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.

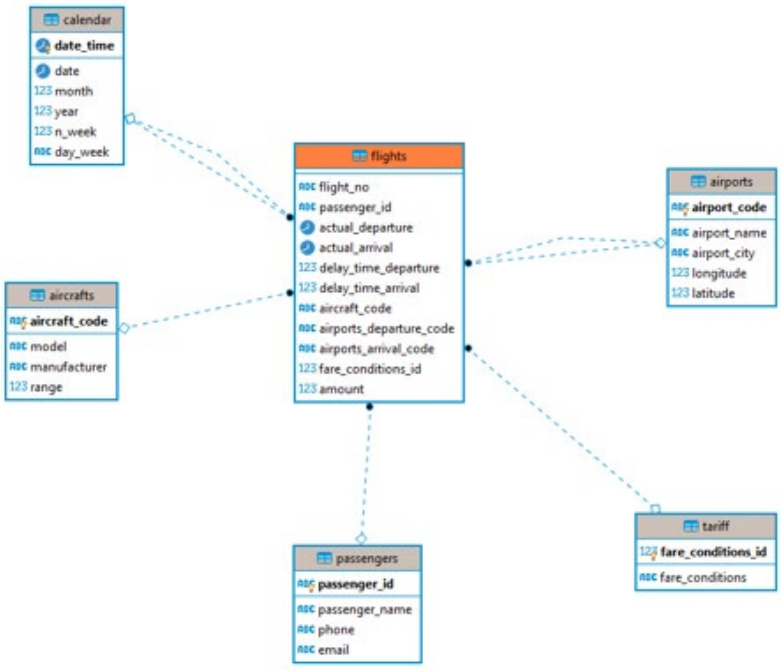


Рисунок 1 Диаграмма БД

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

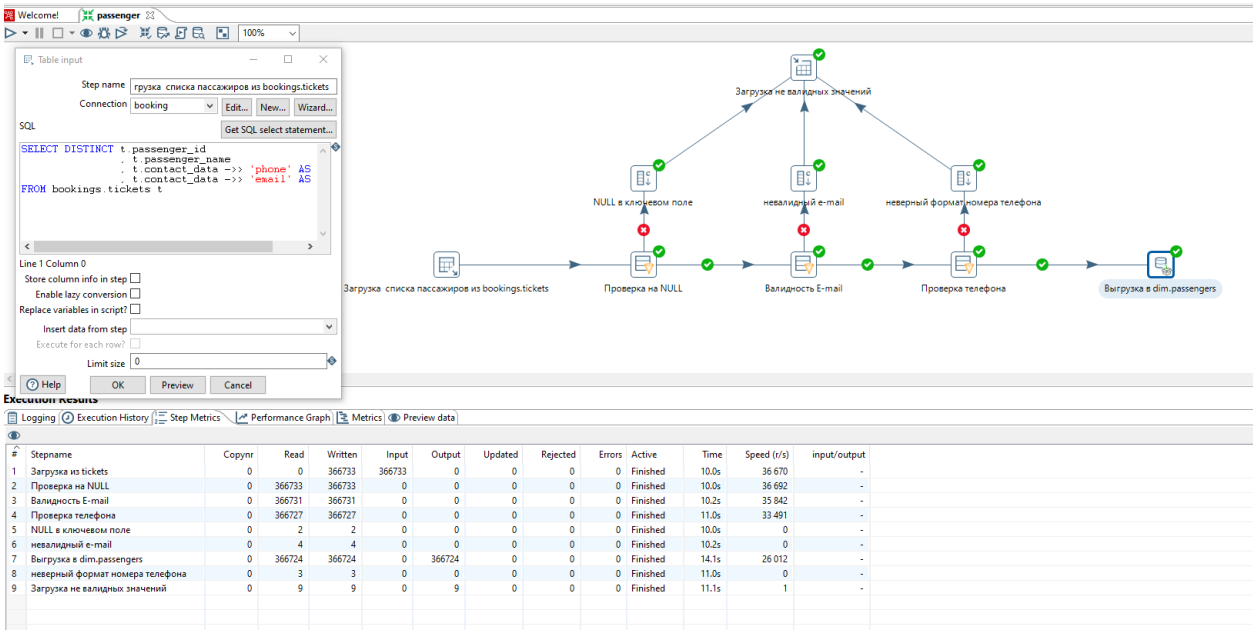
Таблица представляет собой справочник пассажиров. Гранулярность – один пассажир.
Таблица содержит поля:

- 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.



