

# DWH

Лекция №4  
Проектирование схемы БД по схеме Data Vault,  
Anchor modeling.



В прошлой  
лекции



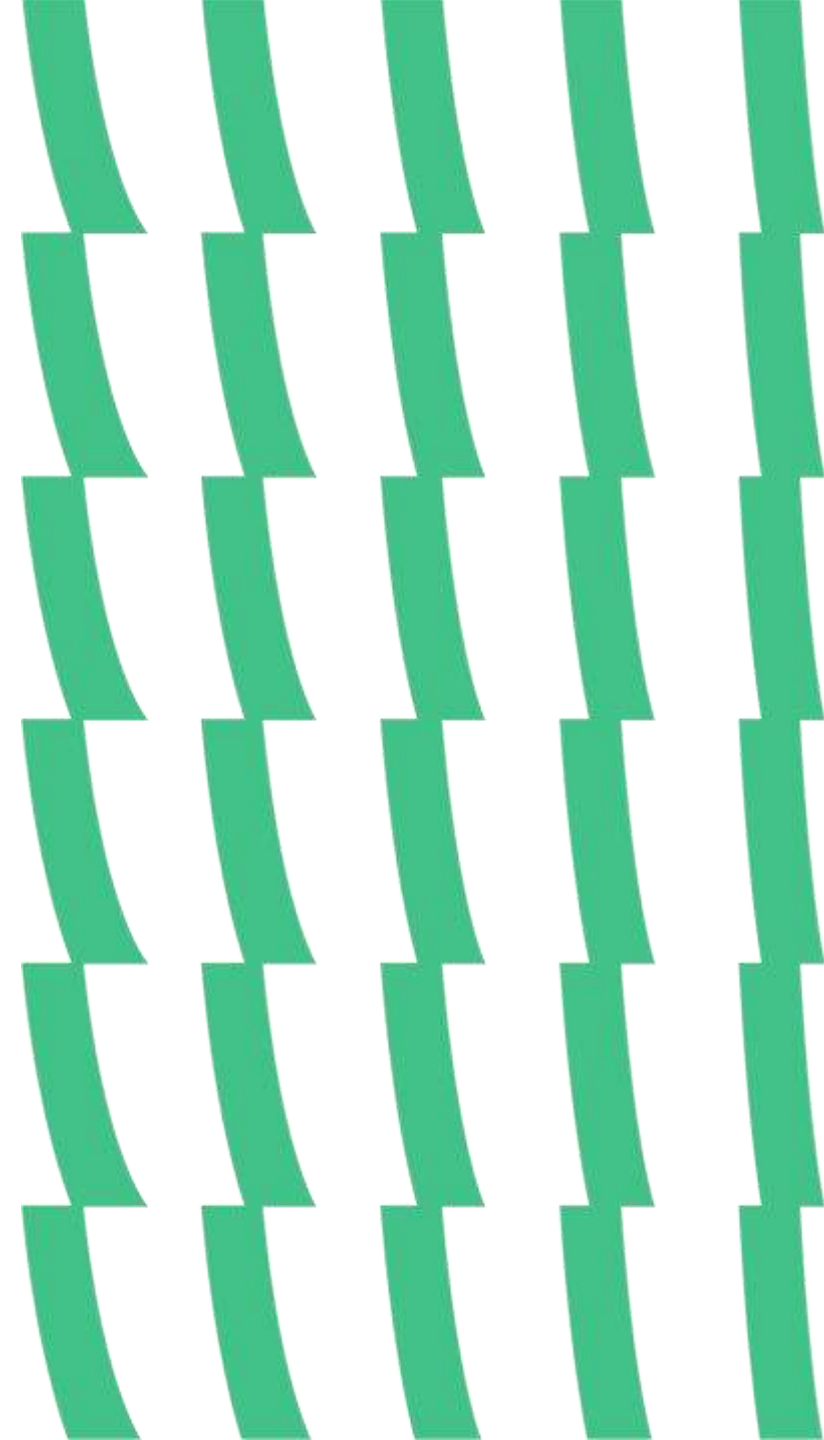
# Факты и измерения

**факты** - численные параметры, данные, которые количественно описывают процесс, они непрерывны, т. е. могут принимать бесконечное количество значений.

**измерения** - текстовые параметры для предоставления максимально возможного контекста для фактов(могут быть числовыми, но в любом случае это дискретные данные).

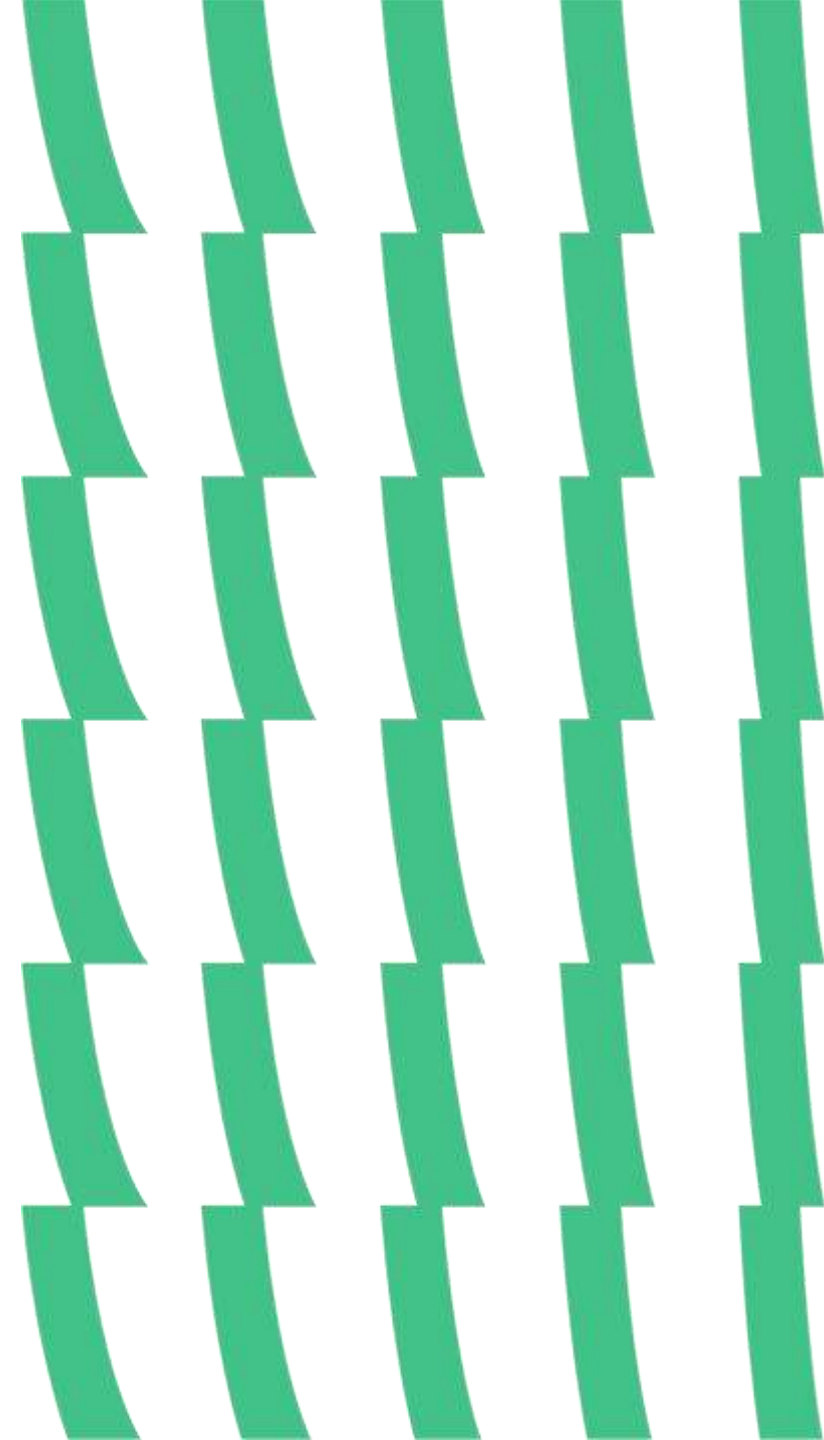
Основной таблицей хранилища данных является **таблица фактов**. Обычно такие таблицы содержат сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться.

**Таблицы измерений** содержат практически неизменяемые данные.



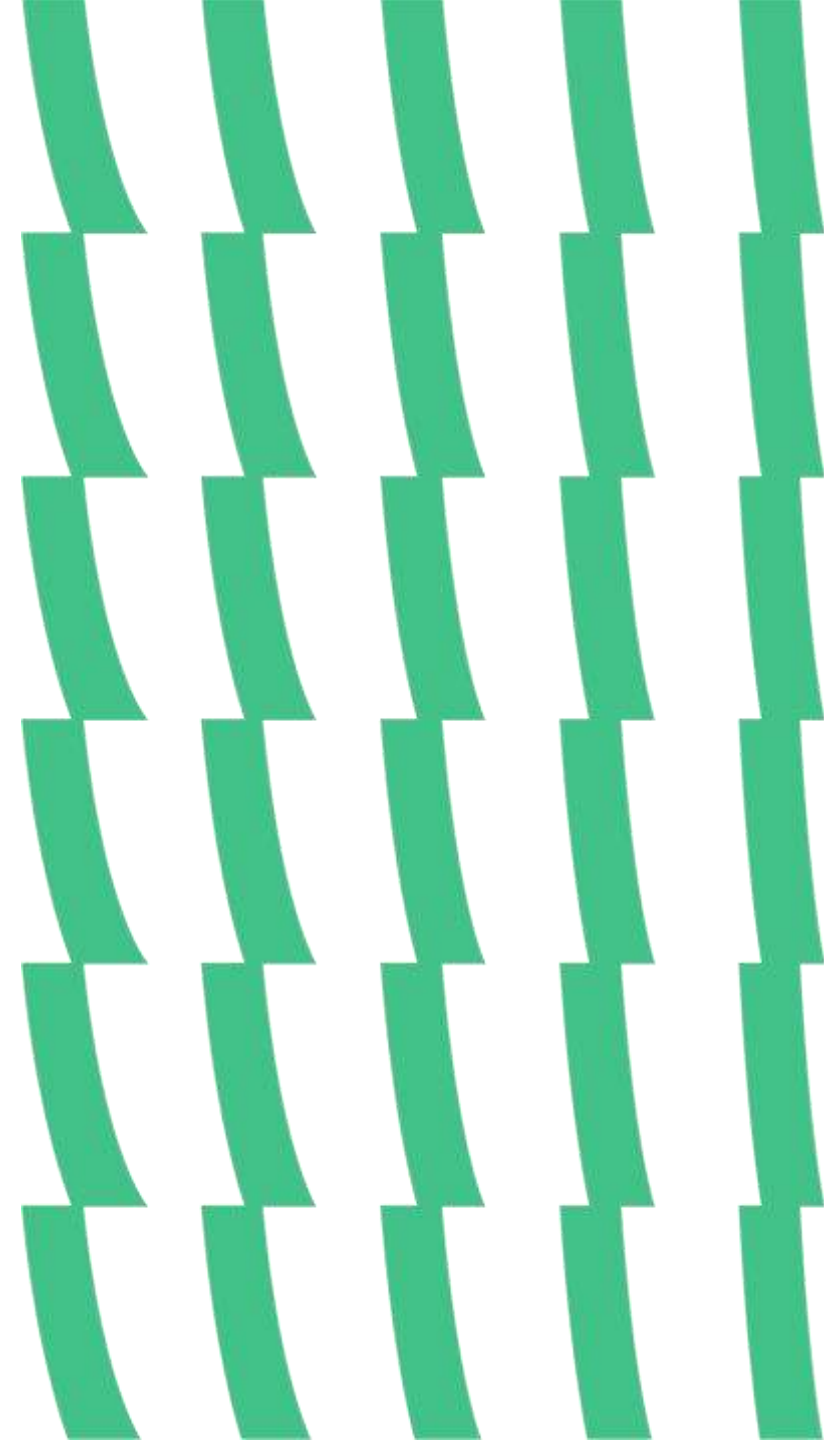
# Категории фактов

1. события (event)
2. мгновенные снимки (snapshot)
3. совокупные мгновенные снимки (cumulative snapshot)

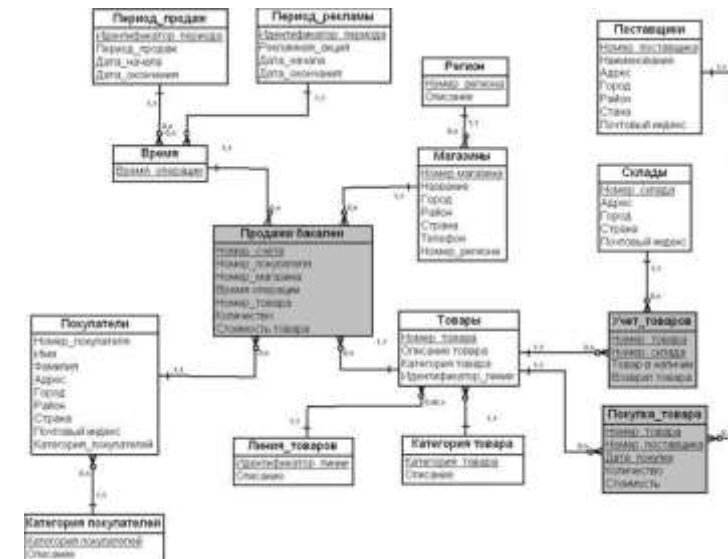
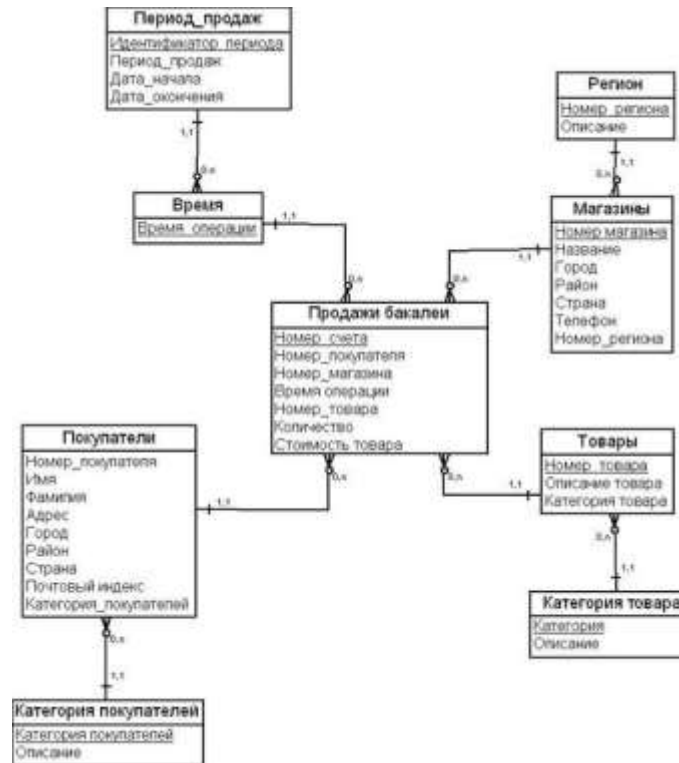


# Виды таблиц фактов

- Аддитивные факты (Additive facts).
- Полуаддитивные факты (Semiadditive facts).
- Неаддитивные факты (Non-additive facts).



# Модель «Звезда», «Снежинка», «Созвездие»



# Медленно меняющиеся измерения

Тип 0 — “Не изменяй”

Тип 1 — “Перезаписывай” — это обычная перезапись старых данных новыми. В чистом виде этот метод тоже не содержит версионности и используется лишь там, где история фактически не нужна

Тип 2 — “Добавляй строку” — для каждой версии создается отдельная запись в таблице с добавлением поля — ключевого атрибута данной версии.

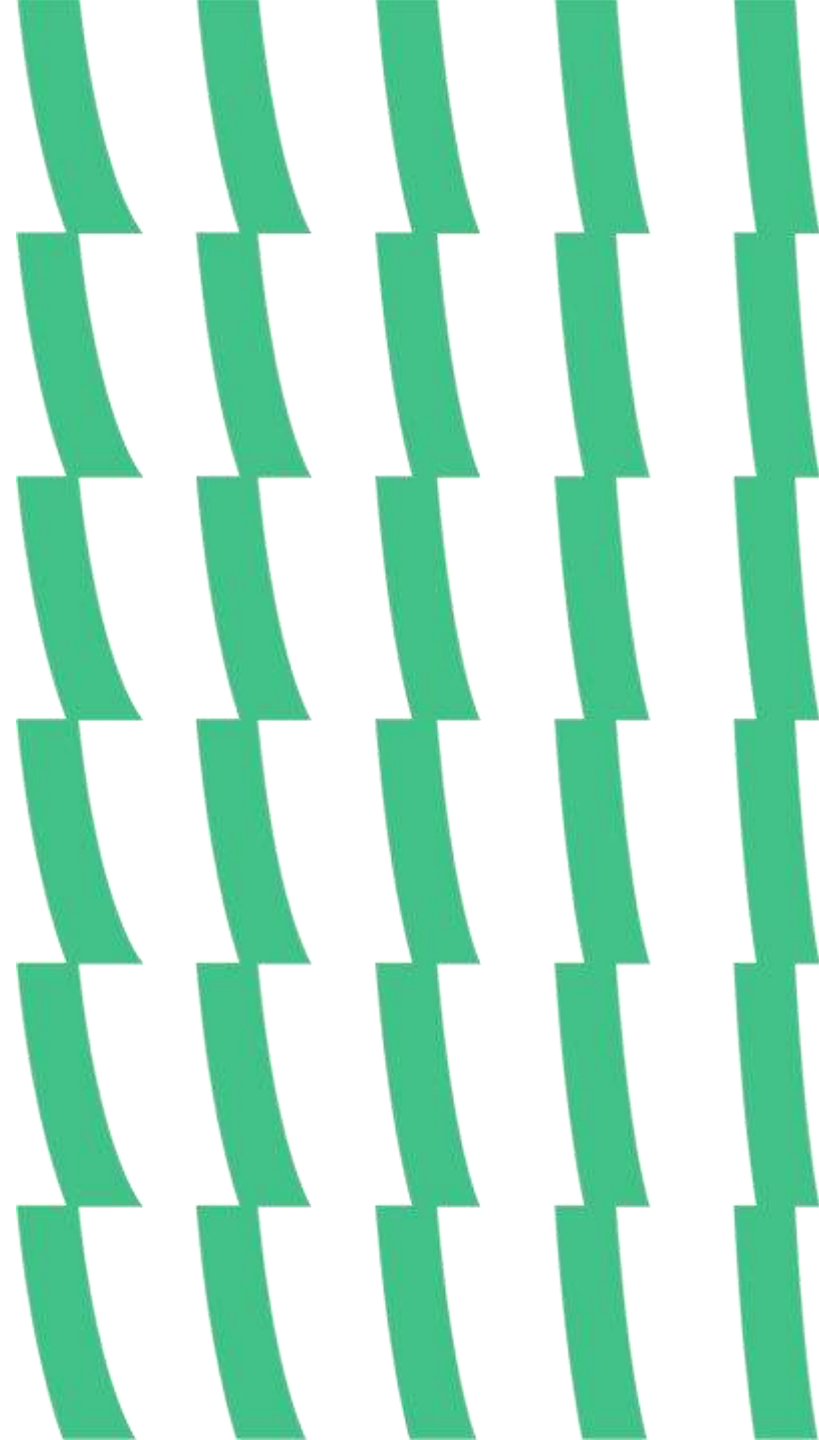
Тип 3 — “Добавляй атрибут” — в самой записи содержатся дополнительные поля для предыдущих значений атрибута. При получении новых данных, старые данные перезаписываются текущими значениями.

