Проектная работа по модулю “DWH”

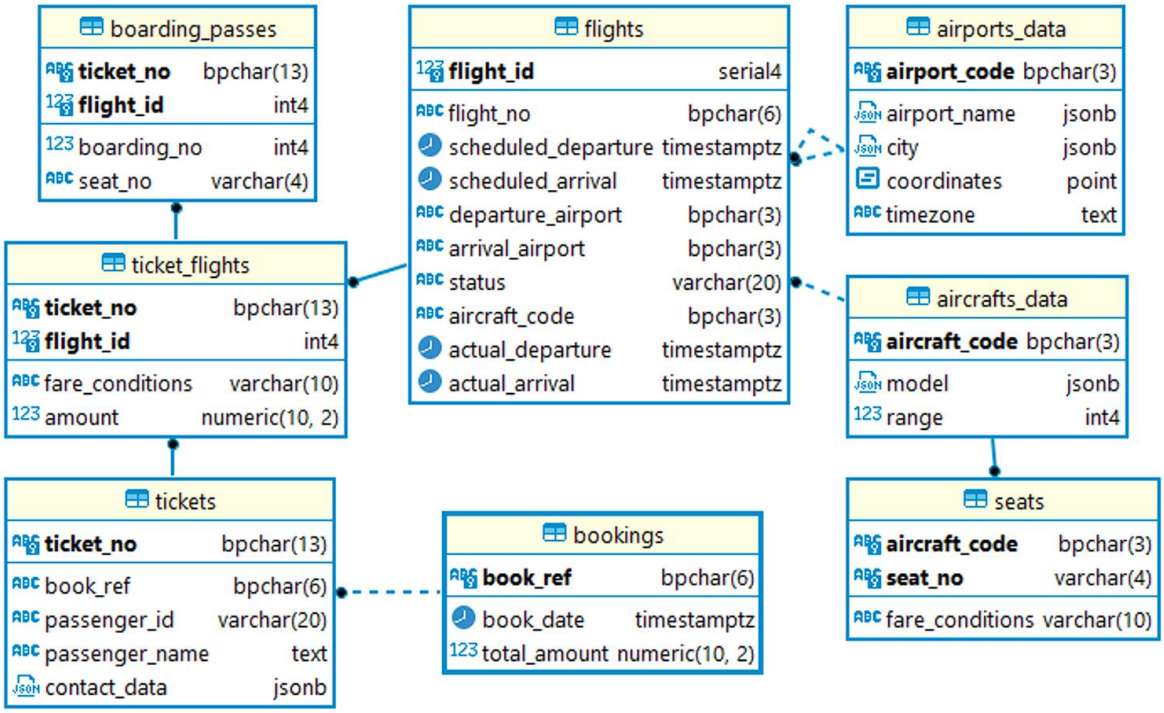
Тимохин Виталий
Ноябрь 2022

1. Источник данных

Источником данных для работы является база данных bookings -«Авиаперевозки».
Описание базы данных: https://edu.postgrespro.ru/bookings.pdf
Подключение к базе данных:

1. Яндекс облако Host - 84.201.153.170
2. Порт – 19001
3. База данных – demo
4. Схема – bookings
5. Пользователь – netology
6. Пароль - <password>

ER-диаграмма согласно подключению



Используемые поля таблиц будут описаны в разделе 4.

2. Экспорт данных БД bookings в файлы CSV.

Экспорт данных БД bookings в файлы CSV можно произвести различными способами.

В данной работе воспользуемся возможностями изученного в рамках курса «Data Warehouse для разработчика баз данных» программного обеспечения Spoon пакета Pentaho data integration.

Для этого создадим трансформацию, в которой для каждой таблицы БД:

1. С помощью элемента Table Input подключаемся к облачной базе данных bookings, считываем данные из таблицы базы данных с помощью указанного в параметрах SQL скрипта «SELECT * FROM bookings.table_name».

2. Связываем элемент Table Input с элементом Text file output.

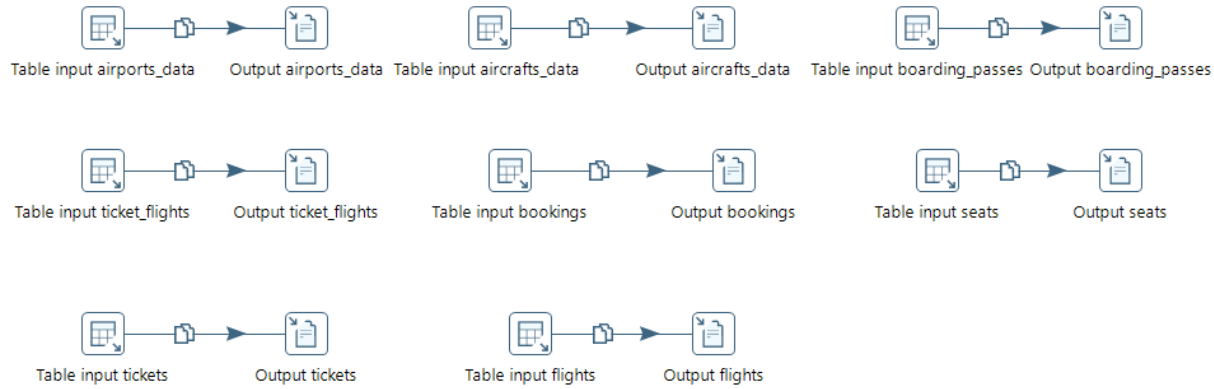
3. В параметрах элемента Text file output для сохранения в CSV указываем разделитель «;» и перечисляем все поля исходной таблицы.

4. Все параметры подключения к БД bookings, пути, имена файлов задаем с помощью переменных в файле kettle.properties и указываем в трансформации в формате ${variable_name}.

Трансформация находится в файле db_to_csv.ktr.

Данные таблиц БД сохраняются bookings сохраняются в одноименные с именем таблиц файлы с расширением .csv

Трансформация переноса таблиц БД bookings в CSV файлы в Spoon Pentaho.



Результат работы трансформации db_to_csv.ktr

Имя	Размер
tickets.csv	82 280 КБ
ticket_flights.csv	84 200 КБ
seats.csv	23 КБ
flights.csv	7 621 КБ
bookings.csv	23 307 КБ
boarding_passes.csv	50 710 КБ
airports_data.csv	16 КБ
aircrafts_data.csv	1 КБ

3. Целевая база данных.

В PostgeSQL создаем локальную базу данных. Эту базу данных будем использовать в качестве целевого хранилища Data Warehouse.

Подключение к базе данных:

1. Host - localhost
2. Порт – 5432
3. База данных – dwh_project
4. Схема – dwh
5. Пользователь – postgres
6. Пароль - <password>

Все параметры подключения к хранилищу dwh_project задаем с помощью переменных в файле kettle.properties.

4. Создание таблиц в хранилище.

4.1 Создание таблиц измерений.

Создадим с помощью скрипта на языке SQL таблицу-справочник календарь

dim-calendar в хранилище:

```
CREATE TABLE dim_calendar (
        id_calendar SERIAL PRIMARY KEY,
        "date" date UNIQUE NOT NULL,
        day_week integer NOT NULL,
        "month" integer NOT NULL,
        "year" integer NOT NULL);
```

Где:
id_calendar - первичный ключ таблицы;
date – поле дата;
day_week – поле день недели;
month – месяц;
year – год.

Таблица dim-calendar необходима для построения отчетов для анализа данных по перелетам в разрезе даты, дня недели, месяца, года.

Создадим с помощью скрипта на языке SQL таблицу- справочник аэропортов dim_airports в хранилище:

```
CREATE TABLE dim_airports (
        airport_code varchar(3) PRIMARY KEY,
        airport_name_en varchar(100) NOT NULL,
        airport_name_ru varchar(100) NOT NULL,
        city_en varchar(60) NOT NULL,
        city_ru varchar(60) NOT NULL,
        longitude  double precision NOT NULL,
        latitude  double precision NOT NULL,
        timezone varchar(30) NOT NULL);
```

Где:
airport_code – код аэропорта, первичный ключ таблицы;
airport_name_en – наименование аэропорта на английском языке;
airport_name_ru – наименование аэропорта на русском языке;
city_en - наименование города аэропорта на английском языке;
city_ru - наименование города аэропорта на русском языке;
longitude – географическая координата аэропорта - долгота;
latitude - географическая координата аэропорта -широта;
timezone – временная зона аэропорта.

Создадим с помощью скрипта на языке SQL таблицу-справочник пассажиров dim_passengers в хранилище:

```
CREATE TABLE dim_passengers (
        passenger_id varchar(15) PRIMARY KEY,
        passenger_name varchar(30) NOT NULL,
        phone varchar(20) NOT NULL,
        email varchar(80));
```

Где:
passenger_id - первичный ключ таблицы;
passenger_name – Имя Фамилия пассажира;
phone – телефон пассажира;

email – email пассажира.

Создадим с помощью скрипта на языке SQL таблицу-справочник воздушных суден dim_aircrafts в хранилище:

```
CREATE TABLE dim_aircrafts (
        aircraft_code varchar(3) PRIMARY KEY,
        model_en varchar(70) NOT NULL,
        model_ru varchar(70) NOT NULL,
        "range" integer NOT NULL);
```

Где:

aircraft_code - первичный ключ таблицы;
model_en - наименование воздушного судна на английском языке;
model_ru - наименование воздушного судна на русском языке;
range – максимальная дальность перелета в километрах.

Создадим с помощью скрипта на языке SQL таблицу-справочник тарифов dim_aircrafts в хранилище:

```
CREATE TABLE dim_tariff (
        aircraft_code varchar(3) NOT NULL,
        seat_no varchar(3) NOT NULL,
        fare_conditions varchar(10) NOT NULL,
        PRIMARY KEY (aircraft_code, seat_no));
```

Где:

aircraft_code – код воздушного судна;
seat_no - номер места воздушного судна;
fare_conditions – класс обслуживания.

В таблице dim_tariff применен составной первичный ключ, состоящий из полей aircraft_code и seat_no.

Так же все поля кроме поля email таблицы dim_passengers применено ограничение NOT NULL либо ограничение первичного ключа.

Скрипт SQL для создания таблиц-справочников находится в файле dwh_sql.sql

4.2 Создание таблицы фактов fact_flights.

