

KARPOV.COURSES >>>

КОНСПЕКТ



> Конспект > 4 урок > Методология Data Vault

> Причины создания

Так как же избежать этих проблем?

> Data Vault и Anchor Modeling

Из чего состоит Data Vault

Хаб

Линк

Сателлит

> Отображение Data Vault в различных аспектах

Перевод классической ЗНФ таблицы в Data Vault

Расположение DV в слоях данных

Загрузка данных в DV 1.0

> Data Vault 2.0

Сравнение подходов Кимбалла, Инмона, DV 1.0, DV 2.0

Новые сущности в DV 2.0

PIT

Bridge

Predefined Derivations

Загрузка DV 2.0

> Воссоздание таблицы измерений и таблицы фактов из DV

> Как проектировать по Data Vault?

> Причины создания

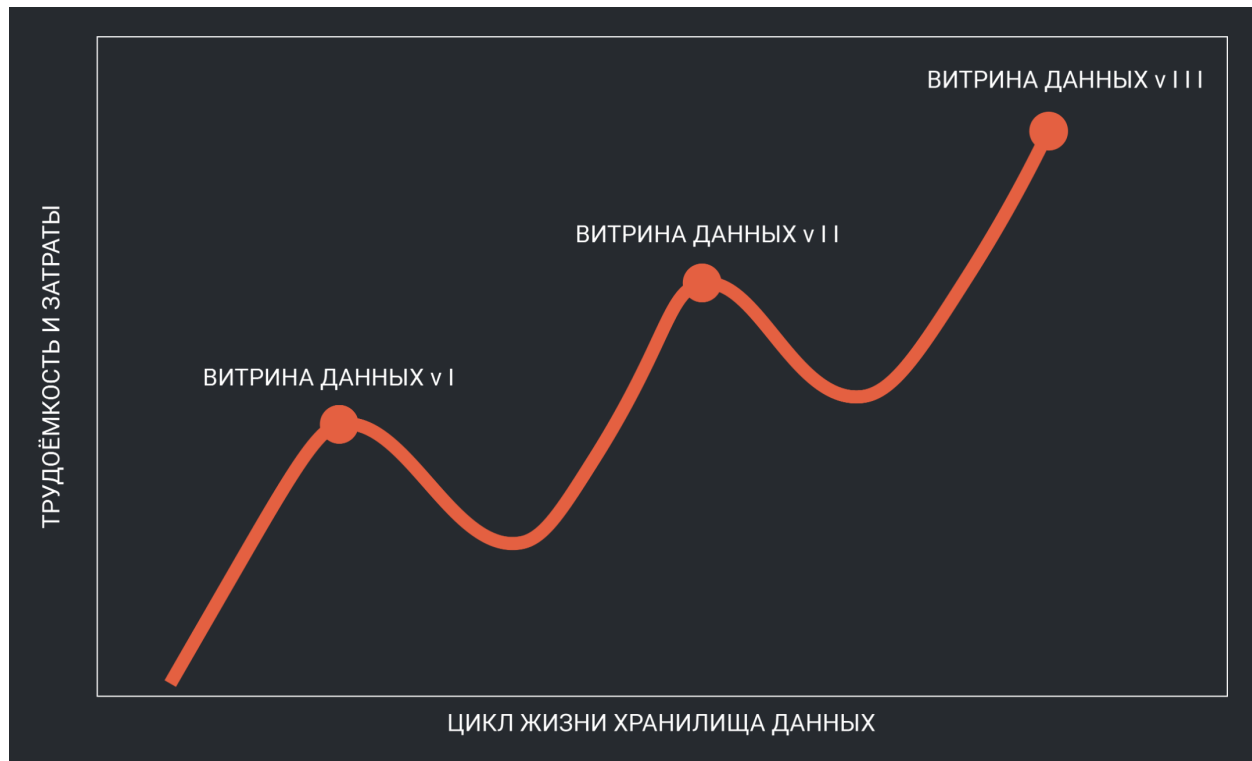


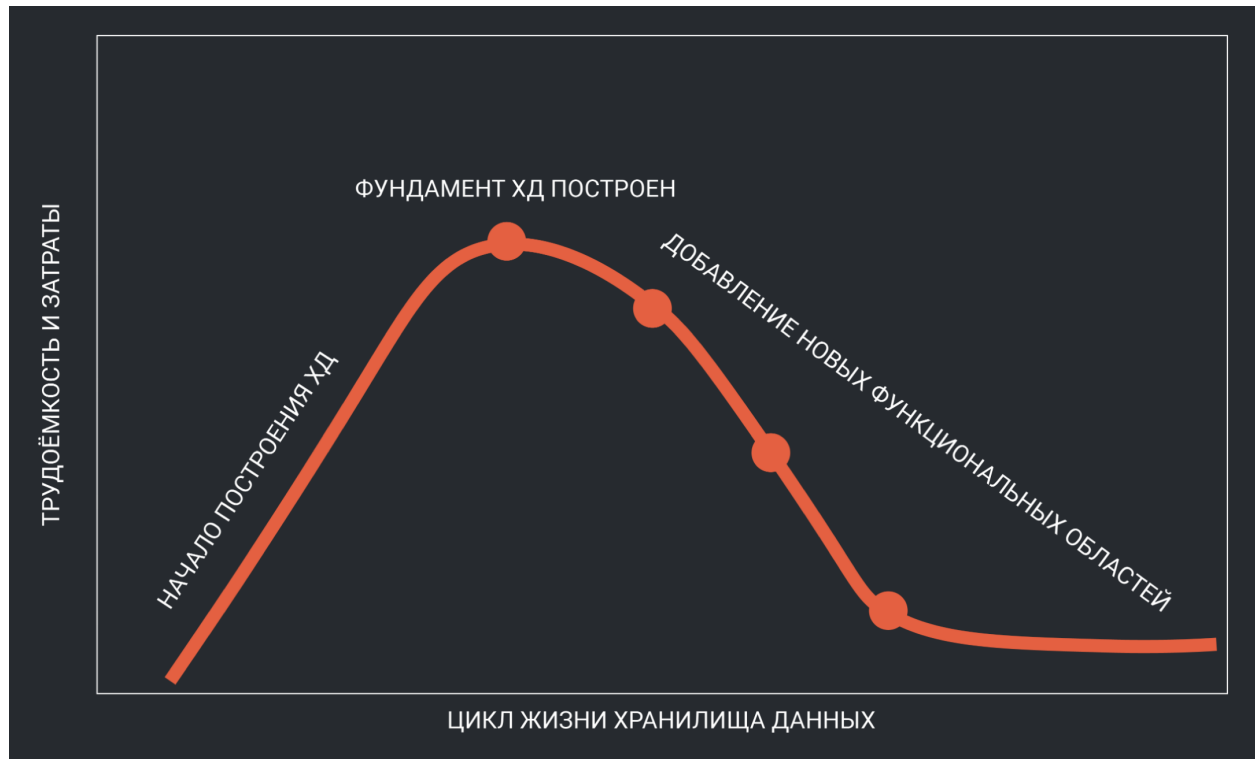
График выше отображает трудоемкость развития хранилища данных. Можно заметить, что чем сильнее расширяется хранилище данных, чем больше витрин мы делаем, тем более затратно их содержать/развивать.

Например, для Кимбалловской модели это действительно так: каждая последующая витрина использует те же источники, что были, так как отсутствует центральный слой.

Также это справедливо и для Инмоновской модели: она не устойчива к изменениям в бизнесе. Рассмотрим пример: сервис по заказу такси.

Изначально, считалось, что есть только пассажир (клиент такси) и его заказ. Один заказ = один пассажир. Однако, в определенный момент потребовалось ввести функцию разделения поездки с другими пассажирами. Связь заказов с пассажирами изменилась с **один ко многим** на **многие ко многим**.

Так как же избежать этих проблем?



Нужен новый принцип проектирования, добавляющий гибкости ядру ХД. На графике видно, что мы вкладываемся в создание фундамента хранилища данных, однако дальнейшее развитие обходится сильно дешевле.

