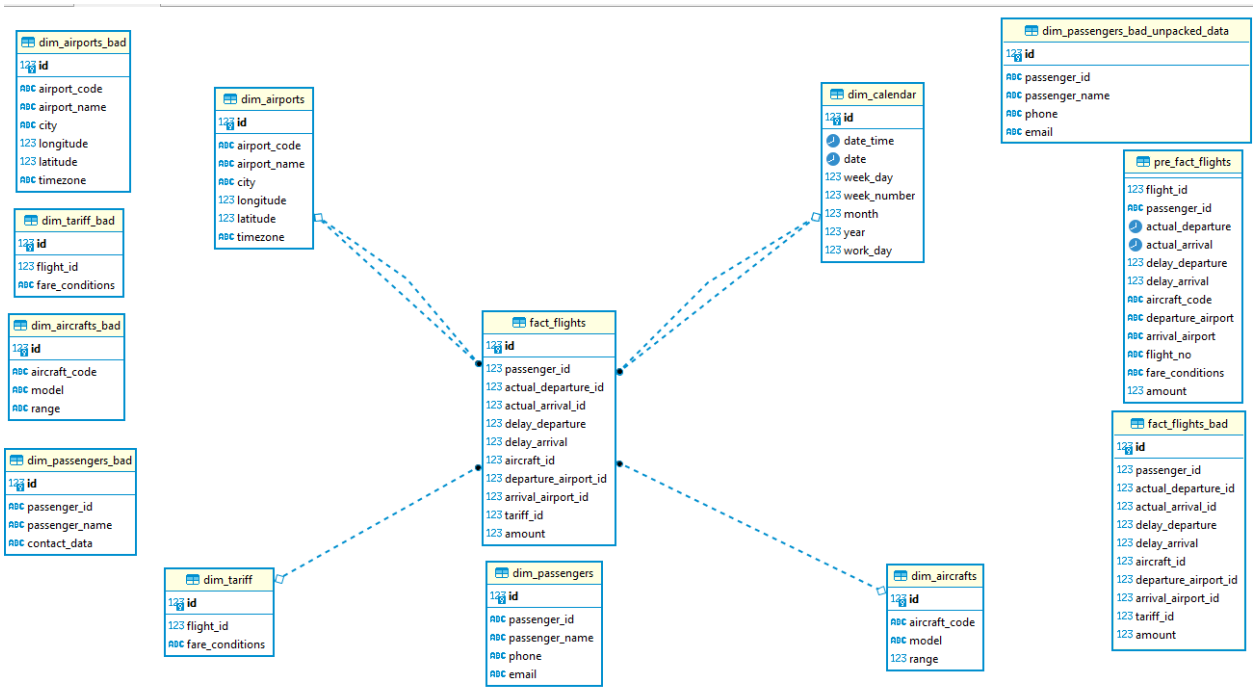
Цели проекта:

1) Создать таблицы измерений и таблицу фактов в PostreSQL. Можете развернуть базу данных локально или в Docker-контейнере.

В качестве альтернативы можете таблицы с фактами и размерностями в csv файлы (каждая таблица - отдельный csv-файл), если не удается запустить PostgreSQL.

2)Наполнить базу данными из бд bookings при помощи ETL

Конечный результат в виде диаграммы схемы dwhfinal :



Источником данных в схеме является локальная схема БД Bookings

Справочники:

Dim_Calendar - справочник дат

Dim_Passengers - справочник пассажиров

Dim_Aircrafts - справочник самолетов

Dim_Airports - справочник аэропортов

Dim_Tariff - справочник тарифов (Эконом/бизнес и тд)

Справочники с данными, которые не прошли проверку в рамках ETL процесса:

Dim_aircrafts_bad

Dim_airports_bad

Dim_passengers_bad

Dim_passengers_bad_unpacked_data (некорректные данные после распаковки JSON)

Dim_tariff_bad

Fact_flights_bad

Подготовительная таблица с данными по перелетам.

Pre_fact_flights

Итоговая таблица

Fact_flights

Описание и наполнение таблиц базы данных

Справочник dim_calendar

Таблица представляет собой справочник дат и времени совершения вылетов самолетов.