Создадим с помощью скрипта на языке SQL таблицу фактов fact_flights в хранилище:

```
CREATE TABLE fact_flights (
        id_fact_flights SERIAL PRIMARY KEY,
        fk_aircraft_code varchar(3) NOT NULL,
        fk_seat_no varchar(3) NOT NULL,
        fk_passenger_id varchar(15) NOT NULL,
        fk_id_calendar integer NOT NULL,
        fk_departure_airport_code varchar(3) NOT NULL,
        fk_arrival_airport_code varchar(3) NOT NULL,
        passenger_name varchar(30) NOT NULL,
```

```
        actual_departure timestamp NOT NULL,
        actual_arrival timestamp NOT NULL,
        delay_departure integer NOT NULL,
        delay_arrival integer NOT NULL,
        aircraft_en varchar(70) NOT NULL,
        aircraft_ru varchar(70) NOT NULL,
        departure_airport_en varchar(100) NOT NULL,
        departure_airport_ru varchar(100) NOT NULL,
        arrival_airport_en varchar(100) NOT NULL,
        arrival_airport_ru varchar(100) NOT NULL,
        fare_conditions varchar(10) NOT NULL,
        amount numeric(10,2) NOT NULL,
        FOREIGN KEY (fk_departure_airport_code) REFERENCES dim_airports(airport_code),
        FOREIGN KEY (fk_arrival_airport_code) REFERENCES dim_airports(airport_code),
        FOREIGN KEY (fk_id_calendar) REFERENCES dim_calendar(id_calendar),
        FOREIGN KEY (fk_passenger_id) REFERENCES dim_passengers(passenger_id),
        FOREIGN KEY (fk_aircraft_code) REFERENCES dim_aircrafts(aircraft_code),
        FOREIGN KEY (fk_aircraft_code, fk_seat_no) REFERENCES dim_tariff(aircraft_code,
seat_no)
    );
```

Где:

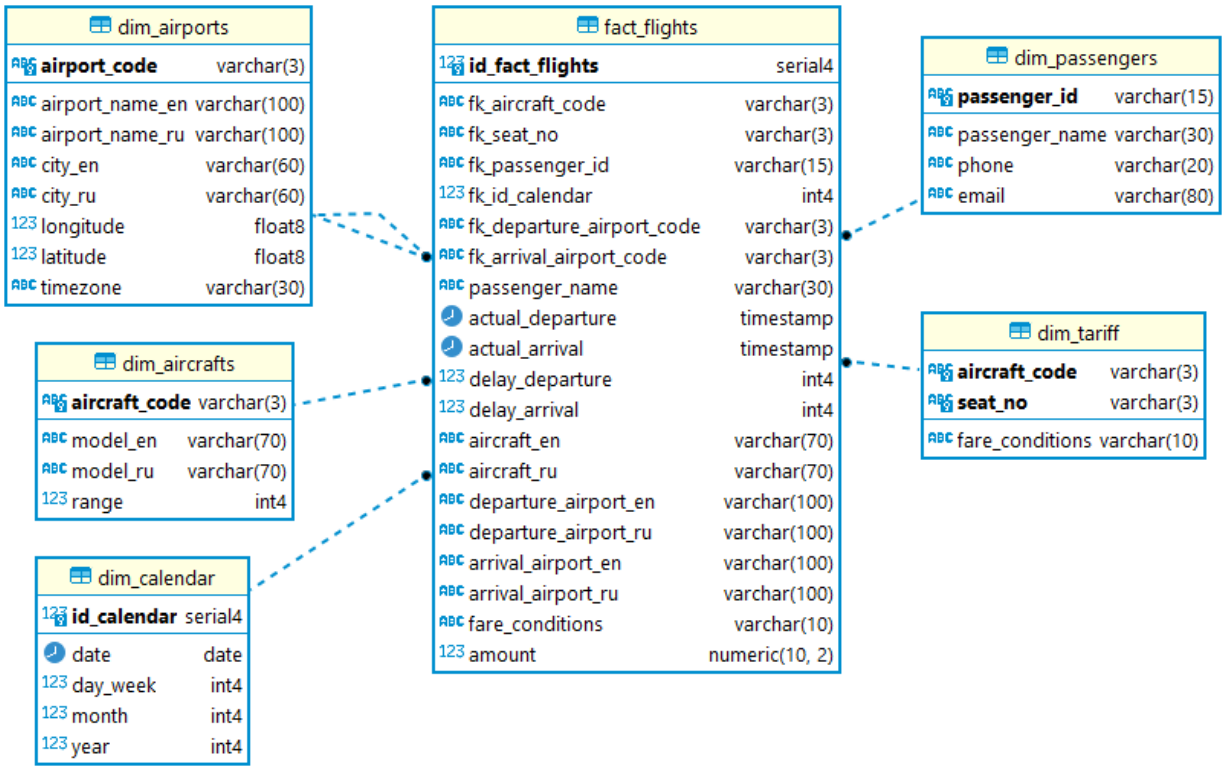
id_fact_flights - первичный ключ таблицы;
fk_aircraft_code – внешний ключ, связь - с таблицей-справочником dim_aircrafts, связь - с таблицей-справочником dim_tariff;
fk_seat_no – внешний ключ, связь - с таблицей-справочником dim_tariff;
fk_passenger_id – внешний ключ, связь - с таблицей-справочником dim_passengers;
fk_id_calendar – внешний ключ, связь - с таблицей-справочником dim_calendar;
fk_departure_airport_code – внешний ключ, связь - с таблицей-справочником dim_airports;
fk_arrival_airport_code – внешний ключ, связь - с таблицей-справочником dim_airports;
passenger_name – Имя Фамилия пассажира;
actual_departure – фактическая дата и время вылета;
actual_arrival - фактическая дата и время прибытия;
delay_departure – задержка вылета в секундах;
delay_arrival – задержка прибытия в секундах;
aircraft_en - наименование воздушного судна на английском языке;
aircraft_ru varchar(70) NOT NULL, - наименование воздушного судна на русском языке;
departure_airport_en - наименование аэропорта вылета на английском языке;
departure_airport_ru - наименование аэропорта вылета на русском языке;
arrival_airport_en - наименование аэропорта прибытия на английском языке;
arrival_airport_ru - наименование аэропорта прибытия на русском языке;

