

**Частное учреждение профессионального образования**

**«Высшая школа предпринимательства»**

**(ЧУПО «ВШП»)**

КУРСОВОЙ ПРОЕКТ

«Разработка базы данных для компьютерной мастерской»

Выполнил:

студент 3-го курса специальности

09.02.07 «Информационные системы и программирование»  
Лешкевич Станислав Сергеевич

подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

проректор по информатизации,  
зав. каф. ИСиТ при ЧУВО «ВШП»,  
к.ф.н. Ткачев П.С.

оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тверь, 2023г

Оглавление

[**Введение** 3](#_Toc136539673)

[**Глава 1. Теоретические основы проектирования баз данных для компьютерных мастерских** 5](#_Toc136539674)

[**Виды Баз Данных и их особенности** 6](#_Toc136539675)

[**Нормализация РБД** 9](#_Toc136539676)

[**Компьютерные мастерские и хранение данных** 14](#_Toc136539677)

[**Примеры компьютерных мастерских и БД-ных для хранения их данных** 16](#_Toc136539678)

[**Глава 2. Проектирование и разработка БД** 18](#_Toc136539679)

[**Распределение ролей таблицы и создание представления (VIEW)** 22](#_Toc136539680)

[**Заключение** 24](#_Toc136539681)

[**Список источников:** 25](#_Toc136539682)

[**Приложения** 26](#_Toc136539683)

# **Введение**

В настоящее время, в любой современной организации, сложилась знакомая всем ситуация – информации иногда даже слишком много, но она не структурирована, не согласована, разрознена, не всегда достоверна, её практически невозможно найти и получить в реальные сроки. Разрешению этих проблем способствует концепция создания и использования современных баз данных, таких как -MySQL.

Актуальность темы:

Разработка базы данных для компьютерной мастерской имеет высокую актуальность в наши дни, поскольку повсеместно распространено использование электронной техники в повседневной жизни. Это приводит к значительному увеличению числа обращений в компьютерные мастерские для ремонта и обслуживания компьютеров, ноутбуков, планшетов и мобильных устройств. Компьютерные мастерские обязаны отслеживать информацию о своих клиентах и заказах, а также обеспечивать ее безопасное хранение. Разработка базы данных для компьютерной мастерской существенно повышает эффективность ведения бизнеса, включая экономию времени на поиск и обработку данных, уменьшение числа ошибок и повышение качества обслуживания клиентов. Многие компьютерные мастерские используют MySQL в качестве инструмента управления базами данных, благодаря его высокой производительности, гибкости и бесплатности использования.

Цель работы:

Провести проектирование и разработку базы данных для компьютерной мастерской.

Задачи:

* Изучение баз данных и их видов
* Изучение потребностей компьютерной мастерской в хранении данных о клиентах, заказах и др.
* Оценка существующих решений и выбор подходящего инструмента для управления базами данных.
* Определение структуры данных и разработка схемы базы данных.
* Разработка базы данных.

# **Глава 1. Теоретические основы проектирования баз данных для компьютерных мастерских**

«База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.» [1]

Так говорит нам Википедия, по первому же запросу «Базы данных» в поисковой строке.

Так же там пишется, что данное понятие может рассматриваться и в двух смыслах.

В широком: «…понятие истории баз данных обобщается до истории любых средств, с помощью которых человечество хранило и обрабатывало данные.» [1]

И в узком: «История баз данных в узком смысле рассматривает базы данных в традиционном (современном) понимании. Эта история начинается с 1955 года, когда появилось программируемое оборудование обработки записей. Программное обеспечение этого времени поддерживало модель обработки записей на основе файлов. Для хранения данных использовались перфокарты.» [1]

Сам термин появился еще в начале 1960-х годов, что относительно не так давно, и в основном употреблялся на симпозиумах, которые организовывала компания System Development Corporation (SDC), которая считается первым в мире производителем ПО. По началу термин **база данных** понимался лишь в контексте систем искусственного интеллекта, а в современном понимании в широкое употребление вошел только на следующий десяток лет, в 70-е годы ХХ века. [4]

Основной принцип работы баз данных заключается в хранении, организации и обработке большого объема структурированных и неструктурированных данных с помощью специально разработанных программных средств.

Для этого данные разбиваются на небольшие части, так называемые таблицы, которые связаны между собой по определенным правилам. Также базы данных имеют набор системных функций, позволяющих выполнять операции с данными, такие как добавление, изменение, удаление, поиск и сортировка.

Получается, что БД – это своего рода книжный шкаф, который содержит в себе полки (таблицы), на которых находятся книги (Столбцы и строки), в которых содержится информация.

## **Виды Баз Данных и их особенности**

Всего различают примерно до 10 видов БД, но обычно самыми популярными являются следующие 5:

* иерархические;
* объектные или объектно-ориентированные;
* реляционные;
* сетевые;
* функциональные.

Итак, начнем по порядку:

Иерархические базы данных

Здесь происходит установление связей между объектами, которые делятся на родительские классы или категории и дочерние экземпляры этих классов или категорий. Каждый дочерний объект имеет только одного родителя. Древовидная структура служит графическим представлением такой базы данных(рис.1).

