
Яндекс Go

Яндекс Go

highly Normalized hybrid Model

Евгений Ермаков, руководитель DWH
Николай Гребенщиков, руководитель команды DE

Глава I. Как мы к этому пришли?

1. Архитектура хранилища Я.Go и детальный слой
2. Data Vault vs Anchor modeling => hNhM
3. hNhM 101

Глава II. Что мы в итоге сделали?

1. hNhM Framework
2. Использование hNhM
3. Оптимизация: Атрибуты vs Группы

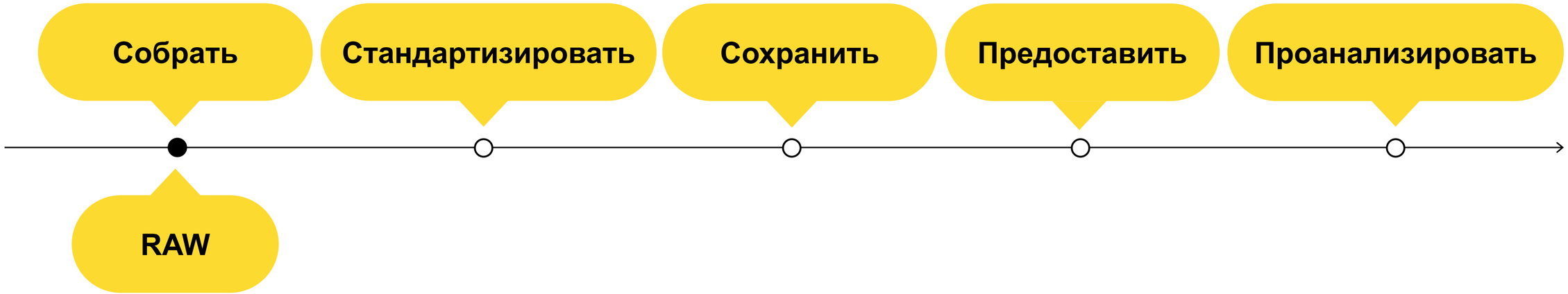
I-1

Архитектура DWH

- Архитектура слоев данных
- Инструменты хранения и обработки данных
- Место детального слоя в архитектуре

Архитектура слоев данных

Архитектура слоев данных



Цель

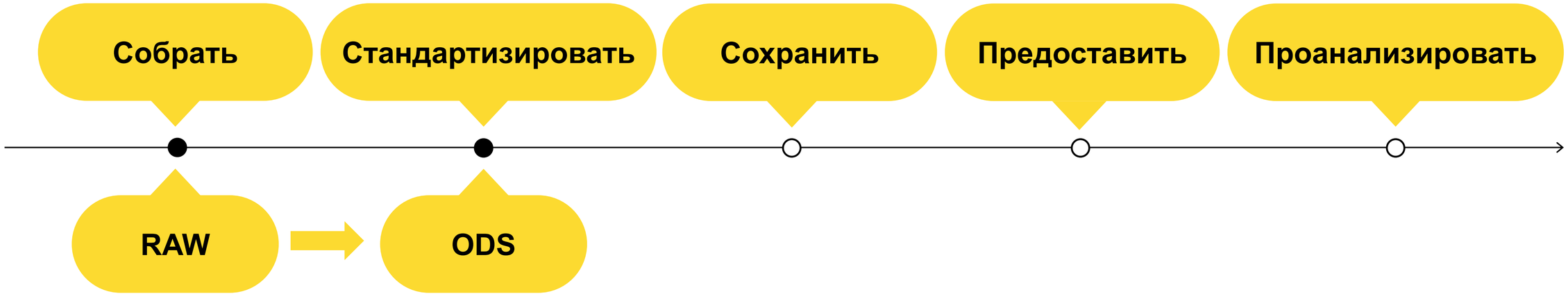
› Сохранить информацию с источника

Задачи

› Собрать данные с источника **as-is**

› Преобразовать в устойчивый к изменениям формат

Архитектура слоев данных



Цель

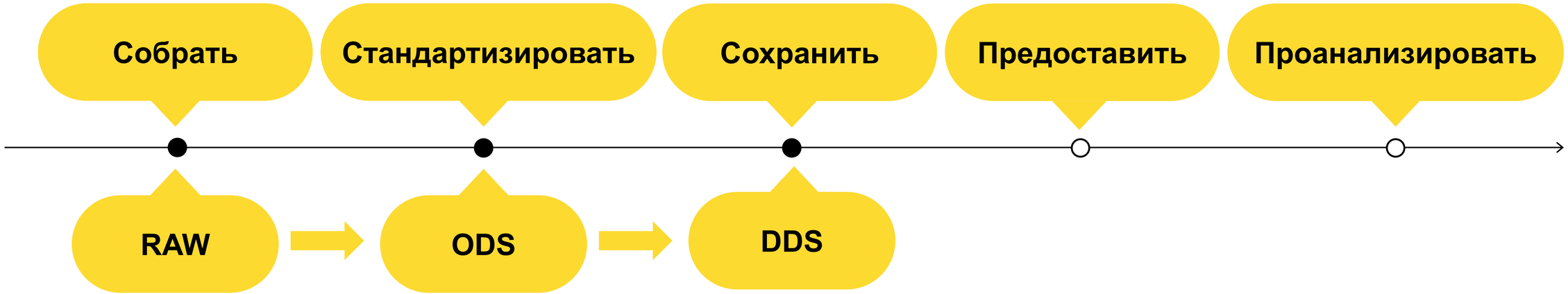
› Хранить операционные данные источника

Задачи

› Сформировать набор сущностей источника иразложить данные сущностям

› Предоставить стандартный интерфейс доступа к данным вне зависимости от особенностей

Архитектура слоев данных



Цель

› Накапливать данные о сущностях доменной модели

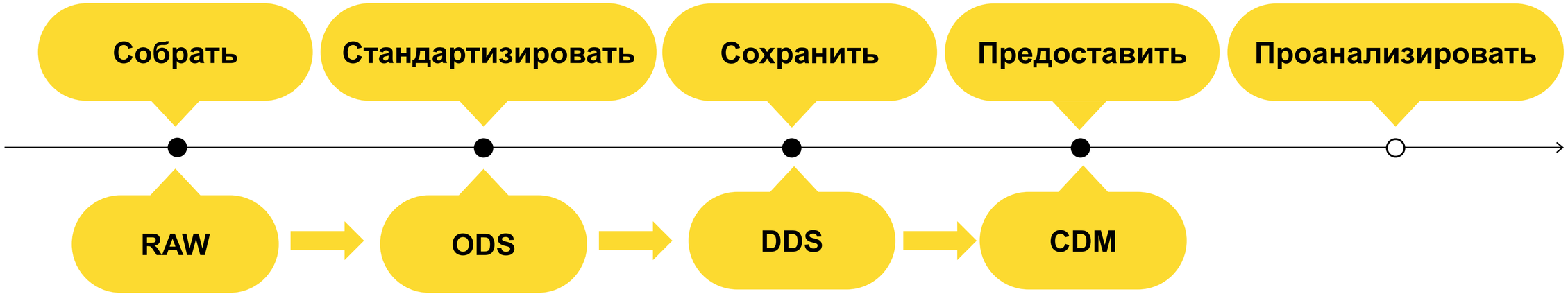
Задачи

› Хранить детальную историю изменений

› Консолидировать данные между источниками

› Предоставлять стандартный интерфейс доступа к сущностям

Архитектура слоев данных



Цель

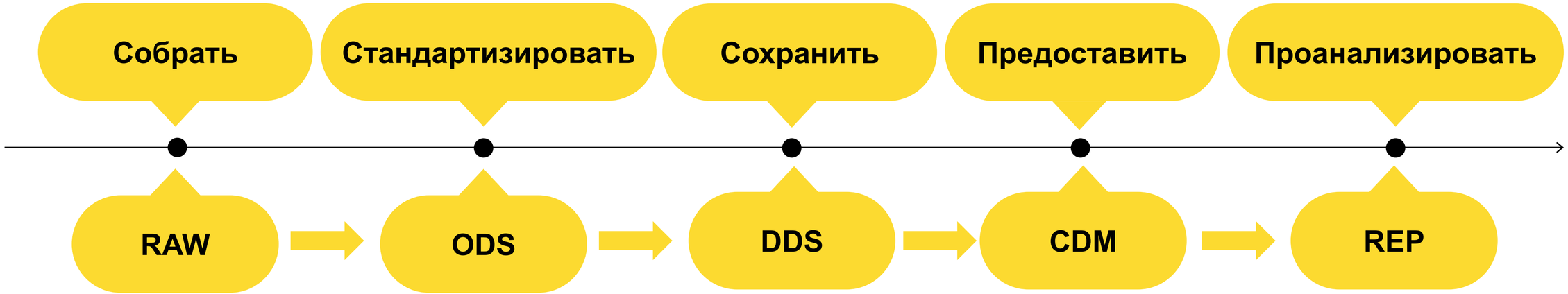
› Предоставлять витрины данных для анализа

Задачи

› Формировать данные в контексте бизнес-потребностей

› Оптимизировать доступ на чтение

Архитектура слоев данных



Цель

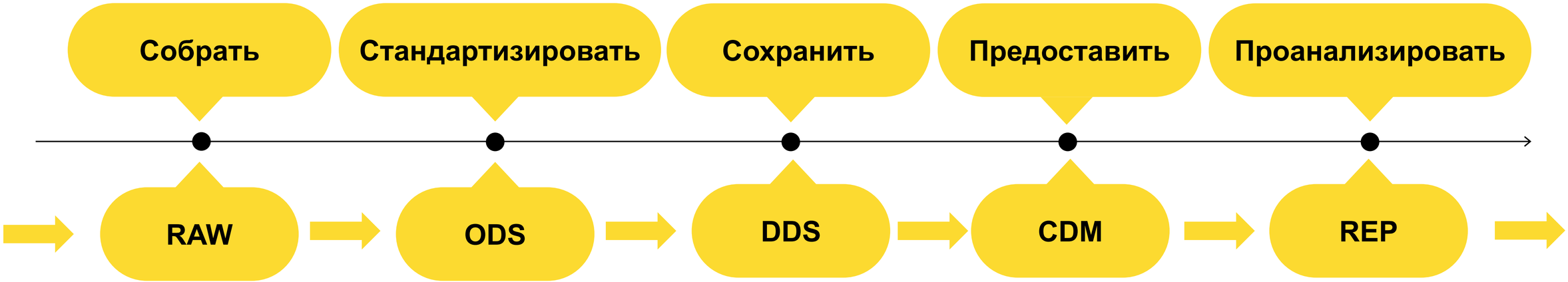
› Хранить отчетные срезы

Задачи

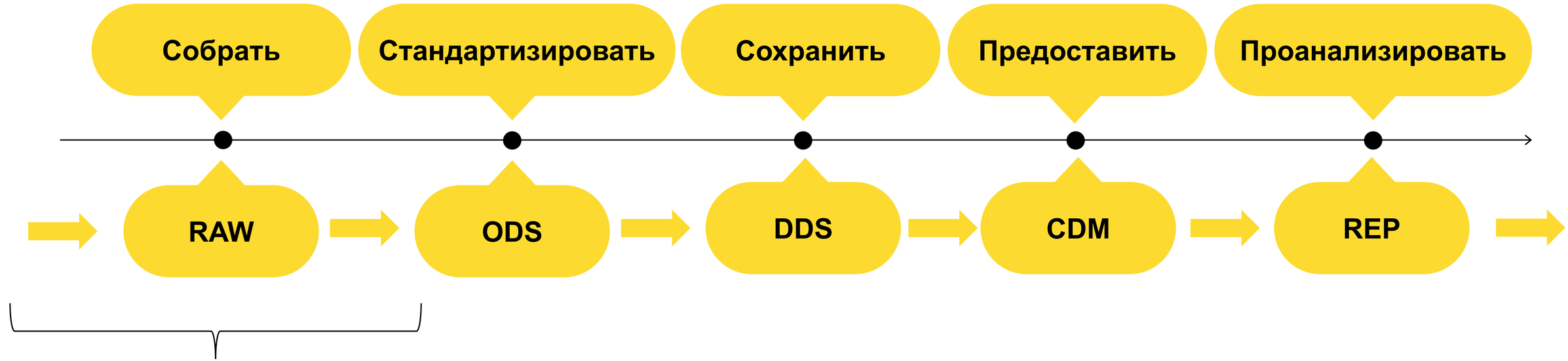
› Формировать данные в контексте бизнес-потребностей

› Готовить агрегированные отчеты

Архитектура слоев данных



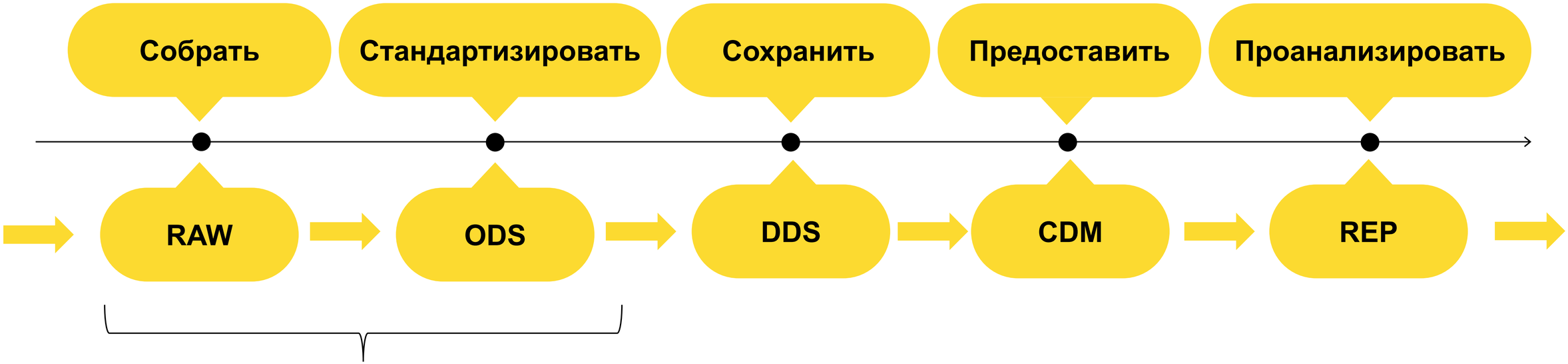
Архитектура слоев данных



Сервис репликации данных

- Забирает инкременты и снепшоты с источников различных типов
- Преобразовывает данные вустойчивый к изменениям формат

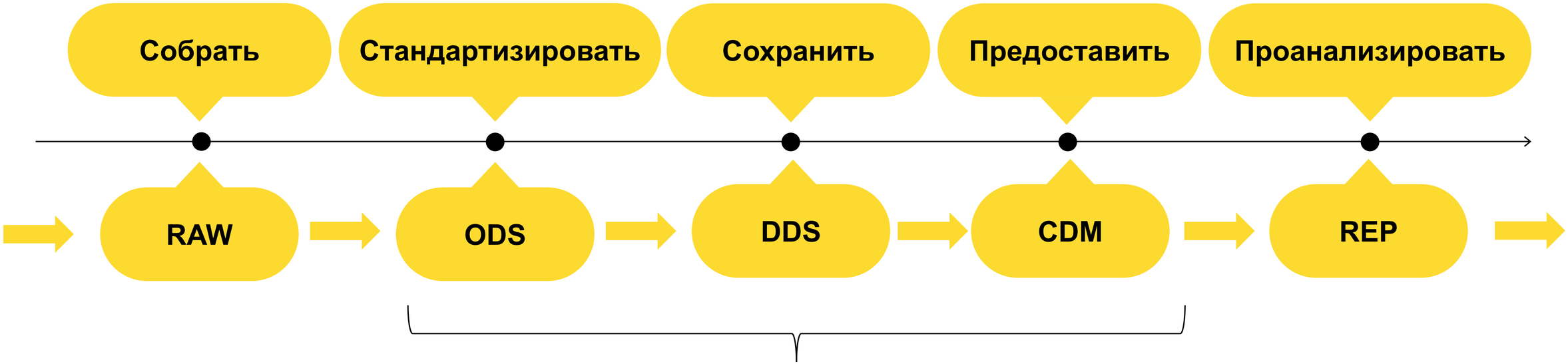
Архитектура слоев данных



Data Lake

› Полуструктурированные данные
› Каркас MapReduce
› Аналоги экосистемы hadoop

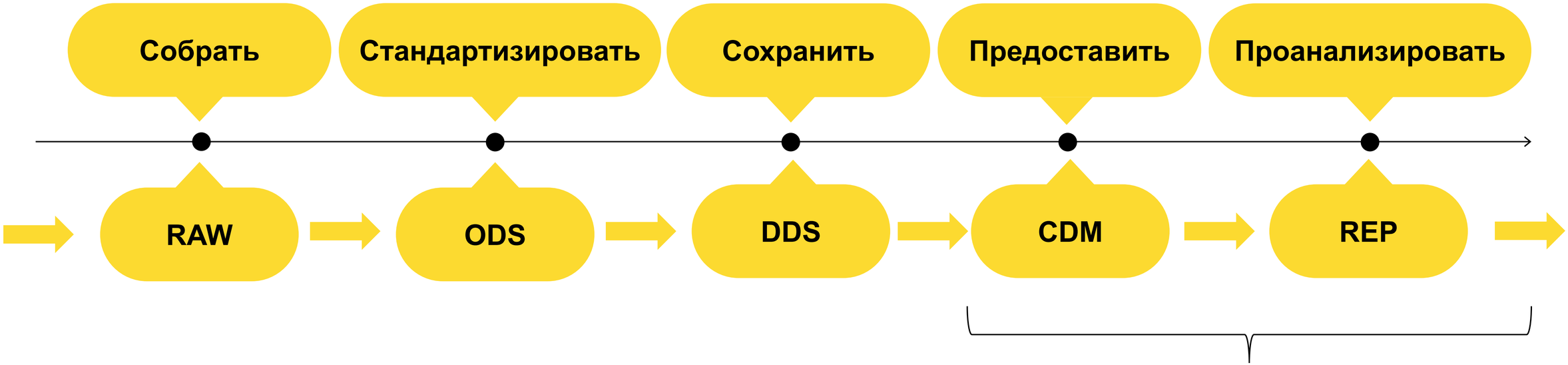
Архитектура слоев данных



Data warehouse

- Различные ad-hoc запросы
- Большое количество join
- Малое время отклика

Архитектура слоев данных

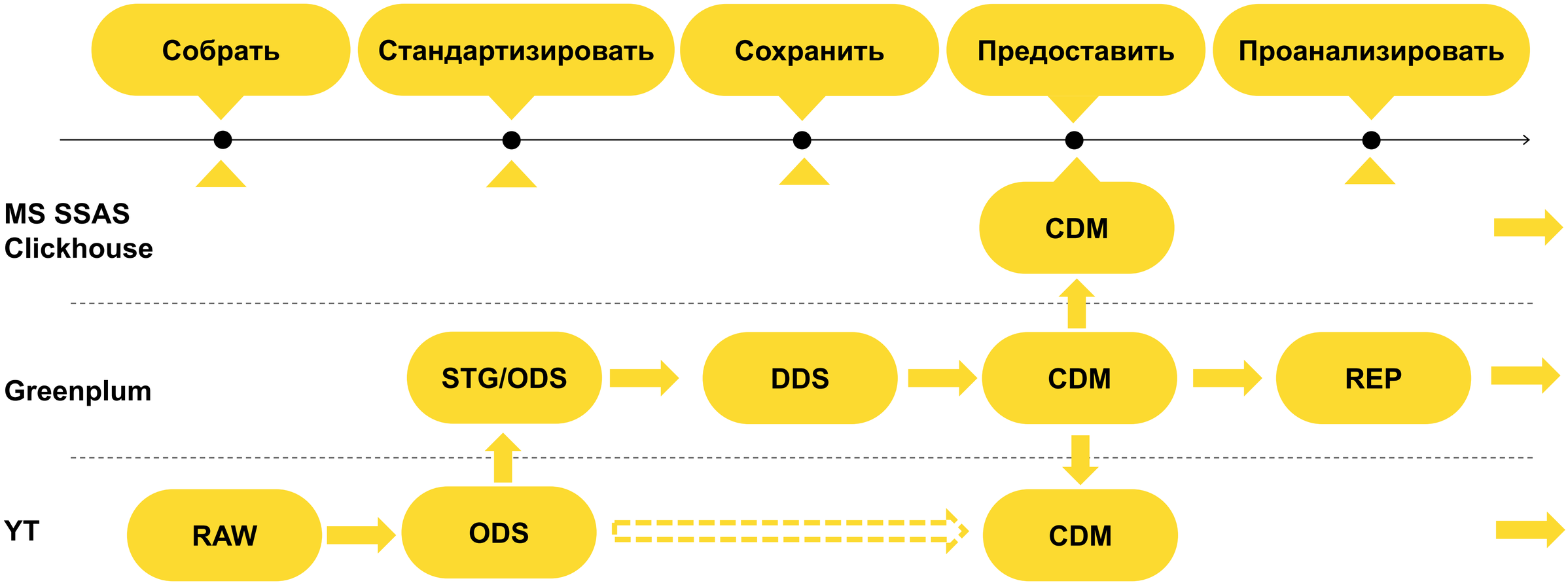


Системы анализа и визуализации

› Кубы данных

› Отчеты и дашборды

Архитектура слоев данных



Детальный слой

Детальный слой – ключевой для построения доменной модели

- Хранит историю изменений сущностей и связей между ними
- Отвечает за консолидацию данных между источниками
- Устойчив к изменению в бизнесе (?)
- Модульный и масштабируемый (?)

Greenplum

Подходы к проектированию

Никакого

- Денормализация до 1НФ
- Можно использовать без подготовки
- Неустойчиво к изменениям
- Дублирование информации
- Нет join

Подходы к проектированию

Никакого

- Денормализация до 1НФ
- Можно использовать без подготовки
- Неустойчиво к изменениям
- Дублирование информации
- Нет join

Звезда и снежинка

- Нормализация до 3НФ
- Можно использовать с минимальной подготовкой
- Неудобно перестраивать
- Минимальное дублирование информации
- Приемлемое количество join

Подходы к проектированию

Никакого

- Денормализация до 1НФ
- Можно использовать без подготовки
- Неустойчиво к изменениям
- Дублирование информации
- Нет join

Звезда и снежинка

- Нормализация до 3НФ
- Можно использовать с минимальной подготовкой
- Неудобно перестраивать
- Минимальное дублирование информации
- Приемлемое количество join

Data Vault

- Строгая нормализация
- Нельзя использовать без подготовки
- Не надо перестраивать (с ограничениями)
- Минимальное дублирование информации
- Большое количество join

Anchor modeling

- Ультра строгая нормализация
- Нельзя использовать без подготовки
- Не надо перестраивать
- Нет дублирования информации
- Ультра количество join

Подходы к проектированию

сложно эксплуатировать, просто внесения изменений

Никакого	Звезда и снежинка	Data Vault	Anchor modeling
› Денормализация до 1НФ	› Нормализация до 3НФ	› Строгая нормализация	› Ультра строгая нормализация
› Можно использовать без подготовки	› Можно использовать с минимальной подготовкой	› Нельзя использовать без подготовки	› Нельзя использовать без подготовки
› Неустойчиво к изменениям	› Неудобно перестраивать	› Не надо перестраивать (с ограничениями)	› Не надо перестраивать
› Дублирование информации	› Минимальное дублирование информации	› Минимальное дублирование информации	› Нет дублирования информации
› Нет join	› Приемлемое количество join	› Большое количество join	› Ультра количество join

легко эксплуатировать, сложно вносить изменения

I-2

Data Vault vs Anchor Modeling

- Сравнение подходов
- Плюсы и минусы
- Что выбрать?

Классическое DWH

Красивые витрины по схеме звезда или снежинка, все лежит аккуратненько в 3НФ

Классическое DWH

Красивые витрины по схеме звезда или снежинка, все лежит аккуратненько в 3НФ

По waterfall сперва проанализируем требования, потом подумаем, потом согласуем, обмозгуем еще хорошенько, в общем, DWH – это не быстро, это основательно

Классическое DWH

Красивые витрины по схеме звезда или снежинка, все лежит аккуратненько в 3НФ

По waterfall сперва проанализируем требования, потом подумаем, потом согласуем, обмозгуем еще хорошенько, в общем, DWH – это не быстро, это основательно

Но заказчик почему-то не знает, что хочет, и вообще готов смотреть на то, что получается, а потом уточнять, что ему хочется.

Классическое DWH

Красивые витрины по схеме звезда или снежинка, все лежит аккуратненько в 3НФ

По waterfall сперва проанализируем требования, потом подумаем, потом согласуем, обмозгуем еще хорошенько, в общем, DWH – это не быстро, это основательно

Но заказчик почему-то не знает, что хочет, и вообще готов смотреть на то, что получается, а потом уточнять, что ему хочется.

Да и бизнес-модель меняется раз в две недели, нам выживать надо, а не хранилища строить

Классическое DWH

Красивые витрины по схеме звезда или снежинка, все лежит аккуратненько в 3НФ

По waterfall сперва проанализируем требования, потом подумаем, потом согласуем, обмозгуем еще хорошенько, в общем, DWH – это не быстро, это основательно

Но заказчик почему-то не знает, что хочет, и вообще готов смотреть на то, что получается, а потом уточнять, что ему хочется.

Да и бизнес-модель меняется раз в две недели, нам выживать надо, а не хранилища строить

И вообще, мы agile и гибкие, давайте все переделаем…

Классическое DWH

Красивые витрины по схеме звезда или снежинка, все лежит аккуратненько в 3НФ

По waterfall сперва проанализируем требования, потом подумаем, потом согласуем, обмозгуем еще хорошенько, в общем, DWH – это не быстро, это основательно

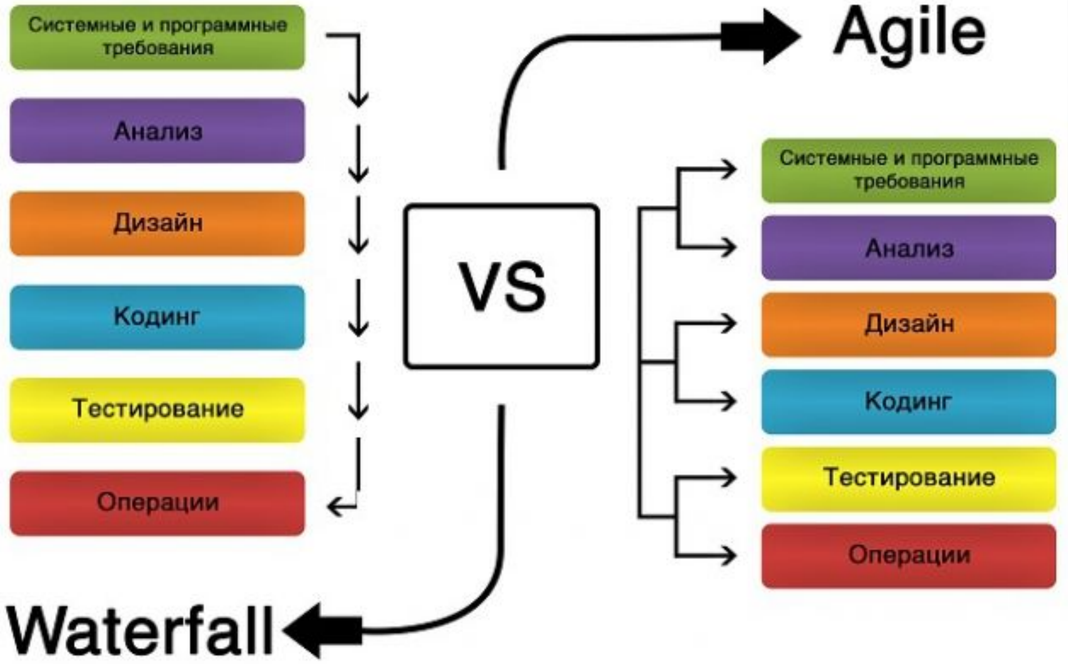
Но заказчик почему-то не знает, что хочет, и вообще готов смотреть на то, что получается, а потом уточнять, что ему хочется.

Да и бизнес-модель меняется раз в две недели, нам выживать надо, а не хранилища строить

И вообще, мы agile и гибкие, давайте все переделаем…

..и желательно завтра результат уже пощупать.

Может ли DWH быть agile?



Да, может – с Data Vault и Anchor Modeling

https://learndatavault.com/

Dan Linstedt

http://www.anchormodeling.com/

Lars Rönnbäck

Да, может – с Data Vault и Anchor Modeling

https://learndatavault.com/

Dan Linstedt

http://www.anchormodeling.com/

Lars Rönnbäck

Обе методологии:

- Повышают градус нормализации выше 3НФ
- Вводят свои типы таблиц и накладывают жесткие ограничение на их использование
- При использовании создают over 9000 таблиц

Да, может – с Data Vault и Anchor Modeling

https://learndatavault.com/

Dan Linstedt

http://www.anchormodeling.com/

Lars Rönnbäck

Обе методологии:

- Повышают градус нормализации выше 3НФ
- Вводят свои типы таблиц и накладывают жесткие ограничение на их использование
- При использовании создают over 9000 таблиц

Взамен обещают:

- Уменьшить постоянное дублирования данных в с SCD2 от изменения всего одного атрибута
- Избавить от деструктивных изменений, только расширение модели (даже при изменении кардинальности связи, боль классического хранилища)
- Позволить дорабатывать хранилище легко и быстро (мистика)

Data Vault

Data Vault вводит и регламентирует основные типы таблиц: Хаб (Hub), Связь (Link) и Сателлит (Satellite).

- **Хаб (Hub)** хранит сущности
- **Связь (Link)** обеспечивает транзакционную интеграцию между Хабами (связи между сущностями)
- **Сателлит (Satellite)** предоставляет контекст первичного ключа Хаба (атрибуты, описания).
- **Мост (Bridge)** упрощает соединение данных через несколько связей
- **PIT (point in time)** упрощает получение информации из саттелитов одной сущности с разной частотой обновления

Data Vault

Data Vault вводит и регламентирует основные типы таблиц: Хаб (Hub), Связь (Link) и Сателлит (Satellite).

Хаб

Хабы (Hub) являются отдельными таблицами, содержащими как минимум уникальный список бизнес ключей.

Атрибуты Хаба включают:

- Ключ бизнес-сущности из внешней системы
- Суррогатный ключ
- Временная отметка даты загрузки
- Код источника данных

Data Vault

Data Vault вводит и регламентирует основные типы таблиц: Хаб (Hub), Связь (Link) и Сателлит (Satellite).

Хаб

Хабы (Hub) являются отдельными таблицами, содержащими как минимум уникальный список бизнес ключей.

Атрибуты Хаба включают:

- Ключ бизнес-сущности из внешней системы
- Суррогатный ключ
- Временная отметка даты загрузки
- Код источника данных

Линк

Связи (Link) представляет отношения или транзакцию между двумя или более компонентами бизнеса (два или более бизнес ключа).

Атрибуты линка включают:

- Суррогатный ключ (Surrogate Key)
- Ключи Хабов: от 1-го Хаба до N-го Хаба
- Временная отметка даты загрузки
- Код источника данных

Data Vault

Data Vault вводит и регламентирует основные типы таблиц: Хаб (Hub), Связь (Link) и Сателлит (Satellite).

Хаб

Хабы (Hub) являются отдельными таблицами, содержащими как минимум уникальный список бизнес ключей.

Атрибуты Хаба включают:

- Ключ бизнес-сущности из внешней системы
- Суррогатный ключ
- Временная отметка даты загрузки
- Код источника данных

Линк

Связи (Link) представляет отношения или транзакцию между двумя или более компонентами бизнеса (два или более бизнес ключа).

Атрибуты линка включают:

- Суррогатный ключ (Surrogate Key)
- Ключи Хабов: от 1-го Хаба до N-го Хаба
- Временная отметка даты загрузки
- Код источника данных

Саттелит

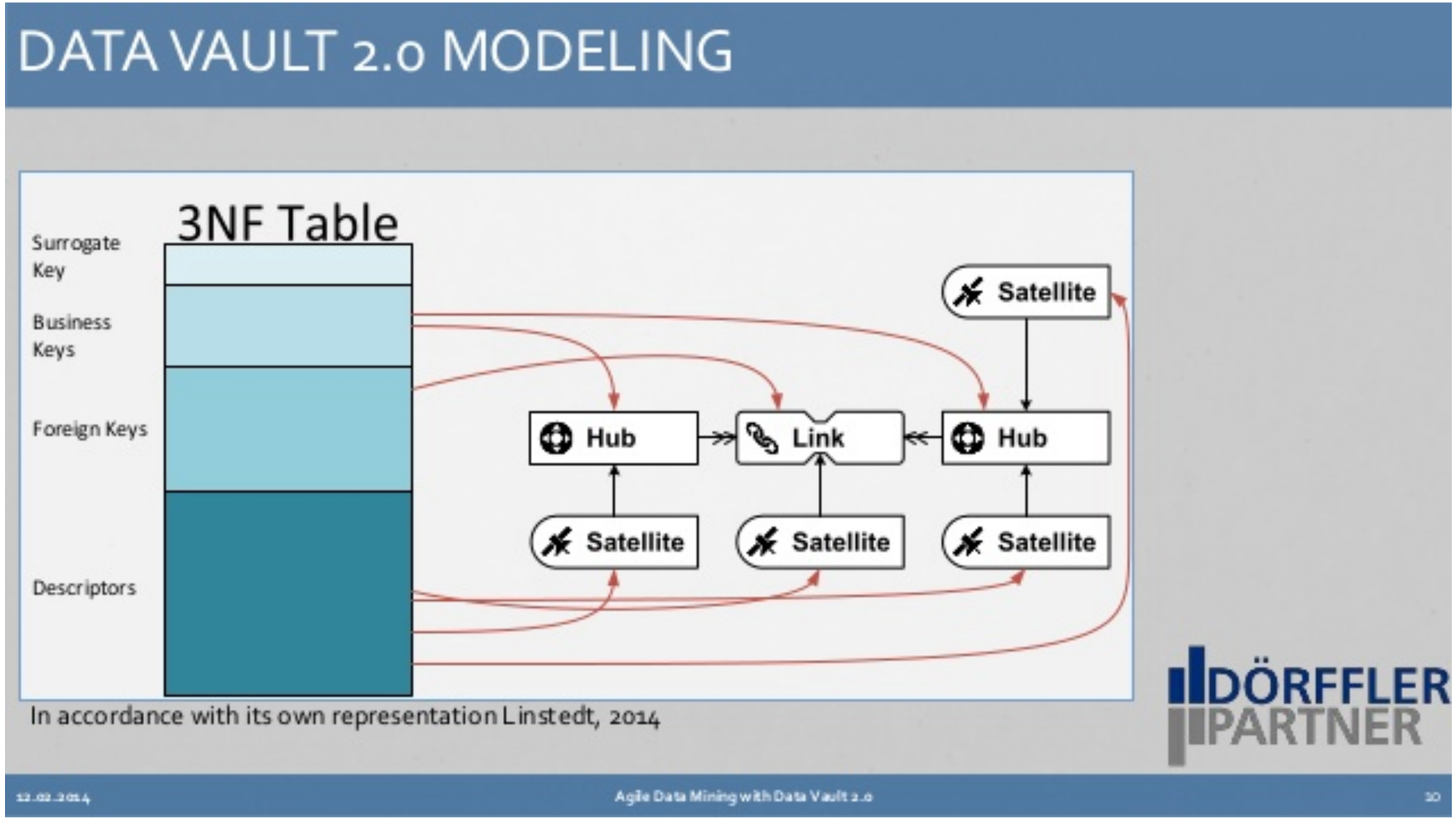
Сателлиты (Satellite) являются контекстной (описательной) информацией ключа Хаба, обычно с историзмом по SCD2.

Атрибуты саттелита включают:

- Первичный ключ Сателлита: Первичный ключ Хаба или первичный ключ Связи
- Даты действия записи (SCD2)
- Временная отметка даты загрузки
- Код источника данных

Data Vault

Data Vault вводит и регламентирует основные типы таблиц: Хаб (Hub), Связь (Link) и Сателлит (Satellite).



https://de.slideshare.net/molschimke/agiles-data-mining-mit-data-vault-20

Anchor Modeling

Якорное моделирование - это технология моделирования гибкой базы данных, подходящая для информации, которая со временем изменяется как по структуре, так и по содержанию.

В методике моделирования используются четыре типа таблиц: якорь, атрибут, связь и узел, - каждый из которых отражает различные аспекты моделируемого домена. Полученные модели могут быть переведены в физические проекты баз данных с использованием формализованных правил. Когда такой перевод сделан, таблицы в реляционной базе данных будут в основном в шестой нормальной форме.

Anchor Modeling

Якорное моделирование - это технология моделирования гибкой базы данных, подходящая для информации, которая со временем изменяется как по структуре, так и по содержанию.

Anchor

Anchor (Якорь) — это существительное, объект реального мира.

Anchor таблица должна хранить

- Суррогатный ключ
- Временная отметка даты загрузки

Tie

Tie (Связь) — это таблица для хранения связей между объектами.

У Tie не может быть атрибутов!

Attribute

Attribute (Атрибут) — это таблица для хранения свойства, атрибута объекта, при этом на каждое свойство ровно одна таблица.

