

# Итоговая работа

## по курсу «DWH (Data Warehouse)»

14 мая 2021г.

группа DWH-5

Новоселов П.В.

В работе использовался удаленный тип подключения к базе данных, используя ПО Docker Desktop. Была создана пустая база данных PostgreSQL, в которой были созданы таблицы:

- измерений:
  - dim\_calendar - справочник дат
  - dim\_passengers - справочник пассажиров
  - dim\_aircrafts - справочник самолетов
  - dim\_airports - справочник аэропортов
  - dim\_tariff - справочник тарифов
- фактов:
  - fact\_flights - содержит совершенные перелеты.

Диаграмма базы данных представлена на рисунке 1.

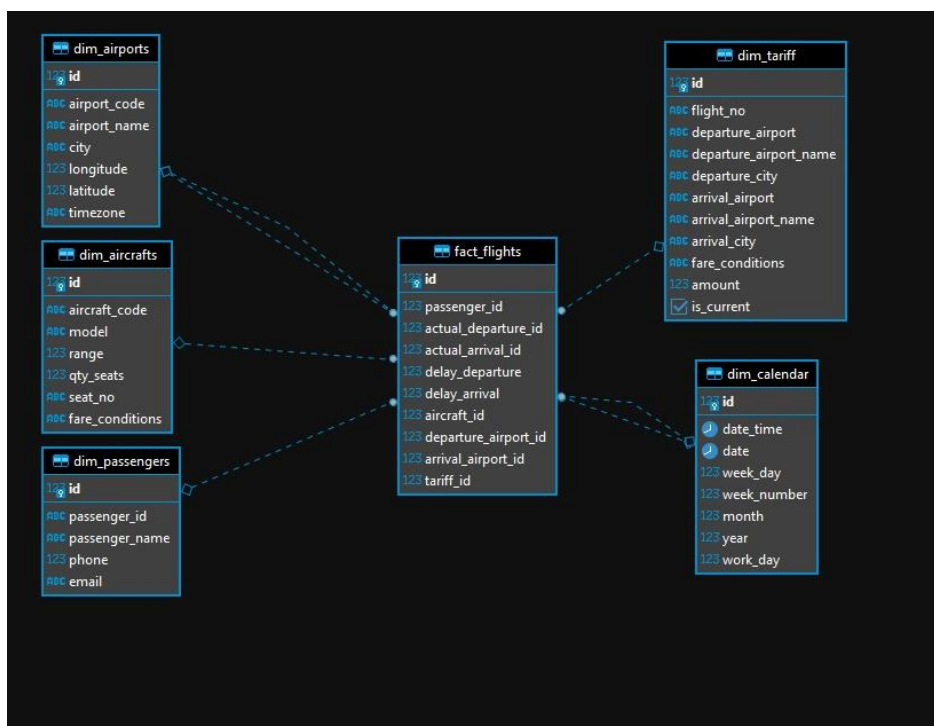


Рисунок 1 Диаграмма БД

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 содержит поля:

id – суррогатный первичный ключ в формате YYYYMMDDHHMI;

date\_time – дата и время в формате timestamptz;

date – дата в формате date;

week\_day – номер дня в неделе (понедельник - 1, вторник – 2 и т.д.);

week\_number – номер недели в году;

month – номер месяца в году;

year – номер года

work\_day – признак дня: рабочий – 1, выходной – 0.

Наполнение таблицы произведено при помощи SQL-запроса:

```
CREATE TABLE dim_calendar as
with dates as (
    select dd as date_time
    from generate_series('2016-09-01'::timestampz, '2016-11-30'::timestampz, '1
minute'::interval) dd
)
(
select
    to_char(date_time, 'YYYYMMDDHH24MI')::bigint as id,
    date_time,
    date_time::date as date,
    date_part('isodow', date_time)::int as week_day,
    date_part('week', date_time)::int as week_number,
    date_part('month', date_time)::int as month,
    date_part('isoyear', date_time)::int as year,
    (date_part('isodow', date_time)::smallint between 1 and 5)::int as work_day
from dates
order by date_time
);
```

### Таблица dim\_passengers

Таблица представляет собой справочник пассажиров. Гранулярность – один пассажир. Тип SCD – 1, подразумеваем, что, если у пассажира изменится контактная информация, то значения в таблице будут обновляться (изменившееся значение будет перезаписано новым), если изменится ФИО, то будет создан новый пассажир.

