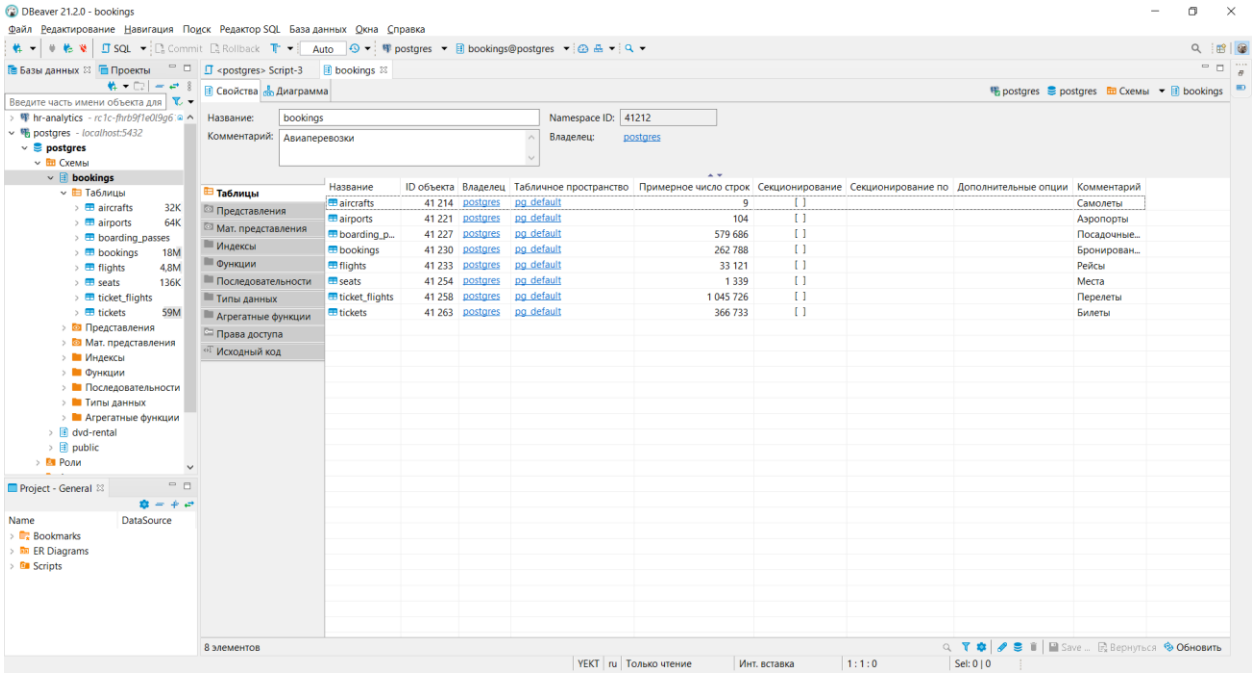
Проектная работа по модулю «SQL и получение данных»

Выполнила:
Мокшина О.В.,
гр. SQL-32

Теоретическая часть

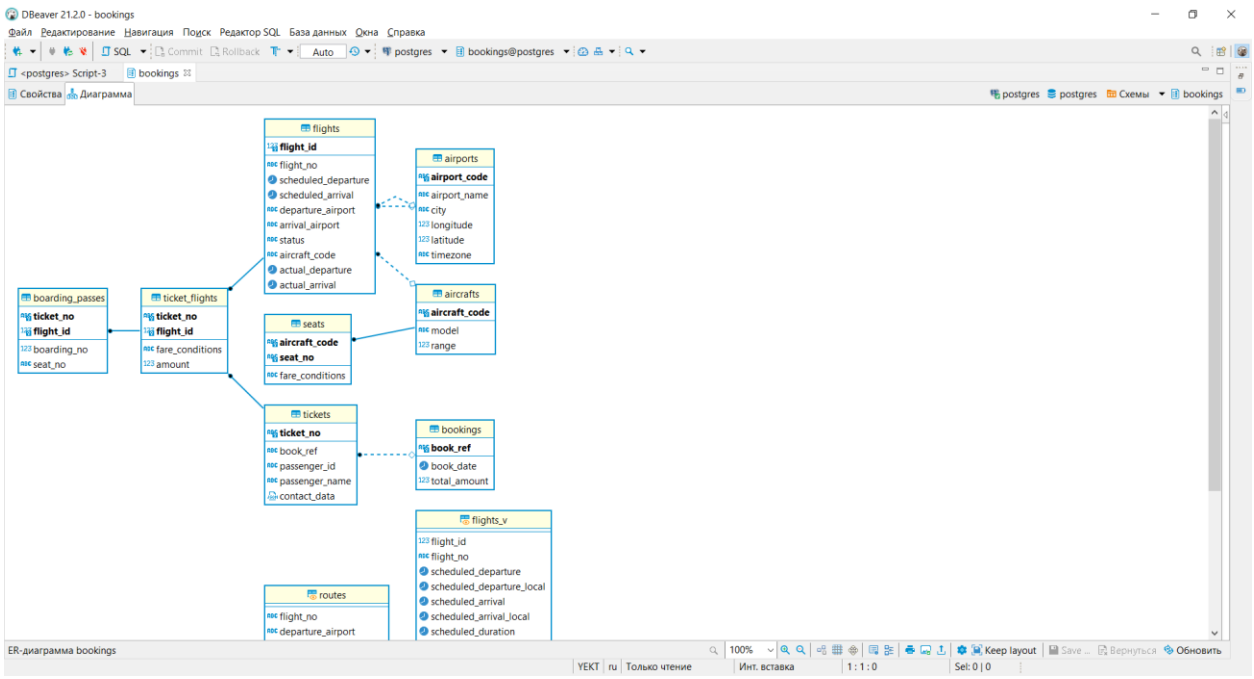
1. Тип подключения.

