Тема 9. Установка MySQL сервера. Резервное копирование. Репликация. Настройка бинарного логирования в MySQL. Восстановление данных. настройка логирования.

Цель занятия:

Научиться устанавливать MySQL, выполнять резервное копирование и восстановление данных, настраивать репликацию и логирование для обеспечения стабильной и безопасной работы базы данных.

Учебные вопросы:

1. Установка MySQL сервера.
2. Резервное копирование MySQL.
3. Восстановление данных.
4. Настройка бинарного логирования в MySQL.
5. Репликация в MySQL.
6. Настройка логирования в MySQL.

1. Установка MySQL сервера.

MySQL — это одна из самых популярных систем управления реляционными базами данных (СУБД), которая использует язык SQL (Structured Query Language) для работы с данными. MySQL широко используется благодаря своей скорости, надежности и легкости в использовании. Она открыта для использования на условиях свободной лицензии, что делает ее привлекательной как для небольших проектов, так и для крупных веб-приложений.

Области применения MySQL:

- **Веб-приложения**: MySQL часто используется в веб-разработке для хранения данных пользователей, постов, комментариев и т. д. Примеры: WordPress, Joomla, Drupal.
- **Корпоративные системы**: Базы данных в MySQL могут обслуживать ERP, CRM, бухгалтерские системы.
- **Электронная коммерция**: Магазины с большим количеством товаров и пользователей используют MySQL для управления заказами, пользователями и транзакциями.
- **Обработка данных и аналитика**: MySQL может использоваться для хранения данных и выполнения аналитических запросов с использованием агрегации и фильтрации.

MySQL — это надежное решение для большинства приложений, требующих реляционного хранения данных и поддерживающих масштабируемость и высокую производительность.

Поддерживаемые операционные системы:

- Windows: Версии Windows 10, Windows Server 2012 и выше.
- macOS: Начиная с версии macOS 10.13 (High Sierra) и выше.
- Linux: Debian/Ubuntu, RHEL/CentOS, Fedora, SUSE, Arch Linux, FreeBSD, Solaris.

Установка MySQL на Ubuntu.

```
sudo apt update
sudo apt install mysql-server
```

```
sudo mysql_secure_installation
```

```
sudo systemctl status mysql
```

2. Резервное копирование MySQL.

Резервное копирование (или бэкап) — это процесс создания копий базы данных, которые можно использовать для восстановления данных в случае их утраты, повреждения или изменения.

В MySQL резервное копирование осуществляется для предотвращения потерь данных из-за сбоев, ошибок пользователей, атак или других непредвиденных обстоятельств.

Зачем нужно резервное копирование:

- Предотвращение потери данных: Регулярные бэкапы помогают восстановить данные в случае аппаратных или программных сбоев.
- Защита от человеческих ошибок: Неправильные команды, удаление данных или изменения в структуре базы могут быть легко устранены путем восстановления из резервной копии.
- Безопасность при атаках: В случае вирусной атаки, взлома или нарушения безопасности можно вернуть состояние данных до инцидента.
- Миграция и обновление системы: Резервные копии используются при миграции базы данных на новый сервер или при переходе на новую версию MySQL.
- Обеспечение непрерывности бизнеса: Восстановление данных после сбоя помогает минимизировать время простоя и избежать больших убытков.
- Тестирование и разработка: Копии базы данных могут использоваться для тестирования новых функций или обновлений без риска повредить основную базу данных.

Основные методы резервного копирования в MySQL:

1. Логическое резервное копирование (mysqldump):

Используется утилита mysqldump, которая сохраняет данные и структуру базы данных в виде SQL-запросов. Это текстовые файлы, которые могут быть восстановлены с помощью выполнения SQL-запросов.

Пример создания бэкапа:

```
mysqldump -u root -p имя_базы > backup.sql
```

Пример восстановления:

```
mysql -u root -p имя_базы < backup.sql
```

Преимущества:

- Простота использования.
- Копия легко переносима между различными версиями MySQL.
- Подходит для небольших и средних баз данных.

