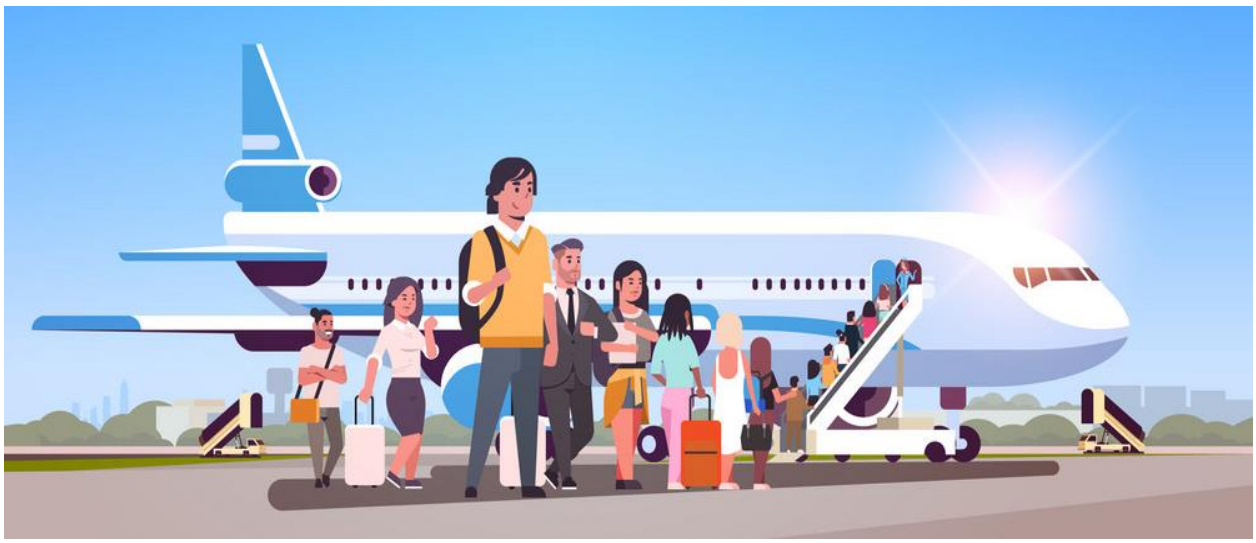

АВИАПЕРЕВОЗКИ ПАССАЖИРОВ

ИСПОЛНИТЕЛЬ ПРОЕКТА: Кляпко Владислав Андреевич

ЦЕЛИ ПРОЕКТА:

1 Спроектировать структурированное хранилище данных;

2 Организовать процесс извлечения данных из внешнего источника;

3 Организовать процесс трансформации извлечённых данных;

4 Организовать процесс загрузки преобразованных данных.

ЗАДАЧИ ПРОЕКТА:

1 Создать чистую базу данных с таблицами фактов и измерений;

2 Обогатить созданную базу сущностями из демонстрационной базы данных demo.bookings при помощи ETL-процессов;

3 Для каждой таблицы в хранилище, необходимо реализовать несколько проверок качества данных.

Требования заказчика к проекту

Хранилище данных должно содержать следующие таблицы:

- **dim_calendar -** справочник дат;
- **dim_passengers -** справочник пассажиров;
- **dim_aircrafts -** справочник самолетов;
- **dim_airports -** справочник аэропортов;
- **dim_tariff -** справочник тарифов (эконом/бизнес и т.д.);
- **fact_flights –** содержит совершенные перелеты.

Перечень столбцов в таблице **fact_flights**:

- Пассажир;
- Дата и время вылета (факт);
- Дата и время прилета (факт);
- Задержка вылета (разница между фактической и запланированной датой в секундах);
- Задержка прилета (разница между фактической и запланированной датой в секундах);
- Самолет;
- Аэропорт вылета;
- Аэропорт прилета;
- Класс обслуживания;
- Стоимость.

Если в рамках билета был сложный маршрут с пересадками – каждый сегмент учитывается независимо.

1 Анализ исходных данных

В работе использовался локальный тип подключения, программное обеспечение - DBeaver. В качестве исходных данных используется демонстрационная база данных «Авиаперевозки» (версия medium, СУБД PostgreSQL) с данными о перелётах за три месяца.

