**Итоговая работа**

**по курсу «DWH** (**Data Warehouse)»  
студента группы DWH-8 (DEG-7)  
Синельниковой Наталии Леонидовны**

# Запуск БД

В работе использовался удаленный тип подключения к базе данных, используя ПО Docker Desktop.

Пустая БД была запущена при помощи следующего скрипта в командной сроке:

docker run --name netology\_DWH -p "5433:5432" -e POSTGRES\_PASSWORD=DWH123 -e POSTGRES\_USER=DWH -e POSTGRES\_DB=DWH-db-final -d postgres:12.8

Далее были созданы схема dim с таблицами измерений:

* dim.calendar - справочник дат
* dim.passengers - справочник пассажиров
* dim.aircrafts - справочник самолетов
* dim.airports - справочник аэропортов
* dim.tariff - справочник тарифов

схема fact с таблицей фактов

* fact.flights - содержит совершенные перелеты.

и схема rejected

-Поля данных rejected-таблицы идентичны полям таблиц справочников, за исключением:

1. Поля не содержат PRIMARY KEY и FOREIGN KEY
2. Поля не содержат какие либо ограничения
3. Тип автоматически сформированных полей serial изменен на int4
4. Добавлено поле reason\_for\_rejection, в которое будет помещена причина отсеивания данных в rejected-таблицы.

Диаграмма базы данных представлена на рисунке 1.

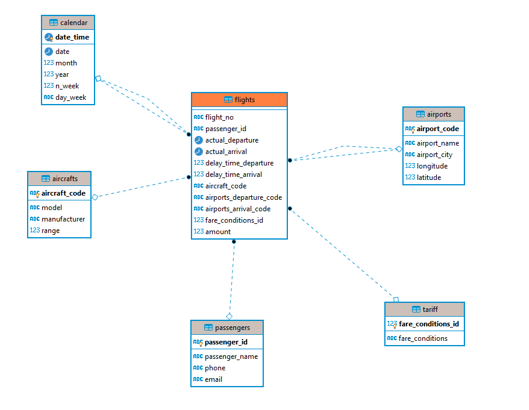


Рисунок 1 Диаграмма БД

Заполнение таблиц производилось с использованием локальной базы данных Bookings. Для написания ETL-трансформаций использовано ПО Pentaho Data Integration.

# Описание и наполнение таблиц базы данных

## Таблица dim.calendar

Таблица представляет собой справочник дат и времени совершения вылетов самолетов. Гранулярность – 1мин. Тип SCD – 0, подразумеваем, что значения в таблице не изменяются.

Таблица создана с помощью SQL запроса

**CREATE** **TABLE** dim.calendar (

date\_time **timestamptz** **PRIMARY** **KEY** **NOT** **NULL**, -- дата-время - ключ

"date" **date** **NOT** **NULL** , -- дата

"month" **int4** **NOT** **NULL** , -- месяц

**YEAR** **int4** **NOT** **NULL**, -- год

n\_week **int4** **NOT** **NULL**, -- неделя

day\_week **varchar**(10) **NOT** **NULL**-- день недели

);

Таблица содержит поля:

date\_time – дата и время в формате timestamptz, также являющаяся ключом;

date – дата в формате date;

month – номер месяца;

year – номер года;

n\_week – номер недели в году;

day\_week – день недели

Наполнение таблицы произведено при помощи SQL-запроса:

**INSERT** **INTO** dim.calendar(date\_time, "date", "month", "year", n\_week, day\_week)

**SELECT** gs **AS** date\_time

, gs::**date**, **date\_part**('month', gs)

, **date\_part**('year', gs)

, **date\_part**('week', gs)

, **to\_char**(gs, 'day')

**FROM** **generate\_series**('2016-09-13', **current\_date**, **interval** '1 minute') **as** gs;

## Таблица dim\_passengers

Таблица представляет собой справочник пассажиров. Гранулярность – один пассажир. Таблица содержит поля:

passenger\_id – идентификатор пассажира в исходной таблице, валяется ключом;

passenger\_name – ФИО пассажира;

phone – контактный телефон;

email – адрес электронной почты.

Запрос SQL на создание таблицы

**CREATE** **TABLE** dim.passengers (

passenger\_id **varchar**(20) **PRIMARY** **KEY**, -- ключ пасажира (bookings.tickets.passenger\_id)

