Содержание

[Введение 4](#_Toc469656191)

[1.Анализ литературных источников 5](#_Toc469656192)

[1.1Реляционная база данных 5](#_Toc469656193)

[1.2 CRM- системы 7](#_Toc469656194)

[1.3 MySQL 8](#_Toc469656195)

[1.4 MySQL Workbench 10](#_Toc469656196)

[2.Формирование требований 13](#_Toc469656197)

[2.1 Требования к проектируемой базе данных 13](#_Toc469656198)

[2.2 Разработка функциональных требований 13](#_Toc469656199)

[3 Проектирование базы данных 15](#_Toc469656200)

[3.1 Описание предметной области 15](#_Toc469656201)

[3.1.1 Назначение и область применения 15](#_Toc469656202)

[3.1.2 Предполагаемые пользователи 15](#_Toc469656203)

[3.2 Построение инфологической модели 15](#_Toc469656204)

[3.2.1 Выделение объектов предметной области 15](#_Toc469656205)

[3.2.2 Выделение атрибутов объектов предметной области 15](#_Toc469656206)

[3.2.3 Выделение атрибутов-кандидатов в первичные ключи и назначение первичных ключей 16](#_Toc469656207)

[3.2.4 Описание модели в форме таблиц 17](#_Toc469656208)

[4 Создание базы данных 19](#_Toc469656209)

[4.1 Построение модели базы данных 19](#_Toc469656210)

[4.2 Генерация SQL 21](#_Toc469656211)

[4.3 Написание SQL запросов 24](#_Toc469656212)

[Заключение 26](#_Toc469656213)

[Список используемой литературы 27](#_Toc469656214)

[Приложение 28](#_Toc469656215)

# 

# **Введение**

В современном бизнесе необходимость автоматизация различных процессов стала уже привычным явлением. Уже становится сложно представить себе складской или бухгалтерский учет без применения специализированного программного обеспечения, торговые представители используют специальные приложения для оформления и отправки заказа в офис прямо с плашнета или мобильного телефона, достаточно большая часть заказов приходит с сайта уже в виде готовых к обработке документов. Но при этом взаимоотношения с клиентами, по крайней мере, в среднем и малом бизнесе, почему-то очень часто ведутся без внедрения автоматизации и достаточного внимания к учету.

Каждый менеджер по продажам работает так, как ему удобнее, ведет фиксацию звонков, других видов взаимодействия с клиентами по собственному усмотрению: кто-то – на бумаге, кто-то – в Excel таблицах, а кто-то вообще не считает нужным фиксировать процесс своей работы.  
  
 Входящие звонки или заявки с сайта от новых заказчиков также не фиксируются, зачастую даже сложно понять, кто из менеджеров занимается входящей заявкой. В результате реальный учет ведется только на уровне оплаченных заказов и отгрузки товара. А насколько эффективно работает отдел продаж и проводится ли какая-то работа с уже имеющимися контактами, определить оказывается невозможно.

Выходом из данной ситуации является внедрение в компании CRM-системы (от англ. Customer Relationship Management), что в переводе на русский язык звучит как «управление взаимоотношениями с клинетами». Одним из ключевых компонентов таких систем является база данных, которая используется для хранения данных.

В данном курсовом проекте была поставлена задача: разработать базу данных для CRM-системы предприятия, оказывающего бухгалтерские услуги. Она должна позволять хранить данные о клиентах, работниках, истории переписки, встреч, телефонных разговора с клиентами, шаблоны основных документов.

# 

# **1.Анализ литературных источников**

## **1.1Реляционная база данных**

Реляционная база данных — это совокупность взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Строка таблицы содержит данные об одном объекте (например, товаре, клиенте), а столбцы таблицы описывают различные характеристики этих объектов — атрибутов (например, наименование, код товара, сведения о клиенте). Записи, т. е. строки таблицы, имеют одинаковую структуру — они состоят из полей, хранящих атрибуты объекта. Каждое поле, т. е. столбец, описывает только одну характеристику объекта и имеет строго определенный тип данных. Все записи имеют одни и те же поля, только в них отображаются различные информационные свойства объекта.

В реляционной базе данных каждая таблица должна иметь первичный ключ — поле или комбинацию полей, которые единственным образом идентифицируют каждую строку таблицы. Если ключ состоит из нескольких полей, он называется составным. Ключ должен быть уникальным и однозначно определять запись. По значению ключа можно отыскать единственную запись. Ключи служат также для упорядочивания информации в БД.

Таблицы реляционной БД должны отвечать требованиям нормализации отношений. Нормализация отношений — это формальный аппарат ограничений на формирование таблиц, который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудозатраты на ведение базы данных.

Пусть создана таблица Студент, содержащая следу-рэщие поля: № группы, ФИО, № зачетки, дата рождения, шазвание специальности, название факультета. Такая организация хранения информации будет иметь ряд недостатков:

* дублирование информации (наименование специальности и факультета повторяются для каждого студента), следовательно, увеличится объем БД;
* процедура обновления информации в таблице затрудняется из-за необходимости редактирования каждой записи таблицы.

Нормализация таблиц предназначена для устранения этих недостатков. Имеется три нормальные формы отношений.

Первая нормальная форма. Реляционная таблица приведена к первой нормальной форме тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто. Так, если из таблицы Студент требуется получать сведения по имени студента, то поле ФИО следует разбить на части Фамилия, Имя, Отчество.

Вторая нормальная форма. Реляционная таблица задана во второй нормальной форме, если она удовлетворяет требованиям первой нормальной формы и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом. Чтобы привести таблицу ко второй нормальной форме, необходимо определить функциональную зависимость полей. Функциональная зависимость полей — это зависимость, при крторой в экземпляре информационного объекта определенному значению ключевого реквизита соответствует только одно значение описательного реквизита.

Третья нормальная форма. Таблица находится в третьей нормальной форме, если она удовлетворяет требованиям второй нормальной формы, ни одно из ее неключевых полей не зависит функционально от любого другого неключевого поля. Например, в таблице Студент (№ группы, ФИО, № зачетной книжки, Дата рождения, Староста) три поля — № зачетной книжки, № группы, Староста находятся в транзитивной зависимости. № группы зависит от № зачетной книжки, а Староста зависит от № группы. Для устранения транзитивной зависимости необходимо часть полей таблицы Студент перенести в другую таблицу Группа. Таблицы примут следующий вид: Студент (№ группы, ФИО, № зачетной книжки, Дата рождения), Группа (№ группы, Староста).

Над реляционными таблицами возможны следующие операции:

* Объединение таблиц с одинаковой структурой. Результат— общая таблица: сначала первая, затем вторая (конкатенация).
* Пересечение таблиц с одинаковой структурой. Результат — выбираются те записи, которые находятся в обеих таблицах.
* Вычитание таблиц с одинаковой структурой. Результат — выбираются те записи, которых нет в вычитаемом.
* Выборка (горизонтальное подмножество). Результат — выбираются записи, отвечающие определенным условиям.
* Проекция (вертикальное подмножество). Результат — отношение, содержащее часть полей из исходных таблиц.
* Декартово произведение двух таблиц Записи результирующей таблицы получаются путем объединения каждой записи первой таблицы с каждой записью другой таблицы.

Реляционные таблицы могут быть связаны друг с другом, следовательно, данные могут извлекаться одновременно из нескольких таблиц. Таблицы связываются между собой для того, чтобы в конечном счете уменьшить объем БД. Связь каждой пары таблиц обеспечивается при наличии в них одинаковых столбцов.

Существуют следующие типы информационных связей:

* один-к-одному;
* один-ко-многим;
* многие-ко-многим.

Связь один-к-одному предполагает, что одному атрибуту первой таблицы соответствует только один атрибут второй таблицы и наоборот.

Связь один-ко-многим предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы.

Связь многие-ко-многим предполагает, что одному атрибуту первой таблицы соответствует несколько атрибутов второй таблицы и наоборот.

## **1.2 CRM- системы**

**Система управления взаимоотношениями с клиентами** (**CRM**, **CRM-система**, сокращение от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Customer Relationship Management*) — [прикладное программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для организаций, предназначенное для [автоматизации](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) стратегий взаимодействия с [заказчиками](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%B0%D0%B7%D1%87%D0%B8%D0%BA) ([клиентами](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%B0%D0%B7%D1%87%D0%B8%D0%BA)), в частности для повышения уровня продаж, оптимизации [маркетинга](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D0%BA%D0%B5%D1%82%D0%B8%D0%BD%D0%B3) и улучшения обслуживания клиентов путём сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) и последующего анализа результатов.