> **Data Vault** и **Anchor Modeling**

Рассмотрим два подхода к построению ХД, позволяющим сделать его гибким и модульным.

Подробнее почитать можно тут – [Data Vault](#), [Anchor Modeling](#).

Можно сказать, что якорная модель (Anchor Modeling) – это крайняя теоретическая модель Data Vault.

Обе методологии:

- Повышают градус нормализации выше 3НФ
- Вводят свои типы таблиц и накладывают жесткие ограничения на их использование

- При использовании создают очень много таблиц

Взамен обещают:

- Уменьшить постоянное дублирование данных в SCD2 от изменения всего одного атрибута
- Избавить от деструктивных изменений, использовать только расширение модели (даже при изменении кардинальности связи)
- Позволить дорабатывать хранилище легко и быстро (agile-based)

Из чего состоит Data Vault

Методология DV (Data Vault) вводит и регламентирует следующие типы таблиц:

1. **Хаб (Hub)** – хранит сущности.
2. **Связь (Link)** – обеспечивает транзакционную интеграцию между Хабами (связи между сущностями).
3. **Сателлит (Satellite)** – предоставляет контекст первичного ключа Хаба (атрибуты, описания).

С версии 2.0 (мы рассмотрим эти типы позже):

1. **Мост (Bridge)** – упрощает соединение данных через несколько связей.
2. **PIT (point in time)** – упрощает получение информации из Саттелитов одной сущности с разной частотой обновления.
3. **Predefined Derivations** представляет собой предрасчитанные значения.

Хаб

HUB_LOCATION		
HUB_LOCATION_KEY	char(32)	PK
LOCATION_NAME	varchar2(50)	
HUB_Load_DTS	date	
HUB_Rec_SRC	varchar2(12)	

