**Проектная работа по модулю**

**“SQL и получение данных”**

Содержание

1 [Установка базы на локальном компьютере](#_heading=h.30j0zll)

2 [Диаграмма схемы данных](#_heading=h.1fob9te)

3 [Описание схемы](#_heading=h.3znysh7)

4 [Объекты схемы](#_heading=h.2et92p0)

5 [Задания. Создание запросов](#_heading=h.tyjcwt)

1. Установка базы на локальном компьютере

Базу данных разворачиваем в Docker-контейнере. За основу взят докерфайл: <https://github.com/mgramin/docker-postgres-up-from-dump>

Описание:

# Загружаем официальный образ PostgreSQL

FROM postgres:12

# Создаем необходимые каталоги

RUN mkdir -p /var/lib/postgresql-static/data

ENV PGDATA /var/lib/postgresql-static/data

# Образ установлен на ОС Debian. Обновляем систему и устанавливаем unzip

RUN apt-get update

RUN apt-get install wget unzip -y

# Загружаем и распаковываем дамп базы

ARG DUMP\_URI=https://edu.postgrespro.ru/demo-small.zip

RUN wget --output-document=dbdump.zip $DUMP\_URI

RUN unzip dbdump.zip -d dbdump

RUN echo "" > /docker-entrypoint-initdb.d/run\_dbdump.sh

# С помощью утилиты psql создаем шаблон базы, устанавливаем библиотеки, в том # числе ''cube'',''earthdistance'' для определения расстояний # Загружаем дамп базы

RUN echo "psql -t template1 -U postgres -c 'alter system set shared\_preload\_libraries=''pg\_stat\_statements'',''pg\_buffercache'',''cube'',''earthdistance'';'" >> /docker-entrypoint-initdb.d/run\_dbdump.sh

RUN echo "psql -U postgres -f /dbdump/demo-small-20170815.sql" >> /docker-entrypoint-initdb.d/run\_dbdump.sh