CRM — модель взаимодействия, основанная на постулате, что центром всей философии бизнеса является клиент, а главными направлениями деятельности компании являются меры по обеспечению эффективного маркетинга, [продаж](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D1%8F%D0%BC%D1%8B%D0%B5_%D0%BF%D1%80%D0%BE%D0%B4%D0%B0%D0%B6%D0%B8) и обслуживания клиентов. Поддержка этих бизнес-целей включает сбор, хранение и анализ информации о потребителях, поставщиках, партнёрах, а также о внутренних процессах компании. Функции для поддержки этих бизнес-целей включают продажи, маркетинг, [поддержку](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D1%80%D0%B6%D0%BA%D0%B0) потребителей.

CRM-система может включать:

* фронтальную часть, обеспечивающую обслуживание клиентов на точках продаж с автономной, распределенной или централизованной обработкой информации;
* операционную часть, обеспечивающую авторизацию операций и оперативную отчётность;
* [хранилище данных](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BB%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85);
* аналитическую подсистему;
* распределенную систему поддержки продаж: [реплики](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) данных на точках продаж или [смарт-карты](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82-%D0%BA%D0%B0%D1%80%D1%82%D0%B0).

**Основные принципы**

* Наличие единого хранилища информации, куда собираются сведения о взаимодействии с клиентами — [клиентской базы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%81%D0%BA%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0).
* Использование многих каналов взаимодействия: обслуживание на точках продаж, телефонные звонки, [электронная почта](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0), мероприятия, встречи, регистрационные формы на веб-сайтах, рекламные ссылки, [чаты](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82), [социальные сети](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_(%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)).
* Анализ собранной информации о клиентах и подготовка данных для принятия соответствующих организационных решений — например, сегментация клиентов на основе их значимости для компании, потенциальном отклике на те или иные [промоакции](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D0%BE%D0%B0%D0%BA%D1%86%D0%B8%D1%8F), прогнозе потребности в тех или иных продуктах компании.

Основной целью внедрения, как правило, ставится увеличение степени удовлетворённости клиентов за счёт анализа накопленной информации о клиентском поведении, регулирования тарифной политики, настройки инструментов маркетинга. Благодаря применению автоматизированной централизованной обработки данных появляется возможность эффективно и с минимальным участием сотрудников учитывать индивидуальные потребности заказчиков, а за счёт оперативности обработки — осуществлять раннее выявление рисков и потенциальных возможностей.

В торговой сфере за счёт CRM обеспечивается более эффективное применение метода [перекрёстных продаж](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B5%D1%80%D0%B5%D0%BA%D1%80%D1%91%D1%81%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B4%D0%B0%D0%B6%D0%B0&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [*cross-selling*](https://en.wikipedia.org/wiki/cross-selling)) и техники [апсейла](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D1%81%D0%B5%D0%B9%D0%BB).

## **1.3 MySQL**

**MySQL** — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года Sun Microsystems приобрела MySQL AB за 1 млрд долларов, 27 января 2010 года Oracle приобрела Sun Microsystems за 7,4 млрд долларов и включила MySQL в свою линейку СУБД.

Сообществом разработчиков MySQL созданы различные ответвления кода, такие как Drizzle (англ.), OurDelta, Percona Server и MariaDB. Все эти ответвления уже существовали на момент поглощения компании Sun корпорацией Oracle. [1]

*О происхождении MySQL*

MySQL возникла как попытка применить mSQL к собственным разработкам компании: таблицам, для которых использовались ISAM — подпрограммы низкого уровня. В результате был выработан новый SQL-интерфейс, но API-интерфейс остался в наследство от mSQL. Откуда происходит название «MySQL» — доподлинно неизвестно. Разработчики дают два варианта: либо потому, что практически все наработки компании начинались с префикса My, либо в честь девочки по имени My, дочери Майкла Монти Видениуса, одного из разработчиков системы.

Логотип MySQL в виде дельфина носит имя «Sakila». Он был выбран из большого списка предложенных пользователями «имён дельфина». Имя «Sakila» было отправлено Open Source-разработчиком Ambrose Twebaze.

*Лицензирование*

MySQL имеет двойное лицензирование. MySQL может распространяться в соответствии с условиями лицензии GPL. Однако по условиям GPL, если какая-либо программа использует библиотеки (или включает в себя другой GPL-код) MySQL, то она тоже должна распространяться по лицензии GPL. Это может расходиться с планами разработчиков, не желающих открывать исходные тексты своих программ. Для таких случаев предусмотрена коммерческая лицензия, которая также обеспечивает качественную сервисную поддержку. Для свободного программного обеспечения Oracle предоставляет отдельное исключение из правил, явным образом разрешающее использование и распространение MySQL вместе с ПО, распространяемым под лицензией из определённого Oracle списка. [1]

*Платформы*

MySQL портирована на большое количество платформ: AIX, BSDi, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Server 2003, WinCE, Windows Vista, Windows 7 и Windows 10. Существует также порт MySQL к OpenVMS. Важно отметить, что на официальном сайте СУБД для свободной загрузки предоставляются не только исходные коды, но и откомпилированные и оптимизированные под конкретные операционные системы готовые исполняемые модули СУБД MySQL. [1]

*Языки программирования*

MySQL имеет API для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентный Паскаль и Tcl, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

MyODBC представляет собой драйвер ODBC (2.50) уровня 0 (с некоторыми возможностями уровней 1 и 2) для подсоединения совместимого с ODBC приложения к MySQL. MyODBC работает на всех системах Microsoft Windows и на большинстве платформ Unix. [1]

## **1.4 MySQL Workbench**

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL. Является преемником DBDesigner 4 от FabForce.

MySQL Workbench предлагается в двух редакциях:

* Community Edition — распространяется под свободной лицензией GNU GPL;
* Standard Edition — доступна по ежегодной оплачиваемой подписке. Эта версия включает в себя дополнительные функции, которые повышают производительность разработчиков и администраторов БД.

Поддерживаемые платформы:

* Windows 7
* Windows XP
* Mac OS X 10.5+
* Ubuntu 10.04 32bit/64bit , 12.04 32/64bit.
* Fedora 12 32bit/64bit
* Oracle Enterprise Linux 6 32bit/64bit
* Red Hat Enterprise Linux 6 32bit/64bit
* Gentoo [3]

*GUI Tools*

Пакет MySQL GUI Tools содержит кросс-платформенное свободное программное обеспечение (под лицензией GNU GPL) для администрирования серверов баз данных MySQL, а также для создания и манипулирования данными внутри баз данных MySQL. Данный пакет разрабатывался Sun Microsystems, но позже его разработка была остановлена, и сейчас он доступен только из архивов загрузок на сайте MySQL.

Пакет GUI Tools был полностью заменён MySQL Workbench начиная с бета-версии 5.2. Тем не менее, MySQL Support Team продолжала оказывать поддержку пакета GUI Tools до 30 июня 2010 г.[5] В будущих версиях MySQL Workbench будет добавлен плагин для миграции (англ. migration plugin), полностью совместимый с MySQL Migration Toolkit, входящий в состав пакета GUI Tools. [3]

*Workbench*

Первая версия MySQL Workbench была выпущена в сентябре 2005 года.