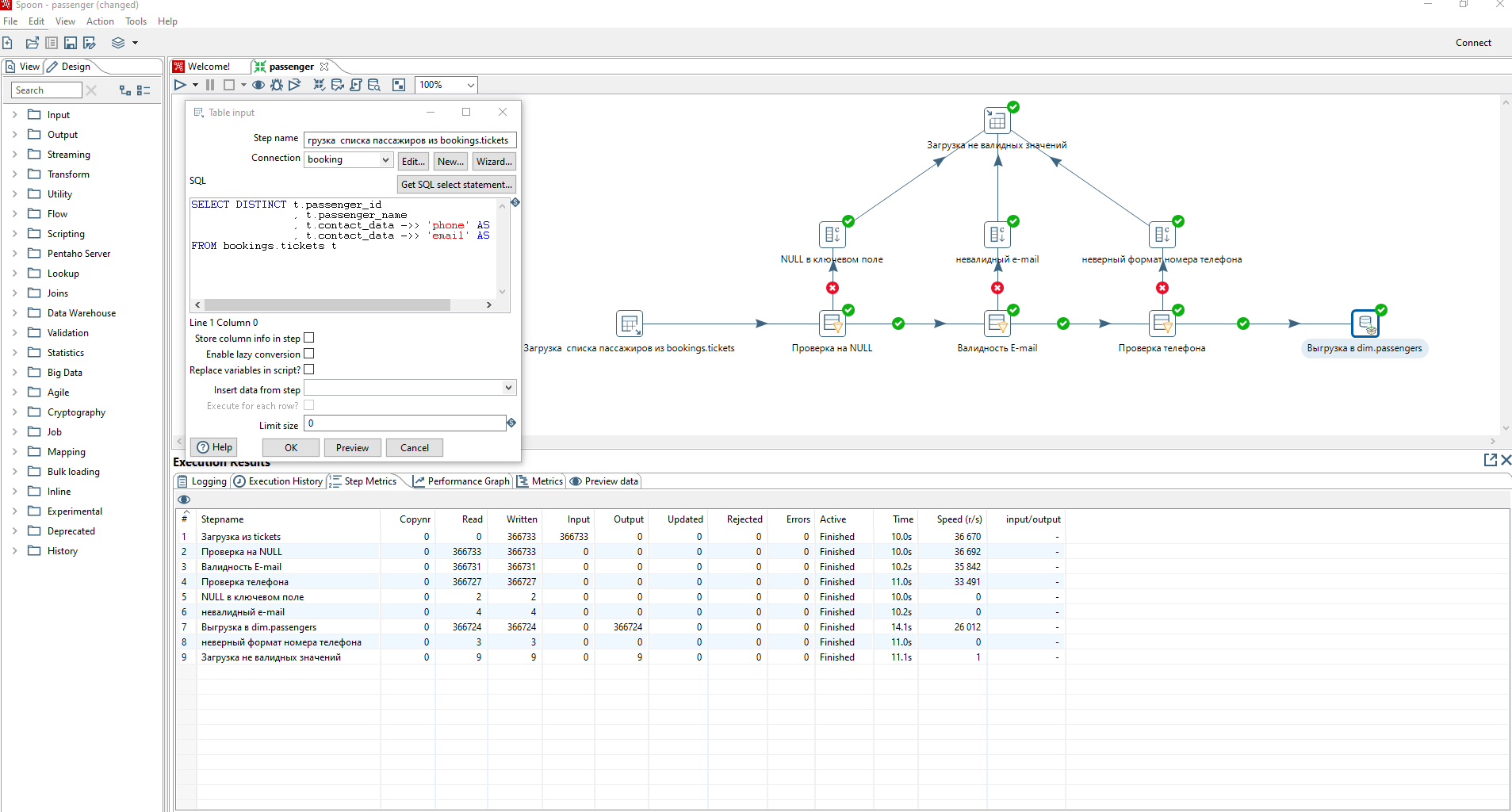
passenger\_name **text** **NOT** **NULL**, -- ФИО пасажира (bookings.tickets.passenger)

phone **varchar**(20), -- телефон пассажира (bookings.tickets.contact\_data ->> 'phone')

email **varchar**(150) -- email пассажира (bookings.tickets.contact\_data ->> 'email')

);

ETL-трансформация по наполнению и проверке данных в таблице dim.passengers представлена на рисунке 2.

Рисунок 2 Заполнение dim.passengers

### Описание трансформации.

Первым шагом производится чтение данных из таблицы bookings.tickets.

Далее проверяем поле ключевые поля на NULL, отсутствующие значения записываем в таблицу rejected.passengers, созданную на этапе создания схемы rejected с помощью SQL скрипта

**CREATE** **TABLE** rejected.passengers (

passenger\_id **varchar**(20),

passenger\_name **text**,

phone **varchar**(20),

email **varchar**(150),

reason\_for\_rejection **TEXT** -- поле с причиной отклонения

);.

После проверяем формат адреса почты на корректность, а также номер телефона через регулярное выражение на отсутствие нечисловых символов и длины номера.

После всех проверок на последнем шаге отбираем нужные поля и производим запись в таблицу dim.passengers

## Таблица dim.aircrafts

Таблица представляет собой справочник самолетов

Таблица содержит поля:

aircraft\_code – код самолета;

model – наименование самолета;

manufacturer - производитель

range – дальность полета;

Запрос SQL на создание таблицы

**CREATE** **TABLE** dim.aircrafts (

aircraft\_code **bpchar**(3) **PRIMARY** **KEY**, -- ключ (bookings.aircrafts.aircraft\_code)

model **varchar**(20) **NOT** **NULL**, -- модель (bookings.aircrafts.model)

manufacturer **varchar**(20) **NOT** **NULL**, -- производитель - первое слово в названии "range" **int4** **NOT** **NULL** -- расстояние (bookings.aircrafts."range")

);

ETL-трансформация по наполнению таблицы представлена на рисунке 3.

