Postgre-DBA

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Резервное копирование и восстановление

Ведущий разработчик PostgreSQL/Greenplum в Сбере

Специалист в области разработки и проектирования витрин данных в PostgreSQL/Greenplum, а также в области разработки хранимых процедур в таких СУБД как PostgreSQL/Greenplum, Oracle, MS SQL Server.

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

- Best practice
- Логический бекап
- Физический бекап
- Практика
- Рефлексия

Цели вебинара

К концу занятия вы сможете

1. Настраивать бэкапы
2. Восстанавливать данные

Смысл

Зачем вам это уметь

1. Чтобы предотвратить потерю базы

Best practice и терминология

С чего начать?

Основные рекомендации

1. Бэкап должен быть всегда.
2. Бэкап должен быть автоматическим.
3. Восстановление из бэкапа — это крайняя мера.
4. Бэкап нужно хранить отдельно от данных и минимум 2 недели.
5. Бэкап нужно регулярно проверять.
6. Полезно дублировать бэкап на удаленную площадку.
7. Бэкап – это нагрузка на работающую систему.

Правило 3-2-1 в резервировании данных

1. Иметь не менее трех экземпляров данных.
2. Хранить копии не менее чем на двух носителях.
3. Хранить не менее одной копии данных за пределами офиса.