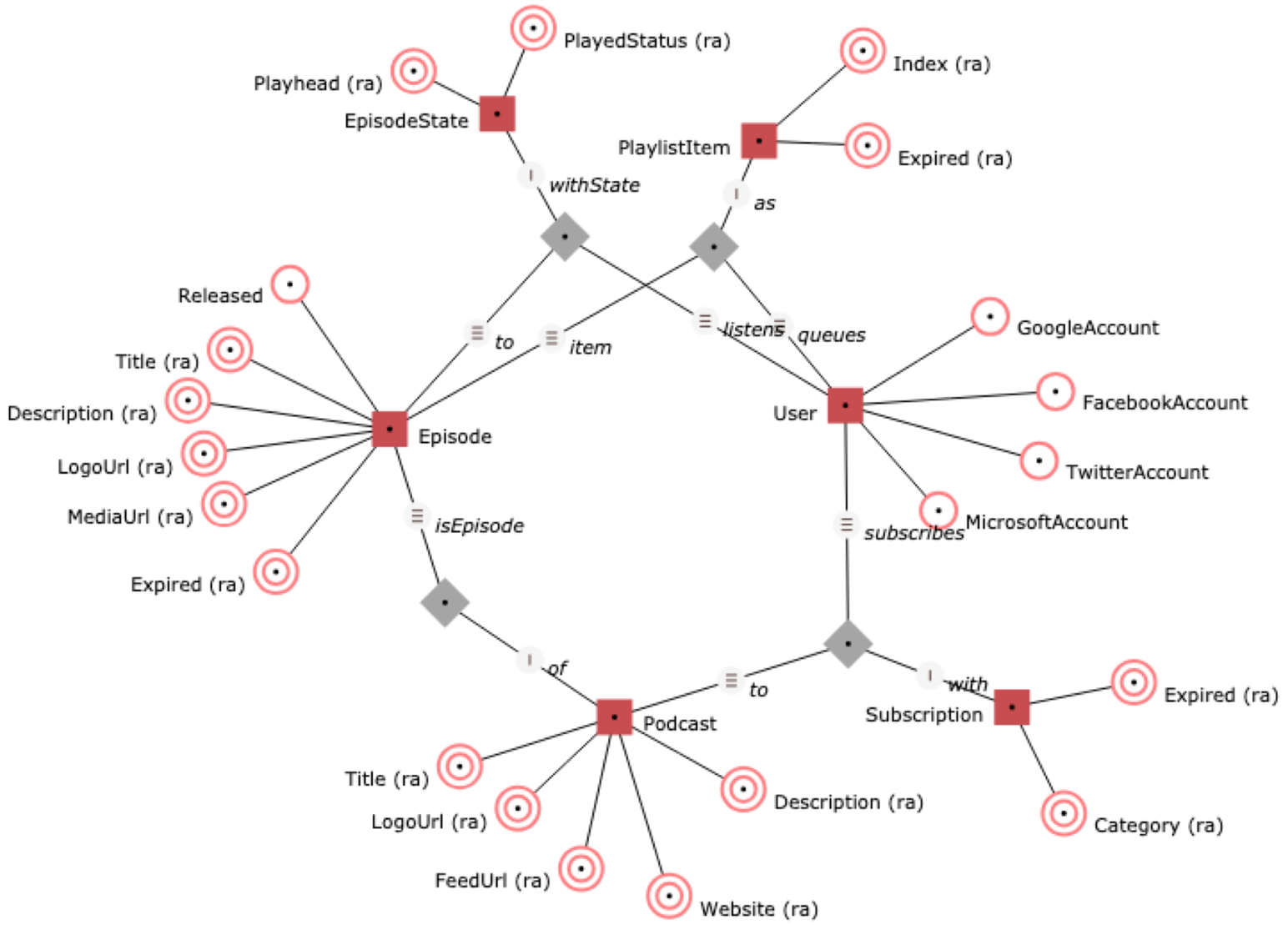
Каждая Attribute-таблица содержит

- Суррогатный ключ
- Временная отметка даты загрузки
- Непосредственно значение

Anchor Modeling

3NF

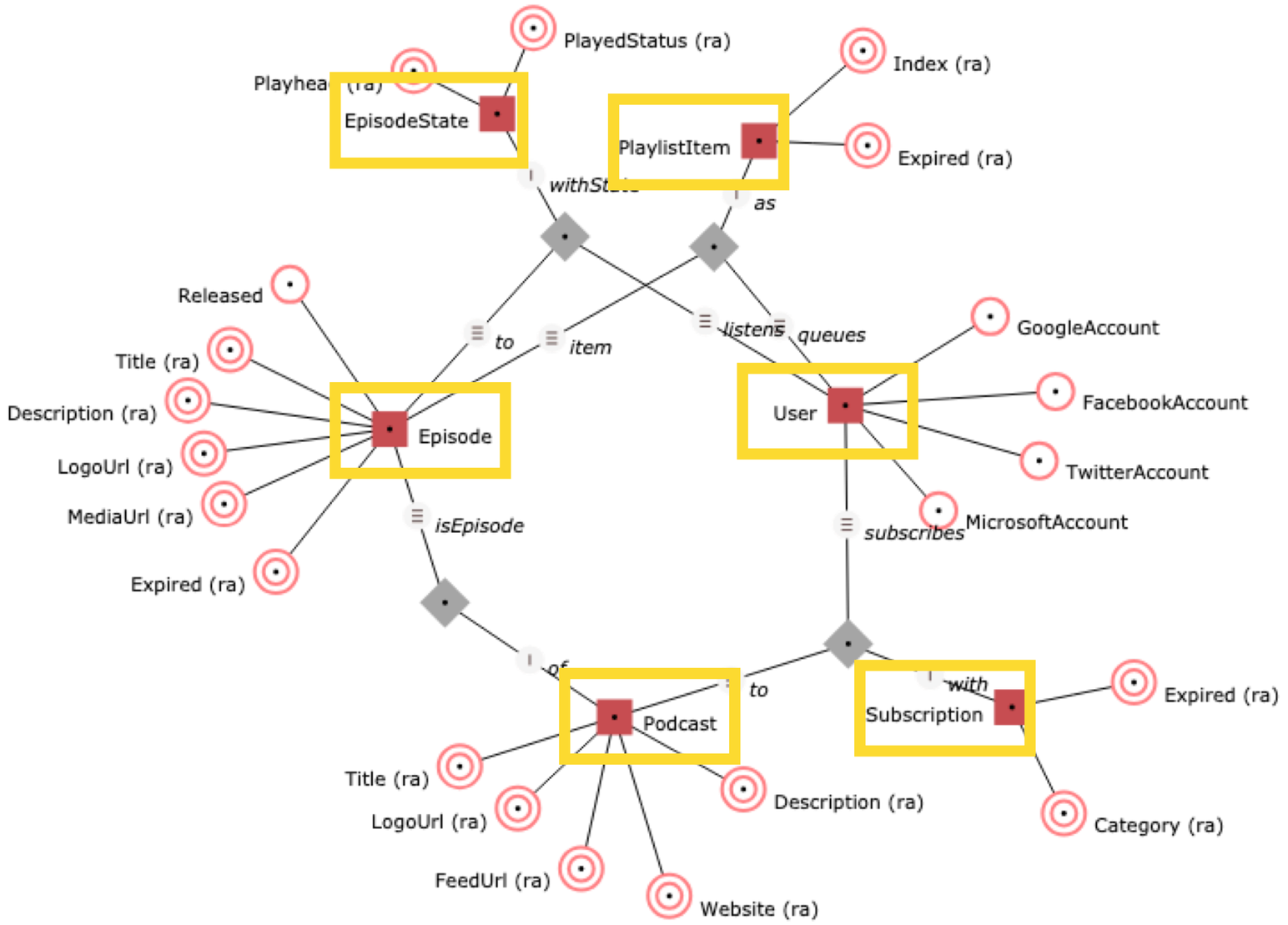
Surrogate key
Business keys
Foreign keys
Descriptors



Anchor Modeling

3NF

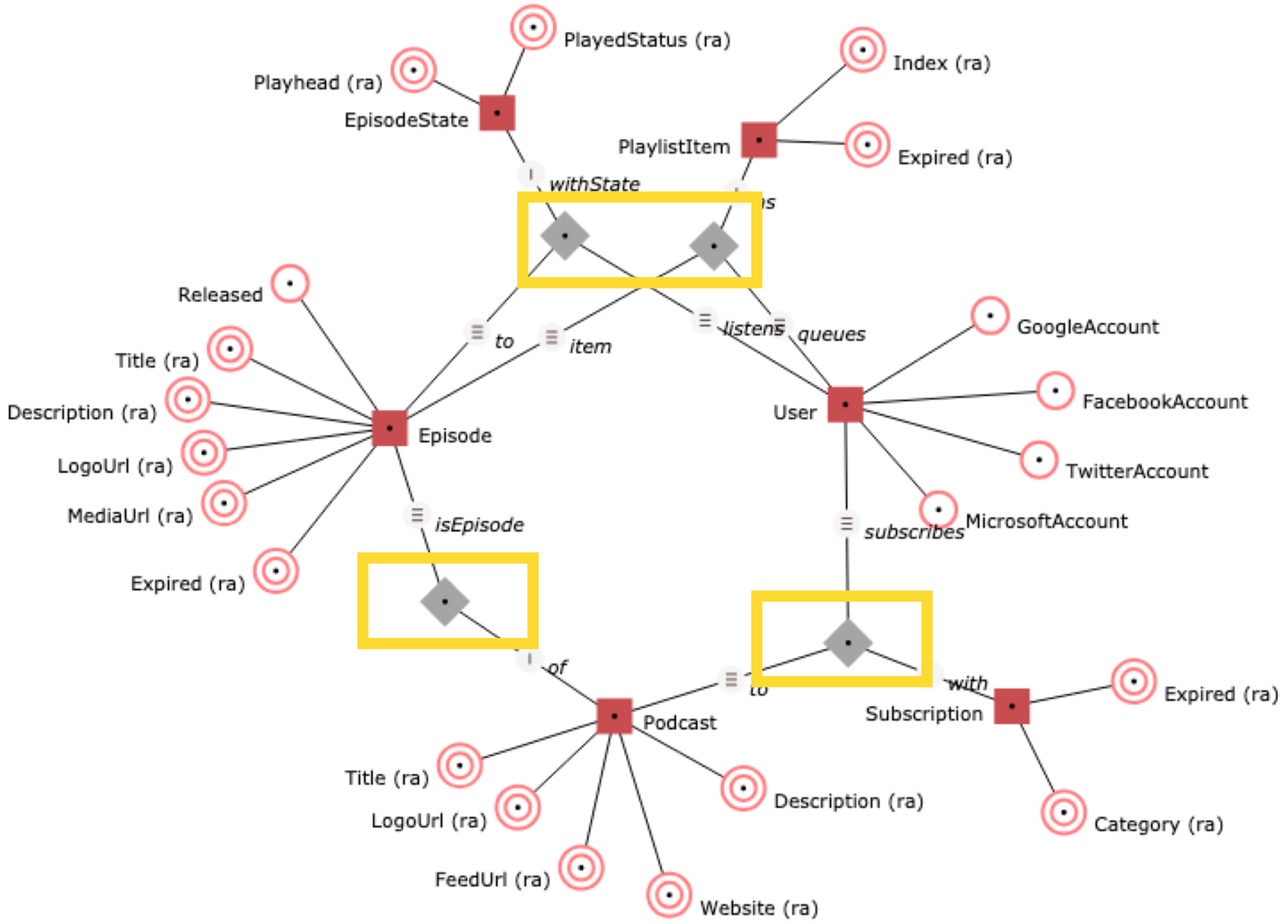
Surrogate key
Business keys
Foreign keys
Descriptors



Anchor Modeling

3NF

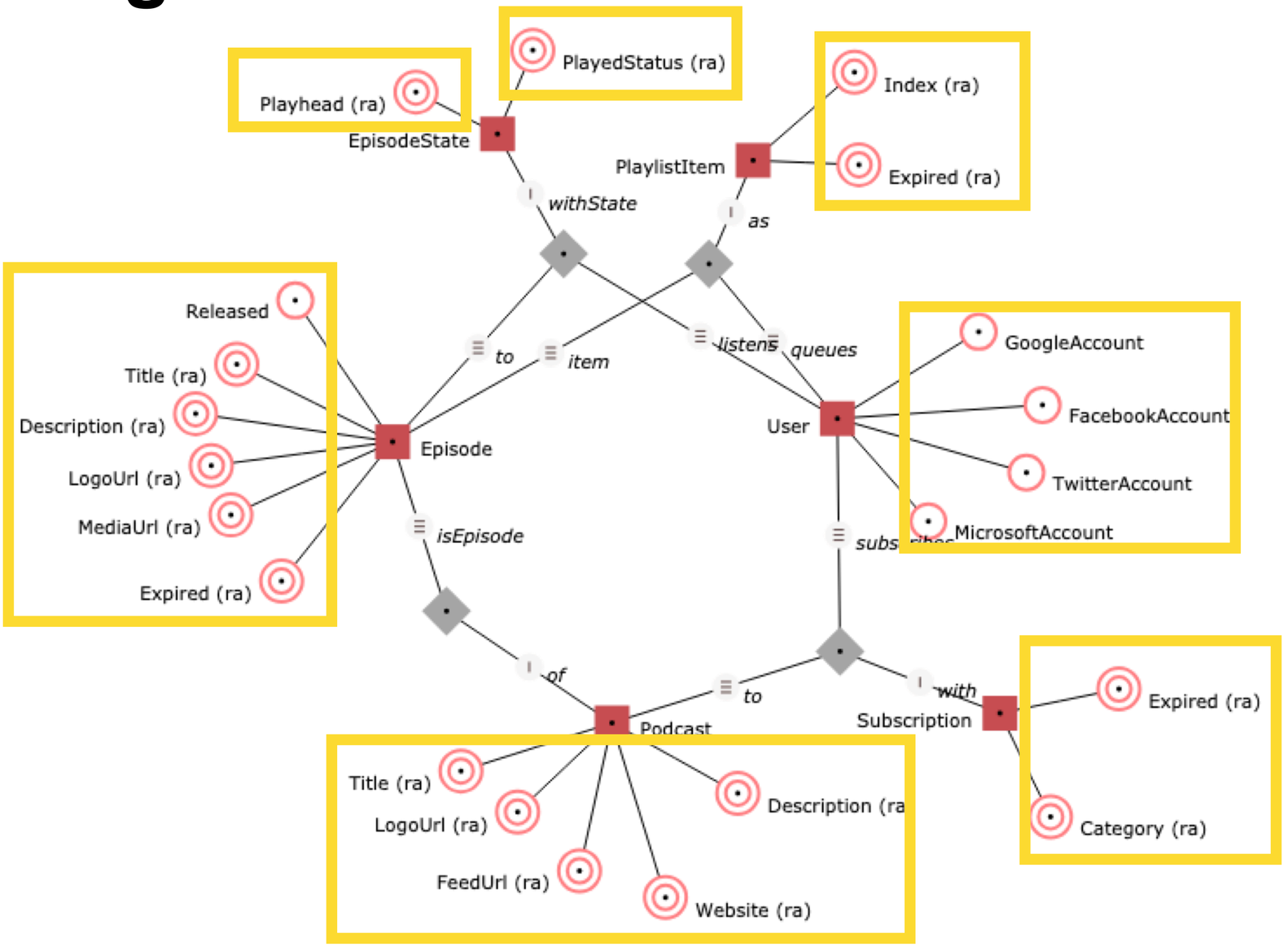
Surrogate key
Business keys
Foreign keys
Descriptors



