Содержание

Краткое описание

БД состоит из таблиц aircrafts_data, airports_data, boarding_passes, bookings, flights, seats, ticket_flights, tickets и представлений flights_v и routes.

Таблица отношений

Schema	Name	Type	Owner
bookings	aircrafts_data	table	elenasuslova
bookings	airports_data	table	elenasuslova
bookings	boarding_passes	table	elenasuslova
bookings	bookings	table	elenasuslova
bookings	flights	table	elenasuslova
bookings	flights_flight_id_seq	sequence	elenasuslova
bookings	flights_v	view	elenasuslova
bookings	routes	materialized view	elenasuslova
bookings	seats	table	elenasuslova
bookings	ticket_flights	table	elenasuslova
bookings	tickets	table	elenasuslova

Основной сущностью является бронирование (bookings). В одном бронировании может быть выписано несколько билетов (tickets), по одному билету на каждого пассажира. 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».

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

При регистрации на рейс пассажиру выдается посадочный талон (boarding_passes), в котором указано место в самолете. Соответственно, если посадочного талона на рейс нет в базе, значит перелет по этому билету не совершался – пассажир еще не зарегистрировался на рейс, либо его пропустил.

Количество мест (seats) в самолете и их распределение по классам обслуживания зависит от модели самолета (aircrafts_data), выполняющего рейс.

Список SQL запросов с их описанием

1. В каких городах больше одного аэропорта?

Запрос группирует строки из таблицы AIRPORTS_DATA по городу расположения, считает количество аэропортов для каждого города и выводит города, где количество аэропортов больше единицы.

```
SELECT
city ->> 'ru' city, COUNT (airport_code )
FROM airports_data ad
GROUP BY 1
HAVING COUNT (airport_code ) > 1 ;
```

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

Подзапрос определяет самолеты с максимальной дальностью полета, затем основной запрос из таблицы flights отбирает аэропорты отправления, из которых летят самолеты с кодами из подзапроса. В данном случае такой самолет оказался один, но на случай, что их могла быть несколько, используется оператор IN.

```
SELECT
DISTINCT f2.departure_airport
FROM flights f2
WHERE f2.aircraft_code IN (
        SELECT
        ad.aircraft_code
        FROM aircrafts_data ad
        WHERE ad.range = (SELECT MAX(a.range) FROM aircrafts_data a) ) ;
```

3. Были ли брони, по которым не совершались перелеты?

По логике БД посадочный талон выдается, когда пассажир регистрируется на рейс, значит если на бронь нет посадочного талон, значит перелет по ней не совершался. Для отображения таких броней запрос последовательно обогащает таблицу бронирований информацией из таблиц tickets, ticket_flights и boarding_passes и затем отбирает уникальный значения бронирований с пустым значением посадочного талона.

```
SELECT
DISTINCT b.book_ref , bp.boarding_no
FROM bookings b
LEFT JOIN tickets  USING (book_ref)
LEFT JOIN ticket_flights t USING (ticket_no)
LEFT JOIN boarding_passes bp USING (ticket_no)
WHERE bp.boarding_no  IS NULL;
```

4. Самолеты каких моделей совершают наибольший % перелетов?

Запрос обогащает таблицу flights значениями из таблицы aircrafts_data, группирует по значению модели самолета и считает долю перелетов для каждой модели путем деления числа полетов конкретной модели на общее число полетов.

Результаты запроса отсортированы по убыванию частоты полетов, соответственно первые строки показывают модели самолетов, которые совершают наибольший % перелетов.

```
SELECT
ad.model ->> 'ru' model,
(ROUND (
                COUNT (*)::real /
                (SELECT COUNT (*) FROM flights f2 )
                * 100
                ) || '%' ) as percent
FROM flights f JOIN aircrafts_data ad USING (aircraft_code)
GROUP BY 1
ORDER BY COUNT (*) DESC;
```

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

Чтобы узнать, куда можно добраться бизнес - классом дешевле, чем эконом-классом, нужно сравнить максимальный тариф за Эконом и минимальный тариф за Бизнес. Тарифы сравниваются на уровне отдельного перелета flight_id, так как цены для одного рейса но в разные даты могут быть разными.

