



# > Конспект > 6 урок > Сравнение методологий проектирования

> OLAP vs OLTP

> Аналитическое хранилище данных

Сравнение

Что выбрать?

> Аналитическое хранилище данных и Озеро данных

> Lambda-архитектура

> Нормализация

1НФ, 2НФ и 3НФ

> Витрина

> Трудоемкость развития хранилища данных

> Общее сравнение подходов к проектированию

> Дополнительные материалы

---

## > OLAP vs OLTP

OLAP-системы должны быть организованы иначе, чем OLTP-системы.

- Для выполнения аналитических запросов необходима обработка информации из разных источников.

- Для выполнения запросов, связанных с прогнозированием, анализом тенденций, необходимы исторические данные, накопленные за достаточно длительный период, что не всегда обеспечивается OLTP-системами
- Данные, используемые для целей анализа данных обслуживания аналитических запросов, отличаются от используемых в OLTP-системах. При аналитической нагрузке можно пользоваться не детальными, а агрегированными данными.

<u>Property</u>	<u>OLAP</u>	<u>OLTP</u>
<u>Степень детализации</u>	Хранение детализированных и обобщенных данных	Хранение обобщенных данных
<u>Формат хранения</u>	Единый согласованный	Варьируется от задач
<u>Допущение избыточности</u>	Контролируемая избыточность	Максимальная нормализация
<u>Управление данными</u>	Периодическое добавление данных	Добавление/удаление/изменение в любое время
<u>Количество хранимых данных</u>	Должны быть доступны все оперативные данные	Должны быть доступны все данные, в том числе исторические
<u>Характер запросов к данным</u>	Произвольные запросы (анализ данных)	Заранее составленные запросы

Понимание различий этих систем понадобится, когда вам, как дата-инженеру, нужно будет разговаривать с backend-разработчиками, которые, как правило, имеют профессиональный уклон в сторону OLTP-систем. Вам нужно уметь объяснить, чем системы отличаются, почему ваша система работает именно так, характер нагрузки этой системы и другие многие аспекты.

## > Аналитическое хранилище данных

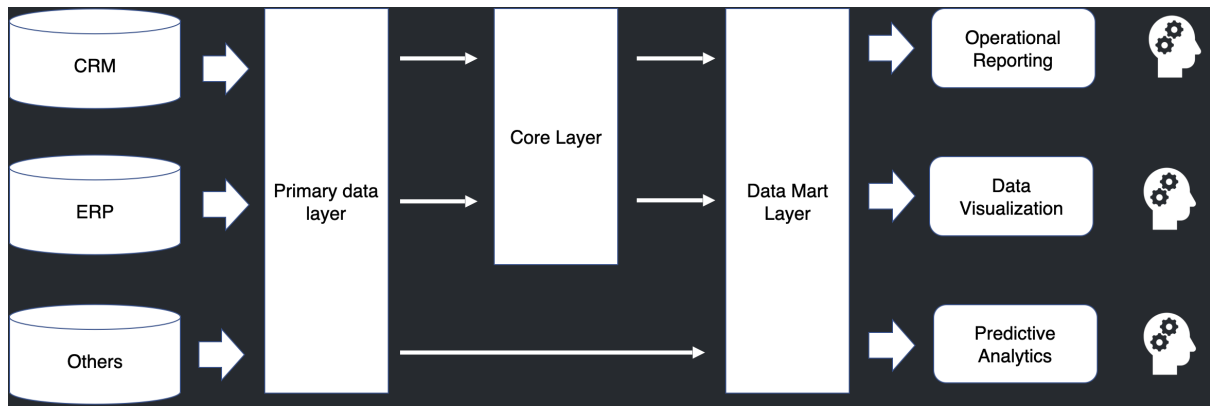
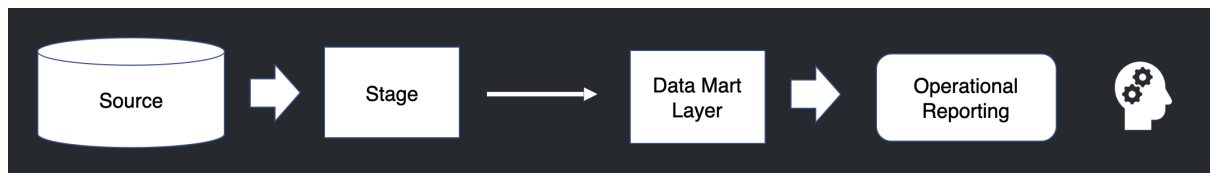


