

# Демонстрационная база данных Авиаперевозки



## **Авторские права**

© Postgres Professional, 2019–2022

Авторы: Егор Рогов, Павел Лузанов, Павел Толмачев, Илья Баштанов

## **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

## **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:  
[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

## **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Цели и задачи

Предметная область и общая схема демобазы

Подробное описание объектов



Демонстрационная база данных создавалась

- для самостоятельного изучения языка запросов SQL,
- для подготовки книг, пособий и учебных курсов по языку SQL,
- для демонстрации возможностей PostgreSQL в статьях и заметках.

При разработке демонстрационной базы данных мы преследовали несколько целей:

- схема данных должна быть достаточно простой, чтобы быть понятной без особых пояснений;
- в то же время схема данных должна быть достаточно сложной, чтобы позволять строить осмысленные запросы;
- база данных должна быть наполнена данными, напоминающими реальные, с которыми будет интересно работать.

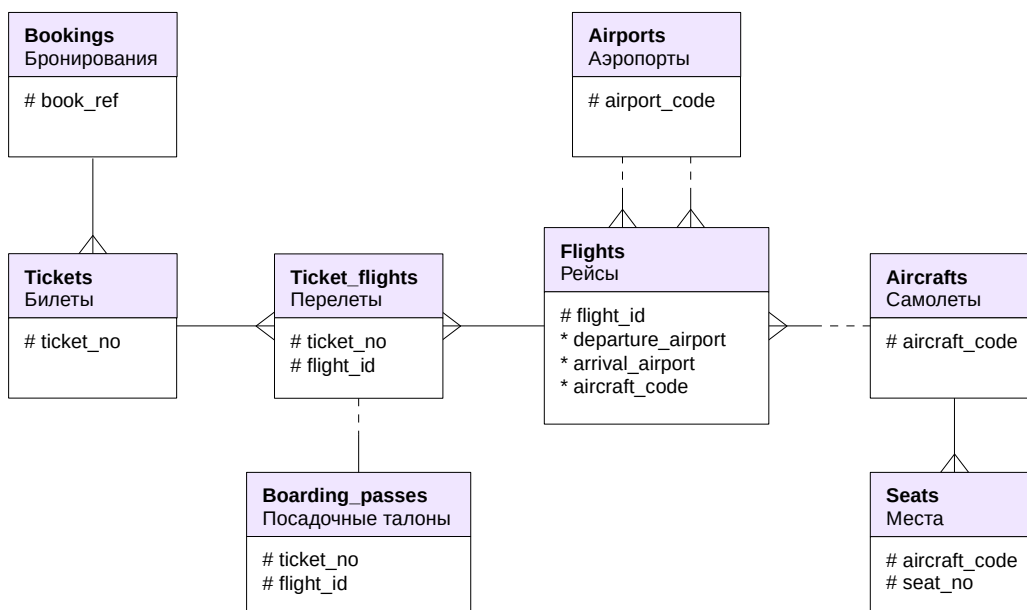
Демонстрационная база данных распространяется под лицензией PostgreSQL.

База данных доступна в трех вариантах, отличающихся размером. Например, в курсе по оптимизации запросов используется база большого объема, содержащая данные по полетам за один год.

В данной теме рассматривается версия демобазы от 15.08.2017.

<https://postgrespro.ru/education/demodb>

# Общая схема



4

Основной сущностью является **бронирование** (bookings).

В одно бронирование можно включить нескольких пассажиров, каждому из которых выписывается отдельный **билет** (tickets).

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) в другой. Рейсы с одним номером имеют одинаковые пункты вылета и назначения, но будут отличаться датой отправления.

При регистрации на рейс пассажиру выдается **посадочный талон** (boarding\_passes), в котором указано место в самолете. Пассажир может зарегистрироваться только на тот рейс, который есть у него в билете. Комбинация рейса и места в самолете уникальна.

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

На приведенной схеме отмечены только столбцы, соответствующие первичным и внешним ключам. Далее мы рассмотрим основные объекты демонстрационной базы данных подробнее.

## Bookings

пассажир заранее (за месяц) бронирует билет себе и, возможно, нескольким другим пассажирам

|              |   |
|--------------|---|
| book_ref     | номер бронирования (комбинация букв и цифр)       |
| book_date    | дата бронирования                                 |
| total_amount | общая стоимость включенных в бронирование билетов |

5

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.

| Столбец      | Тип           | Модификаторы | Описание                  |
|--------------|---------------|--------------|---------------------------|
| book_ref     | char(6)       | not null     | Номер бронирования        |
| book_date    | timestampz    | 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)

## Бронирование

Начнем с бронирования и выберем какое-нибудь одно:

```
=> SELECT * FROM bookings b WHERE b.book_ref = '0824C5';
```

| book_ref | book_date              | total_amount |
|----------|------------------------|--------------|
| 0824C5   | 2017-07-25 23:36:00+03 | 112400.00    |

(1 row)

Мы видим дату бронирования и общую сумму.

Если сравнивать дату с текущей, то бронирование сделано довольно давно:

```
=> SELECT now();
```

| now                           |
|-------------------------------|
| 2022-07-03 15:04:08.048821+03 |

(1 row)

Но для демобазы «текущим» моментом является другая дата:

```
=> SELECT bookings.now();
```

| now                    |
|------------------------|
| 2017-08-15 18:00:00+03 |

(1 row)

Так что «на самом деле» билеты забронированы 20 дней назад:

```
=> SELECT bookings.now() - b.book_date  
FROM bookings b  
WHERE b.book_ref = '0824C5';
```

| ?column?         |
|------------------|
| 20 days 18:24:00 |

(1 row)

## Tickets

билет выдается на одного пассажира и может включать несколько перелетов

ни идентификатор пассажира, ни имя не являются постоянными; нельзя однозначно найти все билеты одного и того же пассажира

|                |  |
|----------------|--|
| ticket_no      | <i>номер билета</i>                              |
| book_ref       | <i>номер бронирования</i>                        |
| passenger_id   | <i>идентификатор пассажира (номер документа)</i> |
| passenger_name | <i>имя пассажира</i>                             |
| contact_data   | <i>контактные данные пассажира</i>               |

7

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\_data).

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.

| 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)
```

Билеты

Посмотрим, какие билеты включены в выбранное бронирование:

```
=> SELECT t.*
FROM bookings b
      JOIN tickets t ON t.book_ref = b.book_ref
WHERE b.book_ref = '0824C5';
```

| ticket_no     | book_ref | passenger_id | passenger_name    | contact_data              |
|---------------|----------|--------------|-------------------|---------------------------|
| 0005435126781 | 0824C5   | 7247 393204  | ALEKSANDR MATVEEV | {"phone": "+70095062310"} |
| 0005435126782 | 0824C5   | 1745 826066  | NINA KRASNOVA     | {"phone": "+70876976071"} |

(2 rows)

Летят два человека; на каждого оформляется собственный билет с информацией о пассажире.



## Ticket\_flights

перелет соединяет билеты с рейсами

|                 |                     |
|-----------------|---------------------|
| ticket_no       | номер билета        |
| flight_id       | идентификатор рейса |
| fare_conditions | класс обслуживания  |
| amount          | стоимость перелета  |

Перелет соединяет билет с рейсом и идентифицируется их номерами. Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare\_conditions).

| 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)
```

## Перелеты

По какому маршруту летят пассажиры? Добавим в запрос перелеты.

```
=> SELECT tf.*
FROM tickets t
      JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no
WHERE t.ticket_no = '0005435126781';
```

| ticket_no     | flight_id | fare_conditions | amount   |
|---------------|-----------|-----------------|----------|
| 0005435126781 | 22566     | Economy         | 11700.00 |
| 0005435126781 | 71439     | Economy         | 3200.00  |
| 0005435126781 | 74643     | Economy         | 8800.00  |
| 0005435126781 | 94335     | Economy         | 11700.00 |
| 0005435126781 | 95726     | Economy         | 3200.00  |
| 0005435126781 | 206625    | Business        | 26400.00 |

(6 rows)

Здесь мы смотрим только на один билет — все маршруты в одном бронировании всегда совпадают.

Видим, что в билете 6 перелетов; из них один бизнес-классом, другие — экономом.

