Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**ИВАНОВСКИЙ ГОСУДАРСТВЕННЫЙ ХИМИКО-ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет техники, управления и цифровой инфраструктуры**

**Кафедра информационных технологий и цифровой экономики**

**КУРСОВОЙ ПРОЕКТ**

**ПО ДИСЦИПЛИНЕ**

**УПРАВЛЕНИЕ ДАННЫМИ**

**Тема:** Разработка базы данных MS SQL

и приложения средствами Visual Studio.

Вариант №3

**Выполнил студент: Крайнова Анастасия Максимовна курс/гр. 3/184**

ФИО

**Проверил: Захарова Наталья Александровна**

ФИО

Иваново 2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**ИВАНОВСКИЙ ГОСУДАРСТВЕННЫЙ ХИМИКО-ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет** ТУиЦИ

**Кафедра** ИТиЦЭ

**Направление** 09.03.02 Информационные системы и технологии

**Профиль** Анализ данных и цифровые финансовые технологии

**Дисциплина** Управление данными

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**Студенту** Крайновой Анастасии Максимовне курс/гр. **3/184**

(фамилия, имя, отчество - полностью)

**Тема:** Разработка базы данных MS SQL и приложения средствами Visual Studio. Вариант №3

Исходные данные: *законодательные и подзаконные нормативные акты; энциклопедическая и справочная литература; статистические и аналитические материалы; монографии; данные профессиональных периодических изданий; интернет-ресурсы; самостоятельно собранные первичные материалы.*

Содержание работы:

Введение: *цель и задачи работы; предмет и объект исследования; обоснование структуры работы.*

Теоретическая часть должна содержать: *системный анализ предметной области, инфологическую модель БД, список запросов, которые должны быть реализованы в БД, список требований к приложению*

В практической части необходимо: *сгенерировать БД, реализовать SQL-запросы, разработать программное приложение к БД.*

Заключение*: основные выводы по проекту, обобщение основных предложений и рекомендаций.*

Руководитель работы: *Захарова Н.А.*

**Календарный план-график**

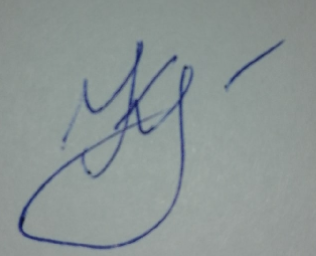
|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов выполнения курсовой работы | Срок выполнения этапов | Примечание |
| 1 | Исследование предметной области и составление логической модели БД | 17.02.20 – 17.03.20 |  |
| 2 | Создание физической модели БД | 18.03.20 – 31.03.20 |  |
| 3 | Генерация БД средствами MSSQL и реализация SQL-запросов | 01.04.20 – 14.04.20 |  |
| 4 | Разработка приложения средствами VisualStudio | 15.04.20 – 12.05.20 |  |
| 5 | Оформление пояснительной записки | 13.05.20 – 17.05.20 |  |
| 6 | Защита курсового проекта | 18.05.20 - 23.05.20 |  |

Дата выдачи задания 17.02.2020

Дата представления законченной работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись /Ф.И.О.)

Студент  Крайнова А.М

(Подпись /Ф.И.О.)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc58923449)

[1 СИСТЕМНЫЙ АНАЛИЗ 6](#_Toc58923450)

[1.1Описание предметной области 6](#_Toc58923451)

[1.2 Логическая модель «Как есть» 8](#_Toc58923452)

[1.3 Логическая модель «Как должно быть» 9](#_Toc58923453)

[1.4 Список запросов, которые должны быть реализованы в БД 11](#_Toc58923454)

[1.5 Набор требований, которым должно отвечать разрабатываемое приложение 11](#_Toc58923455)

[2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 13](#_Toc58923456)

[3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 24](#_Toc58923457)

[3.1 Таблица «Document table» 25](#_Toc58923458)

[3.2 Таблица «Categories products table» 26](#_Toc58923459)

[3.3 Таблица «Price History table 26](#_Toc58923460)

[3.4 Таблица «Client table» 27](#_Toc58923461)

[3.5 Поиск записей 27](#_Toc58923462)

[3.6 Экспорт таблицы Excel 28](#_Toc58923463)

[ЗАКЛЮЧЕНИЕ 29](#_Toc58923464)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 30](#_Toc58923465)

[ПРИЛОЖЕНИЕ А. СКРИПТ БАЗЫ ДАННЫХ 31](#_Toc58923466)

[ПРИЛОЖЕНИЕ Б. ПЕРЕЧЕНЬ ФАЙЛОВ 58](#_Toc58923467)

# ВВЕДЕНИЕ

Качество проектирования базы данных может влиять на работу с ней. С хорошо спроектированной базой данных легче работать, легче писать к ней запросы.

Под БД понимается набор сведений, хранящихся некоторым упорядоченным способом. Можно сравнить базу данных со шкафом, в котором хранятся документы. Иными словами, база данных - это хранилище данных.

Для качественного проектирования базы данных существуют различные методики, различные последовательности шагов или этапов, которые во многом похожи. И в целом мы можем выделить следующие этапы:

* Выделение сущностей и их атрибутов, которые будут храниться в базе данных, и формирование по ним таблиц. Атомизация сложных атрибутов на более простые.
* Определение уникальных идентификаторов (первичных ключей) объектов, которые хранятся в строках таблицы
* Определение отношений между таблицами с помощью внешних ключей
* Нормализация базы данных.

Важным вопросом является создание автоматизированных систем, которые требуют минимальных человеческих затрат и выполняют огромные расчеты за доли секунд. Примером такой автоматизированной системы является Среда SQL Server Management Studio в которой будем проектировать БД.

Данный курсовой проект описывает процесс создания БД, проектирование приложения и написание запросов. Построение инфологических моделей «как есть» и «как должно быть». В последней части-работа с таблицами, добавление, удаление и редактирование записей, поиск записей в БД и отчетность.

ОСНОВНАЯ ЧАСТЬ

# 1 СИСТЕМНЫЙ АНАЛИЗ

ЗАДАНИЕ «ЛОМБАРД»

## 1.1Описание предметной области

Вы работаете в ломбарде. Вашей задачей является отслеживание финансовой стороны его работы.

Деятельность компании организована следующим образом: к вам обращаются различные лица с целью получения денежных средств под залог определенных товаров. У каждого из приходящих к вам клиентов вы запрашиваете фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара вы определяете сумму, которую готовы выдать на руки клиенту, а также свои комиссионные. Кроме того, определяете срок возврата денег. Если клиент согласен, то ваши договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается у вас. В случае, если в указанный срок не происходит возврата денег, товар переходит в вашу собственность.

*Таблицы*

Клиенты (Код клиента, Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Код категории товаров, Название, Примечание). Сдача в ломбард (Код, Код категории товаров, Код клиента, Описание

товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

*Развитие постановки задачи*

После перехода прав собственности на товар ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке. (Например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы.) Помимо текущей цены, нужно хранить все возможные значения цены для данного товара.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Рассмотрим подробнее, что такое системный анализ.