fare_conditions – класс обслуживания;
amount numeric – стоимость перелета в руб.

Скрипт SQL для создания таблицы фактов fact_flights находится в файле dwh_sql.sql

4.3 ER-диаграмма хранилища.

На рисунке представлена ER-диаграмма хранилища, построенная по схеме «звезда»



5. ETL загрузки в хранилище.

5.1 Загрузка данных в таблицу-справочник dim_calendar.

Реализуем загрузку данных в хранилище. Первой наполним данными таблицу-справочник dim_calendar. Наполнять таблицу будем с помощью SQL скрипта:

```
DO $$
DECLARE
   i INTEGER = 0;
   T_DATE  date = '2017-01-01'; -- начальная дата
   E_DATE  date = '2030-12-31'; -- конечная дата
BEGIN
  WHILE (T_DATE+i)<=E_DATE
  LOOP
    INSERT INTO dim_calendar ("date", day_week,"month","year")
    VALUES (T_DATE+i, EXTRACT(isodow FROM T_DATE+i),EXTRACT(month FROM
T_DATE+i),EXTRACT(year FROM T_DATE+i));
    i:= i + 1;
  END LOOP;
END $$;
```

Скрипт SQL для наполнения данными таблицы dim_calendar находится в файле dwh_sql.sql. Для данной таблицы отсутствует ручной ввод данных, и проверки реализованы на в скрипте SQL для создания таблицы оператором CREATE.

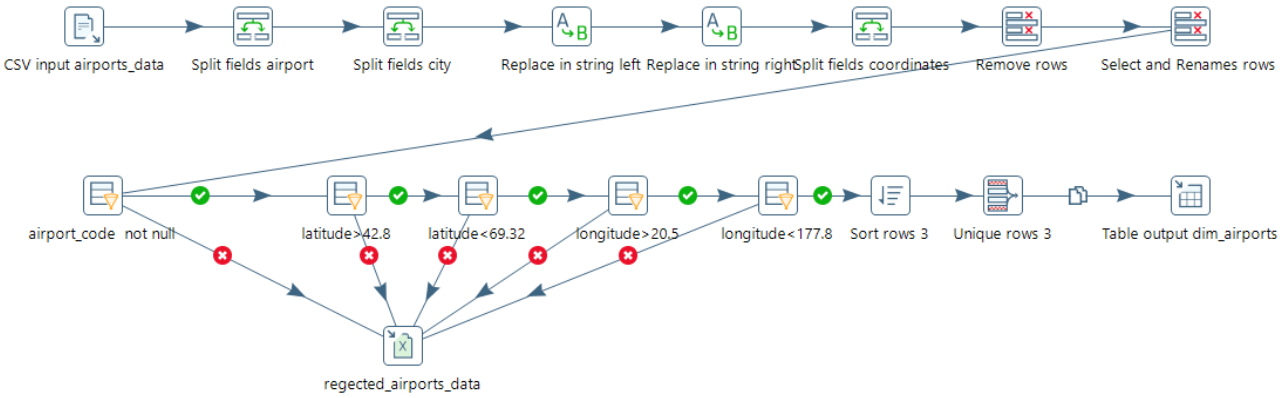
5.2 Загрузка данных в таблицу-справочник dim_airports.

Наполнять данными таблицу-справочник dim_airports будем с помощью ETL трансформации в Spoon Pentaho:

1. С помощью элемента «CSV file input» считываем из файла airports_data.csv экспорта данных из БД bookings, и отбираем поля в поток данных:
   airport_code
   airport_name
   city
   coordinates
   timezone
2. С помощью элемента «Split fields» разобьем поле airport_name со значением поля вида «{"en": "Yakutsk Airport", "ru": "Якутск"}» на два поля airport_name_en «{"en": "Yakutsk Airport"» и airport_name_ru « "ru": "Якутск"}» по разделителю «,».
3. Аналогично с помощью элемента «Split fields» разобьем поле city со значением поля вида «{"en": "Yakutsk", "ru": "Якутск"}» на два поля city_en «{"en": "Yakutsk"» и city_ru « "ru": "Якутск"}» по разделителю «,».
4. С помощью последовательного применения двух элементов «Replace in string» удаляем лишние символы вначале и конце полей airport_name_en, airport_name_ru. city_en, city_ru, а также скобки у поля coordinates.
5. С помощью элемента «Split fields» разобьем поле coordinates на два поля долгота longitude и широта latitude.
6. Удаляем далее не нужные промежуточные поля. Этот элемент лишний, не нужен в связи с применением следующего элемента. Применен в учебных целях.
7. С помощью элемента «Select values» отбираем и переименовываем только необходимые поля в целевой таблице dim_airports.
8. С помощью последовательного применения пяти элементов «Filter rows» осуществляем проверки:
   airport_code не является NULL;
   latitude>42,8 – проверка не выхода координаты за допустимое южное значение координаты;
   latitude>69,32 – проверка не выхода координаты за допустимое северное значение координаты;
   longitude>20,5 – проверка не выхода координаты за допустимое западное значение координаты;
   longitude>177,8 – проверка не выхода координаты за допустимое восточное значение координаты;
   Строки с прошедшие проверку попадают в выходной поток, не прошедшие проверку сохраняются в таблицу в файл rejected_airports_data.xls.
9. С помощью элемента «Sort rows» сортируем данные по полю airport_code для подготовки к следующему действию.
10. Отбираем уникальные строки по полю airport_code с помощью элемента «Unique rows».
11. При помощи элемента «Table output» сохраняем полученные данные в целевое хранилище в таблицу справочник dim_airports, с полями:

airport_code
airport_name_en
airport_name_ru
city_en
city_ru
longitude
latitude
timezone

ETL трансформация наполнения данными таблицы dim_airports в хранилище из CSV файла.



5.3 Загрузка данных в таблицу-справочник dim_passengers.

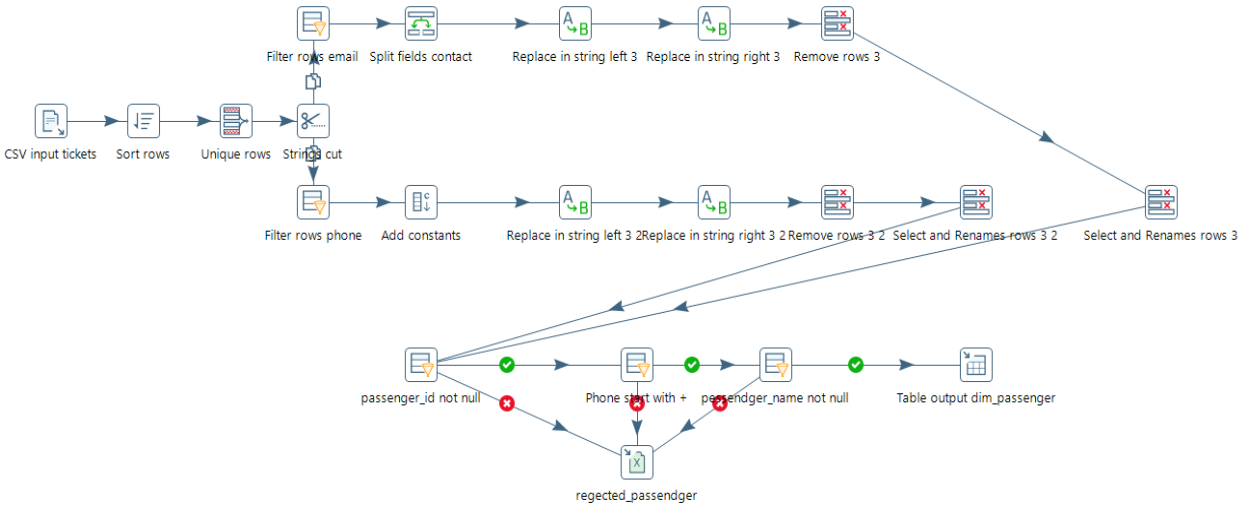
1. С помощью элемента «CSV file input» считываем из файла tickets.csv экспорта данных из БД bookings, и отбираем поля в поток данных:

passendger_id
passendger_name
contacts_data

2. С помощью элементов «Sort rows», «Unique rows» сортируем данные и отбираем уникальные строки по полю passindger_id.
3. С помощью элемента «Strings cut» из поля contacts_data выделяем поле pr1 содержащее:
   phone – если в поле contacts_data содержится только телефон пассажира;
   email - если в поле contacts_data содержится email и телефон пассажира.
4. Фильтруем строки по признаку pr1 = email, разделяем поле contacts_data на два поля email и phone, очищаем полученные поля от лишних символов в начале и конце, убираем ненужные промежуточные колонки(лишняя учебная операция), отбираем нужные колонки в нужном порядке с помощью последовательности элементов «Filter rows», «Split fields», «Replace in string», «Replace in string», «Select values», «Select values». Подробно эта последовательность действий была описана на примере формирования таблицы-справочника dim_airports.
5. Аналогично предыдущему шагу поступаем со строками pr1 = phone, за исключением того, что данные по email у нас отсутствуют в поле contacts_data и формируются не элементом «Split fields», а добавлением константы NULL в поле email с помощью элемента «Add constans». Данные этой последовательности точно по полям и их порядку соответствуют предыдущему шагу для последующего слияния.
6. Объединяются данные по предыдущим двум шагам и аналогично проверкам для таблицы-справочника dim_airports проверяются:
   passenger_id – поле не равно NULL;
   phone – поле начинается с символа «+»;
   passenger_name - – поле не равно NULL.
   Строки с прошедшие проверку попадают в выходной поток, не прошедшие проверку сохраняются в таблицу в файл rejected_passengers.xls.
7. При помощи элемента «Table output» сохраняем полученные данные в целевое хранилище в таблицу справочник dim_passengers, с полями:

passenger_id
passenger_name
phone
email

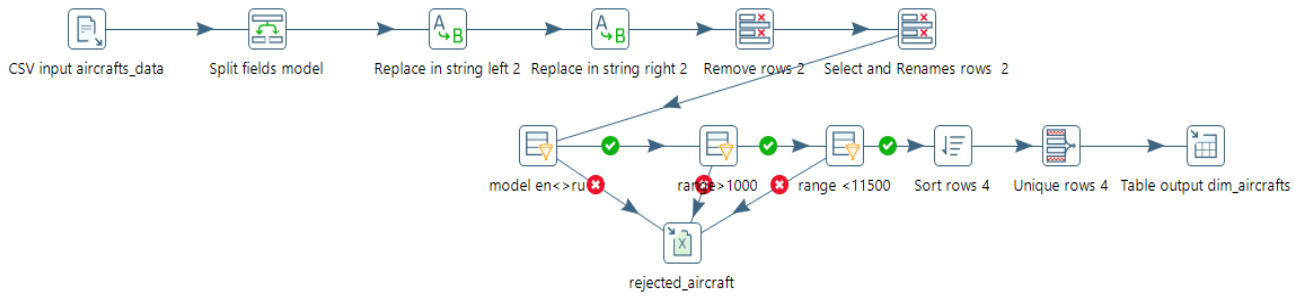
ETL трансформация наполнения данными таблицы dim_passengers в хранилище из CSV файла.



5.4 Загрузка данных в таблицу-справочник dim_aircrafts.

1. С помощью элемента «CSV file input» считываем из файла tickets.csv экспорта данных из БД bookings, и отбираем поля в поток данных:
   aircraft_code
   model
   range
2. Поле model разбивается на два поля model_en и model_en. Действия аналогичны ранее рассмотренным для полей airport_name_en и airport_name_ru для таблицы dim_airports.
3. Аналогично проверкам для таблицы-справочника dim_airports проверяются:
   model_en<>model_ru – наименование воздушного судна на английском языке не равно наименованию на русском языке;
   range > 1000 - максимально возможная дальность перелета воздушного судна более 1000 км;
   range <11500 - максимально возможная дальность перелета воздушного судна меньше 11500 км.
   Строки с прошедшие проверку попадают в выходной поток, не прошедшие проверку сохраняются в таблицу в файл rejected_aircraft.xls.
4. При помощи элемента «Table output» сохраняем полученные данные в целевое хранилище в таблицу справочник dim_aircrafts, с полями:
   aircraft_code
   model_en
   model_ru

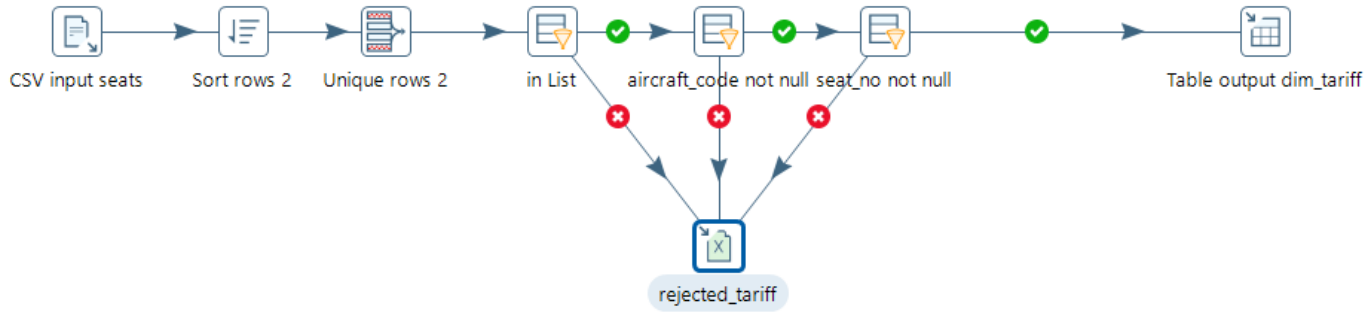
ETL трансформация наполнения данными таблицы dim_aircrafts в хранилище из CSV файла.



5.5 Загрузка данных в таблицу-справочник dim_tariff.

1. С помощью элемента «CSV file input» считываем из файла seats.csv экспорта данных из БД bookings, и отбираем поля в поток данных:
   aircraft_code
   seat_no
   fare_conditions
2. С помощью элементов «Sort rows», «Unique rows» сортируем данные и отбираем уникальные строки по двум полям aircraft_code и seat_no.
3. Аналогично проверкам для таблицы-справочника dim_airports проверяются:
   fare_conditions – поле равно одному из трех значений Economy, Business, Comfort;
   aircraft_code – поле не равно NULL;
   seat_no – поле не равно NULL.
   Строки с прошедшие проверку попадают в выходной поток, не прошедшие проверку сохраняются в таблицу в файл rejected_tariff.xls.
4. При помощи элемента «Table output» сохраняем полученные данные в целевое хранилище в таблицу справочник dim_tariff, с полями:
   aircraft_code
   seat_no
   fare_conditions

ETL трансформация наполнения данными таблицы dim_tariff в хранилище из CSV файла.



5.6 Загрузка данных в таблицу фактов fact_flights.