Схема слоев данных

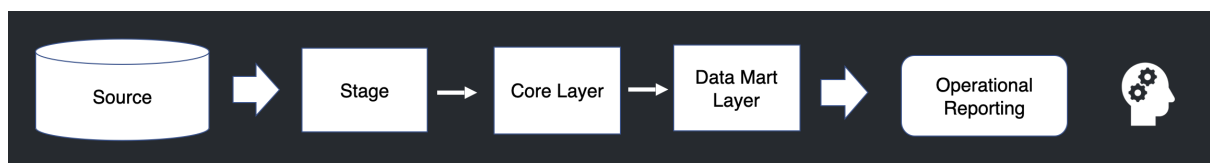
Слои данных позволяют снизить градус сложности и изолировать каждый следующий слой. Без них наше хранилище это как один большой черный ящик, с которым очень сложно работать.

В зависимости от наличия центрального слоя (Core Layer) разделяются подходы – по Кимбаллу и по Инмону.

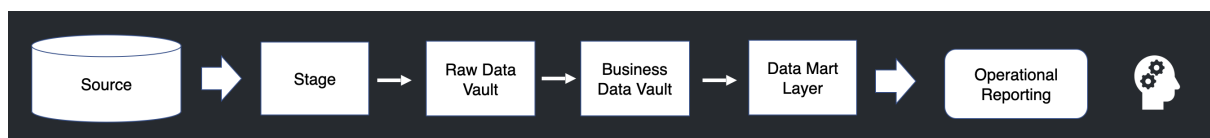
## Сравнение



Кимбалл



Инмон и Data Vault 1.0



Data Vault 2.0

По Кимбаллу мы имеем только Stage-слой (первичный), в котором мы захватываем данные из источника, и сразу по этим данным мы строим витрины.

Инмон, DV 1.0 и Якорное моделирование предполагают наличие центрального слоя, в который мы проецируем бизнес на сущности, и уже поверх этих сущностей строим витрины.

В DV 2.0 так же присутствует центральный слой, отличается он тем, что разделяется еще на два дочерних слоя: Raw Data Vault и Business Data Vault.

## Что выбрать?

**Сроки.** Нужно помнить, что на проектирование по Инмону требуется больше времени, чем на реализацию подхода Кимбалла. В условиях сжатых временных рамок следует строить ХД по Кимбаллу.

### Требования заказчика.

- Если требования заказчика носят более общий, стратегический характер, если необходимо показывать состояние бизнеса в целом, отслеживать взаимосвязи между бизнес-сущностями, следует выбрать Инмона.
- Если поставлены конкретные цели и нужны определенные отчеты по ограниченному множеству процессов и с ними связанных сущностей, следует выбрать Кимбалла.

**Частота изменений.** В условиях развивающегося бизнеса, когда постоянно добавляются новые и изменяются старые сущности, следует выбрать Инмона.