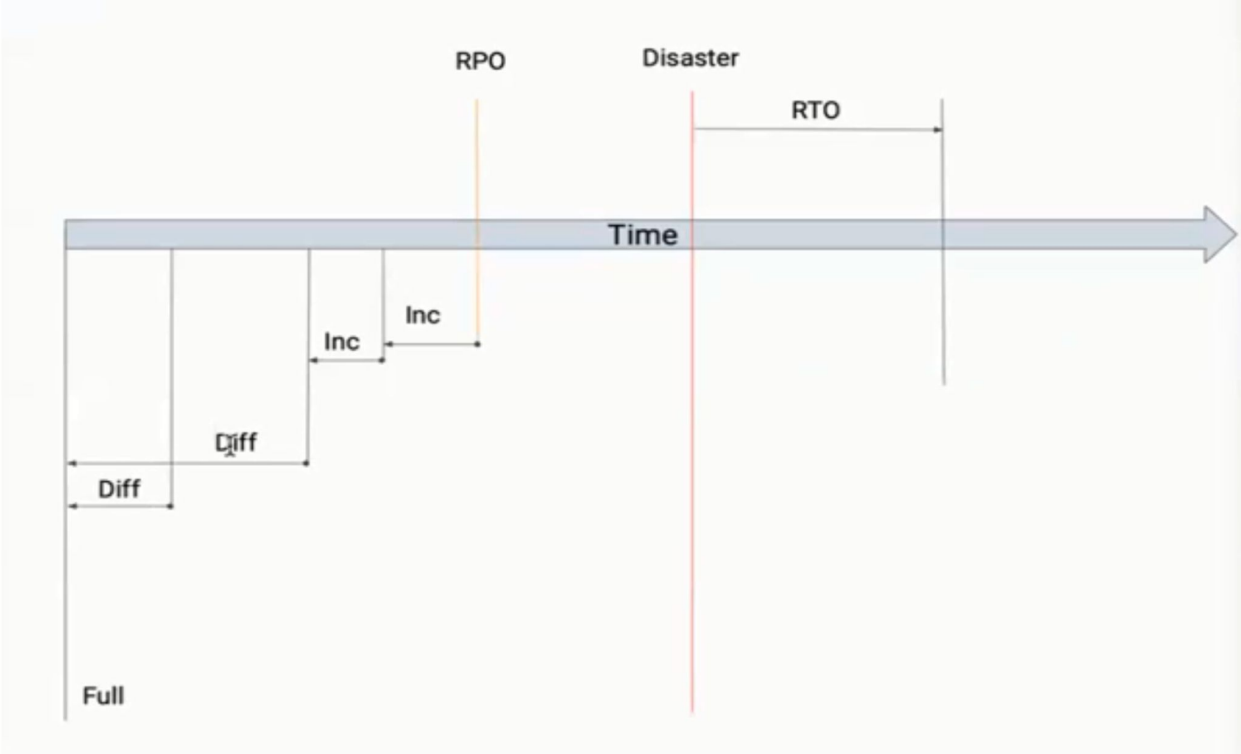
Терминология

- **RTO (Recovery Time Objective)** – определение времени, требуемого на восстановление резервной копии. Например: восстановление из РК должно занимать не более 1 часа.
- **RPO (Recovery Point Objective)** – точка во времени (Point in Time) на которую должны быть восстановлены данные. Например, данные должны быть восстановлены на состояние не «дольше», чем 24 часа с момента сбоя.
- **Backup Level** – уровень резервного копирования (0 – Full, 1 – Differential, 2 – Incremental), это стратегии выбора данных для копирования.
- **Глубина резервного копирования** – определяет, как долго хранятся копии.

Уровни резервного копирования

- **Full** – полное резервное копирование, для восстановления требуется только эта резервная копия.
- **Differential** – разностное резервное копирование: копируется только то, что изменилось с последнего полного резервного копирования. Для восстановления требуется последняя полная версия + последняя дифференциальная копия.
- **Incremental** – инкрементальное резервное копирование: копируется только то, что изменилось с последнего прохода резервного копирования. Для восстановления требуется последняя полная версия + последняя дифференциальная копия (если есть) + ВСЕ инкрементальные копии с момента последней полной/дифференциальной копии.

Уровни резервного копирования



Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Логическое копирование

Логическое копирование

+ можно сделать копию отдельного объекта или базы
+ можно восстановиться на кластере другой основной версии
+ можно восстановиться на другой архитектуре

- невысокая скорость относительно физической

COPY

```
COPY table_name [ ( column_name [, ...] ) ] FROM { 'filename' | PROGRAM 'command' | STDIN } [ [ WITH
] ( option [, ...] ) ] [ WHERE condition ]

COPY { table_name [ ( column_name [, ...] ) ] | ( query ) } TO { 'filename' | PROGRAM 'command' |
STDOUT }
    [ [ WITH ] ( option [, ...] ) ]

    FORMAT format_name
    FREEZE [ boolean ]
    DELIMITER 'delimiter_character'
    NULL 'null_string'
    DEFAULT 'default_string'
    HEADER [ boolean | MATCH ]
    QUOTE 'quote_character'
    ESCAPE 'escape_character'
    FORCE_QUOTE { ( column_name [, ...] ) | * }
    FORCE_NOT_NULL ( column_name [, ...] )
    FORCE_NULL ( column_name [, ...] )
    ENCODING 'encoding_name'
```

PG_DUMP

+ выдает на консоль или в файл либо SQL-скрипт,
+ либо архив в специальном формате с оглавлением
+ поддерживает параллельное выполнение
+ позволяет ограничить набор выгружаемых объектов (таблицы --table, схемы --schema-only, данные --data-only и т.п.) - по умолчанию не создает tablespace и юзеров

```
$ pg_dump -d backup --create
$ pg_dump -d backup --create | gzip > backup.gz
$ pg_dump -d backup -Fc > 1.gz - для pg_restore
```

Восстановление

Так как это простой SQL скрипт:

```
$ psql < 1.sql
```

- заранее должны быть созданы роли и табличные пространства

pg_restore - если архив с оглавлением

- (позволяет ограничить набор объектов при восстановлении)
- поддерживает параллельное выполнение
- заранее должны быть созданы роли, табличные пространства и БД!!!
- после восстановления имеет смысл выполнить сбор статистики (ANALYZE)

```
$ pg_restore 2.gz
```

PG_DUMPALL

- сохраняет весь кластер, включая роли и табличные пространства
- выдает на консоль или в файл SQL-скрипт
- параллельное выполнение не поддерживается
- можно выгрузить только глобальные объекты и воспользоваться pg_dump

```
$ pg_dumpall > backup.sql
$ pg_dumpall --clean --globals-only > globals.sql
$ pg_dumpall --clean --schema-only > schema.sql
```

Восстановление

```
$ psql < backup.sql
```

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Физическкое копирование

Физическое копирование

Используется механизм восстановления после сбоя - копия данных и журналы предзаписи.

