САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 6
Резервное копирование. PostgreSQL. MongoDB

Агафонов Антон Александрович
к.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- Настройка резервного копирования в PostgreSQL:
  - логическое копирование;
  - физическое копирование на уровне файлов;
  - непрерывное архивирование.

- Настройка резервного копирования в MongoDB:
  - физическое копирование;
  - логическое копирование.

Физическое и логическое резервное копирование

- *Физическая резервная копия* — это копия исходных двоичных данных (файлов базы 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.

- *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.

Резервное копирование в PostgreSQL

Существует три разных подхода к резервному копированию данных в PostgreSQL:

- Логическое копирование - выгрузка в SQL.
- Физическое копирование на уровне файлов.
- Непрерывное архивирование.

PostgreSQL. Логическое резервное копирование

Логическое резервное копирование в СУБД PostgreSQL выполняется с помощью утилиты pg_dump. Эта программа генерирует файл с командами SQL, которые при выполнении на сервере пересоздадут объекты базы данных в том же самом состоянии, в котором они были на момент выгрузки. Результат работы программы записывается в стандартный поток вывода. Резервное копирование выполняется в оперативном режиме.

pg_dump [***параметр-подключения***...] [***параметр***...] [***имя_бд***]

Параметры:

- `--host, -h` – адрес сервера;
- `--port, -p` – порт сервера;
- `--username, -U` – имя пользователя, под которым производится подключение;
- `--password, -W` – принудительно запрашивать пароль перед подключением к БД;
- `--format, -F` – (plain, custom, directory, tar) – формат вывода копии.

```
$> pg_dump имя_базы > имя_файла
$> pg_dump -U postgres pagila > pagila.sql
```

PostgreSQL. Логическое резервное копирование

Текстовые файлы, созданные pg_dump, используются для последующего чтения программой psql. Общий вид команды для восстановления дампа:

```
$> psql имя_базы < файл_дампа
```

База данных, заданная параметром *имя_базы* должна быть создана перед запуском psql.

Дампы, выгруженные не в текстовом формате, восстанавливаются утилитой pg_restore.

```
$> pg_restore -d имя_базы файл_дампа
```

По умолчанию, если происходит ошибка SQL, программа psql продолжает выполнение. Если запустить psql с установленной переменной ON_ERROR_STOP, работа psql завершится в случае возникновения ошибки:

```
$> psql --set ON_ERROR_STOP=on имя_базы < файл_дампа
```

Для восстановления БД в рамках одной транзакции можно использовать опцию -1 или --single-transaction .

Можно скопировать базу данных непосредственно с одного сервера на другой, например:

```
$> pg_dump -h host1 имя_базы | psql -h host2 имя_базы
```

PostgreSQL. Логическое резервное копирование

Программа pg_dump выгружает только одну базу данных в один момент времени и не включает в дамп информацию о ролях и табличных пространствах (так как это информация уровня кластера, а не самой базы данных).

Для удобства создания дампа всего содержимого кластера баз данных предоставляется программа pg_dumpall, которая делает резервную копию всех баз данных кластера, а также сохраняет данные уровня кластера, такие как роли и определения табличных пространств.

```
$> pg_dumpall > файл_дампа
```

Полученную копию можно восстановить с помощью psql:

```
$> psql -f файл_дампа postgres
```

Для резервного копирования больших баз данных можно использовать сжатые дампы, разбивать выводимые данные на небольшие файлы средствами Unix или использовать специальный формат дампа pg_dump.

PostgreSQL. Резервное копирование на уровне файлов

Альтернативной стратегией резервного копирования является непосредственное копирование файлов, в которых PostgreSQL хранит содержимое базы данных.

Недостатки такого метода:

1. Чтобы полученная резервная копия была работоспособной, сервер баз данных должен быть остановлен. Запрещение всех подключений к серверу не даст необходимого результата из-за того, что `tar` и подобные средства не получают мгновенный снимок состояния файловой системы, кроме того, в сервере есть внутренние буферы.
2. Копирование и восстановление отдельных таблиц или баз данных в соответствующих файлах или каталогах также невозможно, потому что информацию, содержащуюся в этих файлах, нельзя использовать без файлов журналов транзакций, которые содержат состояние всех транзакций. Без этих данных файлы таблиц непригодны к использованию. Таким образом, копирование на уровне файловой системы будет работать, только если выполняется полное копирование и восстановление всего кластера баз данных.

Ещё один подход к резервному копированию файловой системы заключается в создании «целостного снимка» каталога с данными, если это поддерживает файловая система.

PostgreSQL. Непрерывное архивирование

Другое возможное решение для создания физической резервной копии — использование механизма непрерывного архивирования и восстановление на момент времени (Point-in-Time Recovery, PITR).

В процессе работы PostgreSQL ведётся журнал предзаписи (Write-Ahead Log, WAL), который расположен в подкаталоге `pg_wal/` каталога с данными кластера баз данных. В этот журнал записываются все изменения, вносимые в файлы данных.