Таблица содержит поля:

id - суррогатный первичный ключ;

passenger\_id – идентификатор пассажира в исходной таблице;

passenger\_name – ФИО пассажира;

phone – контактный телефон;

email – адрес электронной почты.

ETL-трансформация по наполнению таблицы представлена на рисунке 2.

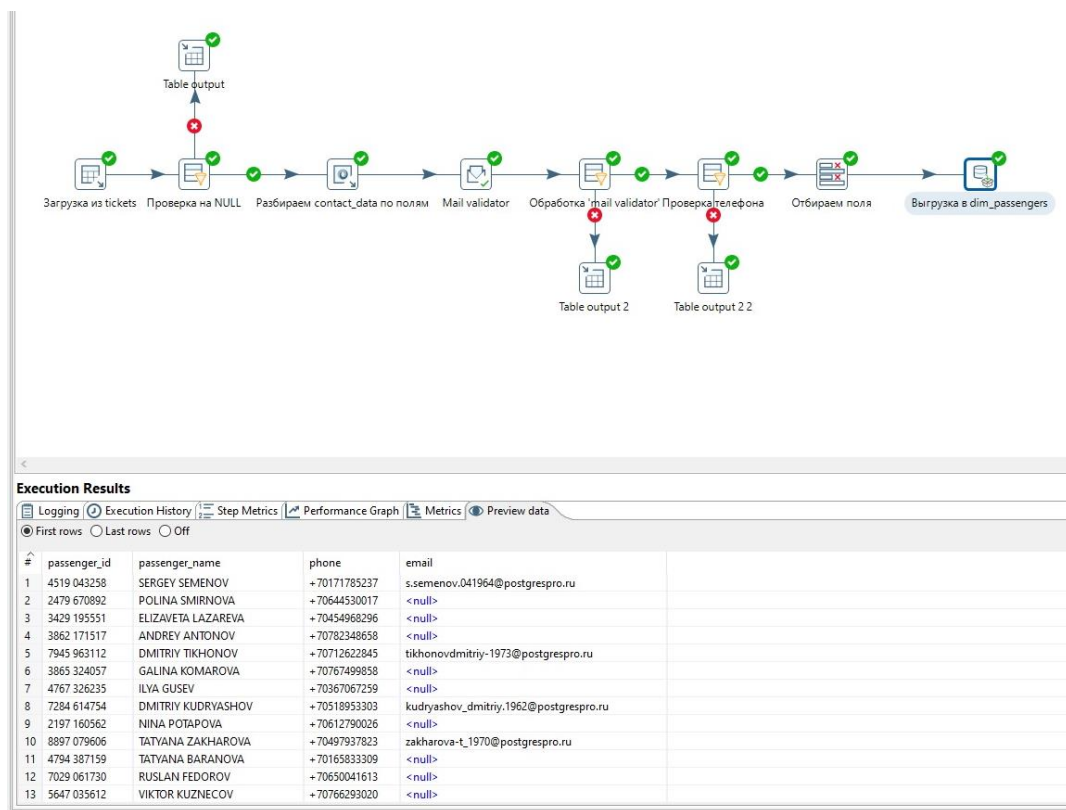


Рисунок 2 Заполнение dim\_passengers

Описание трансформации.

Первым шагом производится чтение данных из таблицы bookings.tickets.

Далее проверяем поле passenger\_name на NULL, отсутствующие значения записываем в отдельную таблицу.

Следующим шагом разбираем json-массив в поле contact\_data по составляющим, чтобы потом записать отдельно телефон и адрес почты.

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 отдельном шаге отбираем нужные поля, после чего производится запись данных в целевую таблицу dim\_passengers.

