

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ОТЧЕТ

по дисциплине «Проектный практикум по разработке ETL-решений»
Самостоятельная работа 1. Интеграция данных из разных источников (баз
данных)

Направление подготовки – 38.03.05 «Бизнес-информатика».
профиль подготовки – «Аналитика данных и эффективное управление»

Выполнила:

студентка группы АДЭУ-211
st92

Руководитель:

,
Кандидат технических наук, доцент

Москва
2025 год

Цель работы: изучить разработку ETL-процесса для интеграции данных между PostgreSQL и MySQL с использованием Pentaho Data Integration

Задачи:

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

Ход работы:

Шаг 1. Проверим доступность СУБД PostgreSQL (локальная СУБД).

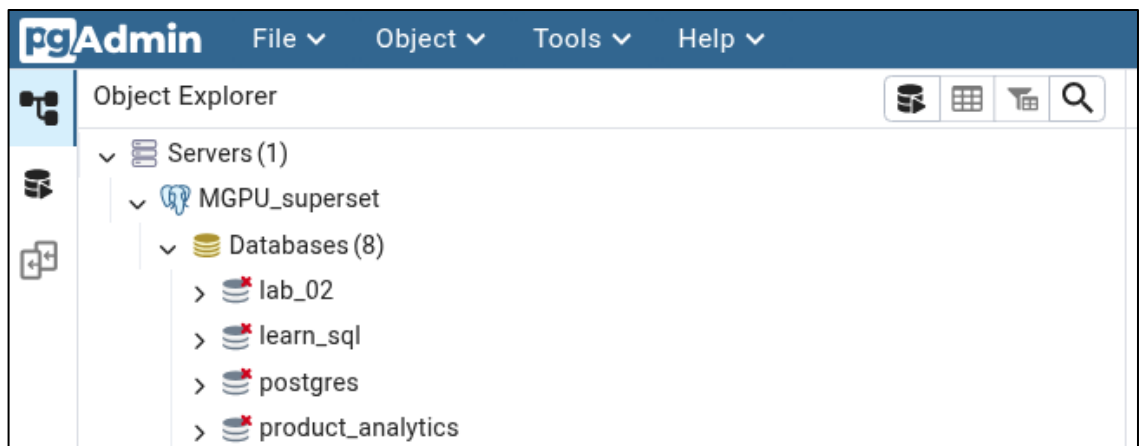


Рисунок 1 – Доступность СУБД Postgres

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

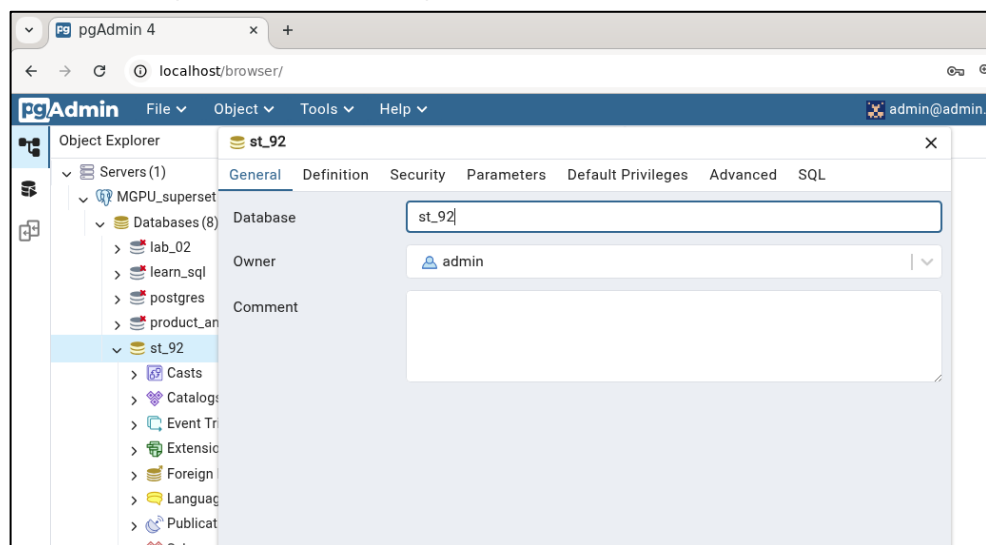
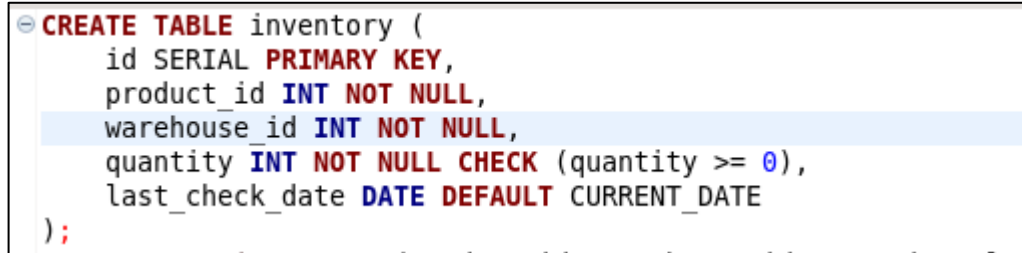


