Практическая работа № 2.
Настройка подключения к серверу баз данных

Цель работы: научиться настраивать разные способы подключения к СУБД Postgres.

Теоретический материал

1. Клиент-серверная архитектура

PostgreSQL использует модель клиент-сервер, состоящую из:

- **Сервера** — отвечает за управление базами данных, выполнение запросов и обеспечение сохранности данных.
- **Клиента** — приложение, которое взаимодействует с сервером, отправляя запросы и получая данные.

Передача данных осуществляется через собственный протокол PostgreSQL. Протокол поддерживает текстовый и бинарный форматы, работая поверх TCP/IP или Unix-сокетов. Стандартный порт для подключения — 5432.

Сама СУБД PostgreSQL может выполнять роль не только сервера, но и клиента. Как правило, это используется для интеграции разных информационных систем или для аналитики данных, чтобы разгрузить рабочую базу данных в момент выполнения аналитических запросов.

2. Основные этапы взаимодействия

Процесс клиент-серверного взаимодействия проходит через:

1. **Установление соединения:** клиент отправляет стартовое сообщение с параметрами подключения (имя пользователя, база данных, версия протокола).
2. **Аутентификация:** сервер проверяет подлинность клиента с использованием выбранного метода.
3. **Обычная работа:** обмен SQL-запросами и ответами.
4. **Завершение соединения:** соединение закрывается клиентом или сервером; все незавершённые транзакции откатываются.

3. Методы аутентификации

PostgreSQL поддерживает:

- **trust:** подключение без проверки подлинности (рекомендуется только для тестов).
- **md5:** хеширование паролей с MD5 (устаревший).
- **scram-sha-256:** современный метод с улучшенной безопасностью.
- **ssl:** защита через шифрованное соединение.
- **gssapi:** интеграция с корпоративными системами (например, Kerberos).

Практическая часть

В рамках учебного задания мы будем пользоваться одним компьютером, на котором развёрнуто два сервера: один имитирует базу данных для анализа, другой – для хранения текущих данных предприятия. В реальной ситуации они должны размещаться на физически разных машинах.

1. Создание нового сервера

Для создания нового сервера сделайте следующее.

1. Создайте необходимые условия для возможности выполнения команд СУБД из командной строки операционной системы. Для этого необходимо сделать одно из двух действий:
    a. войти в командном интерпретаторе в поддиректорию bin директории, куда установлена СУБД Postgres;
    b. прописать путь к директории bin в переменную окружения операционной системы PATH.

    Первый способ позволяет развернуть новый сервер с минимальными вмешательствами в работу операционной системы. Второй способ позволяет настроить операционную систему один раз и больше к этому не возвращаться, когда потребуется разворачивать новые сервера.
2. Создайте директорию для сервера. Проще всего будет настроить эту директорию, если она не находится в системных директориях операционной системы (Windows или Program Files).
3. Запустите командную строку из-под администратора. В командной строке введите:

```
initdb -D "путь к созданной на шаге 2 директории"
```

Дополнительно можно указать пользователя Windows, под которым будет работать сервер, и задать пароль для создаваемого сервера. Для указания пользователя используйте опцию **`-U`** с указанием нужного пользователя (скорее всего, это будет пользователь postgres). Для задания пароля используйте опцию **`-W`**. В этом случае перед выполнением команды система запросит новый пароль для создаваемого сервера. С помощью параметров **`--locale`** и **`--encoding`** можно задать локализацию и кодировку по умолчанию. Рекомендуется использовать для этих параметров значения **`Russian_Russia`** и **`UTF-8`** соответственно.

В результате выполнения команды **`initdb`** в указанной директории создадутся все необходимые для работы сервера файлы и поддиректории. Как правило, для настройки сервера Postgres требуется редактировать конфигурационные файлы **`postgresql.conf`** и **`pg_hba.conf`**. Некоторые настройки задаются путём выполнения определённых SQL-команд. В частности, файл **`postgresql.conf`** содержит параметр **`port`**, в котором указывается, на каком порту будет работать создаваемый сервер.

Чтобы выбрать порт, выполните в командном интерпретаторе операционной системы команду, которая выводит список занятых портов:

```
netstat -a
```

После этого в параметр **port** нужно ввести любой не занятый порт.

Ещё один параметр, который требуется указать перед первым запуском созданного сервера – это **listen_addresses**. Впишите туда значение **localhost** или ***** по желанию.

Для запуска созданного сервера сделайте следующее.

4. Создайте необходимые условия для возможности выполнения команд СУБД из командной строки операционной системы.
5. Запустите командную строку из-под администратора. В командной строке введите (на одной строке):

```
pg_ctl register -N "имя_новой_службы" –S auto
  -D "путь к созданной на шаге 1 директории"
```

6. С помощью инструмента Windows «Панель управления /Администрирование/Службы» убедитесь, что службы успешно созданы (Рисунок 1).
7. Запустите созданную службу либо через графический интерфейс инструмента «Службы», либо выполнив такую команду:

```
net start имя_новой_службы
```

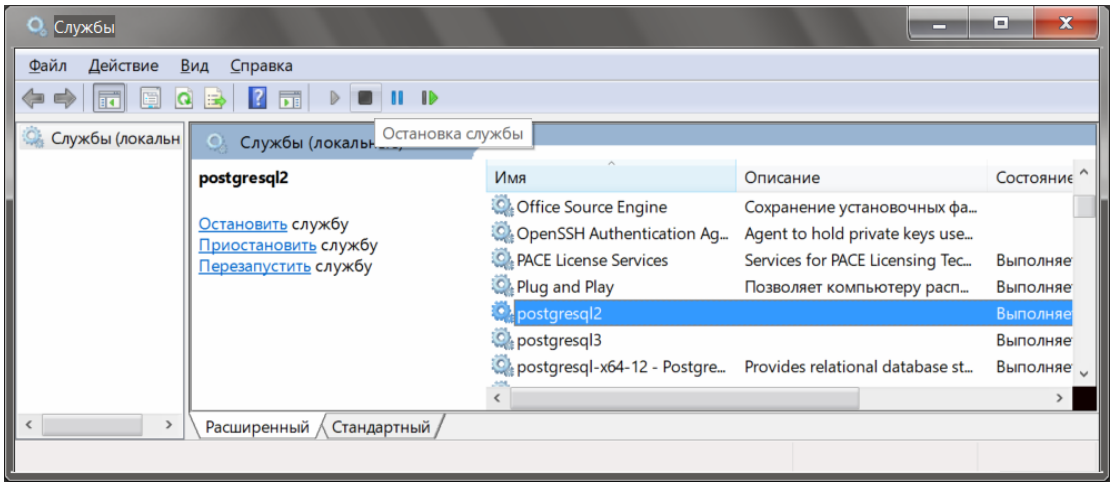

Рисунок 1. Успешное создание службы Windows для нового сервера СУБД Postgres.

Для проверки работоспособности созданного сервера необходимо зайти в pgAdmin и создать новый сервер. В данной утилите «сервер» – это не сам экземпляр сервера, который мы создали на предыдущих шагах, а набор настроек подключения к нему. Введите в качестве настроек localhost (или IP адрес локальной машины) и порт, указанный в конфигурационном файле. Если на шаге 3 был задан пароль, то его тоже нужно будет ввести (Рисунок 2).

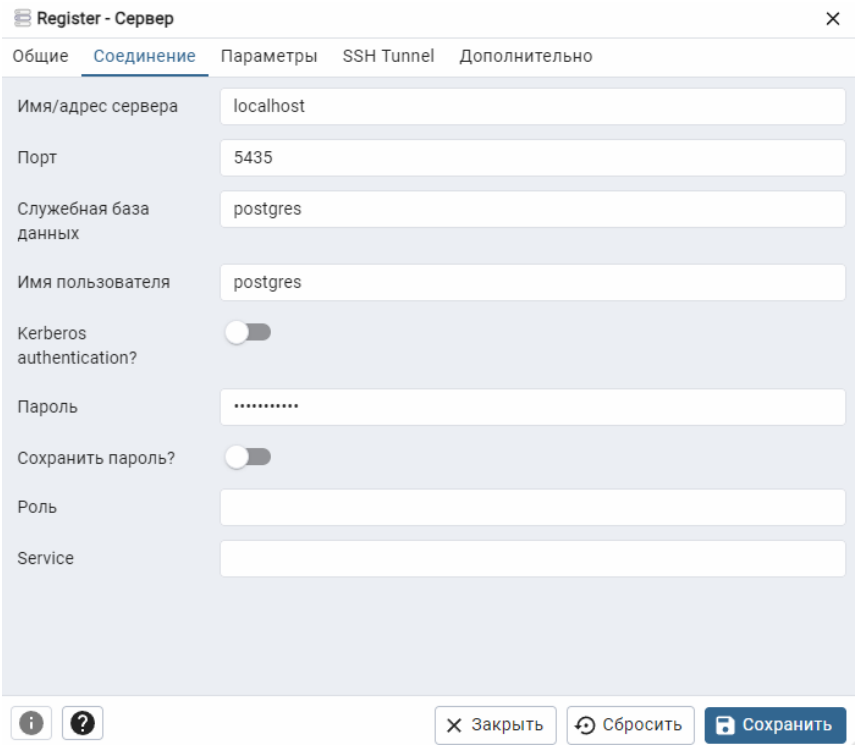

Рисунок 2. Создание нового сервера в pgAdmin.

Проверьте, что подключение через pdAdmin проходит успешно.

Примечание. Иногда порт, прописанный в конфигурационном файле, не подхватывается службой Windows при старте. В этом случае службу необходимо остановить, выполнив команду:

```
net stop <имя_службы>
```

Затем службу необходимо удалить, выполнив команду:

```
pg_ctl unregister -N <имя_службы>
```

Ещё службу можно удалить такой командой (как вариант):

```
sc delete <имя_службы>
```

После того, как служба удалена, можно заново её зарегистрировать, явно указав порт в команде регистрации службы:

```
pg_ctl register -N <имя_службы> -D <каталог_данных>
      -o "-p <порт>"
```

Например:

```
pg_ctl register -N "postgresql-x64-15" -D
      "C:\Program Files\PostgreSQL\15\data" -o "-p 5433"
```

После этого зарегистрированная служба будет работать на указанном при регистрации порте и игнорировать порт, прописанный в конфигурационном файле. Но надо иметь в виду, что данный способ регистрации служб является вспомогательной мерой, использовать его без необходимости не рекомендуется.

2. Настройка шифрования

По умолчанию в новом экземпляре сервера Postgres не настроена защита данных, поэтому дальше необходимо её настроить. Для шифрования необходимо, чтобы и шифрование пароля, и расшифровка пароля выполнялись одним и тем же методом. Перед заданием пароля необходимо указать метод шифрования в параметре **password_encryption** в файле **postgresql.conf**. Рекомендуется использовать **scram-sha-256**. После внесения изменений в этот файл необходимо перезагрузить сервер. Затем необходимо задать новый пароль. Это делается следующей командой в командном интерпретаторе SQL Shell или Query Tool в pgAdmin:

```
ALTER USER <user> WITH PASSWORD '<password>';
```

Затем необходимо задать тот же самый метод расшифровки пароля в файле **pg_hba.conf** (колонка METHOD). После этого переподключитесь к созданному экземпляру сервера Postgres через pgAdmin и убедитесь, что подключение успешно. Для проверки, что всё корректно настроено, в Query Tool выполните следующие команды:

```
show port;
show listen_addresses;
select inet_server_addr();
```

Первая из этих команд должна вывести порт, указанный в **postgresql.conf**. Вторая – указанные там же адреса, с которых могут приходить запросы. Третья выведет адрес локальной стороны соединения.

3. Интеграция серверов

Ещё одним важным аспектом настройки подключения к серверу СУБД Postgres является интеграция разных серверов через механизм Foreign Data Wrapper. При этом один из серверов будет работать как клиент другого. Например, сервер для аналитики данных часто является клиентом по отношению к серверу, на котором развёрнута рабочая база данных. Чтобы задействовать этот механизм, необходимо сделать следующее.

1. На сервере, где развёрнута «рабочая» база данных, откройте Query Tool и выполните команду создания нового пользователя:

```
CREATE USER имя_пользователя WITH PASSWORD 'пароль';
```

2. Предоставьте ему права на использование «рабочей» базы данных. Если имя базы данных или имя пользователя содержит прописные буквы или пробельные символы, его необходимо заключить в кавычки:

```
GRANT CONNECT ON DATABASE "имя_базы_данных"
    TO имя_пользователя;
```

3. Предоставьте этому пользователю права на использование объектов в нужной схеме базы данных:

```
GRANT USAGE ON SCHEMA public TO canteenUser;
```

В данном примере команды используется схема **public**, которая в Postgres является схемой по умолчанию. Но в реальной ситуации объекты базы данных могут быть в разных схемах.