### Таблица dim\_ aircrafts

Таблица представляет собой справочник самолетов с указанием мест и класса обслуживания. Гранулярность – одно посадочное место в самолете. Тип SCD – 0, подразумеваем, что значения в таблице не изменяются.

Таблица содержит поля:

id - суррогатный первичный ключ;

aircraft\_code – код самолета;

model – наименование самолета;

range – дальность полета;

qty\_seats – количество посадочных мест;

seat\_no – номер посадочного места;

fare\_conditions – класс обслуживания, закрепленный за посадочным местом.

ETL-трансформация по наполнению таблицы представлена на рисунке 3.

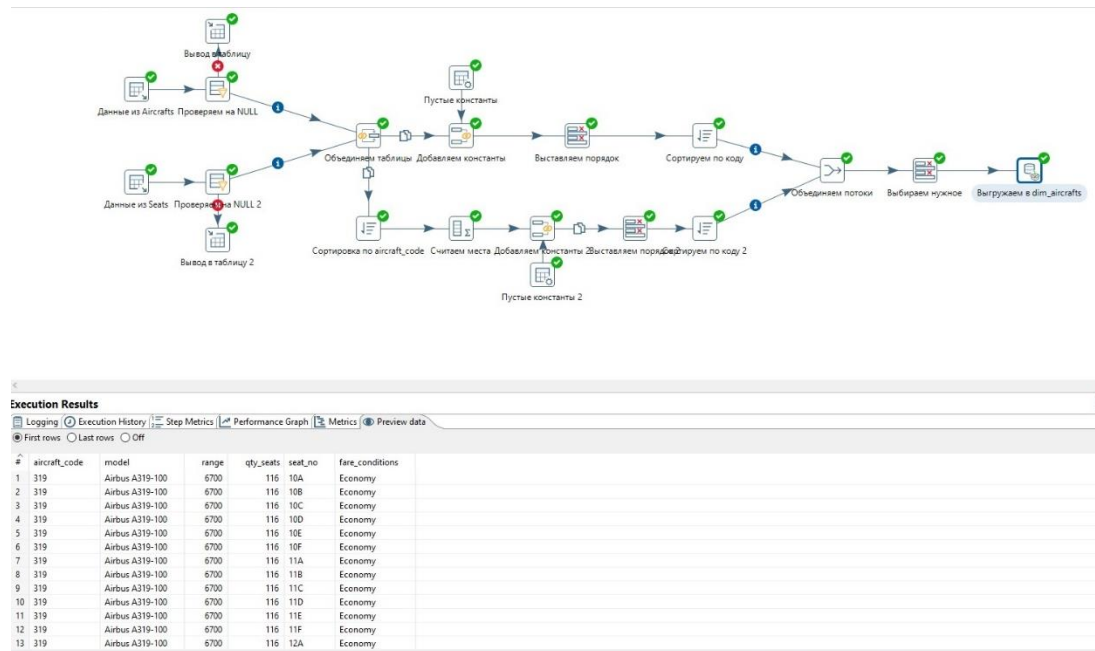


Рисунок 3 Заполнение dim\_aircrafts

Описание трансформации.

Сначала производится чтение данных из таблиц bookings.aircrafts и bookings.seats, данные сразу сортируются по aircraft\_code.

Далее поток данных из bookings.aircrafts проверяется на отсутствие NULL в поле model, а поток из bookings.seats – в поле seat\_no, отсутствующие значения записываем в отдельную таблицу.

После проверки потоки объединяются при помощи шага Merge Join по коду aircraft\_code (потоки уже отсортированы).

Чтобы получить для каждого самолета общее число мест, разделим поток на копии.

В одной копии потока произведём сортировку по aircraft\_code и добавим шаг Group By, в котором посчитаем для каждого самолета количество мест. Подмешаем к потоку в виде пустых констант недостающие поля из другой копии потока, произведём выбор нужных полей и отсортируем для объединения.