В работе использовался локальный тип подключения. База данных была восстановлена из *.backup файла.



2. ER-диаграмма.

ER-диаграмма используемой в настоящей проектной работе схемы *bookings* выглядит следующим образом:



3. Краткое описание БД.

В качестве предметной области рассматриваемой базы данных выбраны авиаперевозки по России.

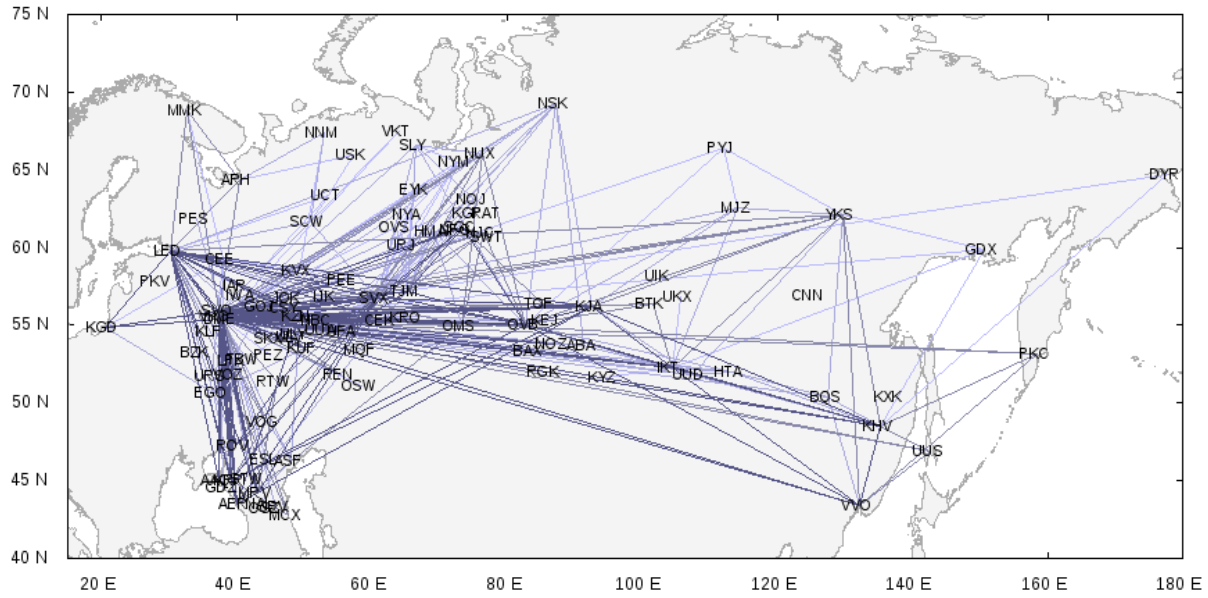


Схема *bookings* состоит из восьми таблиц:

- *aircrafts* (самолеты);
- *airports* (аэропорты);
- *boarding_passes* (посадочные талоны);
- *bookings* (бронирования);
- *flights* (рейсы);
- *seats* (места)
- *tickets* (билеты);
- *ticket_flights* (перелеты);

и двух представлений, одно из которых материализованное:

- *flights_v* (представление создано над таблицей *flights* и содержит дополнительную информацию о рейсах);
- *routes* (материализованное представление, содержит маршруты, не зависящие от конкретных дат рейсов).

