# Введение

Учет и ведение документации по товарообороту является сложной и важной задачей любой торговой организации. Чем больше структура компании, чем большее она имеет в наличии торговых точек – тем сложнее данный вид документооборота. Данная система должна решать комплекс задач: учет проданных и имеющихся в наличии товаров, оформление заказов на требуемые товары поставщикам, быстрая возможность поиска конкретного товара по торговым точкам и складам, ведение документооборота по заказу/поставке/продаже товаров, контроль исполнения заказов и поставок. Хорошо организованная и не имеющая ошибок система обеспечивает организации возможность предоставить клиенту полные сведения о товаре и его местонахождении в быстрый промежуток времени, не допустив ошибок и сохранив лицо компании. Упростить процесс отслеживания товарооборота со стороны старших менеджеров и уменьшить нагрузку на персонал, и создать тем самым нормальную здоровую рабочую атмосферу. Чем больше предоставляет подобная система возможностей – тем проще процесс обслуживания клиентов и процесс заказа новых требующихся товаров.

Существенно облегчается работа менеджеров по продажам. Имея автоматизированную систему, разработанную в рамках курсового проекта, отпадает необходимость постоянной связи с кладовщиками рядовых менеджеров, исчезает возможность ошибок омрачающих репутацию компании, сохраняет время клиентов и их хорошее отношение. Имея возможность быстро предоставить информацию о товаре и оперативно оформить заказ, менеджер по продажам тратит гораздо меньше времени на данную работу позволяя уделить больше времени работе с клиентом. Также система сильно облегчает работу старших менеджеров по планированию заказов, выявлении наиболее популярных и покупаемых товаров, контролю своевременного исполнения заказов и поставок.

Назначение программы, разработанной в рамках курсового проекта, облегчить и улучшить труд работников торговли, позволив им больше времени уделять другим рабочим нуждам, тем самым улучшив атмосферу в рабочем коллективе и улучшить качество обслуживание клиентов, получив тем самым их хорошее отношение. Решение всех этих задач приближает компанию к получению наибольшей прибыли и стабильной работе.

# Построение инфологической концептуальной модели

## Анализ предметной области и выявление необходимого набора сущностей

Предполагается создать базу данных для компьютерной фирмы, которая имеет ряд торговых точек. По условию, каждая торговая точка имеет свой торговый зал с выставленным на продажу оборудованием, а оборудование поступает с центрального склада фирмы. Предположим следующие ситуации: торговая точка может иметь несколько торговых залов и собственные складские помещения; центральный склад может состоять из множества отдельных складских помещений, которые не обязательно располагаются по единому адресу. Исходя из данных ситуаций, можно утверждать, что фирма владеет рядом помещений, которые могут дифференцироваться на помещения центрального склада (базы), торговые залы и склады торговых точек. Притом, помещения могут быть обособленными (складскими либо незадействованными) или закрепленными за определенными торговыми точками, то есть являться помещениями торговой точки.

По заданию фирма осуществляет торговлю различным компьютерным оборудованием: компьютерной техникой, устройствами периферии и сопутствующими товарами. Так как детальное рассмотрение видов и моделей различного оборудования выходит за рамки данного курсового проекта, выделим такое понятие как «категория оборудования**»** (например: ноутбуки, мониторы, принтеры, мыши и так далее).

Каждая единица оборудования характеризуется организацией-поставщиком, ведь одну и ту же модель оборудования могут поставлять различные организации. Так как в зависимости от, например, надбавки поставщика, отпускная цена нашего товара либо процент прибыли от каждой его единицы будет варьироваться, то следует выделить понятие «товар конкретного поставщика».

Ведение документации требует указание ответственных лиц – работников, к примеру, принявших товар по накладной либо оформивших заказ на некоторые товары. Так как в любой фирме присутствует естественный оборот кадров, то есть с некоторой регулярностью в фирму устраиваются новые работники, могут быть переведены на другое место работы либо уволены уже работающие, то можно утверждать, что нам потребуется информация о текущем составе. Поэтому введем понятие «наём», одной из характеристик которого будет являться занимаемая должность, так как мы должны контролировать степень ответственности работника. Так как по условию один работник в разное время может работать на разных торговых точках, а на одной торговой точке может одновременно работать несколько работников, потребуется вести график работы персонала.

Так как цель любого коммерческого предприятия это получение прибыли, то первостепенно требуется ввести понятие «продажа». При обращении покупателя оформляется счет, содержащий перечень продаваемых покупателю товаров. Так как в контексте данной курсовой работы предполагается наличный и безналичный расчет, то мы будем производить учет следующих документов: чеков, напечатанных при помощи КСА (кассовых суммирующих аппаратов) соответствующих торговых точек, если это наличный расчет; платежных поручений, если расчет безналичный.

Так как после определенного количества продаж торговой точке потребуется восполнение товаров, то она может произвести заказ товаров на базе, другой торговой точке или у одной из фирм-поставщиков. Так как заказ потребует времени для его удовлетворения, потребуется хранение списка заказываемых товаров. Если заказ должен быть произведен у одной из фирм поставщиков, то на основании собранных заказов за неделю формируются заявки фирмам-поставщикам с перечнем товаров заявки.

При удовлетворении заказа пересылаемые товары сопровождаются соответствующей накладной. Накладные могут быть различного типа: торговые, торгово-транспортные либо внутренние (накладные, сопровождаемые пересылку товаров между торговыми точками и складами). При получении товара по накладной происходит регистрация всех полученных товаров.

Таким образом, в ходе анализа предметной области был выделен следующий необходимый набор сущностей:

1. Торговая точка
2. Помещение
3. Помещения торговой точки
4. Оборудование
5. Категория оборудования
6. Поставщик
7. Товар поставщика
8. Работник
9. Наём
10. Должность
11. График работы
12. Продажа
13. Продаваемый товар
14. Чек
15. КСА (кассовое суммирующее устройство)
16. Платежное поручение
17. Заказ
18. Заказываемый товар
19. Заявка
20. Товар заявки
21. Накладная
22. Тип накладной
23. Получение товаров по накладной
24. Полученный товар

На этапе анализа предметной области было бы полезным составление функциональной схемы IDEF0 для процесса товарооборота фирмы, который в контексте данной курсовой работы является ключевым. С помощью наглядного графического языка IDEF0, изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков – в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы.

Для построения диаграммы IDEF0 был выбрано программное средство BPwin.

Контекстная диаграмма процесса товарооборота, отражающая взаимосвязь процесса товарооборота с внешней средой, представлена на рисунке 1.1.

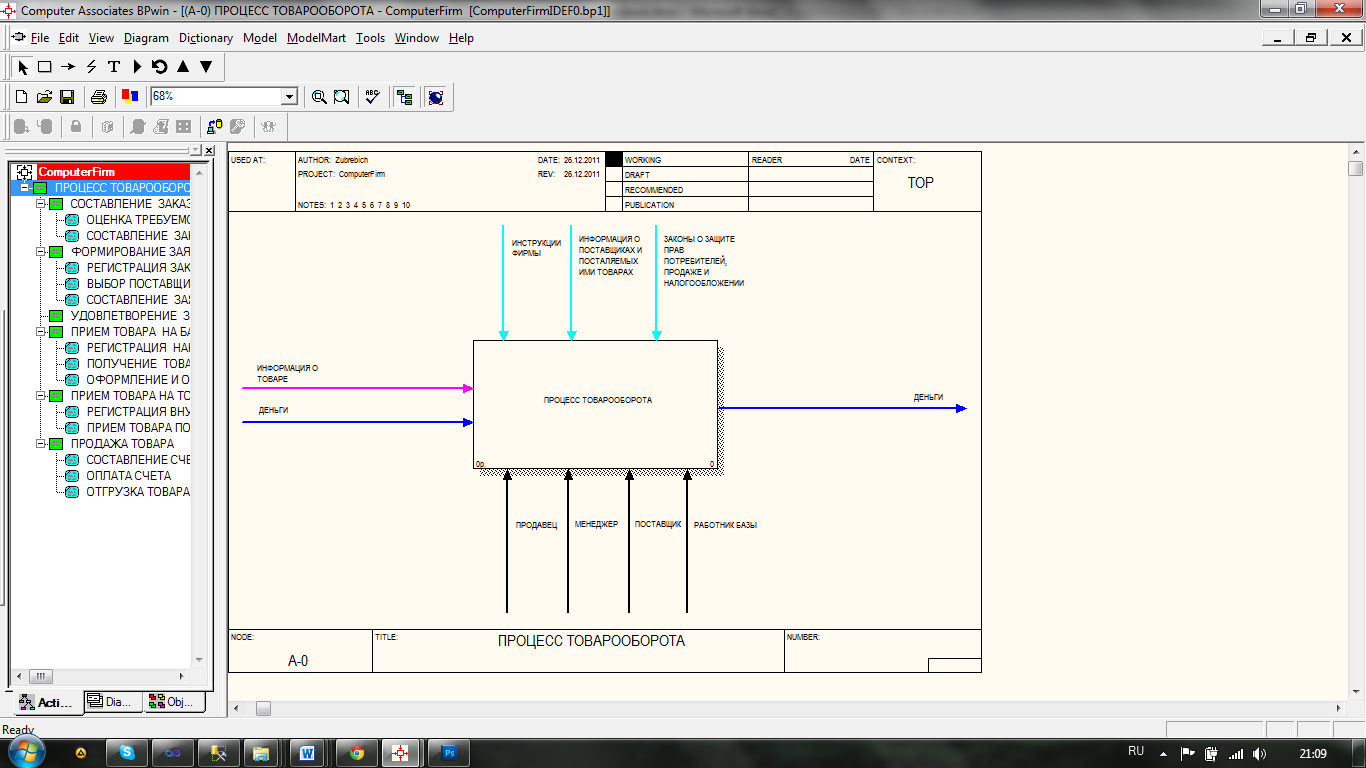
–

Рисунок 1.1 – Контекстная диаграмма процесса товарооборота (IDEF0)

По результатам разработки контекстной диаграммы выполняется процесс декомпозиции модели (результат представлен на рисунке 1.2).

Произведем детализацию IDEF0-диаграммы (рис.1.2) при помощи DFD (Data Flow Diagram) для более наглядного отображения текущих операций документооборота в системах. Главная цель DFD – показать, как каждая работа преобразует свои входные данные в выходные, а также выявить отношения между этими работами.

Графический материал, отражающий результаты детализации процесса товарооборота при помощи DFD-диаграмм, содержится в приложении D.

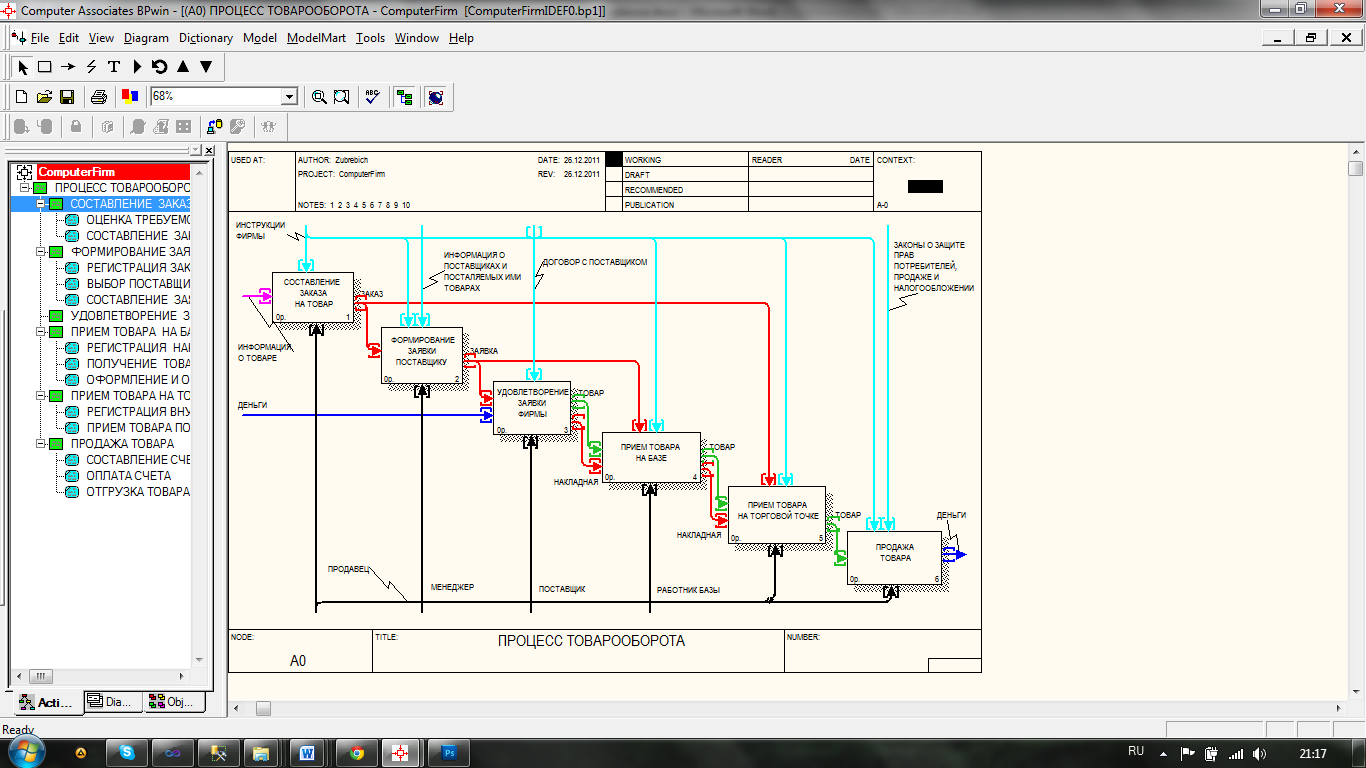


Рисунок 1.2 – Декомпозиция контекстной диаграммы процесса товарооборота (IDEF0)

## Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

Ниже представлены сущности и определенные для них атрибуты, указаны первичные и вторичные ключи (названия вторичных ключей будут соответствовать названию связанной сущности, за исключением базы: под базой будет пониматься помещение с выставленным флагом «база»):

1. Торговая точка:

* номер торговой точки (первичный ключ);
* название;
* безналичный расчет.

1. Помещение:

* номер помещения (первичный ключ);
* торговый зал;
* база;
* адрес здания;
* номер помещения в здании.

1. Помещения торговой точки:

* номер помещения торговой точки (первичный ключ);
* торговая точка (вторичный ключ);
* помещение (вторичный ключ).

1. Оборудование:

* штрих-код (первичный ключ);
* категория оборудования (вторичный ключ);
* гарантийный период;
* срок службы.

1. Категория оборудования:

* номер категории (первичный ключ);
* название категории.

1. Поставщик:

* номер поставщика (первичный ключ);
* название;
* страна;
* адрес;
* контактный телефон;
* электронная почта;
* факс.

1. Товар поставщика:

* номер товара поставщика (первичный ключ);
* поставщик (вторичный ключ);
* оборудование (вторичный ключ);
* цена без наценки;
* наценка поставщика.

1. Работник:

* номер сотрудника (первичный ключ);
* фамилия;
* имя;
* отчество;
* дата рождения;
* серия паспорта;
* номер паспорта.

1. Наём

* номер найма (первичный ключ);
* работник (вторичный ключ);
* должность (вторичный ключ);
* дата найма;
* дата увольнения.

1. Должность:

* номер должности (первичный ключ);
* название.

1. График работы:

* номер пункта графика (первичный ключ);
* наём (вторичный ключ);
* торговая точка (вторичный ключ);
* база (вторичный ключ);
* день недели;
* смена.

1. Продажа:

* номер продажи (первичный ключ);
* торговая точка (вторичный ключ);
* чек (вторичный ключ);
* платежное поручение (вторичный ключ);
* работник (вторичный ключ);
* дата продажи;
* сумма продажи.

