САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 5
Резервное копирование. MySQL

Агафонов Антон Александрович
д.т.н., профессор кафедры ГИиИБ

Самара

План лекции

- Типы резервного копирования
  - По резервируемым данным
  - По способу создания резервных копий
  - По доступности сервера
- Факторы планирования резервного копирования базы данных
- Общие рекомендации по резервированию данных

- Настройка резервного копирования в MySQL

Резервное копирование данных

Резервное копирование базы данных является одной из наиболее важных задач администрирования БД.

Резервное копирование необходимо в следующих случаях:

- Аварийное восстановление данных, необходимое из-за выхода из строя аппаратного обеспечения, физического уничтожения сервера, потери данных из-за несанкционированного доступа и т.д.
- Ошибки пользователей, например, удаление таблицы или данных.
- Аудит: необходимость получения данных или схемы базы данных в некоторый момент времени в прошлом.
- Тестирование: периодическое обновление данных на тестовом сервере с использованием последних рабочих данных позволит проводить тестирование на реальных данных.

Типы резервного копирования. По резервируемым данным

- *Полное резервное копирование*.

  Этот тип резервного копирования создает копии всей базы данных. Это самый очевидный и самый надежный тип резервного копирования базы данных, а также тот, который требует больше всего времени и ресурсов. Поэтому, как правило, полное резервное копирование баз данных выполняется реже всего.

- *Дифференциальное (разностное) резервное копирование*.

  При таком типе резервного копирования выполняется резервное копирование только тех данных, которые изменились в базе данных с момента последнего **полного** резервного копирования. Этот способ гораздо быстрее, чем полное резервное копирование (конечно если запланировано регулярное полное резервное копирование и размер дифференциальных резервных копий не приближается к размеру полных резервных копий), но сопряжен с дополнительным риском того, что произойдет что-то непредвиденное, что приведет к невозможности восстановления базы данных.

Типы резервного копирования. По резервируемым данным

- *Инкрементное резервное копирование*.

  Инкрементное резервное копирование аналогично дифференциальному, за исключением того, что в качестве момента времени, к которому относятся измененные данные, будет использоваться дата последней резервной копии, инкрементной или полной. Таким образом, при восстановлении из инкрементной резервной копии может потребоваться восстановить последнюю полную резервную копию, а также одну или несколько инкрементных резервных копий, чтобы добраться до текущего момента.

  Для СУБД, которые поддерживают журналы транзакций, можно создавать инкрементные резервные копии журналов и использовать их для восстановления базы данных. В журнале транзакций фиксируются все транзакции и производимые ими в базе изменения, и его можно использовать для восстановления базы данных на определенный момент времени. При выполнении резервной копии журнала транзакций происходит его усечение, что является необходимым для всех БД с активированным журналом транзакций. В противном случае, за счет журнала транзакций, размер базы данных будет довольно быстро расти, вплоть до завершения места на диске. В журнале транзакций, если он активирован, фиксируются все транзакции с момента его последнего **усечения**.

Типы резервного копирования. По способу создания

- *Физические резервные копии*.

  Физическая резервная копия — это копия исходных двоичных данных, часто создаваемая на уровне операционной системы. Любой метод резервного копирования, включающий копирование данных без использования построчного доступа к БД, считается методом физического резервного копирования. Такой тип резервного копирования подходит для любых баз данных, но рекомендуем для критически важных баз, которые необходимо быстро восстанавливать после сбоев.

- *Логические резервные копии*.

  Логическая резервная копия — это набор команд SQL, содержащий логическую структуру базы данных (запросы DDL) и ее содержимое (запросы INSERT). Резервное копирование выполняется посредством сканирования таблицы с прохождением каждой строки данных. Такой тип резервного копирования подходит для небольших баз данных, не требовательных к скорости восстановления.

Физическое резервное копирование

Достоинства:

- Время создания физических резервных копий ниже, чем логических, т.к. в процессе создания резервной копии происходит только запись данных содержащихся в файлах БД, в то время как при создании логической копии происходит преобразование физических данных к их логической структуре и запись соответствующих команд SQL в файл.
- Время восстановления физических резервных копий также ниже, чем логических, т.к. фактически происходит перезапись файлов базы данных данными из копии, в то время как при восстановлении логической копии происходит последовательное исполнение SQL-скрипта для создания объектов БД и вставки данных.