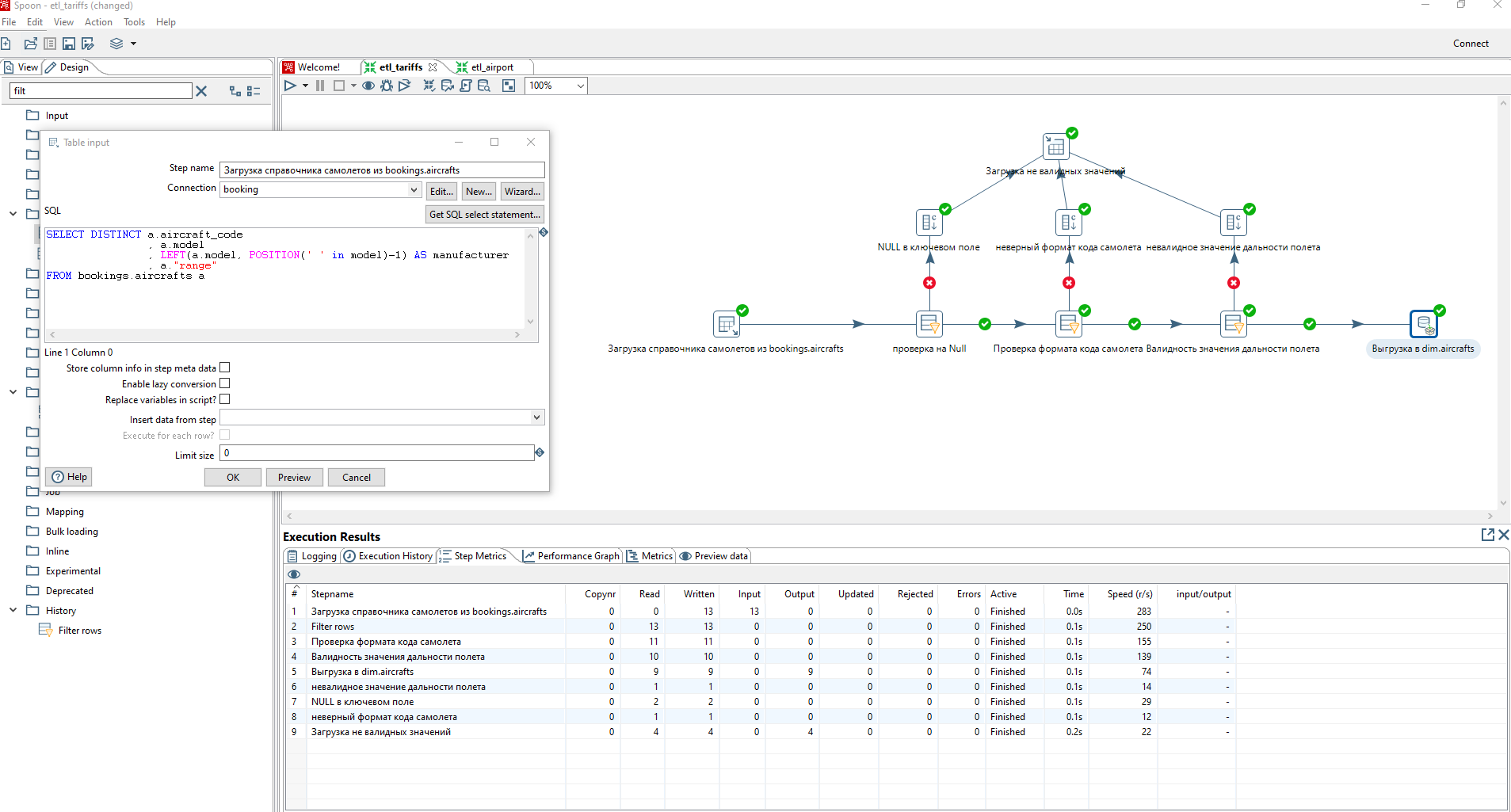


Рисунок 3 Заполнение dim.aircrafts

### Описание трансформации.

Первым шагом производится чтение данных из таблицы bookings.aircrafts.

Далее проверяем поле ключевые поля на NULL, отсутствующие значения записываем в таблицу rejected.aircrafts, созданную на этапе создания схемы rejected с помощью SQL скрипта

**CREATE** **TABLE** rejected.aircrafts (

aircraft\_code **bpchar**(3),

model **varchar**(20),

manufacturer **varchar**(20),

"range" **int4**,

reason\_for\_rejection **TEXT**

);

После проверяем формат кода самолета на корректность, а также валидность дальности полета.

После всех проверок, на последнем шаге отбираем нужные поля и производим запись в таблицу dim.aircrafts

## Таблица dim.airports

Таблица представляет собой справочник самолетов

Запрос SQL на создание таблицы

**CREATE** **TABLE** dim.airports (

airport\_code **bpchar**(3) **PRIMARY** **KEY**, -- ключ (bookings.airports.airport\_code)

airport\_name **varchar**(50) **NOT** **NULL**, -- название аэропорта (bookings.airports.airport\_name)

airport\_city **varchar**(50) **NOT** **NULL**, -- город аэропорта (bookings.airports.city)

longitude **float**(8) **NOT** **NULL**, -- долгота города (bookings.airports.longitude)

latitude **float**(8) **NOT** **NULL** -- широта города (bookings.airports.latitude)

);

ETL-трансформация по наполнению таблицы представлена на рисунке 4.

