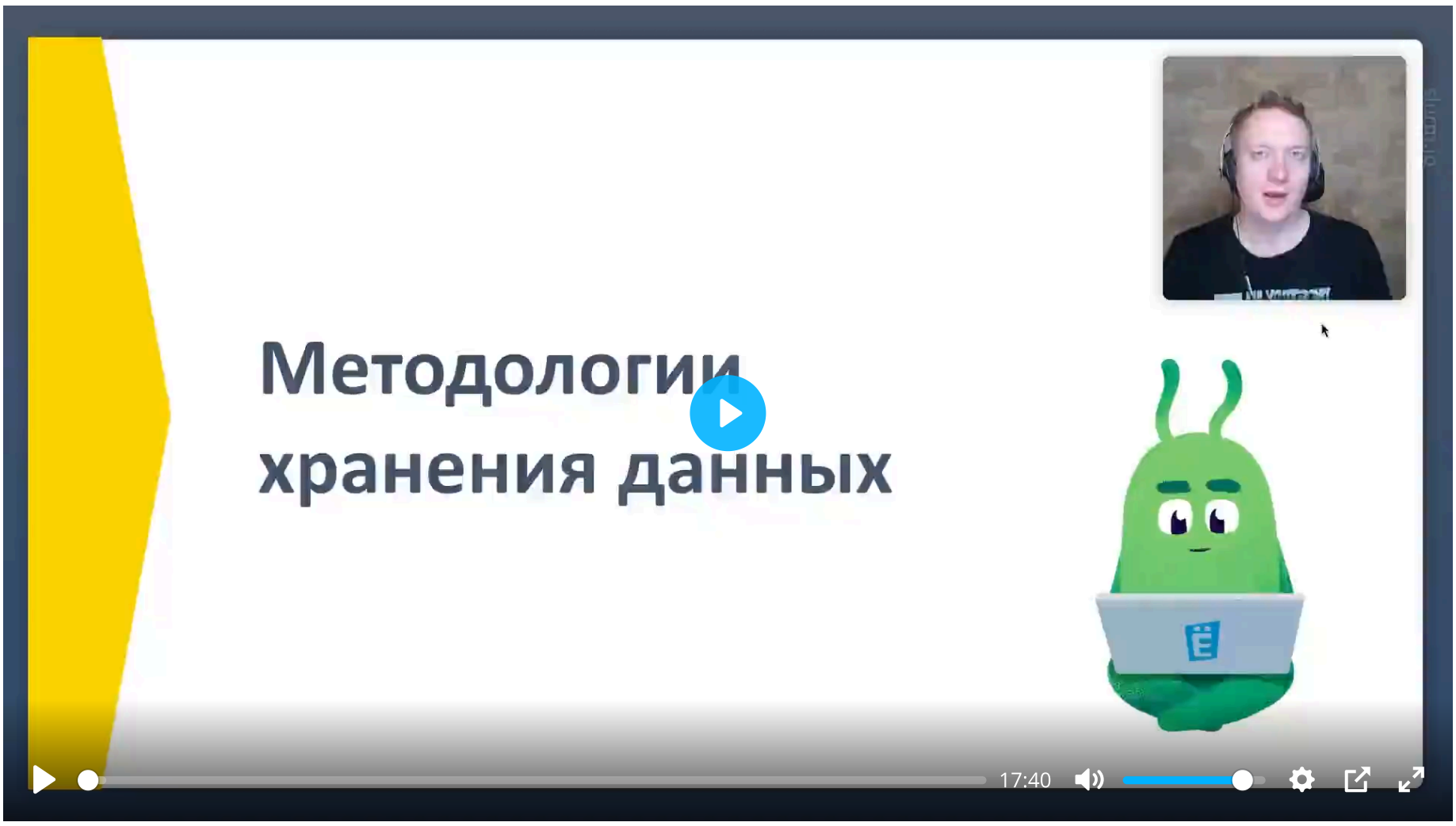
11.2 Традиционная архитектура хранилищ данных