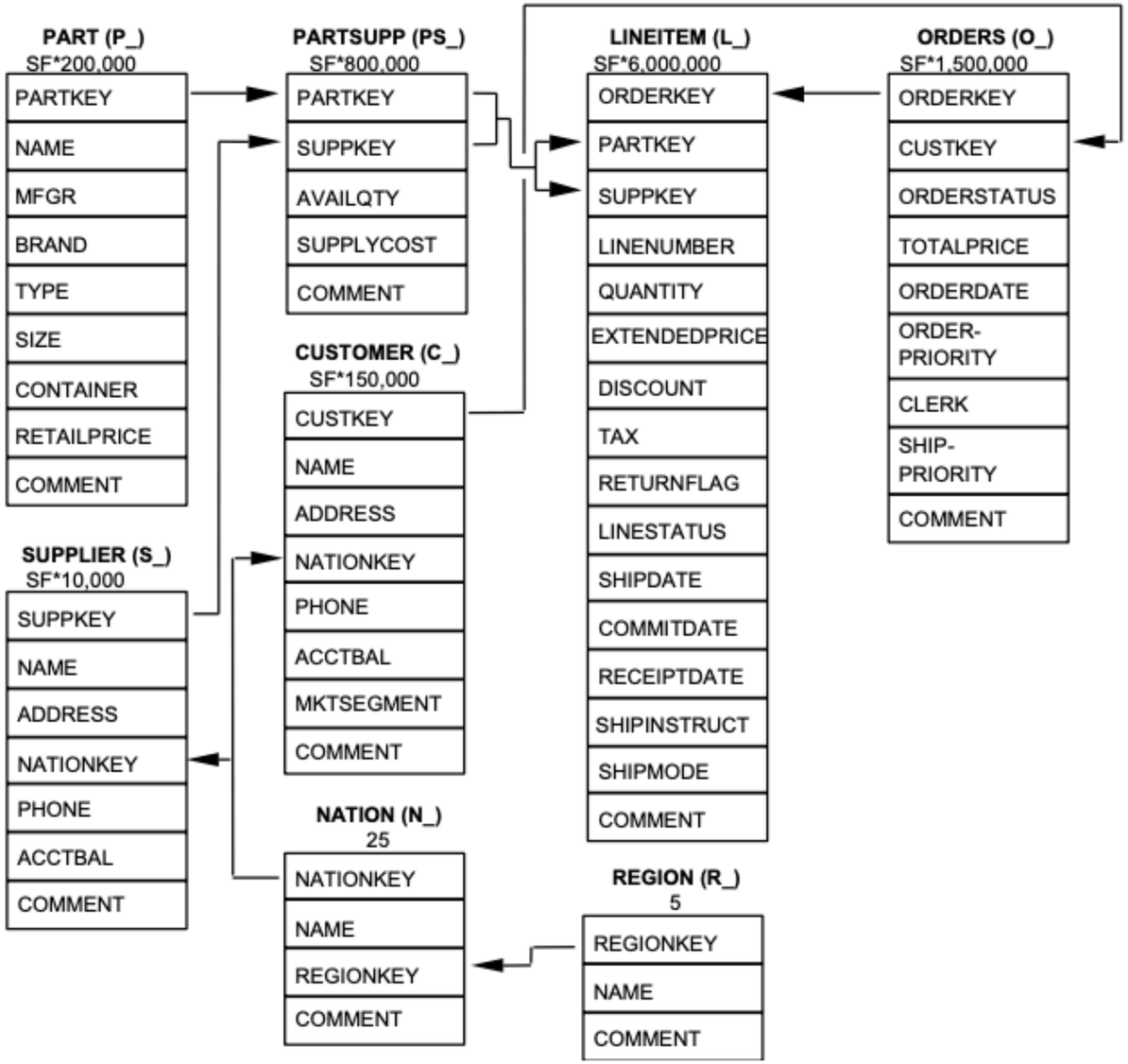
Anchor Modeling

3NF

- Surrogate key
- Business keys
- Foreign keys
- Descriptors

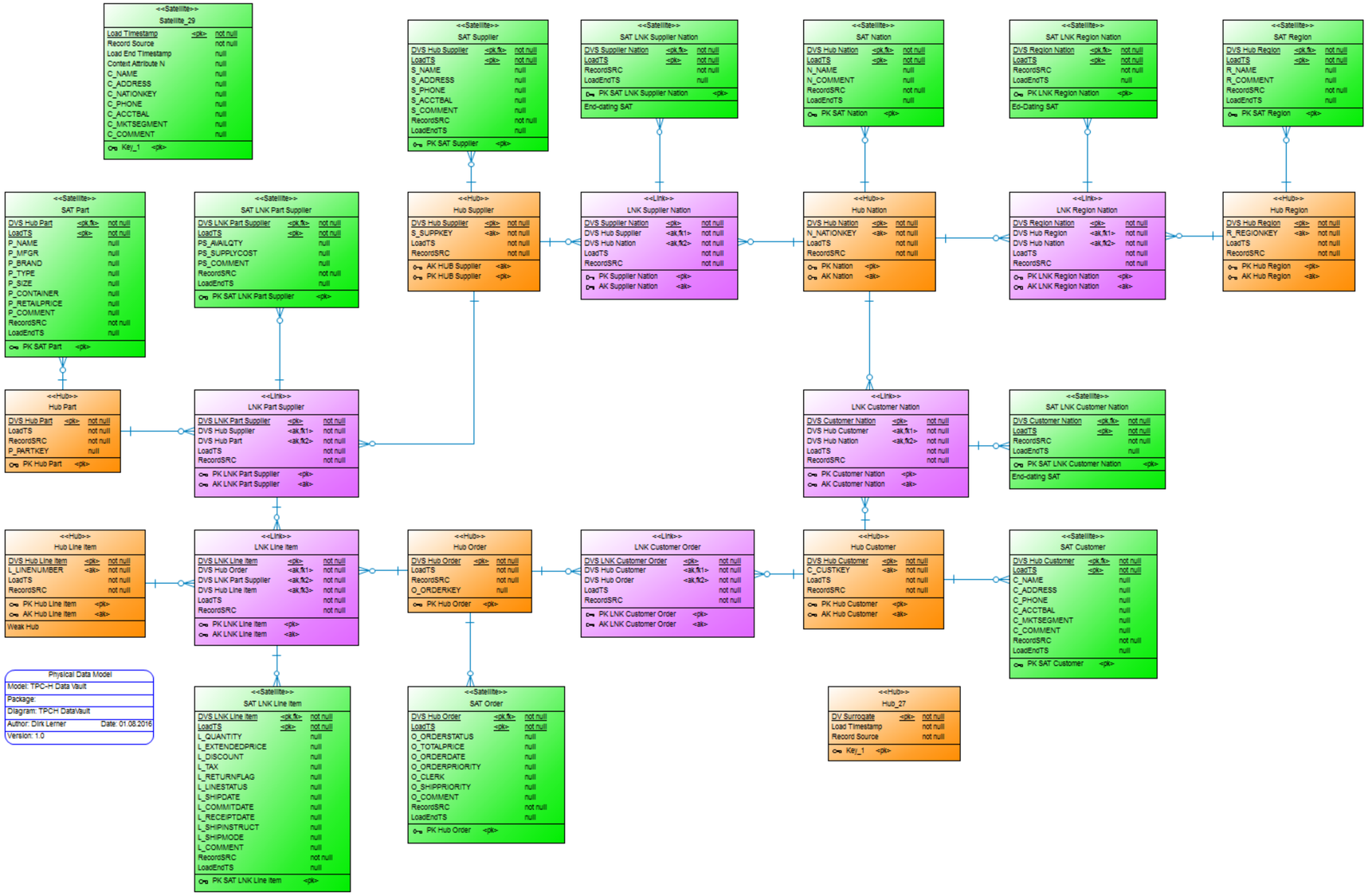


TPC-H



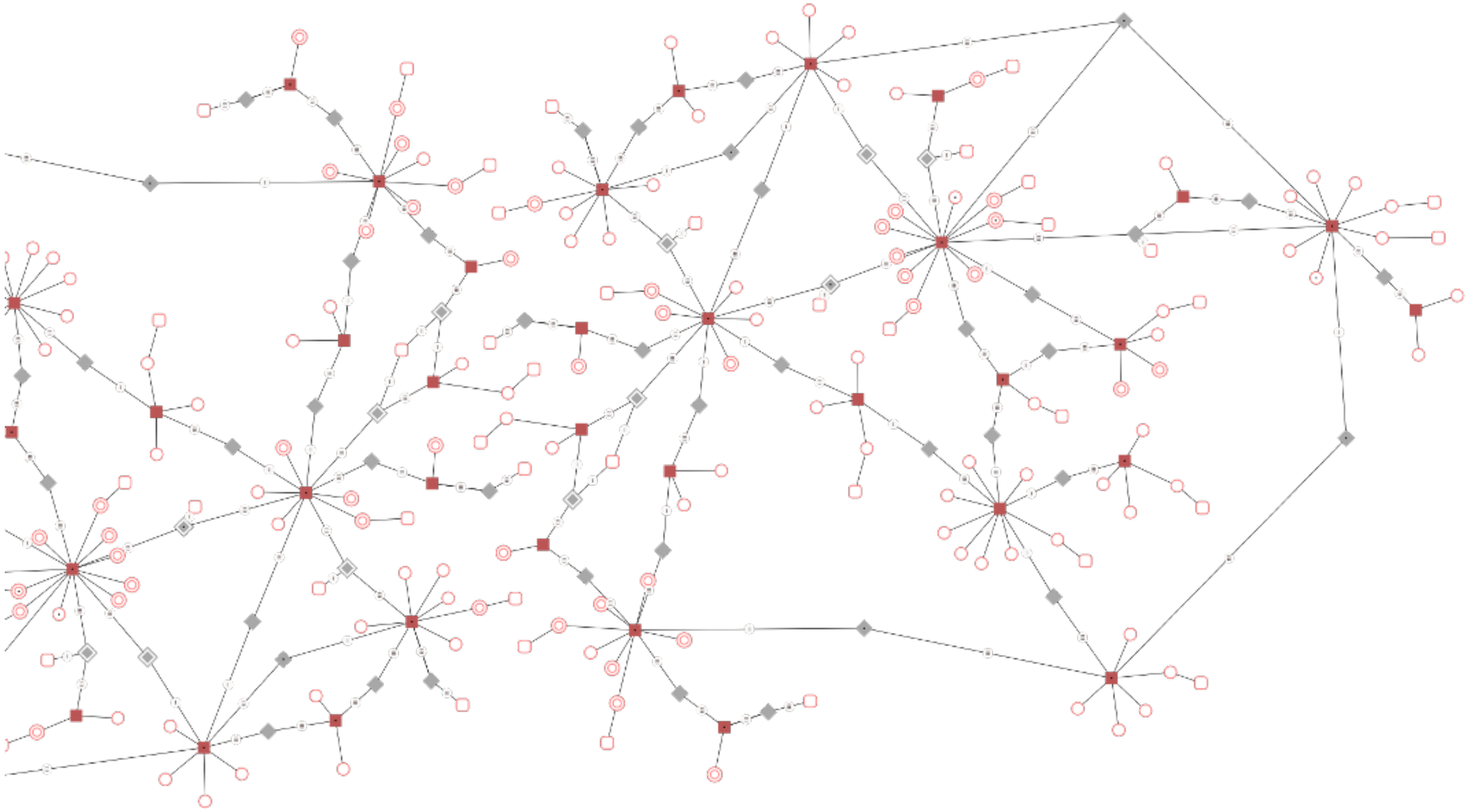
http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf

TPC-H DV



https://tedamoh.com/data-vault/79-data-vault-on-exasol-modeling-and-implementation

TPC-H AM



В чем схожесть и различие?

- На каждую сущность создается **hub** – таблица с бизнес-ключом и суррогатным ключом

- На каждую сущность создается **anchor** – таблица только с суррогатным ключом

В чем схожесть и различие?

- На каждую сущность создается **hub** – таблица с бизнес-ключом и суррогатным ключом
- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования

- На каждую сущность создается **anchor** – таблица только с суррогатным ключом
- Один **атрибут** – одна таблица, да здравствует 6НФ

В чем схожесть и различие?

- На каждую сущность создается **hub** – таблица с бизнес-ключом и суррогатным ключом
- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования
- **Связи** только через отдельные таблицы, на них можно навесить сателлит

- На каждую сущность создается **anchor** – таблица только с суррогатным ключом
- Один **атрибут** – одна таблица, да здравствует 6НФ
- **Связи** только через отдельные таблицы, никаких атрибутов – только хардкор

В чем схожесть и различие?

- На каждую сущность создается **hub** – таблица с бизнес-ключом и суррогатным ключом
- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования
- **Связи** только через отдельные таблицы, на них можно навесить сателлит
- Есть специальные таблицы **Point-in-Time** и **Bridge**

- На каждую сущность создается **anchor** – таблица только с суррогатным ключом
- Один **атрибут** – одна таблица, да здравствует 6НФ
- **Связи** только через отдельные таблицы, никаких атрибутов – только хардкор
- Есть специальная таблица **knot** – статический справочник


Learn Data Vault
Business Information Intelligence
ANCHOR™


Learn Data Vault
Business Information Intelligence
ANCHOR™


Создать свою
модель

Надо выбирать не между, а лучшее!

Highly Normalized Hybrid Model

Data Vault

Яндекс Такси

Anchor modeling

I-3

hNhM 101

- Разделение логического и физического моделирования
- Бизнес-дата и историзм
- Сокрытие физической модели
- Гибрид DV и AM

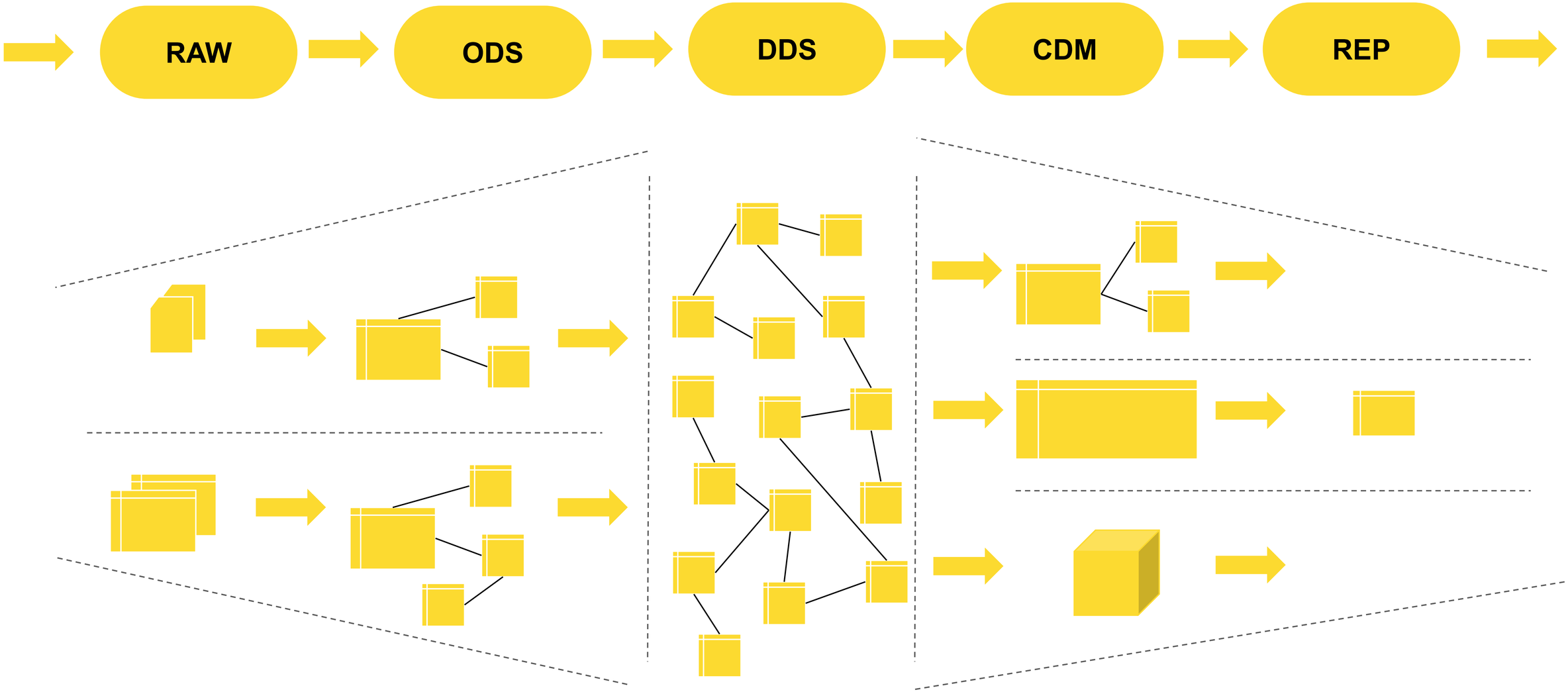
Highly Normalized Hybrid Model (hNhM)

Ключевая идея: выбирать оптимальный формат хранения для каждого конкретного случая

Требования:

- Разделение логического и физического моделирования
- Возможность эмулировать как Data Vault, так и Anchor Modeling, высокая нормализация (вплоть до предельной 6НФ)
- Параллельная загрузка из разных источников, идемпотентность при повторной загрузке
- Устойчивость к изменению в бизнесе, модульность и масштабируемость
- Удобство использования при построении витрин

Highly Normalized Hybrid Model (hNhM)



Логический и физический уровень

Закрепить разделение уровней проектирования в методологии

Логический уровень / ER diagram

- Сущности и их ключи
- Связи между сущностями
- Атрибутивный состав сущностей и необходимость их историзации

Когнитивно сложная часть проектирования, не зависит от СУБД

Физический уровень / DDL Script

- Партицирование сущностей
- Объединение атрибутов в группы
- Дистрибьюция в MPP системах
- Индексы для ускорения запросов

Зависит от СУБД и технических ограничений

Логический и физический уровень

Закрепить разделение уровней проектирования в методологии

Логический уровень / Data Partner

Физический уровень / Data Engineer

Концептуальная модель

Какие направления бизнеса у нас есть?

Какие взаимосвязи между ними?

Какими системами они пользуются?

Логическая модель

Какие сущности у нас есть?

Какие взаимосвязи между ними?

Как часто будут изменяться атрибуты?

Физическая модель

Как хранить атрибуты?

Нужны ли партиции?

Нужно ли закрытие SCD2?

ETL-процесс

Как обеспечить расчет данных по инкременту?

Как правильно пересчитать историю?

Платформа / Core Developer

Как сделать инструментарий по работе с данными удобнее?

Сущность (Entity)

Базовый кубик любого описания домена

Логический уровень

Сущности описываются:

- Именованием и комментарием
- Бизнес-ключом: одно или несколько полей, определяющих сущность
- Набором атрибутов

Атрибуты описываются:

- Именованием и комментарием
- Типом сущности
- Необходимостью хранить историю по атрибуту

Физический уровень

Логическая сущность состоит из:

- Hub – таблица, содержащая техническую информацию и суррогатный ключ (хеш от бизнес-ключа)
- Attribute – таблица, содержащая информацию об атрибуте; может содержать историю (SCD2) или нет
- Group – таблица, содержащая информацию о нескольких атрибутах; может содержать историю (SCD2) или нет; все атрибуты приходят их одного источника и имеют один тип историзма

Связь (Relation)

Связь базовых кубиков

Логический уровень

Связи описываются:

- Списком связанных сущностей
- Типом каждой из связи (M-M, 1-M)

У связи не может быть атрибутов – если есть необходимость добавить в связь атрибут, то необходимо выделить сущность.

Физический уровень

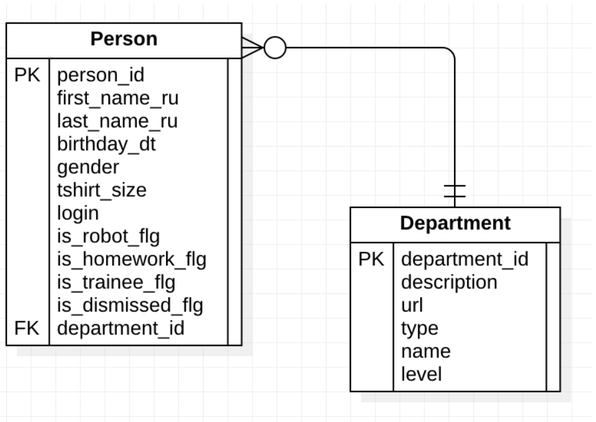
Физически связь – это таблица M-M, которая содержит:

- Суррогатные ключи всех входящих в связь сущностей
- Поля историзма (SCD2)
- Поля - партиции сущностей

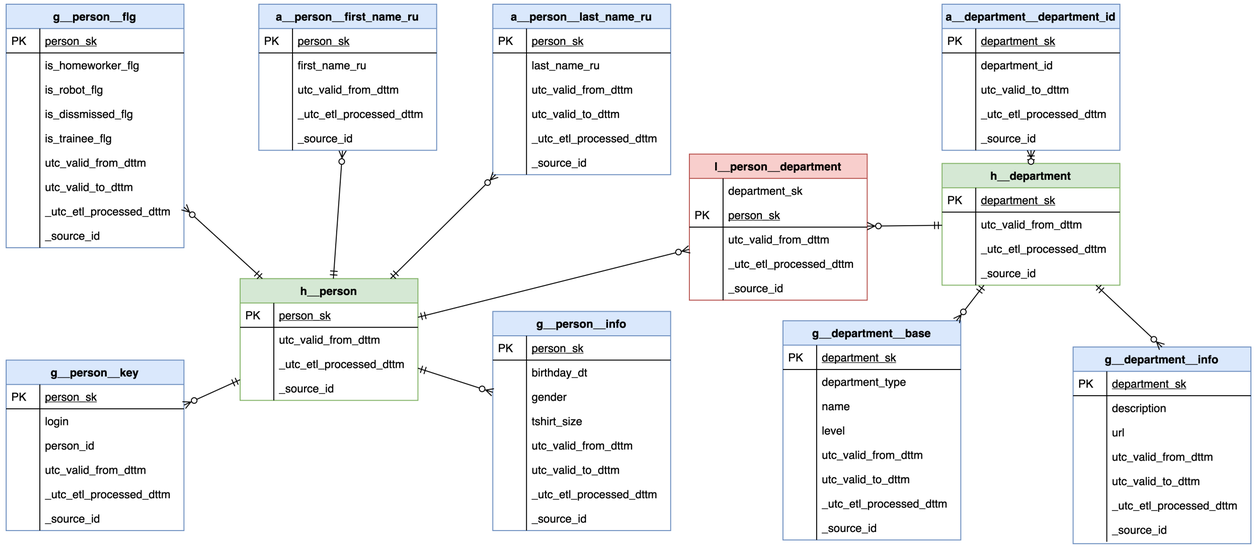
Логический и физический уровень

Закрепить разделение уровней проектирования в методологии

Логический уровень



Физический уровень



Сокрытие физической реализации

Предоставить интерфейс работы с сущностями и связями, сокрыв реализацию

Загрузка данных

- Идемпотентность – если мы загружаем одни и те же данные несколько раз, то результат не меняется
- Параллельность – все таблицы физической реализации могут грузиться параллельно
- Сокрытие сложности SCD2 загрузки

Важно! Во всех загружаемых данных должна быть бизнес-дата, атрибуты одной группы грузятся из одного источника

Построение витрин

- Работа с сущностями на логическом уровне, а не с таблицами на физическом
- Сокрытие сложности исторической обработки записей
- Максимизация простоты работы с инкрементом

Highly Normalized Hybrid Model (hNhM)

