База данных бронирования авиабилетов:

Краткое описание и развернутый анализ базы данных,
ее таблиц и представлений, ER-диаграмма,
написание SQL запросов с описаниями.

Краткое описание базы данных

Имя	Тип	Описание
aircrafts	таблица	Самолеты: модели и максимальная дальность полета в км
airports	таблица	Аэропорты: коды, названия, города, координаты, временные зоны
boarding_passes	таблица	Посадочные талоны: номера талонов, номера билетов, идентификатор рейса, номер места
bookings	таблица	Бронирования: номера, дата бронирования, полные суммы бронирования
flights	таблица	Рейсы: номера, время вылета и прилета по расписанию, фактическое время вылета и прилета, аэропорты отправления и прибытия, статусы рейсов, коды самолетов
flights_v	представление	Рейсы: колонки таблицы flights дополнены местным временем вылета и прилета по расписанию и фактическим, городами и названиями аэропортов вылета и прилета, расчетной и фактической продолжительностью полета
routes	материализованное представление	Маршруты: номера рейсов, коды и названия аэропортов отправления и прибытия, города отправления и прибытия, коды самолетов, продолжительности полетов, дни недели выполнения рейсов
seats	таблица	Места на борту самолета: коды самолета, номера мест, класс обслуживания
ticket_flights	таблица	Перелеты: номера билетов, идентификаторы рейса, классы обслуживания, стоимость перелета
tickets	таблица	Билеты: номера билетов и бронирования, идентификаторы, имена и контактная информация пассажиров

Развернутый анализ базы данных (БД)

Бронирование (bookings) – основная сущность БД. В одно бронирование можно включить несколько пассажиров, каждому из которых выписывается отдельный билет (tickets). Каждый пассажир уникален, то есть не является отдельной сущностью, поэтому однозначно найти все билеты одного и того же пассажира невозможно. В бронировании указывается общая стоимость, включенный в бронирование перелетов всех пассажиров (total_amount).

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». 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. Для города не предусмотрено отдельной сущности, но название города (city) указывается и может служить для того, чтобы определить аэропорты одного города.

При регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон (boarding_passes), в котором указан номер места в самолете (seat_no). Пассажир может зарегистрироваться только на тот рейс, который есть у него в билете (UNIQUE CONSTRAINT). Комбинация рейса и места в самолете уникальна, чтобы не допустить выдачу двух посадочных талонов на одно место (UNIQUE CONSTRAINT). Посадочным талонам присваиваются последовательные номера (boarding_no) в порядке регистрации пассажиров на рейс, и эти номера будут уникальны только в пределах данной рейса.

Количество мест (seats) в самолете и их распределение по классам обслуживания зависит от модели самолета (aircrafts), выполняющего рейс. Каждая модель самолете имеет только одну компоновку салона.

Каждый рейс (flight) уникально идентифицируется по номеру и дате отправления (UNIQUE CONSTRAINT), и соединяет две точки – аэропорты вылета и прибытия. Понятие «рейс с пересадками» отсутствует. Если из одного аэропорта до другого нет прямого рейса, в билет просто включаются несколько необходимых рейсов. Если рейс задержан, то фактическое время вылета и прилета отличаются от запланированных.

Таблица aircrafts:

Включает информацию по всем самолетам. На эту таблицу ссылаются таблица flights (рейсы) и таблица seats (места) внешними ключами aircraft_code.

Колонки:

- aircraft_code: код самолета, IATA - первичный натуральный ключ;
- model: модель самолета (на русском языке, jsonb);
- range: максимальная дальность полета в км.

Таблица airports:

Включает информацию по всем аэропортам. На эту таблицу ссылается таблица flights (рейсы) внешними ключами arrival_airport и departure_airport.

Колонки:

- airport_code: трехбуквенный код аэропорта – первичный натуральный ключ;
- airport_name: название аэропорта (на русском языке, jsonb);
- city: город аэропорта (на русском языке, jsonb);
- longitude: координаты аэропорта - долгота;
- latitude: координаты аэропорта - широта;
- timezone: временная зона аэропорта.

