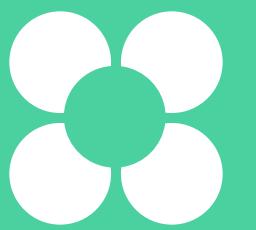


MySQL

Сергей Андрюнин

DevOps-инженер, «Центр биометрических технологий»



План занятия

- 1 Архитектура БД
- 2 Типы данных
- 3 Производительность и масштабирование
- 4 Безопасность



Архитектура БД

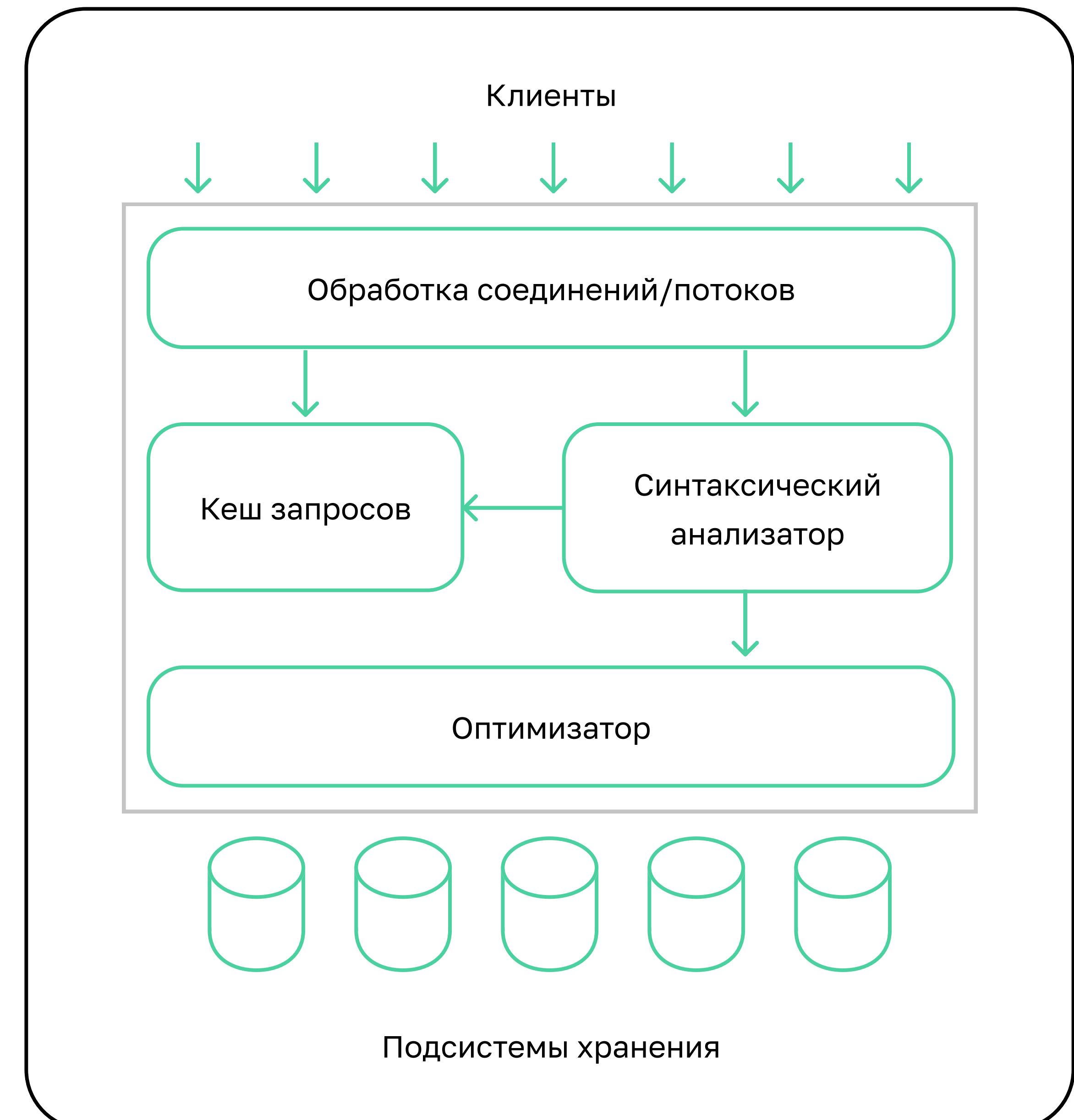
Сергей Андрюнин

DevOps-инженер, «Центр биометрических технологий»

Введение в архитектуру

Архитектуру MySQL
можно представить в виде
3 логических уровней:

- службы обеспечения клиентских соединений
- обработка, анализ запросов и инструменты подсистемы хранения
- подсистемы хранения данных



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 аутентификация

Введение в архитектуру

Подсистемы хранения данных являются «движками» MySQL.

Вывести список «движков» можно, используя команду **show engines**:

Пример вызова 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
```

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	-	+
Мультипроцессинг на одной таблице	-	+

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 иставить **exclusive lock**
- **exclusive lock** – запрещает другим транзакциям блокировать строку, а также может блокировать её 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 снижает производительность.

В системах, где транзакционность не нужна, можно выбрать исключающий её «движок» и повысить скорость операций БД

Транзакции в MySQL

Поддерживаемые уровни изоляции в MySQL

- **Read uncommitted** – чтение незафиксированных данных
- **Read committed** – чтение зафиксированных данных
- **Repeatable read** – повторяемое чтение
- **Serializable** – сериализуемость

Транзакции в MySQL

Чтобы узнать текущий уровень изоляции, используется команда
`SHOW VARIABLES LIKE '%tx_isolation%';`

Для установки уровня изоляции транзакций используется команда/
`SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL;`

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

- GLOBAL – во всех последующих сессиях
- SESSION – ко всем последующим транзакциям, выполняемым в текущей сессии

Level – уровень применяемой транзакции

Производительность и масштабирование

Сергей Андрюнин

DevOps-инженер, «Центр биометрических технологий»

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 буфера кеширования данных и индексов.

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

Например, при серверной ОЗУ 32 ГБ для буфера можно выделить 24 ГБ (~ 75%).

Производительность

`innodb_log_file_size`

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

Файлов-логов всегда 2, таким образом, занятое место на диске будет:
 $\text{total_disk_space} = \text{innodb_log_file_size} * 2$

Чем больше выделено пространства для этого файла, тем быстрее будут производиться IO-операции, но тем медленнее будет восстанавливаться сервер БД

Производительность

`innodb_log_buffer_size`

Размер буфера, в который помещаются транзакции
в незакомиченном состоянии.

После коммита транзакции из буфера попадают в `log_file`.

В большинстве случаев достаточно выставлять эту величину 1 МБ

Производительность

`innodb_file_per_table`

По умолчанию InnoDB сохраняет все таблицы в один файл.

При включении этой опции таблицы хранятся по разным файлам.

Включение этого параметра требуется, когда есть необходимость:

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

Производительность

`innodb_flush_method`

Этот параметр определяет логику сброса данных на диск.

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

→ O_DIRECT

→ O_DSYNC

O_DSYNC работает быстрее, но O_DIRECT обеспечивает большую надёжность процесса записи

Производительность

`innodb_flush_log_at_trx_commit`

Этот 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`

Этот параметр определяет объём памяти, выделенный под кеш запросов.

