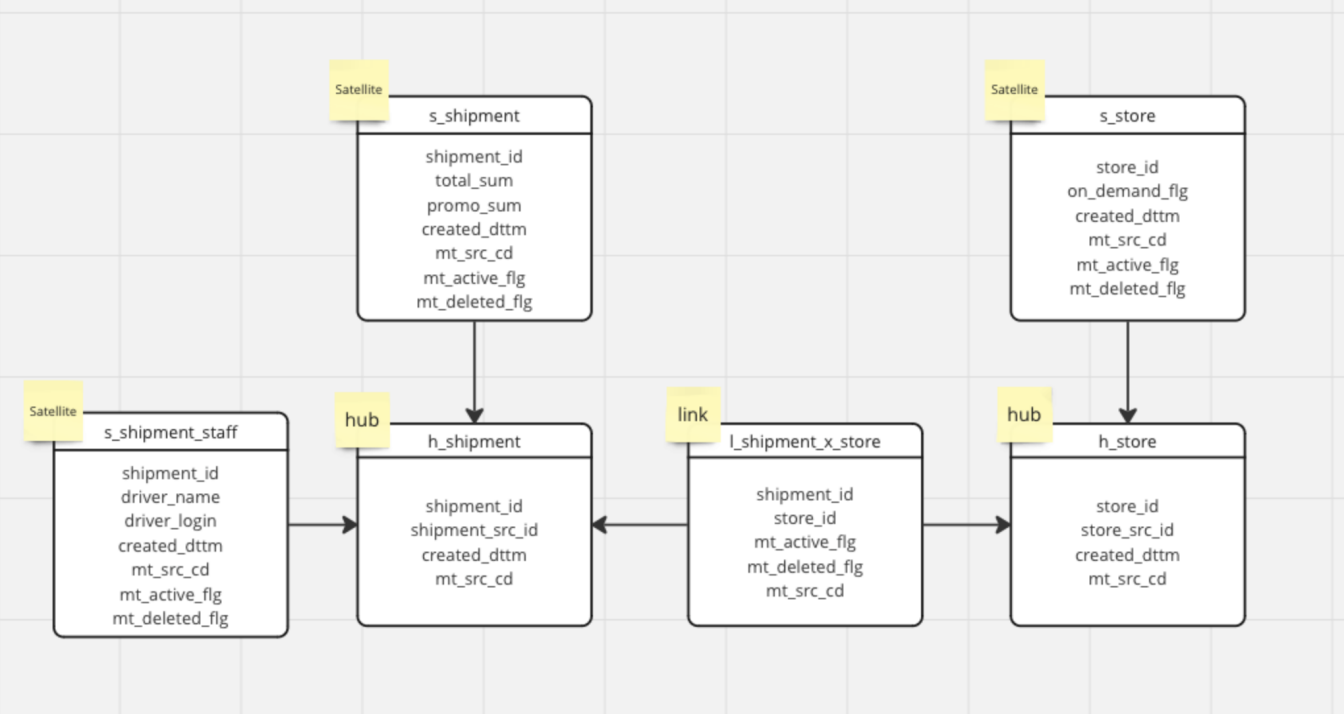
Слой DDS (detail)

DDS - детальный слой данных

Детальный слой хранилища проектируется по методологии Data Vault.

Что такое Data Vault

В нашем ХД все сущности делятся на 5 типов. На схеме ниже представлен пример проектирования по стандартным правилам Data Vault.



Правила нейминга Data Vault

- <entity_name> - всегда в единственном числе
- <src> - как выбрать
    - Если уже есть такой источник - Перечень источников данных
    - А если новый - придумай сам, если сомневаешься - вынеси в мм на обсуждение
    - ограничение - 25 символов

Raw Data Vault

Hub (хаб)

Хабы содержат:

- суррогатный ключ ХД - <entity_name>_id
    - Если в хабе составной ключ, то всегда используем универсальную хэш-конструкцию для конкатенации значений

      ```
      md5(CONCAT_WS('||', coalesce(nullif(<key_field_1>::varchar,''),'^^'), coalesce(nullif
      (<key_field_2>::varchar,''),'^^')))::uuid
      ```

      Собирать такой конструкцией опасно

      ```
      select md5(key_field1::varchar || key_field2::varchar)::uuid
           , ...
      ```

      Это потенциально ошибочная конструкция, которая может привести к искажению данных. Пример, на котором все ломается:

      ```
      --   ,
      select (md5('001' || '12'))::uuid union all
      select (md5('0011' || '2'))::uuid union all
      select (md5('00112' || ''))::uuid;
      ```

      Чтобы этого избежать , следует пользоваться универсальной хэш-конструкцией.

