Самостоятельная работа 1. Интеграция данных из разных источников (баз данных)

Тема: разработка ETL-процесса для интеграции данных между PostgreSQL и MySQL с использованием Pentaho Data Integration.

Задачи:

- Создать исходные таблицы в PostgreSQL с различными наборами данных.
- Настроить целевые таблицы в MySQL для приема данных.
- Разработать процессы трансформации данных в Pentaho.
- Реализовать механизмы обработки ошибок и валидации данных.
- Создать представления для связанных данных.

Необходимое ПО:

- Конфигурация **devops_dba_25.ova, можно скачать в разделе** **http://95.131.149.21/moodle/mod/folder/view.php?id=1429**

- учетная запись в Mysql.

Теоретическая часть

Особенности работы с различными типами данных при переносе между СУБД.

Методы валидации и очистки данных в процессе ETL.

Техники оптимизации производительности при массовой загрузке.

Способы обработки несоответствий в структурах данных.

Особенности работы с различными типами данных при переносе между СУБД PostgreSQL и MySQL

1. Типы данных: PostgreSQL поддерживает более сложные типы данных, такие как hstore, jsonb, и пользовательские типы, в то время как MySQL имеет более ограниченный набор типов. При переносе данных может потребоваться преобразование некоторых типов.

2. Дата и время: В PostgreSQL есть типы timestamp и timestamptz, которые учитывают часовые пояса, тогда как в MySQL это делается через отдельное поле или функции.

3. Серверные настройки: Настройки сервера (например, кодировка) могут отличаться, что влияет на хранение и обработку строковых данных.

Методы валидации и очистки данных в процессе ETL (Pentaho)

1. Валидация данных: Использование шагов в Pentaho Data Integration (PDI), таких как "Filter rows", "Validator" и "Data Validator", для проверки соответствия данных определенным правилам.

2. Очистка данных: Применение шагов "Replace in string", "Regex evaluation" и "String operations" для удаления нежелательных символов, нормализации данных и т.д.

3. Обработка ошибок: Использование шагов "Error handling" и "Exception handling" для управления ошибками и несоответствиями в данных.

Техники оптимизации производительности при массовой загрузке

1. Использование Bulk Load: использование специальных инструментов для массовой загрузки данных, таких как COPY в PostgreSQL и LOAD DATA INFILE в MySQL, которые работают быстрее обычных INSERT-запросов.

2. Отключение индексов и ограничений: Отключение индексов и внешних ключей перед загрузкой данных и их последующее включение после завершения загрузки для ускорения процесса.

3. Параллельная загрузка: Разделение данных на части и загрузка их параллельно для ускорения процесса.

Способы обработки несоответствий в структурах данных

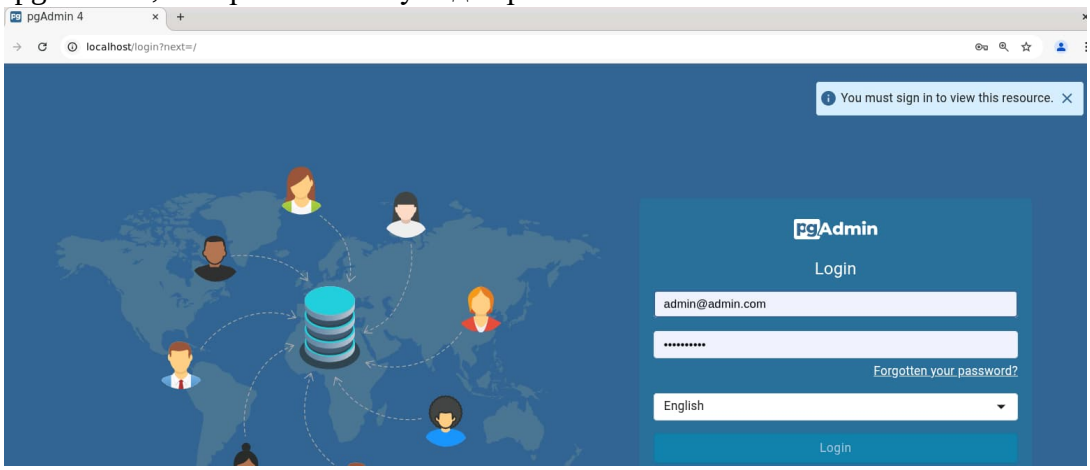
1. **Миграция схемы**: использование инструментов миграции схемы, таких как pg_dump и mysqldump, для создания скриптов создания таблиц и последующего ручного редактирования этих скриптов для адаптации к новой СУБД.

