**Лист задания**

1. **Провести анализ предметной области по следующему описанию:**

Пункт проката занимается предоставлением во временное пользование спортивного инвентаря.  Весь инвентарь делится на несколько категорий. За использование инвентаря с посетителей взимается плата, которая зависит от вида инвентаря и времени его использования. Надо контролировать поступление оплаты. Надо хранить информацию о посетителях. Надо вести учет того, какой инвентарь, когда и на какой срок выдан.

**Перечень входных (первичных) документов.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название категории | Название инвентаря | ФИО посетителя | Цена инвентаря | Номер мобильного | Даты выдачи инвентаря | Дата возврата | Внесенные средства | Дата внесения |
| Металл | Гиря | Петров Иван Петрович | 250 | 89687547690 | 05/04/2016 | 25/04/2016 | 250 | 05/04/2016 |
| Атлетика | Гиря | Петров Иван Петрович | 250 | 89687547690 | 05/04/2016 | 25/04/2016 | 250 | 05/04/2016 |
| Бадминтон | Ракетка | Мулатов Герман | 450 | 8945675822 | 25/04/2016 | 25/04/2016 | 450 | 25/04/2016 |

**Ограничения предметной области:**

1. Каждый посетитель может взять несколько типов инвентаря в одной заявке.
2. Цена товара указывается за 1 день.
3. Один инвентарь может относиться сразу к нескольким категориям (Пример: Гиря - Категория Изделия из метала или Категория Тяжелая атлетика.)
4. Дата выдачи не может быть позже чем дата возврата (ОШИБКА!)
5. **Выполнить концептуальное (инфологическое) проектирование.**
6. **Выполнить даталогическое проектирование для реляционной базы данных.**
7. **Выбрать СУБД (обосновав выбор) и выполнить физическое проектирование.**
8. **Создать базу данных в выбранной СУБД с учетом ограничений предметной области.**
9. **Реализовать следующие отчеты (запросы):**
   1. Найти человека, который имеет максимальную сумму оплат в категории, заданной пользователем.
   2. Найти категорию с максимальным количеством выдачи и в ней вывести 2 самых не популярных товара.
   3. Вывести людей, которые арендовали строго 3 товара из 6 категорий.
   4. Вывести за каждый месяц года, заданного пользователем, количество фактов выдачи инвентаря.
10. **Выбрать язык программирования и разработать приложение для работы с БД (формы ввода/редактирования данных и отчеты).**
11. **Оформить пояснительную записку.**

Оглавление

[**1.** **Анализ предметной области** 5](#_Toc128328707)

[**2.** **Концептуальная модель базы данных** 8](#_Toc128328708)

[**3.** **Логическая модель базы данных** 11](#_Toc128328709)

[**4.** **Физическое проектирование базы данных.** 15](#_Toc128328710)

[**5.** **Реализовать следующие запросы(отчеты)** 20](#_Toc128328711)

[**6.** **Создание и реализация оконного приложения для работы с базой данных.** 22](#_Toc128328712)

[**7.** **Заключение.** 25](#_Toc128328713)

[**8.** **Список литературы.** 26](#_Toc128328714)

[**9.** **Приложения.** 27](#_Toc128328715)

# **Анализ предметной области**

С развитием информационных технологий базы данных с их широкими функциональными возможностями, связанными с обработкой информации и представлением данных, становятся неотъемлемой частью ведения успешного бизнеса.

В современном обществе, которое функционирует в жестких рыночных условиях, своевременная обработка информации способствует совершенствованию организации производства, оперативному и долгосрочному планированию, прогнозированию и анализу хозяйственной деятельности, что позволяет успешно конкурировать на рынке. Каждая организация стремится минимизировать издержки материальных, трудовых и временных ресурсов в ходе своей деятельности,

На сегодняшний день информационный технологии прочно вошли в нашу жизнь, помогают автоматизировать многие процессы, а повышение уровня материально-технического оснащения ведет к ускорению принятия оптимальных решений на всех уровнях.

Целью данного проекта является разработка базы данных для предметной области «Пункт проката» и реализация клиентского приложения для работы с данными.

Объектом исследования является пункт проката, который занимается предоставлением во временное пользование спортивного инвентаря.  Весь инвентарь делится на несколько категорий. За использование инвентаря с посетителей взимается плата, которая зависит от вида инвентаря и времени его использования.