1. Продаваемый товар:

* номер продаваемого товара (первичный ключ);
* продажа (вторичный ключ);
* полученный товар (вторичный ключ);
* количество;
* наценка;
* отгружен.

1. Чек:

* идентификатор чека (первичный ключ);
* КСА (вторичный ключ);
* торговая точка (вторичный ключ);
* работник (вторичный ключ);
* номер чека;
* дата чека;
* сумма.

1. КСА (кассовое суммирующее устройство):

* регистрационный номер (первичный ключ);
* торговая точка (вторичный ключ);
* номер фискального регистратора;
* описание модели КСА;
* дата ввода в эксплуатацию;
* дата окончания гарантии;
* лимит кассы.

1. Платежное поручение:

* идентификатор платежного поручения (первичный ключ);
* номер документа;
* дата документа;
* сумма;
* плательщик.

1. Заказ:

* номер заказа (первичный ключ);
* торговая точка - заказчик (вторичный ключ);
* база – заказчик (вторичный ключ);
* торговая точка – исполнитель (вторичный ключ);
* база – исполнитель (вторичный ключ);
* работник (вторичный ключ);
* дата заказа;
* выполнен.

1. Заказываемый товар:

* номер заказываемого товара (первичный ключ);
* заказ (вторичный ключ);
* оборудование (вторичный ключ);
* поставщик (вторичный ключ);
* количество;
* выполненное количество.

1. Заявка:

* номер заявки (первичный ключ);
* поставщик (вторичный ключ);
* работник (вторичный ключ);
* дата заявки;
* выполнена.

1. Товар заявки:

* номер товара заявки (первичный ключ);
* заявка (вторичный ключ);
* заказываемый товар (вторичный ключ).

1. Накладная:

* номер накладной (первичный ключ);
* тип накладной (вторичный ключ);
* база - отправитель (вторичный ключ);
* торговая точка - отправитель (вторичный ключ);
* номер документа.

1. Тип накладной:

* номер типа накладной (первичный ключ);
* тип накладной.

1. Получение товаров по накладной:

* номер поступления (первичный ключ);
* база (вторичный ключ);
* торговая точка (вторичный ключ);
* работник (вторичный ключ);
* дата получения.

1. Полученный товар:

* номер полученного товара (первичный ключ);
* поступление по накладной (вторичный ключ);
* товар поставщика (вторичный ключ);
* количество;
* осталось.

## Определение связей между объектами

Если торговая точка и помещение могут существовать как обособленные объекты, то помещение торговой точки нет, одна торговая точка может иметь несколько помещений, но одно помещение может принадлежать только одной точке.

Так как торговые точки осуществляют продажу конкретных товаров (компьютерной техники, устройств периферии и сопутствующих товаров), то предполагается, что товар обязательно характеризуется какой-либо категорией оборудования. Притом одна категория может характеризовать множество оборудования. Один поставщик может поставлять множество оборудования, а один товар может поставляться множеством поставщиков.

Процесс найма представляет собой прием определенного работника на определенную должность, притом один работник может быть нанят на разные должности, а на одну должность может быть нанято несколько работников.

График работы представляет собой совокупность сочетаний времени работы, места работы и данных нанятого работника. Один работник может работать в разное время на разных точках, а на одной точке может работать одновременно множество сотрудников.

Продавец может оформить множество продаж, но одна продажа оформляется только одним продавцом. Притом каждая продажа оформляется между двумя сторонами - покупателем (который в случае наличного расчета не указывается) и торговой точкой. Одна продажа оформляется только на 1 торговую точку, но торговая точка может осуществлять множество продаж. Притом в зависимости от типа оплаты (наличного или безналичного), указывается соответствующий документ – чек или платежное поручение соответственно. Один документ соответствует одной продаже.

Продажа может содержать множество продаваемых товаров, но продаваемый товар соответствует единственной продаже. Полученный фирмой товар может выступать как множество продаваемых товаров, но один продаваемый товар соответствует только одному полученному.

Чек может быть выбит только на одном кассовом аппарате торговой точки одним работником, но работник на кассовом аппарате может выбить множество чеков.

Один заказ может быть оформлен одним продавцом на торговую точку, но один продавец может оформить множество заказов на несколько торговых точек. Заказ может быть оформлен на торговую точку, базу или быть оставлен для составления заявки поставщику. Заказ может содержать множество заказываемых товаров, но один заказываемый товар может содержаться в одном лишь заказе. Притом оборудование может выступать как множество заказываемых товаров, а заказываемый товар может соответствовать только одному оборудованию.

Заявка может быть оформлена только одним продавцом и отправлена только одному поставщику, но поставщик может получать множество заявок, а продавец оформлять несколько заявок. Одна заявка может содержать перечень товаров, но один товар заявки может быть включен только в единственную заявку.

Накладная может соответствовать только одному типу накладной, но один тип накладной может быть присущ множеству накладных. Накладная может быть оформлена единственной торговой точкой, базой или поставщиком. Каждая торговая точка, база либо поставщик может оформлять множество накладных. Получение по накладной может осуществляться на единственной торговой точке или базе, а полученный товар относиться к единственной накладной (то есть получению по накладной).

1. Торговая точка – Помещение торговой точки: (1:M).
2. Помещение – Помещение торговой точки: (1:1).
3. Категория оборудования – Оборудование: (1:М).
4. Оборудование – Товар поставщика: (1:М).
5. Поставщик – Товар поставщика: (1:М).
6. Работник – Наём: (1:М).
7. Должность – Наём: (1:М).
8. Наём – График работы: (1:М).
9. Торговая точка – График работы: (0..1:М).
10. База – График работы: (0..1:М).
11. Торговая точка – Продажа: (1:М).
12. Чек – Продажа: (0..1:1).
13. Платежное поручение – Продажа: (0..1:1).
14. Продавец – Продажа: (1:М).
15. Продажа – Продаваемый товар: (1:М).
16. Полученный товар – Продаваемый товар: (1:М).
17. КСА – Чек: (1:М).
18. КСА – Торговая точка (1:M)
19. Торговая точка – Чек: (1:М).
20. Продавец – Чек: (1:М).
21. Торговая точка (заказчик) – Заказ: (0..1:М).
22. База (заказчик) – Заказ: (0..1:М).
23. Торговая точка (исполнитель) – Заказ: (0..1:М).
24. База (исполнитель) – Заказ: (0..1:М).
25. Продавец – Заказ: (1:М).
26. Заказ – Заказанный товар: (1:М).
27. Оборудование – Заказанный товар: (1:М).
28. Поставщик – Заказанный товар: (0..1:М)
29. Поставщик – Заявка: (1:М).
30. Работник – Заявка: (1:М).
31. Заявка – Товар заявки: (1:М).
32. Заказываемый товар – Товар заявки: (1:М).
33. Тип накладной – Накладная: (1:М)
34. База-отправитель – Накладная: (0..1:М).
35. Торговая точка-отправитель – Накладная: (0..1:М).
36. Накладная – Получение по накладной: (1:1).
37. Работник – Получение по накладной: (1:М).
38. База-получатель – Получение по накладной: (0..1:М).
39. Торговая точка-получатель – Получение по накладной: (0..1:М).
40. Получение по накладной – Полученный товар: (1:М).
41. Товар поставщика – Полученный товар: (1:М).

## Описание полученной модели на языке инфологического проектирования

Проектирование базы данных (БД) включает описание отношений между данными, которые накапливаются и обрабатываются информационной системой. Это описание выражается в виде инфологической модели (ИЛМ) предметной области. ИЛМ содержит, в частности, описание объектов и связей между ними, которые могут задаваться диаграммой “сущность-связь” (ER-диаграммой).

ER-диаграммы стали основой для более совершенных методологий создания ИЛМ, учитывающих такие требования, как простота изучения и возможность автоматизации. В частности, такой методологией является методология IDEF1X, реализованная в ряде специальных программ автоматизации проектирования БД, например, в программе ERwin.

Таким образом, на основании результатов предыдущих пунктов опишем инфологическую концептуальную модель в соответствии с методологией IDEF1X, учитывая тип связи (идентифицирующая либо неидентифицирующая) и её кардинарность. Результаты построения данной модели представлены в приложении А.

**2. ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ**

## Построение набора необходимых отношений базы данных

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте мы создали инфологическую концептуальную модель базы данных «Компьютерная фирма», построенной с помощью IDEF1X. На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения для базы данных компьютерной фирмы.

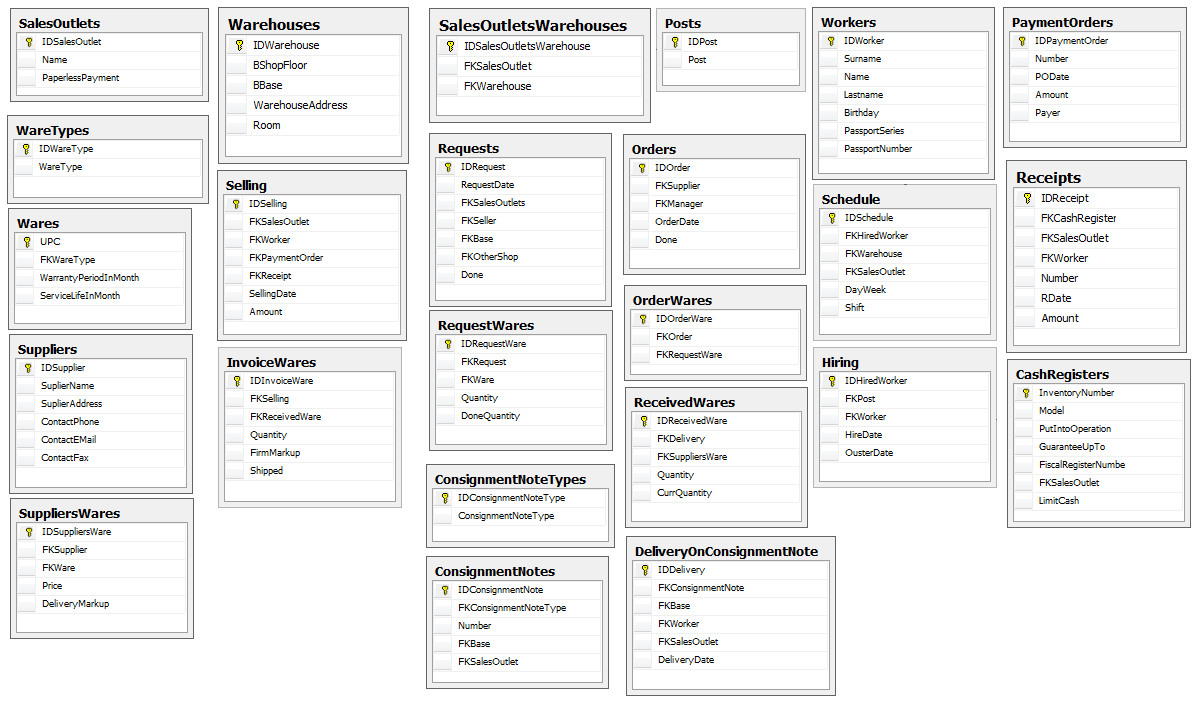


Рисунок 2.1 – Набор необходимых отношений базы данных

## Задание первичных и внешних ключей определённых отношений

Поле первичного ключа служит уникальным определением записи. Оно также служит для связи таблиц. В связанных таблицах первичный ключ родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ дочерней таблицы отсылает к сведениям родительской таблицы.

Первичные ключи будут иметь постфикс ID, вторичные – FK. Также в подавляющем числе случаев названия вторичных ключей будут соответствовать названию связанных таблиц (без множественного числа), при несоответствии укажем название связанной таблицы во избежание путаницы. Также следует упомянуть, что в контексте данной работы первичные ключи будут являться единственным полем в таблице.

Таблица 2.1 – Первичные и внешние ключи отношений

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название таблицы | Первичный ключ | Внешние ключи |
|  | SalesOutlets (Торговая точка) | IDSalesOutlet | - |
|  | Warehouses (Помещение) | IDWarehouse | - |
|  | SalesOutletsWarehouses (Помещения торговой точки) | IDSalesOutletsWarehouse | FKWarehouse  FKSalesOutlet |
|  | CashRegisters (КСА) | InventoryNumber (Регистрационный номер) | FKSalesOutlet |
|  | Posts (Должности) | IDPost | - |
|  | Workers (Работники) | IDWorker | - |
|  | Hiring (Наём) | IDHiredWorker | FKPost  FKWorker |
|  | Schedule (График работы) | IDSchedule | FKHiring  FKWarehouse  FKSalesOutlet |
|  | WareTypes (Категория оборудования) | IDWareType | - |
|  | Wares (Оборудование) | UPC | FKWareType |
|  | Suppliers (Поставщики) | IDSupplier | - |
|  | SuppliersWares (Товары поставщиков) | IDSuppliersWare | FKSupplier  FKWare |
|  | Requests (Заказы) | IDRequest | FKSeller (табл.Workers)  FKBase (табл.Warehouses)  FKSalesOutlet  FKOtherShop (табл. SalesOutlets) |
|  | RequestWares (Товары заказа) | IDRequestWare | FKRequest  FKWare |
|  | Orders (Заявки) | IDOrder | FKManager (табл.Workers)  FKSupplier |
|  | OrderWares (Товары заявки) | IDOrderWare | FKOrder  FKRequestWare |
|  | ConsignmentNoteTypes (Типы накладных) | IDConsignmentNoteType | - |
|  | ConsignmentNotes (Накладные) | IDConsignmentNote | FKConsignmentNoteType |
|  | DeliveryOnConsignmentNote (Поступления по накладной) | IDDelivery | FKConsignmentNote  FKBase (табл.Warehouses)  FKSalesOutlet  FKWorker |
|  | ReceivedWares (Полученные товары) | IDReceivedWare | FKDelivery (табл. DeliveryOnConsignmentNote)  FKSuppliersWare |

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название таблицы | Первичный ключ | Внешние ключи |
|  | PaymentOrders (Платежные поручения) | IDPaymentOrder | - |
|  | Receipts (Чеки) | IDReceipt | FKSalesOutlet  FKWorker  FKCashRegister |
|  | Selling (Продажа) | IDSelling | FKSalesOutlet  FKWorker  FKReceipt  FKPaymentOrder |
|  | InvoiceWares (Продаваемые товары) | IDInvoiceWare | FKSelling  FKReceivedWare |

## Приведение отношений БД к третьей нормальной форме

Нормальная форма — свойство [отношения](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)) в [реляционной модели данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией.

Так как все атрибуты наших отношений атомарны, а каждое отношение имеет первичный ключ, то это означает, что отношения базы находятся в первой нормальной форме (1НФ).

Так как зависимости неключевых атрибутов от части составного ключа отсутствуют (все ключи в вышеописанных отношениях несоставные), а отношения базы находятся в 1НФ, то можно утверждать, что отношения базы удовлетворяют требованиям второй нормальной формы (2НФ).

Так как в отношениях базы отсутствуют зависимости неключевых атрибутов от других неключевых атрибутов, а присутствие 2НФ указано выше, можно сказать, что отношения базы находятся в третьей нормальной форме (3НФ).

Таким образом, отношения БД находятся в 3НФ.

**2.4. Определение ограничений целостности для внешних ключей отношений и для отношений в целом**

Целостность (от англ. integrity – нетронутость, неприкосновенность, сохранность, целостность) – понимается как правильность данных в любой момент времени. Но эта цель может быть достигнута лишь в определенных пределах: СУБД не может контролировать правильность каждого отдельного значения, вводимого в базу данных. Для этого существует ряд средств, помогающих разработчику минимизировать возможность нарушения целостности данных базы: триггеры, проверки («check»), уникальность («unique») и др.

Таблица 2.2 – Ограничения и триггеры таблиц

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название таблицы | Ограничение с указанием типа | Описание |
|  | SalesOutlets | CHECK CH\_SalesOutletsName | Название торговой точки должно содержать хотя бы 1 символ. |
| TRIGGER DTrigger | При удалении торговой точки освобождает все помещения, ей принадлежащие. |
|  | CashRegisters | UNIQUE  UQ\_FiscalRegisterNumber | Номер фискального регистратора КСА должен быть уникальным. |
| CHECK  CH\_CashRegisterModel | Описание модели не должно быть пустой строкой. |
| CHECK  CH\_CashRegisterPutIntoOperation | Поле «Введен в эксплуатацию» должен содержать дату не позже сегодняшнего дня. |
| CHECK  CH\_FiscalRegisterNumber | Номер фискального регистратора должен быть больше ноля. |
| CHECK  CH\_LimitCash | Лимит кассы должен быть больше ноля. |
|  | Posts | UNIQUE  UQ\_PostName | Название должности должно быть уникальным. |
| CHECK  CH\_PostName | Название должности должно содержать хотя бы один символ. |
|  | Workers | CHECK  CH\_WorkerSurname | Фамилия должна содержать хотя бы 1 символ. |
| CHECK  CH\_WorkerName | Имя должно содержать хотя бы 1 символ. |
| CHECK  CH\_WorkerBirthday | Возраст работника должен быть не меньше 14 лет. |
| UNIQUE  UQ\_WorkerPassport | Номер и серия паспорта должны быть уникальны. |
| CHECK  CH\_WorkerPassportNumber | Номер паспорта должен быть больше ноля. |
|  | Hiring | CHECK  CH\_HireDate | Дата найма должна быть не больше сегодняшней. |
| CHECK  CH\_OusterDate | Дата увольнения должна отсутствовать или быть позже даты найма. |
|  | Warehouses | CHECK  CH\_ShopFloorOrBaseOrShopWarehouse | Помещение не может быть одновременно и базой и торговым залом. |

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название таблицы | Вид ограничения | Описание |
| 6 | Warehouses | CHECK  CH\_Room | Номер помещения должен отсутствовать или быть больше ноля. |
| UNIQUE  UQ\_Address | Адрес помещения должен быть уникальным. |
|  | SalesOutletsWarehouses | UNIQUE  UQ\_FKWarehouse | Одно помещение может принадлежать только одной точке. |
| TRIGGER  SalesOutletsWarehousesIUTrigger | У одной торговой точки все торговые залы должны быть по одному адресу здания. |
|  | Schedule | CHECK  CH\_Schedule\_WarehouseOrSalesOutlet | Работник может работать в одно и то же время или на базе, или на торговой точке. |
| CHECK  CH\_DayWeek | День недели должен быть в диапазоне от 1 до 7. |
| UNIQUE  UQ\_Day | Сочетание работник, день недели и смена должны быть уникальны. |
| TRIGGER  ScheduleIUTrigger | Один человек может работать в одно время только в одном месте. |
| 9 | ConsignmentNoteTypes | UNIQUE  UQ\_ConsignmentNoteType | Название типа накладной должно быть уникально. |
| 10 | ConsignmentNotes | CHECK  CH\_ConsignmentNoteNumber | Номер накладной должен быть больше ноля. |
| CHECK  UQ\_ConsignmentNote | Номер накладной и её тип должны быть уникальны. |
| CHECK  CH\_ConsignmentNote\_BaseOrShopOrSupplier | Накладная не может быть одновременно отправлена и торговой точкой, и базой. |
| 11 | WareTypes | UNIQUE  UQ\_WareType | Категория товара должна иметь уникальное название. |
| 12 | Wares | CHECK  CH\_WarrantyPeriodInMonth | Гарантийный период не должен быть отрицательным. |
| CHECK  CH\_ServiceLifeInMonth | Срок службы должен быть не указан или должен быть положительным. |
| CHECK  CH\_WarrantyAndServicePeriods | Если срок службы указан, он должен превосходить гарантийный. |
| 13 | Suppliers | UNIQUE  UQ\_SuplierName | Название поставщика должно быть уникальным значением. |

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название таблицы | Вид ограничения | Описание |
| 13 | Suppliers | UNIQUE  UQ\_SuplierAddress | Адрес поставщика должен быть уникален. |
| UNIQUE  UQ\_SuplierPhone | Телефон поставщика должен быть уникален |
| UNIQUE  UQ\_SuplierEMail | Электронный адрес поставщика должен быть уникален |
| UNIQUE  UQ\_SuplierFax | Факс поставщика должен быть уникален |
| CHECK  CH\_ContactEMail | Электронный адрес поставщика должен соответствовать шаблону:  пользователь@домен |
| 14 | Orders | CHECK  CH\_OrderDate | Дата заказа не должна быть позже сегодняшней. |
| 15 | DeliveryOnConsignmentNote | UNIQUE  UQ\_FKConsignmentNote | Накладная, по которой оформляется поступление, должна быть уникальна. |
| CHECK  CH\_Delivery\_BaseOrShopOrSupplier | Накладная не может быть одновременно принята и торговой точкой, и базой. |
| 16 | SuppliersWares | CHECK  CH\_DeliveryMarkup | Надбавка поставщика должна быть положительным числом. |
| CHECK  CH\_ReceivedWarePrice | Цена должна быть положительным числом. |
| UNIQUE  UQ\_FKSupplier\_FKWare | Сочетание конкретного оборудования и поставщика должно быть уникально. |
| 17 | ReceivedWares | CHECK  CH\_ReceivedWareQuantity | Поступившее количество товара должно быть больше ноля. |
| TRIGGER  ReceivedWares\_ITrigger | Обновляет данные заказов торговых точек и баз. Предотвращает ввоз товара сверх заказанного, обновляет текущее количество товара отправителя товаров. |
| 18 | PaymentOrders | CHECK  CH\_PaymentOrderIssureDate | Платежное поручение должно быть выписано не позже, чем сегодня. |
| CHECK  CH\_PaymentOrderNumber | Номер платежного поручения должен быть больше ноля. |
| CHECK  CH\_PaymentOrderAmount | Сумма платежного поручения должна быть больше ноля. |
| CHECK  CH\_PaymentOrderPayer | Имя плательщика должно быть больше ноля. |
| № п/п | Название таблицы | Вид ограничения | Описание |
| 19 | Receipts | CHECK  CH\_ReceiptAmount | Сумма чека должна быть больше ноля. |
| UNIQUE  UQ\_FKCashRegisterAndNumber | Сочетание номера фискального регистратора и выбитого чека должны быть уникальны. |
| CHECK  CH\_ReceiptNumber | Номер чека должен быть больше ноля. |
| CHECK  CH\_ReceiptDate | Дата чека должна быть не позже сегодняшней. |
| 20 | Selling | CHECK  CH\_SellingAmount | Сумма продажи должна быть больше ноля. |
| CHECK  CH\_ReceiptOrPaymentOrder | Должно присутствовать или платежное поручение, или чек. |
| UNIQUE  UQ\_ReceiptOrPaymentOrder | Сочетание платежное поручение и чек должно быть уникальным. |
| TRIGGER  SellingIUTrigger | Проверяет, чтобы сумма документа оплаты совпадала с суммой продажи, а дата платежного документа не была большей, чем дата продажи. |
| 21 | InvoiceWares | CHECK  CH\_InvoiceWareQuantity | Количество проданного товара должно быть больше ноля. |
| CHECK  CH\_InvoiceWareFirmMurkup | Надбавка фирмы на продаваемый товар должна быть больше ноля. |
| TRIGGER  InvoiceWareIUTrigger | Надбавка поставщика и фирмы в сумме не должны превышать 30% |
| 22 | Requests | CHECK  CH\_RequestDate | Дата заказа не должна быть позже сегодняшней. |
| CHECK  CH\_RequestClients | Заказ может быть произведен или базой, или торговой точкой, и никак иначе. |
| 23 | RequestWares | CHECK  CH\_RequestWareQuantity | Количество заказываемого товара должно быть больше ноля. |
| TRIGGER  ReceivedWares\_ITrigger | Закрывает заказы, когда они были удовлетворены полностью. |
| CHECK  CK\_RequestWares | Количество привезенного товара не должно превышать количество запрашиваемого. |
| 24 | OrderWares | OrderWares\_UTrigger | Закрывает заявки, когда они были удовлетворены. |

Для примера, приведём полный текст триггера таблицы RequestWares «ReceivedWares\_ITrigger»:

ALTER TRIGGER [dbo].[RequestWares\_UTrigger]

ON [dbo].[RequestWares]

AFTER Update

AS

BEGIN

-- Получаем требуемые обновленные данные

DECLARE @id INT

SET @id = (SELECT inserted.IDRequestWare FROM inserted)

DECLARE @request INT

SET @request = (SELECT inserted.FKRequest FROM inserted)

DECLARE @quantity INT

SET @quantity = (SELECT inserted.Quantity - inserted.DoneQuantity

FROM inserted)

-- Проверяем на корректность разницу между запрашиваемым и привезенным

-- количеством товара

IF (@quantity < 0)

BEGIN

ROLLBACK TRAN

PRINT ‘Количество привезенного товара не может превышать заказанное’

END

-- Если обновленный пункт заказа полностью выполнен – проверим, выполнен ли

-- весь заказ целиком. В положительном случае – помечаем его как выполненный.

IF (@quantity = 0)

BEGIN

IF ((SELECT SUM(Quantity-DoneQuantity)

FROM RequestWares

WHERE FKRequest = @request) = 0)

UPDATE Requests SET Done = 1 WHERE IDRequest = @request

END

END

# Создание спроектированной базы данных

Для реализации спроектированной базы данных была выбрана система управления базами данных MS SQL Server 2008. Это обусловлено тем, что, во-первых, данная СУБД получилась широкое распространение, а во-вторых, имеются свободно распространяемые сборки, а в-третьих, Microsoft и другие компании производят большое число программных средств разработки, позволяющих разрабатывать бизнес-приложения с использованием баз данных Microsoft SQL Server.

Произведем описание структуру каждой из таблиц с описанием типа полей.

Таблица SalesOutlets содержит основные сведения о торговых точках. Её структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы SalesOutlets

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDSalesOutlet | smallint | - | Первичный ключ. |
| Name | varchar(100) | - | Название точки. |
| PaperlessPayment | bit | - | Возможность безналичного расчета при значении 1. |

Таблица Warehouses содержит информацию обо всех принадлежащих фирме помещениях. Её структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы Warehouses

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDWarehouse | int | - | Первичный ключ. |
| BShopFloor | bit | - | Если помещение является торговым залом, значение данного флага равно 1. |
| BBase | bit | - | Если помещение является базой, значение данного флага равно 1. |
| WarehouseAddress | varchar(100) | - | Адрес здания. |
| Room | smallint | - | Номер комнаты |

Таблица SalesOutletsWarehouses определяет пренадлежность помещения торговой точке. Её структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы SalesOutletsWarehouses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя атрибута | | Тип | NULL | Описание |
| IDSalesOutletsWarehouse | | int | - | Первичный ключ. |
| FKSalesOutlet | | smallint | - | Вторичный ключ. Связанная таблица - SalesOutlets |
| Имя атрибута | Тип | | NULL | Описание |
| FKWarehouse | int | | - | Вторичный ключ. Связанная таблица - Warehouses |

Таблица CashRegisters содержит информацию обо всех кассовых аппаратах, принадлежащих фирме. Её структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы CashRegisters

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| InventoryNumber | numeric(10,0) | - | Первичный ключ. |
| Model | varchar(150) | - | Описание модели КСА. |
| PutIntoOperation | date | - | Дата ввода в эксплуатацию. |
| GuaranteeUpTo | date | - | Дата окончания гарантии. |
| FiscalRegisterNumber | bigint | - | Номер фискального регистратора, альтернативный ключ. |
| FKSalesOutlet | smallint | allowed | Вторичный ключ. Связанная таблица – SalesOutlets. Торговая точка-владелец. |
| LimitCash | money | - | Лимит кассы. |

Таблица Posts содержит все присутствующие в фирме должности. Её структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы Posts

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDPost | smallint | - | Первичный ключ. |
| Post | varchar(30) | - | Название должности. |

Таблица Workers содержит сведения обо всех работавших и работающих в фирме людях. Её структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы Workers

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDWorker | int | - | Первичный ключ. Номер работника |
| Surname | varchar(25) | - | Фамилия работника. |
| Name | varchar(25) | - | Имя работника. |
| Lastname | varchar(25) | - | Отчество работника. |
| Birthday | date | - | Дата рождения работника. |
| Имя атрибута | Тип | NULL | Описание |
| PassportSeries | char(5) | - | Серия паспорта. |
| PassportNumber | numeric(7,0) | - | Номер паспорта. |

Таблица Hiring содержит сведения обо всех наймах в фирме и позволяет отслеживать текущий состав персонала. Её структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы Hiring

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDHiredWorker | int | - | Первичный ключ. |
| FKPost | smallint | - | Вторичный ключ. Связанная таблица – Posts. |
| FKWorker | int | - | Вторичный ключ. Связанная таблица – Workers. |
| HireDate | date | - | Дата приема на должность. |
| OusterDate | date | allowed | Дата увольнения с должности. |

Таблица Schedule содержит график работы всех работников на различных базах и торговых точках. Её структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDSchedule | int | - | Первичный ключ. |
| FKHiredWorker | int | - | Вторичный ключ. Связанная таблица – Hiring. |
| FKWarehouse | int | allowed | Вторичный ключ. Связанная таблица – Warehouses. |
| FKSalesOutlet | smallint | allowed | Вторичный ключ. Связанная таблица – SalesOutlets. |
| DayWeek | numeric(1, 0) | - | День недели в числовом представлении. |
| Shift | bit | - | Смена работы: 0 – первая смена, 1 – вторая смена. |

Таблица WareTypes содержит перечень всех категорий товаров. Её структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы WareTypes

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDWareType | smallint | - | Первичный ключ. Порядковый номер категории. |
| WareType | varchar(30) | - | Название категории товара. |

Таблица Wares описывает все доступное продаваемое оборудование. Следует отметить, что в контексте данной курсовой работы понятие «оборудование» максимально абстрагировано ввиду акцента на экономической стороне фирмы. Структура таблицы Wares приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы Wares

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| UPC | numeric(13,0) | - | Первичный ключ. Штрих-код товара. Предполагается, что применяется *универсальный код товара* (Universal Product Code) и кодируется 13 цифрами |
| FKWareType | smallint | - | Вторичный ключ. Связанная таблица – WareTypes. |
| WarrantyPeriodInMonth | smallint | - | Гарантийный период в месяцах. |
| ServiceLifeInMonth | smallint | allowed | Срок службы в месяцах. |

Таблица Suppliers содержит список всех поставщиков фирмы. Её структура приведена в таблице 3.11.

Таблица 3.11 – Характеристика атрибутов таблицы Suppliers

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDSupplier | int | - | Первичный ключ. Порядковый номер поставщика. |
| SuplierName | varchar(50) | - | Название поставщика. Уникально. |
| SuplierAddress | varchar(150) | - | Адрес поставщика (полный, включая страну) |
| ContactPhone | varchar(20) | - | Контактный телефон. |
| ContactEMail | varchar(100) | - | Электронная почта поставщика. |
| ContactFax | varchar(20) | allowed | Факс поставщика, если имеется. |

Таблица SuppliersWares содержит перечень всех поставляемых каждым поставщиком товаров. Её структура приведена в таблице 3.12.

Таблица 3.12 – Характеристика атрибутов таблицы SuppliersWares

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDSuppliersWare | int | - | Первичный ключ. Номер товара поставщика. |
| FKSupplier | smallint | - | Вторичный ключ. Связанная таблица – WareTypes. |
| FKWare | numeric(13,0) | - | Вторичный ключ. Связанная таблица – WareTypes. |
| Имя атрибута | Тип | NULL | Описание |
| Price | money | - | Первоначальная цена товара (без наценки поставщика). Цена содержится именно тут, а не в таблице Wares, по причине предположения, что товар может быть собран на разных заводах, для разных локалей либо иметь какие-либо другие причины различий первоначальной цены. |
| DeliveryMarkup | numeric(2,0) | - | Наценка поставщика. |

Таблица ConsignmentNoteTypes содержит список всех доступных типов накладных, включая внутренние. Её структура приведена в таблице 3.13.

