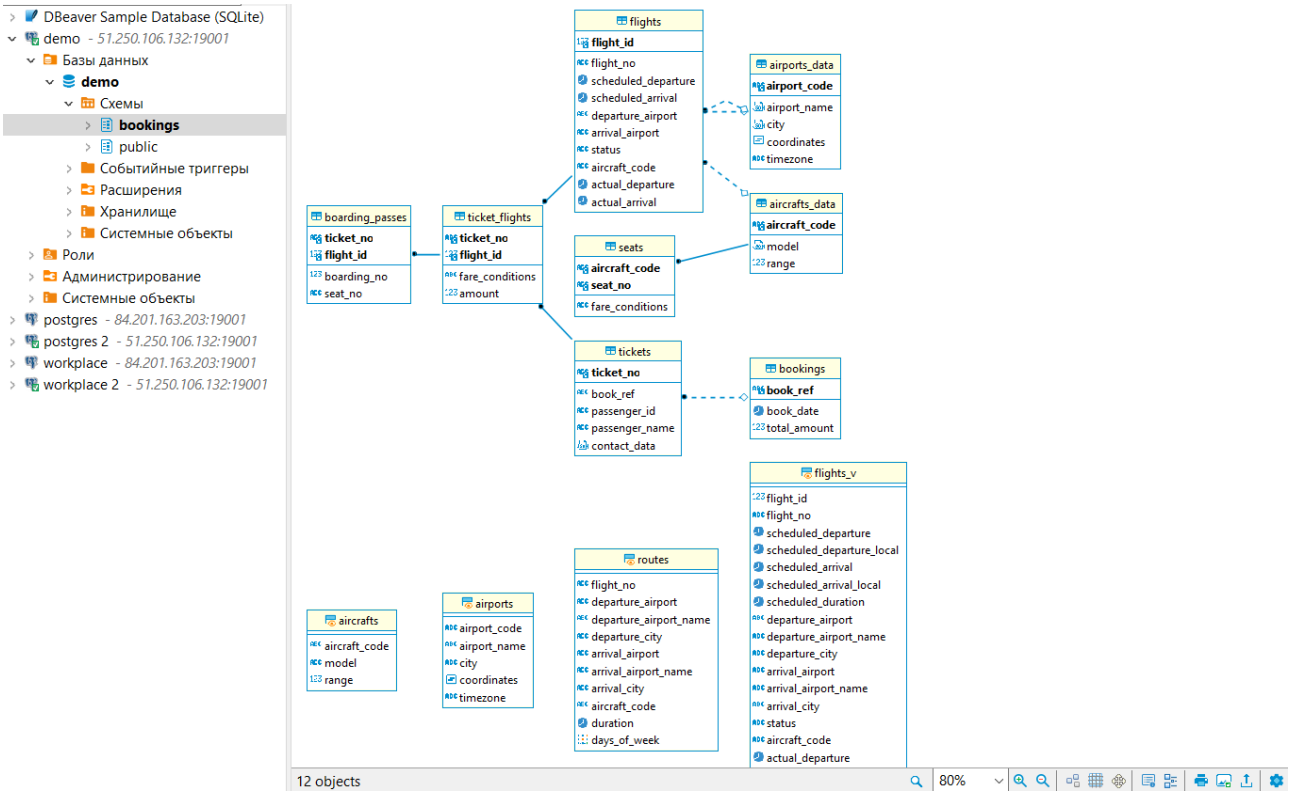
Итоговая работа

1. В работе использовался облачный тип подключения.



2.

3. БД состоит из таблиц:
   - bookings.aircrafts_data
   - bookings.airports_data
   - bookings.boarding_passes
   - bookings.bookings
   - bookings.flights
   - bookings.seats
   - bookings.ticket_flights
   - bookings.tickets

   И представлений:
   - "bookings.flights_v"
   - bookings.routes
   - bookings.aircrafts
   - bookings.airports

4. Развернутый анализ БД
    - **Таблица bookings.aircrafts_data**

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

      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
    - **Таблица bookings.airports_data**

      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).

      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)
    - **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. Посадочный талон идентифицируется — номером билета и номером рейса (первичный ключ). 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).

      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)

- **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.

  Индексы: PRIMARY KEY, btree (book_ref)

  Ссылки извне: TABLE "tickets" FOREIGN KEY (book_ref)
  REFERENCES bookings(book_ref)

- **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.

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)

- **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.
  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
- **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)
  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)
- **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).
  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)

- **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).
- **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.
- **Представление bookings.aircrafts**
  Выводит таблицу bookings.aircrafts_data с учетом локали.
- **Представление bookings.airports**
  Выводит таблицу bookings.airports_data с учетом локали.

Также в схеме есть 11 индексов:

aircrafts_pkey		aircrafts data			[v]		16K	btree
aircraft_code	aircraft code		[v]	[ ]		bpchar ops		
airports_data_pkey		airports data			[v]		16K	btree
airport_code	airport code		[v]	[ ]		bpchar ops		
boarding_passes_flight_id_boarding_no_key		boarding passes			[v]		40M	btree
flight_id	flight id		[v]	[ ]		int4 ops		
boarding_no	boarding no		[v]	[ ]		int4 ops		
boarding_passes_flight_id_seat_no_key		boarding passes			[v]		40M	btree
flight_id	flight id		[v]	[ ]		int4 ops		
seat_no	seat no		[v]	[ ]		text ops		
boarding_passes_pkey		boarding passes			[v]		73M	btree
ticket_no	ticket no		[v]	[ ]		bpchar ops		
flight_id	flight id		[v]	[ ]		int4 ops		
bookings_pkey		bookings			[v]		12M	btree
book_ref	book ref		[v]	[ ]		bpchar ops		
flights_flight_no_scheduled_departure_key		flights			[v]		2M	btree
flight_no	flight no		[v]	[ ]		bpchar ops		
scheduled_departure	scheduled departure		[v]	[ ]		timestamptz ops		
flights_pkey		flights			[v]		1,4M	btree
flight_id	flight id		[v]	[ ]		int4 ops		
seats_pkey		seats			[v]		48K	btree
aircraft_code	aircraft code		[v]	[ ]		bpchar ops		
seat_no	seat no		[v]	[ ]		text ops		
ticket_flights_pkey		ticket flights			[v]		91M	btree
ticket_no	ticket no		[v]	[ ]		bpchar ops		
flight_id	flight id		[v]	[ ]		int4 ops		
tickets_pkey		tickets			[v]		24M	btree
ticket_no	ticket no		[v]	[ ]		bpchar ops		

В схеме также есть две функции:

1. now() - т временный «срез» данных — так, как будто в некоторый момент была сделана резервная копия реальной системы.
2. lang() - возвращает локаль в формате text

Можно решить следующие бизнес задачи, используя эту БД:

1. Вывести топ, посещаемых городов
2. Узнать, в какие города чаще всего летают компанией

3. Подвести статистику, как часто бронируют, но не покупают/покупают
4. Сколько рейсов в среднем задерживается/отменяется
5. Насколько прибыльны/затратны какие то рейсы

5. Список SQL запросов из приложения №2 с описанием логики их выполнения.
    1. В каких городах больше одного аэропорта?
       ЛОГИКА
       * В таблице airoports_data группируем по городам и выбираем те города,
       * в которых число аэропортов больше 1

    2. В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?
       ЛОГИКА
       * К таблице flights присоединяем подзапрос ad по ключу f.arrival_airport = aird.airport_code
       * Из подзапроса ad мы узнаем типы самолетов в максимальной дальностью полета
       * Группируем по аэропортам

    3. Вывести 10 рейсов с максимальным временем задержки вылета
       ЛОГИКА
       * Из таблички flights берем данные, в которых actual_departure не null, т.к. самолет может еще не вылететь
       * и сортируем по времени задержки по убыванию
       * берем первые 10 рейсов

    4. Были ли брони, по которым не были получены посадочные талоны?
       ЛОГИКА:
       * в подзапросе находим количество билетов по броням
       * в основном запросе выбираем только те, в которых не было получено ни одного билета

    5. Найдите количество свободных мест 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 в течении дня.
       ЛОГИКА:
       * В cte necessary_flights выбираем только те рейсы, которые уже прошли или уже идут
       * В cte flights_with_seats соединяем necessary_flights с табличкой seats и считаем сколько всего мест в самолете
       * В cte not_free_sets к таблице boarding_passes присоединяем таблицу нужных рейсов necessary_flights и ищем
       * сколько мест занято
       * В cte first_task таблице flights_with_seats присоединяем таблицу not_free_sets и считаем показатели

* В cte second_task в подзапросе t в окне считаем количесво человек, вывезенных из аэропорта на каждый день
* в основном запросе шруппируем одинаковые значения
* В основном запросе присоединяем first_task и first_task и выводим основныке показатели

6. Найдите процентное соотношение перелетов по типам самолетов от общего количества.
   ЛОГИКА
   * в подзапросе t группирую по типу самолета и считаю количество перелетов с каждым типом
   * В select считаю процент перелетов по типам самолетов

7. Были ли города, в которые можно добраться бизнес - классом дешевле, чем эконом-классом в рамках перелета?
   ЛОГИКА
   * В CTE small_ticket_flights я группирую по flight_id, fare_conditions, amount чтобы убрать лишние значения
   * Далее в CTE ecomomy_tickets и business_tickets я разделяю таблицу small_ticket_flights на 2 билеты эконом
   * класса и билеты бизнес класса
   * Далее в CTE flights_where_economy_more_business я их соединяю и проверяю, есть ли среди них такие, что цена эконом класса больше цены бизнес класса
   * Далее к таблице flights присоединяю таблицы flights_where_economy_more_business и airports_data и вывожу города, являющиеся решением задачи

8. Между какими городами нет прямых рейсов?
   ЛОГИКА
   * В cte small_flights группируем по ad.city ->> 'ru' и ad2.city ->> 'ru' для того, чтобы убрать повторения
   * В подзапросе t мы производим декартово произведение тыблицы small_flights на саму себя, чтобы найти маршруты,
   * в которых город прилета равен городу отлета
   * Из этих маршрутов мы вычитаем те, между которыми есть прямой маршрут

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