

# KARPOV.COURSES >>> КОНСПЕКТ



## > Конспект > 4 урок > Методология Data Vault: Практикум

- > Базовые сущности
- > Преобразование таблицы курсов
  - Базовая информация
  - Дополнительная информация
  - Деление информации на сателлиты
- > Преобразование таблицы блоков
- > Преобразование таблицы лекторов
  - Связь между лектором и курсом
- > Преобразование таблицы лекций
  - Сателлит
  - Итоговая схема
- > Связи между курсами, блоками, лекциями и лекторами
  - Связь между блоком и курсом
  - Связь между блоком и лекцией
  - Связь между блоком и лектором
  - Связь между лектором и лекцией
- > Обзор промежуточной схемы
- > Добавление информации о студентах

[Основные таблицы](#)

[Информация о просмотрах лекций](#)

[Сателлит с информацией о просмотре](#)

[Итоговая схема](#)

> [Информация о платежах](#)

[Ссылки](#)

> [Введение сущностей из Data Vault 2.0](#)

[Агрегированная информация по курсу](#)

[PIT для курсов](#)

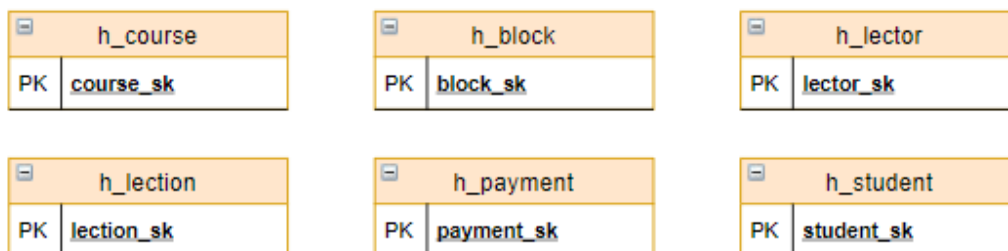
[Bridge для курсов, блоков и лекций](#)

> [Обзор конечной схемы](#)

## > Базовые сущности

В ходе этой практики мы снова вернемся к модели платформы самообразования и попробуем спроектировать модель по Data Vault.

Для начала создадим все базовые сущности.



Базовые сущности

Наши базовые сущности, от которых мы будем отталкиваться, это **курсы, блоки, лекторы, лекции, платежи и студенты**.

## > Преобразование таблицы курсов

Создадим хаб **h\_course**. Все хабы будут иметь префикс **h\_**.

Введем **course\_sk** – суррогатный ключ хаба. Также добавим обязательные поля:

- **load\_dttm** – дата выгрузки
- **source\_id** – код источника

В качестве бизнес-ключа возьмем наименование курса – `course_Name` (считаем, что оно никогда не повторяется).

h_course	
PK	<u>course_sk</u>
	course_Name
	load_dttm
	source_id

Хаб курсов

---

Добавим к курсам описательные характеристики с помощью сателлитов.

Сателлиты будут иметь префикс `S_`

## Базовая информация

Сначала добавим сателлит с базовой информацией о курсе – `s_course_info`. В качестве ключа, очевидно, будет выступать `course_sk`. Добавим информационные атрибуты:

- `description` – описание курса
- `for_whom` – для кого данный курс
- `target_profession` – для какой профессии курс

Связь между этим сателлитом и хабом будет **один-к-одному**. Для каждой записи в таблице хаба будет одна запись базовой информации о хабе.

s_course_info	
PK	<u>Course_sk</u>
	description
	for_whom
	target_profession
	load_dttm
	source_id

Сателлит базовой информации

## Дополнительная информация

В сателлите с дополнительной информацией **s\_course\_info** добавим следующую информацию:

- `difficulty_lvl` – сложность курса
- `duration` – длительность курса

Эту информацию мы будем хранить по SCD2, поэтому также добавим поля `valid_from_dttm` (дата начала действия записи, входит в первичный ключ) и `valid_to_dttm` (дата окончания действия записи).

Связь `{h_course, s_course_difficulty}` будет **один-ко-многим**.

s_course_difficulty	
PK	<u>Course_sk</u>
PK	valid_from_dttm
	valid_to_dttm
	difficulty_lvl
	duration
	source_id
	load_dttm

Сателлит дополнительной информации

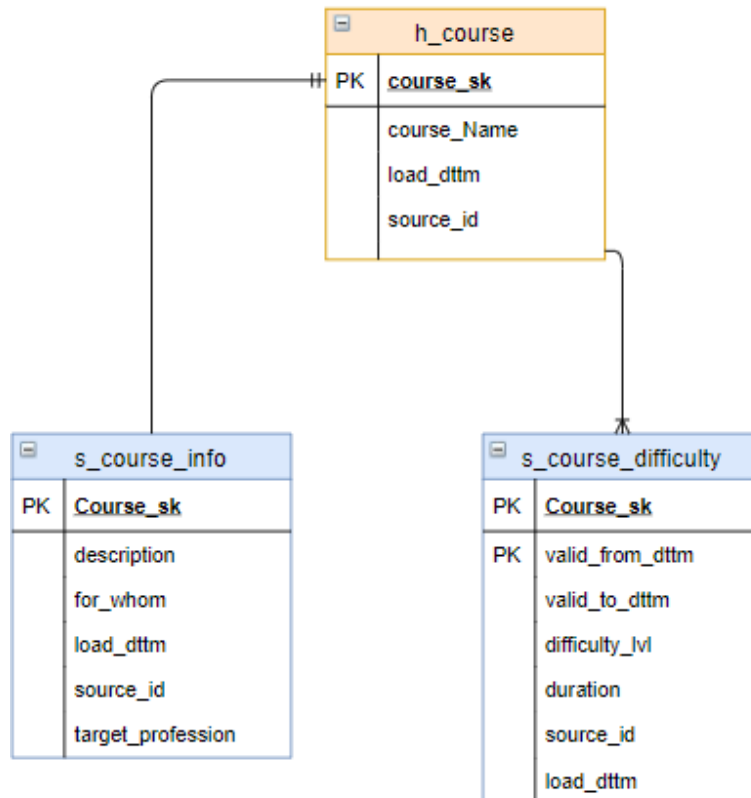
## Деление информации на сателлиты

Есть базовые правила, например, если информация приходит из одного источника, то она попадает в один сателлит. Мы же представим, что вся информация по курсам поступает из одного источника.