Тип 4 — “Добавляй таблицу” — комбинация SCD1 и SCD2.

1. Первая таблица содержит только текущее состояние данных (SCD1).

2. Вторая таблица содержит полную историю изменения (SCD2).

Тип 6 комбинация подходов SCD 1, 2 и 3 ( $1+2+3=6$ ).





# СВЯЗИ

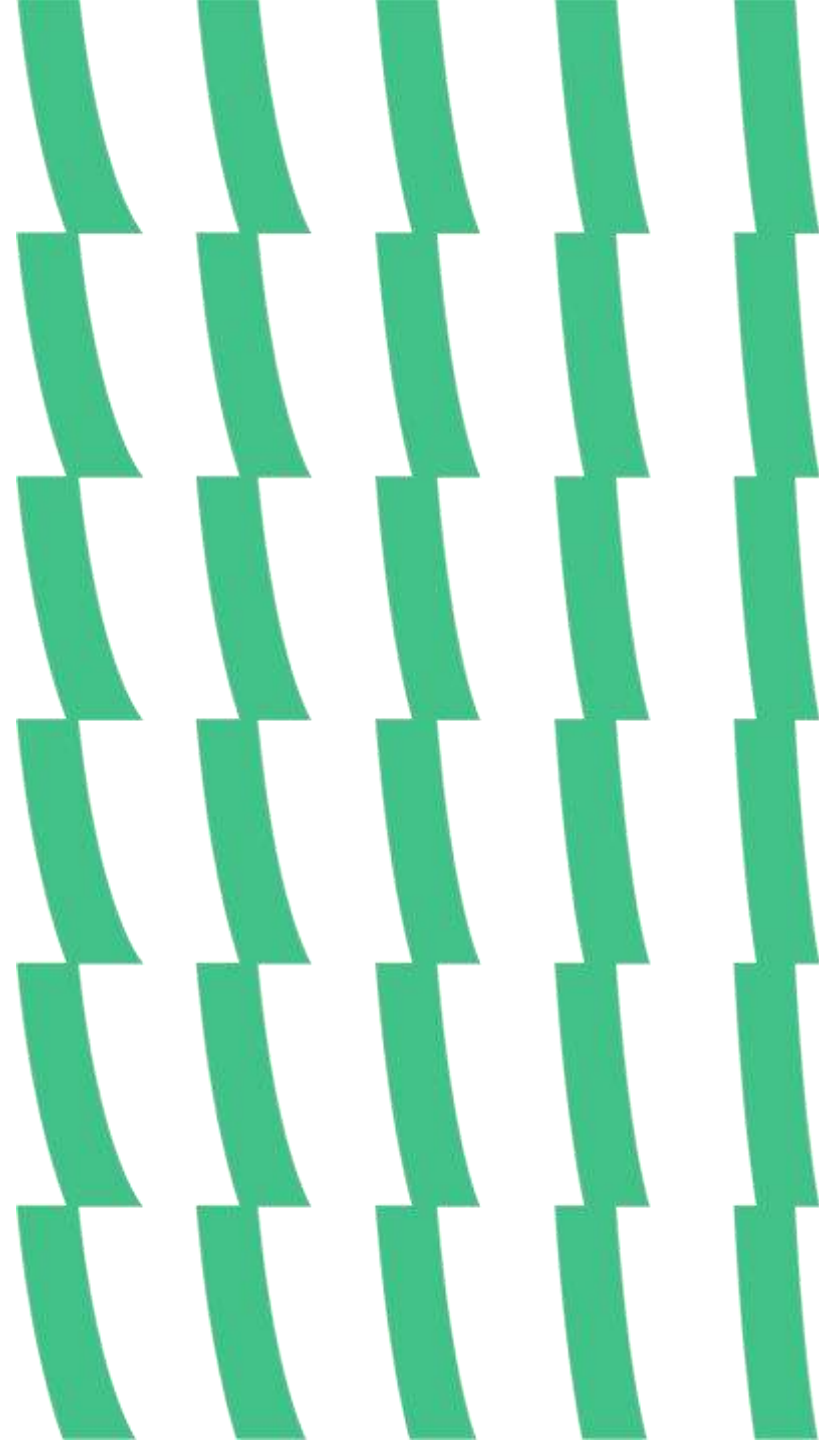
Для организации связи используются внешние ключи. Внешний ключ представляет один или несколько столбцов из одной таблицы, который одновременно является потенциальным ключом из другой таблицы. Внешний ключ необязательно должен соответствовать первичному ключу из главной таблицы. Хотя, как правило, внешний ключ из зависимой таблицы указывает на первичный ключ из главной таблицы.

Связи между таблицами бывают следующих типов:

Один к одному (One to one)

Один к многим (One to many)

Многие ко многим (Many to many)





# Data Vault



# Проектирование по Инмону

1. Стейджинговый слой - исторические данные из источника
2. Детальный слой - модель данных
3. Слой витрин - то, что видит потребитель



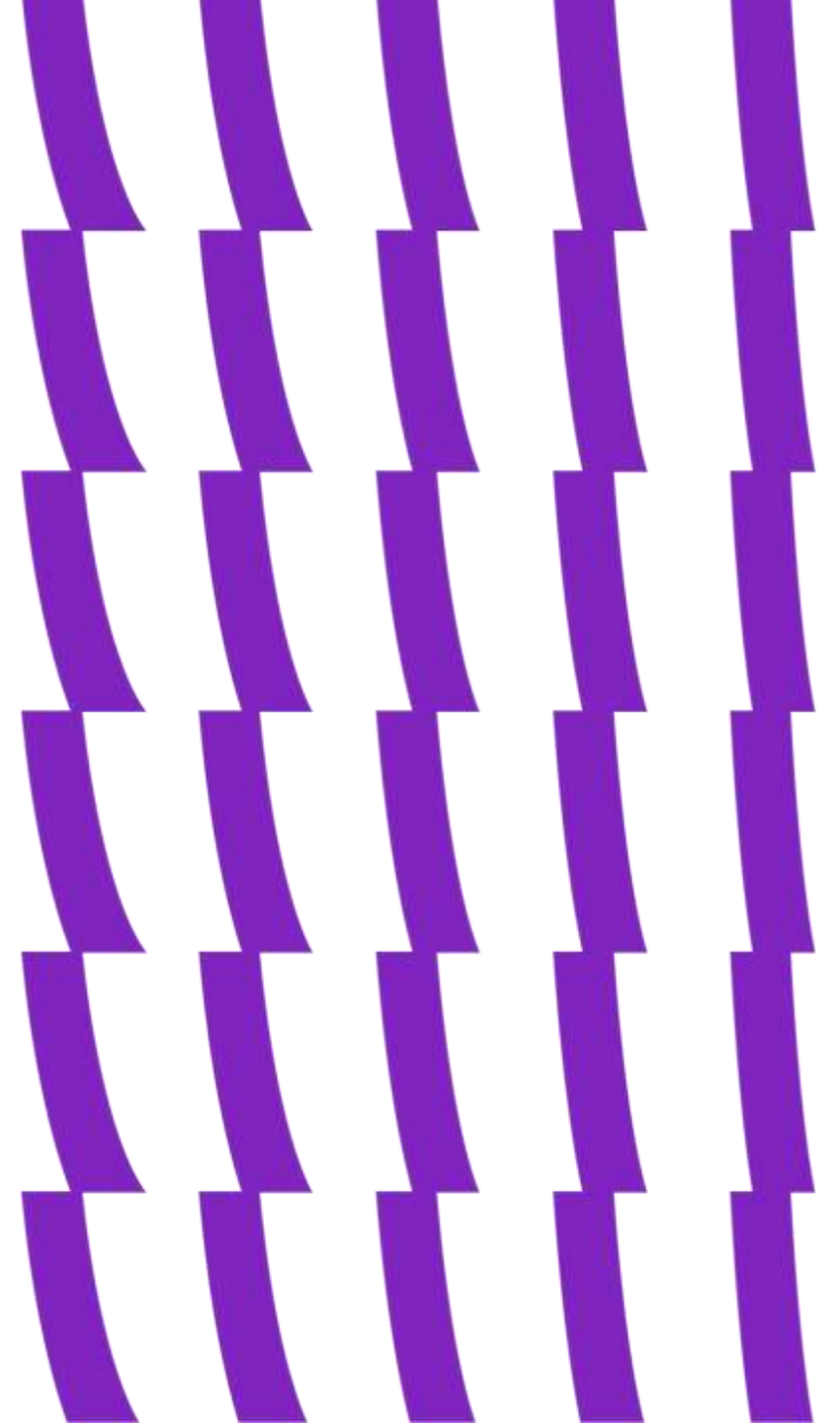
# Концепция Data Vault и Anchor modeling

Гибкая методология

Не все укладывается в звезды, не способны вместить всю сложность

Метод удачно сочетает требования нормализации и возможности схемы "звезда".

Различие лежит в более детальном представлении взаимосвязей и элементов данных, структурированных и детализованных во временном изменении.



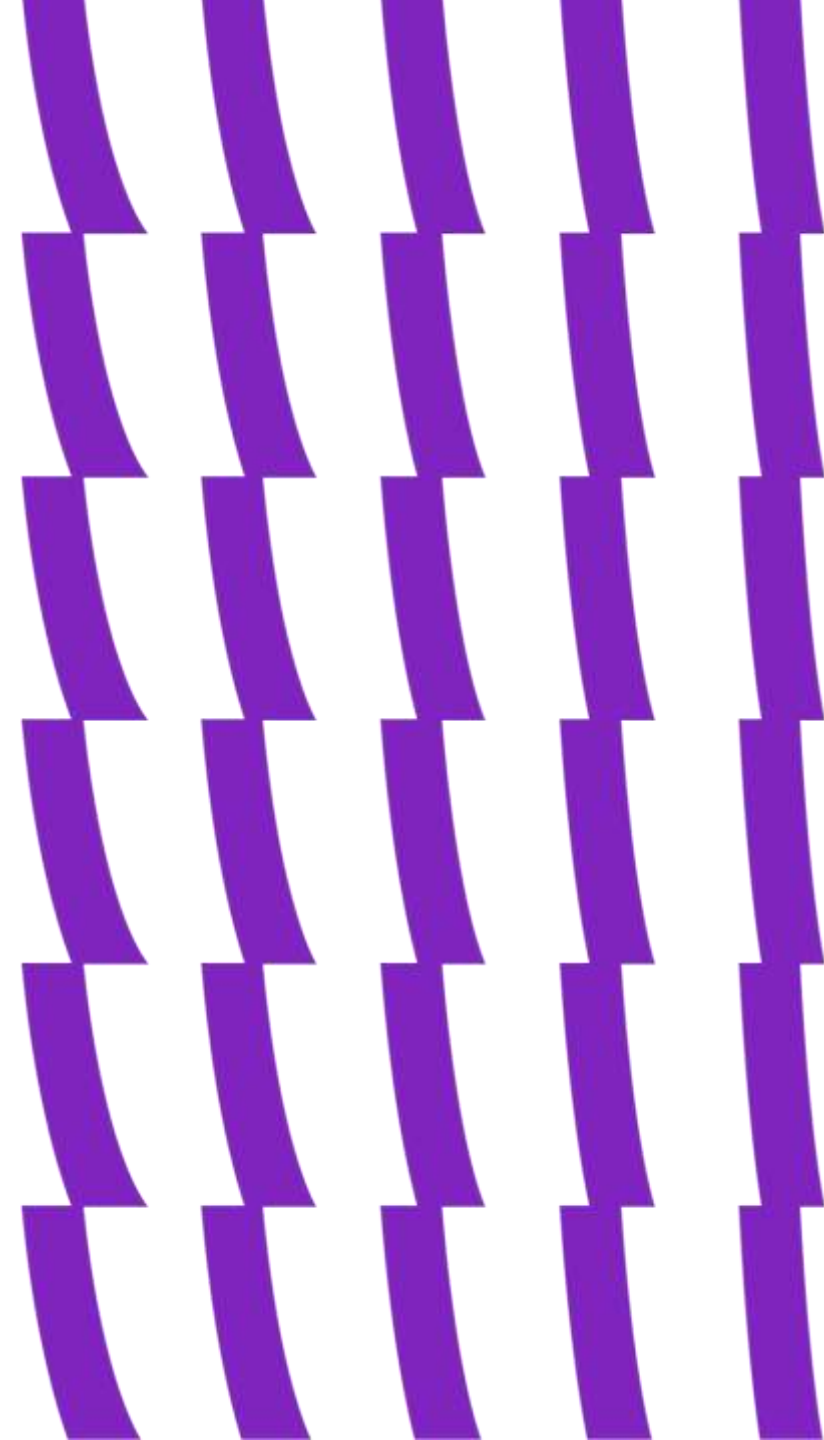
# Концепция Data Vault

**Data Vault** - набор уникально связанных нормализованных таблиц, содержащих детальные данные, отслеживающих историю изменений и предназначенных для поддержки одной или нескольких функциональных областей бизнеса.

Автор: Дэн Линстедт (Dan E. Linstedt)

Дизайн Data Vault - гибок, масштабируем, последователен и приспособляем к потребностям предприятия. Это модель данных, спроектированная специально для удовлетворения потребностей хранилищ данных предприятия.

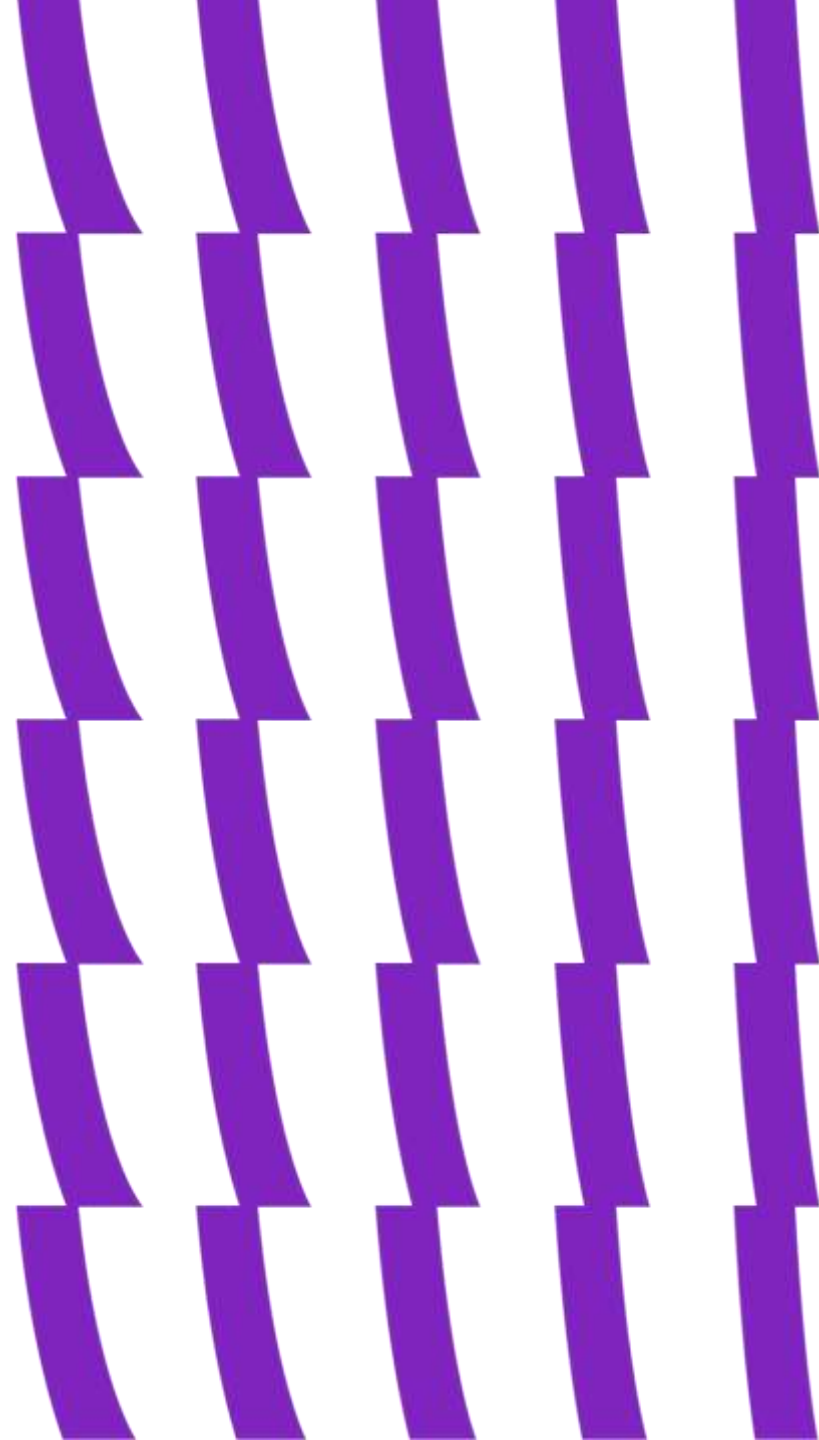
В модели Data Vault содержатся структуры, в чём-то похожие на традиционные модели звезды и 3NF, включая: измерения, связи многие ко многим и стандартные табличные структуры. Различия заключаются в представлении отношений, структурировании полей и хранении детальных данных с учётом историчности.



# Концепция Data Vault

**a single version of the truth** — классический подход. Если данные не вписываются в логику бизнес-процесса, они корректируются или удаляются.

**all the data, all of the time** — подход Data Vault. Нет различий между “хорошими” и “плохими” данными, поэтому структура не так зависима от изменений в бизнес-процессах. Хорошо подходит для итеративной разработки в условиях часто меняющейся модели данных.



