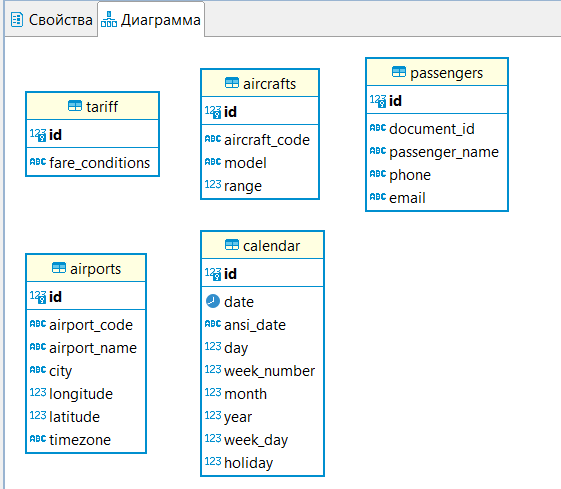
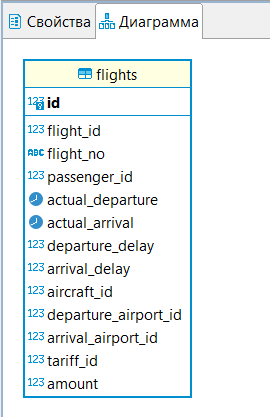
# Реализация ETL-процесса для базы данных bookings

## Новая база данных

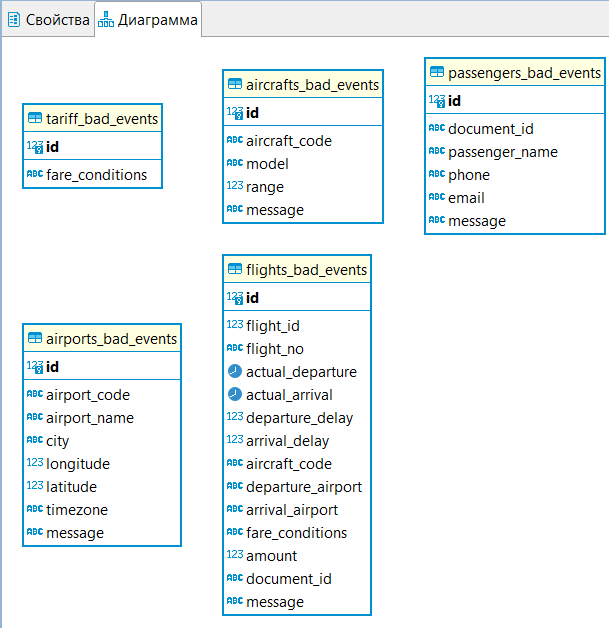
* Создана новая база данных в docker(см. **postgres\_in\_docker.ps1**)
* Созданы схемы(см. **create\_tables\_in\_docker\_bookings.sql**):
  + dim – схема с таблицами измерений
  + fact – схема с таблицами фактов
  + etl\_log – схема с rejected-таблицами ETL процесса
* Созданы таблицы(см. **create\_tables\_in\_docker\_bookings.sql**):
  + dim.calendar - справочник дат
  + dim.passengers - справочник пассажиров
  + dim.aircrafts - справочник самолетов
  + dim.airports - справочник аэропортов
  + dim.tariff - справочник тарифов (Эконом/бизнес и тд)
  + fact.flights - справочник совершенных перелетов
  + etl\_log.aircrafts\_bad\_events - rejected-таблица самолетов
  + etl\_log.airports\_bad\_events - rejected-таблица аэропортов
  + etl\_log.passengers\_bad\_events - rejected-таблица пассажиров
  + etl\_log.tariff\_bad\_events - rejected-таблица тарифов
  + etl\_log.flights\_bad\_events - rejected-таблица перелетов
* ER-диаграмма схемы dim:



* ER-диаграмма схемы fact:

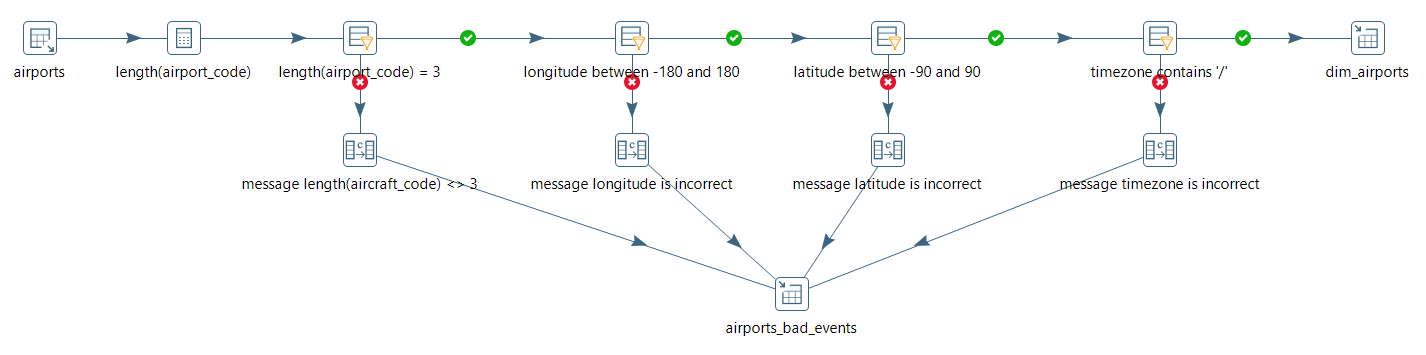


* ER-диаграмма схемы etl\_log:

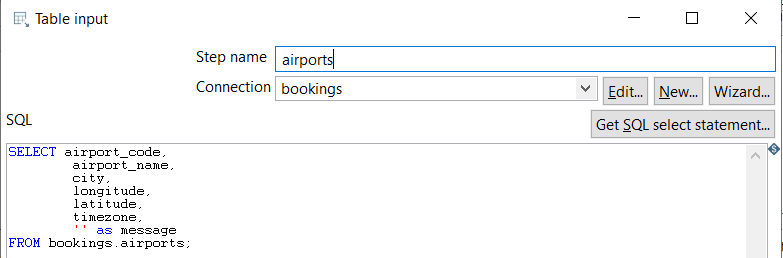


## ETL аэропотров (см. airports.ktr)

Общая схема процесса:



На первом шаге импортируем данные из таблицы **airports** исходной базы данных. Добавляем пустое поле **message**, в которое будем записывать текст ошибки.



Далее выполняем ряд проверок на качество данных:

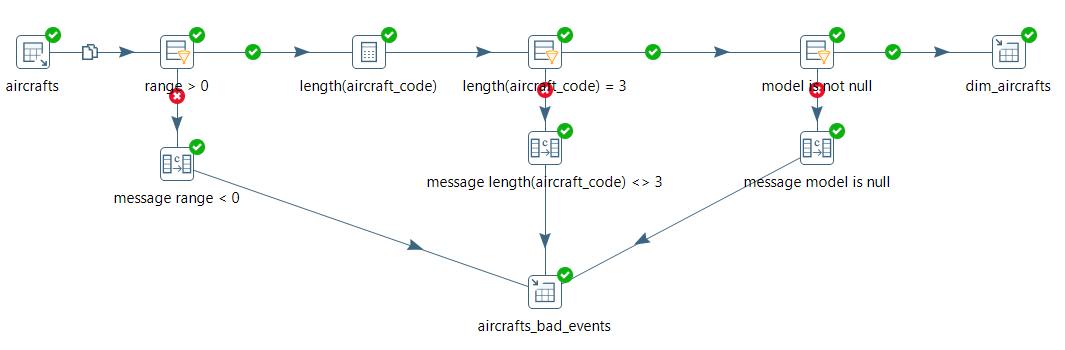
* + Длина кода аэропорта = 3
  + Долгота находится в диапазоне от -180 до 180
  + Широта находится в диапазоне от -90 до 90
  + Временная зона содержит обязательный знак «/»

При невыполнении условий проверки вся строка с соответствующим сообщением записывается в rejected-таблицу **etl\_log.airports\_bad\_events**.