Мы не хотим, чтобы имя, описание и формат курса дублировались каждый раз при изменении сложности или длительности курса.

Также, если некоторая информация из сателлита часто используется для аналитических вопросов, то имеет смысл выделить её в отдельный сателлит.

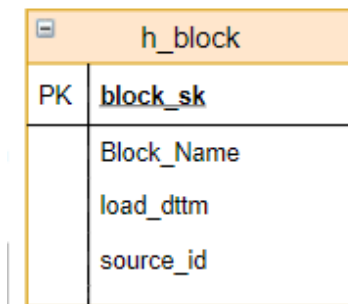
Итоговая схема выглядит следующим образом:



Сущность курса в DV

## > Преобразование таблицы блоков

Создадим хаб **h\_block**. Пусть имя блока будет бизнес-ключом **Block\_Name** (считаем, что оно уникально). Также стандартные поля **load\_dttm** и **source\_id**.



Хаб блока

Добавим также спутник с базовой информацией о блоке **s\_block\_info**. В него включим поля:

- **Name** – наименование блока
- **Description** – описание блока

Связь между хабом блока и спутником будет один-к-одному.

s_block_info	
PK	<u>block_sk</u>
	description
	name
	load_dttm
	source_id

Спутник с базовой информацией о блоке

Аналогично курсу, создадим еще один спутник с дополнительной информацией о блоке **s\_block\_difficulty**. В него войдут поля:

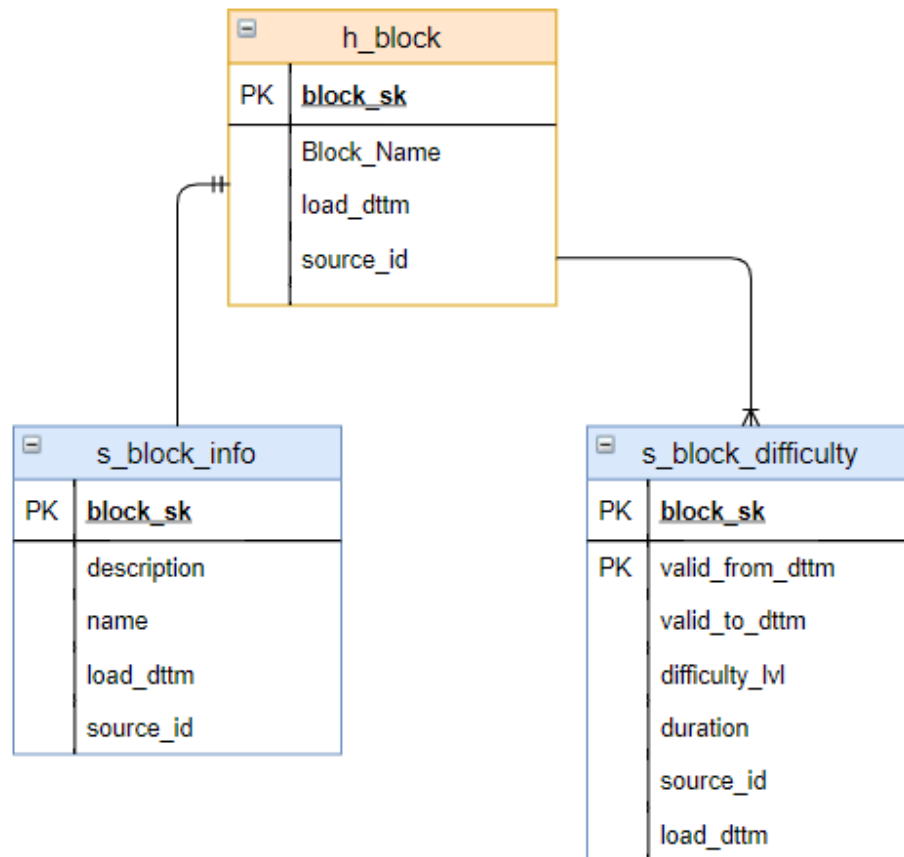
- **difficulty\_lvl** - уровень сложности блока
- **duration** - продолжительность блока

Связь {h\_block, s\_block\_difficulty} будет **один-ко-многим**.

s_block_difficulty	
PK	<u>block_sk</u>
PK	<u>valid_from_dttm</u>
	valid_to_dttm
	difficulty_lvl
	duration
	source_id
	load_dttm

Спутник с дополнительной информацией о блоке

Итоговая схема связей:



Сущность блока в DV

## > Преобразование таблицы лекторов

Создадим хаб лекторов **h\_lector**. В качестве бизнес-ключа добавим поле `FIO` (считаем, что все лекторы имеют разные ФИО). Также добавим стандартные поля `load_dttm` и `source_id`.



h_lector	
PK	<u>lector_sk</u>
	FIO
	load_dttm
	source_id

Хаб лекторов

Добавим сателлит с информацией о лекторах. В него включим следующие поля:

- **Company** – компания, в которой работает лектор
- **Profession** – профессия лектора

Связь между хабом и этим сателлитом будет один-к-одному.

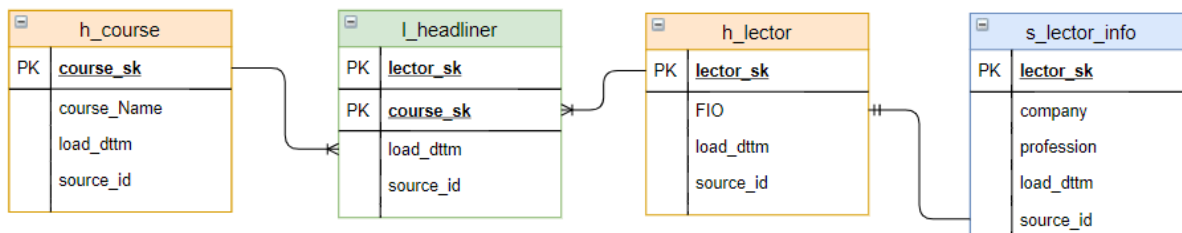
s_lector_info	
PK	<u>lector_sk</u>
	company
	profession
	load_dttm
	source_id

Сателлит с информацией о лекторе

## Связь между лектором и курсом

Добавим новую таблицу связи между лекторами и курсами – линк **l\_headliner**. Линки мы будем обозначать префиксом **l\_**.