Особенности:

* Отношения между объектами реализованы в виде физических указателей. Например, в файловой системе путь к папке или файлу строится из имен корневых и вложенных каталогов;
* Моделирование отношений вложенности и подчиненности.

Ограничения: Технология иерархической организации не предполагает связи «многие-ко-многим», а значит, система хранения данных довольно ограничена. [2]

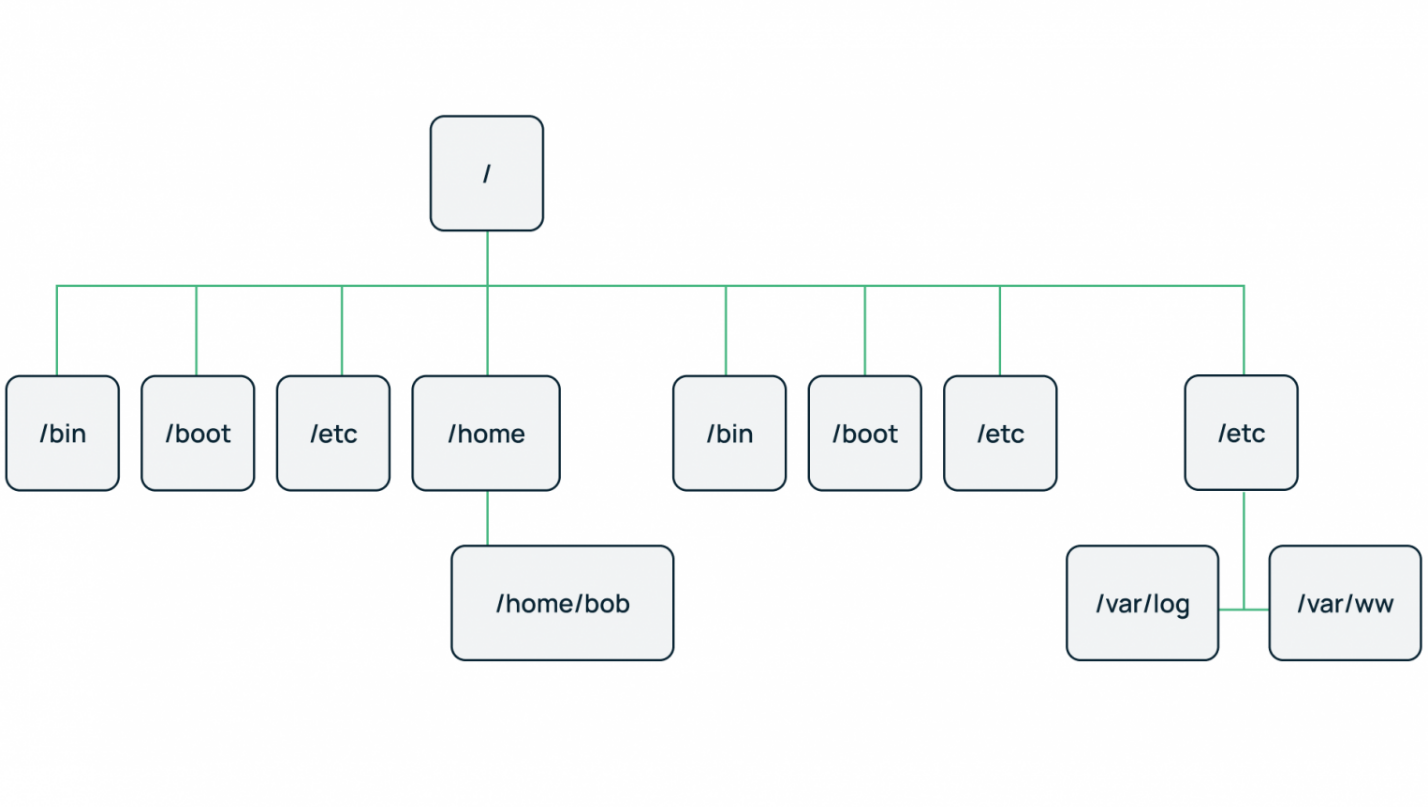


Рисунок 1. Пример иерархической базы данных

Объектно-ориентированные базы данных

Объектно-ориентированная база данных (ООБД) является типом баз данных, в которых данные рассматриваются как наборы объектов, обладающих свойствами и методами. ООБД позволяют моделировать объекты реального мира вместо таблиц и связей, что облегчает работу программистов и повышает качество программного продукта.

Основные особенности ООБД:

* Данные хранятся в виде объектов, имеющих методы и свойства.
* ООБД поддерживает концепцию наследования, что позволяет наследовать свойства и методы объектов.
* ООБД предоставляет механизмы для управления транзакциями и обеспечения целостности данных.
* ООБД обладает расширенными возможностями индексирования данных, что позволяет ускорять поиск при обработке больших объемов данных.

Одним из основных ограничений ООБД является то, что они не всегда поддерживают язык SQL, который является стандартом для реляционных баз данных. Кроме того, ООБД могут оказаться более сложными для использования и требовать больших затрат на обучение программистов по сравнению с теми же реляционными базами данных.

Реляционные базы данных

Здесь остановимся поподробнее, ведь это самый старый тип БД, в котором много всего интересного.