2. **Преобразование данных**: использование ETL-инструментов для преобразования данных из одной структуры в другую, например, объединение нескольких столбцов в один или разбиение одного столбца на несколько.

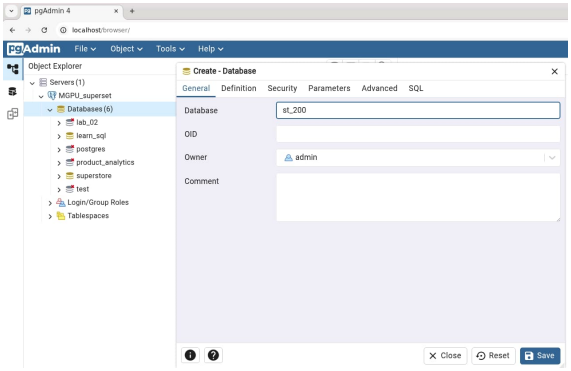
3. Ручная корректировка: в некоторых случаях может потребоваться ручная корректировка структуры данных после автоматической миграции для решения проблем, связанных с несоответствиями в типах данных или структуре таблиц.

Ход работы

Шаг 1. Проверить доступность СУБД Postgre SQL (локальная СУБД). Выполнение задания лучше перенести в pgAdmin4, который использует для работы HTTP.



Шаг 2. Создать базу данных со своим уникальным логином id.

