**Компьютерная академия «ШАГ»**

**Донецкий филиал**

**Кафедра Разработки программного обеспечения**

**КУРСОВОЙ ПРОЕКТ**

**“** **Программа для склада ”**

**Студента группы ППВ-23-01  
Москалёва Д. А.**

**Научный руководитель:  
Стрюк Д. О.**

**Донецк, 2014**

Техническое задание

Склад хранит товары за деньги. Программа должна обеспечивать прием и выдачу товаров, учет оплаты. Вывод различных отчетов по складу. Когда товар привозят на склад нужно записать: название, занимаемую площадь, время хранения, стоимость хранения, клиента и т.д. Далее товар ставится на хранение и отмечается место, где он хранится. Когда товар вывозят нужно отметить, что место освободилось. Когда товар не забирают в срок, нужно начислять пеню. Нужно организовать удобную систему поиска свободного места на складе. Также нужна система поиска товара при выдаче. По запросу пользователя должен генерироваться финансовый отчет для заданного периода. Также нужны отчеты о загруженности склада, должниках, напоминание о позициях, срок хранения которых заканчивается.

Технологии

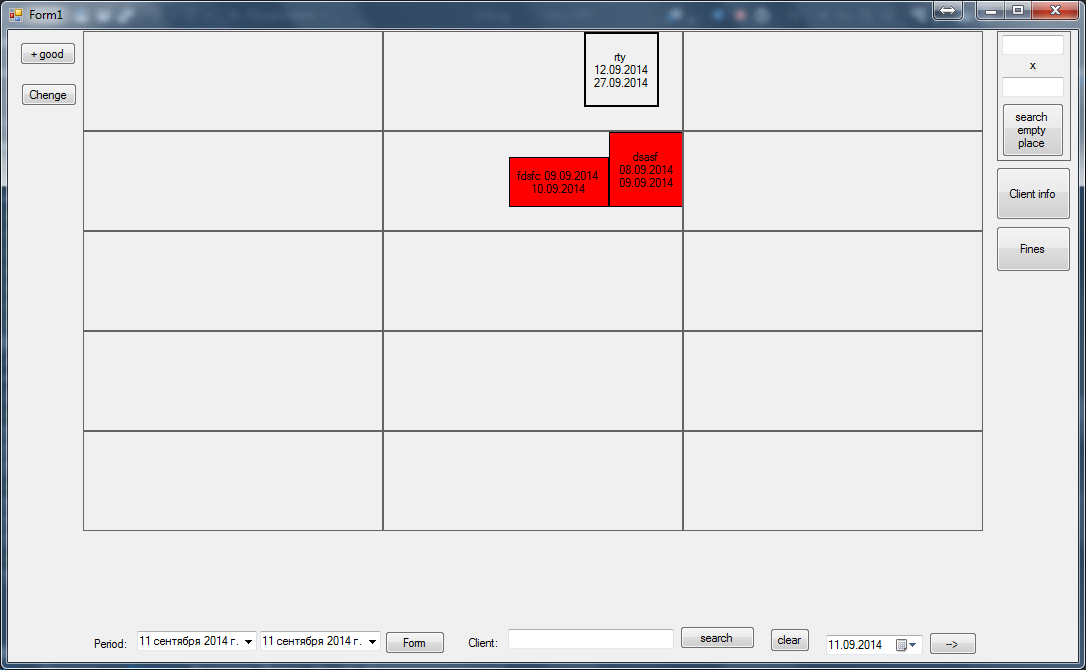
При выполнении курсовой работы были использованы следующие технологии:

* C#
* WinForms
* GDI+
* Transact-SQL
* ADO.NET

Программа написана в API Windows Forms.