#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	Загрузка из tickets	0	0	366733	366733	0	0	0	0	Finished	10.0s	36 670	-
2	Проверка на NULL	0	366733	366733	0	0	0	0	0	Finished	10.0s	36 692	-
3	Валидность E-mail	0	366731	366731	0	0	0	0	0	Finished	10.2s	35 842	-
4	Проверка телефона	0	366727	366727	0	0	0	0	0	Finished	11.0s	33 491	-
5	NULL в ключевом поле	0	2	2	0	0	0	0	0	Finished	10.0s	0	-
6	невалидный e-mail	0	4	4	0	0	0	0	0	Finished	10.2s	0	-
7	Выгрузка в dim.passengers	0	366724	366724	0	366724	0	0	0	Finished	14.1s	26 012	-
8	неверный формат номера телефона	0	3	3	0	0	0	0	0	Finished	11.0s	0	-
9	Загрузка не валидных значений	0	9	9	0	9	0	0	0	Finished	11.1s	1	-

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.

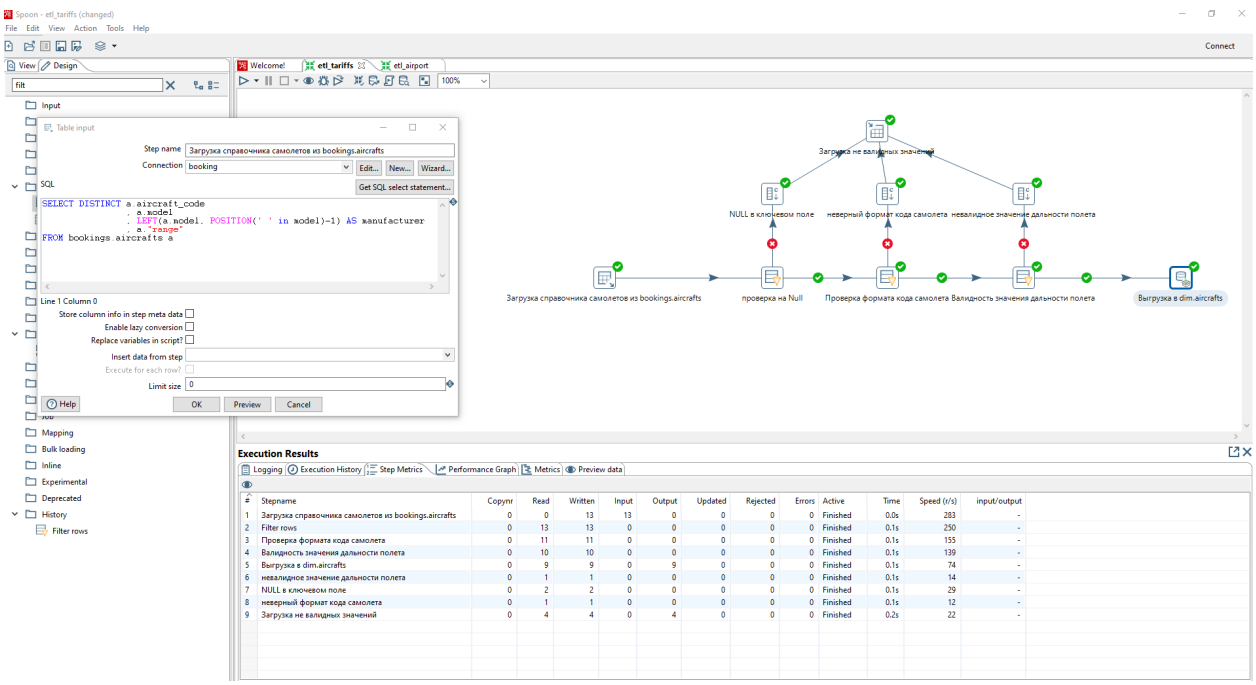


Рисунок 3 Заполнение dim.aircrafts

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.