Таблица 3.13 – Характеристика атрибутов таблицы ConsignmentNoteTypes

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDConsignmentNoteType | int | - | Первичный ключ. Порядковый номер типа накладной. |
| ConsignmentNoteType | varchar(50) | - | Название типа. |

Таблица ConsignmentNotes содержит информацию обо всех зарегистрированных накладных. Её структура приведена в таблице 3.14.

Таблица 3.14 – Характеристика атрибутов таблицы ConsignmentNotes

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDConsignmentNote | int | - | Первичный ключ. Порядковый номер накладной. |
| FKConsignmentNoteType | int | - | Вторичный ключ. Связанная таблица – ConsignmentNoteTypes. Указывает тип приходящего документа. |
| Number | numeric(9,0) | - | Номер накладной. Максимальное количества взято с сайта проверки накладных РБ. |
| FKBase | int | allowed | Вторичный ключ. Связанная таблица – Warehouses. Представляет собой базу, которая отослала товары по накладной. |
| FKSalesOutlet | smallint | allowed | Вторичный ключ. Связанная таблица – SalesOutlets. Представляет собой торговую точку, которая отослала товары по накладной. |

Таблица DeliveryOnConsignmentNote содержит информацию обо всех поступлениях по накладной. Её структура приведена в таблице 3.15.

Таблица 3.15 – Характеристика атрибутов таблицы DeliveryOnConsignmentNote

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDDelivery | int | - | Первичный ключ. Порядковый номер поступления. |
| FKConsignmentNote | int | - | Вторичный ключ. Связанная таблица – ConsignmentNotes. |
| FKBase | int | allowed | Вторичный ключ. Связанная таблица – Warehouses. Представляет собой базу, на которую производится завоз товаров. |
| FKWorker | int | - | Вторичный ключ. Связанная таблица – Workers. Работник, принявший товары и являющийся ответственным. |
| FKSalesOutlet | smallint | allowed | Вторичный ключ. Связанная таблица – SalesOutlets. Представляет собой торговую точку, на которую производится завоз товаров. |
| DeliveryDate | date | - | Дата поступления товаров по накладной. |

Таблица ReceivedWares содержит перечень всех поступлений товаров по накладной за все время и является основным источником информации о текущем количестве товаров на каждом объекте фирмы. Структура таблицы приведена в таблице 3.16.

Таблица 3.16 – Характеристика атрибутов таблицы ReceivedWares

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDReceivedWare | int | - | Первичный ключ. Порядковый номер полученного товара. |
| FKDelivery | int | - | Вторичный ключ. Связанная таблица – DeliveryOnConsignmentNote. Позволяет получить полную информацию о получении данного товара. |
| FKSuppliersWare | int | - | Вторичный ключ. Связанная таблица – SuppliersWares. Позволяет получить информацию о стоимости товара, категории, наценке и др. |
| Quantity | int | - | Привезенное по накладной количество. |
| CurrQuantity | int | - | Текущее количество. |

Таблица Requests содержит в себе перечни всех заказов торговых точек и баз, как к другим торговым точкам, базам, поставщикам. Её структура приведена в таблице 3.17.

Таблица 3.17 – Характеристика атрибутов таблицы Requests

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDRequest | int | - | Первичный ключ. Порядковый номер заказа. |
| RequestDate | date | - | Дата заказа. |
| FKShopClient | smallint | allowed | Вторичный ключ. Связанная таблица – SalesOutlets. Содержит торговую точку – заказчика. |
| Имя атрибута | Тип | NULL | Описание |
| FKBaseClient | int | allowed | Вторичный ключ. Связанная таблица – Warehouses. Содержит базу – заказчика. |
| FKSeller | int | - | Вторичный ключ. Связанная таблица – Workers. Позволяет получить информацию о лице, совершившем заказ. |
| FKShopExec | smallint | allowed | Вторичный ключ. Связанная таблица – SalesOutlets. Содержит торговую точку – исполнителя. |
| FKBaseExec | int | allowed | Вторичный ключ. Связанная таблица – Warehouses. Содержит базу – исполнителя. |
| Done | bit | - | Флаг выполнения заказа. |

Таблица RequestWares хранит информацию обо всех заказываемых товарах. Её структура приведена в таблице 3.18.

Таблица 3.18 – Характеристика атрибутов таблицы RequestWares

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDRequestWare | int | - | Первичный ключ. Порядковый номер заказываемого товара. |
| FKRequest | int | - | Вторичный ключ. Связанная таблица – Requests. Номер заказа. |
| FKWare | numeric(13,0) | - | Вторичный ключ. Связанная таблица – Wares. Заказываемый товар. |
| Quantity | int | - | Заказываемое количество. |
| DoneQuantity | int | - | Привезенное количество. |
| FKSupplier | smallint | allowed | Вторичный ключ. Связанная таблица – Suppliers. Нужен только для внутренних накладных. |

Таблица Orders содержит информацию обо всех заявках фирмы поставщикам. Её структура приведена в таблице 3.19.

Таблица 3.19 – Характеристика атрибутов таблицы Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDOrder | int | - | Первичный ключ. Номер заявки. |
| FKSupplier | smallint | - | Вторичный ключ. Связанная таблица – Suppliers. Поставщик, которому отсылается заявка. |
| FKManager | int | - | Вторичный ключ. Связанная таблица – Workers. Ответственный. |
| OrderDate | date | - | Дата составления заявки. |
| Done | bit | - | Флаг выполнения заявки. |

Таблица OrderWares содержит информацию обо всех товарах в заявках. Её структура приведена в таблице 3.20.

Таблица 3.20 – Характеристика атрибутов таблицы OrderWares

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDOrderWare | int | - | Первичный ключ. Порядковый номер товара заявки. |
| FKOrder | int | - | Вторичный ключ. Связанная таблица – Orders. Номер заявки. |
| FKRequestWare | int | - | Вторичный ключ. Связанная таблица – RequestWares. Номер товара заказа, на основании которого составлялась заявка. |
| ReceivedQuantity | int | - | Привезенное количество товара по заявке. |

Таблица Receipts содержит сведения о чеках, по которым осуществлялась продажа. Её структура приведена в таблице 3.21.

Таблица 3.21 – Характеристика атрибутов таблицы Receipts

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDReceipt | int | - | Первичный ключ. Порядковый номер зарегистрированного чека. |
| FKCashRegister | numeric(10, 0) | - | Вторичный ключ. Связанная таблица – CashRegisters. КСА, на котором выбит чек. |
| FKSalesOutlet | smallint | - | Вторичный ключ. Связанная таблица – SalesOutlets. Торговая точка, выдавшая чек. |
| FKWorker | int | - | Вторичный ключ. Связанная таблица – Workers. Продавец, выдавший чек. |
| Number | int | - | Номер чека (выштамповываемый КСА) |
| RDate | smalldatetime | - | Дата выдачи чека. |
| Amount | money | - | Сумма чека |

Таблица PaymentOrders содержит информацию обо всех платежных поручениях, по которым осуществлялась продажа товара. Её структура приведена в таблице 3.22.

Таблица 3.22 – Характеристика атрибутов таблицы Gr\_Группы

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDPaymentOrder | int | - | Первичный ключ. Порядковый номер зарегистрированного платежного поручения. |
| Number | int | - | Номер в документе. |
| PODate | date | - | Дата в документе. |
| Amount | money | - | Сумма. |
| Имя атрибута | Тип | NULL | Описание |
| Payer | varchar(50) | - | Плательщик |

Таблица Selling содержит информацию обо всех продажах фирмы. Её структура приведена в таблице 3.23.

Таблица 3.23 – Характеристика атрибутов таблицы Selling

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDSelling | int | - | Первичный ключ. Порядковый номер продажи. |
| FKSalesOutlet | smallint | - | Вторичный ключ. Связанная таблица – SalesOutlets. Торговая точка, осуществившая продажу. |
| FKWorker | int | - | Вторичный ключ. Связанная таблица – Workers. Продавец, оформивший продажу. |
| FKPaymentOrder | int | allowed | Вторичный ключ. Связанная таблица – PaymentOrders. Платежное поручение. |
| FKReceipt | int | allowed | Вторичный ключ. Связанная таблица – Receipts. Чек. |
| SellingDate | date | - | Дата продажи |
| Amount | money | - | Сумма продажи. |

Таблица InvoiceWares содержит информацию обо всех проданных товарах. Её структура приведена в таблице 3.24.

Таблица 3.24 – Характеристика атрибутов таблицы InvoiceWares

|  |  |  |  |
| --- | --- | --- | --- |
| Имя атрибута | Тип | NULL | Описание |
| IDInvoiceWare | int | - | Первичный ключ. Порядковый номер продаваемого товара. |
| FKSelling | int | - | Вторичный ключ. Связанная таблица – Selling. Номер продажи. |
| FKReceivedWare | int | - | Вторичный ключ. Связанная таблица – ReceivedWares. Непосредственно сам продаваемый товар. |
| Quantity | int | - | Количество единиц товара. |
| FirmMarkup | numeric(2,0) | - | Наценка фирмы. |
| Shipped | bit | - | Флаг отгрузки товара. |

# Запись выражений указанных в варианте задания типов запросов на языке SQL

* 1. **Процедура выборки перечня товаров, находящихся в указанной торговой точке**

CREATE PROCEDURE getAllSalesOutletsWares

@soid int

AS

BEGIN

SELECT Wares.UPC AS [Штрих-код товара], WareType AS [Категория],

SuplierName AS Поставщик, SUM(CurrQuantity) AS [Всего в наличии],

ISNULL(SUM(RequestWares.Quantity - DoneQuantity),0) AS [Из них заказано]

FROM ReceivedWares AS rw

JOIN DeliveryOnConsignmentNote AS d

ON (rw.FKDelivery = d.IDDelivery AND d.FKSalesOutlet = @soid)

JOIN SuppliersWares AS sw ON sw.IDSuppliersWare = rw.FKSuppliersWare

JOIN Suppliers ON Suppliers.IDSupplier = sw.FKSupplier

JOIN Wares ON sw.FKWare = Wares.UPC

JOIN WareTypes ON WareTypes.IDWareType = Wares.FKWareType

LEFT JOIN Requests ON (d.FKSalesOutlet = Requests.FKShopExec)

LEFT JOIN RequestWares ON (RequestWares.FKRequest = Requests.IDRequest

AND RequestWares.FKWare = sw.FKWare AND RequestWares.FKSupplier = sw.FKSupplier)

WHERE CurrQuantity > 0

GROUP BY Wares.UPC, WareType, SuplierName

END

Тестирование данной процедуры представлено на рисунке 4.1.

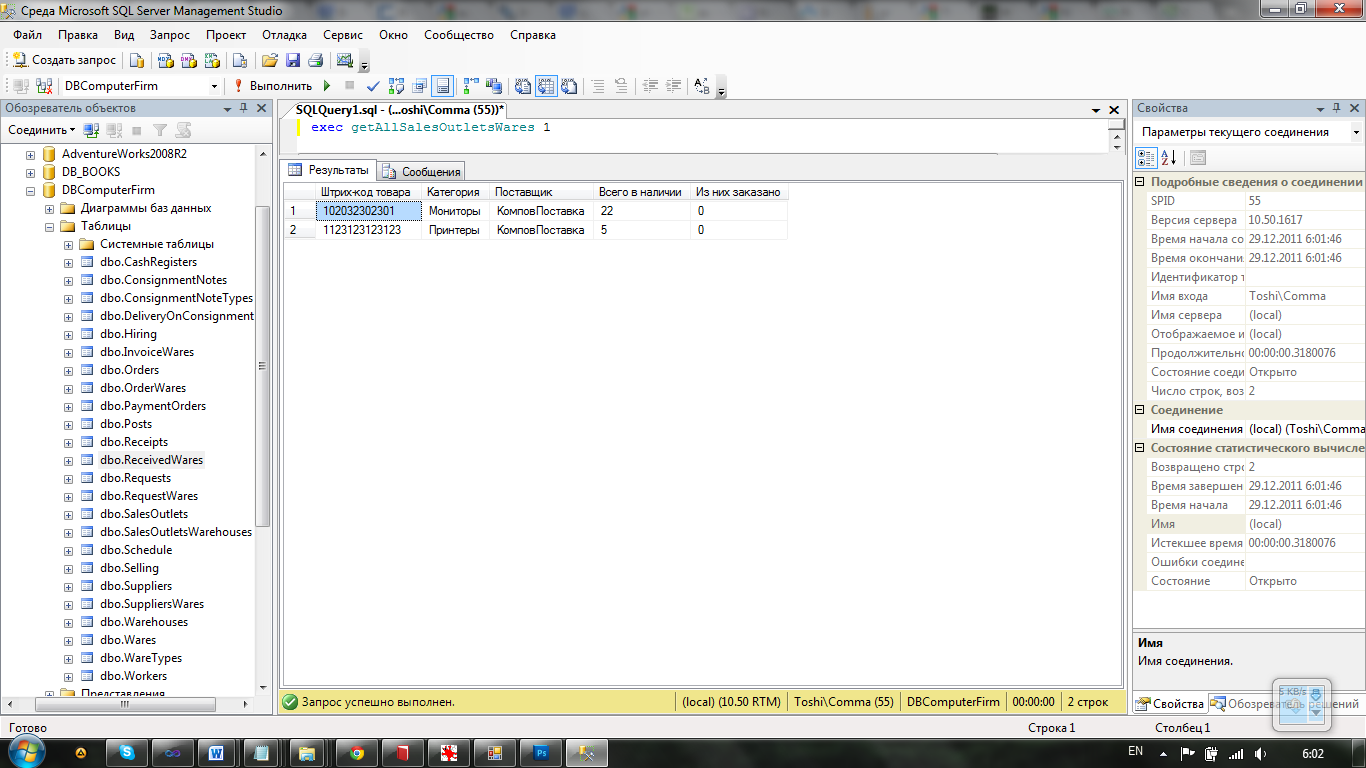


Рисунок 4.1 – Результат выполнения процедуры выборки перечня товаров ,   
находящихся в указанной торговой точке

* 1. **Процедура для вывода состояния центрального склада фирмы по состоянию на заданное число**

CREATE PROCEDURE getAllBaseWaresByDate

@date date

AS

BEGIN

SELECT UPC AS [Штрих-код товара], WareType AS Категория, SuplierName AS Поставщик, SUM(summ) AS [Кол-во на конец дня]

FROM

( -- Найдем все прибывшие на базы товары до данной даты вкл.

(SELECT FKSuppliersWare AS id,SUM(Quantity) AS summ

FROM ReceivedWares AS rw JOIN DeliveryOnConsignmentNote AS d

ON rw.FKDelivery = d.IDDelivery

WHERE d.FKBase IS NOT NULL AND d.DeliveryDate <= @date

GROUP BY FKSuppliersWare)

UNION

-- Найдем все выбывшие с баз товары до данной даты вкл. c отриц. знаком

(SELECT FKSuppliersWare AS id,-SUM(Quantity) AS summ

FROM ReceivedWares AS rw JOIN DeliveryOnConsignmentNote AS d

ON rw.FKDelivery = d.IDDelivery JOIN ConsignmentNotes AS cn

ON cn.IDConsignmentNote = d.FKConsignmentNote

WHERE cn.FKBase IS NOT NULL AND d.DeliveryDate <= @date

GROUP BY FKSuppliersWare)) AS aw

JOIN SuppliersWares AS sw ON aw.id = sw.IDSuppliersWare

JOIN Suppliers ON Suppliers.IDSupplier = sw.FKSupplier

JOIN Wares ON Wares.UPC = sw.FKWare

JOIN WareTypes ON WareTypes.IDWareType = Wares.FKWareType

GROUP BY UPC, WareType, SuplierName

HAVING SUM(summ)> 0

END

GO

В данной процедуре осуществляются 2 выборки: во-первых, выбираются все поступления на базы до заданного числа включительно, во-вторых, все выбывания по накладным до этого же числа со знаком минус. Далее 2 выборки объединяются, группируются по товару, а количество товара складывается (за сче знака минус выбывшие товары отнимаются). Продажи не учитываются, так как продавать товары право имеют только торговые точки – склады используются сугубо для промежуточного хранения товаров.