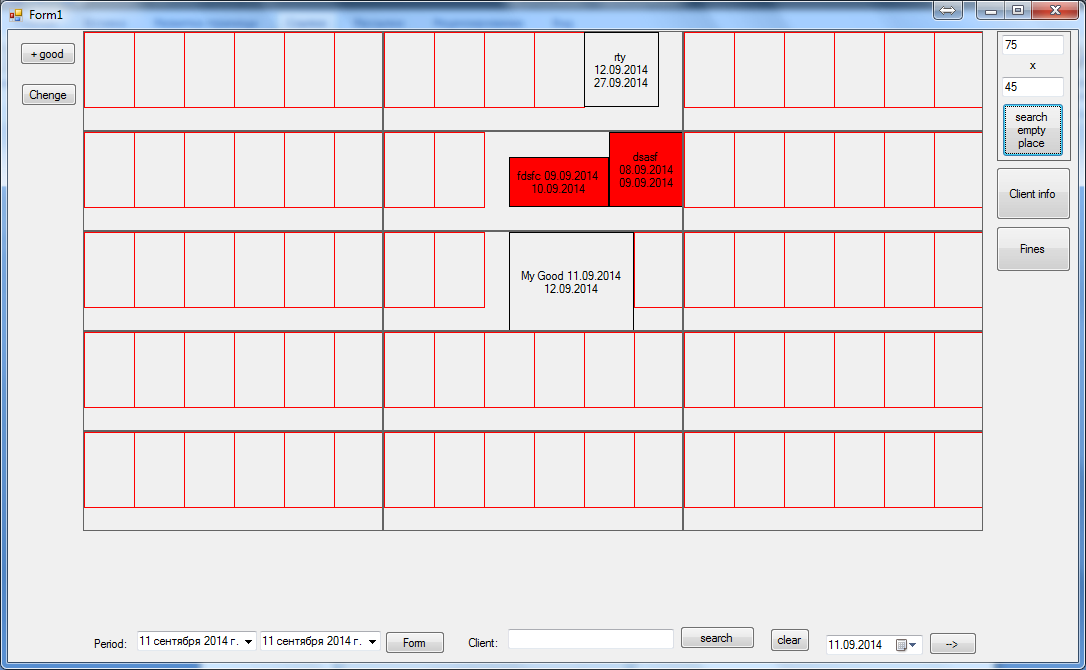
Решение состоит из двух проектов. Первый проект содержит в себе инструментарий для работы с базой данных и адаптирует данные из базы данных для работы в программе (создание новых классов: клиент, товар и т.д.) Второй проект отвечает за внешний вид программы и функциональные возможности.

При запуске программы администратор увидит перед собой окно следующего свойства. На иллюстрации можно увидеть 15 секций для размещения товаров. Товары, находящиеся на хранении отображены на соответствующих их расположению секциям.



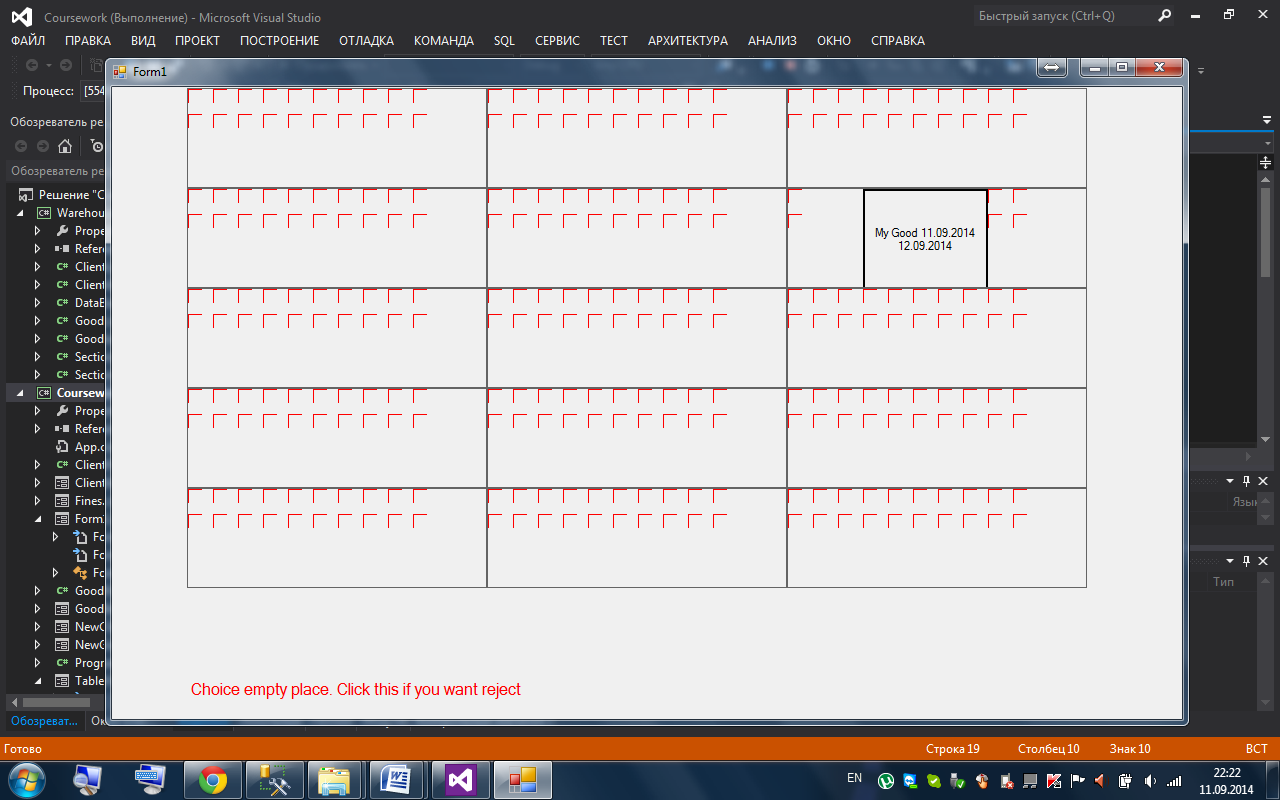
После инициализации основных компонентов выполняется запрос к базе данных . При удачном запросе на присоединения к базе данных происходит экспорт данных из базы данных в экземпляры соответствующих классов, хранящихся в коллекции типа List<T>. После добавлении данных в экземпляры классов заполняются словари Dictionary<Section, Panel> и Dictionary<Button, Good> для связывания графического представления элементов с данными

