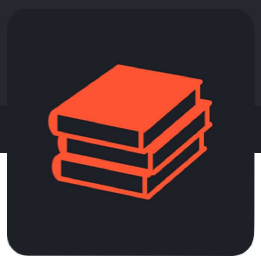
KARPOV.COURSES >>>
КОНСПЕКТ



> 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

Линки

> Введение сущностей из Data Vault 2.0

Агрегированная информация по курсу

PIT для курсов

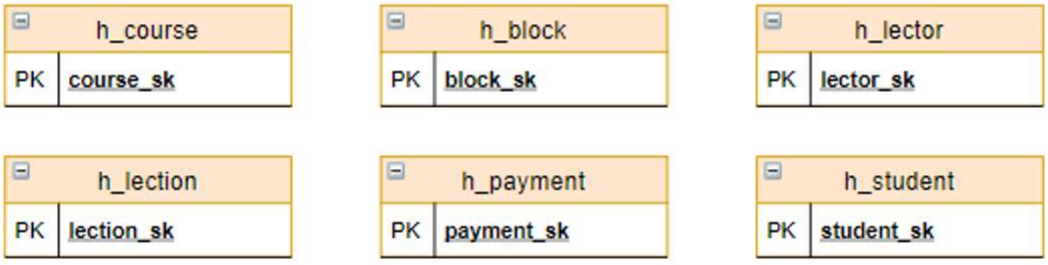
Bridge для курсов, блоков и лекций

> 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.**

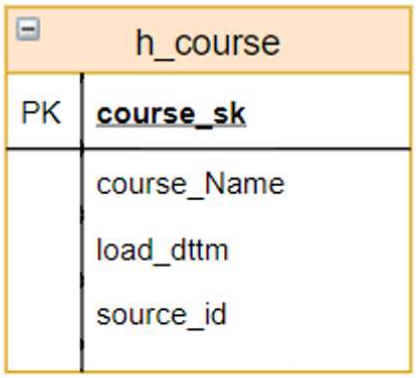
> Преобразование таблицы курсов

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).



Хаб курсов

Добавим к курсам описательные характеристики с помощью сателлитов.

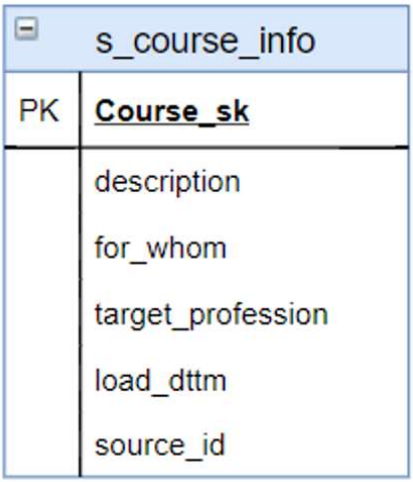
Сателлиты будут иметь префикс s_

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.



Сателлит базовой информации

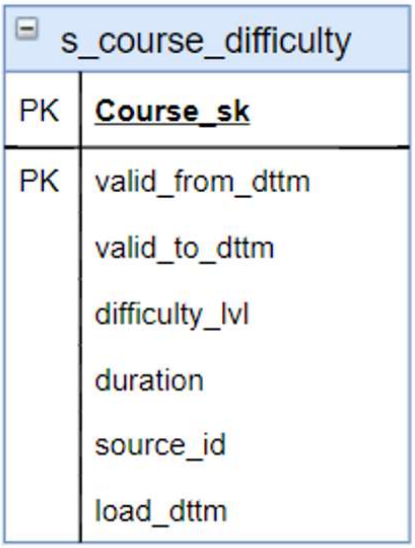
Дополнительная информация

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.