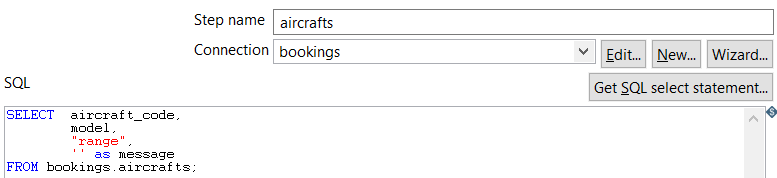
Если все условия выполнены – записываем в таблицу измерений **dim.airports**.

## ETL самолетов (см. aircrafts.ktr)

Общая схема процесса:



На первом шаге импортируем данные из таблицы **aircrafts** исходной базы данных. Добавляем пустое поле **message**, в которое будем записывать текст ошибки.



Далее выполняем ряд проверок на качество данных:

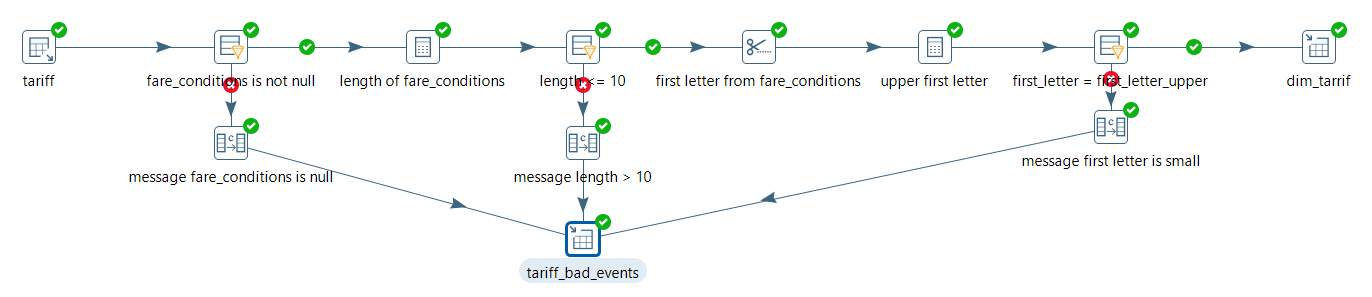
* + Дальность полета > 0
  + Длина кода самолета = 3
  + Модель самолета указана

При невыполнении условий проверки вся строка с соответствующим сообщением записывается в rejected-таблицу **etl\_log.aircrafts\_bad\_events**.

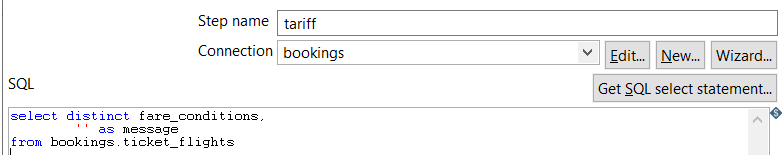
Если все условия выполнены – записываем в таблицу измерений **dim.aircrafts**.

## ETL тарифов (см. tariff.ktr)

Общая схема процесса:



На первом шаге импортируем данные из таблицы **ticket\_flights** исходной базы данных. Добавляем пустое поле **message**, в которое будем записывать текст ошибки.



Далее выполняем ряд проверок на качество данных:

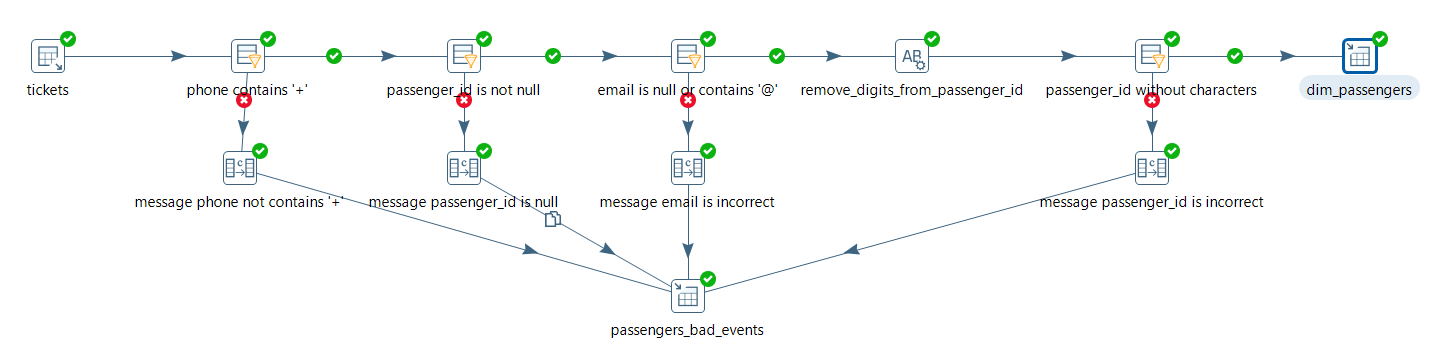
* + Класс обслуживания указан
  + Длина класса обслуживания < 10
  + Класс обслуживания начинается с большой буквы

При невыполнении условий проверки вся строка с соответствующим сообщением записывается в rejected-таблицу **etl\_log.tariff\_bad\_events**.

Если все условия выполнены – записываем в таблицу измерений **dim.tariff**.

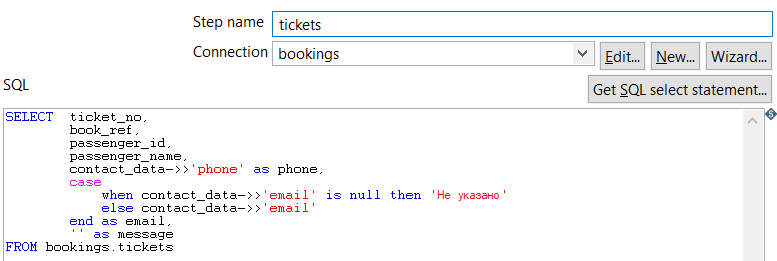
## ETL пассажиров (см. passengers.ktr)

Общая схема процесса:



На первом шаге импортируем данные из таблицы **tickets** исходной базы данных. Добавляем пустое поле **message**, в которое будем записывать текст ошибки.

Также достаем телефон и электронную почту из контактных данных. Если почта is null, явно пишем «Не указано».



Далее выполняем ряд проверок на качество данных:

* + Телефон в формате со знаком «+»
  + Номер документа пассажира заполнен
  + Электронная почта содержит обязательный символ «@»
  + Номер документа содержит только числа и пробел

При невыполнении условий проверки вся строка с соответствующим сообщением записывается в rejected-таблицу **etl\_log.passengers\_bad\_events**.

Если все условия выполнены – записываем в таблицу измерений **dim.passengers**

## ETL перелетов (см. flights.ktr и flights\_2.ktr)

При обновлении таблицы фактов, нам необходимо предварительно получить натуральные ключи из наших таблиц измерений.

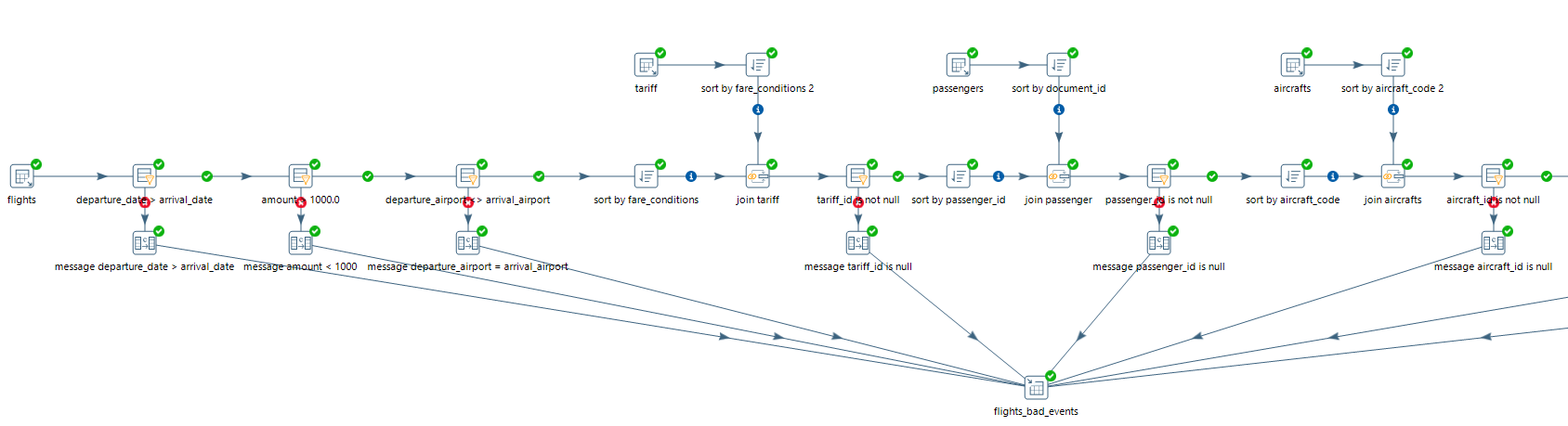
Так как наши таблицы измерений и исходные данные фактов большие, нужен эффективный способ получения этих ключей. После чтения документации и экспериментирования я выбрал шаг **merge join,** который дает мне наилучшую производительность, но не обновляет таблицы измерений в случае отсутствия совпадения.