Запрос содержит два аналогичных подзапроса: один выводит таблицу вида номер перелета и максимальную оплату для класса Эконом, второй – таблицу вида номер перелета и минимум для класса Бизнес. Результат запроса имеет вид таблицы из столбцов «Маршрут» в виде Номер_перелета (город_отправления, город_прибытия), «Максимум за Эконом» и «Минимум за Бизнес». Результат запроса показывает, что маршрутов, где Бизнес был дешевле Эконома не было.

```
SELECT
DISTINCT a.flight_no || ' ' || (f.departure_city , f.arrival_city ) route,
a.max_ecn,
b.min_bsn
FROM
        (
        SELECT
        fv.flight_no ,
        MAX (t.amount) max_ecn
        FROM flights_v fv
        JOIN ticket_flights t USING (flight_id)
```

```
        GROUP BY 1, t.fare_conditions
        HAVING t.fare_conditions = 'Economy'
        ) as a
JOIN
        (
        SELECT
        fv.flight_no ,
        MIN (t.amount) min_bsn
        FROM flights_v fv
        JOIN ticket_flights t USING (flight_id)
        GROUP BY 1, t.fare_conditions
        HAVING t.fare_conditions = 'Business'
        ) as b
        USING (flight_no)
JOIN  flights_v f USING (flight_no)
WHERE a.max_ecn > b.min_bsn
ORDER BY 1;
```

Второй вариант запроса короче, но выполняется дольше.
Здесь вместо подзапросов используется оконная функция. Для определения максимума за Эконом функция делит таблицу на окна по значению flight_id и отбирает максимум из суммы оплаты для значений где класс полета равен Эконому; и аналогично для определения минимума за Бизнес.

```
SELECT
DISTINCT fv.flight_no || ' ' || (fv.departure_city , fv.arrival_city ) route,
MAX (t.amount) FILTER (WHERE t.fare_conditions = 'Economy') OVER (PARTITION BY fv.flight_no) AS max_ecn,
MIN (t.amount) FILTER (WHERE t.fare_conditions = 'Business') OVER (PARTITION BY fv.flight_no) AS min_bsn
FROM flights_v fv
JOIN ticket_flights t USING (flight_id)
ORDER BY 1;
```

6. Узнать максимальное время задержки вылетов самолетов

Для определения задержки вылета самолета достаточно сравнить запланированное время вылета и реальное.

```
SELECT
MAX ( f.actual_departure - f.scheduled_departure )
FROM flights f ;
```

7. Между какими городами нет прямых рейсов?

Для того, чтобы сказать, между какими городами нет прямы перелетов, нужно составить все возможные комбинации между известными городами и сравнить с уникальными маршрутами город_вылета – город_прилета.

Все возможные маршруты выдает подзапрос через CROSS JOIN. Затем эта таблица обогащается информацией из таблицы flights через левое соединение. Соединение происходит на основе совпадения пары значений (город_вылета, город_прилета). Запрос группирует таблицу по паре значений (город_вылета, город_прилета) и отбирает строки, для которых нет перелетов (значение flight_id пусто, а количество соответственно равно 0).

```
SELECT
combin.dep city_departure,
combin.arr city_arrival
FROM (
        -- combinations with repeat
        SELECT DISTINCT
        a.city as dep,
        b.city as arr
        FROM airports a
        CROSS JOIN  airports b
        WHERE a.city <> b.city
        ORDER BY 1
        ) as combin
-- left join will show which couples of cities how many flights have
LEFT JOIN flights_v fv ON (combin.dep = fv.departure_city and combin.arr = fv.arrival_city )
GROUP BY 1, 2
-- filter couples of cities with no direct flights
HAVING COUNT (fv.flight_id ) = 0
ORDER BY 1;
```

8. Между какими городами пассажиры делали пересадки?

Оконная функция LEAD в подзапросе обогащает информацию о билетах и совершаемых по ним рейсах путем добавления информации о времени отправления следующего рейса и городе прилета следующего рейса. Разбивка на окна происходит по номеру билета с сортировкой по времени вылета, что дает логически выстроенный маршрут для каждого билета.

Основной запрос выводит город изначального вылета, город-трансфер и город прилета следующего рейса, если между временем прилета по расписанию и временем вылета по расписанию следующего рейса меньше суток.

Маршруты, содержащие две пересадки, в данном случае отразятся как города_вылета – второй_трансфер и первый_трансфер-город_прилета.

```
SELECT
a.dep_city,
```

```
a.arr_city transfer,
a.next_arr_city
FROM (
	SELECT
	tf.ticket_no ,
	f.flight_no ,
	a1.city ->> 'ru' dep_city,
	a2.city ->> 'ru' arr_city,
	f.scheduled_departure ,
	f.scheduled_arrival ,
	LEAD (f.scheduled_departure) OVER (PARTITION BY tf.ticket_no ORDER BY
f.scheduled_departure) next_scheduled_departure,
	LEAD (a2.city ->> 'ru' ) OVER (PARTITION BY tf.ticket_no ORDER BY f.scheduled_departure)
next_arr_city
	FROM ticket_flights tf
	JOIN flights f using (flight_id)
	JOIN airports_data a1 ON (f.departure_airport = a1.airport_code )
	JOIN airports_data a2 ON (f.arrival_airport = a2.airport_code)
) AS a
GROUP BY 1,2,3
HAVING MIN(a.next_scheduled_departure - a.scheduled_arrival) < '24:00:00';
```

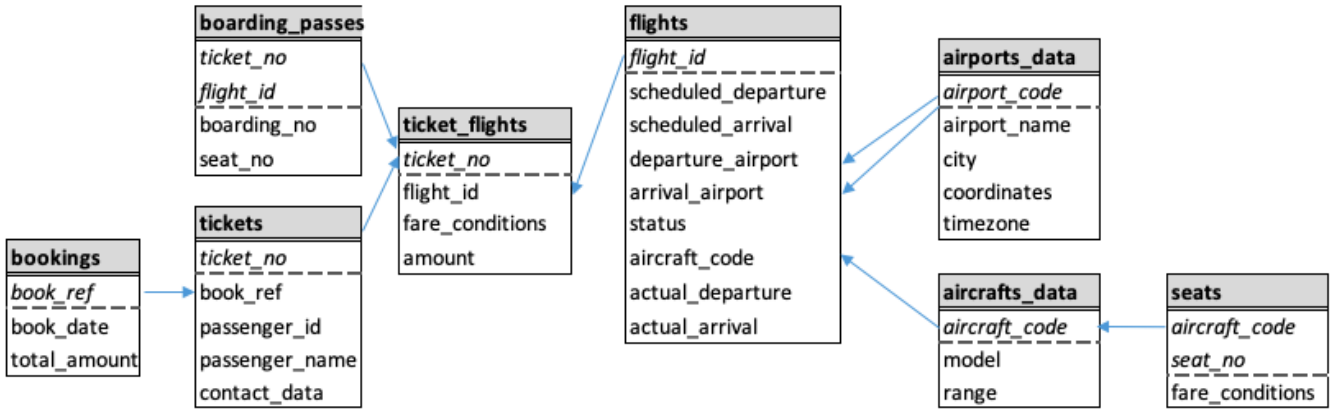
9. Вычислите расстояние между аэропортами, связанными прямыми рейсами, сравните с допустимой максимальной дальностью перелетов в самолетах, обслуживающих эти рейсы.

CTE route собирает таблицу с уникальными значениями рейсов, значениями аэропорта вылета, аэропорта прилета, дистанцией между городами и кодом самолета, совершающего перелет. Дистанция между городами рассчитывается по формуле, приведенной в задании. Долгота и широта из координат приводятся из градусов в радианы путем умножения на π и деления на 180. Затем запрос обогащает CTE информацией о дальности полета самолетов из таблицы aircrafts_data и сравнивает дистанцию между аэропортами с дальностью полета.

```
WITH route AS (
SELECT
	DISTINCT f.flight_no ,
	f.departure_airport,
	f.arrival_airport,
	ROUND (6372.795 * ACOS(	SIN(a.coordinates[1]*PI()/180)  *
				SIN(b.coordinates[1]*PI()/180) +
				COS(a.coordinates[1]*PI()/180) *
				COS(b.coordinates[1]*PI()/180) *
				COS(a.coordinates[0]*PI()/180 - b.coordinates[0]*PI()/180)
					) :: decimal , 3 ) AS distance_km ,
```

```
        f.aircraft_code
        FROM flights f
        JOIN airports a ON (f.departure_airport = a.airport_code)
        JOIN airports b ON (f.arrival_airport = b.airport_code)
        ORDER BY 1)
SELECT
route.flight_no ,
route.departure_airport ,
route.arrival_airport ,
route.distance_km distance_km ,
ad.range aircraft_range_km,
route.distance_km < ad.range IS TRUE  AS "check"
FROM route
JOIN aircrafts_data ad ON (route.aircraft_code = ad.aircraft_code )
ORDER BY 1;
```

ER Диаграмма



Развернутый анализ БД

Таблица bookings.aircrafts_data

```
   Column    |    Type     | Nullable | Default | Storage  |          Description
-------------+-------------+----------+---------+----------+--------------------------------
 aircraft_code | character(3) | not null |         | extended | Код самолета, IATA
 model         | json         | not null |         | extended | Модель самолета
 range         | integer      | not null |         | plain    | Максимальная дальность полета, км
Indexes:
    "aircrafts_pkey" PRIMARY KEY, btree (aircraft_code)
Check constraints:
    "aircrafts_range_check" CHECK (range > 0)
Referenced by:
    TABLE "flights" CONSTRAINT "flights_aircraft_code_fkey" FOREIGN KEY (aircraft_code) REFERENCES aircrafts(aircraft_code)
    TABLE "seats" CONSTRAINT "seats_aircraft_code_fkey" FOREIGN KEY (aircraft_code) REFERENCES aircrafts(aircraft_code) ON
DELETE CASCADE
```

Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft_code). В Таблице содержится информация о название модели (model) и максимальной дальности полета в километрах (range).
Первичный ключ: aircraft_code. В таблице предусмотрена логическая проверка дальности полета больше 0. На таблицу aircrafts_data ссылаются по внешнем ключу таблицы flights и seats.

Таблица bookings.airports_data

```
   Column    |      Type       | Nullable | Default | Storage  |          Description
-------------+-----------------+----------+---------+----------+-------------------------------
 airport_code | character(3)    | not null |         | extended | Код аэропорта
 airport_name | jsonb           | not null |         | extended | Название аэропорта
 city         | jsonb           | not null |         | extended | Город
 coordinates  | point           | not null |         | plain    | Координаты аэропорта: долгота
 timezone     | text            | not null |         | extended | Временная зона аэропорта
Indexes:
    "airports_pkey" PRIMARY KEY, btree (airport_code)
Referenced by:
    TABLE "flights" CONSTRAINT "flights_arrival_airport_fkey" FOREIGN KEY (arrival_airport) REFERENCES airports(airport_code)
    TABLE "flights" CONSTRAINT "flights_departure_airport_fkey" FOREIGN KEY (departure_airport) REFERENCES airports(airport_code)
```

Аэропорт идентифицируется трехбуквенным кодом (airport_code) и имеет свое имя (airport_name), в таблице также указан город (city), в котором он находится. Также в таблице

указаны координаты в виде пары значений (долгота, широта) (longitude, latitude) и часовой пояс (timezone).
Первичный ключ: airport_code. В таблице предусмотрена логическая проверка дальности полета больше 0. На таблицу airports_data ссылается по внешнем ключу таблица flights для проверки значений flights.departure_airport и flights.arrival_airport.

Таблица bookings.boarding_passes

```
  Column    |         Type         | Nullable | Default | Storage  |       Description
------------+----------------------+----------+---------+----------+--------------------------
 ticket_no  | character(13)        | not null |         | extended | Номер билета
 flight_id  | integer              | not null |         | plain    | Идентификатор рейса
 boarding_no | integer             | not null |         | plain    | Номер посадочного талона
 seat_no    | character varying(4) | not null |         | extended | Номер места
Indexes:
    "boarding_passes_pkey" PRIMARY KEY, btree (ticket_no, flight_id)
    "boarding_passes_flight_id_boarding_no_key" UNIQUE CONSTRAINT, btree (flight_id, boarding_no)
    "boarding_passes_flight_id_seat_no_key" UNIQUE CONSTRAINT, btree (flight_id, seat_no)
Foreign-key constraints:

    "boarding_passes_ticket_no_fkey" FOREIGN KEY (ticket_no, flight_id) REFERENCES ticket_flights(ticket_no, flight_id)
```

При регистрации на рейс пассажиру выдается посадочный талон.
Посадочным талонам присваиваются последовательные номера (boarding_no) в порядке регистрации пассажиров на рейс. В посадочном талоне указывается номер места (seat_no), номер билет (ticket_no) и номер перелета (flight_id).
Первичный ключ: сочетание полей ticket_no, flight_id. Таблица имеет ограничение уникальности для номер посадочного талона уникален и номера места в рамках одного рейса (flight_id). Таблица boarding_passes ссылается по внешнем ключу на таблицу ticket_flights.

Таблица bookings.bookings

```
   Column    |           Type           | Nullable | Default | Storage  |       Description
-------------+--------------------------+----------+---------+----------+--------------------------
 book_ref    | character(6)             | not null |         | extended | Номер бронирования
 book_date   | timestamp with time zone | not null |         | plain    | Дата бронирования
 total_amount | numeric(10,2)           | not null |         | main     | Полная сумма бронирования
Indexes:
    "bookings_pkey" PRIMARY KEY, btree (book_ref)
Referenced by:
```

TABLE "tickets" CONSTRAINT "tickets_book_ref_fkey" FOREIGN KEY (book_ref) REFERENCES bookings(book_ref) Таблица bookings.flights

Бронирование идентифицируется номером (book_ref, шестизначная комбинация букв и цифр). В одном бронировании может содержаться несколько билетов для одного или нескольких пассажиров. Поле book_date содержит дату бронирования билетов. Поле total_amount хранит общую стоимость билетов, включенных в бронирование.
Первичный ключ: book_ref. На таблицу bookings ссылается по внешнем ключу таблица tickets.

Таблица bookings.flights

```
Column            |          Type          | Nullable | Storage  |        Description
------------------+------------------------+----------+----------+--------------------
flight_id         | integer                | not null | plain    | Идентификатор рейса
flight_no         | character(6)           | not null | extended | Номер рейса
scheduled_departure | timestamp with time zone | not null | plain    | Время вылета по расписанию
scheduled_arrival   | timestamp with time zone | not null | plain    | Время прилёта по расписанию
departure_airport   | character(3)           | not null | extended | Аэропорт отправления
arrival_airport     | character(3)           | not null | extended | Аэропорт прибытия
status            | character varying(20)  | not null | extended | Статус рейса
aircraft_code       | character(3)           | not null | extended | Код самолета, IATA
actual_departure    | timestamp with time zone |          | plain    | Фактическое время вылета
actual_arrival      | timestamp with time zone |          | plain    | Фактическое время прилёта
Indexes:
    "flights_pkey" PRIMARY KEY, btree (flight_id)
    "flights_flight_no_scheduled_departure_key" UNIQUE CONSTRAINT, btree (flight_no, scheduled_departure)
Check constraints:
    "flights_check" CHECK (scheduled_arrival > scheduled_departure)
    "flights_check1" CHECK (actual_arrival IS NULL OR actual_departure IS NOT NULL AND actual_arrival IS NOT NULL AND actual_arrival
> actual_departure)
    "flights_status_check" CHECK (status::text = ANY (ARRAY['On Time'::character varying::text, 'Delayed'::character varying::text,
'Departed'::character varying::text, 'Arrived'::character varying::text, 'Scheduled'::character varying::text, 'Cancelled'::character varying::text]))
Foreign-key constraints:
    "flights_aircraft_code_fkey" FOREIGN KEY (aircraft_code) REFERENCES aircrafts_data(aircraft_code)
    "flights_arrival_airport_fkey" FOREIGN KEY (arrival_airport) REFERENCES airports_data(airport_code)
    "flights_departure_airport_fkey" FOREIGN KEY (departure_airport) REFERENCES airports_data(airport_code)
Referenced by:
    TABLE "ticket_flights" CONSTRAINT "ticket_flights_flight_id_fkey" FOREIGN KEY (flight_id) REFERENCES flights(flight_id)
```

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). 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, если рейс задержан.
В таблице присутствует проверка, что запланированное время прибытия позже времени отправления.
В Таблице присутствует проверка, что статус рейса (status) может принимать одно из следующих значений:

Scheduled	Рейс доступен для бронирования;
On Time	Рейс доступен для регистрации и не задержан;
Delayed	Рейс доступен для регистрации, но задержан;
Departed	Самолет уже вылетел и находится в воздухе;
Arrived	Самолет прибыл в пункт назначения;
Cancelled	Рейс отменен.

Таблица flights по внешним ключам ссылается на таблицы aircrafts_data и airports_data.
На таблицу flights по внешнему ключу ссылается таблица ticket_flights.

Таблица bookings.seats

```
    Column     |         Type          | Nullable | Default | Storage  |     Description
---------------+-----------------------+----------+---------+----------+--------------------
 aircraft_code | character(3)          | not null |         | extended | Код самолета, IATA
 seat_no       | character varying(4)  | not null |         | extended | Номер места
 fare_conditions | character varying(10) | not null |       | extended | Класс обслуживания
Indexes:
    "seats_pkey" PRIMARY KEY, btree (aircraft_code, seat_no)
Check constraints:
    "seats_fare_conditions_check" CHECK (fare_conditions::text = ANY (ARRAY['Economy'::character varying::text, 'Comfort'::character varying::text, 'Business'::character varying::text]))
Foreign-key constraints:
    "seats_aircraft_code_fkey" FOREIGN KEY (aircraft_code) REFERENCES aircrafts_data(aircraft_code) ON DELETE CASCADE
```

Места определяют схему салона каждой модели самолета. Каждое место определяется своим номером (seat_no) и имеет закреплённый за ним класс обслуживания (fare_conditions) — Economy, Comfort или Business.
Первичный ключ: сочетание полей aircraft_code, seat_no. В таблице предусмотрена проверка, что значение fare_conditions соответствует одному из значений: Economy, Comfort или Business.
Таблица seats по внешнему ключу ссылается на таблицу aircrafts_data.

Таблица bookings.ticket_flights

```
    Column     |         Type          | Nullable | Default | Storage  |     Description
---------------+-----------------------+----------+---------+----------+--------------------
```

```
 ticket_no       | character(13)         | not null |         | extended | Номер билета
 flight_id       | integer               | not null |         | plain    | Идентификатор рейса
 fare_conditions | character varying(10) | not null |         | extended | Класс обслуживания
 amount          | numeric(10,2)         | not null |         | main     | Стоимость перелета
Indexes:
    "ticket_flights_pkey" PRIMARY KEY, btree (ticket_no, flight_id)
Check constraints:
    "ticket_flights_amount_check" CHECK (amount >= 0::numeric)
    "ticket_flights_fare_conditions_check" CHECK (fare_conditions::text = ANY (ARRAY['Economy'::character varying::text,
'Comfort'::character varying::text, 'Business'::character varying::text]))
Foreign-key constraints:
    "ticket_flights_flight_id_fkey" FOREIGN KEY (flight_id) REFERENCES flights(flight_id)
    "ticket_flights_ticket_no_fkey" FOREIGN KEY (ticket_no) REFERENCES tickets(ticket_no)
Referenced by:
    TABLE "boarding_passes" CONSTRAINT "boarding_passes_ticket_no_fkey" FOREIGN KEY (ticket_no, flight_id) REFERENCES
ticket_flights(ticket_no, flight_id)
```

Промежуточная таблица ticket_flights связывает таблицы tickets и flights, перечисляя все перелеты для каждого билета. При этом для каждого перелёта указан класс обслуживания (fare_conditions) и стоимость перелёта (amount).
Первичный ключ: сочетание полей ticket_no, flight_id. В таблице предусмотрена логическая проверка на значение стоимости перелёта больше 0, а также проверка, что значение fare_conditions соответствует одному из значений: Economy, Comfort или Business. Таблица по внешним ключам ссылается на таблицы flights и tickets. На таблицу по внешнему ключу ссылается таблица boarding_passes.