Необходимо:

* Контролировать поступление оплаты.
* Хранить информацию о клиентах пункта проката.
* Вести учет того, какой инвентарь, когда и на какой срок выдан.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить необходимую литературу по требуемой теме
* проанализировать предметную область
* разработать концептуальную и даталогическую модели данных
* обосновать выбор сред разработки приложения и базы данных
* реализовать физическую модель базы данных в выбранной среде
* реализовать запросы
* разработать экранные формы для ввода и вывода данных
* реализовать отчеты
* обеспечить поддержку целостности данных

Пользователем данной БД будет являться работник пункта проката, который будет заносить информация о клиентах, имеющемся инвентаре и инвентаре выданном клиенту, а так же на основе имеющихся данных формировать следующие отчеты:

* Найти человека, который имеет максимальную сумму оплат в категории, заданной пользователем.
* Найти категорию с максимальным количеством выдачи и в ней вывести 2 самых не популярных товара.
* Вывести людей, которые арендовали строго 3 товара из 6 категорий.
* Вывести за каждый месяц года, заданного пользователем, количество фактов выдачи инвентаря.

Входными данными для проектирования БД являются:

* Название инвентаря.
* Категория принадлежности инвентаря.
* Цена аренды инвентаря в сутки.
* ФИО клиента.
* Номер телефона клиента.
* Дата начала аренды инвентаря.
* Срок возврата инвентаря.
* Внесенная клиентом сумма к оплате.

Также необходимо выделить следующие ограничения предметной области:

* Каждый посетитель может взять несколько типов инвентаря в одной заявке.
* Цена товара указывается за 1 день.
* Один инвентарь может относиться сразу к нескольким категориям.
* Дата выдачи не может быть позже, чем дата возврата.

Дальнейшие необходимые ограничения предметной области для сохранения целостности данных и атрибуты сущностей будут выяснены при построении моделей БД.

# **Концептуальная модель базы данных**

Концептуальная (инфологическая) модель базы данных — это наглядная диаграмма, нарисованная в принятых обозначениях и подробно отображающая связь между объектами и их характеристиками. Концептуальная модель проектируется для дальнейшего проектирования логической модели и реализации, например реляционной, физической модели базы данных. Концептуальная модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

На основании имеющихся данных предметной области выделим список сущностей, который представлен в таблице 1.

|  |  |  |
| --- | --- | --- |
| **№** | **Сущность** | **Описание сущности** |
| **1** | Инвентарь | Содержит данные об имеющемся в наличии инвентаре |
| **2** | Клиент | Содержит данные о клиенте, воспользовавшемся услугами пункта проката |
| **3** | Учетная запись | Содержит данные о факте аренды клиентом инвентаря в пункте проката |
| **4** | Платеж | Слабая сущность, не может существовать без учетной записи, содержит информацию о произведенном клиентом факте оплаты взятого в аренду инвентаря |

Таблица 1 – Сущности предметной области «Пункт Проката»

Список атрибутов с кратким описанием каждой сущности представлен в таблице 2.

|  |  |  |
| --- | --- | --- |
| **№** | **Сущность** | **Атрибуты сущности** |
| **1** | Инвентарь | 1. Категория (многозначный атрибут) 2. Наименование инвентаря 3. Цена (стоимость одних суток аренды инвентаря) |
| **2** | Клиент | 1. ФИО клиента 2. Телефон клиента |
| **3** | Учетная запись | 1. Дата выдачи инвентаря 2. Дата возврата инвентаря 3. Сумма к оплате (вычисляемый атрибут, на основе срока аренды и цены взтяого инвентаря) |
| **4** | Платеж | 1. Дата внесения платежа 2. Сумма платежа |

Таблица 2 – Список атрибутов с описанием сущностей предметной области «Пункт Проката»

В таблице 3 представлены связи между сущностями предметной области «Пункт проката».

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Сущности** | | **Связь** |
| **1** | Инвентарь | Заявка | Многие-ко-многим |
| **2** | Клиент | Заявка | Один-ко-многим |
| **4** | Платеж | Заявка | Один-ко-многим |

Таблица 3 – Список связей атрибутов предметной области «Пункт Проката»

Сущности с атрибутами и связи между ними представлены на диаграмме «Сущность-Связь» в нотации Питера Чена на рисунке 1.

