Итоговая работа:

Проектная работа по модулю "DWH"

Выполнил:

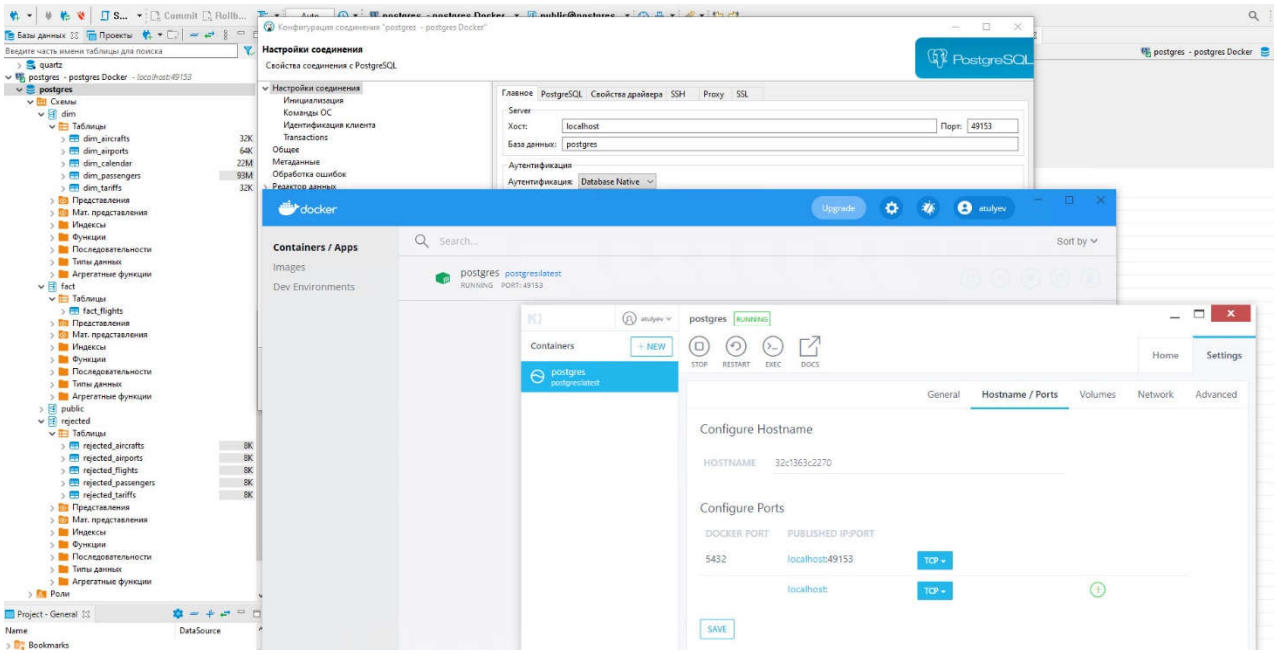
Тульев Александр Сергеевич

Группа: DEG-4

апрель 2021г.