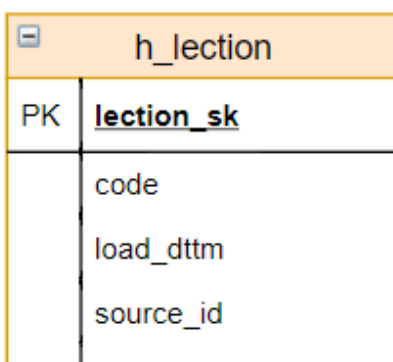
Этот линк будет содержать лишь ключи **lector\_sk** и **course\_sk** (ну и стандартные **load\_dttm** и **source\_id**). Дополнительной информации у этого линка не будет.



Итоговая схема вместе с линком к курсам

## > Преобразование таблицы лекций

Создадим хаб **h\_lection**. В качестве бизнес-ключа выберем `code` – код лекции (считаем, что он уникален у всех лекций). Ну и стандартные поля `load_dttm` и `source_id`.



Хаб лекции

## Сателлит

Создадим сателлит **s\_lection\_info** с информацией о лекции. В нем будут:

- `duration` – длительность лекции
- `difficulty_lvl` – сложность лекции

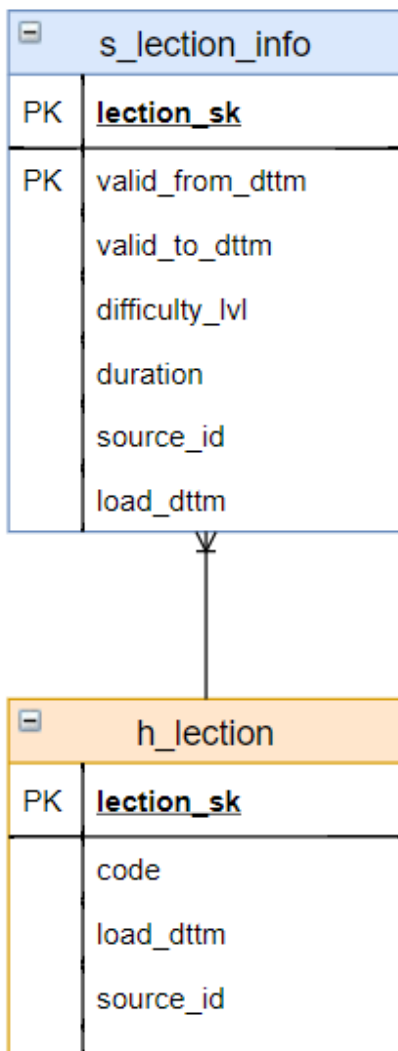
Также для хранения истории добавим поля `valid_from_dttm` и `valid_to_dttm`.

s_lection_info	
PK	<u>lection_sk</u>
PK	valid_from_dttm
	valid_to_dttm
	difficulty_lvl
	duration
	source_id
	load_dttm

Связь между этим сателлитом и хабом лекций будет **один-ко-многим**.

---

## Итоговая схема

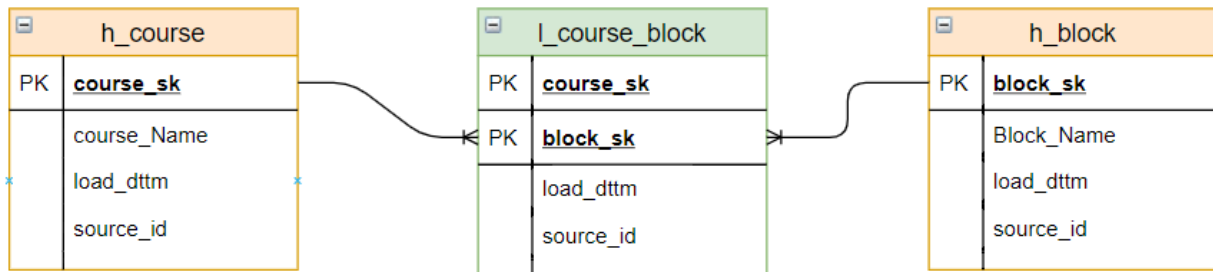


Сущность лекции в DV

## > Связи между курсами, блоками, лекциями и лекторами

### Связь между блоком и курсом

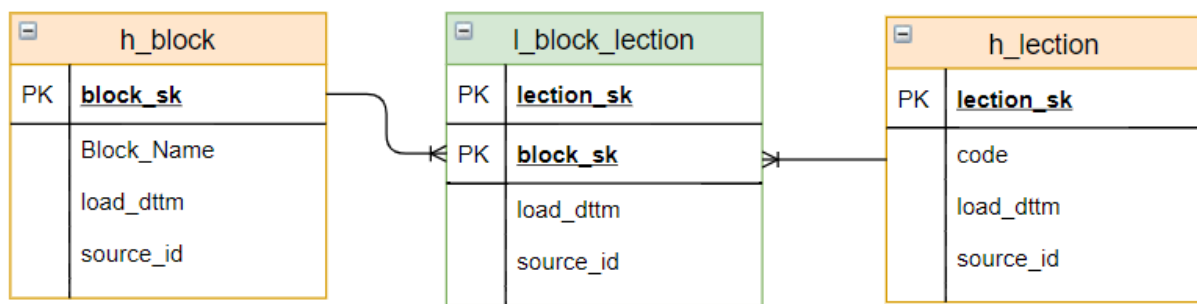
Добавим линк между блоками и курсами **l\_course\_block**. В нем, очевидно, будут ключи **course\_sk** и **block\_sk**.



