Методические указания к выполнению лабораторной работы №1 по дисциплине «Управление данными»

Оглавление

Введение

В данном документе содержатся методические указания для выполнения лабораторных работ по проектированию реляционных баз данных.

В разделе «Анализ предметной области» показано, что нужно сделать после ознакомления со своим вариантом прежде, чем приступить к концептуальному и логическому проектированию.

В разделе «Концептуальное проектирование» по данной теме.

1 Анализ предметной области

В данном разделе изложено несколько возможных способов анализа предметной области. Анализ предметной области в рамках лабораторных работ необходим для того, чтобы студент смог погрузиться в заданную тему: ответить на вопросы «Как это работает?», «Какие объекты есть в предметной области и как они связаны друг с другом?», «Какие данные необходимо хранить в базе данных?».

Описание вариантов, которое даётся на лабораторных работах, имеет такие недостатки, как: разреженность информации, неполнота описания, допущения, противоречия, неточность. Таким образом, можно сделать вывод о том, что изначальная задача не является чётко сформулированной. И для более качественного выполнения задания нужно потратить время и силы на анализ предметной области.

По желанию студента анализ может производиться с разной степенью детализации. Если речь идёт о начале изучения проектирования баз данных, то обычно можно ограничиться расширенным текстовым описанием предметной области и выделением бизнес - правил.

В данном разделе внимание уделяется расширению описания заданного варианта и формированию бизнес-правил.

1.1 Расширение описания варианта

Под расширенным описанием предметной области понимается дополнение исходного описания. Необходимую для дополнения информацию о той или иной предметной области проще всего получать из интернета и литературы. Например, благодаря изучению сайтов, посвящённых музыкальным магазинам, можно примерно понять, каким образом устроена работа таких магазинов и описать это в текстовом варианте или в виде простых схем взаимодействия между элементами предметной области. Дополнительная информация также частично позволит понять, какие данные нужно хранить в базе данных, и с чем вообще будет работать будущее приложение.

На данном этапе можно самостоятельно детализировать задачу, то есть придумать, как будет что-либо работать в заданной предметной области. Однако, в этом случае выдуманная логика работы не должна сильно расходиться с реальностью.

Пример исходного описания темы:

«Необходимо разработать информационную систему для спортзала. База данных должна содержать информацию об инвентаре (название, марка, модель, предназначение и др.), тренерах (ФИО, паспортные данные, контактная информация и т.д.), посетителях (ФИО, информация об абонементе, наличие в группах и т.д.), индивидуальных и групповых занятиях, услугах (название, тип услуги, стоимость), товарах (название, тип товара, стоимость), заказах на товар и прочую необходимую информацию. В базе данных необходимо предусмотреть хранение информации о различных типах абонементов (например, разного срока действия, с

разными привилегиями и уровнями доступа и т.д.). Также необходимо предусмотреть скидочные программы для студентов и корпоративных клиентов. Необходимо хранить фотографии клиентов и сотрудников.»

Как видно из описания, работа спортзала требует уточнения. Неясны все связи между элементами и, банально, подробности оказания услуг, скидочной системы и т.д.

Что нужно сделать, чтобы детализировать описание?

Пожалуй, самым важным здесь является выделение элементов и частей описания/предметной области, работа которых неясна/пояснена слишком поверхностно или размыто.

Лучше всего на данном этапе задавать себе вопросы: «А что мне непонятно?», «Что можно дополнить?», «Есть ли здесь противоречия/неточности?».

Например, выше указано, что необходимо предусмотреть скидочные программы для студентов и корпоративных клиентов. Но нет никакой информации о том, какой размер скидки должен быть, как эта программа должна работать, кто вообще является корпоративным клиентом (уж кто является студентом ясно). Также отсутствует информация о том, какие абонементы можно приобрести.

Что можно сделать, чтобы устранить недостаток этой информации?

Как уже было сказано выше, можно изучить работу реальных спортзалов, фитнес-клубов и т.п. И на основании изученной информации и собственных представлений дополнить тему.

Пример дополнения темы:

Было	Стало
Необходимо предусмотреть скидочные программы для студентов и корпоративных клиентов.	Необходимо предусмотреть скидочные программы для студентов и корпоративных клиентов. Студенческая скидка составляет 15% от стоимости абонемента и не действует на «VIP-абонементы» и платные услуги. Для получения студенческой скидки нужно предъявить действующий студенческий билет сотруднику-оператору спортзала. Его нужно предъявлять при продлении абонемента. Корпоративным клиентом считается сотрудник организации-партнёра. При этом партнёры

	должны сами предоставлять список сотрудников, которые будут посещать зал. Корпоративная скидка составляет 10% от стоимости абонемента и не действует на «VIP-абонементы» и платные услуги.

Можно заметить, что теперь вопросов о работе спортзала стало меньше. При этом нет очевидных выходов за рамки варианта. Главное в этом деле - не закопаться. При этом данное описание затронуло другую тему - типы абонементов, которые также обязательно следует однозначно определить.

Таким образом, расширенное описание предметной области снижает риск запутаться в теме и позволяет рассмотреть её с разных сторон. Всё это также снижает вероятность возникновения в дальнейшем проектировании логических ошибок, которые могут привести к тому, что база данных будет противоречить действительности и реальной работе спортзалов. Грубо говоря, чем больше студент знает о теме, с которой работает, тем проще будет в дальнейшем. Расширение варианта также поможет в разработке технического задания.

1.2 Формирование бизнес-правил

Другим распространённым способом проектирования информационных систем и баз данных является выделение бизнес-правил.

Бизнес-правила представляют собой специализированный вид логики, описывающей ограничения на образ действий, которые система или люди должны учитывать в своем поведении. Эти правила определяются целым рядом факторов, включая директивы распорядительных органов, промышленные стандарты, деловую хватку и простой здравый смысл. Нередко они изменяются от страны к стране, от отрасли к отрасли, и даже от бизнеса к бизнесу. В качестве примера бизнес-правила в банковском деле можно привести закон, по которому о любой сделке, превышающей сумму 10 000 долларов наличными, государство должно ставиться в известность. Несомненно, данное бизнес-правило необходимо принимать во внимание при создании банковской системы вложения/снятия наличных денег.

Бизнес-правила существуют на разных уровнях. Некоторые из них оказывают влияние на всю систему, и многие системы, на самом деле, целиком создаются лишь для того, чтобы ввести в действие бизнес-правила. Бизнес-правила также могут значительно различаться по размерам области влияния. Но, несмотря на это, все бизнес-правила имеют одно общее свойство: они управляют некоторой составляющей бизнеса. По определению, бизнес-правило есть ограничение, применяемое по отношению к человеку, бизнесу, составляющей бизнеса или действию.

Бизнес-правила в БД представляют собой условия того, что данные соответствуют предметной области.

Бизнес-правила можно разделить на элементарные и расширенные.

Элементарные правила ограничивают значения конкретного атрибута или набора атрибутов.

Расширенные правила выражаются в виде некоторой зависимости между атрибутами.

Ограничения могут быть статическими и динамическими.

Статические ограничения должны выполняться для каждого состояния базы данных.

Динамические ограничения проявляются при переходе базы данных из одного состояния в другое (например, при повышении заработной платы новое значение должно быть выше старого).

Ограничения могут накладываться на базу данных:

- В концептуальной, логической и физической модели;
- В запросах на выборку данных из таблиц;
- В хранимых процедурах, функциях и триггерах.

В общем случае правила делятся на три типа:

- Правила вывода
- Правила ограничения
- Инвариантные правила

Правила вывода преобразуют полученную информацию в возвращаемые значения. Например, скидки на товары можно вычислить с помощью специального алгоритма, учитывающего размер заказа, рекламную поддержку и значимость клиента, которому будут поставляться товары. Правила этого типа допускают изменения, и поэтому, прежде чем с ними можно будет работать, их требуется выделить.

Правила ограничения проверяют значения транзакций или операций на непротиворечивость. Например, чтобы заказчик смог оформить доставку и получить товар, индекс, указанный в адресе доставки, должен соответствовать индексу, указанному в документах на отправку товара.

Инвариантные правила проверяют множественные изменения и обеспечивают непротиворечивость итоговых результатов. Например, баланс сберегательного счёта должен быть равен предыдущему балансу + сумма прихода или - сумма расхода.

Часть бизнес-правил может быть слишком очевидной и простой для восприятия. Тем не менее не следует упускать детали из виду, иначе во время разработки и эксплуатации базы данных и приложения, работающего с ним, не избежать логических ошибок.

Примеры бизнес-правил:

1. Каждый автор может написать несколько книг;
2. Одна книга может быть написана несколькими авторами;
3. Посетитель библиотеки может взять несколько книг на руки (не более 3).
4. (на основании примера про «Спортзал») Любой абонемент перестаёт быть действительным на следующий день после последнего дня прежнего срока действия. В качестве часового пояса принимается

GMT+3. Идентификация клиента с недействующим абонементом невозможна.

5. (на основании примера про «Спортзал») Любой абонемент может быть продлён, аннулирован, преобразован в «VIP» и наоборот.
6. (на основании примера про «Спортзал») В одно и то же время не может быть 2 групповых занятия в одном и той же секции «Спортзала».

1.3 Пример разбора варианта

1.3.1 Исходное описание варианта

«Необходимо разработать информационную систему для спортзала. База данных должна содержать информацию об инвентаре (название, марка, модель, предназначение и др.), тренерах (ФИО, паспортные данные, контактная информация и т.д.), посетителях (ФИО, информация об абонементе, наличие в группах и т.д.), индивидуальных и групповых занятиях, услугах (название, тип услуги, стоимость), товарах (название, тип товара, стоимость), заказах на товар и прочую необходимую информацию. В базе данных необходимо предусмотреть хранение информации о различных типах абонементов (например, разного срока действия, с разными привилегиями и уровнями доступа и т.д.). Также необходимо предусмотреть скидочные программы для студентов и корпоративных клиентов. Необходимо хранить фотографии клиентов и сотрудников.»

Что здесь не так?

Информация разрозненная, размытая.

Детализация низкая.

То есть задача не является чётко сформулированной.

Что надо делать?

Расширить описание, додумав логику работы данной информационной системы. Информацию также можно и даже стоит брать из различных источников (например, изучить веб-сайты существующих спортзалов).