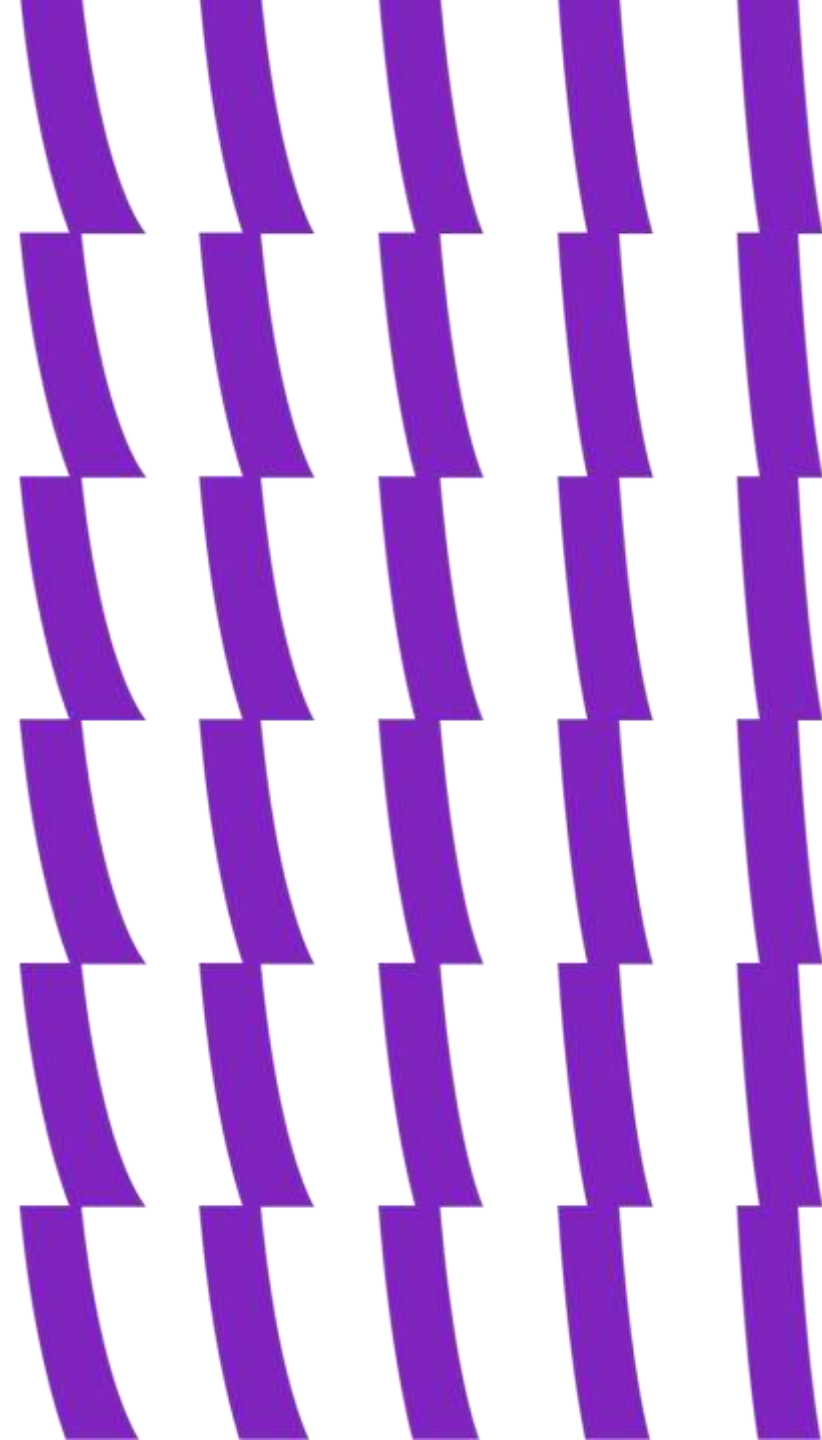
# Сущности Data Vault

Чтобы сохранить дизайн простым и изящным, используется минимальное количество типов сущностей: Хаб (Hub), Связь (Link) и Сателлит (Satellite).

Дизайн Data Vault сосредоточен вокруг функциональных областей бизнеса.

- Хаб (Hub) хранит сущности, соответствующие реальным объектам бизнес-процесса.
- Связь (Link) обеспечивает транзакционную интеграцию между Хабами (связи между сущностями).
- Сателлит (Satellite) предоставляет контекст первичного ключа Хаба (атрибуты, описания).

Каждая сущность предназначена для обеспечения максимальной гибкости и масштабируемости, сохраняя при этом большинство традиционных навыков моделирования данных.





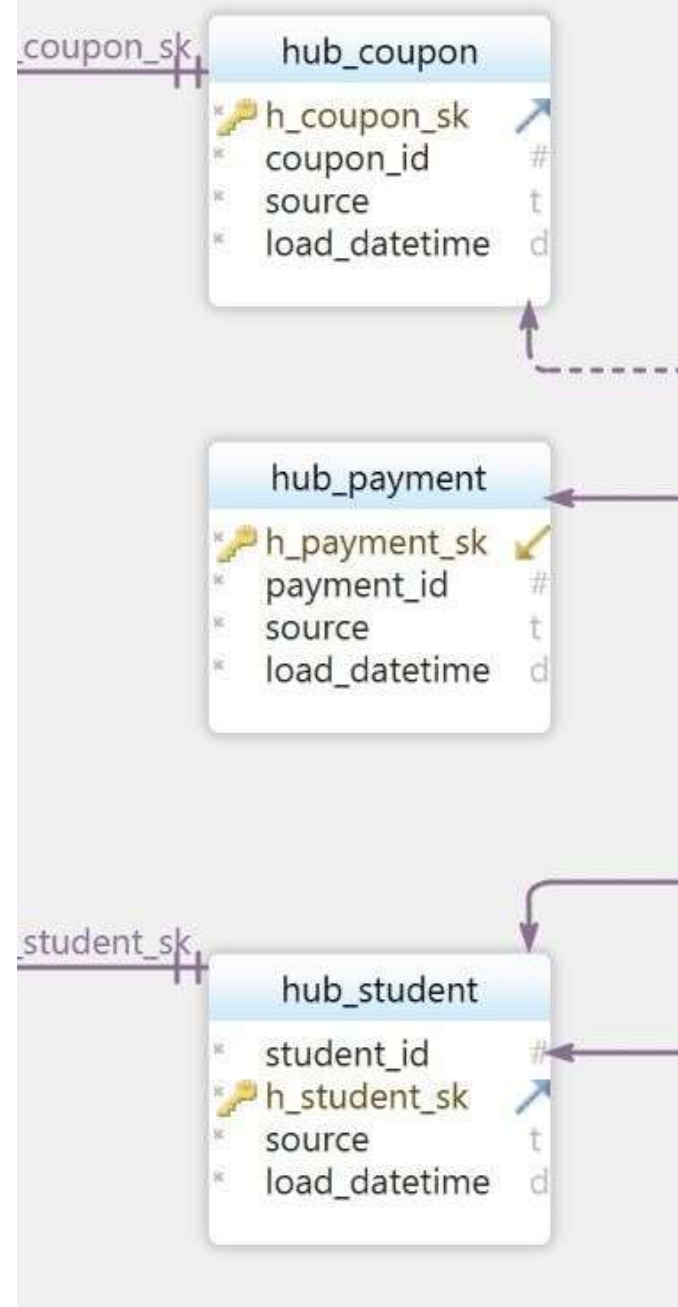
# Сущности Data Vault — Hub

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

Атрибуты Хаба включают:

- Ключ бизнес-сущности из внешней системы
- Суррогатный ключ (Surrogate Key)
- Временная метка даты загрузки (Load Timestamp), регистрирующая, когда ключ впервые был загружен в хранилище.
- Источник данных (Record Source) - регистрация исходной системы, используется для обратного аудита (отслеживания) данных.

Записи в Хабах никогда не изменяются и не имеют версий.





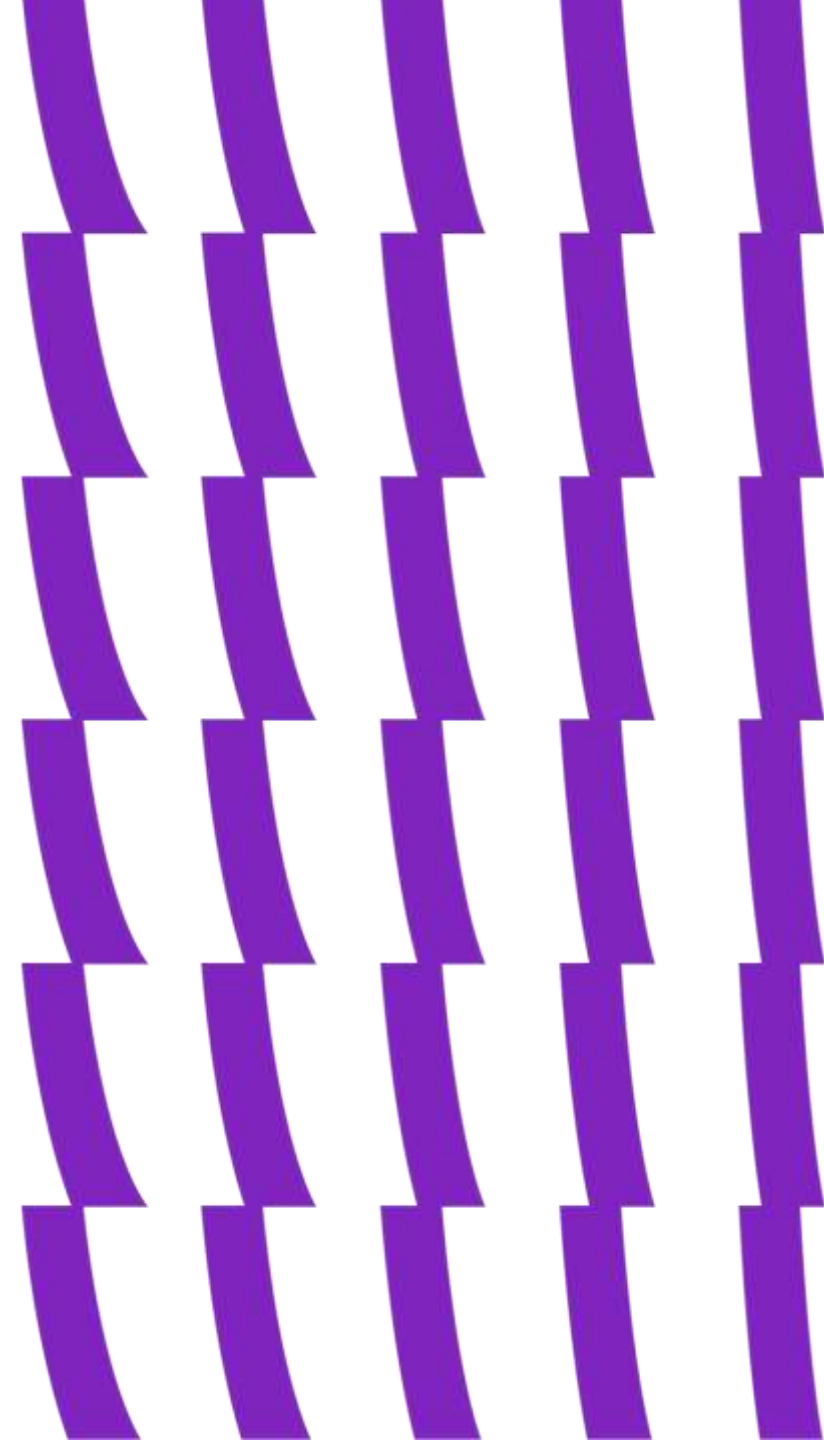
# Сущности Data Vault — Link

**Связи (Link)** - физическое представление связей “многие ко многим” третьей нормальной формы (3NF).

Связь представляет отношения или транзакцию между двумя или более компонентами бизнеса (два или более бизнес-ключа).

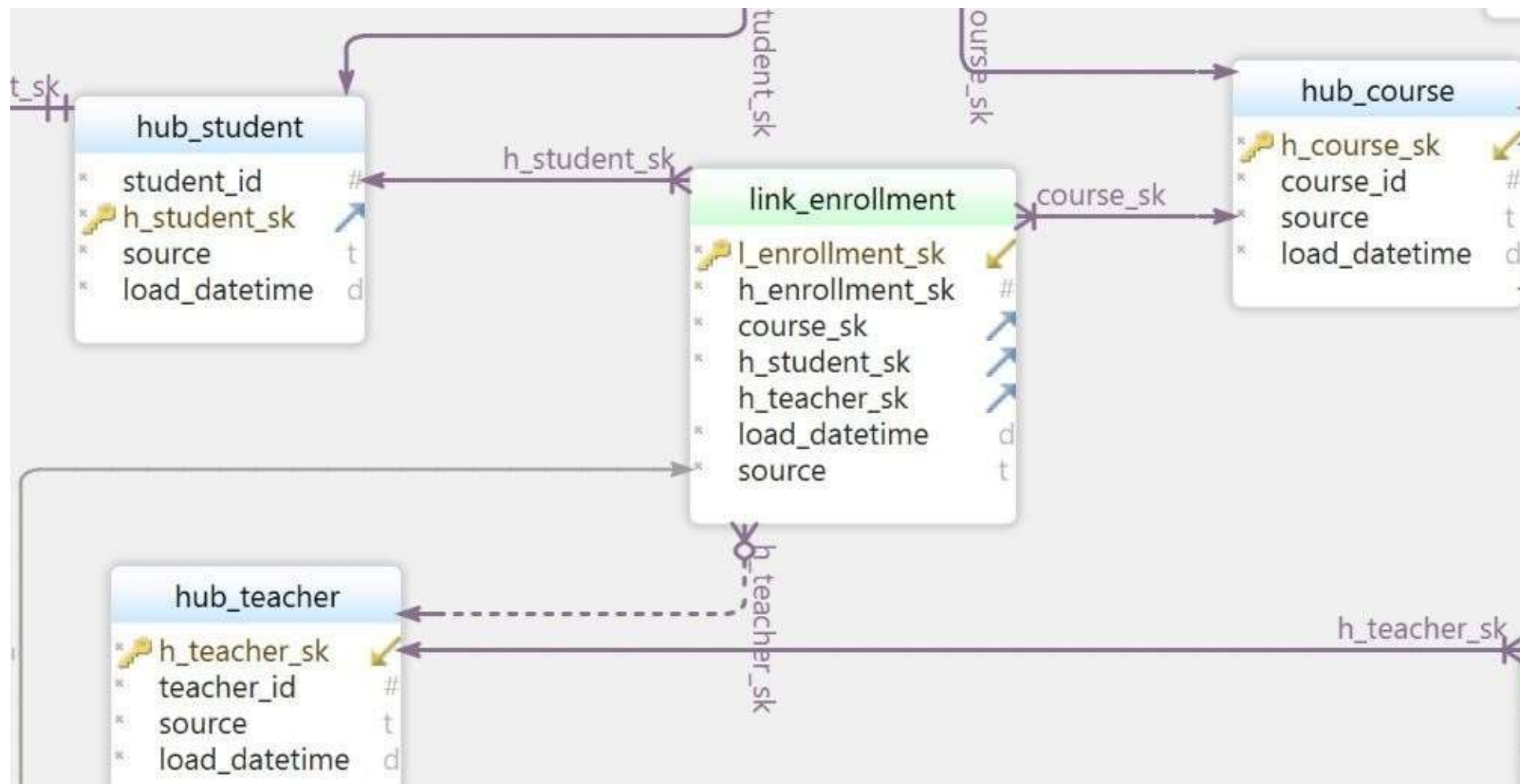
Связь содержит следующие атрибуты:

- Суррогатный ключ (Surrogate Key)
- Ключи Хабов - ключи из всех связанных Хабов мигрируют в сущность Связь, образуя составной ключ, и представляют взаимодействия и связи между Хабами.
- Временная отметка даты загрузки (Load Date Time Stamp), регистрирующая, когда связь/транзакция впервые была создана в хранилище.
- Источник данных (Record Source) - регистрация исходной системы, используется для обратной трассировки (отслеживания) данных.



# Сущности Data Vault – Link.

## Пример.

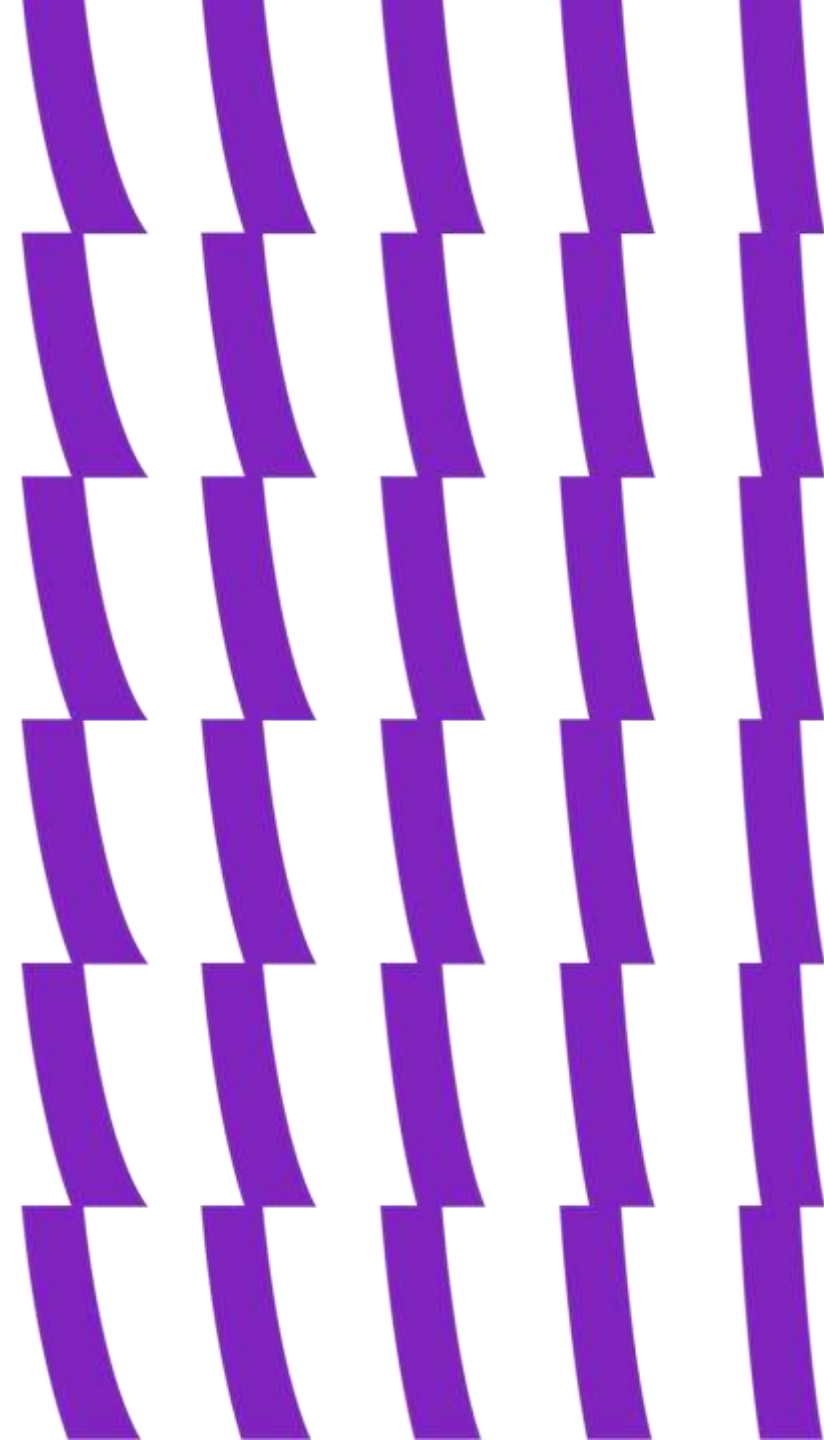


# Сущности Data Vault — Satellite

**Сателлиты (Satellite)** являются контекстной (описательной) информацией ключа Хаба. Описание подвергается изменениям с течением времени, и поэтому структура Сателлитов должна быть способна хранить новые или измененные детальные данные. сателлиты обеспечивают весь контекст и изменения с течением времени.