MySQL Workbench был первым семейством продуктов, который был доступен в двух вариантах. Чтобы привлечь разработчиков в основную команду разработки, коммерческая стандартная версия программы (англ. Standard Edition) предлагается поверх свободной версии (англ. Community Edition), распространяемой под лицензией GNU GPL.[7] «Community Edition» является полнофункциональным продуктом, обладающим всеми основными возможностями коммерческого варианта. Являясь основой для всех будущих релизов, он будет получать пользу от всех будущих усилий, прилагаемых для развития продукта. «Standart Edition» расширяет «Community Edition» серией модулей и плагинов, позволяющих оптимизировать рабочий процесс и, тем самым, сэкономить время и избежать ошибок.[3]

*Возможности программы:*

* Позволяет наглядно представить модель базы данных в графическом виде.
* Наглядный и функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей.
* Reverse Engineering — восстановление структуры таблиц из уже существующей на сервере БД (связи восстанавливаются в InnoDB, при использовании MyISAM — связи необходимо устанавливать вручную).
* Удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы.
* Возможность редактирования данных в таблице в визуальном режиме. [3]

# 

# **2.Формирование требований**

## **2.1 Требования к проектируемой базе данных**

При проектировании базы данных необходимо учитывать тот факт, что база данных должна удовлетворять комплексу требований. Эти требования следующие:

1. целостность базы данных – требование полноты и непротиворечивости данных;
2. возможность пользователю возможность создавать новые БД и определять их схему (логическую структуру данных) с помощью специального языка — языка определения данных; поддерживать разнообразные представления одних и тех же данных;
3. возможность многократно использовать данные;
4. возможность осуществлять быстрый поиск и получение информации по запросам пользователей
5. возможность осуществлять обновление данных
6. Возможность интегрировать базу данных в другие приложения

База должна быть спроектирована при использовании реляционной СУБД и разбита на таблицы, связанные между собой, в соответствии с третьей нормальной формой.

## **2.2 Разработка функциональных требований**

Разрабатываемая база данных должна содержать следующую информацию:

1. Список клиентов и информацию о них
2. Список работников предприятия
3. Список договоров
4. Список протоколов об оказании услуг
5. Информацию о банковских счетах клиентов
6. Информацию об оказанных услугах
7. Историю переписки между клиентом и работником по email
8. Историю встреч между клиентом и работником
9. Историю звонков между клиентом и работником
10. Список предъявляемых услуг
11. Список цен за предъявляемые услуги
12. Список генерируемых документов
13. Список шаблонов для генерация документов

# **3 Проектирование базы данных**

## **3.1 Описание предметной области**

### **3.1.1 Назначение и область применения**

Предметная область представляет собой БД для CRM-системы фирмы осуществляющей бухгалтерские услуги. Эта БД является необходимым компонентом в формировании учета сотрудников, клиентов, договоров и всего того, из чего состоит структура бухгалтерской фирмы.

### **3.1.2 Предполагаемые пользователи**

Проектируемая база данных в первую очередь предназначена для работников и руководства фирмы. С помощью этой БД есть возможность просматривать список работников, клиентов и их документы, оказываемых услуг, список автоматически генерируемых документов и соответствующих им шаблонов генерации, историю общения с клиентом.

Также возможен частичный доступ клиентам. В базе хранится выставленные счета и документы, закрепленные за каждым клиентом. С помощью этой БД, клиентом сможет просмотреть выставленные ему на оплату счета, а также список документов.

## **3.2 Построение инфологической модели**

### **3.2.1 Выделение объектов предметной области**

Выделение объектов производится из описания предметной области. Выделены следующие объекты: клиент, работник фирмы, договор, протокол об оказании услуг, услуга, договор, шаблон документа.

Это не все объекты, которые будут использованы в базе данных, а лишь только основные.

### **3.2.2 Выделение атрибутов объектов предметной области**

* Клиент. Содержит: id клиента, название, УНП, ОКПО, id руководителя, id банка обслуживания, id типа клиента .
* Работник. Содержит: id работника, фамилия, имя, отчество, дату рождения, логин, пароль.
* Услуга. Содержит: id услуги, название, id цены .
* Договор. Содержит: id договора, номер, название, дата начала, дата окончания, id клиента, id работника, id шаблона документа.
* Протокол об оказании услуг. Содержит: id протокола, дата заключения, номер протокола, id договора(по которому выставляется договор).
* Шаблон документа. Содержит: id шаблона, назавние шаблона, имя файла, тип шаблона.

### **3.2.3 Выделение атрибутов-кандидатов в первичные ключи и назначение первичных ключей**

|  |  |  |  |
| --- | --- | --- | --- |
| Объект предметной области | Кандидаты в первичные ключи | Назначенный первичный ключ | примечание |
| Клиент | Фамилия,  Имя, Отчество | ID\_ клиента | Введен суррогатный ключ |
| Работник | Фамилия,  Имя, Отчество | ID\_работника | Введен суррогатный ключ |
| Услуга | Название услуги | ID\_услуги | Введен суррогатный ключ |
| Договор | Номер договора, названия договора | ID\_договора | Введен суррогатный ключ |
| Протокол | Номер протокола | ID\_протокола | Введен суррогатный ключ |
| Шаблон документа | Название шаблона, имя файла | ID\_шаблона | Введен суррогатный ключ |

### 

### **3.2.4 Описание модели в форме таблиц**

Клиент

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Название** | **УНП** | **ОКПО** | **Id руководителя** | **Id банковской информации** |
| 1 | ООО “Мидивисана” | 192668964 | 382925255000 | 3 | 2 |
| 2 | ОАО “Газпром” | 192668962 | 382925254000 | 5 | 3 |

Сотрудник школы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Фамилия** | **Имя** | **Отчество** | **ID должности** | **Дата рождения** | **Логин** | **Пароль** |
| 1 | Иванов | Сергей | Иванович | 1 | 28-09-1972 | fox | 12365 |
| 2 | Коток | Иван | Владимирович | 2 | 28-09-1973 | cat | 45789 |

Услуга

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Название** | **Тип** | **Цена** |
| 1 | Оформление договора | Единовременная | 10 |
| 2 | Подсчёт бухгалтерии | Почасовая | 1 |

Договор

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Номер** | **Префикс** | **Id тип класса** | **ID аудитория** |
| 1 | 6 | А | 1 | 65 |
| 2 | 10 | Б | 2 | 14 |

Протокол

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Номер** | **Дата** | **Id договора** | **Id скидки** |
| 1 | 1 | 30.12.2016 | 4 | 102 |
| 2 | 1 | 14.10.2016 | 3 | 204 |

Шаблон документа

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Имя** | **Имя файла** | **Id тип шаблона** |
| 1 | 3 | 30.06.2016 | 1 |
| 2 | 4 | 14.06.2016 | 2 |

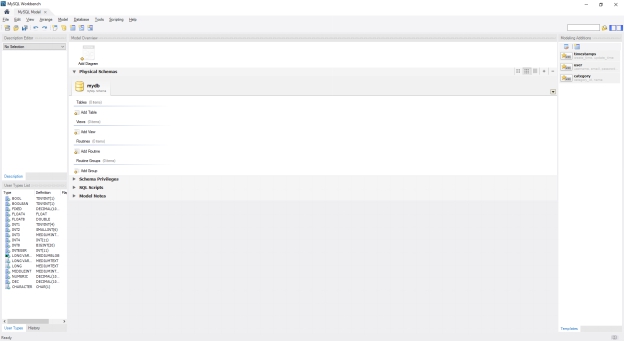
# **4 Создание базы данных**

## **4.1 Построение модели базы данных**

Для построения модели базы данных и генерации скрипта, создающий базу данных, я использовал MySQL Workbench.

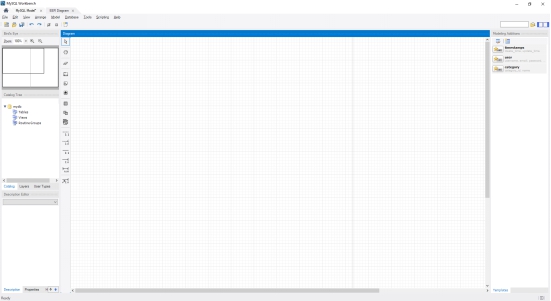
При создании модели я буду опираться на описанную инфологическую модель, а также придерживаться требованиям, которые изложены ранее в пункте 2.

Для создания модели жмем File -> New Model. После чего, на появившемся окне жмём кнопку “Add Diagram” и добавляем новую диаграмму (Рисунок 1).



