### **Требования к Информационной Системе**

В текущем разделе описывается алгоритм формирования сущности request\_data\_hist – детального справочника с данными по заявкам.

### **Требования к выходным данным**

|  |  |  |
| --- | --- | --- |
| **Схема** | **Объект** | **Описание** |
| dwh\_ods | request\_data\_hist | Версионный справочник по заявкам |

### **Источники**

|  |  |  |  |
| --- | --- | --- | --- |
| **Схема** | **Объект** | **Описание** | **Цель использования** |
| oltp\_src\_system | request\_data | Заявки клиентов на обслуживание | Реализация при помощи триггера механизма cdc |

### **Регламент и механизм загрузки**

Тип загрузки: триггером, построчно, сразу же после внесения изменений

Регламент загрузки: по событиям

|  |  |  |  |
| --- | --- | --- | --- |
| **Схема** | **Объект** | **Условия отбора записей** | **Описание** |
| oltp\_src\_system | request\_data | On insert, update, delete | По событию триггер передает изменённую строку на следующий слой |
| oltp\_cdc\_src\_system | request\_data\_changes |  | Слой cdc. История операций с заявками пользователей |
| dwh\_stage | request\_data\_changes\_dwh |  | Первый слой хранилища. История статуса заявок пользователей, с хэш ключем и вспомогательным полем - признаком обработки записей |

|  |  |
| --- | --- |
| **Название шага** | **Описание алгоритма** |
| triger request\_data\_changes() | По событиям вставки, обновления, удаления формируется вставка записи в oltp\_cdc\_src\_system.request\_data\_changes, добавляется поле operation – принимает значение “I”,”U”,”D” и в поле stamp текужее время  Рассчитываем атрибуты для oltp\_cdc\_src\_system.request\_data\_changes:   * operation * stamp |
| function  dwh\_stage.load\_dwh\_stage() | Процедура вставляет в dwh\_stage.request\_data\_changes\_dwh отсутствующие запси записи из oltp\_cdc\_src\_system.request\_data\_changes |
| table  dwh\_stage.request\_data\_changes\_dwh | В таблице рассчитывается автоматически хэш ключ на основании полей id и create\_dttm – для удобства идентификации заявки. Добавляется поле is\_processed\_flg, по умолчанию значение false. Поле означает обработана ли запись для передачи на следующий слой.  Рассчитываем атрибуты:   * hash\_key * is\_processed\_flg |
| function  dwh\_ods.load\_from\_stage\_hist\_request() | Забираем из dwh\_stage.request\_data\_changes\_dwh необработанные записи (is\_processed\_flg= false)  Вставляем в dwh\_ods.request\_data\_hist новые записи о поступлении новых заявок, в поле valid\_to\_dttm вносим значение '2999/12/31 23:59:59'.  Вставляем записи о новых статусах существующих заявок, в поле valid\_to\_dttm заносим значение '2999/12/31 23:59:59'. У удаленных записей поле status\_nm устанавливаем в null, заполняем поле deleted\_dttm временем stamp и deleted\_flg устанавливаем в true.  Корректируем поле valid\_to\_dttm, для этого группируем по ключу hash\_key, сортируем по valid\_from\_dttm. И используя lead, для каждой записи находим следующее значение для valid\_from\_dttm и заносим его в valid\_to\_dttm, если следующее отсутствует, то вносим '2999/12/31 23:59:59'.  Далее, объединяем касающиеся интервалы.  Для этого все записи, отличающиеся только valid\_from\_dttm, valid\_to\_dttm сортируем по valid\_from\_dttm, valid\_to\_dttm. И если, максимальное valid\_to\_dttm среди предшествующих записей больше или равно valid\_from\_dttm, то в вспомогательное поле заносим 0, иначе 1. Для каждой записи производим суммирование этого вспомогательного поля, для упорядоченных записей по valid\_from\_dttm, valid\_to\_dttm до текущей записи включительно. Группируем по данному полю.  Помечаем обработанные из dwh\_stage.request\_data\_changes\_dwh флагом is\_processed\_flg = true.  Рассчитываем атрибуты для dwh\_ods.request\_data\_hist:   * valid\_to\_dttm * deleted\_flg * deleted\_dttm |

#### **Описание алгоритмов заполнения сущности**