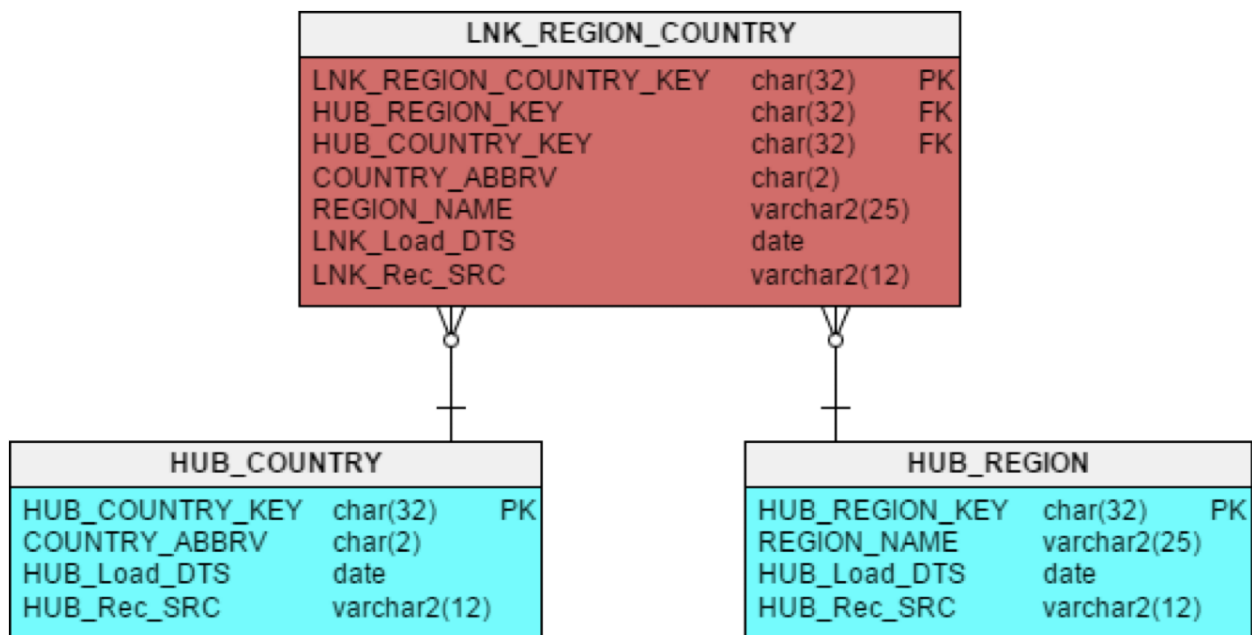
Пример Хаба

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



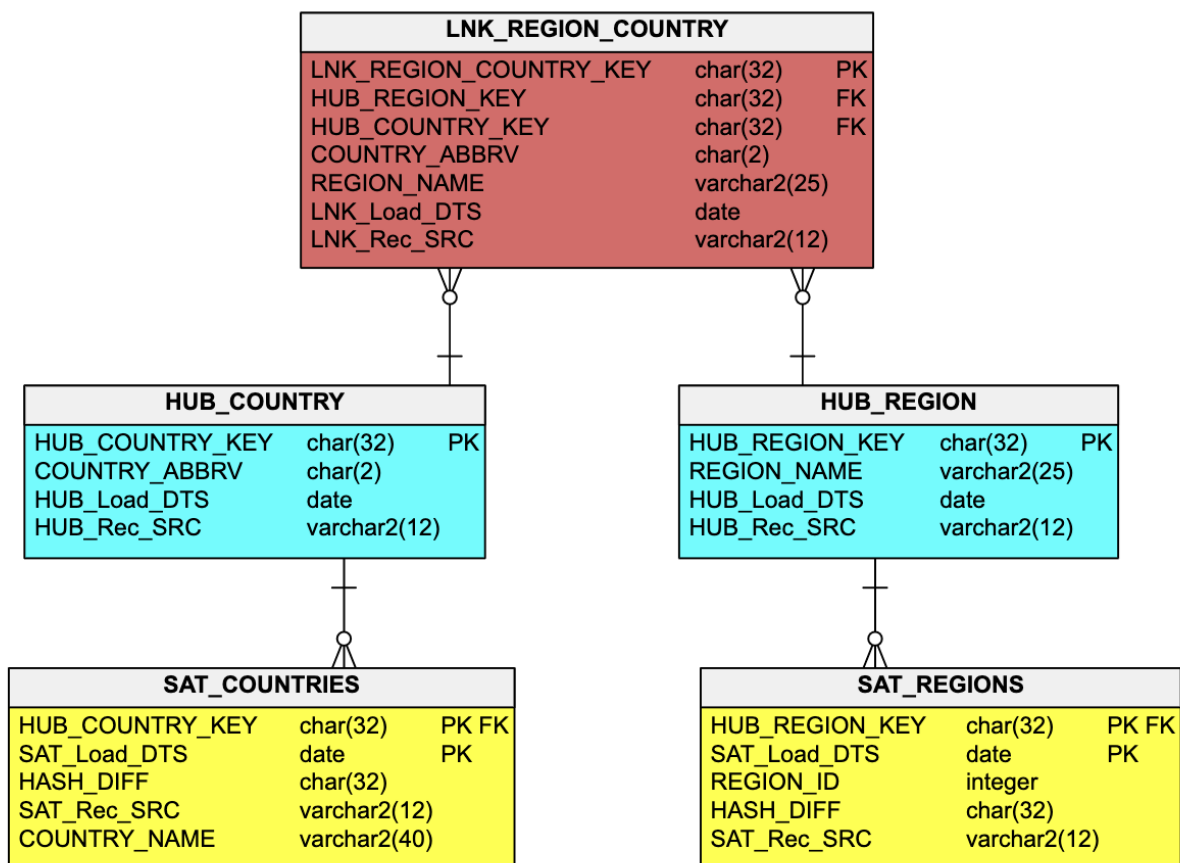
Пример Линка

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

Атрибуты линка включают:

- Суррогатный ключ (Surrogate Key)
- Ключи Хабов: от 2-ух Хабов до N Хабов, где N – количество входящих в Link Хабов
- Временная отметка даты загрузки
- Код источника данных

Сателлит






Пример Сателлита

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

Атрибуты сателлита включают:

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

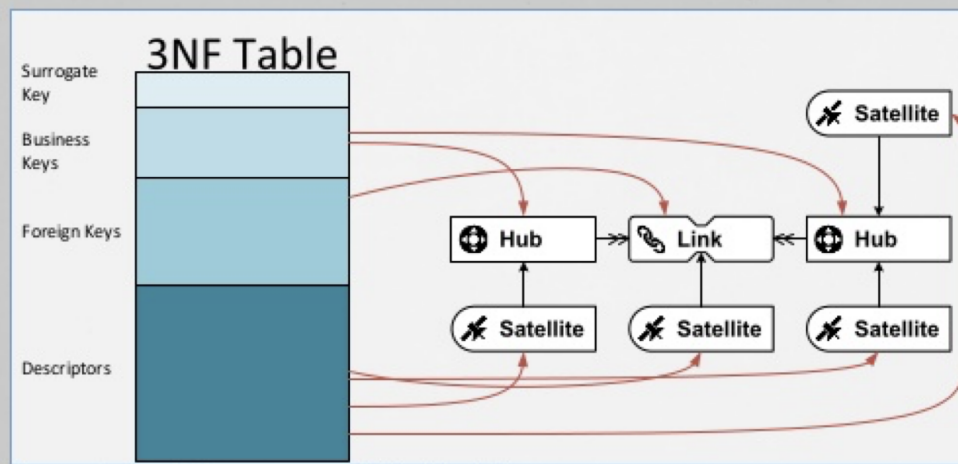
Сравнение основных составляющих DV

 Хаб	 Линк	 Сателлит
<u>Суррогатный ключ</u>	Суррогатный ключ	Первичный ключ Хаба или первичный ключ Связи
<u>Ключ бизнес-сущности из внешней системы</u>	Ключи Хабов (первичные/бизнес): от 2-ух Хабов до N Хабов	Даты действия записи (SCD2)
<u>Временная отметка даты загрузки</u>	Временная отметка даты загрузки	Временная отметка даты загрузки
<u>Код источника данных</u>	Код источника данных	Код источника данных

> Отображение Data Vault в различных аспектах

Перевод классической 3НФ таблицы в Data Vault

DATA VAULT 2.0 MODELING



In accordance with its own representation Linstedt, 2014

DÖRFFLER
PARTNER

12.02.2014

Agile Data Mining with Data Vault 2.0

30

Схема превращения таблицы в 3НФ в Data Vault

- Суррогатные ключи исходной таблицы не используются
- Бизнес-ключи переходят в Хабы
- Внешние ключи превращаются в Линки
- Все дескрипторы (описательная информация) переходят в Сателлиты

Важно, что Сателлит может быть как у Хаба, так и у Линка. Линк при этом становится транзакционной информацией, **НО** тогда Data Vault становится таким же неустойчивым к изменению кардинальности связи.

Расположение DV в слоях данных

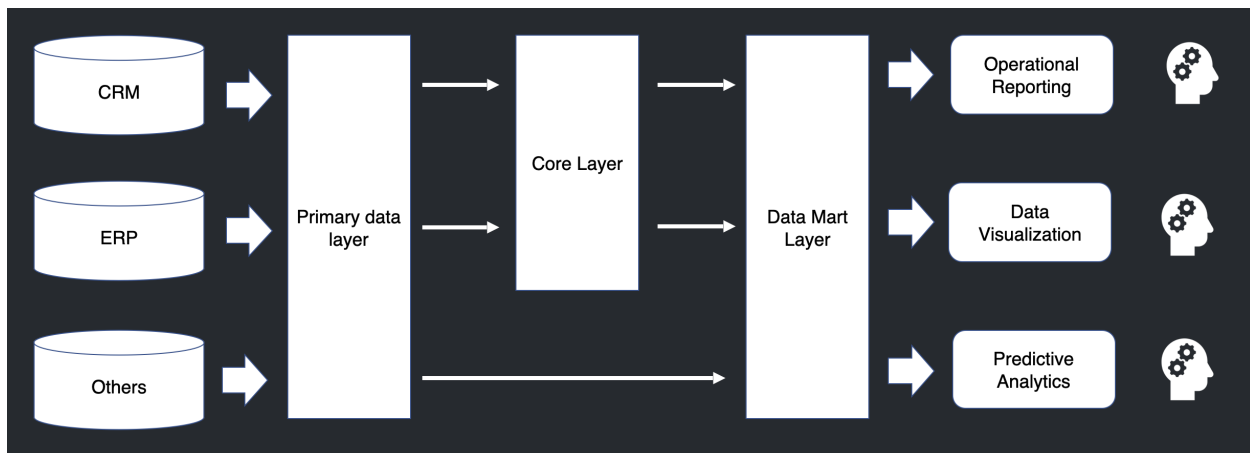
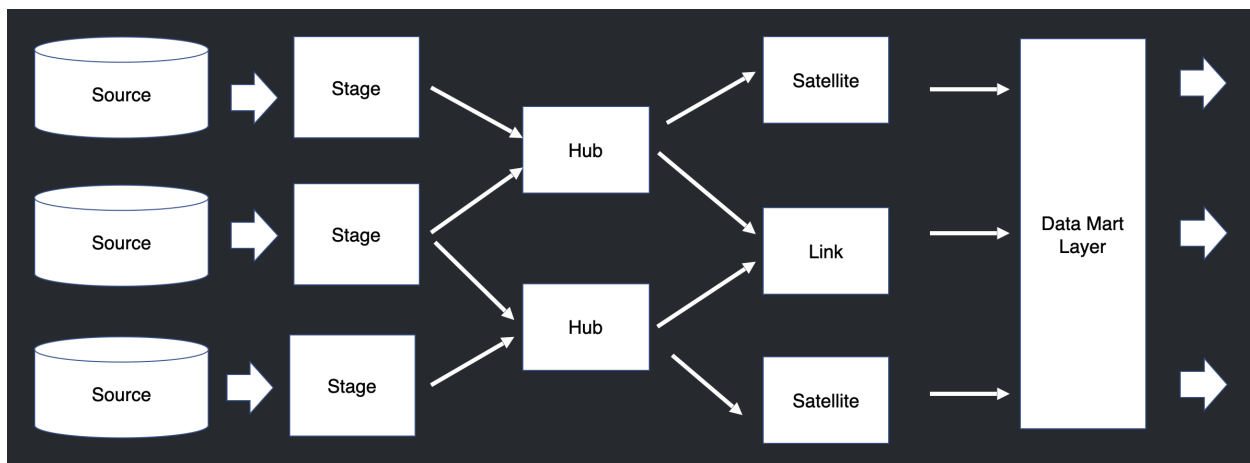