Недостатки:

- Для выполнения физической резервной копии сервер, как правило, должен быть остановлен.
- Объем физических копий, как правило, больше.
- Физические резервные копии обладают низкой степенью переносимости, как правило базы данных из этих копий могут быть восстановлены только на сервере с аналогичной архитектурой, под управлением той же ОС и в той же версии СУБД.

Логическое резервное копирование

Достоинства:

- Логическое резервное копирование не требует остановки сервера.
- Логические копии могут быть относительно легко перенесены на компьютер/БД с другой архитектурой системы.

Недостатки:

- Время создания логической резервной копии существенно выше, чем физической.
- Время восстановления из логической резервной копии также выше, т.е. требуется загружать и интерпретировать команды добавления данных и перестраивать индексы.
- Выполнения скрипта восстановления из логической резервной копии может завершиться с ошибкой, например, из-за проблем с кодировкой.

Типы резервного копирования. По доступности сервера

- *Автономное (холодное) резервное копирование*.

  Автономным резервным копированием называют процесс создания резервной копии на остановленном сервере, т.е. на момент резервного копирования работа с БД не ведется, файлы БД не изменяются, и все данные находятся в согласованном состоянии. Благодаря этому можно быстро скопировать файлы, не беспокоясь о сохранении состояния на текущий момент, пока другие процессы читают и записывают данные. Такие виды резервного копирования проще в реализации, но требуют остановки сервера, что часто невозможно.

- *Оперативное (горячее) резервное копирование*.

  Оперативным резервным копированием называют процесс создания резервной копии, не требующий остановки сервера, т.е. на момент резервного копирования пользователи продолжают работу с БД. Кроме того, при оперативном резервном копировании возрастает нагрузка на сервер. Такой вид резервного копирования сложнее, т.к. после копирования файлов данных требуется еще и анализ журнала транзакций, но является единственным доступным видом резервирования для критических систем.

1) *Показатели RTO и RPO.*
   a) RPO (Recovery Point Objective) – это максимальный период времени, за который могут быть потеряны данные в результате инцидента. Допустим имеется информационная система и показатель RPO для нее определен в 1 час. Это значит, что при аварии мы готовы к тому, что после восстановления системы допускается потеря данных не более, чем за один последний час. При определенном везении количество потерянных данных может быть и меньше, но ни при каких условиях не более 1 часа. Этот показатель говорит о том, как часто необходимо выполнять резервирование данных, и какие технологии применять, чтобы выполнить этот показатель.
   b) RTO (Recovery Time Objective) – это промежуток времени, в течение которого система может оставаться недоступной в случае аварии. Необходимо планировать резервирование и восстановление данных так, чтобы за указанный промежуток времени восстановить работоспособность информационной системы на резервном оборудовании или площадке.

2) *Возможность тестирования резервных копий.*

При минимальной возможности тестирования резервных копий в первую очередь следует рассматривать стратегию полного резервного копирования БД по причине его максимальной надежности. Однако, если имеется возможность детального тестирования дифференциальных резервных копий или резервных копий журнала транзакций, можно запланировать преимущественное выполнение таких типов резервирования, а полное резервное копирование выполнять нечасто, например, раз в неделю или в месяц.

3) *Характеристики базы данных.*

Объем хранимых данных и режим их использования также влияют на планирование стратегии резервного копирования. Если данные меняются достаточно редко, можно обойтись только полным резервным копированием, которое выполняется достаточно часто. При высокой интенсивности изменения данных желательно подключать периодическое разностное копирование или копирование журнала транзакций.

4) *Ограничения на ресурсы.*

Ограничения на ресурсы, такие как оборудование, персонал, объем хранилища резервных копий и его физическая безопасность также будут влиять на стратегии резервного копирования.

Стратегии резервного копирования

- *«Простая» стратегия*.

  Самой простой является стратегия ежедневного полного резервного копирования, обычно ночью. При этом обычно хранятся все резервные копии за последнюю неделю или две, а также резервные копии на начало месяца. Такая стратегия максимально проста в реализации, не требует тестовых серверов для проверки восстановления, и имеет хороший показатель RTO. Несмотря на простоту и надежность такая стратегия может быть рекомендована только для БД с низкой интенсивностью изменения данных.

