**Проектная работа по модулю «DWH»**

**Пояснительная записка**

# Задача

В рамках поставленной учебной задачи требовалось сымитировать загрузку raw данных в Stage-слой КХД.

Источником raw данных является БД avia, расположенная в облачном хранилище Yandex.

Параметры подключения:

* Host - 84.201.153.170
* порт – 19001
* база данных – demo
* пользователь – netology
* пароль - NetoSQL2019

# Результат

Для целей имитации Stage-слоя КХД на локальном сервере была создана БД DWH\_FLIGHTS. Загрузка данных, как измерений, так и таблицы фактов, осуществлялась в схему solution.

Согласно условий задания в БД DWH\_FLIGHTS были созданы 4 таблицы измерений, 1 таблица фактов (**fact\_flights**), 1 вспомогательная таблица-календарь (**dim\_calendar**) и 4 rejected-таблицы, повторяющих по своей структуре соответствующие таблицы измерений и фактов, предназначенные для анализа отклонений данных.

Таблицы измерений:

|  |  |
| --- | --- |
| * **dim\_passengers** | Справочник пассажиров |
| * **dim\_aircrafts** | Справочник воздушных судов |
| * **dim\_airports** | Справочник аэропортов |
| * **dim\_tariff** | Справочник тарифов |

Были созданы ETL процессы для загрузки данных из БД avia в БД DWH\_FLIGHTS, а также реализован ряд логических проверок данных.

# Описание созданных таблиц БД DWH\_FLIGHTS

## **dim\_ passengers**

Справочник пассажиров.

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип** | **Описание** |
| pass\_id | serial | Составной суррогатный первичный ключ |
| document\_id | varchar(30) | Номер паспорта пассажира |
| passenger\_name | text | ФИО пассажира |
| date\_start | date | Техническое поле; дата внесения записи в таблицу dim\_passengers |
| date\_end | date | Техническое поле; показывает срок актуальности записи |
| num\_version | integer | Составной суррогатный первичный ключ  Техническое поле; номер версии кортежа в рассматриваемом отношении. |

Наличие атрибутов *date\_start, date\_end, num\_version* обусловлено тем, что отношение **dim\_passengers** является SCD 2-го типа.

## **dim\_aircrafts**

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

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип** | **Описание** |
| aircraft\_id | int2 | Составной суррогатный первичный ключ |
| aircraft\_code | char(3) | Код самолета, IATA |
| model | varchar(30) | Модель самолета |
| range | integer | Максимальная дальность полета, км |
| date\_start | date | Техническое поле; дата внесения записи в таблицу dim\_aircrafts |
| date\_end | date | Техническое поле; показывает срок актуальности записи |
| num\_version | integer | Составной суррогатный первичный ключ  Техническое поле; номер версии кортежа в рассматриваемом отношении. |

Наличие атрибутов *date\_start, date\_end, num\_version* обусловлено тем, что отношение **dim\_aircrafts** является SCD 2-го типа.

## **dim\_airports**

Справочник аэропортов. Аэропорт идентифицируется трехбуквенным кодом (*airport\_code*) и имеет свое имя (*airport\_name*). Для города не предусмотрено отдельной сущности, но название (*city*) указывается и может служить для того, чтобы определить аэропорты одного города. Также указывается часовой пояс (*timezone*).

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип** | **Описание** |
| airport\_id | int2 | Составной суррогатный первичный ключ |
| airport\_code | char(3) | Код аэропорта |
| airport\_name | varchar(30) | Название аэропорта |
| city | varchar(30) | Город |
| timezone | varchar(30) | Временная зона аэропорта |
| date\_start | date | Техническое поле; дата внесения записи в таблицу dim\_airports |
| date\_end | date | Техническое поле; показывает срок актуальности записи |
| num\_version | integer | Составной суррогатный первичный ключ  Техническое поле; номер версии кортежа в рассматриваемом отношении. |

Наличие атрибутов *date\_start, date\_end, num\_version* обусловлено тем, что отношение **dim\_airports** является SCD 2-го типа.