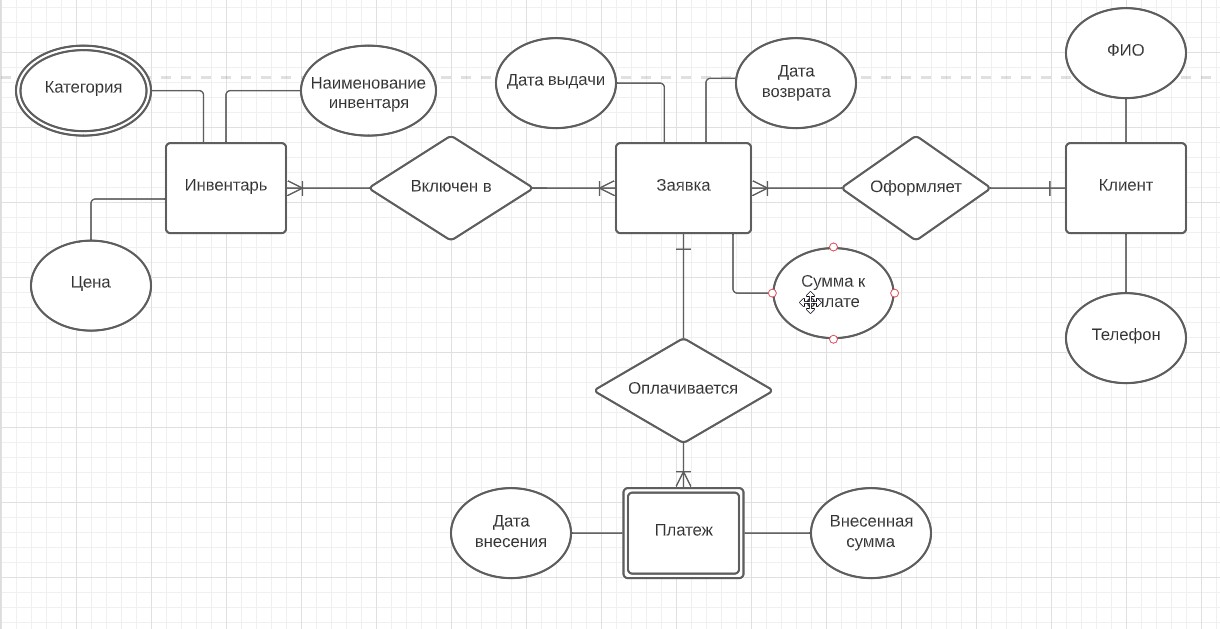


Рисунок 1 – Концептуальная модель в нотации Питера Чена

# **Логическая модель базы данных**

Основная цель логического проектирования заключается в разработке логической схемы базы данных, которая бы соответствовала выбранной СУБД. В процессе логического проектирования создается логическая модель базы данных, которая должна пройти проверку с использованием процедуры последовательной нормализации и контроль на возможность выполнения всех необходимых пользователю транзакций. Этот этап предполагает преобразование концептуальной модели данных в логическую модель, в результате которого будет определена схема реляционной модели данных. При этом исключаются связи типа "многие ко многим", сложные и рекурсивные связи, связи с атрибутами, множественные атрибуты и избыточные связи. Важным этапом является проверка модели с помощью концепций нормализации, чтобы каждое из отношений, полученных на основе концептуальной модели, находилось, по крайней мере, в третьей нормальной форме. Также происходит проверка поддержки целостности данных.

Даталогическая модель является моделью логического уровня и отображает логические связи между элементами данных безотносительно к их содержанию и среде хранения. Эта модель строится в терминах информационных единиц, доступных в той конкретной СУБД, в среде, где проектируется база данных. В процессе даталогического проектирования информация делится на несколько таблиц. Классический процесс проектирования базы данных основан на последовательности переходов от предыдущей нормальной формы к последующей. Для уменьшения избыточности информации и исключения аномалий проводится нормализация исходных схем отношений проекта базы данных. Нормализация — это процесс организации данных в базе данных, включая создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные отношения.

Одной из основных целей нормализации базы данных является устранение избыточности и дублирования информации. При нормализации необходимо обеспечить хранение каждого значения в базе данных только в одном экземпляре, не получая его расчетным путем из других данных. База данных считается нормализованной, если ее таблицы соответствуют по крайней мере третьей нормальной форме.

Чтобы привести БД к нормальной форме, необходимо:

1. Объединить имеющиеся данные в группы.
2. Выяснить логические связи между группами. Чтобы обеспечить правильность связей, связываемые поля должны иметь один тип.

Построим первую нормальную форму. Первая нормальная форма (1НФ) требует, чтобы каждое поле таблицы:

* было неделимым,
* не содержало повторяющихся групп.