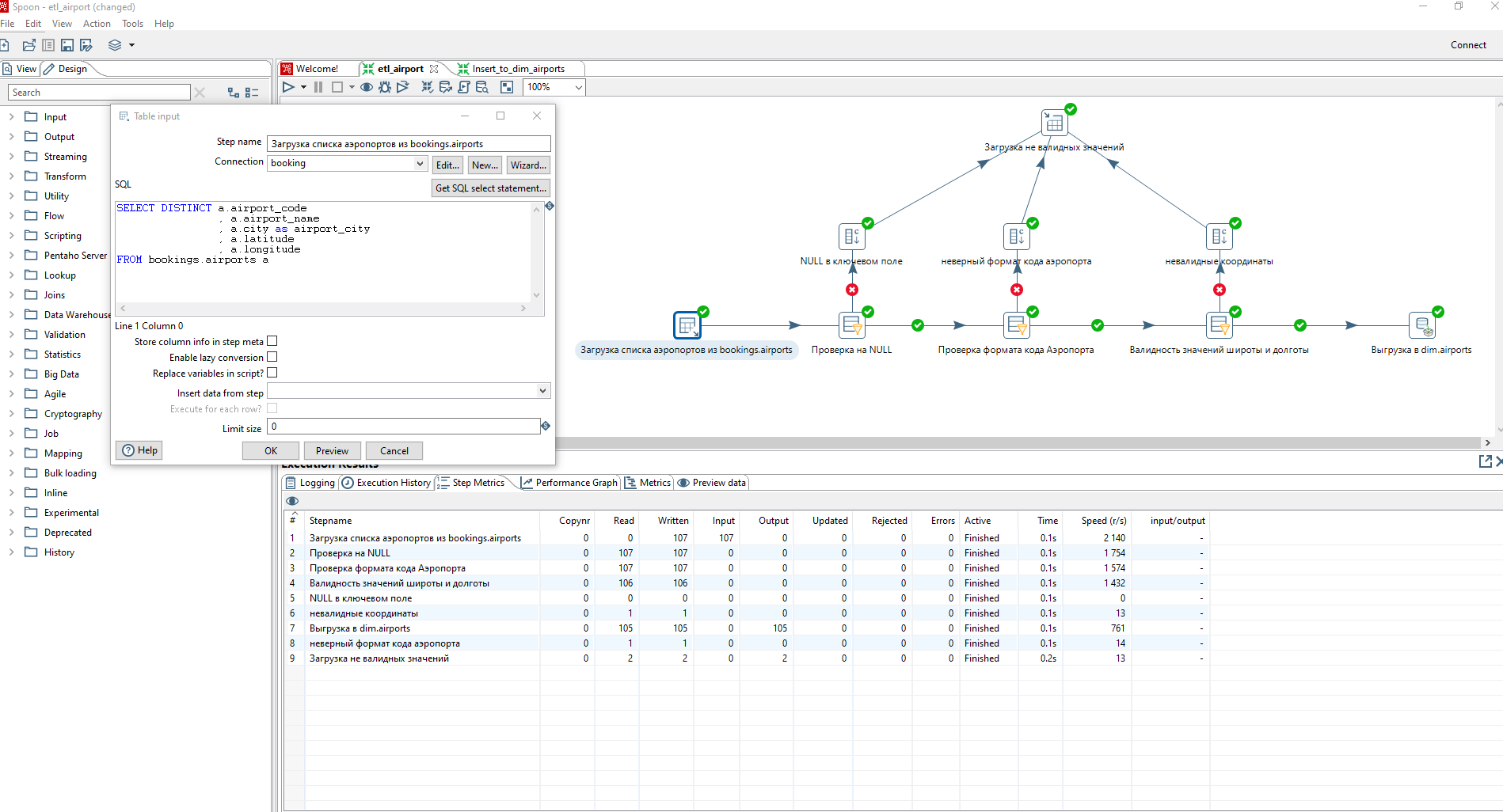


Рисунок 4. Заполнение dim.airports

### Описание трансформации.

Первым шагом производится чтение данных из таблицы bookings.airports.

Далее проверяем поле ключевые поля на NULL, отсутствующие значения записываем в таблицу rejected.airports, созданную на этапе создания схемы rejected с помощью SQL скрипта

**CREATE** **TABLE** rejected.airports (

airport\_code **bpchar**(3),

airport\_name **varchar**(50),

airport\_city **varchar**(50),

longitude **float**(8),

latitude **float**(8),

reason\_for\_rejection **TEXT** -- поле с причиной отклонения

);

После проверяем формат кода аэропорта на корректность, а также валидность значений широты и долготы.

После всех проверок, на последнем шаге отбираем нужные поля и производим запись в таблицу dim.airports.

## Таблица dim.tariff

Таблица представляет собой справочник тарифов

Запрос SQL на создание таблицы

**CREATE** **TABLE** dim.tariff (

fare\_conditions **varchar**(10) **PRIMARY** **KEY** -- тариф (bookings.seats.fare\_conditions)

);

ETL-трансформация по наполнению таблицы представлена на рисунке 4.

