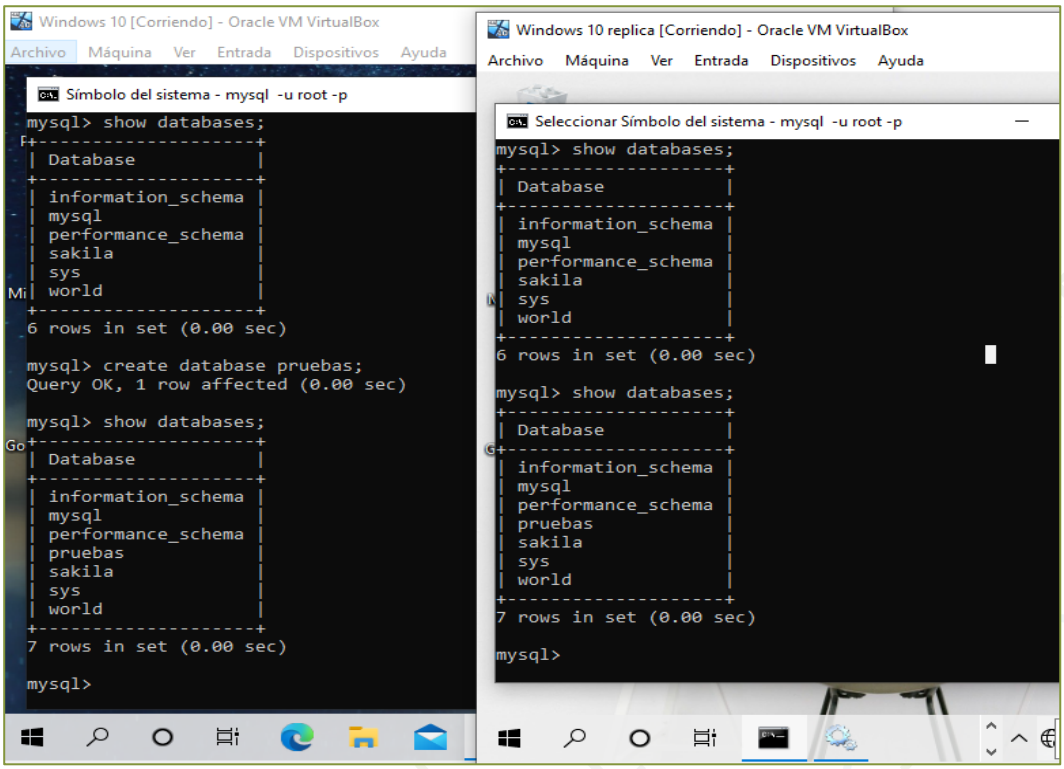
- Reiniciamos el servidor.

```
mysql> start slave;
Query OK, 0 rows affected (0.00 sec)
```

- Comprobamos las variables del esclavo y debería estar en escucha.

```
mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.1.100
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000001
```

Ahora solo queda comprobar que los cambios que realizamos en el master se replican en el esclavo. Creamos una base de datos en el maestro y debería aparecer en el esclavo.