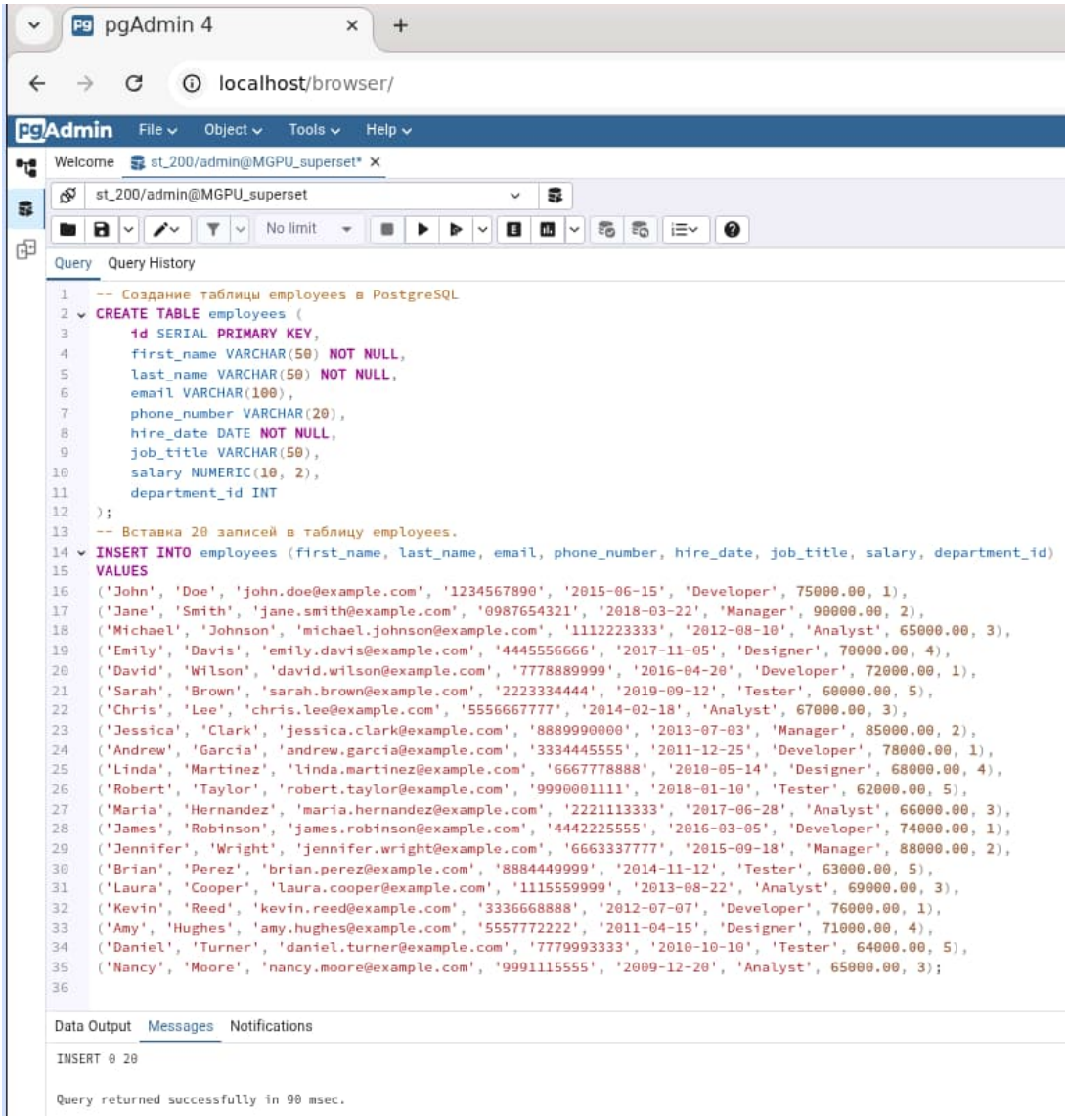


Шаг 3. Создать таблицу и данные, согласно вашего варианта. Для примера показано создание таблицы **employees**.

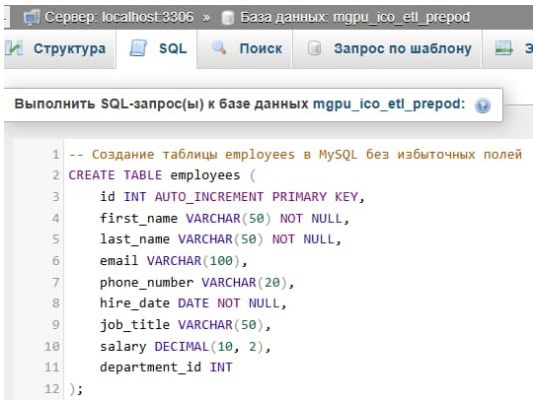
SQL-скрипт создания таблицы employees в PostgreSQL:

```
-- Создание таблицы employees в PostgreSQL
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100),
    phone_number VARCHAR(20),
    hire_date DATE NOT NULL,
    job_title VARCHAR(50),
    salary NUMERIC(10, 2),
    department_id INT
);
-- Вставка 20 записей в таблицу employees.
INSERT INTO employees (first_name, last_name, email, phone_number, hire_date, job_title, salary,
department_id)
```