Рисунок 2 – Созданная таблица st_92

Шаг 3. Создать таблицу и данные, согласно вашего варианта
Создать таблицу inventory (id, product_id, warehouse_id, quantity, last_check_date)

```
CREATE TABLE inventory (  
    id SERIAL PRIMARY KEY,  
    product_id INT NOT NULL,  
    warehouse_id INT NOT NULL,  
    quantity INT NOT NULL CHECK (quantity >= 0),  
    last_check_date DATE DEFAULT CURRENT_DATE
```



```
CREATE TABLE inventory (  
    id SERIAL PRIMARY KEY,  
    product_id INT NOT NULL,  
    warehouse_id INT NOT NULL,  
    quantity INT NOT NULL CHECK (quantity >= 0),  
    last_check_date DATE DEFAULT CURRENT_DATE  
);
```

Рисунок 3 – Создание таблицы inventory

Вставим 30 записей в таблицу inventory:

```
INSERT INTO inventory (product_id, warehouse_id, quantity,  
last_check_date)
```

VALUES

```
(1, 1, 100, '2023-10-01'),  
(2, 1, 50, '2023-10-02'),  
(3, 2, 200, '2023-10-03'),  
(4, 2, 75, '2023-10-04'),  
(5, 3, 150, '2023-10-05'),  
(6, 3, 300, '2023-10-06'),  
(7, 4, 120, '2023-10-07'),  
(8, 4, 90, '2023-10-08'),  
(9, 5, 250, '2023-10-09'),  
(10, 5, 60, '2023-10-10'),  
(11, 6, 180, '2023-10-11'),  
(12, 6, 40, '2023-10-12'),  
(13, 7, 220, '2023-10-13'),  
(14, 7, 80, '2023-10-14'),  
(15, 8, 150, '2023-10-15'),  
(16, 8, 70, '2023-10-16'),  
(17, 9, 300, '2023-10-17'),  
(18, 9, 110, '2023-10-18'),  
(19, 10, 200, '2023-10-19'),
```

(20, 10, 50, '2023-10-20');

```
INSERT INTO inventory (product_id, warehouse_id, quantity, last_check_date)
VALUES
(1, 1, 41, '2023-10-01'),
(2, 1, 50, '2023-10-02'),
(3, 2, 34, '2023-10-03'),
(4, 2, 75, '2023-10-04'),
(5, 3, 28, '2023-10-05'),
(6, 3, 11, '2023-10-06'),
(7, 4, 22, '2023-10-07'),
(8, 4, 90, '2023-10-08'),
(9, 5, 15, '2023-10-09'),
(10, 5, 60, '2023-10-10'),
(11, 6, 18, '2023-10-11'),
(12, 6, 40, '2023-10-12'),
(13, 7, 11, '2023-10-13'),
(14, 7, 80, '2023-10-14'),
(15, 8, 150, '2023-10-15'),
(16, 8, 40, '2023-10-16'),
(17, 9, 55, '2023-10-17'),
(18, 9, 110, '2023-10-18'),
(19, 10, 6, '2023-10-19'),
(20, 10, 50, '2023-10-20'),
(21, 3, 34, '2023-10-11'),
(22, 6, 40, '2023-10-12'),
(23, 7, 23, '2023-10-13'),
(24, 7, 80, '2023-10-14'),
(25, 8, 141, '2023-10-15'),
(26, 8, 70, '2023-10-16'),
(27, 9, 10, '2023-10-17'),
(28, 9, 195, '2023-10-18'),
(29, 10, 49, '2023-10-19'),
(30, 10, 4, '2023-10-20');
```

Рисунок 4 – Вставка 30 записей в таблицу inventory

Шаг 4. Проверим сетевой доступ к целевой СУБД Mysql
(<http://95.131.149.21/>).