- *«Сбалансированная» стратегия*.

  Сбалансированная стратегия резервного копирования использует все три вида резервного копирования. Эта стратегия заключается в том, что полное резервное копирование выполняется раз в неделю (например, в воскресенье), разностное резервное копирование выполняется каждую ночь, и несколько раз в день выполняется резервное копирование журналов транзакций. В отличии от «простой» стратегии, реализация этой стратегии требует тестирования резервных копий, т.к. использует потенциально ненадёжные типы резервного копирования. Также «сбалансированная» стратегия несколько хуже по показателю RTO, но может быть значительно лучше по показателю RPO, который меньше и зависит от частоты резервных копий журнала транзакций в течение рабочего дня.

Общие рекомендации по резервированию данных

- *Хранение резервных копий*.

  Резервные копии не следует хранить на том же физическом хранилище, что и файлы исходной базы данных. В таком случае, если возникнет какая-либо проблема с диском, будут потеряны и база данных, и ее резервная копия. Также стоит помнить, что физические устройства, на которых хранятся резервные копии, также подвержены износу. Поэтому рекомендуется хранить несколько резервных копий на разных физических дисках или в хранилищах с зеркалированием данных. При организации хранения резервных копий также стоит помнить о том, что резервная копия — это абсолютно та же база данных, для которой необходимо учитывать вопросы конфиденциальности данных, и уделить надлежащее внимание безопасности хранилища.

- *Автоматизация процессов резервирования*.

  Автоматизацию процессов резервного копирования необходимо настроить по заданному расписанию. В этом случае исключается человеческий фактор и можно быть уверенным, что все резервные копии, необходимые для восстановления в случае инцидента, сделаны и актуальны. Также необходимо проверять, что запланированные задачи по созданию резервных копий успешно выполняются.

Общие рекомендации по резервированию данных

- *Тестирование резервных копий*.

  Рекомендуется использовать тестовый сервер для проверки процедуры восстановления резервных копий. Даже если процесс резервного копирования завершился успешно, есть шанс, что восстановление данных из этой резервной копии невозможно. Единственный способ убедиться в корректности резервной копии – это выполнить процесс восстановления в реальном сценарии, на тестовом сервере. Это минимизирует вероятность ошибки восстановления в случае инцидента. Кроме того, проведение восстановления на тестовом сервере позволяет оценить показатель RTO и, соответственно, адекватность стратегии резервирования данных в целом.

- *Резервирование системных БД*.

  Как правило, в любой СУБД критические данные хранятся в системных базах данных (в MySQL это БД `mysql`, в PostgreSQL – БД `postgres`). В системных базах данных хранятся параметры конфигурации сервера, параметры конфигурации отдельных баз данных, права доступа к объектам и т.д. В связи с этим рекомендуется резервировать не только пользовательские БД, но и служебные, т.к. при возникновении инцидента, если повреждены системные базы данных, невозможно будет восстановить доступ к данным в разумное время, не имея их резервных копий.

Общие рекомендации по резервированию данных

- *Внеплановое резервирование данных*.

  Не всегда риск прекращения нормального функционирования базы данных и клиентского ПО сопряжен с какими-либо инцидентами. Действия по изменению структуры базы данных, массовое изменение данных, операции массового импорта данных из сторонних источников, ввод новой функциональности в клиентское ПО и т.д. – всё это может привести к несогласованности данных и, соответственно, к невозможности штатного функционирования БД и клиентских приложений. В связи с чем рекомендуется выполнять внеплановое полное резервное копирование данных в указанных случаях.

- *Использование встроенных методов верификации резервных копий*.

  Рекомендуется использовать все предоставляемые СУБД доступные параметры верификации при выполнении резервного копирования. Это позволит быть уверенным в том, что все создаваемые резервные копии будут созданы корректно и последовательны с точки зрения транзакций.

Резервное копирование в MySQL

- *Создание горячей резервной копии с помощью MySQL Enterprise Backup*.