## > Аналитическое хранилище данных и Озеро данных

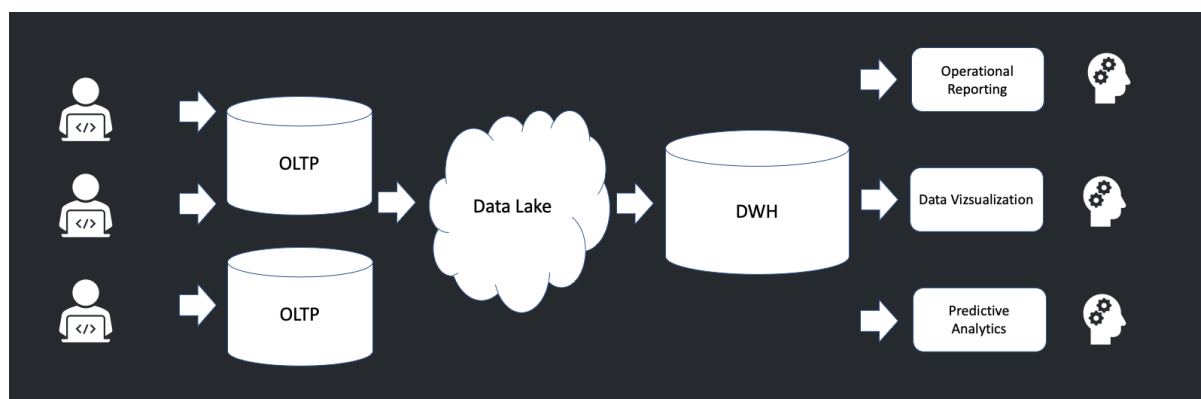




Схема использования OLTP-систем с Data Lake и DWH

На схеме выше мы имеем и Озеро данных, и DWH – оба этих подхода могут быть использованы вместе. При этом может не быть как Озера (останется только DWH), так и наоборот.

## Сравнение

 Аналитическое хранилище	 Озеро данных
<u>Хранятся только полезные данные, актуальные в текущем периоде времени</u>	Хранятся все данные, в том числе и «бесполезные», которые могут пригодиться в будущем или же не понадобятся никогда
<u>Четко структурированные данные одного формата</u>	Структурированные, полуструктурированные и неструктурированные разнородные данные любых форматов: от мультимедиа файлов до текстовых и бинарных из разных источников
<u>Низкая гибкость: структура и типы данных продумываются заранее и не подлежат изменению в процессе эксплуатации</u>	Высокая гибкость, которая позволяет в процессе эксплуатации добавлять новые типы и структуры данных
<u>Благодаря четкой структуре данных процесс их извлечения и обработки происходит быстро</u>	Из-за отсутствия четкой структуры необходима дополнительная обработка данных для их практического использования
<u>Достаточно высокая стоимость проектирования из-за сложности</u>	Озеро данных дешевле DWH с точки зрения проектирования

Озеро данных позволяет быстрее проводить необходимые эксперименты (так как дешево обходится в проектировании). При этом на Озере данных можно построить подобие аналитического хранилища, однако не стоит реализовывать на Озере данных что-то, обильно использующее join'ы вроде (DV/AM). Озера данных, как правило, построены на подходе Map-Reduce, который не позволяет эффективно делать join'ы.

В аналитическом хранилище, напротив, не стоит хранить неструктурированные данные – ваша реляционная СУБД, скорее всего, плохо к этому отнесется.

Поэтому полезно иметь как Аналитическое хранилище (для структурированных данных), так и Озеро данных (для неструктурированных).

