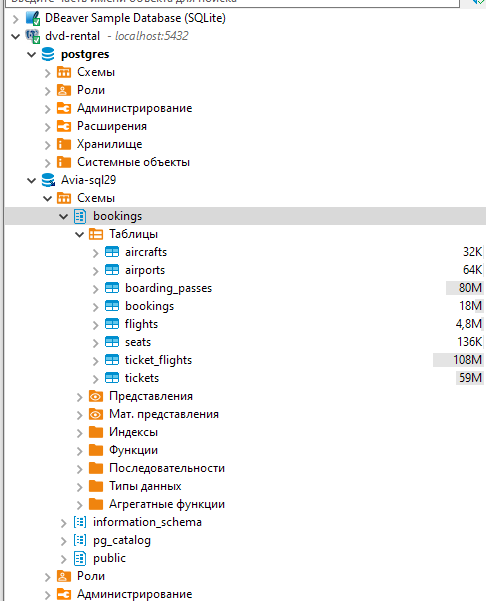
**SQL-29**

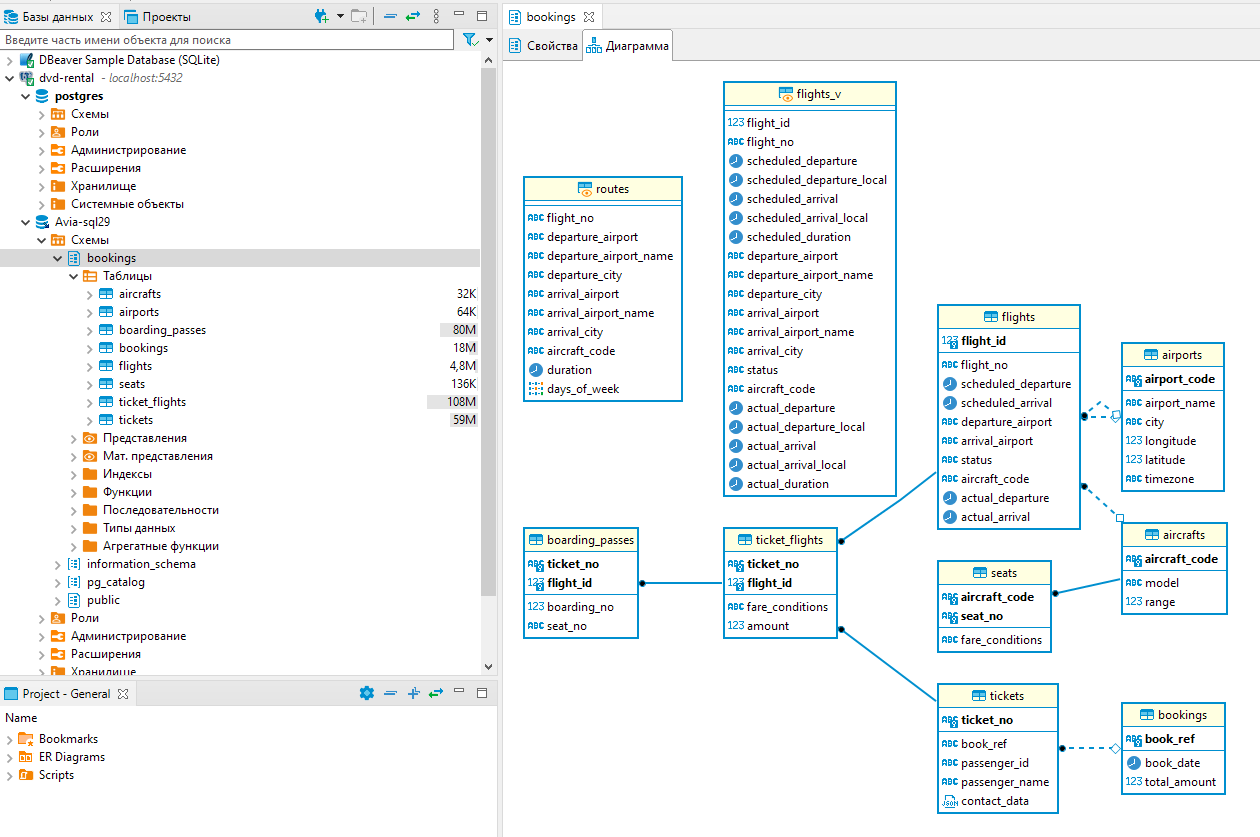
**Итоговая работа**

Приложение №1

1. В работе использовался **локальный** тип подключения.