В другой копии потока произведём добавление пустой константы в виде поля qty\_seats для однообразия обеих копий потоков, выберем нужные поля и отсортируем для объединения.

После объединения копий потоков производим окончательную выборку нужных полей и производим запись данных в целевую таблицу dim\_aircrafts.

### Таблица dim\_airports

Таблица представляет собой справочник аэропортов. Гранулярность – один аэропорт. Тип SCD – 0, подразумеваем, что значения в таблице не изменяются.

Таблица содержит поля:

id - суррогатный первичный ключ;

airport\_code – код аэропорта;

airport\_name – наименование аэропорта;

city – дальность полета;

longitude – географическая долгота расположения аэропорта;

latitude – географическая широта расположения аэропорта;

timezone – часовой пояс места расположения аэропорта.

ETL-трансформация по наполнению таблицы представлена на рисунке 4.

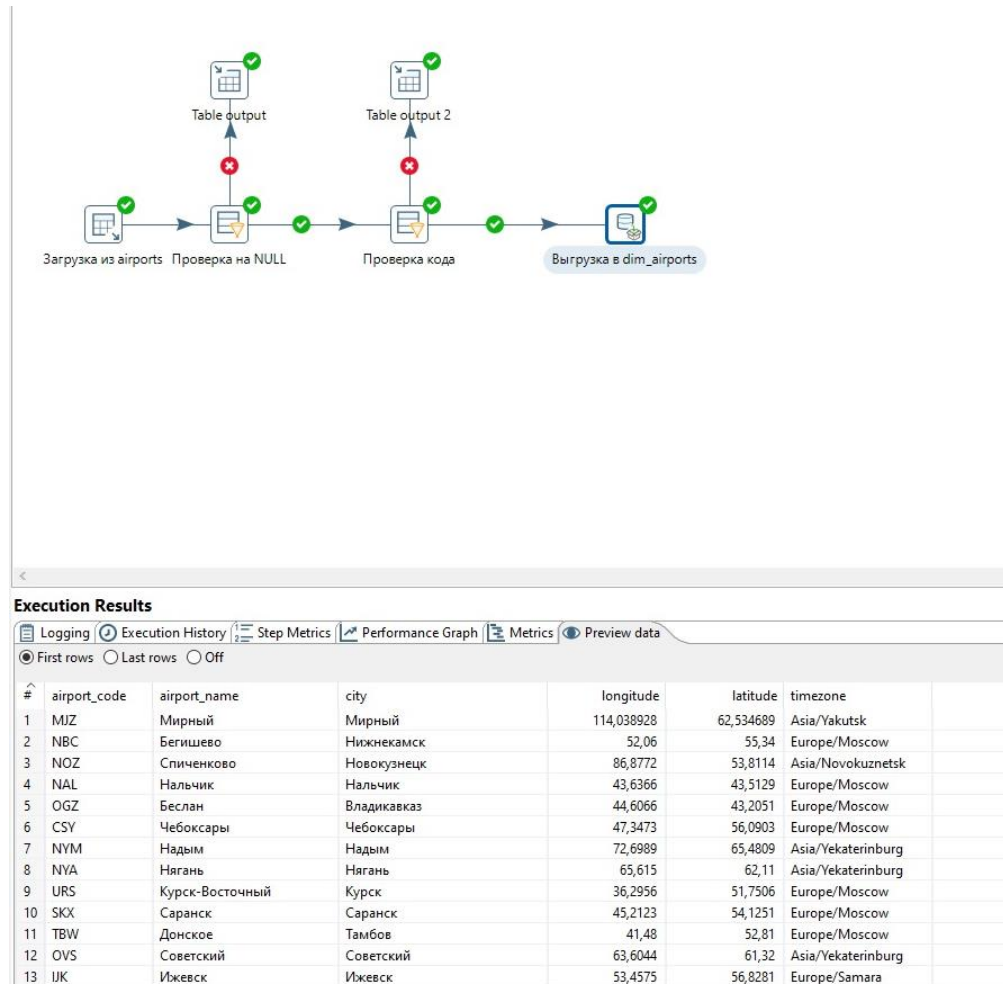


Рисунок 4 Заполнение dim\_airports

Описание трансформации.