Наглядное отражение объектов схемы bookings:

Имя	Тип	Small	Medium	Big	Описание
aircrafts	таблица	16 kB	16 kB	16 kB	Самолеты
airports	таблица	48 kB	48 kB	48 kB	Аэропорты
boarding_passes	таблица	31 MB	102 MB	427 MB	Посадочные талоны
bookings	таблица	13 MB	30 MB	105 MB	Бронирования
flights	таблица	3 MB	6 MB	19 MB	Рейсы
flights_v	представление	0 kb	0 kB	0 kB	Рейсы
routes	мат. предст.	136 kB	136 kB	136 kB	Маршруты
seats	таблица	88 kB	88 kB	88 kB	Места
ticket_flights	таблица	64 MB	145 MB	516 MB	Перелеты
tickets	таблица	47 MB	107 MB	381 MB	Билеты

4. Развернутый анализ БД.

Основной сущностью рассматриваемой базы данных является бронирование (**таблица *bookings***).

```
   Столбец    |      Тип      | Модификаторы |         Описание
--------------+---------------+--------------+--------------------------
 book_ref     | char(6)       | NOT NULL     | Номер бронирования
 book_date    | timestamptz   | NOT NULL     | Дата бронирования
 total_amount | numeric(10,2) | NOT NULL     | Полная сумма бронирования
Индексы:
    PRIMARY KEY, btree (book_ref)
Ссылки извне:
    TABLE "tickets" FOREIGN KEY (book_ref) REFERENCES bookings(book_ref)
```

Каждое бронирование имеет свой уникальный номер (*book_ref*), который является первичным ключом в данной таблице и внешним в таблице *tickets*. В одно бронирование можно включить несколько пассажиров, каждому из которых выписывается отдельный билет (таблица *tickets*). Столбец *total_amount* отражает полную сумму бронирования, то есть стоимость всех перелетов, включенных в выбранное бронирование. Также предполагается, что дата бронирования (*book_date*) отличается от даты бронируемого рейса максимум на месяц.

В **таблице *tickets*** хранится информация о пассажирах (имя и фамилия (*passenger_name*), номер документа, удостоверяющего личность (*passenger_id*), контактные данные (телефон, email (*contact_date,* тип данных *json*)). Для простоты предполагается, что все пассажиры уникальны, поскольку ни идентификатор пассажира, ни имя не являются постоянными (можно поменять паспорт, можно сменить фамилию). Каждый билет имеет уникальный номер (*ticket_no*), который является первичным ключом в таблице *tickets* и внешним в таблице *ticket_flights.* Кроме того, каждому билету соответствует номер бронирования из таблицы *bookings (*внешний ключ *book_ref),* разные билеты могут иметь один номер бронирования.

```
    Столбец     |     Тип     | Модификаторы |          Описание
----------------+-------------+--------------+----------------------------
 ticket_no      | char(13)    | NOT NULL     | Номер билета
 book_ref       | char(6)     | NOT NULL     | Номер бронирования
 passenger_id   | varchar(20) | NOT NULL     | Идентификатор пассажира
 passenger_name | text        | NOT NULL     | Имя пассажира
 contact_data   | jsonb       |              | Контактные данные пассажира
Индексы:
    PRIMARY KEY, btree (ticket_no)
Ограничения внешнего ключа:
    FOREIGN KEY (book_ref) REFERENCES bookings(book_ref)
Ссылки извне:
    TABLE "ticket_flights" FOREIGN KEY (ticket_no) REFERENCES tickets(ticket_no)
```

Билет включает один или несколько перелетов (**таблица *ticket_flights***). Перелет соединяет билет (*ticket_no* из таблицы *tickets*) с рейсом (*flight_id* из таблицы *flights*) и идентифицируется их номерами (составной первичный ключ для таблицы *ticket_flights и составной внешний ключ для таблицы boarding_pass*). Несколько перелетов могут включаться в билет в нескольких случаях:

- Нет прямого рейса, соединяющего пункты отправления и назначения (полет с пересадками);
- Взят билет «туда и обратно».

В схеме данных нет жесткого ограничения, но предполагается, что все билеты в одном бронировании имеют одинаковый набор перелетов.

Таблица *ticket_flights* хранит для каждого перелета информацию о классе обслуживания (значения могут быть только из списка: бизнес, комфорт или эконом), а также о стоимости (предусмотрена проверка на неотрицательность значения).

```
     Столбец     |      Тип      | Модификаторы |      Описание
-----------------+---------------+--------------+---------------------
 ticket_no       | char(13)      | NOT NULL     | Номер билета
 flight_id       | integer       | NOT NULL     | Идентификатор рейса
 fare_conditions | varchar(10)   | NOT NULL     | Класс обслуживания
 amount          | numeric(10,2) | NOT NULL     | Стоимость перелета
Индексы:
    PRIMARY KEY, btree (ticket_no, flight_id)
Ограничения-проверки:
    CHECK (amount >= 0)
    CHECK (fare_conditions IN ('Economy', 'Comfort', 'Business'))
Ограничения внешнего ключа:
    FOREIGN KEY (flight_id) REFERENCES flights(flight_id)
    FOREIGN KEY (ticket_no) REFERENCES tickets(ticket_no)
Ссылки извне:
    TABLE "boarding_passes" FOREIGN KEY (ticket_no, flight_id)
        REFERENCES ticket_flights(ticket_no, flight_id)
```