## **dim\_ tariff**

Справочник тарифов обслуживания.

|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип** | **Описание** |
| tariff\_id | serial | Составной суррогатный первичный ключ |
| fare\_conditions | varchar(10) | Класс обслуживания |
| date\_start | date | Техническое поле; дата внесения записи в таблицу dim\_ passengers |
| date\_end | date | Техническое поле; показывает срок актуальности записи |
| num\_version | integer | Составной суррогатный первичный ключ  Техническое поле; номер версии кортежа в рассматриваемом отношении. |

Наличие атрибутов *date\_start, date\_end, num\_version* обусловлено тем, что отношение **dim\_airports** является SCD 2-го типа.

## **fact\_flights**

Рейс всегда соединяет две точки — аэропорты вылета и прибытия. Такое понятие, как «рейс с пересадками» отсутствует: если из одного аэропорта до другого нет прямого рейса, в билет просто включаются несколько необходимых рейсов.

У каждого рейса есть запланированные дата и время вылета и прибытия. Реальные время вылета (actual\_departure) и прибытия (actual\_arrival) могут отличаться: обычно не сильно, но иногда и на несколько часов, если рейс задержан.

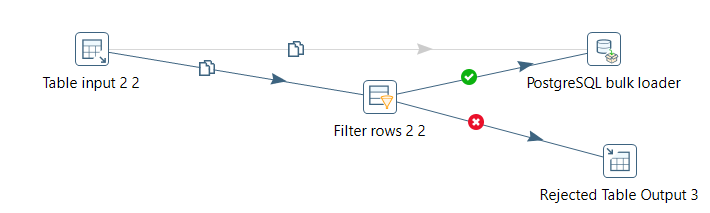
|  |  |  |
| --- | --- | --- |
| **Столбец** | **Тип** | **Описание** |
| id | serial4 | Идентификатор рейса |
| passenger\_id | int4 | Идентификатор пассажира |
| act\_departure | timestamp | Фактическое время вылета |
| act\_arrival | timestamp | Фактическое время прилёта |
| dep\_delay | varchar(20) | Фактическая задержка вылета |
| arr\_delay | varchar(20) | Фактическая задержка прилета |
| departure\_airport\_id | int2 | Идентификатор аэропорта отправления |
| arrival\_airport\_id | int2 | Идентификатор аэропорта прибытия |
| aircraft\_id | int2 | Идентификатор воздушного судна |
| tarif\_id | int2 | Идентификатор класса обслуживания |
| amount | numeric(10,2) | Стоимость билета |

На Stage-слое отключены внешние ключи, однако для таблицы фактов на уровне DDS должны быть установлены внешние ключи для атрибутов *passenger\_id, departure\_airport\_id, arrival\_airport\_id, aircraft\_id* и *tarif\_id* ссоответствующими таблицами измерений.

# Описание ETL процессов

## **Загрузка данных в dim\_passenger**

Трансформация выглядит следующим образом:



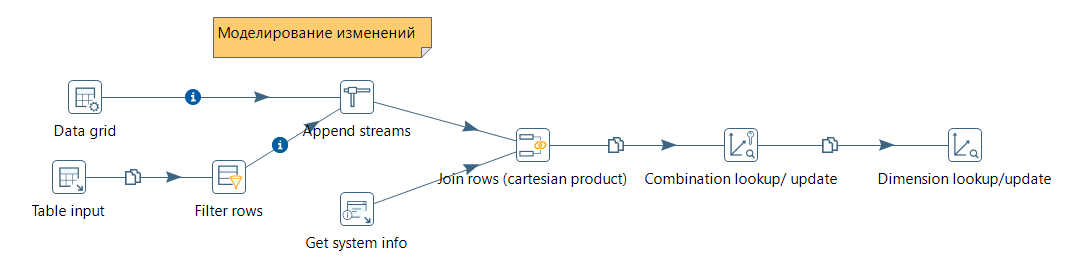
* На шаге «Table Input 2 2» данные копируются из БД Avia (таблица **dim.passenger**)
* на шаге «Filter Rows» производится проверка качества данных (см. раздел «Описание логических проверок»)
* данные, не прошедшие проверку, отправляются в таблицу **rejected\_dim\_passengers** на шаге «Rejected Table Output 3»
* данные, прошедшие проверку, записываются в целевую таблицу **dim\_passengers** на шаге «PostgreSQL bulk loader»

## **Обновление данных в dim\_passenger**

В качестве личной инициативы была смоделирована ситуация обновления данных в источнике и построена трансформация, выполняющая UPDATE отношения dim\_passengers.

