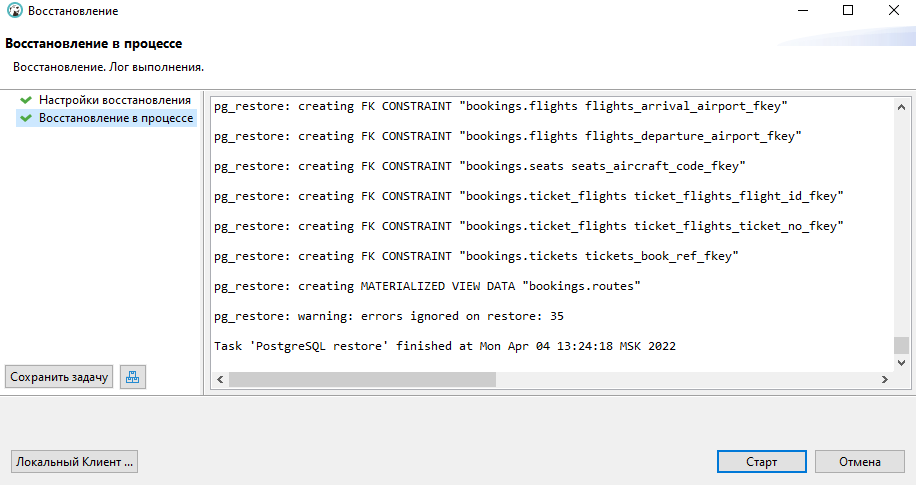
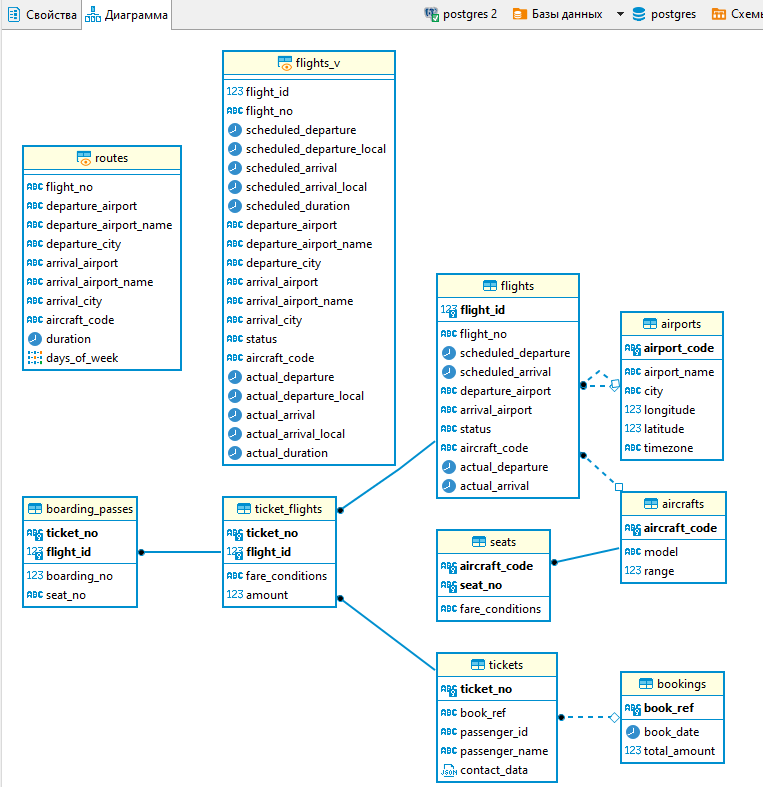
ИТОГОВАЯ РАБОТА ПО КУРСУ «SQL И ПОЛУЧЕНИЕ ДАННЫХ».

1. В работе использовался локальный тип подключения.