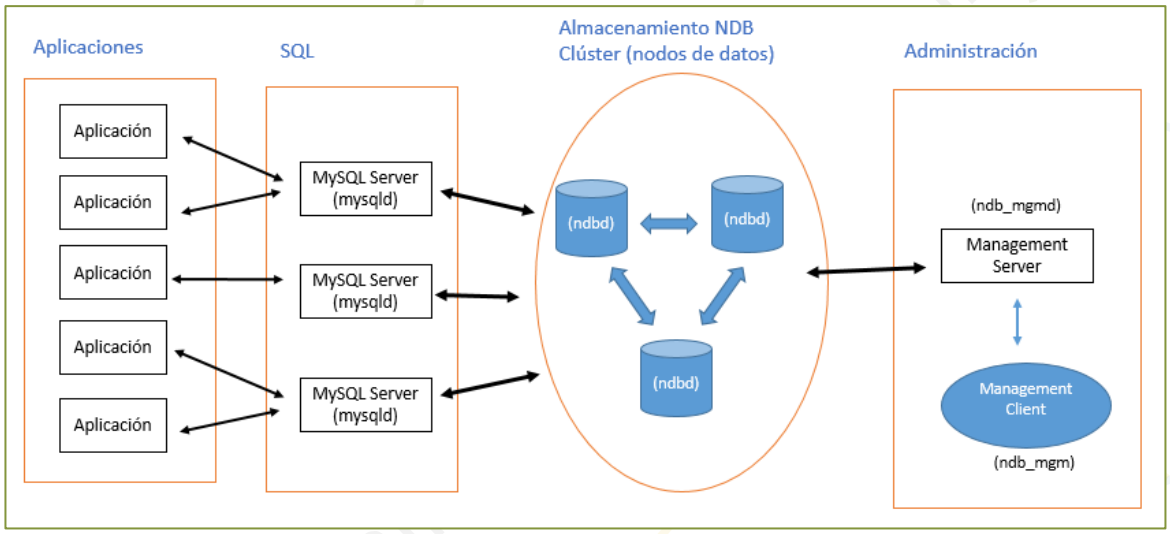


6.4. Clúster de datos

MySQL Cluster es una versión de mysql para un entorno de bases de datos distribuidas, que soporta alta disponibilidad y alta redundancia. Se pueden ejecutar varios servidores MySQL utilizando el motor de almacenamiento NDB Clúster.

Un Clúster MySQL integra el servidor MySQL estándar con un motor de almacenamiento clusterizado en memoria llamado NDB.

Un Clúster MySQL consiste en un conjunto de máquinas, cada una ejecutando un número de procesos incluyendo servidores MySQL, nodos de datos para Clúster NDB, servidores de administración, y aplicaciones que acceden a los datos. Los componentes los vemos en el siguiente esquema:



Todos estos programas funcionan juntos para formar un Clúster MySQL. Cuando se almacenan los datos en el motor NDB, las tablas se almacenan en los nodos de datos. Las tablas son accesibles desde todos los otros servidores MySQL en el clúster.

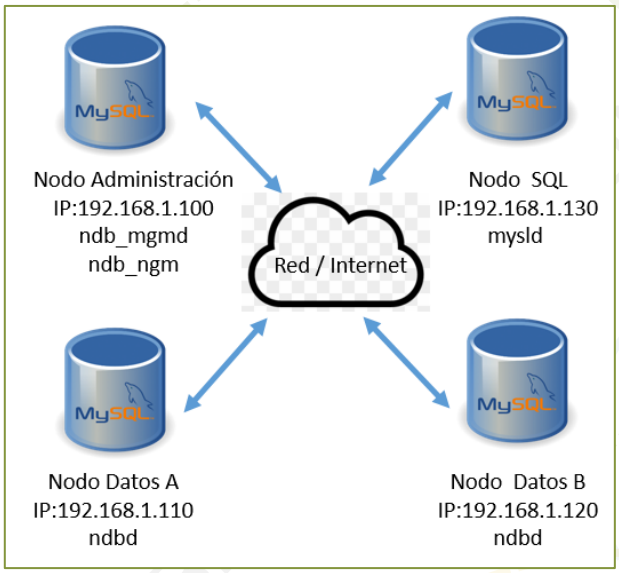
En el Clúster de MySQL hay tres tipos de nodos clúster, y en cualquier configuración de clústeres de Mysql por pequeña que sea, como mínimo debe haber uno de cada tipo.

- **El nodo de administración (MGM):** El rol de este tipo de nodo es administrar los otros nodos dentro del Clúster de MySQL. Proporciona los datos de la configuración del clúster y es el primer nodo que debe arrancar antes de cualquier otro. Se utiliza el comando ndb_mgmd para arrancar un nodo MGM.
- **El nodo de datos:** Este nodo que almacena los datos del clúster. Hay tantos nodos de datos como réplicas, multiplicado por el número de fragmentos. Con un nodo de datos hay suficiente, pero si buscamos redundancia y alta disponibilidad real necesitaremos más nodos. Para iniciar el nodo de datos utilizamos el comando ndbd.

- **El nodo SQL:** Este es el nodo que accede a los datos del clúster. En el caso de MySQL es un servidor MySQL tradicional que usa el motor NDB. El sistema puede incorporar varios nodos SQL.

6.4.1. Implementación de un clúster en MySQL.

Planteamos la implementación de un clúster con 4 nodos. Dos de datos, uno de SQL y otro de administración el esquema seria el siguiente.



La configuración de un clúster implica configurar cada nodo individual en el clúster y inicializar los enlaces de comunicación individual entre los nodos. En un único fichero de configuración se almacenan todos los datos de configuración del clúster.