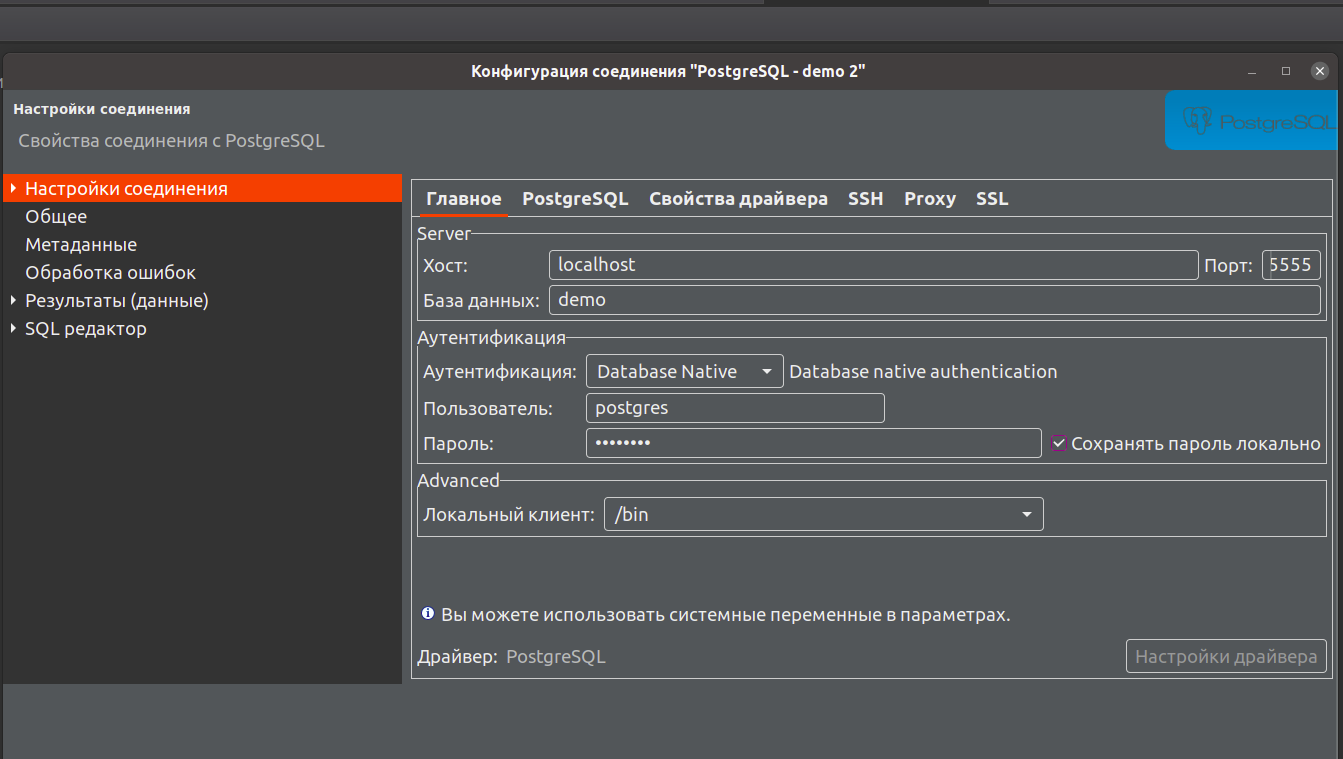
Сборка образа командой

docker build --build-arg DUMP\_URI=https://edu.postgrespro.ru/demo-small.zip -t airbooking .

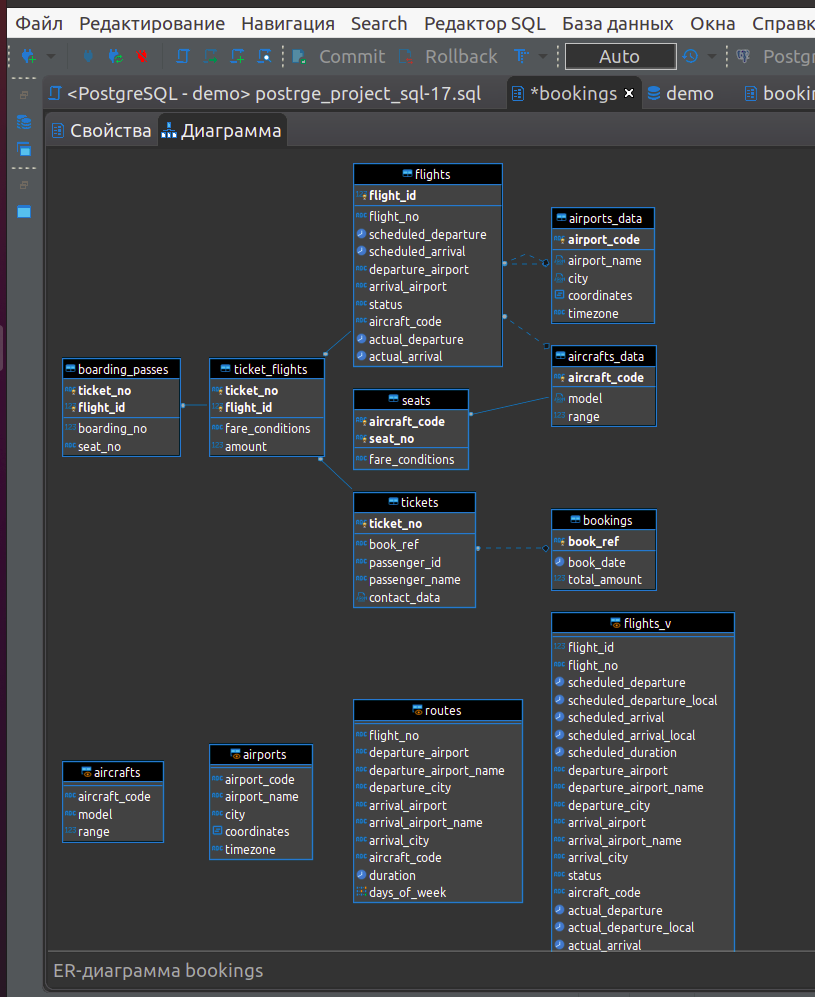
Запуск образа на порту 5555 хоста, так как на нем уже установлена Postgres

docker run -t -p 5555:5432 -e POSTGRES\_PASSWORD=password airbooking

Конфигурация соединения в dbeaver



1. Диаграмма схемы данных



1. Описание схемы

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.