Схема слоев данных

В классической схеме слоев данных Data Vault является ядром (**Core Layer**). После сбора информации из различных источников данные раскладываются в ядре по DV, а после из DV собираются удобные для аналитиков витрины (денормализованные).

Загрузка данных в DV 1.0



Пайплайн загрузки данных в Data Vault 1.0

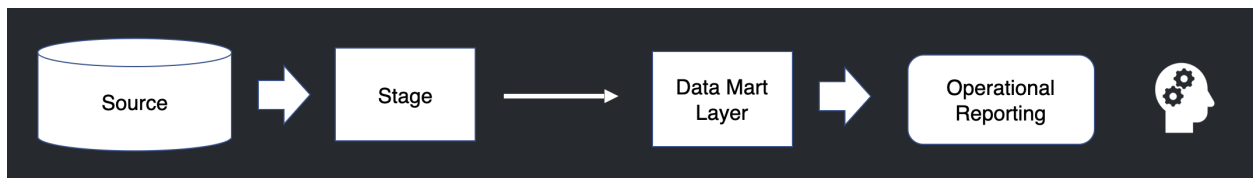
Загрузка происходит следующим образом:

- Из источников данные собираются в стейджинг
- Стейджинг прогружается в **Хабы**
- Из **Хабов** генерируются суррогатные ключи

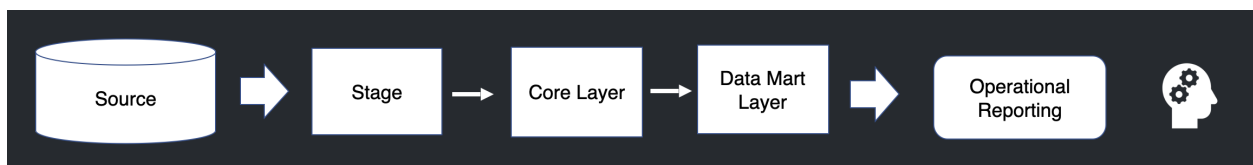
- Имея суррогатные ключи, можно генерировать **Сателлиты** и **Линки**

> Data Vault 2.0

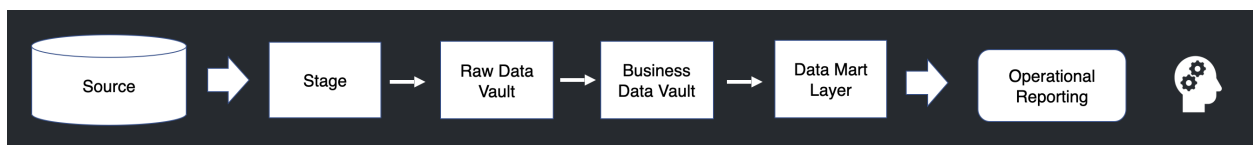
Сравнение подходов Кимбалла, Инмона, DV 1.0, DV 2.0



Кимбалл



Инмон и Data Vault 1.0



Data Vault 2.0

Инмон и DV 1.0 похожи. В Кимбалле отсутствует центральный слой как таковой, в то время как в Инмоне и DV 1.0 он есть, однако сделан по-разному:

- В подходе Инмона это таблицы в ЗНФ (классическое проектирование)
- В DV это знакомые нам Хабы, Линки и Сателлиты

При трансляции ER-диаграммы в DV 1.0 существует несколько правил:

- Все связи выносятся как **многие-ко-многим** — это **Линк**
- Все атрибуты сущности разделяются по **Сателлитам**