Итак, как я уже сказал - это самый старый тип баз данных. Его теоретические основы которого были заложены в 1970 году британским ученым Эдгаром Коддом. Данные в этом типе баз данных организованы в таблицы, которые содержат строки и столбцы. В строках находятся значения свойств объектов, а в столбцах – сами свойства, называемые полями.

Сложные взаимоотношения объектов в реляционных БД моделируются с помощью внешних ключей – ссылок на другие таблицы. Это позволяет подходить к вопросу проектирования базы данных с позиций **нормализации** – минимизации избыточности при описании свойств объектов. [2]

Например, если речь идет о меню ресторана, то у каждого блюда есть вес, цена, наименование, калорийность и категория, к которой оно относится — горячие закуски, холодные закуски, первые блюда, десерты, салаты и так далее. Связь между блюдами и категорией выполняется посредством ссылочного поля индекса категории в таблице блюд. [2]

Такой подход, как нормализация базы данных, имеет ряд преимуществ, среди которых:

1. Минимизация объема базы данных: не требуется добавлять название категории для каждого нового блюда.
2. Повышение целостности системы: все блюда связаны с категориями меню, и невозможно добавить новое блюдо без указания соответствующей категории или ссылаясь на несуществующую категорию.
3. Упрощение масштабирования: новые блюда могут быть легко добавлены в существующие категории, а также могут быть созданы новые категории для добавления новых блюд.
4. Повышение отказоустойчивости: проектирование базы данных с учетом нормализации позволяет повысить скорость выполнения запросов, так как данные хранятся в оптимальной форме. При увеличении количества записей в таблицах в будущем это поможет сохранять качество пользовательского опыта.

Думаю, для понимания нужно еще немного рассказать о нормализации РМД.

## **Нормализация РБД**

Сама по себе нормализация в БД – это «процесс удаления избыточных данных.» [3]

Также можно рассматривать нормализацию как процесс проектирования базы данных, где целью является разбиение таблицы на более мелкие отношения для обеспечения логической целостности и максимальной эффективности работы с данными. Тогда:

Нормализация – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности. [3]

Избыточность устраняется, как правило, за счёт декомпозиции отношений (таблиц), т.е. разбиения одной таблицы на несколько. [3]

Всего существует 5 основных форм БД:

* Первая нормальная форма (1NF) – каждая строка таблицы должна содержать только атомарные значения (не разбиваемые на более мелкие части) и при этом следует избегать повторяющихся групп.
* Вторая нормальная форма (2NF) – таблица находится в 2НФ, если она находится в 1НФ и каждый не ключевой атрибут зависит от всего первичного ключа.
* Третья нормальная форма (3NF) – таблица находится в 3НФ, если она находится в 2НФ и каждый не ключевой атрибут зависит только от первичного ключа и не зависит от других не ключевых атрибутов.
* Четвертая нормальная форма (4NF) – таблица находится в 4НФ, если она находится в 3НФ и атрибуты, которые не являются ключевыми, не зависят друг от друга.
* Пятая нормальная форма (5NF) – таблица находится в 5НФ, если она находится в 4НФ и содержит только те отношения, которые не могут быть декомпозированы даже в рамках третьего уровня нормализации.

Обычно в реальном мире нормализация базы данных до третьей нормальной формы (3NF) считается стандартной практикой, так как она способна устранить большинство аномалий, не нанося при этом существенного ущерба производительности и удобству использования базы данных. В то же время, высшие нормальные формы могут снижать производительность и создавать трудности при работе с данными. Собственно говоря, поэтому я и буду придерживаться 3NF в своём проекте.

Теперь, когда мы немного понимаем про нормализацию, можно продолжать изучать Реляционные БД.

Наглядный пример моделирования сложных взаимоотношений в реляционных БД приведен в прил.2. Здесь мы видим модель базы данных учебного заведения, где есть следующие объекты: ученик, курс, преподаватель, отдел, направление обучения. [2]

Для организации связи между преподавателем и отделом используются внешние ключи id курса и id преподавателя в таблице Секция, а также Отдел в таблице Курс. А связь между учеником и направлением обучения реализована через таблицу Направление обучения студента и ее внешние ключи id студента и id направления обучения.

Чтобы получить статистику по количеству студентов, посещающих курс и их привязке к определенным преподавателям, необходимо написать запрос с использованием JOIN, присоединяющий таблицы учеников, направлений, курсов и преподавателей и сделав соответствующую группировку по преподавателям.

Запросы в реляционных базах данных формируют с помощью структурированного языка SQL. Его предложения позволяют:

* делать выборки,
* проводить агрегации и группировки,
* изменять и удалять данные,
* модифицировать структуру БД (создавать таблицы, поля),
* управлять доступом пользователей к тем или иным операциям и пр.

**Преимущества реляционного подхода**:

* определение сложных отношений между объектами,
* нормализация и денормализация данных,
* структурированный язык запросов,
* богатая история развития и широкое распространение (основной инструмент при разработке различных приложений и сервисов).