Таблица bookings.tickets

```
     Column     |         Type          | Nullable | Default | Storage  |         Description
----------------+-----------------------+----------+---------+----------+-----------------------------
 ticket_no      | character(13)         | not null |         | extended | Номер билета
 book_ref       | character(6)          | not null |         | extended | Номер бронирования
 passenger_id   | character varying(20) | not null |         | extended | Идентификатор пассажира
 passenger_name | text                  | not null |         | extended | Имя пассажира
 contact_data   | jsonb                 |          |         | extended | Контактные данные пассажира
Indexes:
    "tickets_pkey" PRIMARY KEY, btree (ticket_no)
Foreign-key constraints:
    "tickets_book_ref_fkey" FOREIGN KEY (book_ref) REFERENCES bookings(book_ref)
Referenced by:
    TABLE "ticket_flights" CONSTRAINT "ticket_flights_ticket_no_fkey" FOREIGN KEY (ticket_no) REFERENCES tickets(ticket_no)
```

Билет имеет уникальный номер (ticket_no), состоящий из 13 цифр, и привязан к номеру бронирования (book_ref). Билет содержит идентификатор пассажира (passenger_id) — номер документа, удостоверяющего личность, — его фамилию и имя (passenger_name) и контактную информацию (contact_date).
Первичный ключ: ticket_no. Таблица по внешнему ключу ссылается на таблицу bookings. На таблицу по внешнему ключу ссылается таблица ticket_flights.

Представление bookings.flights_v

Column	Type	Description
flight_id	integer	Идентификатор рейса
flight_no	character(6)	Номер рейса
scheduled_departure	timestamp with time zone	Время вылета по расписанию
scheduled_departure_local	timestamp without time zone	Время вылета по расписанию, местное время в пункте отправления
scheduled_arrival	timestamp with time zone	Время прилёта по расписанию
scheduled_arrival_local	timestamp without time zone	Время прилёта по расписанию, местное время в пункте прибытия
scheduled_duration	interval	Планируемая продолжительность полета
departure_airport	character(3)	Код аэропорта отправления
departure_airport_name	text	Название аэропорта отправления
departure_city	text	Город отправления
arrival_airport	character(3)	Код аэропорта прибытия
arrival_airport_name	text	Название аэропорта прибытия
arrival_city	text	Город прибытия
status	character varying(20)	Статус рейса
aircraft_code	character(3)	Код самолета, IATA
actual_departure	timestamp with time zone	Фактическое время вылета
actual_departure_local	timestamp without time zone	Фактическое время вылета, местное время в пункте отправления
actual_arrival	timestamp with time zone	Фактическое время прилёта
actual_arrival_local	timestamp without time zone	Фактическое время прилёта, местное время в пункте прибытия
actual_duration	interval	Фактическая продолжительность полета

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

```
       Column        |    Type    |           Description
---------------------+------------+---------------------------------
 flight_no           | character(6) | Номер рейса
 departure_airport   | character(3) | Код аэропорта отправления
 departure_airport_name | text       | Название аэропорта отправления
 departure_city      | text       | Город отправления
 arrival_airport     | character(3) | Код аэропорта прибытия
 arrival_airport_name | text       | Название аэропорта прибытия
 arrival_city        | text       | Город прибытия
 aircraft_code       | character(3) | Код самолета, IATA
 duration            | interval   | Продолжительность полета
 days_of_week        | integer[]  | Дни недели, когда выполняются рейсы
```

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.

Самостоятельно развернутая СУБД

(base) elenasuslova@MacBook-Air-Elena ~ % psql

psql (12.1)

Type "help" for help.

elenasuslova=# \list

List of databases

Name	Owner	Encoding	Collate	Ctype	Access privileges
demo	elenasuslova	UTF8	en_US.UTF-8	en_US.UTF-8	
elenasuslova	elenasuslova	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + | | | | | postgres=CTc/postgres
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres + | | | | | postgres=CTc/postgres

(5 rows)