Восстановление БД из avia.backup: 

1. ER – диаграмма: 
2. БД bookings состоит из:

Таблиц:

1. aircrafts - Самолеты
2. aircraft - Аэропорты
3. boarding\_passes - Посадочные
4. bookings - Бронирования
5. flights - Рейсы
6. seats - Места
7. ticket\_flights - Перелеты
8. tickets – Билеты

Представлений:

1. flights\_v - Рейсы
2. routes – Маршруты (материальное)

**4.**

**4**.**1** Описание таблиц:

1. **aircrafts**

Состоит из полей:

* + aircraft\_code – код модели самолета, тип данных bpchar (3), ограничение NOT NULL
  + model – наименование модели самолета, тип данных text, ограничение NOT NULL
  + range – максимальная дальность полета (км), тип данных int4, ограничение NOT NULL

Первичный ключ – поле aircraft\_code. По нему **aircrafts** связана с таблицами **flights** и **seats** Ограничение на поле range – range > 0

1. **airports**

Состоит из полей:

* airport\_code – код аэропорта, тип данных bpchar (3), ограничение NOT NULL
* airport\_name – Название аэропорта, тип данных text, ограничение NOT NULL
* city – Город, тип данных text, ограничение NOT NULL
* longitude – Координаты аэропорта: долгота, тип данных float8, ограничение NOT NULL
* latitude – Координаты аэропорта: широта, тип данных float8, ограничение NOT NULL
* timezone – Временная зона аэропорта, тип данных text, ограничение NOT NULL

Первичный ключ – поле airport\_code. По нему **airports** связана с таблицей **flights**

1. **boarding\_passes**

Состоит из полей:

* ticket\_no – Номер билета, тип данных bpchar (13), ограничение NOT NULL
* flight\_id – Идентификатор рейса, тип данных int4, ограничение NOT NULL
* boarding\_no – Номер посадочного талона, тип данных int4, ограничение NOT NULL
* seat\_no – Номер места, тип данных varchar (4), ограничение NOT NULL

Первичный ключ составной – поля ticket\_no + flight\_id. По нему осуществляется ограничение внешнего ключа с таблицей **ticket\_flights**

1. **bookings**

Состоит из полей:

* book\_ref – Номер бронирования, тип данных bpchar (6), ограничение NOT NULL
* book\_date – Дата бронирования, тип данных timestamptz, ограничение NOT NULL
* total\_amount – Полная сумма бронирования, тип данных numeric (10,2), ограничение NOT NULL

Первичный ключ – поле book\_ref. По нему **bookings** связана с таблицей **tickets**

1. **flights**

Состоит из полей:

* flight\_id – Идентификатор рейса, тип данных serial4, ограничение NOT NULL
* flight\_no – Номер рейса, тип данных bpchar (6), ограничение NOT NULL
* scheduled\_departure – Время вылета по расписанию, тип данных timestamptz, ограничение NOT NULL
* scheduled\_arrival – Время прилета по расписанию, тип данных timestamptz, ограничение NOT NULL
* departure\_airport – Аэропорт отправления, тип данных bpchar (3), ограничение NOT NULL
* arrival\_airport – Аэропорт прибытия, тип данных bpchar (3), ограничение NOT NULL
* status – Статус рейса, тип данных varchar (20), ограничение NOT NULL
* aircraft\_code – Код самолета, тип данных bpchar (3), ограничение NOT NULL
* actual\_departure – Фактическое время вылета, тип данных timestamptz
* actual\_arrival – Фактическое время прилета, тип данных timestamptz

Первичный ключ – поле flight\_id (суррогатный ключ). По нему **flights** связана с таблицей **ticket\_flights.** Также есть естественный составной ключ – поля flight\_no + scheduled\_departure.

Ограничения на поля: scheduled\_arrival > scheduled\_departure,

actual\_arrival IS NULL OR (actual\_departure IS NOT NULL AND actual\_arrival IS NOT NULL)

status – может принимать значения: 'On Time', 'Delayed', 'Departed', 'Arrived', 'Scheduled', 'Cancelled'

Ограничения внешнего ключа: по полю aircraft\_code с таблицей **aircrafts**,

по полям airport\_code, departure\_airport с таблицей airports

1. **seats**

Состоит из полей:

* aircraft\_code – код самолета, тип данных bpchar (3), ограничение NOT NULL
* seat\_no – Номер места, тип данных varchar (4), ограничение NOT NULL
* fare\_conditions – Класс обслуживания, тип данных varchar (10), ограничение NOT NULL

Первичный ключ составной – поля aircraft\_code + seat\_no

Ограничение на поле fare\_conditions – может принимать значения: 'Economy', 'Comfort', 'Business'

Ограничение внешнего ключа: по полю aircraft\_code с таблицей aircrafts

1. **tickets**

Состоит из полей:

* ticket\_no – Номер билета, тип данных bpchar (13), ограничение NOT NULL
* flight\_id – Идентификатор рейса, тип данных int4, ограничение NOT NULL
* fare\_conditions – Класс обслуживания, тип данных varchar (10), ограничение NOT NULL
* amount – Стоимость перелета, тип данных numeric (10,2), ограничение NOT NULL

Первичный ключ составной – поля ticket\_no + flight\_id. По нему **tickets** связана с таблицей **ticket\_flights**

Ограничения на поля: amount >= 0,

fare\_conditions может принимать значения 'Economy', 'Comfort', 'Business'

Ограничения внешнего ключа: по полю flight\_id с таблицей **flights**,

по полю ticket\_no с таблицей **tickets**

1. **tickets**

Состоит из полей:

* ticket\_no – Номер билета, тип данных bpchar (13), ограничение NOT NULL
* book\_ref – Номер бронирования, тип данных bpchar (13), ограничение NOT NULL
* passenger\_id – Идентификатор пассажира, тип данных varchar (20), ограничение NOT NULL
* passenger\_name – Имя пассажира, тип данных varchar (20), ограничение NOT NULL
* contact\_data – Контактные данные пассажира, тип данных jsonb

Первичный ключ – поле ticket\_no. По нему **tickets** связана с таблицей **ticket\_flights**

Ограничения внешнего ключа: по полю book\_ref с таблицей **bookings**

**4.2** Описание представлений:

1. **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).

2. **routes** (материализованное представление)

Данное представление также создано из таблицы flights, исключив из нее избыточные данные (идентификатор рейса, время вылета/время прилета (по расписанию/фактическое, статус) и добавив дополнительные данные (название аэропортов отправления/прибытия, города аэропортов отправления/прибытия, продолжительность полета, дни недели выполняемых рейсов). Так как данная информация меняется достаточно редко, имело смысл создать именно материализованное представление.

С помощью запросов к этой БД можно решать множество бизнес-задач:

Проанализировав загруженность рейсов на каждом направлении за период, можно добавить/убрать рейсы, если наблюдается изменение загруженности по неким временным отрезкам (летом большая загруженность, зимой низкая), можно ввести сезонное расписание. Если провести анализ по заполненности рейсов по дням недели, времени вылета, то также можно скорректировать расписание, добиваясь максимальной заполненности рейсов.

Проанализировав заполненность мест на рейсах в разрезе классов обслуживания, можно подобрать оптимальные модели самолетов на данные рейсы.

Проанализировав количество пассажиров, летящих из аэропорта А в аэропорт Б, с которым нет прямого сообщения, с пересадкой в аэропорту В, можно задуматься о введении прямых рейсов между аэропортами А и Б.

Проанализировав расстояние между аэропортами, можно подобрать оптимальную модель самолета для перелетов между этими аэропортами. Дополнительно можно посмотреть, какова заполненность мест по классам обслуживания на этих рейсах, и оценить целесообразность использования той или иной модели самолета.

**5. Запросы**

1. В каких городах больше одного аэропорта?

