Описание базы данных

Демонстрационная база данных “Авиаперевозки” - это база данных авиаперевозок по России. Представляет собой временной срез данных на 13.10.2016.

Все объекты - таблицы и представления - данной базы находятся в схеме bookings, поэтому при обращении к ним указывается имя схемы.

База данных состоит из восьми таблиц и двух представлений, в которых содержится следующая информация:

1) таблица Бронирование (Bookings):

данные о номере, дате и общей сумме бронирования.

2) таблица Билеты (Tickets):

данные о номере билета, номере бронирования, данные о пассажире (ID, имя и контактные данные).

Tickets связана с Bookings номером бронирования.

3) таблица Самолеты (Aircrafts):

данные о коде и модели самолета, максимальной дальности перелета (в километрах).

4) таблица Места (Seats):

данные о коде самолета, номере места и классе обслуживания.

Класс обслуживания принимает одно из трех значений:

business - бизнес-класс,

comfort - комфорт-класс,

economy - эконом-класс.

Seats связана с Aircrafts кодом самолета.

5) таблица Аэропорты (Airports):

данные о коде, названии и городе аэропорта, координаты расположения (долгота и широта) и временная зона аэропорта.

6) таблица Рейсы (Flights):

данные о рейсе (ID и номер), время вылета и прилета по расписанию, аэропорты отправления и прибытия, статус рейса, код самолета, фактическое время вылета и прилета.

Статус рейса принимает одно из шести значений:

scheduled (доступен для бронирования),

on time (доступен для регистрации),

delayed (доступен для регистрации, но задержан),

departed (самолет уже в воздухе),

arrived (самолет прибыл в пункт назначения),

cancelled (рейс отменен).

Flights связана с Aircrafts кодом самолета, с Airports - связь с кодом аэропорта от аэропортов отправления и прибытия.

7) таблица Перелеты (Ticket flights):

данные о номере билета, ID рейса, классе обслуживания (варианты значений описаны ранее) и стоимости перелета.

Ticket flights связана с Tickets номером билета, с Flights - через ID рейса.

8) таблица Посадочные талоны (Boarding passes):

данные о номере билета, ID рейса, номере посадочного талона и номере места.

Чтобы исключить случаи выдачи нескольких посадочных талонов на одно место, комбинация “рейс - место” должна быть уникальной.

Поэтому Boarding passes связана с Ticket flights двумя атрибутами: номер билета и ID рейса.

В таблице Flights отражена не вся информация о рейсах. Дополнительные параметры, помимо тех, что даны в таблице Flights, приведены в Представлении Flights\_v. Это:

местное время отправления и прибытия по расписанию, продолжительность полета планируемая и фактическая, названия аэропортов отправления и прибытия,

города отправления и прибытия,

местное время фактического вылета и прилета.

Flights\_v связано с теми же таблицами, что и Flights (по тем же атрибутам).

Материализованное представление Routes содержит данные о маршрутах (часть таблицы Flights), которые не связаны с конкретными датами рейсов:

номер рейса,

код аэропорта отправления и прибытия,

название аэропорта отправления и прибытия,

город отправления и прибытия,

код самолета,

продолжительность полета (плановая),

дни недели, когда выполняются рейсы.

Данные в таблицах, а также связи между ними, отражают реальный процесс бронирования и перелетов. Например, в одно бронирование может быть включено несколько пассажиров, на каждого из которых выписывается отдельный билет; каждый билет может включать один или несколько перелетов (предполагается, что все билеты в одном бронировании имеют одинаковый набор перелетов); посадочный талон выдается каждому пассажиру на тот рейс, на который он зарегистрирован (за сутки до вылета); в каждой модели самолета свое количество мест по классам обслуживания пассажиров и т.д.

Демонстрационная база данных “Авиаперевозки” содержит достаточно данных для тренировочных запросов, а также получения ответов на такие бизнес-вопросы, как:

* требуется ли изменение (сокращение/увеличение) парка самолетов (в зависимости от количества и дальности выполняемых перелетов),
* каких расходных материалов (деталей, блоков и т.д.) нужно закупать больше (в зависимости от моделей самолетов, совершающих большее количество рейсов),
* по каким маршрутам можно оптимизировать количество рейсов (в зависимости от заполняемости салонов самолетов).

SQL запросы и их описание

1) города с более чем одним аэропортом (“1\_more than 1 air.sql”)

Логика решения: подсчет количества аэропортов в разрезе городов.

-- просмотр всех данных в таблице

select \* from bookings.airports\_data ad

-- основной запрос

with cte\_code as (

select ad.city->>'ru' as name\_city,

ad.airport\_name->>'ru' as name\_air,

count(ad.airport\_code) over (partition by ad.city->>'ru') as code\_air

from bookings.airports\_data ad

)

select cc.name\_city, cc.name\_air

from cte\_code cc

where cc.code\_air>1

order by cc.name\_city

Данные берутся из таблицы bookings.airports\_data ad, где информация по городам и названиям аэропортов представлена в виде json объектов.

Создается общее табличное выражение (CTE) (cte\_code), состоящее из данных:

* название города (как name\_city),
* название аэропорта (как name\_air),
* подсчет количества аэропортов по их кодам в разрезе каждого города с использованием оконной функции (как code\_air).

Из cte\_code выбираются только те значения, которые соответствуют условию: количество аэропортов больше 1 (code\_air > 1).

Выводятся данные name\_city и name\_air, которые сортируются по name\_city.

Результат: в двух городах - Москва и Ульяновск - больше 1 аэропорта ( 3 и 2 соответственно).

2) аэропорты, в которых есть рейсы на самолеты с максимальной дальностью перелета (“2\_max distance.sql”)

Логика решения: выявление самолета с максимальной дальностью перелетов, определение перечня аэропортов, из которых выполняет рейсы данный самолет.

-- просмотр парка самолетов

select \* from bookings.aircrafts\_data ad

order by ad."range" desc

-- максимальная дальность полета у Боинг 777-300, код 773

-- просмотр всех данных в bookings.routes r

select \* from bookings.routes r

-- основной запрос

select r.departure\_airport as code\_airport, r.departure\_airport\_name as name\_airport, r.departure\_city as city

from bookings.routes r

where r.aircraft\_code = '773'

group by 1, 2, 3

order by city

Для решения задачи просматриваются данные по парку самолетов (bookings.aircrafts\_data ad) для определения самолета с максимальной дальностью перелета.

Им оказывается Боинг 777-300, код 773.

Информация об аэропортах берется из материализованного представления bookings.routes r (здесь содержатся все данные о маршрутах, включая названия городов).

Из таблицы bookings.routes r выбираются данные, которые соответствуют условию: код самолета равен 773 (r.aircraft\_code = '773').

Эти данные группируются по выбранным (выводимым) параметрам.

Выводится следующая информация:

* код аэропорта (как code\_airport),
* название аэропорта (как name\_airport),
* город аэропорта (как city),

которая сортируется по city.

Результат: из 7 аэропортов (три в Москве, по одному в Екатеринбурге, Новосибирске, Перми, Сочи) выполняются рейсы с максимальной дальностью перелета.

3) брони, по которым не совершалось перелетов (“3\_booking no flight.sql”)

Логика решения: нахождение забронированных билетов, по которым не получены посадочные талоны, на самолеты, которые уже улетели.

select t.book\_ref as book\_num, t.ticket\_no as ticket\_num, bp.boarding\_no as board\_num, f.status

from bookings.tickets t

join bookings.ticket\_flights tf using (ticket\_no)

join bookings.flights f using (flight\_id)

left join bookings.boarding\_passes bp on tf.ticket\_no = bp.ticket\_no and tf.flight\_id = bp.flight\_id

where bp.boarding\_no is null

and f.status in ('Arrived','Departed')

Необходимая информация содержится в объединяемых таблицах: “билеты”, “перелеты” (используется для связи со следующими двумя таблицами), “рейсы” и “посадочные талоны”.

Выбираются данные, которые соответствуют условиям:

* посадочного талона нет (bp.boarding\_no is null)

и

* статус рейса или “самолет прибыл в пункт назначения”, или “самолет уже вылетел и находится в воздухе” (f.status in ('Arrived','Departed')).

Выводится информация по номеру бронирования (как book\_num), номеру билета (как ticket\_num), номеру посадочного талона (как board\_num) и статусу рейса (*по факту данные отсутствуют*).

Результат: броней, по которым не совершались перелеты, нет.

4) модели самолетов, совершающих наибольший % перелетов (“4\_percent flights.sql”)

Логика решения: по каждому самолету (по коду) расчет отношения количества рейсов, совершаемых им, к общему числу рейсов; определение модели самолета с наибольшим %.

-- основной запрос

with cte\_percent as (

select r.aircraft\_code,

count(r.flight\_no) over (partition by r.aircraft\_code) as count\_flight,

(select count(r.flight\_no)

from bookings.routes r) as total\_count

from bookings.routes r

)

select cp.aircraft\_code,

(cp.count\_flight \* 100 / cp.total\_count) as percent\_flights

from cte\_percent cp

group by 1, 2

order by percent\_flights desc

limit 1

-- модель самолета с кодом CR2

select ad.aircraft\_code, ad.model->>'ru' as model

from bookings.aircrafts\_data ad

where ad.aircraft\_code='CR2'

-- наибольший процент перелетов у Бомбардье CRJ-200

Данные о рейсах и кодах самолетов берутся из материализованного представления (bookings.routes r).

Создается общее табличное выражение CTE (cte\_percent), состоящее из данных:

* код самолета,
* подсчет количества рейсов в разрезе самолетов (их кодов) с использованием оконной функции (как count\_flight),
* подсчет общего количества рейсов (как total\_count).

Данные из cte\_percent группируются по коду самолетов и количеству рейсов (count\_flight).

Выбирается следующая информация:

* код самолета,
* рассчитанный по формуле процент перелетов (количество рейсов, совершенных самолетом, х 100 и деленное на общее количество рейсов) (как percent\_flights).

Данные сортируются по значению percent\_flights в порядке убывания и выводится одно значение, соответствующее коду самолета с максимальным % перелетов.

Этот код самолета - CR2.

Из таблицы самолетов (bookings.aircrafts\_data ad) выбирается модель с данным кодом (ad.aircraft\_code='CR2').

Результат: самолет Бомбардье CRJ-200 совершает наибольший % перелетов.

5) города, в которые можно добраться бизнес-классом дешевле, чем эконом-классом (“5\_business and economy.sql”)

Логика решения: выбор билета эконом-класса с максимальной стоимостью и билета бизнес-класса с минимальной стоимостью на тот же рейс, сравнение их между собой.

-- просмотр всех данных в bookings.ticket\_flights tf

select \* from bookings.ticket\_flights tf

-- основной запрос

select max, min

from

(select tf.flight\_id, max(tf.amount) from bookings.ticket\_flights tf

where tf.fare\_conditions = 'Economy'

group by 1) as ma

join

(select tf.flight\_id, min(tf.amount) from bookings.ticket\_flights tf

where tf.fare\_conditions = 'Business'

group by 1) as mi

on ma.flight\_id = mi.flight\_id

where max > min

-- рейсов, на которых стоимость билетов Business меньше стоимости Economy, нет

Данные берутся из таблицы перелетов (bookings.ticket\_flights tf).

Создаются две таблицы, которые объединяются по id рейсов (flight\_id):

* первая - по классу обслуживания “Economy” выбираются id рейсов и билеты с максимальной стоимостью,
* вторая - по классу обслуживания “Business” выбираются id рейсов и билеты с минимальной стоимостью.

Из этих объединенных таблиц выбираются те строки с максимальной (max) и минимальной (min) стоимостью, где max > min (*по факту данные отсутствуют*).

Результат: билетов бизнес-класса дешевле, чем эконом-класса, нет, соответственно, и городов, куда можно добраться бизнес-классам дешевле, нет.

6) максимальное время задержки вылета самолета (“6\_max time wait.sql”)

Логика решения: нахождение максимальной разницы между фактическим временем вылета и временем вылета по расписанию.

select f.flight\_no, (f.actual\_departure - f.scheduled\_departure) as time\_wait

from bookings.flights f

where f.actual\_departure is not null

order by time\_wait desc

limit 1

С учетом условия о наличии фактического времени вылета (f.actual\_departure is not null) из таблицы рейсов (bookings.flights f) выбираются данные по номеру рейса (f.flight\_no) и рассчитанному значению разницы между фактическим и плановым временем вылета (f.actual\_departure - f.scheduled\_departure) (как time\_wait).

Для определения максимального времени задержки вылета данные сортируются по значению time\_wait в порядке убывания и выводится одно (первое) значение.

Результат: максимальное время задержки вылета самолета составляет 4 часа 41 минуту.

7) города, между которыми нет прямых рейсов (“7\_no right flights.sql”)

Логика решения: создание таблицы с данными о возможных комбинациях городов, ее сравнение с таблицей существующих (прямых) рейсов.

select r1.departure\_city as city1, r2.departure\_city as city2

from bookings.routes r1

cross join bookings.routes r2

where r1.departure\_city <> r2.departure\_city

group by 1, 2

except

select r.departure\_city, r.arrival\_city

from bookings.routes r

Данные берутся из материализованного представления (bookings.routes r).

Столбец с городами отправления (departure\_city) составляет первую таблицу - bookings.routes r1. Последняя, с помощью оператора cross join, соединяется сама с собой, с таблицей bookings.routes r2. При этом исключаются соединения города с собой же: рейсов из любого города в тот же самый город быть не должно (r1.departure\_city <> r2.departure\_city).

Получается итоговая первая таблица, в которой каждый город соединен со всеми остальными городами.

Вторая таблица - это города вылета и прилета по существующим рейсам (r.departure\_city и r.arrival\_city соответственно).

С помощью оператора EXCEPT создается итоговое множество из строк первой таблицы с возможными комбинациями городов, откуда исключаются строки второй таблицы с рейсами (строки-дубликаты не включаются).

Результат: сформирован перечень городов, между которыми нет прямых рейсов.

ER-диаграмма (самостоятельно построенная)

Приложена отдельным файлом в виде скриншота.

Использование представлений или функций при решении задач

**CTE** - общее табличное выражение, временный объект для выполнения конкретного запроса. Создание представлений не потребовалось: каждая из решенных задач не связана друг с другом.

**Count()** - подсчет количества строк.

**Оконная функция** - выполнение вычислений с набором строк.

**Min() и Max()** - нахождение минимальных и максимальных значений.

Локально установленная база данных “demo”

Скриншот установки приложен отдельным файлом.