Сущностями проектируемой системы являются:

* Инвентарь
* Клиенты
* Заявка
* Платеж

Согласно правилам, все атрибуты в данных таблицах должны быть простыми, все сохраняемые данные на пересечении столбцов и строк должны содержать лишь скалярные значения. Также не должно быть повторяющихся строк.

Рассмотрим сущность «Инвентарь». Так как в одному предмету инвентаря может соответствовать несколько категорий (многозначный атрибут), введём таблицу «Категории» с атрибутами «Название», а также установим таблицу «КатегорияИнвентаря», соответствия сущностей «Инвентарь» и «Категория».

Так одной заявке может соответствовать несколько предметов инвентаря, поэтому потребуется дополнительная таблица «СоставЗаявки» для сохранения целостности.

В результате получаем 7 таблиц, удовлетворяющие требованиям первой нормальной формы.

Вторая нормальная форма (2НФ) требует, чтобы все поля таблицы зависели от первичного ключа, то есть, чтобы первичный ключ однозначно определял запись и не был избыточен. Те поля, которые зависят только от части первичного ключа, должны быть выделены в составе отдельных таблиц. Добавим в каждую таблицу уникальный идентификатор сущности (ID), для нормализации базы данных.

Третья нормальная форма (ЗНФ) требует, чтобы в таблице не имелось транзитивных зависимостей между не ключевыми полями, то есть, чтобы значение любого поля таблицы, не входящего в первичный ключ, не зависело от значения другого поля, не входящего в первичный ключ. Нормализованноя база данных представлена в таблице 4.

| № | Сущность | Ключ сущности и его обозначение | Атрибуты сущности |
| --- | --- | --- | --- |
| 1 | Категория | КодКатегории / PK | Название категории |
| 2 | Инвентарь | КодИнвентаря /PK | Название инвентаря |
| 3 | КатегорияИнвентаря | КодИнвентаря/PK, FK  КодКатегории/PK, FK |  |
| 4 | Клиенты | КодКлиента/PK | Полное имя  Телефон |
| 5 | Заявка | КодЗаявки/PK  КодКлиента/FK | Дата выдачи  Дата возврата |
| 6 | СоставЗаявки | КодЗаявки/PK, FK  КодИнвентаря/PK, FK |  |
| 7 | Платеж | КодПлатежа/PK  КодЗаявки/FK | Дата платежа  Сумма Платежа |

Таблица 4 – Ключевые поля и атрибуты предметной области «Пункт Проката»

# **Физическое проектирование базы данных.**

Для привязки даталогической модели к конкретной среде хранения данных, необходимо создать физическую модель данных. Эта модель определяет, какие запоминающие устройства будут использоваться, а также способы физической организации данных в среде хранения. Модель физического уровня также учитывает возможности, которые предоставляет СУБД.

На этапе физического проектирования базы данных описывается физическая структура базы данных, которая называется схемой хранения.

Среди множества доступных СУБД на рынке наиболее распространенной и используемой является MS SQL Server. Эта система предлагает широкий набор функций, необходимых как при создании баз данных, так и при их администрировании. Для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server разработана среда SQL Server Management Studio. Она объединяет большое количество графических средств с полнофункциональными редакторами сценариев для доступа к SQL Server разработчиков и администраторов с любым уровнем опыта. Разработчики получают знакомую среду, а администраторы баз данных - единую полнофункциональную программу, объединяющую простые в использовании графические средства и богатые возможности для создания сценариев.

Одним из главных преимуществ СУБД MS SQL Server является ее тесная интеграция с другими программными продуктами от Microsoft, а также возможность экспорта и импорта в большинство распространенных форматов данных. В настоящее время данная СУБД работает не только на машинах под управлением ОС Windows, но и под ОС Linux, что позволяет использовать на недорогих VPS серверах.

Учитывая перечисленные выше преимущества и возможность доступа к удаленному VPS-серверу, было принято решение выполнить физическое проектирование курсовой работы с использованием SQL Server.

Для удобства проектирования и дальнейшей работы с SQL запросами названия сущностей и их атрибутов были переведены на английский язык.