**Недостатки подхода:**жесткая структура сведений об объектах.

Сетевые базы данных

Эта технология развивает иерархический подход за счет моделирования сложных отношений между объектами. Здесь потомки могут иметь более одного родителя, однако ограничения иерархического подхода сохраняются. [2]

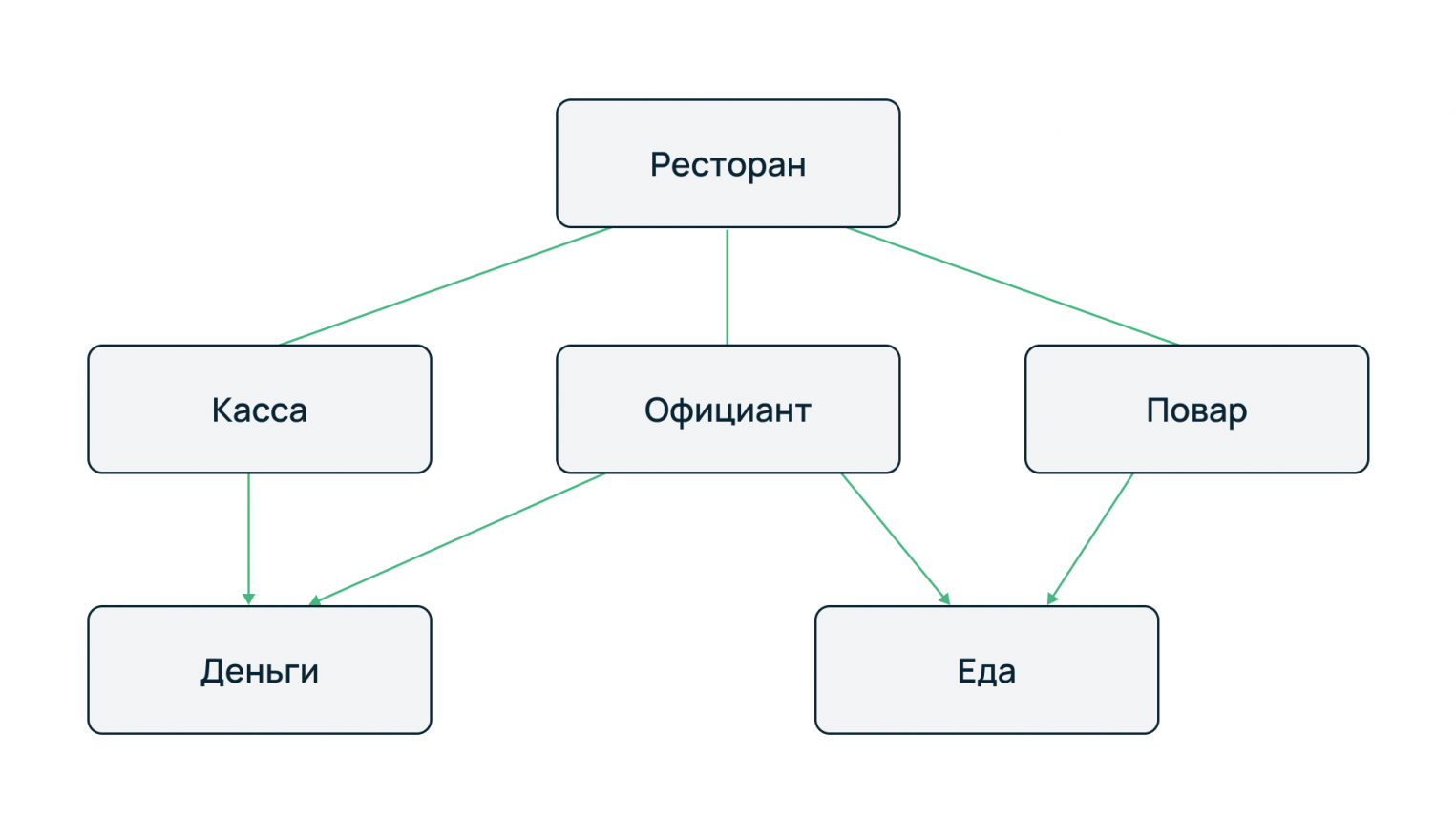


Рисунок 2. Пример сетевой БД

Функциональные базы данных

Функциональные базы данных используются для решения аналитических задач, таких как финансовое моделирование и управление производительностью. Функциональная база данных отличается от реляционной модели. Функциональная база данных организует данные в виде функций, которые описываются математическими выражениями. ФБД применяется в случае, когда необходимо доказывать теоремы и доказуемость результатов является целью.

Реляционная база данных, в свою очередь, организует данные в виде таблиц, которые состоят из строк и столбцов.

## **Компьютерные мастерские и хранение данных**

Компьютерная мастерская – это место, где производятся ремонт, обслуживание, модернизация компьютерной техники. Иными словами, в большинстве случаев это сервисные центры. В таких мастерских могут работать специалисты-техники, занимающиеся ремонтом и обслуживанием компьютеров, но также могут быть программисты и другие специалисты в области iT. В компьютерной мастерской можно получить помощь в установке и настройке программного и аппаратного обеспечения, восстановлении данных, устранении неполадок и других технических вопросах, связанных с компьютерами и сопутствующей техникой.

Хранение данных в компьютерной мастерской является одним из самых важных и ответственных этапов работы с клиентскими компьютерами. Во-первых, необходимо убедиться, что все личные данные на компьютере пользователя сохранены и надежно защищены. Кроме того, часто требуется сохранить все программы и настройки, чтобы вернуть компьютер в первоначальное состояние после выполнения ремонта.