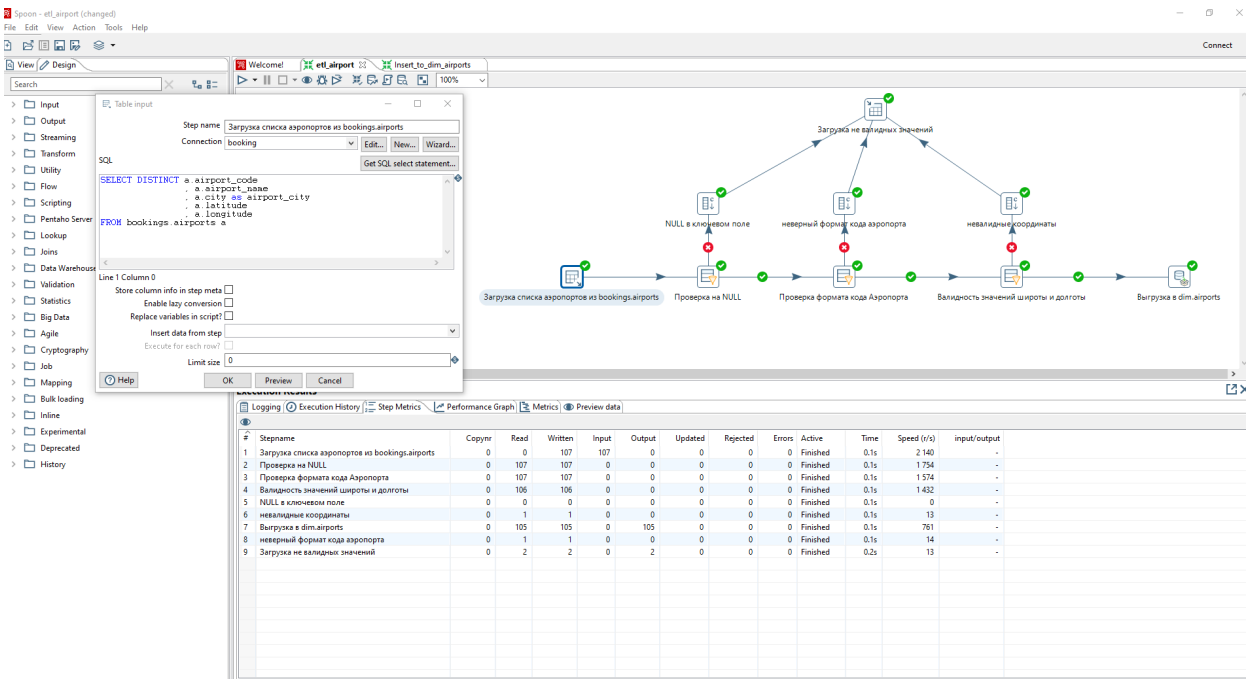


Рисунок 4. Заполнение dim.airports

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_id serial PRIMARY KEY, -- ключ тарифа - не задан, задаем сами
	fare_conditions varchar(10) NOT NULL -- тариф (bookings.seats.fare_conditions)
);
```