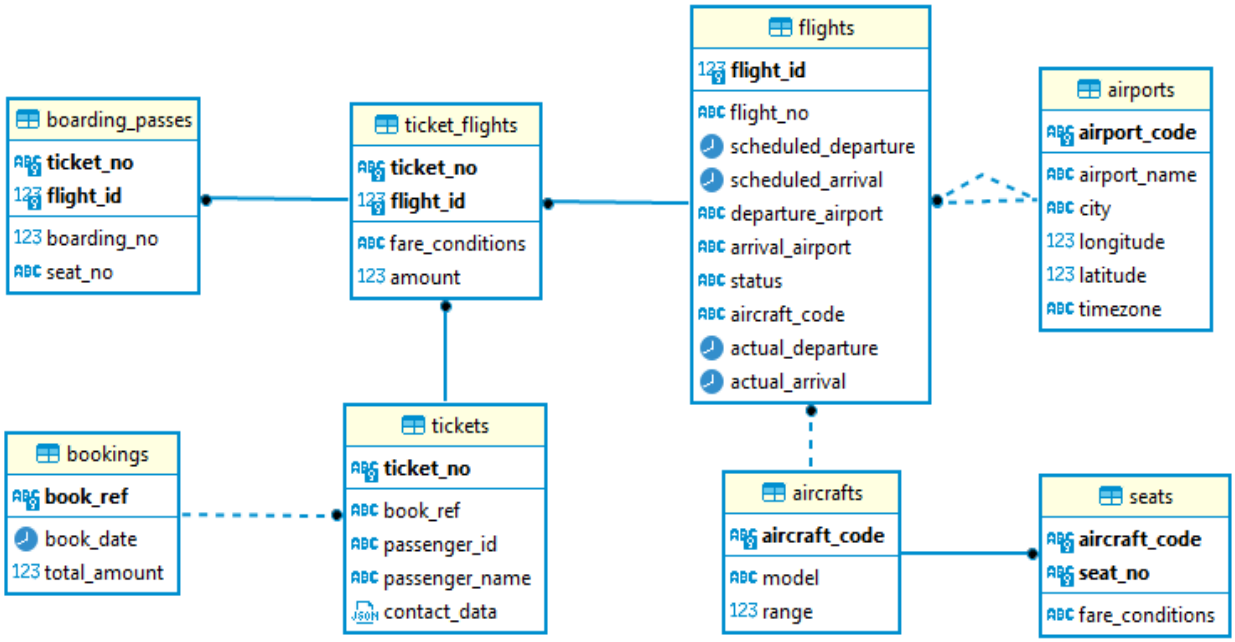

Рисунок 1 – Диаграмма схемы данных демонстрационной базы данных PostgreSQL «Авиаперевозки» (версия medium)

1.1 Описание схемы источника данных

Основной сущностью является бронирование (bookings).

В одно бронирование можно включить несколько пассажиров, каждому из которых выписывается отдельный билет (tickets). Билет имеет уникальный номер и содержит информацию о пассажире. Как таковой пассажир не является отдельной сущностью. Как имя, так и номер документа пассажира могут меняться с течением времени, так что невозможно однозначно найти все билеты одного человека; для простоты можно считать, что все пассажиры уникальны.

Билет включает один или несколько перелетов (ticket_flights). Несколько перелетов могут включаться в билет в случаях, когда нет прямого рейса, соединяющего пункты отправления и назначения (полет с пересадками), либо когда билет взят «туда и обратно». В схеме данных нет жесткого ограничения, но предполагается, что все билеты в одном бронировании имеют одинаковый набор перелетов.

Каждый рейс (flights) следует из одного аэропорта (airports) в другой. Рейсы с одним номером имеют одинаковые пункты вылета и назначения, но будут отличаться датой отправления.

При регистрации на рейс пассажиру выдается посадочный талон (boarding_passes), в котором указано место в самолете. Пассажир может зарегистрироваться только на тот рейс, который есть у него в билете. Комбинация рейса и места в самолете должна быть уникальной, чтобы не допустить выдачу двух посадочных талонов на одно место.

Количество мест (seats) в самолете и их распределение по классам обслуживания зависит от модели самолета (aircrafts), выполняющего рейс. Предполагается, что каждая модель самолета имеет только одну компоновку салона. Схема данных не контролирует, что места в посадочных талонах соответствуют имеющимся в самолете (такая проверка может быть сделана с использованием табличных триггеров или в приложении).