Также в таблице «Rentals» будем использовать вычисляемые поля «Payed» где будем хранить сумму произведенных платежей по этой аренде и вычисляемое поле «Total» в котором будем хранить полную сумму к оплате по данной заявке. Также установим флажок «is\_paid» будет иметь 2 варианта значений:

* 0 по умолчанию
* 1 когда «Payed» становится равен «Total»

Для создания таблиц применялся SQL скрипт 

Структура созданных таблиц представлена в таблицах 5.1-5.7.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Note |
| 1 | ID | int IDENTITY(1,1) | PRIMARY KEY | NOT NULL |
| 2 | Title | nvarchar(50) | PRIMARY KEY | NOT NULL  UNIQUE |

Таблица 5.1 – Структура таблицы «Categories»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Constraint |
| 1 | ID | int IDENTITY(1,1) | PRIMARY KEY | NOT NULL |
| 2 | Title | nvarchar(50) | – | NOT NULL |
| 3 | Price | money | – | NOT NULL  CHECK (Price > 0) |

Таблица 5.2 – Структура таблицы «Equipments»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Constraint |
| 1 | EquipID | int | PRIMARY KEY/  FOREIGN KEY  REFERENCES Equipments(ID) | NOT NULL |
| 2 | CategoryID | int | PRIMARY KEY/  FOREIGN KEY  REFERENCES Categories(ID) | NOT NULL |

Таблица 5.3 – Структура таблицы «EquipCategory»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Constraint |
| 1 | ID | int IDENTITY(1,1) | PRIMARY KEY | NOT NULL |
| 2 | FullName | nvarchar(100) | – | NOT NULL |
| 3 | Phone | Char(11) | – | NOT NULL  CHECK (Phone LIKE '8[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')  UNIQUE |

Таблица 5.4 – Структура таблицы «Clients»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Constraint |
| 1 | ID | int IDENTITY(1,1) | PRIMARY KEY | NOT NULL |
| 2 | ClientID | int | FOREIGN KEY  REFERENCES Clients(ID) | NOT NULL |
| 3 | DateBegin | date | – | NOT NULL |
| 4 | DateEnd | date | – | NOT NULL  CHECK (DateBegin <= DateEnd) |
| 5 | Payed | money | – | NOT NULL  DEFAULT 0  CHECK (Payed >= 0) |
| 6 | Total | money | – | NOT NULL  DEFAULT 0  CHECK (Total >= 0)  CHECK (Payed <=Total) |
| 7 | is\_paid | tinyint | – | NOT NULL |

Таблица 5.5 – Структура таблицы «Clients»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Constraint |
| 1 | RentalID | int | PRIMARY KEY/  FOREIGN KEY  REFERENCES Rentals(ID) | NOT NULL |
| 2 | EquipID | int | PRIMARY KEY/  FOREIGN KEY  REFERENCES Equipments(ID) | NOT NULL |

Таблица 5.6 – Структура таблицы «RentEquip»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Column | Type | Key | Constraint |
| 1 | ID | int IDENTITY(1,1) | PRIMARY KEY | NOT NULL |
| 2 | RentalID | int | FOREIGN KEY  REFERENCES Rentals(ID) | NOT NULL |
| 3 | PaymentDate | date | – | NOT NULL |
| 4 | Amount | money | – | NOT NULL  CHECK (Amount > 0) |

Таблица 5.7 – Структура таблицы Payments»

Физическая модель базы данных представлена на рисунке 2

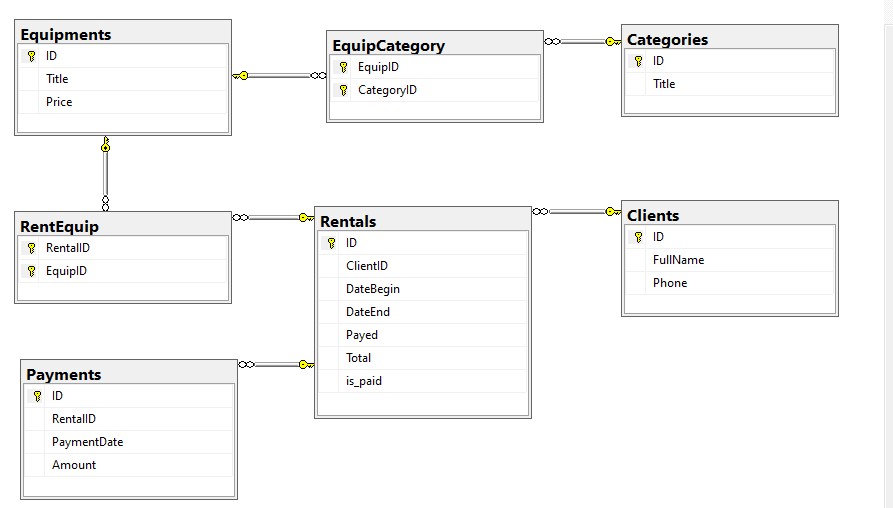


Рисунок 2 – Физическая модель базы данных

Для обеспечения ссылочной целостности базы данных, все связи между таблицами реализованы с применением каскадного обновления и бездействия при попытке удаления.