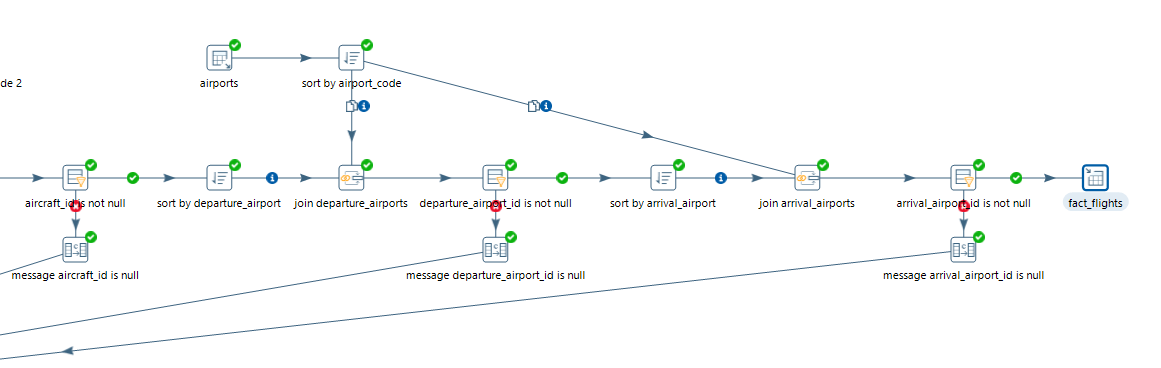
Если мы предполагаем, что ключи в таблицах измерений всегда будут найдены (в нашей исходной БД это гарантировано связями), то на этом можно остановиться. Но если мы, например, загружаем данные из разных источников без какой-либо согласованности, нам все же нужно обновить таблицы измерений с помощью шага **combination lookup/update**.

Поэтому, я разбил весь процесс на 2 этапа.

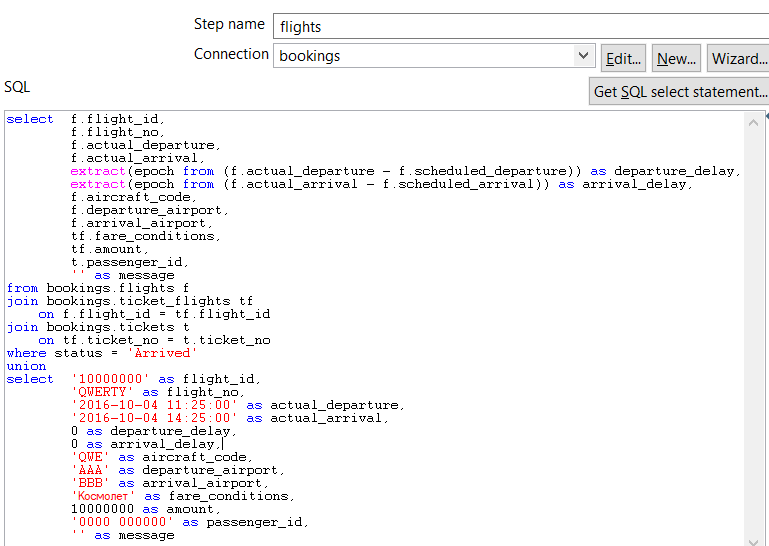
**1 этап**:

Общая схема процесса (не умещается на одном изображении):





На первом шаге импортируем данные о перелетах из таблицы flights исходной базы данных. И сразу подтягиваем информацию о билетах, классе обслуживания и пассажире. Добавляем пустое поле **message**, в которое будем записывать текст ошибки.



Также, я добавил тестовую запись, содержащую коды самолета и аэропортов, а также номер документа и класс обслуживания, которых нет в наших таблицах измерений.

Далее выполняем ряд проверок на качество данных:

* + Время отправления < времени прибытия
  + Цена билета > 1000
  + Аэропорт отправления не равен аэропорту прибытия

При невыполнении условий проверки вся строка с соответствующим сообщением записывается в rejected-таблицу **etl\_log.flights\_bad\_events**.

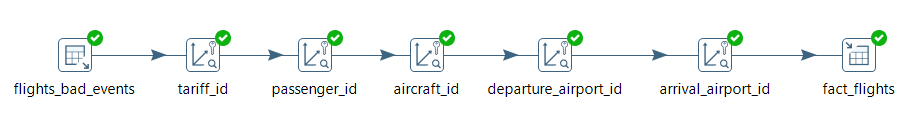
После этого, мы импортируем таблицы измерений и применяем к ним левое внешнее соединение с помощью шагов **merge join.** Предварительно сортируем как наш поток, так и таблицы измерений.

После каждого соединения выполняется проверка – id измерения is not null. Если ключ найден, переходим к следующему шагу, если нет – записываем аналогично ошибке в таблицу **etl\_log.flights\_bad\_events.**

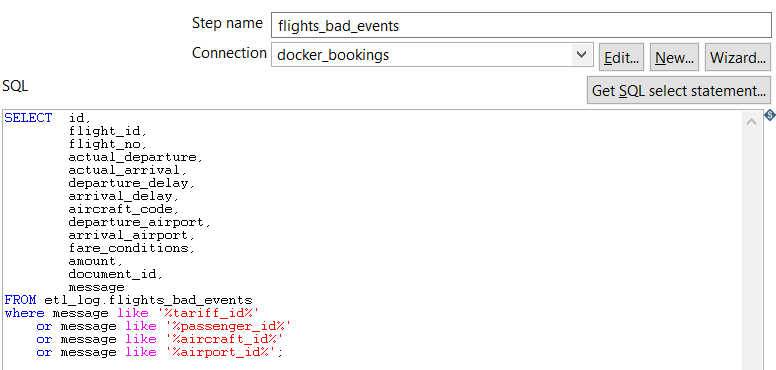
Если все условия выполнены – записываем в таблицу фактов **fact.flights.**

Таким образом, на первом этапе эффективно выполняется ETL основного потока данных.

**2 этап**:

Общая схема процесса:

На первом шаге мы импортируем события, отсеченные на 1 этапе из-за отсутствия ключей.



А далее выполняем шаги combination lookup/update для всех таблиц измерений уже без дополнительных проверок на малой части исходных данных и записываем в нашу таблицу фактов **fact.flights**

Таким образом, если мы предполагаем, что обновление таблиц измерений в процессе обновления фактов не происходит - 2 этап не нужен и мы записываем случаи, когда ключ не подтянулся, как ошибка.

Если обновлять таблицы измерений нужно и данные большие – загружаем сначала основную часть данных, а потом 2-м этапом обновляем таблицы измерений и фактов для оставшейся малой порции. И в этом случае лучше передавать записи непосредственно в следующую трансформацию, а не использовать логирующую таблицу.

## Весь процесс ETL(см. full\_etl.kjb)

Для удобства создано задание, выполняющее последовательно все описанные трансформации.

Общая схема процесса:

