нетология

MySQL

Сергей
Андрюнин

Сергей Андрюнин

План занятия

1. Историческая справка
2. Текущая версия
3. CAP-теорема
4. Введение в архитектуру
5. Транзакции в MySQL
6. Типы данных
7. Индексация данных
8. Расширяемость
9. Производительность
10. Безопасность
11. Бэкап и восстановление
12. Масштабирование
13. Итоги
14. Домашнее задание

Историческая справка

Историческая справка

В 1994 году выпущена **mSQL** (MiniSQL) - легковесная клиент-серверная реляционная СУБД.

При попытках добавить низкоуровневый драйвер таблиц ISAM к mSQL выяснили, что mSQL является недостаточно быстродейственным и гибким инструментом.

В мае 1995 выпущена первая версия **MySQL**, которая обладала API интерфейсом, аналогичным mSQL, но обладающая большей гибкостью и быстродействием.

На данный момент MySQL является ПО из линейки продуктов компании Oracle.

Историческая справка

В процессе разработки MySQL как Open-source продукта, появилось **несколько “ответвлений кода”.**

Самые известные из них:

- Drizzle (упор на производительность);
- OurDelta (расширение функциональности MySQL);
- Percona (российская разработка, основанная на собственном “движке” XtraDB);
- MariaDB (использование устойчивого к сбоям и производительного “движка” Maria Engine).

CAP-теорема

CAP-теорема

БД MySQL является классической RDBMS системой.

Данные системы являются CA, то есть обеспечивают доступность и согласованность данных.

Но в зависимости от конфигурации, MySQL может являться и CP системой.

Например, кластер может полностью отключаться, при отсутствии указанных узлов.

Введение в архитектуру

Введение в архитектуру

Архитектуру MySQL можно представить в виде 3-х логических уровней:

- Службы обеспечения клиентских соединений
- Обработка, анализ запросов и инструменты подсистемы хранения
- Подсистемы хранения данных

Взято с сайта: storage.piter.com/upload/contents/978544610696/978544610696_p.pdf

Введение в архитектуру

Каждое клиентское соединение внутри серверного процесса выполняется в отдельном потоке.

Сервер хранит потоки в кеше, их не нужно создавать или уничтожать для каждого соединения.

При подключении к серверу - происходит аутенфикация.

Введение в архитектуру

Подсистемы хранения данных являются “движками” MySQL.

Вывести список “движков” можно используя команду:

show engines;

Пример вызова данной команды:

```
mysql> show engines;
| Engine     | Support  | Comment

| MyISAM     | DEFAULT  | Двигатель по умолчанию с MySQL 3.23 с отличной производительностью
| InnoDB     | YES      | Поддерживает транзакции, блокировку на уровне строк и внешние ключи
| BerkeleyDB | DISABLED | Поддерживает транзакции и блокировку на уровне страницы
```

Взято с сайта: ru.bmstu.wiki/InnoDB

Введение в архитектуру

“Движок”	Транзакционность	Активная разработка
Archive	-	+
CSV	-	+
Falcon	+	-
InnoDB	+	+
MyISAM	-	-
NDB	+	+

Введение в архитектуру

Кейс	MyISAM	InnoDB
Полнотекстовый поиск	+	>5.6.4
Транзакционность	-	+
Частые Select запросы	+	-
Частые Insert/Update/Delete запросы	-	+
Мульти-процессинг на одной таблице	-	+

Транзакции в MySQL

Транзакции в MySQL

Транзакция - это группа запросов SQL, обрабатываемых атомарно, как единое целое.

Если БД не может выполнить какой-то запрос из группы, то ни один из запросов не будет выполнен.

При использовании транзакций - на изменяемые данные в таблице накладывается блокировка, запрещающая двум транзакциям изменять данные в одной строке.

Транзакции в MySQL

Базовые типы блокировок:

- shared lock

Позволяет читать, но не позволяет изменять данные и ставить exclusive lock

- exclusive lock

Запрещает другим транзакциям блокировать строку, а также может блокировать ее на запись и на чтение, в зависимости от уровня изоляции

Транзакции в MySQL

MySQL соответствует принципу ACID (при выборе соответствующего “движка”):

- atomicity - атомарность;
- consistency - согласованность;
- isolation - изолированность;
- durability - долговечность.

ACID понижает производительность.

В системах, где транзакционность не нужна - можно выбрать “движок”, исключающий ее и повысить скорость операций БД.

Транзакции в MySQL

Поддерживаемые **уровни изоляции в MySQL**:

- Read uncommited - чтение незафиксированных данных;
- Read committed - чтение зафиксированных данных;
- Repeatable read - повторяемое чтение;
- Serializable - сериализуемость.

Транзакции в MySQL

Чтобы узнать текущий уровень изоляции, используется следующая команда:

SHOW VARIABLES LIKE '%transaction_isolation%';

Для установки уровня изоляции транзакций используется следующая команда:

*SET **[GLOBAL | SESSION]** TRANSACTION ISOLATION LEVEL **level**;*

В зависимости от выбора опциональных параметров **[GLOBAL|SESSION]** изоляции будет применена соответственно:

- *GLOBAL* - во всех последующих сессиях,
- *SESSION* - ко всем последующим транзакциям, выполняемых в текущей сессии.

level - это уровень применяемой транзакции.

Типы данных

Типы данных

Поддерживаемые типы данных указаны в документации на MySQL: dev.mysql.com/doc/refman/8.0/en/data-types.html

Стоит отметить, что дополнительно к стандартным типам данных, принятых в SQL поддерживаются:

- **Spatial Data** (например gis-data)
- **Json Data** (в том числе возможно их частичное обновление)

Индексация данных

Индексация данных

MySQL поддерживает несколько полезных алгоритмов индексации:

- **Индексы в B-деревьях** - такие как INDEX, FULLTEXT, PRIMARY KEY и UNIQUE.
- **Индексы в R-деревьях**, например, индексы для пространственных типов данных.
- **Хэш-индексы и инвертированные списки** при использовании индексов FULLTEXT.

Расширяемость

Расширяемость

MySQL позволяет расширять свой функционал, путем написания плагинов и процедур.

Для обеспечения расширяемости осуществляется поддержка следующих ЯП: C/C++, Delphi, Erlang, Go, Java, Lisp, Node.js, Perl, PHP, R.

Возможные расширения MySQL:

- механизмы хранения;
- полнотекстовые анализаторы;
- различные демоны;
- репликация;
- аудит.

Производительность

Производительность

Производительность MySQL обеспечивается выбором “правильного движка” и тонкой настройкой сервера БД.

В общем виде настройка сервера заключается в изменении параметров конфигурационного файла *my.cnf*

Производительность

Список основных параметров конфигурационного файла настройки MySQL (при использовании InnoDB):

- innodb_buffer_pool_size
- innodb_log_file_size
- innodb_log_buffer_size
- innodb_file_per_table
- innodb_flush_method
- innodb_flush_log_at_trx_commit
- query_cache_size
- max_connections

Производительность

innodb_buffer_pool_size

Размер буффера кеширования данных и индексов.

Обычно устанавливается как 70-80% от всей доступной серверу БД памяти.

Например при серверной ОЗУ 32 Гб - для буффера можно выделить 24 Гб (~ 75%).

Производительность

innodb_log_file_size

Размер файла-лога операций. Данный файл требуется для восстановления работоспособности сервера БД после сбоя.

Файлов логов всегда 2, таким образом - занятое место на диске будет:

*total_disk_space = innodb_log_file_size * 2*

Чем больше выделено пространства для данного файла, тем быстрее будут производится io операции, но тем медленнее будет восстанавливаться сервер БД.

Производительность

innodb_log_buffer_size

Размер буффера, в который помещаются транзакции в незакомиченном состоянии.

После коммита транзакции из буффера попадают в log_file.

В большинстве случаев достаточно эту величину выставлять 1 Mb.

Производительность

innodb_file_per_table

По умолчанию InnoDB сохраняет все таблицы в один файл.

При включении данной опции - таблицы хранятся по разным файлам.

Включение данного параметра требуется в случаях необходимости:

- освобождения места на диске при удалении таблиц (общий файл может только увеличиваться)
- компрессии таблиц для экономии места на диске

Производительность

innodb_flush_method

Данный параметр определяет логику сброса данных на диск.

На текущий момент оптимальные, возможные для установки, значения данной настройки:

- O_DIRECT
- O_DSYNC

O_DSYNC работает быстрее, но O_DIRECT обеспечивает большую надежность процесса записи.

Производительность

innodb_flush_log_at_trx_commit

Данный параметр определяет поведение сброса операций в лог файл на диск.

- innodb_flush_log_at_trx_commit = 1

Сохранность данных важнее скорости IO

- innodb_flush_log_at_trx_commit = 2

Скорость IO важнее сохранности данных

Производительность

query_cache_size

Данный параметр определяет объем памяти, выделенный под кеш запросов.

Является неэффективным и чаще его выставляют 0.

Неправильно выставленные параметр может замедлить сервер БД.

Производительность

max_connections

Данный параметр определяет количество одновременных соединений с сервером БД.

Изменять его следует только в том случае, если вы уверены в нехватке текущего значения

Например в логах сервера БД видите ошибку “*Too many connections*”.

Безопасность

Безопасность

Безопасность MySQL основана **на использовании ACL (access control list)** для всех пользовательских операций.

Также, при необходимости, можно настроить **SSL** на уровне клиент-серверного соединения.

Безопасность

Управление доступом сервером MySQL осуществляется в два шага:

- Сервер принимает или отклоняет соединение, основываясь на Вашей личности и том, может ли проверить Вашу личность, поставляя правильный пароль.
- Предположив, что вы можете соединиться, сервер проверяет каждый запрос, который Вы делаете, чтобы определить, есть ли у Вас достаточные привилегии, чтобы выполнить это.

Безопасность

Систему привилегий MySQL можно разделить на 3 типа:

- **Административные привилегии**

  Позволяют пользователям управлять работой сервера MySQL

- **Привилегии базы данных**

  Относятся к базе данных и всем объектам в ее пределах.

- **Привилегии для объектов базы данных**

  Для конкретных целей в пределах БД, для всех объектов данного типа в пределах БД или глобально для всех объектов данного типа во всех БД.

Бэкап и восстановление

Бэкап и восстановление

Файлы бэкапов в MySQL могут быть использованы как:

- В качестве резервной копии для восстановления данных
- Как источник данных для настройки реплик
- В качестве источника данных для экспериментов:
  - Сделать копию базы данных, которую можно использовать без изменения исходных данных.
  - Чтобы проверить возможные несовместимости обновлений

Бэкап и восстановление

В MySQL для создания бэкапов используется утилита mysqldump.

В общем виде вызов утилиты выглядит следующим образом:

```
shell> mysqldump [arguments] > file_name
```

- arguments - аргументы вызова утилиты
- file_name - имя файла с данными бэкапа.

Взято с сайта: dev.mysql.com

Бэкап и восстановление

Примеры использования утилиты mysqldump:

- для бэкапа всех БД на сервере:

```
shell> mysqldump --all-databases > dump.sql
```

- для бэкапа выбранных БД на сервере:

```
shell> mysqldump --databases db1 db2 db3 > dump.sql
```

Взято с сайта: dev.mysql.com

Бэкап и восстановление

Восстановление из файла-бэкапа производится с помощью бинарного файла mysql:

```
shell> mysql < backup_sunday_1_PM.sql
```

Для восстановления изменений над БД после бэкапа - можно воспользоваться записями из бинарных лог-файлов:

```
shell> mysqlbinlog gbichot2-bin.000007 gbichot2-bin.000008 | mysql
```

Масштабирование

Масштабирование

Масштабирование - это распределение данных для увеличения производительности и отказоустойчивости.

Масштабирование делится на:

- **шардирование** - разделение таблиц на куски по какому-либо принципу.
- **репликацию** - дублирование данных на разных экземплярах СУБД и формирование правил выполнения io операций.

Каждый из видов масштабирования также имеет деление на подтипы.

Эффективная работа с БД достигается путем правильного комбинирования типов шардирования и репликации.

Масштабирование

Шардирование можно разделить на:

- **вертикальное** - разделение таблиц на куски по какому-либо условию в рамках одного экземпляра СУБД.
- **горизонтальное** - разделение таблиц на куски по какому-либо условию в рамках нескольких экземпляров СУБД.

В MySQL шардирование может быть настроено через AutoSharding, либо выполнено на стороне клиентского приложения.

Масштабирование

Репликацию можно разделить следующим образом:

- master-slave репликация;
- multi-master репликация;
- two masters, many slaves репликация.

Режим репликации (синхрон/асинхрон) может изменяться, в зависимости от версии MySQL.