Таблица boarding_passes:

Включает информацию по посадочным талонам и ссылается на таблицу ticket_flights составным внешним ключом ticket_no и flight_id.

Колонки:

- ticket_no: номер билета – первичный ключ;
- flight_id: идентификатор рейса – первичный ключ;
- boarding_no: номер посадочного талона;
- seat_no: номер места в самолете.

Таблица bookings

Включает информацию по бронированию. На эту таблицу ссылается таблица tickets (билеты) внешним ключом booking_ref.

Колонки:

- book_ref: номер бронирования – первичный ключ;
- book_date: дата бронирования;
- total_amount: полная сумма бронирования.

Таблица flights:

Включает информацию по рейсам и ссылается на таблицу airports внешними ключами arrival_airport и departure_airport, а также на таблицу aircrafts внешним ключом aircraft_code. На эту таблицу ссылается таблица ticket_flights внешним ключом flight_id.

Колонки:

- flight_id: идентификатор рейса – первичный суррогатный ключ;
- flight_no: номер рейса;
- scheduled_departure: время вылета по расписанию;
- scheduled_arrival: время прилета по расписанию;
- departure_airport: аэропорт отправления;

- arrival_airport: аэропорт прибытия;
- status: статус рейса;
- aircraft_code: код самолета, IATA;
- actual_departure: фактическое время вылета;
- actual_arraval: фактическое время прилета.

Таблица seats:

Включает информацию по местам в салоне самолета. Эта таблица ссылается на таблицу aircrafts_data внешним ключом aircraft_code.

Колонки:

- aircraft_code: код самолета, IATA – первичный ключ;
- seat_no: номер места – первичный ключ;
- fare_condition: класс обслуживания.

Таблица ticket_flights:

Включает информацию по перелетам, объединяя данные по билету и рейсу, ссылается на таблицу tickets внешним ключом ticket_no и на таблицу flights внешним ключом flight_id. На эту таблицу ссылается таблица boarding_passes внешними ключами ticket_no и flight_id.

Колонки:

- ticket_no: номер билета – первичный ключ;
- flight_id: идентификатор рейса – первичный ключ;
- fare_condition: класс обслуживания;
- amount: стоимость перелета.

Таблица tickets:

Включает информацию по билетам и ссылается на таблицу bookings внешним ключом book_ref. На эту таблицу ссылается таблица ticket_flights внешним ключом ticket_no.

Колонки:

- ticket_no: номер билета – первичный ключ;
- book_ref: номер бронирования;
- passenger_id: идентификатор пассажира;
- passenger_name: имя пассажира;
- contact_data: контактные данные пассажира, json.

Представление flight_v:

Включает дополнительную информацию над таблицей flights (рейсы), а именно расшифровки данных об аэропортах вылета и прилета, местное время вылета и прилета и продолжительность полетов.

Колонки:

- flight_id: идентификатор рейса;
- flight_no: номер рейса;
- scheduled_departure: время вылета по расписанию;
- scheduled_departure_local: время вылета по расписанию, местное время в пункте вылета;
- scheduled_arrival: время прилета по расписанию;
- scheduled_arrival_local: время прилета по расписанию, местное время в пункте прилета;
- scheduled_duration: планируемая продолжительность полета;
- departure_airport: код аэропорта вылета;
- departure_airport_name: название аэропорта вылета;
- departure_city: город отправления;
- arrival_airport: код аэропорта прибытия;
- arrival_airport_name: название аэропорта прибытия;
- arrival_city: город прибытия;
- status: статус рейса;
- aircraft_code: код самолета, IATA;
- actual_departure: фактическое время вылета;
- actual_departure_local: фактическое время вылета, местное время в пункте отправления;
- actual_arrival: фактическое время прилета;
- actual_arrival_local: фактическое время прилета, местное время в пункте прилета;
- actual_duration: фактическая продолжительность полета;