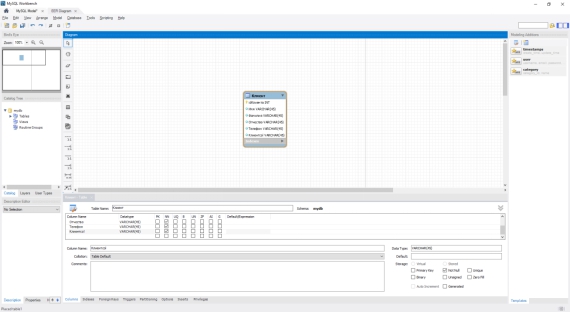
**Рисунок 1** - **Создание новой модели в MySQL Workbench**

Далее появляется редактор диаграмм. В левом крае рабочего окна находится панель инструменты для создания полноценной диаграммы: таблицы, связи, представления и т.д. (Рисунок 2)



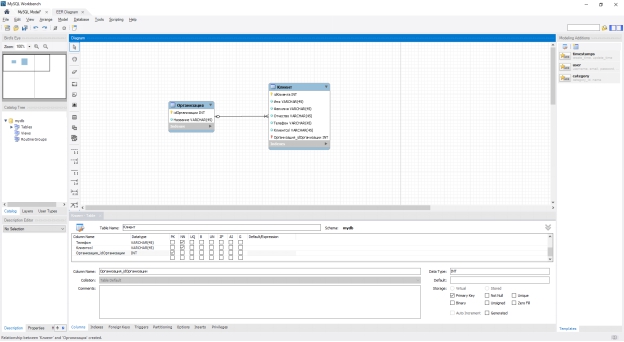
**Рисунок 2** - **Редактор диаграмм**

Для того, чтобы добавить новую таблицу, нужно перетянуть значок таблицы в поле редактора таблиц. После чего его можно задать поля, которые он будет содержать, а также указать индексы. (Рисунок 3)

****

**Рисунок 3** - **Таблица «Клиент»**

Далее, создаем новую таблицу, задаём поля. После этого нужно сделать связь между двумя таблицами и определиться, какая из них будет дочерняя, а какая родительская. После установки связи между таблицами можно увидеть, как в дочерней появился внешний(мигрирующий) ключ, который ссылается на родительскую таблицу.



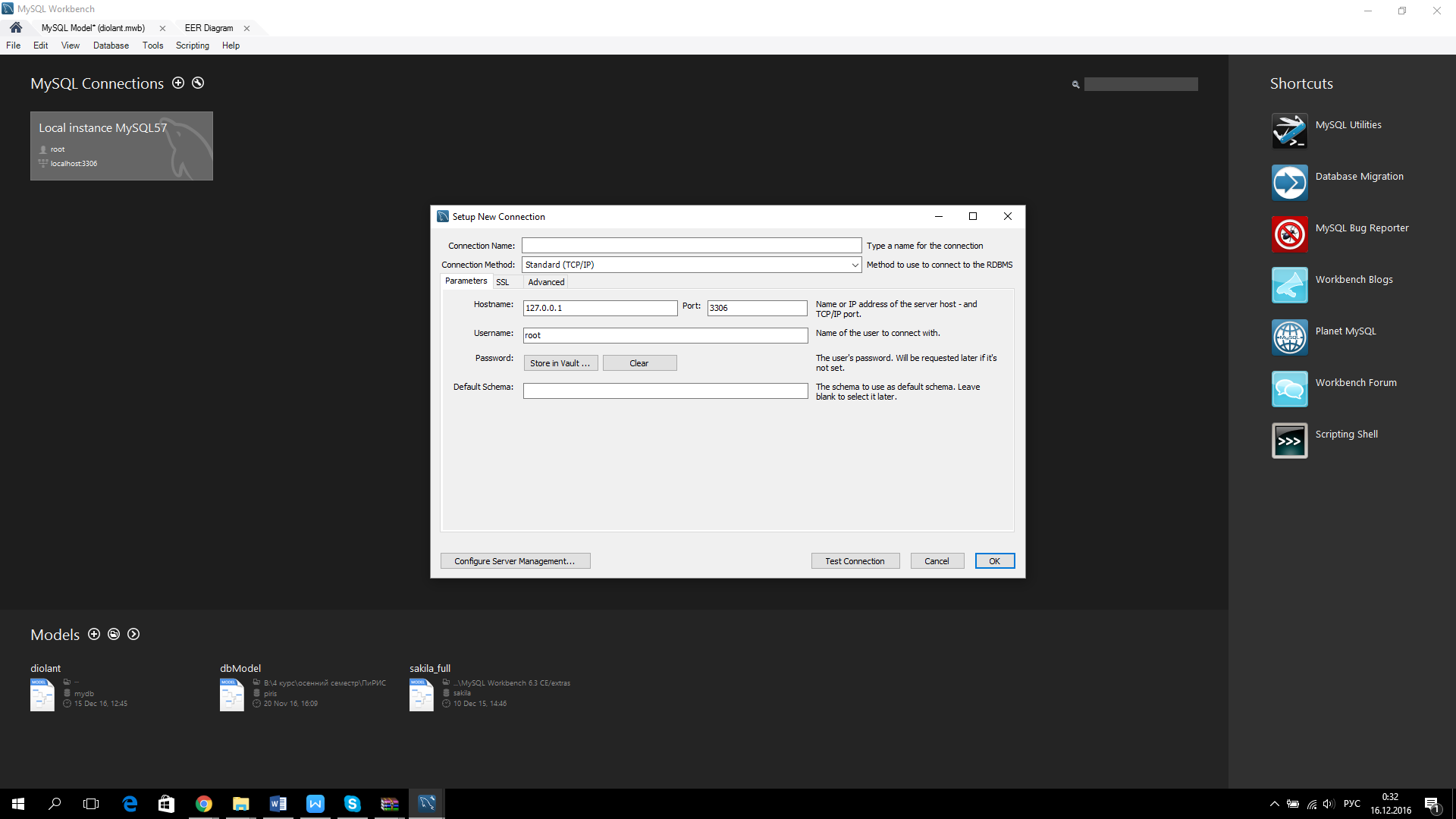
**Рисунок 4** -**Таблицы и связи между ними**

Внешние ключи мигрируют от изначальной таблицы, до последней, которая входит в данную “цепочку” таблиц, из-за этого в некоторых таблицах приходится удалять мигрирующие ключи вручную.

Используя эти инструменты, создадим полную диаграмму. Результат представлен в приложении 1.

## **4.2 Генерация SQL**

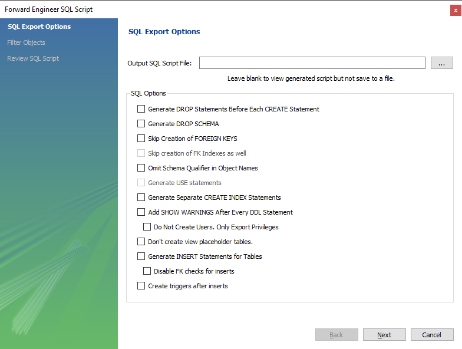
Перед тем как сгенерировать скрипт которй создаст базу данных требуется подключить MySQL Server к MySQL WorkBench. Для этого требуется в стартовом окне нажать кнопку “+”, после этого указать адрес сервера и пароль и нажать кнопку “Ок”.



**Рисунок 6 – Подключение MySQL сервера**

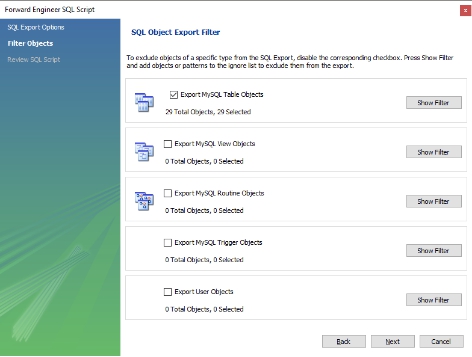
Для генерации SQL запроса по созданию полученной базы данных воспользуемся стандартным средством MySQL Workbench : Forward Engineer SQL Script. Для этого:

1. Жмем File -> Export -> Forward Engineer (Рисунок 7)



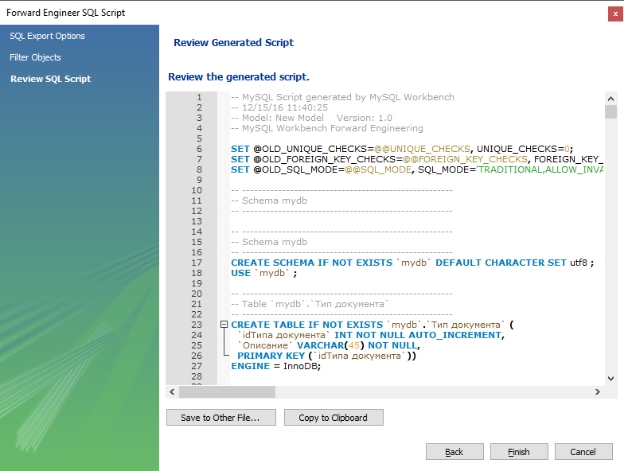
**Рисунок 7** - **Forward Engineer SQL Script**

1. Далее выбираем объекты, которые хотим чтобы были в сгенерированном SQL запросе (Рисунок 8)



**Рисунок 8** - **Forward Engineer SQL Script**

1. Получение готового запроса (Рисунок 9)



**Рисунок 9** - **Запрос для создания БД**

После всех проделанных действий этот запрос нужно нажать кнопку “Finish” и база данных будет создана.