- бизнес-ключ данных источника
    - тип всегда varchar
    - Если в хабе одинарный ключ, то src_<entity_name>_id
        - не важно, как называется поле на источнике
    - Если в хабе составной ключ, то все поля называем src_<field_name>
        - <field_name> - поля из источника, написанное по нашим правилам, как в сателлите
- код источника - src_cd
- дату и время вставки строки - load_dttm
- номер процесса которым загрузился - load_proc_id

В нашем ХД названия хабов начинаются с **'h_'**.

- h_<entity_name>

Суррогатный ключ генерируется с помощью алгоритма хеширования MD5 на основе бизнес-ключа данных источника.

Одинаковые сущности из всех источников собираются в одном хабе (например, все ритейлеры из разных источников). Фильтр по источнику для хабов не используется, при этом в хабе для сущности в техническом поле mt_src_cd сохраняется источник, из которого она впервые попала в хаб (см. также параметр hub_wo_src_cd_flg). Технически это реализовано через union all + row number (см. пример h_retailer_bk). Ключ сущности в хабе уникален.

"Сложные" хабы из нескольких источников можно обновлять несколько раз в день.

Описательные атрибуты хабов хранятся в сателлитах.

Связи между хабами хранятся в линках.

Link (линк)

Линки содержат:

- суррогатные ключи ХД для связываемых сущностей
- код источника связи - при необходимости
- метаданные для версионирования по SCD2

В нашем ХД названия линков начинаются с **'l_'**.

- l_<entity_name>_x_<entity_name>
    - <entity_name>_id
    - <entity_name>_id
- Если соединение не однозначно - в исходной таблице > 1 ключа:
    - l_<entity_name>_x_<entity_name>__on_<смысл связи>
    - пример l_task_x_user__on_moderator

Пары ключей могут повторяться для разных источников, в линке собираем пары из всех источников через union all. Чтобы выделить пары для конкретного источника, нужно использовать фильтр по полю mt_src_cd.

Примечания:

- Не плодим и не создаем лишние линки: в одном линке только 2 хаба
- Мы не делаем сателлиты на линки

Контекстный линк - используется, когда нужно указать какого типа связь существует между объектами. Например, между сделкой и пользователем может быть связь нескольких типов - пользователь может быть создателем/редактором/модератором /ответственным лицом для сделки. На источнике это может выглядеть как несколько FK в таблице сделок на таблицу пользователей. Для таких линков тип связи (контекст) отражаем в имени линка.

Формат имени:
l_<entity1>_x_<entity2>__on_<context>

Примеры:

- l_deal_x_user__on_updated_by
- l_deal_x_user__on_created_by
- l_deal_x_user__on_moderator

Возможны варианты, когда определенная контекстная связь существует не для всех сущностей, а для подгрупп. Например, связь между промоакциями "СоИнвест" и пользователями по создателю. В таком случае в нейминге используется постфикс и указывается контекст.

Формат имени:
l_<entity1>__<context>_x_<entity2>__on_<context>

Примеры:

- l_promo_action__coinvest_x_user__on_updated_by (связь между промоакциями СоИнвест и пользователями по роли "редактор")
- l_promo_action__coinvest_x_user__on_created_by (связь между промоакциями СоИнвест и пользователями по роли "создатель")
- l_promo_action__ad_x_user__on_updated_by (связь между промоакциями платного продвижения и пользователями по роли "редактор")
- l_promo_action__ad_x_user__on_created_by (связь между промоакциями платного продвижения и пользователями по роли "создатель")

Satellite (сателлит)

Сателлиты содержат:

- <entity_name>_id - суррогатный ключ ХД описываемой сущности
- <key_name> - суррогатные/натуральные ключи из источника, правила нейминга ниже
- набор необходимых бизнес-атрибутов по нашим правилу неймнига
- метаданные для версионирования по SCD2

В нашем ХД названия сателлитов начинаются с **'s_'.**

- s_<entity_name>__<src>

Бизнес-атрибуты группируются по сателлитам по двум параметрам:

1. Бизнес-смысл атрибутов
2. Частота изменения атрибутов для избежания дублирования редко-изменяемых данных

Сателлитов у одной сущности может быть несколько, тогда называем их:

s_<entity_name>__<postfix>__<src>

Правила формирования <postfix>:

- если 1 атрибут - <key_name>
- если атрибутов много - то бизнес-смысл атрибутов
  - например, s_shipment_staff
- если для формирование SAT имеет смысл частота изменения атрибутов, то можно добавить одним словом в постфикс, например slow

Пример, когда для одной сущности может быть построено несколько сателлитов на одном источнике: сущность на источнике имеет подвиды, хранится в разных таблицах, имеет различный атрибутивный состав.