**select** city "Города, где больше одного аэропорта", **count**(air) "Количество аэропортов"

**from** (

**select** r.departure\_airport air, r.departure\_city city

**from** bookings.routes r

**group** **by** 2,1) s

**group** **by** city

**having** **count**(air) > 1

В подзапросе s выбираем из мат. представления все города и все находящиеся в них аэропорты, группируя по городам, а затем по аэропортам. Из получившейся таблицы в основном запросе выбираем название города и количество аэропортов в нем, при условии, что это количество больше 1.

1. В каких аэропортах есть рейсы, выполняемые самолетом с максимальной дальностью перелета?

**select** departure\_airport\_name "Аэропорт с рейсами максимальной дальности перелета"

**from** bookings.routes r

**where** (**select** aircraft\_code **from** bookings.aircrafts **order** **by** "range" **desc** **limit** 1) = aircraft\_code

**group** **by** 1

В подзапросе в таблице aircrafts сортируем самолеты по дальности перелета от большей к меньшей, берем только первую строку и из нее берем код самолета. В основном запросе из мат. представления выбираем аэропорты по условию, что рейсы совершает самолет с кодом, равным коду самолета из подзапроса, а затем группируем по названию аэропорта.

1. Вывести 10 рейсов с максимальным временем задержки вылета

**select** f.flight\_no "Номер рейса", f.actual\_departure::**date** "Дата вылета",

f.actual\_departure::**time** "Время вылета",

**extract** (**day** **from** (f.actual\_departure - f.scheduled\_departure)) || ' д. ' ||

**extract** (**hour** **from** (f.actual\_departure - f.scheduled\_departure)) || ' ч. ' ||

**extract** (**minute** **from** (f.actual\_departure - f.scheduled\_departure)) || ' мин.' "Время задержки рейса"

**from** bookings.flights\_v f

**where** f.scheduled\_departure <> f.actual\_departure

**order** **by** f.actual\_departure - f.scheduled\_departure **desc**

**limit** 10

Из таблицы flights выбираем рейсы по условию, что планируемое дата/время вылета не совпадает с фактическим датой/временем вылета, сортируем по разнице между этими значениями в обратном порядке и выводим 10 первых записей. Для удобства восприятия

выводим номер рейса, дату вылета и задержку вылета в удобочитаемом виде.

1. Были ли брони, по которым не были получены посадочные талоны?

**select** b.book\_ref "Номер бронирования, по которому не получен посадочный талон"

**from** bookings.bookings b

**join** bookings.tickets t **on** t.book\_ref = b.book\_ref

**left** **join** bookings.boarding\_passes bp **on** bp.ticket\_no = t.ticket\_no

**where** boarding\_no **is** **null**

Объединяем таблицы bookings и tickets по равенству полей book\_ref в обоих таблицах, затем присоединяем справа таблицу boarding\_passes по равенству полей ticket\_no, если не находится соответствия по полям ticket\_no, то для этих строк в присоединяемой таблице проставляется NULL. В результат запроса выводим номер бронирования, для которого номер посадочного талона содержит NULL.

1. 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.

**with** cte1 **as** (

**select** s.aircraft\_code ,**count**(s.seat\_no) ColFull

**from** bookings.seats s

**group** **by** s.aircraft\_code),

cte2 **as**

(**select** bp.flight\_id, **count**(bp.seat\_no) ColFact

**from** bookings.boarding\_passes bp

**group** **by** 1

)

**select** f.flight\_no "Номер рейса", f.actual\_departure::**date** "Дата вылета",

f.departure\_airport "Аэропорт вылета", cte1.ColFull - cte2.ColFact "Количество свободных мест",

**round**(((cte1.ColFull - cte2.ColFact)::**numeric** / cte1.ColFull)\*100, 2) "Процент свободных мест",

**sum**(cte2.ColFact) **over** (**partition** **by** f.departure\_airport, f.actual\_departure::**date** **order** **by** f.actual\_departure) "Вылетело пассажиров"