## **4.3 Написание SQL запросов**

SQL запросы обычно выполняют следующие задачи:

* добавление новых записей в таблицы базы данных;
* изменение записей в таблицах базы данных;
* выборка записей из таблиц базы данных;
* удаление записей из базы данных.

В связи с этим существует четыре основных типа запросов данных в SQL, которые относятся к языку манипулирования данными (Data Manipulation Language):

* INSERT - добавить строки в таблицу;
* UPDATE - изменить строки в таблице;
* DELETE - удалить строки в таблице;
* SELECT - выбрать строки из таблиц;

Работа с базой данных осуществляется с помощью окна запросов MySQL Workbench:

* Добавление
  + INSERT INTO `mydb`.`bank` (`idbank`, `name`, `adress\_idadress`) VALUES ('1', 'ВТБ', 'г.Минск ул.Гикало д.2');
  + INSERT INTO `mydb`.`country` (`country\_code`, `name`) VALUES ('231', 'Республика Беларусь');
  + INSERT INTO `mydb`.`client` (`name`, `type\_client\_id\_type\_client`, `unp`, `okpo`, `bank\_inf\_checking\_account`, `manager\_idmanager`, `manager\_position\_idposition`) VALUES ('ОАО \"Газпром\"', '1', '192668962', '382925254000', '24', '23', '2');
  + INSERT INTO `mydb`.`services` (`name`, `type`, `price\_integer\_part`) VALUES ('Подсчёт налогов за месяц', 'PER\_HOURS', '1');
* Изменение
  + UPDATE `mydb`.`bank` SET `, `adress\_idadress` = 'г.Минск ул.Сурганова д.2' WHERE `idbank` = 1;
  + UPDATE `mydb`.`country` SET `name`=’Австрия’ WHERE `country\_code` = 32;
* Удаление
  + DELETE FROM `mydb`.`country` WHERE `country\_code` = 32;
* Выборка
  + SELECT `email\_text`, workers.`name`, workers.`surname`, client.`name` FROM mydb.email\_history JOIN mydb.workes ON email\_hisory.`id\_worker` = workers.`id\_worker` JOIN client ON email\_history.`client.idClient` = client.idClient;SELECT `название` FROM `город`
  + SELECT idmails, mail, client\_idclient FROM mydb.mails;

# **Заключение**

В результате выполнения курсовой работы была спроектирована база данных CRM-системы фирмы, осуществляющей бухгалтерские услуги. Была изучена теория нормализации реляционных баз данных, закреплены теоретические и практические знания по курсу дисциплины.

В спроектированной базе заложены следующие возможности:

* Хранение и просмотр списка клиентов и информацию о них
* Хранение и просмотр списка работников предприятия
* Хранение и просмотр списка договоров
* Хранение и просмотр списка протоколов об оказании услуг
* Хранение и просмотр информации о банковских счетах клиентов
* Хранение и просмотр информации об оказанных услугах
* Хранение и просмотр истории переписки между клиентом и работником по email
* Хранение и просмотр истории встреч между клиентом и работником
* Хранение и просмотр истории звонков между клиентом и работником
* Хранение и просмотр списка предъявляемых услуг
* Хранение и просмотр списка цен за предъявляемые услуги
* Хранение и просмотр генерируемых документов
* Хранение и просмотр шаблонов для генерации документов

Работа была разделена на этапы, такие как анализ литературных источников, формирование требований, проектирование базы данных, создание базы данных, создание SQL-запросов. После последовательного выполнения вышеперечисленных этапов разработки, была получена исправно работающая база данных.

# **Список используемой литературы**

1.MySQL Workbench [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/MySQL\_Workbench – Дата доступа: 12.11.2016.

2.MySQL [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/MySQL – Дата доступа: 14.11.2016.

3.MySQL [Электронный ресурс]. Режим доступа: <http://www.mysql.ru/docs/man/> – Дата доступа: 14.11.2016.

4.Карпова Т.С. Базы данных. Модели, разработка, реализация/СПб.: Питер,2002. – 304 с.

5. Ханcен Г., Ханcен Д. Базы данных. Разработка и управление / М.: Бином, 1999 – 704 с.

6.Куликов С.С. Работа с MySQL, MS SQL Server и Oracle в примерах

# 

# **Приложение**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`type\_client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`type\_client` (

`id\_type\_client` INT NOT NULL AUTO\_INCREMENT,

`form` VARCHAR(45) NOT NULL,

`type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_type\_client`),

INDEX `uniq\_type` (`form` ASC, `type` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`country`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`country` (

`country\_code` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`country\_code`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`city`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`city` (

`country\_country\_code` INT NOT NULL,

`index` VARCHAR(45) NOT NULL,

INDEX `fk\_city\_country1\_idx` (`country\_country\_code` ASC),

PRIMARY KEY (`index`),

CONSTRAINT `fk\_city\_country1`

FOREIGN KEY (`country\_country\_code`)

REFERENCES `mydb`.`country` (`country\_code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`adress`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`adress` (

`idadress` INT NOT NULL,

`city\_index` VARCHAR(45) NOT NULL,

`street` VARCHAR(45) NOT NULL,

`number\_house` VARCHAR(5) NOT NULL,

PRIMARY KEY (`idadress`),

INDEX `fk\_adress\_city1\_idx` (`city\_index` ASC),

CONSTRAINT `fk\_adress\_city1`

FOREIGN KEY (`city\_index`)

REFERENCES `mydb`.`city` (`index`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`bank`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`bank` (

`idbank` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(40) NOT NULL,

`adress\_idadress` INT NOT NULL,

PRIMARY KEY (`idbank`),

INDEX `fk\_bank\_adress1\_idx` (`adress\_idadress` ASC),

CONSTRAINT `fk\_bank\_adress1`

FOREIGN KEY (`adress\_idadress`)

REFERENCES `mydb`.`adress` (`idadress`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`bank\_inf`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`bank\_inf` (

`checking\_account` VARCHAR(20) NOT NULL,

`bank\_idbank` INT NOT NULL,

PRIMARY KEY (`checking\_account`),

INDEX `fk\_bank\_inf\_bank1\_idx` (`bank\_idbank` ASC),

CONSTRAINT `fk\_bank\_inf\_bank1`

FOREIGN KEY (`bank\_idbank`)

REFERENCES `mydb`.`bank` (`idbank`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`position`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`position` (

`idposition` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idposition`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`manager`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`manager` (

`idmanager` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`surname` VARCHAR(45) NOT NULL,

`patronymic` VARCHAR(45) NOT NULL,

`position\_idposition` INT NOT NULL,

PRIMARY KEY (`idmanager`, `position\_idposition`),

INDEX `fk\_manager\_position1\_idx` (`position\_idposition` ASC),

CONSTRAINT `fk\_manager\_position1`

FOREIGN KEY (`position\_idposition`)

REFERENCES `mydb`.`position` (`idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`client` (

`idclient` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NULL,

`type\_client\_id\_type\_client` INT NOT NULL,

`unp` VARCHAR(45) NOT NULL,

`okpo` VARCHAR(45) NOT NULL,

`bank\_inf\_checking\_account` VARCHAR(20) NOT NULL,

`manager\_idmanager` INT NOT NULL,

`manager\_position\_idposition` INT NOT NULL,

PRIMARY KEY (`idclient`, `bank\_inf\_checking\_account`, `manager\_idmanager`, `manager\_position\_idposition`),

INDEX `fk\_client\_type\_client\_idx` (`type\_client\_id\_type\_client` ASC),

INDEX `fk\_client\_bank\_inf1\_idx` (`bank\_inf\_checking\_account` ASC),

INDEX `fk\_client\_manager1\_idx` (`manager\_idmanager` ASC, `manager\_position\_idposition` ASC),