- **Закрепить разделение логического и физического проектирования в методологии**
- **Скрыть физическую модель при загрузке данных и использовании модели**
- **Выбирать оптимальный формат хранения для каждого конкретного случая**

Highly Normalized Hybrid Model (hNhM)

Закрепить разделение логического и физического проектирования в методологии

Скрыть физическую модель при загрузке данных и использовании модели

Выбирать оптимальный формат хранения для каждого конкретного случая

- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования
- Есть специальные таблицы **Point-in-Time** и **Bridge**

- На каждую сущность создается **anchor** – таблица с суррогатным ключом
- **Связи** только через отдельные таблицы, никаких атрибутов – только хардкор

Highly Normalized Hybrid Model (hNhM)

Закрепить разделение логического и физического проектирования в методологии

Скрыть физическую модель при загрузке данных и использовании модели

Выбирать оптимальный формат хранения для каждого конкретного случая

- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования
- Есть специальные таблицы **Point-in-Time** и **Bridge**

- На каждую сущность создается **anchor** – таблица с суррогатным ключом
- **Связи** только через отдельные таблицы, никаких атрибутов – только хардкор

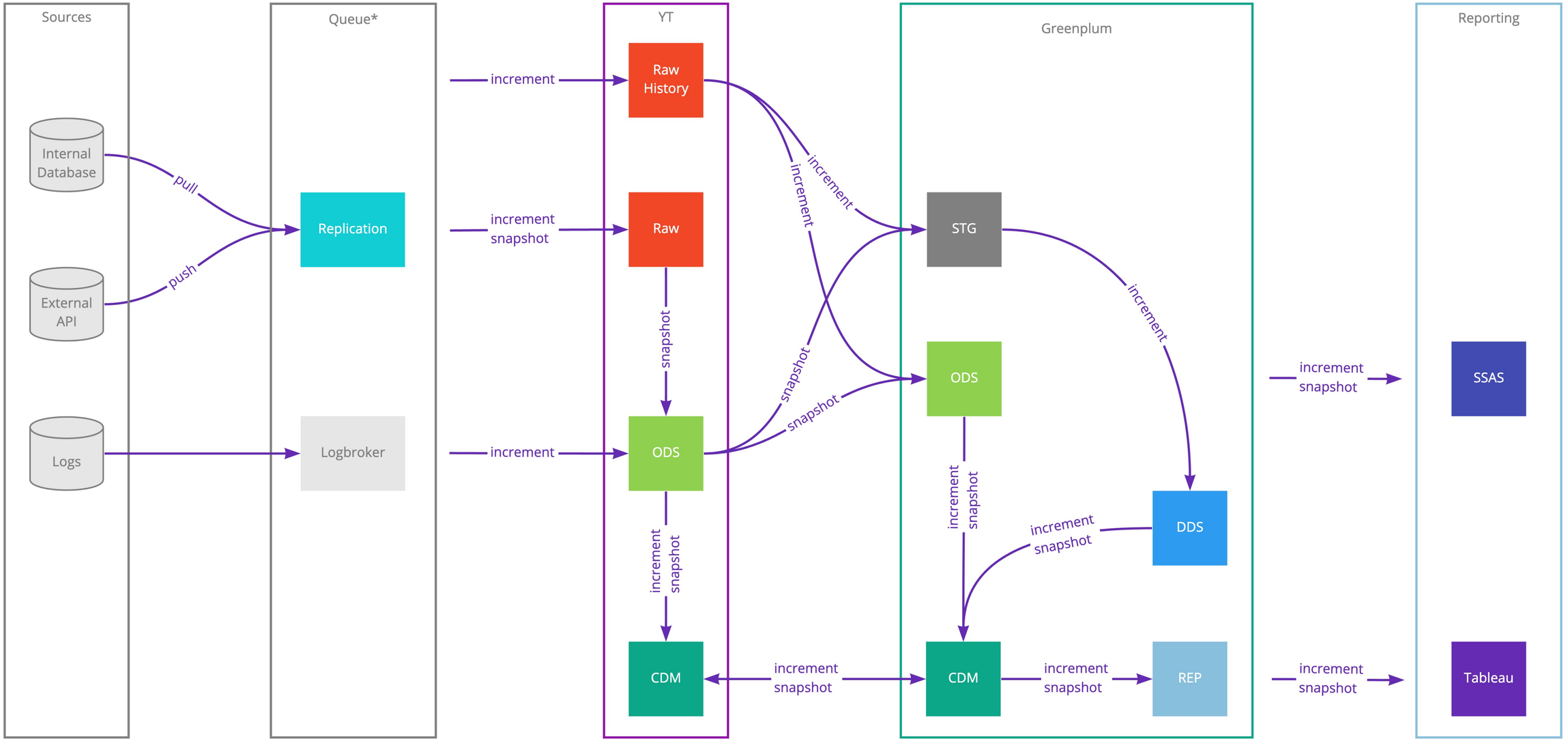
Невозможно управлять без framework

II-1

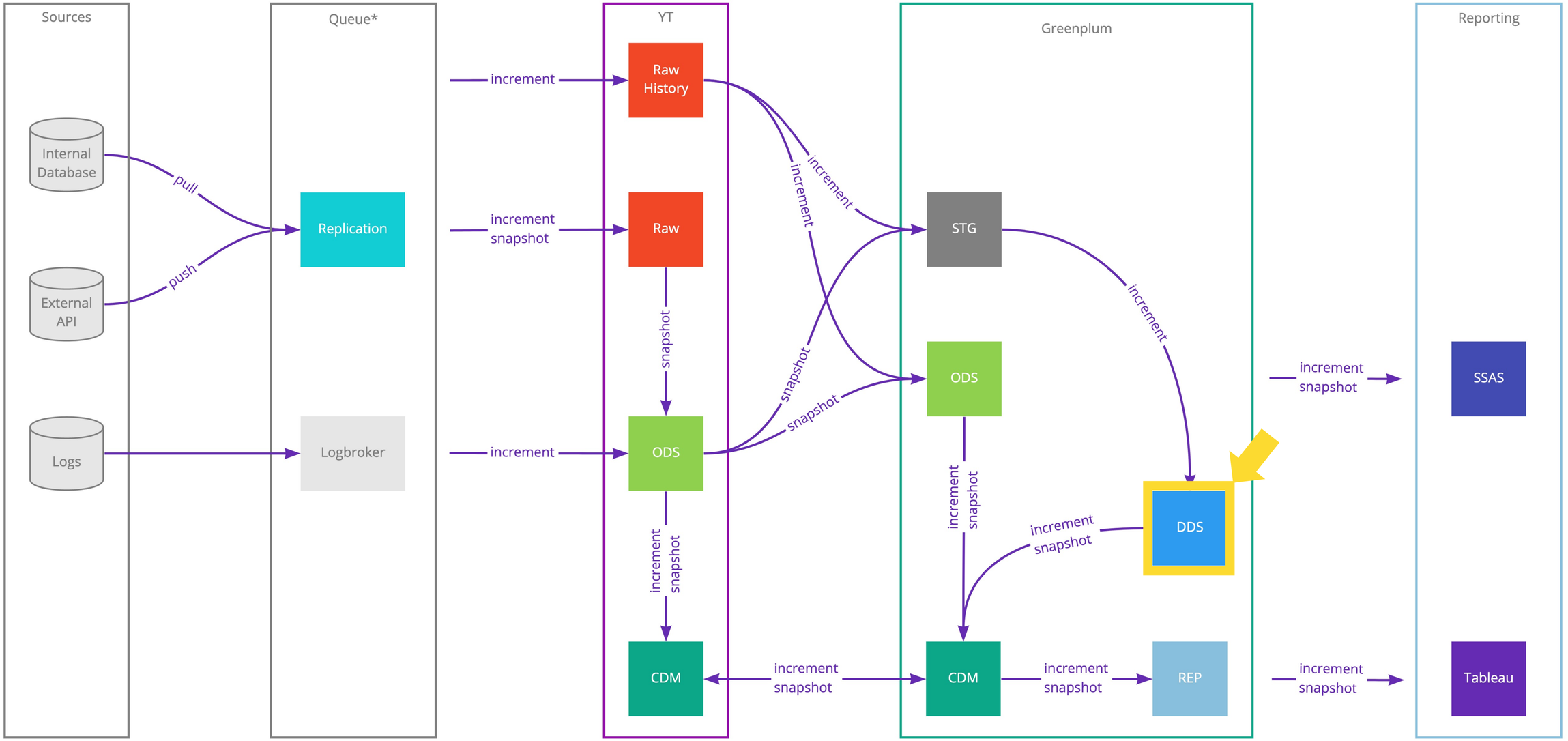
hNhM.Framework

- Описание метаданных
- Генерация DDL
- Загрузка данных

DMP – Data Management Platform



DMP – Data Management Platform



Объявление сущности

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```

Объявление сущности

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```

Объявление сущности

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```

Объявление сущности

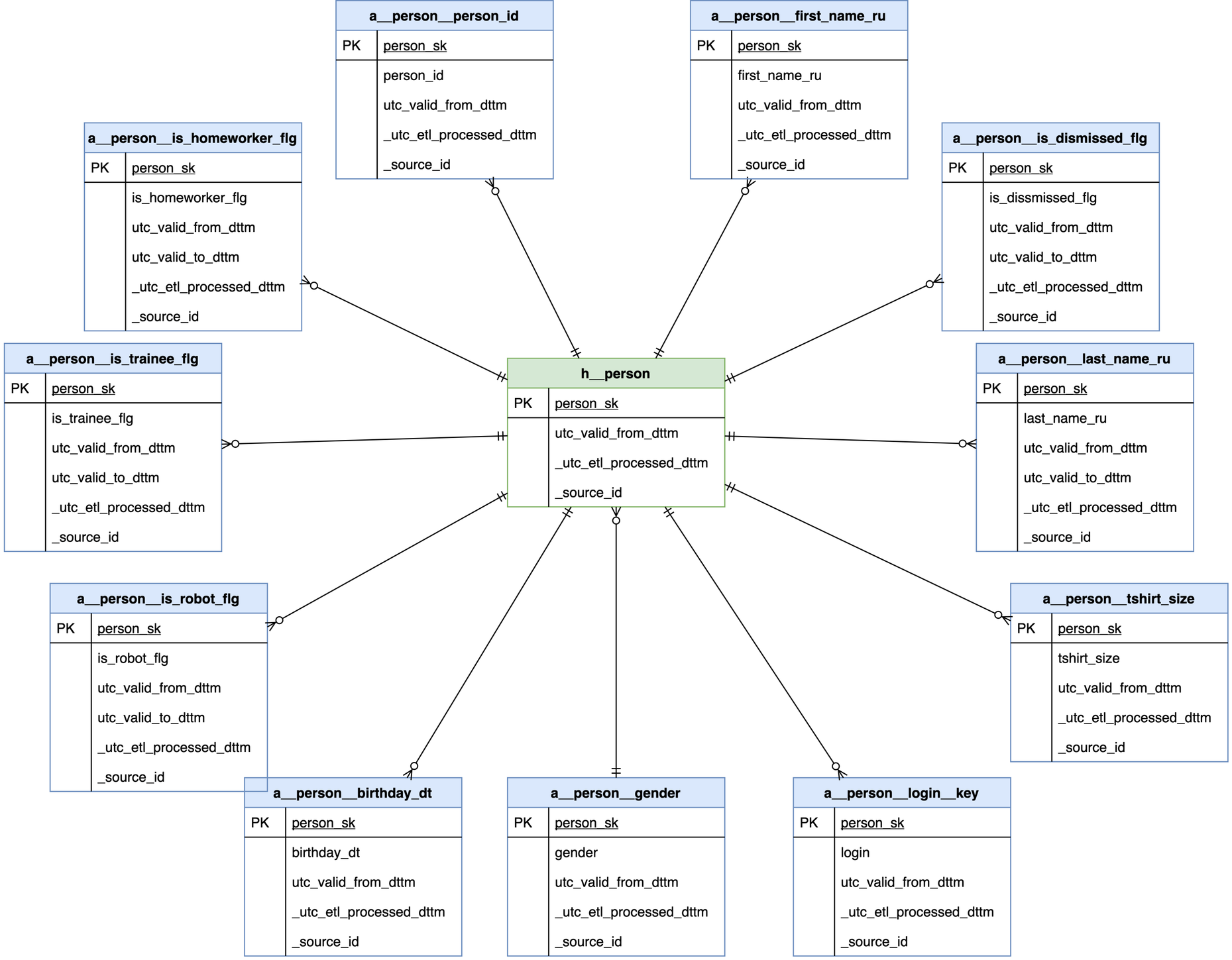
```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```

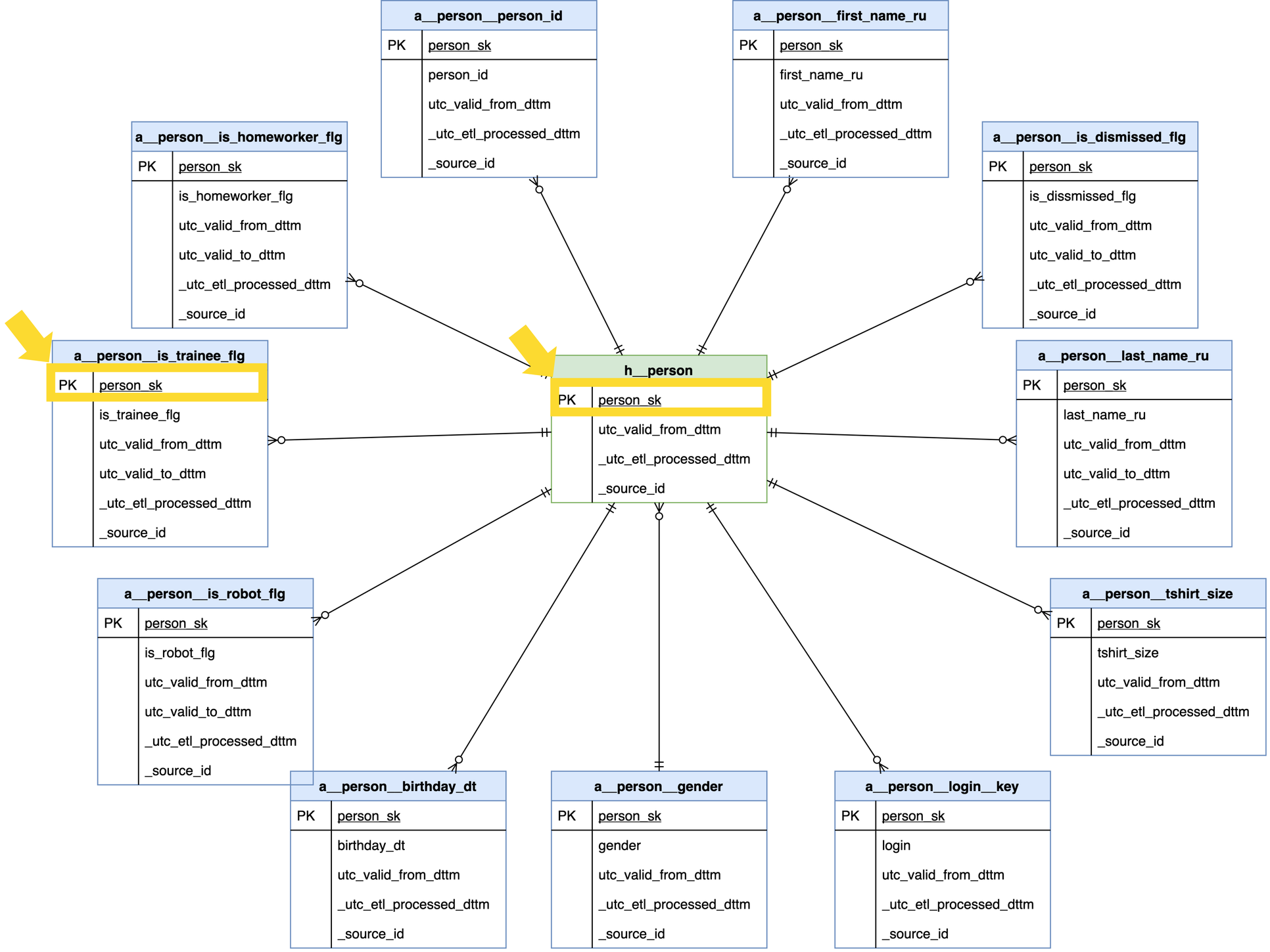
Объявление сущности

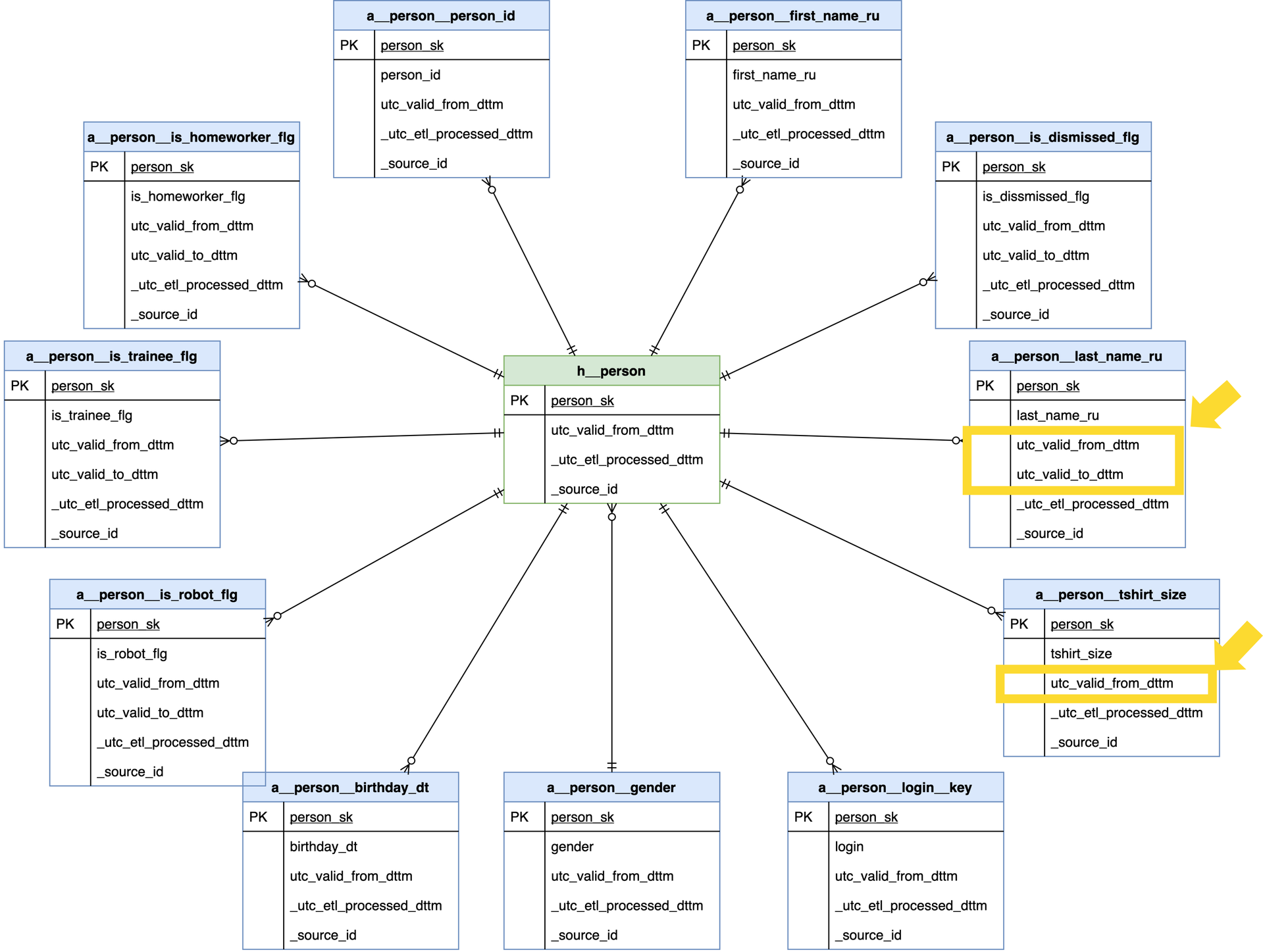
```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```