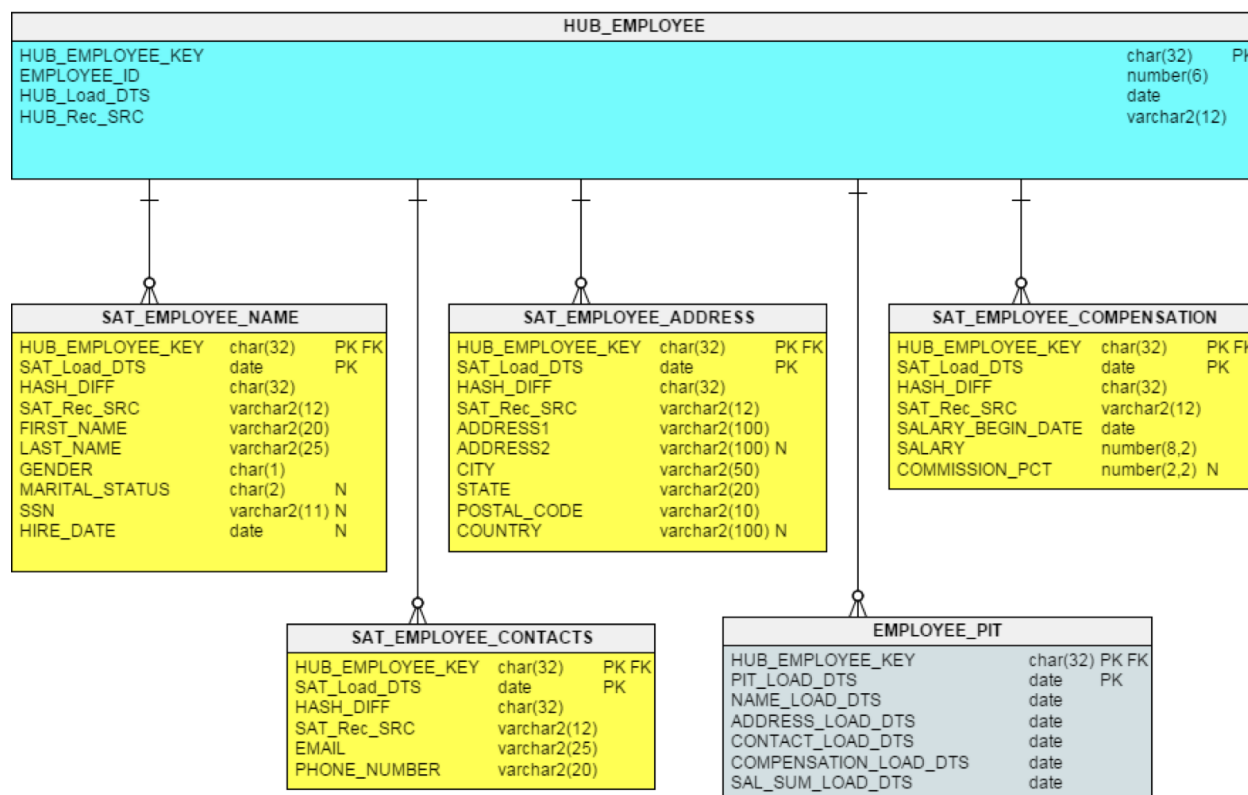
В DV 2.0 центральный слой делится на две части: *Raw Data Vault* и *Business Data Vault*.

В DV данные попадают такими, какими они были в источниках. При этом не происходит объединение данных в мета-сущности, такой как пользователи – нам могут приходиться различные идентификаторы пользователей (номера телефонов, рекламные идентификаторы и т.д.) и каждый из этих идентификаторов может являться отдельной сущностью в *Raw Data Vault*.

Когда такие сущности (как идентификаторы) объединяются в некоторую общую сущность (в пользователя), то это уже относится к *Business Data Vault*.