Для этих целей используются различные методы хранения данных, в зависимости от характеристик клиентского компьютера и особенностей выполняемых задач. Например, для хранения резервной копии данных могут использоваться внутренние или внешние жесткие диски, USB-накопители, облачные хранилища или сетевые серверы.

Однако надежность и безопасность хранения данных в компьютерной мастерской зависит не только от выбранного метода, но и от качества использованных инструментов и программного обеспечения. Важно, чтобы мастерские имели достаточно мощные компьютеры и серверы для быстрого и эффективного резервного копирования, а также они должны использовать надежные антивирусные и антишпионские программы для защиты от вирусов и других опасных угроз.

## **Примеры компьютерных мастерских и БД-ных для хранения их данных**

В мире проблемы с компьютерной техникой это уже обыденность, поэтому существуют различные мастерские по починке и настройке ПК и ноутбуков различных как ценовых, так и специализированных категорий по всему миру. Например, в Твери одна из таких мастерских – IT Global сервис.

Эта компания, помимо сервиса, занимается так же и продажами различных комплектующих. Я связался с владельцем одного из магазинов, Александром, чтобы выяснить некоторые вопросы, интересующие меня.

Из нашего разговора я узнал, что данная компания пользуется отечественным ПО ASC CRM. Для них это наиболее удобная база данных.

Здесь так же используются таблицы с данными, как и в большинстве БД. Вот его слова на мой вопрос, какие сущности в БД они используют для своего сервис-центра: - «Данные о клиенте, данные о принятом устройстве и его неисправности, далее этапы работы. проведение диагностики, ожидание деталей, выполнение ремонта, устройство готово к выдаче, клиент оповещение о готовности. Это основной функционал, которым пользуется сервисный центр».

Как я понял, таблицы примерно следующие:

* Клиенты
* Устройства клиентов и их неисправности
* Список работ (или же услуг)
* Детали

Это основные таблицы, которым пользуется сервисный центр.

Таблица "Клиенты" обычно содержит информацию о клиентах, такую как их ФИО, адрес электронной почты, телефонные номера и другие данные контакта. Эта таблица может также содержать информацию о заказах клиента и его устройствах.

Таблица "Устройства клиентов и их неисправности" содержит информацию о устройствах клиентов, а также о неисправностях, которые были замечены на этих устройствах. Эта таблица может также содержать информацию о гарантийных условиях и сроках ремонта, в теории.

Таблица "Список работ" (или же услуг) обычно содержит информацию о том, какие работы (или услуги) предлагает сервисный центр, а также стоимость каждой из них.

Таблица "Детали" содержит информацию обо всех деталях, которые используются для ремонта устройств клиентов. В теории эта таблица может содержать информацию о производителях деталей, их стоимости и наличии в магазине.

Также он сказал, что в настоящее время без хранения данных на серверах, в Базах данных было бы намного тяжелее. Базы данных действительно облегчают хранение информации, особенно в каких-либо бизнесах, в области обучения, производства, медицинские учреждения.

В этой мастерской довольно индивидуальный подход к заказам клиентов – каждое комплектующее, которое необходимо для ремонта техники, привозится курьером на заказ, поэтому создавать отдельный склад не требуется. Мне кажется, это довольно разумный подход, дабы не захламлять склады ненужными товарами.

Проанализировав все необходимые данные, думаю я готов создавать собственную БД, на основе анализа.

# **Глава 2. Проектирование и разработка БД**

Ну вот теперь, когда изучена вся необходимая информация, можно приступать непосредственно к реализации собственной базы данных со всеми необходимыми данными, связями, ролями и т.д.

Начнём с легенды. У нас есть некая компьютерная мастерская. Допустим, она находится в каком-либо крупном городе, где есть конкуренты, что естественно. Там живет много людей, находится много складов производителей различных товаров, и множество магазинов техники.

У этой мастерской есть список клиентов, список мастеров, список заказов, список товаров на складе, система отзывов, а также список услуг.

Итак, давайте создадим ER-диаграмму для нашей БД.

Распишем сущности:

Таблица клиентов – Clients. У каждого из клиентов будет уникальный id, чтобы можно было быстро найти клиента. Он будет являться Уникальным Идентификатором, ключом – Primary key, и подставляться автоматически при добавлении Клиента, в чем нам поможет Auto\_Increment.

|  |
| --- |
| Clients |
| id INT |
| full\_name VARCHAR(255) |
| Phone INT |
| e-mail VARCHAR(255) |

Для клиентов необходимы ФИО, иначе как мы поймём, какой человек у нас заказал какую-либо работу? – full\_name.

Так же нужно поддерживать с клиентом связь, на случай каких-либо уточнений или непредвиденных обстоятельств. Берем обязательным Номер телефона – phone.

Также опционально возьмём e-mail человека – e-mail.

