На самом деле, считаю что, тест-кейс не нужен, так как есть постоянное поступление данных, есть логирование, есть представление, показывающее полную разницу между исходными данными и восстановленными из хранилища, представление с частичной разницей, без поля «дата обновления записи. ( см «0\_2 сопроводительная.docx», «Рассмотрим представления в витрине данных»).

Скорее это будет кейс описывающий движение данных и работу системы.

Для понимания движения данных, рекомендую открыть файл «3\_4 схема\_и\_мапинг.xlsx» на вкладке «схема».

Рекомендую остановить автоматическое поступление данных – остановить dag «st34\_case\_3\_control\_v3» в airflow. Это не обязательно, но так будет удобней работать.

Найдем максимальный номер записи в исходной таблице:

**select** **max**(id) max\_id **from** oltp\_src\_system.request\_data;



в дальнейшем будем использовать номер новой записи как N = max\_id +1, для текущей реализации N=340. Это не обязательно, но так будет удобней работать.

Создадим запись о новой заявке:

**insert** **into** oltp\_src\_system.request\_data

**values**

(N,'accepted', **now**(), **now**());

В текущей реализации:

**insert** **into** oltp\_src\_system.request\_data

**values**

(340,'accepted', **now**(), **now**());

Теперь поменяем статус нашей заявки (далее я буду указывать номер заявки в моей, текущей реализации равный 334, напоминаю у вас его надо пересчитать по вышеприведенному алгоритму)на выполняемый'executed'

**update** oltp\_src\_system.request\_data

**set**

status\_nm = 'executed',

update\_dttm = now()

**where** id = 340;

Поставим заявку на паузу

**update** oltp\_src\_system.request\_data

**set**

status\_nm = 'paused',

update\_dttm = **now**()

**where** id = 340;

Еще раз сгенерируем второй раз подряд постановку на паузу

**update** oltp\_src\_system.request\_data

**set**

status\_nm = 'paused',

update\_dttm = **now**()

**where** id = 340;

Проверим , что в исходной таблице заявка с номером 340 находится в состоянии паузы.

**select** \* **from** oltp\_src\_system.request\_data **where** id = 340;

Так и есть:



Заведем новую заявку

**insert** **into** oltp\_src\_system.request\_data

**values**

(341,'accepted', **now**(), **now**());

Сменим статус на выполнен «complited»

**update** oltp\_src\_system.request\_data

**set**

status\_nm = 'complited',

update\_dttm = **now**()

**where** id = 341;

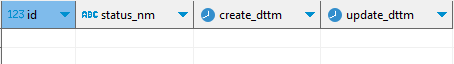
Удалим данную заявку

**delete** **from** oltp\_src\_system.request\_data

**where** id = 341;

Проверим что в исходной таблице заявки с номером 341 нет:

**select** \* **from** oltp\_src\_system.request\_data **where** id = 341;



Теперь рассмотрим таблицу на следующем слое oltp\_cdc\_src\_system.request\_data\_changes .

Данные в эту таблицу попадают при помощи триггера request\_data\_changes().

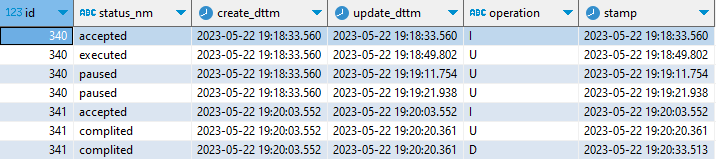
Запрос

**select** \* **from** oltp\_cdc\_src\_system.request\_data\_changes

**where** id **in** (340,341)

**order** **by** id, stamp;

показывает, что все наши манипуляции с заявками сохранены:

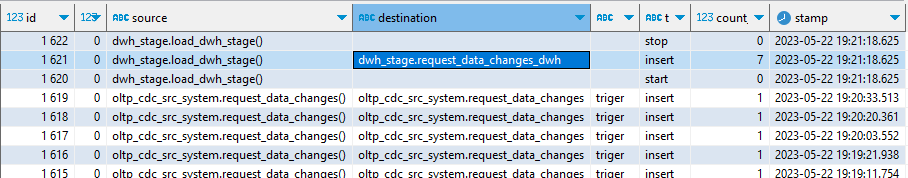


Замечательно, теперь что бы перейти на следующий слой нам нужно запустить загрузку данных на следующий слой

**select** \* **from** dwh\_stage.load\_dwh\_stage();

Что происходит при выполнении функций мы можем увидеть в логе:

**select** \* **from** **log**.v\_show\_log;



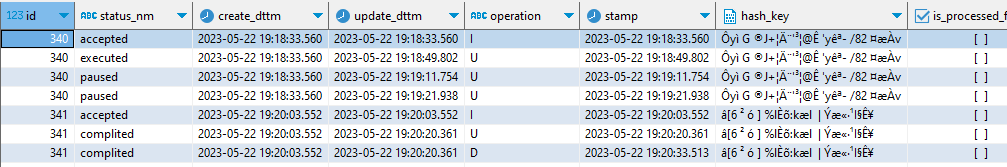
Как видно в логе триггер по одной записи добавлял в oltp\_cdc\_src\_system.request\_data\_changes. Процедура dwh\_stage.load\_dwh\_stage вставила некоторое количество записей в таблицу dwh\_stage.request\_data\_changes\_dwh.

Проверим, есть ли записи о наших заявках в dwh\_stage.request\_data\_changes\_dwh:

**select** \* **from** dwh\_stage.request\_data\_changes\_dwh

**where** id **in** (340,341)

**order** **by** id, stamp;



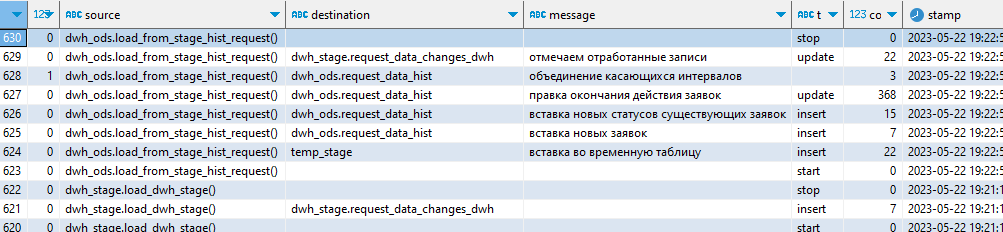
Вся информация прогрузилась в первый слой хранилища. Как видим добавлены поля hash\_key и is\_processing. hash\_key – генерируется на основании полей id и create\_dttm – для удобства идентификации заявки; поле is\_processed\_flg, по умолчанию значение false, означает обработана ли запись для передачи на следующий слой.

Для переноса данных на следующий слой нужно вызвать процедуру dwh\_ods.load\_from\_stage\_hist\_request().

**select** \* **from** dwh\_ods.load\_from\_stage\_hist\_request();

Посмотрим, что скажет лог:

**select** \* **from** **log**.v\_show\_log;

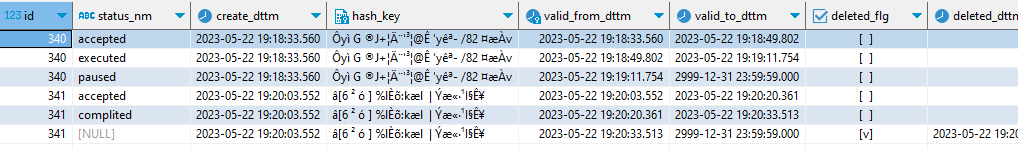


Как видим здесь более подробно расписано, что делает данная процедура. Сейчас заметил, лог говорит, что у меня в алгоритме не оптимально и можно изменить. Обновление даты окончания заявок, проходит по всем заявкам, в то время как это не нужно, нужно было бы добавить дополнительное ограничение и улучшить алгоритм ☺. Но все работает, проверим:

**select** \* **from** dwh\_ods.request\_data\_hist

**where** id **in** (340,341)

**order** **by** id, valid\_from\_dttm;



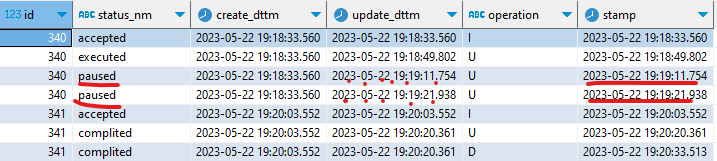
Как видим, вся информация есть в детальном слое, все версии сохранены. Заметим, что по 340 заявке мы дважды генерировали состояние «paused». Но интервал был объединён. Информацию, что происходило объединение интервалов, вы можете видеть на скрине лога.

Ещё раз, приведу запросы и результаты скринов, что было понятней:

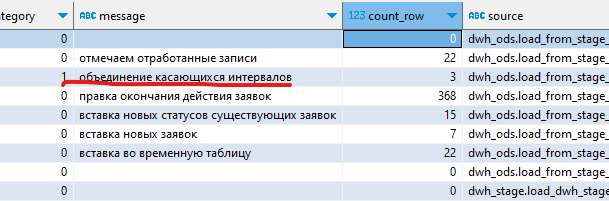
**select** \* **from** oltp\_cdc\_src\_system.request\_data\_changes

**where** id **in** (340,341)

**order** **by** id, stamp;



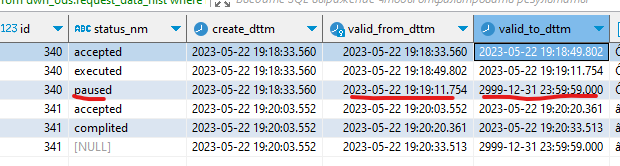
**select** \* **from** **log**.v\_show\_log;



**select** \* **from** dwh\_ods.request\_data\_hist

**where** id **in** (340,341)

**order** **by** id, valid\_from\_dttm;



Это необходимо было объяснить, что бы двинуться дальше.

Дальше рассмотрим представление report.v\_recover\_request\_data, которое формирует на основании данных хранилища исходное актуальное.

**select** \* **from** report.v\_recover\_request\_data

**where** id **in** (340,341)

**order** **by** id;

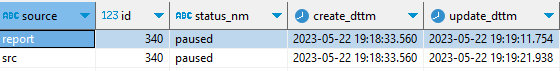


А теперь сравним с исходным при помощи представления report.v\_dif\_source\_target

**select** \* **from** report.v\_dif\_source\_target

**where** id **in** (340,341)

**order** **by** id;



Видим что по 340 заявке данные между исходной таблицей ( поле «source» , значение «src») и восстановленной ( поле «source» , значение «report»). Как, объяснял раньше, это происходит из-за алгоритма работы хранилища, мы склеиваем рядом стоящие интервалы с одним и тем же значением операции. Поэтому, у нас в восстановленные данные более достоверные, ведь обновления данных не было :)