Сателлит дополнительной информации

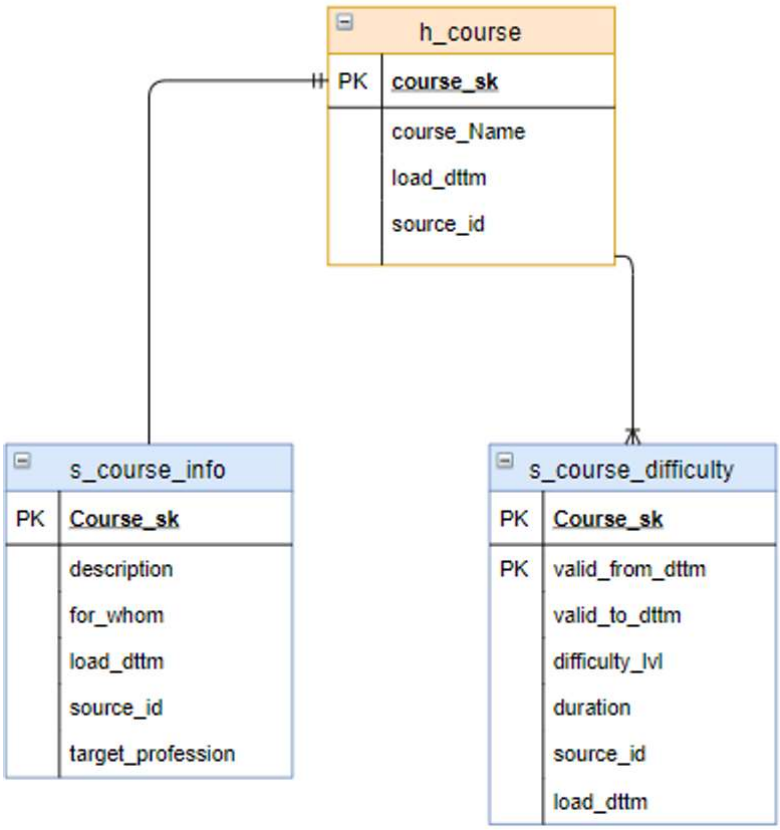
Деление информации на сателлиты

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.

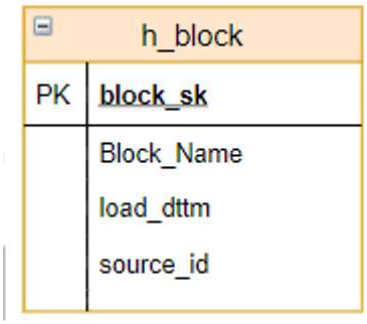
Итоговая схема выглядит следующим образом:



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`.

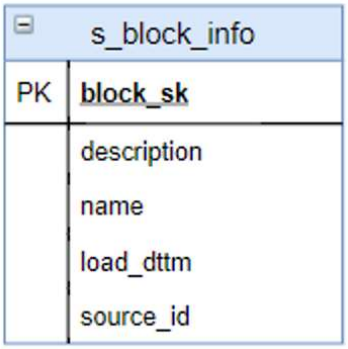


Хаб блока

Добавим также сателлит с базовой информацией о блоке **s_block_info**. В него включим поля:

- `Name` – наименование блока
- `Description` – описание блока

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

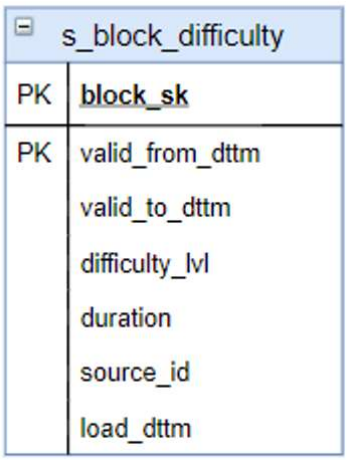


Сателлит с базовой информацией о блоке

Аналогично курсу, создадим еще один сателлит с дополнительной информацией о блоке **s_block_difficulty**. В него войдут поля:

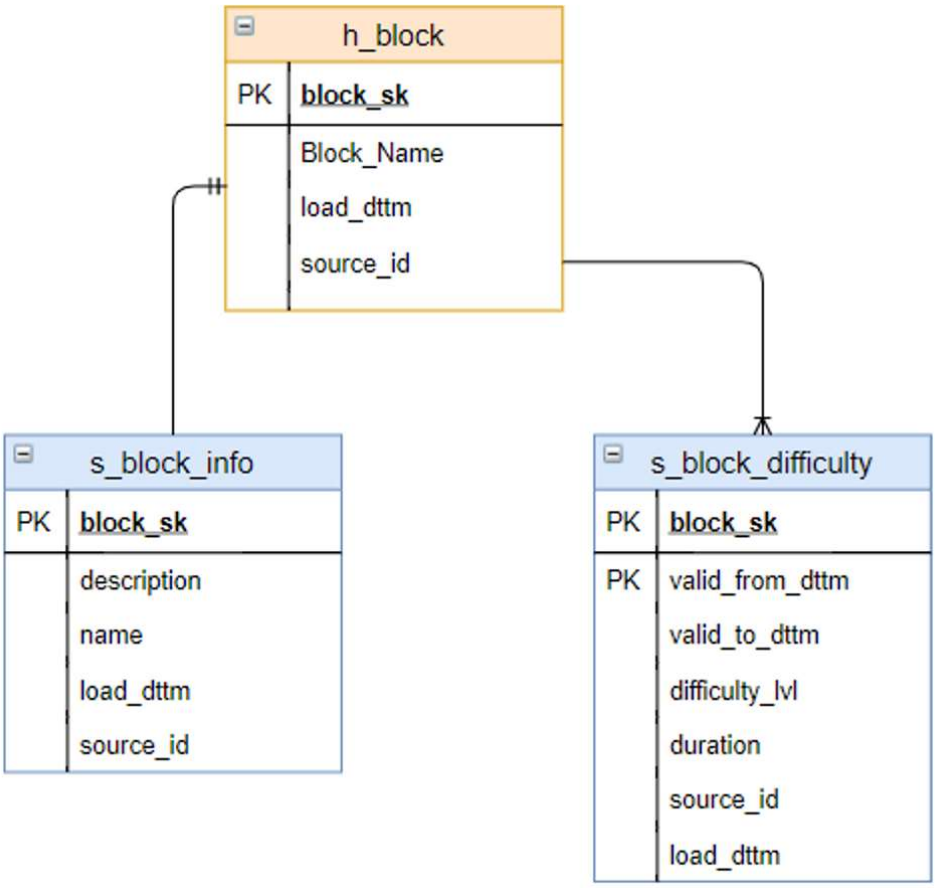
- `difficulty_lvl` - уровень сложности блока
- `duration` - продолжительность блока

Связь *{h_block, s_block_difficulty}* будет один-ко-многим.