Для сохранения целостности данных при заполнении таблиц был выбран путь заведения данных через хранимые на сервере процедуры, которые принимают данные, обрабатывают их и заносят в таблицы, либо сообщают об ошибке пользователю. Код в приложении. 

# **Реализовать следующие запросы(отчеты)**

1. Найти человека, который имеет максимальную сумму оплат в категории, заданной пользователем.

Задача была решена с использованием хранимой процедуры и с передачей заданной категории в виде теста в процедуру как параметр.

CREATE PROC spCourseQueryA

@Category nvarchar(50)

AS

WITH RentalStat AS

(SELECT c.ID ClientID, c.FullName ClientName, e.Title Equipment, cat.Title Category, e.Price\*DATEDIFF(d, r.DateBegin, r.DateEnd) Total

FROM Clients c JOIN Rentals r ON c.ID = r.ClientID

JOIN RentEquip re ON r.ID = re.RentalID

JOIN Equipments e ON re.EquipID = e.ID

JOIN EquipCategory ec ON e.ID = re.EquipID

JOIN Categories cat ON ec.CategoryID = cat.ID

WHERE re.EquipID = ec.EquipID AND cat.Title = @Category)

SELECT TOP 1 Category, ClientName, SUM(Total) TotalSum

FROM RentalStat

GROUP BY Category, ClientName

ORDER BY TotalSum DESC

1. Найти категорию с максимальным количеством выдачи и в ней вывести 2 самых не популярных товара.

Задача была решена путем создания представления, выводящего искомые данные, однако, для сохранения единого стиля доступа к данным из приложения, решено было так же создать процедуру, выводящую данное преставление.

CREATE VIEW courseQueryB AS (

SELECT TOP 2

c.Title TopCategory, e.Title Equipment, COUNT(re.RentalID) RentalCount

FROM RentEquip re JOIN Equipments e ON re.EquipID = e.ID

JOIN EquipCategory ec ON ec.EquipID = e.ID JOIN Categories c ON ec.CategoryID = c.ID

WHERE ec.CategoryID = (SELECT CategoryID FROM (SELECT TOP 1 c.ID CategoryID, c.Title, COUNT(re.RentalID) RentalCount FROM RentEquip re

JOIN Equipments e ON re.EquipID = e.ID JOIN EquipCategory ec ON e.ID = ec.EquipID JOIN Categories c ON ec.CategoryID = c.ID

GROUP BY c.ID, c.Title ORDER BY RentalCount DESC) TopCategory)

GROUP BY e.Title, c.Title

ORDER BY RentalCount)

CREATE PROC spCourseQueryB

AS

SELECT \* FROM courseQueryB

1. Вывести людей, которые арендовали строго 3 товара из 6 категорий.

Задача была решена путем создания представления, выводящего искомые данные, однако, для сохранения единого стиля доступа к данным из приложения, решено было так же создать процедуру, выводящую данное преставление.

CREATE VIEW CourseQueryC AS (

SELECT ClientName, ClientPhone FROM

(SELECT c.ID, c.FullName ClientName, c.Phone ClientPhone, COUNT(DISTINCT re.EquipID) Equips, COUNT(DISTINCT ec.CategoryID) Cats

FROM Clients c JOIN Rentals r ON c.ID = r.ClientID

JOIN RentEquip re ON r.ID = re.RentalID JOIN Equipments e ON re.EquipID = e.ID

JOIN EquipCategory ec ON e.ID = ec.EquipID JOIN Categories cat ON ec.CategoryID = cat.ID

GROUP BY c.ID, c.FullName, c.Phone) inner\_table

WHERE Equips = 3 AND Cats = 6)

CREATE PROC spCourseQueryC

AS

SELECT \* FROM CourseQueryC

1. Вывести за каждый месяц года, заданного пользователем, количество фактов выдачи инвентаря.

Задачу решено было решить в общем виде, с предоставлением отчетности за любой выбранный период, а если период не выбран, а введена только одна дата, то за период до 30 дней от введенной даты. Решено также, через хранимые процедуры.