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:

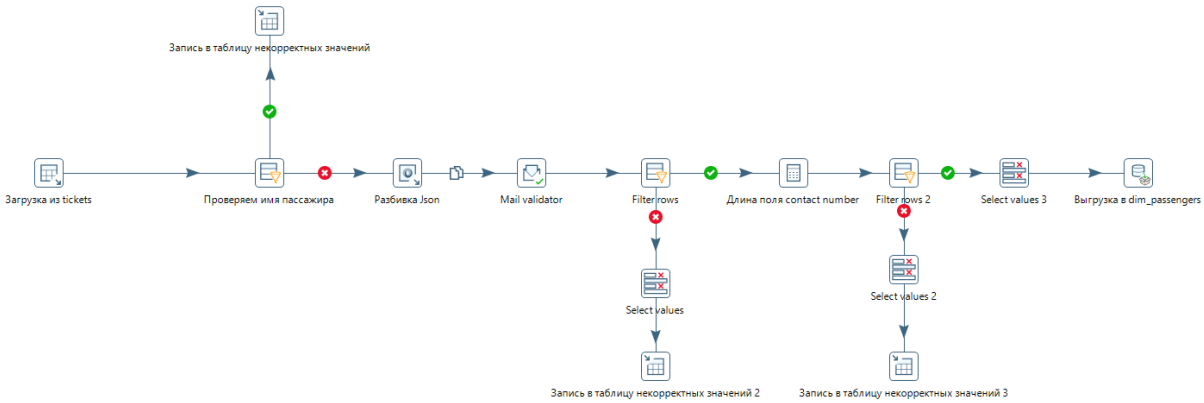
```
create table dwhfinal.dim_calendar as
with dates as (
	select dd as date_time
	from generate_series('2016-08-01'::timestamptz, '2016-12-30'::timestamptz, '1
minute'::interval) dd
                    )
(
select
	to_char(date_time, 'YYYYMMDDHH24MI')::bigint as id,
	date_time,
	date_time::date as date,
	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
);
alter table dwhfinal.dim_calendar add primary key(id);
```

Справочник dim_passengers

Таблица представляет собой справочник пассажиров.

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 **dim_passengers:**



Данные получаем с помощью sql запроса из таблицы bookings.tickets

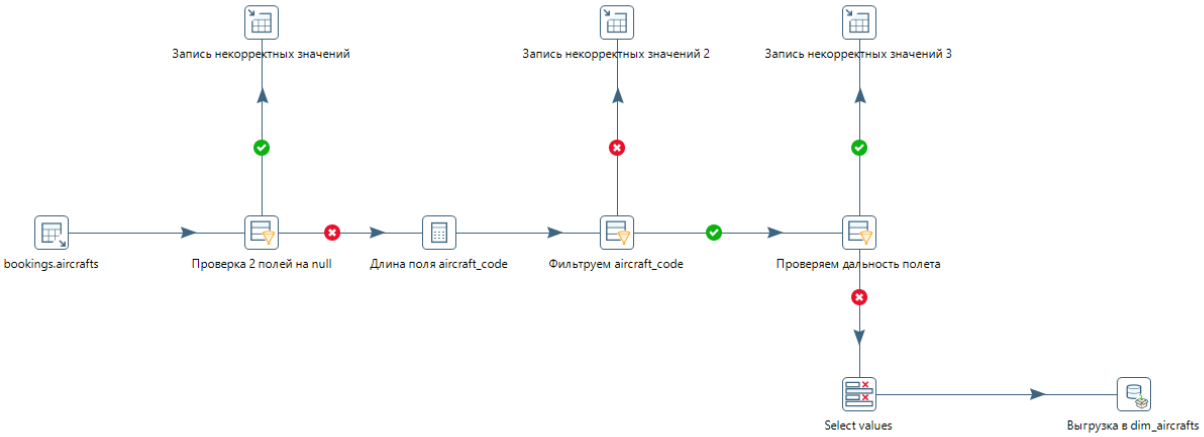
```
SELECT passenger_id, passenger_name, contact_data
FROM bookings.tickets
```

Проверки внутри трансформации:
1. Passenger name не должно быть null
2. Проверка Mail validator
3. Проверка длины поля номер телефона (должно быть 12 символов включая +)
Данные не прошедшие проверку отправляются в таблицы Dim_passengers_bad
и Dim_passengers_bad_unpacked_data (некорректные данные после распаковки JSON)

Данные прошедшие проверку отправляются в справочник **dim_passengers.**

Справочник dim_aircrafts
Таблица представляет собой справочник самолетов
Таблица содержит поля:
id - суррогатный первичный ключ;
aircraft_code – код самолета;
model – наименование самолета;
range – дальность полета;

