Replicación de Bases de Datos

# **1.- Concepto de replicación y tipos**

Concepto

Una **replicación de base de datos** es una técnica mediante la cual copiamos de forma exacta en otra ubicación una instancia de la base de datos. Se utiliza en entornos distribuidos de Sistemas de Gestión de Bases de Datos donde una sola base de datos tiene que ser utilizada y actualizada en varios lugares de forma simultánea.

Permite que el contenido de uno o más servidores (llamados maestros) se refleje en uno o más servidores (llamados esclavos).

Los términos maestro y esclavo se han utilizado históricamente en la replicación, pero ahora se prefieren los términos primario y réplica .

Tipos de replicación

Podemos hablar al menos de 3 tipos de **replicación de base de datos**:

* **Replicación Instantánea**: los datos de un servidor son simplemente copiados a otro servidor o a otra base de datos dentro del mismo servidor. Al copiarse todo no necesitas un control de cambios. Se suele utilizar cuando los datos cambian con muy poca frecuencia.
* **Replicación Transaccional**: primero se envía una copia completa de la base de datos y luego se van enviando de forma periódica (o a veces continua) las actualizaciones de los datos que cambian. Se utiliza cuando necesitas que todos los nodos con todas las instancias de la base de datos tengan los mismos datos a los pocos segundos de realizarse un cambio.
* **Replicación de mezcla**: los datos de dos o más bases de datos se combinan en una sola base de datos. En primer lugar se envía una copia completa de la base de datos. Luego el Sistema de Gestión de Base de Datos va comprobando los cambios que van apareciendo en los distintos nodos y a una hora programada o a petición los datos se sincronizan. Es sobre todo útil cuando cada nodo suele utilizar solo los datos que se actualizan allí pero que por circunstancias necesita tener también los datos de los otros sitios.

# **2.- Ventajas e inconvenientes de la replicación**

La replicación se usa mucho en sistemas de acceso a datos ya que puede ofrecer grandes beneficios relacionados principalmente con el rendimiento, disponibilidad y seguridad de los datos

* **Aumento de la fiabilidad**: mediante la replicación de base de datos a través de múltiples servidores, te aseguras que los datos van a estar disponibles incluso en el caso de que una de las máquinas tenga un fallo grave de hardware. El sistema distribuido de gestión de bases de datos debe ser capaz de enrutar a los usuarios afectados a otro de los nodos disponibles.
* **Mejora en el rendimiento**: al estar los datos distribuidos en diferentes servidores, los múltiples accesos no saturan los servidores, dándose el paralelismo de consultas. Esto es importante sobre todo en el caso de aplicaciones que pueden tener miles o cientos de miles de peticiones simultáneas. El rendimiento de las aplicaciones aumenta notablemente y hay una menor carga del sistema ya que disminuye el tráfico de red.
* **Mejora en la seguridad de los datos**: ya que las actualizaciones están siendo escritas en varios servidores. Es decir, varios discos, varias fuentes de alimentación, CPU’s, etc. son utilizadas para asegurar que tus datos estarán a salvo en algunos servidores, aunque pueda ocurrir un desastre en otros lo que le otorga **fiabilidad.**
* [**Alta disponibilidad**](https://es.wikipedia.org/wiki/Disponibilidad)**:** si falla algún nodo esa relación puede estar en otro nodo y el sistema continúa
* **Procesamiento desconectado** la replicación puede funcionar por instantáneas, lo cual permite trabajar en un conjunto de datos, aunque no estén conectados al servidor principal. Cuando vuelvan a conectarse tendrán que sincronizarse.

En cuanto a los inconvenientes, hay que mencionar los siguientes:

* Las técnicas de control de concurrencia y recuperación son más costosas que en un sistema que no está replicado.
* Se transmiten muchos datos en el momento de actualizar la replicación, lo que implica ocupar las líneas de comunicaciones.

# **3.- Componentes de la replicación transaccional**

* Objeto de replicación: puede ser una base de datos completa, un índice, una viste, etc.
* Grupo de replicación: es un conjunto de objetos de replicación que tienen alguna lógica
* Sitio de replicación: son los nodos donde se encuentra cada grupo de replicación. Hay dos tipos de sitios: maestros y esclavos

En función del momento en que se realiza la réplica, esta puede ser:

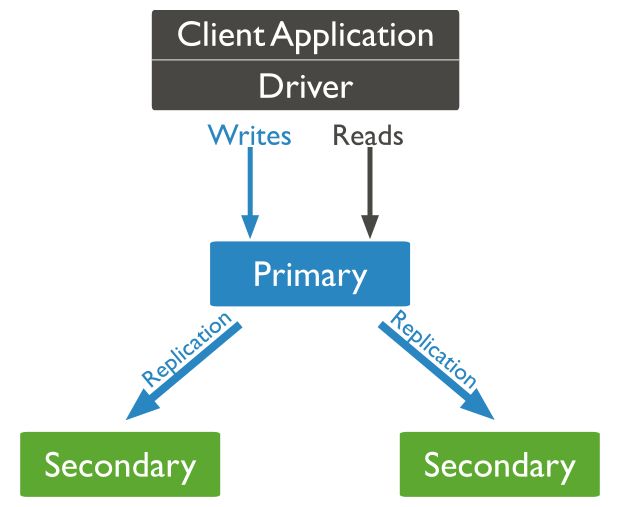
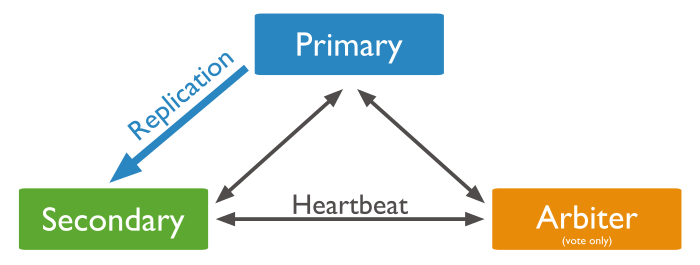
* Síncrona: los datos replicados se actualizan a la vez en toda la base como en el momento de actualizarse en el sitio donde se origina la transacción.
* Asíncrona: en este procedimiento las demás bases de datos replicadas se actualizan más tarde de la réplica que se modificó.

**4.- Replicación NoSql**

*MongoDB* es una de las bases de datos NoSQL más conocidas. Sigue un modelo de datos documental, donde los documentos se basan en JSON.

*MongoDB* destaca porque:

* Soporta esquemas dinámicos: diferentes documentos de una misma colección pueden tener atributos diferentes.
* No soporta *joins*, ya que no escalan bien.
* No soporta transacciones. Lo que en un SGDB puede suponer múltiples operaciones, con *MongoDB* se puede hacer en una sola operación al insertar/actualizar todo un documento de una sola vez.

En *MongoDB* se replican los datos mediante un conjunto de réplicas (*Replica Set*), que es un grupo de servidores (nodos) donde hay uno que ejerce la función de **primario** y por tanto recibe las peticiones de los clientes, y el resto de servidores hace de **secundarios**, manteniendo copias de los datos del primario.

Si el nodo primario se cae, los secundarios eligen un nuevo primario entre ellos mismos, en un proceso que se conoce como votación. La aplicación se conectará al nuevo primario de manera transparente. Cuando el antiguo nodo primario vuelva en sí, será un nuevo nodo secundario.

Los tipos de nodos que podemos encontrar en un conjunto de réplica son:

* Regular: Es el tipo de nodo más común.

**Primario**: Acepta todas las operaciones de escritura de los clientes. Cada conjunto de réplicas tendrá sólo un primario, y como sólo un miembro acepta operaciones de escritura, ofrece consistencia estricta para todas las lecturas realizadas desde él.

**Secundario**: Los secundarios replican el *oplog* primario y aplican las operaciones a sus conjuntos de datos. De este modo, los nodos secundarios son un espejo del primario. Si el primario deja de estar disponible, el conjunto de réplica elegirá a un secundario para que sea el nuevo primario, mediante un proceso de votación.

* **Árbitro**: se emplea sólo para votar. No contiene copia de los datos, no se puede convertir en primario y no requieren hardware dedicado. Solo está presente cuando hay un número par de nodos.
* Retrasado (*delayed*): nodo que se emplea para la recuperación del sistema ante un fallo. Para ello, hay que asignar la propiedad priority:0. Este nodo nunca será un nodo primario.
* Oculto: empleado para analíticas del sistema.

##### **oplog (operations log)**

Para soportar la replicación, el nodo primario almacena todos los cambios en su ***oplog***.

De manera simplificada, el *oplog* es un diario de todos los cambios que la instancia principal realiza en las bases de datos con el propósito de replicar dichos cambios en un nodo secundario para asegurar que las dos bases de datos sean idénticas. El *oplog* crea un *timestamp* para cada entrada.