Линк между курсами и блоками

## Связь между блоком и лекцией

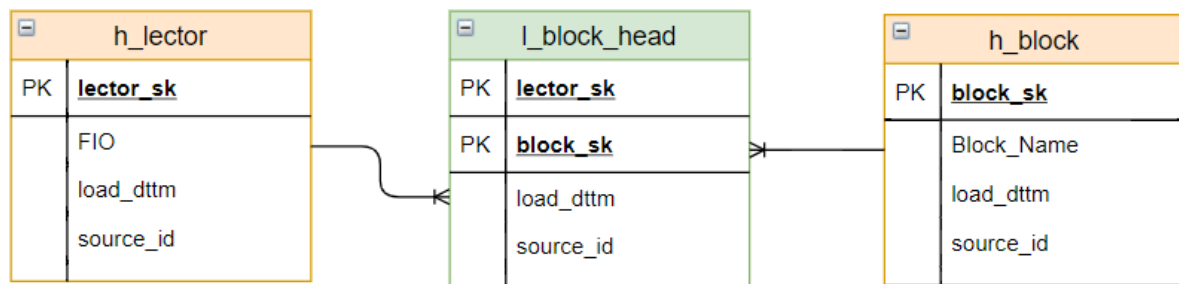
Добавим линк **l\_block\_lection**. Ключи – **block\_sk** и **lection\_sk**.



Линк между блоками и лекциями

## Связь между блоком и лектором

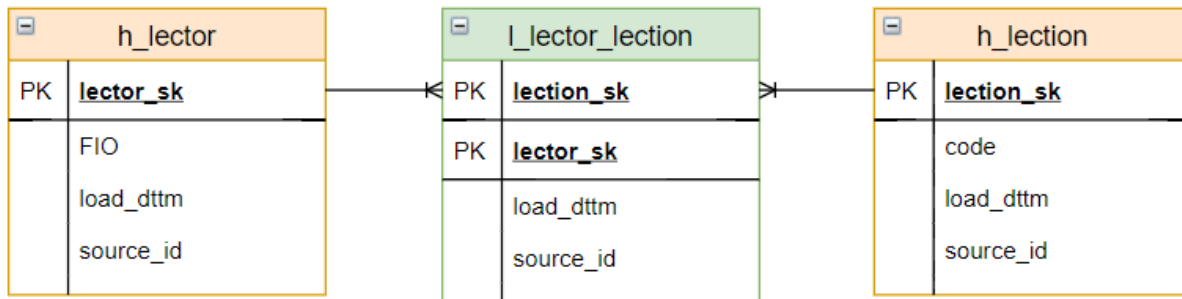
Добавим линк **l\_block\_head**. Ключи будут соответственно **block\_sk** и **lector\_sk**.



Линк между блоками и лекторами

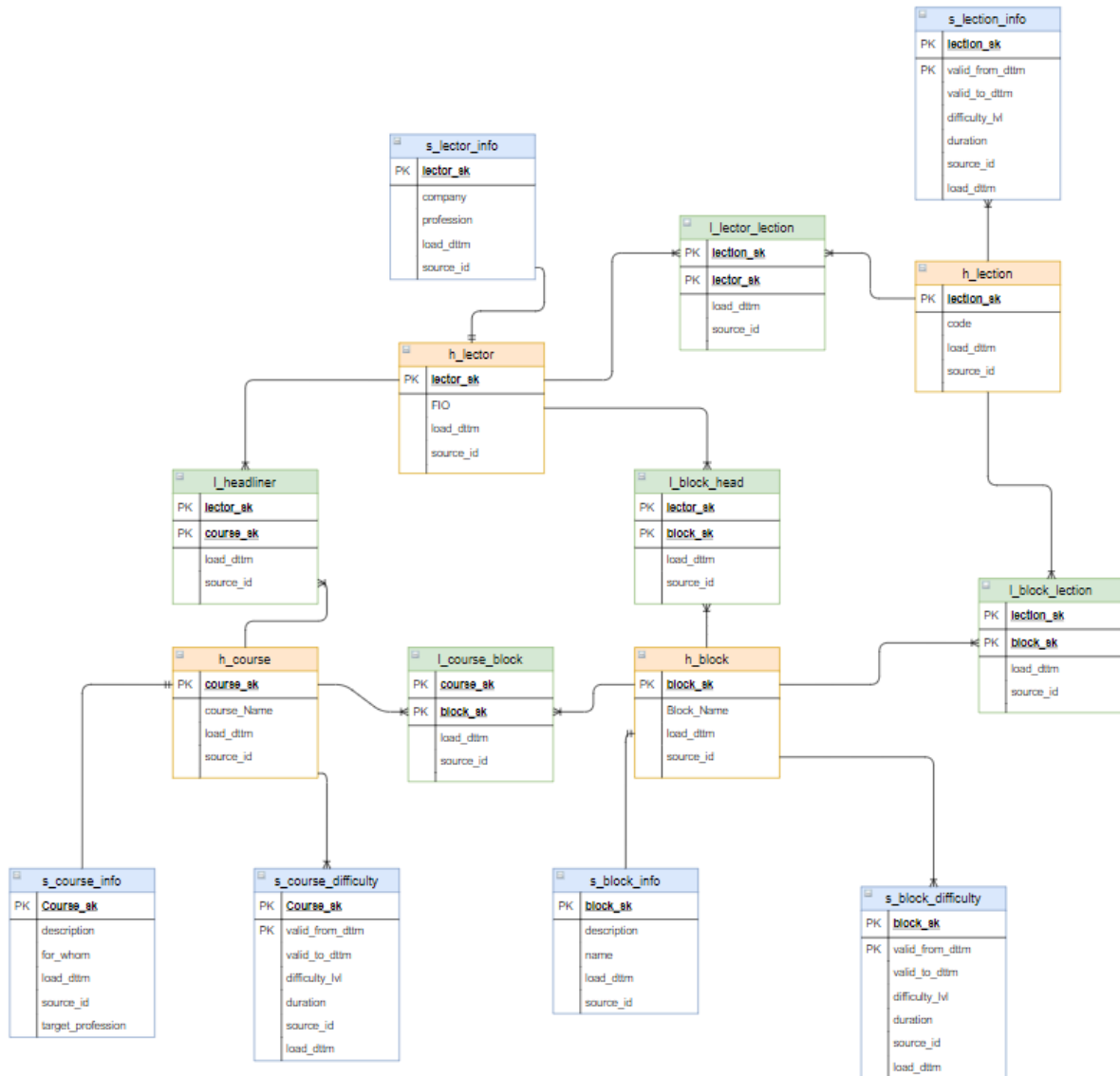
## Связь между лектором и лекцией

Добавим линк I\_lector\_lection. Ключи – `lector_sk` и `lection_sk`.