ETL-трансформация по наполнению таблицы **dim_aircrafts:**



Данные получаем с помощью sql запроса из таблицы bookings.aircrafts

```
select distinct * from bookings.aircrafts
order by aircraft_code
```

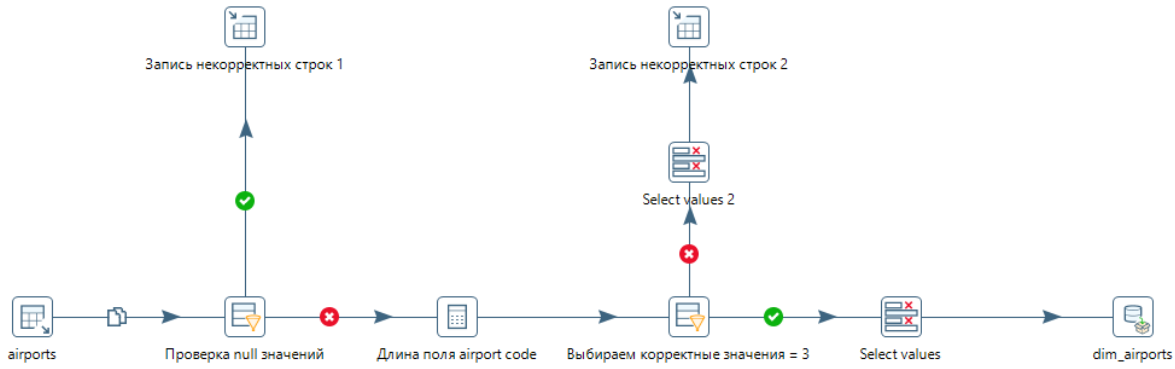
Проверки внутри трансформации:
1. aircraft_code и model не должны быть null
2. Проверка длины поля aircraft_code (должно быть 3 символа)
3. Проверка дальности полета (поле не должно быть больше 20000)
Данные не прошедшие проверку отправляются в таблицу Dim_aircrafts_bad
Данные прошедшие проверку отправляются в справочник **dim_aircrafts.**

Справочник dim_airports

Таблица представляет собой справочник аэропортов.

Таблица содержит поля:
id - суррогатный первичный ключ;
airport_code – код аэропорта;
airport_name – наименование аэропорта;
city – город;
longitude – географическая долгота расположения аэропорта;
latitude – географическая широта расположения аэропорта;
timezone – часовой пояс места расположения аэропорта.

ETL-трансформация по наполнению таблицы **dim_airports**



Данные получаем с помощью sql запроса из таблицы bookings.airports

```
SELECT *
FROM bookings.airports
```

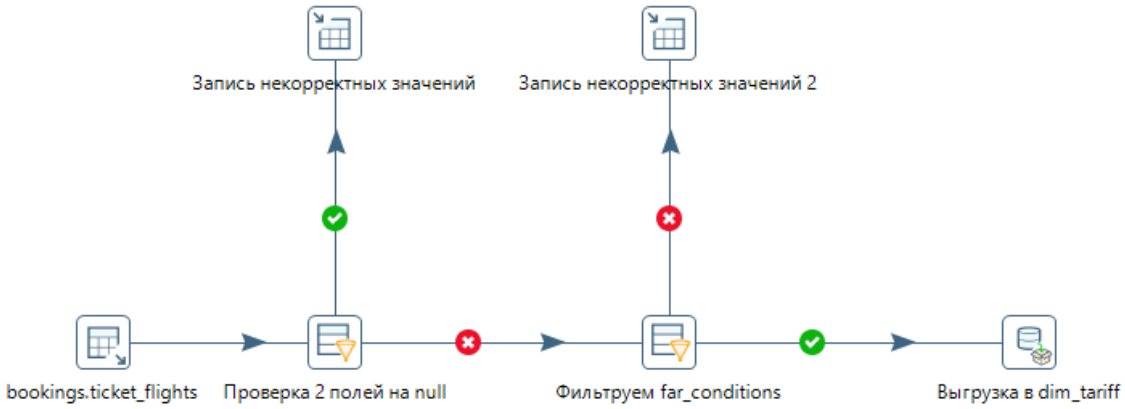
Проверки внутри трансформации:
1. airport_code и airport_name не должны быть null
2. Проверка длины поля airport_code (должно быть 3 символа)
Данные не прошедшие проверку отправляются в таблицу Dim_airports_bad
Данные прошедшие проверку отправляются в справочник **dim_airports.**

Справочник **dim_tariff**

Таблица представляет собой справочник тарифов перелета между городами.