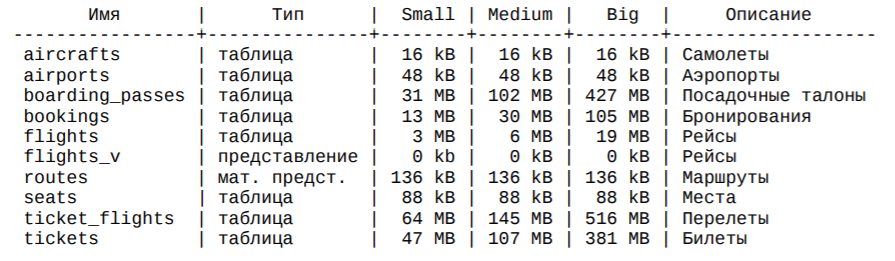
Билет включает один или несколько перелетов (ticket\_flights). Несколько перелетов могут включаться в билет в случаях, когда нет нет прямого рейса, соединяющего пункты отправления и назначения (полет с пересадками), либо когда билет взят «туда и обратно». В схеме данных нет жесткого ограничения, но предполагается, 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.

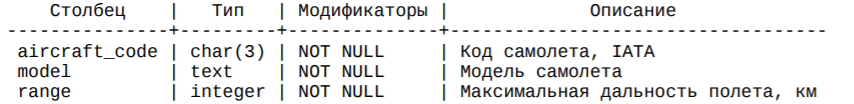
Количество мест (seats) в самолете и их распределение по классам обслуживания зависит от модели самолета (aircrafts), выполняющего рейс. Предполагается, что каждая модель самолета имеет только одну компоновку салона. Схема данных не контролирует, что места в посадочных талонах соответствуют имеющимся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

1. Объекты схемы
   1. Список отношений



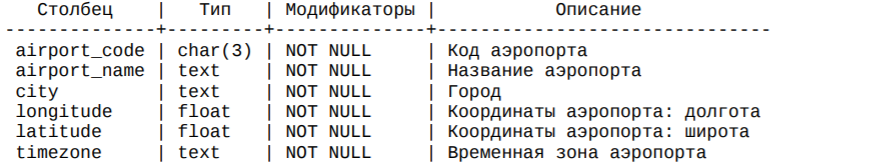
* 1. bookings.aircrafts

Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft\_code). Указывается также название модели (model) и максимальная дальность полета в километрах (range).



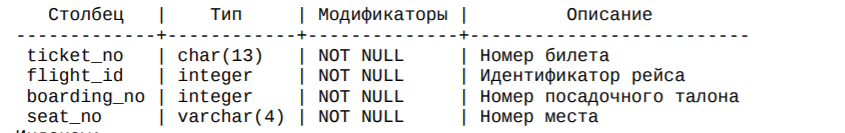
* 1. bookings.airports

Аэропорт идентифицируется трехбуквенным кодом (airport\_code) и имеет свое имя (airport\_name). Для города не предусмотрено отдельной сущности, но название (city) указывается и может служить для того, чтобы определить аэропорты одного города. Также указывается широта (longitude), долгота (latitude) в формате point(latitude, longitude) , и часовой пояс (timezone).



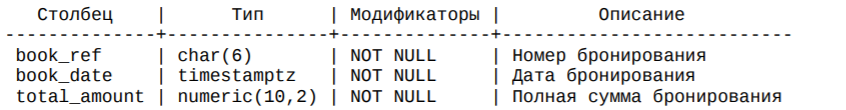
* 1. bookings.boarding\_passes

При регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон. Он идентифицируется также, как и перелет — номером билета и номером рейса. Посадочным талонам присваиваются последовательные номера (boarding\_no) в порядке регистрации пассажиров на рейс (этот номер будет уникальным только в пределах данного рейса). В посадочном талоне указывается номер места (seat\_no).



* 1. bookings.bookings

Пассажир заранее (book\_date, максимум за месяц до рейса) бронирует билет себе и, возможно, нескольким другим пассажирам. Бронирование идентифицируется номером (book\_ref, шестизначная комбинация букв и цифр). Поле total\_amount хранит общую стоимость включенных в бронирование перелетов всех пассажиров.