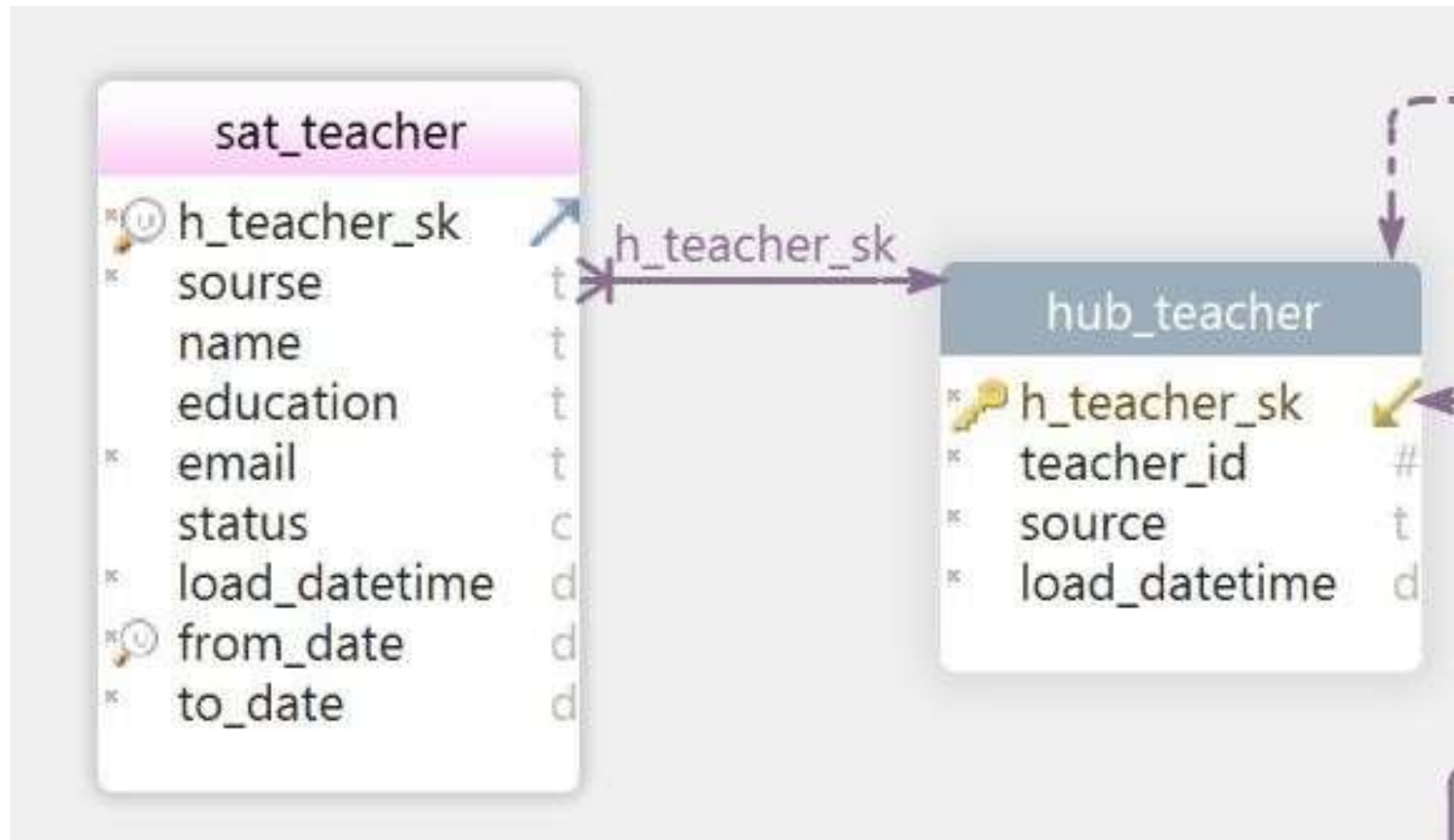
Сателлит состоит из следующих обязательных атрибутов:

- Первичный ключ Сателлита — первичный ключ Хаба или первичный ключ Связи — мигрирует в Сателлит из Хаба или Связи.
- Временная отметка даты загрузки (Load Date Time Stamp) — регистрация, когда информация стала доступна в хранилище (всегда вставляется новая строка).
- Источник данных (Record Source) - регистрация исходной системы, используется для обратного аудита (отслеживания) данных.



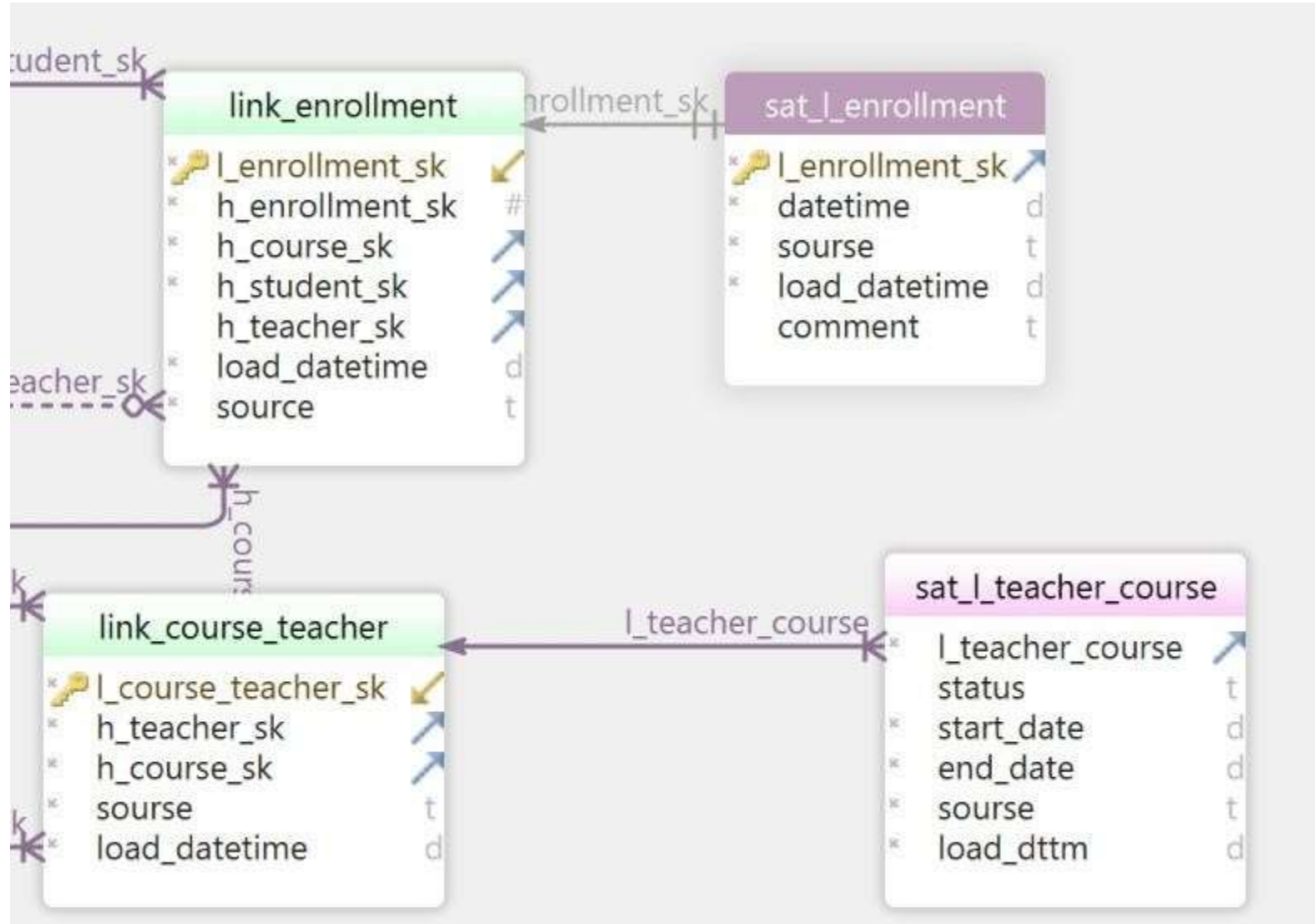
# Сущности Data Vault – Satellite. Пример.

Сателлит наиболее близок к медленно меняющимся измерениям второго типа в определении Ральфа Кимбалла. Он хранит изменения на детальном уровне, а его функция заключается в обеспечении описательного контекста экземплярам Хаба или Связи.



# Сущности Data Vault – Satellite.

## Пример линк-сателлита

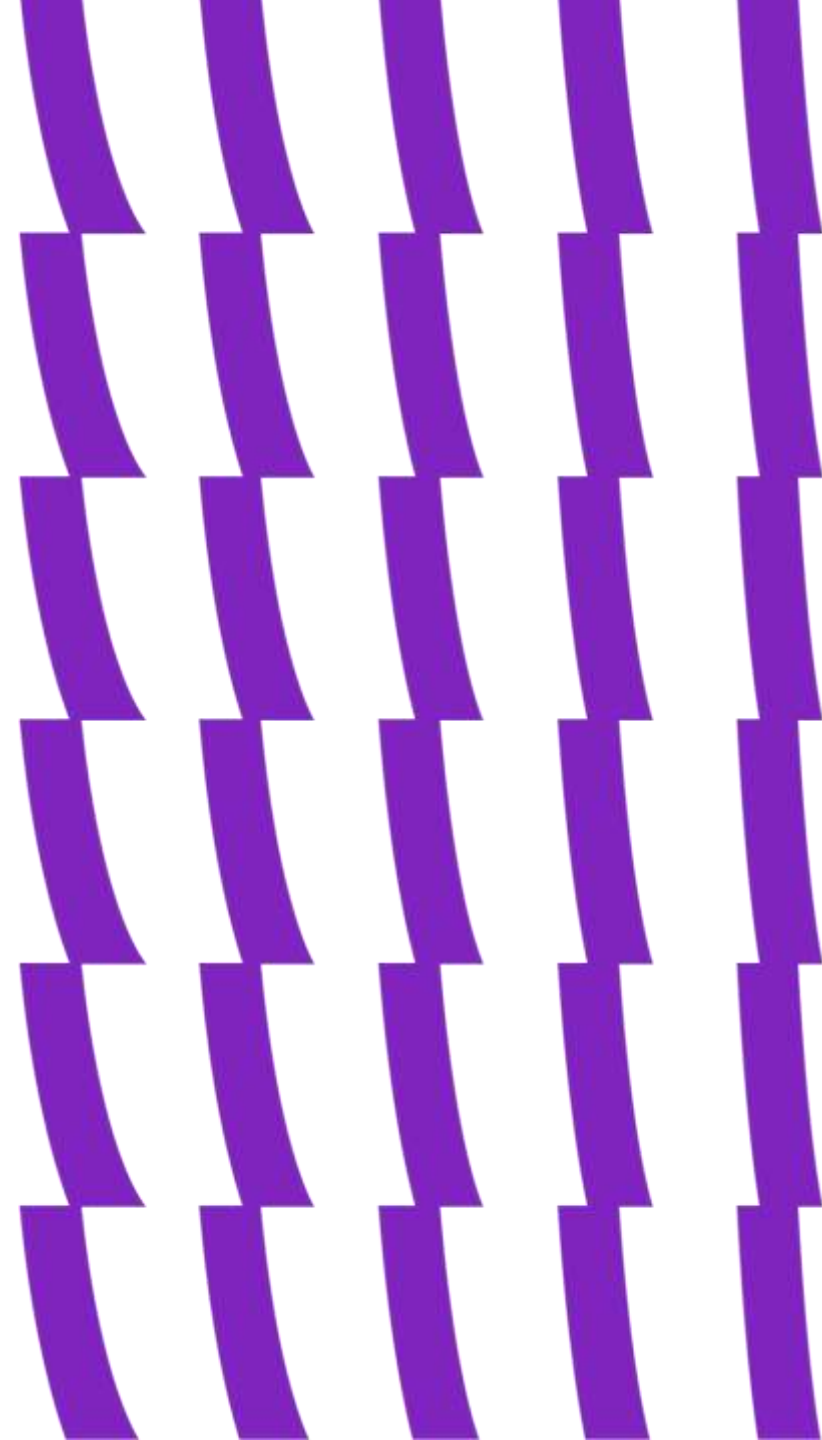




# Проектирование Data Vault

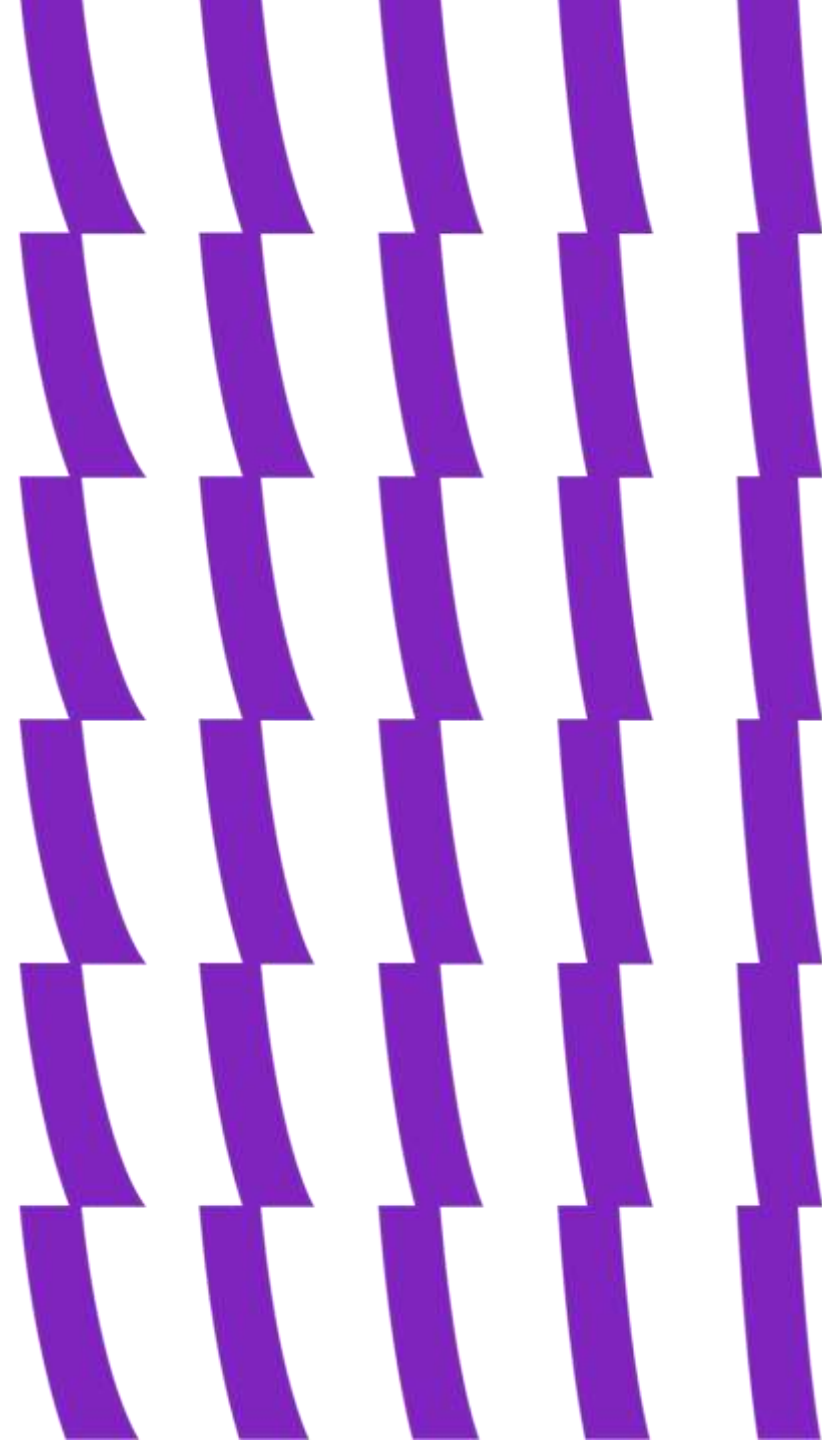
Алгоритм построения модели Data Vault:

1. Смоделируйте Хабы. Для этого требуется понимать основные бизнессущности (и их бизнес-ключи) и как они используются в выбранной области.
2. Смоделируйте Связи. Выявление возможных отношений между ключами - требует формулировки, как бизнес работает в контексте каждого бизнес-ключа.
3. Смоделируйте Сателлиты. Обеспечение контекста как каждой бизнес сущности (бизнес-ключу), так и транзакциям/сделкам (Связи), соединяющим Хабы вместе. С этого начинает проявляться полная картина о бизнесе.