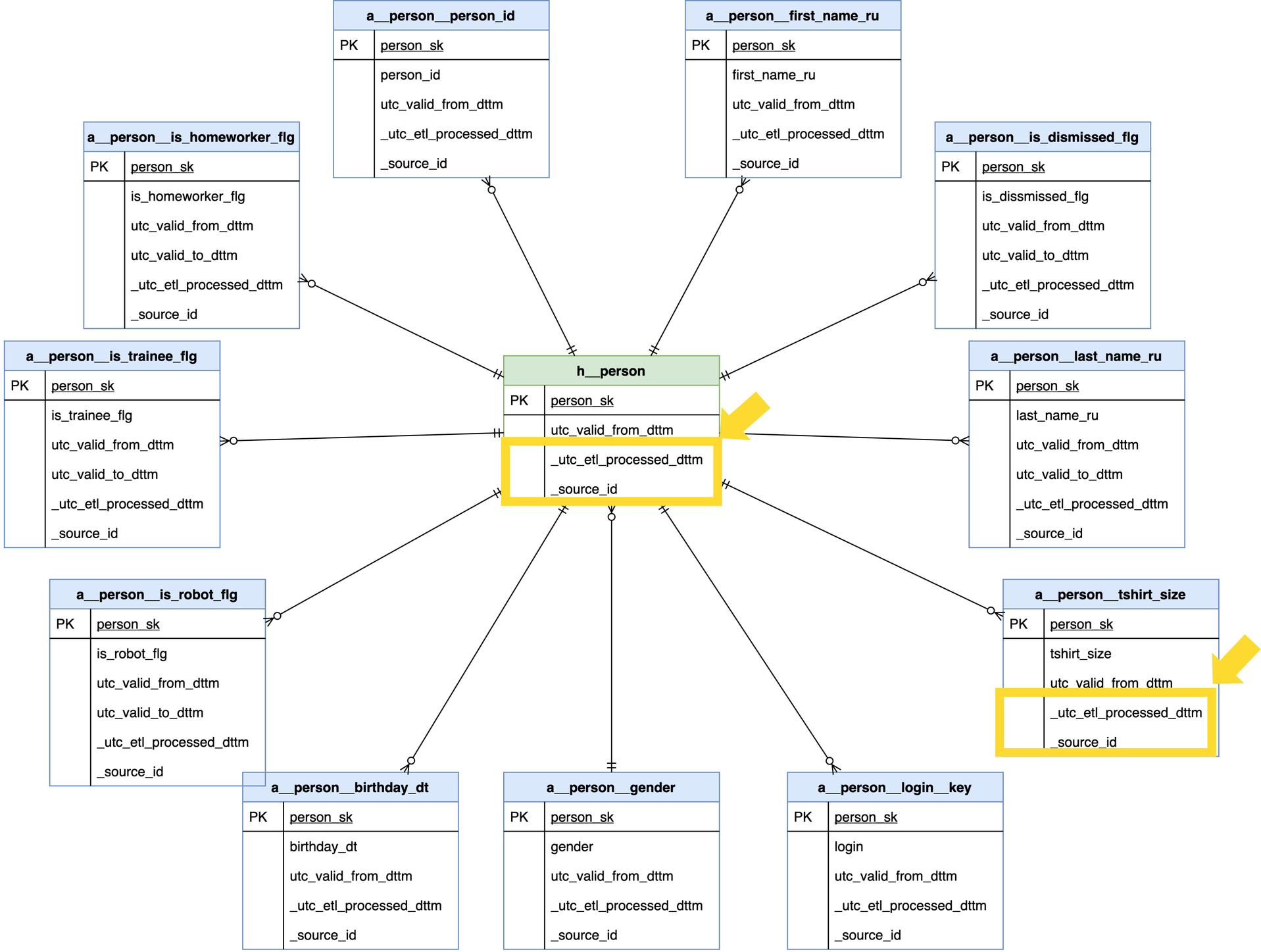
Объявление сущности

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```


a__person__person_id
PK
person_sk
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_homeworker_flg
PK
person_sk
is_homeworker_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_dismissed_flg
PK
person_sk
is_dissmissed_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_trainee_flg
PK
person_sk
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_robot_flg
PK
person_sk
is_robot_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__tshirt_size
PK
person_sk
tshirt_size
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__birthday_dt
PK
person_sk
birthday_dt
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__gender
PK
person_sk
gender
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__login__key
PK
person_sk
login
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id


a__person__person_id
PK
person_sk
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_homeworker_flg
PK
person_sk
is_homeworker_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_dismissed_flg
PK
person_sk
is_dissmissed_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_trainee_flg
PK
person_sk
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_robot_flg
PK
person_sk
is_robot_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__tshirt_size
PK
person_sk
tshirt_size
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__birthday_dt
PK
person_sk
birthday_dt
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__gender
PK
person_sk
gender
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__login__key
PK
person_sk
login
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id


a__person__person_id
PK
person_sk
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_homeworker_flg
PK
person_sk
is_homeworker_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_dismissed_flg
PK
person_sk
is_dissmissed_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_trainee_flg
PK
person_sk
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_robot_flg
PK
person_sk
is_robot_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__tshirt_size
PK
person_sk
tshirt_size
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__birthday_dt
PK
person_sk
birthday_dt
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__gender
PK
person_sk
gender
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__login__key
PK
person_sk
login
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id

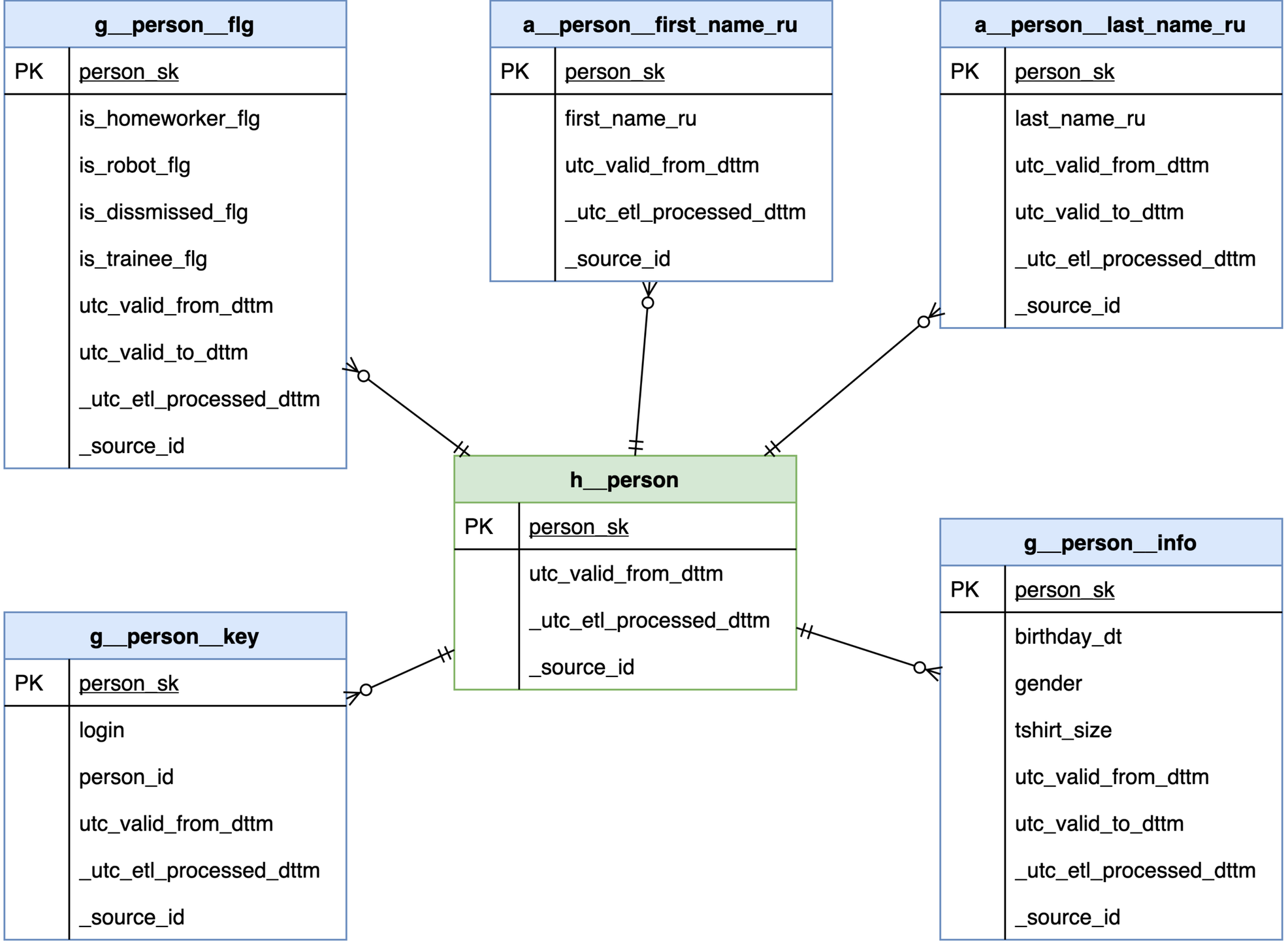

a__person__person_id
PK
person_sk
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_homeworker_flg
PK
person_sk
is_homeworker_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_dismissed_flg
PK
person_sk
is_dissmissed_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_trainee_flg
PK
person_sk
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_robot_flg
PK
person_sk
is_robot_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__tshirt_size
PK
person_sk
tshirt_size
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__birthday_dt
PK
person_sk
birthday_dt
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__gender
PK
person_sk
gender
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__login__key
PK
person_sk
login
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id

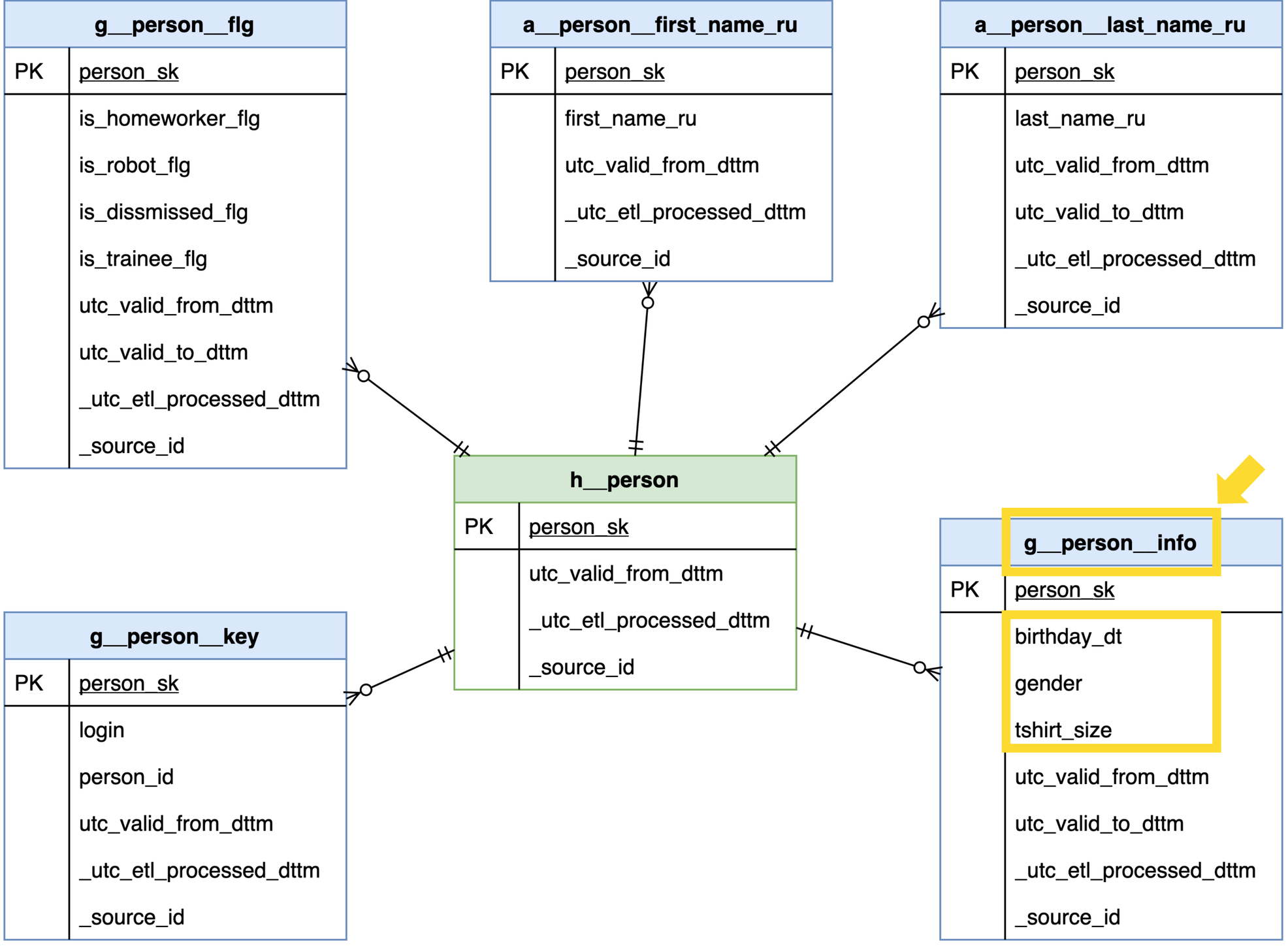
Объявление сущности с группами атрибутов

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE, group='key')
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE, group='key')
    gender = String(comment='Пол', change_type=ChangeType.UPDATE, group='info')
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE, group='info')
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE, group='info')
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW, group='flg')
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW, group='flg')
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW, group='flg')
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW, group='flg')

    __keys__ = [login]
```

Объявление сущности с группами атрибутов

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE, group='key')
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE, group='key')
    gender = String(comment='Пол', change_type=ChangeType.UPDATE, group='info')
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE, group='info')
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE, group='info')
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW, group='flg')
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW, group='flg')
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW, group='flg')
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW, group='flg')

    __keys__ = [login]
```


g__person__flg
PK
person_sk
is_homeworker_flg
is_robot_flg
is_dissmissed_flg
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__key
PK
person_sk
login
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__info
PK
person_sk
birthday_dt
gender
tshirt_size
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id

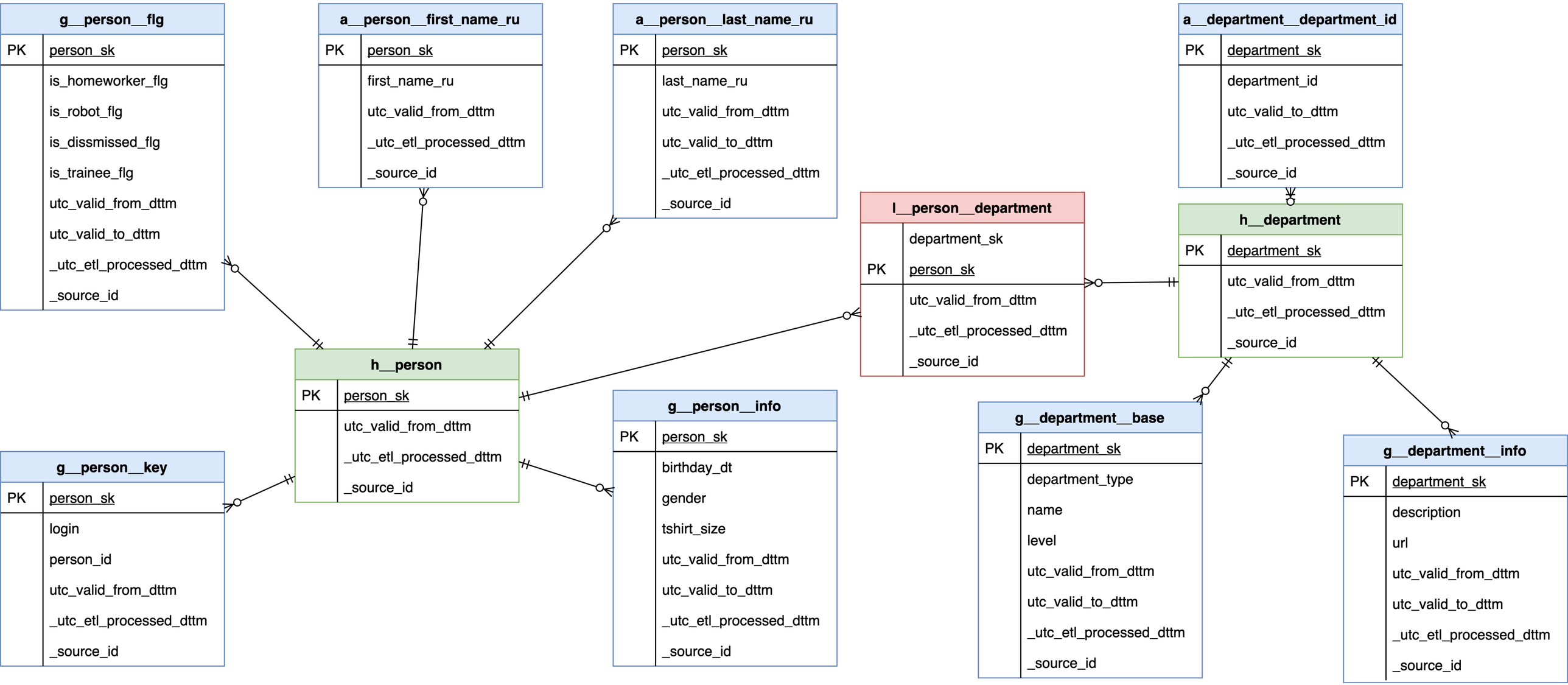

g__person__flg
PK
person_sk
is_homeworker_flg
is_robot_flg
is_dissmissed_flg
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__key
PK
person_sk
login
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__info
PK
person_sk
birthday_dt
gender
tshirt_size
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id