Недостатки:

- Медленное восстановление для больших баз данных.
- Логические бэкапы занимают больше времени и места, чем физические.

2. Физическое резервное копирование (снимки файловой системы):

Физические копии базы данных представляют собой копирование файлов данных MySQL. Это быстрый способ создания бэкапов, особенно для больших баз данных.

Пример:

Копирование файлов данных MySQL вручную или с использованием инструментов (например, rsync или tar), когда сервер остановлен, или с помощью снимков файловой системы (snapshot) на уровне хранилища.

Преимущества:

- Быстрое создание копии даже для больших баз данных.
- Можно использовать инструменты для горячих бэкапов (например, LVM snapshots).

Недостатки:

- Менее гибкие в переносе данных между разными системами.
- Рекомендуется для опытных администраторов.

3. Инкрементное и дифференциальное резервное копирование (XtraBackup):

Percona XtraBackup — это утилита для создания горячих резервных копий (не останавливая сервер). Она поддерживает полные, инкрементные и дифференциальные бэкапы.

Полный бэкап: полная копия всех данных.

Инкрементный бэкап: копируются только изменения, сделанные с момента последнего полного или инкрементного бэкапа.

Дифференциальный бэкап: копируются все изменения, сделанные после последнего полного бэкапа.

Преимущества:

- Не требуется останавливать сервер для создания копии.
- Более эффективное использование места и времени благодаря инкрементным копиям.

Недостатки:

- Требует дополнительных настроек и знаний для восстановления инкрементных бэкапов.

Пример стратегии резервного копирования:

- Полное резервное копирование — раз в неделю.
- Инкрементное резервное копирование — каждый день.
- Сохранение бэкапов — минимум на 7-30 дней (в зависимости от потребностей).
- Автоматизация — настройка регулярных задач с помощью планировщика (например, cron на Linux).

Рекомендации по резервному копированию:

- Создавать копии регулярно (ежедневно или еженедельно).
- Хранить резервные копии на разных физических или облачных серверах.
- Проверять и тестировать восстановление данных с резервных копий, чтобы убедиться в их работоспособности.
- Защищать резервные копии с помощью шифрования и ограничения доступа.

3. Восстановление данных.

Восстановление данных — это процесс возврата базы данных к предыдущему состоянию с использованием резервной копии.

Этот процесс может быть необходим в случаях сбоя системы, потери данных, ошибок пользователей, а также при миграции или обновлении базы данных.

Способ восстановления зависит от того, какой тип резервной копии использовался (логическое или физическое копирование).

1. Восстановление из логической резервной копии (mysqldump)

Если резервная копия была сделана с помощью утилиты mysqldump, восстановление выполняется путем исполнения SQL-запросов из файла резервной копии. Это один из самых простых методов восстановления.

Пример восстановления:

Убедитесь, что MySQL-сервер запущен.

Используйте команду для восстановления

```
mysql -u root -p имя_базы < backup.sql
```

Если необходимо восстановить структуру и данные целой базы данных:

```
mysql -u root -p < backup.sql
```

Примечания:

Убедитесь, что база данных, в которую вы восстанавливаете данные, существует. Если нет, создайте ее:

```
mysql -u root -p
CREATE DATABASE имя_базы;
```

Файл резервной копии backup.sql должен быть доступен на сервере.

2. Восстановление из физической резервной копии.

Физические резервные копии включают копирование файлов базы данных, такие как файлы *.ibd и *.frm.

Этот метод требует особой осторожности, поскольку восстановление может вызвать проблемы с целостностью данных, если не соблюсти нужную последовательность действий.