2 Создание хранилища данных конечного использования

В качестве модели данных была спроектирована схема «звезда», представляющая собой централизованное хранилище данных, состоящее из таблиц фактов и измерений.

Схема разбивает таблицу фактов на ряд денормализованных таблиц измерений. Таблица фактов содержит агрегированные данные, которые будут использоваться для составления отчетов, а таблица измерений описывает хранимые данные.

Таблицы измерений:

- **dim_calendar** - справочник дат;
- **dim_passengers** - справочник пассажиров;
- **dim_aircrafts** - справочник самолетов;
- **dim_airports** - справочник аэропортов;
- **dim_tariff** - справочник тарифов.

Таблица фактов:

- **fact_flights** - совершенные перелеты.

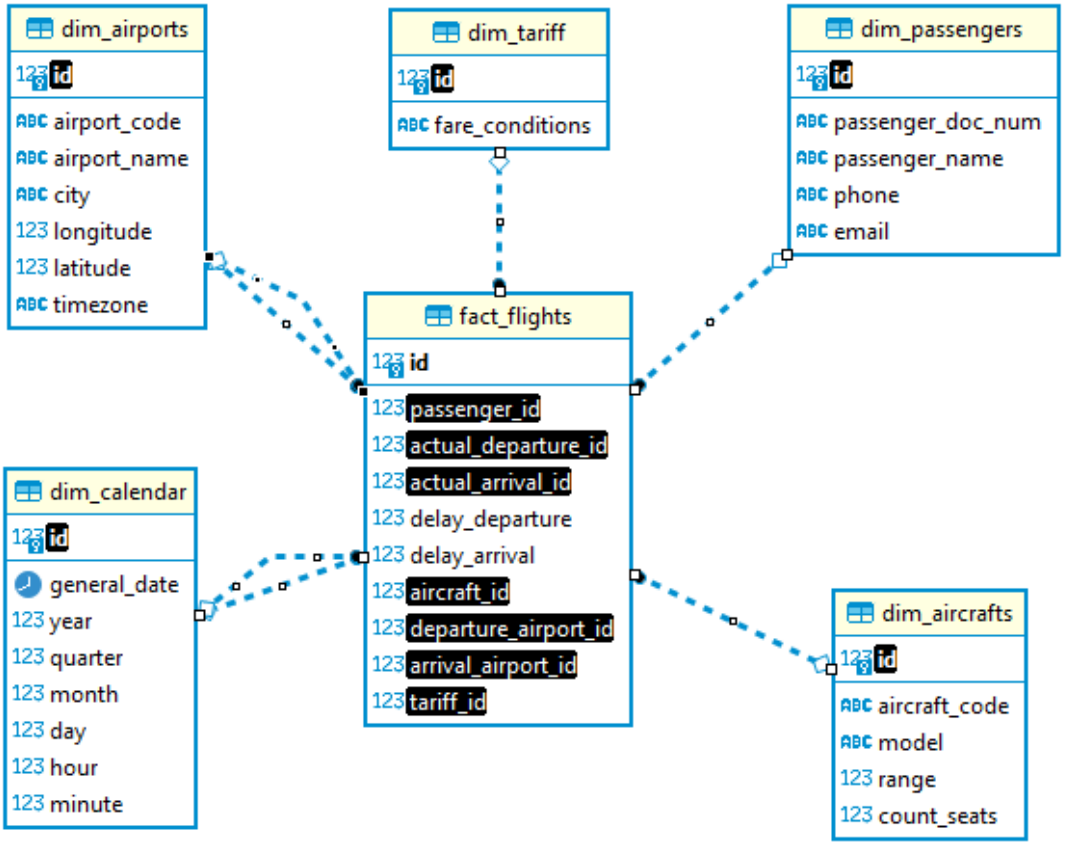

Рисунок 2 – Диаграмма схемы данных пустой базы данных PostgreSQL с таблицами измерений (справочников) и фактов

2.1 Описание таблиц хранилища данных конечного использования

Таблица dim_calendar

Таблица содержит информацию о дате и времени перелётов. Гранулярность – 1 мин. Тип SCD – 0, это означает, что значения в таблице не изменяются.