* 1. bookings.flights

Естественный ключ таблицы рейсов состоит из двух полей — номера рейса (flight\_no) и даты отправления (scheduled\_departure). Чтобы сделать внешние ключи на эту таблицу компактнее, в качестве первичного используется суррогатный ключ (flight\_id). Рейс всегда соединяет две точки — аэропорты вылета (departure\_airport) и прибытия (arrival\_airport). Такое понятие, как «рейс с пересадками» отсутствует: если из одного аэропорта до другого нет прямого рейса, в билет просто включаются несколько необходимых рейсов. У каждого рейса есть запланированные дата и время вылета (scheduled\_departure) и прибытия (scheduled\_arrival). Реальные время вылета (actual\_departure) и прибытия (actual\_arrival) могут отличаться: обычно не сильно, но иногда и на несколько часов, если рейс задержан. Статус рейса (status) может принимать одно из следующих значений:

• Scheduled Рейс доступен для бронирования. Это происходит за месяц до плановой даты вылета; до этого запись о рейсе не существует в базе данных.

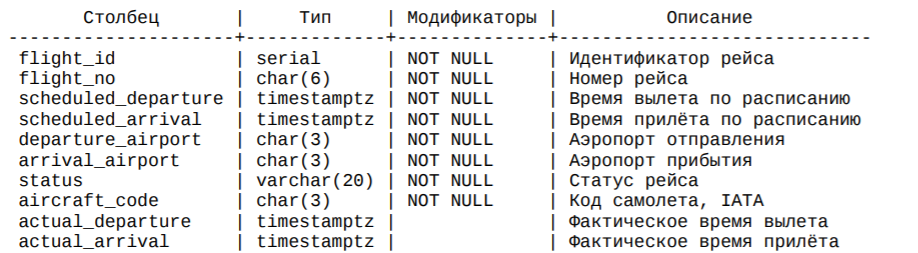
• On Time Рейс доступен для регистрации (за сутки до плановой даты вылета) и не задержан.

• Delayed Рейс доступен для регистрации (за сутки до плановой даты вылета), но задержан.

• Departed Самолет уже вылетел и находится в воздухе.

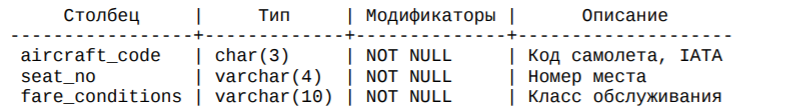
• Arrived Самолет прибыл в пункт назначения.

• Cancelled Рейс отменен.



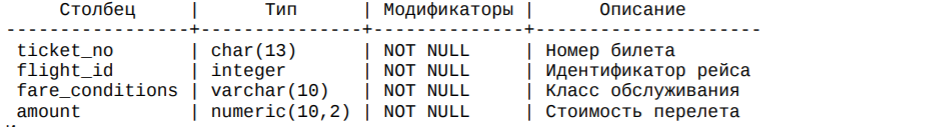
* 1. Таблица bookings.seats

Места определяют схему салона каждой модели. Каждое место определяется своим номером (seat\_no) и имеет закрепленный за ним класс обслуживания (fare\_conditions) — Economy, Comfort или Business



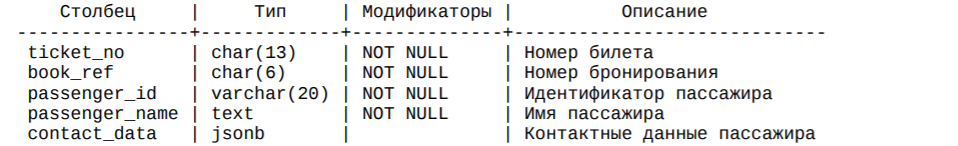
* 1. Таблица bookings.ticket\_flights

Перелет соединяет билет с рейсом и идентифицируется их номерами. Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare\_conditions).