Линк между лекциями и лекторами

## > Обзор промежуточной схемы



Промежуточная схема

Схема получилась достаточно объемной, хотя:

- Мы даже не добавили информации о студентах
- В схеме толком нет ничего, что потребовалось бы для анализа данных (платежи, просмотры, домашние задания)

На схеме лишь информация о структуре курсов/блоков/лекций.

## > Добавление информации о студентах

## Основные таблицы

Создадим хаб **h\_student**. В качестве бизнес ключа используем ФИО студента **FIO**, регион **region\_name** и также дату рождения **bday** (считаем, что такая комбинация будет уникальна). Добавим стандартные **load\_dttm** и **source\_id**.

h_student	
PK	<u>student_sk</u>
	FIO
	bday
	region_name
	load_dttm
	source_id

Хаб студентов

Добавим также сателлит **s\_student\_socdem** с нашей собственной информацией о студенте. Поля сателлита:

- **age\_cls** – возраст
- **gender** – пол

Добавляем поля **valid\_from\_dttm** и **valid\_to\_dttm** для хранения истории.

Постфикс **\_cls** обозначает, что мы используем разделение по классам (уровням), так как, например, точной информации о возрасте у нас нет.

Связь хаба с сателлитом будет **один-ко-многим**.



s_student_socdem	
PK	<u>student_sk</u>
	age_cls
	gender
	valid_to_dttm
PK	valid_from_dttm
	source_id
	load_dttm

Сателлит с информацией о студенте

Также создадим еще один сателлит **s\_student\_ext\_socdem** с расширенной информацией о студентах. В него добавим:

- `salary_cls` - класс зарплаты
- `age_cls` - возраст
- `interest_cls` - класс интересов

Также добавляем историю и, соответственно, поля `valid_from_dttm` и `valid_to_dttm`.

Данный сателлит будет соединяться связью один-ко-многим.

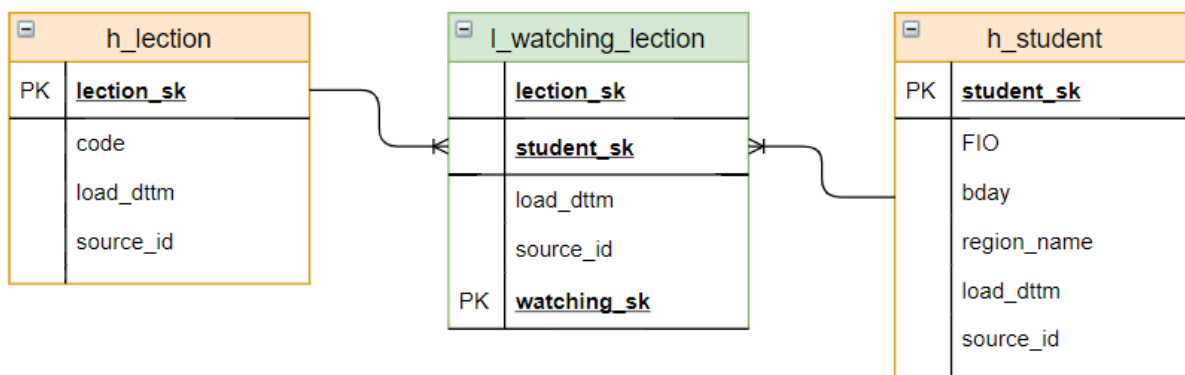
s_student_ext_socdem	
PK	<u>student_sk</u>
PK	valid_from_dttm
	valid_to_dttm
	salary_cls
	age_cls
	interest_cls
	source_id
	load_dttm

Сателлит с расширенной информацией о студенте

## Информация о просмотрах лекций

Добавим линк **I\_watching\_lection** с суррогатным ключом **watching\_sk**. В нем будут поля **student\_sk** и **lection\_sk**. Также добавим стандартные **load\_dttm** и **source\_id**.

Связи с хабами будут **ОДИН-КО-МНОГИМ**.



Линк с информацией о просмотрах лекций

## Сателлит с информацией о просмотре

Добавим сателлит **s\_watching\_info** с информацией о просмотре:

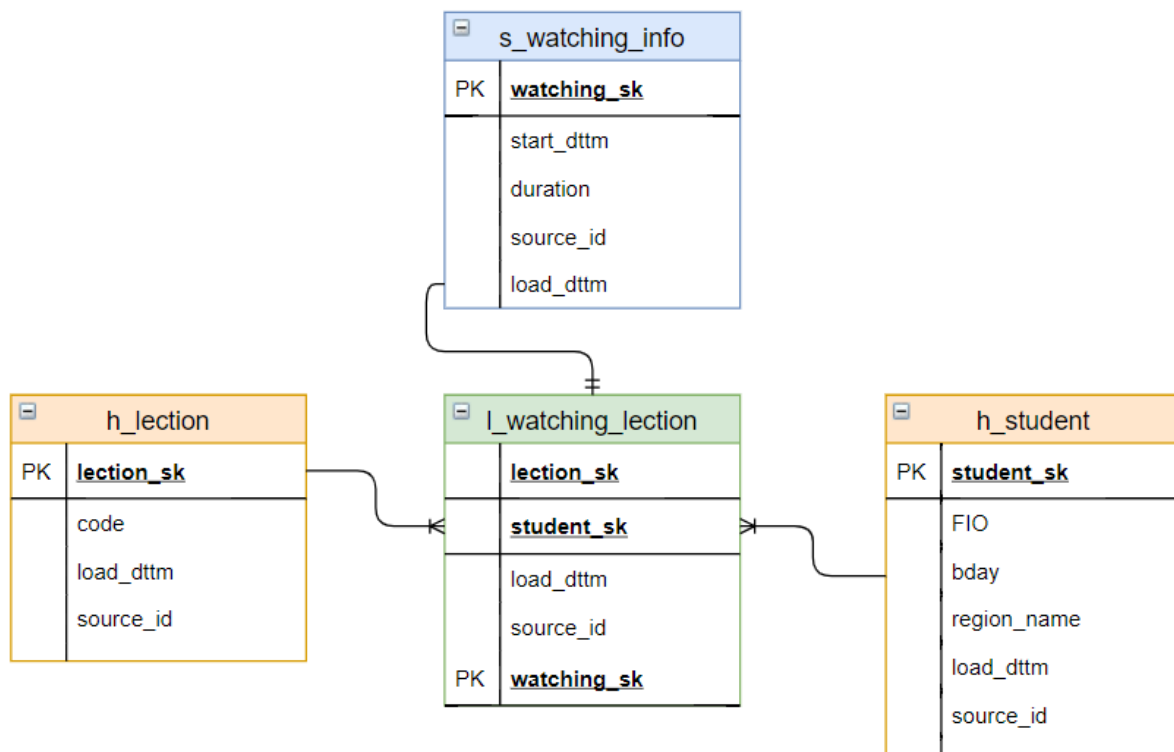
- **Duration** – длительность просмотра
- **Start\_dttm** – Начало просмотра