Далее вся эта информация будет представлены в функциональных, поведенческих и прочих моделях.

1.3.2 Пример расширения варианта

Спортзал с некоторым названием имеет единственный филиал в городе и всей стране.

Спортзал включает в себя несколько секций, оборудованных тренажёрами, детскую комнату и бассейн.

Каждая секция имеет название в виде цвета: «Желтая», «Красная», «Зеленая» и «Синяя» (бассейн) и т.д.

Посещение спортзала осуществляется по абонементам. Есть несколько типов абонементов: разовое посещение, абонемент на месяц, абонемент на полгода, абонемент на год.

При посещении спортзала клиент должен провести своим абонементом у турникета.

У каждого абонемента своя стоимость:

- абонемент на разовое посещение (бесплатный, если это гостевое посещение по предварительной записи/350 рублей, если это не гостевое посещение);
- абонемент на месяц (7000 рублей);
- абонемент на полгода (22000 рублей);
- абонемент на год (30000 рублей);
- VIP-абонемент на месяц/полгода/год (9000, 25000, 35000 рублей).

VIP - абонементы отличаются от обычных тем, что позволяют безлимитно посещать спортзал и все его секции, в т.ч. бассейн. Обычные абонементы позволяют посещать все секции, кроме бассейна. При этом для обычных абонементов количество занятий в месяц/полгода/год ограничено 15/50/100 занятиями соответственно.

Для студентов и корпоративных клиентов действуют скидки:

- 15% для студентов;
- 10% для корпоративных клиентов.

Каждый абонемент, кроме абонемента на разовое посещение, является именным. Для получения студенческой скидки необходимо очно предоставить действующий студенческий билет. Корпоративная скидка предоставляется сотрудникам компаний - партнёров, включённых в список от этой компании. Списки сотрудников для получения скидки присылает сама компания. На корпоративном абонементе помимо ФИО и № абонемента отражается также и название компании - партнёра.

Студенческие и корпоративные скидки не действуют на VIP-абонементы.

Абонемент сроком на полгода/год можно вернуть, если с момента его приобретения не прошло больше половины срока действия абонемента. При этом будет осуществлён возврат денежных средств после перерасчёта.

В спортзале есть несколько типов занятий:

- индивидуальные;
- индивидуальные с тренером;
- групповые с тренером;
- групповые детские с тренером.

Групповые тренировки также имеют несколько типов, например:

- Йога;
- Суставная гимнастика;
- BODY PUMP;
- STRETCHING;
- TAE-BO;
- BOSU;
- детские занятия в бассейне;
- PILATES и др.

При этом некоторые типы групповых занятий могут быть платными, несмотря на наличие абонемента. Все групповые занятия осуществляются

по расписанию. В зависимости от типа группового занятия количество мест может изменяться.

Спортзал имеет собственный магазин, в ассортимент которого включаются подарочные абонементы на месяц/полгода/год; спортивное питание, одежда, обувь, инвентарь.

В спортзале есть несколько должностей у сотрудников, например, тренеры, операторы (стоят на входе, регистрируют новых клиентов), продавцы (осуществляют контроль за складом и осуществляют продажи товаров). При этом у тренеров есть разная специализация (например, массажист).

Спортзал закупает товары для своего магазина у поставщиков.

Выше была кратко описана логика работы спортзала. Это описание отчасти структурировало информацию о предметной области и дало некоторую детализацию. Также это задало ряд ограничений темы, о которых речь пойдёт ниже. Тем не менее при пристальном изучении детального описания всё равно остаётся много вопросов.

1.3.3 Выделение бизнес-правил

Для данной темы можно выделить следующие бизнес-правила.

1. Посещать спортзал могут только клиенты, достигшие совершеннолетнего возраста.

Таким образом нельзя оформить абонемент на человека, младше 18 лет.

2. Студенческая скидка рассчитывается только при предъявлении действительного студенческого билета во время оформления абонемента. Студенческая скидка рассчитывается согласно следующей логике: если это не "VIP"-абонемент, то сумма абонемента = "Сумма выбранного абонемента" - "Сумма выбранного абонемента" * 0,15

3. Корпоративная скидка рассчитывается только для сотрудников компаний-партнёров, включённых в список. Список таких сотрудников спортзал получает от партнёров. Корпоративная скидка рассчитывается согласно следующей логике: если клиент присутствует в списке компании-партнёра, и если это не "VIP"-абонемент, то сумма абонемента = "Сумма выбранного абонемента" - "Сумма выбранного абонемента" * 0,1

4. До момента первого посещения спортзала, если это не разовое посещение, клиент должен предоставить о себе следующие данные: ФИО, паспортные данные, номер мобильного телефона, фотографию.

5. Идентификация клиента осуществляется через его абонемент, который клиент должен предъявлять при каждом посещении спортзала.

6. Абонемент перестаёт быть действительным на следующий день после последнего дня действия абонемента. Часовым поясом по умолчанию считается GMT+3. Идентификация клиента по данному абонементу невозможна до момента продления абонемента.

7. Любой абонемент может быть продлён, аннулирован, преобразован в "VIP" абонемент и обратно.

8. В одно и то же время не может проходить 2 групповых занятия в одной секции или бассейне.

9. Для попадания на любое групповое занятие необходима предварительная запись хотя бы за 30 минут до начала занятия. Если период < 30 минут, то запись на это занятие считается невозможным.

10. Стоимости услуг персональных тренировок не являются фиксированными.

11. Существует возможность возврата товара в течение 14 дней, если этот товар не относится к типу спортивного белья, подарочного абонемента или спортпита.

12. Для посещения зала необходимо соответствующее врачебное заключение, которое нужно предоставить до начала тренировок. Идентификация клиента с действующим абонементом и отсутствующим врачебным заключением не должна происходить.

13. Стоимость закупки товара у поставщика не может превышать стоимость его продажи клиенту.

14. Покупки во внутреннем магазине могут осуществлять не постоянные клиенты спортзала, т.е. у них может отсутствовать абонемент. Абонемент даёт право посещать занятия.

Выше представлен всего лишь небольшой перечень ограничений, накладываемых на предметную область, разрабатываемую базу данных и приложение. Часть ограничений будет реализовываться на уровне базы данных, а часть будет реализована в логике приложения, работающего с этой БД.

2 Концептуальное проектирование базы данных

Концептуальное проектирование — это создание схемы БД, включающей определение важнейших сущностей (таблиц) и связей между ними, но не зависящей от модели БД (иерархической, сетевой, реляционной и т.д.) и физической реализации (особенностей реальной СУБД).

Целью концептуального проектирования является создание концептуальной схемы данных на основе представлений о предметной области.

Концептуальная схема — это описание основных сущностей (таблиц) и связей между ними.

Концептуальное проектирование делится на 3 этапа:

1. Выделение сущностей (будущих таблиц)
2. Выделение атрибутов (будущих столбцов таблиц)
3. Определение связей между сущностями.

Перед тем, как пойти дальше, нужно ввести ряд определений.

Сущность — это физическое представление логической группировки данных. Может представлять как реальные, так и абстрактные объекты. В реляционной модели данных под сущностью понимается таблица.

Атрибут — это характеристика сущности, которая каким-либо образом её описывает. В реляционной модели данных атрибут = столбец таблицы.

Связь — это отношение между сущностями, графически отображаемая ассоциация (в виде стрелки или линии) между двумя сущностями.

Первичный ключ (Primary Key, PK) — это ключевой уникальный атрибут (в таком случае ключ будет называться простым) или несколько атрибутов сразу (в таком случае ключ будет называться составным), который идентифицирует текущую запись в таблице (экземпляр сущности).

Под уникальностью значений одного атрибута или набора атрибутов понимается то, что для каждой новой строки в таблице (для каждого нового экземпляра сущности) значение первичного ключа будет уникальным.

Примеры первичных ключей: «Табельный номер сотрудника» (у каждого сотрудника есть свой отличный от других табельный номер), «Код товара» (у каждого товара есть свой уникальный идентификатор), «Номер абонемента» и т.д.

Первичные ключи делятся на:

а) по составу:

- простые (состоят из одного атрибута). Встречаются наиболее часто.
- составные (состоят из нескольких атрибутов).

б) по естественности:

- естественные (реально существующие атрибуты сущностей (например, № зачётной книжки, табельный номер и т.д.).
- суррогатные (искусственно введённые атрибуты, обычно являющиеся столбцами с числовым типом данных). Чаще всего используются именно искусственные ключи, даже если можно выделить естественные.

Если на каком-либо этапе проектирования нельзя однозначно определить, что будет являться первичным ключом, то вводится так называемый «Потенциальный ключ».

Потенциальный ключ (CK) — это столбец или набор столбцов, удовлетворяющих следующим условиям:

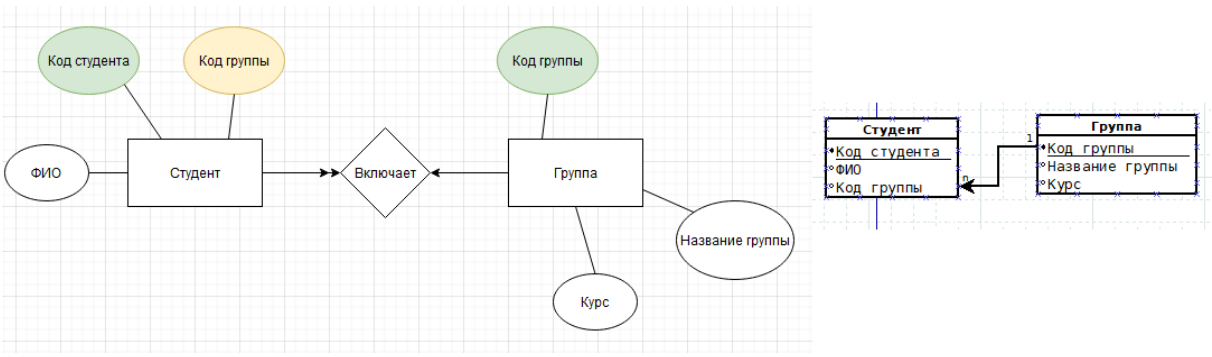
а) неприводимость - столбец или набор столбцов содержит минимально необходимый набор атрибутов для того, чтобы можно было идентифицировать запись в таблице;

б) уникальность - значение потенциального ключа должно быть уникальным всегда вне зависимости от изменений в строке;

в) наличие значения - столбец или набор столбцов потенциального ключа всегда должен быть заполнен.