С точки зрения проектирования БД в рамках системного анализа, необходимо осуществить первый этап, то есть провести подробное словесное описание объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами. Желательно, чтобы данное описание позволяло корректно определить все взаимосвязи между объектами предметной области.

В общем случае существуют два подхода к выбору состава и структуры предметной области:

* Функциональный подход — он реализует принцип движения "от задач" и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД.
* Предметный подход — когда информационные потребности будущих пользователей БД жестко не фиксируются.

Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который, с одной стороны, ориентирован на конкретные задачи или функциональные потребности пользователей, а с другой стороны, учитывает возможность наращивания новых приложений.

Системный анализ должен заканчиваться подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая, должна храниться в БД, формулировкой конкретных задач, которые будут решаться с использованием данной БД с кратким описанием алгоритмов их решения, описанием выходных документов, которые должны генерироваться в системе, описанием входных документов, которые служат основанием для заполнения данными БД. Далее представим логическую модель «Как есть».

## 1.2 Логическая модель «Как есть»

Первым шагом при создании логической модели БД является построение диаграммы ERD (Entity Relationship Diagram). ERD-диаграммы состоят из трех частей: сущностей, атрибутов и взаимосвязей. Сущностями являются существительные, атрибуты - прилагательными или модификаторами, взаимосвязи - глаголами.

ERD-диаграмма позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации.

ERD-диаграммы можно подразделить на отдельные куски, соответствующие отдельным задачам, решаемым проектируемой системой. Это позволяет рассматривать систему с точки зрения функциональных возможностей, делая процесс проектирования управляемым.

Исходя из представленного задания, составим начальную модель.

Клиенты (Код клиента, Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Код категории товаров, Название, Примечание).

Сдача в ломбард (Код, Код категории товаров, Код клиента, Описание товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

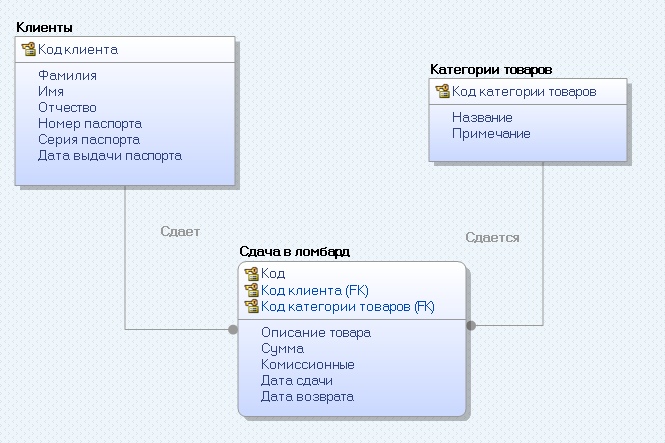


Рисунок 1.1 – Модель «как есть»

Исходя из проделанного анализа, мы видим, что появилась новая сущность «История цен».

* В Сущность «История цен» входят атрибуты: (Код, Код истории цен, Код клиента, Код категории товаров, Дата выставления цены, Цена, Продан)

## 1.3 Логическая модель «Как должно быть»

Клиенты (Код клиента, Фамилия, Имя, Отчество, Номер паспорта, Серия паспорта, Дата выдачи паспорта).

Категории товаров (Код категории товаров, Название, Примечание).

Сдача в ломбард (Код, Код категории товаров, Код клиента, Описание товара, Дата сдачи, Дата возврата, Сумма, Комиссионные).

История цен (Код, Код истории цен, Код клиента, Код категории товаров, Дата выставления цены, Цена, Продан).

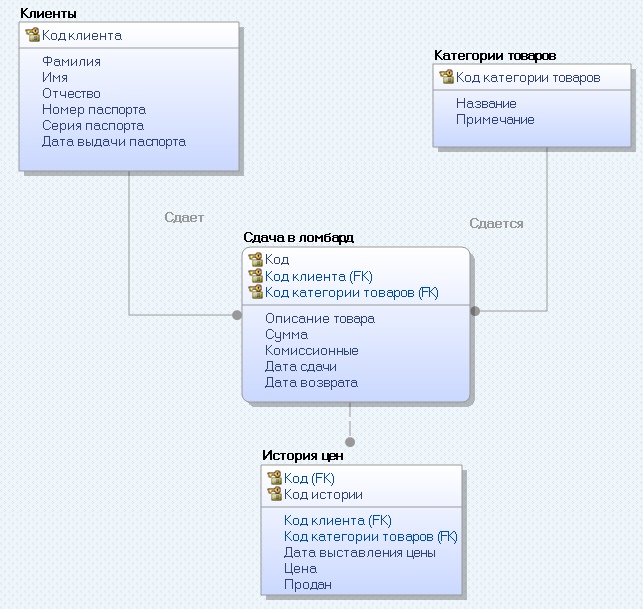


Рисунок 1.2 – Построение модели в нотации IDEF1x «Как должно быть»

Главной задачей системного анализа является разрешение проблемной ситуации, возникшей перед объектом проводимого системного исследования. Системный анализ занимается изучением проблемной ситуации, выяснением её причин, выработкой вариантов её устранения, принятием решения и организацией дальнейшего функционирования системы, разрешающего проблемную ситуацию. Начальным этапом любого системного исследования является изучение объекта проводимого системного анализа с последующей его формализацией. На этом этапе возникают задачи, в корне отличающие методологию системных исследований от методологии других дисциплин, а именно, в системном анализе решается двуединая задача. С одной стороны, необходимо формализовать объект системного исследования, с другой стороны, формализации подлежит процесс исследования системы, процесс постановки и решения проблемы.

## 1.4 Список запросов, которые должны быть реализованы в БД

1. Вывести имя и описание товара, id категории которого, больше 2?
2. Вывести имя категории товара, где нет описания?
3. Вывести количество клиентов с именем Олег?
4. Вывести сумму и id категории, где сумма больше 50000?
5. Вывести максимальную сумму комиссионных?
6. Вывести среднюю стоимость товаров?
7. Вывести имя и сумму заказа клиента?
8. Вывести id документов товары, в которых можно продавать?
9. Вывести проданные товары?
10. Вывести имя, id клиента и цену продажи?
11. Вывести имена клиентов и комиссионные для товаров, которые можно продать?
12. Вывести имя клиента, который заплатил больше всего комиссионных?
13. Вывести имена клиентов в алфавитном порядке?
14. Вывести цены и категории товаров, которые сейчас?
15. Вывести информацию о клиенте с фамилией, начинающейся на 'О'?

## 1.5 Набор требований, которым должно отвечать разрабатываемое приложение

* Обеспечение ведения списка клиентов:
* Обеспечение ведения списка документов:
* Обеспечение ведения списка комиссионных:
* Обеспечение ведения списка историй цен:
* Обеспечить возможность сортировки по всем имеющимся полям:
* Обеспечить возможность поиска записей в базе данных:
* Обеспечить выгрузку данных в таблицы Excel;

## 2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Основная задача данного этапа заключается в преобразовании инфологической модели в физическую модель базы данных, преобразовании ее в схему базы данных для MS SQL Server 2008, используя Transact-SQL, проверки корректности сделанных преобразований и соответствию исходных требований, а также проектированию приложения к базам данных.