s_watching_info	
PK	<u>watching_sk</u>
	start_dttm
	duration
	source_id
	load_dttm

Сателлит с информацией о просмотрах лекций

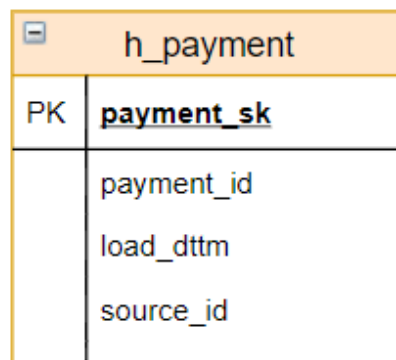
---

## Итоговая схема



## > Информация о платежах

Добавим хаб **h\_payment** с суррогатным ключом `payment_sk`. Бизнес ключ - `payment_id`. Также добавим стандартные `load_dttm` и `source_id`.

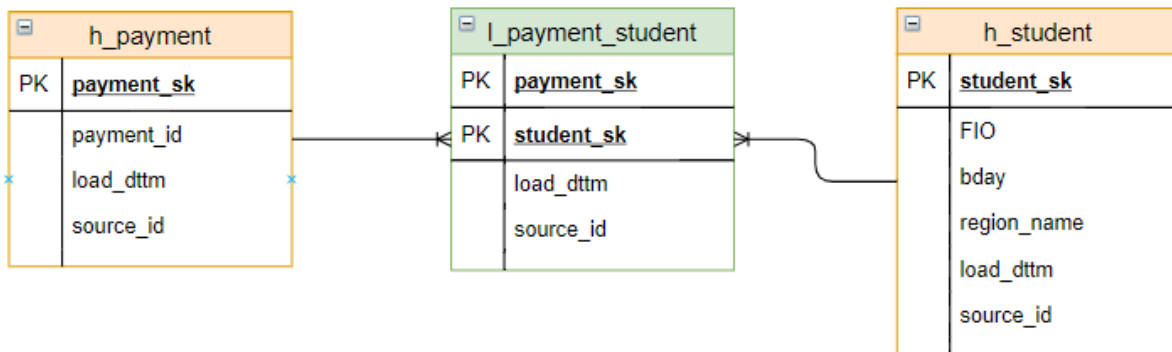


Хаб платежа

## Линки

Свяжем хабы платежа и студента через отдельный линк **l\_payment\_student**. Ключи

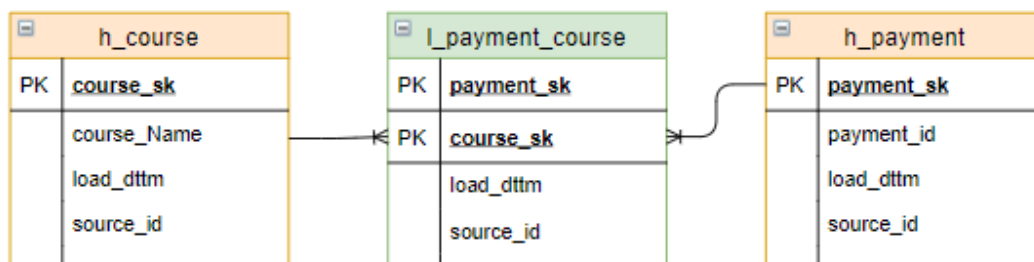
– `payment_sk` и `student_sk`.



Линк студента и платежа

Таким же образом свяжем хабы платежа и курса линком **l\_payment\_course**. Ключи

– `payment_sk` и `course_sk`.



Линк курса и платежа

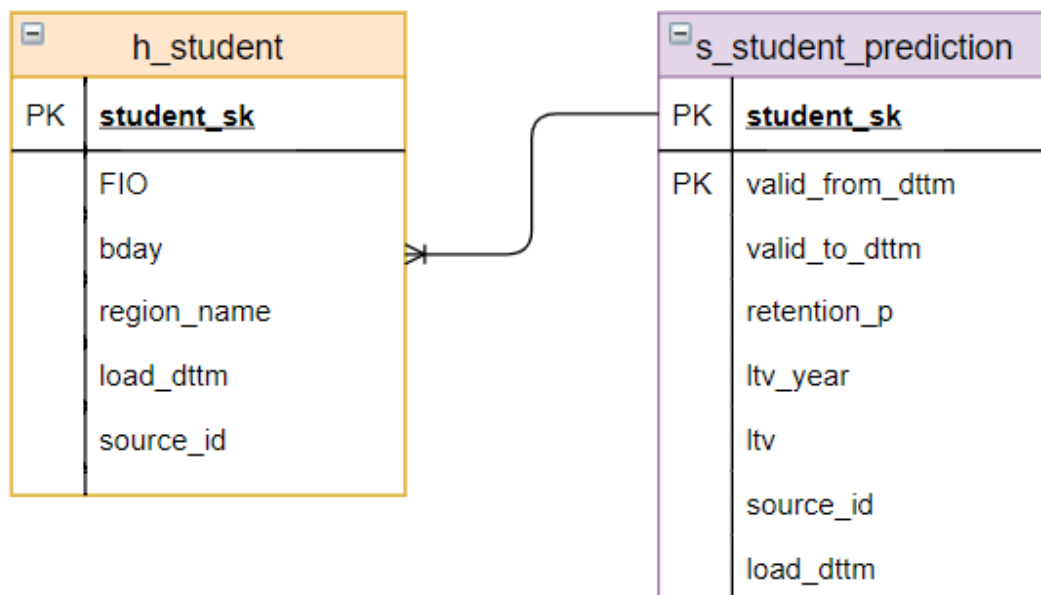
## > Введение сущностей из Data Vault 2.0