## > Lambda-архитектура

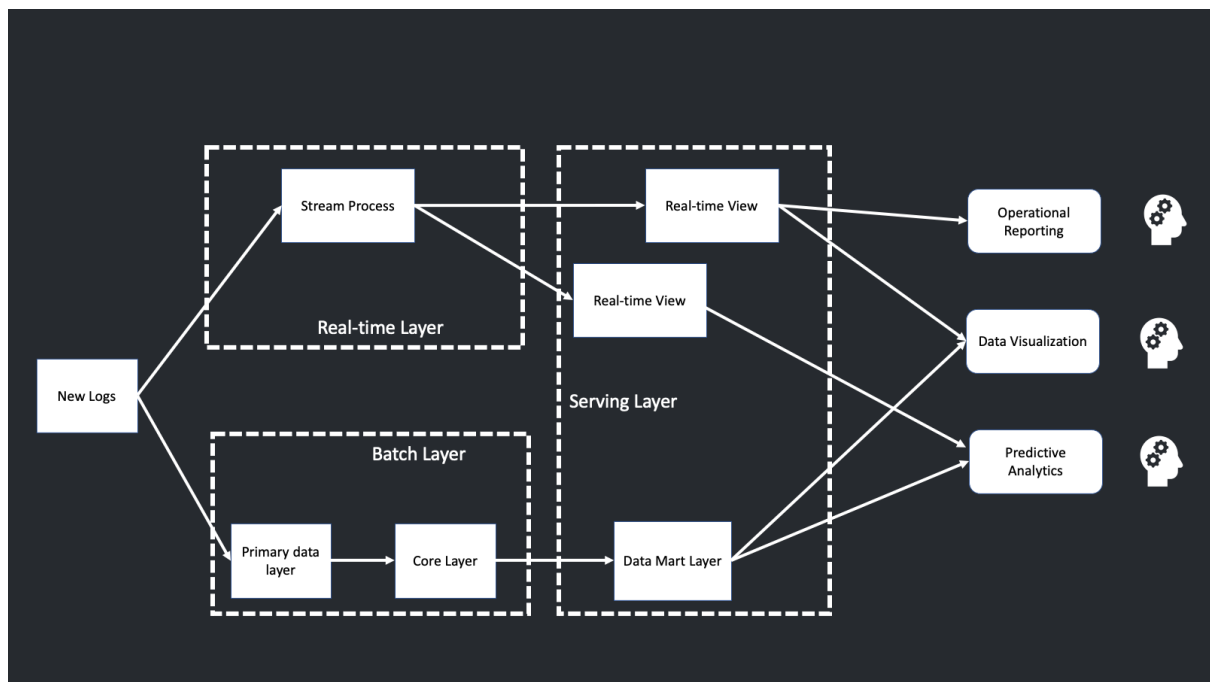


Схема Lambda-архитектуры

В Lambda-архитектуре мы имеем как потоковую обработку, так и пакетную. Аналитики же обращаются в оба слоя с помощью BI-инструментов.

### Сравнение

Аа Потоковая обработка	≡ Пакетная обработка
<u>Результат в режиме реального времени</u>	Данные доставляются с задержкой
<u>Сложная разработка и тестирование (относительно).</u>	Простая разработка и тестирование (относительно)
<u>Низкая эффективность для OLAP-систем</u>	Высокая эффективность для OLAP-систем
<u>Равномерная нагрузка на железо</u>	Пиковая нагрузка на железо

В целом, сейчас потоковая обработка данных и Lambda-архитектура в целом – это современный стандарт построения хранилища данных.

## > Нормализация

### 1НФ, 2НФ и 3НФ

Первая нормальная форма (1НФ) – это обычное отношение. Отношение в 1НФ обладает следующими свойствами: в отношении нет одинаковых

кортежей; кортежи не упорядочены; атрибуты не упорядочены; все значения атрибутов атомарны.

Отношение находится во **второй нормальной форме (2НФ)** тогда и только тогда, когда отношение находится в 1НФ и нет неключевых атрибутов, зависящих от части сложного ключа.

Отношение находится в **третьей нормальной форме (3НФ)** тогда и только тогда, когда отношение находится в 2НФ и все неключевые атрибуты взаимно независимы.

---

В целом, нормальные формы необходимы для **устранения аномалий**.

---

### Сравнение нормальных форм

<u>Аа</u> Критерий	☰ Отношения слабо нормализованы (1НФ, 2НФ)	☰ Отношения сильно нормализованы (3НФ)
<u>Адекватность базы данных предметной области</u>	ХУЖЕ (-)	ЛУЧШЕ (+)
<u>Легкость разработки и сопровождения базы данных</u>	СЛОЖНЕЕ (-)	ЛЕГЧЕ (+)
<u>Скорость выполнения вставки, обновления, удаления</u>	МЕДЛЕННЕЕ (-)	БЫСТРЕЕ (+)
<u>Скорость выполнения выборки данных</u>	БЫСТРЕЕ (+)	МЕДЛЕННЕЕ (-)

Как слабая нормализация, так и сильная подходят для построения хранилища данных.

Если аналитики (или другие пользователи данных) менее квалифицированные специалисты и хуже разбираются в различных методологиях и написании сложных запросов, то лучше выбрать слабую нормализацию с большими плоскими таблицами.