Таблица содержит поля:
id - суррогатный первичный ключ;
flight_id– идентификатор перелета из исходной таблицы;
fare_conditions – класс обслуживания;

ETL-трансформация по наполнению таблицы **dim_tariff:**



Данные получаем с помощью sql запроса из таблицы bookings.airports

```
select
distinct
flight_id
,fare_conditions
from bookings.ticket_flights
```

Проверки внутри трансформации:
1. flight_id и fare_conditions не должны быть null
2. Поле fare_conditions должно содержать одно из 3 значений (business, economy, comfort)

Данные не прошедшие проверку отправляются в таблицу Dim_tariff_bad
Данные прошедшие проверку отправляются в справочник **dim_tariff.**

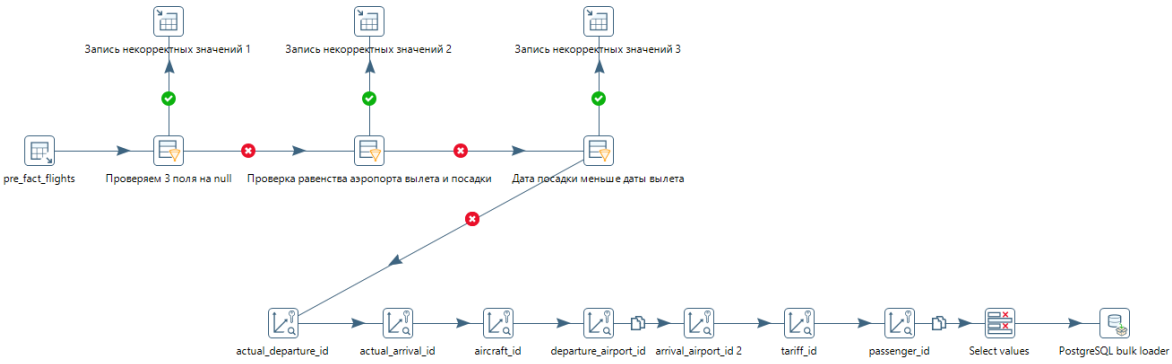
Таблица **fact_flights**

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

Таблица содержит поля:
id - суррогатный 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.
Amount – стоимость перелёта

ETL-трансформация по наполнению таблицы



Для загрузки данных из единой таблицы и для предварительных необходимых вычислений была создана подготовительная таблица с данными по перелетам **Pre_fact_flights.**

Sql код:

```
create table dwhfinal.pre_fact_flights as
select
	f.flight_id,
	t.passenger_id,
	f.actual_departure,
	f.actual_arrival,
	abs(extract(epoch from f.actual_departure) - extract(epoch from
f.scheduled_departure)) as delay_departure,
	abs(extract(epoch from f.actual_arrival) - extract(epoch from
f.scheduled_arrival)) as delay_arrival,
	f.aircraft_code,
	f.departure_airport,
	f.arrival_airport,
	f.flight_no,
	tf.fare_conditions,
	tf.amount
from bookings.flights f
inner join bookings.boarding_passes bp
```

```
        on f.flight_id = bp.flight_id

        inner join bookings.ticket_flights tf
        on f.flight_id = tf.flight_id
        and tf.ticket_no = bp.ticket_no

        inner join bookings.tickets t
        on tf.ticket_no = t.ticket_no

where f.status = 'Arrived'
order by f.flight_id;
```

Данные получаем с помощью sql запроса из таблицы **Pre_fact_flights**

```
SELECT *
FROM dwhfinal.pre_fact_flights
```

Проверки внутри трансформации:
1. passenger_id, arrival_airport, departure_airport не должны быть null
2. departure_airport <> arrival_airport
3. actual_arrival > actual_departure

Данные не прошедшие проверку отправляются в таблицу fact_flights_bad
Данные прошедшие проверку отправляются в таблицу **fact_flights.**

Дополнительная информация:
В рамках выполнения работы столкнулся с проблемами производительности по типу bottle neck: в трансформации по наполнению **fact_flights** были проблемы с матчингом passenger_id. Проблема была решена с помощью создания индекса в таблице **dim_passengers**

```
CREATE INDEX idx_passenger_id
ON dwhfinal.dim_passengers(passenger_id)
```