|  |
| --- |
|  |
| (наименование образовательного учреждения) |

|  |
| --- |
| **ДОПУСКАЮ К ЗАЩИТЕ** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Заместитель директора по УПР *О.В.Корешков*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
| (дата) |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Разработка базы данных учёта кассовых операций | | | | |
| (тема) | | | | |
| Выпускная квалификационная работа должна быть выполнена в виде: | | | | |
| дипломной работы и демонстрационного экзамена | | | | |
| студентом группы | | 3ИСП-11-8 | | |
|  | | (номер группы) | | |
| Мишель Александр Игоревич Бертран | | | |  |
| (И.О.Фамилия) | | | | (подпись, дата) |
| Основная профессиональная образовательная программа по специальности | | | | |
| 09.02.07 Информационные системы и программирование | | | | |
| (шифр и наименование специальности) | | | | |
| Форма обучения | | очная | | |
| Руководитель |  | преподаватель |  | Анастасия Николаевна Виеру |
|  | (ученая степень, должность, И.О.Фамилия) | | |  |
|  |  | | | (подпись, дата) |
| Руководитель |  | мастер п. о. |  | Анастасия Николаевна Виеру |
|  | (ученая степень, должность, И.О.Фамилия) | | |  |
|  |  | | | (подпись, дата) |
| Председатель предметной (междисциплинарной, модульной) комиссии | | | | |
| Кирилл Михайлович Бастрыкин | | |  |  |
| (И.О.Фамилия) | | |  | (подпись, дата) |

Москва

2021

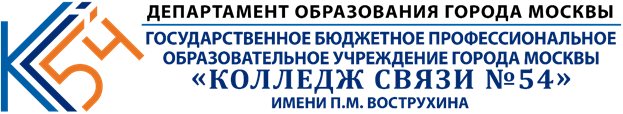
|  |
| --- |
|  |
| (наименование образовательного учреждения) |

|  |
| --- |
| **УТВЕРЖДАЮ** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| Заместитель директора по УПР *О.В.Корешков*  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
| (дата) |
|  |

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Бертран Мишель Александр Игоревич | | |
| (фамилия, имя, отчество полностью) | | |
| **I. Тема выпускной квалификационной работы** | | |
| Разработка базы данных учёта кассовых операций | | |
|  | | |
| **II. Срок сдачи студентом законченной работы** | |  |
| **III. Исходные данные** | | |
|  | Программные средства: Microsoft Office Visio 2019, СУБД SQL Server Management Studio, среда разработки Visual Studio 2019 | |
|  |  | |
| **IV. Перечень подлежащих разработке вопросов** | | |
| 1. | Анализ особенностей предметной области | |
| 2. | Сравнительный анализ программных средств. | |
| 3. | Исследование оптимизация БД учёта кассовых операций. | |
| 4. | Исследование тестирование БД учёта кассовых операций. | |
| 5. | Исследование важности и актуальности БД учёта кассовых операций. | |
| **V. Перечень графического/иллюстрационного материала** | | |
| 1. | Среда разработки SSMS | |
| 2. | Схема базы данных , разработанная в MS Visio | |
| 3. | Схема базы данных , разработанная в MS SQL Server | |
| 4. | Скриншоты экрана (практическая часть) | |
| 5. | Презентация | |
| **VI. Дата выдачи задания** | | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г |

|  |  |  |
| --- | --- | --- |
| Руководитель | А. Н. Виеру |  |
|  | (И.О.Фамилия) | (подпись) |
| Задание принял к исполнению | М. А. И. Бертран |  |
|  | (И.О.Фамилия) | (подпись) |
|  |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_20\_\_г. |



КАЛЕНДАРНЫЙ ПЛАН ВКР

(с указанием сроков выполнения отдельных этапов)

для специальности 09.02.07 Информационные системы и программирование

(группа 3ИСП11-8)

|  |  |  |
| --- | --- | --- |
| **Сроки преддипломной практики** | | 4 недели  с 12.04.21 по 30.04.21  Согласно указа Президента РФ от 23 апреля 2021 г. № 242 |
|  | Выбор темы, руководителя, оформление заявления | с 12.04.21 по 17.04.21 |
|  | Утверждение темы ВКР | 19.04.21 |
|  | Выполнение задания по теме ВКР | с 20.04.21 по 24.04.21 |
|  | Предоставление отчета по практике руководителю | с 26.04.21 по 30.04.21 |
|  | Аттестация по практике | 30.04.21 |
| **Подготовка ВКР** | | 4 недели  с 18.05.21 по 14.06.21  согласно учебному плану |
|  | Утверждение задания на ВКР | 1 неделя  с 18.05.21 по 22.05.21 |
|  | Подбор и анализ исходной информации |
|  | Подготовка и утверждение плана (оглавления) ВКР |
|  | Работа над разделами (главами) и устранение замечаний руководителя ВКР | 1 неделя  с 24.05.21 по 29.05.21 |
|  | Согласование содержания ВКР, устранение замечаний | 1 неделя  с 31.06.21 по 05.06.21 |
|  | **Оформление и представление руководителю полного текста работы. Получение отзыва руководителя ВКР.** | 07.06.21  08.06.21 |
|  | **Предоставление студентом готовой ВКР рецензенту** |
|  | **Сдача ДЭ** | Согласно отдельному графику |
|  | **Предзащита ВКР** | 09.06.21-14.06.21 |

**Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Н. Виеру**

План принял к исполнению «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2021 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Оглавление

[ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА 5](#_Toc72015274)

[ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ 6](#_Toc72015275)

[ВВЕДЕНИЕ 5](#_Toc72015276)

[ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ВЫБОР СРЕДСТВ РАЗРАБОТКИ 7](#_Toc72015277)

[1.1. Анализ предметной области 7](#_Toc72015278)

[1.2. Анализ СУБД SQL Server Management Studio 7](#_Toc72015279)

[1.3. Инфологическое проектирование 10](#_Toc72015280)

[1.4. Логическое проектирование 11](#_Toc72015281)

[1.5. Анализ оптимизации запросов 13](#_Toc72015282)

[1.6 Оптимизация параллельных сортировок и статистика 15](#_Toc72015283)

[ГЛАВА II. РАЗРАБОТКА БАЗЫ ДАННЫХ 18](#_Toc72015284)

[2.1. Проектирование и создание базы данных 18](#_Toc72015285)

[2.2. Заполнение базы данных данными 19](#_Toc72015286)

[2.3. Создание представлений 20](#_Toc72015287)

[2.4. Создание триггеров 22](#_Toc72015288)

[2.6 Создание хранимых процедур 25](#_Toc72015289)

[Выводы 26](#_Toc72015290)

[ЗАКЛЮЧЕНИЕ 27](#_Toc72015291)

[СПИСОК ЛИТЕРАТУРЫ 28](#_Toc72015292)

# ВВЕДЕНИЕ

Кассовые операции — это операции с физическими и юридическими лицами по приёму и выдаче наличных денежных средств (банкнот и монет), осуществляемые предприятиями, банками. Представляют собой совокупность материально-технических процедур, заключающихся в приёме (инкассация), хранении и выдаче наличных денег.

Актуальность данной темы имеет место быть по причине того, что подсчёт большого количества денежной без ошибок и хранение данной информации имеет большое значение в разных сферах в том числе и в магазинах, которые постоянно имеют дело с большим количеством наличности.

Целью выполнения выпускной квалификационной работы является систематизация и закрепление полученных теоретических знаний и практических умений по общепрофессиональным и специальным дисциплинам специальности “Информационные системы и программирование” и разработки базы данных с использованием конкретной СУБД и в ходе выполнения выпускной квалификационной работы необходимо разработать базу данных для автоматизации кассовых операций, где сотрудники предприятия смогу манипулировать данными и выполнять свои должностные обязанности.

Для достижения данной цели необходимо решить следующие задачи:

• выбрать необходимые средства разработки;

• спроектировать базу данных для риелторской деятельности;

• создать представления;

• создать триггеры;

• создать возможность автозаполнения некоторых столбцов на случай, если они не будут заполнены.

Объектом исследования является MS SQL-Server, так как именно в нём будет создана база данных и выполнятся поставленные задачи.

Предметом исследования является автоматизация регистрации кассовых операций, а также хранение и анализ соответствующей информации.

# ГЛАВА I АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ВЫБОР СРЕДСТВ РАЗРАБОТКИ

## 1.1. Анализ предметной области

Целью создания этой базы данных является упрощение работы с кассой в условиях магазина, то есть все кассовые манипуляции будут регистрироваться и отмечаться в БД для дальнейшего анализа и возможности найти несоответствия и обезопасить магазин от недочётов.

Проанализировав предметную область, касающуюся автоматизации кассовых операций в условиях магазина, я пришёл к выводу, о том, что данное база данных должна облегчить труд, связанный с подсчётом и учётом наличности в кассах, а также исключить возможные недочёты из-за человеческого фактора, все эти параметры должны положительно сказаться на работе любого предприятия.

В рамках выпускной квалификационной работы была разработана гибкая база данных, в которой будет содержаться:

• Данные о движение денег в кассе;

• Данные о клиентах;

• Данные о сотрудниках;

• Данные о магазинах;

• Данные о сделках;

• Данные о кассах и где они находятся физически;

• Данные о чеках;

• Данные о продукции;

• Данные о статусах касс;

Данная База данных предоставляет возможность получить подробную информацию о движение денег в кассе.

## 1.2. Анализ СУБД SQL Server Management Studio

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

MSSQL Server подойдёт:

* для управления базами масштабных предприятий,
* если ПО проекта требует обращения к MSSQL,
* при нехватке функционала MySQL,
* для использования не только в веб-проектах, но и в desktop-программах.

На рисунке 1.1 ниже изображен сам интерфейс MSSQL:

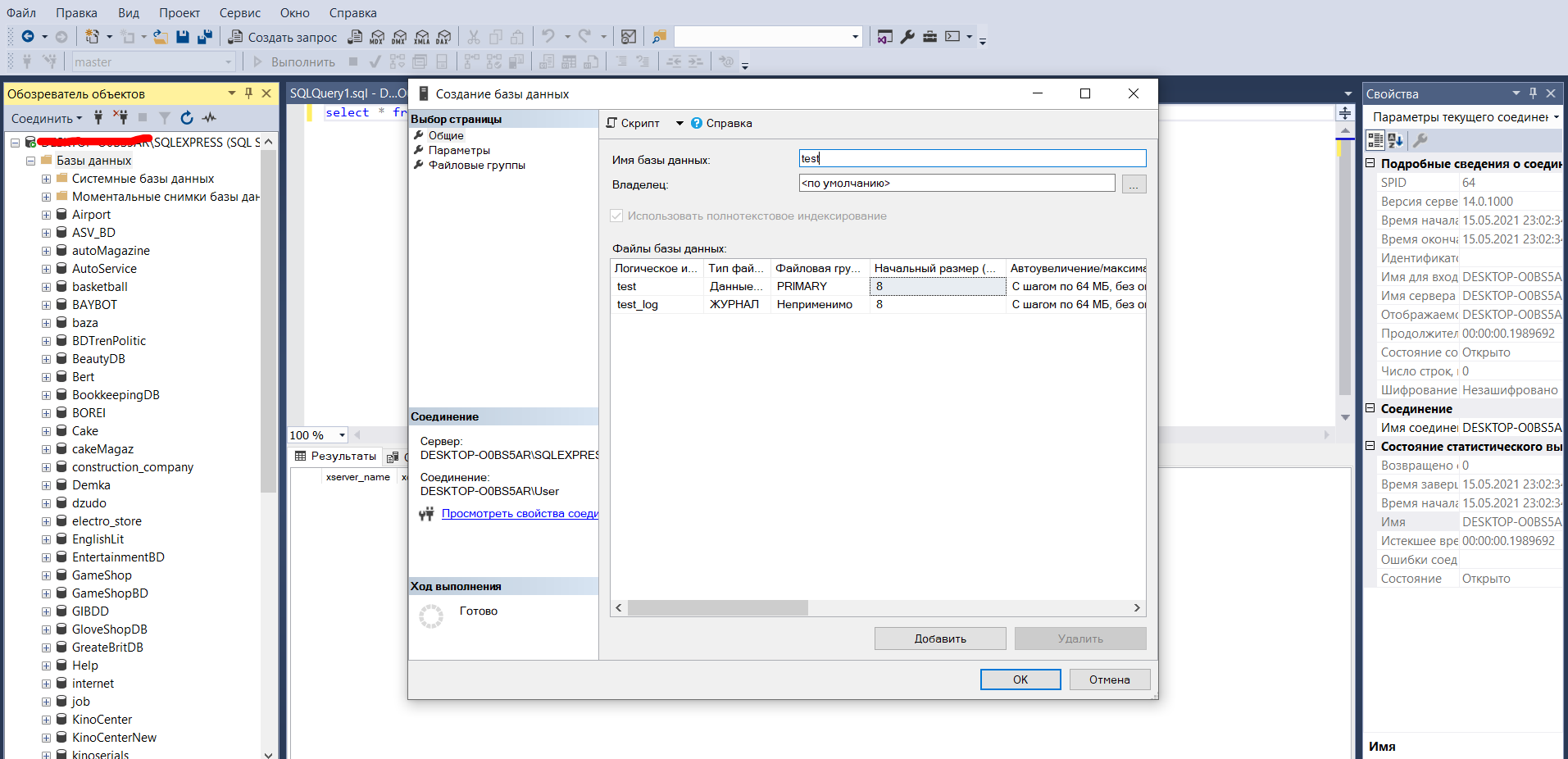


Рисунок 1.1. MSSQL

Как можно наблюдать он адаптивен удобен и понятен и это ещё одна из причин почему я выбрал именно эту СУБД.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

СУБД SQL server используются для создания, размещения, хранения и управления реляционными (табличными) базами данных на специальных серверах или в облаке. Они работают через настольные приложения и web-сайты. К основным преимуществам их функционирования относятся:

* высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД;
* простота работы и администрирования, обусловленные понятной структурой языка программирования SQL;
* безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования.

Специфика работы сервера базы данных SQL server заключается в транзакционной обработке данных. Это означает, что по каждому запросу от СУБД обрабатывается и сохраняется небольшое количество информации.

Применение SQL server позволяет автоматизировать решение различных бизнес-задач, поддерживать проведение аналитики данных в режиме онлайн, отслеживать направление ресурсов СУБД, управлять транзакциями (операциями по обработке данных).

## 1.3. Инфологическое проектирование

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области» (например, « концептуальная модель базы данных » и «концептуальная модель предметной области»), поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

* описание информационных объектов или понятий предметной области и связей между ними.
* описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

Основными задачами инфологического проектирования являются определение предметной области системы и формирование взгляда на предметную область с позиций будущих пользователей БД,

Инфологическая модель базы данных представляет собой описание структуры и динамики предметной области, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД.

Рассмотрим основные подходы к созданию инфологической модели предметной области.

Функциональный -реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Предметный -применяется в тех случаях, когда у разработчиков есть чёткое представление о предметной области, о том, какую именно информацию нужно хранить в БД, но не определена (или не полностью определена) структура запросов. Основное внимание уделяется наиболее адекватному отображению предметной области в БД с учётом информационных запросов к ней.

## 1.4. Логическое проектирование

Логическое проектирование базы данных – это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации.

Второй этап проектирования базы данных называется логическим проектированием базы данных. Его цель состоит в создании логической модели данных для исследуемой части предприятия. Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных. Логическая модель данных учитывает особенности выбранной модели организации данных в целевой СУБД (например, реляционная модель).

Если концептуальная модель данных не зависит от любых физических аспектов реализации, то логическая модель данных создается на основе выбранной модели организации данных целевой СУБД. Иначе говоря, на этом этапе уже должно быть известно, какая СУБД будет использоваться в качестве целевой — реляционная, сетевая, иерархическая или объектно-ориентированная. Однако на этом этапе игнорируются все остальные характеристики выбранной СУБД, например, любые особенности физической организации ее структур хранения данных и построения индексов.

В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей.