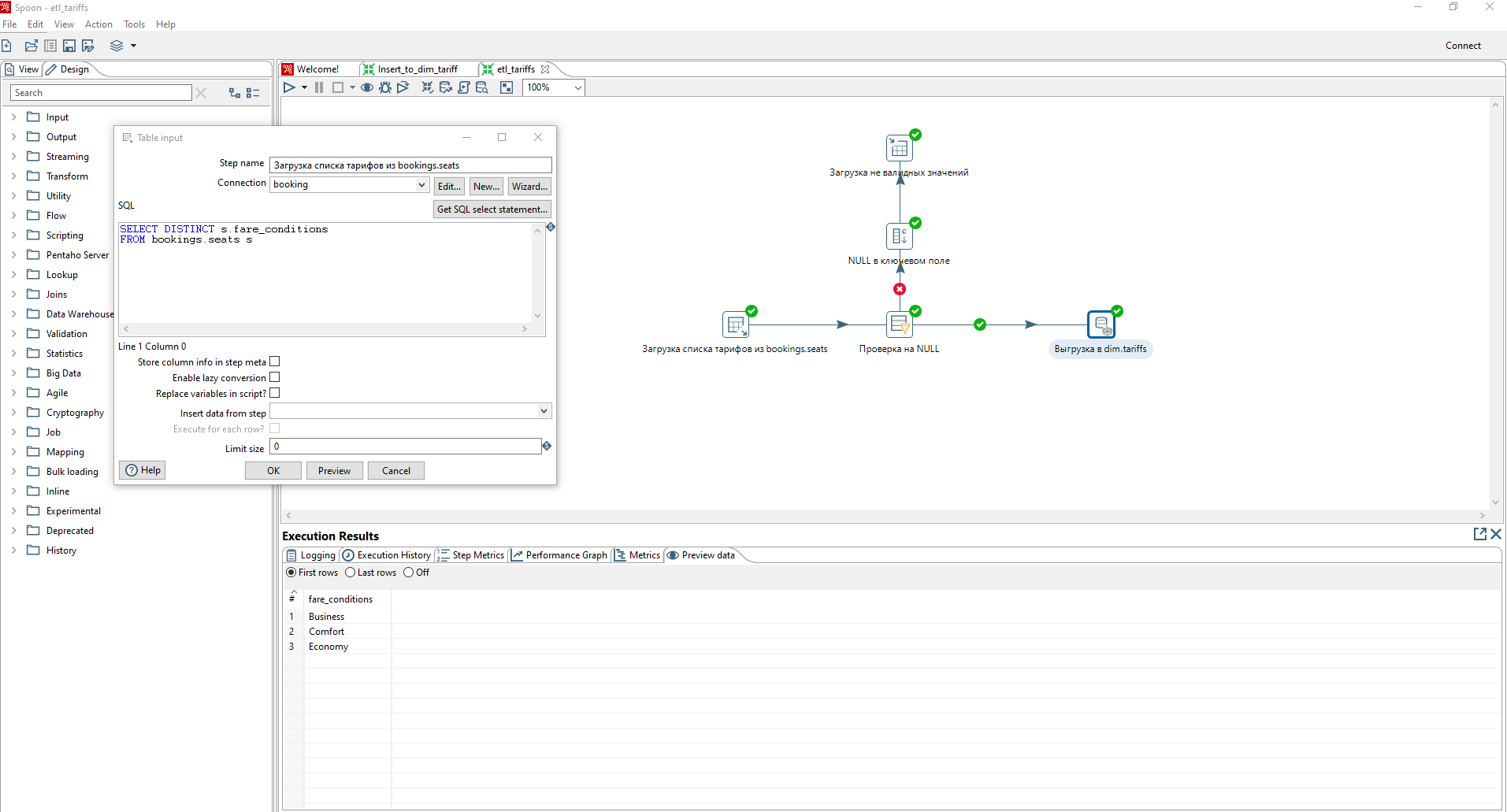


Рисунок 4. Заполнение dim.tariff

### Описание трансформации.

Первым шагом производится чтение данных из таблицы bookings.seats.

Далее проверяем поле ключевые поля на NULL, отсутствующие значения записываем в таблицу rejected.tariff, созданную на этапе создания схемы rejected с помощью SQL скрипта

**CREATE** **TABLE** rejected.tariff (

fare\_conditions **varchar**(10),

reason\_for\_rejection **TEXT** -- поле с причиной отклонения

);

## Таблица dim.flights

Таблица представляет собой справочник совершенных перелетов

Запрос SQL на создание таблицы

**CREATE** **TABLE** dim.flights (

flight\_id **int4** **PRIMARY** **KEY**, -- ключ (bookings.flights.flight\_id)

flight\_no **bpchar**(6), -- номер рейса (bookings.flights.flight\_no)

actual\_departure **timestamp**, -- Дата и время вылета (bookings.flights.actual\_departure)

actual\_arrival **timestamp**, -- Дата и время прилета (bookings.flights.actual\_arrival)

scheduled\_departure **timestamp**, -- Дата и время вылета (bookings.flights.scheduled\_departure)

scheduled\_arrival **timestamp**, -- Дата и время прилета (bookings.flights.scheduled\_arrival)

delay\_time\_departure **int4**, -- Задержка вылета (разница между фактической и запланированной датой в секундах) (bookings.flights.actual\_departure - bookings.flights.scheduled\_departure)

delay\_time\_arrival **int4**, -- Задержка прилета (разница между фактической и запланированной датой в секундах) (bookings.flights.actual\_arrival - bookings.flights.scheduled\_arrival)

aircraft\_code **varchar**(30), -- код самолета (bookings.flights.aircraft\_code)

airports\_departure\_code **varchar**(30), -- Аэропорт вылета (bookings.flights.departure\_airports)

airports\_arrival\_code **varchar**(30) -- Аэропорт прилета (bookings.flights.arrival\_airports)

);

ETL-трансформация по наполнению таблицы представлена на рисунке 4.