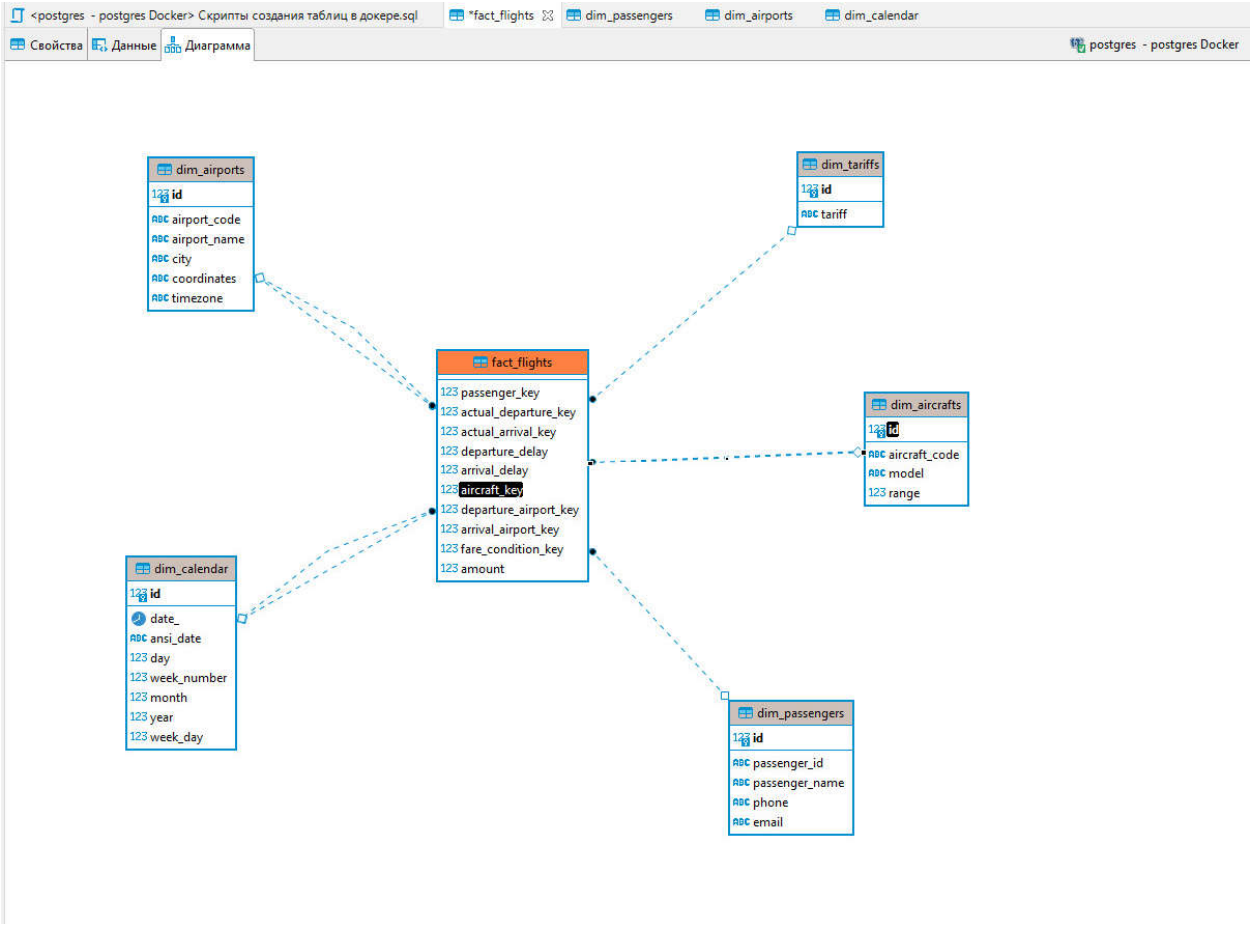
Для данной работы использовал две БД:

1. БД источник: demo в yandex облаке, схема bookings. Некоторые таблицы в БД не соответствовали описанию БД. Использовал те таблицы (представления), которые имеются в БД.
2. БД (фактов и измерений). Создана в Dockere (использовал пустой контейнер postgres).



3. Создал 3 схемы (разнес таблицы по схемам):
    a. dim – для таблиц измерений;
    b. fact – для таблиц фактов;
    c. rejected – для таблиц с некачественными строками. Такие строки можно было выгружать и в Excel, например, но я решил грузить их в отдельные таблицы в БД.

Прикладываю ER-диаграмму созданной БД хранилища:



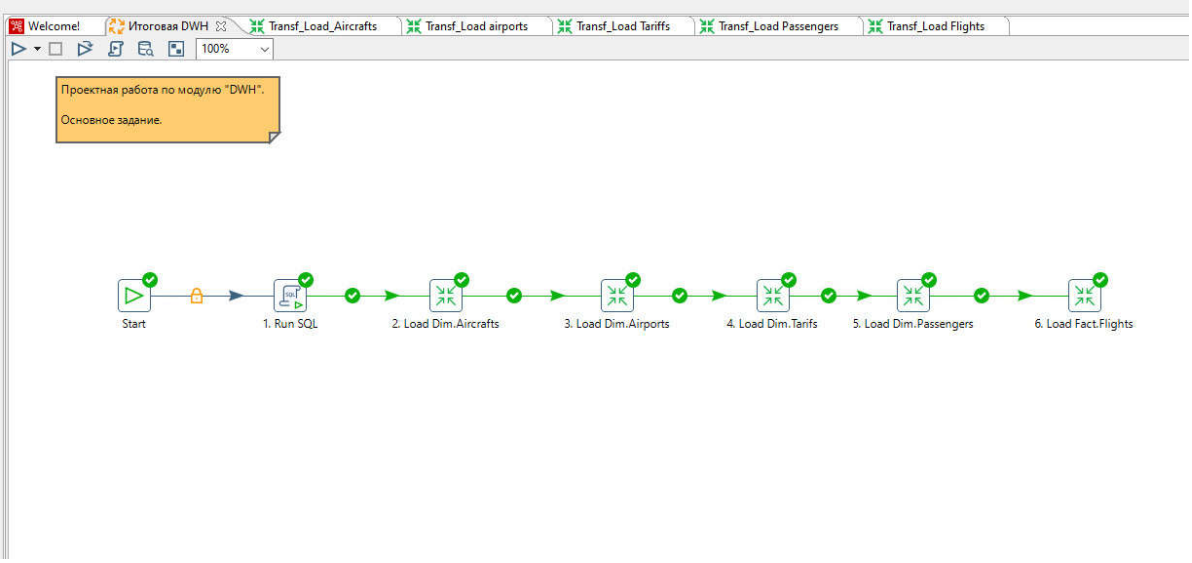
Все вложенные screenshots присутствуют в оригинале на github в отдельной папке.

Ссылка: https://github.com/twins-a/study_netology/tree/main/DWH/final%20work%20DWH

Там же на github в отдельной папке расположил трансформации.

Наполнение базы данных при помощи ETL построил по следующему варианту:

В основу выбрал задание (Job).



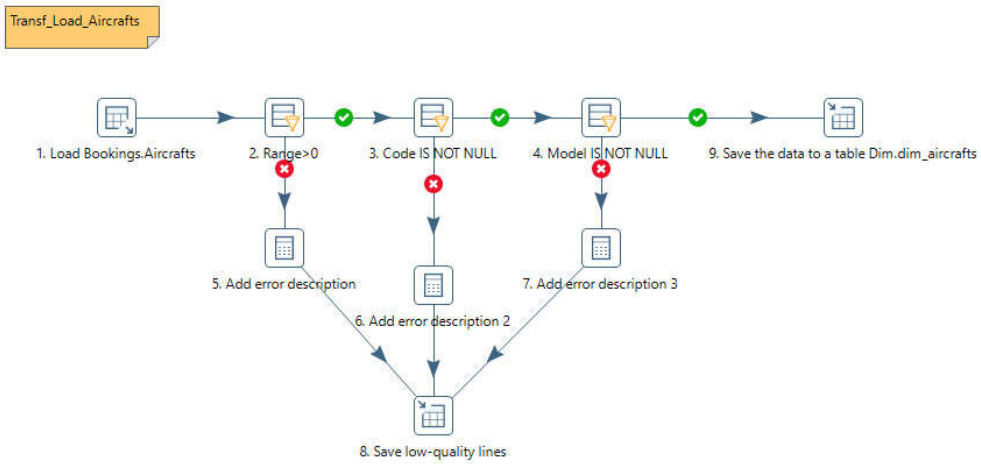
Последовательно выполняю шаги:

1. Шаг 1. SQL-скрипт создания таблиц из внешнего файла. То есть каждый раз при запуске задания удаляю и по новой создаю все таблицы и отношения. Считаю что каждый раз загружаю полный объем информации (нужный период) и работаю только с ним. Времени это много не съедает.
   Таблица фактов имеет внешние ключи на таблицы измерений.
   Таблицы измерений имеют первичный ключ тип serial, то есть автоинкремент.
   Отдельно нужно остановиться на создании таблицы измерений dim.dim_Calendar. Создаю эту таблицу SQL-скриптом.
   Даты генерирую в одном поле со временем с интервалом 1 минута. Правильнее было бы разнести дату и время в отдельные поля. Но на данных объемах сделал в одной колонке и дату и время.
2. Шаги со 2 по 5. Загрузка данных в таблицы измерения.
3. Шаг 6. Используя уже загруженные таблицы измерения, загружаю данные в таблицу фактов.