Созданная логическая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами поиска компромиссов, необходимых для достижения поставленных целей, что очень важно для эффективного проектирования. Логическая модель данных играет также важную роль на этапе эксплуатации и сопровождения уже готовой системы. При правильно организованном сопровождении поддерживаемая в актуальном состоянии модель данных позволяет точно и наглядно представить любые вносимые в базу данных изменения, а также оценить их влияние на прикладные программы и использование данных, уже имеющихся в базе.Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

## 1.5. Анализ оптимизации запросов

Оптимизация запросов — это 1) функция СУБД, осуществляющая поиск оптимального плана выполнения запросов из всех возможных для заданного запроса, 2) процесс изменения запроса и/или структуры БД с целью уменьшения использования вычислительных ресурсов при выполнении запроса. Один и тот же результат может быть получен СУБД различными способами (планами выполнения запросов), которые могут существенно отличаться как по затратам ресурсов, так и по времени выполнения. Задача оптимизации заключается в нахождении оптимального способа.

В реляционной СУБД оптимальный план выполнения запроса — это такая последовательность применения операторов реляционной алгебры к исходным и промежуточным отношениям, которая для конкретного текущего состояния БД (её структуры и наполнения) может быть выполнена с минимальным использованием вычислительных ресурсов.

В настоящее время известны две стратегии поиска оптимального плана:

* грубой силы путём оценки всех перестановок соединяемых таблиц, используемых способов входа в таблицы и типов соединения (то есть полный перебор вариантов);
* на основе генетического алгоритма путём оценки ограниченного числа перестановок.

Также некоторые СУБД позволяют программисту вмешиваться в поиск оптимального плана в различной степени, от минимального влияния до полного и чёткого указания какой именно план запроса использовать.

Планы выполнения запроса сравниваются исходя из множества факторов (реализации в различных СУБД отличаются), в том числе:

* потенциальное число строк, извлекаемое из каждой таблицы, получаемое из статистики;
* наличие индексов;
* возможность выполнения слияний (merge-join);
* способ чтения записей/блоков таблиц/индексов.

В общем случае соединение выполняется вложенными циклами. Однако этот алгоритм может оказаться менее эффективен, чем специализированные алгоритмы. Например, если у сливаемых таблиц есть индексы по соединяемым полям, или одна или обе таблицы достаточно малы, чтобы быть отсортированными в памяти, то исследуется возможность выполнения слияний.

Оценка числа извлекаемых из таблицы строк используется для принятия решения о полном сканировании таблицы вместо доступа по индексу. Решение принимается на том основании, что каждое чтение листовой страницы индекса с диска влечет за собой 1 или более позиционирований и 1 или более чтений страниц таблицы. Поскольку индекс содержит ещё и нелистовые страницы, то извлечение более 0.1-1 % строк из таблицы, как правило, эффективней выполнять полным сканированием таблицы.

Более точная оценка получится на основе следующих показателей:

1. Число извлекаемых строк
2. Средняя длина ключа в индексе
3. Среднее число строк в странице индекса
4. Длина страницы индекса
5. Высота B\*-дерева в индексе
6. Средняя длина строки в таблице
7. Среднее число строк в странице таблицы
8. Длина страницы таблицы

СУБД старается организовать хранение блоков данных одной таблицы последовательно с целью исключить накладные расходы на позиционирование при полном сканировании (СУБД использует предварительное выделение дискового пространства для файлов данных). Эффективность полного сканирования так же увеличивается за счёт упреждающего чтения. При упреждающем чтении СУБД одновременно выдает внешней памяти команды чтения нескольких блоков. Сканирование начинается по завершении чтения любого из блоков. Одновременно продолжается чтение остальных блоков. Эффективность достигается за счёт параллелизма чтения и сканирования.

## 1.6 Оптимизация параллельных сортировок и статистика

Если СУБД запущена на нескольких процессорах, то для уменьшения времени ответа сортировки могут выполняться параллельно. Необходимым условием для этого является возможность поместить все извлекаемые данные в оперативную память. Для выполнения сортировки извлекаемые данные разделяются на фрагменты, число которых равно числу процессоров. Каждый из процессоров выполняет сортировку над одним из фрагментов независимо от других. На финальном шаге выполняется слияние отсортированных фрагментов, либо слияние совмещается с выдачей данных клиенту СУБД.

Если СУБД запущена на нескольких узлах, то сортировка параллельно выполняется каждым из узлов, вовлеченных в выполнение запроса. Затем каждый из узлов отправляет свой фрагмент узлу, отвечающему за выдачу данных клиенту, где выполняется слияние полученных фрагментов.

Для оценки потенциального числа строк, извлекаемого из таблицы, РСУБД использует статистику. Статистика имеет вид гистограмм для каждой колонки таблицы, где по горизонтали располагается шкала значений, а высотой столбца отмечается оценка числа строк в процентах от общего числа строк.

Таким образом, если из таблицы извлекаются строки со значением колонки C с ограничением [V1, V2], то можно оценить число строк, попадающих в этот интервал. Алгоритм оценки числа извлекаемых строк, следующий:

1. Определить, в какие интервалы гистограммы попадает ограничение [V1, V2];
2. Найти оценки числа строк Ri для каждого интервала i в процентах.
3. Если [V1, V2] попадает в некоторый интервал [S1, S2] частично или полностью лежит в интервале, то:
   1. Найти пересечение [V1, V2] и [S1, S2]
   2. Откорректировать число значений в частичном интервале (это либо Ri \* (V1 — S2 + 1) / (S1 — S2 + 1), либо Ri \* (S1 — V2 + 1) / (S1 — S2 + 1), либо Ri \* (V1 — V2 + 1) / (S1 — S2 + 1));
4. Иначе оценка для интервала равна Ri;
5. Просуммировать оценки в процентах для всех интервалов;
6. Перевести оценку в процентах в число строк.