Шаги восстановления:

Остановите сервер MySQL:

```
sudo systemctl stop mysql
```

Восстановите файлы из резервной копии. Копируйте файлы базы данных (например, с помощью rsync):

```
rsync -av /путь_к_резервной_копии /var/lib/mysql
```

Убедитесь, что права доступа к файлам правильные:

```
sudo chown -R mysql:mysql /var/lib/mysql
```

Запустите сервер MySQL:

```
sudo systemctl start mysql
```

Примечания:

Этот метод особенно подходит для больших баз данных, где создание логической копии с помощью mysqldump занимает слишком много времени.

Важно восстанавливать файлы на том же сервере или совместимом сервере, чтобы избежать проблем с версией MySQL и операционной системы.

3. Восстановление из инкрементных и дифференциальных резервных копий

Если вы использовали такие инструменты, как Percona XtraBackup для создания инкрементных или дифференциальных бэкапов, процесс восстановления включает несколько шагов: сначала восстанавливается полный бэкап, а затем применяются инкрементные или дифференциальные копии.

Шаги восстановления:

Полное восстановление:

```
xtrabackup --copy-back --target-dir=/путь_к_полной_резервной_копии
```

Применение инкрементных бэкапов:

```
xtrabackup --apply-log --incremental-dir=/путь_к_инкрементному_бэкапу --target-dir=/путь_к_полной_резервной_копии
```

Запуск MySQL после восстановления данных:

```
sudo systemctl start mysql
```

Этот процесс позволяет быстро восстановить данные даже для больших баз данных, сохраняя только изменения, а не всю базу.

4. Настройка бинарного логирования в MySQL.

Бинарное логирование (binary logging) в MySQL — это механизм, с помощью которого сервер сохраняет все изменения в базе данных, такие как модификации данных (INSERT, UPDATE, DELETE), в бинарные логи.

Эти логи могут быть использованы для восстановления базы данных после сбоев, выполнения репликации и анализа изменений.

Зачем нужно бинарное логирование:

- Восстановление данных: Позволяет восстановить базу данных до последнего коммита, используя резервную копию и бинарные логи.
- Репликация: Бинарные логи используются для передачи изменений с основного сервера на серверы-реплики в режиме master-slave.
- Аудит: Логи можно использовать для отслеживания изменений и аудита активности в базе данных.

Как включить и настроить бинарное логирование.

1. Редактирование файла конфигурации MySQL

Для включения бинарного логирования необходимо отредактировать файл конфигурации MySQL my.cnf (или my.ini на Windows). Этот файл обычно находится по следующим путям:

На Linux: /etc/mysql/my.cnf или /etc/my.cnf

На Windows: C:\ProgramData\MySQL\MySQL Server X.Y\my.ini

Добавьте или измените следующие параметры в секции [mysqld]:

```
[mysqld]
# Включение бинарного логирования
log-bin = /var/log/mysql/mysql-bin.log
# Идентификатор сервера для репликации (должен быть уникальным в кластере)
server-id = 1
# (Опционально) Указание максимального размера бинарного лога (по умолчани
max_binlog_size = 100M
# (Опционально) Настройка очистки бинарных логов через 7 дней
expire_logs_days = 7
# (Опционально) Логирование только изменений (минимальный объем)
binlog_format = ROW
```

log-bin: Включает бинарное логирование и задает путь к файлам бинарного лога.

server-id: Уникальный идентификатор сервера. Обязателен для репликации.

max_binlog_size: Максимальный размер одного файла бинарного лога. Когда этот размер будет достигнут, создается новый файл.

expire_logs_days: Указывает, через сколько дней старые бинарные логи будут автоматически удалены.

binlog_format: Определяет формат бинарного лога. Варианты:

- ROW: Логируются конкретные изменения строк. Рекомендуется для репликации.
- STATEMENT: Логируются SQL-запросы, которые изменяют данные.
- MIXED: Комбинация двух форматов.

Перезапуск MySQL

После внесения изменений в конфигурационный файл необходимо перезапустить MySQL, чтобы настройки вступили в силу.

На Linux:

```
sudo systemctl restart mysql
```

На Windows:

Откройте «Службы» и перезапустите службу MySQL.