CONSTRAINT `fk\_client\_type\_client`

FOREIGN KEY (`type\_client\_id\_type\_client`)

REFERENCES `mydb`.`type\_client` (`id\_type\_client`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_client\_bank\_inf1`

FOREIGN KEY (`bank\_inf\_checking\_account`)

REFERENCES `mydb`.`bank\_inf` (`checking\_account`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_client\_manager1`

FOREIGN KEY (`manager\_idmanager` , `manager\_position\_idposition`)

REFERENCES `mydb`.`manager` (`idmanager` , `position\_idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`type\_agreement`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`type\_agreement` (

`idtype\_agreement` INT NOT NULL,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`idtype\_agreement`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`role`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`role` (

`idrole` INT NOT NULL AUTO\_INCREMENT,

`type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idrole`),

UNIQUE INDEX `type\_UNIQUE` (`type` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`workes`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`workes` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`role\_idrole` INT NOT NULL,

`surname` VARCHAR(45) NOT NULL,

`patronymic` VARCHAR(45) NOT NULL,

`birthday` DATE NOT NULL,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `login\_UNIQUE` (`login` ASC),

CONSTRAINT `fk\_workes\_role1`

FOREIGN KEY (`role\_idrole`)

REFERENCES `mydb`.`role` (`idrole`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`type\_template`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`type\_template` (

`idtype\_template` INT NOT NULL,

`type\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idtype\_template`),

UNIQUE INDEX `type\_name\_UNIQUE` (`type\_name` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`document\_output\_type`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`document\_output\_type` (

`iddocument\_output\_type` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`iddocument\_output\_type`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`converter\_document`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`converter\_document` (

`idconverter\_document` INT NOT NULL AUTO\_INCREMENT,

`location\_document` VARCHAR(150) NOT NULL,

`document\_output\_type\_iddocument\_output\_type` INT NOT NULL,

PRIMARY KEY (`idconverter\_document`),

INDEX `fk\_converter\_document\_document\_output\_type1\_idx` (`document\_output\_type\_iddocument\_output\_type` ASC),

CONSTRAINT `fk\_converter\_document\_document\_output\_type1`

FOREIGN KEY (`document\_output\_type\_iddocument\_output\_type`)

REFERENCES `mydb`.`document\_output\_type` (`iddocument\_output\_type`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`templates\_document`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`templates\_document` (

`id\_template` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(150) NOT NULL,

`file\_name` VARCHAR(150) NOT NULL,

`type\_template\_idtype\_template` INT NOT NULL,

`converter\_document\_idconverter\_document` INT NOT NULL,

PRIMARY KEY (`id\_template`),

UNIQUE INDEX `name\_UNIQUE` (`name` ASC),

UNIQUE INDEX `file\_name\_UNIQUE` (`file\_name` ASC),

INDEX `fk\_templates\_document\_type\_template1\_idx` (`type\_template\_idtype\_template` ASC),

INDEX `fk\_templates\_document\_converter\_document1\_idx` (`converter\_document\_idconverter\_document` ASC),

CONSTRAINT `fk\_templates\_document\_type\_template1`

FOREIGN KEY (`type\_template\_idtype\_template`)

REFERENCES `mydb`.`type\_template` (`idtype\_template`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_templates\_document\_converter\_document1`

FOREIGN KEY (`converter\_document\_idconverter\_document`)

REFERENCES `mydb`.`converter\_document` (`idconverter\_document`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`agreement`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`agreement` (

`idagreement` INT NOT NULL AUTO\_INCREMENT,

`number` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

`date\_start` DATE NOT NULL,

`date\_finish` DATE NOT NULL,

`type\_agreement\_idtype\_agreement` INT NOT NULL,

`client\_idclient` INT NOT NULL,

`client\_bank\_inf\_checking\_account` VARCHAR(20) NOT NULL,

`client\_manager\_idmanager` INT NOT NULL,

`client\_manager\_position\_idposition` INT NOT NULL,

`workes\_role\_idrole` INT NOT NULL,

`templates\_document\_id\_template` INT NOT NULL,

PRIMARY KEY (`idagreement`, `type\_agreement\_idtype\_agreement`),

INDEX `fk\_agreement\_type\_agreement1\_idx` (`type\_agreement\_idtype\_agreement` ASC),

INDEX `fk\_agreement\_client1\_idx` (`client\_idclient` ASC, `client\_bank\_inf\_checking\_account` ASC, `client\_manager\_idmanager` ASC, `client\_manager\_position\_idposition` ASC),

INDEX `fk\_agreement\_workes1\_idx` (`workes\_role\_idrole` ASC),

INDEX `fk\_agreement\_templates\_document1\_idx` (`templates\_document\_id\_template` ASC),

CONSTRAINT `fk\_agreement\_type\_agreement1`

FOREIGN KEY (`type\_agreement\_idtype\_agreement`)

REFERENCES `mydb`.`type\_agreement` (`idtype\_agreement`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_agreement\_client1`

FOREIGN KEY (`client\_idclient` , `client\_bank\_inf\_checking\_account` , `client\_manager\_idmanager` , `client\_manager\_position\_idposition`)

REFERENCES `mydb`.`client` (`idclient` , `bank\_inf\_checking\_account` , `manager\_idmanager` , `manager\_position\_idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_agreement\_workes1`

FOREIGN KEY (`workes\_role\_idrole`)

REFERENCES `mydb`.`workes` (`role\_idrole`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_agreement\_templates\_document1`

FOREIGN KEY (`templates\_document\_id\_template`)

REFERENCES `mydb`.`templates\_document` (`id\_template`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`annex\_to\_changes\_agreement`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`annex\_to\_changes\_agreement` (

`idannex\_to\_changes\_agreement` INT NOT NULL AUTO\_INCREMENT,

`annex\_to\_changes\_agreementcol` VARCHAR(45) NOT NULL,

`agreement\_idagreement` INT NOT NULL,

`agreement\_type\_agreement\_idtype\_agreement` INT NOT NULL,

`date\_annex` DATE NOT NULL,

`templates\_document\_id\_template` INT NOT NULL,

PRIMARY KEY (`idannex\_to\_changes\_agreement`),

INDEX `fk\_annex\_to\_changes\_agreement\_agreement1\_idx` (`agreement\_idagreement` ASC, `agreement\_type\_agreement\_idtype\_agreement` ASC),

INDEX `fk\_annex\_to\_changes\_agreement\_templates\_document1\_idx` (`templates\_document\_id\_template` ASC),

CONSTRAINT `fk\_annex\_to\_changes\_agreement\_agreement1`

FOREIGN KEY (`agreement\_idagreement` , `agreement\_type\_agreement\_idtype\_agreement`)

REFERENCES `mydb`.`agreement` (`idagreement` , `type\_agreement\_idtype\_agreement`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_annex\_to\_changes\_agreement\_templates\_document1`

FOREIGN KEY (`templates\_document\_id\_template`)

REFERENCES `mydb`.`templates\_document` (`id\_template`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`protocol`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`protocol` (

`id\_protocol` INT NOT NULL AUTO\_INCREMENT,

`number` VARCHAR(45) NOT NULL,

`date` DATE NOT NULL,

`agreement\_idagreement` INT NOT NULL,

`agreement\_type\_agreement\_idtype\_agreement` INT NOT NULL,

PRIMARY KEY (`id\_protocol`),

UNIQUE INDEX `protocolcol\_UNIQUE` (`date` ASC),

INDEX `fk\_protocol\_agreement1\_idx` (`agreement\_idagreement` ASC, `agreement\_type\_agreement\_idtype\_agreement` ASC),

CONSTRAINT `fk\_protocol\_agreement1`

FOREIGN KEY (`agreement\_idagreement` , `agreement\_type\_agreement\_idtype\_agreement`)

REFERENCES `mydb`.`agreement` (`idagreement` , `type\_agreement\_idtype\_agreement`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`type\_money`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`type\_money` (

`code` INT NOT NULL,

`short` VARCHAR(5) NOT NULL,

`full\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`code`),

UNIQUE INDEX `short\_UNIQUE` (`short` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`price`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`price` (

`integer\_part` INT NOT NULL DEFAULT 0,

`fraction` VARCHAR(45) NOT NULL DEFAULT 0,

`type\_money\_code` INT NOT NULL,

PRIMARY KEY (`integer\_part`, `type\_money\_code`),

INDEX `fk\_price\_type\_money1\_idx` (`type\_money\_code` ASC),

CONSTRAINT `fk\_price\_type\_money1`

FOREIGN KEY (`type\_money\_code`)