1. ER-диаграмма



1. База данных представляет собой набор данных о пассажирских авиаперевозках. Рассматриваются такие сущности как тип самолета, аэропорты, посадочные талоны, бронирования, рейсы, места, перелеты, билеты. Итого: 8 таблиц и 2 представления.
2. Развернутый анализ БД:

* **Таблица bookings.aircrafts** Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft\_code). Указывается также название модели (model) и максимальная дальность полета в километрах (range).
* **Таблица 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) и часовой пояс (timezone).
* **Таблица 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).
* **Таблица 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.
* **Таблица 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.

* **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.
* **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).
* **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.
* **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).

* **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

**Бизнес задачи, которые можно было бы решать с помощью данного датасета:** например, какое направление наиболее популярное, какие типы самолетов летают чаще всего, какой пассажиропоток был за период в разрезе месяцев, какие самые непопулярные места в самолете, какие рейсы чаще всего отменяются, какие самые дорогие направления, какие самые экономичные, какая средняя стоимость билета по направлению и т.п

Приложение №2

Ссылка на script SQL <https://github.com/borovkovamv/SQL-29/blob/main/Script-final.sql>

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

Берем таблицу аэропортов. Группируем по городу. Отбираем только те, в которых аэропортов больше 1

**select** city "Город", **count**(airport\_code) **as** "Количество аэропортов"

**from** airports a

**group** **by** city

**having** **count**(airport\_code) > 1;

1. **В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?**

Выбираем подзапросом код самолета с максимальной дальностью полета для этого сортируем по убываю и берем первое значение. Внешним запросом смотрим в каких аэропортах есть рейсы с таким типом самолета, но так как получаем код аэропорта в конце еще соединимся с таблицей аэропортов, чтобы вывести название.

**select** aero.\*, a2.airport\_name

**from** (

**select** f.departure\_airport

**from** flights f

**where** f.aircraft\_code =(

**select** a.aircraft\_code

**from** aircrafts a

**order** **by** **range** **desc**

**limit** 1)

**group** **by** f.departure\_airport) aero

**join** airports a2 **on** a2.airport\_code = aero.departure\_airport

1. **Вывести 10 рейсов с максимальным временем задержки вылета**

Из таблицы вылетов считаем разницу по вылетевшим (доп условие на null) самолетам. Сортируем по убываю. Ограничиваем 10ю рейсами

**select** f.flight\_id, f.scheduled\_departure, f.actual\_departure, f.actual\_departure - f.scheduled\_departure "Задержка вылета"

**from** flights f

**where** f.actual\_departure **is** **not** **null**

**order** **by** 4 **desc**

**limit** 10

1. **Были ли брони, по которым не были получены посадочные талоны?**

Выбираем абсолютно все брони и соединяем их с билетами. Смотрим по каким броням нет билета. Оборачиваем в условие, чтобы вывести ответ в виде Да/Нет

**select**

**case**

**when** **count**(b.book\_ref) > 0 **then** 'Да'

**else** 'Нет'

**end** **as** "Брони без посадочных"

**from** bookings b

**left** **join** tickets t **on** b.book\_ref = t.book\_ref

**left** **join** boarding\_passes bp **on** t.ticket\_no = bp.ticket\_no

**where** bp.boarding\_no **is** **null**

1. **Найдите свободные места для каждого рейса, их % отношение к общему количеству мест в самолете.**

**Добавьте столбец с накопительным итогом - суммарное накопление количества вывезенных пассажиров из каждого аэропорта на каждый день. Т.е. в этом столбце должна отражаться накопительная сумма - сколько человек уже вылетело из данного аэропорта на этом или более ранних рейсах за день.**

Будем использовать 2 cte. Первая для подсчета количества посадочных талонов по рейсам. Вторая для подсчета количества мест в каждом типе самолета. Потом соединяем эти две логики по типу самолета и используем оконную функцию для подсчета накопительной суммы

**with** cte1 **as** (

**select**

f.flight\_id,

f.aircraft\_code,

f.departure\_airport,

f.actual\_departure,

**count**(bp.boarding\_no) **as** "Количество посадочных"

**from** flights f

**join** boarding\_passes bp **on** bp.flight\_id = f.flight\_id

**where** f.actual\_departure **is** **not** **null**

**group** **by** f.flight\_id

),

cte2 **as**(

**select**

s.aircraft\_code,

**count**(s.seat\_no) **as** "Количество мест в самолете"

**from** seats s

**group** **by** s.aircraft\_code )

**select**

c1.actual\_departure,

c1.departure\_airport,

c1.flight\_id,

c2."Количество мест в самолете",

c1."Количество посадочных",

c2."Количество мест в самолете" - c1."Количество посадочных" **as** "Свободные места",

**round**((c2."Количество мест в самолете" - c1."Количество посадочных") / c2."Количество мест в самолете" :: **dec**, 2) \* 100 **as** "% свободных мест на рейсе",

**sum**(c1."Количество посадочных") **over** (**partition** **by** (c1.actual\_departure::**date**, c1.departure\_airport) **order** **by** c1.actual\_departure, c1.departure\_airport) "Количество вылетевших пассажиров"

**from** cte1 c1

**join** cte2 c2 **on** c2.aircraft\_code = c1.aircraft\_code

1. **Найдите процентное соотношение перелетов по типам самолетов от общего количества.**

В подзапросе получим общее число полетов и используем его сразу для расчета % соотношения

**select**

aircraft\_code **as** "Тип самолета",

**round**(**count**(flight\_id) \* 100. / (**select** **count**(flight\_id) **from** flights f2)) **as** "% перелетов"

**from** flights f

**group** **by** aircraft\_code

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

В первой cte отбираем максимальную стоимость эконом класса в разрезе перелета, во второй cte отбираем минимальную стоимость перелета бизнес класса. В основном запросе выбираем города, из которых были осуществлены вылеты этих рейсов из двух cte по условию эконом класс больше бизнес класса

**with** cte1 **as** (

**select** flight\_id, **max** (amount) **as** economy

**from** ticket\_flights tf

**where** fare\_conditions = 'Economy'

**group** **by** flight\_id),

cte2 **as** (

**select** flight\_id, **min** (amount) **as** business

**from** ticket\_flights tf

**where** fare\_conditions = 'Business'

**group** **by** flight\_id)

**select** a.city

**from** airports a

**left** **join** flights f **on** a.airport\_code = f.departure\_airport

**left** **join** cte1 **on** f.flight\_id = cte1.flight\_id

**left** **join** cte2 **on** cte1.flight\_id = cte2.flight\_id

**where** cte1.economy > cte2.business

**group** **by** a.city

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

В первом представлении выбираем декартово произведение всех городов, в которых есть аэропорты, во втором представлении выбираем все варианты сочетаний городов из существующих рейсов. Далее вычитаем из первого представления второе, то есть из всех возможных вариаций, те по которым существуют рейсы, получаем города, между которыми нет прямых рейсов

**create** **view** view1 **as**

**select** **distinct** c1.city **as** city1, c2.city **as** city2

**from** airports c1

**cross** **join** airports c2

**where** c1.city != c2.city

**create** **view** view2 **as**

**select** a1.city **as** city1, a2.city **as** city2

**from** flights f

**join** airports a1 **on** f.departure\_airport = a1.airport\_code

**join** airports a2 **on** f.arrival\_airport = a2.airport\_code

**group** **by** a1.city, a2.city

**select** \*

**from** cities

**except**

**select** \*

**from** connected\_cities

**order** **by** city1, city2

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

Выбираем группировочные данные по сочетанию аэропорт вылета – аэропорт прилета на основании рейсов. Сопоставляем модель самолета, которая совершала данный рейс, делаем расчеты расстояния между аэропортами, сравниваем с данными по расстоянию для модели самолета. Если дальность меньше дальности полета, то ОК.

**select**

a1.airport\_name **as** "Вылет",

a2.airport\_name **AS** "Прилет",

a.model **AS** "Модель самолета",

a."range",

**round**((**acos**(**sind**(a1.latitude) \* **sind**(a2.latitude) + **cosd**(a1.latitude) \* **cosd**(a2.latitude) \* **cosd**(a1.longitude - a2.longitude)) \* 6371)::**dec**, 2) **as** "Расстояние между аэропортами",

**case**

**when** a."range" < **acos**(**sind**(a1.latitude) \* **sind**(a2.latitude) + **cosd**(a1.latitude) \* **cosd**(a2.latitude) \* cosd(a1.longitude - a2.longitude)) \* 6371

**then** 'НЕТ, ALARM!'

**else** 'Да, все ОК'

**end** "Проверка, долетит?"

**from** flights f

**join** airports a1 **on** f.departure\_airport = a1.airport\_code

**join** airports a2 **on** f.arrival\_airport = a2.airport\_code

**join** aircrafts a **on** a.aircraft\_code = f.aircraft\_code

**group** **by** a1.airport\_code, a2.airport\_code, a.aircraft\_code

**order** **by** a1.airport\_name, a2.airport\_name, a.model