Тестирование данной процедуры представлено на рисунке 4.2.

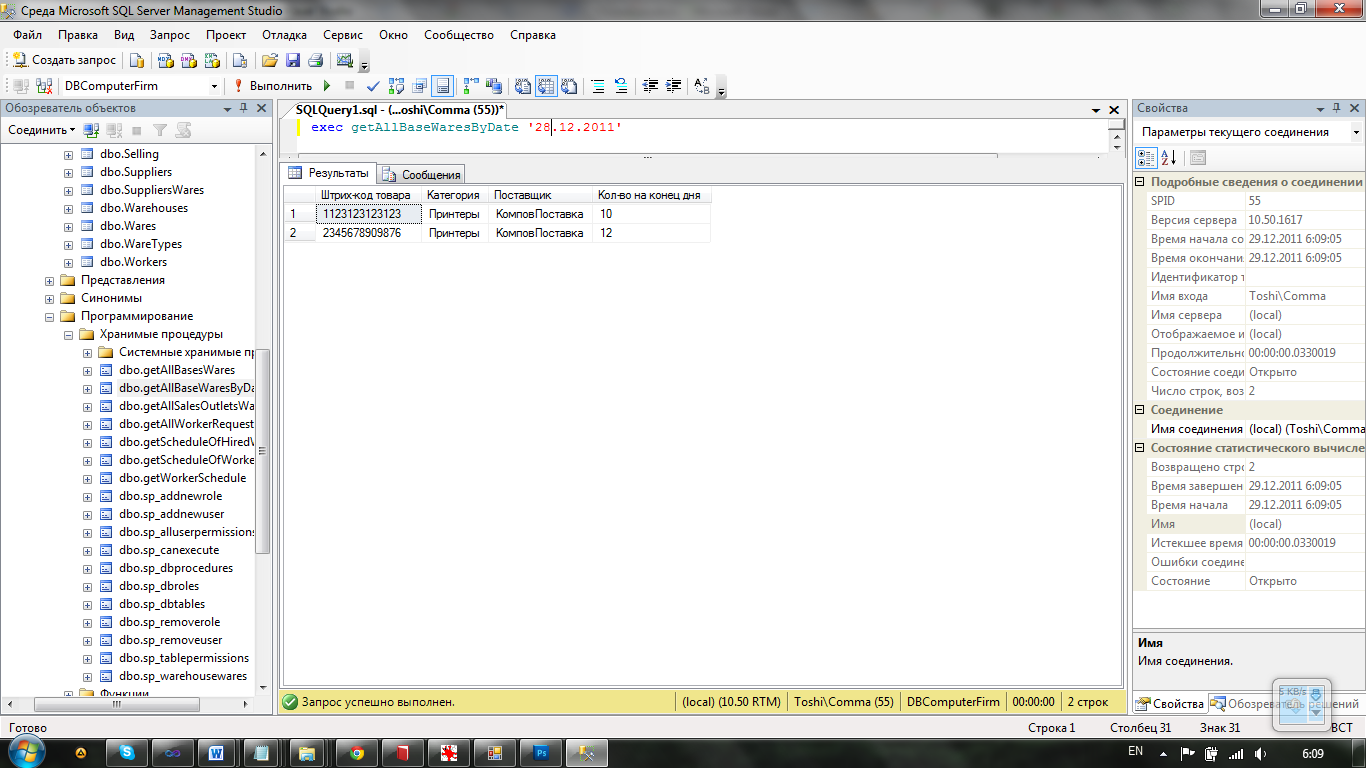


Рисунок 4.2 – Состояние центрального склада фирмы на 28.12.2011

Без объединения и группировки выборка выглядела бы иначе, более наглядно. Это можно продемонстрировать сделав 2 выборки отдельно. Резульатт представлен на рисунке 4.3.

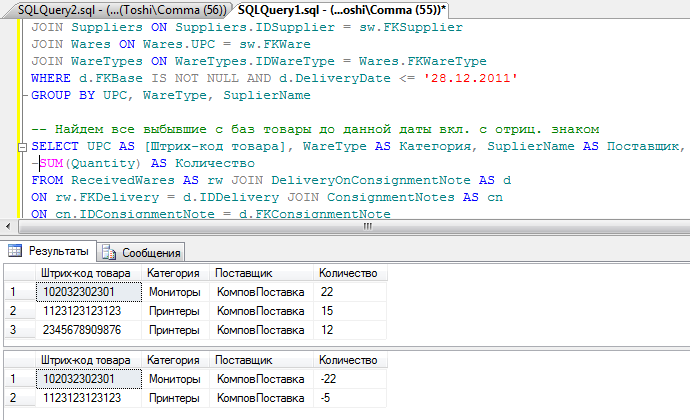


Рисунок 4.3 – Результат работы процедуры, выводящей состояние   
центрального склада в «разобранном» виде на 28.12.2011

* 1. **Представление, отображающее «общий» склад фирмы**

CREATE VIEW AllCurrentWares

AS

SELECT UPC AS [Штрих-код товара], WareType AS Категория,

SuplierName AS Поставщик, SUM(CurrQuantity) AS [Всего в наличии]

FROM ReceivedWares AS rw

JOIN DeliveryOnConsignmentNote AS d ON rw.FKDelivery = d.IDDelivery

JOIN SuppliersWares AS sw ON sw.IDSuppliersWare = rw.FKSuppliersWare

JOIN Suppliers ON Suppliers.IDSupplier = sw.FKSupplier

JOIN Wares ON FKWare = Wares.UPC

JOIN WareTypes ON WareTypes.IDWareType = Wares.FKWareType

WHERE (rw.CurrQuantity > 0)

GROUP BY UPC, WareTypes.WareType, SuplierName

Данное представление не нуждается в пояснениях. Просто продемонстрируем результаты её работы на рисунке 4.4.

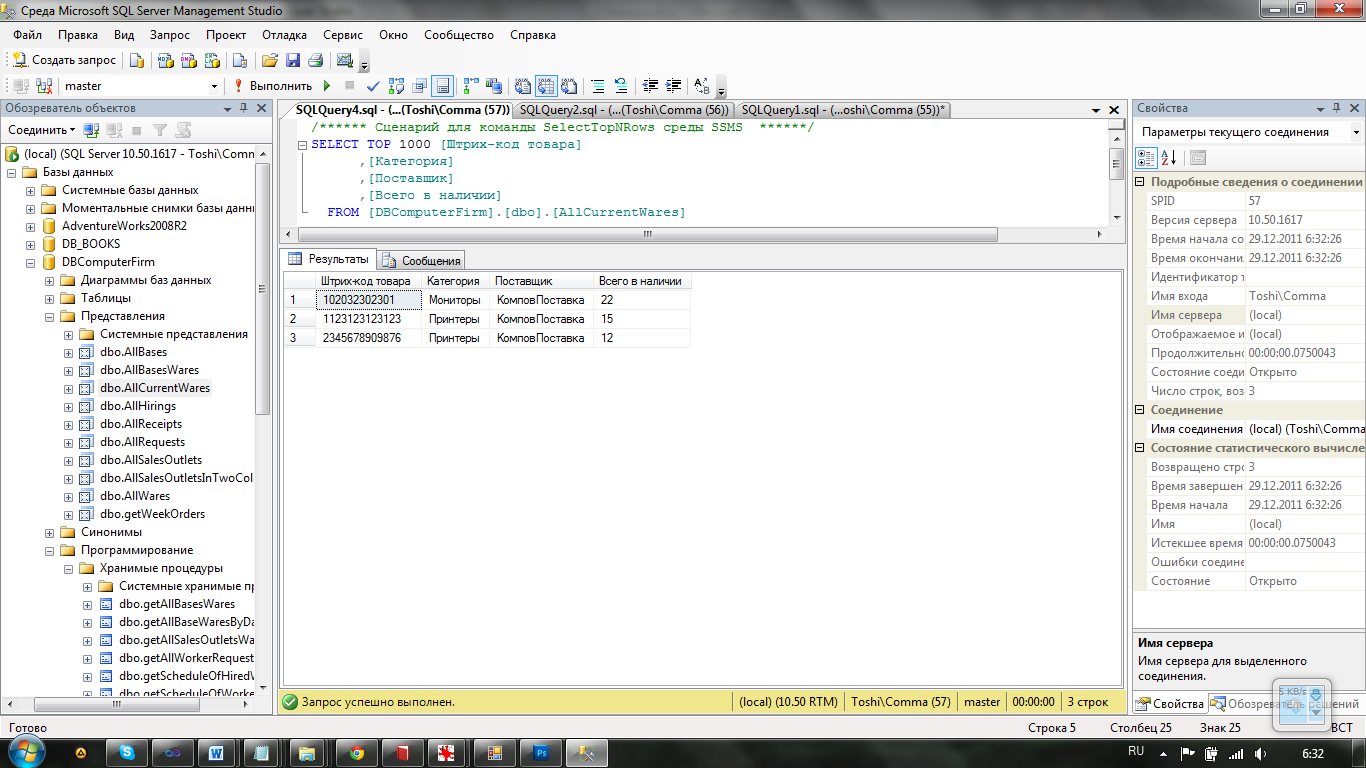


Рисунок 4.4 – Результат выборки из представления, отображающего  
 «общий» склад фирмы (все имеющиеся в наличии товары на базах и торговых точках)

* 1. **Процедура, выводящая информацию о сделках конкретного продавца с делением на наличные и безнал (выводится ворма оплаты, сумма и дата сделки)**

CREATE PROCEDURE AllWorkerSellings

@worker int

AS

BEGIN

SELECT 'наличные' AS [Форма оплаты],

SellingDate AS [Дата продажи], dbo.Selling.Amount AS [Сумма продажи]

FROM Selling

JOIN Workers ON Workers.IDWorker = Selling.FKWorker

JOIN AllSalesOutletsInTwoColumns AS aso ON Selling.FKSalesOutlet = aso.id

WHERE FKPaymentOrder IS NULL AND FKWorker = @worker

UNION

SELECT 'безнал' AS [Форма оплаты],

SellingDate AS [Дата продажи], dbo.Selling.Amount AS [Сумма продажи]

FROM Selling

JOIN Workers ON Workers.IDWorker = Selling.FKWorker

JOIN AllSalesOutletsInTwoColumns AS aso ON Selling.FKSalesOutlet = aso.id

WHERE FKReceipt IS NULL AND FKWorker = @worker

END

GO

Результаты тестирования для работника Быкова К.А. после внесения в базу двух сделок для данного продавца (одной «наличной», второй «безналичной») продемонстрированы на рисунке 4.5.

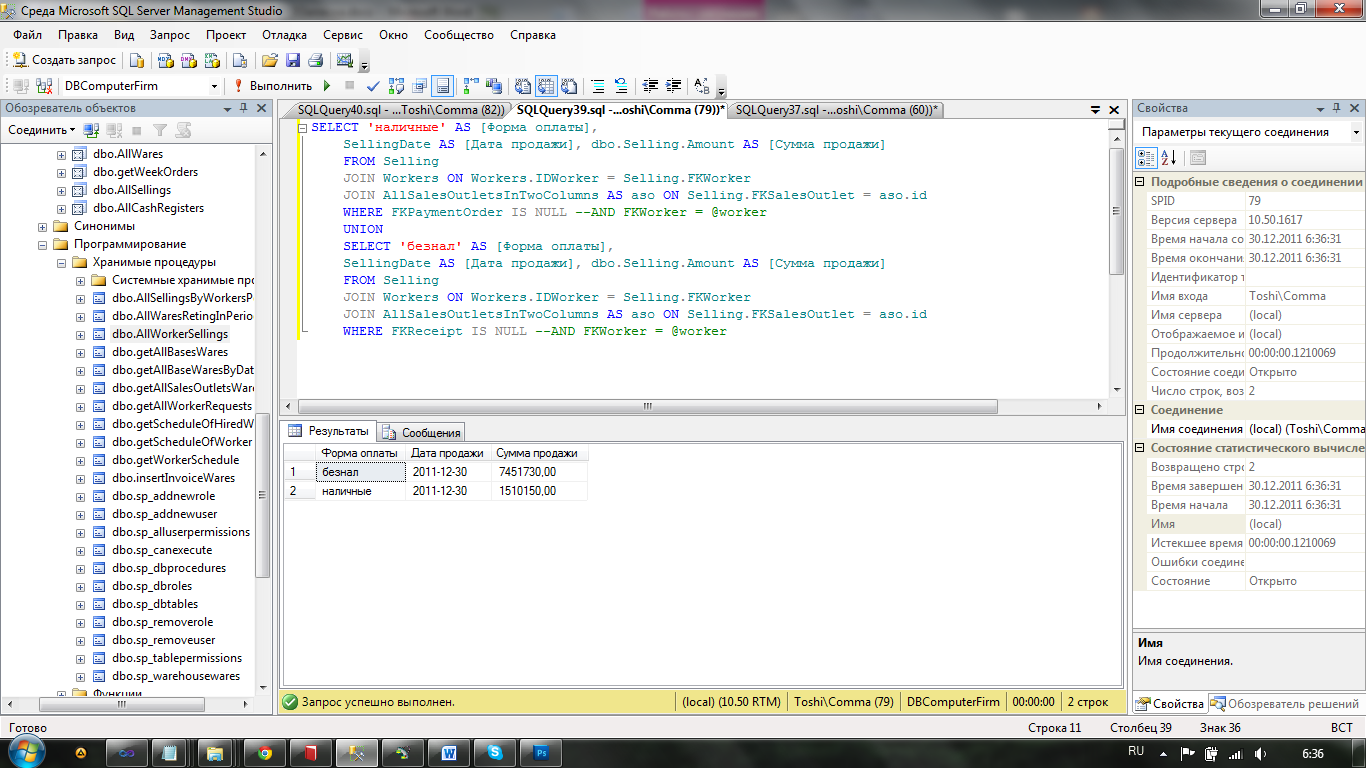


Рисунок 4.5 – Результат выборки всех сделок с делением   
на «наличные» и «безналичные» для продавца Быкова К.А.

* 1. **Процедура, выводящая сведения о продажах продавцов (общая сумма сделок) за заданный перод времени**

CREATE PROCEDURE AllSellingsByWorkersPeriod

@dateMin date,

@dateMax date

AS

BEGIN

SELECT [Продавец], SUM([Сумма продажи]) AS [Сумма продажи]

FROM AllSellings

WHERE [Дата продажи]>=@dateMin AND [Дата продажи]<=@dateMax

GROUP BY [Продавец]

END

GO

В данном случае выборка осуществляется из представления AllSellings, которое имеет следующий скрипт создания:

CREATE VIEW AllSellings

SELECT Selling.IDSelling AS [Номер продажи], aso.so AS [Торговая точка],

(dbo.Workers.Surname + ' ' + LEFT(dbo.Workers.Name, 1) + '.' + ISNULL(NULLIF (LEFT(dbo.Workers.Lastname, 1) + '.', '.'), '') + ' (') + (CAST(dbo.Workers.PassportSeries AS varchar(4)) + CAST(dbo.Workers.PassportNumber AS varchar(7))) + ')' AS Продавец, ISNULL(dbo.Selling.FKPaymentOrder, '-') AS [№ п\п], ISNULL(dbo.Selling.FKReceipt, '-') AS [№ чека], dbo.Selling.SellingDate AS [Дата продажи], dbo.Selling.Amount AS [Сумма продажи]

FROM Selling JOIN Workers ON Workers.IDWorker = Selling.FKWorker JOIN

AllSalesOutletsInTwoColumns AS aso ON dbo.Selling.FKSalesOutlet = aso.id

Результаты тестирования для работника Быкова К.А. после внесения в базу двух сделок для данного продавца (одной «наличной», второй «безналичной») продемонстрированы на рисунке 4.6 (можно сравнить с рис.4.5).