Новые сущности в DV 2.0

PIT



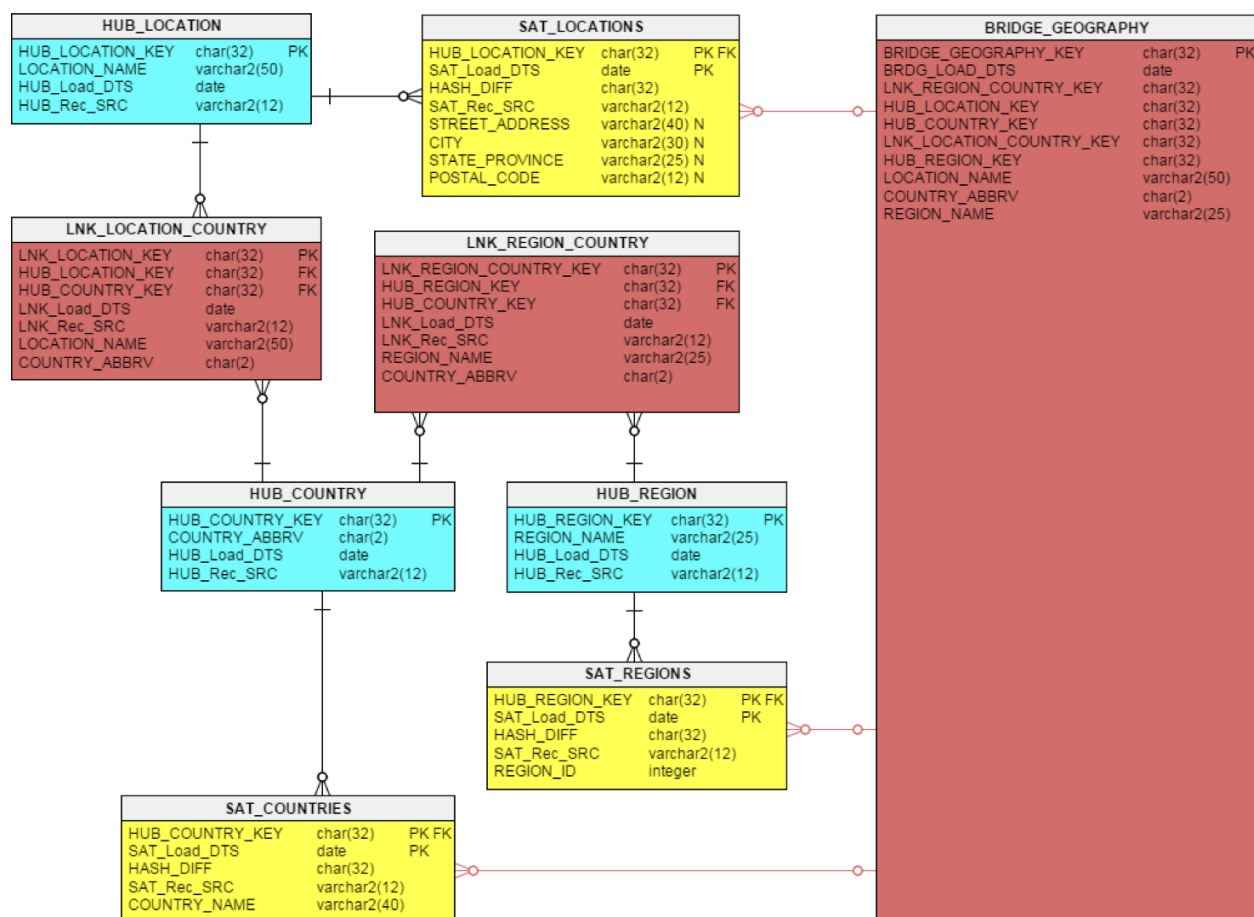
Пример PIT'a

PIT (point in time) упрощает получение информации из Сателлитов одной сущности с разной частотой обновления.

Так как частота изменений каждого отдельного Сателлита может быть разной, не все диапазоны SCD2 будут пересекаться. В связи с этим могут возникнуть трудности с восстановлением состояния записи на некоторый момент времени.

В PIT-таблице добавляются записи на каждое изменение каждого Сателлита. Фактически, это позволяет с помощью join'ов Хаба на PIT и Сателлиты воссоздать всю сущность на необходимый момент времени.

Bridge

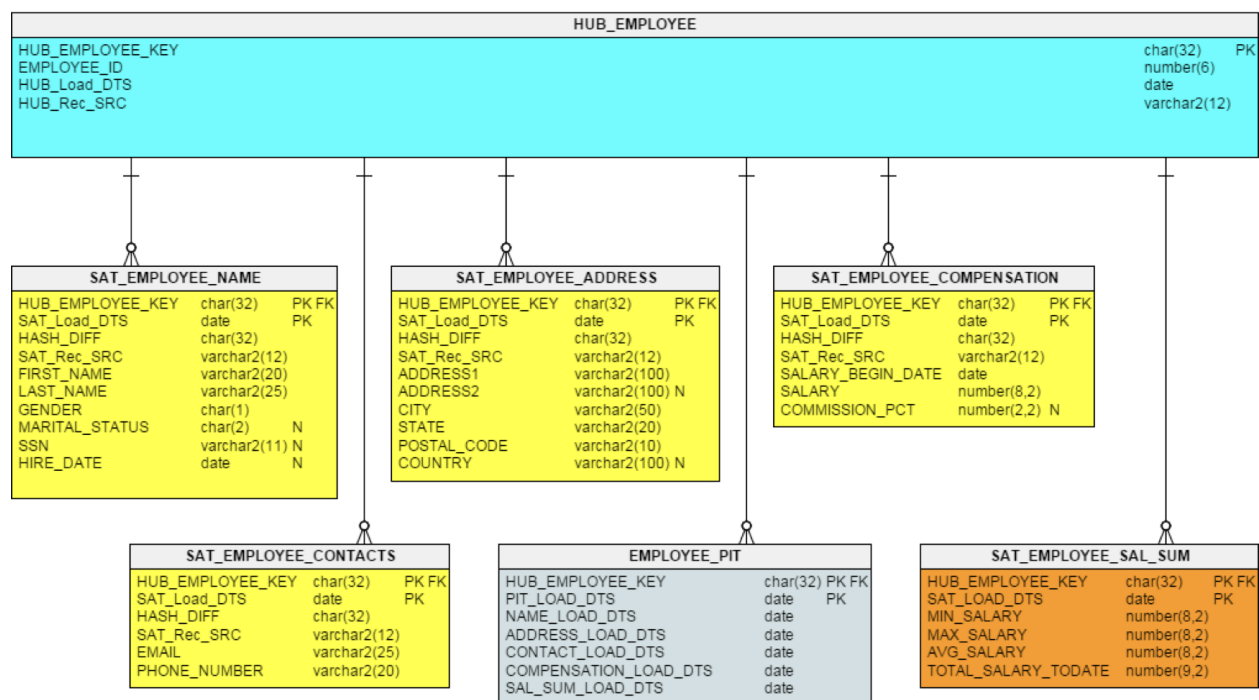


Пример Bridge

Мост (Bridge) упрощает соединение данных через несколько связей.

При наличии нескольких Линков, соединение всех таблиц в некоторую сущность может быть затруднено. Bridge упрощает это, так как фактически содержит в себе все Линки.