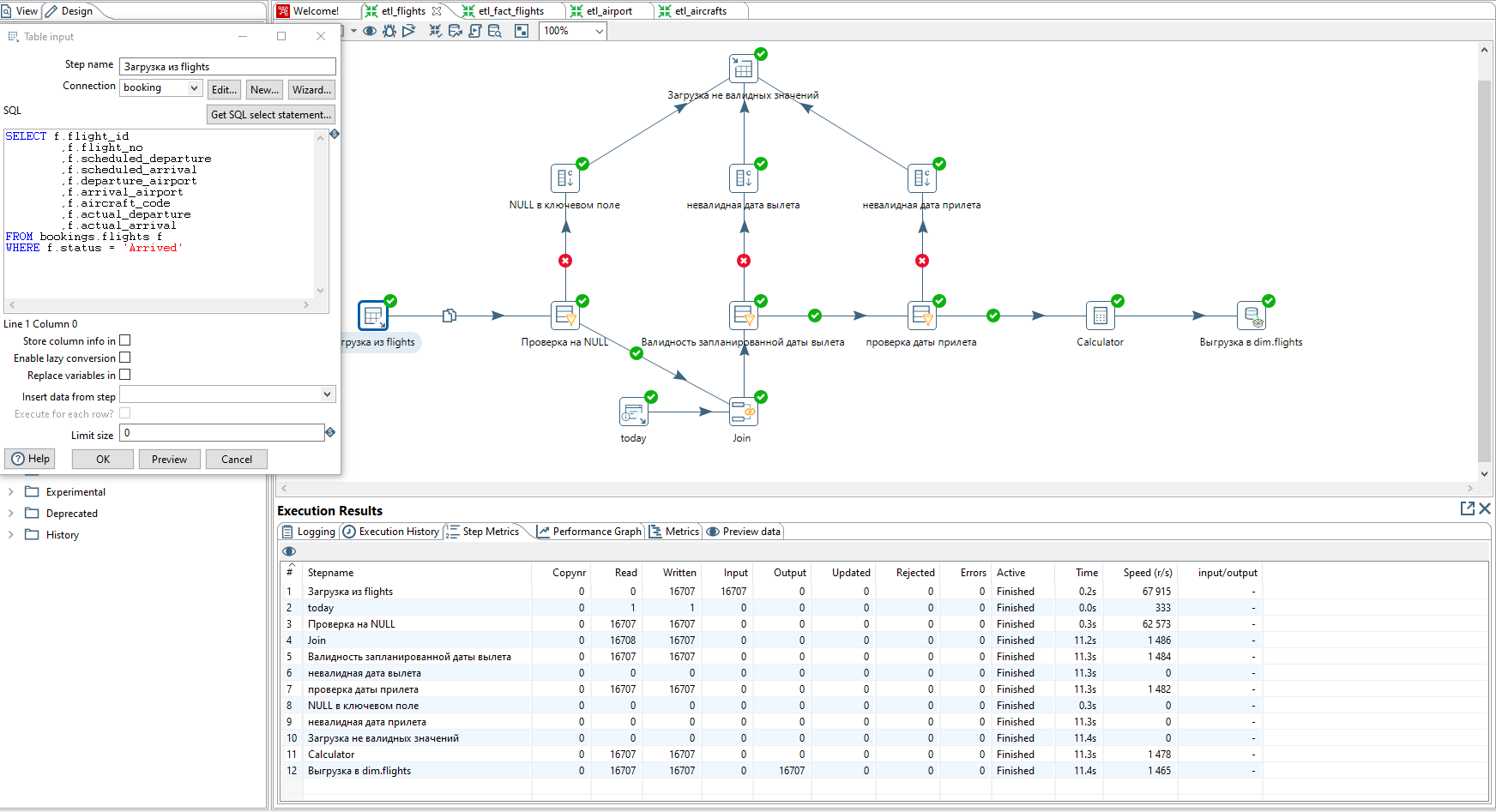


Рисунок 4. Заполнение dim.flights

### Описание трансформации.

Первым шагом производится чтение данных из таблицы bookings.flights.

Далее проверяем поле ключевые поля на NULL, запланированное время вылета проверяем на дату( поскольку это совершенные вылеты дата не должна быть больше текущей даты, также проверяем запланированное время прилета оно не должно быть меньше запланированного времени вылета, не валидные значения записываем в таблицу rejected.flights, созданную на этапе создания схемы rejected с помощью SQL скрипта

**CREATE** **TABLE** rejected.dim\_flights (

flight\_id **int4**,

flight\_no **bpchar**(6),

actual\_departure **timestamp**,

actual\_arrival **timestamp**,

scheduled\_departure **timestamp**,

scheduled\_arrival **timestamp**,

delay\_time\_departure **int4**,

delay\_time\_arrival **int4**,

aircraft\_code **varchar**(30),

airports\_departure\_code **varchar**(30),

airports\_arrival\_code **varchar**(30),

reason\_for\_rejection **TEXT** -- поле с причиной отклонения

);

Далее рассчитываем время задержки вылета и прилета и загружаем в таблицу

## Таблица fact.flights

Таблица содержит информацию о фактически выполненных перелетах. Гранулярность – один пассажир. Таблица связана с таблицами измерений по соответствующим id по схеме «звезда»