### Агрегированная информация по курсу

Добавим спутник с агрегированной информацией по студенту **s\_student\_prediction**. Он будет содержать первичный ключ `student_sk`, а также поля с предрасчитанными значениями:

- `retention_p` – метрика удержания студента
- `ltv_year` – Lifetime value (прибыль от студента за год)
- `ltv` - общий Lifetime value

А также будем вести историю по SCD2 – добавим `valid_from_dttm` и `valid_to_dttm`.



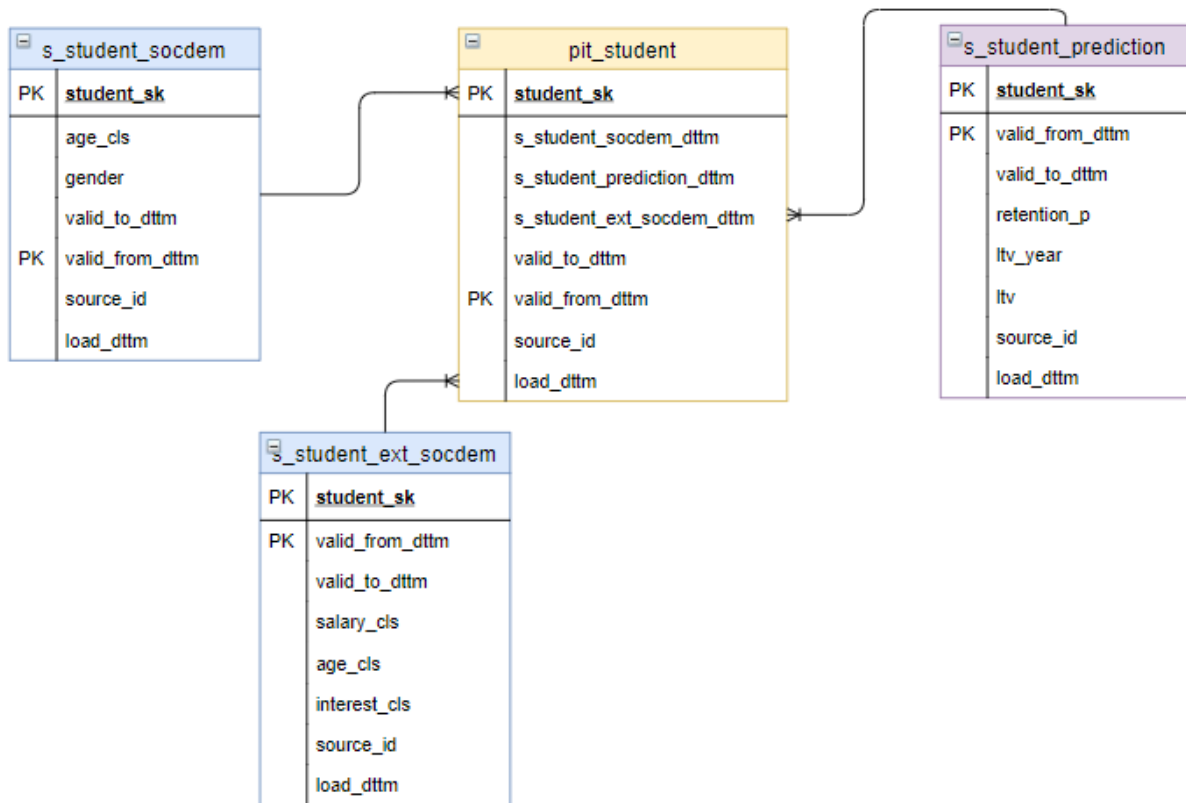
Сателлит с агрегированной информацией по студенту

## PIТ для курсов

Для того, чтобы воссоздать информацию по студенту, используя `s_student_socdem` и `s_student_ext_socdem`, нам нужно делать сложный временной join. Поэтому создадим Point-in-time таблицу `pit_student`.

В качестве первичного ключа будет использоваться `student_sk`. Добавим поля `s_student_socdem_dttm`, `s_student_prediction_dttm` и `s_student_ext_socdem_dttm`. Добавим также `load_dttm`, `source_id`, `valid_from_dttm` и `valid_to_dttm`.

Связь с каждым из сателлитов будет **ОДИН-КО-МНОГИМ**.