Для поиска свободных мест используется форма в правом верхнем углу, в которую вбиваются размеры товара (в дальнейшем они округляются в большую сторону до числа, кратного 25). Красными полями выделены возможные размещения товара



Каждая секция в базе данных имеет своё представление заполненности, хранящееся в числе типа double. При заливке информации о секциях в программу число преобразуется в List<bool> из 48 элементов (замолнена-не заполнена каждая из 48 ячеек для хранения товара) при помощи отдельного алгоритма в файле SectionsManager.cs . Каждый товар также имеет свою локацию (алгоритм в Goodsmanager.cs) (может занимать несколько неделимых ячеек). При поиске свободных мест оценивается топография каждой секции и определяются места, в которые может быть помещён товар (функция Offset в Form1.cs). При необходимости, товар может быть перевёрнут для помещения его в свободное пространство.

Для рационального использования места осуществлена возможность перемещения товаров внутри секции или в других секциях. После нажатия кнопки Change нужно выбрать необходимый товар, после чего выполнить клик в одном из красных уголков (иллюстрация ниже). Туда и будет вставлен товар (при необходимости товар переворачивается).

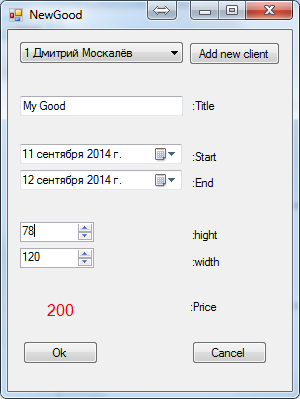


После выбора товара для перемещения выполняется поиск возможных размещений с помощью функции Offset. После выбора нового места происходят следующие пункты:

* Изменение локации товара в соответствующем экземпляре класса
* Изменение локации товара в базе данных
* Изменение топографии участвующих секций в соответствующих экземплярах класса
* Изменение топографии участвующих секций в базе данных

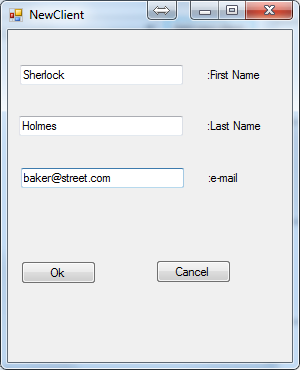
В процессе происходят манипуляции с булевыми значениями локаций с помощью функций, расположенных в файлах SectionsManager.cs и Goodsmanager.cs

Для добавления новых товаров нужно нажав кнопку +Good выбрать существующего клиента и заполнить соответствующие поля. При правильном заполнении отображается кнопа OK. Подсчёт стоимости выполняется после каждого изменения данных.

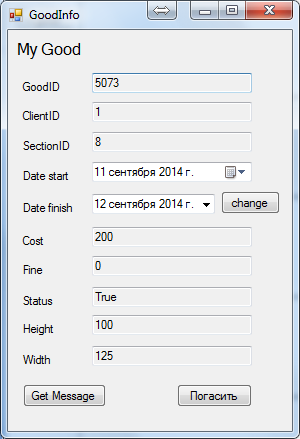


В форме используется comboBox с привязкой DataSource. Для этого создан класс ClientForCB.cs, в котором определены поле типа string для отображения имени и фамилии и поле типа Client для хранения объекта типа Client. (такая-же механика осуществлена в форме информации о клиентах и товарах для клинтов и товаров(GoodForClientInf.cs)).