Внешний ключ (Foreign key, FK) — это столбец в одной таблице, значение которого совпадает со значением первичного ключа в другой таблице. Именно через внешний ключ связываются 2 таблицы (сущности).

Пример связи сущностей в концептуальной и логической схеме:



//В данном материале зелёным цветом будут выделяться первичные ключи, а жёлтым - внешние ключи. Это сделано для удобства.

Ещё один пример изображения связи таблиц по паре значений «первичный-внешний ключ»:

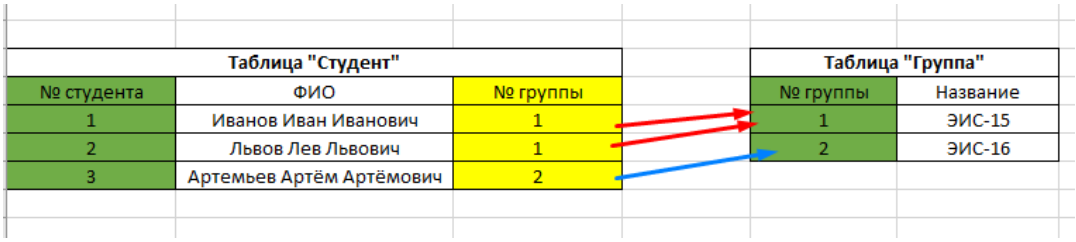


Таблица "Студент"		
№ студента	ФИО	№ группы
1	Иванов Иван Иванович	1
2	Львов Лев Львович	1
3	Артемьев Артём Артёмович	2

Таблица "Группа"	
№ группы	Название
1	ЭИС-15
2	ЭИС-16

В таблице «Студент» внешним ключом является столбец «№ группы». Из него видно, что Иванов Иван и Львов Лев учатся в одной группе с №1. В таблице «Группа» группой с №1 является группа «ЦИС-15». Таким образом, Иван и Лев учатся в группе ЦИС-15.

Здесь таблица «Студент» является зависимой от таблицы «Группа», потому что в ней есть внешний ключ. А таблица «Группа» является независимой, так как в ней внешних ключей, ссылающихся на другие таблицы, нет.

Забегая вперёд, можно отметить, что между таблицей «Группа» и «Студент» существует отношение «один-ко-многим» (то есть в одной группе учится много студентов, а один студент может учиться только в одной группе, таким образом одной записи из таблицы «Группа» соответствует множество записей в таблице «Студент»).

2.1 Сущности

На данном этапе на основании исходного описания предметной области и описания, получившегося в результате анализа, выделяются главные элементы данной области. На этих элементах, так сказать, «держится» вся работа предметной области. То есть эти элементы в основном и участвуют во всех внутренних процессах рассматриваемой темы.

Примеры:

Для библиотеки такими объектами можно считать книгу, посетителя библиотеки, библиотекарей.

Для кафе со сладостями такими объектами можно считать ассортимент (т.е. товар), сотрудников магазина и т.д.

Для кредитной организации такими объектами можно считать клиента, сотрудника банка, кредитные программы и т.д.

Для базы данных «Спортзал» такими объектами можно считать посетителей, сотрудников, товары, занятия и т.д.

По указанным выше примерам можно выделить следующие сущности:

Для библиотеки можно выделить сущности «Книга», «Читатель», «Сотрудник».

Для кафе можно выделить сущности «Товар», «Сотрудник».

Для кредитной организации можно выделить сущности «Клиент», «Сотрудник», «Кредит».

Для базы данных «Спортзал» можно выделить сущности «Клиент», «Сотрудник», «Абонемент».

Каждая сущность включает в себя:

- Атрибуты (столбцы таблицы), т.е. характеристики, описывающие данные.
- Экземпляры сущности (строки таблицы), т.е. данные.

Сущности можно классифицировать.

а) По степени независимости:

- зависимые - в таких сущностях (т.е. таблицах) есть информация, значение которой зависит от данных в других таблицах. Примеры зависимых сущностей: «Кредит» в кредитной организации (он не может существовать без клиента, который берёт кредит), «Товар» в магазине (в случае, если есть множество типов товаров, то типы товаров можно выделить в отдельную сущность).
- независимые - в таких сущностях (т.е. таблицах) нет информации, зависимой от данных в других сущностях. Примеры независимых сущностей: «Читатели» в библиотеки, «Клиенты» в кредитной организации, спортзале и т.д.

б) По смыслу и содержанию:

- Стержневые сущности;
- Справочные сущности;
- Характеристические сущности;
- Ассоциативные сущности.

Стержневая (корневая) сущность - сущность, которая обычно является одним из самых важных объектов в предметной области (например, товар в магазине, книга в библиотеке, студент в университете). Могут быть независимыми и зависимыми (чаще всего).

Справочная (кодовая) сущность (справочник) - обычно независимая сущность, которая хранит в себе перечень значений какой-либо характеристики реального или абстрактного объекта. Например, такой сущностью может являться список кредитных программ в банке, список типов документов, необходимых для подачи с целью поступления в вуз.

Характеристическая сущность (характеристика) - некая реальная характеристика объекта, которую целесообразно выделить в отдельную сущность (таблицу) в связи с наличием большого числа атрибутов. Пример такой сущности: «Паспортные данные» - такую сущность можно выделить в отдельную таблицу, если требуется хранить не только серию и номер, а и прочую информацию, которая занимает несколько столбцов.

Ассоциативная сущность - связующая сущность между двумя или большим числом таблиц, необходимая для устранения связи «многие-ко-многим».

2.2 Атрибуты

У каждой сущности есть какой-то набор данных, которым она характеризуется. То есть каждая сущность содержит некоторый набор атрибутов.

Примеры атрибутов: у сущности «Книга» могут быть атрибуты «Наименование», «Авторы», «Год издания» и т.д.; у сущности «Кредит» могут быть атрибуты «Наименование кредитной программы», «Срок кредита», «Одобренная сумма» и т.д.; у сущности «Клиент» могут быть атрибуты «ФИО», «Дата рождения», «Мобильный телефон», «Адрес для доставки» и т.д.

Атрибуты обычно именуются именами существительными в единственном числе, но на начальных этапах проектирования в сущностях могут быть атрибуты, именуемые множественным числом и хранящие в себе несколько значений.

Атрибуты также имеют собственную классификацию:
а) По количеству хранимых значений:

- Однозначный атрибут (атомарный) - в столбце таблицы хранится только 1 значение из числа возможных. Пример: «Название книги», у книги оно одно.
- Многозначный атрибут (неатомарный) - в столбце таблицы хранится >1 значения из числа возможных. Пример: «Авторы», у книги их может быть несколько.

Код книги	Название книги	Авторы книги
1	Название1	Автор1, Автор2
2	Название2	Автор3, Автор7, Автор9

Однозначный атрибут

Многозначный атрибут

б) По обязательности:

- Обязательный атрибут - столбец обязательно должен быть заполнен при добавлении в таблицу новой строки. Пример: «Имя клиента».
- Необязательный атрибут - столбец, который может быть, не заполнен при добавлении в таблицу новой строки. Пример: «Адрес электронной почты» у посетителя спортзала.

в) По вхождению в состав первичного ключа:

- Ключевой атрибут - столбец является первичным ключом или входит в состав составного первичного ключа (об этом далее).
- Неключевой атрибут - столбец не является простым первичным ключом и не входит в состав составного первичного ключа.

г) По составу:

- Простой - состоит из 1 столбца.
- Составной - состоит из нескольких столбцов.

2.3 Связи между сущностями

Связь — это отношение между сущностями, графически отображаемая ассоциация (в виде стрелки или линии) между двумя сущностями.

Сущности связываются между собой с помощью специальных атрибутов, которые называются «Первичный ключ» и «Внешний ключ». Эти понятия были разобраны в начале главы.

Существует 3 основных типа связей:

1. «Один-к-одному».

Одному экземпляру одной сущности соответствует не более одного экземпляра другой сущности. Т.е. одной строке из первой таблицы соответствует либо 0, либо 1 строка во второй таблице. Обычно при такой связи в одной из таблиц используется такой же первичный ключ, как и в другой таблице. При этом в зависимой таблице этот первичный ключ будет ещё и внешним.

Графически в концептуальной схеме это можно представить следующим образом:

В реляционном, т.е. в табличном представлении это будет выглядеть так:

Таблица Клиент				Таблица паспорт			
Код клиента	**Код паспорта**	**ФИО**		**Код паспорта**	**Серия**	**Номер**	**Дата выдачи**
1	1	Иванов Иван Иванович	→	1	5256	565656	18.02.1998
2	2	Петров Пётр Петрович	→	2	5254	254589	12.10.2023

Каждый клиент имеет только один действующий паспорт и каждый паспорт принадлежит только одному человеку.

2. «Один-ко-многим».

Одному экземпляру одной сущности соответствует более одного экземпляра другой сущности. Т.е. одной строке из первой таблицы соответствует несколько строк из второй таблицы. При этом для каких-то строк из первой таблицы во второй таблице может соответствовать только 1 строка. Но т.к. другим строкам первой таблицы соответствует несколько строк второй таблицы, то связь «один-ко-многим» остаётся актуальной.

Графически в концептуальной схеме это можно представить следующим образом:

В табличном представлении это будет выглядеть так:

Таблица "Клиент"			Таблица "Заказ"		
№	ФИО		№ заказа	Дата	№ клиента
1	Иванов Иван Иванович		1	20.06.2020	1
2	Петров Пётр Петрович		2	24.06.2020	1
			3	27.07.2020	2

Выше клиенту с номером 1 соответствует 2 заказа.

3. «Многие-ко-многим».

Нескольким экземплярам одной сущности соответствует несколько экземпляров другой сущности. Т.е. нескольким строкам одной таблицы соответствует несколько строк другой таблицы.

В графическом представлении это выглядит так:

В табличном представлении это будет выглядеть так:

Таблица "Книга"			Таблица "Автор"		
№	Название	№ автора	№	ФИО	№ книги
1	Двенадцать стульев	1	1	Евгений Петров	1
2	Двенадцать стульев	2	2	Илья Ильф	1
3	Удушье	3	3	Чак Паланик	3
4	Бойцовский клуб	3	4	Чак Паланик	4

Какие проблемы здесь можно заметить?