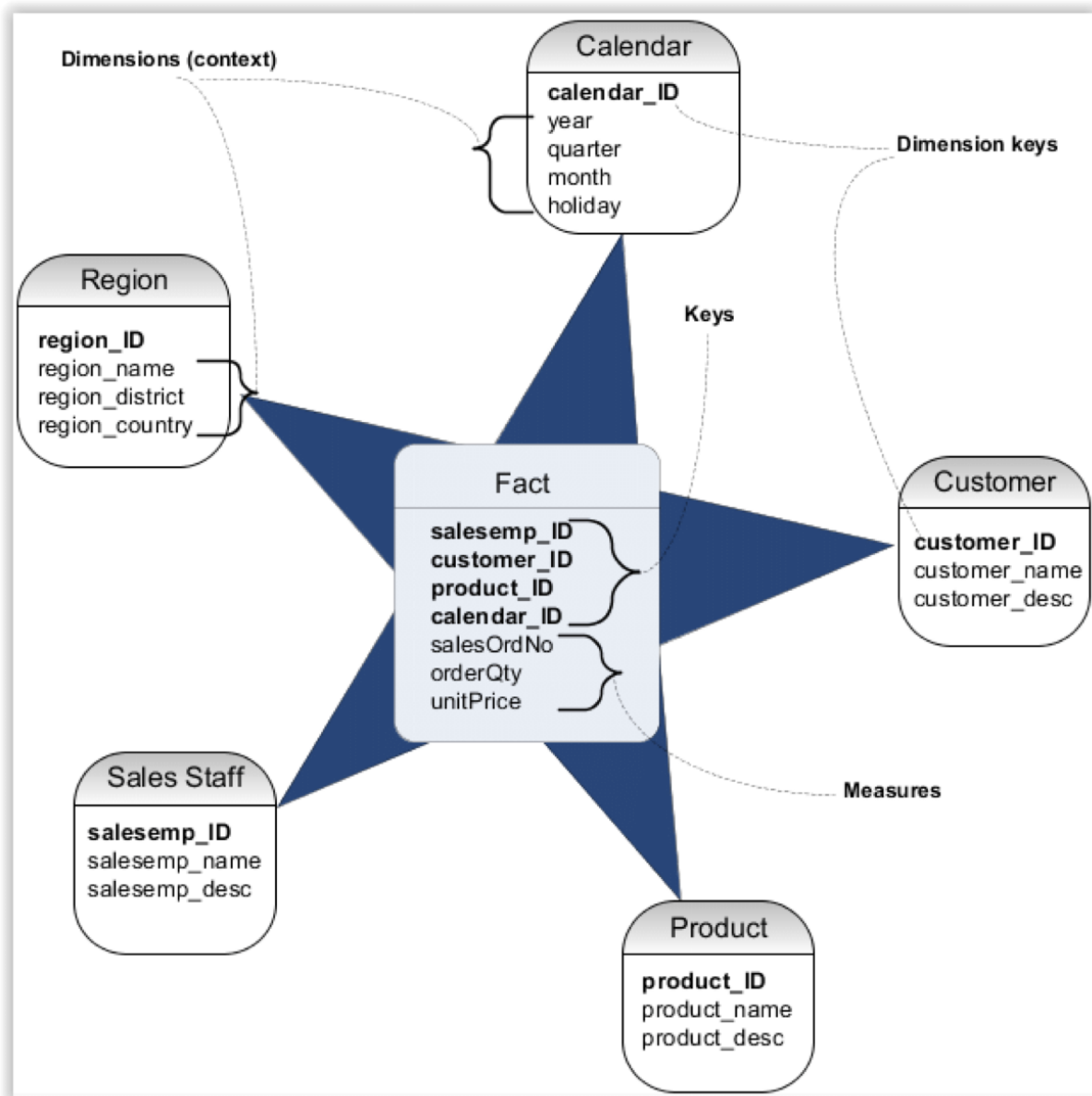
Если аналитики имеют достаточную квалификацию, то лучше выбрать сильную нормализацию – ее проще обслуживать/пересчитывать.

Есть также гибридный подход – когда, например, мы делаем таблицы фактов и измерений, но какие-то таблицы или часто выполняемые запросы с join'ами мы материализуем в виде плоских витрин.