1. С помощью элемента «CSV file input» считываем из файла flights.csv экспорта данных из БД bookings, и отбираем поля в поток данных:

flight_id
flight_no
scheduled_departure
scheduled_arrival
departure_airport
arrival_airport
status
aircraft_code
actual_departure
actual_arrival

2. С помощью элемента «Filter rows» фильтруем стоки по полю status = «Arrived».
3. С помощью элемента «Calculator» рассчитываем поле delay_departure как разницу в секундах между actual_departure и scheduled_departure.

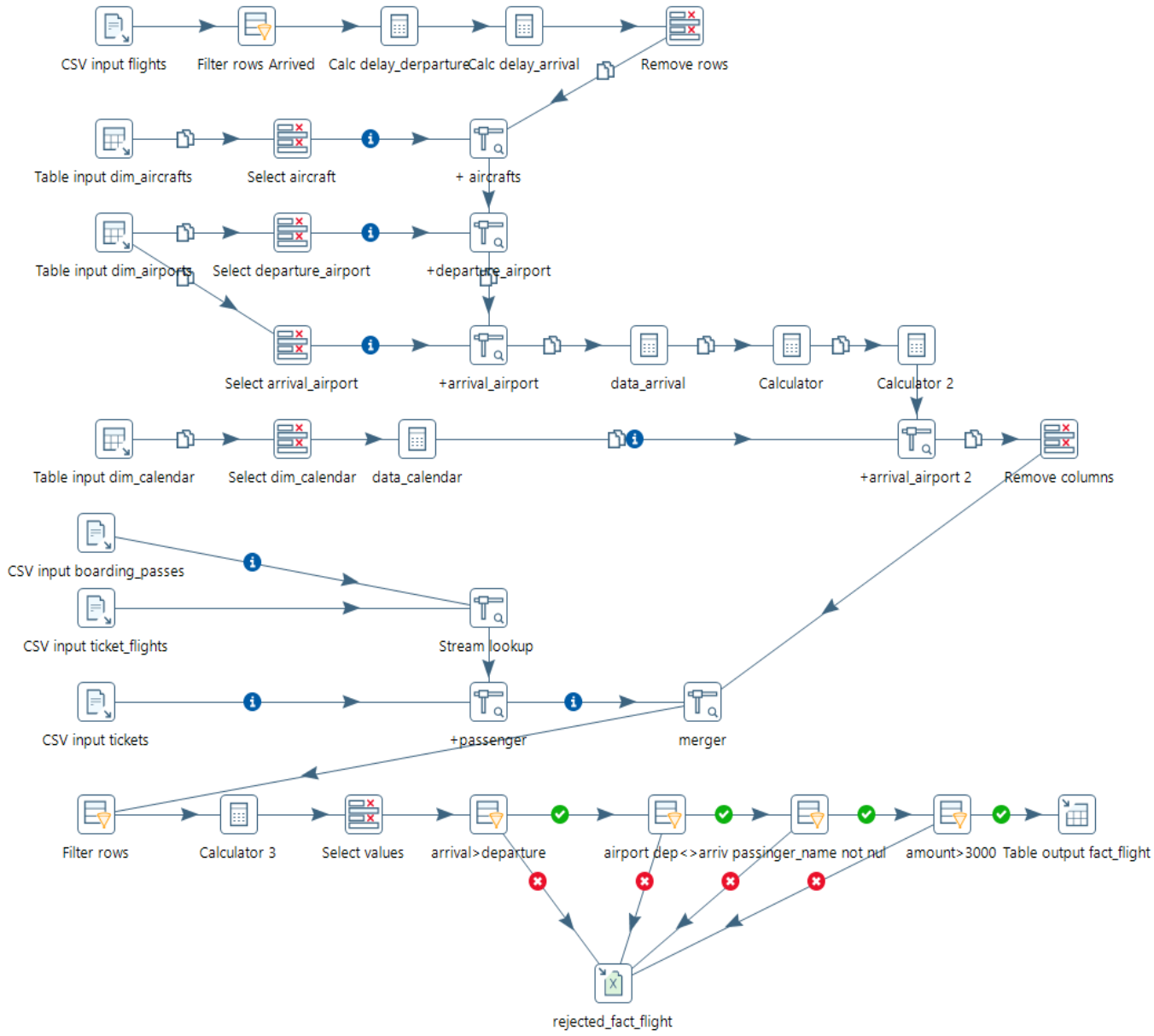
4. С помощью элемента «Calculator» рассчитываем поле delay_arrival как разницу в секундах между actual_arrival и scheduled_arrival.
5. С помощью элемента «Select values» удаляем ненужные далее колонки из потока:
   scheduled_arrival
   scheduled_departure
   status
6. С помощью элемента «Table input» считываем из подготовленной ранее таблицы справочника dim_aircrafts поля:
   aircraft_code
   model_en
   model_ru
7. С помощью элемента «Select values» отбираем и переименовываем колонки. Данные преобразованы в следующие колонки:
   aircraft_code
   aircraft_en
   aircraft_ru
8. С помощью элемента «Stream lookup» в потоке данных, полученном на шаге 5, ищем совпадение по полям aircraft_code = aircraft_code и добавляем данные по полям aircraft_en и aircraft_ru.
9. Аналогично шагам 6-8 добавляем данные из таблицы dim_airports по полям:
   departure_airport_en
   departure_airport_ru
   departure_airport_code
   arrival_airport_en
   arrival_airport_ru
   arrival_airport_code
10. Далее привяжем к нашему потоку данных данные из таблицы справочника dim_calendar. На этом этапе столкнулся со сложностью преобразовав дату прибытия с помощью элемента «Calculator» к виду yyyy/MM/dd и дату из поля date таблицы dim_calendar совпадение не получилось из-за составляющей time. Не помогло и применение действия «Remove time from a date A» элемента «Calculator».
    Для очистки временной составляющей было введено преобразование actual_arrival в поле data_arrival по маске yyyy/MM/dd в формате string с помощью элемента «Calculator». Далее обратное преобразование data_arrival из формата string в формат date.
11. С помощью элементов «Table input» и элемента «Select values» считываем из подготовленной ранее таблицы справочника dim_calendar поля:
    id_calendar
    date
12. С помощью элемента «Calculator» приводим формат поля date к формату data_arrival из шага 10, далее с помощью элемента «Stream lookup» обогащаем поток данных данными Id_calendar шага 11 по совпадению полей date = data_arrival. Напомню поле date таблицы dim_calendar уникально и сгенерировано SQL скриптом и описано в пункте 5 и фактически тоже может служить первичным ключом.
    Для удобства сгенерирован суррогатный первичный ключ id_calendar типа serial.
13. С помощью элемента «Select values» удаляем временные ненужные колонки.
14. С помощью элементов «Table input» считываем из подготовленной ранее таблицы справочника dim_calendar поля:
15. С помощью элемента «CSV file input» считываем из файла boarding_passes.csv экспорта данных из БД bookings, и отбираем поля в поток данных:
    ticket_no
    flight_id