И в таблице с книгами, и в таблице с авторами дублируются данные. Тем более по смыслу записи №1 и 2 в таблице «Книга» не отличаются друг от друга, потому что это одна и та же книга. Конечно, в столбце № автора можно просто перечислить № причастных к написанию авторов. Но когда речь идёт о работе с таблицами, то так делать нежелательно, т.е. в одном столбце таблицы не следует хранить перечисление, т.е. столбец (атрибут) не должен быть многозначным, а должен быть атомарным. Перечисления в столбцах создают трудности при обработке данных. Если с числами работать ещё как-то можно, то со строками проблем куда больше (например, если атрибут «Паспортные данные» не разбивать на части, а хранить все данные о паспорте в одном столбце).

Однако в концептуальной модели наличие многозначных атрибутов (столбцов) допускается, но принято избавляться от связи «многие-ко-многим» между сущностями (таблицами).

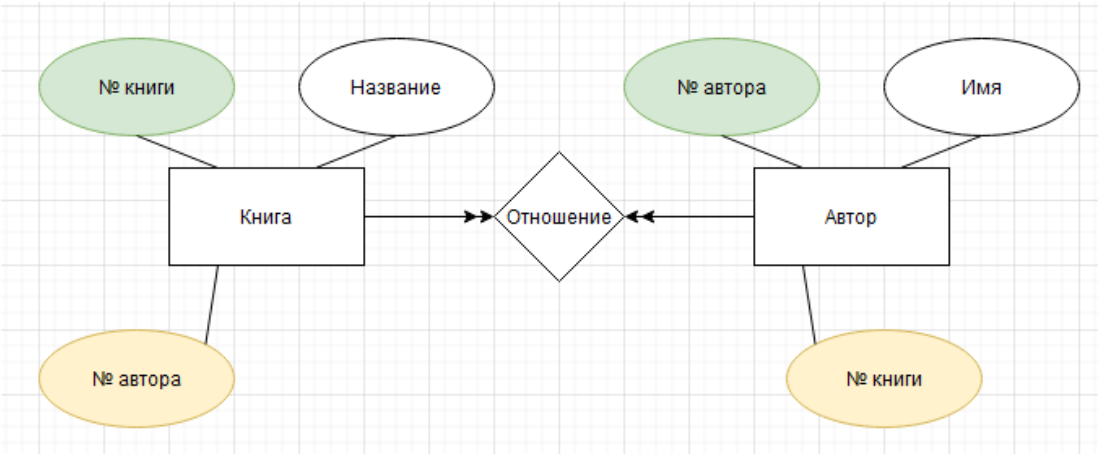
Многозначность полностью или практически полностью будет устранена на этапе логического проектирования во время приведения схемы к 1NF.

Связь «многие-ко-многим» может создать большую избыточность данных.

Как же избавиться от связи многие-ко-многим в концептуальной модели?

Допустим, в базе данных есть 2 сущности: «Книга» и «Автор». Книга может быть написана несколькими авторами, а один автор может написать несколько книг. Получается, что между сущностями «Книга» и «Автор» имеется отношение «многие-ко-многим», ибо одной строке из таблицы «Книга» может соответствовать несколько строк таблицы «Автор», а одной строке таблицы «Автор» может соответствовать несколько строк таблицы «Книга».

Графически сейчас это выглядит так:



Зелёным обозначены первичные ключи, жёлтым - внешние. Именно через эти атрибуты идёт связь между сущностями.

Для устранения связи между сущностями в отношении «многие-ко-многим» обычно вводится дополнительная сущность, которая называется ассоциативной. Гораздо реже связь «многие-ко-многим» устраняется через изменение структуры таблиц без выделения дополнительных (например, в ситуациях, когда данные изначально были некорректно разделены по таблицам).

Ассоциативная сущность будет связывать все таблицы, находящиеся друг с другом в отношениях «многие-ко-многим». В этой сущности обычно выделяется случайный первичный ключ (какое-нибудь числовое поле) и внешние ключи, ссылающиеся на таблицы, которые нужно связать.

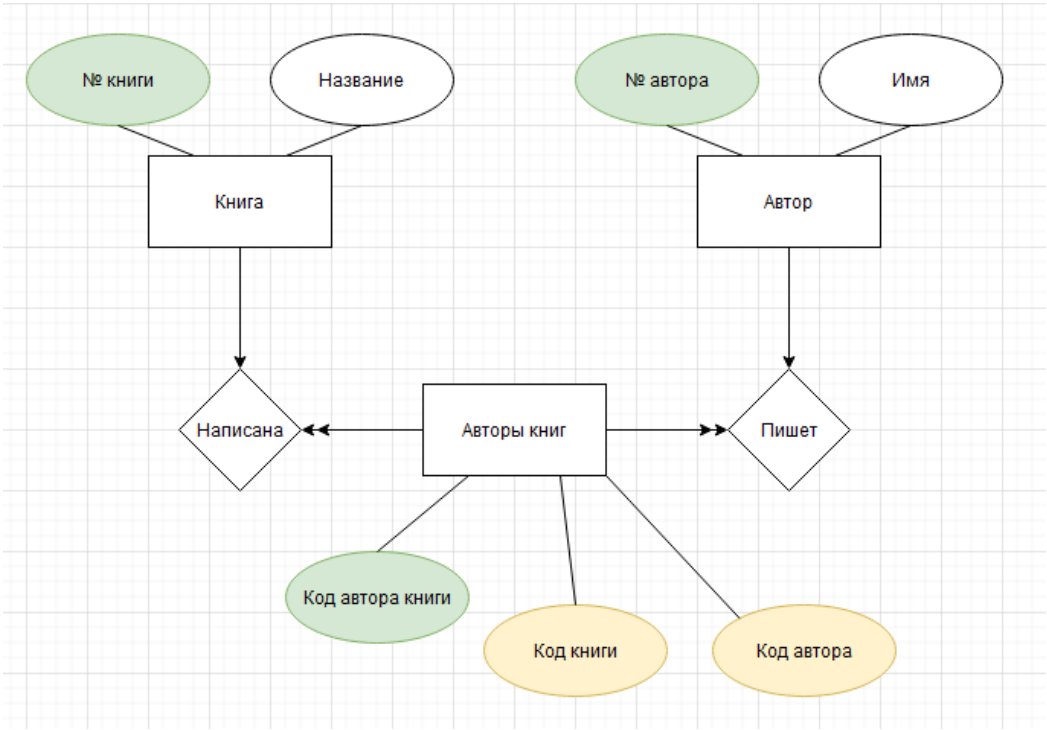
На примере книг и авторов можно поступить так:

- В сущности «Книга» убрать атрибут «№ автора».
- В сущности «Автор» убрать атрибут «№ книги».
- Добавить новую сущность с названием «Авторы книг» и добавить в неё следующие атрибуты: «Код автора книги» (первичный ключ), «Код книги» (внешний ключ для

связи с сущностью «Книга»), «Код автора» (внешний ключ для связи с сущностью «Автор»).

Между таблицами «Книга» и «Автор» теперь не будет отношения «многие-ко-многим», при этом эти таблицы будут иметь отношение «один-ко-многим» с таблицей «Авторы книг».

Графически это будет выглядеть так:



А в табличном виде данные теперь будут организованы следующим образом:

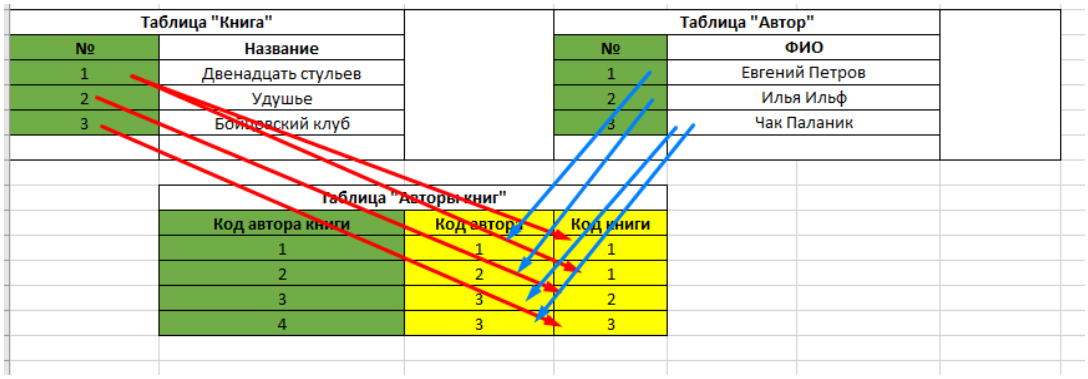


Таблица "Книга"

№	Название
1	Двенадцать стульев
2	Удушье
3	Бойцовский клуб

Таблица "Автор"

№	ФИО
1	Евгений Петров
2	Илья Ильф
3	Чак Паланик

Таблица "Авторы книг"

Код автора книги	Код автора	Код книги
1	1	1
2	2	1
3	3	2
4	3	3

В таком формате представления устраняется дублирование данных в таблицах «Книга» и «Автор». Связь между таблицами осуществляется благодаря идентификаторам - первичным и внешним ключам.

Как понять, какая связь есть между двумя сущностями?

Для этого нужно провести анализ предметной области и задавать себе простой вопрос.

Например, если речь идёт о сущностях «Студент» и «Группа», то можно задать следующие вопросы: «Сколько студентов учится в одной группе?», «В скольких группах учится один студент?». Отвечая на эти вопросы, можно прийти к выводу о том, что студент может учиться только в одной группе, а одна группа может включать в себя множество студентов. Таким образом между сущностями «Группа» и «Студент» формируется отношение «один-ко-многим».

2.4 Именование сущностей, атрибутов и отношений

Название сущности должно полностью и точно отражать сущность, должно указывать на данные, которые будут храниться в этой сущности.

- Название сущности – имя существительное (обычно в единственном числе) или словосочетание на основе существительного (примеры: «Книга», «Автор», «Авторы книг», «Студент», «Группа», «Дисциплины преподавателей» и т.п.).
- Название сущности уникально.
- Название сущности не должно содержать спецсимволы (нижнее подчёркивание разрешено). Цифры тоже могут быть.
- Желательно не использовать сокращения и аббревиатуры, если они являются специфичными. Иногда считается нежелательным в названиях сущностей использовать указания на принадлежность (например, «авторы книг» (альтернативное название – «Авторы_Книги»), но ничего страшного в этом нет.
- Названия атрибутов обычно являются именами существительными в единственном числе
- В качестве названия отношений лучше использовать глаголы.
- Следует отметить, что это лишь общие рекомендации. Это не правила, не требования, не стандарты. В реальных условиях компания – разработчик ПО может иметь собственные внутренние правила именования сущностей, атрибутов, связей и т.п.

2.5 Пример

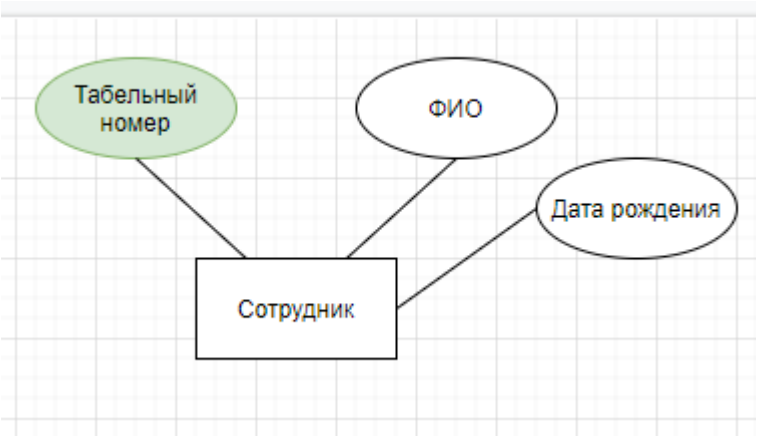
Здесь будет показан пример концептуального проектирования по варианту “Спортзал”. Диаграмма рисовалась на сайте https://app.diagrams.net/

В спортзале работают сотрудники, поэтому можно выделить сущность “Сотрудник”.

Сущность	Атрибуты
“Сотрудник”. Сущность хранит информацию о сотрудниках.	Табельный номер
	ФИО
	Дата рождения
	Паспортные данные
	Контактная информация
	Адрес
	Должность
	Оклад
	Логин
	Пароль
	Фотография

Зелёным цветом выделен атрибут, который можно считать первичным ключом.
В этой сущности есть несколько многозначных атрибутов: ФИО, Паспортные данные, Контактная информация и т.д. На этапе концептуального проектирования избавляться от них не нужно. Поэтому идём дальше.

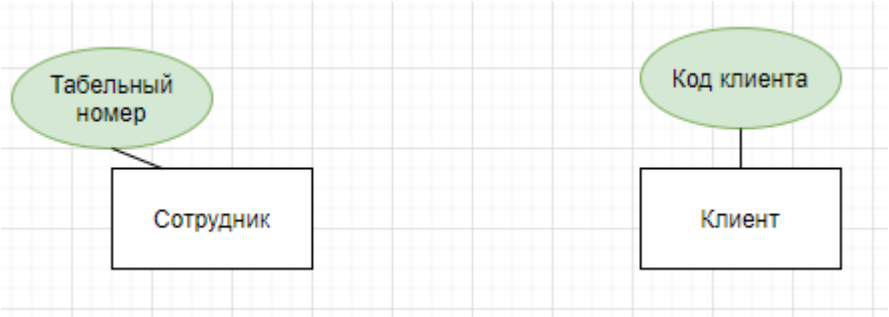
Но сначала посмотрим, как будет выглядеть сущность в концептуальной схеме. Здесь мы обозначили только первичный ключ и парочку атрибутов. Далее мы будем ограничиваться только обозначением первичных и внешних ключей, иначе схема получится слишком громоздкой.



В зал ходят клиенты, поэтому можно выделить сущность “Клиент”.

Сущность	Атрибуты
“Клиент”. Сущность хранит информацию о клиентах.	Код клиента
	ФИО
	Дата рождения
	Паспортные данные
	Контактная информация
	Тип клиента
	Организация
	Наличие справки
	Логин
	Пароль
	Фотография
	Абонемент

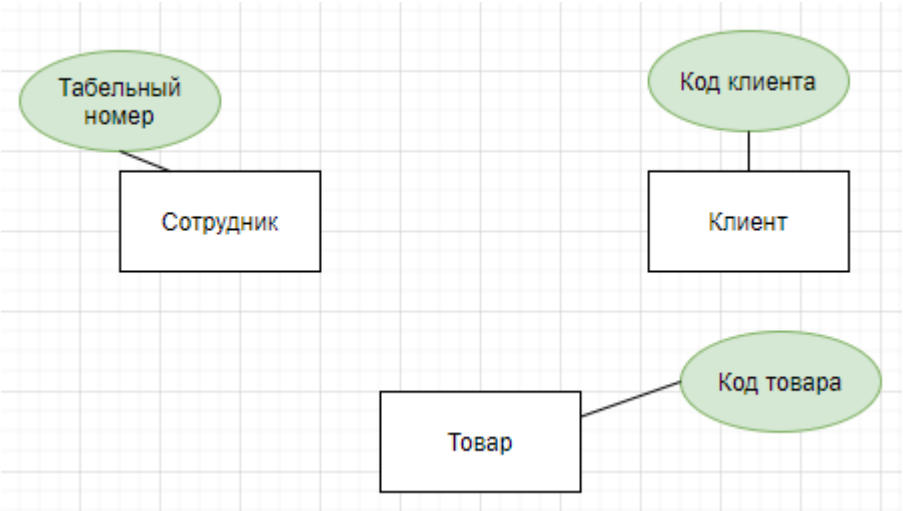
Атрибут “Тип клиента” нужен для того, чтобы можно было хранить информацию о том, является ли клиент студентом или корпоративным клиентом. Атрибут “Организация” нужен для хранения названия организации-партнёра, в которой работает корпоративный клиент. Атрибут “Наличие справки” служит для хранения информации о том, принёс ли разрешение от врача клиент или нет.



В магазине спортзала продают товары. Нужно выделить сущность “Товар”.

Сущность	Атрибуты
“Товар”. Сущность хранит информацию о товарах.	Код товара
	Наименование товара
	Тип товара
	Стоимость закупки
	Стоимость продажи
	Количество
	Фотография

Товары в магазине имеют разные типы. Товар может относиться к продуктам питания, одежде, спортивному инвентарю и пр.



Товары закупаются у поставщиков и продаются клиентам. В данном случае у нас есть два “направления движения”: нужно описать сущности,

относящиеся к поставщикам и закупкам, и сущности, относящиеся к продаже товаров. Выделим сущность “Поставщик”.

Сущность	Атрибуты
“Поставщик”. Сущность хранит информацию о поставщиках.	Код поставщика
	Наименование поставщика
	Контактная информация
	Реквизиты
	Адрес

Т.к. мы выделили сущность “Поставщик”, то теперь можем выделит сущность “Поставка”. Данная сущность будет хранить информацию о поставках товаров в магазин спортзала разными поставщиками.

Сущность	Атрибуты
“Поставка”. Сущность хранит информацию о поставках товаров поставщиками.	Код поставки
	Код поставщика
	Табельный номер
	ДатаВремя поставки
	Состав поставки

Жёлтым цветом выделены внешние ключи - атрибуты, через которые осуществляется связь с другими сущностями.
В атрибуте “Состав поставки” предполагается хранение информации о товарах, которые пришли в магазин.

Раз уж у нас появились внешние ключи, то сразу установим отношения между сущностью “Поставка” и сущностями, от которых она зависит.

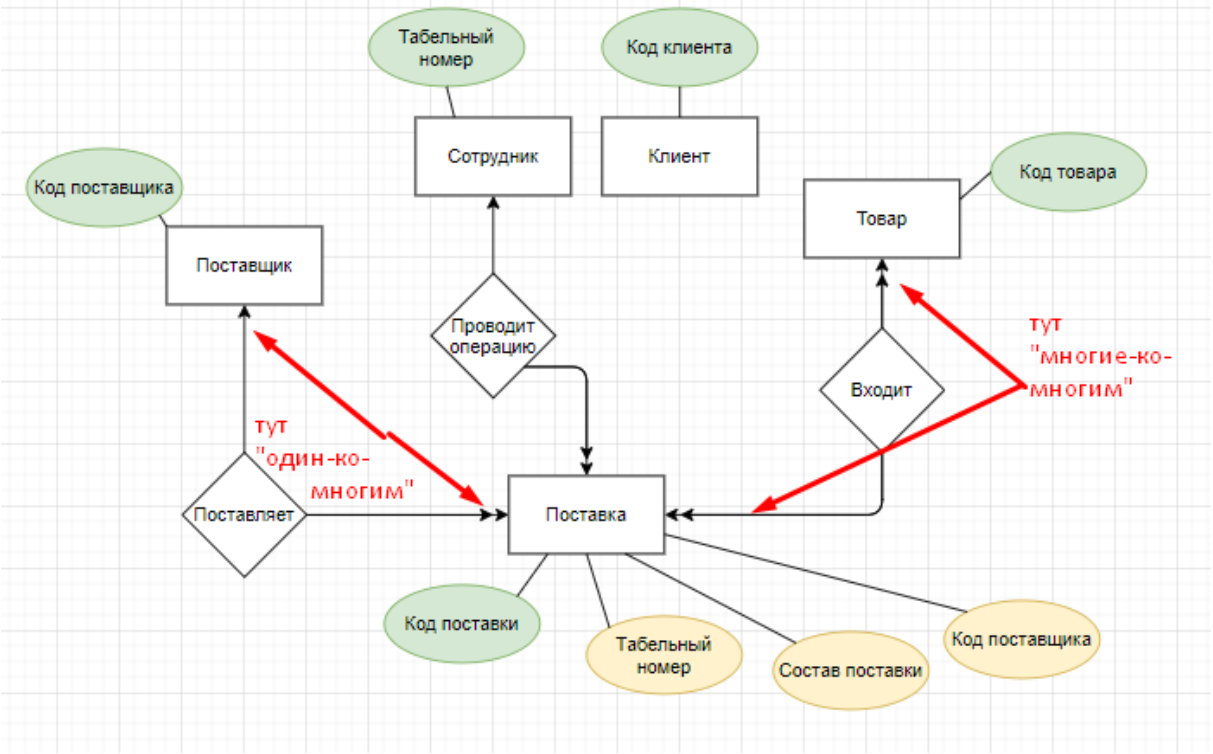
“Поставщик” - “Поставка”
Сколько поставок может отправить поставщик? Много.
Сколько поставщиков может отправить одну и ту же поставку? Один.
Таким образом, связь между сущностями “Поставщик” - “Поставка”: 1:N (“один-ко-многим”).