* 1. Таблица bookings.tickets

Билет имеет уникальный номер (ticket\_no), состоящий из 13 цифр. Билет содержит идентификатор пассажира (passenger\_id) — номер документа, удостоверяющего личность, — его фамилию и имя (passenger\_name) и контактную информацию (contact\_date). Ни идентификатор пассажира, ни имя не являются постоянными (можно поменять паспорт, можно сменить фамилию), поэтому однозначно найти все билеты одного и того же пассажира невозможно.



* 1. Представление "bookings.flights\_v"

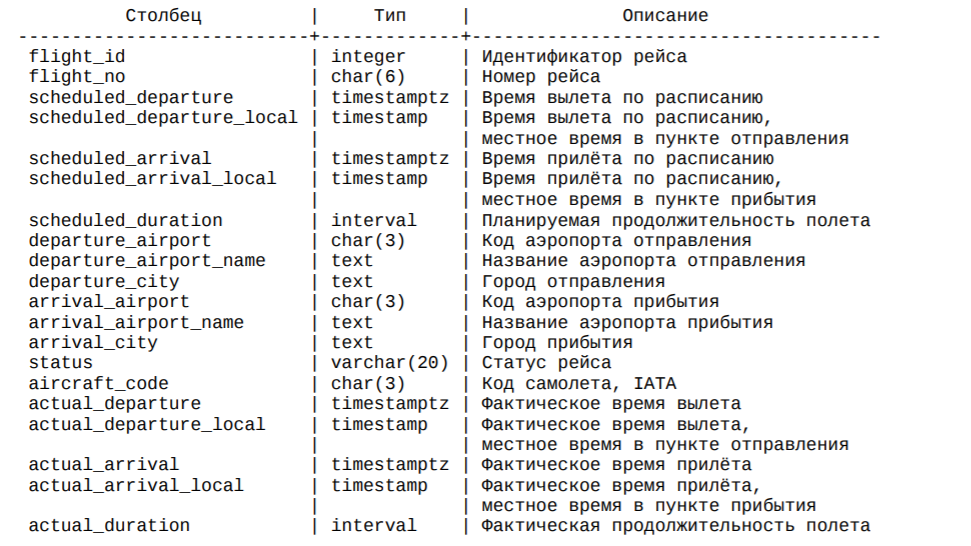
Над таблицей flights создано представление flights\_v, содержащее дополнительную информацию: • расшифровку данных об аэропорте вылета (departure\_airport, departure\_airport\_name, departure\_city),

• расшифровку данных об аэропорте прибытия (arrival\_airport, arrival\_airport\_name, arrival\_city),

• местное время вылета (scheduled\_departure\_local, actual\_departure\_local),

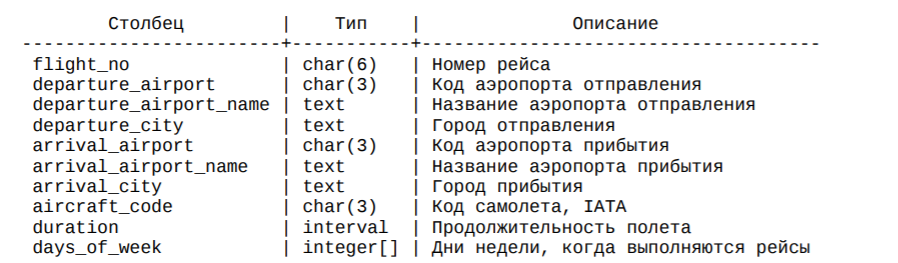
• местное время прибытия (scheduled\_arrival\_local, actual\_arrival\_local),

• продолжительность полета (scheduled\_duration, actual\_duration).



* 1. Материализованное представление bookings.routes

Таблица рейсов содержит избыточность: из нее можно было бы выделить информацию о маршруте (номер рейса, аэропорты отправления и назначения), которая не зависит от конкретных дат рейсов. Именно такая информация и составляет материализованное представление routes.



1. Задания
   1. В каких городах больше одного аэропорта?

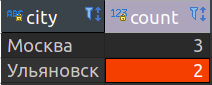
Группируем города из таблицы airports\_data по количеству аэропортов. При этом наименования аэропорта получаем по ключу ‘ru’ из json формата

select a.city ->> 'ru' as city, count(a.airport\_name)

from airports\_data a

group by a.city

having count(a.airport\_name) > 1;



* 1. В каких аэропортах есть рейсы, которые обслуживаются самолетами с максимальной дальностью перелетов?