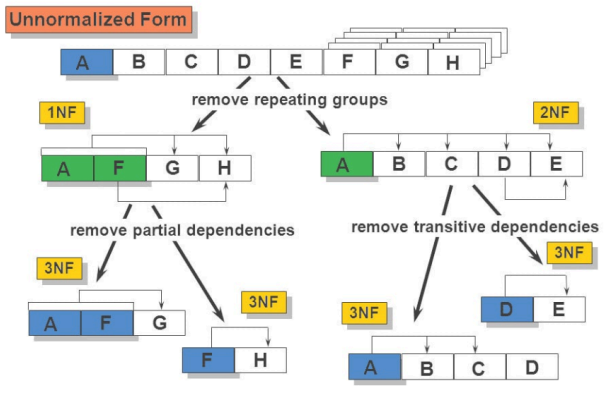
Тексовая расшифровка видео:

МЕТОДОЛОГИИ ХРАНЕНИЯ ДАННЫХ

План:

- Базовые проблемы хранилищ;
- Бессмертная классика;
- Подход Кимбалла;
- Звезда и снежинка;
- Подход Инмона;
- Data Vault 1.0/2.0;
- Anchor Modeling.

Базовые проблемы хранилищ



Вопросы, которые могут возникнуть:

- До какой степени нарезать на кусочки (нормализовать)?
- Как поддерживать историчность данных?

- Как мигрировать и версионировать схему данных?
- В каком формате хранить данные на диске?
- Нужно ли поддерживать SQL?

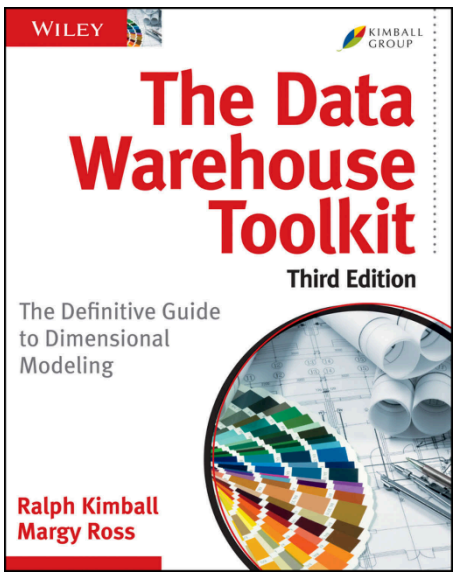
Один из самых важных вопросов: «как именно выстроить хранилище таким образом, чтобы не пришлось его переделывать?».

Появились разные методологии того, как проектировать хранилища, чтобы они жили как можно дольше, а также служили качественным источником информации для растущего бизнеса.

Бессмертная классика

Изначально было несколько исследователей, которые активно занимались работой с Data Warehouse, проектированием, структурой, методологиями и т.д.

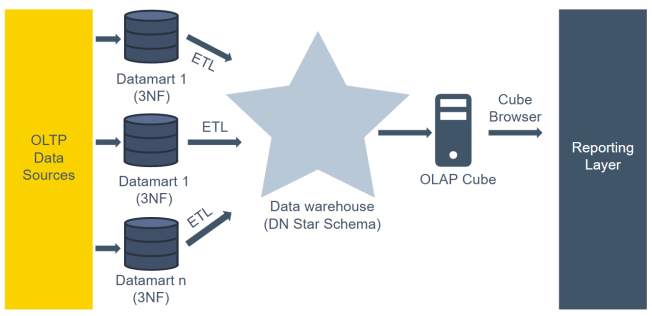
Ralph Kimball считается отцом современного Data Warehouse'инга.



В этой книге находятся основы основ.

Подход Кимбалла

Кимбалл предлагал свою модель хранения, которая сейчас может показаться немного странной.



Основная идея: исследователь предлагал хранить данные по схеме звезда. На изображении вы видите, что Datamart'ы (витрины) находятся не справа, а слева. У нас есть разные подразделения, где каждое генерирует свой собственный набор данных, свои датасеты. Кимбалл предлагал в рамках каждого из этих подразделений сначала сформировать доменную витрину, а потом совокупность всех витрин объединить. Это объединение будет считаться Data Warehouse. Поверх Data Warehouse можно стоить OLAP-кубы, строить многомерные выборки, которые будут использоваться как основной источник для дальнейшей аналитики.

Data Warehouse по Кимбаллу представляет собой совокупность всех Datamart'ов. Каждый Datamart по отдельности находится в третьей нормальной форме. Вместе они объединяются в форму «Звезда».