При необходимости можно добавить нового клиента нажав кнопку Add new client. При правильном заполнении отображается кнопа OK. Добавление нового пользователя возможно только при добавлении товара.



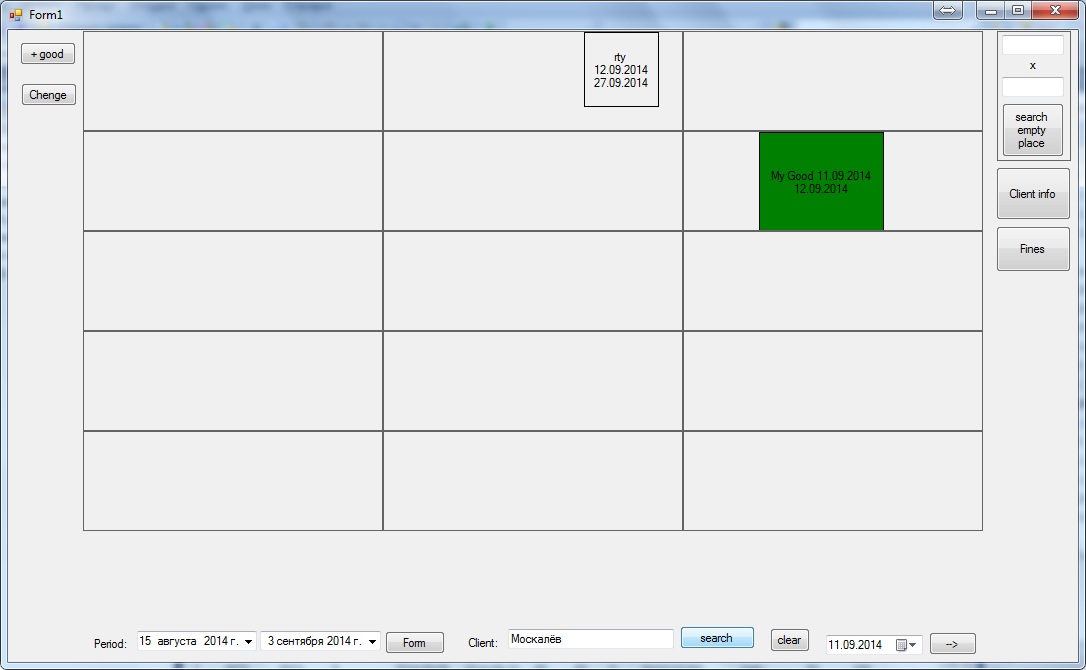
При нажатии на графическое отображение товара, выводится информация следующего вида. Здесь можно изменить конечную дату хранения, но только если текущая дата меньше установленной ранее. При этом стоимость хранения будет пересчитана и информация об этом занесётся в БД и в соответствующий экземпляр класса Good. В этой же форме есть возможность погасить хранение. Если товар забран раньше указанного ранее срока стоимость будет пересчитана, информация занесена в соответствующий экземпляр класса и в БД.



При нажатии кнопки погасить осуществляются следующие шаги:

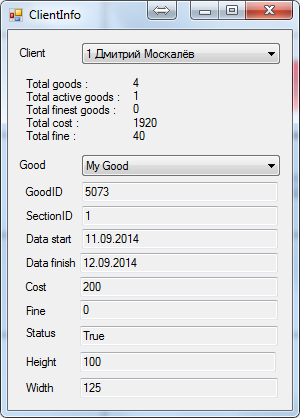
* Товар отмечается как не активный и информация об этом заносится в соответствующий экземпляр класса.
* Корректируется информация о товаре в БД.
* Корректируется информация о топографии секции, в которой располагался товар с помощью методов для работы с булевыми значениями, расположенными в Section.cs и SectionsManager.cs.
* Корректируется информация о секции в БД.
* Удаляется запись в словаре Dictionary<Button, Good> dictButGood
* Удаляется кнопка товара.

Для удобства поиска товаров определённого клиента осуществлена возможность графического выделения кнопок товаров зелёным цветом. На иллюстрации ниже приведен результат поиска

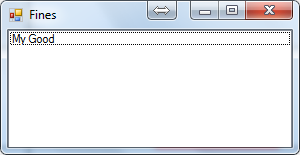


После нажатия кнопки Search происходит сканирование всех товаров, хранящихся в List<Good>, все кнопки, соответствующие активным товары с помощью Dictionary<Button, Good> dictButGood окрашиваются в зелёный цвет.