4. Предоставьте созданному пользователю права на использование объектов базы данных CanteenDishes:

```
GRANT SELECT, INSERT, UPDATE, DELETE
    ON ALL TABLES IN SCHEMA public TO имя_пользователя;
GRANT USAGE, SELECT ON ALL SEQUENCES
    IN SCHEMA public TO имя_пользователя;
```

Если «рабочая» база данных уже заполнена данными, в первой команде можно прописать не **SELECT**, **INSERT**, **UPDATE**, **DELETE**, а только **SELECT**.

5. Проверьте, что на «рабочую» базу данных можно зайти под созданным пользователем, создав новый объект «сервер» в pgAdmin с нужными настройками подключения. Проверьте, что у пользователя есть права на те действия, которые были указаны командой **GRANT**, выполнив несколько запросов на чтение и редактирование данных.

6. На «аналитическом» сервере включите стадартное расширение для подключения к внешним источникам данных Postgres. Для этого откройте Query Tool и введите команду:

```
CREATE EXTENSION postgres_fdw;
```

Это включит расширение, которое позволяет подключаться из одной базы данных PostgreSQL к другой. Аналогичные расширения для подключения к другим СУБД нужно устанавливать отдельно.

7. Используя установленное расширение, добавьте внешнее подключение из «аналитической» базы данных к «рабочей». Для этого в «аналитической» базе данных в Query Tool введите команду:

```
CREATE SERVER my_work_server_name
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'localhost', port '5435',
    dbname 'my_work_db')
```

На место **my_work_server_name**, **my_work_db** и порта необходимо подставить желаемое имя внешнего подключения, порт и имя базы данных, которые использовались при разворачивании «рабочего» сервера.

8. Задайте сопоставление пользователя «аналитической» базы данных пользователю «рабочей» базы данных. Для этого введите команду:

```
CREATE USER MAPPING FOR postgres
SERVER my_work_server
OPTIONS(USER 'имя', password 'пароль');
```

В качестве имени пользователя и пароля нужно указать параметры пользователя, который был создан на шаге 1. Желательно в «аналитической» базе данных тоже создать отдельного пользователя и подключение для него, и в этом случае вместо пользователя **postgres** нужно указать имя созданного пользователя базы данных.

9. Создайте таблицы с внешними данными для таблиц из «рабочей» базы данных. Для этого в «рабочей» базе данных на каждой таблице выполните команду Scripts/CREATE Script из контекстного меню. В результате выполнения этой команды должен сгенерироваться скрипт создания таблицы и отобразиться в окне Query Tool (Рисунок 3).

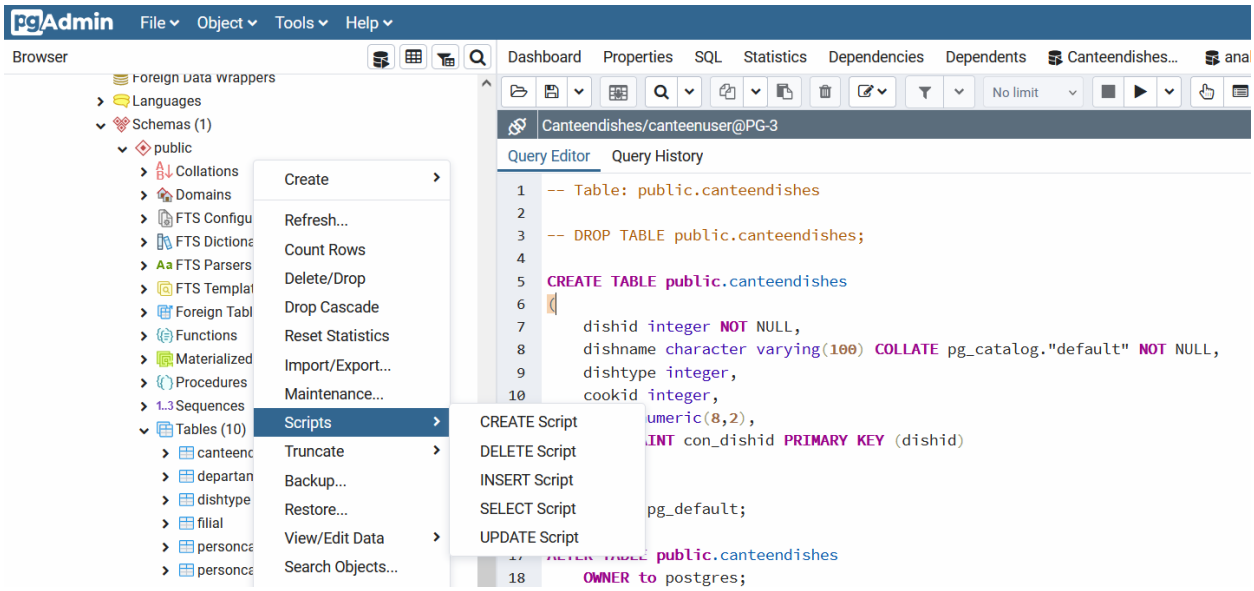

Рисунок 3. Получение скрипта создания таблицы.

Из этого скрипта необходимо скопировать определения колонок и имя таблицы. Их нужно вставить в скрипт создания внешней таблицы в «аналитической» базе данных:

```
CREATE FOREIGN TABLE имя_таблицы
(
    скопированные_определения_колонок
)
SERVER my_work_server_name
OPTIONS (schema_name 'public',
   table_name 'имя_таблицы');
```

Имя создаваемой внешней таблицы в «аналитической» базе данных может быть произвольным. Но имя таблицы, указанной в параметре **table_name** обязательно должно совпадать с именем таблицы, на которую делается ссылка. Как правило, в учебных базах данных используется схема **public**, но если в конкретной базе данных используется другая схема, в данной команде схему необходимо заменить на ту, которая используется.

10. Выполните запрос к созданной внешней таблице и убедитесь, что он проходит успешно:

```
SELECT * FROM имя_таблицы;
```

Для отслеживания того, как работают внешние подключения, можно использовать два инструмента. Во-первых, можно на «рабочей» базе данных выполнить следующую команду:

```
SELECT * FROM pg_stat_activity;
```

Эта команда выведет информацию, показанную на рисунке 4. Можно увидеть таблицу, кто, когда и с какого IP-адреса подключался к базе данных. Примерный вид этой таблицы показан на рисунке 4.

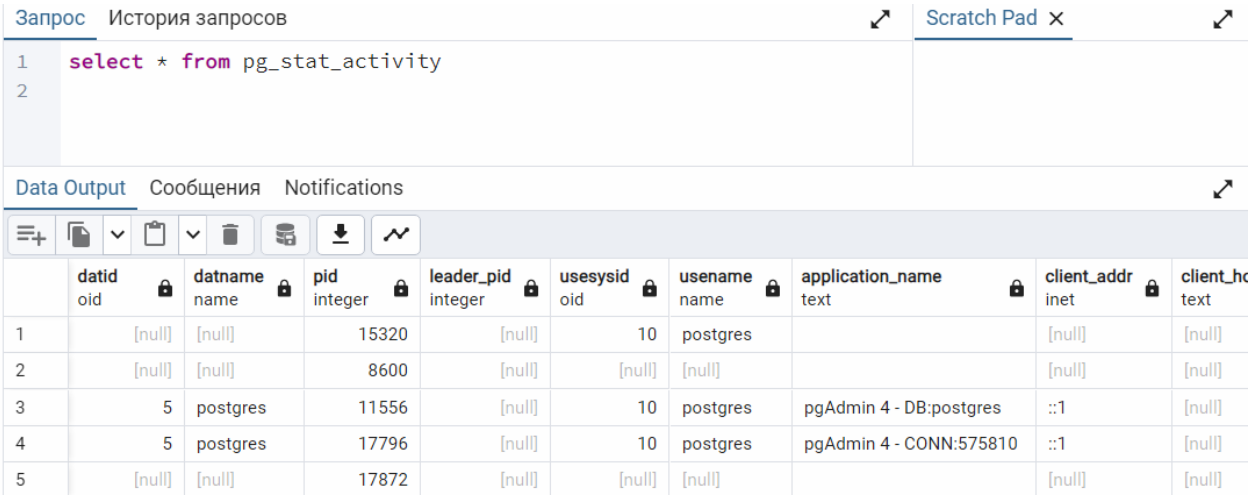

Рисунок 4. Просмотр активности сервера.

Кроме этого, можно на вкладке Dashboard посмотреть эту же информацию на вкладке «Активность», как показано на рисунке 5. Более подробные логи можно посмотреть в C:\Program Files\PostgreSQL\16\ data\log.