Сначала производится чтение данных из таблицы bookings.airports.

Далее производится проверка на отсутствие NULL в полях airport\_code и city, отсутствующие значения записываем в отдельную таблицу.

Следующим шагом производится проверка на ошибки в коде аэропорта через регулярное выражение: проверяется, что количество символов равно трём, ошибочные записи записываем в отдельную таблицу.

После проверок производим запись данных в целевую таблицу dim\_airports.

### Таблица dim\_tariff

Таблица представляет собой справочник тарифов перелета между городами с указанием класса обслуживания и соответствующей ему стоимости.

Гранулярность – один перелет. Тип SCD – 2, для его реализации в таблицу добавлен столбец «is\_current», содержащий флаг актуальности.

Таблица содержит поля:

id - суппрогатный первичный ключ;

flight\_no – идентификатор перелета из исходной таблицы;

departure\_airport – код аэропорта отправления;

departure\_airport\_name – название аэропорта отправления;

departure\_city – город отправления;

arrival\_airport – код аэропорта прибытия;

arrival\_airport\_name – название аэропорта прибытия;

arrival\_city – город прибытия;

fare\_conditions – класс обслуживания;

amount – стоимость тарифа;

is\_current – флаг актуальности тарифа.

ETL-трансформация по наполнению таблицы представлена на рисунке 5.

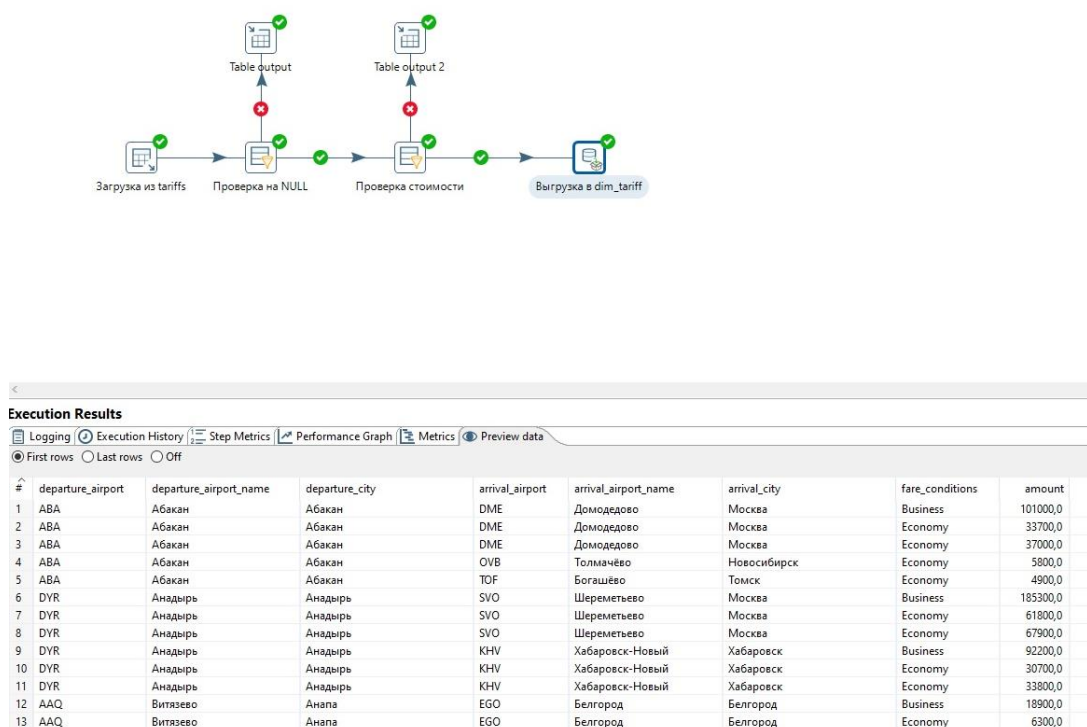


Рисунок 5 Заполнение dim\_tariff

Описание трансформации.