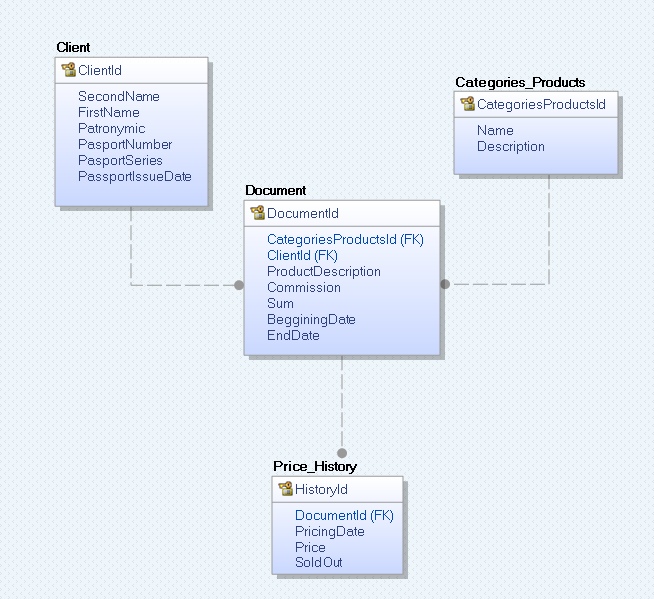


Рисунок 2.1 – Физическая модель

Физические модели данных служат для отображения моделей данных. Основными понятиями модели данных являются поле, логическая запись, логический файл. Слово «логический» введено, чтобы отличать понятия, относящиеся к логической модели данных, от понятий, относящихся к физической модели данных.

Физическая модель данных зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД.

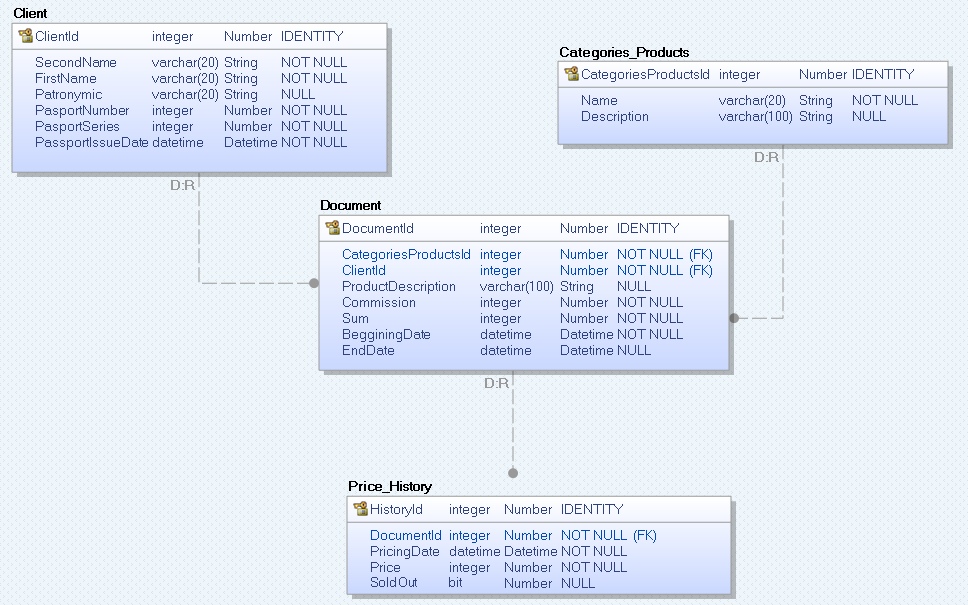


Рисунок 2.2 – Физическая модель со всеми типами данных

Физическая модель данных, как правило, создается на основе логической, поэтому каждому объекту логической модели соответствует объект физической модели (хотя соответствие может быть неоднозначным). В физической модели данных сущности логической модели данных соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы. Кроме перечисленных выше объектов, физическая модель может содержать объекты, тип которых зависит от СУБД: индексы, представления, последовательности, триггеры, процедуры и т.п.

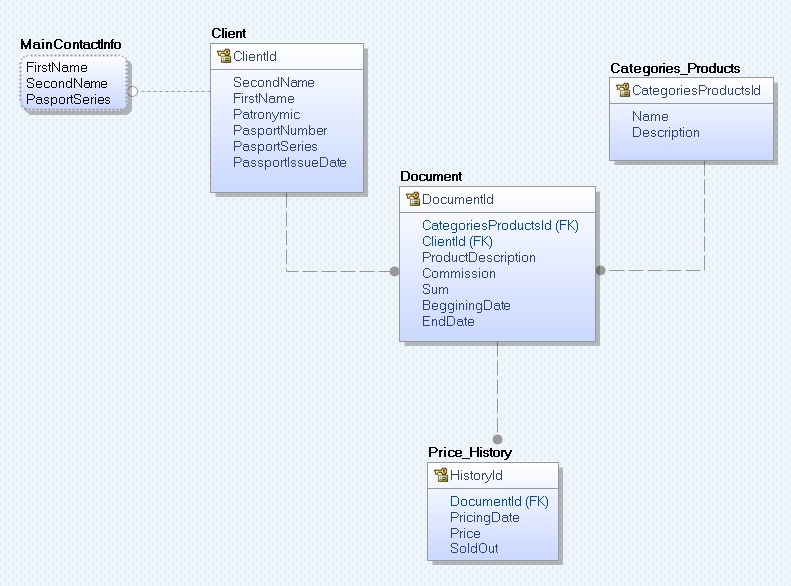
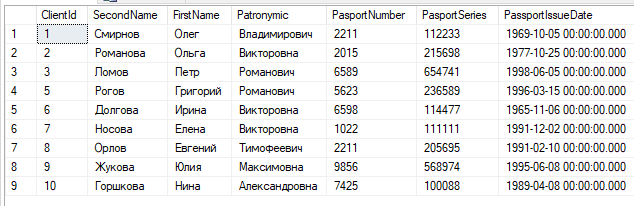


Рисунок 2.3 – Представление

Далее представлены таблицы базы данных и запросы по заполнению таблиц.