Окно информации о клиентах представляет собой не редактируемые поля с информацией по каждому товару, выбираемому из выпадающего списка и суммарную информацию по клиенту.

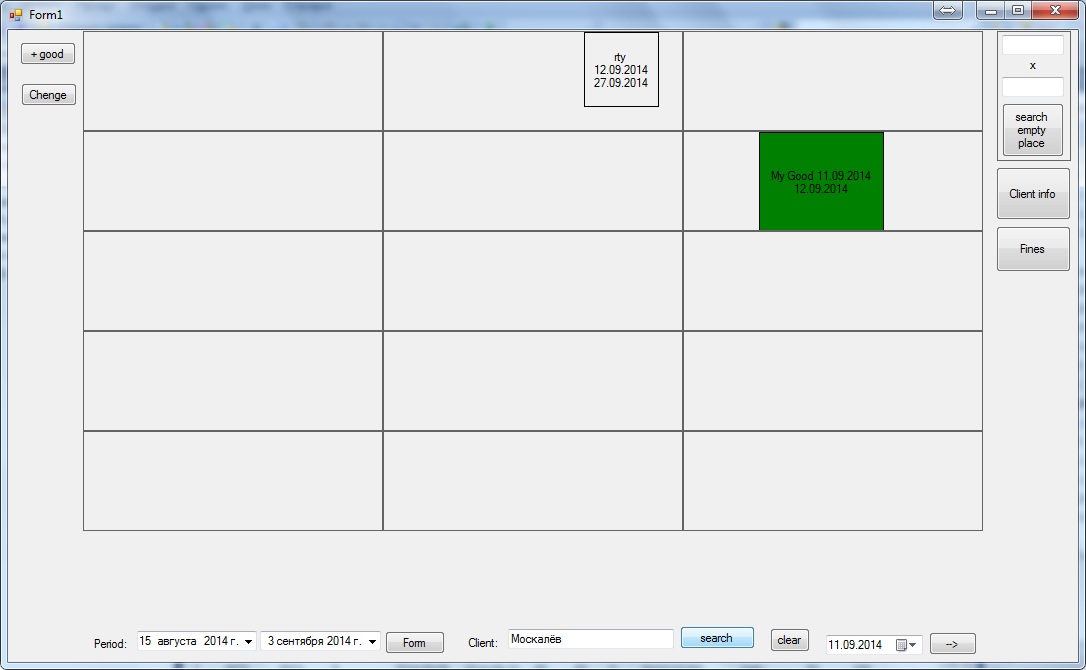


Нажатие кнопки Fines отображает товара с просроченными сроками. Двойной клик по товару отобразит форму информации о товаре.



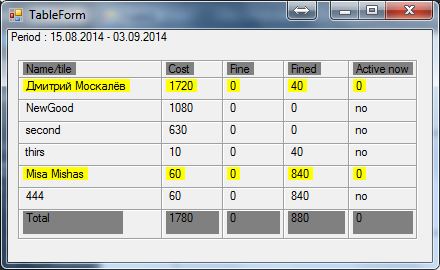
В данной форме также реализована привязка DataSource в ListBox-е.

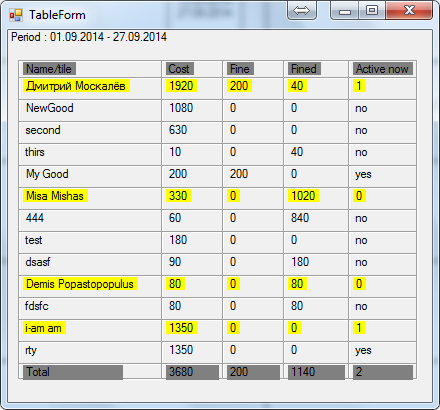
Нажатие кнопки со стрелкой эмулирует переход на следующий день. При этом кнопки товаров окрашиваются в жёлтый цвет, когда конечный срок хранения на 3 дня больше от текущей даты и красный цвет при просроченном сроке хранения товара.