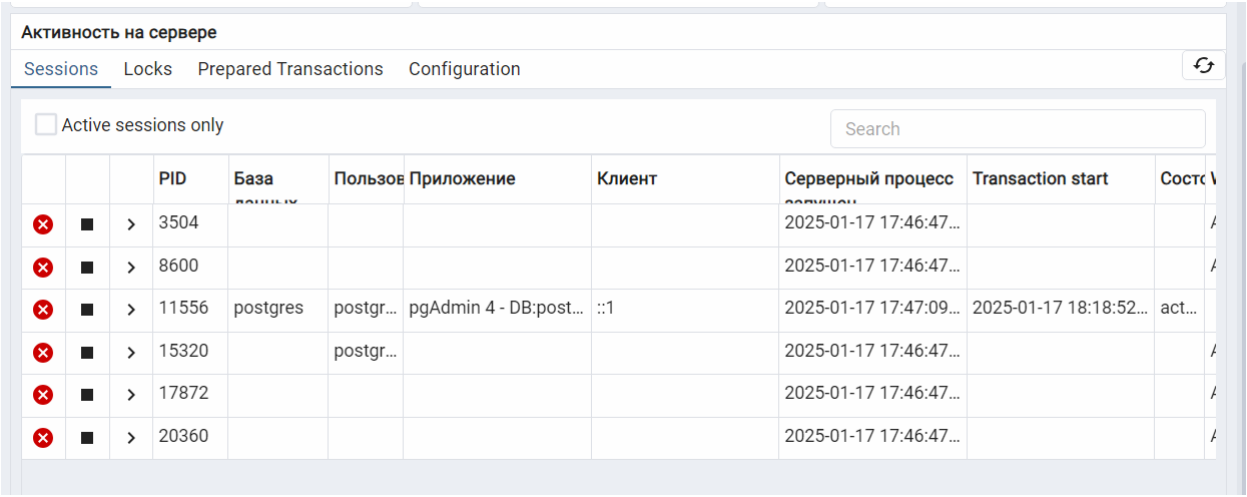

Рисунок 5. Просмотр логов в pgAdmin.

Задание

1. Создайте два сервера на разных портах и запустите их.
2. Настройте шифрование паролей на обоих серверах, желательно **`scram-sha-256`**.
3. На первом сервере разверните условную «рабочую» базу данных. Если есть база данных, заполненная данными, от курсового проекта по курсу «Управление данными», то рекомендуется взять её. В противном случае можно взять одну из учебных баз данных Canteen Dishes или Novosibirsk.
4. На втором сервере создайте «аналитическую» базу данных. В ней нужно создать произвольную таблицу с суррогатным ключом. Эта таблица по семантике должна дополнять «рабочую» базу данных.
5. Создайте в «рабочей» базе данных пользователя с правами на чтение из любых таблиц.
6. В «аналитической» базе данных создайте пользователя, который может читать и редактировать данные в таблице, созданной на шаге 4, и связать его с пользователем «рабочей» базы данных.
7. Создайте в «аналитической» базе данных внешние таблицы для таблиц «рабочей» базы данных.
8. Создайте в «аналитической» базе данных представление, которое берёт данные из внешних таблиц и таблицы, созданной на шаге 4.
9. Зайдите в «аналитическую» базу данных из-под созданного для неё пользователя и убедитесь, что запрос к созданному на шаге 8 представлению успешно проходит.
10. Зайдите в «рабочую» базу данных и посмотреть с использованием Dashboard и **`pg_stat_activity`**, внешнее подключение к серверу.
11. Измените данные в «рабочей» базе данных и убедитесь, что изменения видны в «аналитической» базе данных.

Оформление отчёта

Отчёт должен иметь титульный лист с указанием ФИО и группы студента, а так же темы практической работы. После этого должен быть скриншот командной строки со списком занятых портов и скриншоты командной строки с созданием серверов и их запуском. Затем должны быть указаны порты и настройки внешних подключений к созданным серверам с шифрованием (выписки из конфигурационных файлов). Затем должны быть указаны имена баз данных и скрипты создания новых пользователей на обоих серверах с предоставлением им необходимых прав. Кроме того, должны быть представлены скриншоты, показывающие, что подключение к обеим базам данных проходит успешно, и у пользователя «рабочей» базы данных есть права на чтение, но не редактирование. Затем должны быть приведены скрипты: какая таблица была создана в «аналитической» базе данных, какие внешние таблицы были созданы, и какое было из всего этого создано представление. На скриншоте должны быть видны данные, отображающиеся в этом представлении. Затем на скриншотах должно быть показано, что на «рабочей» базе данных видны подключения из «аналитической» базы данных. Ещё в отчёте нужно привести запрос на изменение данных в «рабочей» базе данных и скриншот, на котором видно, что данные в представлении в «аналитической» базе данных изменились.