Id: Цифры, значит тип данных будет integer(INT)

full\_name: ФИО, значит тип данных – VARCHAR

phone: Цифры, значит тип данных - integer(INT)

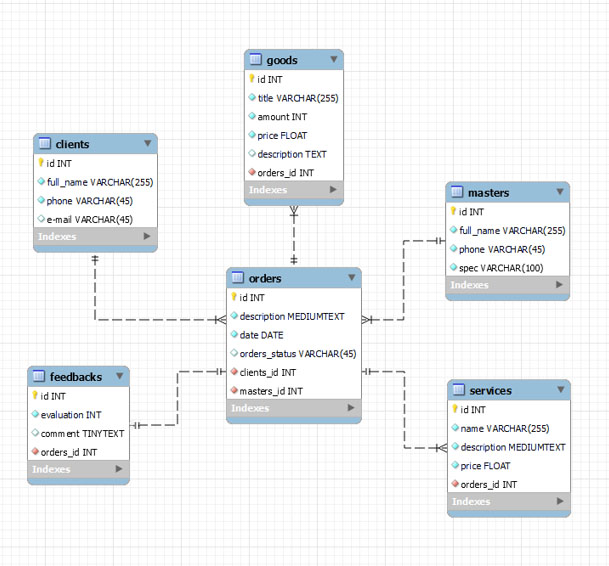
e-mail: Разные символы, значит тоже VARCHAR

Примерно такая таблица у нас получится.

Таким же образом проектируем и остальные таблицы.

После того, как мы закончили все 6 таблиц, нужно определиться со связями между ними. Основной таблицей у нас будет считаться Таблица заказов – Orders. Соответственно, все остальные таблицы будут взаимосвязаны с ней так или иначе.

В итоге у меня получилась следующая ERD:



Как можно видеть – везде применяются связи типа «один ко многим».

Логика связей следующая:

Клиент может сделать несколько заказов, соответственно связь «один клиент – несколько заказов». Та же ситуация и с мастерами, ведь один мастер может заниматься несколькими заказами.

К одному заказу может быть только один отзыв.

К одному заказу может быть применены несколько услуг (services). Также и с товарами, ведь для выполнения заказа может использоваться несколько, к примеру, комплектующих.

Далее нужно заполнить данными все таблицы, чтоб было с чем работать. Это можно сделать запросами вроде:

**INSERT** **INTO** `clients` (`full\_name`, `phone`, `e-mail`)

**VALUES** ('Иванов Иван Иванович','79105553355’, NULL),

('Иванов Андрей Сергеевич','79105553355', **NULL**),

('Краснов Владимир Дмитриевич','79101111212','hismail321@mail.ru'),

('Сидоренко Никита Михайлович','79157502525', **NULL**),

('Петрова Екатерина Александровна','79209516585', **NULL**),

('Васильев Денис Евгеньевич','78005553535','mymail123@gmail.com'),

('Петров Максим Петрович','76508591929','MadMax@gmail.com'),

('Шевченко Анастасия Олеговна','79105005454','ABCD\_123@yandex.ru'),

('Коваленко София Игоревна','79106900069', **NULL**),

('Третьяков Владимир Петрович','79157347907','nismo69@gmail.com')

;

Этим запросом мы заполнили таблицу клиентов. Почему нет указания id для каждого человека? А всё потому, что id имеет атрибут AUTO\_INCREMENT, которое само подставляет значения. Это облегчает работу с БД, уменьшает время написания кода.

Так же не каждый человек указывает электронную почту, ведь она не так обязательна, как номер телефона, на мой взгляд. Лично я не всегда указываю её, если это не обязательно, ведь лишние, к примеру, рекламные рассылки «засоряют» почту.

Так же заполним таблицу мастеров:

**INSERT** **INTO** **masters** (`full\_name`, `phone`, `spec`)

**VALUES** ('Куликова Анна Владимировна', **89356431651**, 'Сетевой Админ'),

('Суворов Андрей Михайлович', **89723034512**, 'Специалист по программному обеспечению'),

('Емельянова Алина Дмитриевна', **89167913247**, 'Техник-сборщик ПК'),

('Кутузов Борис Владимирович', **89104855384**, 'Мастер настроек'),

('Кузнецов Антон Степанович', **89921463295**, 'Техник-сборщик ПК'),

('Багданов Илья Александрович', **89496093078**, 'Техник-сборщик ПК'),

('Белова Оксана Евгеньевна', **89077073597**, 'Менеджер')

;

Таблицу отзывов:

**INSERT** **INTO** **feedbacks** (`evaluation`, `comment`, `orders\_id`)

**VALUES** (**10**, 'Сделали то, что мне и было нужно! Спасибо!', **1**),

(**7**, 'Возникли некоторые проблемы, но в итоге заказ был выполнен. Сойдет.', **5**),

(**10**, 'Сделали все быстро и качественно. Спасибо!', **7**),

(**8**, 'Забыли про термопасту... Пришлось обращаться еще раз. Слава богу, что исправили бесплатно и признали свою ошибку.', **9**),

(**1**, 'Очень долго тянули. Ну вот ооочень долго. Так еще и половину не выполнили того, о чём договорились, хотя была полная оплата. Разочаровали.', **11**)

;

Ну и таким же образом заполняем и остальные таблицы, в том числе довольно обширные. (Как пример – таблица goods. См. приложение 3).

## **Распределение ролей таблицы и создание представления (VIEW)**

После того, как мы заполнили таблицы, нужно распределить роли – кто и как сможет использовать таблицу

У меня их будет 3:

Admin – Полностью владеет функционалом БД, может делать с ней буквально все. Естественно, такой человек будет лишь один. Дополнительное представление, думаю, излишне.