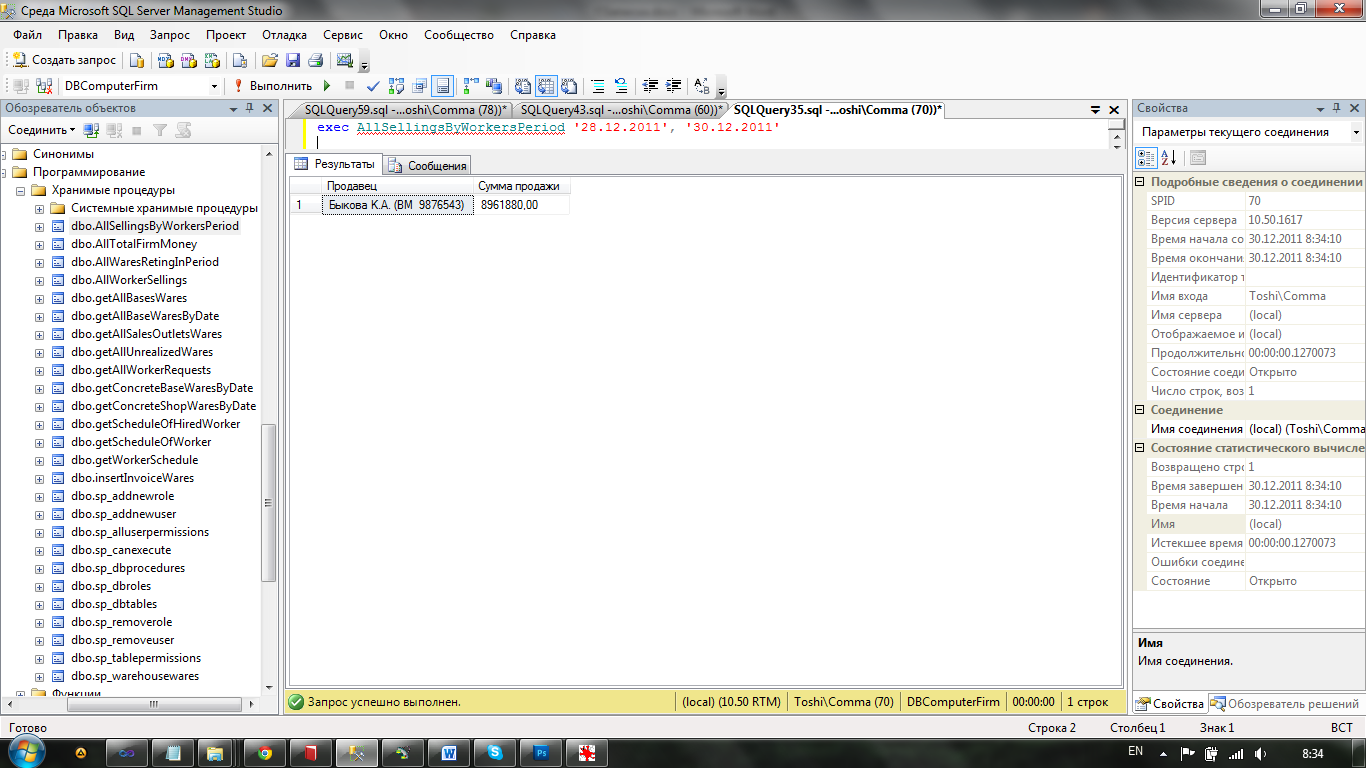


Рисунок 4.6 – Результат выборки общей суммы   
сделок продавцов за заданный период

* 1. **Процедура, выводящая перечень проданных за указанный период товаров в порядке убывания их популярности.**

CREATE PROCEDURE AllWaresRetingInPeriod

@dateMin date,

@dateMax date

AS

BEGIN

SELECT UPC AS [Штрих-код товара], SUM(iw.Quantity) AS [Проданное количество]

FROM AllSellings AS alls

JOIN InvoiceWares AS iw ON alls.[Номер продажи] = iw.FKSelling

JOIN ReceivedWares AS rw ON iw.FKReceivedWare = rw.IDReceivedWare

JOIN SuppliersWares AS sw ON sw.IDSuppliersWare = rw.FKSuppliersWare

JOIN Wares ON Wares.UPC = sw.FKWare

WHERE alls.[Дата продажи] >= @dateMin AND alls.[Дата продажи] <= @dateMax

GROUP BY UPC

ORDER BY SUM(iw.Quantity) DESC

END

Как упоминалось ранее, в базу были внесены две сделки. Были приобретены: 5 экземпляров товара со штрих-кодом 102032302301 и 1 экземпляр товара со штрих-кодом 1123123123123 в контексте первой сделки; 1 экземпляр товара со штрих-кодом 102032302301 – второй. Результаты тестирования продемонстрированы на рисунке 4.7:

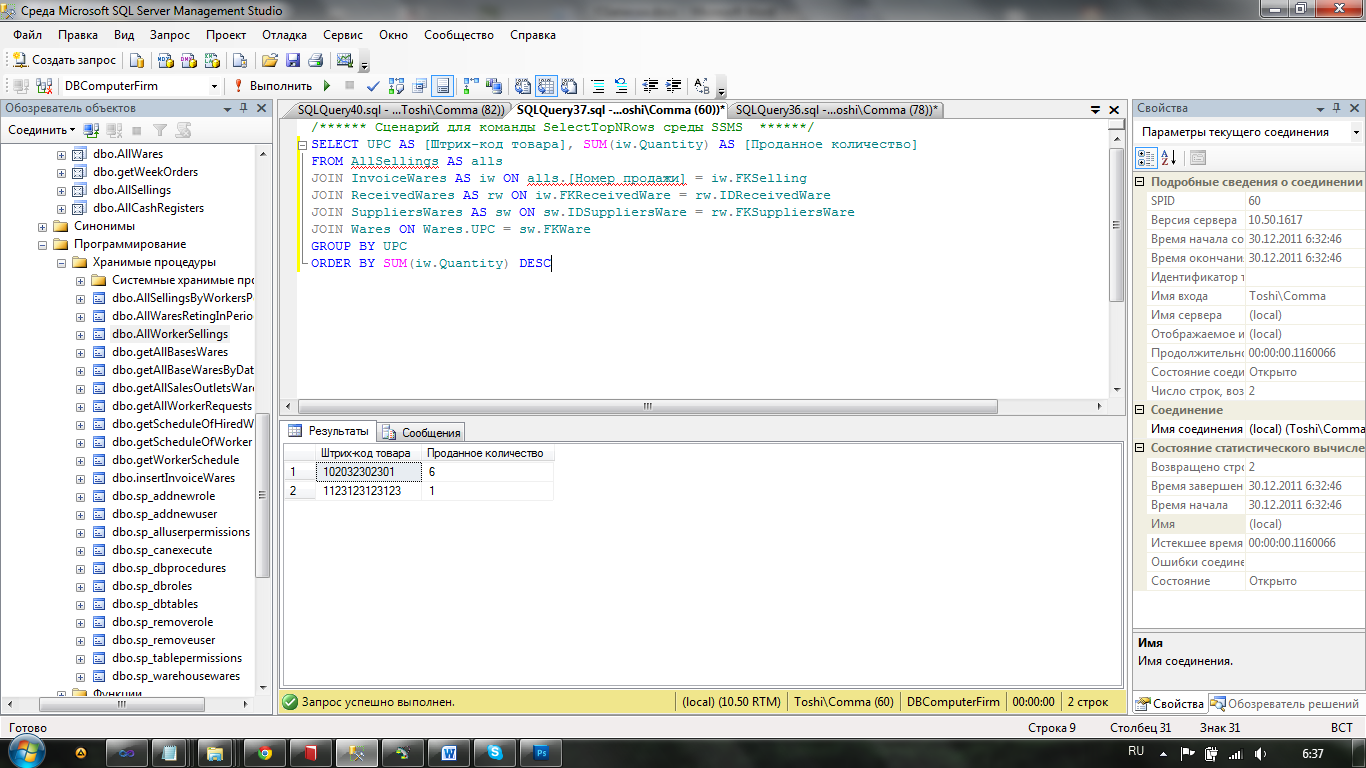


Рисунок 4.7 – Результат выборки проданных в заданном промежутке   
(28.12.2011-30.12.2011) времени товаров с сортировкой по популярности товара

* 1. **Процедура, выводящая суммарный доход фирмы за указанный период времени, отдельно учитывая «наличные» и «безналичные»**

CREATE PROCEDURE [dbo].[AllTotalFirmMoney]

@dateMin date,

@dateMax date

AS

BEGIN

SELECT 'наличные' AS [Форма оплаты], SUM([Сумма продажи]) AS [Сумма продажи]

FROM AllSellings

WHERE [№ п\п] = '-' AND [Дата продажи]>=@dateMin AND [Дата продажи]<=@dateMax

UNION

SELECT 'безнал' AS [Форма оплаты], SUM([Сумма продажи]) AS [Сумма продажи]

FROM AllSellings

WHERE [№ чека] = '-' AND [Дата продажи]>=@dateMin AND [Дата продажи]<=@dateMax

END

Как упоминалось ранее, в базу были внесены две сделки (рис.4.5). Таким образом, суммарный доход за тот же период времени будет соответствовать выборке сделок продавца Быковой К.А. Результаты можно увидеть на рисунке 4.8.

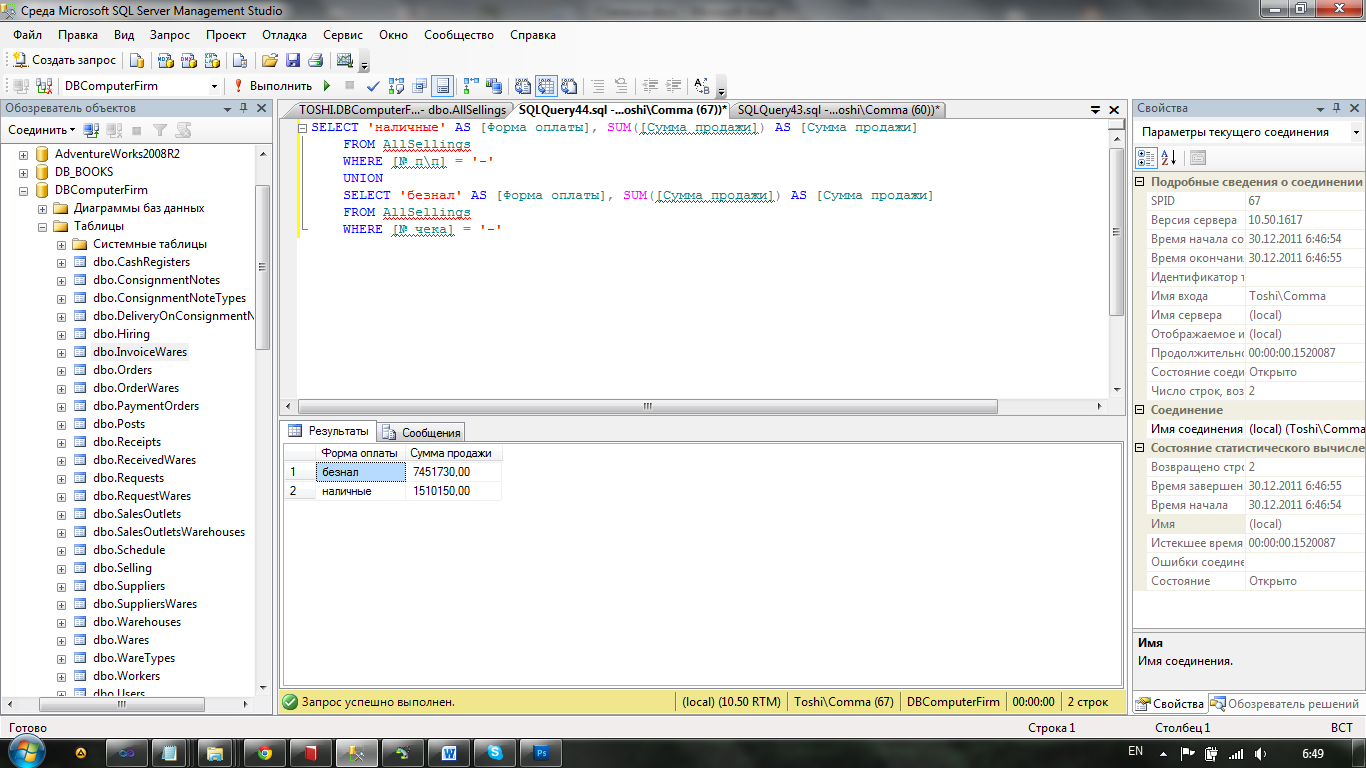


Рисунок 4.8 – Результат выводы суммарного дохода фирмы за заданный промежуток времени (28.12.2011-30.12.2011).

* 1. **Процедура, отображающая перечень товаров с указанием торговой точки, не реализованных в течение указанного периода времени**

Для реализации данной процедуры нам понадобятся 2 вспомогательные процедуры. Вспомогательная процедура, возвращающая все товары баз на определенную дату:

CREATE PROCEDURE [dbo].[getConcreteBaseWaresByDate]

@date Date

AS

BEGIN

SELECT w, UPC AS [Штрих-код товара], WareType AS Категория,

SuplierName AS Поставщик, SUM(summ) AS [Кол-во на конец дня]

FROM(SELECT d.FKBase AS base, FKSuppliersWare AS id,SUM(Quantity) AS summ

FROM ReceivedWares AS rw JOIN DeliveryOnConsignmentNote AS d

ON rw.FKDelivery = d.IDDelivery

WHERE d.FKBase IS NOT NULL AND d.DeliveryDate <= @date

GROUP BY d.FKBase, FKSuppliersWare

UNION

-- Найдем все выбывшие с баз товары до данной даты вкл. c отриц. знаком

SELECT cn.FKBase AS base,FKSuppliersWare AS id,-SUM(Quantity) AS summ

FROM ReceivedWares AS rw JOIN DeliveryOnConsignmentNote AS d

ON rw.FKDelivery = d.IDDelivery JOIN ConsignmentNotes AS cn

ON cn.IDConsignmentNote = d.FKConsignmentNote

WHERE cn.FKBase IS NOT NULL AND d.DeliveryDate <= @date

GROUP BY cn.FKBase, FKSuppliersWare) AS aw

JOIN SuppliersWares AS sw ON aw.id = sw.IDSuppliersWare

JOIN Suppliers ON Suppliers.IDSupplier = sw.FKSupplier

JOIN Wares ON Wares.UPC = sw.FKWare

JOIN WareTypes ON WareTypes.IDWareType = Wares.FKWareType

JOIN AllBases AS ab ON ab.id = base

GROUP BY w, UPC, WareType, SuplierName

HAVING SUM(summ)> 0

GO

Вспомогательная процедура, возвращающая все товары магазинов на определенную дату:

CREATE PROCEDURE [dbo].[getConcreteShopWaresByDate]

@date Date

AS

BEGIN

SELECT so, UPC AS [Штрих-код товара], WareType AS Категория,

SuplierName AS Поставщик, SUM(summ) AS [Кол-во на конец дня]