Проверка настройки бинарного логирования

Чтобы убедиться, что бинарное логирование успешно включено, выполните следующую команду в MySQL:

```
SHOW VARIABLES LIKE 'log_bin';
```

Ответ должен быть:

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
```

Также можно проверить список активных бинарных логов:

```
SHOW BINARY LOGS;
```

Управление бинарными логами

Для просмотра содержимого бинарных логов используется команда mysqlbinlog. Она выводит изменения, записанные в бинарные логи, в читаемом формате.

Пример:

```
mysqlbinlog /var/log/mysql/mysql-bin.000001
```

Для ручного удаления старых бинарных логов можно использовать команду:

```
PURGE BINARY LOGS TO 'mysql-bin.000010';
```

Или удалить логи, которые старше определенной даты:

```
PURGE BINARY LOGS BEFORE '2024-09-01 00:00:00';
```

Использование бинарных логов для восстановления данных.

Сделайте полную резервную копию базы данных.

В случае сбоя базы данных можно восстановить её из резервной копии, а затем применить бинарные логи для восстановления последних изменений.

Пример:

```
mysqlbinlog /var/log/mysql/mysql-bin.000001 | mysql -u root -p
```

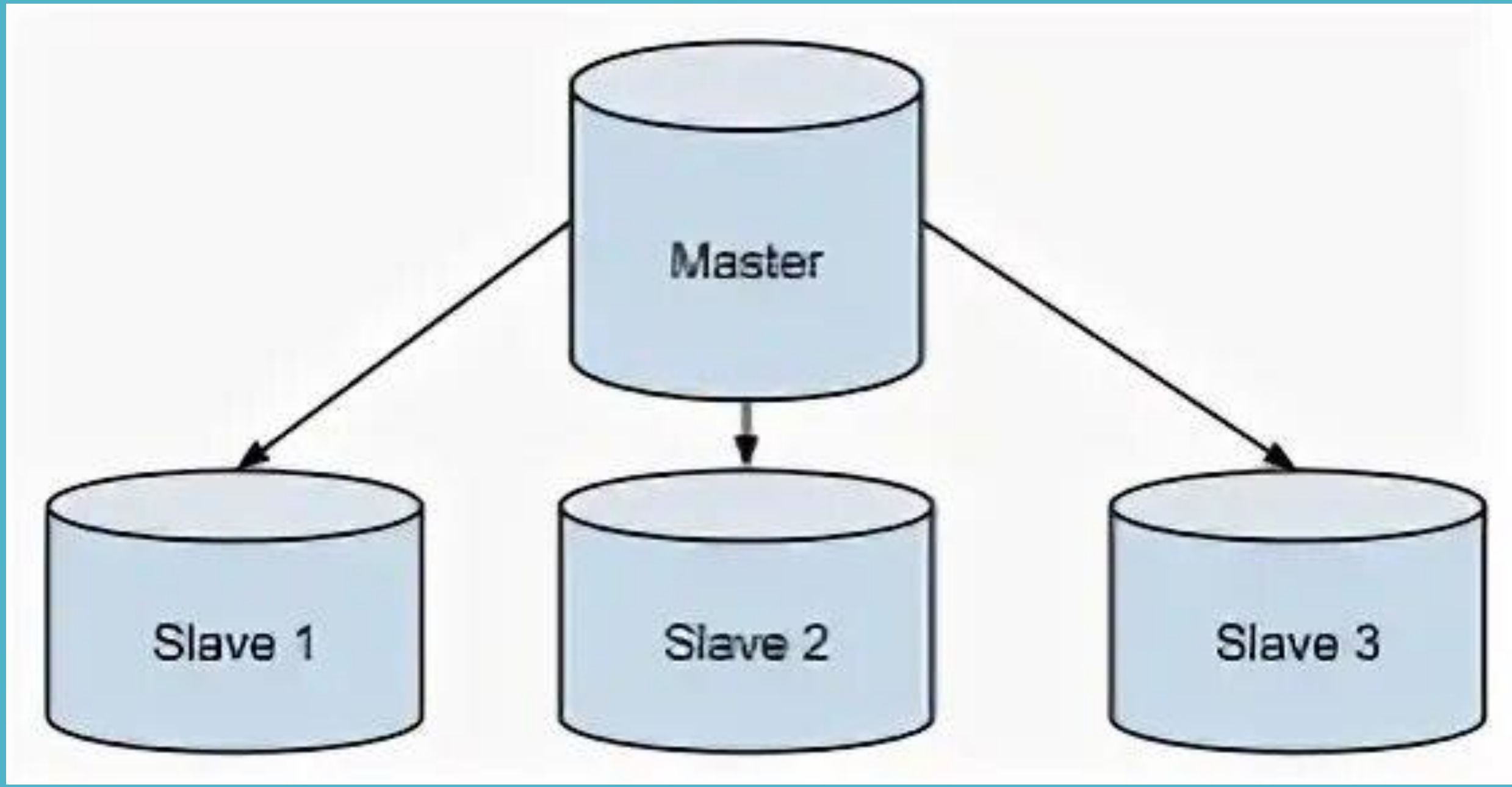
Рекомендации:

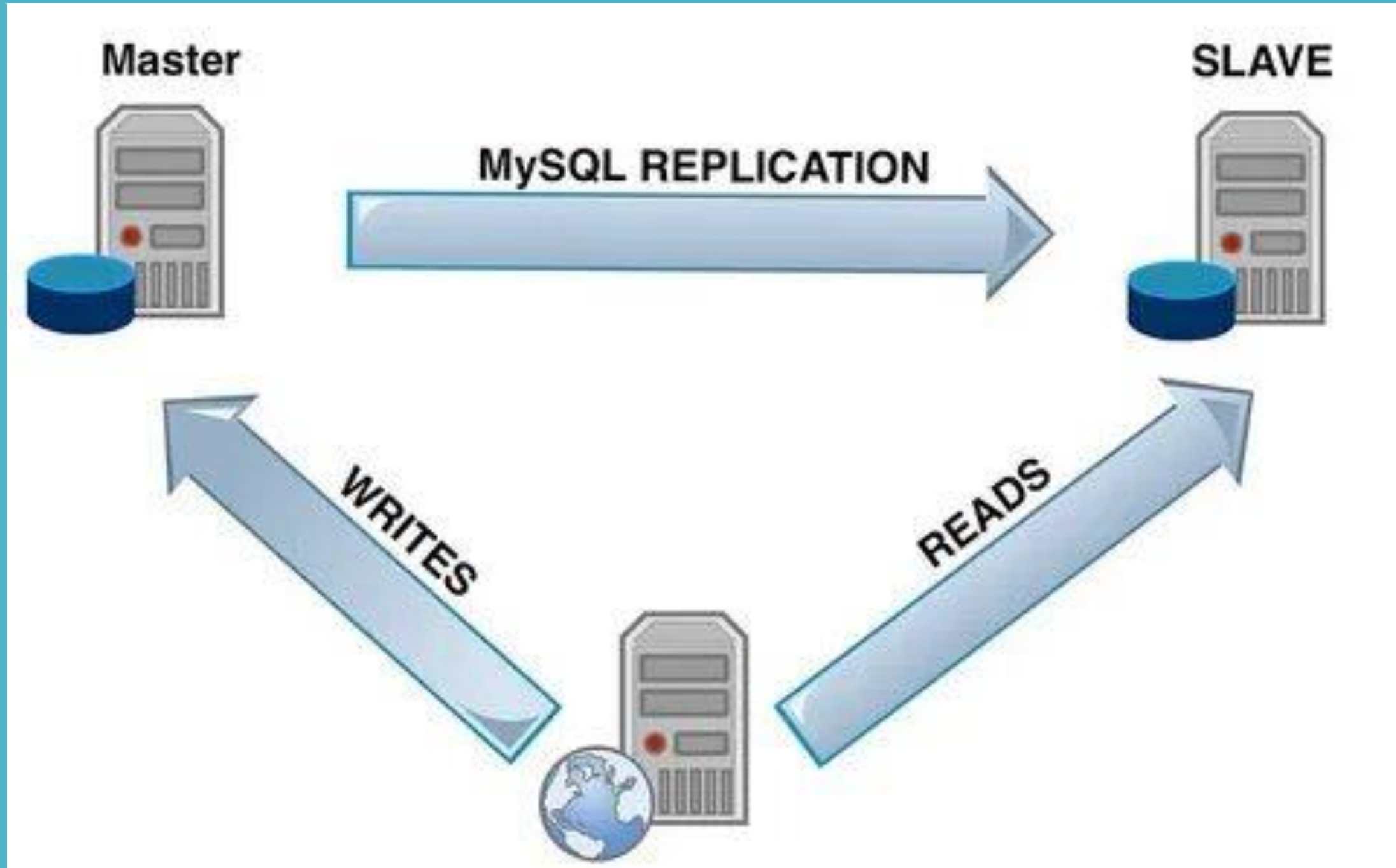
- Регулярно настраивайте автоматическую очистку старых бинарных логов с помощью параметра expire_logs_days, чтобы избежать переполнения диска.
- Для критических систем рекомендуется хранить резервные копии бинарных логов, чтобы иметь возможность восстановления данных на случай потерь.
- Определите оптимальный формат бинарного логирования (ROW предпочтителен для большинства задач, особенно для репликации).

5. Репликация в MySQL.

Репликация в MySQL — это процесс копирования данных с одного MySQL сервера (основного, или Master) на другие сервера (реплики, или Slave).

Этот процесс позволяет синхронизировать несколько баз данных для обеспечения доступности данных, повышения отказоустойчивости и увеличения производительности за счет распределения нагрузки между серверами.


Master
Slave 1
Slave 2
Slave 3


Master
SLAVE
MySQL REPLICATION
WRITES
READS

Основные типы репликации:

Асинхронная репликация:

- Самый распространенный тип репликации. Изменения на основном сервере передаются на реплику с небольшой задержкой.
- Реплика не всегда имеет актуальные данные, так как могут быть задержки в передаче изменений.

Полусинхронная репликация:

- Основной сервер ожидает подтверждения от хотя бы одной реплики, что изменения были записаны, прежде чем завершить транзакцию.
- Обеспечивает более высокий уровень согласованности данных по сравнению с асинхронной репликацией.

Синхронная репликация (Galera Cluster):

- Все изменения моментально синхронизируются между всеми серверами.
- Используется в системах с высокой доступностью, например, в MySQL Cluster.

Как работает репликация:

- Основной сервер (Master) записывает все изменения данных в бинарный лог (binlog).
- Реплика (Slave) подключается к основному серверу и читает бинарный лог, применяя изменения у себя.

В MySQL для репликации изменений используются два потока:

- IO-поток: читает бинарный лог с основного сервера и записывает его в файл релея (relay log) на сервере реплики.
- SQL-поток: применяет изменения из файла релея к базе данных реплики.

Пример настройки асинхронной репликации

Шаг 1. Настройка основного сервера (Master)

Редактирование файла конфигурации my.cnf или my.ini:

Добавьте следующие строки в секцию [mysqld]:

```
[mysqld]
# Включаем бинарное логирование
log-bin = /var/log/mysql/mysql-bin.log
# Уникальный идентификатор сервера
server-id = 1
# (Опционально) Настройка количества дней для хранения бинарных логов
expire_logs_days = 7
```

Перезапустите MySQL для применения изменений:

```
sudo systemctl restart mysql
```

Создание пользователя для репликации: Выполните следующий запрос на основном сервере для создания пользователя, который будет использоваться репликой для подключения:

```
CREATE USER 'repl'@'%' IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
FLUSH PRIVILEGES;
```

Получение позиции бинарного лога: До начала репликации необходимо узнать текущее состояние бинарного лога:

```
SHOW MASTER STATUS;
```

Результат будет примерно таким:

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 | 154      |              |                  |
+------------------+----------+--------------+------------------+
```

Запомните имя файла бинарного лога (**File**) и позицию (**Position**), они понадобятся для настройки реплики.

Шаг 2. Настройка реплики (Slave)

Редактирование файла конфигурации my.cnf или my.ini на сервере реплики:

Добавьте следующие строки в секцию [mysqld]:

```
[mysqld]
# Уникальный идентификатор сервера реплики
server-id = 2
# (Опционально) Отключаем бинарное логирование на реплике, если оно не
log-bin = /var/log/mysql/mysql-bin.log
relay-log = /var/log/mysql/mysql-relay-bin
```

Перезапуск MySQL на реплике:

```
sudo systemctl restart mysql
```

Настройка репликации: Выполните следующие команды на реплике для подключения к основному серверу и начала процесса репликации:

```
CHANGE MASTER TO
    MASTER_HOST = 'IP_адрес_основного_сервера',
    MASTER_USER = 'repl',
    MASTER_PASSWORD = 'password',
    MASTER_LOG_FILE = 'mysql-bin.000001',
    MASTER_LOG_POS = 154;
```

Здесь MASTER_LOG_FILE и MASTER_LOG_POS — это значения, полученные из команды SHOW MASTER STATUS на основном сервере.

Запуск репликации: После настройки подключите реплику к основному серверу:

```
START SLAVE;
```

Проверка статуса репликации: Убедитесь, что репликация работает корректно, выполнив команду:

```
SHOW SLAVE STATUS \G
```

В выводе вы должны обратить внимание на следующие параметры:

- Slave_IO_Running: Yes
- Slave_SQL_Running: Yes

Если оба параметра установлены в Yes, значит репликация работает корректно.

Репликация нескольких серверов

MySQL поддерживает сценарии с несколькими репликами. Один основной сервер может реплицировать данные на множество серверов-реплик. Также возможна репликация "цепочкой", когда один сервер является одновременно репликой и основным сервером для других.

Репликация с фильтрацией данных

MySQL позволяет настроить репликацию с фильтрацией определенных баз данных или таблиц. Для этого можно использовать параметры binlog-do-db и replicate-do-db в файле конфигурации.

Рекомендации по настройке репликации:

- Мониторинг репликации: Регулярно проверяйте статус репликации с помощью SHOW SLAVE STATUS и настройте автоматический мониторинг возможных ошибок.
- Обработка сбоев: Если репликация остановилась из-за ошибок, важно изучить ошибки и решить их. Можно использовать команды STOP SLAVE, START SLAVE и RESET SLAVE.
- Синхронизация: Регулярно проверяйте, чтобы данные на репликах соответствовали данным на основном сервере. Можно использовать инструменты, такие как pt-table-checksum и pt-table-sync, для обнаружения расхождений.

6. Настройка логирования в MySQL.

Логирование в MySQL позволяет отслеживать различные события, действия пользователей, а также ошибки и запросы, что важно для аудита, диагностики проблем и повышения производительности базы данных.

MySQL поддерживает несколько типов логов:

- Общий лог (General Query Log) — записывает все запросы к базе данных.
- Лог медленных запросов (Slow Query Log) — фиксирует запросы, которые выполнялись дольше установленного порога времени.
- Бинарный лог (Binary Log) — хранит все изменения данных (INSERT, UPDATE, DELETE), используется для восстановления данных и репликации.
- Лог ошибок (Error Log) — записывает все ошибки сервера MySQL, а также информацию о его запуске и остановке.

Как включить и настроить различные типы логов

1. Общий лог (General Query Log)

Этот лог фиксирует все SQL-запросы, которые отправляются на сервер MySQL.