Содержание полей:

- **id** – суррогатный первичный ключ;
- **general_date** – дата и время;
- **year** – год;
- **quarter** – квартал;
- **month** – месяц;
- **day** – день;
- **hour** – час;
- **minute** – минута.

Таблица dim_passengers

Таблица содержит информацию о пассажирах. Гранулярность – один пассажир. Тип SCD – 1, это означает, что в случае, 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.

Содержание полей:

- **id** – суррогатный первичный ключ;
- **passenger_doc_num** – номер документа пассажира;
- **passenger_name** – фамилия и имя пассажира;
- **phone** – контактный телефон;
- **email** – адрес электронной почты.

Таблица dim_aircrafts

Таблица содержит информацию о самолетах с указанием мест и класса обслуживания. Гранулярность – одно посадочное место в самолете. Тип SCD – 0.

Содержание полей:

- **id** - суррогатный первичный ключ;
- **aircraft_code** – код самолета;

- **model** – модель самолета;
- **range** – дальность полета;
- **count_seats** – количество посадочных мест.

Таблица dim_airports

Таблица содержит информацию об аэропортах отправления и прибытия. Гранулярность – один аэропорт. Тип SCD – 0.

Содержание полей:

- **id** - суррогатный первичный ключ;
- **airport_code** – код аэропорта;
- **airport_name** – наименование аэропорта;
- **city** – город, в котором расположен аэропорт;
- **longitude** – географическая долгота расположения аэропорта;
- **latitude** – географическая широта расположения аэропорта;
- **timezone** – часовой пояс места расположения аэропорта.

Таблица dim_tariff

Таблица содержит информацию о тарифах перелета с указанием класса обслуживания и стоимости (стоимость соответствует выбранному классу). Гранулярность – один перелет. Тип SCD – 0.

Содержание полей:

- **id** - суррогатный первичный ключ;

Таблица fact_flights

Таблица содержит информацию о фактически совершённых перелетах. Гранулярность – один пассажир. Таблица связана с таблицами измерений (справочниками) по соответствующим id.

Содержание полей:

- **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.

3 ETL-процессы

3.1 ETL dim_passengers

Рисунок 3 – ETL-трансформация обогащения данными таблицы dim_passengers

Описание трансформации:

На первом этапе происходит считывание информации из исходной демонстрационной базы данных. Чтение происходит из таблицы bookings.tickets (выбираются столбцы passenger_id, passenger_name, contact_data).

Далее реализуется проверка качества передаваемых данных (в зависимости от результата проверки, в лог для пассажиров отправляются строки, у которых как минимум в одном из полей passenger_id, passenger_name, содержится значение NULL). Принимаем, что в хранилище данных должны попасть только полностью заполненные строки.

Следующим шагом происходит разбиение json-массива (поле contact_data) по составляющим, с целью записи данных о телефоне и адресе электронной почты пассажира в отдельные поля;

Затем реализуется проверка, адреса электронной почты и телефона на корректность данных и фильтрация «бракованных» сведений. Принимаем, что в хранилище должны попасть только корректные данные.

Заключительными шагами данной трансформации будет являться выбор столбцов для записи в таблицу dim_passengers, и непосредственная загрузка данных в таблицу.