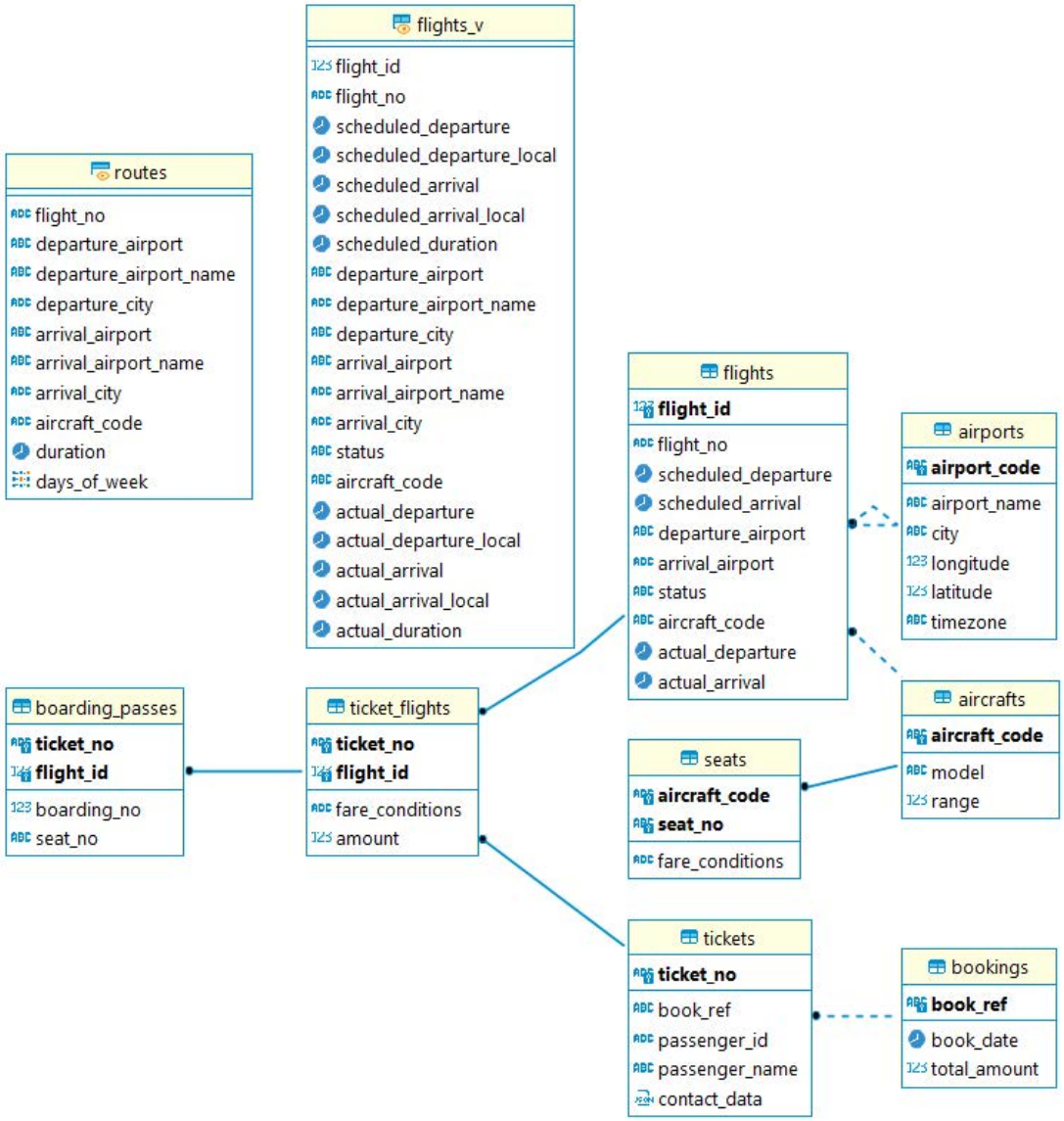
Материализованное представление routes:

Содержит информацию о маршрутах рейсов без привязки к конкретным датам.

Колонки:

- flight_no: номер рейса;
- departure_airport: код аэропорта отправления;
- departure_airport_name: название аэропорта отправления;
- departure_city: город отправления;
- arrival_airport: код аэропорта прибытия;
- arrival_airport_name: название аэропорта прибытия;
- arrival_city: город прибытия;
- aircraft_code: код самолета, IATA;
- duration: продолжительность полета;
- days_of_week: дни недели, когда выполняются рейсы.

ER-диаграмма



SQL запросы и их описания

- Вывести названия самолетов, которые имеют менее 50 посадочных мест;
- Вывести процентное изменение ежемесячной суммы бронирования билетов, округленной до сотых;
- Вывести названия самолетов без бизнес-класса. В запросе обязательно использовать функцию array_agg;
- Вывести накопительный итог количества мест в самолетах по каждому аэропорту на каждый день. Учесть только те самолеты, которые летали пустыми и только те дни, когда из одного аэропорта вылетело более одного такого самолета;
- Вывести процентное соотношение перелетов по маршрутам от общего количества перелетов;
- Вывести количество пассажиров по каждому коду сотового оператора;
- Классифицировать финансовые обороты (сумму стоимости билетов) по маршрутам.

Вывести названия самолетов, которые имеют менее 50 посадочных мест

Код :

```
select model
from(
	select a.model, count(s.seats_no) as seats_number
	from aircrafts a
	join seats s using (aircraft_code)
	group by a.model) t
where seats_number < 50
```

Описание:

Соединяю таблицы aircrafts и seats по ключу aircraft_code, группирую по моделям самолетов и вывожу только названия тех моделей, которые имеют менее 50 посадочных мест.

Вывести процентное изменение ежемесячной суммы бронирования билетов, округленной до сотых

Код:

```
with recursive r as(
	select min(date_trunc('month', book_date)) x from bookings
	union
	select x + interval '1 month' as x
	from r
	where x < (select max(date_trunc('month', book_date)) x from bookings))
select x as "month", sa as sum_amount,
	coalesce(round((coalesce(sa, 0) - lag(coalesce(sa, 0), 1, 0)
over(order by x)) / lag(coalesce(sa, 0), 1, null) over (order by x)*100, 2),0)
as persentage_change
	from r
left join(
	select date_trunc('month', book_date) cm, sum(total_amount) sa
	from bookings
	group by 1) t on t.cm = r.x
order by 1
```

Описание:

Создаю рекурсивный запрос для того, чтобы последовательно перебрать все месяцы. Затем последовательно вывожу месяц, сумму бронирования билетов, сгруппированную по месяцам в блоке left join и процентное изменение суммы бронирования билетов для текущего месяца по сравнению с предыдущим. С помощью оператора round округляю значение до второго знака после запятой. С помощью оператора coalesce вывожу 0 в случае, если значения не существует (для первого месяца не существует предыдущего, поэтому для него процентное изменение по сравнению с предыдущим месяцем будет равно 0). С помощью оператора lag выбираю предыдущее значение, отличное от текущего на 1 позицию.

Вывести названия самолетов без бизнес-класса. В запросе обязательно использовать функцию array_agg

Код:
```
select model
from(
      select a.model, array_agg(distinct s.fare_conditions)::_text as arrays
      from seats s
      join aircrafts a using (aircraft_code)
      group by a.model) t
where not arrays @> array['Business']
```

Описание:
С помощью функции array_agg создаю массив из уникальных значений столбца fare_conditions таблицы seats (Economy, Business, Comfort), присоединяю таблицу aircrafts по ключу aircraft_code. С помощью оператора @> ищу в столбце arrays значение Business и с помощью оператора where not вывожу названия только тех самолетов, в которых нет бизнесс-класса. Так как в столбце fare_conditions находятся значения типа данных varchar, для использования оператора @> я меняю тип данных столбца arrays на _text.