“Сотрудник” - “Поставка”
Сколько сотрудников может “обработать” поставку? Один.

Сколько поставок может “обработать” сотрудник? Много.
Связь между сущностями “Сотрудник” - “Поставка”: 1:N

“Товар” - “Поставка”
Сколько поставок может содержать один и тот же товар? Много.
В скольких поставках может быть один и тот же товар? Во многих.
Связь между сущностями “Товар” - “Поставка”: N:M (“многие-ко-многим”).

Если вы внимательно читали теорию, то на этом моменте можете воскликнуть что-то вроде “Связи “многие-ко-многим” не может быть!”. От этой связи мы избавимся потом. А пока просто дорисуем схему.



Введём сущность “Продажа”, в которой будет храниться информация о продажах.

Сущность	Атрибуты
“Продажа”. Сущность хранит информацию о продажах товаров.	Код продажи
	Код клиента
	Табельный номер
	ДатаВремя продажи

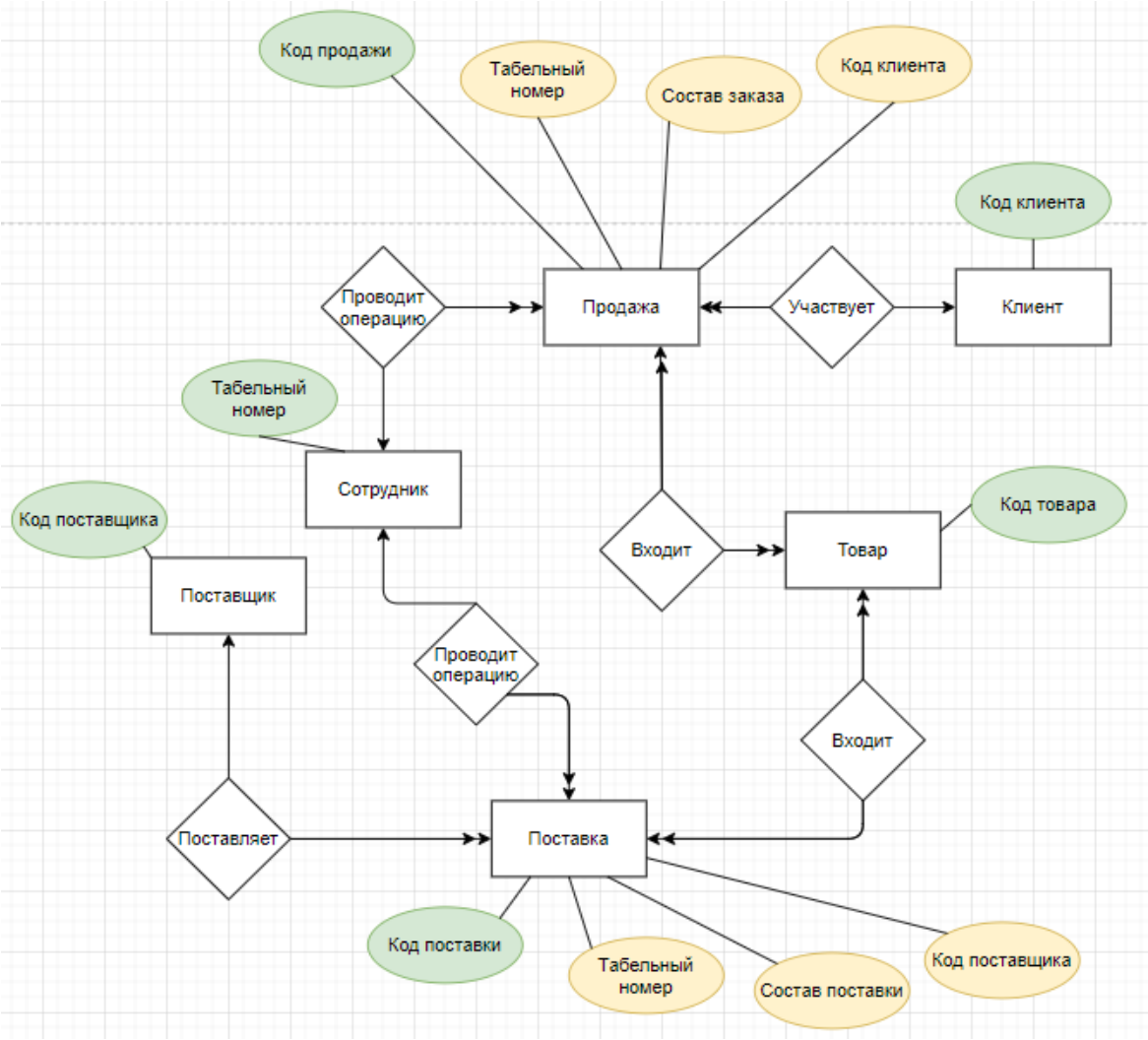
	Состав заказа

На схеме она будет выглядеть аналогично сущности “Поставка”.
Но для начала выделим отношения.

“Продажа” - “Клиент”.
Сколько покупок может делать клиент? Много.
Сколько клиентов может совершать одну покупку в указанный момент времени и с определённым составом товаров? Один.
Связь между сущностями “Клиент” - “Покупка”: 1:N

“Сотрудник” - “Продажа”.
Сколько продаж может осуществлять сотрудник? Много.
Сколькими сотрудниками может проводиться одна продажа? Одним.
Связь между сущностями “Сотрудник” - “Продажа”: 1:N

“Товар” - “Продажа”.
Сколько товаров может быть в одном заказе? Много.
В скольких заказах может быть один товар? Во многих.
Связь между сущностями “Товар” - “Продажа”: N:M



В спортзале есть...залы, которые называются секциями. У каждой секции есть своё имя. Введём сущность “Секция”.

Сущность	Атрибуты
“Секции”. Сущность хранит информацию о секциях.	Код секции
	Название секции

А в секциях есть спортивный инвентарь, тренажеры и т.д., и т.п. У каждой единицы инвентаря есть своё имя, тип.

Сущность	Атрибуты
“Инвентарь”. Сущность хранит информацию об инвентаре.	Код инвентаря
	Название инвентаря

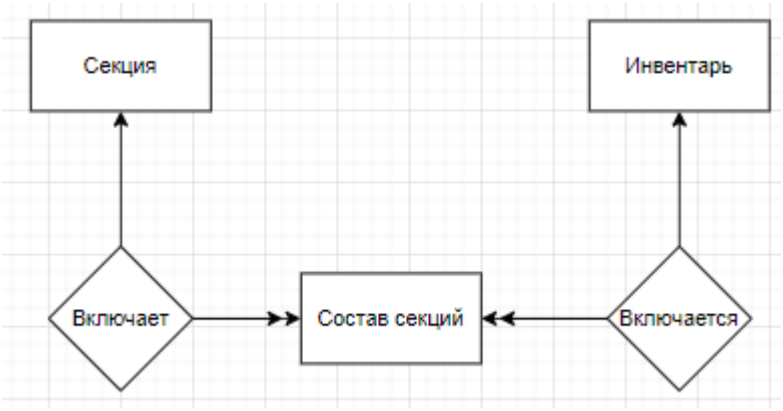
	Тип инвентаря

Инвентарь находится в секциях.
В одной секции может быть, множество разных единиц инвентаря.
Один и тот же тип инвентаря может быть в разных секциях.
Это значит, что между сущностями “Секция” и “Инвентарь” формируется отношение “многие-ко-многим”.

Избавимся от этой связи сразу. Для этого нужно ввести связующую, ассоциативную сущность, которая будет связывать секции с инвентарём. В этой сущности должны быть “ссылки” на сущность “Секция” и “Инвентарь”.

Сущность	Атрибуты
“Состав секций”. Сущность хранит информацию о том, какой инвентарь в какой секции находится.	Код состава секции
	Код секции
	Код инвентаря

Дорисуем схему.



Избавимся теперь от связи “многие-ко-многим” в продажах и покупках.
Сейчас в поле “Состав заказа” придётся хранить информацию о товаре, который приобретается, и о приобретаемом количестве.

Сущность	Атрибуты
“Продажа”. Сущность хранит информацию о продажах товаров.	Код продажи
	Код клиента
	Табельный номер
	ДатаВремя продажи
	Состав заказа

Для решения проблемы нужно поступить так: в сущности, “Продажа” хранить только общую информацию о заказах, а состав заказа вынести в отдельную сущность “Состав заказа”. При этом атрибут “Состав заказа” из сущности “Продажа” нужно удалить, потому что он становится неактуальным.