2.4 – Таблица Client

Запрос SQLна добавление записи в таблицу Client

INSERT INTO Client(SecondName, FirstName, Patronymic, PasportNumber, PasportSeries, PassportIssueDate)  
VALUES('Фамилия', 'Имя', 'Отчество', 'Номер паспорта', 'Серия паспорта' 'Дата выдачи паспорта')

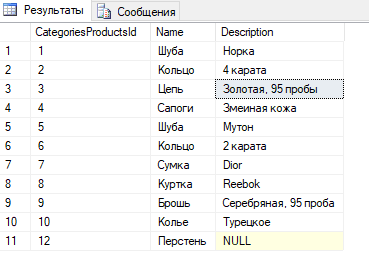


Рисунок 2.5 – Таблица Categories\_Products

Запрос SQL на добавление записи в таблицу Categories\_Products

INSERT INTO Categories\_Products(Name, Description)

VALUES('Название', 'Описание')

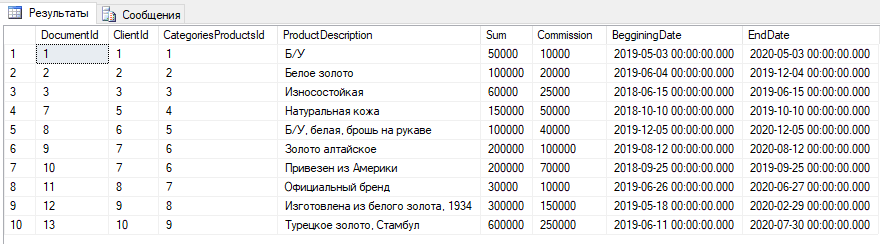


Рисунок 2.6 – Таблица Document

Запрос SQL на добавление записи в таблицу Document

INSERT INTO Document(Client, CategoriesProductsId, ProductDescription, Sum, Commission, BegginingDate, EndDate)  
 VALUES(‘Клиент’, ‘Категории товаров’, ‘Описание продукта’, ‘Сумма’, ’Комиссионные’, ’Дата начала’, ’Дата окончания’)

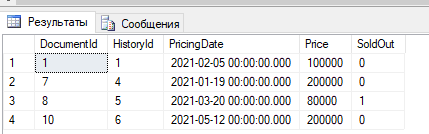


Рисунок 2.7 – Таблица Price\_History

Запрос SQLна добавление записи в таблицу Price\_History

INSERT INTO Price\_History (History, PricingDate, Price,

SoldOut)  
 VALUES('История', ‘Дата выставления цены’, ‘Цена’, ‘Продан’)

Далее проверяем работу представлений.

Представления -это таблицы чье содержание выбирается или получается из других таблиц. Они работают в запросах и операторах DML точно также как и основные таблицы, но не содержат никаких собственных данных. Представления - подобны окнам, через которые вы просматриваете информацию (как она есть, или в другой форме, как вы потом увидите), которая фактически хранится в базовой таблице. Представление -это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления.

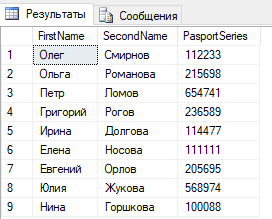


Рисунок 2.8 – Вывод представления

Создадим ограничение для таблицы Document. Нам нужно чтобы нельзя было добавить запись с неверной датой создания и неверной датой окончания, так же что дата начала должна быть раньше даты конца.

Создание ограничений:

ALTER TABLE Document add constraint ValidDates CHECK (BeginningDate < EndDate)  
 Проверка ограничения ValidDates:  
INSERT INTO Document (ClientID, BeginningDate, EndDate)  
VALUES (5, '01.01.2023', '01.01.2013')

1. Вывести имя и описание товара id категории которого больше 2?  
   Select Name,Description from Categories\_Products Where CategoriesProductsId > 2

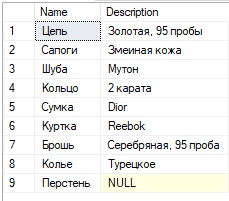


Рисунок 2.9 – Вывод запроса

2. Вывести имя категории товара, где нет описания?

Select Name,Description from Categories\_Products Where Description = nul



Рисунок 2.10 – Вывод запроса

1. Вывести количество клиентов с именем Олег?  
   Select count(ClientId) as Count from Client where FirstName ='Олег'



Рисунок 2.11 – Вывод запроса

1. Вывести сумму и id категории где сумма больше 50000?  
   Select d.Sum, d.CategoriesProductsId from Document d where d.Sum > 50000

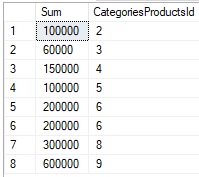


Рисунок 2.12 – Вывод запроса

1. Вывести максимальную сумму комиссионных?  
   Select MAX(Commission) as Max from Document



Рисунок 2.13 – Вывод запроса

1. Вывести среднюю стоимость товаров?  
   Select AVG(sum)as avg from Document



Рисунок 2.14 – Вывод запроса

1. Вывести имя и сумму заказа клиента?  
   Select c.FirstName,Sum from Document d  
   Join Client c on d.ClientId = c.ClientId



Рисунок 2.15 – Вывод запроса

1. Вывести id документов товары в которых можно продавать  
   Select DocumentId From Document Where EndDate < GETDATE()

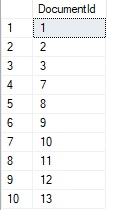


Рисунок 2.16 – Вывод запроса

1. Вывести проданные товары?  
   Select Name from Price\_History ph  
   Join Document d on d.DocumentId = ph.DocumentId  
   Join Categories\_Products cp on cp.CategoriesProductsId = d.CategoriesProductsId  
   Where ph.SoldOut = 'true'



Рисунок 2.17 – Вывод запроса

1. Вывести имя, id клиента и цену продажи?  
   Select FirstName, c.ClientId, Price From Document d  
   Join Client c on d.ClientId = c.ClientId  
   Join Price\_History ph on ph.DocumentId = d.DocumentId  
   Where EndDate < GETDATE() AND ph.SoldOut = 'true'



Рисунок 2.18 – Вывод запроса

1. Вывести имена клиентов и комиссионные для товаров которые можно продать?  
   Select c.FirstName, d.Commission from Client c  
   Join Document d on d.ClientId = c.ClientId  
   Where d.EndDate <GETDATE()  
   Order by d.Commission

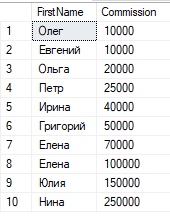


Рисунок 2.19 – Вывод запроса

1. Вывести имя клиента который заплатил больше всего комиссионных?  
   Select top 1 c.FirstName, max(d.Commission) as Max from Client c  
   Join Document d on d.ClientId = c.ClientId  
   Group by c.FirstName  
   Order by Max desc



Рисунок 2.20 – Вывод запроса

1. Вывести имена клиентов в алфавитном порядке?  
   Select FirstName from Client  
   Order by FirstName asc



Рисунок 2.21 – Вывод запроса

1. Вывести цены и категории товаров которые сейчас продаются?  
   Select Name, Price From Document d  
   Join Price\_History ph on ph.DocumentId = d.DocumentId  
   Join Categories\_Products cp on cp.CategoriesProductsId = d.CategoriesProductsId  
   Where EndDate < GETDATE() AND ph.SoldOut = 'false'  
   Group by Price, PricingDate, Name Having PricingDate = MAX(PricingDate)



Рисунок 2.22 – Вывод запроса

1. Вывести информацию о клиенте с фамилией начинающейся на 'О'  
   Select \* from Client Where SecondName like 'О%'



Рисунок 2.23 – Вывод запроса

На данном этапе мы спроектировали БД с использование Erwin. Для этого создали логическую модель данных. Также были написаны запросы и представлены фрагменты вывода данных запросов.