Note: При реализации данной задачи столкнулся с рядом вопросов, на которые прошу преподавателей курса дать пояснения, буду очень признателен за это! Более подробно описал вопросы в отдельном архиве «Вопрос по UPDATE dim\_passengers»

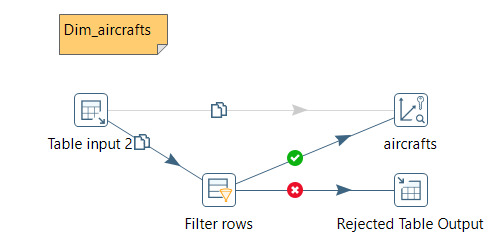
Трансформация выглядит следующим образом:



* В Data Grid внесена 1 новая запись, которой нет в источнике, и 1 изменение по существующей записи
* На шаге Append Streams в поток добавлены новые данные и убраны старые
* На шаге «Get system info» добавляется текущая дата
* На шаге «» происходит формирование суррогатного ключа pass\_id и добавление новых строк в **dim\_passengers** с новым pass\_id
* На последнем шаге новые строки отношения dim\_passengers обогащаются данными неключевых атрибутов

## **Загрузка данных в dim\_aircrafts**

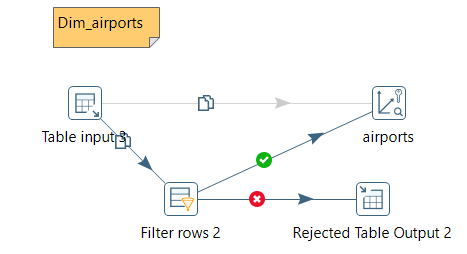
Трансформация выглядит следующим образом:



* На шаге «Table Input 2» данные копируются из БД Avia (таблица **dim.aircrafts**)
* на шаге «Filter Rows» производится проверка качества данных (см. раздел «Описание логических проверок»)
* данные, не прошедшие проверку, отправляются в таблицу **rejected\_dim\_aircrafts** на шаге «Rejected Table Output»
* данные, прошедшие проверку, записываются в целевую таблицу **dim\_aircrafts** на шаге «Combination lookup/update» (aircrafts)

## **Загрузка данных в dim\_airports**

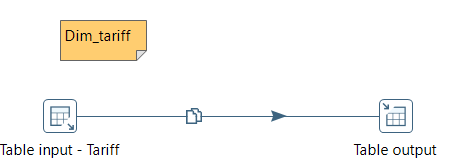
Трансформация выглядит следующим образом:



* На шаге «Table Input 3» данные копируются из БД Avia (таблица **dim.airports**)
* на шаге «Filter Rows 2» производится проверка качества данных (см. раздел «Описание логических проверок»)
* данные, не прошедшие проверку, отправляются в таблицу **rejected\_dim\_ airports** на шаге «Rejected Table Output 2»
* данные, прошедшие проверку, записываются в целевую таблицу **dim\_ airports** на шаге «Combination lookup/update» (airports)

## **Загрузка данных в dim\_tariff**

Трансформация выглядит следующим образом:



* На шаге «Table Input - Tariff» данные копируются из БД Avia (таблица **dim.tariff**)
* на шаге «Table Output» данные записываются в целевую таблицу **dim\_tariff**
* Ввиду простоты справочника (всего 3 значения) проверки не были добавлены для этой трансформации

## **Загрузка данных в fact\_flights**

Общий вид трансформации можно посмотреть в приложенном скриншоте «ETL Fact Flights» (трансформация большая, в MS Word будет нечитаема).