Для удобства в исходной схеме Bookings было создано представление bookings.tariffs, в котором при помощи SQL-запроса были объединены необходимые таблицы, содержащие информацию по аэропортам, перелетам, классу обслуживания и стоимости.

Текст SQL-запроса:

```
create or replace view tariffs as
select distinct flight_no,
       departure_airport,
       departure_airport_name,
       departure_city,
       arrival_airport,
       arrival_airport_name,
       arrival_city,
       fare_conditions,
       amount
from flights_v fv join ticket_flights tf using(flight_id)
order by departure_city, arrival_city, fare_conditions;
```

Первым шагом трансформации производится чтение данных из представления bookings.tariffs.

Далее производится проверка на отсутствие NULL в полях fare\_conditions и amount, отсутствующие значения записываем в отдельную таблицу.

Следующим шагом производится проверка на ошибки в стоимости: проверяется, что стоимость больше 10 и меньше 1000000 (автоматически проверяются неотрицательные значения стоимости), ошибочные записи записываем в отдельную таблицу.

После проверок производим запись данных в целевую таблицу dim\_tariff.

### Таблица fact\_flights

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

Таблица содержит поля:

id - супрогатный первичный ключ;

passenger\_id – внешний ключ к id пассажира в таблице измерений dim\_passengers;

actual\_departure\_id – внешний ключ к id фактической даты и времени вылета в таблице dim\_calendar;

actual\_arrival\_id – внешний ключ к id фактической даты и времени прибытия в таблице dim\_calendar;

delay\_departure - задержка вылета (разница между фактической и запланированной датой/временем в секундах);

delay\_arrival - задержка прибытия (разница между фактической и запланированной датой/временем в секундах);

aircraft\_id - внешний ключ к id самолета в таблице измерений dim\_aircraft;

departure\_airport\_id - внешний ключ к id аэропорта отправления в таблице измерений dim\_airports;

arrival\_airport\_id - внешний ключ к id аэропорта прибытия в таблице измерений dim\_airports;

tariff\_id - внешний ключ к id тарифа в таблице измерений dim\_tariff.

ETL-трансформация по наполнению таблицы представлена на рисунке 6.

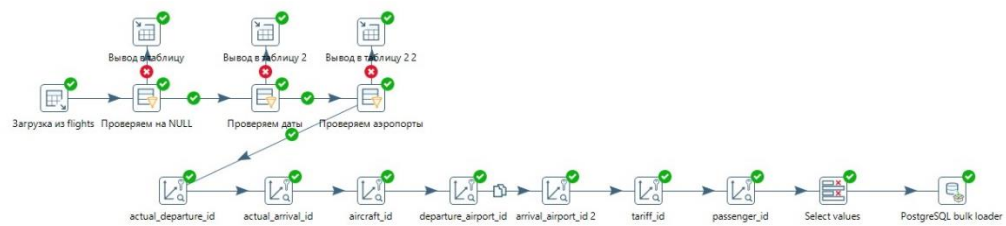


Рисунок 6 Заполнение fact\_flights

Описание трансформации.

Для удобства в исходной схеме Bookings было создано представление bookings.fact\_flights, в котором при помощи SQL-запроса были объединены необходимые таблицы и выполнены необходимые вычисления.

Текст SQL-запроса:

```
create or replace view fact_flights as
select
    f.flight_id,
    passenger_id,
    actual_departure,
    actual_arrival,
    abs(extract(epoch from actual_departure) - extract(epoch from scheduled_departure)) as delay_departure,
    abs(extract(epoch from actual_arrival) - extract(epoch from scheduled_arrival)) as delay_arrival,
    aircraft_code,
    seat_no,
    departure_airport,
    arrival_airport,
    flight_no,
    fare_conditions
from flights f join ticket_flights tf using(flight_id)
join boarding_passes bp
on tf.flight_id = bp.flight_id and tf.ticket_no = bp.ticket_no
join tickets t
on tf.ticket_no = t.ticket_no
where f.status = 'Arrived'
order by f.flight_id;
```

Первым шагом трансформации производится чтение данных из представления bookings.fact\_flights.

Далее производится проверка на отсутствие NULL в полях passenger\_id, departure\_airport и arrival\_airport, отсутствующие значения записываем в отдельную таблицу.

Следующим шагом производится проверка дат: дата отправления должна быть меньше даты прибытия, ошибочные записи записываем в отдельную таблицу.

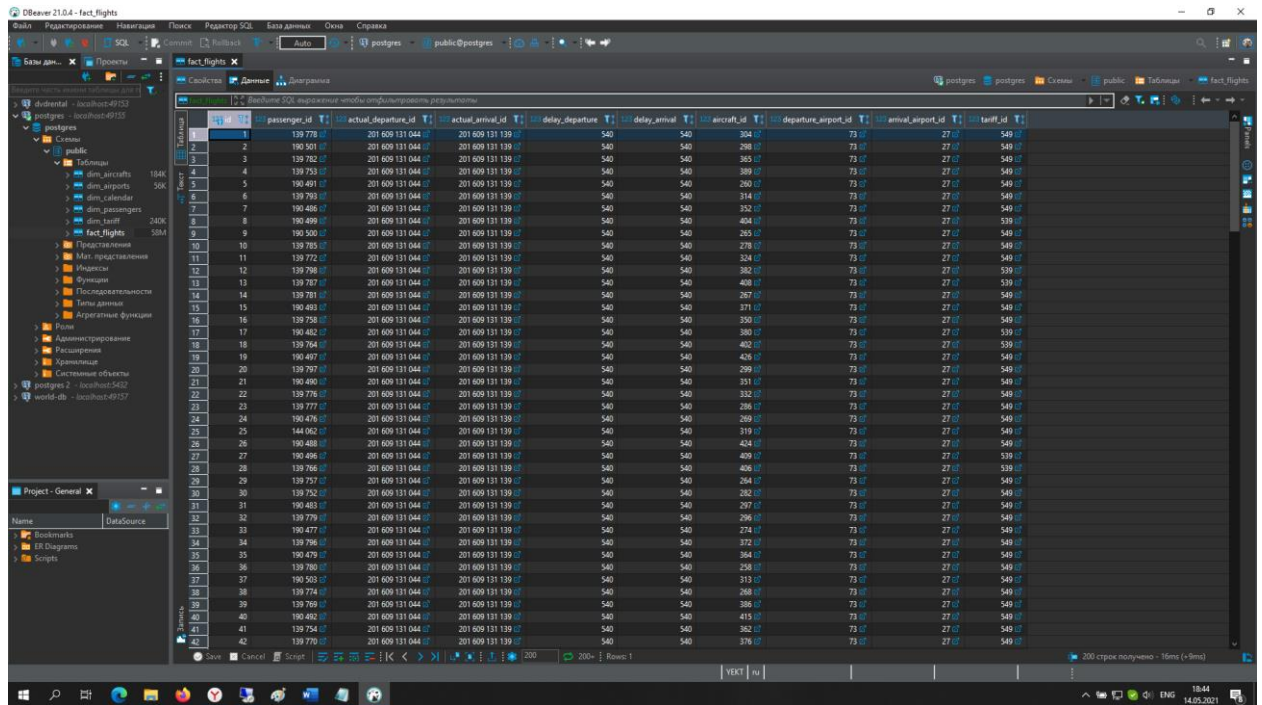
Далее проверяем аэропорты: аэропорт отправления должен быть не равен аэропорту прибытия, ошибочные записи записываем в отдельную таблицу.



После проверок производим серию шагов «Combination lookup/update» по извлечению id из таблиц измерений и замене ими соответствующих значений в исходной таблице.

После того, как все значения исходной таблицы (кроме значений delay\_departure и delay\_arrival) заменены соответствующими значениями их id из таблиц измерений при помощи шага «Select values» отбираем только нужные поля и производим запись данных в целевую таблицу fact\_flights.

Фрагмент таблицы fact\_flights приведён на рисунке 7.



	passenger_id	actual_departure_id	actual_arrival_id	delay_departure	delay_arrival	aircraft_id	departure_airport_id	arrival_airport_id	tariff_id
1	139 778	201 609 131 044	201 609 131 139	540	540	304	73	27	549
2	190 501	201 609 131 044	201 609 131 139	540	540	298	73	27	549
3	139 782	201 609 131 044	201 609 131 139	540	540	365	73	27	549
4	139 751	201 609 131 044	201 609 131 139	540	540	389	73	27	549
5	190 481	201 609 131 044	201 609 131 139	540	540	282	73	27	549
6	139 793	201 609 131 044	201 609 131 139	540	540	314	73	27	549
7	190 486	201 609 131 044	201 609 131 139	540	540	352	73	27	549
8	190 489	201 609 131 044	201 609 131 139	540	540	404	73	27	539
9	190 500	201 609 131 044	201 609 131 139	540	540	265	73	27	549
10	139 785	201 609 131 044	201 609 131 139	540	540	278	73	27	549
11	139 772	201 609 131 044	201 609 131 139	540	540	324	73	27	549
12	139 798	201 609 131 044	201 609 131 139	540	540	382	73	27	539
13	139 787	201 609 131 044	201 609 131 139	540	540	408	73	27	539
14	139 781	201 609 131 044	201 609 131 139	540	540	267	73	27	549
15	190 493	201 609 131 044	201 609 131 139	540	540	371	73	27	549
16	139 758	201 609 131 044	201 609 131 139	540	540	350	73	27	549
17	190 482	201 609 131 044	201 609 131 139	540	540	380	73	27	539
18	139 784	201 609 131 044	201 609 131 139	540	540	402	73	27	539
19	190 497	201 609 131 044	201 609 131 139	540	540	426	73	27	549
20	139 797	201 609 131 044	201 609 131 139	540	540	299	73	27	549
21	190 490	201 609 131 044	201 609 131 139	540	540	331	73	27	549
22	139 776	201 609 131 044	201 609 131 139	540	540	332	73	27	549
23	139 777	201 609 131 044	201 609 131 139	540	540	286	73	27	549
24	190 476	201 609 131 044	201 609 131 139	540	540	269	73	27	549
25	144 062	201 609 131 044	201 609 131 139	540	540	319	73	27	549
26	190 488	201 609 131 044	201 609 131 139	540	540	424	73	27	549
27	190 496	201 609 131 044	201 609 131 139	540	540	409	73	27	539
28	139 766	201 609 131 044	201 609 131 139	540	540	406	73	27	539
29	139 757	201 609 131 044	201 609 131 139	540	540	264	73	27	549
30	139 752	201 609 131 044	201 609 131 139	540	540	282	73	27	549
31	190 483	201 609 131 044	201 609 131 139	540	540	297	73	27	549
32	139 779	201 609 131 044	201 609 131 139	540	540	296	73	27	549
33	190 477	201 609 131 044	201 609 131 139	540	540	274	73	27	549
34	139 796	201 609 131 044	201 609 131 139	540	540	372	73	27	549
35	190 479	201 609 131 044	201 609 131 139	540	540	364	73	27	549
36	139 780	201 609 131 044	201 609 131 139	540	540	258	73	27	549
37	190 503	201 609 131 044	201 609 131 139	540	540	313	73	27	549
38	139 774	201 609 131 044	201 609 131 139	540	540	268	73	27	549
39	139 769	201 609 131 044	201 609 131 139	540	540	386	73	27	549
40	190 492	201 609 131 044	201 609 131 139	540	540	415	73	27	549
41	139 754	201 609 131 044	201 609 131 139	540	540	362	73	27	549
42	139 770	201 609 131 044	201 609 131 139	540	540	376	73	27	549

Рисунок 7 Фрагмент таблицы fact\_flights

Приложение:

SQL-скрипты создания таблиц, ETL-трансформации.