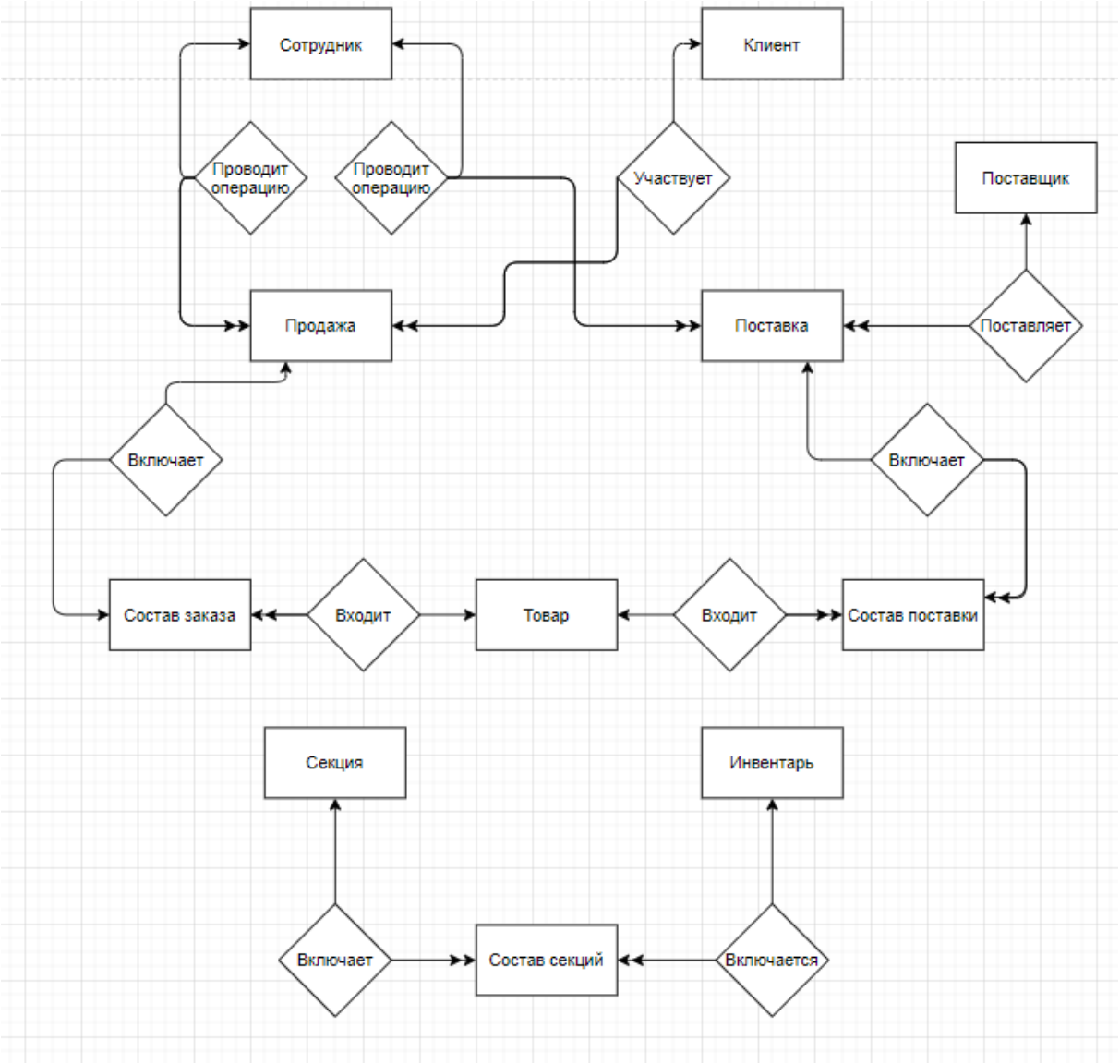
Сущность	Атрибуты
“Состав заказа”. Сущность хранит информацию о составе заказов.	Код состава заказа
	Код продажи
	Код товара
	Количество товара

В этой сущности “Код продажи” нужен для связи заказа с составом, а “Код товара” - для связи с товаром, включённым в заказ.

Абсолютно аналогичная ситуация с поставками. В этом случае нужно выделить сущность “Состав поставки”.

Сущность	Атрибуты
“Состав поставки”. Сущность хранит информацию о составе заказов.	Код состава поставки
	Код поставки
	Код товара
	Количество товара

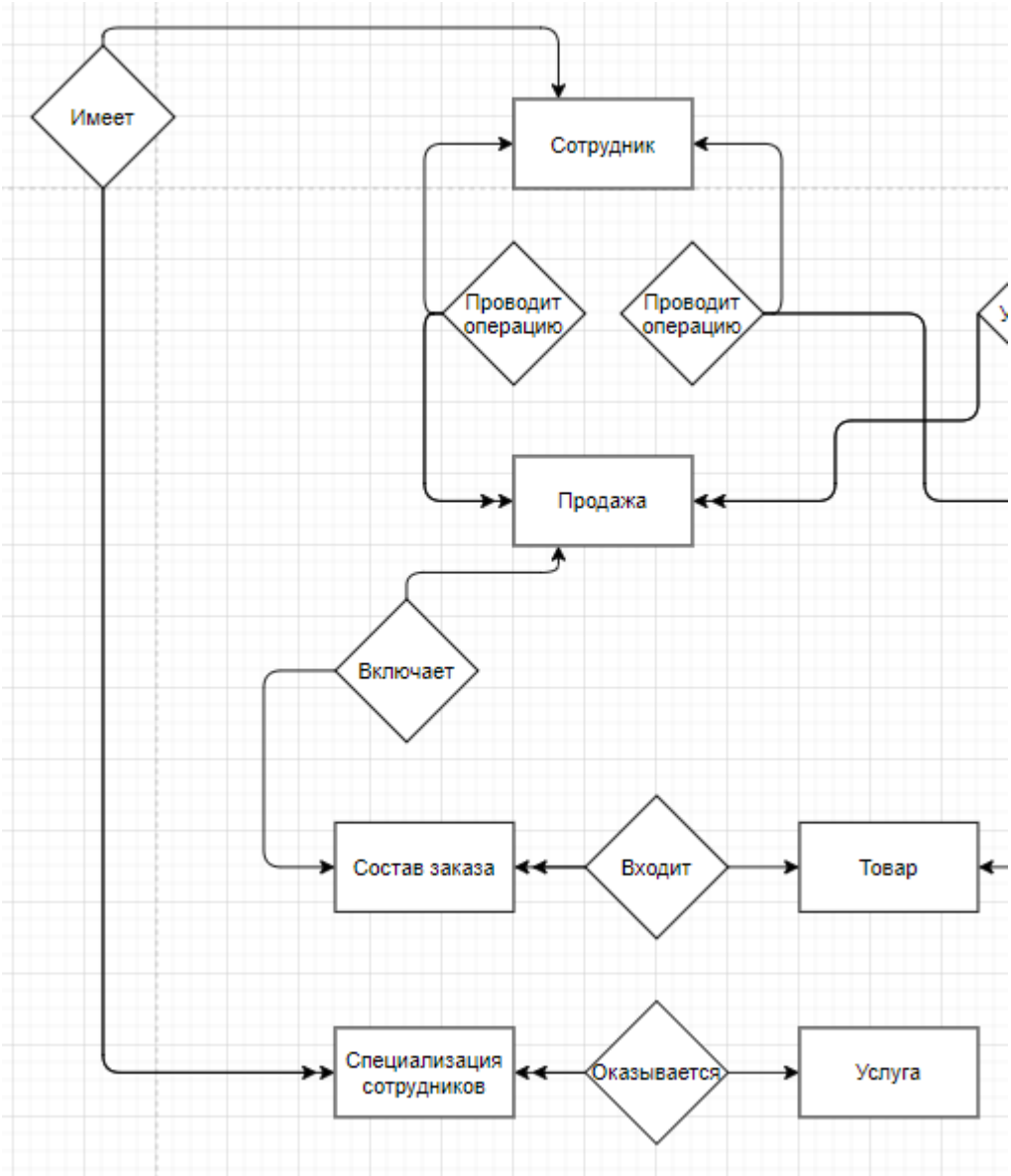
Дорисуем схему. Вынуждены убрать атрибуты с неё, т.к. схема становится слишком большой.



В спортзале оказываются услуги. Под услугами можно считать занятия, составление программ питания, посещение SPA, массаж и т.д.
Услуги оказывают сотрудники. Один сотрудник (например, тренер) может оказывать различные виды услуг (например, персональные тренировки, составление программы питания), но одна и та же услуга может быть оказана разными людьми. Получается, что между сотрудниками и услугами формируется отношение “многие-ко-многим”. Получается, что у персонала есть своя специализация. Выделим сущность “Услуга”, в которой будет храниться список услуг. И выделим сущность “Специализация сотрудников”, которая позволит устранить связь “многие-ко-многим” и будет хранить в себе информацию о том, кто и чем занимается.

Сущность	Атрибуты
“Услуга”. Сущность хранит информацию об услугах.	Код услуги
	Название услуги
	Тип услуги
	Стоимость услуги

Сущность	Атрибуты
“Специализация сотрудников”. Сущность хранит информацию о специализации сотрудников.	Код специализации
	Табельный номер
	Код услуги



Далее нужно решить вопрос с хранением информации о расписании. В нём указывается дата и время начала занятия, секция, в котором будет занятие, оказываемая услуга (клиент может прийти не на персональное занятие, допустим, а ради какой-то иной услуги), сотрудник, который оказывает услугу, информация о клиенте.

Сущность	Атрибуты
“Расписание”. Сущность хранит информацию расписании занятий и услуг.	Код занятия
	Табельный номер
	Код услуги
	Код секции
	Код клиента
	ДатаВремя