VALUES
('John', 'Doe', 'john.doe@example.com', '1234567890', '2015-06-15', 'Developer', 75000.00, 1),
('Jane', 'Smith', 'jane.smith@example.com', '0987654321', '2018-03-22', 'Manager', 90000.00, 2),
('Michael', 'Johnson', 'michael.johnson@example.com', '1112223333', '2012-08-10', 'Analyst', 65000.00, 3),
('Emily', 'Davis', 'emily.davis@example.com', '4445556666', '2017-11-05', 'Designer', 70000.00, 4),
('David', 'Wilson', 'david.wilson@example.com', '7778889999', '2016-04-20', 'Developer', 72000.00, 1),
('Sarah', 'Brown', 'sarah.brown@example.com', '2223334444', '2019-09-12', 'Tester', 60000.00, 5),
('Chris', 'Lee', 'chris.lee@example.com', '5556667777', '2014-02-18', 'Analyst', 67000.00, 3),
('Jessica', 'Clark', 'jessica.clark@example.com', '8889990000', '2013-07-03', 'Manager', 85000.00, 2),
('Andrew', 'Garcia', 'andrew.garcia@example.com', '3334445555', '2011-12-25', 'Developer', 78000.00, 1),
('Linda', 'Martinez', 'linda.martinez@example.com', '6667778888', '2010-05-14', 'Designer', 68000.00, 4),
('Robert', 'Taylor', 'robert.taylor@example.com', '9990001111', '2018-01-10', 'Tester', 62000.00, 5),
('Maria', 'Hernandez', 'maria.hernandez@example.com', '2221113333', '2017-06-28', 'Analyst', 66000.00, 3),
('James', 'Robinson', 'james.robinson@example.com', '4442225555', '2016-03-05', 'Developer', 74000.00, 1),
('Jennifer', 'Wright', 'jennifer.wright@example.com', '6663337777', '2015-09-18', 'Manager', 88000.00, 2),
('Brian', 'Perez', 'brian.perez@example.com', '8884449999', '2014-11-12', 'Tester', 63000.00, 5),
('Laura', 'Cooper', 'laura.cooper@example.com', '1115559999', '2013-08-22', 'Analyst', 69000.00, 3),
('Kevin', 'Reed', 'kevin.reed@example.com', '3336668888', '2012-07-07', 'Developer', 76000.00, 1),
('Amy', 'Hughes', 'amy.hughes@example.com', '5557772222', '2011-04-15', 'Designer', 71000.00, 4),
('Daniel', 'Turner', 'daniel.turner@example.com', '7779993333', '2010-10-10', 'Tester', 64000.00, 5),
('Nancy', 'Moore', 'nancy.moore@example.com', '9991115555', '2009-12-20', 'Analyst', 65000.00, 3);



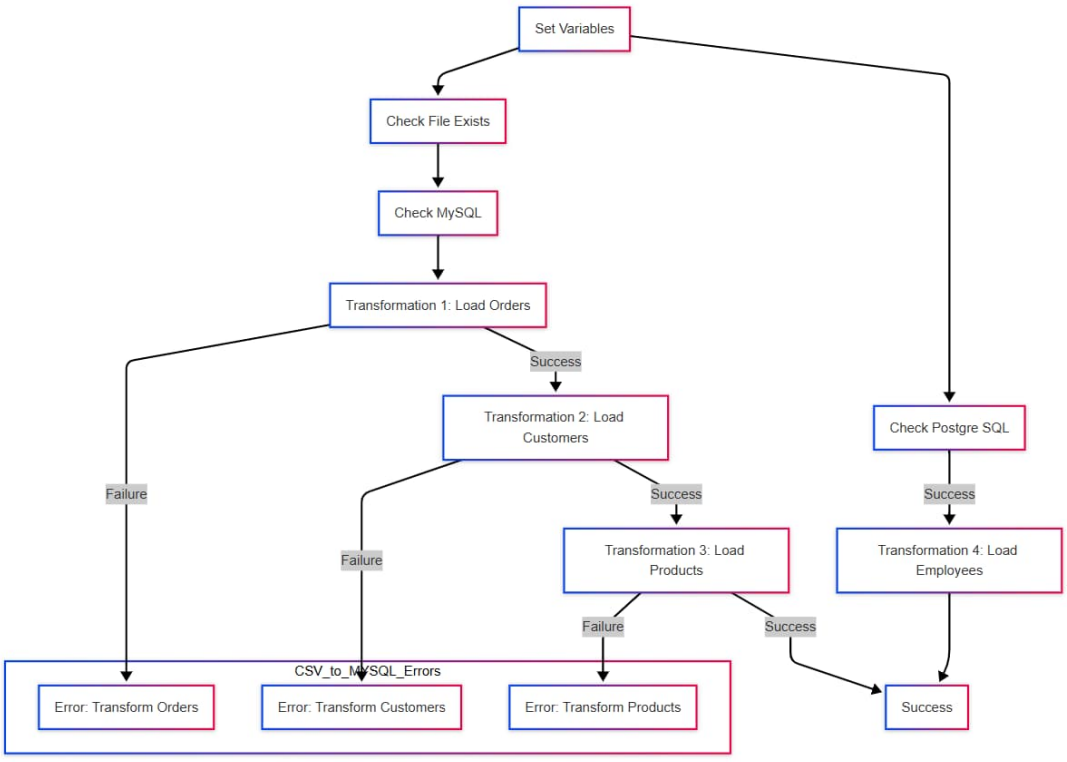
Шаг 4. Проверить сетевой доступ к целевой СУБД Mysql (http://95.131.149.21/). Создать целевую таблицу employees в СУБД Mysql.