Как правило, СУБД не знает и не может знать точное число строк в таблице (даже для выполнения запроса SELECT COUNT (\*) FROM TABLE выполняется сканирование первичного индекса), поскольку в базе могут храниться одновременно несколько образов одной и той же таблицы с различным числом строк. Для оценки числа строк используются следующие данные:

1. Число страниц в таблице
2. Длина страницы
3. Средняя длина строки в таблице

Статистика так же может храниться нарастающим итогом. В этом случае каждый интервал содержит суммарную оценку всех предыдущих интервалов плюс собственную оценку. Для получения оценки числа строк для ограничения [V1, V2] достаточно из оценки интервала, в который попадает V2, вычесть оценку интервала, в который попадает V1.

Сбор статистики для построения гистограмм осуществляется либо специальными командами СУБД, либо фоновыми процессами СУБД. При этом ввиду того, что база может содержать существенный объём данных, делается выборка меньшего объёма из всей генеральной совокупности строк. Оценка репрезентативности (достоверности) выборки может осуществляться, например, по критерию согласия Колмогорова.

Если данные в таблице существенно изменяются в короткий промежуток времени, то статистика перестает быть актуальной и оптимизатор принимает неверные решения о полном сканировании таблиц. Режим работы базы данных должен быть спланирован таким образом, чтобы поддерживать актуальную статистику, либо не использовать оптимизацию на основе статистики.

## 1.7 Выводы по теоретической части

Очень важно подобрать нужные средства разработки, а также провести комплексный анализ предметной области и на основе всего вышеперечисленного сделать решение в сторону того или иного инструмента разработки или программы, а также грамотно построить модели, схемы и графики.

В данном случае исходя из анализа и поставленных задач можно сделать вывод что среда разработки SSMS подходит для разработки базы данных учёта кассовых операций, а саму базу данных лучше всего развернуть на сервере Microsoft SQL Server, по причинам, описанным в данной главе.

Исходя из анализа набора инструментов, которые выбраны для разработки БД можно полноценную разработку базы данных, так как их функционала вполне хватить для того чтобы построить полноценную систему.

# ГЛАВА II. РАЗРАБОТКА БАЗЫ ДАННЫХ

В практической части выпускной квалификационной работы будут разобраны основные аспекты проектирования базы данных «Кассовые операции»

## 2.1. Проектирование и создание базы данных

Создание таблиц в SQL Management Studio происходит последовательно и поэтапно, сначала проектирование потом создание базы данных.

Проектирование базы данных начинаем с составление ERD модели.

ENTITY RELATIONAL (ER) MODEL — это концептуальная модель модели данных высокого уровня. ER моделирование помогает систематически анализировать требования к данным для создания хорошо спроектированной базы данных. Модель сущности-отношения представляет сущности реального мира и отношения между ними. Лучше всего завершить моделирование ER перед внедрением вашей базы данных.

ER моделирование помогает систематически анализировать требования к данным для создания хорошо спроектированной базы данных. Таким образом, считается наилучшей практикой завершить моделирование ER перед внедрением вашей базы данных.

ERD отображает отношения набора сущностей, хранящиеся в базе данных. Другими словами, мы можем сказать, что ER-диаграммы помогут вам объяснить логическую структуру баз данных. На первый взгляд диаграмма ER выглядит очень похоже на блок-схему. Однако ER-диаграмма включает в себя множество специализированных символов, и ее значения делают эту модель уникальной. Цель ER Diagram — представить инфраструктуру структуры объекта.

Основные причины использовать диаграммы ER:

* Помогает вам определить термины, связанные с моделированием отношений сущностей
* Предоставьте предварительный просмотр того, как все ваши таблицы должны соединиться, какие поля будут на каждой таблице
* Помогает описать сущности, атрибуты, отношения
* ER-диаграммы переводятся в реляционные таблицы, что позволяет быстро создавать базы данных.
* ER-диаграммы могут быть использованы разработчиками базы данных в качестве образца для реализации данных в конкретных программных приложениях.
* Разработчик базы данных получает лучшее понимание информации, которая будет содержаться в базе данных с помощью диаграммы ERP
* ERD позволяет общаться с логической структурой базы данных пользователям

Подробная схема базы данных представлена в приложение 1.

После составления ERD базы данных по ней создаются таблицы с определёнными столбцами, пограничьями и т.д.