FROM(

-- Найдем все прибывшие по накладным товары до данной даты

SELECT d.FKSalesOutlet AS shop, FKSuppliersWare AS id,SUM(Quantity) AS summ

FROM ReceivedWares AS rw JOIN DeliveryOnConsignmentNote AS d

ON rw.FKDelivery = d.IDDelivery

WHERE d.FKSalesOutlet IS NOT NULL AND d.DeliveryDate <= @date

GROUP BY d.FKSalesOutlet, FKSuppliersWare

UNION

-- Найдем все выбывшие с баз товары до данной даты вкл. c отриц. знаком

SELECT cn.FKSalesOutlet AS shop,FKSuppliersWare AS id,-SUM(Quantity) AS summ

FROM ReceivedWares AS rw JOIN DeliveryOnConsignmentNote AS d

ON rw.FKDelivery = d.IDDelivery JOIN ConsignmentNotes AS cn

ON cn.IDConsignmentNote = d.FKConsignmentNote

WHERE cn.FKSalesOutlet IS NOT NULL AND d.DeliveryDate <= @date

GROUP BY cn.FKSalesOutlet, FKSuppliersWare

UNION

-- Найдем все проданные товары до данной даты вкл. c отриц. знаком

SELECT FKSalesOutlet AS shop, FKSuppliersWare AS id,-SUM(iw.Quantity) AS summ

FROM ReceivedWares AS rw JOIN InvoiceWares AS iw

ON rw.IDReceivedWare = iw.FKReceivedWare JOIN

Selling ON Selling.IDSelling = iw.FKSelling

WHERE Selling.SellingDate <= @date

GROUP BY FKSalesOutlet, FKSuppliersWare) AS aw

JOIN SuppliersWares AS sw ON aw.id = sw.IDSuppliersWare

JOIN Suppliers ON Suppliers.IDSupplier = sw.FKSupplier

JOIN Wares ON Wares.UPC = sw.FKWare

JOIN WareTypes ON WareTypes.IDWareType = Wares.FKWareType

JOIN AllSalesOutletsInTwoColumns AS allso ON shop = allso.id

GROUP BY so, UPC, WareType, SuplierName

HAVING SUM(summ)> 0

END

GO

Текст самой процедуры:

CREATE PROCEDURE [dbo].[getAllUnrealizedWares]

@dateMin Date,

@dateMax Date

AS

BEGIN

DECLARE @date Date

SET @date = DATEADD(day,-1,@dateMin)

DECLARE @table TABLE (id varchar(100), UPC numeric(13,0), wtype varchar(30),

supplier varchar(50), summ int)

INSERT INTO @table

EXEC getConcreteBaseWaresByDate @date

INSERT INTO @table

EXEC getConcreteBaseWaresByDate @dateMax

INSERT INTO @table

EXEC getConcreteShopWaresByDate @date

INSERT INTO @table

EXEC getConcreteShopWaresByDate @dateMax

SELECT id AS [Точка или склад], UPC AS [Штрих-код товара],

supplier AS [Поставщик], MIN(summ) AS [Нереализованное количество]

FROM @table

GROUP BY id, UPC, supplier

END

В данной процедуре находится текущее количество товаров на утро меньшей даты и на вечер большей. Далее после группировки по товару на конкретной точке либо базе выбирается минимальное количество товара за эти два дня. Так как, если за заданный промежуток времени был совершен завоз, и товара стало больше, можно сказать, что товары, имеющиеся на меньшую дату, были полностью и всецело не реализованы; если же товар был частично распродан, то остаток можно считать за нереализованный товар. Результаты тестирования представлены на рисунке 4.9.

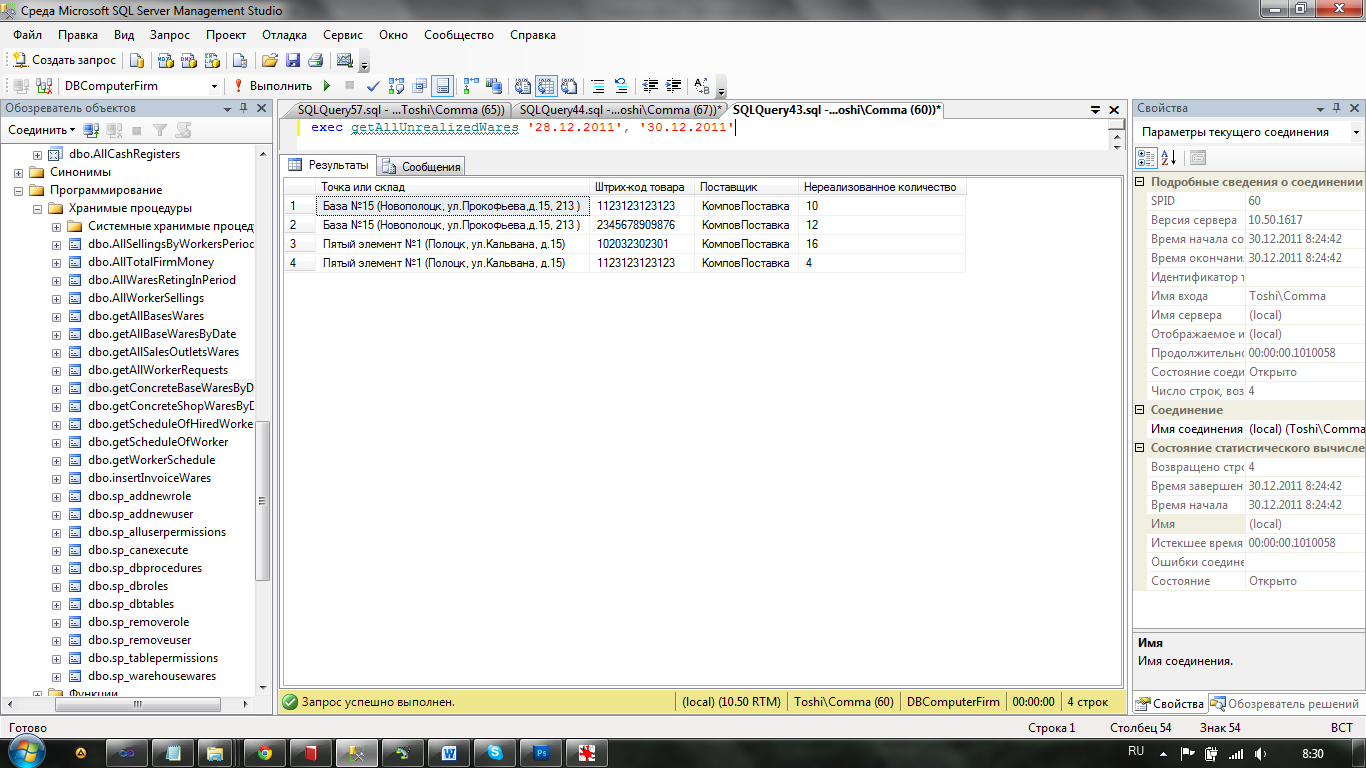


Рисунок 4.9 – Результат вывода нереализованного в течение   
заданного промежутка времени (28.12.2011-30.12.2011) товара

* 1. **Процедура, отображающая заказы указанного продавца за текущую неделю.**

CREATE PROCEDURE [dbo].[getAllWorkerRequests]

@workerid int

AS

BEGIN

SELECT so AS [На торговую точку], WareType AS [Категория товара], UPC AS [Штрих-код товара],

SUM(Quantity) AS Количество

FROM Requests JOIN RequestWares ON Requests.IDRequest = RequestWares.FKRequest

JOIN AllSalesOutletsInTwoColumns AS allshops ON allshops.id = Requests.FKShopClient

JOIN Wares ON RequestWares.FKWare = Wares.UPC

JOIN WareTypes ON Wares.FKWareType = WareTypes.IDWareType

WHERE Requests.FKSeller = 7

AND DATEDIFF(day, RequestDate, GETDATE()) < 8

GROUP BY so, WareType, UPC

END

Результат тестирования для работника Волохи Г.В. (Порядковый номер - 9) представлены на рисунке 4.10.

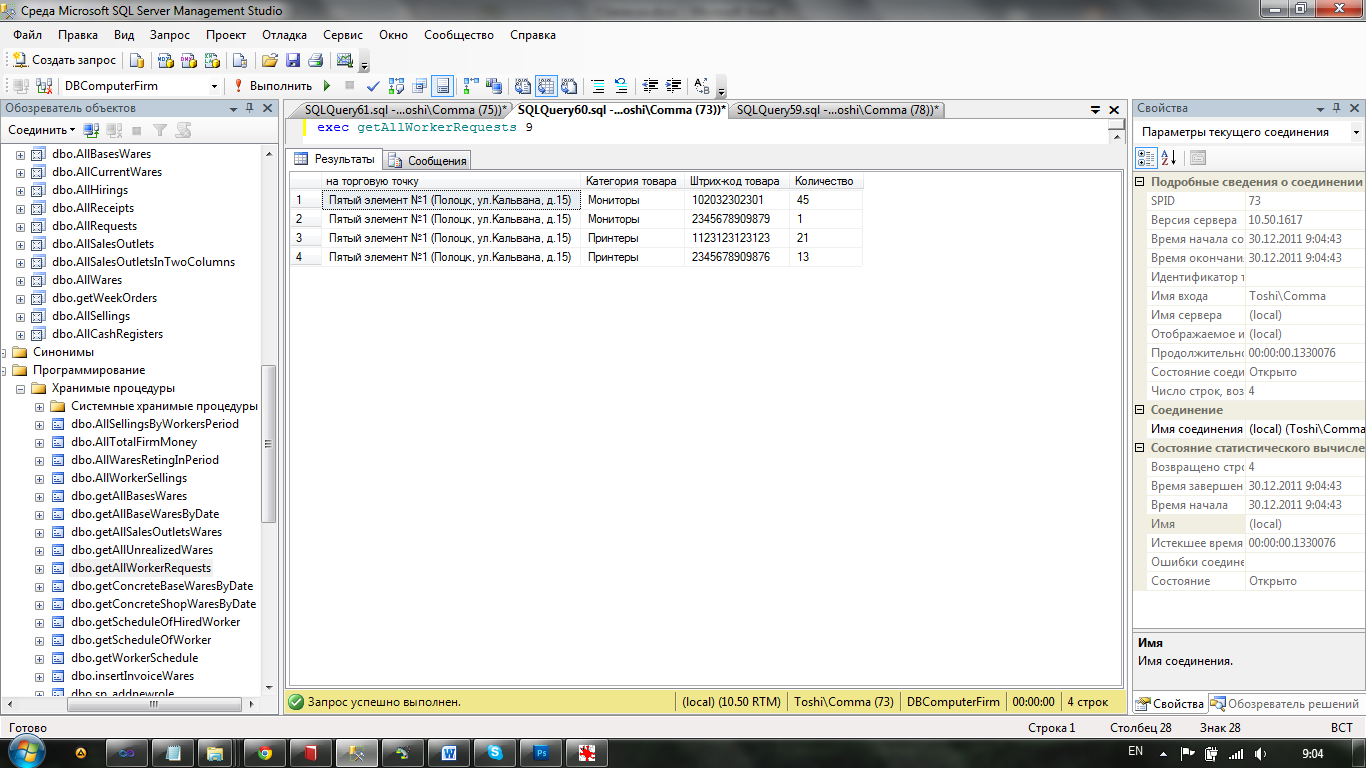


Рисунок 4.10 – Все заказы заданного работника

**4.10 Представление, отображающее общий недельный заказ фирмы с указанием фирм-поставщиков.**

CREATE VIEW getWeekOrders

AS

SELECT dbo.WareTypes.WareType AS [Категория товара], dbo.RequestWares.FKWare AS Товар, SUM(dbo.RequestWares.Quantity) AS Количество,

dbo.Suppliers.SuplierName AS Поставщик

FROM dbo.Orders INNER JOIN

dbo.Suppliers ON dbo.Suppliers.IDSupplier = dbo.Orders.FKSupplier INNER JOIN

dbo.OrderWares ON dbo.OrderWares.FKOrder = dbo.Orders.IDOrder INNER JOIN

dbo.RequestWares ON dbo.OrderWares.FKRequestWare = dbo.RequestWares.IDRequestWare INNER JOIN

dbo.Wares ON dbo.Wares.UPC = dbo.RequestWares.FKWare INNER JOIN

dbo.WareTypes ON dbo.WareTypes.IDWareType = dbo.Wares.FKWareType

WHERE (DATEDIFF(DAY, dbo.Orders.OrderDate, GETDATE()) < 8)

GROUP BY dbo.WareTypes.WareType, dbo.RequestWares.FKWare, dbo.Suppliers.SuplierName

Результат выполнения отображен на рисунке 4.11.

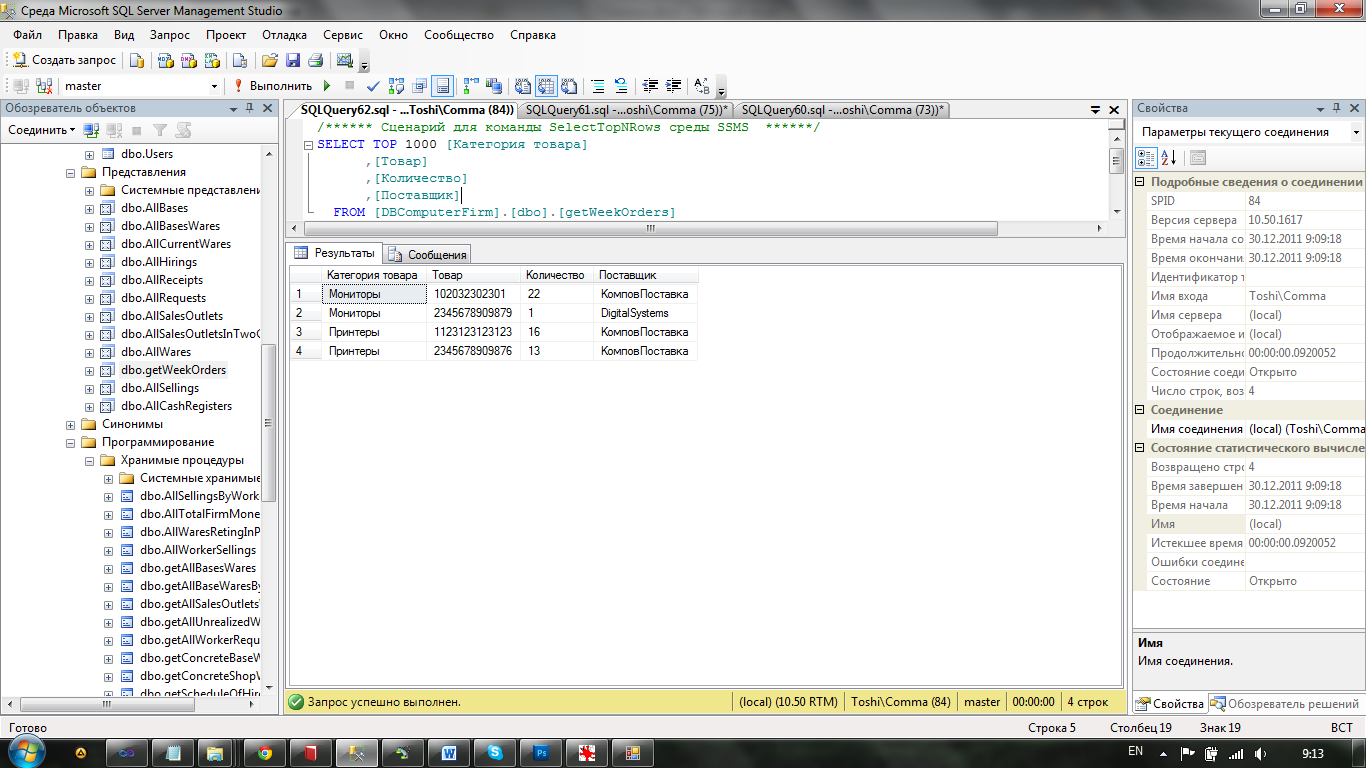


Рисунок 4.11 – Содержание всех заявок поставщкам за текущую  
 неделю вне зависимости от того, удовлетворена заявка или нет

* 1. **Процедура, возвращающая график работы конкретного продавца за месяц.**

CREATE PROCEDURE [dbo].[getWorkerSchedule] (

@worker INT

)

AS

BEGIN

DECLARE @date DATETIME

SET @date = GETDATE()

DECLARE @daycount INT

SET @daycount = (SELECT DAY(DATEADD(Month, 2, @Date) - DAY(DATEADD(Month, 1, @Date))))