**from** bookings.flights f, cte1, cte2

**where** cte1.aircraft\_code = f.aircraft\_code **and** f.flight\_id = cte2.flight\_id

**order** **by** 2,3,6

В cte1 с помощью агрегатной функции вычисляем количество мест для каждой модели самолета, в cte2 с помощью агрегатной функции вычисляем количество получивших посадочные талоны пассажиров. Затем в основном запросе, обращаясь к cte1 и cte2, с помощью агрегатной функции вычисляем процент заполненности каждого рейса. Далее, используя агрегатную функцию внутри оконной, считаем нарастающим итогом количество вылетевших пассажиров для каждого аэропорта для каждого дня. Для удобства восприятия в результат запроса также выводим дату вылета и сортируем все данные по этой дате, затем по коду аэропорта, затем по количеству вылетевших пассажиров.

1. Найдите процентное соотношение перелетов по типам самолетов от общего количества.

**select** p.aircraft\_code, **round**((p.Col/p.ColFull)\*100,2) "Процент от общего количества перелетов", p.ColFull "Общее количество перелетов"

**from** (

**select** **distinct** f.aircraft\_code, **count**(\*) **over** (**partition** **by** f.aircraft\_code)::**numeric** Col, **count**(\*) **over** ()::**numeric** ColFull

**from** flights f) **as** p

В подзапросе из таблицы flights берем коды самолетов, с помощью оконных функций вычисляем общее количество перелетов и количество перелетов, приходящееся на каждый тип самолета, приводим их к типу данных numeric (для возможности дальнейшей работы с числами с дробной частью), и выбираем только уникальные значения. Затем, в основном запросе, для каждого типа самолета, выбираемого из подзапроса, вычисляем процент перелетов относительно общего числа перелетов и округляем получившееся значение до сотых.

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

**with** cte **as** (

**select** f.flight\_id, **case** **when** tf.fare\_conditions = 'Business' **then** **min**(tf.amount) **end** AmountB

**from** ticket\_flights tf

**join** flights f **on** f.flight\_id = tf.flight\_id

**group** **by** 1, tf.fare\_conditions

**order** **by** 1),

cte2 **as** (

**select** cte.flight\_id , cte.AmountB, **case** **when** tf.fare\_conditions = 'Economy' **then** **max**(tf.amount) **end** AmountE

**from** cte

**left** **join** ticket\_flights tf **on** tf.flight\_id = cte.flight\_id **and** cte.AmountB **is** **not** **null**

**where** cte.AmountB **is** **not** **null**

**group** **by** 1,2, tf.fare\_conditions

)

**select** fv.departure\_city "Город вылета", fv.arrival\_city "Город прибытия",

cte2.AmountB "Стоимость перелета Бизнес", cte2.AmountE "Стоимость перелета Эконом"

**from** cte2

**join** flights\_v fv **on** fv.flight\_id = cte2.flight\_id

**where** cte2.AmountB **is** **not** **null** **and** cte2.AmountE **is** **not** **null**

**group** **by** 1,2,3,4

**having** cte2.AmountB < cte2.AmountE

В первом cte присоединяем по равенству поля flight\_id таблицу flights к таблице ticket\_flights, отправляя в результат flight\_id и минимальную стоимость перелета на этом рейсе, при условии, что категория обслуживания = «Business», сгруппированные по идентификатору рейса и категории обслуживания. Во втором cte присоединяем по равенству поля flight\_id и существованию данных в поле таблицу ticket\_flights, отправляя в результат flight\_id, минимальную стоимость перелета на этом рейсе, при условии, что категория обслуживания = «Business», максимальную стоимость перелета на этом рейсе, при условии, что категория обслуживания = «Economy», сгруппировав по идентификатору рейса, стоимости перелета в категории «Business» и категория обслуживания.

В основном запросе к результату второго cte присоединяем представление flights\_v по равенству поля flight\_id, откуда берем названия городов вылета и прибытия и выводим результат, по условию, что в полях стоимости перелета по категориям обслуживания «Business» и «Economy» есть значения, группируя по всем полям, чтобы исключить повторяющиеся строки, и ограничивая условием, что стоимость перелета в категории «Business», меньше, чем в категории «Economy».

1. Между какими городами нет прямых рейсов?

Создание представления:

**create** **materialized** **view** flights\_v2 **as**

**select** a.city "DepCity", a2.city "ArrCity"

**from** flights f

**join** airports a **on** a.airport\_code = f.departure\_airport

**join** airports a2 **on** a2.airport\_code = f.arrival\_airport

**group** **by** 1,2

**with** **data**;

Представление создаем над таблицей flights, помещая туда по равенству поля airport\_code названия городов вылета и прибытия, группируя по ним получившуюся выборку. В рамках текущей задачи представление делаем материализованным, с заполнением данными при создании, для увеличения скорости отработки запроса.

Запрос:

**select** **distinct** a.city "DepCity", a2.city "noArrCity"

**from** airports a, airports a2

**where** a.city != a2.city

**group** **by** 1,2

**except**

**select** **distinct** fv."DepCity" , fv2."ArrCity"

**from** flights\_v2 fv, flights\_v2 fv2

**where** fv2."DepCity" = fv."DepCity" **and** fv."ArrCity" = fv2."ArrCity"

**order** **by** 1,2

В первой части запроса, используя декартово произведение, получаем все возможные пары городов вылета и прибытия, исключив случаи, когда город вылета совпадает с городом прибытия. Во второй части запроса из созданного ранее представления, используя декартово произведение, получаем пары городов вылета и прибытия, которые имели место в рамках существующих рейсов. Вычитая из первой части запроса вторую с помощью Except, получаем пары городов, между которыми нет прямых рейсов.

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

**select** a.city "Город вылета", a2.city "Город прибытия", a3.model "Модель самолета", a3."range" "Дальность полета самолета", **Round**(**acos**(**sind**(a.latitude)\***sind**(a2.latitude) + **cosd**(a.latitude)\***cosd**(a2.latitude)\***cosd**(a.longitude - a2.longitude))\*6371) "Расстояние между городами",

**case** **when** (a3."range" - **Round**(**acos**(**sind**(a.latitude)\***sind**(a2.latitude) + **cosd**(a.latitude)\***cosd**(a2.latitude)\***cosd**(a.longitude - a2.longitude))\*6371)) > 100

**then** 'Да'

**else** 'Нет' **end** "Применимость самолета на маршруте"

**from** flights f

**join** airports a **on** a.airport\_code = f.departure\_airport

**join** airports a2 **on** a2.airport\_code = f.arrival\_airport

**join** aircrafts a3 **on** a3.aircraft\_code = f.aircraft\_code

**group** **by** 1,2,3,4,5,6

**order** **by** 1,2

К таблице flights по два раза присоединяем таблицу airports по равенству полей departure\_airport = airport\_code и arrival\_airport = airport\_code, а также таблицу aircraft по равенству поля aircraft\_code. В итоге получаем таблицу, где для каждого рейса есть город отправления, город прибытия, их координаты, модель самолета и максимальную дальность его полета. В результат выводим город вылета, город прибытия, модель самолета для этого перелета, его максимальную дальность полета, вычисляемое значение расстояния между городами вылета и прибытия, а также вычисляемую колонку, в которой будет отображаться пригодность данной модели самолета для выполнения рейсов между этими городами (если максимальная дальность – расстояние между городами больше 100 км, выводится «Да», иначе «Нет»). Затем группируем результат по всем колонкам, чтобы избежать дублирования строк, и для лучшего восприятия сортируем по городу вылета, а затем по городу прибытия.