Каждый рейс (**таблица *flights***) следует из одного аэропорта (таблица *airports*) в другой. Аэропорты отправления (*departure_airport*) и назначения (*arrival_airport*) являются внешними ключами из таблицы *airports*. Уникальность рейса обеспечивается его номером (*flight_no*) и датой вылета (*scheduled_departure*), т.е. составным естественным ключом, тем не менее в таблице *flights* для простоты введен суррогатный первичный ключ (*flight_id*), имеющий тип *serial* (числовой автоинкремент). Рейсы с одним номером имеют одинаковые пункты вылета (*departure_airport*) и назначения *(arrival_airport*), но отличаются датой отправления (*scheduled_departure*, тип данных *timestamptz*, т.е. дата и время (с часовым поясом)). Каждый рейс имеет как плановое время вылета/прилета (*scheduled_departure, scheduled_arrival*), так и фактическое (*actual_departure, actual_arrival*), которое может существенно отличаться. При этом предусмотрены проверки:

- время прилета позже времени вылета (для плановых и фактических данных),
- фактическое время вылета (*actual_departure*) и фактическое время прилета (*actual_arrival*) имеют значение null одновременно
- фактическое время вылета (*actual_departure*) и фактическое время прилета (*actual_arrival*) имеют значение not null одновременно

Для каждого рейса обязательно указывается код обслуживающего самолета (*aircraft_code*, внешний ключ из таблицы *aircrafts*) и статус *status:*

- *Scheduled* - доступен для бронирования, происходит за месяц до плановой даты вылета, до этого запись о рейсе не существует в базе данных.
- *On Time* - доступен для регистрации (за сутки до плановой даты вылета) и не задержан.
- *Delayed* - доступен для регистрации (за сутки до плановой даты вылета), но задержан.
- *Departed* - самолет находится в воздухе.

```
       Столбец       |     Тип     | Модификаторы |          Описание
---------------------+-------------+--------------+----------------------------
 flight_id           | serial      | NOT NULL     | Идентификатор рейса
 flight_no           | char(6)     | NOT NULL     | Номер рейса
 scheduled_departure | timestamptz | NOT NULL     | Время вылета по расписанию
 scheduled_arrival   | timestamptz | NOT NULL     | Время прилёта по расписанию
 departure_airport   | char(3)     | NOT NULL     | Аэропорт отправления
 arrival_airport     | char(3)     | NOT NULL     | Аэропорт прибытия
 status              | varchar(20) | NOT NULL     | Статус рейса
 aircraft_code       | char(3)     | NOT NULL     | Код самолета, IATA
 actual_departure    | timestamptz |              | Фактическое время вылета
 actual_arrival      | timestamptz |              | Фактическое время прилёта
Индексы:
    PRIMARY KEY, btree (flight_id)
    UNIQUE CONSTRAINT, btree (flight_no, scheduled_departure)
Ограничения-проверки:
    CHECK (scheduled_arrival > scheduled_departure)
    CHECK ((actual_arrival IS NULL)
       OR  ((actual_departure IS NOT NULL AND actual_arrival IS NOT NULL)
            AND (actual_arrival > actual_departure)))
    CHECK (status IN ('On Time', 'Delayed', 'Departed',
                      'Arrived', 'Scheduled', 'Cancelled'))
Ограничения внешнего ключа:
    FOREIGN KEY (aircraft_code)
        REFERENCES aircrafts(aircraft_code)
    FOREIGN KEY (arrival_airport)
        REFERENCES airports(airport_code)
    FOREIGN KEY (departure_airport)
        REFERENCES airports(airport_code)
Ссылки извне:
    TABLE "ticket_flights" FOREIGN KEY (flight_id)
        REFERENCES flights(flight_id)
```