# 3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

На третьем этапе необходимо спроектировать приложение-ассистент для быстрой работы с базой данных. Проектирование приложения основывалось по следующим требованиям:

* Обеспечение ведения списка клиентов:
* Обеспечение ведения списка документов:
* Обеспечение ведения списка комиссионных:
* Обеспечение ведения списка историй цен:
* Обеспечить возможность сортировки по всем имеющимся полям:
* Обеспечить возможность поиска записей в базе данных:
* Обеспечить выгрузку данных в таблицы Excel;

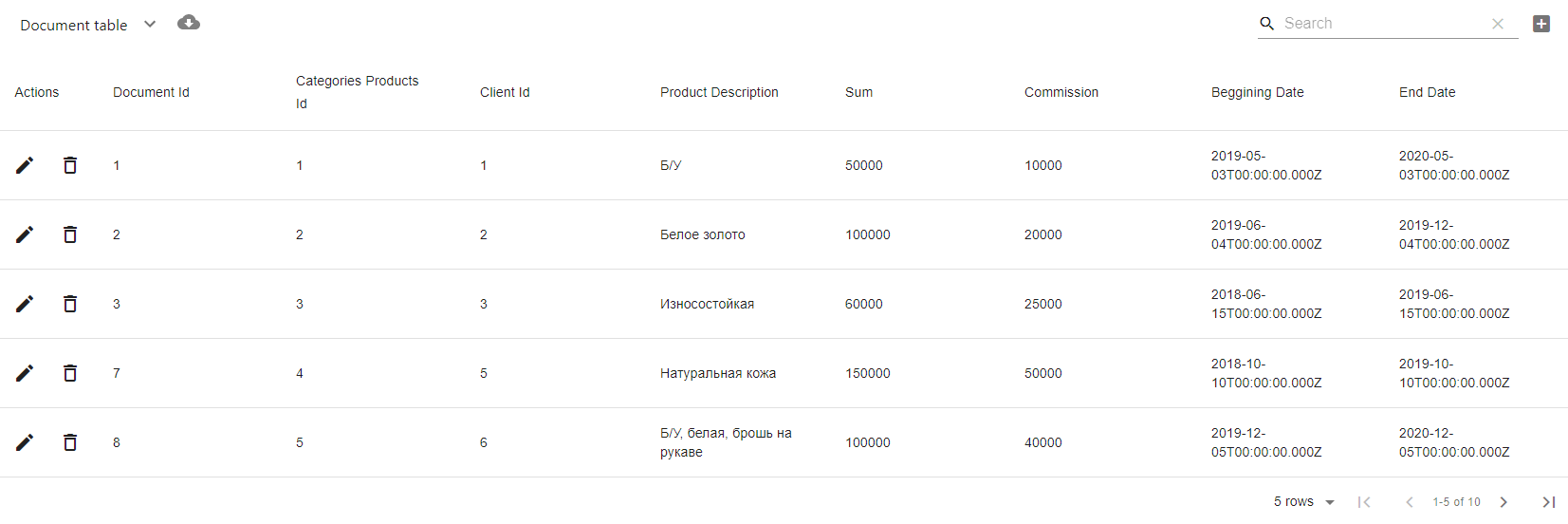


Рисунок 3.1 – Таблица «Document table»

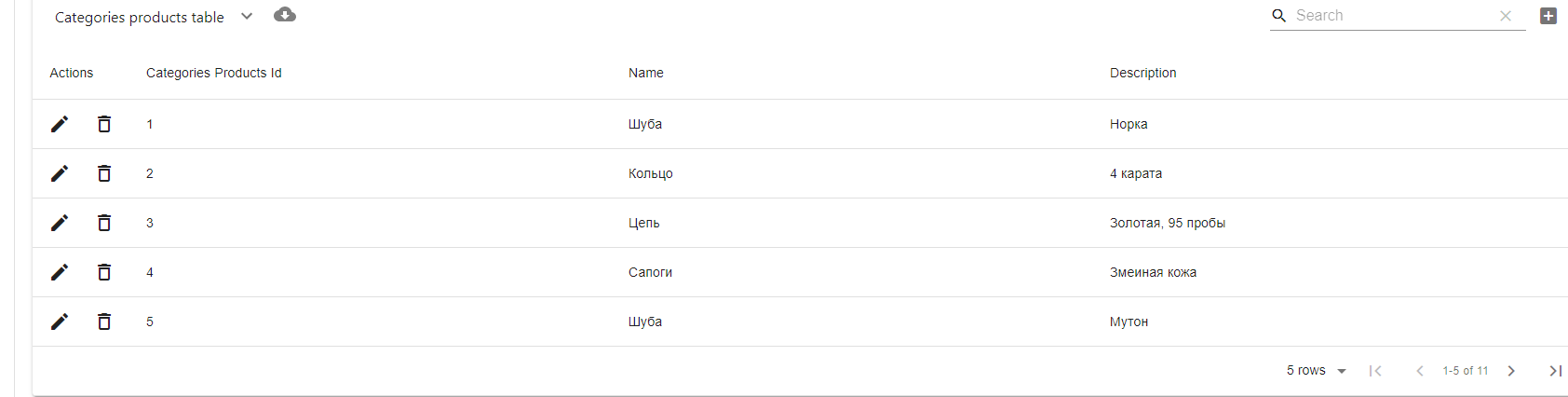


Рисунок 3.2 – Таблица «Categories products table»

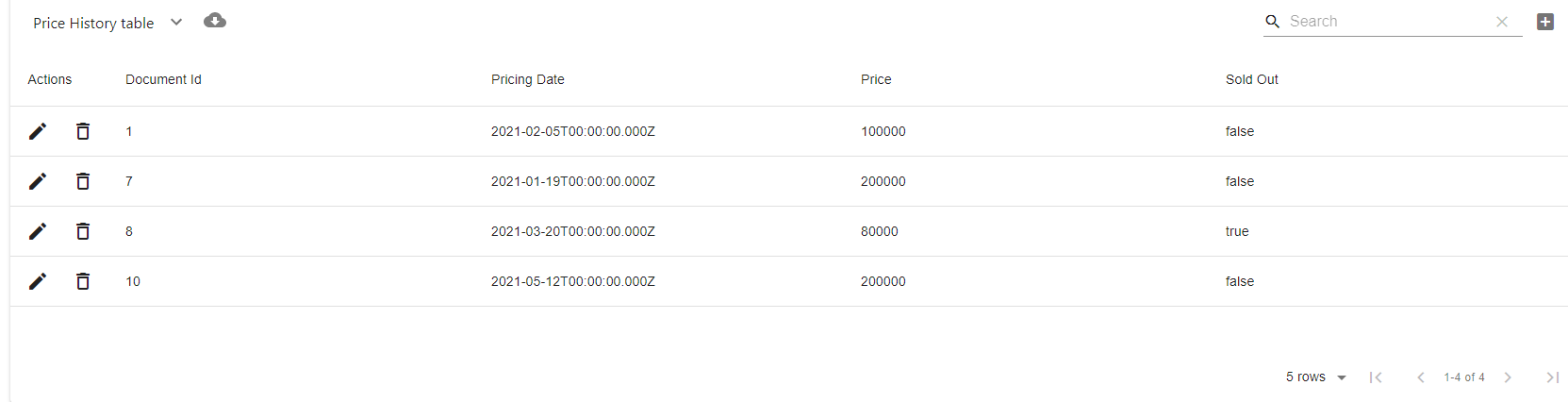


Рисунок 3.3 – Таблица «Price History table»

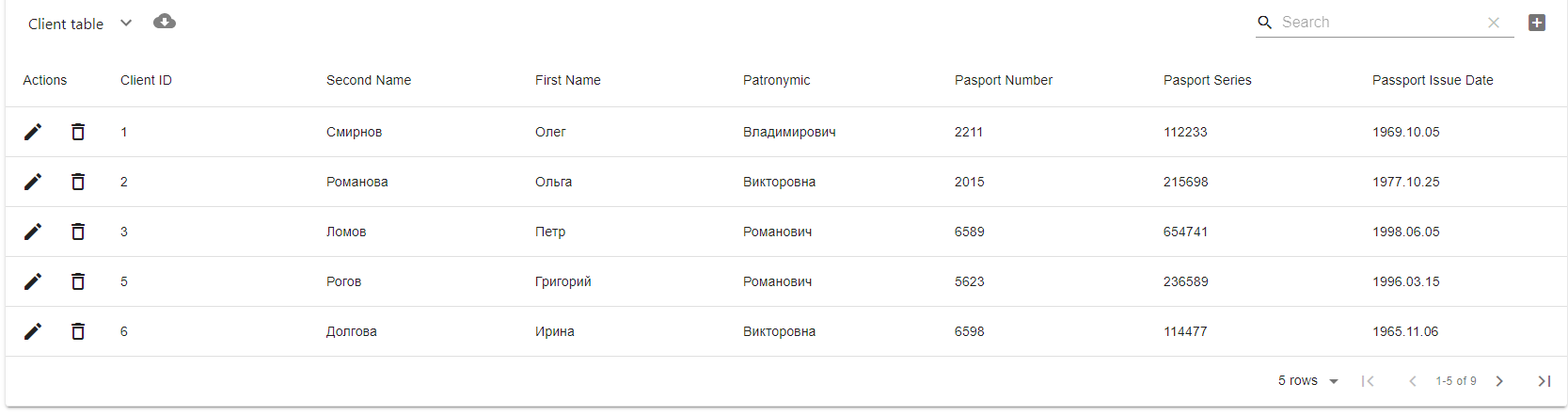


Рисунок 3.4 – Таблица «Client table»

Далее рассмотрим подробнее вид окон и работу с разными таблицами:

## 3.1 Таблица «Document table»

Кнопка «+» в правом верхнем углу добавляет новую запись в таблицу (Рисунок 3.5).

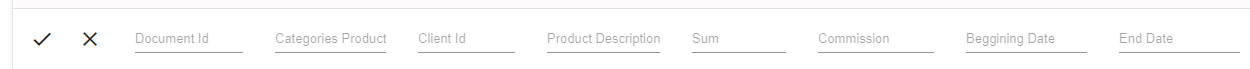


Рисунок 3.5 – Добавление

Кнопка «карандаш» загружает в форму данные строки, чтобы ее изменить (Рисунок 3.6).



Рисунок 3.6 – Изменение

Кнопка «Корзина» удаляет выбранную запись из таблицы (Рисунок 3.7).

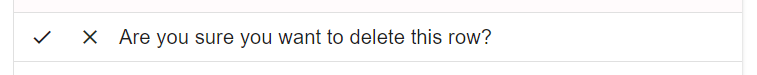


Рисунок 3.7 – Удаление записи

## 3.2 Таблица «Categories products table»

Кнопка «+» в правом верхнем углу добавляет новую запись в таблицу (Рисунок 3.8).

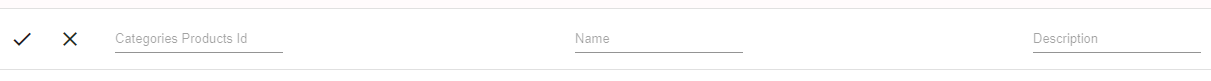


Рисунок 3.8 – Добавление

Кнопка «Карандаш» загружает в форму «Categories products table» данные строки, чтобы ее изменить (Рисунок 3.9).

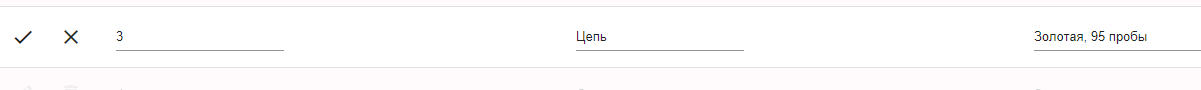


Рисунок 3.9 – Изменение

Кнопка «Корзина» удаляет выбранную запись из таблицы (Рисунок 3.10).

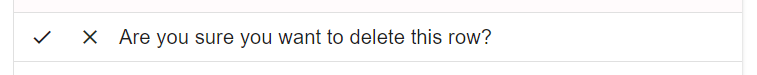


Рисунок 3.10 – Удаление

## 3.3 Таблица «Price History table

Кнопка «+» в правом верхнем углу добавляет новую запись в таблицу (Рисунок 3.11).



Рисунок 3.11 – Добавление

Кнопка «Карандаш» загружает в форму «Price History table» данные строки, чтобы ее изменить (Рисунок 3.12).



Рисунок 3.12 – Изменение

Кнопка «Корзина» удаляет выбранную запись из таблицы (Рисунок 3.13).

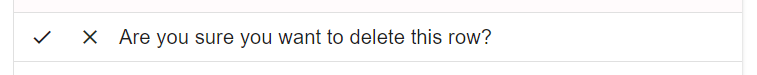


Рисунок 3.13 – Удаление

## 3.4 Таблица «Client table»

Кнопка «+» в правом верхнем углу добавляет новую запись в таблицу (Рисунок 3.14).

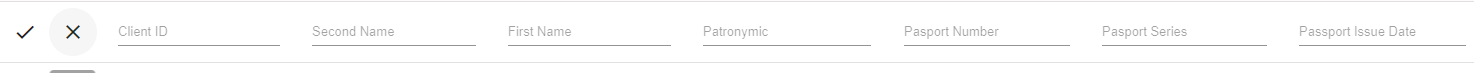


Рисунок 3.14 – Добавление

Кнопка «Карандаш» загружает в форму «Client table» данные строки, чтобы ее изменить (Рисунок 3.15).



Рисунок 3.15 – Изменение

Кнопка «Корзина» удаляет выбранную запись из таблицы (Рисунок 3.16).

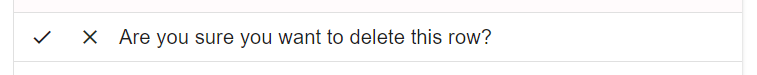


Рисунок 3.16 – Удаление

## 3.5 Поиск записей

Для поиска нужных записей в БД, необходимо выбрать нужную таблицу, затем при заполнении строки поиска записи, будут отображаться результаты поиска. Строка поиска представлена на рисунке 3.17.