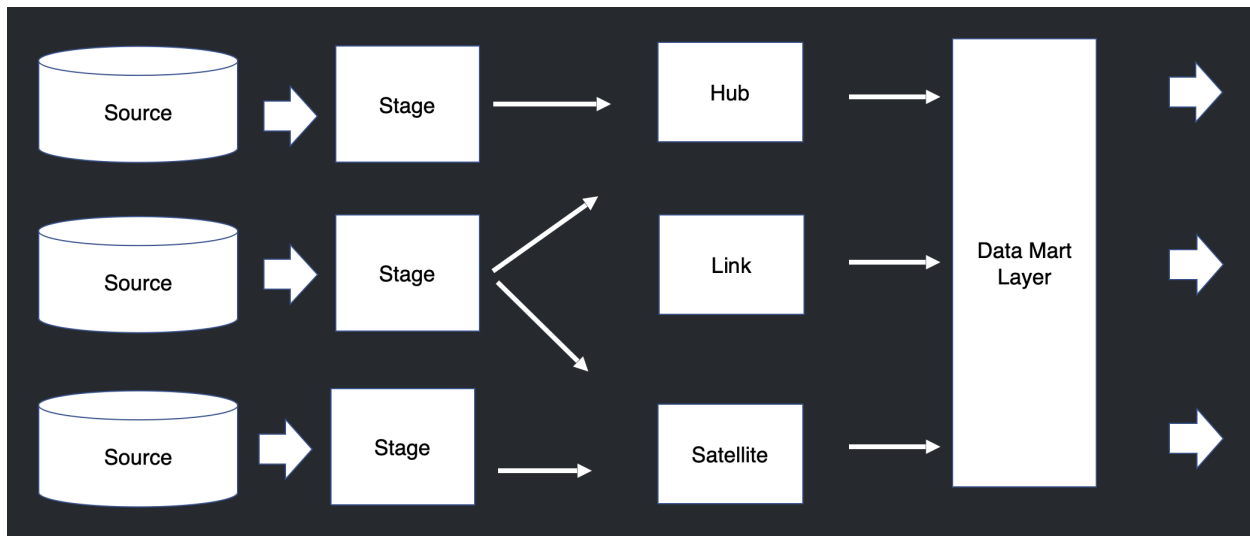
Predefined Derivations



Пример Predefined Derivations

Predefined Derivations (предагрегаты) представляет собой готовые, предагрегированные (или предкалькулированные) данные, такие как дата первой поездки или минимальное время нахождения на сервисе. Эти данные появляются на следующем этапе после загрузки Сателлитов.

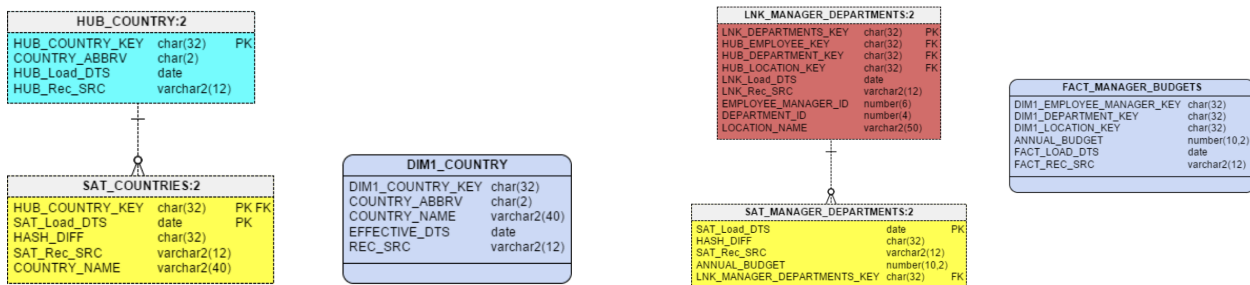
Загрузка DV 2.0



Пайплайн загрузки данных Data Vault 2.0

Важная особенность процесса загрузки заключается в **параллельности** загрузки Хабов, Линков и Сателлитов. Это достигается тем, что идентификатор генерируется как хешированное значение бизнес-ключа.

> Воссоздание таблицы измерений и таблицы фактов из DV



Построение витрин

Если присоединить Сателлит к Хаб, то мы получим готовое **измерение** по SCD2.

Построение таблицы фактов зависит от того, где находится транзакционная информация.

- Если транзакционная информация находится в Линке, то необходимо присоединить Сателлит к Линку. При этом все суррогатные ключи сущности – это ключи измерений.
- Если таблица фактов строится из какой-то сущности, то необходимо будет через Линки соединить все Хабы вокруг, чтобы получить их ключи и необходимую информацию из Сателлитов транзакционного Хаба. Данная таблица фактов будет построена на базе Хаба.

> Как проектировать по Data Vault?

1. Смоделируйте **Хабы**. Для этого требуется понимать основные бизнес-сущности (бизнес-ключи) и как они используются в выбранной области.
2. Смоделируйте **Связи**. Выявление возможных отношений между ключами – требует формулировки, как бизнес работает сегодня в контексте каждого бизнес ключа.
3. Смоделируйте **Сателлиты**. Это обеспечивает контекст, как каждой бизнес-сущности (бизнес-ключу), так и транзакциям/сделкам (Связям), соединяющим Хабы вместе. После этого начинается проявляться полная картина о бизнесе.

Основная проблема при проектировании по DV – Join'ы

Уже на этапе добавления Сателлитов и Линков, мы сильно увеличиваем количество join'ов, необходимых для построения цельных сущностей.

Неподготовленному аналитику будет очень тяжело работать с DV – много join'ов повышают когнитивную сложность для человека и вычислительную для системы.

Для менее опытных аналитиков имеет смысл собирать классические витрины, с которыми ему будет сильно проще работать.

Не следует применять Data Vault там, где join'ы плохо работают, ознакомьтесь со спецификой своей СУБД – позволяет ли она быстро делать много join'ов?

> Дополнительные материалы

- Хеширование
- Транзакционные ссылки – что это и почему это плохо