Прекрасно понимаю, что со 2 по 5 шаги можно было запустить параллельно, но сделал так.

SQL-скрипт публикую отдельным файлом, в описание не стал загружать.

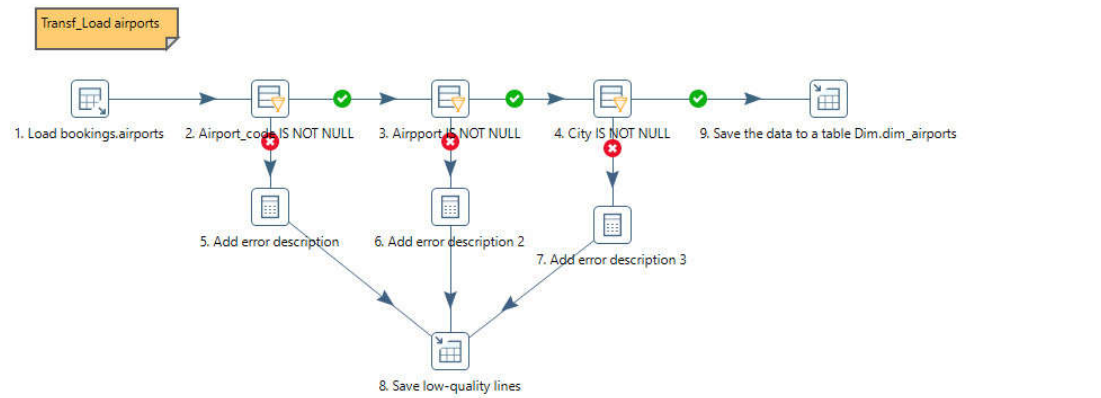
Теперь посмотрим на трансформации (шаги 2-5 из задания).



1. Шаг 1. Читаю таблицу bookings.aircrafts;
2. Шаги 2-4 фильтрую для проверки качества данных на разные ошибки. Проверяю:
   а) Дальность полета > 0;
   б) Code aircraft не должен быть Null;
   в) Model не должна быть Null.
   Строки с ошибками уходят на Шаги 5-7, где добавляется колонка с описанием ошибки и на шаге 8 записываем ошибочные строки в rejected таблицу rejected. rejected_aircrafts.
3. На шаге 9 сохраняем нормальные данные, поступившие после фильтров, в таблицу dim. dim_aircrafts. Truncate table не использую так как таблицы при запуске задания каждый раз пересоздаются.

Аналогично работают следующие трансформации (шаги 3,4,5 задания).
Посмотрим на них.

Создание таблицы измерений dim.dim_airports (шаг 3 задания).



Все идентично работе предыдущего шага, только использую другие условия в фильтрах, ну и конечно входные и выходные таблицы.
Фильтры:
а) Airport_code IS NOT NULL;
б) Airport IS NOT NULL;
в) City IS NOT NULL.

Создание таблицы измерений dim.dim_tariffs (шаг 4 задания).

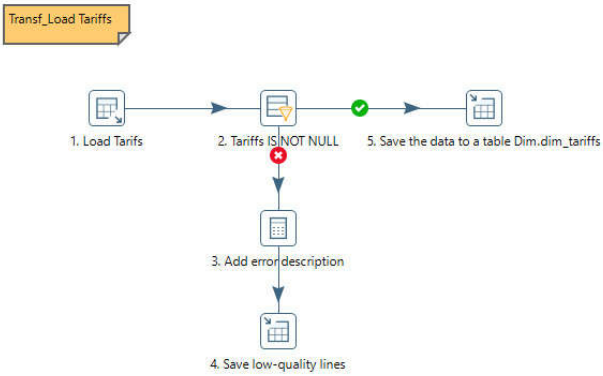
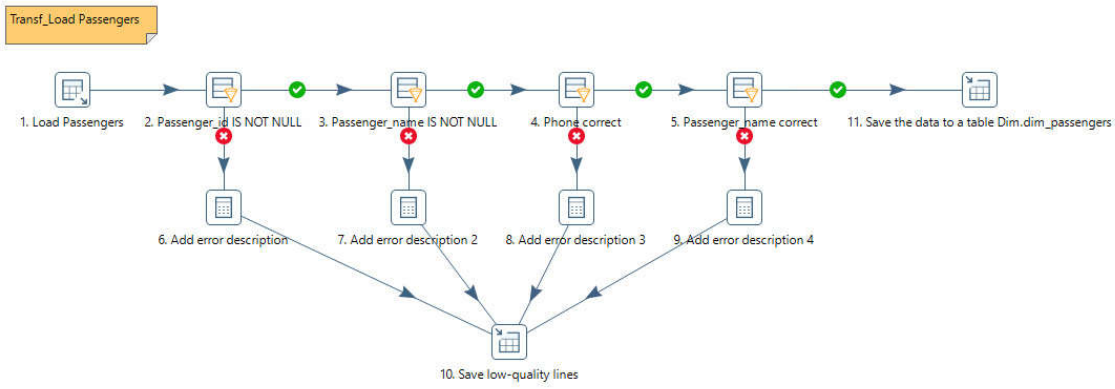


Схема работы та же самая. Из таблицы загружается всего одно поле fare_conditions. В запросе использую Distinct. Здесь смог придумать только одно правило для проверки качества:
А) Tariffs IS NOT NULL.

Создание таблицы измерений dim. dim_passengers (шаг 5 задания).



Общая схема работы та же самая что и на предыдущих шагах.

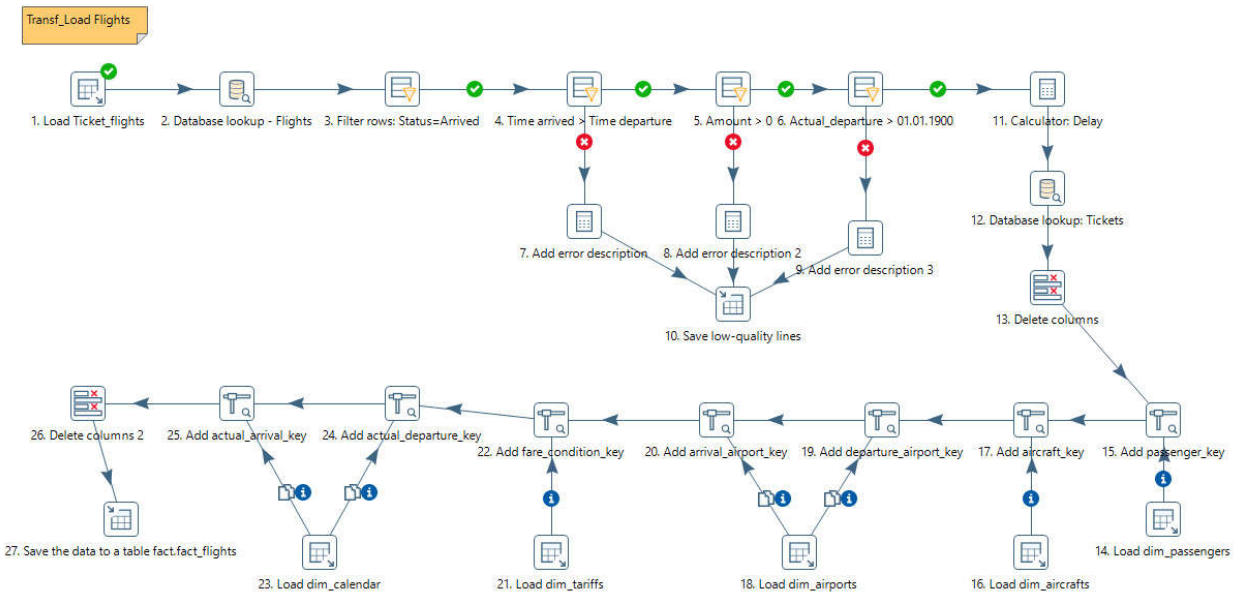
- Здесь при чтении таблицы контактные данные разбиваю на два отдельных поля phone и email.
- Данных в этой таблице больше, чем в других, поэтому commit size я увеличил до 50000. Пробовал с другими значениями, но то которое установил показалось оптимальным по скорости работы.
  Фильтры (проверка качества):
  а) Passenger_id IS NOT NULL;

б) Passenger_name IS NOT NULL;
в) Phone correct;
г) Passenger_name correct.