Данная трансформация состоит из 6-ти фрагментов:

1. Загрузка данных из первоисточника «как есть»
2. Добавление суррогатного ключа для подключения таблицы измерений dim\_airports
3. Добавление суррогатного ключа для подключения таблицы измерений dim\_aircrafts
4. Добавление суррогатного ключа для подключения таблицы измерений dim\_tariff
5. Добавление суррогатного ключа для подключения таблицы измерений dim\_passengers
6. Проверка данных

Рассмотрим каждый из фрагментов более подробно:

* Фрагмент 1 - Загрузка данных из первоисточника «как есть»

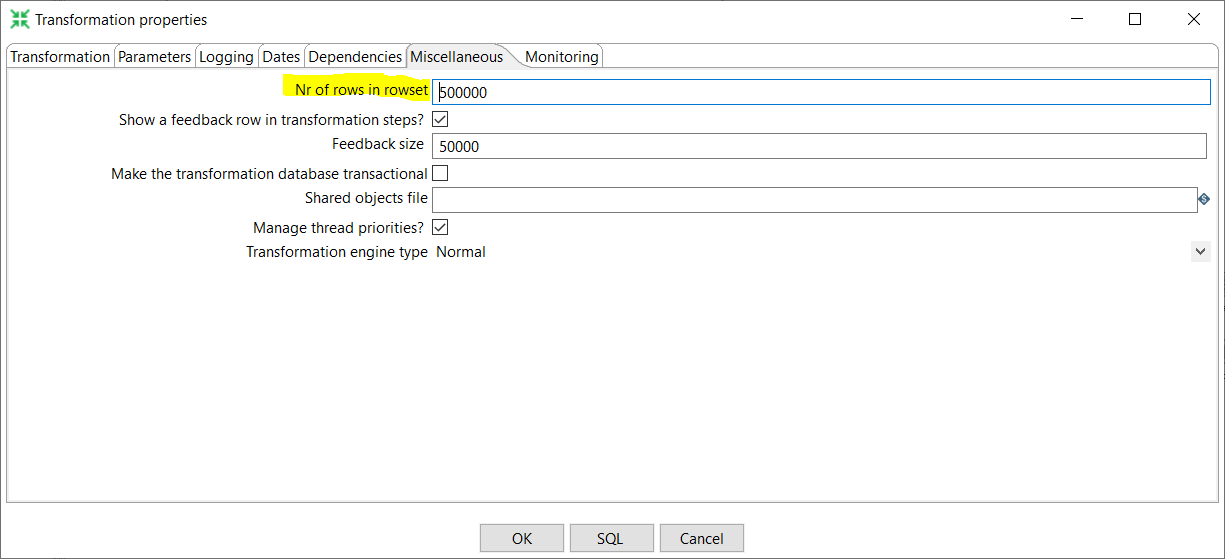
Используется шаг трансформации «Table Input», загрузка из БД Avia (**схема dwh\_2, таблица fact\_flights**).

Для ограничения выборки и сокращения времени отработки учебной трансформации добавлено условие

WHERE date\_part('month',date\_dep) IN (5)

Даже с учетом этого ограничения поток состоит из 456 тысяч строк.

При отработке трансформации неизбежно возникали deadlocks, устранить которые удалось только с помощью увеличения значения «Nr of rows in rowset» до 500 000 в параметрах трансформации:



Также были опробованы:

* копирование шага «Table input - Origin 1» на 4 копии и уход от распараллеливания потоков
* добавление в трансформацию шагов «Block this step until steps finish» с целью управления временем распределения данных в потоке

Однако, как было сказано выше, единственно эффективным оказалось увеличения «Nr of rows in rowset»

Note – есть ли другие способы бороться с deadlock? Дайте пожалуйста ссылки на литературу / источники в сети ?