**CREATE** **TABLE** fact.flights (

flight\_no **bpchar**(6) **NOT** **NULL**, -- номера рейса (flight\_no)

passenger\_id **text** **references** dim.passengers(passenger\_id), -- Пассажир (dim.passengers.passenger\_id)

actual\_departure **timestamp** **NOT** **NULL** **REFERENCES** dim.calendar(date\_time), -- Дата и время вылета (факт) (dim.flights.actual\_departure)

actual\_arrival **timestamp** **NOT** **NULL** **REFERENCES** dim.calendar(date\_time), -- Дата и время прилета (факт) (dim.flights.actual\_arrival)

delay\_time\_departure **int4** **NOT** **NULL**, -- Задержка вылета (разница между фактической и запланированной датой в секундах) (dim.flights.delay\_time\_departure)

delay\_time\_arrival **int4** **NOT** **NULL**, -- Задержка прилета (разница между фактической и запланированной датой в секундах) (dim.flights.delay\_time\_arrival)

aircraft\_code **varchar**(30) **NOT** **NULL** **REFERENCES** dim.aircrafts(aircraft\_code), -- код самолета (dim.flights.aircraft\_code)

airports\_departure\_code **varchar**(30) **NOT** **NULL** **REFERENCES** dim.airports(airport\_code), -- Аэропорт вылета (dim.flights.airports\_departure)

airports\_arrival\_code **varchar**(30) **NOT** **NULL** **REFERENCES** dim.airports(airport\_code), -- Аэропорт прилета (dim.flights.airports\_arrival)

fare\_conditions **varchar**(10) **NOT** **NULL** **REFERENCES** dim.tariff(fare\_conditions), -- ключ класса обслуживания (dim.tariff)

amount **numeric**(10,2) -- стоимость (bookings.ticket\_flights.amount)

);

ETL-трансформация по наполнению таблицы представлена на рисунке 5.

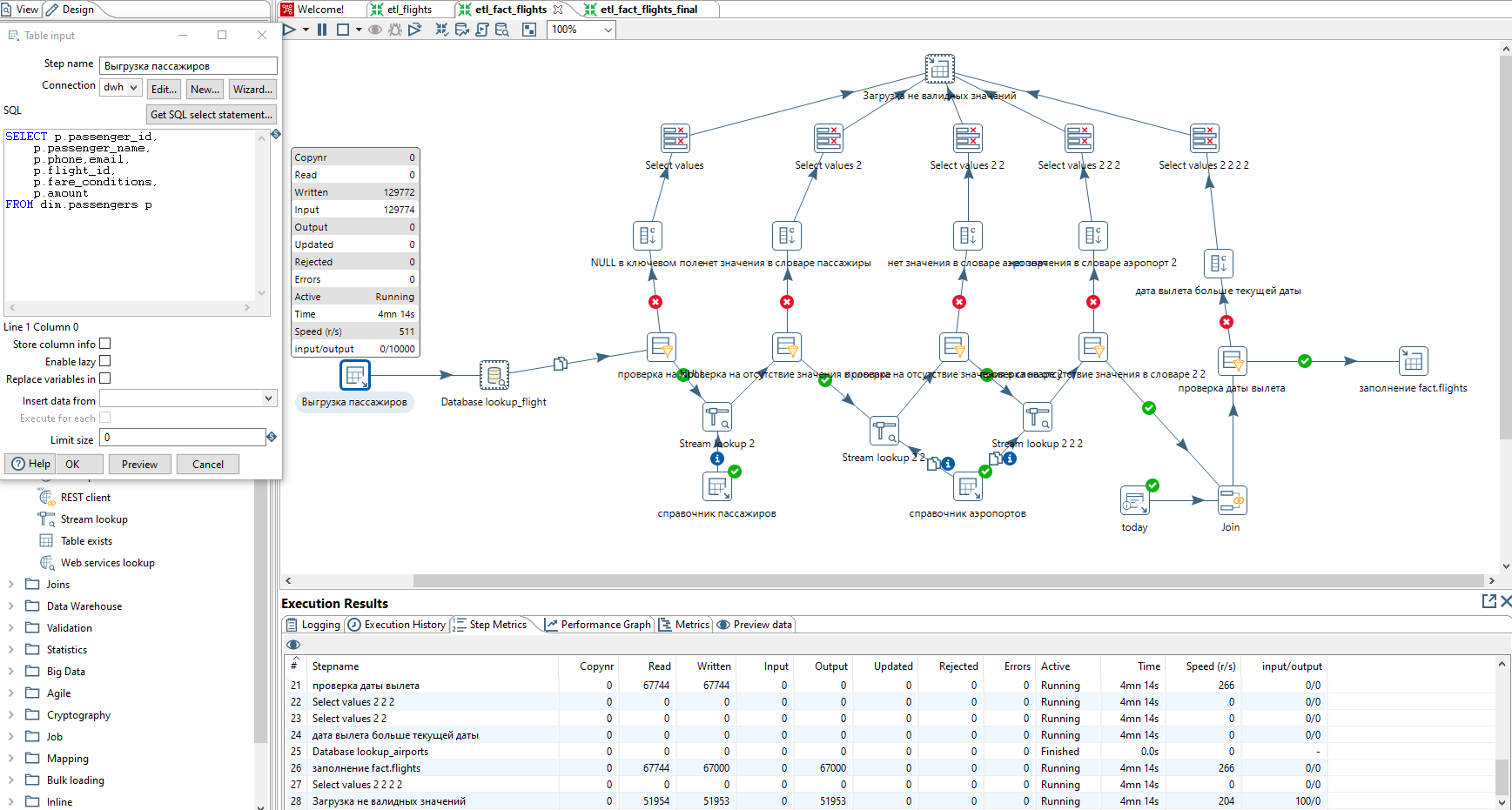


Рисунок 5. Заполнение fact.flights

### Описание трансформации.

### Из таблиц измерений dim.passengers и dim.flights выгружаем данные по всем пассажирам и их состоявшимся вылетах через проверку полей:

проводим проверки на Null в ключевых полях и наличие значений словарей в данных таблицы, также проводится проверка даты вылета относительно текущей даты.