Например, на источнике хранятся промоакции различных типов, для каждого типа своя таблица и свой атрибутивный состав. Для сателлитов назначены следующие имена:
- s_promo_action__coinvest__rtl_office (промоакции СоИнвест из источника retailer-office)
- s_promo_action__ad__rtl_office (промоакции платного продвижения из источника retailer-office)

Сателлиты также могут содержать уникальные суррогатные и прочие ключи сущности сателлита.

Для подобных сателлитов у нас правило - один ключ - один сателлит

Такие сателлиты будут иметь вид:

- s_<entity_name>_unq_<key_name>__<src>
  - <entity_name>_id
  - <key_name>

например

- s_shipment_unq_shipment_number_bk__inst

В **Сателлитах** используем следующие правила формирования <key_name>:

- если в названии <field_name> есть <entity_name> или его понятная часть, то:
  - <field_name> - напр. shipment_number
- если в названии <field_name> нет <entity_name>, то:
  - если <field_name> = id, то
    - orig_<entity_name>_<field_name>, напр. orig_shipment_id, если поле в таблице-источнике shipment называется id
  - иначе
    - <entity_name>_<field_name>, напр. shipment_number, если поле в таблице-источнике shipment называется number

При построении **Сателлита** добавлять общую бизнес-логику (что является общей бизнес-логикой - решает владелец сущности) - это норма. Можно и изменять существующие поля, и добавлять новые расчетные.

Business Data Vault

Same-as-Link (связь одного хаба с самим собой)

The recommended best practice is to load all business keys, regardless of their specific format, to one common hub (in this case the customer hub) and create a special link, called a same-as link (or SAL), to indicate the business keys that identify the same business object.

Same as link содержат:

- суррогатный ключ ХД
- дубль суррогатного ключа ХД
- источник связи
- метаданные для версионирования по SCD2

В нашем ХД таблицы Связь хаба с самим собой обозначаются приставкой **'sal_'**

- sal_<entity_name>
  - <entity_name>_id
  - <entity_name>_same_id

Пример (на примере есть ключ самого линка, мы такие ключи не используем):

Table 5.1
Passenger Hub

Passenger HashKey	Load Date	Record Source	Passenger Number
8473d2a…	2014-06-26	DomesticFlight	1234
9d8e72a…	2014-06-26	DomesticFlight	1257
1a4e2c2…	2014-06-26	InternationalFlight	C21X9
238aaff…	2014-06-26	InternationalFlight	C43Z8

Table 5.2
Same-as-Link for Passenger

SALPassenger HashKey	Load Date	Record Source	Master Passenger HashKey	Duplicate Passenger HashKey
38dfa8…	2014-06-26	Dedupe	238aaff…	8473d2a…
937aae…	2014-06-26	Dedupe	1a4e2c2…	9d8e72a…

Hierarchical Link (HAL)

Линк, который используется для указания рекурсивных или иерархических отношений.

Hierarchical link содержат:

- суррогатный ключ ХД
- родительский суррогатный ключ ХД
- источник связи
- метаданные для версионирования по SCD2

В нашем ХД такие таблицы обозначаются приставкой **'hal_'**

- hal_<entity_name>
  - <entity_name>_id
  - parent_<entity_name>_id

Predefined derivations (предрассчитанные показатели, преагрегаты)

Предрассчитанные показатели содержат:

- суррогатный ключ ХД
- набор расчетных атрибутов, отражающих некоторую бизнес-логику
- метаданные для версионирования по SCD2

Архитектурно они представляют из себя "ещё один тип сателлитов". В нашем ХД названия таких таблиц начинаются с **'p_'.**

Предрассчитанные показатели необходимы для избежания дублирования расчетов и разной логики для расчета одних и тех же значений.

Где мы пропускаем RAW Data Vault:

- Экстремально-большой объект - строим SAT/LINK снапшотом
  - критерии - определяем экспертно
  - префиксы
    - s_snp_
    - l_snp_
  - в остальном - правила для SAT/LINK
- Логи
  - строим преагрегат на ODS
- Эксели пользователей
  - строим преагрегат на ODS