## Flights

рейс выполняется по расписанию из одного аэропорта в другой  
естественный ключ — номер рейса и дата отправления,  
но используется суррогатный ключ

|  |   |
|--|---|
| <code>flight_id</code>                   | <i>идентификатор рейса</i>              |
| <code>flight_no</code>                   | <i>номер рейса</i>                      |
| <code>scheduled_departure/arrival</code> | <i>вылет и прилет по расписанию</i>     |
| <code>actual_departure/arrival</code>    | <i>фактический вылет и прилет</i>       |
| <code>departure/arrival_airport</code>   | <i>аэропорты отправления и прибытия</i> |
| <code>status</code>                      | <i>статус рейса</i>                     |
| <code>aircraft_code</code>               | <i>код самолета</i>                     |

11

Рейс соединяет аэропорты вылета и прибытия. Такое понятие, как «рейс с пересадками» отсутствует: если нет прямого рейса, в билет просто включаются несколько рейсов.

| Столбец                          | Тип                      | Модификаторы          | Описание                    |
|----------------------------------|--------------------------|-----------------------|-----------------------------|
| <code>flight_id</code>           | <code>serial</code>      | <code>not null</code> | Идентификатор рейса         |
| <code>flight_no</code>           | <code>char(6)</code>     | <code>not null</code> | Номер рейса                 |
| <code>scheduled_departure</code> | <code>timestampz</code>  | <code>not null</code> | Время вылета по расписанию  |
| <code>scheduled_arrival</code>   | <code>timestampz</code>  | <code>not null</code> | Время прилёта по расписанию |
| <code>departure_airport</code>   | <code>char(3)</code>     | <code>not null</code> | Аэропорт отправления        |
| <code>arrival_airport</code>     | <code>char(3)</code>     | <code>not null</code> | Аэропорт прибытия           |
| <code>status</code>              | <code>varchar(20)</code> | <code>not null</code> | Статус рейса                |
| <code>aircraft_code</code>       | <code>char(3)</code>     | <code>not null</code> | Код самолета, IATA          |
| <code>actual_departure</code>    | <code>timestampz</code>  |                       | Фактическое время вылета    |
| <code>actual_arrival</code>      | <code>timestampz</code>  |                       | 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)
```

## Рейсы

Теперь разберемся, какие рейсы скрываются за выбранными перелетами.

```
=> SELECT f.flight_id, f.scheduled_departure,
       f.departure_airport dep, f.arrival_airport arr,
       f.status, f.aircraft_code aircraft
FROM tickets t
  JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no
  JOIN flights f ON f.flight_id = tf.flight_id
WHERE t.ticket_no = '0005435126781'
ORDER BY f.scheduled_departure;
```

| flight_id | scheduled_departure    | dep | arr | status    | aircraft |
|-----------|------------------------|-----|-----|-----------|----------|
| 22566     | 2017-08-12 11:00:00+03 | VKO | PEE | Arrived   | 773      |
| 95726     | 2017-08-12 15:30:00+03 | PEE | SVX | Arrived   | SU9      |
| 74643     | 2017-08-13 11:30:00+03 | SVX | SGC | Arrived   | SU9      |
| 206625    | 2017-08-15 14:45:00+03 | SGC | SVX | Departed  | SU9      |
| 71439     | 2017-08-16 08:50:00+03 | SVX | PEE | On Time   | SU9      |
| 94335     | 2017-08-16 18:55:00+03 | PEE | VKO | Scheduled | 773      |

(6 rows)

Видим три рейса «туда» и три «обратно». «Туда» все рейсы уже совершены (Arrived), а в настоящее время пассажир летит «обратно» (Departed). Следующий рейс будет по расписанию (On Time), а на последний еще не открыта регистрация (Scheduled).

Посмотрим внимательнее на все столбцы одного из рейсов.

```
=> SELECT * FROM flights f WHERE f.flight_id = 22566 \gx
```

```
-[ RECORD 1 ]-----+-----
flight_id      | 22566
flight_no      | PG0412
scheduled_departure | 2017-08-12 11:00:00+03
scheduled_arrival  | 2017-08-12 12:25:00+03
departure_airport  | VKO
arrival_airport    | PEE
status          | Arrived
aircraft_code     | 773
actual_departure   | 2017-08-12 11:01:00+03
actual_arrival     | 2017-08-12 12:25:00+03
```

Реальное время может отличаться от времени по расписанию (обычно не сильно).

Номер flight\_no одинаков для всех рейсов, следующих по одному маршруту по расписанию:

```
=> SELECT f.flight_id, f.flight_no, f.scheduled_departure
FROM flights f
WHERE f.flight_no = 'PG0412'
ORDER BY f.scheduled_departure
LIMIT 10;
```

| flight_id | flight_no | scheduled_departure    |
|-----------|-----------|------------------------|
| 22784     | PG0412    | 2016-08-15 11:00:00+03 |
| 22746     | PG0412    | 2016-08-16 11:00:00+03 |
| 22721     | PG0412    | 2016-08-17 11:00:00+03 |
| 22691     | PG0412    | 2016-08-18 11:00:00+03 |
| 22749     | PG0412    | 2016-08-19 11:00:00+03 |
| 22508     | PG0412    | 2016-08-20 11:00:00+03 |
| 22493     | PG0412    | 2016-08-21 11:00:00+03 |
| 22496     | PG0412    | 2016-08-22 11:00:00+03 |
| 22483     | PG0412    | 2016-08-23 11:00:00+03 |
| 22501     | PG0412    | 2016-08-24 11:00:00+03 |

(10 rows)

## Airports

город не выделен в отдельную таблицу

реализация: многоязычное представление над airports\_data

|              |   |
|--------------|---|
| airport_code | код аэропорта                           |
| airport_name | название аэропорта                      |
| city         | город                                   |
| coordinates  | координаты аэропорта (долгота и широта) |
| timezone     | часовой пояс                            |

13

Аэропорт идентифицируется трехбуквенным кодом (airport\_code) и имеет свое имя (airport\_name).

Для города не предусмотрено отдельной сущности, но введено поле с названием города (city), позволяющее найти аэропорты одного города. Это представление также включает координаты аэропорта (coordinates) и часовой пояс (timezone).

Значения полей airport\_name и city определяются в зависимости от выбранного в конфигурационном параметре *bookings.lang* языка.

| Столбец      | Тип     | Модификаторы | Описание               |
|--------------|---------|--------------|------------------------|
| airport_code | char(3) | not null     | Код аэропорта          |
| airport_name | text    | not null     | Название аэропорта     |
| city         | text    | not null     | Город                  |
| coordinates  | point   | not null     | Координаты аэропорта   |
| timezone     | text    | not null     | Часовой пояс аэропорта |

Определение представления:

```
SELECT ml.airport_code,  
       ml.airport_name ->> lang() AS airport_name,  
       ml.city ->> lang() AS city,  
       ml.coordinates,  
       ml.timezone  
FROM airports_data ml;
```

## Аэропорты

В качестве ключа для аэропортов используется общепринятый трехбуквенный код. Посмотрим полную информацию об одном аэропорте:

```
=> SELECT * FROM airports WHERE airport_code = 'VKO' \gx
```

```
-[ RECORD 1 ]+-----  
airport_code | VKO  
airport_name | Внуково  
city         | Москва  
coordinates  | (37.2615013123,55.5914993286)  
timezone     | Europe/Moscow
```

Помимо названия и города, хранятся координаты аэропорта и часовой пояс.

Теперь мы можем расшифровать сведения о рейсах:

```
=> SELECT f.scheduled_departure,  
       dep.airport_code || ' ' || dep.city || ' (' || dep.airport_name || ')' departure,  
       arr.airport_code || ' ' || arr.city || ' (' || arr.airport_name || ')' arrival  
FROM tickets t  
     JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no  
     JOIN flights f ON f.flight_id = tf.flight_id  
     JOIN airports dep ON dep.airport_code = f.departure_airport  
     JOIN airports arr ON arr.airport_code = f.arrival_airport  
WHERE t.ticket_no = '0005435126781'  
ORDER BY f.scheduled_departure;
```

| scheduled_departure    | departure                   | arrival                     |
|------------------------|-----------------------------|-----------------------------|
| 2017-08-12 11:00:00+03 | VKO Москва (Внуково)        | PEE Пермь (Пермь)           |
| 2017-08-12 15:30:00+03 | PEE Пермь (Пермь)           | SVX Екатеринбург (Кольцово) |
| 2017-08-13 11:30:00+03 | SVX Екатеринбург (Кольцово) | SGC Сургут (Сургут)         |
| 2017-08-15 14:45:00+03 | SGC Сургут (Сургут)         | SVX Екатеринбург (Кольцово) |
| 2017-08-16 08:50:00+03 | SVX Екатеринбург (Кольцово) | PEE Пермь (Пермь)           |
| 2017-08-16 18:55:00+03 | PEE Пермь (Пермь)           | VKO Москва (Внуково)        |

(6 rows)

Чтобы не выписывать каждый раз подобный запрос, существует представление flights\_v:

```
=> SELECT * FROM flights_v f WHERE f.flight_id = 22566 \gx
```

```
-[ RECORD 1 ]+-----  
flight_id      | 22566  
flight_no      | PG0412  
scheduled_departure | 2017-08-12 11:00:00+03  
scheduled_departure_local | 2017-08-12 11:00:00  
scheduled_arrival   | 2017-08-12 12:25:00+03  
scheduled_arrival_local | 2017-08-12 14:25:00  
scheduled_duration  | 01:25:00  
departure_airport   | VKO  
departure_airport_name | Внуково  
departure_city      | Москва  
arrival_airport      | PEE  
arrival_airport_name | Пермь  
arrival_city         | Пермь  
status              | Arrived  
aircraft_code        | 773  
actual_departure     | 2017-08-12 11:01:00+03  
actual_departure_local | 2017-08-12 11:01:00  
actual_arrival       | 2017-08-12 12:25:00+03  
actual_arrival_local | 2017-08-12 14:25:00  
actual_duration      | 01:24:00
```

Здесь видим и локальное время в часовых поясах городов отправления и прибытия, длительность полета, названия аэропортов.

Поскольку в демобазе маршруты не меняются со временем, из таблицы рейсов можно выделить информацию, которая не зависит от конкретной даты вылета. Такая информация собрана в представлении routes:

```
=> SELECT * FROM routes r WHERE r.flight_no = 'PG0412' \gx
```

```
-[ RECORD 1 ]-----+-----
flight_no      | PG0412
departure_airport | VKO
departure_airport_name | Внуково
departure_city  | Москва
arrival_airport | PEE
arrival_airport_name | Пермь
arrival_city    | Пермь
aircraft_code   | 773
duration        | 01:25:00
days_of_week    | {1,2,3,4,5,6,7}
```

Видно, что рейсы выполняются ежедневно (массив days\_of\_week).

## Aircrafts

модели самолетов, выполняющие рейсы

реализация: многоязычное представление над aircrafts\_data

|               |                                   |
|---------------|-----------------------------------|
| aircraft_code | код самолета                      |
| model         | модель самолета                   |
| range         | максимальная дальность полета, км |

15

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

Значение поля model определяется в зависимости от выбранного в конфигурационном параметре *bookings.lang* языка.

| 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 |

Определение представления:

```
SELECT ml.aircraft_code,  
       ml.model ->> lang() AS model,  
       ml.range  
FROM aircrafts_data ml;
```



## Самолеты

Модели самолетов, обслуживающих рейсы, также используют стандартные трехсимвольные коды в качестве первичных ключей.

```
=> SELECT a.*  
FROM flights f  
      JOIN aircrafts a ON a.aircraft_code = f.aircraft_code  
WHERE f.flight_id = 22566;
```

| aircraft_code | model         | range |
|---------------|---------------|-------|
| 773           | Боинг 777-300 | 11100 |

(1 row)

## Seats

места определяют схему салона

все самолеты одной модели имеют одну и ту же компоновку салона

|                 |                    |
|-----------------|--------------------|
| aircraft_code   | код самолета       |
| seat_no         | номер места        |
| fare_conditions | класс обслуживания |

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.

| 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

## Места

В демобазе все самолеты одной модели имеют одинаковую конфигурацию салона. Посмотрим на первый ряд:

```
=> SELECT s.*
FROM flights f
      JOIN aircrafts a ON a.aircraft_code = f.aircraft_code
      JOIN seats s ON s.aircraft_code = a.aircraft_code
WHERE f.flight_id = 22566
AND s.seat_no ~ '^1.$';
```

| aircraft_code | seat_no | fare_conditions |
|---------------|---------|-----------------|
| 773           | 1A      | Business        |
| 773           | 1C      | Business        |
| 773           | 1D      | Business        |
| 773           | 1G      | Business        |
| 773           | 1H      | Business        |
| 773           | 1K      | Business        |

(6 rows)

Это бизнес-класс.

А вот общее число мест различных классов обслуживания:

```
=> SELECT s.fare_conditions, count(*)
FROM seats s
WHERE s.aircraft_code = '733'
GROUP BY s.fare_conditions;
```

| fare_conditions | count |
|-----------------|-------|
| Business        | 12    |
| Economy         | 118   |

(2 rows)

## Boarding\_passes

посадочный талон выдается при регистрации на рейс

|                          |   |
|--------------------------|---|
| <code>ticket_no</code>   | <i>номер билета</i>                                     |
| <code>flight_id</code>   | <i>идентификатор рейса</i>                              |
| <code>boarding_no</code> | <i>номер посадочного талона (в порядке регистрации)</i> |
| <code>seat_no</code>     | <i>номер места</i>                                      |

19

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`).

| Столбец                  | Тип                     | Модификаторы          | Описание                 |
|--------------------------|-------------------------|-----------------------|--------------------------|
| <code>ticket_no</code>   | <code>char(13)</code>   | <code>not null</code> | Номер билета             |
| <code>flight_id</code>   | <code>integer</code>    | <code>not null</code> | Идентификатор рейса      |
| <code>boarding_no</code> | <code>integer</code>    | <code>not null</code> | Номер посадочного талона |
| <code>seat_no</code>     | <code>varchar(4)</code> | <code>not null</code> | 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)
```

## Посадочные талоны

На каких местах сидел наш пассажир? Для этого надо заглянуть в посадочный талон, который выдается при регистрации на рейс:

```
=> SELECT f.status, bp.*
FROM tickets t
  JOIN ticket_flights tf ON tf.ticket_no = t.ticket_no
  JOIN flights f ON f.flight_id = tf.flight_id
  LEFT JOIN boarding_passes bp
    ON bp.ticket_no = tf.ticket_no AND bp.flight_id = tf.flight_id
WHERE t.ticket_no = '0005435126781'
ORDER BY f.scheduled_departure;
```

| status    | ticket_no     | flight_id | boarding_no | seat_no |
|-----------|---------------|-----------|-------------|---------|
| Arrived   | 0005435126781 | 22566     | 4           | 22A     |
| Arrived   | 0005435126781 | 95726     | 64          | 19D     |
| Arrived   | 0005435126781 | 74643     | 42          | 8D      |
| Departed  | 0005435126781 | 206625    | 11          | 3F      |
| On Time   |               |           |             |         |
| Scheduled |               |           |             |         |

(6 rows)

На два оставшихся рейса пассажир еще не зарегистрировался.

Конфигурационный параметр

*bookings.lang*

Таблицы для хранения многоязычных названий

airports\_data

aircrafts\_data

Язык, на котором выводятся названия городов, аэропортов и моделей самолетов, переключается с помощью конфигурационного параметра *bookings.lang*. В состав демобазы входят названия на русском (ru) и английском (en) языках.

Можно самостоятельно расширить языковую поддержку, добавив в строки таблиц `airports_data` и `aircrafts_data` названия на произвольном языке.

## Многоязычность

В демобазе заложена возможность перевода названий аэропортов, городов и самолетов на другие языки. Как мы видели, по умолчанию все названия выводятся по-русски:

```
=> SELECT * FROM airports a WHERE a.airport_code = 'VKO' \gx
```

```
-[ RECORD 1 ]+-----  
airport_code | VKO  
airport_name | Внуково  
city         | Москва  
coordinates  | (37.2615013123,55.5914993286)  
timezone     | Europe/Moscow
```

Чтобы сменить язык, достаточно установить конфигурационный параметр:

```
=> SET bookings.lang = 'en';
```

SET

```
=> SELECT * FROM airports a WHERE a.airport_code = 'VKO' \gx
```

```
-[ RECORD 1 ]+-----  
airport_code | VKO  
airport_name | Vnukovo International Airport  
city         | Moscow  
coordinates  | (37.2615013123,55.5914993286)  
timezone     | Europe/Moscow
```

Реализация использует представление над базовой таблицей, которая содержит переводы в формате JSON:

```
=> SELECT * FROM airports_data ml WHERE ml.airport_code = 'VKO' \gx
```

```
-[ RECORD 1 ]+-----  
airport_code | VKO  
airport_name | {"en": "Vnukovo International Airport", "ru": "Внуково"}  
city         | {"en": "Moscow", "ru": "Москва"}  
coordinates  | (37.2615013123,55.5914993286)  
timezone     | Europe/Moscow
```

Схема демобазы достаточно проста, но позволяет писать сложные и интересные запросы

Данные в демобазе похожи на настоящие

Демобазы могут использоваться для изучения языка SQL, демонстрации возможностей PostgreSQL и т. п.



