## ЛР4. PostgreSQL. Резервное копирование. Репликация

### 1. Резервное копирование

Резервное копирование базы данных является одной из наиболее важных задач администрирования БД. Необходимо выполнять резервное копирование БД, чтобы иметь возможность восстановить данные и снова запустить их на сервере в случае возникновения проблем, таких как сбои системы, сбои оборудования или действия пользователей, удаляющих данные по ошибке.

Классификация типов резервного копирования по способу создания:

- *Физические резервные копии* состоят из необработанных копий каталогов и файлов, в которых хранится содержимое базы данных. Этот тип резервного копирования подходит для больших и важных баз данных, которые необходимо быстро восстанавливать при возникновении проблем.
- *Логические резервные копии* сохраняют информацию, содержащую логическую структуру базы данных (запросы `CREATE_DATABASE`, `CREATE_TABLE`) и ее содержимое (запросы `INSERT`). Этот тип резервного копирования подходит для небольших объемов данных, так как время создания логической копии существенно выше, чем физической. Однако, логическая копия может быть легко перенесена на компьютер/БД любой архитектуры.

Классификация типов резервного копирования по резервируемым данным:

- *Полное резервное копирование*. Этот тип резервного копирования создает копии всей базы данных.
- *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.
- *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.

СУБД PostgreSQL поддерживает создание логических и физических резервных копий с помощью входящих в ее состав утилит.

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. Текстовые файлы, созданные `pg_dump`, используются для последующего чтения программой `psql`. Дампы, выгруженные не в текстовом формате, восстанавливаются утилитой `pg_restore`.

Для создания физической резервной копии используется утилита `pg_basebackup`. Данная утилита позволяет создать физическую копию сервера, не выключая и не блокируя его, за счет использования протокола репликации. Данный протокол позволяет получать поток журнальных записей, которые генерирует сервер. Для выполнения копирования утилита `pg_basebackup` использует два подключения по протоколу репликации: одно для передачи данных и одно для передачи потока журнальных записей, которые генерирует работающий сервер во время копирования. Подробное описание синтаксиса команд утилиты представлено на сайте: https://postgrespro.ru/docs/postgresql/15/app-pgbasebackup. В ходе выполнения лабораторной работы будет рассмотрено создание физических копий.

## 2. Репликация

Репликация - набор технологий копирования и распространения данных и объектов баз данных между базами данных и последующей синхронизации баз данных для поддержания их согласованности.

Типы репликации:

- Синхронная репликация: транзакция, модифицирующая данные, считается подтверждённой только тогда, когда все серверы подтвердят эту транзакцию.
- Асинхронная репликация: транзакция, модифицирующая данные, считается подтверждённой тогда, когда она подтверждена локально на том сервере, на котором эти изменения происходили.
- Полусинхронная репликация: транзакция, модифицирующая данные, считается подтверждённой тогда, когда хотя бы один из других серверов подтвердит получение изменений.

В контексте репликации у серверов могут быть две роли: ведущий сервер и подчиненный (ведомый) сервер. При репликации данные копируются с ведущего сервера на подчиненные серверы. Количество серверов с этими ролями определяет различные виды топологии репликации:

- Репликация с одним ведущий сервером. Данные всегда отправляются на один конкретный ведущий узел.
- Репликация с несколькими ведущими серверами. Может существовать несколько узлов, играющих роль ведущих, и каждый ведущий узел должен сохранять данные в кластере.
- Репликация без ведущих серверов. Ожидается, что все узлы могут принимать данные при записи.

СУБД PostgreSQL поддерживает физическую и логическую репликацию.

При физической репликации серверы имеют назначенные роли: один является ведущим («мастер») и один или несколько серверов являются ведомыми («реплики»). Мастер передает на реплику журнальные записи, а реплика применяет эти записи к своим файлам данных. Поскольку журнал является общим для всего кластера, то и реплицировать можно только кластер целиком, а не отдельные базы данных или таблицы. Возможность «отфильтровать» журнальные записи отсутствует. Реплика не может генерировать собственных журнальных записей, она лишь применяет записи мастера. Поэтому при таком подходе реплика может быть доступна только для чтения — никакие операции, изменяющие данные, на реплике не допустимы.

При логической репликации на одном сервере создается публикация, другие серверы могут на нее подписаться. У сервера нет выделенной роли: один и тот же сервер может как публиковать изменения, так и подписываться на другие (или даже свои) подписки. Подписчику передается информация об изменениях строк в таблицах в платформонезависимом виде; двоичная совместимость не требуется. Для работы логической репликации в журнале публикующего сервера необходима дополнительная информация (параметр `wal_level = logical`). Логическая репликация позволяет транслировать не все изменения, а только касающиеся определенных таблиц. Реплицируются только изменения базовых таблиц, вызванные командами `INSERT`, `UPDATE` и `DELETE`. В частности, не реплицируются команды `DDL`, что означает необходимость для подписчика предварительно самостоятельно создать все необходимые таблицы.

### 3. Настройка репликации в PostgreSQL

Рассмотрим кратко процесс настройки физической репликации. Пусть серверы расположены на одной системе и имеют разные порты: 5343 – порт главного сервера и 5344 – порт ведомого сервера.

1. Для каждого сервера создать отдельную папку для хранения данных.
2. Выполнить инициализация директории главного сервера с помощью утилиты `initdb`:

```
initdb -D master_path -U username –W
```

3. Создать папку для хранения журналов предзаписи. Файлы данных журналов используются в качестве инкрементных резервных копий. Настроить архивацию журналов предзаписи в конфигурационном файле сервера `postgresql.conf` (находится в корневой папке директории сервера):

```
archive_mode = ON;
archive_command = 'copy "%p" "PATH\\%f"' # Пример команды архивации для операционной системы Windows
archive_timeout = 60 # время существования не архивированных данных в секундах
```

4. Запустить главный сервер с помощью утилиты `pg_ctl`:

```
pg_ctl start -D master_path -o "-p 5343 "
```

5. Создать пользователя для осуществления репликации:

```
CREATE USER replicator WITH REPLICATION PASSWORD 'password';
```

6. Создать копию главного сервера с помощью утилиты pg_basebackup

```
pg_basebackup -p master_port -U replicator -D path_to_slave -Fp -Xs -P -R
```

7. Запустить ведомый сервер и проверить статус репликации на главном сервере:

```
select * from pg_stat_replication;
```

### 4. Восстановление сервера из резервной копии

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

1. Удалить все содержимое директории сервера, на котором будет выполняться восстановление данных.
2. Скопировать содержимое папки, в которую была выполнена полная резервная копия, в директорию данных сервера.

3. Для восстановления данных из инкрементных копий в конфигурационном файле главного и ведомого серверов написать команду восстановления из архива WAL, например (для Windows):

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

   Для восстановления данных до заданного момента времени используется параметр `recovery_target_time`.
4. Создать пустой файл в директории сервера с названием: "`recovery.signal`".
5. Запустить сервер, процесс восстановления начнется автоматически.

## ЛР4. Вопросы для контроля

1. Классификация типов резервного копирования по резервируемым данным.
2. Физическое резервное копирование.
3. Логическое резервное копирование.
4. Классификация типов резервного копирования по способу создания копий.
5. Репликация. Типы репликации.
6. Репликация. Виды топологии репликации.
7. Репликация с одним ведущий сервером.
8. Репликация с несколькими ведущими серверами.
9. Репликация без ведущих серверов.

## ЛР4. Задание

### Репликация

1. Выполнить настройку физической репликации БД на двух серверах PostgreSQL.
2. Проверить работу репликации:
   2.1. Изменить данные на главном сервере и убедиться, что данные на ведомом сервере также изменились.
   2.2. Выключить ведомый сервер, изменить данные на главном сервере, включить ведомый сервер и убедиться, что данные на ведомом сервере были так же изменены.

### Резервное копирование

3. Настроить и проверить инкрементное резервное копирование БД.
   3.1. После выполнения полного резервного копирования внести некоторые изменения в БД. Сымитировать ситуацию, при которой на главном сервере были потеряны все данные (все содержимое директории сервера удалено).
   3.2. Восстановить потерянные данные из резервной копии (не используя данные ведомого сервера).
   3.3. Восстановить процесс репликации.