При регистрации на рейс пассажиру выдается посадочный талон (**таблица *boarding_passes***), который идентифицируется также, как и перелет - номером билета (*ticket_no*, тип *char* – строка постоянной длины)) и номером рейса (*flight_id*), что обеспечивает связь с таблицей *ticket_flights* через ограничение внешнего ключа. Предполагается, что регистрация на рейс возможна за сутки до плановой даты отправления. Посадочный талон имеет номер (*boarding_no*), который является уникальным в пределах выбранного рейса и присваивается в порядке регистрации пассажиров на рейс. Кроме того, в посадочном талоне указывается номер места (*seat_no*), состоящий из цифр и букв (тип *varchar* – строка переменной длины). Комбинации идентификатор рейса *flight_id* и номер посадочного талона *boarding_no*, идентификатор рейса *flight_id* и номер места *seat_no* являются естественными ключами в рамках выбранного рейса, т.е. предупреждают выдачу двух посадочных талонов на одно место.

```
   Столбец   |    Тип     | Модификаторы |          Описание
-------------+------------+--------------+---------------------------
 ticket_no   | char(13)   | NOT NULL     | Номер билета
 flight_id   | integer    | NOT NULL     | Идентификатор рейса
 boarding_no | integer    | NOT NULL     | Номер посадочного талона
 seat_no     | varchar(4) | NOT NULL     | Номер места
Индексы:
    PRIMARY KEY, btree (ticket_no, flight_id)
    UNIQUE CONSTRAINT, btree (flight_id, boarding_no)
    UNIQUE CONSTRAINT, btree (flight_id, seat_no)
Ограничения внешнего ключа:
    FOREIGN KEY (ticket_no, flight_id)
        REFERENCES ticket_flights(ticket_no, flight_id)
```

Количество мест в самолете и их распределение по классам обслуживания (*fare_conditions*) зависит от модели самолета выполняющего рейс, хранится в **таблице *seats***. Код самолета (*aircraft_code*) связывает таблицу *seats* с родительской таблицей *aircrafts*, при этом предусмотрено каскадное удаление связанных строк при их удалении из таблицы *aircrafts*. Комбинация из кода модели самолета (*aircraft_code*) и номера места (*seat_no*) является уникальной для каждого самолета определенной модели. Предполагается, что каждая модель имеет только одну компоновку салона. Схема данных не контролирует, что места в посадочных талонах соответствуют имеющимся в самолете. Также предусмотрена проверка значения класса обслуживания (fare_conditions): economy, comfort, business.

```
     Столбец     |     Тип     | Модификаторы |      Описание
-----------------+-------------+--------------+--------------------
 aircraft_code   | char(3)     | NOT NULL     | Код самолета, IATA
 seat_no         | varchar(4)  | NOT NULL     | Номер места
 fare_conditions | varchar(10) | NOT NULL     | Класс обслуживания
Индексы:
    PRIMARY KEY, btree (aircraft_code, seat_no)
Ограничения-проверки:
    CHECK (fare_conditions IN ('Economy', 'Comfort', 'Business'))
Ограничения внешнего ключа:
    FOREIGN KEY (aircraft_code)
        REFERENCES aircrafts(aircraft_code) ON DELETE CASCADE
```

Информация о самолетах, обслуживающих рейсы из таблицы *flights* хранится в **таблице *aicrafts***. Каждая модель самолета (*model*) идентифицируется своим трехзначным кодом (*aircraft_code*, тип *char*), который является внешним ключом для таблицы *flights* и *seats*. Предусмотрено, что при удалении связанных строк из таблицы *aircrafts* автоматически удаляются строки из зависимой таблицы *seats*. Также для каждой модели (*model*) указывается максимальная дальность полета (*range*) в километрах, при этом предусмотрена проверка на положительность данной величины.

```
    Столбец    |   Тип   | Модификаторы |             Описание
---------------+---------+--------------+-----------------------------------
 aircraft_code | char(3) | NOT NULL     | Код самолета, IATA
 model         | text    | NOT NULL     | Модель самолета
 range         | integer | NOT NULL     | Максимальная дальность полета, км
Индексы:
    PRIMARY KEY, btree (aircraft_code)
Ограничения-проверки:
    CHECK (range > 0)
Ссылки извне:
    TABLE "flights" FOREIGN KEY (aircraft_code)
        REFERENCES aircrafts(aircraft_code)
    TABLE "seats" FOREIGN KEY (aircraft_code)
        REFERENCES aircrafts(aircraft_code) ON DELETE CASCADE
```

Информация об аэропортах, используемых в таблице *flights* (*departure_airport*, *arrival_airport*) хранится в **таблице *airports***. Аэропорт идентифицируется трехбуквенным кодом (*airport_code*, тип *char*) и имеет свое имя (*airport_name*). Для города не предусмотрено отдельной сущности, но указывается его название (*city*) и может служить для того, чтобы определить аэропорты одного города. Также указываются координаты аэропорта (широта (*latitude*), долгота (*longitude*)) и часовой пояс (*timezone*).

```
   Столбец     |   Тип    | Модификаторы |           Описание
---------------+----------+--------------+-------------------------------
 airport_code  | char(3)  | NOT NULL     | Код аэропорта
 airport_name  | text     | NOT NULL     | Название аэропорта
 city          | text     | NOT NULL     | Город
 longitude     | float    | NOT NULL     | Координаты аэропорта: долгота
 latitude      | float    | NOT NULL     | Координаты аэропорта: широта
 timezone      | text     | NOT NULL     | Временная зона аэропорта
Индексы:
    PRIMARY KEY, btree (airport_code)
Ссылки извне:
    TABLE "flights" FOREIGN KEY (arrival_airport)
        REFERENCES airports(airport_code)
    TABLE "flights" FOREIGN KEY (departure_airport)
         REFERENCES airports(airport_code)
```

Также в схеме *bookings* реализовано два представления. **Представление *flights_v*** (рейсы) содержит дополнительную информацию о рейсах:

- 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*).

То есть представляет собой обогащенную данными таблицу *flights*.

```
          Столбец          |     Тип     |              Описание
---------------------------+-------------+-------------------------------------
 flight_id                 | integer     | Идентификатор рейса
 flight_no                 | char(6)     | Номер рейса
 scheduled_departure       | timestamptz | Время вылета по расписанию
 scheduled_departure_local | timestamp   | Время вылета по расписанию,
                           |             | местное время в пункте отправления
 scheduled_arrival         | timestamptz | Время прилёта по расписанию
 scheduled_arrival_local   | timestamp   | Время прилёта по расписанию,
                           |             | местное время в пункте прибытия
 scheduled_duration        | interval    | Планируемая продолжительность полета
 departure_airport         | char(3)     | Код аэропорта отправления
 departure_airport_name    | text        | Название аэропорта отправления
 departure_city            | text        | Город отправления
 arrival_airport           | char(3)     | Код аэропорта прибытия
 arrival_airport_name      | text        | Название аэропорта прибытия
 arrival_city              | text        | Город прибытия
 status                    | varchar(20) | Статус рейса
 aircraft_code             | char(3)     | Код самолета, IATA
 actual_departure          | timestamptz | Фактическое время вылета
 actual_departure_local    | timestamp   | Фактическое время вылета,
                           |             | местное время в пункте отправления
 actual_arrival            | timestamptz | Фактическое время прилёта
 actual_arrival_local      | timestamp   | Фактическое время прилёта,
                           |             | местное время в пункте прибытия
 actual_duration           | interval    | Фактическая продолжительность полета
```

Материализованное представление *routes* (маршруты) хранит информацию, собранную из таблиц *flights* и *airports*, о маршрутах (номер рейса, аэропорты отправления и назначения и

соответствующие города, обслуживающий самолет, продолжительность полета), которая не зависит от конкретных дат рейсов (указаны дни недели выполняемых рейсов).

Столбец	Тип	Описание
flight_no	char(6)	Номер рейса
departure_airport	char(3)	Код аэропорта отправления
departure_airport_name	text	Название аэропорта отправления
departure_city	text	Город отправления
arrival_airport	char(3)	Код аэропорта прибытия
arrival_airport_name	text	Название аэропорта прибытия
arrival_city	text	Город прибытия
aircraft_code	char(3)	Код самолета, IATA
duration	interval	Продолжительность полета
days_of_week	integer[]	Дни недели, когда выполняются рейсы

С помощью рассматриваемой в данной проектной работе схемы *bookings* можно **решить множество различных бизнес-задач, например**:

- вывести топ самых дорогих / дешевых / популярных направлений в зависимости от времени года;
- узнать, отличаются ли стоимости билетов в зависимости от даты рейса;
- узнать, как отличаются рейсы с запада на восток от рейсов с востока на запад по продолжительности
- определить среди городов, не имеющих прямого рейса, часто посещаемый;
- определить за какое время до даты вылета чаще всего осуществляется бронирование;
- определить среднее количество человек в одном бронировании.

5. Список SQL- запросов.

В рамках настоящей проектной работы были реализованы SQL-запросы, решающие следующие задачи.

- **В каких городах больше одного аэропорта?**

Описание решения: с помощью подзапроса определить в таблице *airports*, какие города встречаются чаще 1 раза (сгруппировав по названию и применив к результату группировки условие, что количество городов в ней больше 1), поскольку это и будут те города, в которых больше 1 аэропорта (в данной таблице аэропорты уникальны, а города - нет). Затем внешним запросом вывести города, сгруппировав их по названию.

```
select a.city
from airports a
where a.city in (select a.city
                 from airports a
                 group by a.city
                 having count(a.city) > 1)
group by a.city;
```

- **В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?** В решении обязательно должен быть использован подзапрос.

Описание решения: учитывая, что у разных моделей самолетов может совпасть максимальная дальность перелета, используются два подзапроса: первый - для определения максимальной дальности перелета *range*, второй - для определения модели самолета *aircraft_code*, которому(ым) свойственно искомое значение максимальной дальности перелета. Затем с помощью внешнего запроса из таблицы *flights* выводятся аэропорты отправления, из которых выполняются рейсы найденными с помощью подзапросов самолетами.

```
select f.departure_airport as "Аэропорты"
from flights f
where f.aircraft_code in (select a.aircraft_code
                          from aircrafts a
                          where a."range" in (select max(a."range") from aircrafts a)
                          )
group by 1;
```

- **Вывести 10 рейсов с максимальным временем задержки вылета.** В решении обязательно должен быть использован оператор limit.