Звезда и снежинка

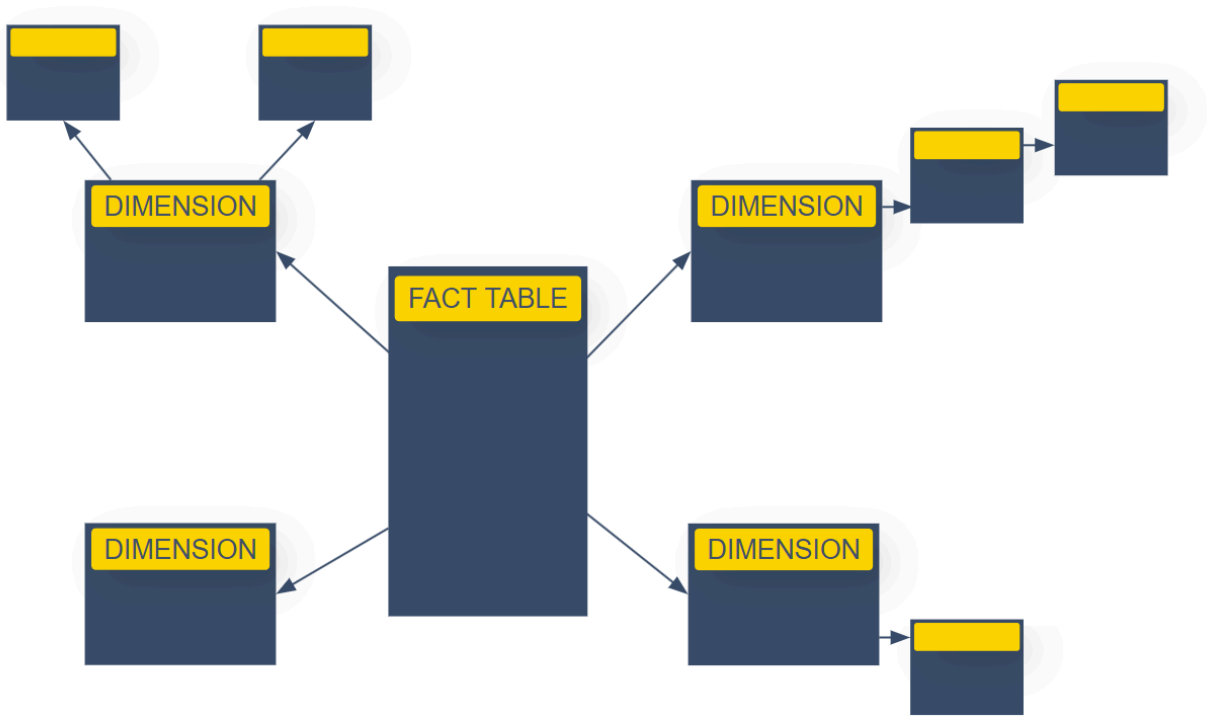


Схема «Звезда» представляет из себя следующее:

Когда в системе происходит какое-то событие, в этом событии задействованы бизнес-объекты, пользователь, продукт и т.д. Получается какое-то количество существительных (здесь они называются «Dimension»). Когда появляется взаимосвязь между этими объектами, появляется запись в табличке фактов. **Факт** – это логи того, что происходит в системе. Каждая строчка лога содержит Foreign key на какие-то Dimension'ы, являющиеся объектами, которые в этой системе поучаствовали.