Сателлит с дополнительной информацией о блоке

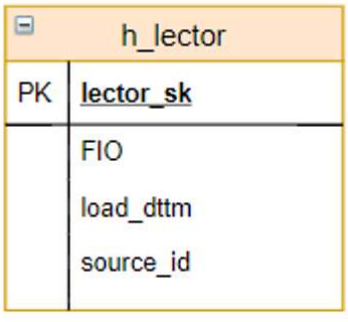
Итоговая схема связей:



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`.



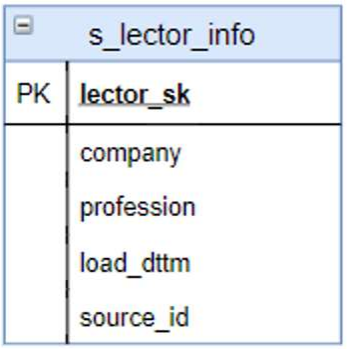
Хаб лекторов

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.

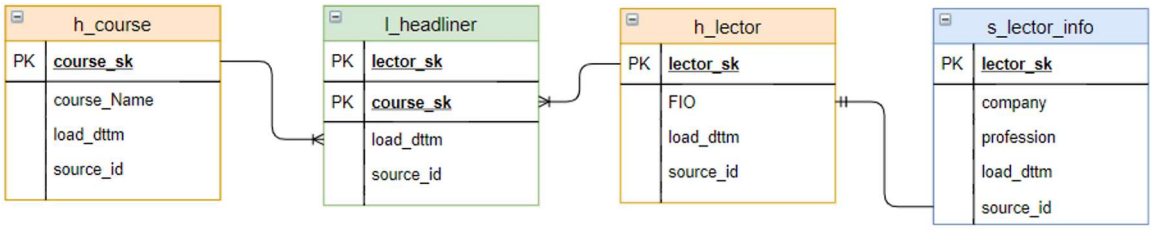


Сателлит с информацией о лекторе

Связь между лектором и курсом

Добавим новую таблицу связи между лекторами и курсами – линк **l_headliner**. Линки мы будем обозначать префиксом l_.

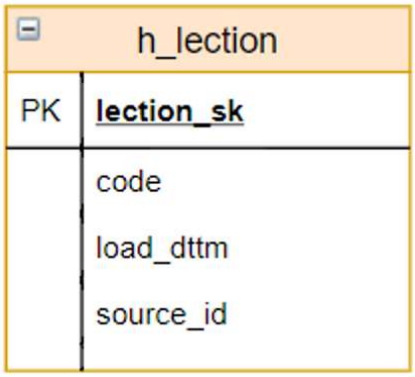
Этот линк будет содержать лишь ключи `lector_sk` и `course_sk` (ну и стандартные `load_dttm` и `source_id`). Дополнительной информации у этого линка не будет.



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`.



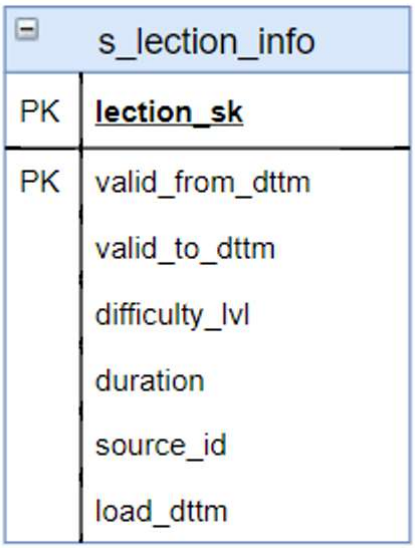
Хаб лекции

Сателлит

Создадим сателлит **s_lection_info** с информацией о лекции. В нем будут:

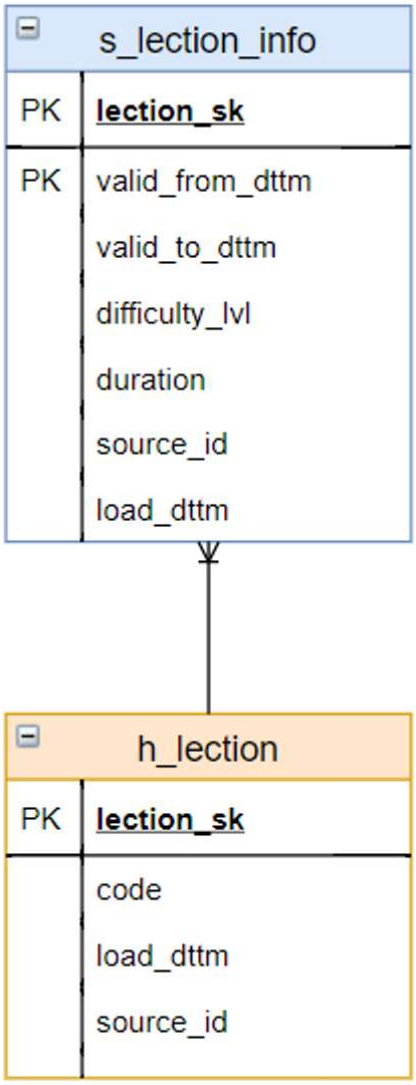
- `duration` – длительность лекции
- `difficulty_lvl` – сложность лекции

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



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

Итоговая схема

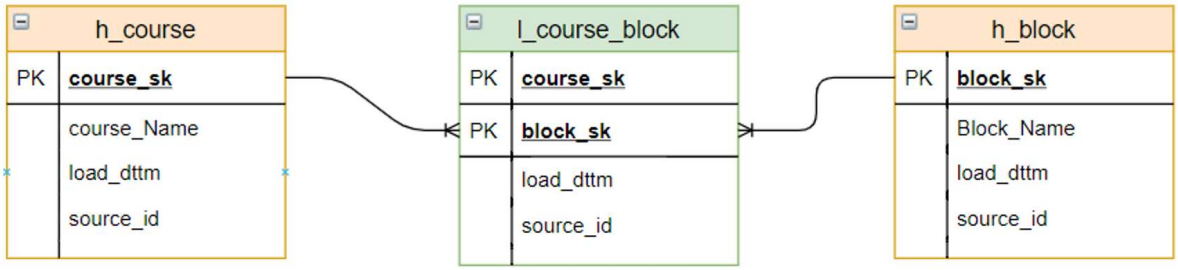


Сущность лекции в DV

> Связи между курсами, блоками, лекциями и лекторами

Связь между блоком и курсом

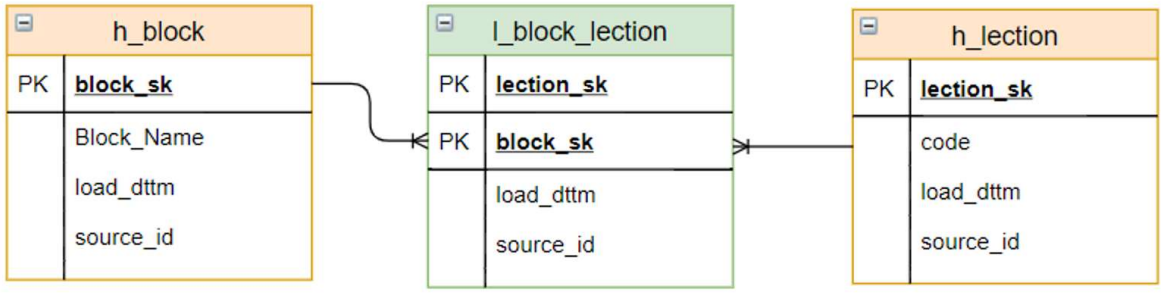
Добавим линк между блоками и курсами **l_course_block**. В нем, очевидно, будут ключи `course_sk` и `block_sk`.



Линк между курсами и блоками

Связь между блоком и лекцией

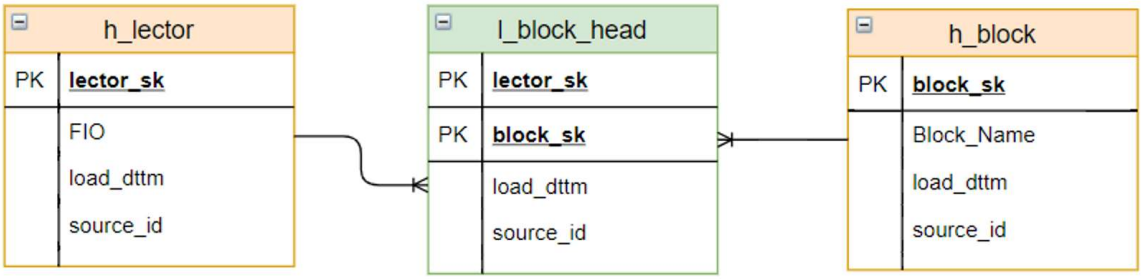
Добавим линк **l_block_lection**. Ключи – `block_sk` и `lection_sk`.



Линк между блоками и лекциями

Связь между блоком и лектором

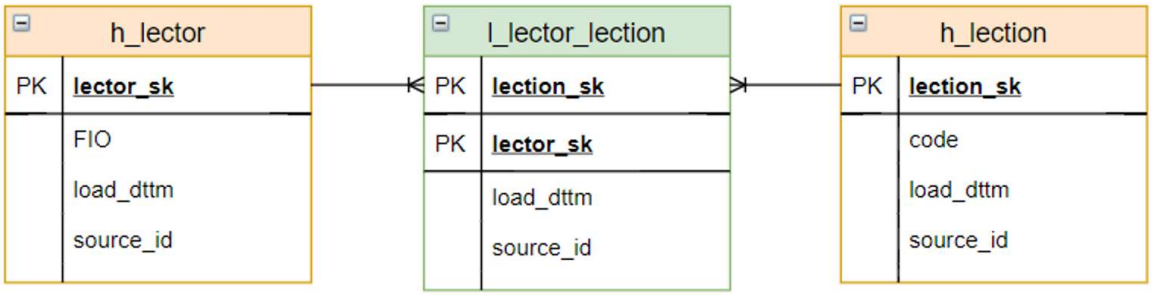
Добавим линк **l_block_head**. Ключи будут соответственно `block_sk` и `lector_sk`.



Линк между блоками и лекторами

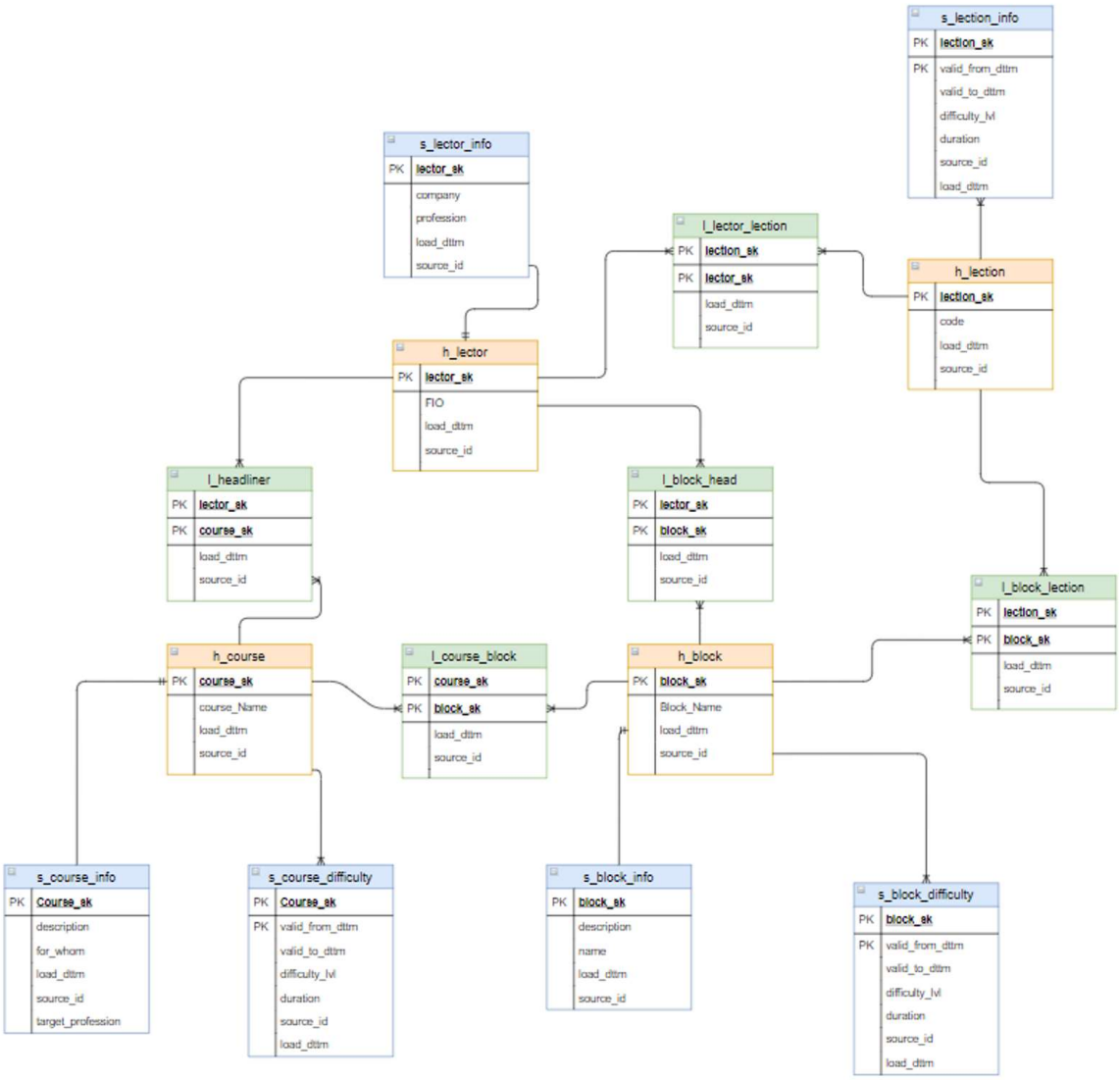
Связь между лектором и лекцией

Добавим линк **l_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

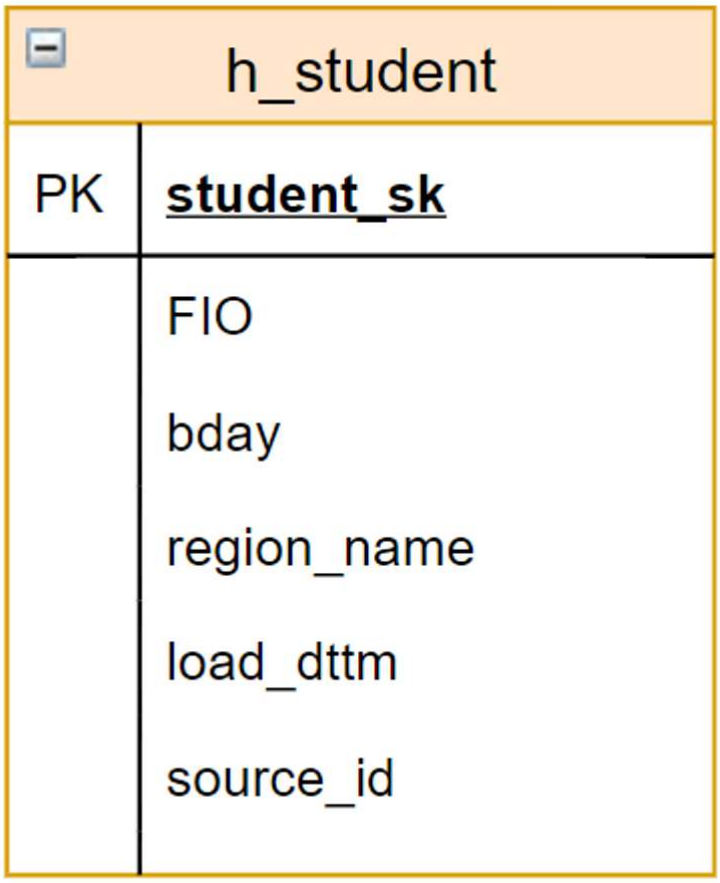
- В схеме толком нет ничего, что потребовалось бы для анализа данных (платежи, просмотры, домашние задания)

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`.



Хаб студентов

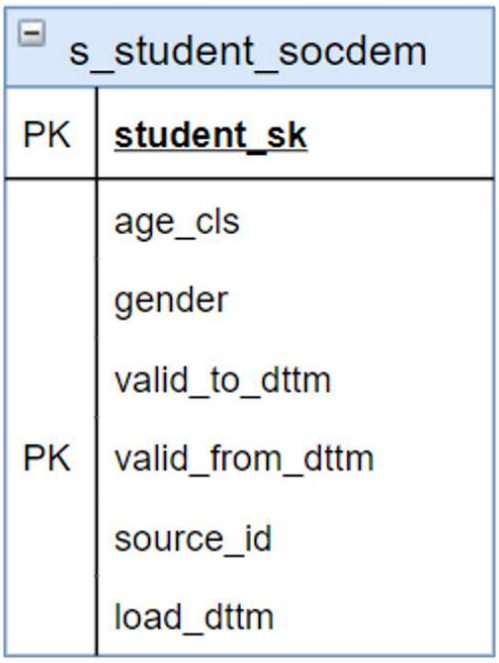
Добавим также сателлит **s_student_socdem** с нашей собственной информацией о студенте. Поля сателлита:

- `age_cls` – возраст
- `gender` – пол

Добавляем поля `valid_from_dttm` и `valid_to_dttm` для хранения истории.

Постфикс _cls обозначет, 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.



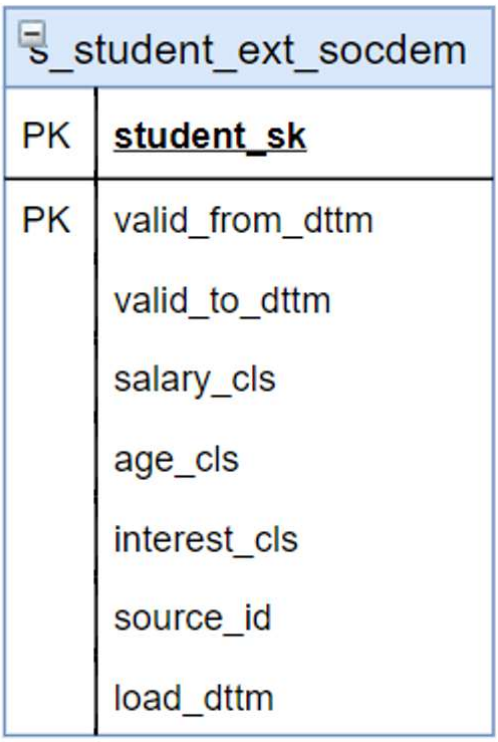
Сателлит с информацией о студенте

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.

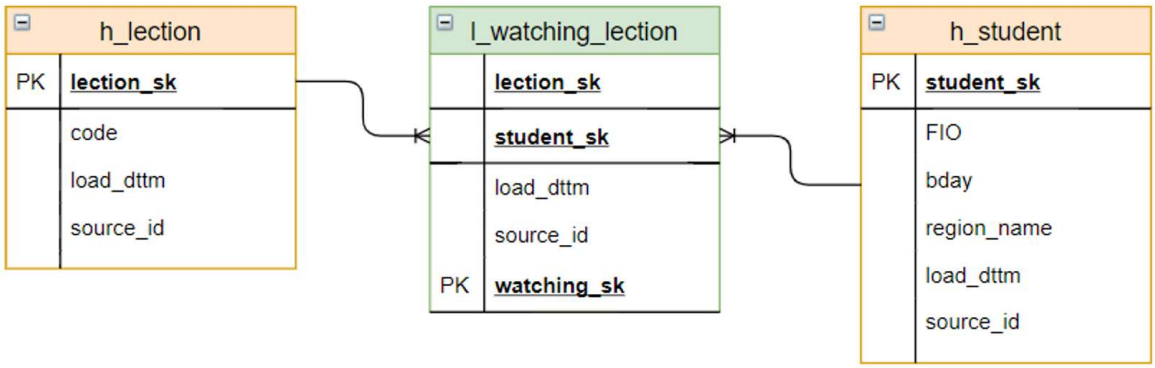


Сателлит с расширенной информацией о студенте

Информация о просмотрах лекций

Добавим линк **l_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

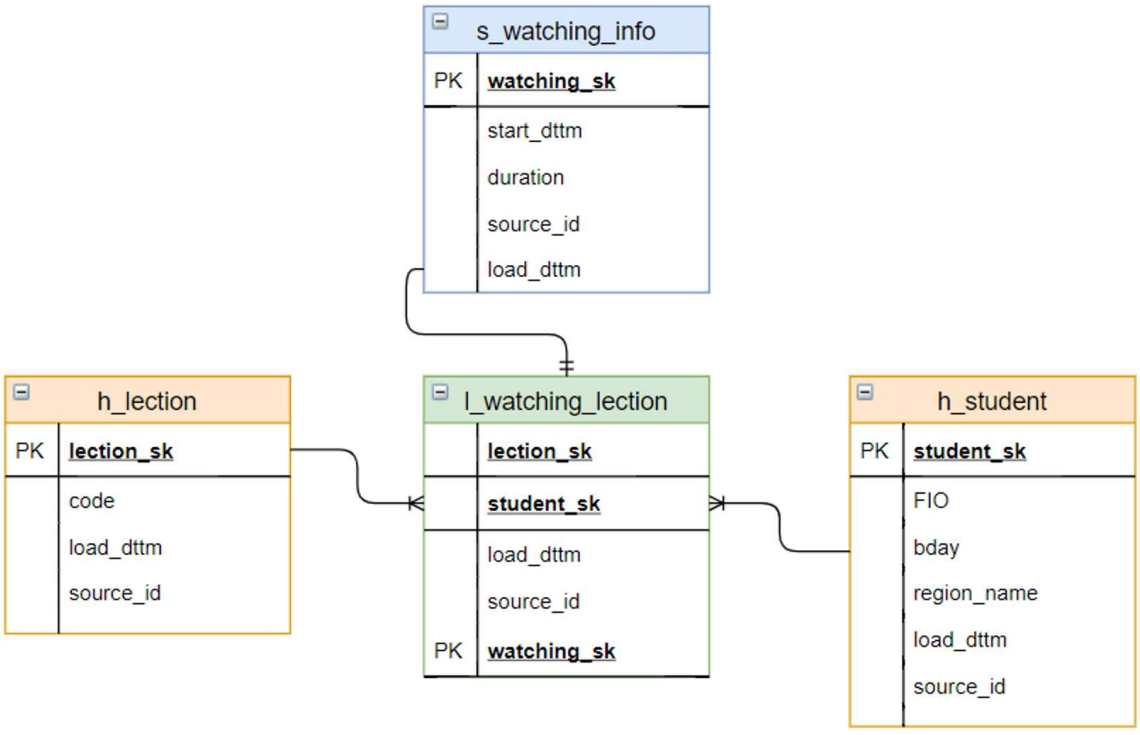
Сателлит с информацией о просмотре

Добавим сателлит **s_watching_info** с информацией о просмотре:

- `Duration` – длительность просмотра
- `Start_dttm` – Начало просмотра

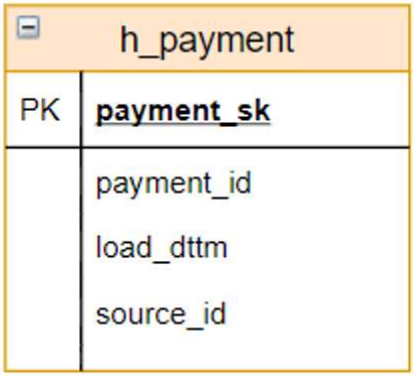
Сателлит с информацией о просмотрах лекций

Итоговая схема



> Информация о платежах

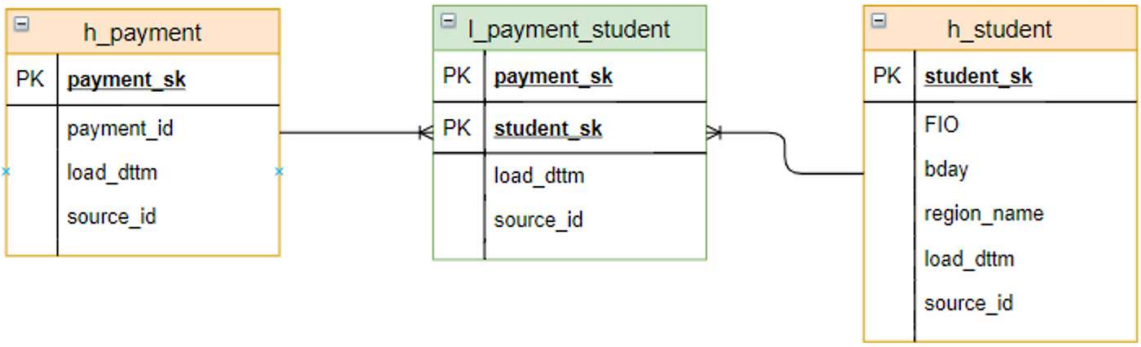
Добавим хаб **h_payment** с суррогатным ключом `payment_sk`. Бизнес ключ - `payment_id`. Также добавим стандартные `load_dttm` и `source_id`.



Хаб платежа

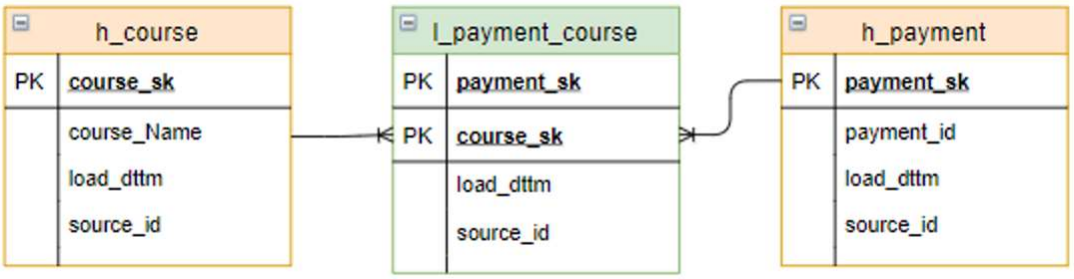
Линки

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

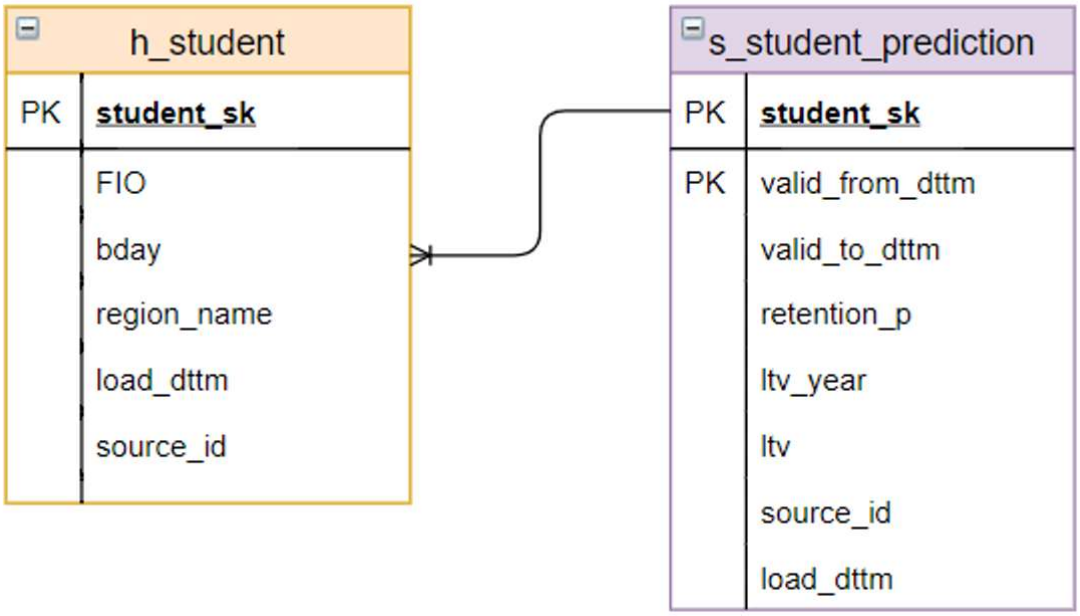
> Введение сущностей из Data Vault 2.0

Агрегированная информация по курсу

Добавим сателлит 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

PIT 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.