Описание решения: добавить в вывод из таблицы *flights* вычисляемый время задержки столбец (разница между плановым и фактическим временем вылета (*actual_departure-scheduled_departure*)). Поскольку в таблице *flights* время фактического вылета *actual_departure* может принимать *null*, то необходимо исключить из рассмотрения такие строки (предусмотреть условия в where). Затем отсортировать по убыванию значений из вычисляемого время задержки столбца.

```
select f.flight_no, f.scheduled_departure, f.actual_departure, (f.actual_departure-
f.scheduled_departure) as "Время задержки"
from flights f
where f.actual_departure is not null
order by (f.actual_departure- f.scheduled_departure) desc
limit 10;
```

- **Были ли брони, по которым не были получены посадочные талоны?** В решении обязательно должен быть использован верный тип JOIN.

Описание решения: номера броней *book_ref* и соответствующие им билеты *ticket_no* отражены в таблице *tickets*, при этом одной брони может соответствовать несколько билетов. Исходя из того, что если пассажир купил билет, то это вовсе не значит, что он обязательно зарегистрировался на рейс и имеет посадочный талон (т.е. билетов больше, чем посадочных талонов), необходимо соединить таблицу *tickets* с таблицей *boarding_passes* методом ***left join***. Чтобы определить номера броней *book_ref*, по которым не были получены посадочные талоны, необходимо отфильтровать посадочные талоны со значением *null* (т.е. номеру билета соответствует *null* значение посадочного талона).

```
select t.book_ref
from tickets t
left join boarding_passes bp on bp.ticket_no = t.ticket_no
where bp.boarding_no is null;
```

- **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. В решении обязательно должны быть использованы: оконная функция и подзапросы или cte.

Описание решения: используя *cte1*, можно посчитать число мест, соответствующее каждой модели самолета (*aircraft_code*), применяя группировку.

Поскольку информация о фактически занятых местах хранится в таблице *boarding_passes*, необходимо соединить с этой таблицей таблицу *flights* по уникальному ключу *flight_id*. Используя *cte2* и оконную конструкцию с функцией *count()*, можно посчитать число занятых мест в группировке по рейсам. Чтобы в основном запросе оставить по одной строке каждой группы, необходимо с помощью оконной функции *row_number()* дополнительно пронумеровать строки каждого окна.

В основном запросе необходимо соединить *cte2* и *cte1* по уникальному ключу *aircraft_code* и вывести запрашиваемую информацию.

Для вывода округленных значений % свободных мест используется функция *round()* с предварительным приведением значений к типу данных *numeric*.

Для вывода накопительного итога по количеству улетевших человек в каждом аэропорту на каждый день используется оконная конструкция с функцией *sum()* с группировкой по аэропортам вылета и дате (т.е. отбрасывается время) и сортировкой по дате с учетом времени.

```
with cte1 as (select aircraft_code, count(s.seat_no) all_s
from seats s
group by aircraft_code),
cte2 as (select f.flight_id, f.flight_no, f.actual_departure, f.departure_airport,  f.aircraft_code,
        count(bp.seat_no) over (partition by f.flight_id) fact_s,
        row_number() over (partition by f.flight_id) r_n
        from flights f
        join boarding_passes bp on bp.flight_id = f.flight_id)
select cte2.flight_no, cte2.actual_departure, cte2.departure_airport, (cte1.all_s - cte2.fact_s) as
"Свободно",
round(((cte1.all_s  - cte2.fact_s)::numeric  / cte1.all_s::numeric)*100, 2) as "% свободных мест",
sum(cte2.fact_s) over (partition by cte2.departure_airport,cte2.actual_departure::date  order by
cte2.actual_departure) as "Улетело, чел."
from cte2
join cte1 on cte1.aircraft_code = cte2.aircraft_code
where cte2.r_n = 1
order by cte2.departure_airport;
```

- **Найдите процентное соотношение перелетов по типам самолетов от общего количества.** В решении обязательно должны быть использованы: подзапрос и оператор ROUND

Описание решения: в подзапросе с помощью оконных конструкций с функцией *count()* выводятся общее количество рейсов и количество рейсов по типам самолетов.

Для расчета процентного соотношения перелетов по типам самолетов от общего количества во внешнем запросе используются результаты оконных функций из подзапроса при этом данные группируются.

```
select t.aircraft_code, round(t.c_fa::numeric /t.c::numeric*100 , 2) as "% отношение перелетов"
from (select aircraft_code, count(flight_id) over(partition by aircraft_code) c_fa, count(flight_id)
over () c
from flights f) t
group by t.aircraft_code, t.c_fa, t.c;
```

- **Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета?** В решении обязательно должно быть использовано CTE.