Вывести накопительный итог количества мест в самолетах по каждому аэропорту на каждый день. Учесть только те самолеты, которые летали пустыми и только те дни, когда из одного аэропорта вылетело более одного такого самолета

Код :
```
select t1.departure_airport, t1.actual_departure::date, t2.seats_count,
sum(t2.seats_count) over (partition by t1.departure_airport,
t1.actual_departure::date order by t1.actual_departure)
from(
      select f.aircraft_code, f.departure_airport, f.actual_departure,
      count(f.flight_id) over (partition by f.departure_airport,
      f.actual_departure::date) as flights_count
      from flights f
      left join boarding_passes bp on f.flight_id = bp.flight_id
      where bp.flight_id is null and f.actual_departure::date is not null) t1
join(
      select a.aircraft_code, count(s.seats_no) as seats_count
      from aircrafts a
      join seats s on a.aircraft_code = s.aircraft_code
      group by a.aircraft_code) t2 on t1.aircraft_code = t2.aircraft_code
where t1.flights_count > 1
```

Описание:
1. Сначала в блоке from я присоединяю таблицу boarding_passes к таблице flights с помощью left join, при этом с помощью оператора where я выбираю только те рейсы, которых нет в таблице boarding_passes, но при этом в таблице flights есть фактическая дата вылета, что говорит о том, что рейс состоялся. Таким образом, благодаря присоединению с помощью left join, мы оставляем только те рейсы, на которых не было пассажиров. Ключевым моментом является проверка отсутствия рейса в таблице boarding_passes(посадочные талоны), так как только отсутствие посадочных талонов на рейс гарантирует, что на нем нет пассажиров.
Наконец, с помощью оконной функции выбираю количество рейсов для каждого аэропорта на каждую дату;

2. В блоке join присоединяю значение количества мест для каждой модели самолета;
3. В блоке select оставляю только те значения, где количество рейсов в день для каждого аэропорта больше 1 и с помощью оконной функции вывожу накопительный суммы мест для каждой даты в каждом аэропорту.

Вывести процентное соотношение перелетов по маршрутам от общего количества перелетов

Код:

```
select departure_airport_name, arrival_airport_name,
flights_count/sum(flights_count) over() * 100 as flights_percentage
from(
	select departure_airport_name, arrival_airport_name,
	count(flight_id) as flights_count
	from routes r
	join flights f using (flight_no)
	group by departure_airport_name, arrival_airport_name) t
```

Описание:

К таблице с маршрутами присоединяю таблицу с рейсами, затем считаю количество рейсов с группировкой по аэропортам вылета и аэропортам прибытия. В результат вывожу аэропорт вылета, аэропорт прибытия (маршрут) и, с помощью оконной функции, долю перелетов по этим маршрутам от общего количества перелетов.

Вывести количество пассажиров по каждому коду сотового оператора

Код:

```
select substring(contact_data->>'phone' from 3 for 3), count(passenger_id)
from tickets
group by substring(contact_data->>'phone' from 3 for 3)
```

Описание:

Работа с форматом json - с помощью оператора ->> обращаюсь к значению ключа phone и с помощью оператора substring извлекаю 3 символа - код оператора. В результат вывожу количество пассажиров с группировкой по коду сотового оператора.

Классифицировать финансовые обороты (сумму стоимости билетов) по маршрутам

Код:

```
select
	case
		when sum_routes < 50000000 then 'low'
		when sum_routes >= 150000000 then 'high'
		else 'middle'
	end c,
	count(sum_routes)
from(
	select departure_airport_name, arrival_airport_name, sum(sum_flights) as sum_routes
	from(
		select f.flight_id, sum(tf.amount) as sum_flights
		from flights f
		join ticket_flights tf on f.flight_id = tf.flight_id
		group by f.flight_id) t
	join flights f on f.flight_id = t.flight_id
	join routes r on r.flight_no = f.flight_no
	group by departure_airport_name, arrival_airport_name) t
group by c
```

Описание:

1. Внутренний блок from: соединяю таблицы fligts и ticket_flights и вывожу сумму стоимости билетов для каждого рейса;
2. Внешний блок from: соединяю таблицу, полученную в результате подзапроса в предыдущем блоке, с таблицей flights, в которой нам нужен столбец flight_no для присоединения таблицы routes. Вывожу сумму стоимости рейсов с группировкой по маршрутам;
3. Блок select: с помощью цикла определяю 3 класса по сумме стоимости билетов на маршрутах и вывожу количество маршрутов, распределенных по данным классам согласно условию.