Создаем общее табличное выражение range\_max, которое выдает максимальную дальность перелета. Объединяем таблицы aircrafts\_data, flights, airports\_data. Выбираем строки где дальность перелета равна макимальной

with range\_max as(

select a\_d."range"

from aircrafts\_data a\_d

order by "range" desc limit 1)

select distinct

a\_d.model ->> 'ru' as model\_name,

a\_d."range",

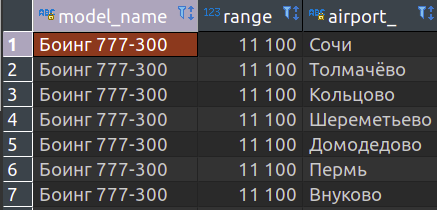
a.airport\_name ->> 'ru' as airport\_

from aircrafts\_data a\_d

join flights f on f.aircraft\_code = a\_d.aircraft\_code

join airports\_data a on a.airport\_code = f.arrival\_airport or airport\_code = f.departure\_airport

where "range" = (select \* from range\_max);



* 1. Были ли брони, по которым не совершались перелеты?

Выбираем бронирования по которым посадочный талон не выдавался

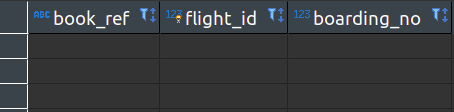
select t.book\_ref, t\_f.flight\_id, b.boarding\_no

from tickets t

join ticket\_flights t\_f using(ticket\_no)

join boarding\_passes b using(ticket\_no)

where t.book\_ref is null;



Таких броней не было

* 1. Самолеты каких моделей совершают наибольший % перелетов?

Создаем общее табличное выражение summ, которое выдает максимальную дальность перелета. Считаем проценты полетов группируя по модели самолета. Упорядочим по убыванию процентов

with summ as (select count (\*) from flights)

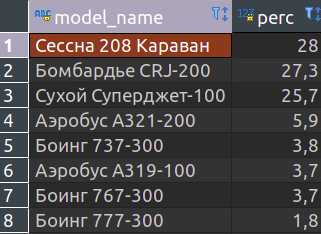
select a\_d.model ->> 'ru' as model\_name, round(100 \* count(\*)/(select \* from summ)::numeric, 1) as perc

from flights f

join aircrafts\_data a\_d using(aircraft\_code)

group by a\_d.model

order by perc desc;



* 1. Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом?

Создаем два представленя с номерам рейса, классом обслуживния, стоимостью и маршрутом для бизнес и эконом класса в отдельности.. Объединяем в одну таблицу по номеру вылета. Выбираем строки, где стоимость перелета бизнес-классом дешевле эконом-классом

create or replace view econimy\_amount as

select f.flight\_id, tf.amount, tf.fare\_conditions, f.departure\_airport || ' - ' || f.arrival\_airport as e\_route

from ticket\_flights tf

join flights f using(flight\_id)

join airports\_data a on a.airport\_code = f.arrival\_airport

where tf.fare\_conditions = 'Economy';

create or replace view business\_amount as

select f.flight\_id, tf.amount, tf.fare\_conditions, f.departure\_airport || ' - ' || f.arrival\_airport as b\_route

from ticket\_flights tf

join flights f using(flight\_id)

join airports\_data a on a.airport\_code = f.arrival\_airport

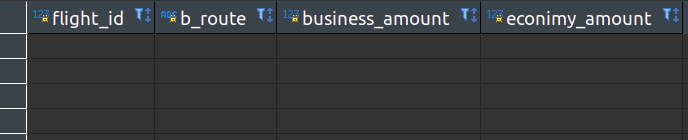
where tf.fare\_conditions = 'Business';

select ba.flight\_id, ba.b\_route, ba.amount business\_amount, ea.amount econimy\_amount

from business\_amount\_ ba

join econimy\_amount\_ ea using(flight\_id)

where ba.amount < ea.amount;



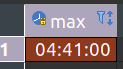
* 1. Узнать максимальное время задержки вылетов самолетов

Время задержки это разница между фактическим временем отправления и расписанием. Убираем те случаи, когда самолет еще не вылетел, тут максимальное время на момент среза данных еще не известно

select max(f.actual\_departure - f.scheduled\_departure)

from flights f

WHERE f.actual\_departure is not null;



* 1. Между какими городами нет прямых рейсов\*?