# Правила Data Vault (Резюме)

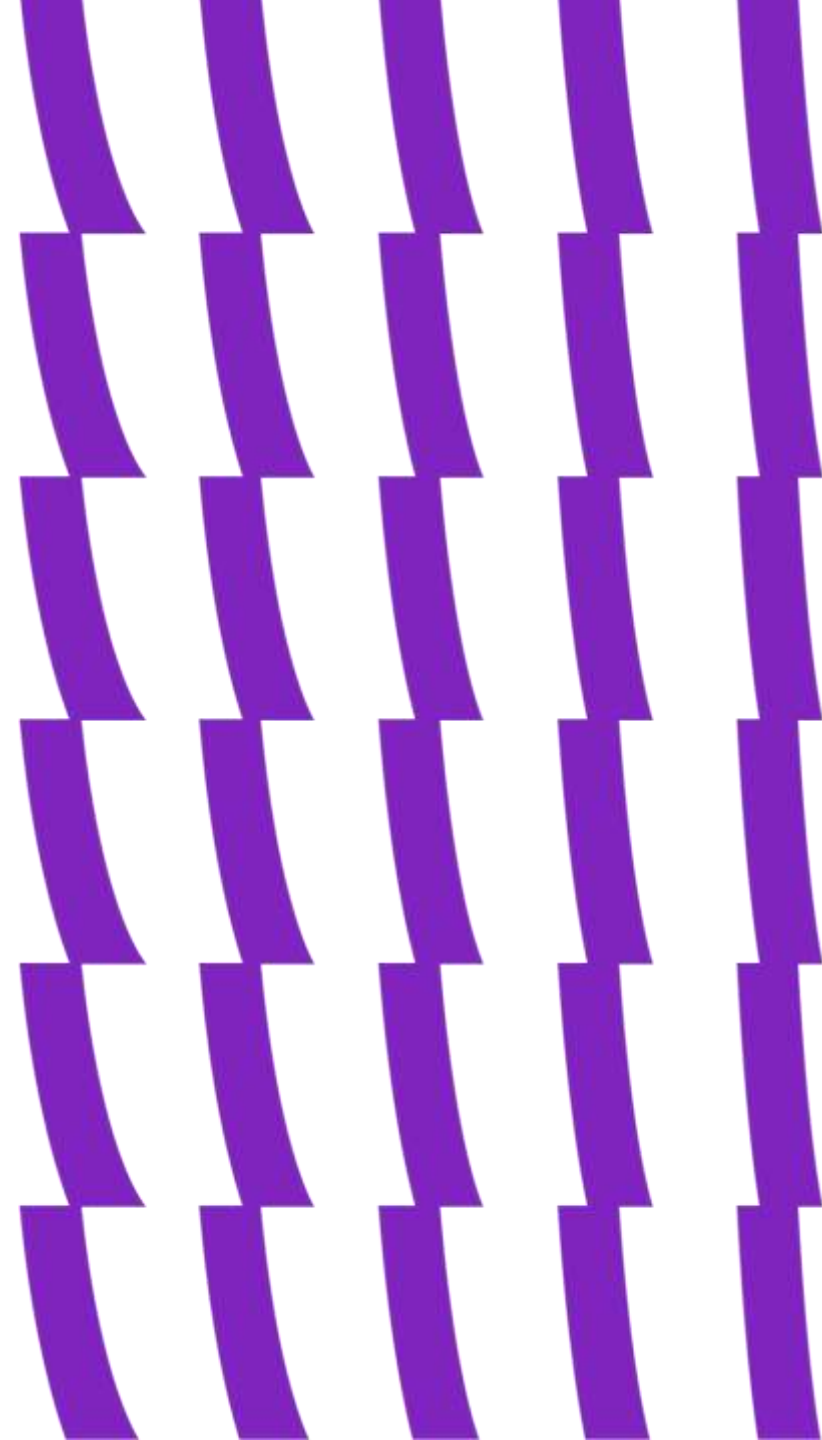
1. Ключи Хабов не могут мигрировать в другие Хабы (не может быть иерархических Хабов). Моделирование в такой манере нарушает гибкость и расширяемость техники моделирования Data Vault.
2. Хабы должны быть связаны с помощью сущностей типа Связь.
3. С помощью Связей может быть связано более двух Хабов.
4. Сущности Связи могут быть связаны с другими сущностями типа Связи.
5. Суррогатные ключи могут быть сгенерированы для Хабов и для Связей.
6. У Сателлита нет своего собственного суррогатного ключа.
7. Ключи Хаба всегда мигрируют наружу





# Правила Data Vault (Резюме)

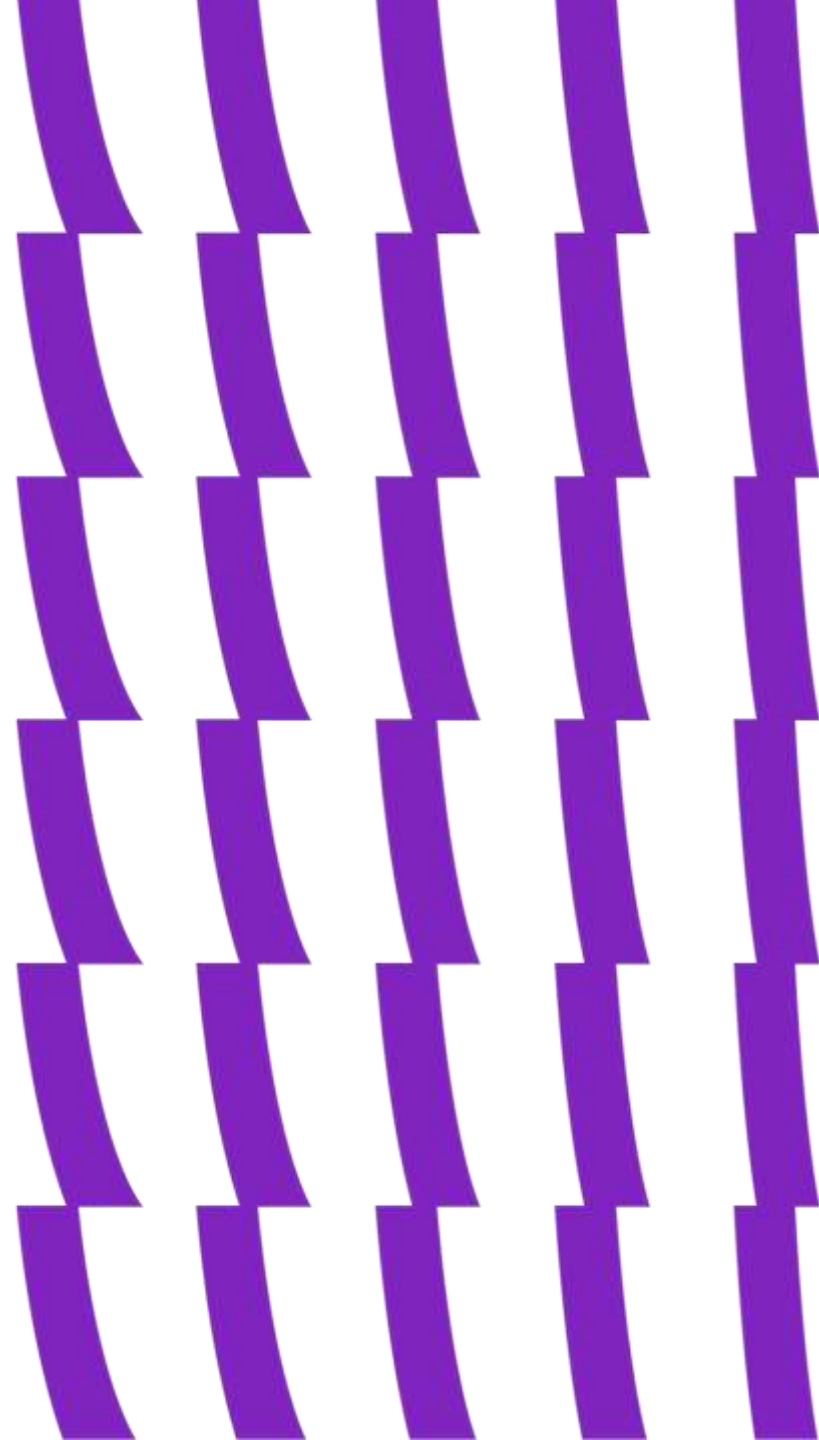
- 8. Бизнес-ключи Хаба никогда не меняются, первичные ключи Хабов никогда не меняются.
- 9. Сателлиты могут быть связаны как с Хабами, так и со Связями.
- 10. Сателлиты всегда содержат временную отметку даты загрузки (Load Date Time Stamp).
- 11. Могут быть использованы автономные таблицы, такие как календари, время, коды и описания.
- 12. Сателлиты фиксируют только изменения, дублирование строк не должно быть.
- 13. Данные распределяются по структуре Сателлитов, основываясь на: 1) типе информации, 2) темпах изменения.

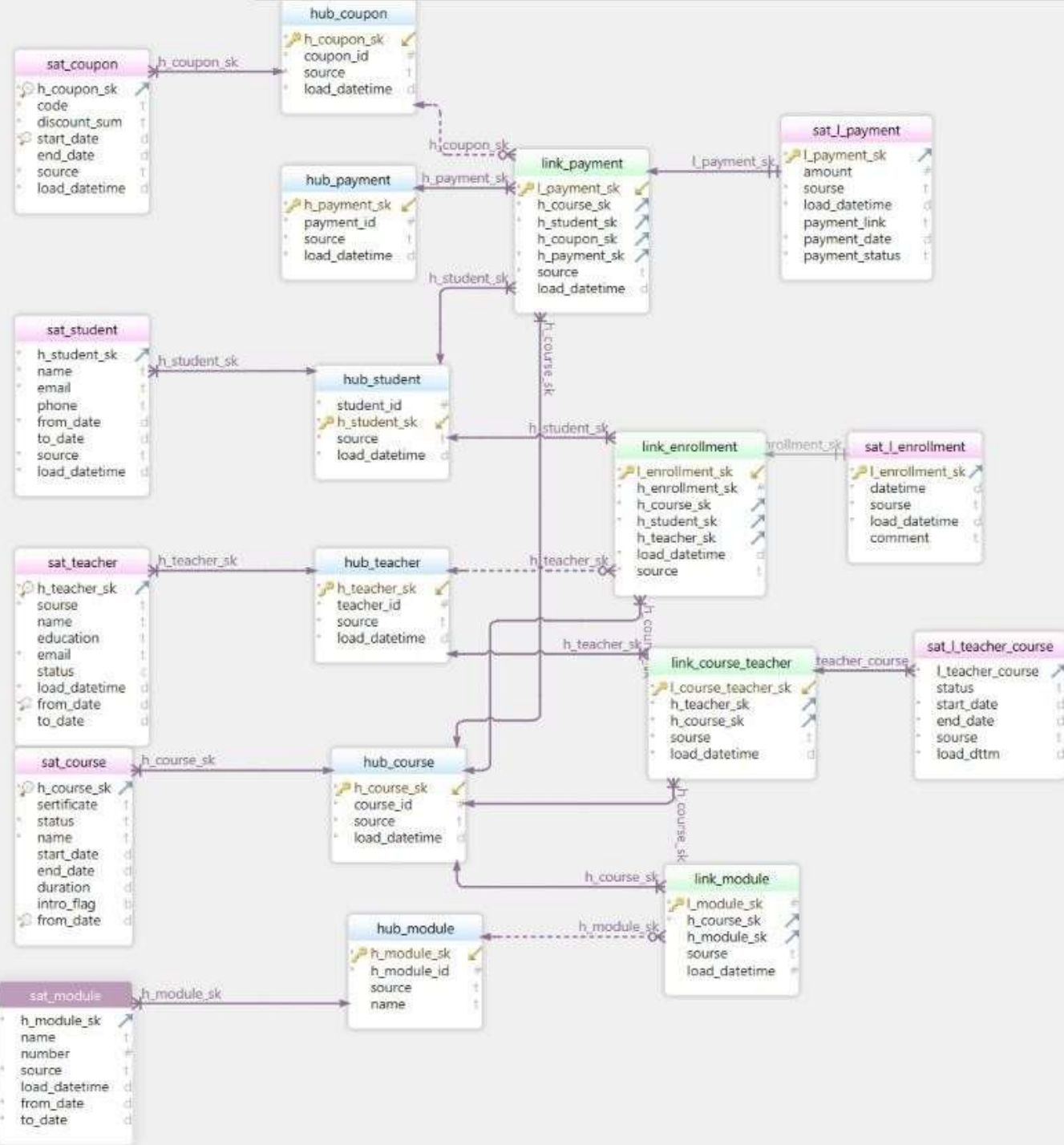


# Загрузка Data Vault

1. Загрузка хабов
2. Загрузка Линков и Сателлитов (параллельно и последовательно).

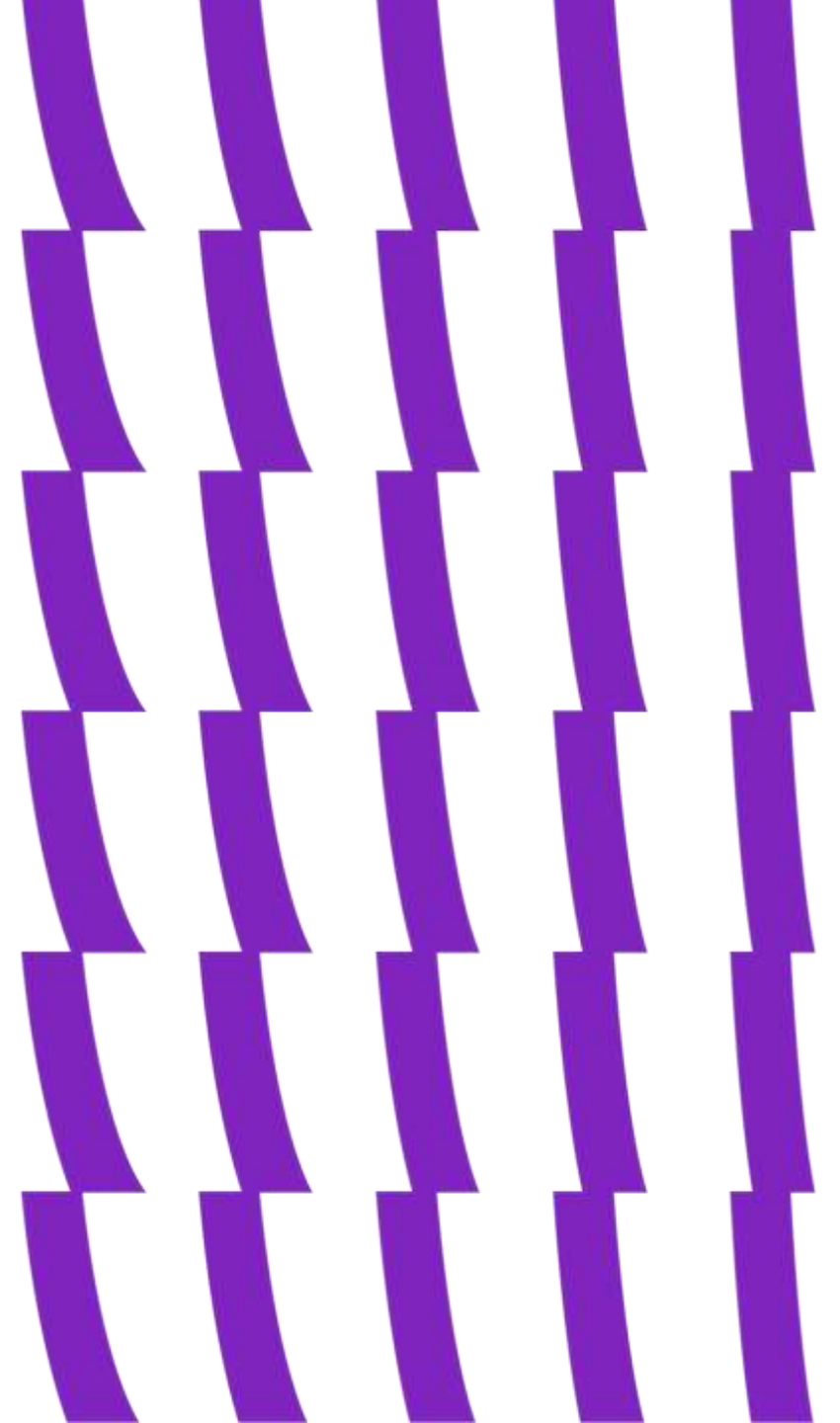
Хабы можно загружать параллельно, так же как и Сателлиты со Ссылками, если конечно не используется связь link-to-link.





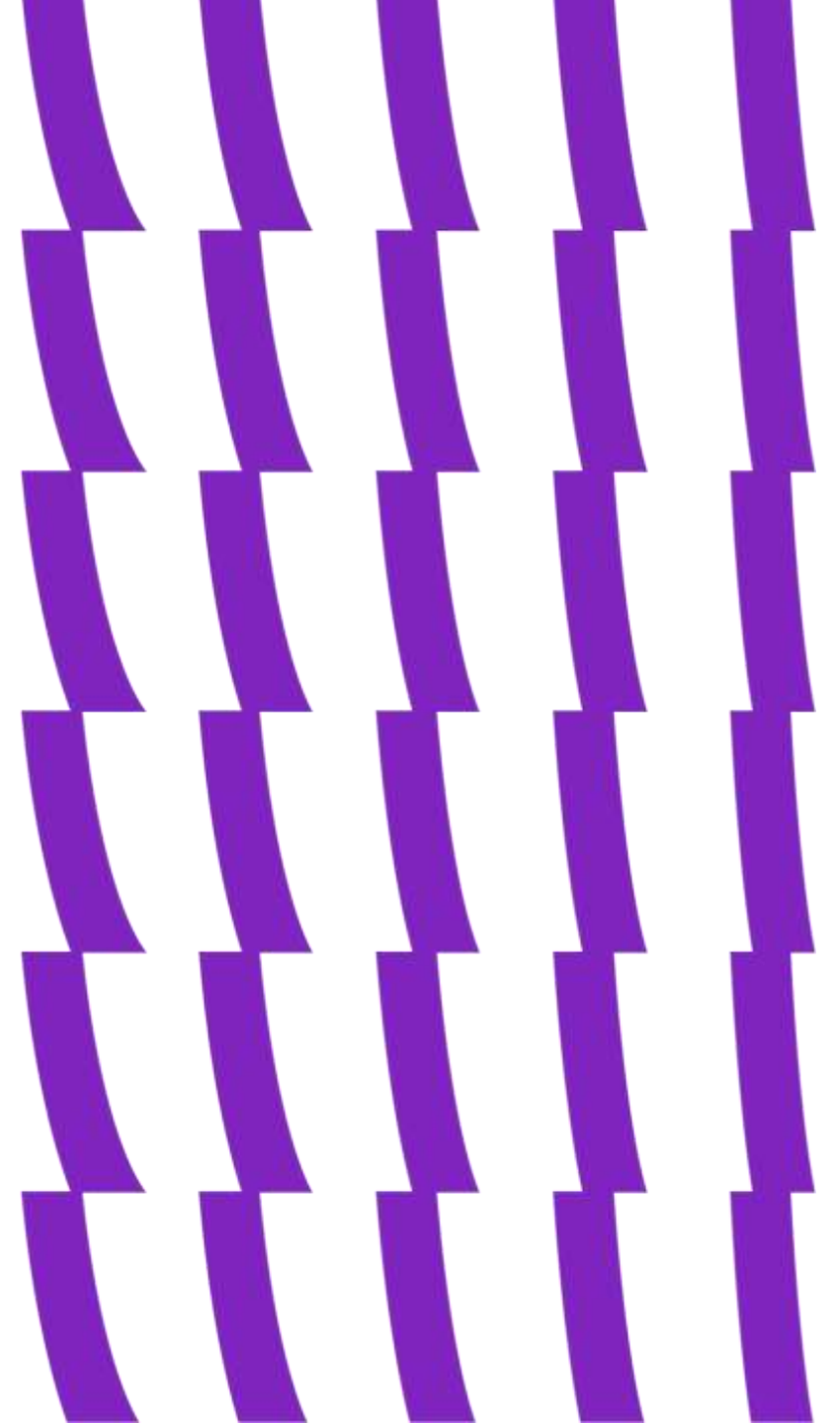
# Преимущества Data Vault

- Гибкость и расширяемость. С Data Vault перестает быть проблемой как расширение структуры хранилища, так и добавление и сопоставление данных из новых источников. Максимально полное хранилище «сырых» данных и удобная структура их хранения позволяют нам сформировать витрину под любые требования бизнеса, а существующие решения на рынке СУБД хорошо справляются с огромными объемами информации и быстро выполняют даже очень сложные запросы, что дает возможность виртуализировать большинство витрин.