  Коммерческая версия MySQL Enterprise Backup реализует физическое копирование экземпляров сервера, выбранных баз данных или таблиц. Поддерживает горячее резервное копирование, в т.ч. инкрементное, шифрование данных алгоритмом AES-256, компрессию резервных копий и другие функции, свойственные подсистемам резервного копирования современных СУБД. Физическое резервное копирование выполняется значительно быстрее, чем логическое копирование с использование команды `mysqldump`.

- *Создание логических резервных копий с использованием* `mysqldump` .

  Позволяет выполнить оперативное логическое резервное копирование, не блокируя таблицы.

- *Создание резервных копий путем копирования файлов таблиц*.

  Свободная версия СУБД MySQL предлагает только один способ полного физического резервного копирования: копирование файлов базы данных средствами ОС. Файловая организация базы данных MySQL (для движка InnoDB) подразумевает следующий набор файлов: файлы данных (ibdata* и. ibd), лог-файлы (ib_logfile*), файл конфигурации (my.cnf). Для копирования указанных файлов требуется остановить сервер.

- *Создание резервных копий в виде текстовых файлов с разделителями*.

  Для создания текстового файла, содержащего данные таблицы, можно использовать команду:

  ```
  SELECT * INTO OUTFILE 'file_name' FROM table_name;
  ```

  Другой способ создания текстовых файлов данных – использование утилиты `mysqldump` с параметром `--tab`

- *Создание инкрементных резервных копий с использованием бинарного лог-файла (двоичного журнала транзакций)*.

  Бинарный лог-файл содержит «события», которые описывают изменения базы данных, такие как операции создания таблицы или изменения данных таблицы.

- *Создание резервных копий с использованием реплик*.

  Если возникают проблемы с производительностью исходного сервера при создании резервных копий, одна из стратегий, которая может помочь — настроить репликацию и выполнять резервное копирование на реплике, а не на источнике.

Логическое резервное копирование в MySQL

Для создания логической резервной копии базы данных в комплекте с MySQL Server поставляется утилита `mysqldump`, которая формирует последовательность операторов SQL, при выполнении которых восстанавливаются исходные определения объектов базы данных и данных таблиц.

Утилита поддерживает два типа выходных данных:

- Без параметра `--tab` операторы SQL записываются в стандартный выходной поток. Выходные данные состоят из операторов `CREATE` для создания объектов (баз данных, таблиц, хранимых процедур и т.д.) и операторов `INSERT` для загрузки данных в таблицы. Вывод можно сохранить в файле и загрузить с помощью `mysql` для воссоздания выгруженных объектов.
- С параметром `--tab` `mysqldump` создает два выходных файла для каждой выгруженной таблицы. Сервер записывает один файл в виде текста с разделителями табуляции, по одной строке на строку таблицы. Этот файл называется `tbl_name.txt` в выходном каталоге. Оператор `CREATE TABLE` записывается в файл с именем `tbl_name.sql` в выходном каталоге.

Логическое резервное копирование в MySQL

По умолчанию `mysqldump` записывает результат в стандартный выходной поток. Результат можно сохранить в файл следующим образом:

```
$> mysqldump [arguments] > file_name
```

Утилита может быть использована для создания резервной копии всех БД сервера или заданных БД:

```
$> mysqldump --all-databases > dump.sql
$> mysqldump --databases db1 db2 db3 > dump.sql
```

В случае создания резервной копии одной БД можно опустить опцию `--databases` :

```
$> mysqldump test > dump.sql
```

В этом случае дамп не содержит инструкций CREATE DATABASE или USE. В этом случае:

- При загрузке копии необходимо указать имя базы данных по умолчанию.
- При загрузке копии можно указать имя базы данных, отличное от исходного имени, что позволит повторно загрузить данные в другую БД.
- БД для загрузки данных должна быть предварительно создана.

Логическое резервное копирование в MySQL

Основные параметры утилиты `mysqldump` :

- `--host` – адрес MySQL сервера;
- `--port` – порт MySQL сервера;
- `--user` – имя пользователя MySQL сервера;
- `--password` – пароль пользователя MySQL сервера;
- `--routines` – включать в резервную копию хранимые процедуры и функции;
- `--single-transaction` – создавать резервную копии в рамках одной транзакции;
- `--source-data` – включить в резервную копию позицию и имя бинарного лог-файла сервера на момент создания копии;
- `--flush-logs` – создать новый бинарный лог-файл перед созданием резервной копии.