REFERENCES `mydb`.`type\_money` (`code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`services`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`services` (

`id\_service` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`type` ENUM('PER\_HOURS', 'ONE') NOT NULL,

`price\_integer\_part` INT NOT NULL,

PRIMARY KEY (`id\_service`),

INDEX `fk\_services\_price1\_idx` (`price\_integer\_part` ASC),

CONSTRAINT `fk\_services\_price1`

FOREIGN KEY (`price\_integer\_part`)

REFERENCES `mydb`.`price` (`integer\_part`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`discount`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`discount` (

`per\_cent` INT NOT NULL,

`protocol\_id\_protocol` INT NOT NULL,

`code` INT NOT NULL AUTO\_INCREMENT,

`date` DATE NOT NULL,

INDEX `fk\_discount\_protocol1\_idx` (`protocol\_id\_protocol` ASC),

PRIMARY KEY (`code`),

CONSTRAINT `fk\_discount\_protocol1`

FOREIGN KEY (`protocol\_id\_protocol`)

REFERENCES `mydb`.`protocol` (`id\_protocol`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`mails`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`mails` (

`idmails` INT NOT NULL,

`mail` VARCHAR(45) NOT NULL,

`client\_idclient` INT NOT NULL,

`client\_bank\_inf\_checking\_account` VARCHAR(20) NOT NULL,

`client\_manager\_idmanager` INT NOT NULL,

`client\_manager\_position\_idposition` INT NOT NULL,

PRIMARY KEY (`idmails`),

INDEX `fk\_mails\_client1\_idx` (`client\_idclient` ASC, `client\_bank\_inf\_checking\_account` ASC, `client\_manager\_idmanager` ASC, `client\_manager\_position\_idposition` ASC),

CONSTRAINT `fk\_mails\_client1`

FOREIGN KEY (`client\_idclient` , `client\_bank\_inf\_checking\_account` , `client\_manager\_idmanager` , `client\_manager\_position\_idposition`)

REFERENCES `mydb`.`client` (`idclient` , `bank\_inf\_checking\_account` , `manager\_idmanager` , `manager\_position\_idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`phones`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`phones` (

`code` INT NOT NULL,

`number` VARCHAR(45) NOT NULL,

`client\_idclient` INT NOT NULL,

PRIMARY KEY (`code`, `number`),

INDEX `fk\_phones\_client1\_idx` (`client\_idclient` ASC),

CONSTRAINT `fk\_phones\_client1`

FOREIGN KEY (`client\_idclient`)

REFERENCES `mydb`.`client` (`idclient`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`account`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`account` (

`idaccount` INT NOT NULL AUTO\_INCREMENT,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`client\_idclient` INT NOT NULL,

PRIMARY KEY (`idaccount`),

UNIQUE INDEX `login\_UNIQUE` (`login` ASC),

INDEX `fk\_account\_client1\_idx` (`client\_idclient` ASC),

CONSTRAINT `fk\_account\_client1`

FOREIGN KEY (`client\_idclient`)

REFERENCES `mydb`.`client` (`idclient`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`external\_documents`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`external\_documents` (

`idexternal\_documents` INT NOT NULL AUTO\_INCREMENT,

`name\_document` VARCHAR(45) NOT NULL,

`date\_incoming` DATE NOT NULL,

PRIMARY KEY (`idexternal\_documents`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`changes\_agreement`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`changes\_agreement` (

`id` INT NOT NULL AUTO\_INCREMENT,

`point` VARCHAR(10) NOT NULL,

`text` VARCHAR(45) NOT NULL,

`agreement\_idagreement` INT NOT NULL,

`agreement\_type\_agreement\_idtype\_agreement` INT NOT NULL,

PRIMARY KEY (`id`, `agreement\_idagreement`, `agreement\_type\_agreement\_idtype\_agreement`),

INDEX `fk\_changes\_agreement\_agreement1\_idx` (`agreement\_idagreement` ASC, `agreement\_type\_agreement\_idtype\_agreement` ASC),

CONSTRAINT `fk\_changes\_agreement\_agreement1`

FOREIGN KEY (`agreement\_idagreement` , `agreement\_type\_agreement\_idtype\_agreement`)

REFERENCES `mydb`.`agreement` (`idagreement` , `type\_agreement\_idtype\_agreement`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`protocol\_has\_services`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`protocol\_has\_services` (

`protocol\_id\_protocol` INT NOT NULL,

`services\_id\_service` INT NOT NULL,

`count\_hours` FLOAT NULL,

PRIMARY KEY (`protocol\_id\_protocol`, `services\_id\_service`),

INDEX `fk\_protocol\_has\_services\_services1\_idx` (`services\_id\_service` ASC),

INDEX `fk\_protocol\_has\_services\_protocol1\_idx` (`protocol\_id\_protocol` ASC),

CONSTRAINT `fk\_protocol\_has\_services\_protocol1`

FOREIGN KEY (`protocol\_id\_protocol`)

REFERENCES `mydb`.`protocol` (`id\_protocol`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_protocol\_has\_services\_services1`

FOREIGN KEY (`services\_id\_service`)

REFERENCES `mydb`.`services` (`id\_service`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`request\_to\_meeting`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`request\_to\_meeting` (

`idrequest\_to\_meeting` INT NOT NULL AUTO\_INCREMENT,

`status` ENUM('CANCELED', 'CHANGED', 'PROCESSED', 'CONFIRMED') NOT NULL,

`departure\_date` DATE NOT NULL,

`start\_meeting` TIME NOT NULL,

`end\_meeting` TIME NOT NULL,

`theme` VARCHAR(150) NOT NULL,

`client\_idclient` INT NOT NULL,

`client\_bank\_inf\_checking\_account` VARCHAR(20) NOT NULL,

`client\_manager\_idmanager` INT NOT NULL,

`client\_manager\_position\_idposition` INT NOT NULL,

`workes\_role\_idrole` INT NOT NULL,

PRIMARY KEY (`idrequest\_to\_meeting`),

INDEX `fk\_request\_to\_meeting\_client1\_idx` (`client\_idclient` ASC, `client\_bank\_inf\_checking\_account` ASC, `client\_manager\_idmanager` ASC, `client\_manager\_position\_idposition` ASC),

INDEX `fk\_request\_to\_meeting\_workes1\_idx` (`workes\_role\_idrole` ASC),

CONSTRAINT `fk\_request\_to\_meeting\_client1`

FOREIGN KEY (`client\_idclient` , `client\_bank\_inf\_checking\_account` , `client\_manager\_idmanager` , `client\_manager\_position\_idposition`)

REFERENCES `mydb`.`client` (`idclient` , `bank\_inf\_checking\_account` , `manager\_idmanager` , `manager\_position\_idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_request\_to\_meeting\_workes1`

FOREIGN KEY (`workes\_role\_idrole`)

REFERENCES `mydb`.`workes` (`role\_idrole`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`last\_meeting`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`last\_meeting` (

`id` INT NOT NULL AUTO\_INCREMENT,

`request\_to\_meeting\_idrequest\_to\_meeting` INT NOT NULL,

`status` ENUM('FINISH', 'CANCELED') NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_last\_meeting\_request\_to\_meeting1`

FOREIGN KEY (`request\_to\_meeting\_idrequest\_to\_meeting`)

REFERENCES `mydb`.`request\_to\_meeting` (`idrequest\_to\_meeting`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`suggections`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`suggections` (

`idsuggections` INT NOT NULL AUTO\_INCREMENT,

`last\_meeting\_id` INT NOT NULL,

`suggection` VARCHAR(150) NOT NULL,

PRIMARY KEY (`idsuggections`),

INDEX `fk\_suggections\_last\_meeting1\_idx` (`last\_meeting\_id` ASC),

CONSTRAINT `fk\_suggections\_last\_meeting1`

FOREIGN KEY (`last\_meeting\_id`)

REFERENCES `mydb`.`last\_meeting` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`changed\_date\_time`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`changed\_date\_time` (