# Преимущества Data Vault

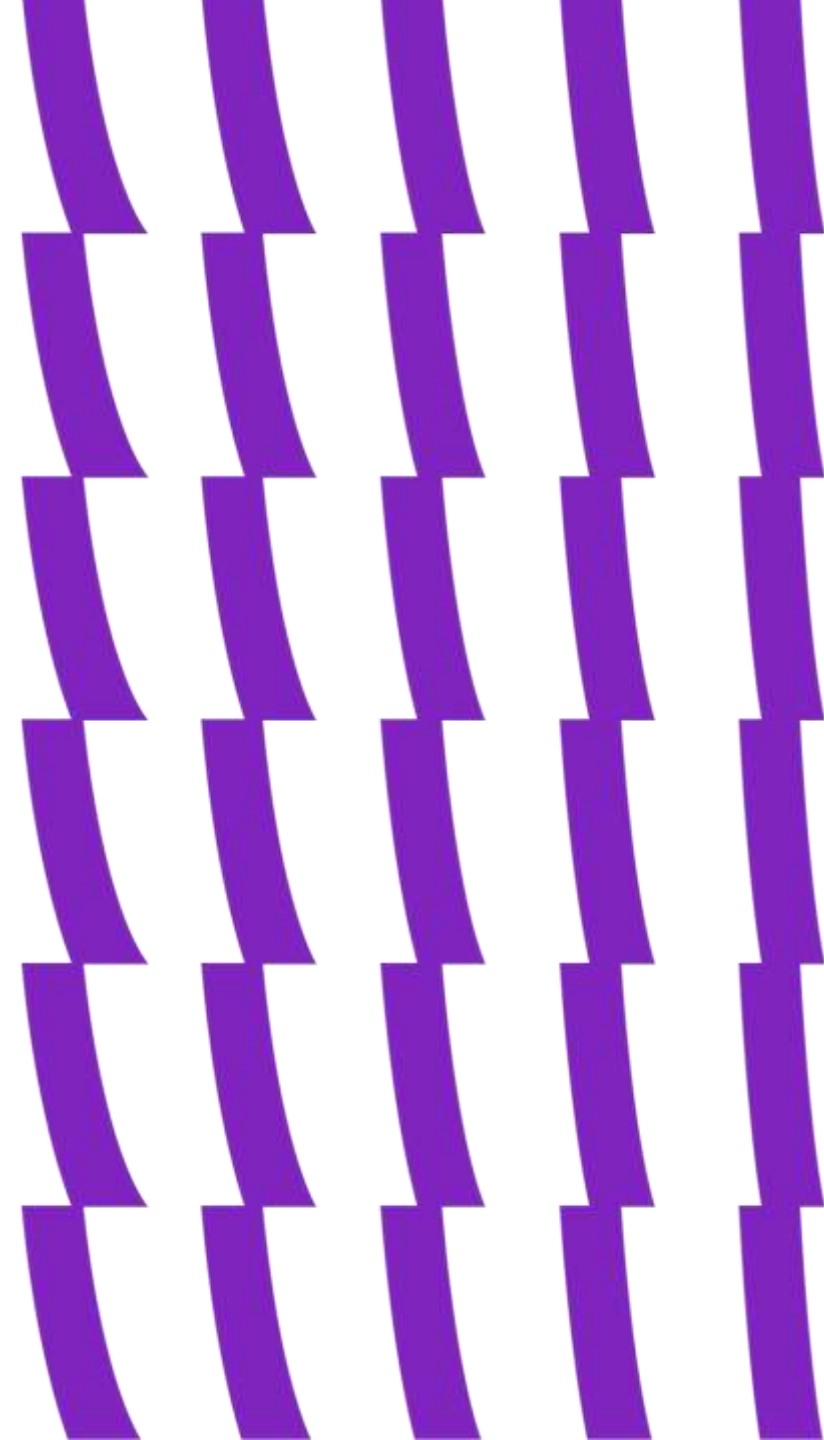
- Agile-подход из коробки. Моделировать хранилище по методологии Data Vault довольно просто. Новые данные просто «подключаются» к существующей модели, не ломая и не модифицируя существующую структуру. При этом поставленную задачу можно решать максимально изолированно, загружая только необходимый минимум, и, вероятно, временная оценка для такой задачи станет точнее. Планирование спринтов будет проще, а результаты предсказуемы с первой же итерации.
- Отсутствие избыточности данных, а для больших данных это весьма важный аргумент





# Недостатки Data Vault

- Обилие JOIN'ов. За счет большого количества операций join запросы могут быть медленнее, чем в традиционных хранилищах данных, где таблицы денормализованы.
- Сложность. В описанной выше методологии есть множество важных деталей, разобраться в которых вряд ли получится за пару часов. К этому можно прибавить малое количество информации в интернете и почти полное отсутствие материалов на русском языке. Как следствие, при внедрении Data Vault возникают проблемы с обучением команды, появляется много вопросов относительно нюансов конкретного бизнеса. Большой недостаток сложности это обязательное требование к наличию витрин данных, так как сам по себе Data Vault плохо подходит для прямых запросов.



# Anchor modelling





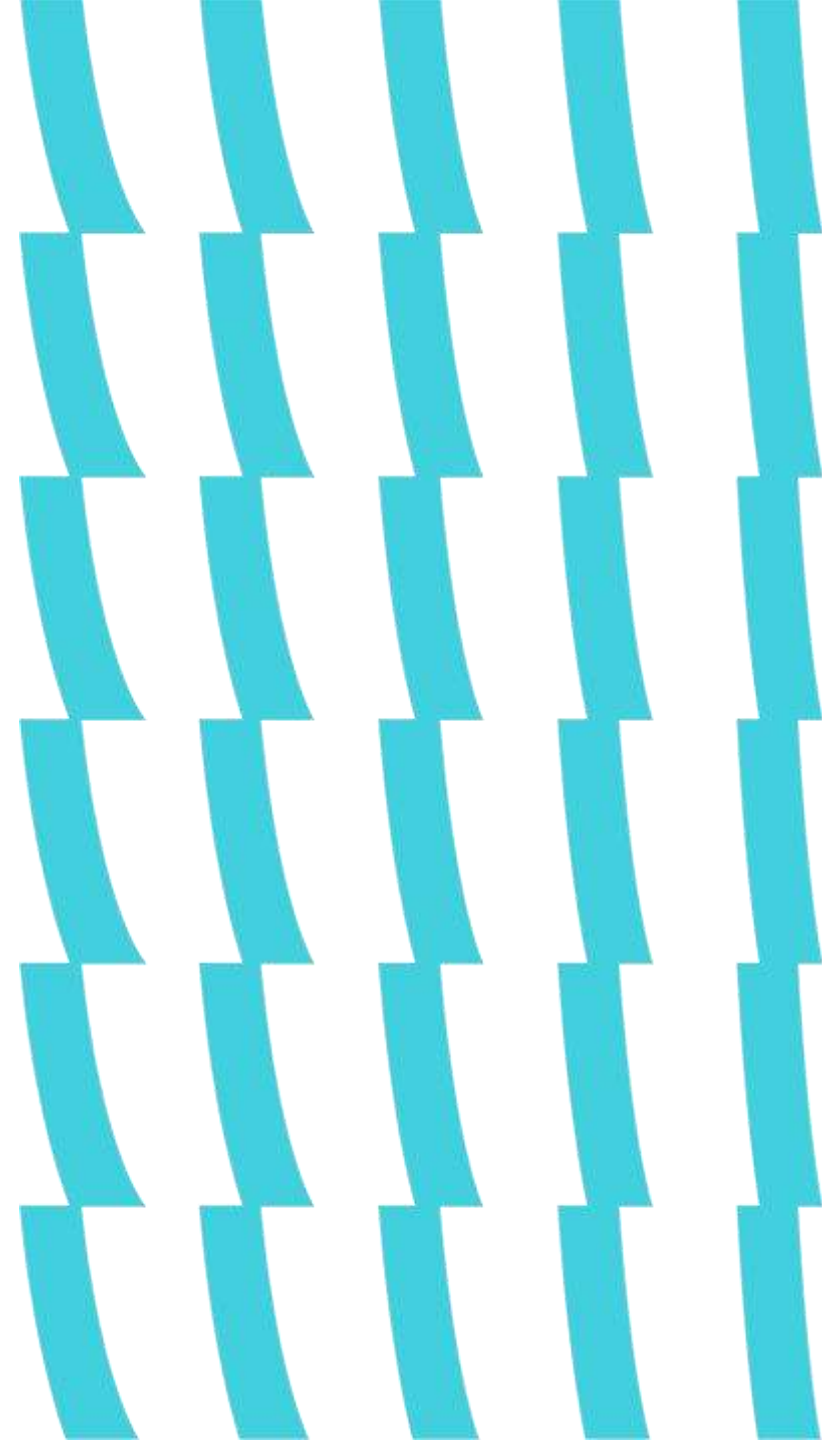
# Якорное моделирование

Автор: Ларс Рёнбэк (Lars Rönnbäck)

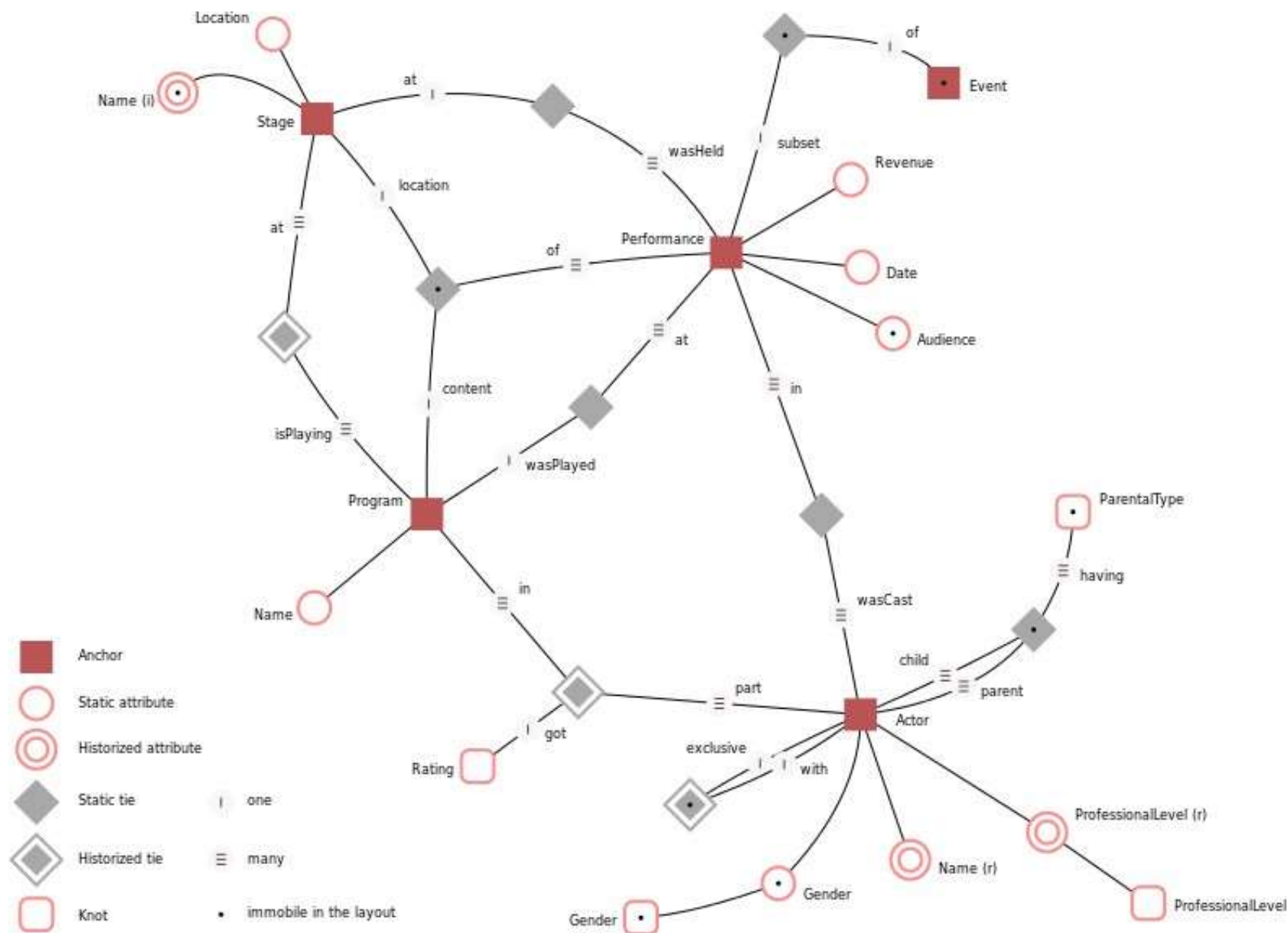
**Якорное моделирование** — это технология моделирования гибкой базы данных, подходящая для информации, которая со временем изменяется как по структуре, так и по содержанию.

В методике моделирования используются четыре основные сущности: **якорь**, **атрибут**, **связь** и **узел**, каждый из которых отражает различные аспекты моделируемого домена. Полученная модель может быть переведена 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

Якорная модель фокусируется на изменениях



# Якорное моделирование



# Якорное моделирование

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

Anchor таблица должна хранить ТОЛЬКО суррогатный ключ и несколько технических полей (система-источник, дата-время загрузки).

Пример — курс, студент, купон

Натуральные ключи с точки зрения Якорной Модели считаются обычными атрибутами

Концептуально Anchor нужен только для одной задачи — грузить каждый уникальный товар/пользователя/платеж только один раз.  
=> Избежать повторной загрузки, и помнить, в какой момент и из какой системы пришла исходная запись.

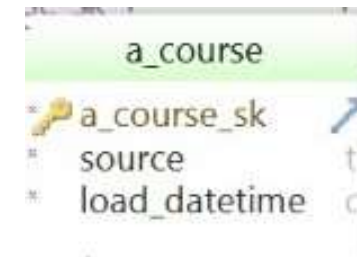


Diagram illustrating the structure of the `a_course` anchor table. It shows a surrogate key `a_course_sk` (indicated by a key icon) and two technical attributes: `source` (type `t`) and `load_datetime` (type `d`).

a_course	
a_course_sk	
source	t
load_datetime	d

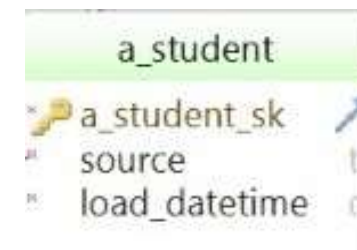


Diagram illustrating the structure of the `a_student` anchor table. It shows a surrogate key `a_student_sk` (indicated by a key icon) and two technical attributes: `source` (type `t`) and `load_datetime` (type `d`).