3.2 ETL dim_aircrafts

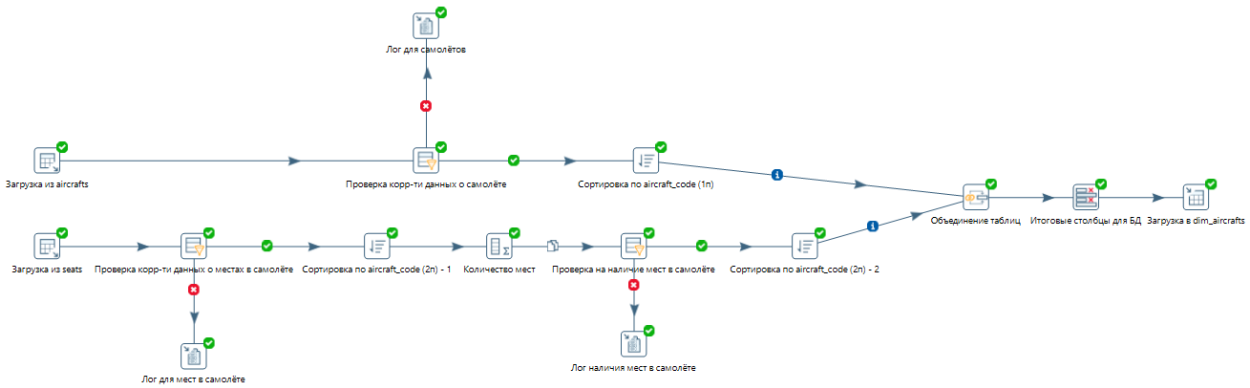

Рисунок 4 – ETL-трансформация обогащения данными таблицы dim_aircrafts

Описание трансформации:

На первом этапе происходит считывание информации из исходной демонстрационной базы данных. Чтение происходит из таблиц bookings.aircrafts и bookings.seats.

Затем, поток данных из bookings.aircrafts проверяется на отсутствие NULL в поле model и удовлетворение условия range > 0. Параллельно происходит проверка потока из bookings.seats на отсутствие NULL в полях airport_code, seat_no, fare_conditions. В том случае, когда условия не выполняются, «бракованные данные» записываются в логи.

Дополнительно стоит отметить, что в потоке из bookings.seats происходит подсчет количества мест в каждом самолёте (это сопровождается процессами сортировки и группировки). В том случае, если у самолёта количество мест равно 0, то подобные принимаются за ошибочные и записываются в лог.

После всех проверок отсортированные потоки объединяются при помощи шага Merge Join (тип INNER) по полю aircraft_code.

Заключительными шагами данной трансформации является окончательная выборка необходимых столбцов и запись данных в таблицу dim_aircrafts.

3.3 ETL dim_airports

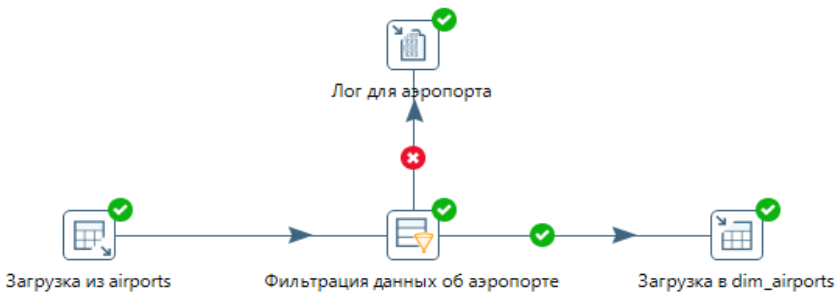

Рисунок 5 – ETL-трансформация обогащения данными таблицы dim_airports

Описание трансформации:

На первом этапе происходит считывание информации из исходной демонстрационной базы данных. Чтение происходит из таблицы bookings.airports.

Затем, поток данных из bookings.airports проверяется на отсутствие NULL в полях airport_code, airport_name, city, а так же на удовлетворение ещё одного условия для airport_code (данное поле должно содержать ровно 3 символа). В том случае, если перечисленные условия не выполнены – данные отправляются в лог.

Заключительным шагом данной трансформации является запись данных в таблицу dim_airports.

3.4 ETL таблицы dim_tariff

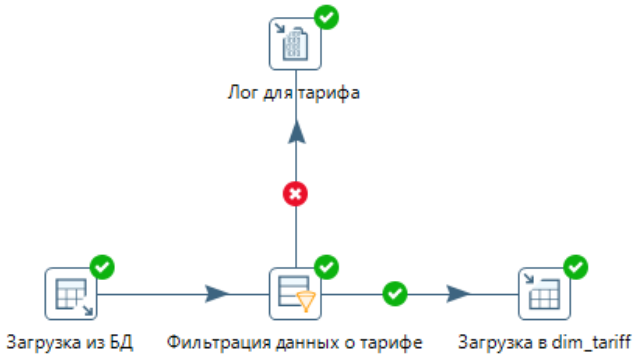

Рисунок 5 – ETL-трансформация обогащения данными таблицы dim_tariff

Описание трансформации:

На первом этапе происходит считывание информации из исходной демонстрационной базы данных. Соответствующий запрос представлен в файле **Script_(tarriff and fact).sql**.