Напишите несколько запросов к демонстрационной базе данных.

1. Сколько человек бывает включено в одно бронирование?
2. До каких городов нельзя добраться без пересадок из Москвы?
3. Какая модель самолета выполняет больше всего рейсов, а какая — меньше всего?
4. А какая модель перевозит больше всего пассажиров?

## 1. Сколько человек в одном бронировании?

Посчитаем количество человек в каждом бронировании, а затем число бронирований для каждого количества.

```
=> SELECT tt.cnt, count(*)
FROM (
  SELECT count(*) cnt
  FROM tickets t
  GROUP BY t.book_ref
) tt
GROUP BY tt.cnt
ORDER BY tt.cnt;
```

| cnt | count   |
|-----|---------|
| 1   | 1388875 |
| 2   | 613356  |
| 3   | 101440  |
| 4   | 7245    |
| 5   | 194     |

(5 rows)

## 2. До каких городов нельзя добраться без пересадок из Москвы?

Найдем города, куда можно добраться, и выведем все остальные.

```
=> SELECT a.city
FROM airports a
EXCEPT
SELECT arr.city
FROM flights f
  JOIN airports dep ON f.departure_airport = dep.airport_code
  JOIN airports arr ON f.arrival_airport = arr.airport_code
WHERE dep.city = 'Москва';
```

| city                 |
|----------------------|
| Калуга               |
| Когалым              |
| Якутск               |
| Новокузнецк          |
| Сургут               |
| Иркутск              |
| Удачный              |
| Кызыл                |
| Стрежевой            |
| Ярославль            |
| Иваново              |
| Усть-Кут             |
| Магадан              |
| Чита                 |
| Череповец            |
| Комсомольск-на-Амуре |
| Усть-Илимск          |
| Москва               |
| Благовещенск         |
| Ухта                 |
| Нижнекамск           |

(21 rows)

Интересно, что из Москвы в Москву без пересадок добраться не получится.

## 3. Какие модели выполняют больше всего и меньше всего рейсов?

```
=> SELECT a.model, f.cnt
FROM aircrafts a
LEFT JOIN (
  SELECT f.aircraft_code, count(*) cnt
  FROM flights f
  GROUP BY f.aircraft_code
) f
ON f.aircraft_code = a.aircraft_code
ORDER BY cnt DESC NULLS LAST;
```

| model               | cnt   |
|---------------------|-------|
| Сессна 208 Караван  | 60196 |
| Бомбардье CRJ-200   | 58611 |
| Сухой Суперджет-100 | 55213 |
| Аэробус A321-200    | 12672 |
| Боинг 737-300       | 8263  |
| Аэробус A319-100    | 8032  |
| Боинг 767-300       | 7920  |
| Боинг 777-300       | 3960  |
| Аэробус A320-200    |       |

(9 rows)

Больше всех трудится маленькая Сессна, а одна модель авиапарка вообще не используется на рейсах.

#### 4. Какая модель перевозит больше всего пассажиров?

Число пассажиров на рейсе можно посчитать по посадочным талонам.

```
=> SELECT a.model, count(*) cnt
FROM boarding_passes bp
     JOIN flights f ON f.flight_id = bp.flight_id
     JOIN aircrafts a ON a.aircraft_code = f.aircraft_code
GROUP BY a.model
ORDER BY count(*) DESC;
```

| model               | cnt     |
|---------------------|---------|
| Сухой Суперджет-100 | 2767457 |
| Бомбардье CRJ-200   | 1155683 |
| Боинг 777-300       | 1111547 |
| Боинг 767-300       | 945568  |
| Аэробус A321-200    | 777370  |
| Боинг 737-300       | 649730  |
| Аэробус A319-100    | 407361  |
| Сессна 208 Караван  | 111096  |

(8 rows)