`idchanged\_date\_time` INT NOT NULL,

`new\_date` DATE NOT NULL,

`new\_start\_time` TIME NOT NULL,

`new\_end\_time` TIME NOT NULL,

`request\_to\_meeting\_idrequest\_to\_meeting` INT NOT NULL,

PRIMARY KEY (`idchanged\_date\_time`, `request\_to\_meeting\_idrequest\_to\_meeting`),

INDEX `fk\_changed\_date\_time\_request\_to\_meeting1\_idx` (`request\_to\_meeting\_idrequest\_to\_meeting` ASC),

CONSTRAINT `fk\_changed\_date\_time\_request\_to\_meeting1`

FOREIGN KEY (`request\_to\_meeting\_idrequest\_to\_meeting`)

REFERENCES `mydb`.`request\_to\_meeting` (`idrequest\_to\_meeting`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`agreement\_has\_last\_meeting`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`agreement\_has\_last\_meeting` (

`agreement\_idagreement` INT NOT NULL,

`agreement\_type\_agreement\_idtype\_agreement` INT NOT NULL,

`last\_meeting\_id` INT NOT NULL,

PRIMARY KEY (`agreement\_idagreement`, `agreement\_type\_agreement\_idtype\_agreement`, `last\_meeting\_id`),

INDEX `fk\_agreement\_has\_last\_meeting\_last\_meeting1\_idx` (`last\_meeting\_id` ASC),

INDEX `fk\_agreement\_has\_last\_meeting\_agreement1\_idx` (`agreement\_idagreement` ASC, `agreement\_type\_agreement\_idtype\_agreement` ASC),

CONSTRAINT `fk\_agreement\_has\_last\_meeting\_agreement1`

FOREIGN KEY (`agreement\_idagreement` , `agreement\_type\_agreement\_idtype\_agreement`)

REFERENCES `mydb`.`agreement` (`idagreement` , `type\_agreement\_idtype\_agreement`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_agreement\_has\_last\_meeting\_last\_meeting1`

FOREIGN KEY (`last\_meeting\_id`)

REFERENCES `mydb`.`last\_meeting` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`last\_meeting\_has\_protocol`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`last\_meeting\_has\_protocol` (

`last\_meeting\_id` INT NOT NULL,

`protocol\_id\_protocol` INT NOT NULL,

PRIMARY KEY (`last\_meeting\_id`, `protocol\_id\_protocol`),

INDEX `fk\_last\_meeting\_has\_protocol\_protocol1\_idx` (`protocol\_id\_protocol` ASC),

INDEX `fk\_last\_meeting\_has\_protocol\_last\_meeting1\_idx` (`last\_meeting\_id` ASC),

CONSTRAINT `fk\_last\_meeting\_has\_protocol\_last\_meeting1`

FOREIGN KEY (`last\_meeting\_id`)

REFERENCES `mydb`.`last\_meeting` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_last\_meeting\_has\_protocol\_protocol1`

FOREIGN KEY (`protocol\_id\_protocol`)

REFERENCES `mydb`.`protocol` (`id\_protocol`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`cancellation`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`cancellation` (

`idcancellation` INT NOT NULL,

`date` DATE NOT NULL,

`location` VARCHAR(150) NOT NULL,

`agreement\_idagreement` INT NOT NULL,

`agreement\_type\_agreement\_idtype\_agreement` INT NOT NULL,

`templates\_document\_id\_template` INT NOT NULL,

PRIMARY KEY (`idcancellation`),

INDEX `fk\_cancellation\_agreement1\_idx` (`agreement\_idagreement` ASC, `agreement\_type\_agreement\_idtype\_agreement` ASC),

INDEX `fk\_cancellation\_templates\_document1\_idx` (`templates\_document\_id\_template` ASC),

CONSTRAINT `fk\_cancellation\_agreement1`

FOREIGN KEY (`agreement\_idagreement` , `agreement\_type\_agreement\_idtype\_agreement`)

REFERENCES `mydb`.`agreement` (`idagreement` , `type\_agreement\_idtype\_agreement`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_cancellation\_templates\_document1`

FOREIGN KEY (`templates\_document\_id\_template`)

REFERENCES `mydb`.`templates\_document` (`id\_template`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`debets`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`debets` (

`iddebets` INT NOT NULL AUTO\_INCREMENT,

`isCLosed` TINYINT NOT NULL,

`protocol\_has\_services\_protocol\_id\_protocol` INT NOT NULL,

`protocol\_has\_services\_services\_id\_service` INT NOT NULL,

`maturity date` DATE NULL,

PRIMARY KEY (`iddebets`),

INDEX `fk\_debets\_protocol\_has\_services1\_idx` (`protocol\_has\_services\_protocol\_id\_protocol` ASC, `protocol\_has\_services\_services\_id\_service` ASC),

CONSTRAINT `fk\_debets\_protocol\_has\_services1`

FOREIGN KEY (`protocol\_has\_services\_protocol\_id\_protocol` , `protocol\_has\_services\_services\_id\_service`)

REFERENCES `mydb`.`protocol\_has\_services` (`protocol\_id\_protocol` , `services\_id\_service`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`email\_history`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`email\_history` (

`idemail\_history` INT NOT NULL,

`is\_client\_sender` TINYINT NOT NULL,

`email\_text` VARCHAR(500) NOT NULL,

`client\_idclient` INT NOT NULL,

`client\_bank\_inf\_checking\_account` VARCHAR(20) NOT NULL,

`client\_manager\_idmanager` INT NOT NULL,

`client\_manager\_position\_idposition` INT NOT NULL,

`workes\_role\_idrole` INT NOT NULL,

PRIMARY KEY (`idemail\_history`),

INDEX `fk\_email\_history\_client1\_idx` (`client\_idclient` ASC, `client\_bank\_inf\_checking\_account` ASC, `client\_manager\_idmanager` ASC, `client\_manager\_position\_idposition` ASC),

INDEX `fk\_email\_history\_workes1\_idx` (`workes\_role\_idrole` ASC),

CONSTRAINT `fk\_email\_history\_client1`

FOREIGN KEY (`client\_idclient` , `client\_bank\_inf\_checking\_account` , `client\_manager\_idmanager` , `client\_manager\_position\_idposition`)

REFERENCES `mydb`.`client` (`idclient` , `bank\_inf\_checking\_account` , `manager\_idmanager` , `manager\_position\_idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_email\_history\_workes1`

FOREIGN KEY (`workes\_role\_idrole`)

REFERENCES `mydb`.`workes` (`role\_idrole`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`phone\_hystory`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`phone\_hystory` (

`idphone\_hystory` INT NOT NULL,

`date` DATE NOT NULL,

`description` VARCHAR(150) NOT NULL,

`client\_idclient` INT NOT NULL,

`client\_bank\_inf\_checking\_account` VARCHAR(20) NOT NULL,

`client\_manager\_idmanager` INT NOT NULL,

`client\_manager\_position\_idposition` INT NOT NULL,

`workes\_id` INT NOT NULL,

`phones\_code` INT NOT NULL,

`phones\_number` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idphone\_hystory`, `phones\_code`, `phones\_number`),

INDEX `fk\_phone\_hystory\_client1\_idx` (`client\_idclient` ASC, `client\_bank\_inf\_checking\_account` ASC, `client\_manager\_idmanager` ASC, `client\_manager\_position\_idposition` ASC),

INDEX `fk\_phone\_hystory\_workes1\_idx` (`workes\_id` ASC),

INDEX `fk\_phone\_hystory\_phones1\_idx` (`phones\_code` ASC, `phones\_number` ASC),

CONSTRAINT `fk\_phone\_hystory\_client1`

FOREIGN KEY (`client\_idclient` , `client\_bank\_inf\_checking\_account` , `client\_manager\_idmanager` , `client\_manager\_position\_idposition`)

REFERENCES `mydb`.`client` (`idclient` , `bank\_inf\_checking\_account` , `manager\_idmanager` , `manager\_position\_idposition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_phone\_hystory\_workes1`

FOREIGN KEY (`workes\_id`)

REFERENCES `mydb`.`workes` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_phone\_hystory\_phones1`

FOREIGN KEY (`phones\_code` , `phones\_number`)

REFERENCES `mydb`.`phones` (`code` , `number`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

USE `mydb`;

DELIMITER $$

USE `mydb`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `mydb`.`request\_to\_meeting\_AFTER\_UPDATE` AFTER UPDATE ON `request\_to\_meeting` FOR EACH ROW

BEGIN

END

$$

DELIMITER ;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;