На рисунке 2.1 приведён скрипт на создание таблицы:

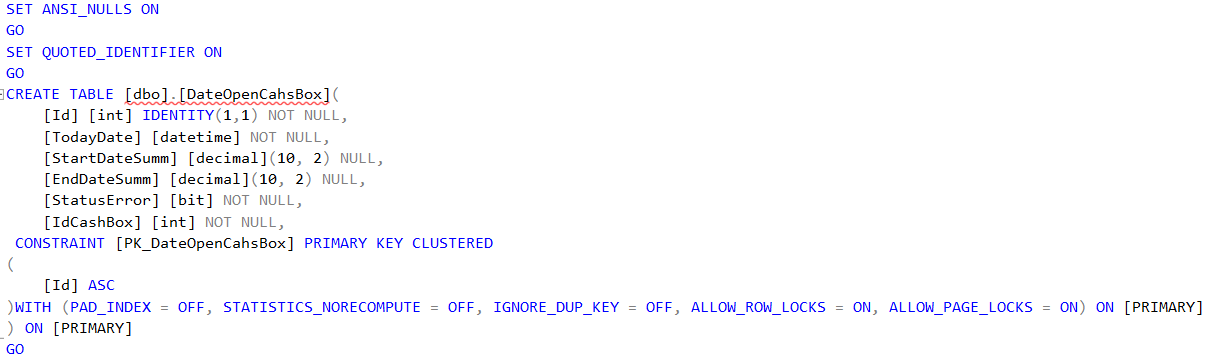


Рисунок 2.1. Скрипт для создания таблицы

## 2.2. Заполнение базы данных данными

После создание проектирование и создание БД, следует заполнить её актуальными данными.

Существуют два основных синтаксиса инструкции INSERT INTO:

В первом варианте столбец1, столбец2, столбец3,… столбец N — это названия столбцов в таблице, в которые вы хотите вставить данные, это продемонстрированно на рисунке 2.2.



Рисунок 2.2. Первый вариант добавление данных в БД

Во втором варианте столбцы вовсе не указываются, подразумевая последовательность заполнения по столбцам, мы можем это наблюдать на рисунке 2.3.



Рисунок 2.3. Второй вариант добавление данных в БД

Для заполнения базы данных я прибегнул к первому способу, так как не заполнял некоторые поля, а также этот способ мне кажется более интуитивно понятным, пример показан на рисунке 2.4.

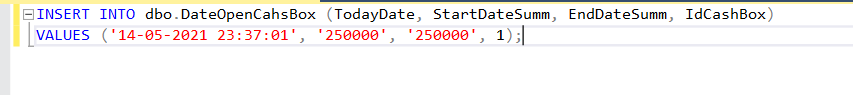


Рисунок 2.4. Добавление данных в базу данных

## 2.3. Создание представлений

Представления или Views представляют виртуальные таблицы. Но в отличии от обычных стандартных таблиц в базе данных представления содержат запросы, которые динамически извлекают используемые данные.

Представления дают нам ряд преимуществ. Они упрощают комплексные SQL-операции. Они защищают данные, так как представления могут дать доступ к части таблицы, а не ко всей таблице. Представления также позволяют возвращать отформатированные значения из таблиц в нужной и удобной форме.

Для создания представления используется команда CREATE VIEW, которая имеет следующую форму. Пример показан на рисунке 2.5:

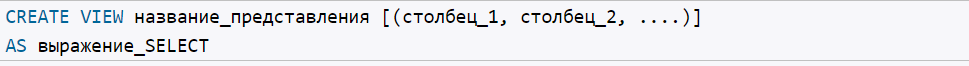


Рисунок 2.5. Схема создания представления

Представление выполняет функцию фильтра базовых таблиц, на которые оно ссылается. Определяющий представление запрос может быть инициирован в одной или нескольких таблицах, или в других представлениях текущей или других баз данных. Кроме того, для определения представлений с данными из нескольких разнородных источников можно использовать распределенные запросы. Это полезно, например, если нужно объединить структурированные подобным образом данные, относящиеся к разным серверам, каждый из которых хранит данные конкретного отдела организации.

Представления обычно используются для направления, упрощения и настройки восприятия каждым пользователем информации базы данных. Представления могут использоваться как механизмы безопасности, давая возможность пользователям обращаться к данным через представления, но не предоставляя им разрешений на непосредственный доступ к базовым таблицам, лежащим в основе представлений. Представления могут использоваться для обеспечения интерфейса обратной совместимости, моделирующего таблицу, которая существует, но схема которой изменилась. Представления могут также использоваться при прямом и обратном копировании данных в SQL Server для повышения производительности и секционирования данных.

Кроме основных определяемых пользователем представлений, выполняющих стандартные роли, в SQL Server предусмотрены следующие типы представлений, которые соответствуют специальным назначениям в базе данных.

Индексированные представления. Индексированным называется материализованное представление. Это означает, что определение представления вычисляется, а результирующие данные хранятся точно так же, как и таблица. Индексировать представление можно, создав для него уникальный кластеризованный индекс. Индексированные представления могут существенно повысить производительность некоторых типов запросов. Индексированные представления эффективнее всего использовать в запросах, группирующих множество строк. Они не очень хорошо подходят для часто обновляющихся базовых наборов данных.

Секционированные представления. Секционированным называется представление, соединяющее горизонтально секционированные данные набора таблиц-элементов, находящихся на одном или нескольких серверах. При этом данные выглядят так, как будто находятся в одной таблице. Представление, соединяющее таблицы-элементы одного экземпляра SQL Server , называется локальным секционированным представлением.

Системные представления. Системные представления предоставляют доступ к метаданным каталога. Системные представления можно использовать для получения сведений об экземпляре SQL Server или объектах, определенных в экземпляре. Например, получить сведения об определяемых пользователем базах данных, доступных в экземпляре, можно через представление каталога sys.databases.