* Фрагмент 2 - Добавление суррогатного ключа для dim\_airports
* Используется шаг «Database Lookup» для поиска идентификатора аэропорта (*airport\_id*) по наименованию аэропорта (*airport\_name*) в таблице измерений **dim\_airports**
* Ввиду того, что каждый рейс содержит данные как об аэропорте вылета, так и об аэропорте прилета, поток пришлось разделить на 2, для присвоения *airport\_id* аэропортам вылета и аэропортам прилета по отдельности
* После «Database Lookup» используются стандартные шаги для выравнивания данных в потоках и возможности их объединения на щаге «Stream lookup 2» и последующей фильтрации (удаления ненужных столбцов) с помощью шага «Select Values 5 2»
* Фрагмент 3 - Добавление суррогатного ключа для dim\_aircrafts

Аналогично предыдущему фрагменту с помощью «Database Lookup» осуществляется поиск *aircraft\_id* из таблицы измерений **dim\_aircrafts**. Поиск осуществляется по полю *model*.

* Фрагмент 4 - Добавление суррогатного ключа для dim\_tariff

Здесь с помощью «Stream lookup» осуществляется поиск *tariff\_id* из таблицы измерений **dim\_tariff**. Поиск осуществляется по полю *fare\_condition*.

Использование «Stream lookup» обусловлено тем, что в процессе создания трансформации обновил справочник тарифов в БД, после чего в Pentaho перестали корректно подтягиваться поля на шаге «Database Lookup» (т.е. это своеобразный «костыль»)

Note – как можно исправить данный баг, если перестают корректно подтягиваться нужные поля из таблиц БД ? Пробовал удалять шаг/ перезапускать Pentaho – не помогло.

* Фрагмент 5 - Добавление суррогатного ключа для dim\_passengers

Особенностью данного фрагмента является использование шага «Replace in string» для конвертирования текстового формата атрибута, содержащего номер паспорта, в числовой. Для этого в «Replace in string» убран пробел, далее с помощью «Select values» у атрибута *passenger* изменен тип данных на Integer.

Необходимость данных манипуляций обусловлена тем, что шаг «Database Lookup» не срабатывал, падал с ошибкой из-за попытки сравнить 2 текстовых поля через оператор =

На шаге «Stream lookup 3» происходит объединение идентификаторов пассажиров и идентификаторов аэропортов.

* Фрагмент 6 - Проверка данных

С помощью шага «Data Grid» добавлена текущая дата, на следующем шаге она добавлена к каждой строке потока, после чего на шаге «Filter Rows» реализованы проверки данных (см. раздел «Описание логических проверок»)

• данные, не прошедшие проверку, отправляются в таблицу **rejected\_fact\_flights** на шаге «Table output»

• данные, прошедшие проверку, записываются в целевую таблицу **fact\_flights** на шаге «PostgreSQL bulk loader».

«PostgreSQL bulk loader» используется вместо стандартного «Table Output» как более эффективный по скорости шаг, выполняющий загрузку больших объемов данных в меньшие сроки.

# Описание логических проверок

## **Проверки dim\_passengers**

На шаге «Filter Rows» с помощью регулярного выражения [0-9]{4} [0-9]{6} проверяется соответствие значений атрибута *passenger\_id* общепринятому формату записи номеров паспортов.

## **Проверки dim\_aircrafts**

На шаге «Filter Rows» проверяется:

1. соответствие значений кода самолета (атрибут *aircraft\_code*) общепринятому трехсимвольному формату обозначения воздушных судов с помощью регулярного выражения [A-Z0-9]{3}
2. дальность полета (атрибут *range*) – данное поле не может быть отрицательным. Также установлено логическое ограничение дальности перелета в 30 000 км.

## **Проверки dim\_airports**

На шаге «Filter Rows» с помощью регулярного выражения [A-Z0-9]{3} проверяется соответствие значений атрибута *airport\_code* общепринятому трехсимвольному формату обозначения аэропортов в системе IATA.

## **Проверки fact\_flights**

На шаге «Filter Rows» проверяется:

1. фактическое время прилёта (*act\_arrival*) не может быть больше текущей даты
2. время задержки вылета/прилета (*dep\_delay/ arr\_delay*) не может быть отрицательным
3. сумма (*amount*) не может быть меньше 0