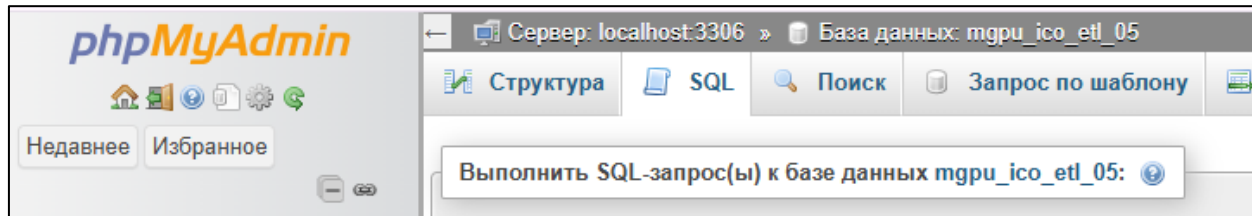


Рисунок 5 – Доступность целевой СУБД

Создадим целевую таблицу *stock_levels* в СУБД Mysql.

```
CREATE TABLE stock_levels (
    min_quantity INT NOT NULL CHECK (min_quantity >= 0),
    max_quantity INT NOT NULL CHECK (max_quantity >= 0)
);
```

✓ MySQL вернула пустой результат (т.е. ноль строк). (Запрос занял 0.0003 сек.)

```
CREATE TABLE stock_levels ( min_quantity INT NOT NULL CHECK (min_quantity >= 0), max_quantity INT NOT NULL CHECK (max_quantity >= 0) );
```

Рисунок 6 – Создание таблицы stock_levels

Шаг 5. Построим ETL-процесс в Pentaho, согласно графу данных, представленного ниже. Выполним трансформацию с таблицей,

полученной из Postgre SQL (агрегация полей, фильтр и т.д согласно варианту).

Создадим таблицу для хранения данных о критическом балансе

```
CREATE TABLE critical_balances (  
    id SERIAL PRIMARY KEY,  
    product_id INT NOT NULL,  
    warehouse_id INT NOT NULL,  
    quantity INT NOT NULL  
);
```

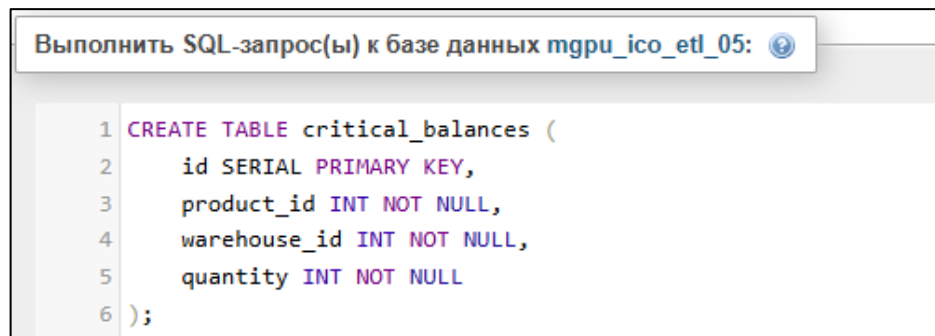


Рисунок 7 – Создание таблицы critical_balances

Создадим таблицу для хранения данных о количестве необходимых к закупке товаров

```
CREATE TABLE replenishment (  
    id SERIAL PRIMARY KEY,  
    product_id INT NOT NULL,  
    warehouse_id INT NOT NULL,  
    quantity INT NOT NULL,  
    to_replenish INT NOT NULL  
);
```