Затем, поток данных проверяется на отсутствие NULL в полях fare_conditions и amount, а также на удовлетворение дополнительного условия для airport_code (0 < amount < 1 000 000). В том случае, если перечисленные условия не выполнены – данные отправляются в лог.

Заключительным шагом данной трансформации является запись данных в таблицу dim_tariff.

3.5 ETL fact_flights

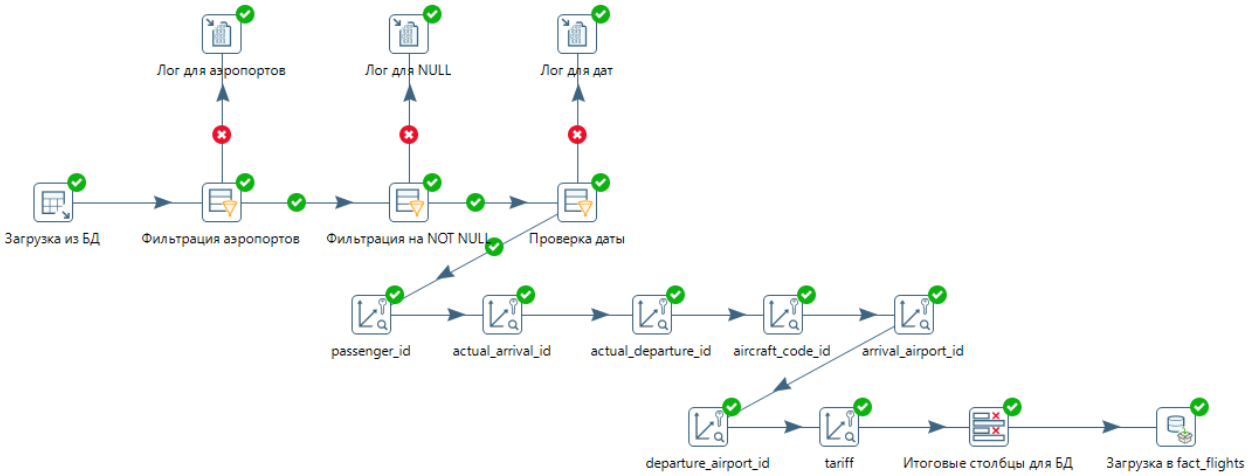

Рисунок 6 – ETL-трансформация обогащения данными таблицы fact_flights

Описание трансформации:

На первом этапе происходит считывание информации из исходной демонстрационной базы данных. Соответствующий запрос представлен в файле **Script_(tarriff and fact).sql.**

Затем проверяются данные об аэропортах. Аэропорт отправления не должен совпадать с аэропортом прибытия, ошибочные записи записываются лог.

Следующим шагом реализуется проверка на отсутствие NULL в полях flight_id и passenger_id, отсутствующие значения записываются в лог.

Следующим шагом производится проверка даты и времени. Дата и время вылета должны быть меньше даты и времени прилёта, ошибочные данные записываются лог.

После всех необходимых проверок реализуется серия шагов «Combination lookup/update» по извлечению id из таблиц измерений с целью замены значений на извлечённые id в исходной таблице.

В финальной части данной трансформации, т.е. когда все значения исходной таблицы (кроме значений delay_departure и delay_arrival) заменены соответствующими id из таблиц измерений, отбираются только нужные поля (шаг «Select values») и производится запись данных в целевую таблицу фактов fact_flights.

ФАЙЛЫ ПРОЕКТА

1 **Script.sql** – скрипты для создания таблиц фактов и измерений (включая запрос на обогащение таблицы dim_calendar);

2 **Script_(tarriff and fact).sql** – скрипты для трансформацию данных о тарифах и для таблицы фактов из внешнего источника;

3 **ETL_aircrafts.ktr** - описание ETL-процессов на обогащение таблицы dim_aircrafts;

4 **ETL_airports.ktr** - описание ETL-процессов на обогащение таблицы dim_airports;

5 **ETL_passengers.ktr** - описание ETL-процессов на обогащение таблицы dim_ passengers;

6 **ETL_tariff.ktr** - описание ETL-процессов на обогащение таблицы dim_ tariff;

7 **ETL_fact_flights.ktr** - описание ETL-процессов на обогащение таблицы fact_flights;

8 **demo_small.sql –** скрипт для поднятия демонстрационной БД.