Рисунок 3.17 – Окно поиска

## 3.6 Экспорт таблицы Excel

Для выгрузки таблицы Excelнеобходимо сначала выбрать таблицу для сохранения, затем нажать на кнопку ‘облака’, после этого сразу загрузиться, и откроется файл excelс нужной таблицей.

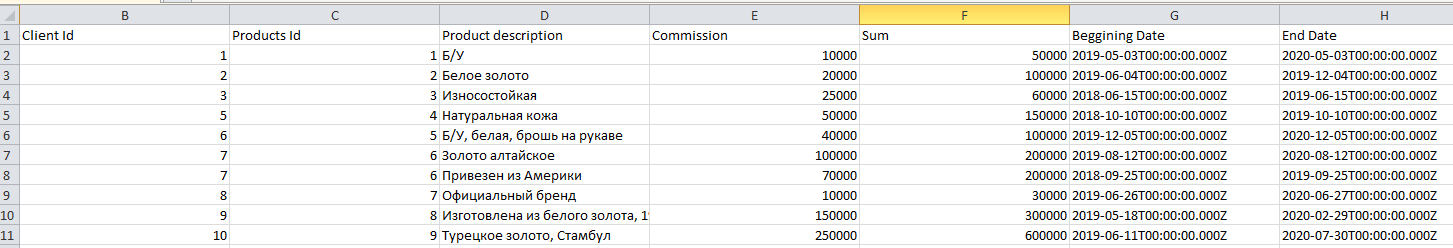


Рисунок 3.29 – Результат экспорта в MicrosoftExcel

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта по заданию «Ломбард» были выполнены различные задачи по анализу поставленной задачи, создания физической модели.

На основе физической модели была создана реляционная база данных, в которой дополнительно были созданы триггеры, ограничения, хранимые процедуры для корректной работы базы данных.

Также, было спроектировано приложение-ассистент для работы с базой данных. В приложении были реализованы функции просмотра, добавления, изменения и удаления записей базы данных, поиска записей, фильтрация для записей и экспорт таблиц.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Требования к оформлению квалификационных работ: метод. указания для студентов по направлению 230200 «Информационные системы» / Сост.: А.П.Власов, Н.А. Марчук: Иван. гос. хим.-технол. ун-т. – Иваново, 2010, 35 с.
2. Проектирование баз данных: лабораторный практикум / Э.Г. Галиаскаров, А.Ю. Крылов; Иван. гос. хим.-технол. ун-т.- Иваново, 2012.- 96 с.
3. Разработка приложений баз данных: лабораторный практикум / Э.Г. Галиаскаров и др.; Иван. гос. хим.-технол. ун-т. – Иваново, 2015. – 112 с.
4. Упражнения по SQL<http://sql-ex.ru/>
5. Татьяна Карпова, курс «Базы данных: модели, разработка, реализация» <https://www.intuit.ru/studies/courses/1001/297/info> Сайт «НОУ ИНТУИТ»
6. Учебник <http://www.sql-tutorial.ru/>
7. Учебный сайт <https://metanit.com/sql/postgresql/>

# ПРИЛОЖЕНИЕ А. СКРИПТ БАЗЫ ДАННЫХ

CREATE TABLE [Client]

(

[ClientID] int NOT NULL IDENTITY ( 1,1 ) ,

[Title] varchar(50) NULL ,

[Requisites] varchar(50) NULL ,

[Address] varchar(50) NULL ,

[Telephone] varchar(20) NOT NULL ,

[ContactPerson] varchar(50) NOT NULL

)

ALTER TABLE [Client]

ADD CONSTRAINT [XPKClient] PRIMARY KEY CLUSTERED ([ClientID] ASC)

CREATE TABLE [Contract]

(

[ContractID] int NOT NULL IDENTITY ( 1,1 ) ,

[ClientID] int NOT NULL ,

[BeginningDate] datetime NOT NULL ,

[EndDate] datetime NULL

)

ALTER TABLE [Contract]

ADD CONSTRAINT [XPKContract] PRIMARY KEY CLUSTERED ([ContractID] ASC)

CREATE TABLE [Outlets]

(

[OutletID] int NOT NULL IDENTITY ( 1,1 ) ,

[Floor] int NULL ,

[Area] int NOT NULL ,

[WithConditioners] bit NULL ,

[RentalPrice] int NOT NULL

)

ALTER TABLE [Outlets]

ADD CONSTRAINT [XPKOutlets] PRIMARY KEY CLUSTERED ([OutletID] ASC)

CREATE TABLE [Payment]

(

[PaymentID] int NOT NULL IDENTITY ( 1,1 ) ,

[PaymentDate] datetime NOT NULL ,

[SummPayment] int NOT NULL ,

[ContractID] int NOT NULL

)

ALTER TABLE [Payment]

ADD CONSTRAINT [XPKPayment] PRIMARY KEY CLUSTERED ([PaymentID] ASC)

CREATE TABLE [Rent]

(

[RentalID] int NOT NULL IDENTITY ( 1,1 ) ,

[OutletID] int NOT NULL ,

[ContractID] int NOT NULL

)

ALTER TABLE [Rent]

ADD CONSTRAINT [XPKRent] PRIMARY KEY CLUSTERED ([RentalID] ASC)

CREATE VIEW [CompanyInformation]([Title],[Address],[Telephone])

AS

SELECT [Title],[Address],[Telephone]

FROM [Client]

ALTER TABLE [Contract]

ADD CONSTRAINT [R\_4] FOREIGN KEY ([ClientID]) REFERENCES [Client]([ClientID])

ALTER TABLE [Payment]

ADD CONSTRAINT [R\_5] FOREIGN KEY ([ContractID]) REFERENCES [Contract]([ContractID])

ALTER TABLE [Rent]

ADD CONSTRAINT [R\_6] FOREIGN KEY ([ContractID]) REFERENCES [Contract]([ContractID])

ALTER TABLE [Rent]

ADD CONSTRAINT [R\_7] FOREIGN KEY ([OutletID]) REFERENCES [Outlets]([OutletID])

CREATE TRIGGER tD\_Client ON Client FOR DELETE AS

/\* erwin Builtin Trigger \*/

/\* DELETE trigger on Client \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* erwin Builtin Trigger \*/

/\* Client Contract on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00010398", PARENT\_OWNER="", PARENT\_TABLE="Client"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="ClientID" \*/

IF EXISTS (

SELECT \* FROM deleted,Contract

WHERE

/\* %JoinFKPK(Contract,deleted," = "," AND") \*/

Contract.ClientID = deleted.ClientID

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Client because Contract exists.'

GOTO error

END

CREATE TRIGGER tU\_Client ON Client FOR UPDATE AS

/\* erwin Builtin Trigger \*/

/\* UPDATE trigger on Client \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insClientID int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Client Contract on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000126d4", PARENT\_OWNER="", PARENT\_TABLE="Client"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="ClientID" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ClientID)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Contract

WHERE

/\* %JoinFKPK(Contract,deleted," = "," AND") \*/

Contract.ClientID = deleted.ClientID

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Client because Contract exists.'