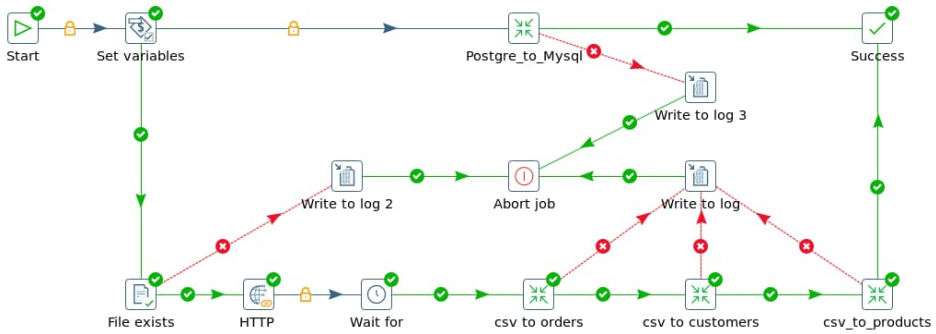
-- Создание таблицы employees в MySQL без избыточных полей

```
CREATE TABLE employees (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100),
    phone_number VARCHAR(20),
    hire_date DATE NOT NULL,
    job_title VARCHAR(50),
    salary DECIMAL(10, 2),
    department_id INT
);
```

Шаг 5. Построить ETL-процесс в Pentaho, согласно графа данных, представленного ниже. Выполнить трансформацию с таблицей полученной из Postgre SQL (агрегация полей, фильтр и т.д согласно вариант).



JOB трансформации представлен следующим графом:



Start
- **Set Variables (FILE_PATH)**
- **Check File Exists**
- **HTTP Download**
- **Transform 1: Load Orders**
- **Transform 2: Load Customers**
- **Transform 3: Load Products**
- **Transformation 4: Load Employees**

Success

Настройка подключения к PostgreSQL.

1. Откройте **Pentaho Data Integration (PDI)**.
2. В меню **View** выберите **Database Connections** (или Get **Table Input** в более новых версиях).
3. Нажмите правой кнопкой мыши на **Database Connections** и выберите **New connection**.
4. Заполните поля для подключения к PostgreSQL:

   Connection Name: установить имя подключению, например, **Postgres_Connection**.

   Database Type: выберите PostgreSQL.

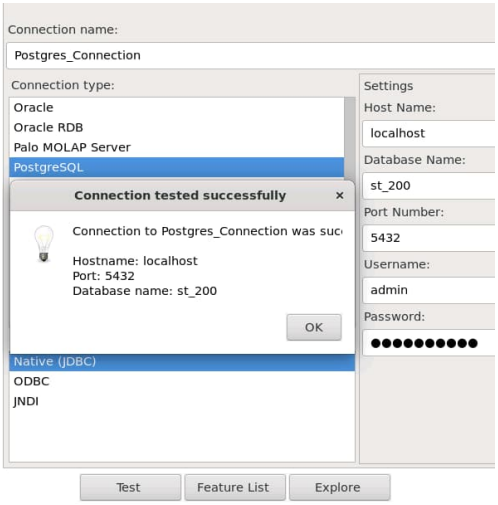
   Host Name: укажите адрес сервера PostgreSQL - **localhost**.