+ скорость восстановления
+ можно восстановить кластер на определенный момент времени

- нельзя восстановить отдельную базу данных, только весь кластер
- восстановление только на той же основной версии и архитектуре

Виды физического резервирования

Холодное - когда БД остановлена

- сервер корректно остановлен (необходимы только файлы данных)
- некорректно выключенный (файлы данных и wal сегменты)

Горячее - на работающем экземпляре

- необходимы как файлы данных, так и wal сегменты, причем нужно проконтролировать, чтобы сервер сохранил все wal файлы на время копирования основных данных

Создание автономной копии

Автономная копия содержит и файлы данных, и WAL

Резервное копирование — утилита **pg_basebackup**:

- подключается к серверу по протоколу репликации
- выполняет контрольную точку
- переключается на следующий сегмент WAL
- копирует файловую систему в указанный каталог
- переключается на следующий сегмент WAL
- сохраняет все сегменты WAL, сгенерированные за время копирования

Восстановление

- разворачиваем созданную автономную копию
- запускаем сервер

Создание автономной копии

Протокол репликации

- получение потока журнальных записей
- команды управления резервным копированием и репликацией
- обслуживается процессом **wal_sender**
- параметр wal_level = **replica**

Слот репликации

- серверный объект для получения журнальных записей
- помнит, какая запись была считана последней
- сегмент WAL не удаляется, пока он полностью не прочитан через слот

Создание автономной копии

```
SELECT name, setting FROM pg_settings WHERE name IN
('wal_level','max_wal_senders');
```

Необходимо настроить файервол в файле pg_hba.conf

```
SELECT type, database, user_name, address, auth_method FROM
pg_hba_file_rules() WHERE database = '{replication}';
```