Логическое резервное копирование в MySQL

Для восстановления БД из резервной копии необходимо использовать файл с дампом в качестве входных данных для клиента `mysql`. Если файл дампа был создан с опцией `--all-databases` или `--databases`, он содержит операторы CREATE DATABASE и USE, и нет необходимости указывать базу данных по умолчанию, в которую следует загружать данные:

```
$> mysql --user root --password < dump.sql
```

Из `mysql` то же действие может быть выполнено с использованием команды `source`:

```
mysql> source dump.sql
```

Если файл представляет собой дамп одной базы данных, не содержащий операторов CREATE DATABASE и USE, сначала необходимо создать базу данных, затем указать ее при импорте данных:

```
$> mysql db1 < dump.sql
```

Инкрементное резервное копирование в MySQL

В СУБД MySQL поддерживается инкрементное резервное копирование, которое основано на копировании бинарных лог-файлов (журналов). Журнал содержит события, которые описывают изменения в БД (например, создание таблиц, изменения в данных таблицы и т.д.) и представляет собой упорядоченную последовательность одноименных файлов (обычно это server_name-bin) с возрастающим числовым расширением (SERV-bin.000001, SERV-bin.000002, SERV-bin.000003…). Запись событий ведется в текущий журнал – последний файл в последовательности.

Новый файл двоичного журнала создается сервером автоматически при каждом перезапуске сервера, по достижении текущим журналом предельного размера или при выполнении команды сброса журнала FLUSH LOGS.

Инкрементное резервное копирование состоит из следующих шагов:

- Сбросить журнал с помощью команды FLUSH LOGS;
- Скопировать все файлы журналов, начиная с момента последнего полного или инкрементного резервного копирования, в хранилище резервных копий.

Инкрементное резервное копирование в MySQL

Для восстановления инкрементной резервной копии используется утилита mysqlbinlog, которая используется для обработки журналов:

```
$> mysqlbinlog [arguments] log_file
```

Для просмотра журнала в консоли или записи содержимого в текстовый файл можно выполнить следующие команды:

```
$> mysqlbinlog AGAFONOV-bin.000024
$> mysqlbinlog AGAFONOV-bin.000024 > incremental.log
```

Сохраненный файл можно использовать для восстановления:

```
$> mysql --user root --password < incremental.log
```

Можно выполнить восстановление без использования промежуточного файла. Если необходимо восстановить из нескольких файлов журнала, лучше это делать одной командой:

```
$> mysqlbinlog AGAFONOV-bin.000024 AGAFONOV-bin.000025 | mysql -u root -p
```

Можно также выполнить восстановление на конкретный момент времени:

```
$> mysqlbinlog --stop-datetime="2022-11-01 9:59:59" AGAFONOV-bin.000024 | mysql
-u root -p
```

Пример резервного копирования в MySQL

Создание полной резервной копии со сбросом журнала

```
$> mysqldump -u root -p --flush-logs sakila > sakila_full.sql
```

```
-- Текущий лог-файл для записи AGAFONOV-bin.000026
```

Внесение изменения в данные и сброс журнала

```
CREATE TABLE test (id INT AUTO_INCREMENT, name VARCHAR(50), value FLOAT,
PRIMARY KEY (id) );
INSERT INTO test(name, value) VALUES ('Ivanov', 3.34), ('Petrov', 4.31);
FLUSH LOGS;
```

```
-- Текущий лог-файл для записи AGAFONOV-bin.000027
```

Пример резервного копирования в MySQL

«Случайное» изменение данных

```
DROP TABLE test;
UPDATE actor SET last_name = '';

Query OK, 200 rows affected (0.02 sec)
```

Восстановление данных

```
$> mysql --user root --password sakila < sakila_full.sql
$> mysqlbinlog AGAFONOV-bin.000026 | mysql -u root -p
```

Проверка данных

```
SELECT COUNT(*) FROM test;
2
SELECT COUNT(*) FROM actor WHERE last_name != '';
200
```

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
д.т.н., профессор кафедры ГИиИБ