---

## > Витрина

**Витрина данных (Data Mart)** представляет собой срез хранилища данных в виде массива тематической, узконаправленной информации, ориентированного, например, на пользователей одной рабочей группы или департамента.



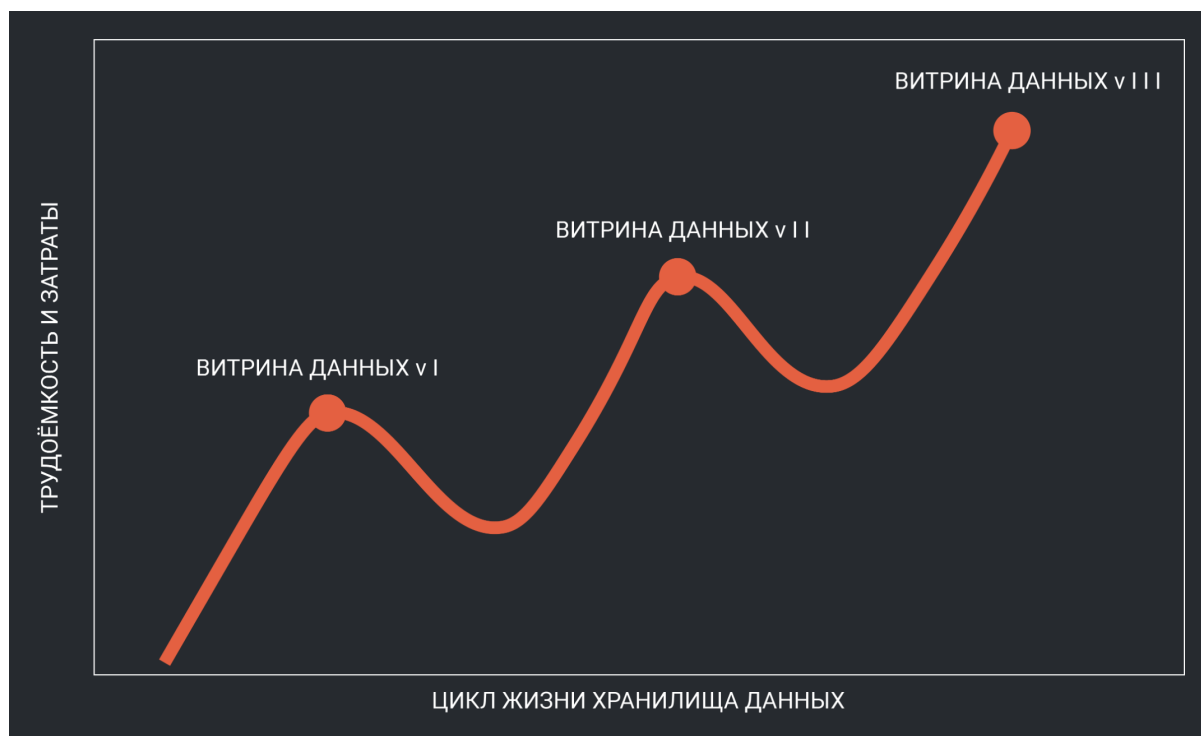
Витрина данных, аналогично дашборду, позволяет аналитику увидеть агрегированную информацию в определенном временном или тематическом разрезе, а также сформировать отчетные данные в виде шаблонизированного документа.

Не все аналитики хорошо разбираются в хранилищах данных и понимают принципы их построения. Если аналитик не работал ранее с таблицей фактов и



измерений, вы, как дата-инженер, должны уметь объяснить ему смысл и принцип использования этих таблиц.



## > Трудоемкость развития хранилища данных



Если идти по подходу Кимбалла, то каждую следующую витрину данных разрабатывать все сложнее.

Для решения этой проблемы существуют два подхода, которые мы рассматривали ранее – Data Vault и Anchor Modeling.