Создание автономной копии

Создадим 2 кластер

$pg_createcluster -d /var/lib/postgresql/16/main2 16 main2

Удалим оттуда файлы

$rm -rf /var/lib/postgresql/16/main2

Сделаем бэкап нашей БД

$pg_basebackup -p 5432 -D /var/lib/postgresql/16/main2

Стартуем кластер

$pg_ctlcluster 16 main2 start

Смотрим как стартовал

$pg_lsclusters

Архив журналов

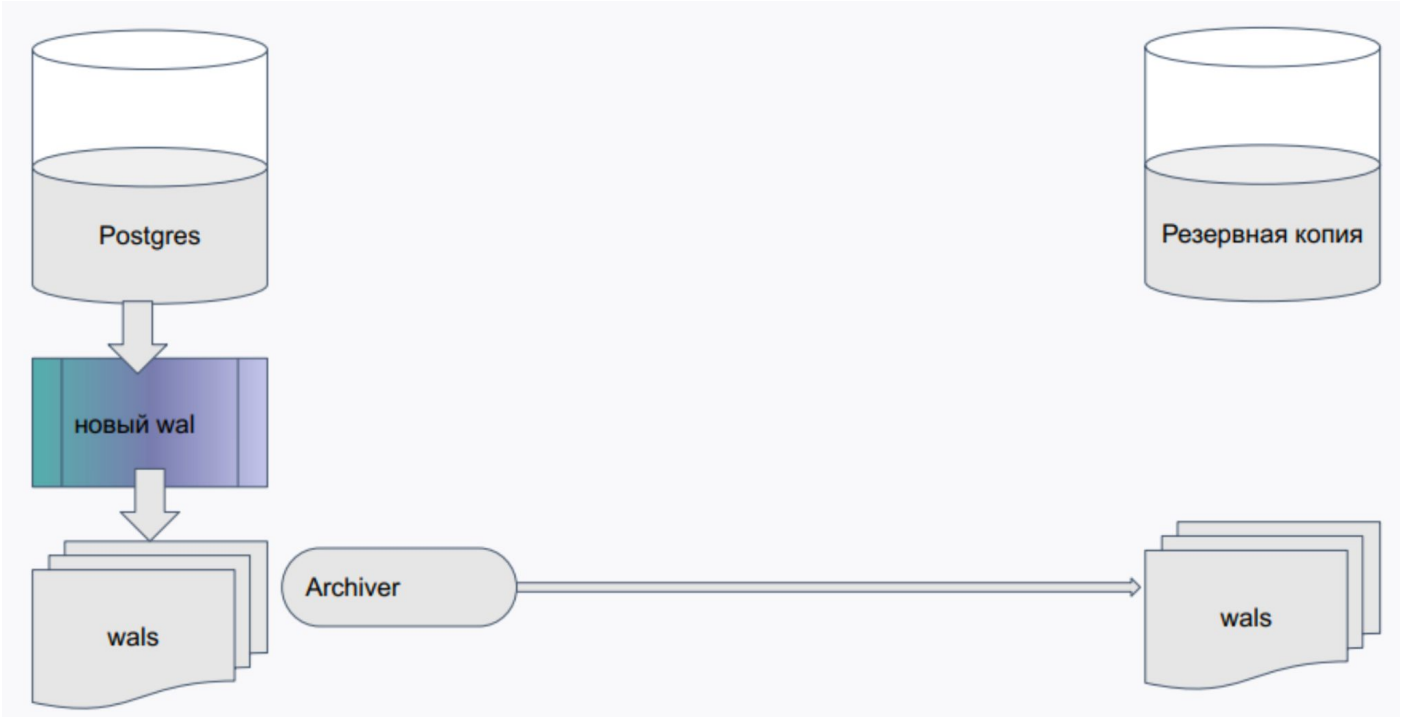
Файловый архив

- сегменты WAL копируются в архив по мере заполнения
- механизм работает под управлением сервера
- неизбежны задержки попадания данных в архив

Потоковый архив

- в архив постоянно записывается поток журнальных записей
- требуются внешние средства
- задержки минимальны

Файловый архив журналов



Файловый архив журналов

Процесс archiver. Параметры

```
SELECT name, setting FROM pg_settings WHERE name IN
('archive_mode','archive_command','archive_timeout');
```