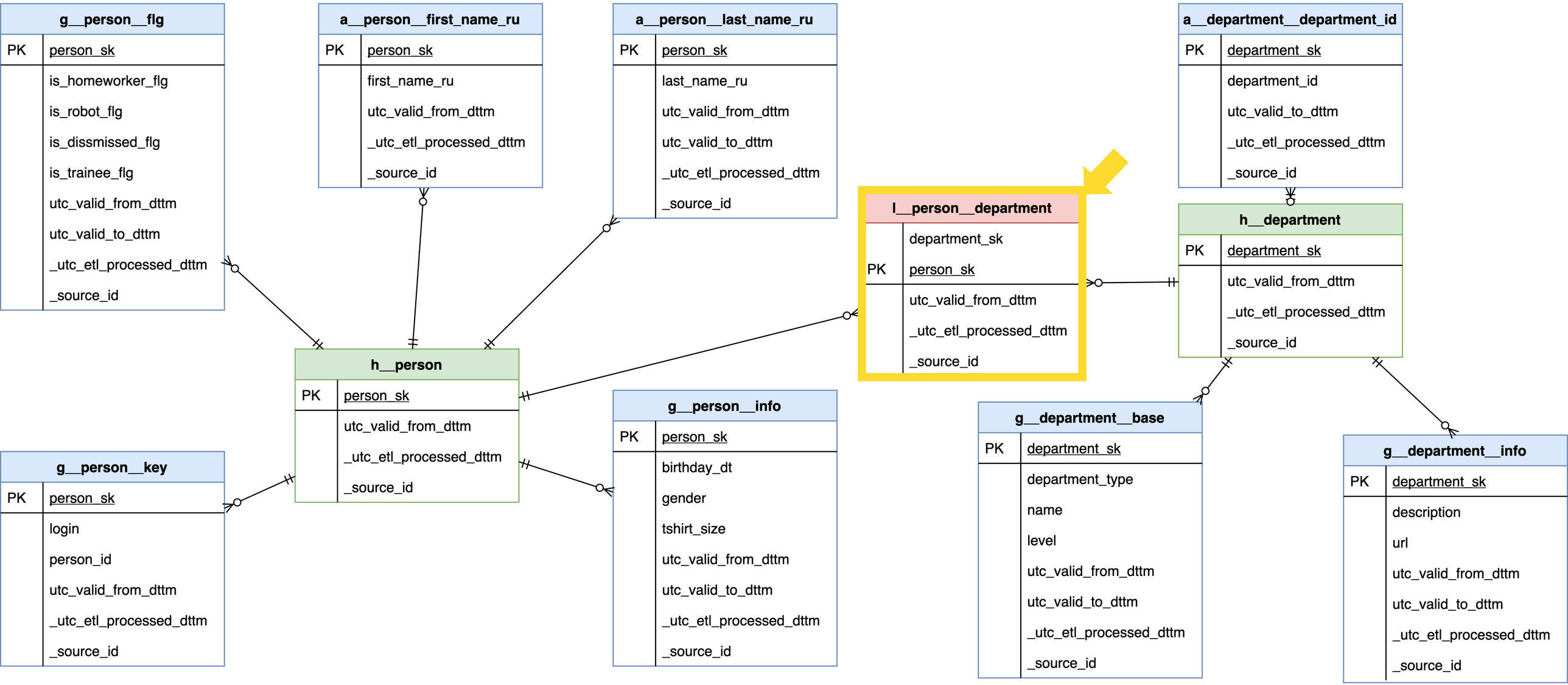
Объявление связи

```python
class Department(HnhmEntity):
    """Департамент со стафа."""
    __layout__ = DdsLayout(name='department', group='staff')
    department_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    department_type = String(comment='Тип подразделения', change_type=ChangeType.NEW, group='base')
    name = String(comment='Название подразделения', change_type=ChangeType.NEW, group='base')
    level = Int(comment='Уровень в иерархии', change_type=ChangeType.NEW, group='base')
    description = String(comment='Описание', change_type=ChangeType.UPDATE, group='info')
    url = String(comment='Ссылка на стафф', change_type=ChangeType.UPDATE, group='info')

    __keys__ = [department_id]


class PersonDepartment(HnhmLink):
    """Связь между департаментом и персоной стафа"""
    __layout__ = DdsLayout(name='person_department', group='link')
    department = HnhmLinkElement(entity=Department(), comment='департамент')
    person = HnhmLinkElement(entity=Person(), comment='персона')

    __keys__ = [person]
```

Объявление связи

```
class Department(HnhmEntity):
    """Департамент со стафа."""
    __layout__ = DdsLayout(name='department', group='staff')
    department_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    department_type = String(comment='Тип подразделения', change_type=ChangeType.NEW, group='base')
    name = String(comment='Название подразделения', change_type=ChangeType.NEW, group='base')
    level = Int(comment='Уровень в иерархии', change_type=ChangeType.NEW, group='base')
    description = String(comment='Описание', change_type=ChangeType.UPDATE, group='info')
    url = String(comment='Ссылка на стафф', change_type=ChangeType.UPDATE, group='info')

    __keys__ = [department_id]


class PersonDepartment(HnhmLink):
    """Связь между департаментом и персоной стафа"""
    __layout__ = DdsLayout(name='person_department', group='link')
    department = HnhmLinkElement(entity=Department(), comment='департамент')
    person = HnhmLinkElement(entity=Person(), comment='персона')

    __keys__ = [person]
```


g__person__flg
PK
person_sk
is_homeworker_flg
is_robot_flg
is_dissmissed_flg
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__department__department_id
PK
department_sk
department_id
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
l__person__department
department_sk
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
h__department
PK
department_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__info
PK
person_sk
birthday_dt
gender
tshirt_size
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
g__department__base
PK
department_sk
department_type
name
level
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
g__person__key
PK
person_sk
login
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__department__info
PK
department_sk
description
url
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id


g__person__flg
PK
person_sk
is_homeworker_flg
is_robot_flg
is_dissmissed_flg
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__department__department_id
PK
department_sk
department_id
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
l__person__department
department_sk
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
h__department
PK
department_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__info
PK
person_sk
birthday_dt
gender
tshirt_size
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
g__person__key
PK
person_sk
login
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__department__base
PK
department_sk
department_type
name
level
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
g__department__info
PK
department_sk
description
url
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id

Загрузка сущностей

```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({
    Person.person_id: Stg.person_id,
    Person.first_name_ru: Stg.first_name_ru,
    Person.last_name_ru: Stg.last_name_ru,
    Person.login: Stg.login,
    Person.is_robot_flg: Stg.is_robot_flg,
    Person.gender: Stg.gender,
    Person.birthday_dt: Stg.birthday_dt,
    Person.tshirt_size: Stg.tshirt_size,
})
```

Загрузка сущностей

```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({
    Person.person_id: Stg.person_id,
    Person.first_name_ru: Stg.first_name_ru,
    Person.last_name_ru: Stg.last_name_ru,
    Person.login: Stg.login,
    Person.is_robot_flg: Stg.is_robot_flg,
    Person.gender: Stg.gender,
    Person.birthday_dt: Stg.birthday_dt,
    Person.tshirt_size: Stg.tshirt_size,
})
```

Загрузка сущностей

```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({
    Person.person_id: Stg.person_id,
    Person.first_name_ru: Stg.first_name_ru,
    Person.last_name_ru: Stg.last_name_ru,
    Person.login: Stg.login,
    Person.is_robot_flg: Stg.is_robot_flg,
    Person.gender: Stg.gender,
    Person.birthday_dt: Stg.birthday_dt,
    Person.tshirt_size: Stg.tshirt_size,
})
```

Загрузка связей

```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({Person.login: Stg.login})\
    .target(Department).mappings({Department.department_id: Stg.department_id})\
    .target(Telegram).mappings({Telegram.login: Stg.telegram})\
    .target(Email, 'personal').mappings({Email.email: Stg.personal_email})\
    .target(Email, 'work').mappings({Email.email: Stg.email})\
    .target(LinkPersonDepartment)\
    .target(LinkPersonTelegram)\
    .target(LinkPersonOffice)\
    .target(LinkDepartmentOrganization)\
    .target(LinkPersonPersonalEmail).mappings(
    {LinkPersonPersonalEmail.person: Person,
     LinkPersonPersonalEmail.personal_email: 'personal'})\
    .target(LinkPersonWorkEmail).mappings(
    {LinkPersonWorkEmail.person: Person,
     LinkPersonWorkEmail.work_email: 'work'}
)
```

Загрузка связей

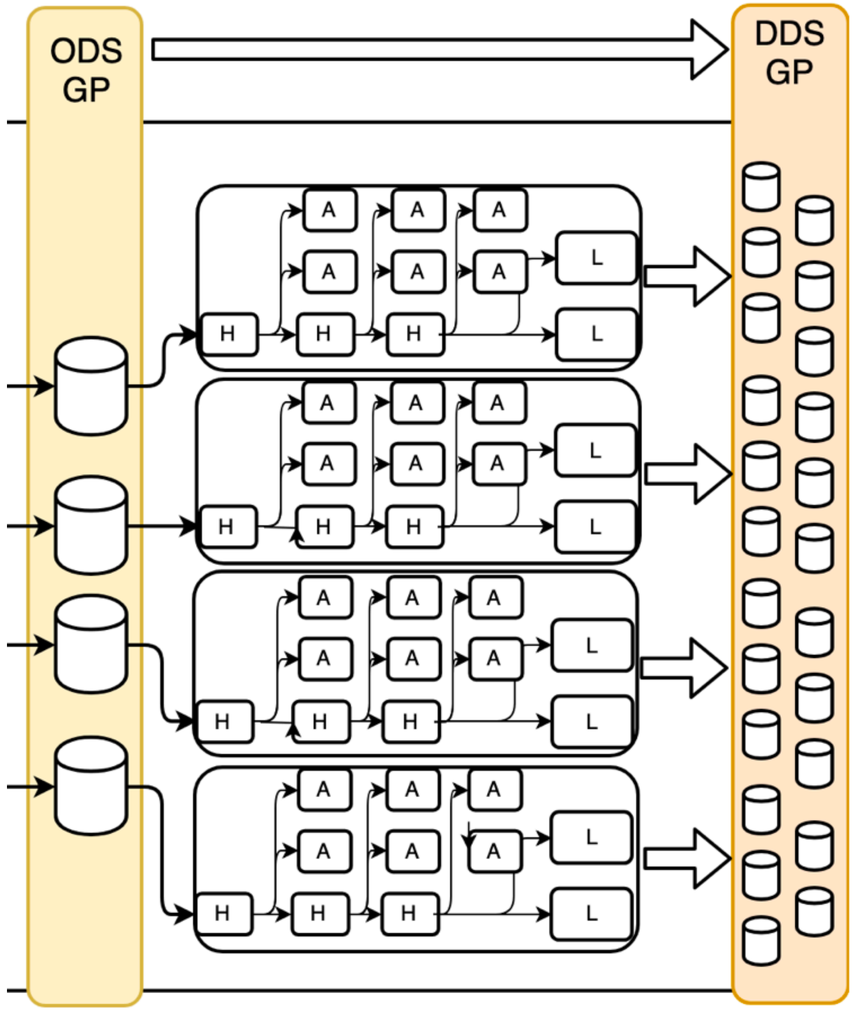
```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({Person.login: Stg.login})\
    .target(Department).mappings({Department.department_id: Stg.department_id})\
    .target(Telegram).mappings({Telegram.login: Stg.telegram})\
    .target(Email, 'personal').mappings({Email.email: Stg.personal_email})\
    .target(Email, 'work').mappings({Email.email: Stg.email})\
    .target(LinkPersonDepartment)\
    .target(LinkPersonTelegram)\
    .target(LinkPersonOffice)\
    .target(LinkDepartmentOrganization)\
    .target(LinkPersonPersonalEmail).mappings(
    {LinkPersonPersonalEmail.person: Person,
     LinkPersonPersonalEmail.personal_email: 'personal'})\
    .target(LinkPersonWorkEmail).mappings(
    {LinkPersonWorkEmail.person: Person,
     LinkPersonWorkEmail.work_email: 'work'}
)
```

Загрузка связей

```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({Person.login: Stg.login})\
    .target(Department).mappings({Department.department_id: Stg.department_id})\
    .target(Telegram).mappings({Telegram.login: Stg.telegram})\
    .target(Email, 'personal').mappings({Email.email: Stg.personal_email})\
    .target(Email, 'work').mappings({Email.email: Stg.email})\
    .target(LinkPersonDepartment)\
    .target(LinkPersonTelegram)\
    .target(LinkPersonOffice)\
    .target(LinkDepartmentOrganization)\
    .target(LinkPersonPersonalEmail).mappings(
    {LinkPersonPersonalEmail.person: Person,
     LinkPersonPersonalEmail.personal_email: 'personal'})\
    .target(LinkPersonWorkEmail).mappings(
    {LinkPersonWorkEmail.person: Person,
     LinkPersonWorkEmail.work_email: 'work'}
)
```

Граф загрузки

- Для каждой такой загрузки генерируется граф загрузки из атомарных загрузок хабов, линков, саттелитов
- Суррогатный ключ генерится как хеш от набора бизнес-ключей
- Все задачи внутри графа выполняются параллельно
- В зависимости от типа изменений это или Insert\Update по SCD2, или Insert
- Данные из разных источников также могут загружаться параллельно



II-2

Использование hNhM

- Почему чистый sql не подходит
- Решение на стороне Python
- Решение на стороне СУБД

Построение витрин на базе hNhM

СУБД

Все сущность покрыты view

- Выглядит как полноценная таблица
- Скрывает внутри, на какие группы разбиты атрибуты
- Удобно для использования внутри СУБД

Python

Генерация SQL на базе описания

- Встроено в таски нашей платформы
- Генерирует или cte, или подготавливает временные таблицы
- Удобно для использования в ETL-процессах

Почему чистый sql не подходит в hNhM

- Надо знать все о сущностях и атрибутах
- Изменение описания влияет на витрины
- Сложнаааа…

Доступ к сущностям из Python

> **Был разработано несколько классов, которые позволяют сформировать SQL запрос к определенной сущности и затем использовать его в построении витрины**

```python
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```



```sql
WITH department AS (
    {department}
), person AS (
    {person}
), link_person_department AS (
    {link_person_department}
)
SELECT *
FROM department d
  LEFT JOIN link_person_department lpdr
        ON d.id = lpdr.department
  LEFT JOIN person p
        ON p.id = lpdr.person
```

Как это выглядит

```
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```

Как это выглядит

```
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```

Как это выглядит

```
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```

Как это выглядит

```
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```

Как выглядит SQL запрос

```
WITH department AS (
    {department}
), person AS (
    {person}
), link_person_department AS (
    {link_person_department}
)
SELECT *
FROM department d
  LEFT JOIN link_person_department lpdr
         ON d.id = lpdr.department
  LEFT JOIN person p
         ON p.id = lpdr.person
```

Как выглядит SQL запрос

```
WITH department AS (
    {department}
), person AS (
    {person}
), link_person_department AS (
    {link_person_department}
)
SELECT *
FROM department d
  LEFT JOIN link_person_department lpdr
         ON d.id = lpdr.department
  LEFT JOIN person p
         ON p.id = lpdr.person
```

Доступ к сущностям из СУБД

Для доступа к сущностям используются специальные процедур

```
SELECT *
FROM get_entity_act(in_entity := 'person'
  , in_cols := '{person_id, gender, tshirt_size}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

	person_id	gender	tshirt_size
1	1	m	L
2	2	f	XS
3	3	f	S
4	4	m	M
5	5	m	L

```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

	person_id				department_id	description
1	1	m	L	2	tmp	test
2	2	f	XS	2	tmp	test
3	3	f	S	1	tmp1	test
4	4	m	M	1	tmp1	test
5	5	m	L	1	tmp1	test

Как это выглядит

```
SELECT *
FROM get_entity_act(in_entity := 'person'
  , in_cols := '{person_id, gender, tshirt_size}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM get_entity_act(in_entity := 'person'
  , in_cols := '{person_id, gender, tshirt_size}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM get_entity_act(in_entity := 'person'
  , in_cols := '{person_id, gender, tshirt_size}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM get_entity_act(in_entity := 'person'
  , in_cols := '{person_id, gender, tshirt_size}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит


```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

Как это выглядит