ETL-трансформация по наполнению таблицы представлена на рисунке 4.

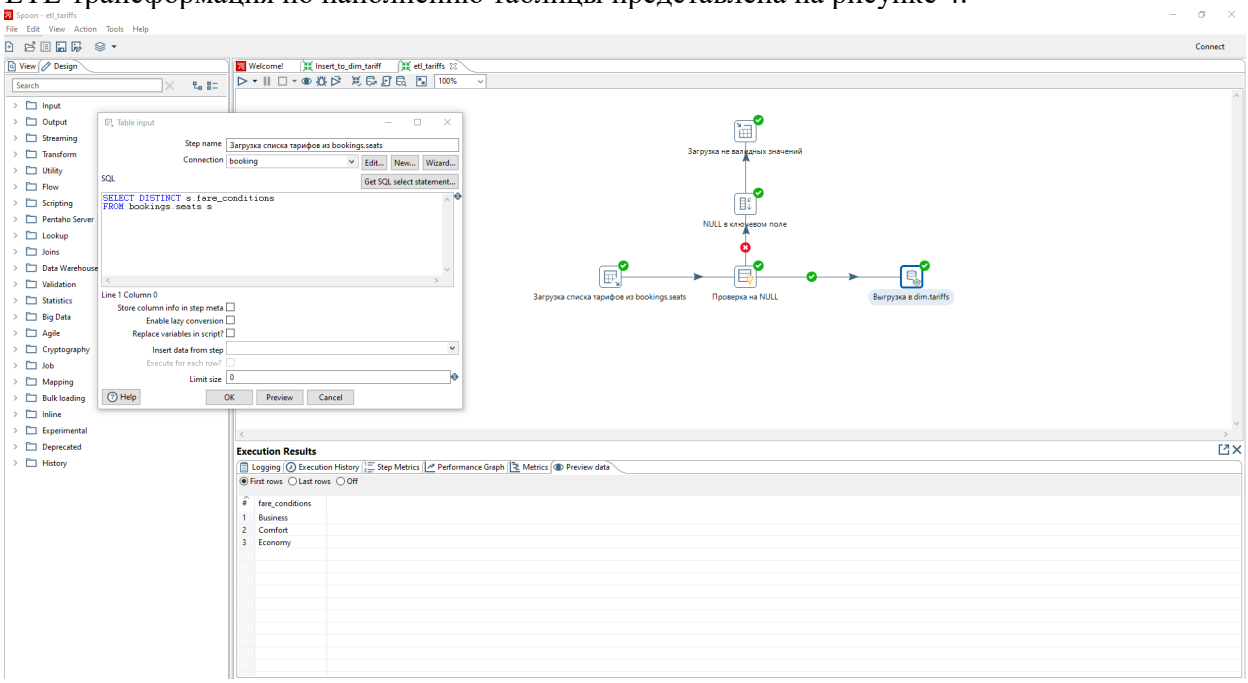


Рисунок 4. Заполнение dim.tariff

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_id int4,
fare_conditions varchar(10),
reason_for_rejection TEXT -- поле с причиной отклонения
);
```