Создание представления в базе данных с помощью запроса. Пример показан на рисунке 2.6:

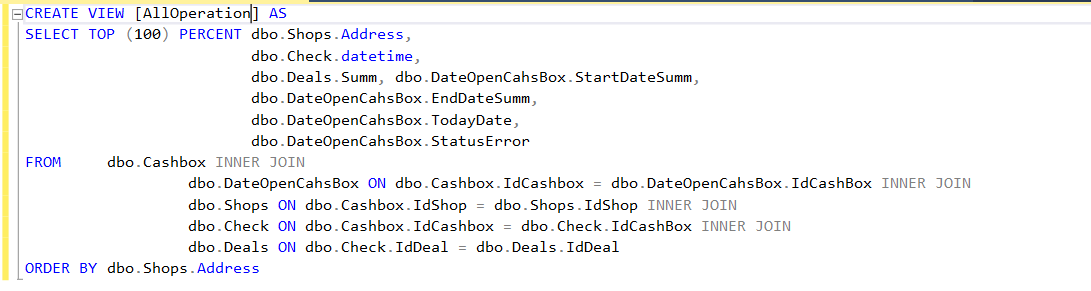


Рисунок 2.6. Создание представления

## 2.4. Создание триггеров

Триггер — это особая разновидность хранимой процедуры, которая автоматически выполняется при возникновении события на сервере базы данных. Триггеры DML выполняются, когда пользователь пытается изменить данные с помощью событий языка обработки данных (DML). Событиями DML являются процедуры INSERT, UPDATE или DELETE, применяемые к таблице или представлению. Эти триггеры срабатывают при запуске любого допустимого события независимо от наличия и числа затронутых строк таблицы. Дополнительные сведения см. в разделе DML Triggers.

Триггеры DDL активируются в ответ на разные события языка описания данных (DDL). Эти события прежде всего соответствуют инструкциям Transact-SQL CREATE, ALTER, DROP и некоторым системным хранимым процедурам, которые выполняют схожие с DDL операции.

Триггеры входа могут срабатывать в ответ на событие LOGON, которое возникает при создании пользовательского сеанса. Вы можете создавать триггеры непосредственно из инструкций Transact-SQL или методов сборок, созданных в среде CLR платформы Microsoft .NET Framework и переданных в экземпляр SQL Server. SQL Server позволяет создавать несколько триггеров для любой инструкции.

Триггеры DML часто используются для применения бизнес-правил и обеспечения целостности данных. В SQL Server декларативное ограничение ссылочной целостности обеспечивается инструкциями ALTER TABLE и CREATE TABLE. Но декларативное ограничение ссылочной целостности не обеспечивает ссылочную целостность между базами данных. Ограничение ссылочной целостности подразумевает выполнение правил связи между первичными и внешними ключами таблиц. Для обеспечения ограничений ссылочной целостности используйте в инструкциях ALTER TABLE и CREATE TABLE ограничения PRIMARY KEY и FOREIGN KEY. Если ограничения распространяются на таблицу триггера, они проверяются после выполнения триггера INSTEAD OF, но до выполнения триггера AFTER. Если будет обнаружено нарушение ограничений, для триггера INSTEAD OF выполняется откат, а триггер AFTER не срабатывает.

Вы можете указать, какой триггер AFTER будет выполняться для таблицы первым, а какой последним, с помощью sp\_settriggerorder. Для таблицы можно определить только один первый и один последний триггер для каждой из операций INSERT, UPDATE и DELETE. Если для таблицы определены другие триггеры AFTER, они выполняются в случайном порядке.

Если инструкция ALTER TRIGGER изменяет первый или последний триггер, для него удаляется метка первого или последнего триггера и порядок сортировки нужно установить заново с помощью sp\_settriggerorder.

Триггер AFTER выполняется только после того, как вызывающая срабатывание триггера инструкция SQL успешно выполняется. Успешное выполнение также подразумевает завершение всех ссылочных каскадных действий и проверки ограничений, связанных с измененными или удаленными объектами. Триггер AFTER не вызывает рекурсивное срабатывание триггера INSTEAD OF для той же таблицы.

Если определенный для таблицы триггер INSTEAD OF выполняет в этой таблице какую-либо инструкцию, которая обычно приводит к срабатыванию триггера INSTEAD OF, этот триггер не вызывается рекурсивно. Вместо этого инструкция обрабатывается так, как если бы у таблицы отсутствовал триггер INSTEAD OF и начинается последовательность применения ограничений и выполнения триггеров AFTER. Для примера предположим, что для таблицы определен триггер INSTEAD OF INSERT. Этот триггер выполняет инструкцию INSERT в той же таблице, и в этом случае выполненная в триггере INSTEAD OF инструкция INSERT не приводит к новому срабатыванию триггера. Выполняемая триггером команда INSERT начинает процесс применения ограничений и срабатывания всех триггеров AFTER INSERT, определенных для этой таблицы.

Если определенный для представления триггер INSTEAD OF выполняет по отношению к этому представлению какую-либо инструкцию, которая обычно приводит к срабатыванию триггера INSTEAD OF, триггер рекурсивно не вызывается. Вместо этого инструкция выполняет изменение базовых таблиц, на которых основано представление. В данном случае определение представления должно удовлетворять всем ограничениям, установленным для обновляемых представлений. Определение обновляемых представлений см. в разделе Изменение данных через представление.

Для примера предположим, что для представления определен триггер INSTEAD OF UPDATE. Этот триггер выполняет инструкцию UPDATE в том же представлении, и в этом случае выполненная в триггере INSTEAD OF инструкция UPDATE не приводит к новому срабатыванию триггера. Выполняемая в триггере инструкция UPDATE обрабатывает представление так, как если бы у него не было триггера INSTEAD OF. Столбцы, измененные с помощью инструкции UPDATE, должны принадлежать одной базовой таблице. Каждая модификация базовой таблицы вызывает применение последовательности ограничений и взвод триггеров AFTER, определенных для данной таблицы.