Прежде всего, журнал необходим для безопасного восстановления после аварийной остановки сервера: в этом случае целостность СУБД может быть восстановлена в результате «воспроизведения» записей, зафиксированных после последней контрольной точки.

Кроме того, наличие журнала делает возможным использование инкрементной стратегии копирования баз данных: можно сочетать резервное копирование на уровне файловой системы с копированием файлов WAL.

PostgreSQL. Непрерывное архивирование WAL-файлов

Преимущества подхода на основе непрерывного архивирования:

- В качестве начальной точки для восстановления необязательно иметь полностью согласованную копию на уровне файлов. Внутренняя несогласованность копии будет исправлена при воспроизведении журнала.
- Поскольку при воспроизведении можно обрабатывать неограниченную последовательность файлов WAL, непрерывную резервную копию можно получить, просто продолжая архивировать файлы WAL.
- Воспроизводить все записи WAL до самого конца нет необходимости. Воспроизведение можно остановить в любой точке и получить целостный снимок базы данных на выбранный момент времени.
- Если непрерывно передавать последовательность файлов WAL другому серверу, получившему данные из базовой копии того же кластера, получается система «тёплого» резерва: в любой момент мы можем запустить второй сервер, и он будет иметь практически текущую копию баз данных.

Программы `pg_dump` и `pg_dumpall` не создают копии на уровне файловой системы и **не могут применяться как часть решения по непрерывной архивации**. Создаваемые ими копии являются логическими и не содержат информации, необходимой для воспроизведения WAL.

PostgreSQL. Непрерывное архивирование. WAL-файлы

В абстрактном смысле, запущенная СУБД PostgreSQL производит неограниченно длинную последовательность записей WAL. СУБД физически делит эту последовательность на файлы сегментов WAL. Когда архивирование WAL не применяется, система обычно создаёт только несколько файлов сегментов и затем перезаписывает их.

При архивировании данных WAL необходимо считывать содержимое каждого файла-сегмента, как только он заполняется, и сохранять эти данные, прежде чем файл-сегмент будет перезаписан. PostgreSQL позволяет администратору указать команду оболочки или библиотеку архивирования, которая будет запускаться для архивирования файла завершённого сегмента.

Чтобы включить архивирование WAL, необходимо установить в параметре конфигурации `wal_level` уровень `replica` или выше, в `archive_mode` — значение `on` и задать команду архивирования в параметре `archive_command`. На практике эти параметры всегда задаются в файле postgresql.conf.

В `archive_command` символы `%p` заменяются полным путём к файлу, подлежащему архивации, а `%f` заменяются только именем файла. Простейшая команда для архивации:

```
archive_command = 'test ! -f /mnt/server/archivedir/%f
                        && cp %p /mnt/server/archivedir/%f' # Unix
archive_command = 'copy "%p" "C:\\server\\archivedir\\%f"' # Windows
```

PostgreSQL. Непрерывное архивирование. Базовая копия

Полная физическая копия кластера баз данных PostgreSQL выполняется с использованием утилиты pg_basebackup. Процедура создания копии выполняется в оперативном режиме, базовая копия сохраняется в виде обычных файлов или в архиве tar. Резервные копии создаются для кластера целиком.

```
pg_basebackup [параметр-подключения...] [параметр...]
```

Простейший вариант вызова утилиты:

```
$> pg_basebackup -D каталог
$> pg_basebackup -U postgres -D "E:\\backup"
```

Процесс создания базовой резервной копии записывает файл истории резервного копирования, который сохраняется в каталоге архивации WAL файлов. Данный файл именуется по имени файла первого сегмента WAL, который потребуется для восстановления скопированных файлов.

Файл истории резервного копирования — это текстовый файл, в который записывается метка, которая была передана pg_basebackup, а также время и текущие сегменты WAL в момент начала и завершения резервной копии.

Для восстановления данных из полной резервной копии (полученной с помощью утилиты `pg_basebackup`) и инкрементных копий (файлы журналов WAL) следует выполнить следующее:

1. Остановить сервер баз данных, если он запущен.
2. Скопировать весь текущий каталог кластера баз данных и все табличные пространства во временный каталог на случай, если они понадобятся. Как минимум, следует сохранить содержимое подкаталога `pg_wal/` каталога кластера, так как он может содержать журналы, не попавшие в архив перед остановкой системы.
3. Удалить все содержимое директории сервера, на котором будет выполняться восстановление данных.
4. Скопировать содержимое папки, в которую была выполнена полная резервная копия, в директорию данных сервера.
5. Удалить все файлы из `pg_wal/`; они восстановились из резервной копии файлов и поэтому, скорее всего, будут старее текущих.
6. Если на шаге 2 были сохранены незаархивированные файлы с сегментами WAL, необходимо скопировать их в `pg_wal/`. Сегменты WAL, которые не найдутся в архиве, система будет искать в `pg_wal/`; благодаря этому можно использовать последние незаархивированные сегменты. Однако файлы в `pg_wal/` будут менее предпочтительными, если такие сегменты окажутся в архиве.