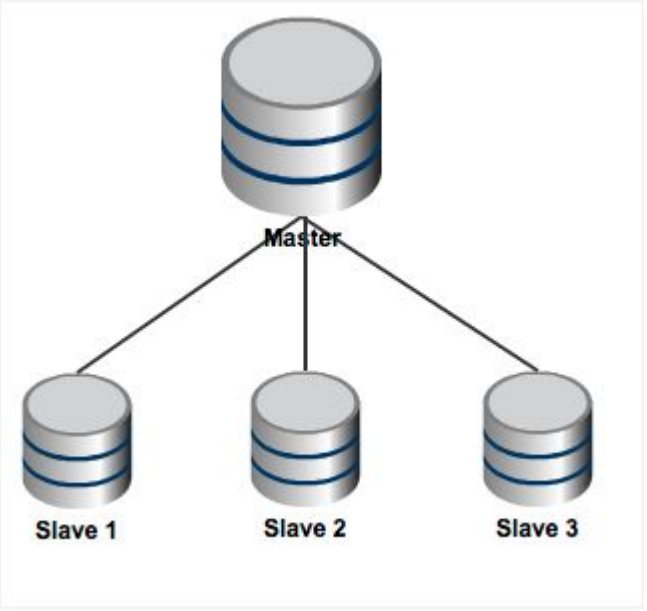
Также возможна репликация на дисках и лог-файлах.

Масштабирование

Master-slave репликация обеспечивает передачу данных на запись с ведущего узла (master) на ведомые узлы (slave).

Ведомые узлы работают в режиме “только для чтения”.

При остановке ведущего узла - все запросы на модификацию данных не будут выполняться



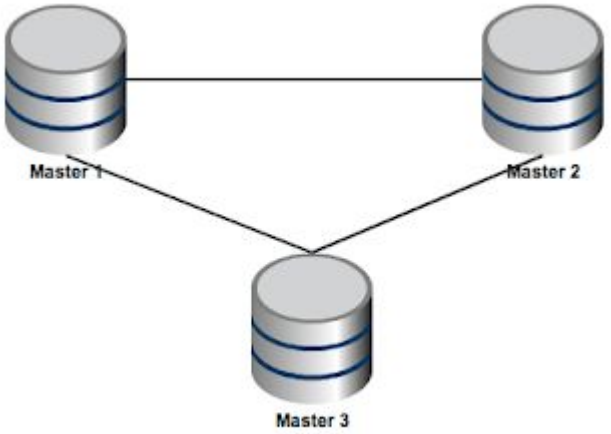
Взято с сайта: www.phphighload.com/

Масштабирование

Multi-master репликация похожа на master-slave репликацию, за исключением наличия нескольких ведущих узлов.

Каждый ведущий узел обрабатывает входящий запрос, затем производит его синхронизацию на других ведущих серверах.

Недостаток данного вида репликации - возможность возникновения конфликтов между ведущими серверами на уровне транзакций.



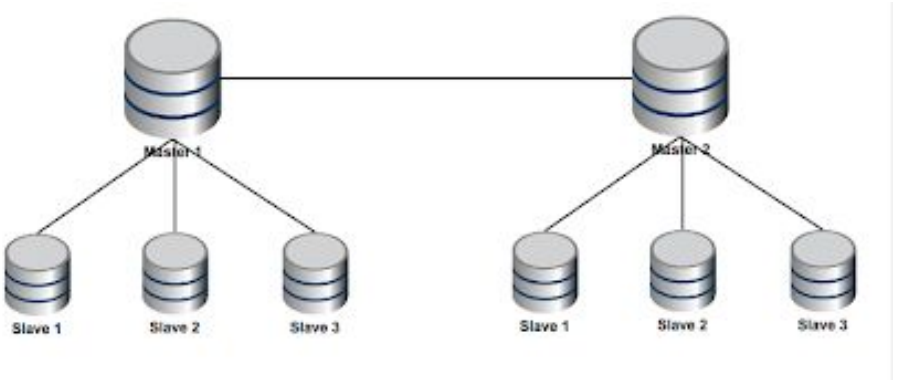
Взято с сайта: www.phphighload.com/

Масштабирование

Two masters, many slaves репликация представляет цепочку из 2x master серверов, которые имеют равные количества slave серверов.

Оба master сервера выполняют запросы на модификацию данных.

При выходе из строя одного master сервера, приложение продолжит работу со вторым.



Взято с сайта: www.phphighload.com/

Итоги

Итоги

MySQL - одна из современных СУБД, позволяющая решать большой набор задач, при этом оставаясь довольно простой в настройке.

В текущей лекции мы узнали, что MySQL:

- По теореме Брюера может быть как CA, так и CP системой
- Имеет множество “движков”, под разные виды задач
- Обеспечивает безопасность системой ACL
- Для восстановления используются файлы бэкапов и бинарные log-файлы
- Позволяет расширять свой функционал
- В обеспечение производительности и отказоустойчивости возможны гибкие настройки шардинга и репликации

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Telegram.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Сергей Андрюнин