GOTO error

END

END

CREATE TRIGGER tD\_Contract ON Contract FOR DELETE AS

/\* erwin Builtin Trigger \*/

/\* DELETE trigger on Contract \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* erwin Builtin Trigger \*/

/\* Contract Rent on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001e4a2", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ContractID" \*/

IF EXISTS (

SELECT \* FROM deleted,Rent

WHERE

/\* %JoinFKPK(Rent,deleted," = "," AND") \*/

Rent.ContractID = deleted.ContractID

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Contract because Rent exists.'

GOTO error

END

/\* erwin Builtin Trigger \*/

/\* Contract Payment on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="ContractID" \*/

IF EXISTS (

SELECT \* FROM deleted,Payment

WHERE

/\* %JoinFKPK(Payment,deleted," = "," AND") \*/

Payment.ContractID = deleted.ContractID

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Contract because Payment exists.'

GOTO error

END

CREATE TRIGGER tI\_Contract ON Contract FOR INSERT AS

/\* erwin Builtin Trigger \*/

/\* INSERT trigger on Contract \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Client Contract on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001643e", PARENT\_OWNER="", PARENT\_TABLE="Client"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="ClientID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ClientID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Client

WHERE

/\* %JoinFKPK(inserted,Client) \*/

inserted.ClientID = Client.ClientID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," and") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30002,

@errmsg = 'Cannot insert Contract because Client does not exist.'

GOTO error

END

END

CREATE TRIGGER tU\_Contract ON Contract FOR UPDATE AS

/\* erwin Builtin Trigger \*/

/\* UPDATE trigger on Contract \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insContractID int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Contract Rent on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000365d4", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ContractID" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ContractID)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Rent

WHERE

/\* %JoinFKPK(Rent,deleted," = "," AND") \*/

Rent.ContractID = deleted.ContractID

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Contract because Rent exists.'

GOTO error

END

END

/\* erwin Builtin Trigger \*/

/\* Contract Payment on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="ContractID" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ContractID)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Payment

WHERE

/\* %JoinFKPK(Payment,deleted," = "," AND") \*/

Payment.ContractID = deleted.ContractID

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Contract because Payment exists.'

GOTO error

END

END

/\* erwin Builtin Trigger \*/

/\* Client Contract on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Client"

CHILD\_OWNER="", CHILD\_TABLE="Contract"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="ClientID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ClientID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Client

WHERE

/\* %JoinFKPK(inserted,Client) \*/

inserted.ClientID = Client.ClientID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Contract because Client does not exist.'

GOTO error

END

END

CREATE TRIGGER tD\_Outlets ON Outlets FOR DELETE AS

/\* erwin Builtin Trigger \*/

/\* DELETE trigger on Outlets \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* erwin Builtin Trigger \*/

/\* Outlets Rent on parent delete restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001030d", PARENT\_OWNER="", PARENT\_TABLE="Outlets"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="OutletID" \*/

IF EXISTS (

SELECT \* FROM deleted,Rent

WHERE

/\* %JoinFKPK(Rent,deleted," = "," AND") \*/

Rent.OutletID = deleted.OutletID

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Outlets because Rent exists.'

GOTO error

END

CREATE TRIGGER tU\_Outlets ON Outlets FOR UPDATE AS

/\* erwin Builtin Trigger \*/

/\* UPDATE trigger on Outlets \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insOutletID int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Outlets Rent on parent update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00011f71", PARENT\_OWNER="", PARENT\_TABLE="Outlets"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="OutletID" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(OutletID)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Rent

WHERE

/\* %JoinFKPK(Rent,deleted," = "," AND") \*/

Rent.OutletID = deleted.OutletID

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Outlets because Rent exists.'

GOTO error

END

END

CREATE TRIGGER tI\_Payment ON Payment FOR INSERT AS

/\* erwin Builtin Trigger \*/

/\* INSERT trigger on Payment \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Contract Payment on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0001565e", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="ContractID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ContractID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Contract

WHERE

/\* %JoinFKPK(inserted,Contract) \*/

inserted.ContractID = Contract.ContractID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," and") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30002,

@errmsg = 'Cannot insert Payment because Contract does not exist.'

GOTO error

END

END

CREATE TRIGGER tU\_Payment ON Payment FOR UPDATE AS

/\* erwin Builtin Trigger \*/

/\* UPDATE trigger on Payment \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insPaymentID int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Contract Payment on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000159ac", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Payment"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="ContractID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ContractID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Contract

WHERE

/\* %JoinFKPK(inserted,Contract) \*/

inserted.ContractID = Contract.ContractID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Payment because Contract does not exist.'

GOTO error

END

END

CREATE TRIGGER tI\_Rent ON Rent FOR INSERT AS

/\* erwin Builtin Trigger \*/

/\* INSERT trigger on Rent \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Outlets Rent on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="000289a8", PARENT\_OWNER="", PARENT\_TABLE="Outlets"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="OutletID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(OutletID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Outlets

WHERE

/\* %JoinFKPK(inserted,Outlets) \*/

inserted.OutletID = Outlets.OutletID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," and") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30002,

@errmsg = 'Cannot insert Rent because Outlets does not exist.'

GOTO error

END

END

/\* erwin Builtin Trigger \*/

/\* Contract Rent on child insert restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ContractID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ContractID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Contract

WHERE

/\* %JoinFKPK(inserted,Contract) \*/

inserted.ContractID = Contract.ContractID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," and") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30002,

@errmsg = 'Cannot insert Rent because Contract does not exist.'

GOTO error

END

END

CREATE TRIGGER tU\_Rent ON Rent FOR UPDATE AS

/\* erwin Builtin Trigger \*/

/\* UPDATE trigger on Rent \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insRentalID int,

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* erwin Builtin Trigger \*/

/\* Outlets Rent on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="0002a2bc", PARENT\_OWNER="", PARENT\_TABLE="Outlets"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_7", FK\_COLUMNS="OutletID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(OutletID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Outlets

WHERE

/\* %JoinFKPK(inserted,Outlets) \*/

inserted.OutletID = Outlets.OutletID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Rent because Outlets does not exist.'

GOTO error

END

END

/\* erwin Builtin Trigger \*/

/\* Contract Rent on child update restrict \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Contract"

CHILD\_OWNER="", CHILD\_TABLE="Rent"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_6", FK\_COLUMNS="ContractID" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ContractID)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Contract

WHERE

/\* %JoinFKPK(inserted,Contract) \*/

inserted.ContractID = Contract.ContractID

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Rent because Contract does not exist.'

GOTO error

END

END

# ПРИЛОЖЕНИЕ Б. ПЕРЕЧЕНЬ ФАЙЛОВ

1. Пояснительная записка к курсовому проекту в папке doc Крайнова пояснительная записка.docx
2. Инфологическая модель в папке erwin logicalModel.erwin
3. Физическая модель базы данных в папке erwin physicalModel.erwin
4. Файлы базы данных в папке db lombard.mdf
5. Папки с исходным кодом приложения Server и Client