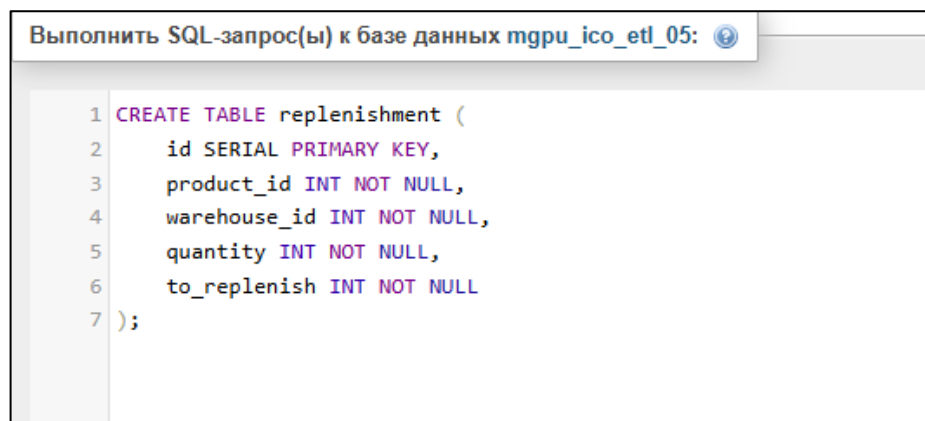


Рисунок 8 – Создание таблицы replenishment

Создадим таблицу для хранения данных о сумме количества товара по складам

```
CREATE TABLE remaining_warehouses (  
    warehouse_id INT NOT NULL,  
    sum_quantity INT NOT NULL  
);
```

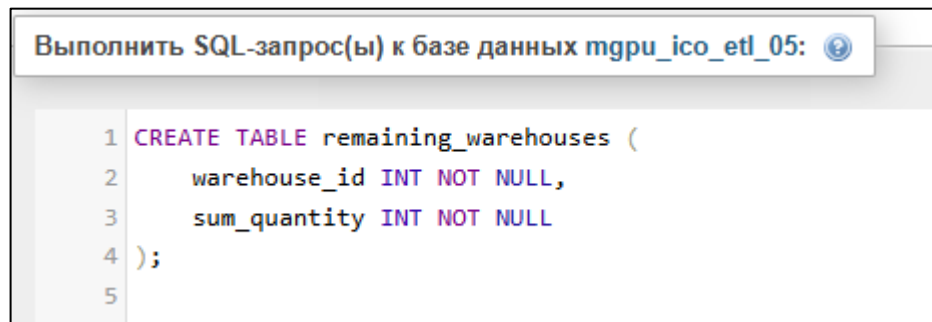


Рисунок 9 – Создание таблицы remaining_warehouses

Создадим трансформацию для передачи таблицы из PostgreSQL в MySQL

Добавим элемент Table input с необходимыми настройками

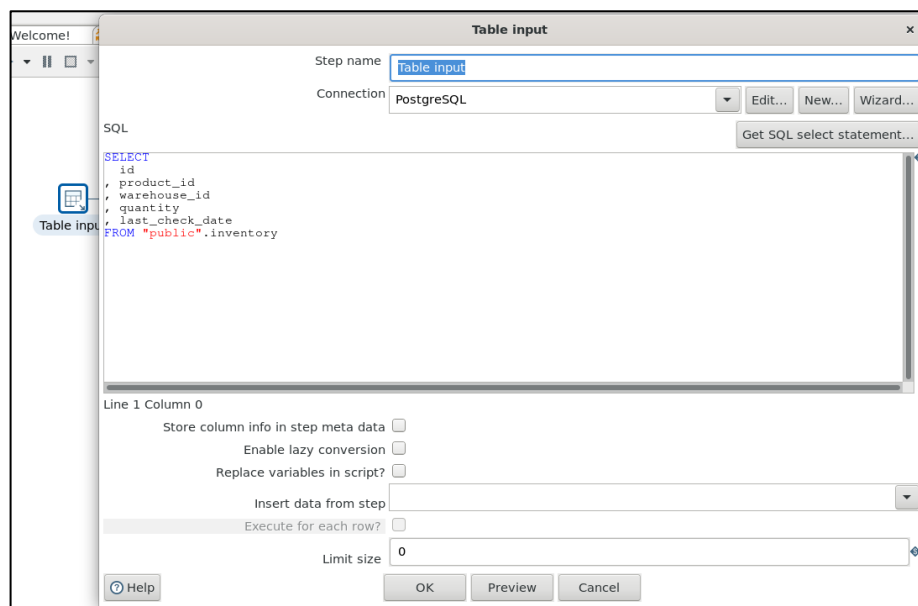


Рисунок 10 – Настройки объекта Table input

Добавим объект Select values с необходимыми настройками

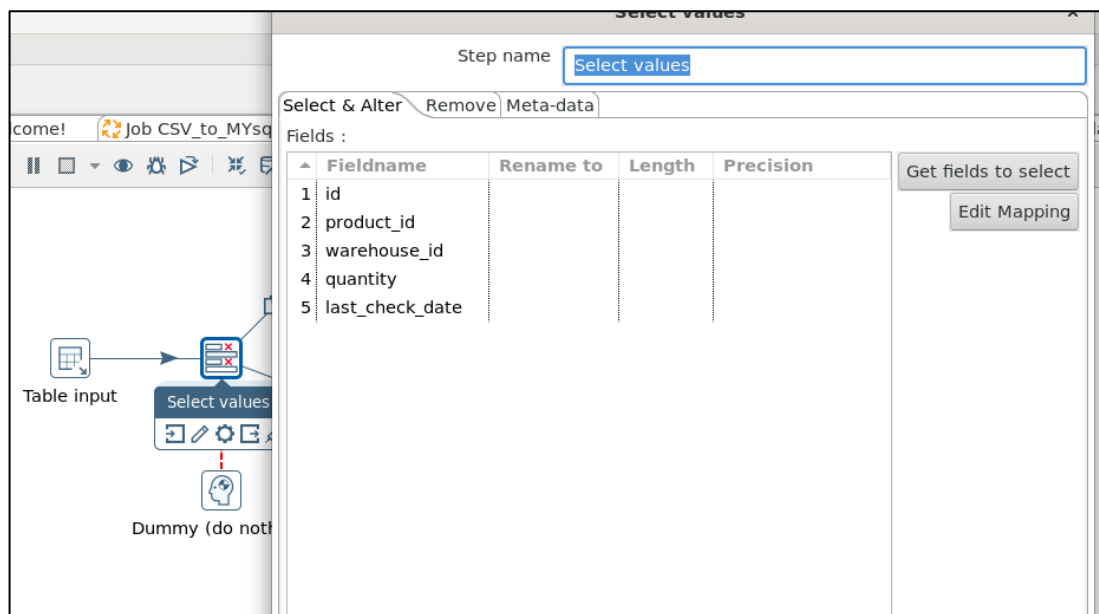


Рисунок 11 – Настройки объекта Select values

Проведем Фильтрацию товаров с критическим остатком:

Добавим объект Filter rows со следующими настройками

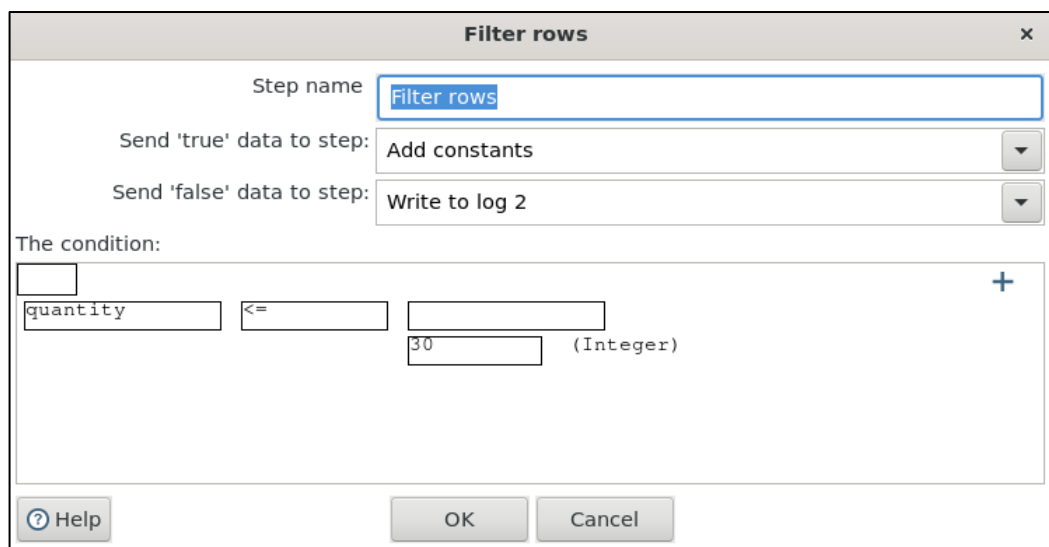


Рисунок 12 – Настройка объекта Filter rows

Подсчитаем Суммарные остатки по складам:

Добавим объект Group By

Group by

Step name:

Include all rows? ☐

Temporary files directory:

TMP-file prefix:

Add line number, restart in each group ☐

Line number field name:

Always give back a result row ☐

The fields that make up the group:

Group field	
1	warehouse_id

Aggregates :

	Name	Subject	Type	Value
1	sum_quantity	quantity	Sum	

Рисунок 13 – Настройки объекта Group by warehouse_id

Добавим объект Table output для вывода данных в таблицу MySQL

Table output

Step name:

Connection:

Target schema:

Target table:

Commit size:

Truncate table ☐

Ignore insert errors ☐

Specify database fields ☒

Main options: **Database fields**

Fields to insert:

	Table field	Stream field
1	warehouse_id	warehouse_id
2	sum_quantity	sum_quantity

Рисунок 14 – Настройка объекта Table output 4

Расчитаем необходимости пополнения склада

Добавим объект add constants, укажем пороговое значение для количества = 45

Add constants

Step name:

Fields :

	Name	Type	Format	Length	Precision	Currency	Decimal	Group	Value	Set em
1	constant_of_quantity	Number	0.00						45	N

Рисунок 15 – Настройки Add constants

Добавим калькулятор для расчета потребности в закупке товара на склад

Calculator

Step name

Calculator

☒ Throw an error on non existing files

Fields:

	New field	Calculation	Field A	Field B	Field C	Value type	Length	Precision
1	to_replenish	A - B	constant_of_quantity	quantity		None		

Help

Рисунок 16 – Настройки объекта Calculator

Добавим объект Select values и выберем необходимые столбцы для таблицы потребностей

Select values

Step name

Select values 3

Select & Alter Remove Meta-data

Fields :

	Fieldname	Rename to	Length	Precision
1	id			
2	product_id			
3	warehouse_id			
4	quantity			
5	to_replenish			