В данном случае нам также помогает созданный на начальном этапе словарь Dictionary<Button, Good> dictButGood

Для отчёта о финансовых поступлениях за указанный период необходимо нажать кнопку Form. Поля таблицы будут содержать суммарную информацию по складу, информацию для каждого клиента и товара





База данных склада состоит из трёх таблиц:

1. Хранимые товары
   1. Идентификатор товара
   2. Идентификатор клиента
   3. Идентификатор секции
   4. Дата приёма товара
   5. Дата окончания хранения
   6. Стоимость
   7. Пеня
   8. Статус (активный/не активный)
   9. Положение в секции на паллете
   10. Название
   11. Длина
   12. Ширина
2. Клиенты
   1. Идентификатор клиента
   2. Имя
   3. Фамилия
   4. e-mail
3. Секции для хранения товаров
   1. Идентификатор секции
   2. Заполненность секции

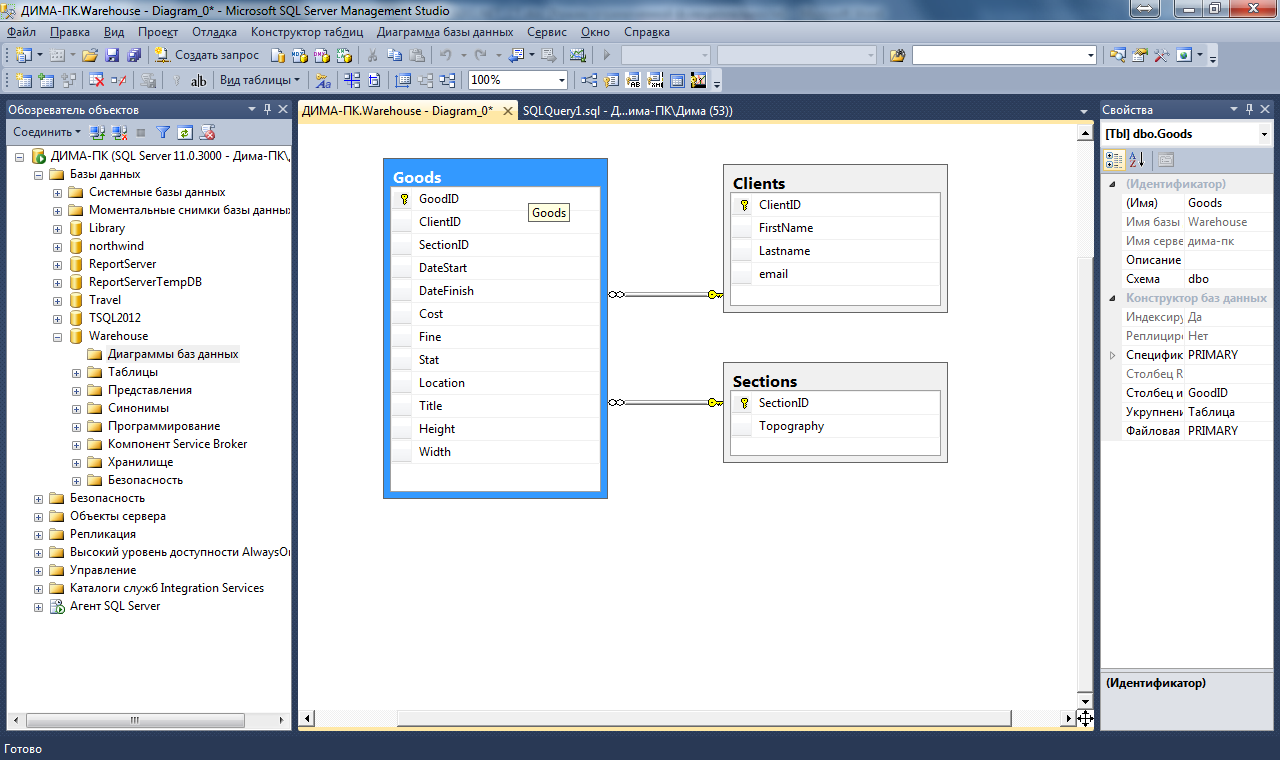


Таблица товаров связана с таблицами клиентов и секций внешними ключами:

ALTER TABLE Goods ADD

CONSTRAINT FK\_Goods\_Clients FOREIGN KEY(ClientID)

REFERENCES Clients(ClientID);

ALTER TABLE Goods ADD

CONSTRAINT FK\_Goods\_Sections FOREIGN KEY(SectionID)

REFERENCES Sections(SectionID);

При добавлении товара происходит проверка на корректность e-mail адреса:

ALTER TABLE Clients ADD CONSTRAINT CK\_Clients\_email

CHECK (email like '%\_@\_\_%.\_\_%');

База заполняется 15 экземплярами секций:

insert into Sections (Topography) values (0);

…