Создаем два представления. Множество всех возможных вариантов пар городов. И множество пар существующих рейсов. Вычитаем одно множество из другого.

create or replace view cities\_ as

select a.city || ' - ' || b.city as routes

from airports a,airports b

where a.city != b.city;

create or replace view existing as

select distinct a.city as departure\_city,

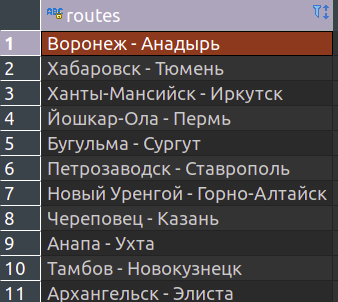
b.city as arrival\_city

from flights f

join airports a on f.departure\_airport = a.airport\_code

join airports b on f.arrival\_airport = b.airport\_code;

(select ci.routes from cities\_ ci) except (select er.departure\_city || ' - ' || er.arrival\_city as routes from existing er);



* 1. Между какими городами пассажиры делали пересадки\*?

Создаем материализованное представление transfers, которое по номеру билета собирает наименования всех аэропортов всех рейсов каждого билета в один массив. Убираем повторяющиеся элементы массива - это аэропорты в которых были пересадки, а нам нужны конечные пункты. Здесь же считаем время задержки в аэропорту и выбираем строки с задержкой от 0 минут до 24 часов.

По обозначению аэропорта выбираем город

create MATERIALIZED VIEW IF NOT EXISTS transfers as

select distinct tf.ticket\_no, max(f.scheduled\_departure) - min(f.scheduled\_arrival) as transfer\_time,

(

(select unnest(regexp\_split\_to\_array(STRING\_AGG(f.departure\_airport, ' '), '\s+'))) except (select unnest(regexp\_split\_to\_array(STRING\_AGG(f.arrival\_airport , ' '), '\s+')))

) as airport\_code

from ticket\_flights tf

join flights f using(flight\_id)

group by tf.ticket\_no

having (EXTRACT(minute from max(f.scheduled\_departure) - min(f.scheduled\_arrival)) > 0) and (

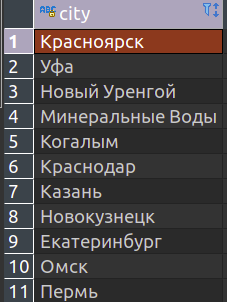
EXTRACT(day from max(f.scheduled\_departure) - min(f.scheduled\_arrival)) < 1)

and (not regexp\_split\_to\_array(STRING\_AGG(f.departure\_airport, ' '), '\s+') @> regexp\_split\_to\_array(STRING\_AGG(f.arrival\_airport , ' '), '\s+'));

select distinct city ->> 'ru' city

from transfers t

join airports\_data a\_d using(airport\_code);



* 1. Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы \*\*

Создаем функцию, вычисляющую расстояние по известной формуле. Предварительно градусы переводим в радианы. применим эту функцию в запросе, который выдает города рейса, модель самолета, максимальную дальность и расстояние по поверхности сферы между этими городами

CREATE OR REPLACE function earthdistance(point, point)

RETURNS double precision

AS '

select round(6371 \* (acos( sin(radians($1[1])) \* sin(radians($2[1])) + cos(radians($1[1])) \* cos(radians($2[1])) \* cos(radians($1[0] - $2[0]))) ));

-- select sin(60);

'

LANGUAGE sql ;

select distinct

ap\_d.airport\_name ->> 'ru' dep\_airport, ap\_a.airport\_name ->> 'ru' arr\_airport,

ad.model ->> 'ru' model, ad."range",

earthdistance(ap\_d.coordinates, ap\_a.coordinates)

from flights f

join aircrafts\_data ad using(aircraft\_code)

join airports\_data ap\_d on f.departure\_airport = ap\_d.airport\_code

join airports\_data ap\_a on ap\_a.airport\_code = f.arrival\_airport ;