Описание решения: в *cte* с помощью условного оператора *case* определим минимальную цену билета для бизнес-класса и максимальную цену билета для эконом-класса для каждого города прибытия в рамках рейсов. Для этого таблицу *ticket_flights* с данными стоимости билета в разрезе классов обслуживания необходимо обогатить данными из таблицы *flights* с помощью *join* по идентификатору рейсов, чтобы затем добраться до названия города прибытия из таблицы *airports*. После чего в основном запросе выводятся те города, для которых справедливо, что минимальная цена билета бизнес-класса ниже максимальной цены билета эконом-класса.

```
with cte as (select  f.flight_no, a.city arrival_city,
min(case when fare_conditions = 'Business' then amount else null end) as business,
max(case when fare_conditions = 'Economy' then amount else null end) as economy
from ticket_flights tf
join flights f on f.flight_id = tf.flight_id
join airports a on a.airport_code = f.arrival_airport
group by f.flight_no, a.city
)
select cte.flight_no, cte.arrival_city
from cte
where cte.business < cte.economy;
```

- **Между какими городами нет прямых рейсов?** В решении обязательно должны быть использованы: декартово произведение в предложении FROM, самостоятельно созданные представления, оператор EXCEPT.

Описание решения: создается материализованное представление *connecting flights*, в котором будут храниться названия городов, между которыми нет рейсов. Для этого из множества всевозможных пар городов из материализованного представления *routes*, собранных с помощью декартова произведения *cross join* (с условием исключения пар с одинаковыми названиями городов), исключаются действительные пары аэропортов (т.е. связанных прямыми рейсами) из материализованного представления *routes*.

```
create materialized view connecting_flights as
select r.departure_city departure_city, r2.arrival_city arrival_city
from routes r
cross join routes r2
where r.departure_city != r2.arrival_city
group by r.departure_city, r2.arrival_city
except
select r3.departure_city, r3.arrival_city
from routes r3;

select *
from connecting_flights;
```

- **Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы.** В решении обязательно должны быть использованы: оператор RADIANS или использование sind/cosd и CASE.

Описание решения: для расчета расстояния используется формула из сферической теоремы косинусов:

d = arccos {sin(latitude_a)·sin(latitude_b) + cos(latitude_a)·cos(latitude_b)·cos(longitude_a - longitude_b)},

где *latitude_a* и *latitude_b* — широты, *longitude_a, longitude_b* — долготы данных пунктов,

d — расстояние между пунктами измеряется в радианах длиной дуги большого круга земного шара.

Расстояние между пунктами, измеряемое в километрах, определяется по формуле:

L = d·R, где R = 6371 км — средний радиус земного шара.

С помощью *join* из таблицы *flights* к аэропортам, связанным прямыми рейсами, добавляются координаты из таблицы *airports* (соединение производится по идентификатору аэропорта, сначала для обогащения данными аэропорта вылета, затем - для аэропорта назначения).

Также с помощью *join* добавляется максимальная дальность перелетов самолетов, обслуживающих рассматриваемые рейсы, из таблицы *aircrafts* (соединение производится по идентификатору модели самолета).

Используя приведенную выше формулу из сферической теоремы косинусов и применяя функции *cosd()* и *sind()* для координат в градусах, рассчитывается расстояние между аэропортами. Сравнение расстояния между аэропортами предусматривается в операторе *case*.
Во избежание дублирования выводимой информации в виде зеркальных пар аэропортов используется условие *f.departure_airport > f.arrival_airport*.

```
select f.departure_airport, f.arrival_airport,
a2."range" as "Range r",
round(6371*acos(sind(a.latitude)*sind(a1.latitude) +
cosd(a.latitude)*cosd(a1.latitude)*cosd(a.longitude - a1.longitude))::numeric,0) as "Расстояние l",
case when (6371*acos(sind(a.latitude)*sind(a1.latitude) +
cosd(a.latitude)*cosd(a1.latitude)*cosd(a.longitude - a1.longitude))) > a2."range"
then 'Превышает' else 'Не превышает' end as "Сравнение l и r"
from flights f
join airports a on a.airport_code = f.departure_airport
join airports a1 on a1.airport_code = f.arrival_airport
join aircrafts a2 on a2.aircraft_code = f.aircraft_code
where f.departure_airport > f.arrival_airport
group by a.airport_code, a1.airport_code, a2."range", f.departure_airport,f.arrival_airport
order by round(6371*acos(sind(a.latitude)*sind(a1.latitude) +
cosd(a.latitude)*cosd(a1.latitude)*cosd(a.longitude - a1.longitude))::numeric,0) DESC;
```