CREATE PROC spCourseQueryD

@DateBegin date,

@DateEnd date = Null

AS

IF @DateEnd IS NULL

SET @DateEnd = DATEADD(m, 1, @DateBegin)

SELECT RentalID, FullName, Phone, SUM(Price) SumPrice, DateBegin, DateEnd, Payed, Total, ToPay FROM

(SELECT r.ID RentalID, c.FullName, c.Phone, e.Price, r.DateBegin, r.DateEnd,

r.Payed, r.Total, r.Total - r.Payed ToPay

FROM Clients c JOIN Rentals r ON c.ID = r.ClientID AND r.DateBegin BETWEEN @DateBegin AND @DateEnd

JOIN RentEquip re ON r.ID = re.RentalID JOIN Equipments e ON re.EquipID = e.ID

JOIN EquipCategory ec ON e.ID = ec.EquipID JOIN Categories cat ON ec.CategoryID = cat.ID) queryD\_table

GROUP BY RentalID, FullName, Phone, DateBegin, DAteEnd, Payed, Total, ToPay

Весь используемый код представлен в приложении.

# **Создание и реализация оконного приложения для работы с базой данных.**

Для решения поставленной задачи решено было воспользоваться технологией WPF (Windows Presentation Foundation) и языком программирования C# (C-Sharp), ведь это отличные инструменты для написания клиентского приложения для работы с базой данных SQL Server по нескольким причинам.

Во-первых, WPF позволяет создавать современные и красивые интерфейсы для приложения. WPF имеет широкий набор инструментов для создания пользовательских элементов управления, анимаций, стилей и тем оформления. Это позволяет разработчикам создавать более эффективные и интуитивно понятные пользовательские интерфейсы для приложения.

Во-вторых, C# является объектно-ориентированным языком программирования с широкой поддержкой .NET Framework. Это означает, что C# обладает мощными функциями для работы с базами данных, такими как ADO.NET. ADO.NET предоставляет различные инструменты для доступа к базам данных, включая Entity Framework, LINQ to SQL и другие. Это позволяет легко и эффективно работать с базой данных SQL Server.

Кроме того, WPF и C# позволяют разработчикам легко создавать графические интерфейсы для взаимодействия с базой данных. Они также позволяют легко создавать связи между элементами пользовательского интерфейса и базой данных.

Наконец, оба инструмента имеют обширную документацию и активное сообщество разработчиков, что позволяет быстро решать возникающие проблемы и быстро разрабатывать качественное приложение.

При разработке Windows-приложений общепринятым является использование паттерна MVVM (Model-View-ViewModel). Однако, из-за ограниченного опыта, не удалось полностью раскрыть потенциал этого паттерна при разработке.

Были созданы следующие модели, характеризующие сущности базы данных (рисунок 3).



Рисунок 3 – Классы представляющие модель данных ADO.NET.

Были разработаны классы, представляющие визуализацию вопросов курсовой (Рисунок 4).

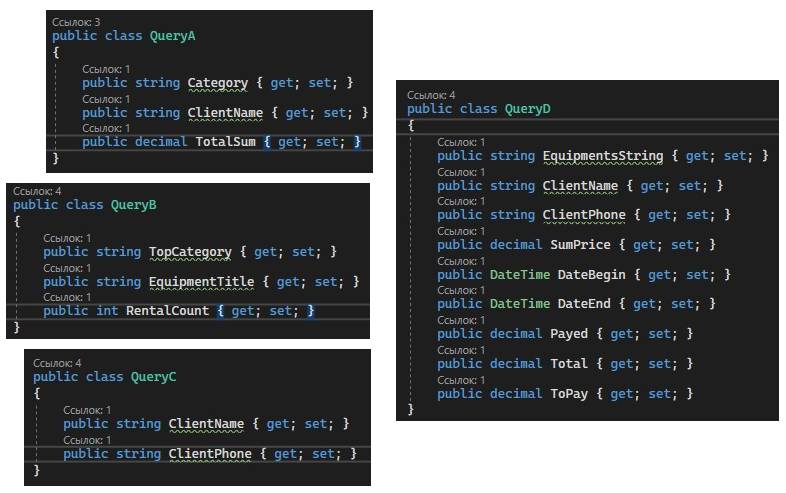


Рисунок 4 – Классы представляющие модель ответов на вопросы курсовой.

Был разработан класс DataWorker.cs (), содержащий все методы для взаимодействия с базой данных.