Master – Частично владеет функционалом, не может удалять и добавлять таблицы, изменять структуру БД, а так же управлять ролями. Может добавлять данные в таблицу и изменять их.

Client – может лишь просматривать данные.

Создадим View для большего удобства наблюдения за данными.

Наше Представление будет выводить имя клиента, дату заказа и его описание, и так же отзыв об этом заказе. Так можно будет узнать, какой клиент оставил конкретный отзыв к конкретному заказу.

Я реализовал это следующим образом:

**CREATE** VIEW `clients\_feedbacks` **AS**

**SELECT**

`clients`.`full\_name` **AS** `full\_name`,

`orders`.`date` **AS** `date`,

`orders`.`description` **AS** `description`,

`feedbacks`.`comment` **AS** `comment`,

`feedbacks`.`evaluation` **AS** `evaluation`

**FROM**

((`clients`

**JOIN** `orders`)

**JOIN** `feedbacks`)

Так же для облегчения работы с БД было разработано две хранимые процедуры, по добавлению новых клиентов и новых работников.

Добавление клиентов:

DELIMITER //

**CREATE** **PROCEDURE** **insert\_user**(**IN** full\_name **VARCHAR**(**50**), **IN** email **VARCHAR**(**100**), **IN** phone **VARCHAR**(**45**))

BEGIN

**INSERT** **INTO** **clients** (full\_name, `e-mail`, phone) **VALUES** (full\_name, email, phone);

END //

DELIMITER ;

Добавление работников:

DELIMITER //

**CREATE** **PROCEDURE** **insert\_master**(**IN** full\_name **VARCHAR**(**255**), **IN** spec **VARCHAR**(**100**), **IN** phone **VARCHAR**(**45**))

BEGIN

**INSERT** **INTO** **masters** (full\_name, spec, phone) **VALUES** (full\_name, spec, phone);

END //

DELIMITER ;

После выполнения тестовых запросов по этим двум Хранимым процедурам, новые данные появились в соответствующих таблицах - они обе рабочие.

# **Заключение**

Разработка базы данных для компьютерной мастерской – это важный и необходимый процесс для эффективной работы и управления данными в компании.

В рамках данной работы была проведена аналитическая работа для определения требований и основных сущностей базы данных. Была разработана логическая и физическая модели базы данных с использованием современных методов проектирования MySQL.

# **Список источников:**

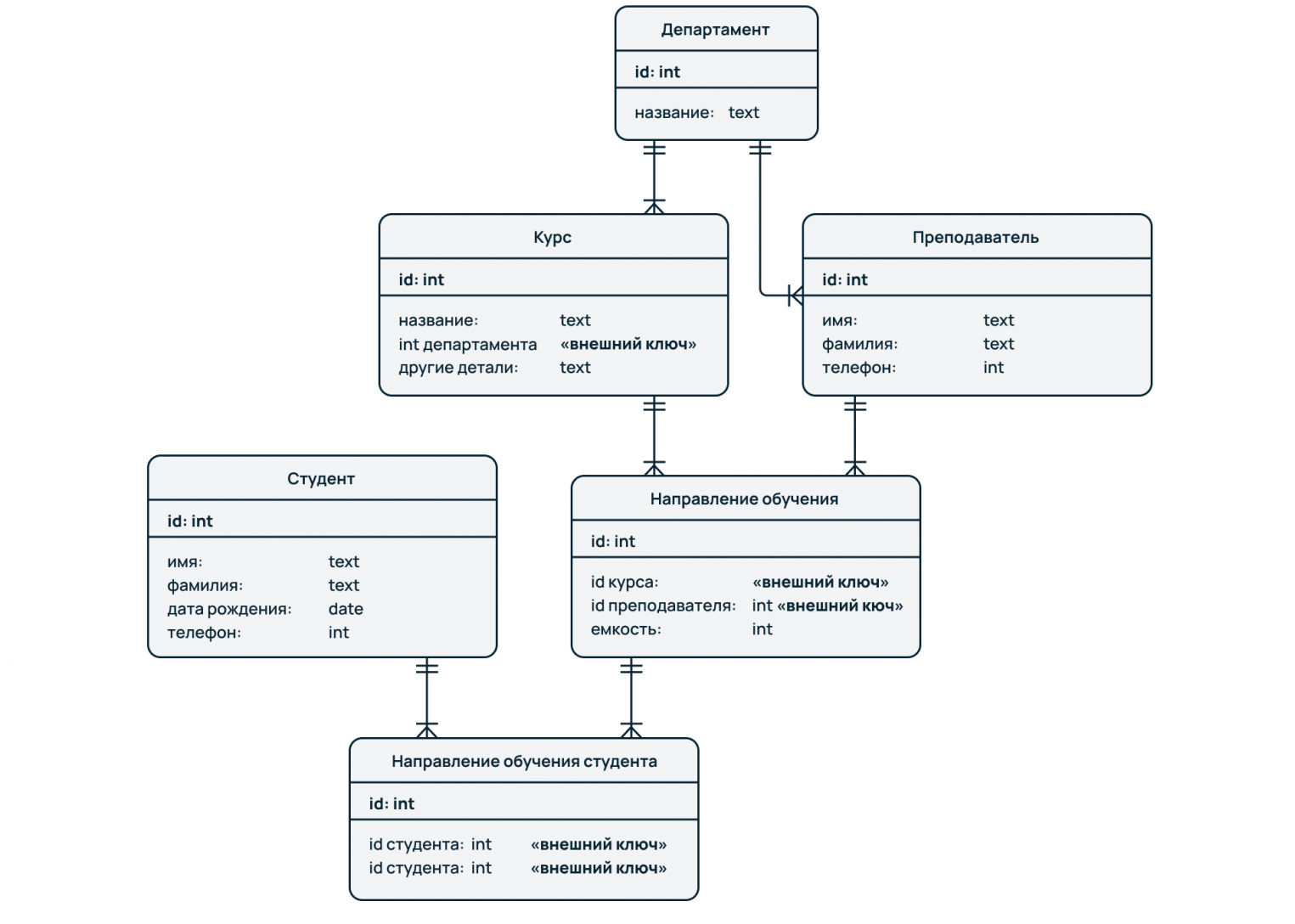
1. База данных [Электронный ресурс] / - Режим доступа: <https://ru.wikipedia.org/wiki/База_данных>
2. Роман Андреев. Виды баз данных [Электронный ресурс] / Роман Андреев – Режим доступа: <https://selectel.ru/blog/databases-types/>
3. Ткачев П.С. Нормализация баз данных [Электронный ресурс] / Ткачев П.С. – Режим доступа: <https://it.vshp.online/#/pages/mdk1101/mdk1101_lection_01>
4. System Development Corporation [Электронный ресурс] / - Режим доступа: <https://ru.wikipedia.org/wiki/System_Development_Corporation>
5. Урок по структуризации и проектированию баз данных [Электронный ресурс] / - Режим доступа: <https://www.lucidchart.com/pages/ru/проектирование-базы-данных>