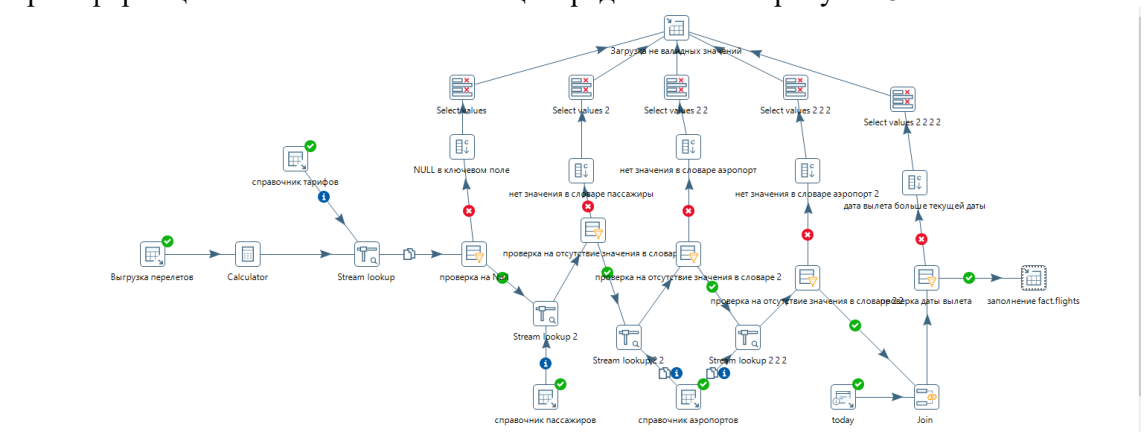
Таблица fact.flights

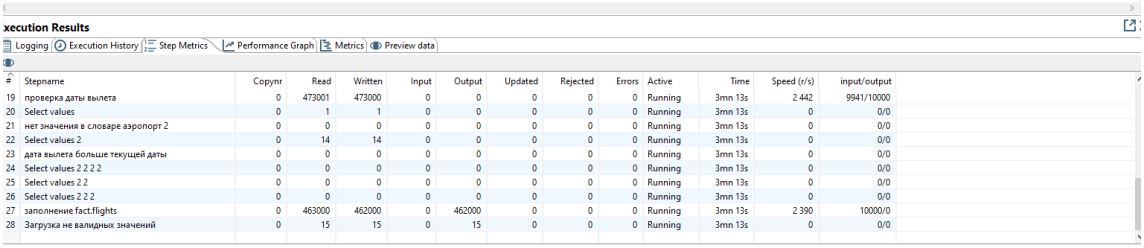
Таблица содержит информацию о фактически выполненных перелетах. Гранулярность – один пассажир. Таблица связана с таблицами измерений по соответствующим id по схеме «звезда»

```
CREATE TABLE fact.flights (
	flight_no bpchar(6) NOT NULL, -- совокупность номера рейса (flight_no) и даты отправления
	(scheduled_departure или actual_departure) являются естественным ключом, поэтому добавление
	данного поля упростит работу конечного пользователя с таблицей фактов
	passenger_id text references dim.passengers(passenger_id),
	-- Пассажир (bookings.tickets.passenger_name)
	actual_departure timestamp NOT NULL REFERENCES dim.calendar(date_time),
	-- Дата и время вылета (факт) (bookings.flights.actual_departure)
	actual_arrival timestamp NOT NULL REFERENCES dim.calendar(date_time),
	-- Дата и время прилета (факт) (bookings.flights.actual_arrival)
```

```
    delay_time_departure int4 NOT NULL,
    -- Задержка вылета (разница между фактической и запланированной датой в секундах)
    (bookings.flights.actual_departure - bookings.flights.scheduled_departure)
    delay_time_arrival int4 NOT NULL,
    -- Задержка прилета (разница между фактической и запланированной датой в секундах)
    (bookings.flights.actual_arrival - bookings.flights.scheduled_arrival)
    aircraft_code varchar(30) NOT NULL REFERENCES dim.aircrafts(aircraft_code),
    -- код самолета (bookings.flights.aircraft_code)
    airports_departure_code varchar(30) NOT NULL REFERENCES dim.airports(airport_code),
    -- Аэропорт вылета (bookings.flights.airports_departure)
    airports_arrival_code varchar(30) NOT NULL REFERENCES dim.airports(airport_code),
    -- Аэропорт прилета (bookings.flights.airports_arrival)
    fare_conditions_id int REFERENCES dim.tariff(fare_conditions_id),
    -- ключ класса обслуживания (dim.tariff)
    amount numeric(10,2)
    -- стоимость (bookings.ticket_flights.amount)
```

ETL-трансформация по наполнению таблицы представлена на рисунке 5.





xecution Results

Logging | Execution History | Step Metrics | Performance Graph | Metrics | Preview data

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
19	проверка даты вылета	0	473001	473000	0	0	0	0	0	Running	3mn 13s	2 442	9941/10000
20	Select values	0	1	1	0	0	0	0	0	Running	3mn 13s	0	0/0
21	нет значения в словаре аэропорт 2	0	0	0	0	0	0	0	0	Running	3mn 13s	0	0/0
22	Select values 2	0	14	14	0	0	0	0	0	Running	3mn 13s	0	0/0
23	дата вылета больше текущей даты	0	0	0	0	0	0	0	0	Running	3mn 13s	0	0/0
24	Select values 2 2 2 2	0	0	0	0	0	0	0	0	Running	3mn 13s	0	0/0
25	Select values 2 2	0	0	0	0	0	0	0	0	Running	3mn 13s	0	0/0
26	Select values 2 2 2	0	0	0	0	0	0	0	0	Running	3mn 13s	0	0/0
27	заполнение fact.flights	0	463000	462000	0	462000	0	0	0	Running	3mn 13s	2 390	10000/0
28	Загрузка не валидных значений	0	15	15	0	15	0	0	0	Running	3mn 13s	0	0/0

Рисунок 5. Заполнение fact.flights

Описание трансформации.