Он неэффективный, и чаще всего его выставляют 0.

Неправильно выставленный параметр может замедлить сервер БД

Производительность

`max_connections`

Этот параметр определяет количество одновременных соединений с сервером Бд.

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

Например, в логах сервера Бд вы видите ошибку
Too many connections

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 делится на подтипы.

Эффективная работа с БД достигается с помощью 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 Auto-Sharding либо выполнено на стороне клиентского приложения

Масштабирование

Репликацию можно разделить следующим образом:

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

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

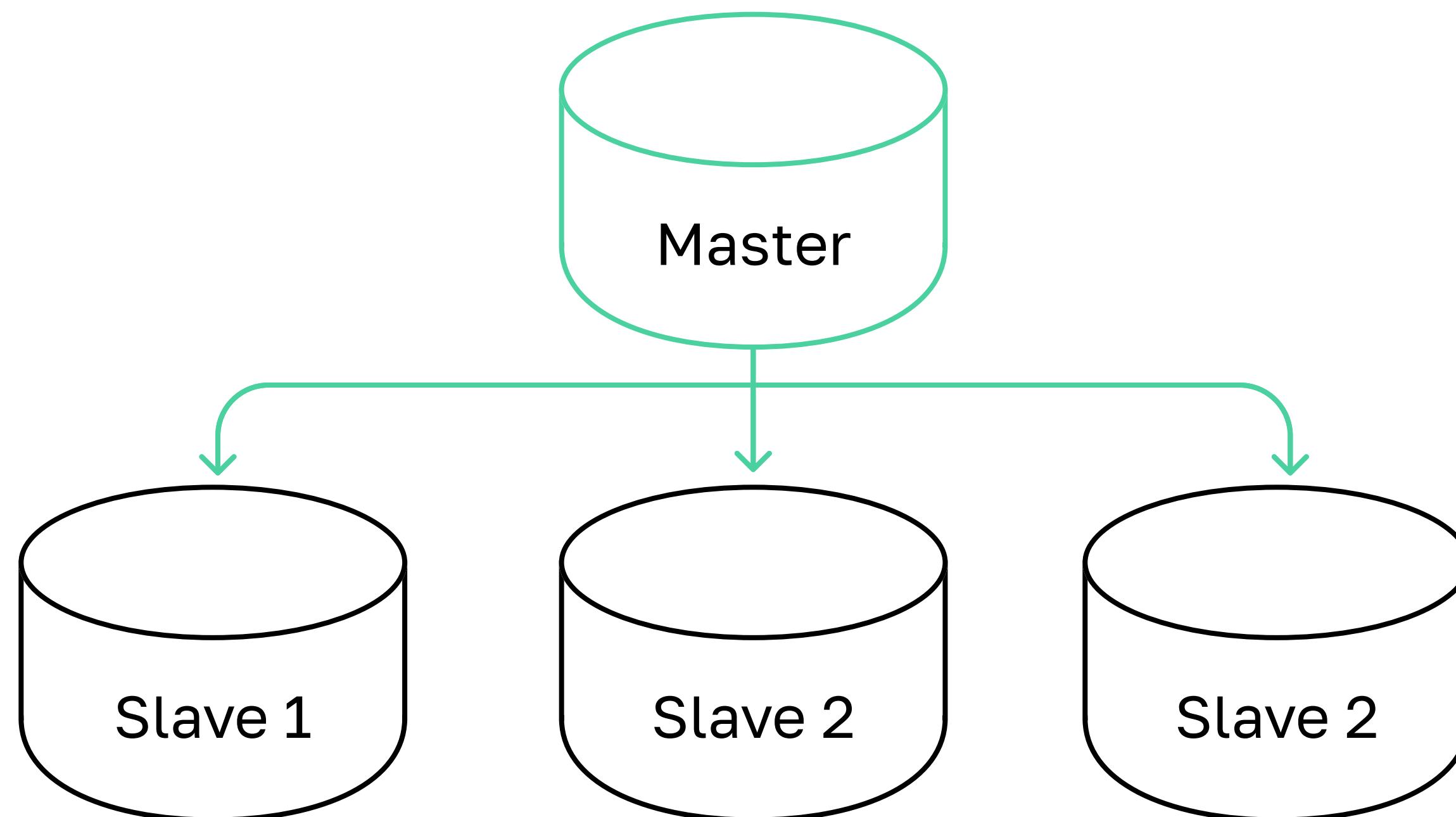
Также возможна репликация на дисках и лог-файлах

Масштабирование

Репликация master-slave 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

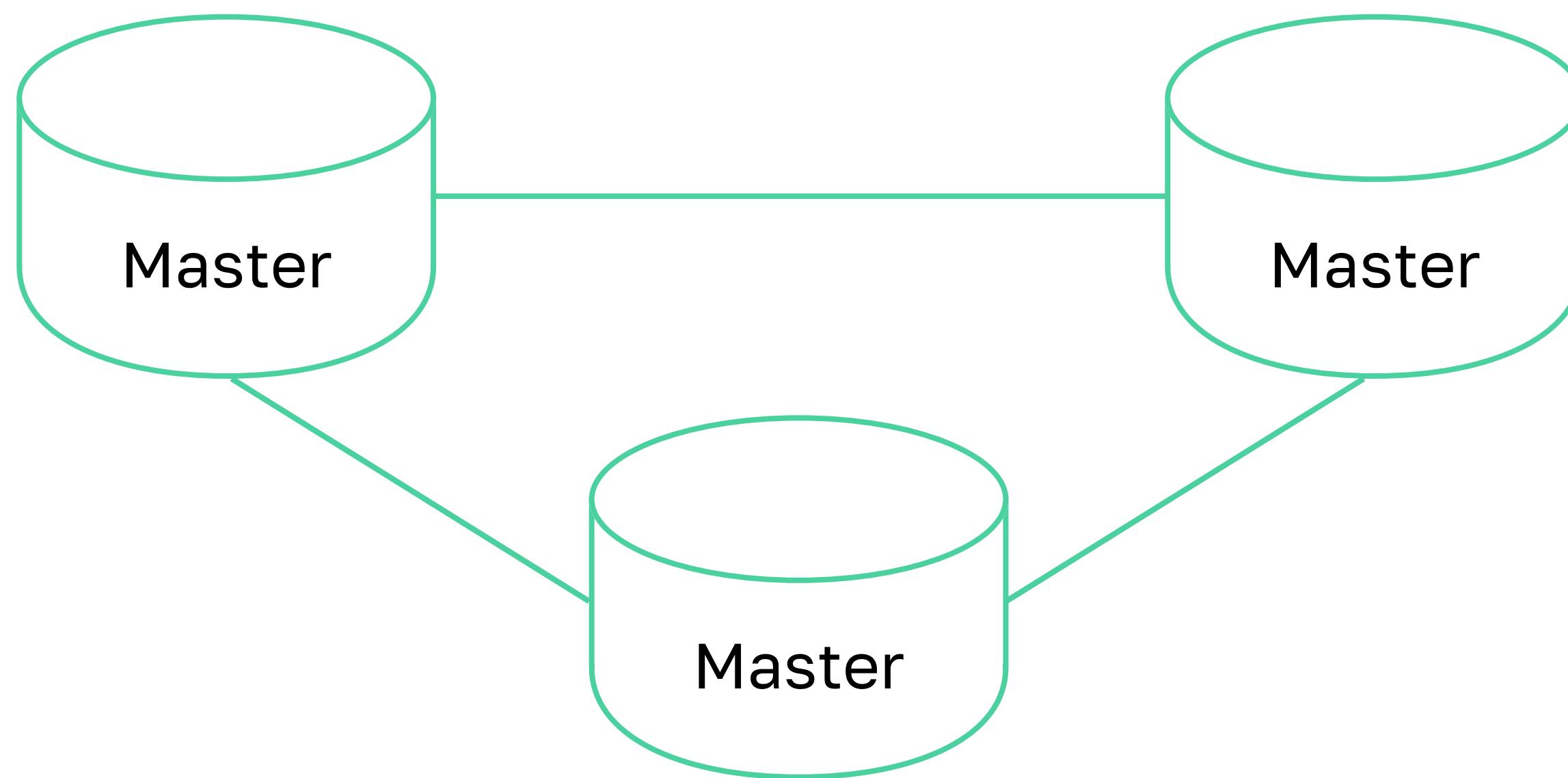


Масштабирование

Репликация multi-master похожа на репликацию master-slave, за исключением наличия нескольких ведущих узлов.

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

Недостаток этого вида репликации в том, что могут возникать конфликты между ведущими серверами на уровне транзакций

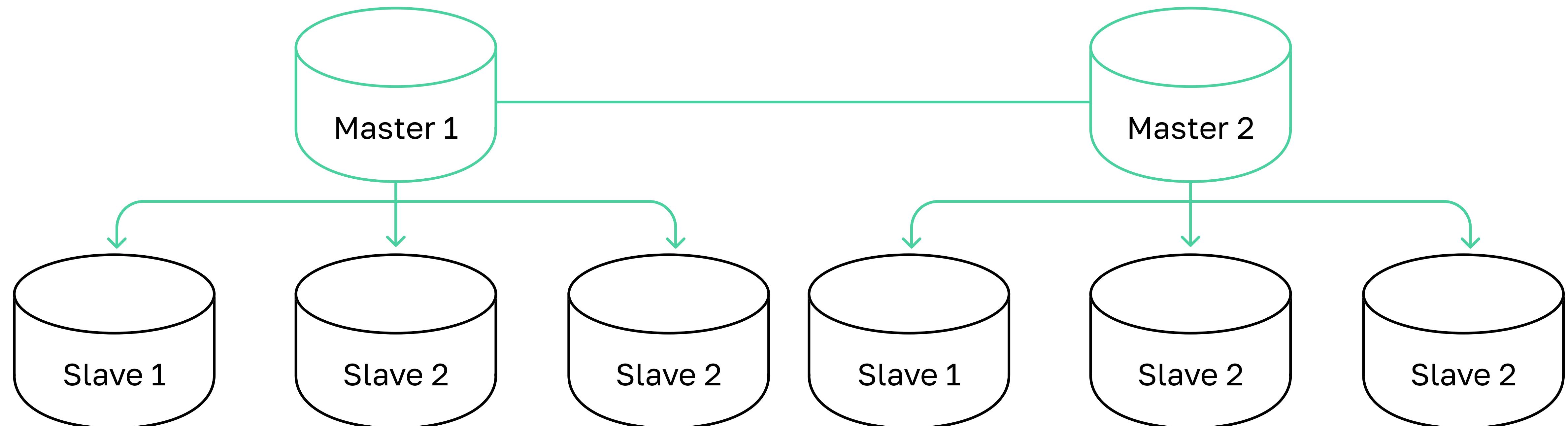


Масштабирование

Репликация two masters, many slaves представляет собой цепочку из двух master-серверов, которые имеют равное количество slave-серверов.

Оба master-сервера выполняют запросы на модификацию данных.

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



Безопасность

Сергей Андрюнин

DevOps-инженер, «Центр биометрических технологий»

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

Сервер MySQL управляет доступом в два шага:

- сервер принимает или отклоняет соединение, основываясь на вашей личности и на 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 её 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

Файлы бэкапов в MySQL могут быть использованы:

- 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
 - проверить возможные несовместимости обновлений

Бэкап и восстановление

В MySQL для создания бэкапов используется утилита `mysqldump`.

В общем виде вызов утилиты выглядит так:

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

- **arguments** – аргументы вызова утилиты
- **file_name** – имя файла с данными бэкапа

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
```

Бэкап и восстановление

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

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

Чтобы восстановить изменения над БД после бэкапа можно воспользоваться записями из бинарных лог-файлов:

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

Расширяемость

MySQL позволяет расширять свой функционал путём написания плагинов и процедур.

Чтобы обеспечить расширяемость, 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