ИТМО ВТ

Лекция 7

Лектор:
Николаев В.В.

1. Резервное копирование

ИТМО ВТ

Терминология

- Стратегия резервного копирования бывает:
  - полная — охватывает всю базу данных;
  - частичная — охватывает только часть базы данных.
- Резервная копия может содержать:
  - все страницы в пределах выбранных файлов;
  - ту информацию, которая изменилась с момента последнего резервного копирования (инкрементная):
    - кумулятивная (изменения — до последнего уровня 0);
    - дифференциальная (изменения — до последнего инкрементного резервного копирования).

ИТМО ВТ

Виды резервного копирования (1)

Виды резервного копирования:

- «холодное» — создание резервной копии, когда экземпляр сервера БД остановлен;
- «горячее» — создание резервной копии во время работы экземпляра сервера БД;

Виды резервного копирования (2)

Виды резервного копирования:

- логическое;
- физическое;

2. Логическое РК

ИТМО ВТ

Логическое РК

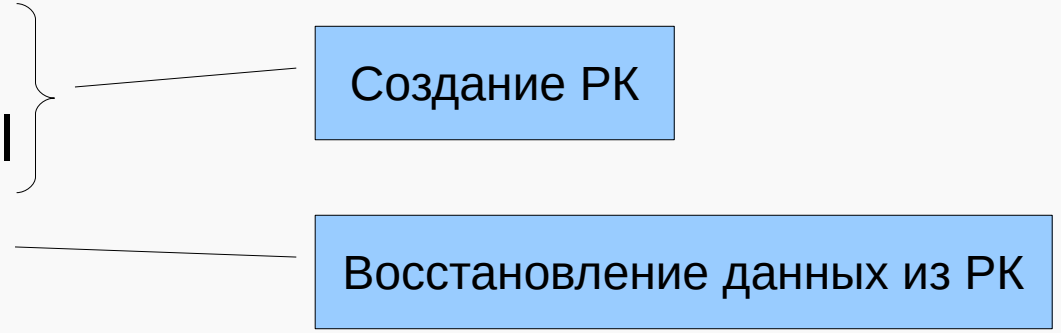
- Копия объектов БД и структур данных создается в виде команд/предложений языка SQL, сами файлы не копируются.

- Утилиты в PostgreSQL для осуществления логического резервного копирования/восстановления:
  - pg_dump
  - pg_dumpall
  - pg_restore

Логическое РК

- Копия объектов БД и структур данных создается в виде команд/предложений языка SQL, сами файлы не копируются.

- Утилиты в PostgreSQL для осуществления логического резервного копирования/восстановления:
  - pg_dump
  - pg_dumpall
  - pg_restore

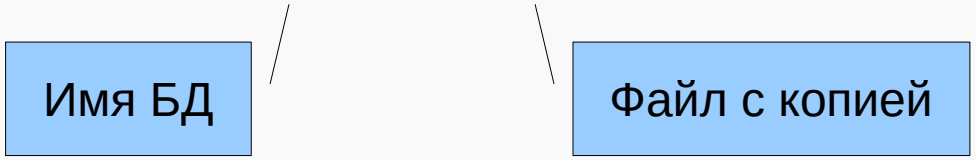


ИТМО ВТ

pg_dump

- Служит для создания резервной копии БД или ее части.
- В общем случае создается SQL-файл с командами, которые позволяют воссоздать логическое состояние БД на момент создания бэкапа.
- Использование:

pg_dump **mydb** > **mydbdump**



pg_dump -U postgres1 -f **mydbdump mydb**

ИТМО ВТ

pg_dump: форматы PK

- **plain text**, текстовый формат — в файле ~SQL-предложения:
  - можно восстановить, выполнив в psql;
- **custom** формат:
  - возможно сжатие (если есть zlib);
  - бинарный файл;
- **в виде директории** — один файл для каждой ~таблицы;
- **tar** — архив PK, в архиве содержимое, как для формата в виде директории;

ИТМО ВТ

pg_dump: форматы PK

- **plain text**:

  pg_dump -f mydbdump mydb

- **custom** формат:

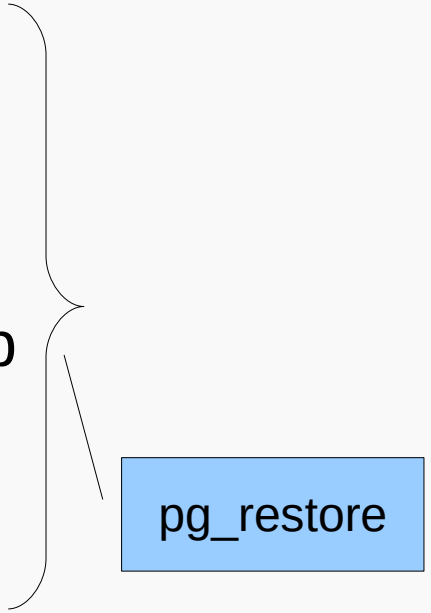
  pg_dump **-Fc** -f mydbdump mydb

- **в виде директории**:

  pg_dump **-Fd** Student -f mydbdump mydb

- **tar**:

  pg_dump **-Ft** Student -f mydbdump mydb



Формат PK в виде директории

pg_dump **-Fd** Student -f mydbdump mydb

- Для отдельных объектов БД создаются отдельные файлы:
  - возможно сжатие файлов в директории;
- В директории есть файл toc.dat:
  - индекс для pg_restore;
  - для поиска файлов внутри директории;

ИТМО ВТ

Текстовый формат

- Используется по умолчанию.
- Содержимое — предложения для восстановления логического состояния БД.
- По умолчанию для **добавления данных** используется PostgreSQL команда **COPY**.
- Для генерации INSERT:

  pg_dump -U postgres1 **--inserts** -f mydbdump mydb

  pg_dump -U postgres1 **--column-inserts** -f mydbdump mydb

Особенности использования pg_dump

- По умолчанию не создаются предложения для создания БД.
- Чтобы восстановить данные нужно подключиться к существующей БД.
- Для добавления создания БД (CREATE DATABASE) можно использовать опцию —create (-C):
  - при использовании опции в бэкап добавляются предложения для подключения к созданной БД:

    pg_dump -U postgres1 **--create** -f mydbdump mydb

Получение скрипта для восстановления части БД

Можно задать создание скрипта для восстановления только определенных таблиц, структуры БД, содержимого БД:

- pg_dump **-t** Student -f mydbdump mydb -- only Student
- pg_dump **-T** Student -f mydbdump mydb -- all except -- Student
- pg_dump **-a** -f mydbdump mydb -- only data
- pg_dump **-s** -f mydbdump mydb -- only **schema (DDL)**

ИТМО ВТ

Восстановление БД

- Использование:

  psql **mydb** < **mydbdump**

- Использующиеся в бэкапе пользователи, роли должны существовать/быть воссозданы до запуска скрипта.

Ошибки при восстановлении БД

- Чтобы остановить скрипт после возникновения первой ошибки (невыполненного предложения):

  psql **--set ON_ERROR_STOP=on** mydb < mydbdump

  Результаты прошедших команд останутся в случае возникновения ошибки

- Можно указать, чтобы восстановление выполнялось в рамках **одной транзакции**:

  psql **--single-transaction** mydb < mydbdump

ИТМО ВТ

pg_restore

- Утилита для восстановления данных на основе скриптов, созданных pg_dump:

  pg_restore **-d** mydb mydbdump

- Можно посмотреть список команд, которые будут исполнены при восстановлении БД:

  pg_restore mydbdump **-f** test.sql

- Если используется директория — можно включить свой toc-файл и сделать частичное восстановление.

ИТМО ВТ

pg_dumpall

- Утилита для создания скриптов восстановления всех БД в кластере:

  pg_dumpall -U postgres1 -f mydbdump

- Создает SQL-скрипт.
- Кроме БД сохраняются глобальные объекты (пользователи, роли, табличные пространства).
- Восстановление БД:

  psql -f dumpfile postgres

ИТМО ВТ

Осуществление резервного копирования

- Возможен параллельный режим:

  pg_dump, опция -j:

  - можно указать число параллельных процессов.

- Есть готовые скрипты-шаблоны для организации резервного копирования:

https://wiki.postgresql.org/wiki/Automated_Backup_on_Linux

3. Физическое РК

ИТМО ВТ

Физическое РК

- При создания резервной копии происходит копирование файлов исходного кластера.

- Утилиты в PostgreSQL для осуществления физического резервного копирования/восстановления:
  - pg_basebackup
  - функции pg_start_backup, pg_stop_backup

ИТМО ВТ

Резервное копирование на уровне файловой системы

- Заключается в полном копировании директории кластера БД средствами файловой системы.
- Обычно производится в «холодном режиме» - сервер БД должен быть остановлен:
  - «горячий режим» возможен если файловая система поддерживает возможность получения единовременного «снимка» данных.
- Может быть произведен только для всех объектов БД.