Сперва выгружаем всю доступную информацию из БД bookings. рис 6

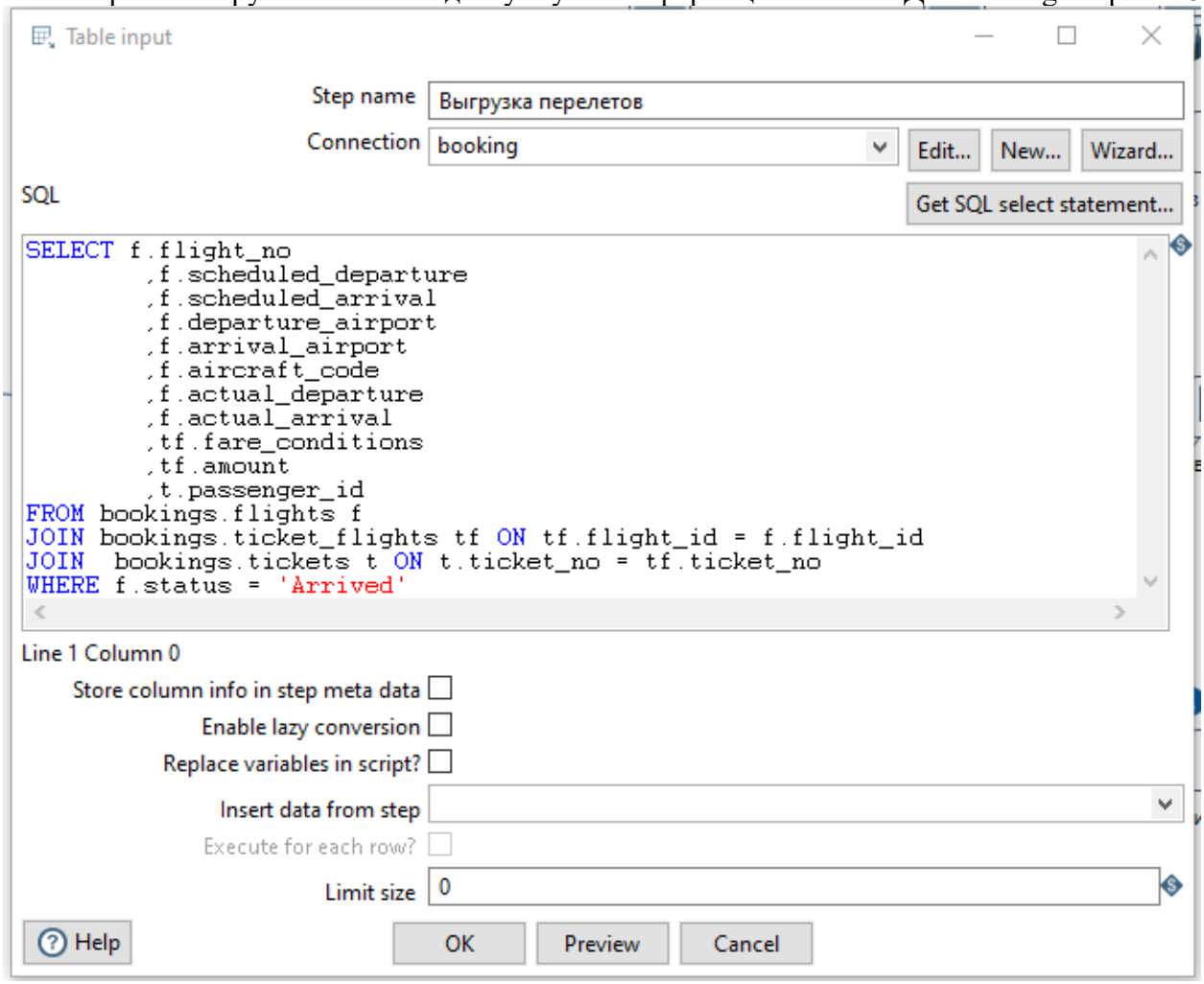


Рисунок 6. SQL запрос к БД источника

Далее рассчитываем время задержки вылета и прилета. Так как в БД bookings нет естественного ключа тарифов, то к значениям подтягиваем данные из справочника.