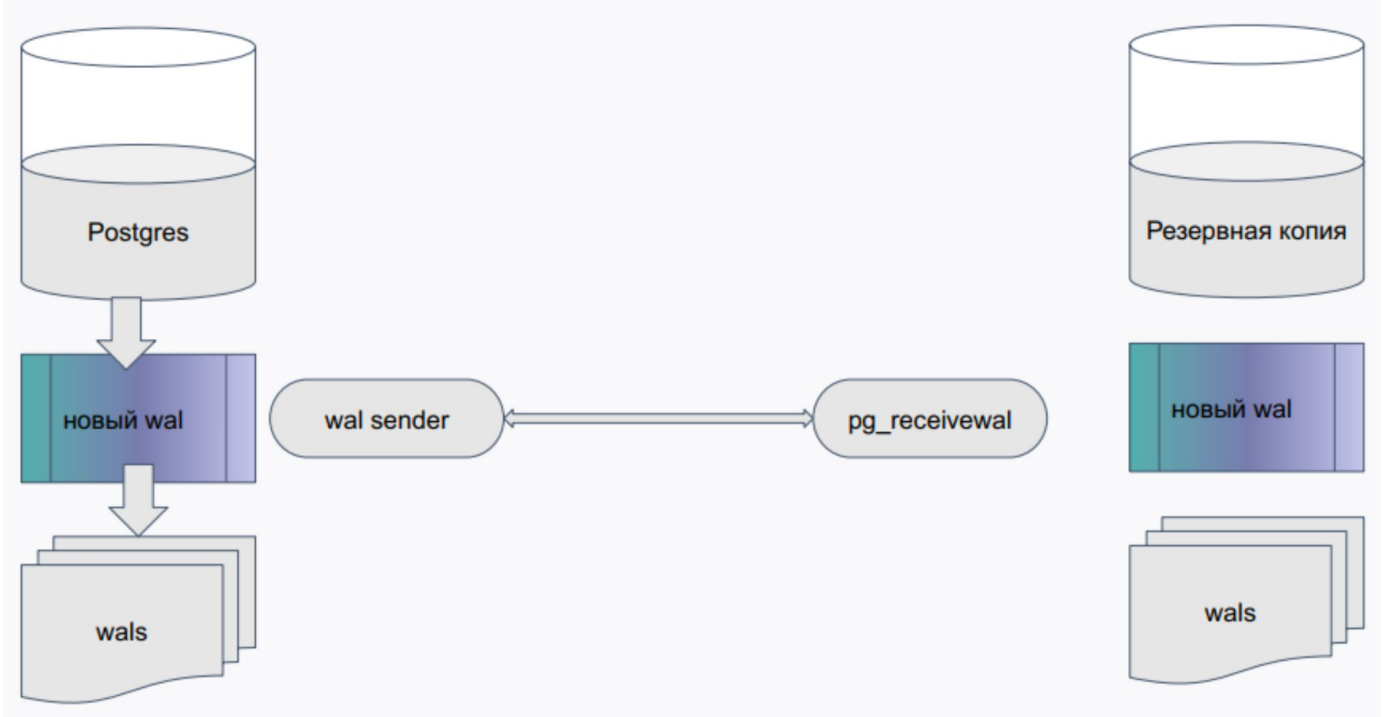
- ALTER SYSTEM SET archive_mode = on
- ALTER SYSTEM SET archive_command - команда shell для копирования сегмента WAL в отдельное хранилище
- ALTER SYSTEM SET archive_timeout - максимальное время для переключения на новыйсегмент WAL
- требуется рестарт сервера

Потоковый архив журналов

Алгоритм

- при заполнении сегмента WAL вызывается команда archive_command
- если команда завершается со статусом 0, сегмент удаляется
- если команда возвращает не 0 (в частности, если команда не задана), сегмент остается до тех пор, пока попытка не будет успешной

Потоковый архив журналов



Потоковый архив журналов

Утилита **pg_receivewal**

- подключается по протоколу репликации (можно использовать слот)
- направляет поток записей WAL в файлы-сегменты
- стартовая позиция — начало сегмента, следующего за последним заполненным сегментом, найденным в каталоге,
- или начало текущего сегмента сервера, если каталог пустой
- в отличие от файлового архива, записи пишутся постоянно
- при переходе на новый сервер надо перенастраивать параметры

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Домашнее задание

ДЗ

1. Создаем ВМ/докер с ПГ.
2. Создаем БД, схему и в ней таблицу.
3. Заполним таблицы автосгенерированными 100 записями.
4. Под линукс пользователем Postgres создадим каталог для бэкапов
5. Сделаем логический бэкап используя утилиту COPY
6. Восстановим в 2 таблицу данные из бэкапа.
7. Используя утилиту pg_dump создадим бэкап в кастомном сжатом формате двух таблиц
8. Используя утилиту pg_restore восстановим в новую БД только вторую таблицу!
9. ДЗ сдается в виде миниотчета на гитхабе с описанием шагов и с какими проблемами столкнулись.

Рефлексия

Список материалов для изучения

1. barman - Многоярусный бэкап PostgreSQL с помощью Barman и синхронногопереноса журналов транзакций
2. wal-e - Разгоняем бэкап. Лекция Яндекса
3. wal-g - WAL-G: бэкапы и восстановление СУБД PostgreSQL
4. BART
5. pg_probackup

Рефлексия

С какими впечатлениями уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Приходите на следующие вебинары

Ведущий разработчик PostgreSQL/Greenplum в Сбере

Специалист в области разработки и проектирования витрин данных в PostgreSQL/Greenplum, а также в области разработки хранимых процедур в таких СУБД как PostgreSQL/Greenplum, Oracle, MS SQL Server.