7. Для восстановления данных из инкрементных копий (архивных WAL-файлов) в конфигурационном файле сервера написать команду восстановления, например (для Windows):

```
restore_command = 'copy "PATH\\%f" "%p"'
```

   Для восстановления данных до заданного момента времени можно использовать параметр `recovery_target_time`.

8. Создать пустой файл в директории сервера с названием `recovery.signal`.
9. Запустить сервер. Сервер запустится в режиме восстановления и начнёт считывать необходимые ему архивные файлы WAL. По завершении процесса восстановления сервер удалит файл `recovery.signal` (чтобы предотвратить повторный запуск режима восстановления), а затем перейдёт к обычной работе с базой данных.

Резервное копирование в MongoDB

В MongoDB существует несколько вариантов резервного копирования кластеров:

- MongoDB Atlas, официальный сервис облачных вычислений MongoDB, предоставляет как непрерывное резервное копирование, так и снимки облачных провайдеров. Непрерывное резервное копирование требует инкрементного резервного копирования данных в кластере. Снимки облачного провайдера предоставляют локализованное хранилище резервных копий с использованием функционала снимков кластера (например, Amazon Web Services, Microsoft Azure или Google Cloud Platform). Лучшее решение для резервного копирования для большинства сценариев – непрерывное резервное копирование.
- MongoDB также обеспечивает возможность резервного копирования с помощью Cloud Manager и Ops Manager. Cloud Manager – это служба резервного копирования, мониторинга и автоматизации для MongoDB. Ops Manager – это локальное решение, функциональность которого аналогична Cloud Manager.
- Для администраторов, управляющих кластерами MongoDB напрямую, существует несколько стратегий резервного копирования.

- Создание снимка файловой системы.

Для резервного копирования путем создания снимка файловой системы используются инструменты системного уровня, такие как LVM. Снимки создают образ всего диска или тома. Если выполнять резервное копирование всей системы не требуется, следует рассмотреть возможность изоляции файлов данных MongoDB, журнала (если это применимо) и конфигурирования на одном логическом диске, который не содержит никаких других данных.

Чтобы сбросить записи на диск и «заблокировать» базу данных, необходимо выполнить метод:

```
db.fsyncLock();
```

Чтобы разблокировать базу данных после создания моментального снимка, используется метод:

```
db.fsyncUnlock();
```

- Копирование файлов данных.

База данных блокируется командой `fsyncLock`, затем происходит копирование файлов с данными средствами операционной системы, затем база разблокируется командой `fsyncUnlock`.

MongoDB. Логическое резервное копирование

Логическое резервное копирование выполняется с использованием утилиты `mongodump`.

- ▼ Медленная работа как при получении резервной копии, так и при восстановлении из нее по сравнению с физическим резервным копированием.
- ▼ Существует ряд проблем, связанных с резервным копированием набора реплик: помимо данных необходимо также фиксировать состояние набора реплик, чтобы обеспечить точный снимок вашего развертывания.

- ▲ Логическое копирование - это хороший способ для резервного копирования отдельных баз данных, коллекций и даже подмножеств коллекций.
- ▲ Логическое резервное копирование не требует остановки сервера.

MongoDB. Логическое резервное копирование

Для резервного копирования данных из экземпляра mongod, работающего на том же компьютере и на порту по умолчанию 27017, используется следующая команда:

```
$> mongodump
```

Утилита mongodump создаст в текущем каталоге каталог dump, в котором содержится дамп всех данных. Чтобы указать другой выходной каталог, используется параметр `--out` или `-o`:

```
$> mongodump --out=/data/backup/
```

Чтобы ограничить объем данных, включенных в дамп базы данных, можно указать параметры `--db` и `--collection` :

```
$> mongodump --collection=inventory --db=db_lectures
```

Чтобы запустить mongodump для создания резервной копии MongoDB с включенным контролем доступа, у пользователи должны быть привилегии, предоставляющие право выполнять метод `find` для каждой резервной копии базы данных. Встроенная роль `backup` предоставляет необходимые привилегии для выполнения резервного копирования любых баз данных.

```
$> mongodump --host=mongodb1.example.net --port=37017 --username=user
   --authenticationDatabase=admin --out=/opt/backup/mongodump-2022-11-02
```

MongoDB. Логическое резервное копирование

Для восстановления из резервной копии mongodump используется утилита mongorestore:

```
$> mongorestore --port=<порт> <путь к дампу>
```

Чтобы восстановить данные в MongoDB с включенным контролем доступа, пользователю необходимо назначить роль restore, которая предоставляет необходимые привилегии для восстановления данных из резервных копий.

```
$> mongorestore  --host=mongodb1.example.net --port=37017 --username=user
--authenticationDatabase=admin /opt/backup/mongodump-2022-11-02
```

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
к.т.н., доцент кафедры ГИиИБ