   Database Name: укажите название базы данных **ваш id Шаг 2.**

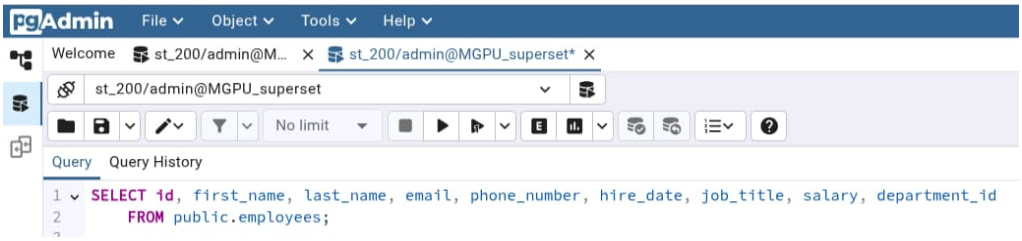
   Port: порт **5432**.

   Username и **Password**: **указаны в docker composer запуска СУБД или в заметках к выпуску конфигурации** для подключения к базе данных PostgreSQL.

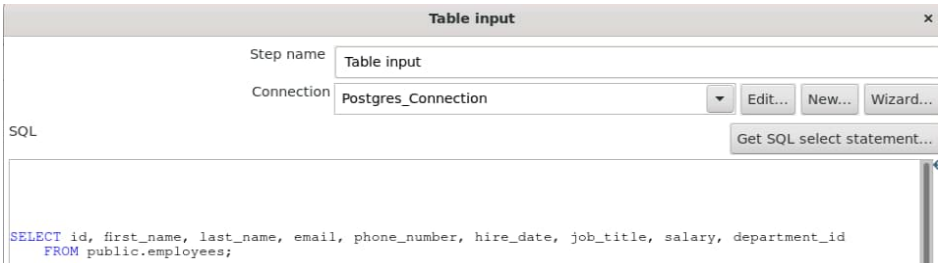
Нажмите **Test** для проверки соединения, и затем нажмите **OK**.



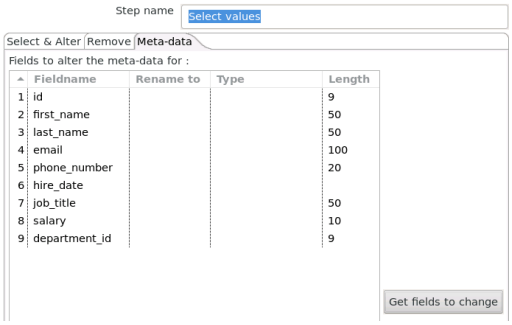
Чтоб сгенерировать запрос, требуется предварительно его получить в Postgre



Переносим его в Pentaho объект **Table Input**:



Импортируем все атрибуты таблицы в объект **Select values, преобразовываем поля в закладке Meta-data.**



Далее выполняем внутренние трансформации с данными согласно варианту.

В итоге, обязательными элементами являются коннекторы к базам данных и трансформации согласно Вашему варианту.

Согласовать данные из Transformation 4: Load Employees и Transform 2: Load Customers путем создания представления в СУБД Mysql представления, где указаны через JOIN только те сотрудники, которые работают в компании.

Варианты индивидуальных заданий

Ознакомиться и приступить к заданию можно, перейдя по ссылке

https://docs.google.com/spreadsheets/d/1lyZ19LejTW9TDW9-9cEf4hcQIf-mwKSQNAUebMb5bk4/edit?usp=sharing

Требования к электронному отчету

Отчет прикрепить одним файлом. Все трансформации, схемы баз данных (в Postgresql и Mysql) выгрузить в Git-репозиторий, в ответе предоставить только ссылку на репозиторий.

1. Титульный лист.
2. Цель и задачи работы.
3. Описание архитектуры решения.
4. Скриншоты настроек компонентов.
5. SQL-скрипты создания структур данных.
6. Примеры обработанных данных. Схему трансформаций в Pentaho.
7. Выводы.

Критерии оценки

Каждое задание оценивается в один балл.