Включение общего лога:

В файле конфигурации MySQL (my.cnf или my.ini добавьте или измените следующие строки в секции [mysqld]:

```
[mysqld]
general_log = 1
general_log_file = /var/log/mysql/general.log
```

general_log: Включает общий лог (1 — включено, 0 — выключено).

general_log_file: Указывает путь к файлу общего лога.

Перезапустите MySQL для применения изменений:

Проверка включения общего лога:

Выполните команду в MySQL:

```
SHOW VARIABLES LIKE 'general_log';
```

Если общий лог включен, вы увидите результат:

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| general_log   | ON    |
+---------------+-------+
```

2. Лог медленных запросов (Slow Query Log)

Этот лог записывает запросы, выполнение которых заняло больше времени, чем указано в параметре long_query_time. Лог медленных запросов полезен для оптимизации производительности базы данных.

Для ключение лога в файле конфигурации MySQL (my.cnf или my.ini) добавьте или измените следующие строки:

```
[mysqld]
slow_query_log = 1
slow_query_log_file = /var/log/mysql/slow-query.log
long_query_time = 2
log_queries_not_using_indexes = 1
```

- **slow_query_log**: Включает лог медленных запросов.
- **slow_query_log_file**: Путь к файлу, в котором будут записаны медленные запросы.
- **long_query_time**: Указывает порог времени выполнения запроса (в секундах), после которого запрос будет записан в лог. Например, 2 означает, что все запросы, выполнявшиеся более 2 секунд, будут записаны.
- **log_queries_not_using_indexes**: Записывает в лог запросы, которые не используют индексы (даже если они выполнялись быстро).

Перезапустите MySQL.

Проверка включения лога медленных запросов:

```
SHOW VARIABLES LIKE 'slow_query_log';
```

Если лог медленных запросов включен, результат будет таким:

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| slow_query_log | ON    |
+---------------+-------+
```

3. Бинарный лог (Binary Log)

Этот лог используется для репликации и восстановления базы данных. Он фиксирует все изменения данных (например, INSERT, UPDATE, DELETE).

Для включения бинарного лога, добавьте в файл конфигурации MySQL следующие строки (подробнее настройка бинарного лога была описана выше):

```
[mysqld]
log-bin = /var/log/mysql/mysql-bin.log
server-id = 1
```

После перезапуска MySQL бинарный лог начнет фиксировать все изменения данных.

4. Лог ошибок (Error Log)

Лог ошибок записывает все ошибки, которые произошли при работе сервера, а также информацию о его запуске и остановке. По умолчанию, этот лог включен в MySQL, и его местоположение зависит от операционной системы.

Чтобы изменить расположение файла лога ошибок, добавьте или измените следующие строки в файле конфигурации MySQL:

```
[mysqld]
log_error = /var/log/mysql/error.log
```

После перезапуска MySQL ошибки будут записываться в указанный файл.

5. Другие параметры логирования

log_output: Определяет, куда записываются логи (файл или таблица). По умолчанию логи пишутся в файлы, но можно настроить запись в таблицы MySQL.

Пример настройки:

```
SET GLOBAL log_output = 'TABLE';
```

log_warnings: Уровень детализации логирования предупреждений и ошибок.

Значение 0: Логирование предупреждений отключено.

Значение 1 (по умолчанию): Логируются основные предупреждения и ошибки.

Рекомендации:

Включайте логирование медленных запросов для мониторинга производительности. Это поможет выявить запросы, которые требуют оптимизации.

Ограничивайте размер логов и настраивайте их ротацию, чтобы избежать переполнения диска.

Анализируйте логи ошибок для своевременного обнаружения проблем с базой данных.

Используйте бинарные логи для репликации и восстановления данных после сбоев.

Домашнее задание:

1. Повторить материал лекции.

Список литературы:

1. В. Ю. Кара-ушанов SQL — язык реляционных баз данных
2. А. Б. ГРАДУСОВ. Введение в технологию баз данных
3. А.Мотеев. Уроки MySQL

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com