Далее 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.

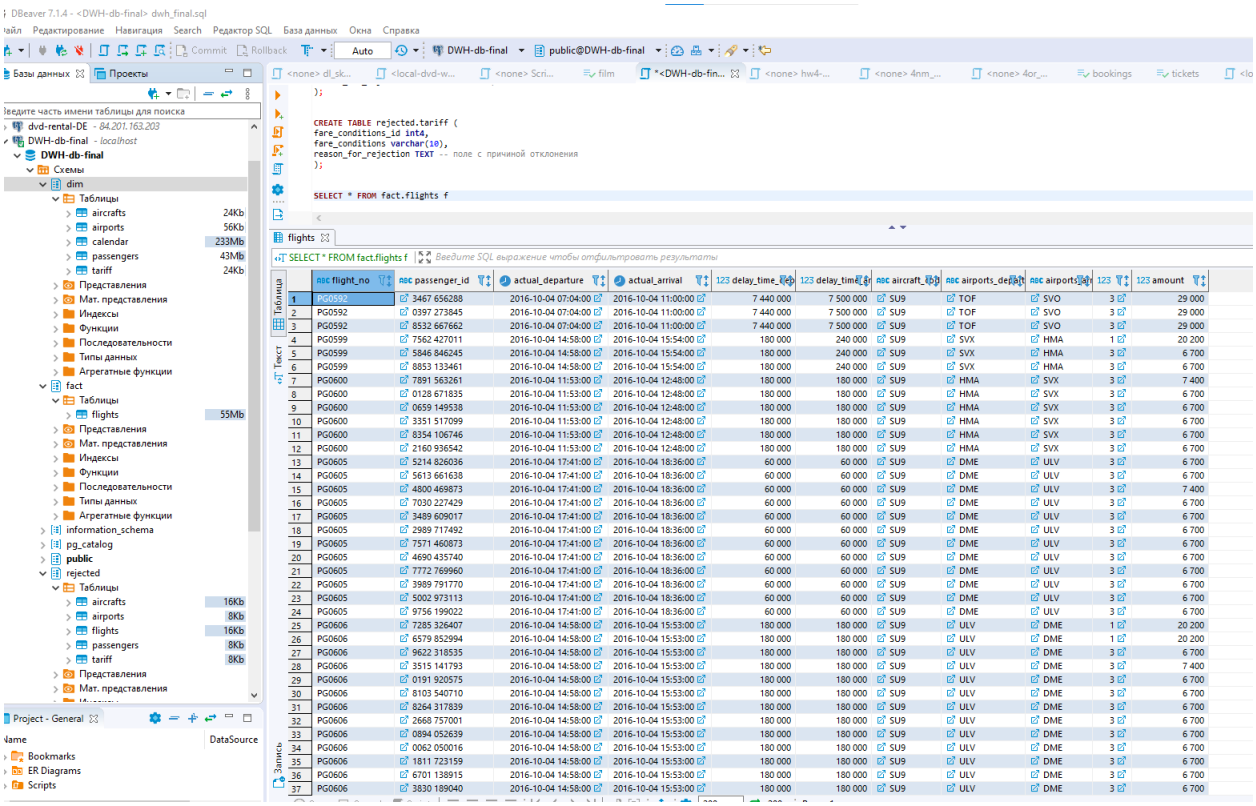


Рисунок 7 Фрагмент таблицы fact_flights

Приложение:

SQL-скрипты создания таблиц

- dwh_final.sql

ETL-трансформации.

- etl_aircrafts.ktr
- etl_airport.ktr
- etl_passenger.ktr
- etl_tariffs.ktr
- etl_flight.ktr

Ссылка на github: https://github.com/finesun16/netology_dwh_final