Отсутствующие и невалидные значения записываем в таблицу rejected.flights, созданную на этапе создания схемы rejected с помощью SQL скрипта

**CREATE** **TABLE** rejected.flights (

flight\_no **bpchar**(6),

passenger\_id **text**,

actual\_departure **timestamp** ,

actual\_arrival **timestamp**,

delay\_time\_departure **int4**,

delay\_time\_arrival **int4**,

aircraft\_code **varchar**(30),

airports\_departure\_code **varchar**(30),

airports\_arrival\_code **varchar**(30),

fare\_conditions\_id **int**,

amount **numeric**(10,2),

reason\_for\_rejection **TEXT** -- поле с причиной отклонения

);

После всех проверок, на последнем шаге отбираем нужные поля, производим маппинг и производим запись в таблицу fact.flights

Фрагмент таблицы fact\_flights приведён на рисунке 7.

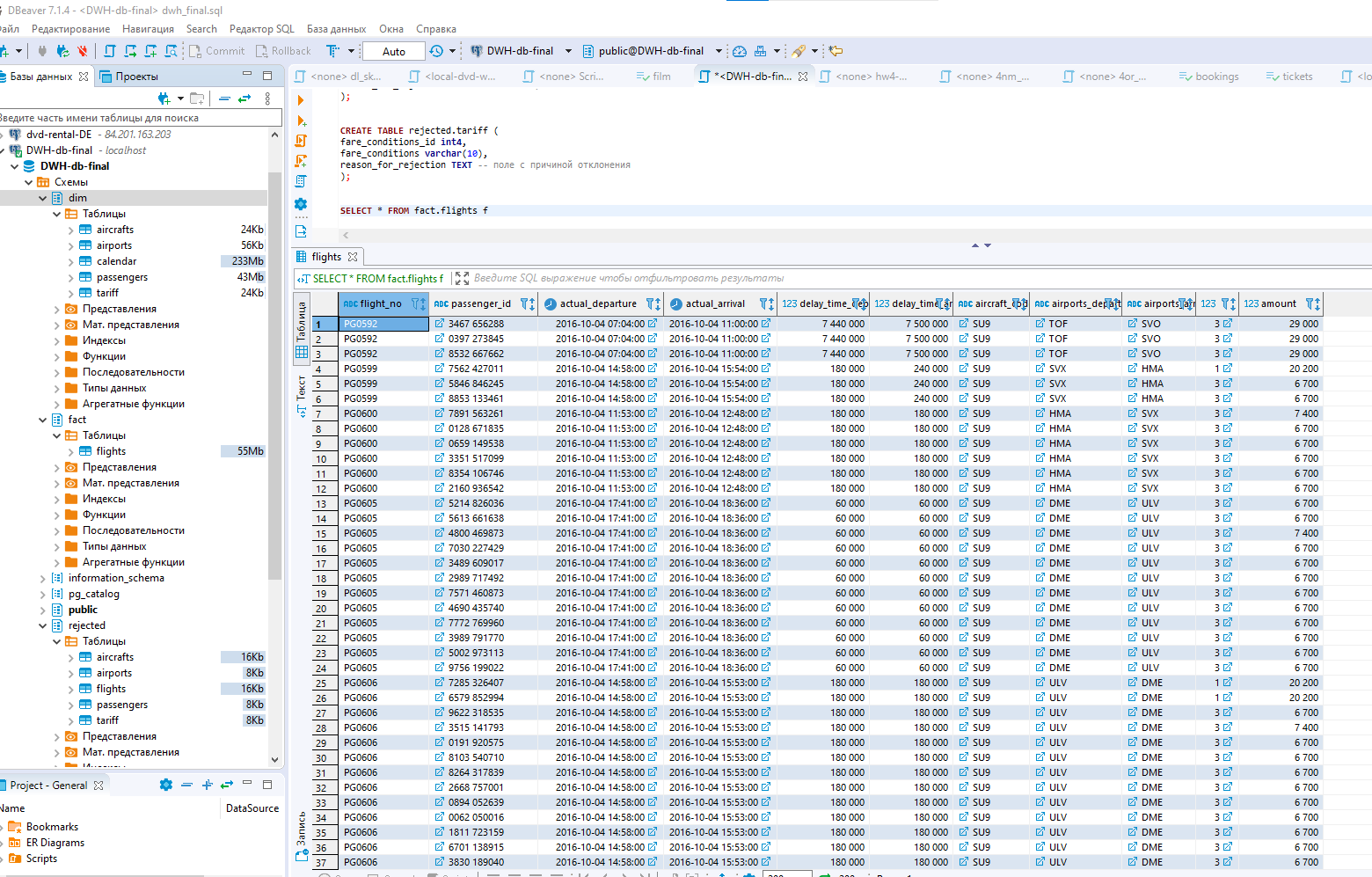


Рисунок 7 Фрагмент таблицы fact\_flights

# Приложение:

SQL-скрипты создания таблиц

* dwh\_final.sql

ETL-трансформации.

* etl\_aircrafts.ktr
* etl\_airport.ktr
* etl\_passenger.ktr
* etl\_tariffs.ktr
* etl\_flights.ktr
* etl\_fact\_flights.ktr

**Ссылка на github:** [**https://github.com/finesun16/netology\_dwh\_final**](https://github.com/finesun16/netology_dwh_final)