En el nodo SQL, editamos o creamos el fichero my.cnf y añadimos las dos siguientes líneas ndbcluser y cuál es el nodo administrador.

```
[mysqld]
ndbcluster
ndb-connectstring=192.168.1.100
```

En los nodos de datos editamos o creamos el fichero my.cnf y añadimos la información del nodo administrador.

```
[mysql_cluster]
ndb-connectstring=192.168.1.100
```

En el nodo de administración creamos un fichero de configuración llamado config.ini, donde indicamos la composición del clúster y de todos sus nodos. Debemos indicar el número de réplicas y los directorios donde cada nodo almacena los datos y los logs.

```
[ndbd default]
NoOfReplicas=2

[ndb_mgmd]
hostname=192.168.1.100
datadir=c:\mysql\mysql-cluster

[ndbd]
hostname=192.168.1.110
datadir=c:\mysql\data

[ndbd]
hostname=192.168.1.110
datadir=c:\mysql\data

[mysqld]
hostname=192.168.1.130
datadir=c:\mysql\data
```

Ahora solo falta iniciar cada nodo, primero el de administración, luego los de datos y para finalizar el de SQL.

Para iniciar el nodo de administración utilizamos la instrucción ndb_mgmd indicándole en qué lugar está el fichero de configuración.

```
C:\mysql\bin>ndb_mgmd -f config.ini
MySQL Cluster Management Server mysql-5.7.26 ndb-7.6.10
```

Para iniciar los nodos de datos.

```
C:\mysql\bin>ndbd --initial
```

Para iniciar el nodo SQL ejecutamos un mysqld.

```
C:\mysql\bin>mysqld
```

Video: Replicación

Recursos y enlaces

- Clúster MySQL

- Replicación en MySQL

- Configuración de un clúster

Conceptos clave

- **Base de datos distribuidas**: Los componentes se encuentran distribuidos en diferentes máquinas.
- **Fragmentación:** División de una tabla o relación con el objetivo de mejorar su rendimiento.
- **Replicación**: Mantener una copia de la base de datos en servidores esclavos para aumentar el rendimiento sobre todo de lectura.
- **Clúster:** Para implementar soluciones de alta disponibilidad.

Test de autoevaluación

En un sistema replicado el Master_log_file almacena:

a) Las transacciones que realiza el maestro.
b) Las transacciones que realiza el esclavo.
c) El estado de la replicación.
d) Los errores de replicación.

El esquema básico de una replicación MYSQL consiste en:

a) Un maestro y dos esclavos.
b) Dos maestros y un esclavo.
c) Un maestro y un esclavo.
d) Un maestro, un esclavo y un sql.

¿Cuál de las siguientes no es un tipo de fragmentación?

a) Vertical
b) Red
c) Horizontal
d) Mixta

Ponlo en práctica

Actividad 1

¿Qué modificaciones realizarías en la configuración del clúster si queremos añadir dos nodos más de datos 192.168.1.140 y 192.168.1.150 y otro nodo SQL 192.168.1.160.?

SOLUCIONARIOS

Test de autoevaluación

En un sistema replicado el Master_log_file almacena:

e) Las transacciones que realiza el maestro.

f) Las transacciones que realiza el esclavo.

g) El estado de la replicación.

h) Los errores de replicación.

El esquema básico de una replicación MYSQL consiste en:

e) Un maestro y dos esclavos.

f) Dos maestros y un esclavo.

g) Un maestro y un esclavo.

h) Un maestro, un esclavo y un sql.

¿Cuál de las siguientes no es un tipo de fragmentación?

e) Vertical

f) Red

g) Horizontal

h) Mixta

Ponlo en práctica

Actividad 1

¿Qué modificaciones realizarías en la configuración del clúster si queremos añadir dos nodos más de datos 192.168.1.140 y 192.168.1.150 y otro nodo SQL 192.168.1.160.?

Solución:

- **En el nodo de datos máquina 192.168.1.140 editamos el fichero my.cnf y añadimos la información del nodo administrador.**

  [mysql_cluster]
  ndb-connectstring=192.168.1.100

- **En el nodo de datos máquina 192.168.1.150 editamos el fichero my.cnf y añadimos la información del nodo administrador.**

  [mysql_cluster]
  ndb-connectstring=192.168.1.100

- **En el nodo SQL máquina 192.168.1.160 editamos el fichero my.cnf y añadimos la información del nodo administrador.**

  [mysqld]
  ndbcluster
  ndb-connectstring=192.168.1.100