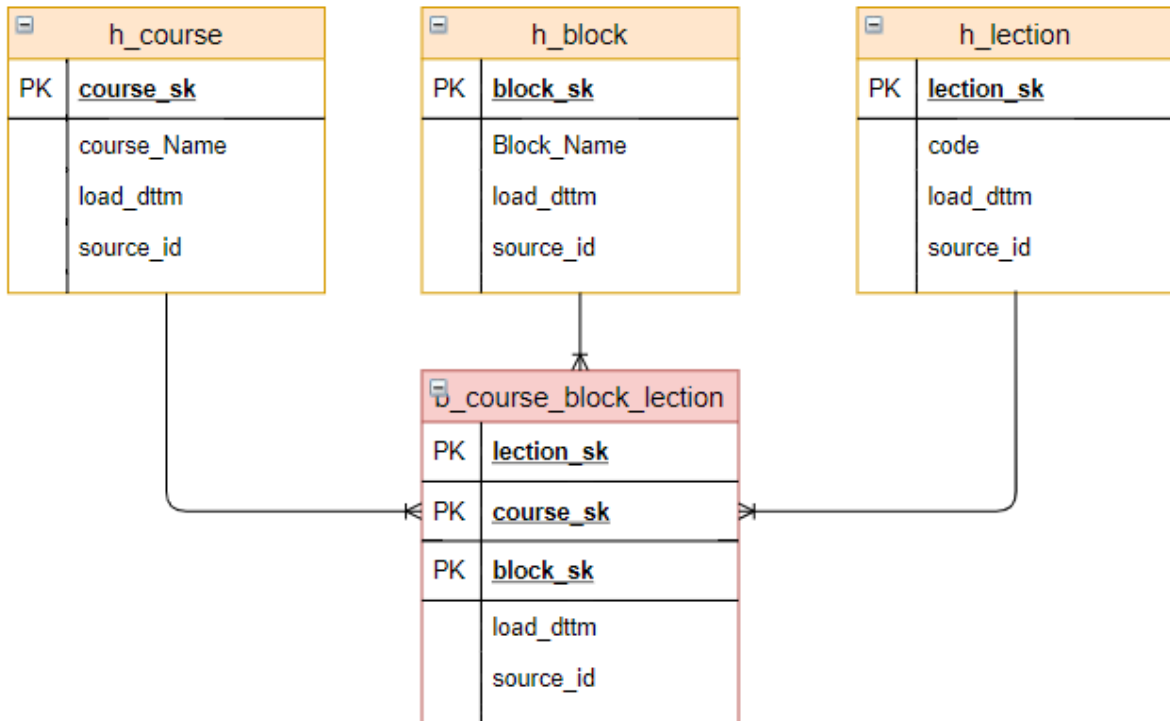


РІТ таблиця студентів

## Bridge для курсов, блоків і лекцій

Bridge поможет нам упростить линки. Добавим bridge **b\_course\_block\_lection**.

Первичный ключ будет состоять из ключей `course_sk`, `lection_sk` и `block_sk`. Связь – **ОДИН-КО-МНОГИМ**.



Bridge курсов, блоков и лекций

## > Обзор конечной схемы



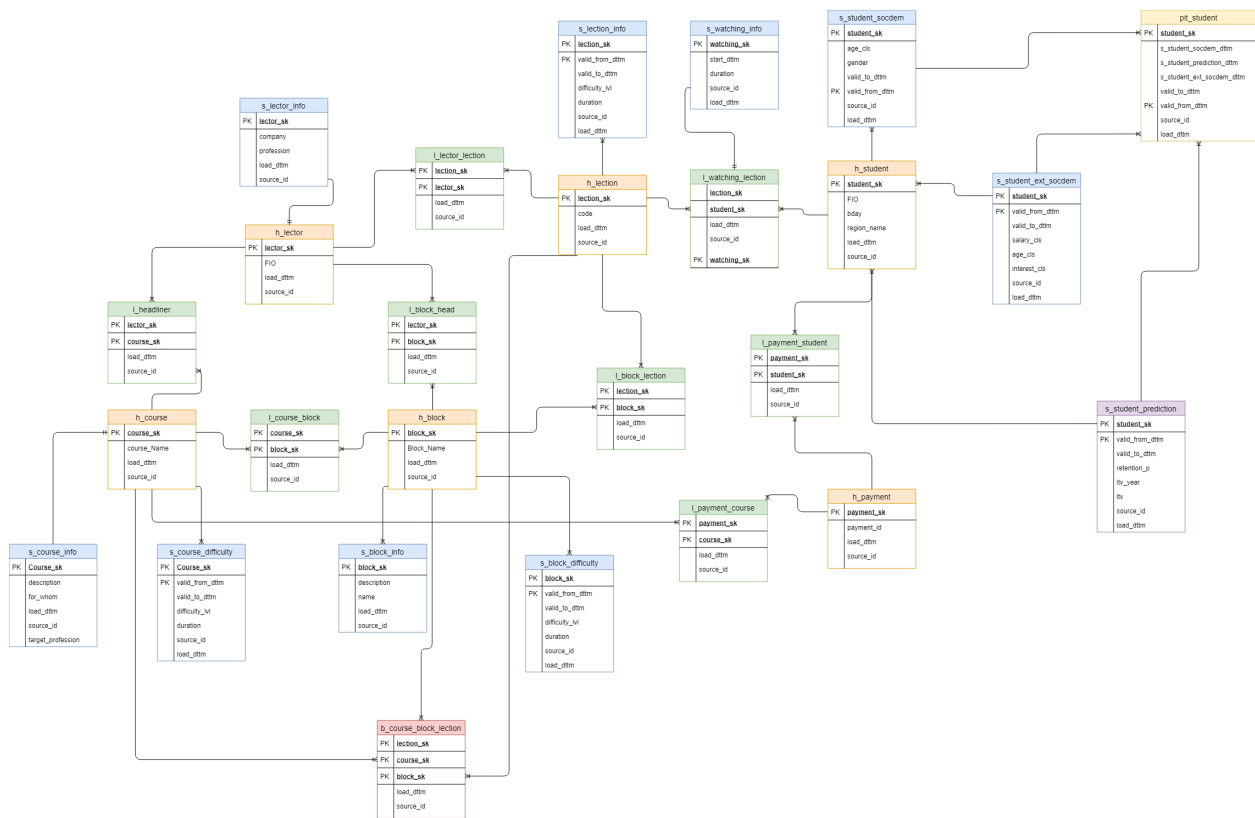


Схема получилась достаточно объемной, хотя мы использовали лишь 6 сущностей и 1 транзакционный линк (просмотры). Схема очевидно подтверждает, что Data Vault генерирует сильно больше таблиц, чем при классическом подходе.

Главное преимущество – расширяемость. Изменение кардинальности не является проблемой. Для добавления новых атрибутов к сущностям достаточно создать новый спателлит.

Скачать файл со всеми построенными схемами можно [здесь](#) а открыть [здесь](#).