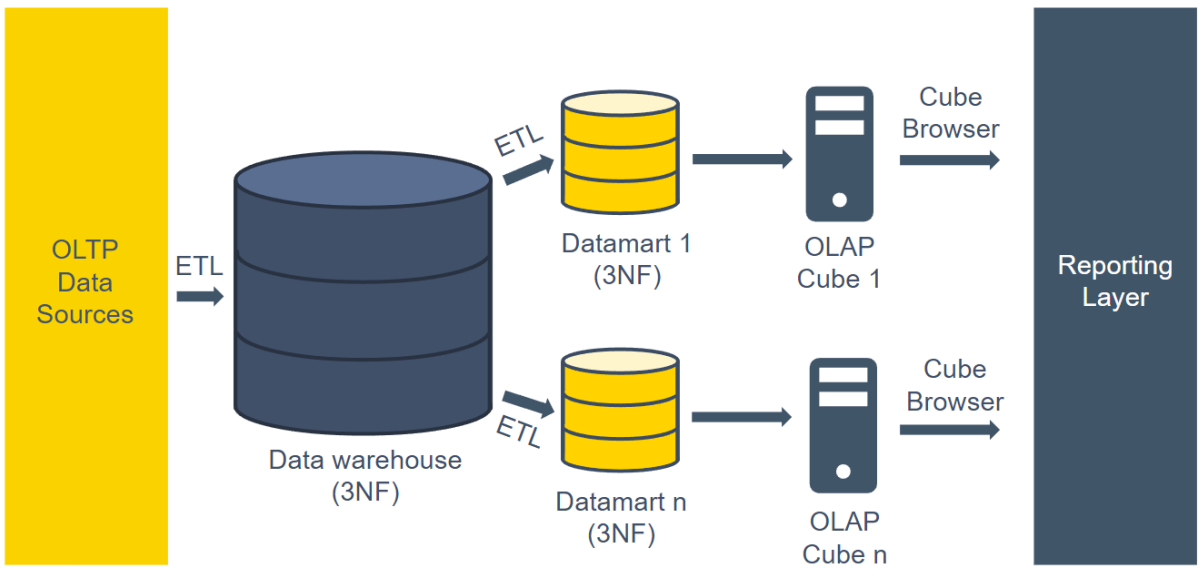
- Схема, если мы рассматриваем только один уровень Dimension'ов, когда есть таблица фактов и первый уровень, называется «**Звезда**».
- Если есть дальнейшие уровни, то схема называется «**Снежинка**».

Удобство этой методологии построения хранилища в том, что оно очень простое. Однако в подобном хранилище не очень удобно отслеживать историю изменений конкретных объектов/параметров. Также есть и другие вопросы, например, «как мы будем заливать данные?».

Существует проблема, которая называется «**Late Arriving Fact**», когда пришел факт, а Dimension'а для этого нет; когда пришла запись об изменении в каком-то Dimension'е, а мы о нем ничего не знаем. Также разные параметры объектов из разных источников могут по-разному обновляться с разной периодичностью.

Подход Инмона

Билл Инмон – исследователь, который описывал схожую структуру, но немного иначе:



Хранилище представляет собой БД в третьей нормальной форме. На основе этого хранилища мы строим отдельные Datamart'ы с помощью ETL-процессов. Поверх этих Datamart'ов можем строить OLAP-кубы, делать выборки и генерировать отчеты.

Data Vault 1.0/2.0

Поговорим о недостатках:

- В частности, из разных источников данные в объекте могут обновляться по-разному.
- Если что-то в структуре серьезно меняется, то единственный способ что-то сделать – создать новую табличку или делать большую миграцию.

Появились современные и молодые методологии проектирования хранилищ. Одна из них на сегодняшний день очень популярна и называется «**Data Vault**».

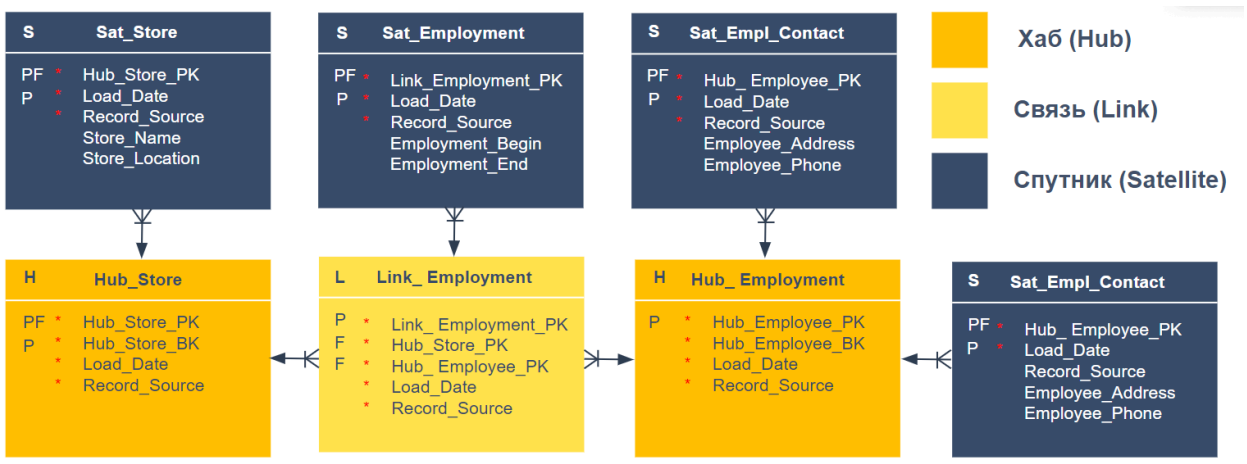
Data Vault существует в двух версиях:

- 1.0;
- 2.0.

Основная идея:

У нас есть три типа объектов:

- **Хабы** (бизнес-существительные), где каждая строчка – набор полей, которая однозначно идентифицирует бизнес-объект.
- У каждого из объектов есть набор полей (они периодически меняются). Появляется такое понятие, как «**Satellite**». Satellite – это таблички с наборами дополнительных полей, которые содержат внешний ключ. Табличка связана с хабом. В ней перечислены наборы полей. Если меняется набор полей, мы можем долгое время обходиться без серьезных миграций.
- Между разными объектами существует **связь (link)**. Иногда существует дополнительная сущность – **Satellite на link**, когда, условно, факт – это осуществление покупки, и есть **Satellite на link,** описывающий параметры этой покупки. Однако старайтесь этого избегать.



Разница между 1.0 и 2.0 заключается в следующем:

Если мы используем в качестве суррогатного ключа, например, поле с автоинкрементом, возникает проблема: если мы записываем записи в таблицу, то, не получив ID новой записи, мы не сможем вставить записи, связанные с ней. Для того, чтобы загрузить данные, нам придется пройтись несколько раз, залив первый уровень, затем второй и третий для того, чтобы решить проблему с ID. Поэтому была придумана концепция Data Vault 2.0.

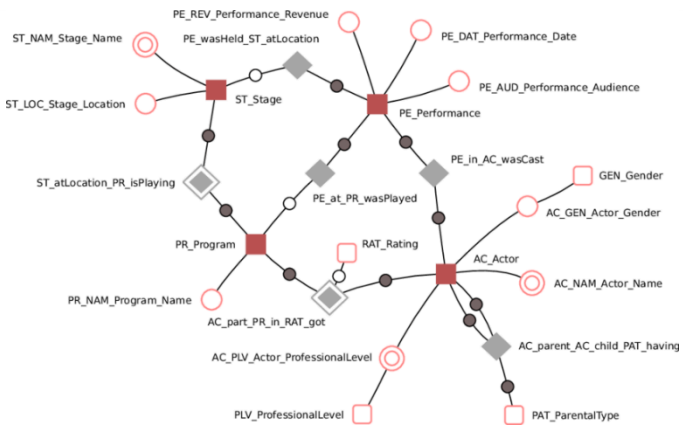
В 2.0. пошли на следующий шаг:

В качестве суррогатных ключей используются не числа, а хэши. Каждый объект идентифицируется набором полей. Если человека идентифицируем по набору параметров, то можно объединить параметры и посчитать хэш. До тех пор, пока есть этот набор полей, мы можем считать хэш в любой момент.

В секции с литературой мы прикрепили ряд полезных книг по Data Vault. Рекомендуем к чтению.

Anchor Modeling

Anchor Modeling – это шестая нормальная форма:



Существует шесть нормальных форм, то есть, на каждый атрибут, который есть у объекта, создается отдельная таблица.

При таком подходе происходит взрывной рост таблиц, но при этом получается идеально-гибкое хранилище, в котором можно любую бизнес-сущность идентифицировать по любому набору полей.

К сожалению, баз, современных хранилищ, которые полноценно поддерживают Anchor Modeling, не так много. В основном это коммерческие решения, например, T**eradata, Vertica, Microsoft Secret Server**. В клауде популярен **Snowflake**.

Как вам урок? 😠 😐 🔥

Изучил, далее ›