### Сравнение

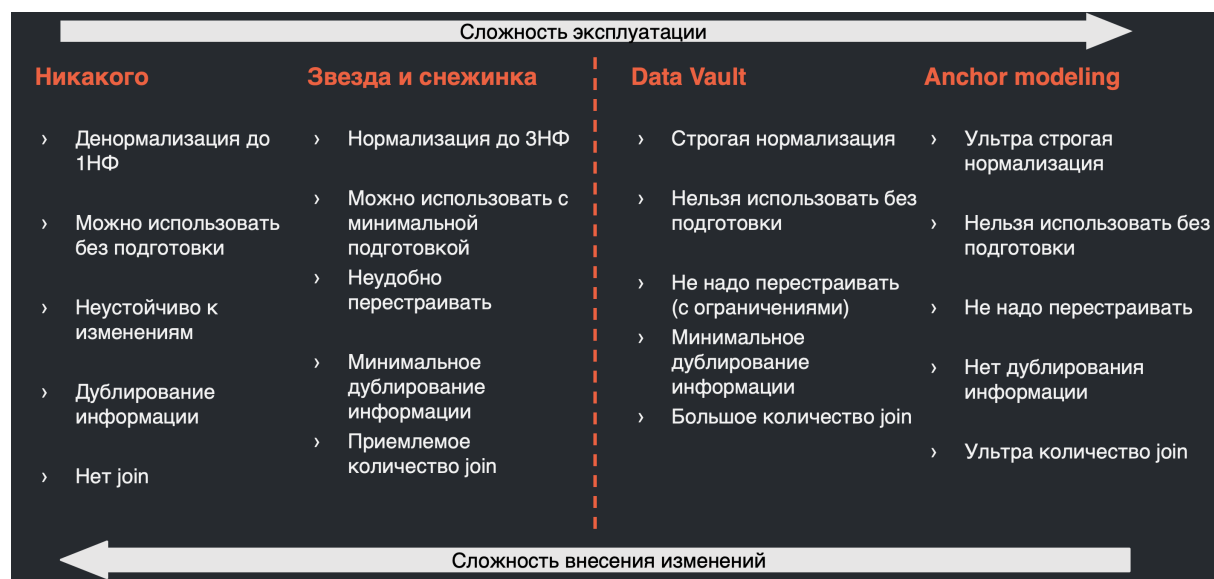
 Data Vault	 Anchor Modeling
<u>На каждую сущность создается hub – таблица с бизнес-ключом и суррогатным ключом</u>	На каждую сущность создается <b>anchor</b> – таблица только с суррогатным ключом
<u>Атрибуты группируются в таблицы-сателлиты по принципам совместности: изменения и/или источника и/или использования</u>	Один <b>атрибут</b> – одна таблица, да здравствует 6НФ

<b>Aa Data Vault</b>	<b>Anchor Modeling</b>
<u>Связи только через отдельные таблицы, на них можно навесить сателлит</u>	<b>Связи</b> только через отдельные таблицы, никаких атрибутов
<u>Есть специальные таблицы Point-in-Time и Bridge</u>	Есть специальная таблица <b>knot</b> – статический справочник

В любом новом хранилище лучше использовать Data Vault, так как это более простой для осмысления подход. Для использования якорной модели нужна хорошая автоматизация процессов.

Также можно комбинировать Data Vault и якорную модель. Так, например, сделали в Яндекс.Go – назвали этот подход **HNHM**.

## > Общее сравнение подходов к проектированию



Чем строже нормализация, чем больше join'ов и сущностей – тем сложнее хранилище в эксплуатации. Но тем проще вносить изменения в хранилище.

Если вы строите хранилище с нуля, начинать стоит с сочетания Data Vault и звезды/снежинки. При этом звезда/снежинка (таблицы фактов и измерений) – это Кимбалловский подход, помогающий быстрее получить результат. Но при этом не стоит оставаться в Кимбалловском подходе надолго – нам все равно необходим core-слой, который как раз таки можно строить на Data Vault.

Якорную модель стоит использовать только при наличии хорошей автоматизации или разработчиков, которые способны реализовать эту автоматизацию. А Data Vault можно поддерживать руками системных аналитиков и инженеров-данных.

---

## > **Дополнительные материалы**

1. Комбинация подходов Data Vault и Anchor Modeling в Яндекс.GO – [HNHM](#)