Get fields to select

Edit Mapping

Рисунок 17 – Настройка объекта Select values 3

Добавим объект Table output для выгрузки таблицы в базу данных

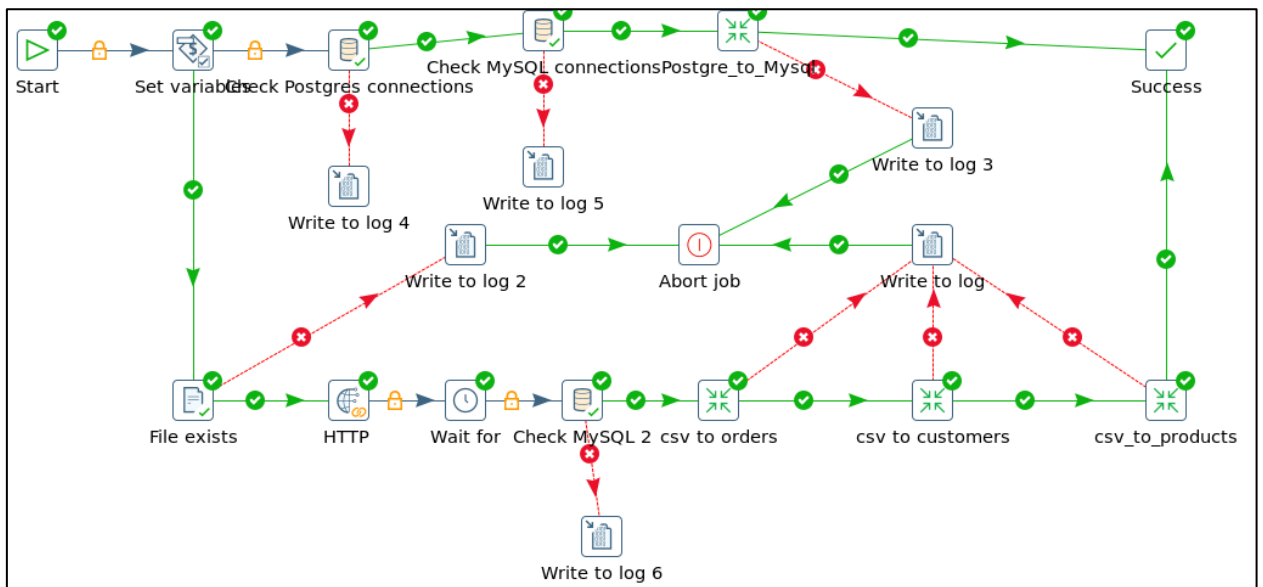


Рисунок 20 – Рабочая область Job CSV_to_MySQL

Убедимся, что таблицы успешно загружены в базу данных

id	product_id	warehouse_id	quantity
5	5	3	28
6	6	3	11
7	7	4	22
9	9	5	15
11	11	6	18
13	13	7	11
19	19	10	6
23	23	7	23
27	27	9	10
30	30	10	4

Рисунок 21 – Таблица critical_balances

warehouse_id	sum_quantity
3	39
4	22
5	15
6	18
7	11
10	6
7	23
9	10
10	4

Рисунок 22 – Таблица remaining_warehouses

Сервер: localhost:3306 » База данных: mgru_ico_etl_05 » Таблица: replenishment

Обзор Структура SQL Поиск Вставить Экспорт Импорт Операции Слежение Триггеры

Отображение строк 0 - 9 (10 всего, Запрос занял 0.0002 сек.)

SELECT * FROM `replenishment`

Профилирование [Построчное редактирование] [Изменить] [Анализ SQL запроса] [Создать PHP-код] [Обновить]

Показать все Количество строк: 25 Фильтровать строки: Поиск в таблице Сортировать по ключу: Ни одного

Extra options

			id	product_id	warehouse_id	quantity	to_replenish
<input type="checkbox"/>	Изменить	Копировать	Удалить	5	5	3	28
<input type="checkbox"/>	Изменить	Копировать	Удалить	6	6	3	34
<input type="checkbox"/>	Изменить	Копировать	Удалить	7	7	4	23
<input type="checkbox"/>	Изменить	Копировать	Удалить	9	9	5	30
<input type="checkbox"/>	Изменить	Копировать	Удалить	11	11	6	27
<input type="checkbox"/>	Изменить	Копировать	Удалить	13	13	7	34
<input type="checkbox"/>	Изменить	Копировать	Удалить	19	19	10	39
<input type="checkbox"/>	Изменить	Копировать	Удалить	23	23	7	22
<input type="checkbox"/>	Изменить	Копировать	Удалить	27	27	9	35
<input type="checkbox"/>	Изменить	Копировать	Удалить	30	30	10	41

Рисунок 23 – Таблица replenishment

Сервер: localhost:3306 » База данных: mgru_ico_etl_05 » Таблица: stock_levels

Обзор Структура SQL Поиск Вставить Экспорт Импорт Операции

Данное выделение не содержит уникального столбца. Изменение сетки, выставление галочки, редактирование, копирование и у

Отображение строк 0 - 0 (1 всего, Запрос занял 0.0001 сек.)

SELECT * FROM `stock_levels`

Профилирование [Построчное редактирование] [Изменить] [Анализ SQL запроса] [Создать PHP-код] [Обновить]

Показать все Количество строк: 25 Фильтровать строки: Поиск в таблице

Extra options

min_quantity	max_quantity
4	195

Показать все Количество строк: 25 Фильтровать строки: Поиск в таблице

Использование результатов запроса

Рисунок 24 – Таблица stock_levels

Выводы:

В ходе работы мы изучили разработку ETL-процесса для интеграции данных между PostgreSQL и MySQL с использованием Pentaho Data Integration.