Схема составления триггера. Пример показан на рисунке 2.7:



Рисунок 2.7. Схема создания триггера

## 2.6 Создание хранимых процедур

Храни́мая процеду́ра — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Хранимые процедуры позволяют повысить производительность, расширяют возможности программирования и поддерживают функции безопасности данных.

Вместо хранения часто используемого запроса, клиенты могут ссылаться на соответствующую хранимую процедуру. При вызове хранимой процедуры её содержимое сразу же обрабатывается сервером.

Кроме собственно выполнения запроса, хранимые процедуры позволяют также производить вычисления и манипуляцию данными — изменение, удаление, выполнять DDL-операторы (не во всех СУБД!) и вызывать другие хранимые процедуры, выполнять сложную транзакционную логику. Один-единственный оператор позволяет вызвать сложный сценарий, который содержится в хранимой процедуре, что позволяет избежать пересылки через сеть сотен команд и, в особенности, необходимости передачи больших объёмов данных с клиента на сервер.

Схема создания хранимой процедуры. Пример показан на рисунке 2.8:

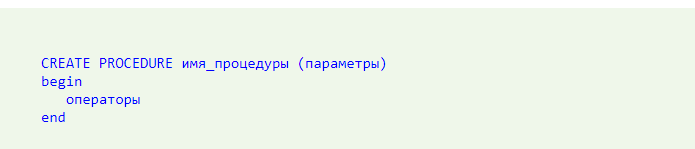


Рисунок 2.8. Схема создания хранимой процедуры

## 2.7 Выводы по практической части

Для контроля денежной массы в кассе любого магазина удобнее использовать автоматизированную систему, которая будет регистрировать и контролироваться приходящую и уходящие деньги. Эта база данных должна помочь в осуществление контроля за кассой и за денежным оборотом, осуществляемым через неё. В осуществление ранее упомянутого контроля ей поможет автоматизация, настроенная в базе данных с помощью триггеров, хранимых процедур, а также функций.

Данная база данных учёта кассовых операций, цель которой облегчить работу с кассовыми операциями, учётом, движением и хранением наличности путём автоматизации достигает ранее поставленные цели.

В качестве доработок базы можно выделить такие направления, как индексация таблиц для более удобного и быстрого поиска по базе данных, а также улучшение разграничение прав пользователей работающих с данной базой данных.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения дипломной работы была создана база данных для учета кассовых операций в магазине, это база данных является актуальной на сегодняшний день и так как учёт кассовых является неотъемлемой частью любого магазина, продающего товары. Это важно автоматизировать и потому что тогда будет легче следить и получать отчёты касающиеся движения денег в кассе и в магазине в общем.

Данная база данных создана с возможностью её будущего модернизированния и улучшения в дальнейшем, также она приведена к третьей нормальной форме, что в сою очередь также облегчает дальнейшую поддержку и развитие базы данных. Для улучшения данной базы данных в ней можно было бы добавить и настроить автоматическое проведение и сохранение бэкапов.

База данных создана с использованием современных инструментов разработки, а также является автоматизированной и гибкой для внесения правок и патчей и оптимизованна для работы с большим количеством данных. Всё вышеперечисленное позволяет быть эффективной системой для контроля кассовых операций.

# СПИСОК ЛИТЕРАТУРЫ

1. Абрамян, Михаил Visual C# на примерах / Михаил Абрамян. - М.: БХВ Петербург, 2015. - 572 c.
2. Албахари, Джозеф C# 6.0. Справочник. Полное описание языка / Джозеф Албахари, Бен Албахари. - М.: Вильямс, 2018. - 406 c.
3. Вагнер, Билл. Эффективное программирование на С# / Билл Вагнер. – М.: ЛОРИ, 2017. – 320 c
4. Ватсон, Б. С# 4.0 на примерах (C# 4.0. How-To) / Б. Ватсон. - М.: БХВПетербург, 2017. - 608 c
5. Голдштейн, Саша Оптимизация приложений на платформе .Net / Саша Голдштейн, Дима Зурбалев, Идо Флатов. - М.: ДМК Пресс, 2017. - 524 c
6. Гэри, Маклин Холл Адаптивный код на C#. Проектирование классов и интерфейсов, шаблоны и принципы SOLID / Гэри Маклин Холл. - М.: Вильямс, 2015. - 432 c.
7. Культин, Н.Б. Microsoft Visual C# в задачах и примерах (+ CD-ROM) / Н.Б. Культин. - М.: БХВ-Петербург, 2017. - 308 c.
8. Культин, Никита Борисович Основы программирования в Microsoft Visual C# 2010 (+ CD-ROM) / Культин Никита Борисович. - М.: БХВПетербург, 2016. - 201 c.
9. Магда, Ю. С. NI Measurement Studio. Практика разработки систем измерения и управления на C# / Ю.С. Магда. - М.: ДМК Пресс, 2017. - 190 c.
10. Джеффри Рихтер: CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е издание/Д. Рихтер. - М.: Питер, 2019. - 436 c.
11. Фримен, А. LINQ. Язык интегрированных запросов в C# 2010 для профессионалов / А. Фримен. - М.: Диалектика / Вильямс, 2015. - 961 c.

Приложение №1. Диаграмма базы данных.