seat_no

16. С помощью элемента «CSV file input» считываем из файла ticket_flights.csv экспорта данных из БД bookings, и отбираем поля в поток данных:
    ticket_no
    flight_id
    fare_conditions
    amount
17. С помощью элемента «Stream lookup» обогащаем данные, полученные на шаге 15 данными поля seat_no из данных, полученных на шаге 16 по совпадению полей flight_id= flight_id и ticket_no =ticket_no.
18. 16. С помощью элемента «CSV file input» считываем из файла tickets.csv экспорта данных из БД bookings, и отбираем поля в поток данных:
    ticket_no
    passenger_id
    passenger_name
19. С помощью элемента «Stream lookup» обогащаем данные, полученные на шаге 17, данными полей passenger_id и passendger_name из данных, полученных на шаге 18 по совпадению поля ticket_no =ticket_no.
20. С помощью элемента «Stream lookup» обогащаем данные, полученные на шаге 13, данными полей amount, passenger_id, passenger_name, seat_no, fare_conditions из данных, полученных на шаге 18 по совпадению поля flight_id= flight_id.
21. С помощью элемента «Filter rows» строки, где seat_no <> NULL.
22. С помощью элемента «Calculator» actual_departure и actual_ arrival в тип данных timestamp.
23. С помощью элемента «Select values» при необходимости переименовываем поля и отбираем необходимые.
24. Аналогично проверкам для таблицы-справочника dim_airports проверяются:
    actual_arrival >actual_departure;
    departure_airport_ru<>arrival_airport_ru;
    passenger_name - поле не равно NULL;
    amount > 3000.
    Строки с прошедшие проверку попадают в выходной поток, не прошедшие проверку сохраняются в таблицу в файл rejected_fact_flight.xls.
25. При помощи элемента «Table output» сохраняем полученные данные в целевое хранилище в таблицу справочник fact_flights, с полями:
    id_fact_flights
    fk_aircraft_code
    fk_seat_no
    fk_passenger_id
    fk_id_calendar
    fk_departure_airport_code
    fk_arrival_airport_code
    passenger_name
    actual_departure
    actual_arrival
    delay_departure
    delay_arrival
    aircraft_en
    aircraft_ru
    departure_airport_en
    departure_airport_ru
    arrival_airport_en

arrival_airport_ru
fare_conditions
amount

ETL трансформация наполнения данными таблицы фактов fact_flights.



На примере данной трансформации отработан навык оптимизации работы трансформации. Первоначально время работы трансформации составило 5 мин 36 с. Узким местом были операции, производимые элементами «Stream lookup». За счет повышения нагрузки на процессор элементами «Stream lookup» (убрал галочку в поле Preserve memory (costs CPU)) время работы трансформации составило 16с. Повышение нагрузки на локальный компьютер вполне допустимо, на сервере я бы этого делать не стал.

6 Загрузка на github.

SQL-скрипт, трансформации, документация, файл с переменными Pentaho (без актуальных паролей) выложены на github.

Список файлов:

dwh_sql.sql	- скрипты SQL создания таблиц в целевом хранилище.
kettle.properties	- файл с переменными Pentaho (без актуальных паролей)
db_to_csv.ktr	- файл Spoon Pentaho трансформации сохранения исходной облачной базы данных в csv файлы.
ETL.ktr	- файл Spoon Pentaho ETL трансформации заполнения данными таблиц-справочников в целевом хранилище.
ETL_fact.ktr	- файл Spoon Pentaho ETL трансформации заполнения данными таблицы-фактов в целевом хранилище.
dwh_doc.pdf	- этот файл документации.