```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

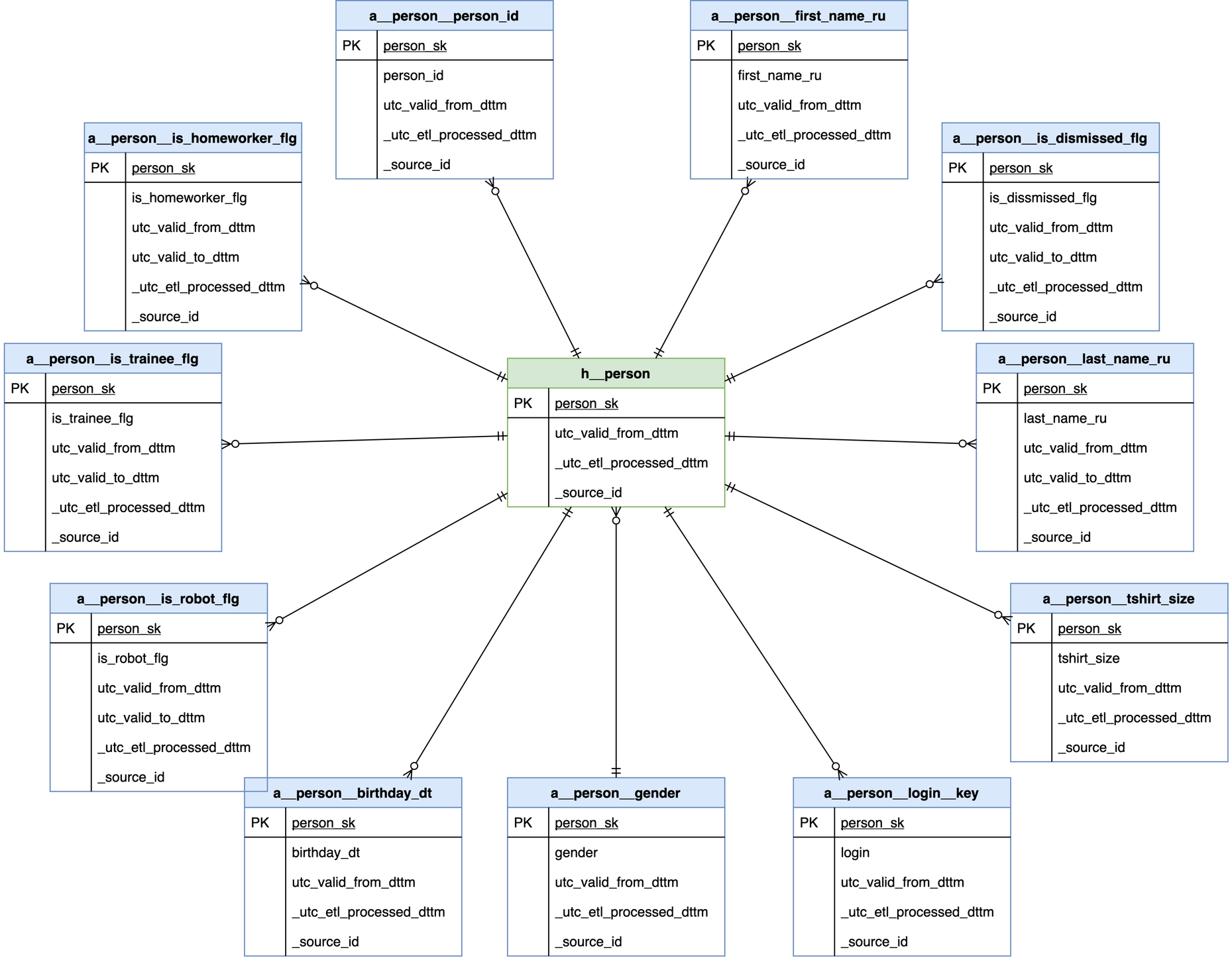
II-3

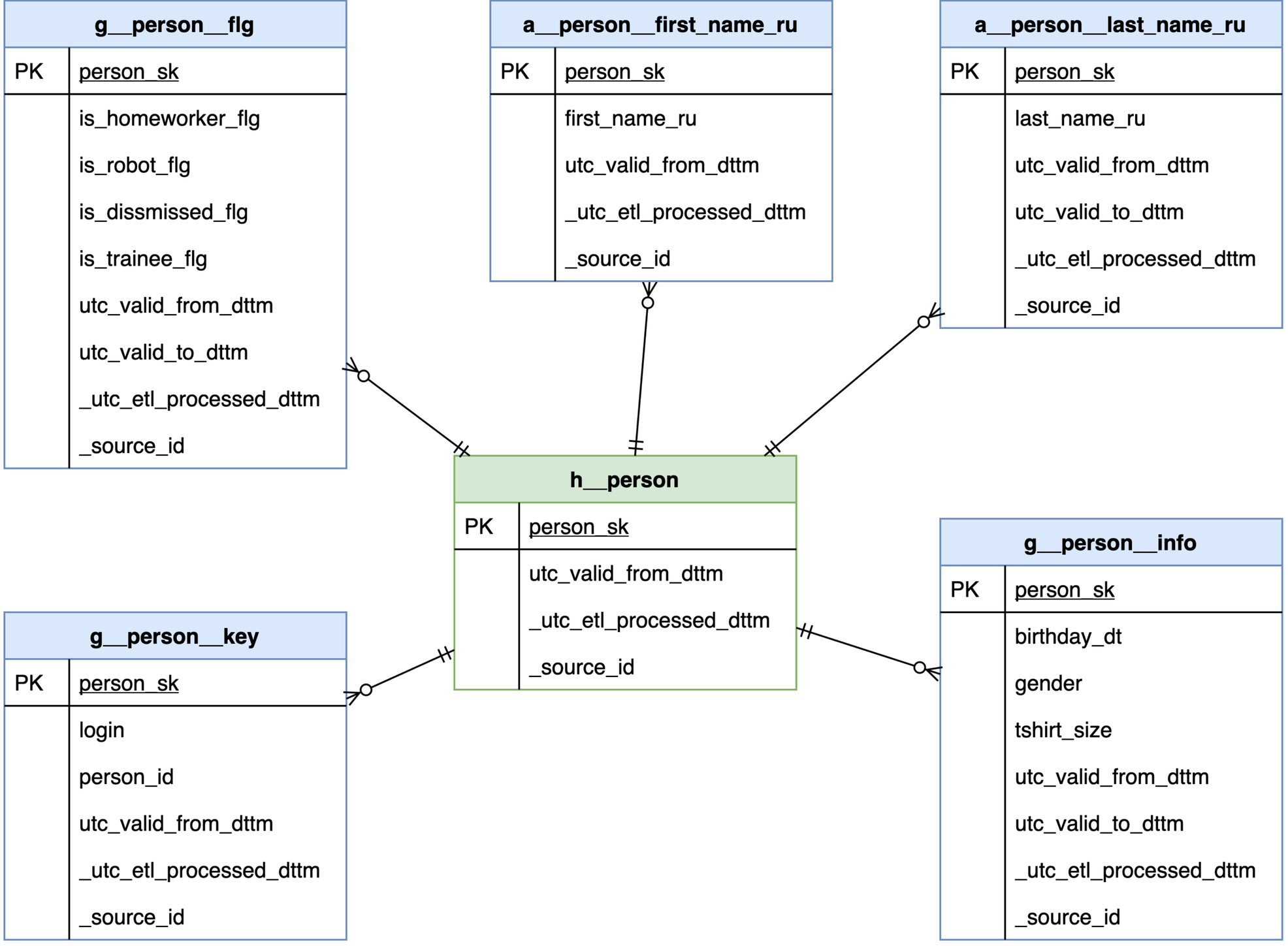
Оптимизация: Атрибуты vs Группы

- Атрибуты или группы атрибутов
- Преобразование физической схемы
- Задача оптимизации

Объявление сущности

```
class Person(HnhmEntity):
    """Сотрудник со staff.yandex-team.ru"""

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```


a__person__person_id
PK
person_sk
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_homeworker_flg
PK
person_sk
is_homeworker_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_dismissed_flg
PK
person_sk
is_dissmissed_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_trainee_flg
PK
person_sk
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__is_robot_flg
PK
person_sk
is_robot_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__tshirt_size
PK
person_sk
tshirt_size
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__birthday_dt
PK
person_sk
birthday_dt
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__gender
PK
person_sk
gender
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__login__key
PK
person_sk
login
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id


g__person__flg
PK
person_sk
is_homeworker_flg
is_robot_flg
is_dissmissed_flg
is_trainee_flg
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
a__person__first_name_ru
PK
person_sk
first_name_ru
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
a__person__last_name_ru
PK
person_sk
last_name_ru
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id
h__person
PK
person_sk
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__key
PK
person_sk
login
person_id
utc_valid_from_dttm
_utc_etl_processed_dttm
_source_id
g__person__info
PK
person_sk
birthday_dt
gender
tshirt_size
utc_valid_from_dttm
utc_valid_to_dttm
_utc_etl_processed_dttm
_source_id

Оптимизационная задача

Вопрос: как оптимально объединить атрибуты по группам?

Дано (и есть в metaDWH):

- Метаданные объектов
- Маппинги полей и загрузчики
- Количество строк в объекте и инкремент
- Накопленное знание о частоте изменений

Ограничения

- Набор полей в метаданных объектов
- Маппинги полей и загрузчики (группа должна загружаться из одного источника)

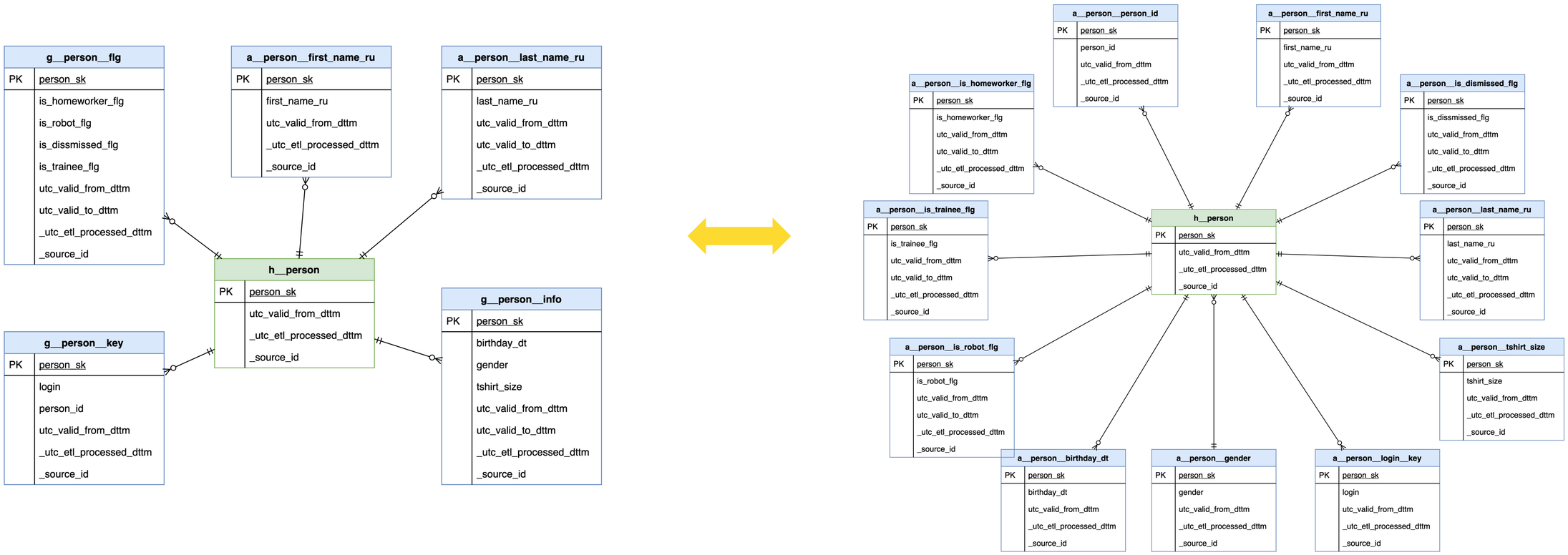
Оптимальность

- Будем минимизировать занимаемое место на диске

Преобразование физической модели

Вводим атомарные операции, меняющие схему, но не меняющие логику

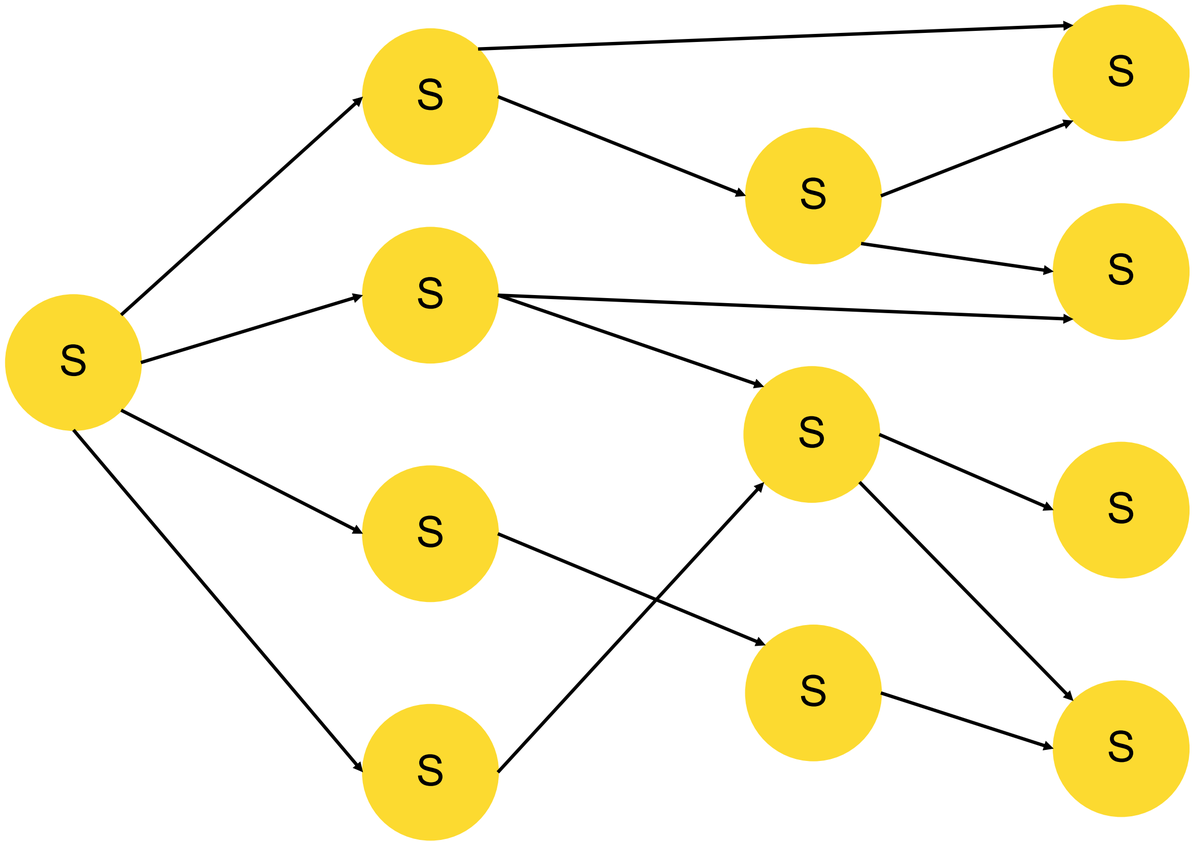
- Объединение групп\атрибутов в группу
- Разъединение группы в набор группы\атрибутов



Преобразование физической модели

Вводим атомарные операции, меняющие схему, но не меняющие логику

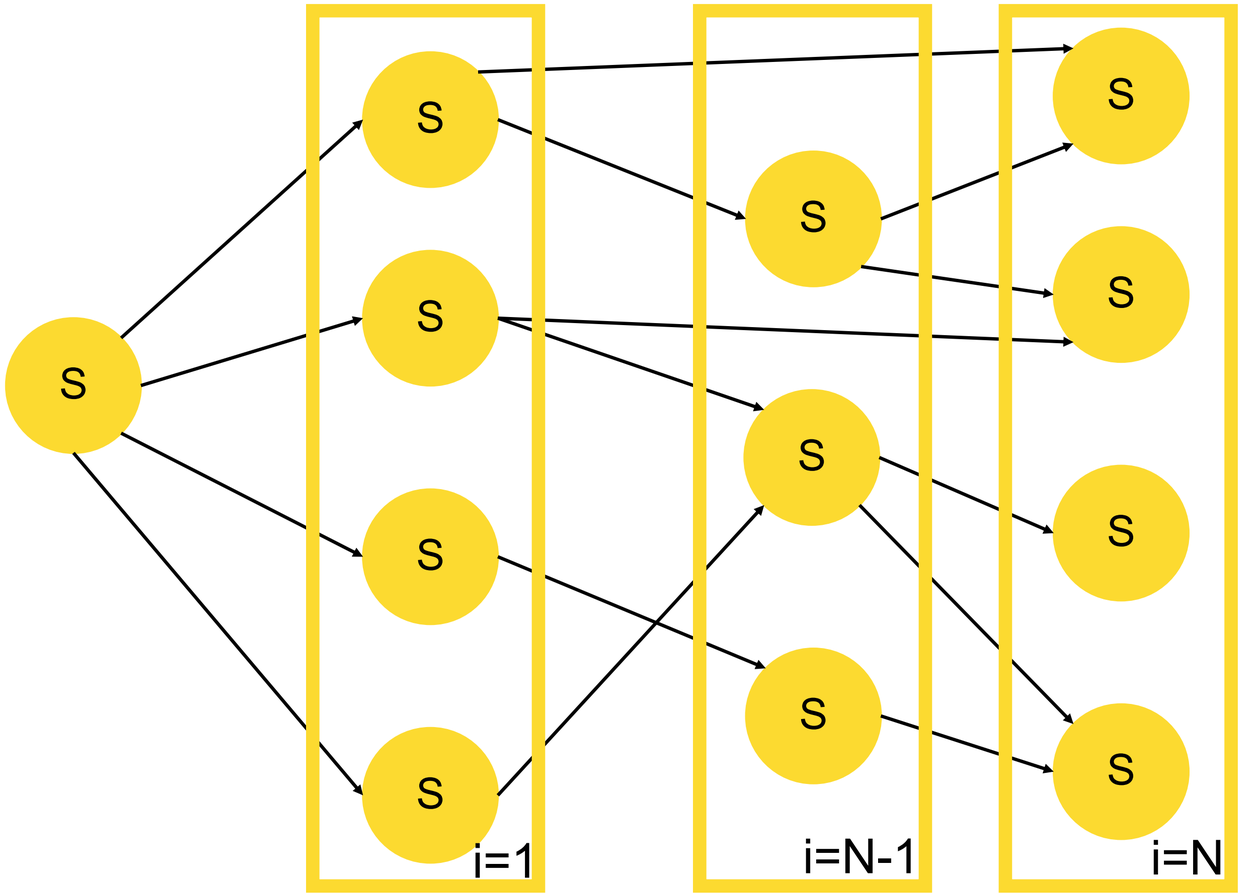
- Объединение групп\атрибутов в группу
- Разъединение группы в набор группы\атрибутов



Сходимость

Вводим атомарные операции, меняющие схему, но не меняющие логику

- Объединение групп\атрибутов в группу
- Разъединение группы в набор группы\атрибутов



Генетический алгоритм

- Из текущего состояния мутациями (=атомарными операциями) создаем стартовую популяцию
- Производим скрещивания и новые мутации
- Каждое состояние оцениваем на оптимальность (в нашем случае по месту)
- При подозрениях на сходимость останавливаемся

Результат

- Получаем итоговое состояние, которое лучше текущего
- Сравниваем метаданные между состояниями и генерируем скрипт миграции
- Миграция – отдельный вопрос на целый доклад ☺

06

Резюме

- Куда мы пришли?
- Как мы пришли?
- Стоит ли повторять наш путь?

Data Vault vs Anchor modeling

- На каждую сущность создается **hub** – таблица с бизнес-ключом и суррогатным ключом
- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования
- **Связи** только через отдельные таблицы, на них можно навесить сателлит
- Есть специальные таблицы **Point-in-Time** и **Bridge**

- На каждую сущность создается **anchor** – таблица только с суррогатным ключом
- Один **атрибут** – одна таблица, да здравствует 6НФ
- **Связи** только через отдельные таблицы, никаких атрибутов – только хардкор
- Есть специальная таблица **knot** – статический справочник

Мы провели сравнение DV и AM – и решили взять лучшее (на наш взгляд) из каждой методологии

Highly Normalized Hybrid Model (hNhM)

Ключевая идея: выбирать оптимальный формат хранения для каждого конкретного случая

- Высокая нормализация (вплоть до предельной 6НФ)
- Параллельная загрузка из разных источников
- Устойчивость к изменению в бизнесе
- Идемпотентность при повторной загрузке
- Модульность и масштабируемость
- Удобство использования при построении витрин
- Возможность эмулировать как Data Vault, так и Anchor Modeling

- Атрибуты группируются в таблицы-**сателлиты** по принципам совместности: изменения и\или источника и\или использования

- Есть специальные таблицы **Point-in-Time** и **Bridge**

- На каждую сущность создается **anchor** – таблица с суррогатным ключом

- **Связи** только через отдельные таблицы, никаких атрибутов – только хардкор

hNhM.Framework

Невозможно управлять таким количеством сущностей без framework

- Базировались на существующем внутреннем решении
- Реализовали описание сущностей и связей между ними
- Скрыли физическую реализацию таблиц
- Замаскировали сложность загрузки, при этом сохранив все требования к ней (идемпотентность)
- Создали инструмента для генерации кода построения витрин

Это сложный путь, который мы бы не советовали повторять без команды разработки

hNhM.Roadmap

Есть много точек роста у нашего подхода и framework

- Добавить больше гибкости – атрибуты у связей, все типы таблиц из DV и AM
- Независимость от системы хранения Greenplum
- Полноценный DSL для построения витрин (с учетом инкрементов)
- Оптимизация не только хранения, но и скорости выполнения запросов
- Автоматизированные миграции
- Визуальное редактирование метаданных

Когда-то (скоро, но без публичных комитов) этот инструмент дорастет до OpenSource – следите за обновлениями

Яндекс Такси

Спасибо

Евгений Ермаков

Руководитель DWH

jkermakov@yandex-team.ru

@iJKos

Николай Гребенщиков

Руководитель Data Engineer

ngrbnshchkv@yandex-team.ru

@ngrebenshchikov