**SQL – запрос.**

В локальном докере имеется база Postgres. В sql-скрипте реализовано создание новой схемы dwh\_final.

Далее идет создание таблиц справочников Dim\_passenger, dim\_airports, dim\_aircrafts, dim\_tariffs, dim\_calendar.  В каждой таблице наложены ограничения в виде основного ключа и отсутствие нулевых значение.

Затем описано создание таблицы фактов Fact\_flights. Основной ключ таблицы – это flight\_id, также заданы ссылки на таблицы справочники. Для таблицы аэропортов задано две ссылки из таблицы фактов (аэропорты вылета и прилета по отдельности).

Затем описано создание таблиц для записи ошибок из ETL-процесса. Для этих таблиц заданы только ограничения по длине строковых значений.

**ETL-процесс.**

В etl-процессе не удалось реализовать наполнение всех таблиц одной трансформацией.

Поэтому созданы 4-х отдельных процесса, причем процесс наполнения таблицы фактов необходимо выполнять последним, т.к. там реализована ссылка на таблицы справочники.

Для процесса созданы два подключения к базам данных – одно входящие данные, второе – запись в другу базу данных.



Наполнение справочников Dim\_Airports и Dim\_Aircrafts происходить одинаково. Сначала происходит считывание из базы источника без указания типа данных и сразу происходит запись, причем ограничения целевой базы данных дают две ошибки – отсутствие нулевых значений и дублированных значений. Соответственно если такие данные появляются, то они идут в поток обработки ошибок, где в два шага – удаление пробелов вокруг описания и потом удаления переноса строк записывается в целевую таблицу с ошибками для каждого потока отдельно.

Для наполнения справочников dim\_passenger и dim\_tariffs реализованы с такими же проверками как и для первых двух, но создал два варианта потока, в одном – происходит сортировка и удаление дублирующихся строк и потом уже запись в базу, но я оставил аналогично первым двум – запись сразу в целевую базу с реализованными проверками и дублирующиеся данные записываются в таблицу с ошибками для соответствующей таблицы. Вариант с удалением дубликатов отдельными шагами работает на много быстрее, но эти ошибки будет не видно.

Запись для таблицы-справочника dim\_calendar делается из двух поток даты прилета и вылета с помощью запроса, выбирающего отдельно год, месяц, день, час и записывается в таблицу, на таблицу наложены ограничения основного ключа и ненулевых значение, это позволяет отсечь дубликаты в таблицу ошибок. Набор из двух таблиц сделан, т.к. дата вылета и прилета могут отличаться и соответственно на самом последнем рейсе возможно одна дата может преобразоваться в две.

Запись данных в таблицу фактов начинается со считывания данных sql-запросом. В запросе для таблицы flights реализован расчет задержки и перевод в секунды с помощью функции Extract. Также в sql-запросах для проверяемых данных не задается тип данных, в запросе указан фильтр по времени вылета (не совершенные рейсы нам не нужны), так указано в задание.

Потом происходит объединение потоков по полю flight\_id. Далее создан шаг конвертации типа данных, на этом шаге если будут ошибки в датах они будут перенаправлены в поток для ошибок и с помощью уже, описанных выше шагов, будут обработаны и записаны в таблицу ошибок для fact\_flights. В поток основных данных на следующем шаге будет добавлена текущая дата с последующим вычетом от нее актуальной даты прилета (можно также реализовать для даты вылета). Затем происходит поиск не цифровых символов (в данном случае это будет «-») и замена их на # для четырех полей: amount, flight\_id, ticket\_no и разницы между датами. Затем происходить обратная конвертация в цифровой тип данных предыдущей полей и при выявлении ошибки опять уходят в поток с записью ошибок. Далее происходит запись в целевую таблицу и также если будут нулевые или дублирующиеся данные они будут направлены в таблицу с ошибками.

Запись таблиц справочников:



Запись таблицы фактов:

