**Информационное моделирование:**

**1. Сущности:**

- Преподаватель

- Атрибуты: ID, Фамилия, Имя, Отчество, Должность, Список тематик

Зачем нужна должность? Если подразумевалось, что это ограничивает то, может ли преподаватель руководить аспирантами (это могут делать только кандидаты или доктора наук), то «кандидат наук» и «доктор наук» — это учёные степени. Например, кандидат наук (степень) может быть старшим преподавателем или доцентом (должность), не путать с учёным званием, которое в нашей задаче вообще не важно.

- Тематика

- Атрибуты: ID, Название тематики

- Студент

- Атрибуты: ID, Фамилия, Имя, Отчество, Группа, Список интересующих тематик

Как вы уже знаете, список интересующих тематик имеет порядок. Это нужно указать в модели, а также учесть в структуре БД (это может быть третье поле в таблице связей).

- Успеваемость

- Атрибуты: ID\_Студента, ID\_Тематики, Оценка

Непонятная сущность, особенно то, откуда берется «оценка».

- Специальность

- Атрибуты: ID\_Специальности , Название\_Специальности, Список\_Предметов

- Оценки\_Студентов

ID\_Оценки, ID\_Студента, ID\_Специальности, ID\_Предмета, Оценка, Номер семестра обучения.

Сущность предмета не описана.

**2. Связи между сущностями:**

- У преподавателя есть тематики (связь многие-ко-многим)

- Студент интересуется тематиками (связь многие-ко-многим)

- Студент имеет Успеваемость по дисциплинам (связь один-ко-многим)

- Студент - Оценки\_Студентов - один-ко-многим

- Связь "Специальность - Оценки\_Студентов" - один-ко-многим,

- Связь "Специальность - Предмет" - многие-ко-многим

В текстовом виде всё это трудно читать, потому что представлено нестандартно. Кроме того, «ID\_предмета» и т. п. относятся не к информационной модели, а уже к проектированию таблиц БД. Например, связь «многие-ко-многим» будет отдельной таблицей с ID на уровне БД, но на уровне модели это просто ромбик с линиями к сущностям. Рекомендую ER-диаграммы ([https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C](https://ru.wikipedia.org/wiki/ER-модель)). Полотнов на СУБД о них рассказывает. Если нравится писать текстом, а компьютер чтобы сам рисовал, см. <https://plantuml.com/er-diagram>. PlantUML может отрисовать и структуру таблиц БД.

Где в модели собственно результат распределения? Можно ли его будет поправить вручную, если автоматика даст не совсем хороший результат, и как это учесть в модели? Можно ли будет в системе закрепить итоговое выбранное распределение? На практике это процесс итеративный, то есть по некоторым студентам сразу будет принято решение, а некоторых раскидывают позже — кто не уверен или кто попадает к преподавателям, у которых много желающих.

**Требования к СУБД:**

1. Многопользовательская система: необходима поддержка аутентификации и авторизации пользователей (преподавателей и студентов).  
Аутентификация и авторизация, как правило, делается на уровне приложения. В СУБД лишь хранится информация о пользователях и правах. Разграничение прав доступа к таблицам и даже строкам (row-level security) на уровне СУБД существует, но не советую в это лезть — такое нужно только для систем с очень серьезными требованиями к безопасности.  
С точки зрения выбора СУБД многопользовательская система означает прежде всего сценарий, когда несколько пользователей хотят выбрать или изменить данные одновременно. Тут два аспекта: 1) конкурентность доступа, 2) конкурентность запросов.  
Конкуретность доступа — могут ли два потока одновременно писать в базу (читать, как правило, могут сколько угодно). Например, SQLite по умолчанию разрешает писать только одному потоку, хотя там есть режим синхронизации писателей. Postgres поддерживает множество писателей штатно. К этому можно добавить соображение, а будет ли интенсивный конкуретный доступ: SQLite в режиме синхронизации заставит одного из писателей подождать сколько-то миллисекунд, но проблема ли это при ожидаемых нагрузках?  
Конкурентность запросов — не поломают ли одновременно работающие с СУБД задачи друг друга. Сюда входит поддержка транзакций в СУБД, то есть, можно ли сделать несколько изменений так, чтобы одновременно работающий читатель увидел либо все из них, либо ничего из них. Важно это или нет, зависит от ожидаемых запросов. В вашем случае кажется, что нет.

2. Реляционная база данных для хранения информации о преподавателях, тематиках, студентах и их успеваемости.

3. Поддержка связей между сущностями (преподаватель-тематика, студент-тематика, студент-успеваемость).  
Если вы имеете в виду, что СУБД не должна позволять записать студенту ID несуществующего руководителя, например, то это называется контролем ссылочной целостности (referential integrity).

4. Возможность выполнения запросов для получения рекомендаций по распределению студентов.  
Разве бывает СУБД, которая не поддерживает запросы? Ну, а сами запросы вам как разработчику понадобится написать.

5. Простой и удобный интерфейс для ввода и редактирования данных через веб-приложение.   
Не имеет отношения к СУБД.

**Структура базы данных:**

1. Таблица "Преподаватели":

- ID\_Преподавателя (первичный ключ)

- Фамилия

- Имя

- Отчество

- Должность

2. Таблица "Тематики":

- ID\_Тематики (первичный ключ)

- Название\_Тематики

3. Таблица "Преподаватель\_Тематика" (связь многие-ко-многим):

- ID\_Преподавателя (внешний ключ из таблицы "Преподаватели")

- ID\_Тематики (внешний ключ из таблицы "Тематики")

4. Таблица "Студенты":

- ID\_Студента (первичный ключ)

- Фамилия

- Имя

- Отчество

- Группа

- Специальность

5. Таблица "Студент\_Тематика" (связь многие-ко-многим):

- ID\_Студента (внешний ключ из таблицы "Студенты")

- ID\_Тематики (внешний ключ из таблицы "Тематики")

6. Таблица "Специальности":

- ID\_Специальности (первичный ключ)

- Название\_Специальности

- Список\_Предметов

7. Таблица "Оценки\_Студентов":

- ID\_Оценки (первичный ключ)

- ID\_Студента (внешний ключ из таблицы "Студенты")

- ID\_Специальности (внешний ключ из таблицы "Специальности")

- ID\_Предмета (внешний ключ из таблицы "Специальности")

- Оценка

- Номер семестра обучения