a_student	
a_student_sk	
source	t
load_datetime	d

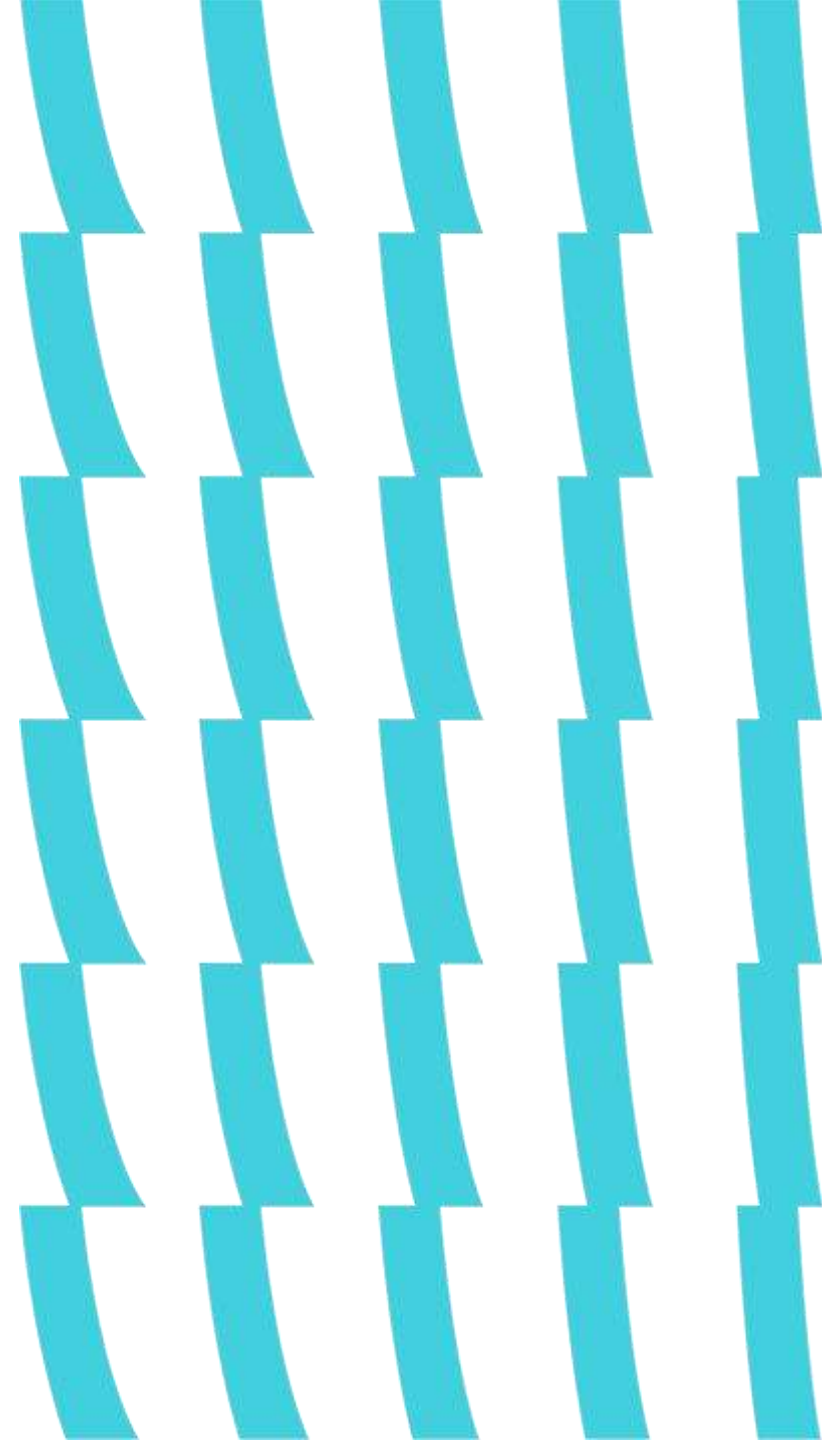
# Якорное моделирование — Attribute

**Attribute (Атрибут)** — это таблица для хранения свойства, атрибута объекта

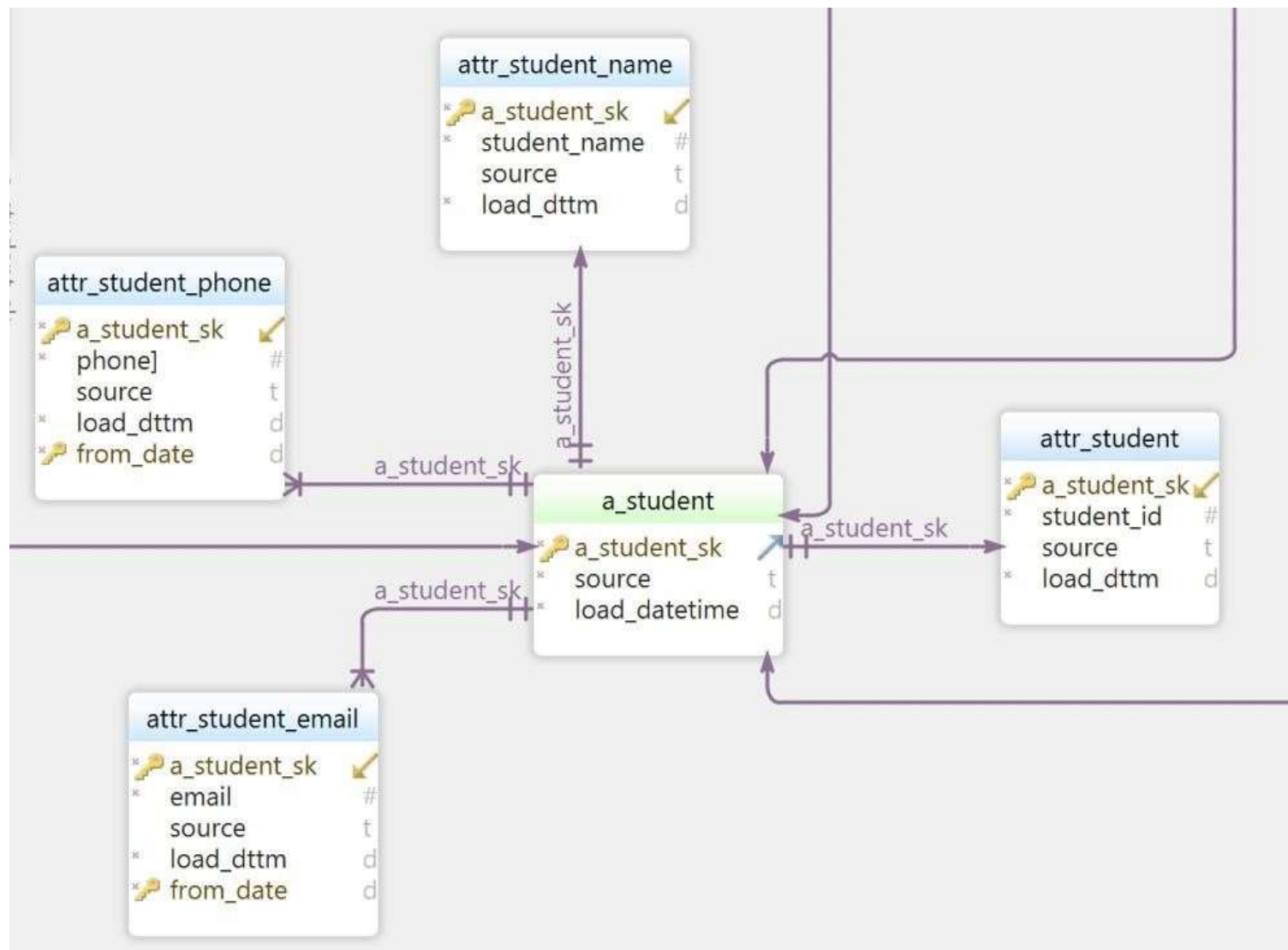
Пример — название курса, логин и дата рождения пользователя, суммы платежа.

Одно свойство у объекта — одна Attribute-таблица. Десять свойств у объекта (имя, фамилия, дата рождения, пол, адрес регистрации, ...) — десять Attribute-таблиц.

Каждая Attribute-таблица содержит суррогатный ключ объекта, которым является ссылка на соответствующий Anchor, поле для значения атрибута, и, опционально, дату для историчности и технические поля.



# Якорное моделирование — Attribute



# Якорное моделирование — Tie

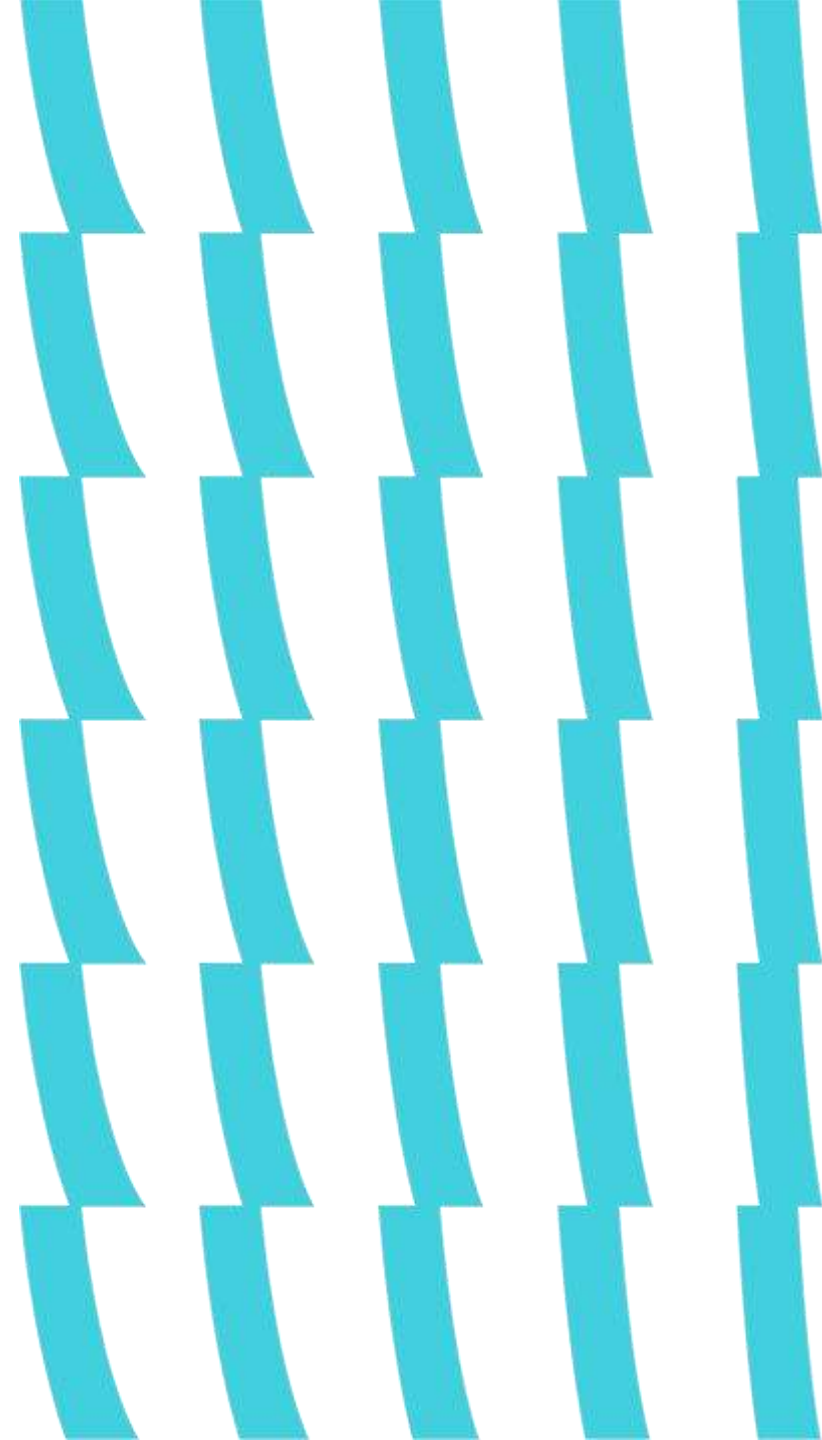
**Tie (Связь)** — это таблица для хранения связей между объектами.

Пример — таблица для хранения факта наличия у покупателя гражданства в определенной стране.

Соответственно, таблица должна содержать суррогатный ключ левого объекта (`customer_id`), правого объекта (`country_id`) и, по необходимости, даты историчности и технических полей.

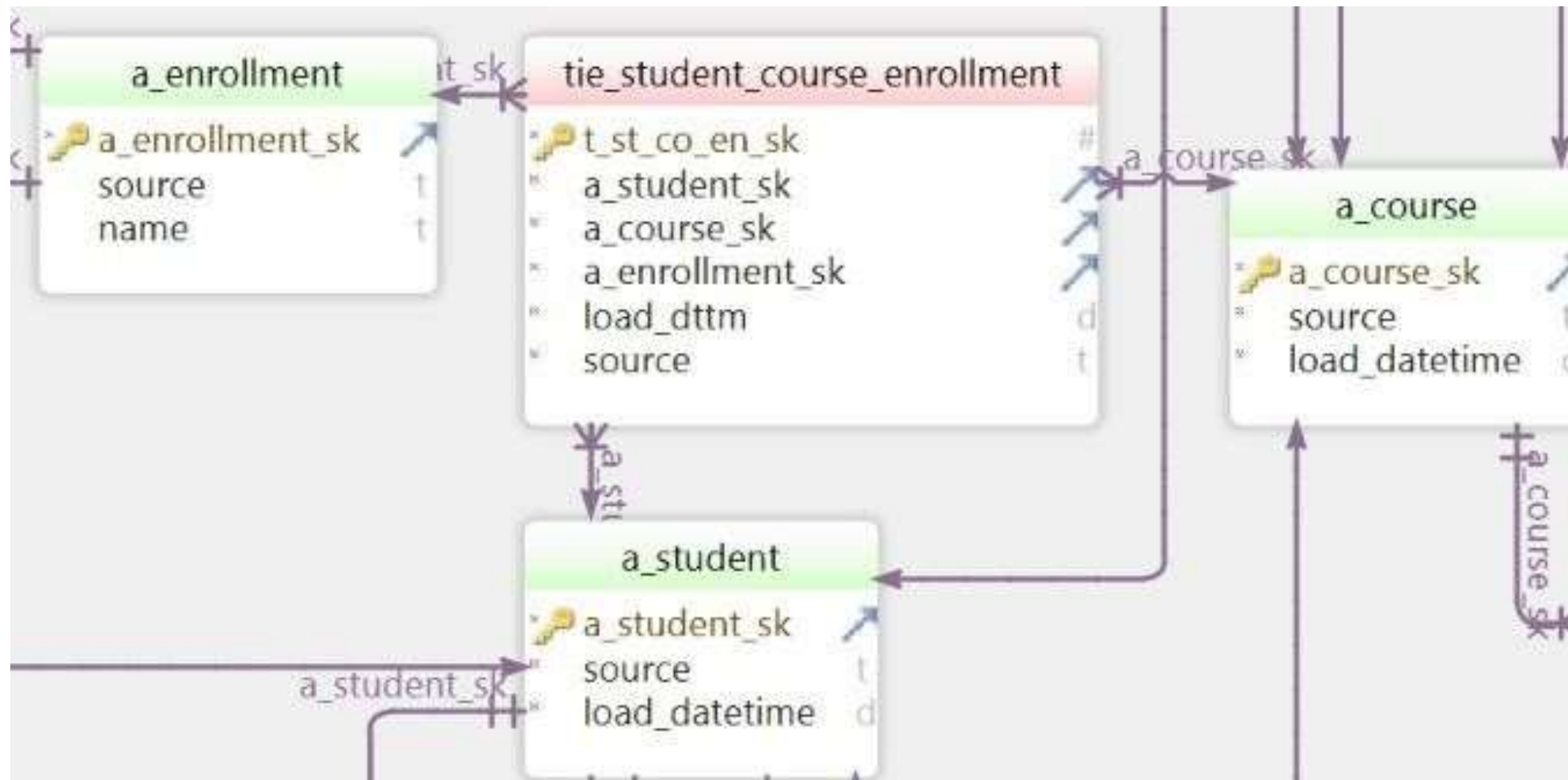
Важный нюанс с точки зрения моделей:

- В Data Vault можно вешать данные (сателлиты) на связь (link)
- В Anchor Modeling данные (Attribute) можно повесить только на Anchor, а на Tie нельзя (это важно! — совсем никак НЕЛЬЗЯ).

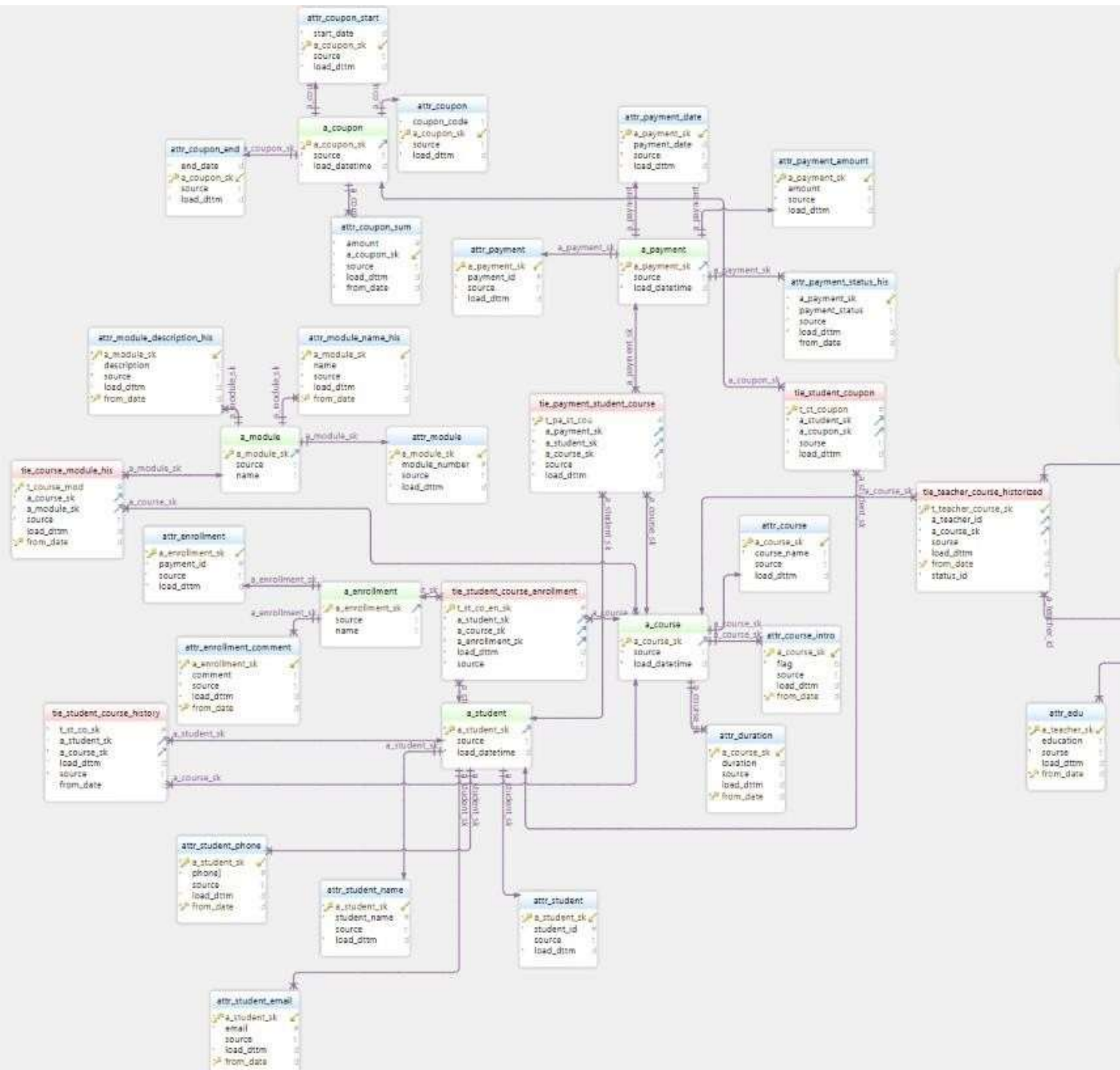




# Якорное моделирование – Tie







# Якорное моделирование — Knot

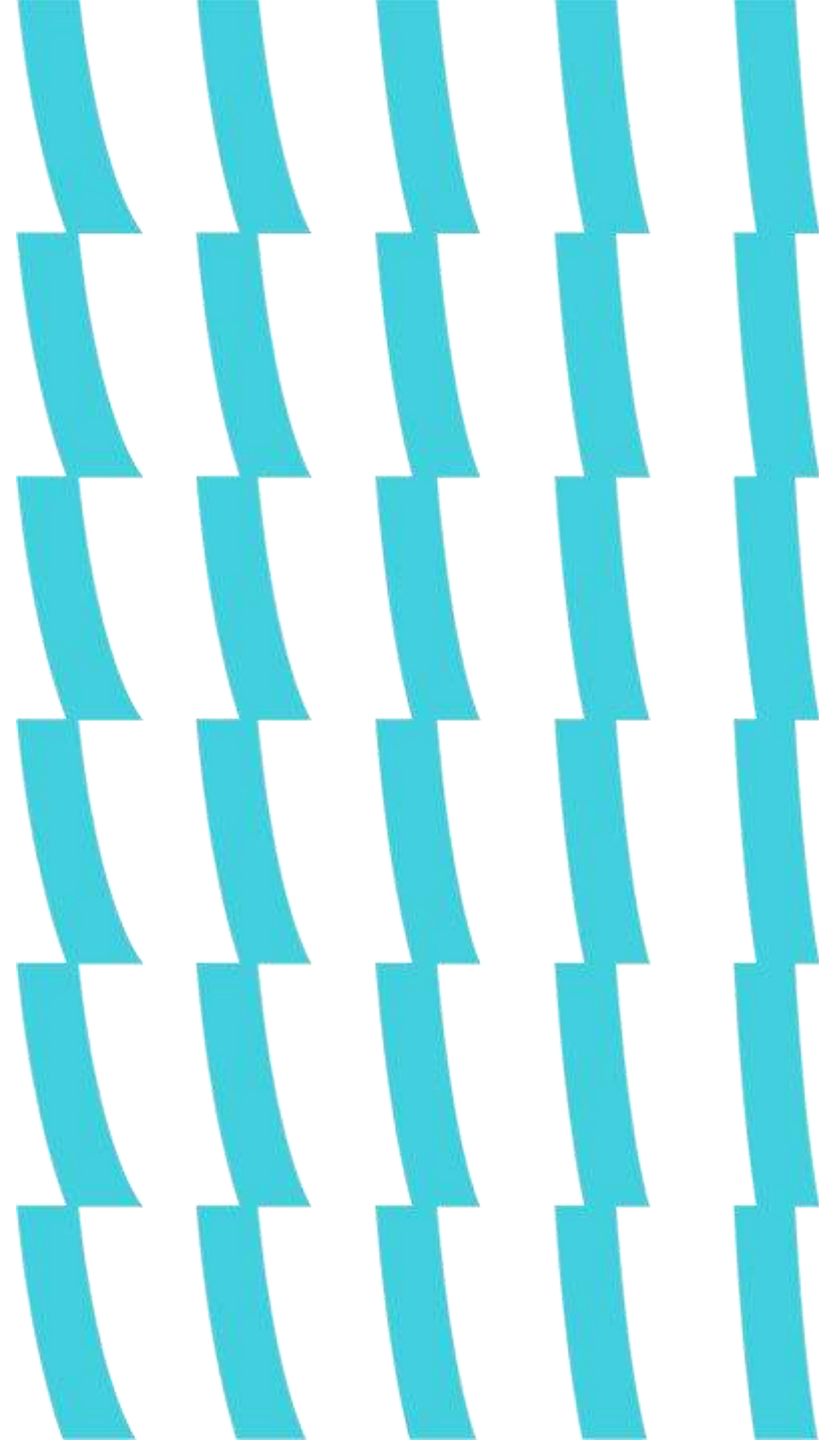
**Knot (Узел)** — таблица-состояние, справочник.