# **5.- Diferencias entre Backup y Replicación**

|  |  |  |
| --- | --- | --- |
|  | **Backup** | **Replicación** |
| **Requisitos** | Un disco o dispositivo de almacenamiento | Dos o más servidores |
| **Recomendado** | Almacenamiento de copias de seguridad | Aplicaciones críticas o siempre online |
| **Coste** | Bajo | Alto |
| **A favor** | Fácil de implementar  Económico  Aislamiento ante amenazas | Alta disponibilidad  Alta recuperación ante desastres  Inicio muy rápido |
| **En contra** | Los tiempos de creación y recuperación son largos | Existe el riesgo que el malware se expanda por todos los sistemas |

# 

# 

# 

# 

# **6.- Prácticas**

## **Configuración Master-Slave en MariaDB**

Necesitamos dos servidores con Mariadb, a los cuales le asignamos una IP.

Ejemplo:

**Master – 192.168.0.2**

**Slave – 192.168.0.3**

### **Paso 1 - Configuración del Master:**

|  |
| --- |
| $ nano /etc/my.cnf |

y luego dejarlo así:

|  |
| --- |
| [mysqld]  log-bin  server\_id=1  replicate-do-db=important  bind-address=192.168.0.2 |

Guardar y luego reiniciar mariadb:

|  |
| --- |
| $ systemctl restart mariadb |

Loguearse:

|  |
| --- |
| $ mysql –u root –p |

Esto creará un nuevo usuario para el esclavo llamado «slaveuser» y le asignará los privilegios necesarios:

|  |
| --- |
| STOP SLAVE;  GRANT REPLICATION SLAVE ON \*.\* TO ‘slaveuser’@’%’ IDENTIFIED BY ‘iamslave’;  FLUSH PRIVILEGES;  FLUSH TABLES WITH READ LOCK;  SHOW MASTER STATUS; |

### **Paso 2 - Crear un backup:**

|  |
| --- |
| $ mysqldump –u root –p prueba\_replia > prueba\_replica\_backup.sql |

Luego:

|  |
| --- |
| $ mysql –u root –p  UNLOCK TABLES; |

|  |
| --- |
| $ scp prueba\_replica\_backup.sql root@IPesclavoservice --status-all  :/data |

Esto completa nuestra configuración en el servidor maestro, ahora pasaremos a configurar nuestro servidor esclavo.

### **Paso 3 - Configuración del SLAVE:**

Comenzaremos de nuevo con la edición del archivo ‘/etc/my.cnf’ y buscaremos la sección [mysqld] e ingresaremos los siguientes detalles:

|  |
| --- |
| [mysqld]  server-id = 2  replicate-do-db=important  […] |

|  |
| --- |
| $ mysql -u root -p < /data/ important\_backup.sql |

Cuando se complete el proceso, le otorgaremos los privilegios a «slaveuser» en la base de datos «important» al iniciar sesión en mariadb en el servidor esclavo.

|  |
| --- |
| $ mysql –u root –p |

|  |
| --- |
| GRANT ALL PRIVILEGES ON important.\* TO ‘slaveuser’@’localhost’ WITH GRANT  OPTION;  FLUSH PRIVILEGES; |

Siguiente reinicio de mariadb para implementar los cambios:

|  |
| --- |
| $ systemctl restart mariadb |

### **Paso 4 - Iniciar la replicación:**

Necesitamos las variables MASTER\_LOG\_FILE y MASTER\_LOG\_POS que obtuvimos al ejecutar «SHOW MASTER STATUS» en mariadb en el servidor maestro. Ahora ingrese a mariadb en el servidor esclavo y le diremos a nuestro servidor esclavo dónde buscar el maestro ejecutando los siguientes comandos:

|  |
| --- |
| STOP SLAVE;  CHANGE MASTER TO MASTER\_HOST=’192.168.0.2′, MASTER\_USER=’slaveuser’,  MASTER\_PASSWORD=’iamslave’, MASTER\_LOG\_FILE=’mariadb-bin.000001′,  MASTER\_LOG\_POS=460;  START SLAVE;  SHOW SLAVE STATUS\G; |

### **Paso 5 - Probar:**

Ahora crearemos una nueva tabla en nuestra base de datos en el maestro para asegurarnos de que la replicación esté funcionando o no. Entonces, ingrese a mariadb en el servidor maestro:

|  |
| --- |
| $ mysql –u root –p |

|  |
| --- |
| use important;  create table x (c int); |

|  |
| --- |
| insert into x (c) value (1); |

|  |
| --- |
| select \* from x; |

Ahora iniciemos sesión en nuestra base de datos esclava para asegurarnos de que nuestra replicación de datos esté funcionando

|  |
| --- |
| $ mysql –u root –p  $ use important;  $ select \* from x; |

Con eso podrá visualizar que la salida muestra el mismo valor que insertamos en el servidor maestro, por lo tanto, la replicación funciona correctamente sin ningún problema.

# **Replicación NoSql**

### 