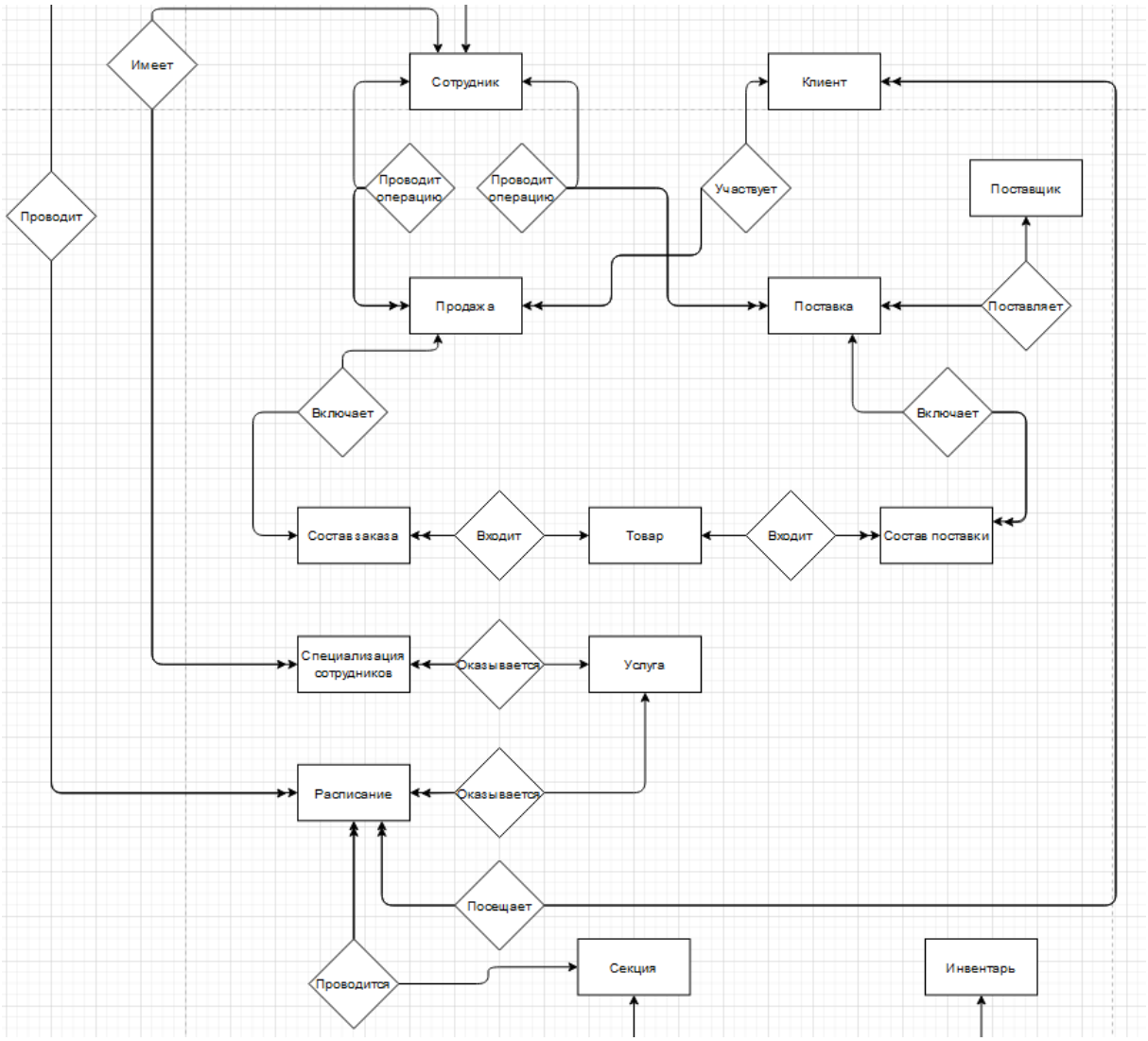
Установим отношения.
“Сотрудник” - “Расписание”.
Сколько занятий может проводить один сотрудник? Много.
Сколько сотрудников может проводить одно занятие? Один.
Связь между сущностями “Сотрудник” - “Расписание”: 1:N

“Секция” - “Расписание”.
Сколько занятий может проходить в одной секции? Много.
В скольких секциях может проходить одно занятие? В одной.
Связь между сущностями “Секция” - “Расписание”: 1:N

“Услуга” - “Расписание”.
Сколько раз может проходить тот или иной вид занятий? Много.
К скольким видам занятий может относиться конкретное занятие в указанное в расписании время и в указанную дату? К одному.
Связь между сущностями “Услуга” - “Расписание”: 1:N

“Клиент” - “Расписание”.
На сколько занятий может ходить клиент? На множество.
Сколько клиентов может приходить на занятие? В зависимости от типа (есть групповые и не групповые тренировки).
Ответ на второй вопрос говорит о том, что формируется необязательная (т.е. такая связь будет не всегда (например, если речь идёт об индивидуальных занятиях)) связь “многие-ко-многим”.

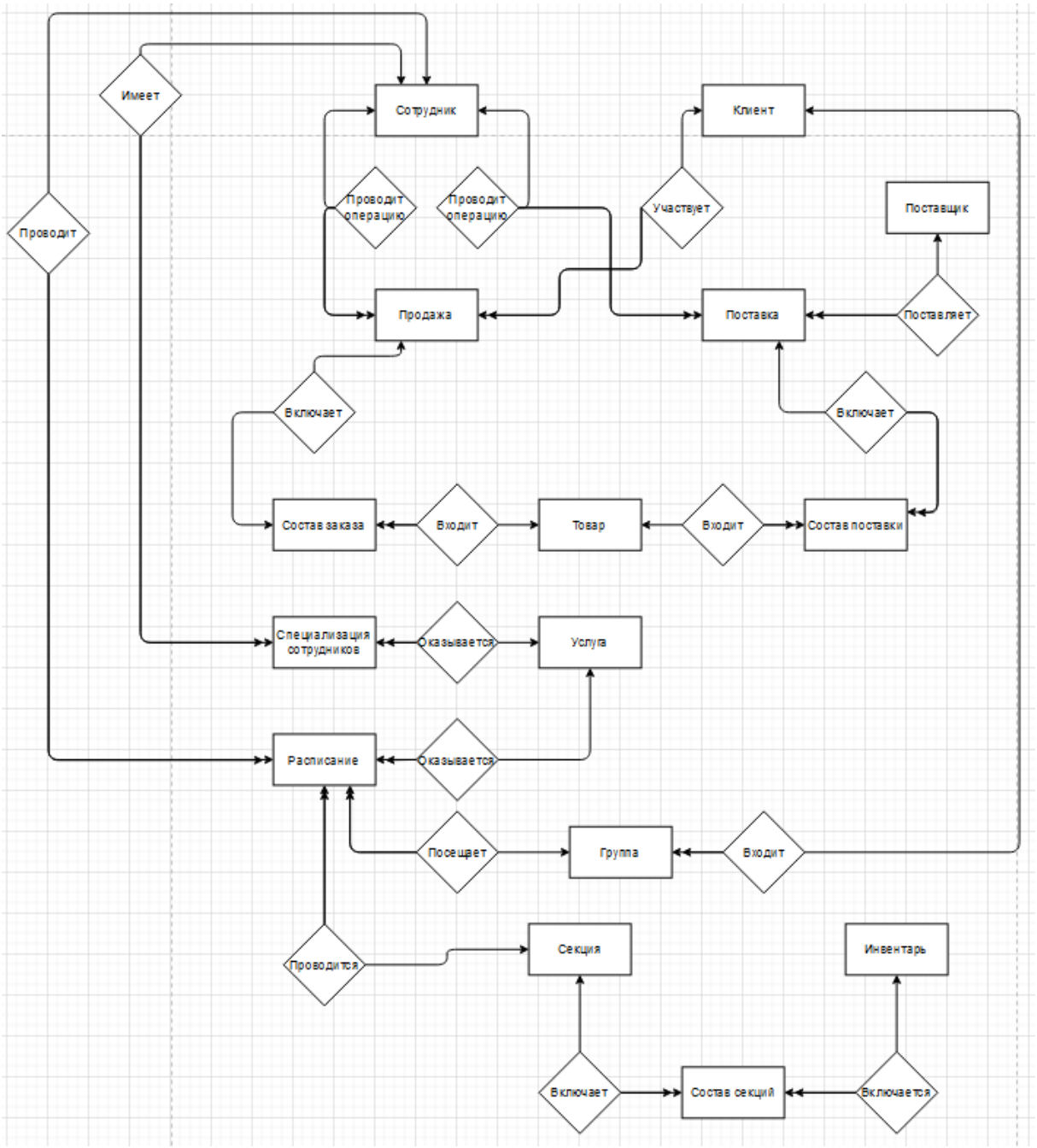
Т.е. связь “многие-ко-многим” между сущностями “Клиент” и “Расписание” будет устанавливаться только при групповых занятиях. Но это не отменяет необходимость в избавлении от такой связи.



Для устранения связи “многие-ко-многим” между сущностями “Расписание” и “Клиент” нужно ввести дополнительную сущность, которая будет хранить информацию о группе, которая должна прийти заниматься на групповую тренировку.

Сущность	Атрибуты
“Группа”. Сущность хранит о группах для групповых занятий	Код группы
	Код клиента
	Код занятия

Итоговая концептуальная схема выглядит так:



Список сущностей и их атрибутов:

Сущность	Атрибуты
“Сотрудник”. Сущность хранит информацию о сотрудниках.	Табельный номер
	ФИО
	Дата рождения
	Паспортные данные

	Контактная информация
	Адрес
	Должность
	Оклад
	Логин
	Пароль
	Фотография

Сущность	Атрибуты
“Клиент”. Сущность хранит информацию о клиентах.	Код клиента
	ФИО
	Дата рождения
	Паспортные данные
	Контактная информация
	Тип клиента
	Организация
	Наличие справки
	Логин
	Пароль
	Фотография
	Абонемент

Сущность	Атрибуты
“Товар”. Сущность хранит информацию о товарах.	Код товара
	Наименование товара
	Тип товара
	Стоимость закупки
	Стоимость продажи

	Количество
	Фотография

Сущность	Атрибуты
“Поставщик”. Сущность хранит информацию о поставщиках.	Код поставщика
	Наименование поставщика
	Контактная информация
	Реквизиты
	Адрес

Сущность	Атрибуты
“Секции”. Сущность хранит информацию о секциях.	Код секции
	Название секции

Сущность	Атрибуты
“Инвентарь”. Сущность хранит информацию об инвентаре.	Код инвентаря
	Название инвентаря
	Тип инвентаря

Сущность	Атрибуты
“Состав секций”. Сущность хранит информацию о том, какой инвентарь в какой секции находится.	Код состава секции
	Код секции
	Код инвентаря

Сущность	Атрибуты
“Состав заказа”. Сущность хранит информацию о составе заказов.	Код состава заказа
	Код продажи
	Код товара
	Количество товара

Сущность	Атрибуты
“Состав поставки”. Сущность хранит информацию о составе заказов.	Код состава поставки
	Код поставки
	Код товара
	Количество товара

Сущность	Атрибуты
“Услуга”. Сущность хранит информацию об услугах.	Код услуги
	Название услуги
	Тип услуги
	Стоимость услуги

Сущность	Атрибуты
“Специализация сотрудников”. Сущность хранит информацию о специализации сотрудников.	Код специализации
	Табельный номер
	Код услуги

Сущность	Атрибуты
“Расписание”. Сущность хранит информацию расписании занятий и услуг.	Код занятия
	Табельный номер
	Код услуги
	Код секции
	ДатаВремя

Сущность	Атрибуты
“Группа”. Сущность хранит о группах для групповых занятий	Код группы
	Код клиента
	Код занятия

Сущность	Атрибуты
“Поставка”. Сущность хранит информацию о поставках товаров поставщиками.	Код поставки
	Код поставщика
	Табельный номер
	ДатаВремя поставки

Сущность	Атрибуты
“Продажа”. Сущность хранит информацию о продажах товаров.	Код продажи
	Код клиента
	Табельный номер
	ДатаВремя продажи