Для улучшения визуального восприятия в приложении были использованы цвета и темы “Material Design” от Google. Основным цветом был выбран Индиго.

Все окна были выполнены в едином стиле внутри элемента <Border>

(Рисунок 5).

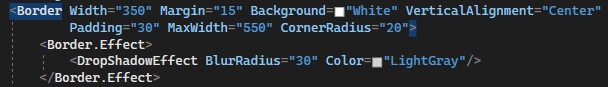


Рисунок 5 – Код элемента Border для MainWindow

Окна, которые должны быть единственным экземпляром в приложении, были инициализированы с использованием паттерна Singleton, который гарантирует, что в системе будет только один экземпляр данного объекта. Для остальных окон использовался обычный метод объявления, без использования Singleton.

Класс подключения к базе данных так же был выполнен в паттерне Singleton (Рисунок 6).

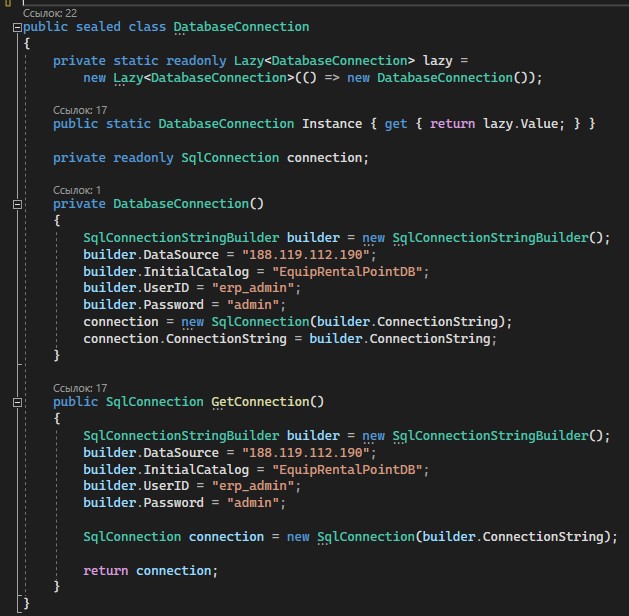


Рисунок 6 – Класс, обеспечивающий соединение с базой данных.

# **Заключение.**

В результате выполнения курсовой работы была разработана база данных для предметной области "Пункт проката", которая позволяет хранить информацию о посетителях, инвентаре и выдаче его во временное пользование, а так же позволяет контролировать внесенную оплату.

При проведении анализа предметной области были выявлены основные требования к базе данных, которые были учтены при ее разработке. Были определены основные сущности и их связи, а также атрибуты, необходимые для хранения информации.

Для реализации функционала были использованы язык программирования C# и технологии WPF, СУБД SQL Server.

В результате работы была создана база данных, которая позволяет контролировать выдачу инвентаря во временное пользование, хранить информацию о посетителях и проводить учет поступления оплаты. Также было разработано клиентское приложение, облегчающее работу с базой данных.

Таким образом, выполнение курсовой работы позволило получить практические навыки разработки базы данных и приложений на их основе, а также углубить знания в области анализа предметной области и проектирования баз данных.

# **Список литературы.**

1. И.П. Карпова. Базы данных: Учебное пособие**.** СПб: Питер, 2013г. – 240с.
2. М.П. Малыхина. Базы данных: основы, проектирование, использование**.** 2-е изд. СПб: БХВ-Петербург, 2007.
3. В.В. Кирилов, Г.Ю. Громов. Введение в реляционные базы данных. – СПб.: БХВ-Петербург, 2009г. – 464с.
4. Уолтер Шилдс. SQL:быстрое погружение. – СПб.: Питер, 2022г – 224с.
5. Н. А. Литвиненко. Декларативное программирование на языке XAML + C#. WPF проекты. – Москва: Горячая Линия – Телеком, 2022г – 320с.

# **Приложения.**

1. (SQL-Scripts.zip) SQL Скрипты:
   1. (DataBase\_Creation.sql) Создание базы данных.
   2. (DataBase\_Insert.sql) Процедуры занесения значений.
   3. (DataBase\_Query.sql) Вопросы курсовой.
   4. (DataBase\_ForClientApp.sql) Для клиентского приложения.
2. (EquipRentalPointApp.zip) Архив исходного кода приложения.
3. (SQL\_SERVER.zip) Архив с базой данных.
   1. (EquipRentalPointDB.mdf) Файл базы данных.
   2. (EquipRentalPointDB\_log.ldf) Файл логов базы данных.