вырожденный вид якоря, может содержать всего один атрибут  
Узлы можно рассматривать как сочетание якоря и одного атрибута.

Таблицы узлов содержат следующие поля:

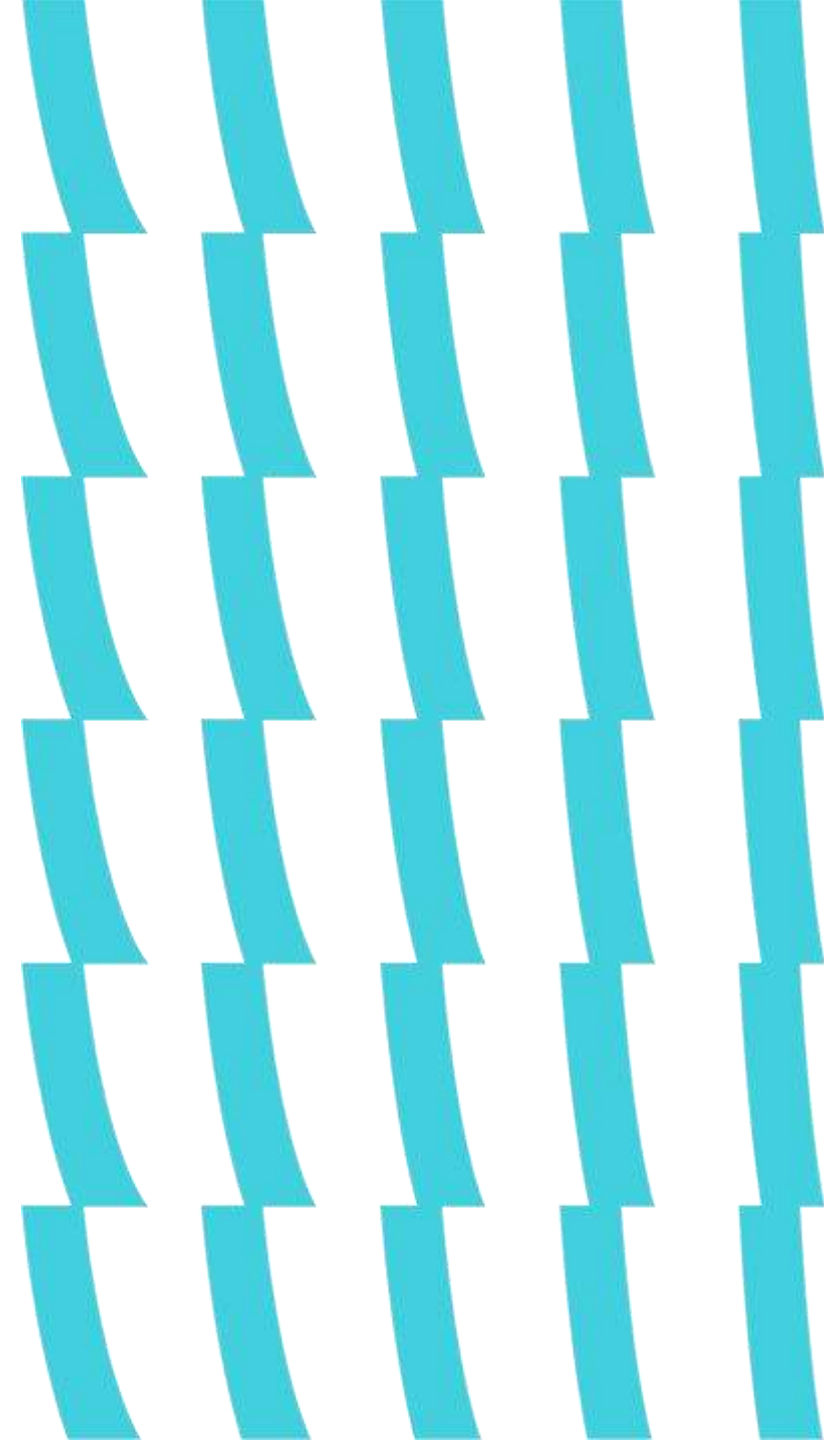
- Суррогатный ключ
- Значение
- Мета-поля

Узлы предполагается использовать для хранения плоских справочников (например пол, семейное положение, категория обслуживания клиентов и т.п). В отличие от Якоря, Узел не имеет связанных таблиц атрибутов, а его единственный атрибут (название) всегда хранится в одной таблице с ключом. Узлы связываются с Якорями таблицами-связями (Tie) также, как якоря друг с другом.



# Проектирование

1. Найдите основные бизнес-сущности (это якоря)
2. Исследуйте их (найдите другие якоря, связи)
3. Опишите их (атрибуты, узлы)
4. Историзируйте их, если нужно (атрибуты)
5. Определите взаимоотношения сущностей (связи)
6. Историзируйте взаимоотношения (связи)

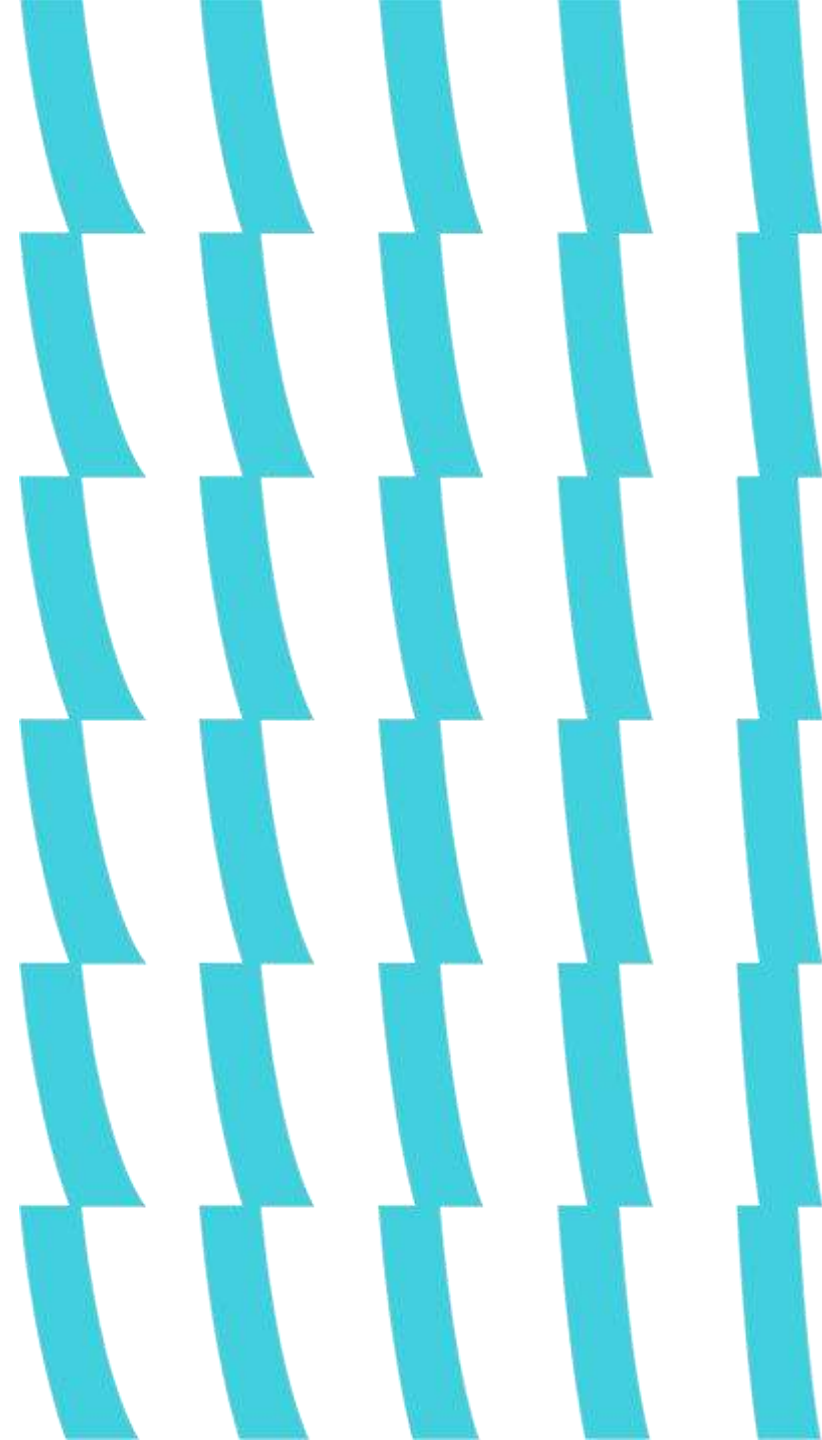


# Отличие от Data Vault

Модель предусматривает шестую и иногда пятую нормальную форму

Из этого вытекает следующее:

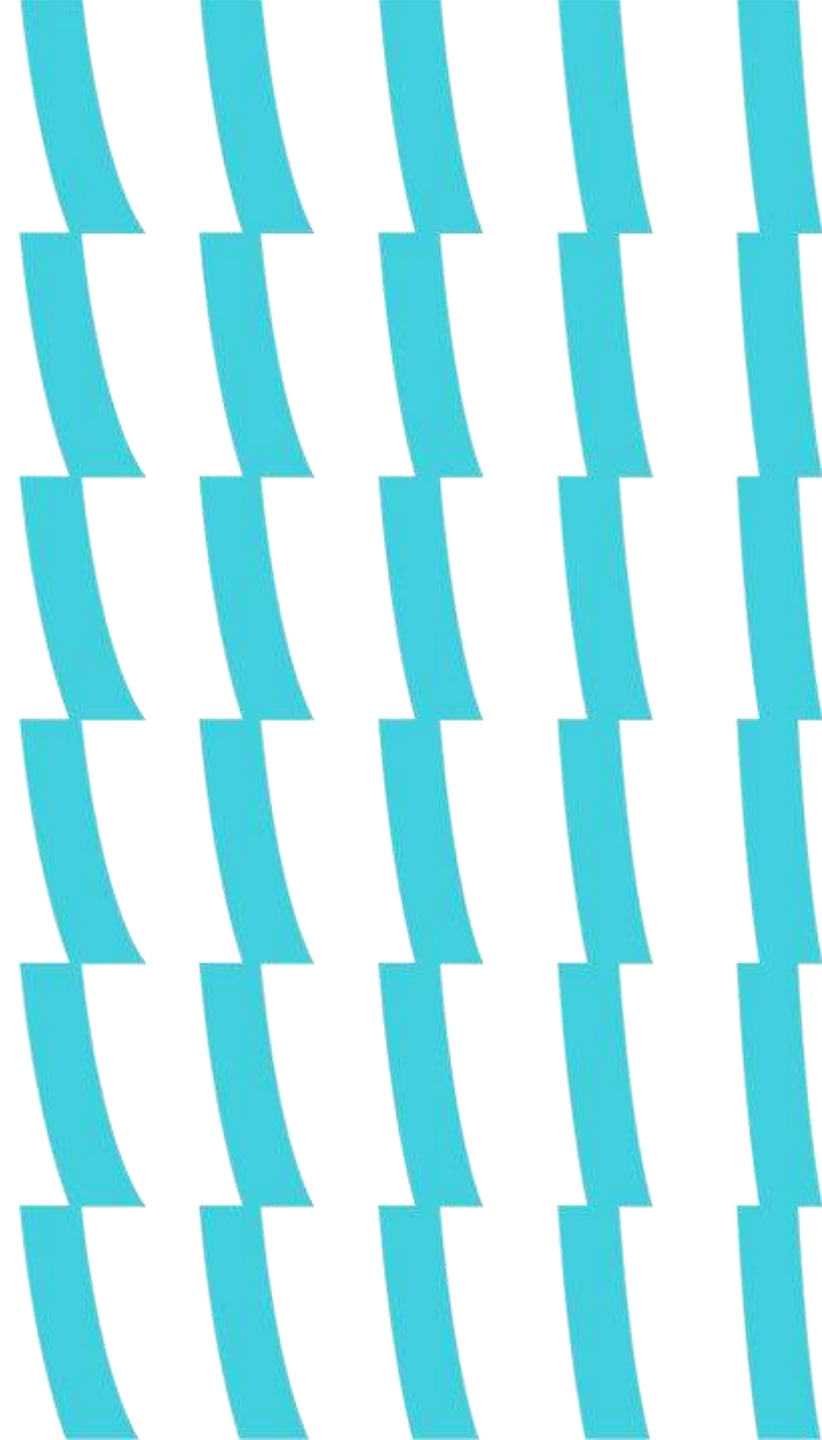
- Поддержка историчности. Data Vault предусматривает реализацию медленно меняющихся измерений типа 2 с одной или двумя датами — датой начала действия данных и опционально датой конца действия данных. В случае с двумя датами, когда приходит новая версия данных, старая версия должна быть обновлена для выставления даты конца действия данных (операция update). Если дата одна — это дата начала действия данных. Такой подход позволяет реализовывать концепцию insert-only ETL — загрузка данных без обновлений, только со вставками. В Якорном моделировании, так как таблицы находятся в 6НФ целостность компонентов обеспечивается наложением простых ограничений при выполнении изменения, записи или удаления.
- В Data Vault хабы содержат как идентификатор из системы-источника, так и суррогатный ключ. Якоря содержат только суррогатные ключи.
- В Data Vault можно вешать сателлиты на связи, а в якорном моделировании — нет.



# Отличие от Data Vault

Data Vault предполагает хранение атрибутов сущностей сгруппированными в таблицах-сателлитах. Например, если сущность обладает 50 атрибутами со сравнимой вероятностью изменений, методология Data Vault рекомендует хранить их в одной таблице. Методология Anchor Modelling в подобном случае требует создать 50 таблиц, по одной для каждого атрибута. Подход Anchor Modelling на первый взгляд кажется избыточным усложнением, хотя и удовлетворяет 6НФ. Сбор актуальных значений всех 50 атрибутов возможен только посредством создания специальных материализованных представлений. Без них аналитику, работающему с хранилищем, собирать атрибуты слишком сложно.

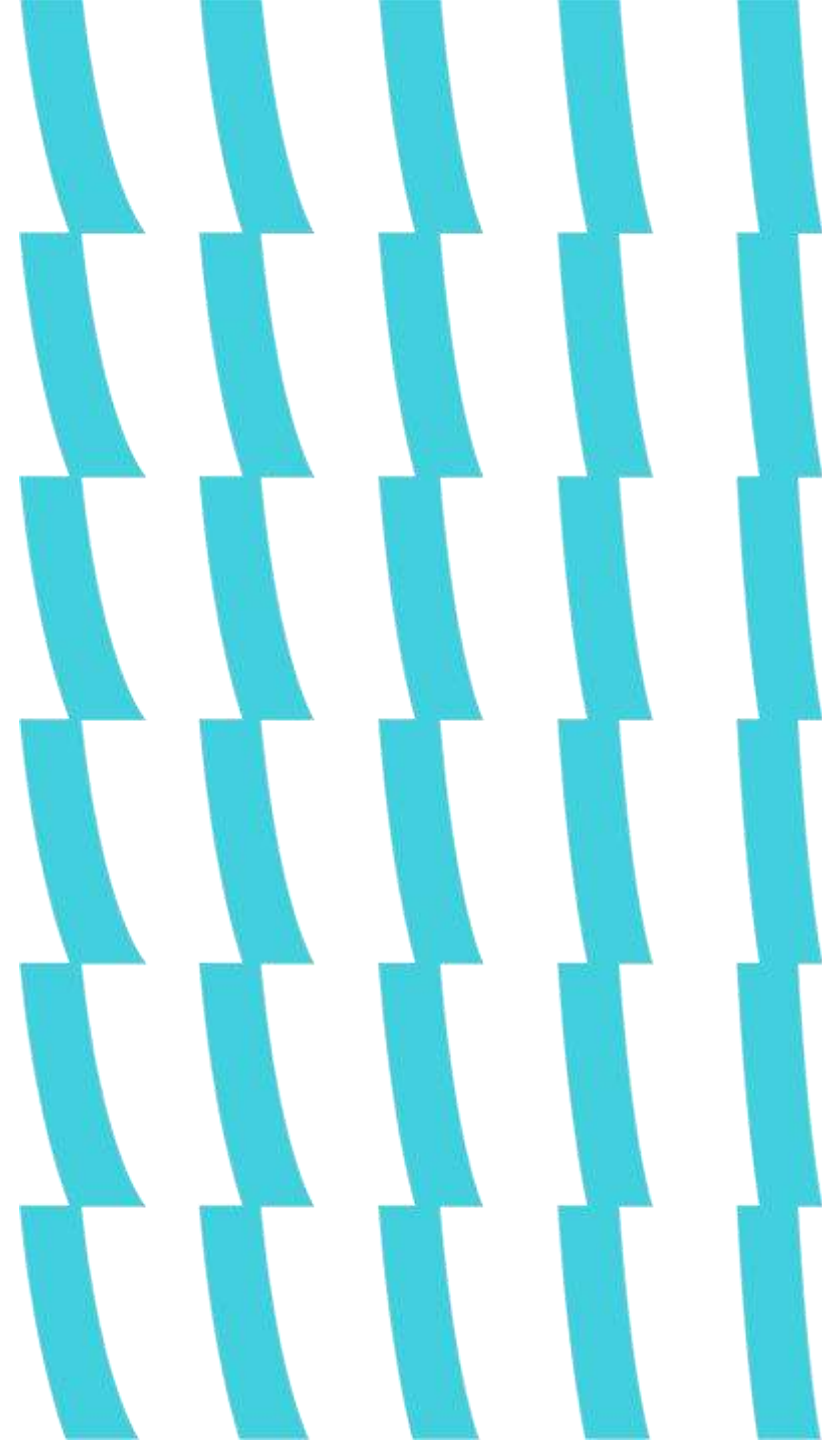
Все это приводит к более высокой управляемости и расширяемости хранилища.





# Достижение гибкости обоих подходов

- Распределяя атрибуты по Сателлитам (в Data Vault) или отдельным таблицам (Anchor Model), мы уменьшается(исключается) дублирование значений одних атрибутов при изменении других.
- Добавление как отдельных атрибутов, так и целых новых предметных областей в такой модели выглядит как **надстройка** над существующим набором объектов без их изменения.
- Благодаря декомпозиции на стандартные элементы, ETL-процессы в таких системах выглядят однотипно, их написание поддается алгоритмизации и, в конечном счете, автоматизации.





Спасибо  
за внимание!

## DIAGRAM EXPLORER

Search

## Tables (filtered by SubjectArea)

- Person.Address
- Person.AddressType
- Person.BusinessEntityAddress
- Person.BusinessEntityContact
- Person.ContactType
- Person.CountryRegion
- Person.EmailAddress
- Person.Password
- Person.Person
- Person.Person\_json
- Person.Person\_Temporal
- Person.Person\_Temporal\_History
- Person.PersonPhone
- Person.PhoneNumberType
- Person.StateProvince

## References

Notes

## Person Subject Area

## Person Diagram