ИТМО ВТ

Непрерывное архивирование

- Базируется на существовании WAL (PGDATA/pg_wal):
  - состояние БД может быть восстановлено путем повтора зафиксированных в логе операций;
  - конфигурационные файлы не восстанавливаются через WAL (postgresql.conf, ...);

Резервная копия = полная копия + WAL-файлы

ИТМО ВТ

Физическое PK + WAL

- БД не обязательно должна быть в полностью согласованном состоянии:
  - для приведения состояния к согласованному может использоваться WAL;
  - С точки зрения работы WAL — механизмы аналогичны тем, которые используются при восстановлении после сбоев.
- Можно восстановить данные в одно из временных состояний между снятием бэкапа и последним WAL.
- Можно восстановить базу данных целиком (а не отдельную часть).

ИТМО ВТ

Этапы для реализации непрерывного архивирования

1) Организация архивирования WAL.
2) Создание базовой резервной копии.

ИТМО ВТ

Этап 1: архивирование WAL

- На диске WAL разделен на **сегменты** по 16 Мб (по умолчанию).
- Необходимо осуществить копирование добавляемых сегментов на внешнее хранилище.
- Для включения возможности резервного копирования:
  - archive_mode = on
  - wal_level = replica # минимум
- PostgreSQL позволяет задать команду для создания копий WAL по мере их создания (postgresql.conf):
  - archive_command = ' some command '

Команда для архивирования

Задается в postgresql.conf:

archive_command = 'cp **%p** /path2/**%f**' # Unix

- %p — заменяется на полное имя файла для архив-я.
- %f — заменяется на имя файла.

Назначение:

- копирование сегментов WAL в path2.
- команда должна возвращать 0 только в случае успешного завершения.

ИТМО ВТ

archive_command

- Значение можно изменить, перегрузив значения конфигурационного файла.

- Частоту создания новых сегментов можно регулировать через archive_timeout:
  - если он небольшой будет создаваться много незаполненных сегментов и тратиться лишнее место.

Этап 2: создание базовой резервной копии - pg_basebackup

- Можно осуществить через pg_basebackup:

pg_basebackup -D /path/backup -T /old/ts=$(pwd)/new/ts

- backup_history-файл в pg_wal (***.backup) — имя содержит название первого WAL-сегмента для осуществления восстановления.
- С помощью pg_basebackup можно осуществить создание обособленной копии (без осуществления непрерывного копирования).

ИТМО ВТ

Этап 2: создание базовой резервной копии — через pg_*_backup

функции pg_start_backup, pg_stop_backup:

- SELECT pg_start_backup('my_label');
  - создается **backup_my_label** файл — содержит различную информацию о резервной копии (время начала, метку, ...);
  - создается tablespace_map;
  - создается контрольная точка;
- пользователем производится копирование файлов кластера БД;
- SELECT pg_stop_backup();

ИТМО ВТ

Восстановление данных на основе бэкапа с непрерывным архивированием (1)

1)Нужно остановить сервер БД.

2)Если возможно, сделать копию — хотя бы pg_wal (на случай, если есть незаархивированные WAL).

3)Перед шагом: **проверить наличие бэкапа**. Если бэкап есть — удалить содержимое PGDATA и, если есть, внешних директорий (с табличными пространствами и т. д).

4)Скопировать в PGDATA и внешние директории данные ранее полученного бэкапа.

5)Очистить директорию PGDATA/pg_wal.

6)Если есть незаархивированные WAL (с шага 2) — поместить их в PGDATA/pg_wal.

Восстановление данных на основе бэкапа с непрерывным архивированием (2)

7)Установить настройки для восстановления в postgresql.conf (**restore_command**, должна возвращать ненулевой код в случае неудачи) + запретить внешние подключения (для пользователей) в pg_hba.conf.

8)Создать файл **recovery.signal** — говорит серверу, что надо стартовать в режиме восстановления (PGDATA/data).

9)Запустить сервер → он перейдет в режим восстановления. По завершении recovery.signal будет удален.

10)Разрешить подключения для пользователей в pg_hba.conf.

ИТМО ВТ

restore_command

- Задает логику получения архивированных WAL-сегментов:

  restore_command = 'cp /path/%f %p'

  - %f - имя файла (WAL-сегмента);
  - %p - путь, куда копировать файлы (WAL-сегменты).
- Сегменты, которые не были найдены в архиве, будут искаться в pg_wal:
  - приоритет тем сегментам, которые в архиве.

ИТМО ВТ

Источники

Документация PostgreSQL:
https://www.postgresql.org/docs/14/index.html

Лицензия PostgreSQL:

https://www.interdb.jp/pg/index.html