# **Приложения**

Приложение 1. Ссылка на репозиторий проекта.



Приложение 2. Пример моделирования сложных взаимоотношений в РБД.



Приложение 3. Заполнение таблицы товаров.

**INSERT** **INTO** `goods` (`title`, `amount`, `price`, `description`, `orders\_id`)

**VALUES** ('Видеокарта GIGABYTE GeForce GT 1030',**4**,**6990**,'Видеокарта GigaByte GeForce GT 1030 [GV-N1030D5-2GL] с 2 ГБ видеопамяти типа GDDR5 способна значительно увеличить возможности домашней \nстанции или рабочего компьютера в сфере обработки графических данных. Она предусматривает подключение к интерфейсу PCI-E 3.0 и отличается малыми габаритами, которые позволят использовать ее \nв небольших системных блоках для настольного размещения. Обладая часовым энергопотреблением на уровне 35 Вт, карта может эффективно работать в системном блоке с мощностью БП 300+ Вт.\nДля стабильного отвода тепла в процессе работы графический процессор GigaByte GeForce GT 1030 [GV-N1030D5-2GL] оснащен активной воздушной системой охлаждения, включающей один осевой вентилятор. \nВзаимодействие с мониторами и проекторами в данной модели осуществляется с помощью портов HDMI, DVI-D, к которым можно одновременно присоединить до 2 устройств вывода графики.',**1**),

('Видеокарта MSI AMD Radeon RX 570 ARMOR OC 8G (8Gb)',**2**,**14499**.**9**,'Видеокарта MSI AMD Radeon RX 570 ARMOR OC [RX 570 ARMOR 8G OC] подойдет для игровых ПК. С ее помощью удается запускать различные игры и наслаждаться отличной \nграфикой. Вы сможете дополнить сборку и получить прекрасный компонент с высокой производительностью. Видеокарта MSI AMD Radeon RX 570 ARMOR OC [RX 570 ARMOR 8G OC] оснащается двумя вентиляторами. \nОни используются для охлаждения и позволяют поддерживать оптимальную температуру. Удается предотвратить перегрев под нагрузками. Модель поддерживает ряд основных стандартов – DirectX 12, OpenGL 4.5. Она оснащается видеочипом, \nизготовленным по технологическому процессу 14 нм. Элемент работает на частоте 1268 МГц. Объем видеопамяти большой – 8 ГБ. Этого должно хватить для подобной модели. Память принадлежит к типу GDDR5. Она работает на высокой частоте \n7000 МГц, обеспечивается пропускная способность 224 ГБ/с. Подключение к материнской плате происходит через интерфейс PCI-E 3.0.

...

('6 ТБ Жесткий диск Seagate SkyHawk',**6**,**10999**,'Жесткий диск Seagate SkyHawk [ST6000VX001] – модель запоминающего устройства с емкостью 6 ТБ, которая разработана для использования в составе систем видеонаблюдения. \nЖесткий диск способен обеспечить стабильность функционирования и отзывчивость в режиме непрерывного круглосуточного использования. Поддержка технологии RAID предусматривает объединение нескольких накопителей в единое хранилище. \nФирменное ПО позволяет осуществлять мониторинг и восстановление данных. Из других особенностей Seagate SkyHawk отмечаются ударостойкая конструкция и низкое потребление энергии.',**16**);