- На шагах 4 и 5 текущей трансформации (в описании п.п. в и г) использую регулярные выражения.
- Email на Null проверять не стал, так как у человека действительно может не быть электронной почты.

После этой трансформации у меня готовы и заполнены данными все таблицы измерения.

Ну и я подобрался к основной трансформации: наполнение таблицы фактов fact .fact_flights.



В этой трансформации долго выбирал что использовать на данной базе данных DataBase lookup или Stream lookup. В итоге использовал и тот и тот элементы. По производительности особо разницы не заметил. Можно было еще использовать Join, но с этим уже не стал играть.

И так логика работы основной трансформации (Шаг 6 задания):

1. Шаг 1. Читаю данные с таблицы bookings.ticket_flights.
2. Шаг 2. Читаю данные с таблицы bookings.flights и дообогащаю данные 1 шага. В роли ключа использую flight_id.
3. Шаг 3. Фильтрую строки со статусом arrived. То есть на следующий шаг передаются только совершенные перелеты. Отмененные, выполняющиеся, доступные для бронирования, доступные для регистрации и задержанные рейсы не рассматриваю.
4. Шаги 4 – 6. Фильтрация строк (проверка качества). Для примера использовал три шага для проверки качества:
   а) актуальное время прилета не может быть раньше актуального времени вылета;
   б) Стоимость перелета не может быть < или = 0;
   в) актуальное время вылета должно быть больше 01.01.1900
5. Шаги 7-9. Добавляю колонку с описанием ошибки для дальнейшего анализа.
6. Шаг 10. Сохраняю ошибочные строки в таблицу rejected.rejected_flights.
7. Шаг 11. Считаю задержку вылета и прилета в секундах. Использую формулу: Date A – Date B (seconds). То есть после этого шага появляется два новых расчетных поля.
8. Шаг 12. Читаю данные с таблицы bookings.tickets и добавляю информацию о пассажирах к предыдущим данным.
9. Шаг 13. Удаляю 5 уже не нужных столбцов из набора данных. Меньше места в памяти должно занимать и работать по шустрее после этого.

После шага 13 у меня готовы все данные для связи таблицы фактов и таблицы измерений. Остается получить ключи из таблиц измерений, удалить лишние столбцы и все данные записать в таблицу фактов.

Продолжаю:

10. Шаги 14-15. Читаю таблицу измерений dim.dim_passengers и подтягиваю к данным ключ из этой таблицы passenger_key.
11. Шаги 16-17. Читаю таблицу измерений dim.dim_aircrafts и подтягиваю к данным ключ из этой таблицы aircraft_key.
12. Шаги 18-20. Читаю таблицу измерений dim.dim_airports и подтягиваю к данным ключи из этой таблицы departure_airport_key и arrival_airport_key.
13. Шаги 21-2. Читаю таблицу измерений dim.dim_tariffs и подтягиваю к данным ключ из этой таблицы fare_condition_key.
14. Шаги 23-25. Читаю таблицу измерений dim.dim_calendar и подтягиваю к данным ключи из этой таблицы actual_departure_key и actual_arrival_key.
15. Шаг 26. Удаляю 7 уже не нужных столбцов из набора данных, которые были необходимы для связи таблиц.
16. Шаг 27. Записываю данные в таблицу фактов fact.fact_flights. Commit size так же установлен в значение 50000.

Ну вот и все наполнил базу данными.

Понимаю, что можно было уменьшить количество шагов, но решил для себя все сделать более подробно.

В итоге я создаю 5 таблиц измерений, 1 таблица фактов и 5 таблиц отклонений. На таблицу dim.dim_Calendar таблица отклонений не нужна.

Общее время отработки задания примерно 3 минуты на моем ПК. Считаю это нормальным временем выполнения.