DECLARE @iterator INT

SET @iterator = 1

DECLARE @day INT

SET @day = 1 - DAY(@date)

SET @date = DATEADD(day, @day, @date)

DECLARE @table table (Дата datetime, Смена bit, Место varchar(200))

DECLARE @dayname varchar(12)

DECLARE @dayweek int

DECLARE @addr varchar(200)

WHILE (NOT(@iterator > @daycount))

BEGIN

SET @dayname = datename(weekday, @date)

SET @dayweek = CASE @dayname

WHEN 'понедельник' THEN 1

WHEN 'вторник' THEN 2

WHEN 'среда' THEN 3

WHEN 'четверг' THEN 4

WHEN 'пятница' THEN 5

WHEN 'суббота' THEN 6

WHEN 'воскресенье' THEN 7

END

SET @addr = (SELECT [dbo].[getAddress](@worker,@dayweek,0))

INSERT INTO @table VALUES(@date, 0, @addr)

SET @addr = (SELECT [dbo].[getAddress](@worker,@dayweek,1))

INSERT INTO @table VALUES(@date, 1, @addr)

SET @iterator = @iterator + 1

SET @date = DATEADD(day, 1, @date)

END

SELECT \* FROM @table

END

Результаты тестирования процедуры на примере отображения графика работы Волохи Г.В. представлены на рисунке 4.12

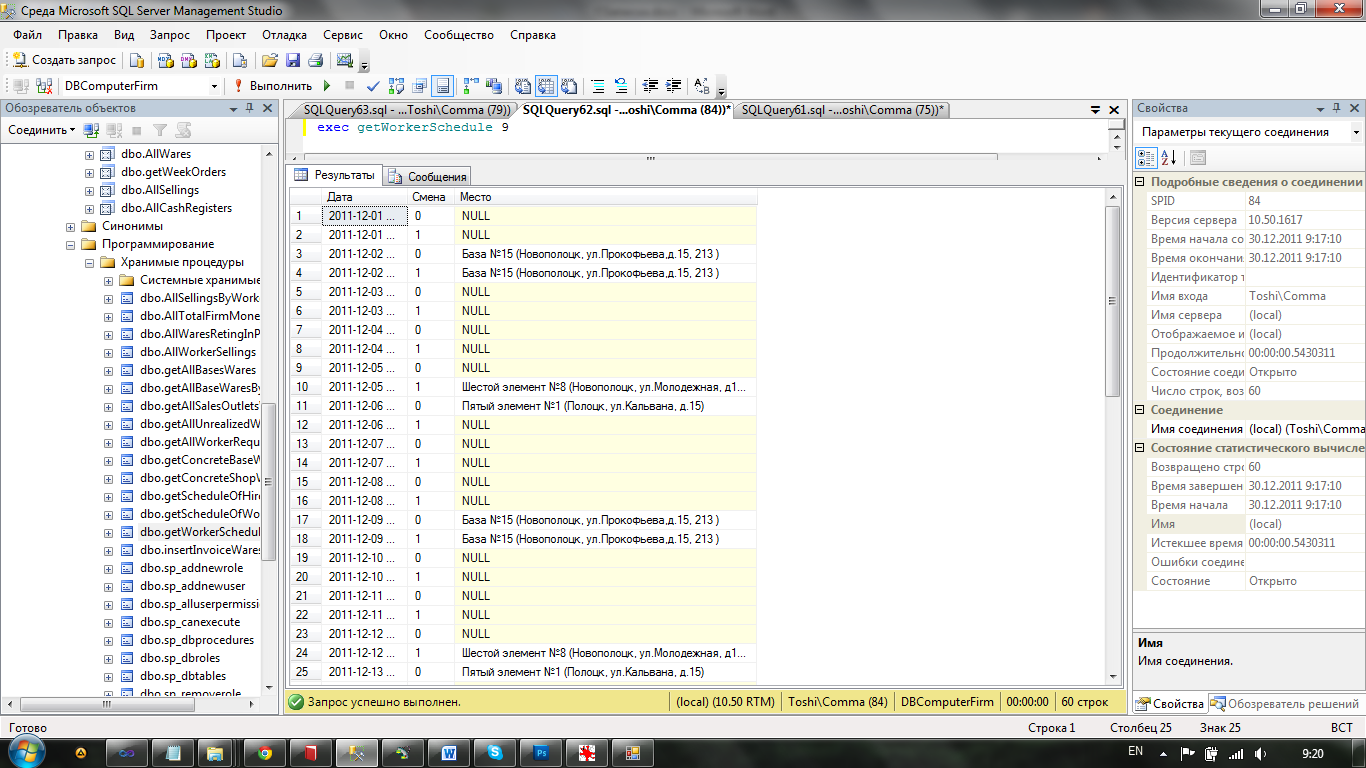


Рисунок 4.12 – Расписание за текущий месяц для работника Волохи Г.В.

* 1. **Представление, отображающее торговый оборот по торговым точкам (сумма выручки за месяц).**

CREATE VIEW AllMoneyByMonth

AS

SELECT [Торговая точка], SUM([Сумма продажи]) AS [Сумма продаж]

FROM dbo.AllSellings

WHERE (DATEDIFF(month, [Дата продажи], GETDATE()) = 0)

GROUP BY [Торговая точка]

Результаты тестирования представлены на рисунке 4.13.

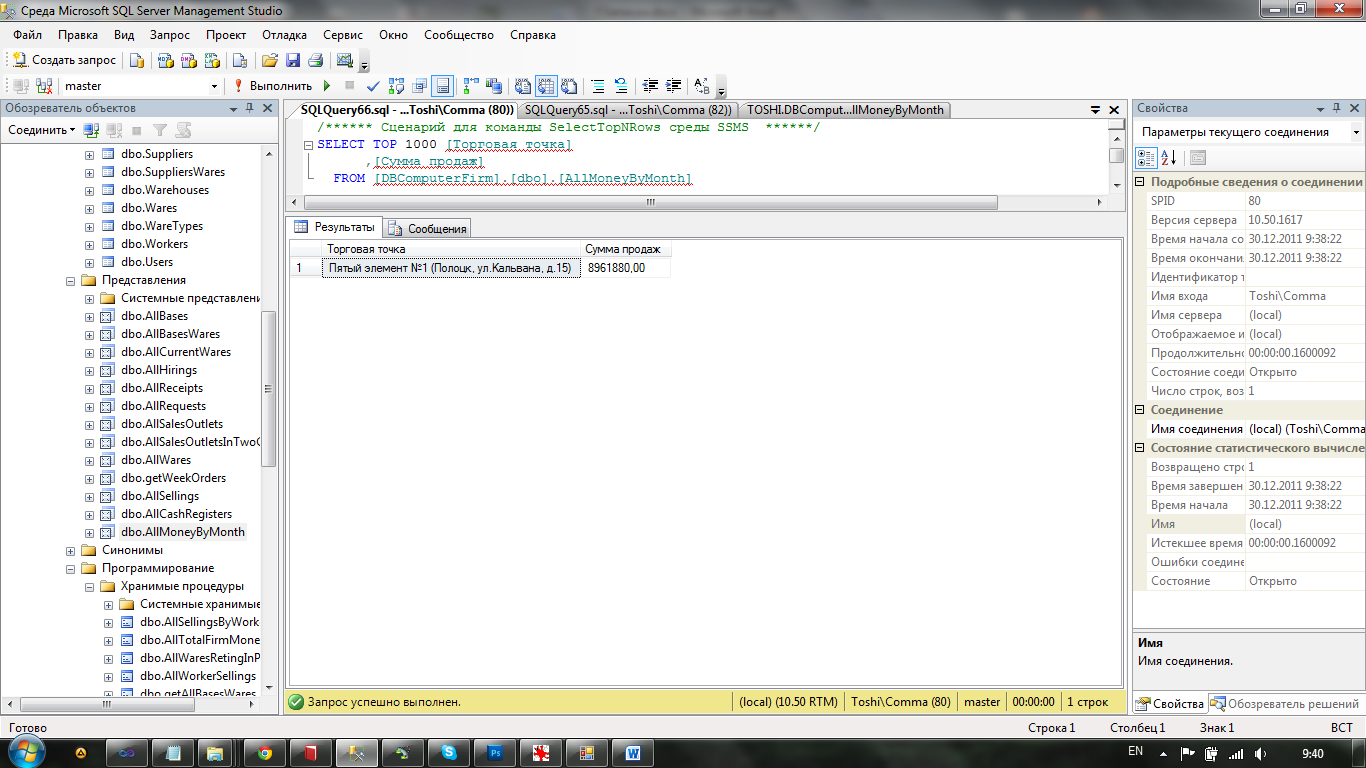


Рисунок 4.13 – Результаты тестирования представления, отображающего выручку торговых точек за месяц

* 1. **Представление, выводящее торговые точки, на которых был превышен лимит кассы.**

Так как лимит кассы – это предельная сумма наличных денег, которая может быть проведена через КСА, то предположим, что это ежедневный отчет. Соответственно, будем смотреть превышения лимита по текущей дате.

CREATE VIEW LimitCash

AS

SELECT dbo.AllSellings.[Торговая точка], dbo.CashRegisters.InventoryNumber AS [Регистрационный номер КСА], dbo.CashRegisters.LimitCash AS [Лимит кассы],

SUM(dbo.AllSellings.[Сумма продажи]) AS [Проведено через КСА], SUM(dbo.AllSellings.[Сумма продажи]) - dbo.CashRegisters.LimitCash AS [Превышено на]

FROM dbo.CashRegisters INNER JOIN

dbo.Receipts ON dbo.CashRegisters.InventoryNumber = dbo.Receipts.FKCashRegister INNER JOIN

dbo.AllSellings ON dbo.AllSellings.[№ чека] = dbo.Receipts.IDReceipt

WHERE (dbo.AllSellings.[Дата продажи] = GETDATE())

GROUP BY dbo.AllSellings.[Торговая точка], dbo.CashRegisters.InventoryNumber, dbo.CashRegisters.LimitCash

Результат тестирования представлен на рисунке 4.14.

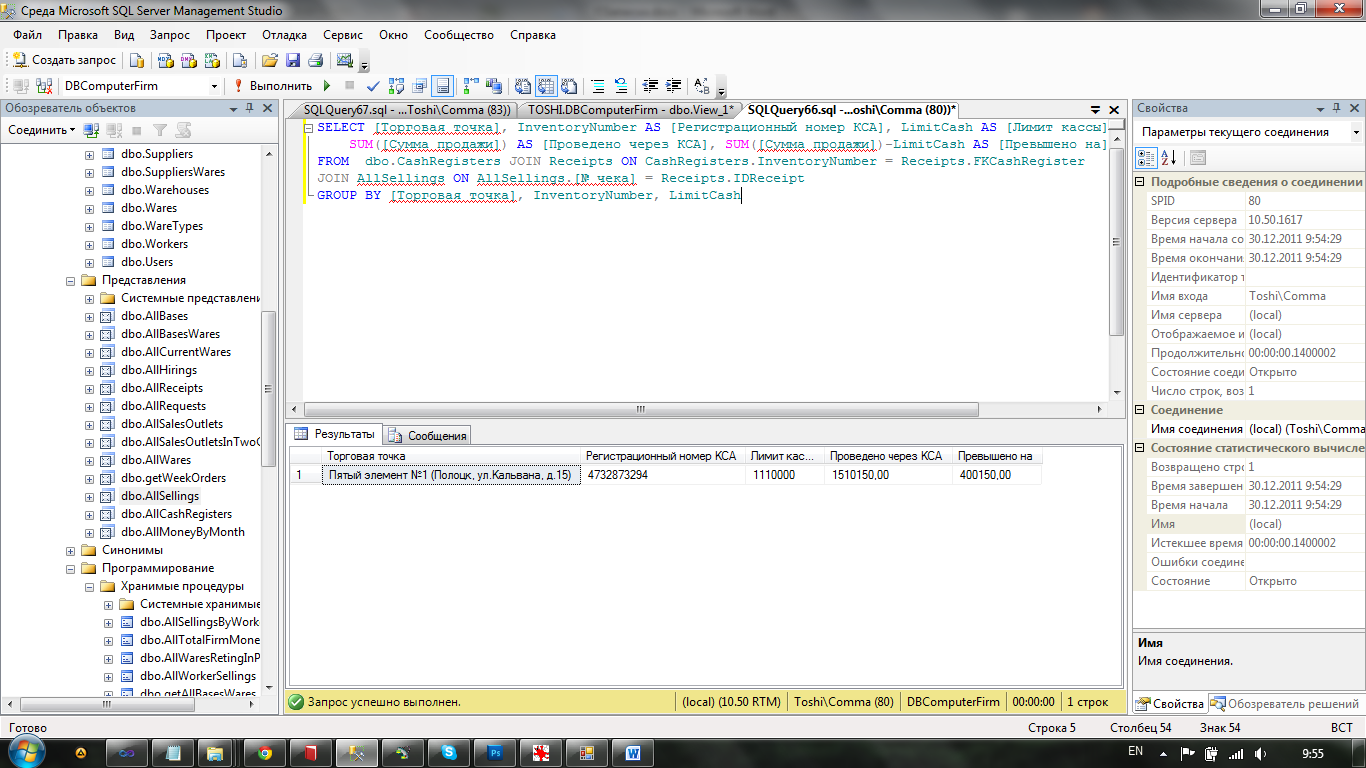


Рисунок 4.12 – Торговые точки на которых был превышен лимит кассы

# Выбор и основание средств разработки приложения

Выбор СУБД является сложной задачей и должен основываться, в первую очередь, на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или и то, и другое). Другими показателями, влияющими на выбор СУБД, являются[2]:

* + удобство и простота использования;
  + качество средств разработки, защиты и контроля базы данных;
  + уровень коммуникационных средств в случае применения ее в сетях;
  + фирма-разработчик;
  + стоимость.

Система SQL Server 2008 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio. SQL Server обеспечивает высочайший уровень безопасности, надежности и масштабируемости для критически важных приложений. Чтобы использовать новые возможности, постоянно возникающие в быстро меняющемся деловом мире, предприятиям нужно быть способными быстро создавать и развертывать решения, управляемые данными. SQL Server 2008 позволяет сократить затраты времени и средств, требуемые на управление и развертывание таких приложений. Также следует учесть, что фирма-разработчик данной СУБД является также разработчиком самой распространенной ОС. В финансовом плане важным фактором является то, что существуют бесплатные сборки данной СУБД (Express).

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2010, в качестве языка программирования – C#.

Достоинства платформы .NET [1]:

1) Вся платформа .NET основана на единой объектно-ориентированной модели. Все сервисы, интерфейсы и объекты, которые платформа предоставляет разработчику объединены в единую иерархию классов. Другими словами, все, что может вам потребоваться при создании приложений под платформу .NET будет всегда у вас под рукой. Причем, все это сгруппировано очень удобно и интуитивно понятно.

2) Приложение, написанное на любом .NET-совместимом языке является межплатформенным (в идеале). Почему в идеале? Дело в том, что приложение, написанное, скажем, на том же C#, не зависит от платформы, на которой будет выполняться, но зато зависит от наличия платформы .NET.

3) В состав платформы .NET входит "сборщик мусора", который освобождает ресурсы. Таким образом, приложения защищены от утечки памяти и от необходимости освобождать ресурсы. Это делает программирование более легким и более безопасным.

6) Приложения .NET используют безопасные типы, что повышает их надежность и совместимость.

7) .NET приложения могут быть сертифицированы на безопасность. Это является особенность промежуточного кода, в который преобразуются все .NET приложения.

8) Абсолютно все ошибки обрабатываются механизмом исключительных ситуаций. Это позволяет избежать разногласий, которые иногда возникают при программировании под Win32.

9) Повторное использование кода стало еще удобнее. Это связано с тем, что промежуточный язык MSIL не зависит от языка программирования. Например, вы можете написать программу на C#, а патч к ней писать уже, скажем, на J#.

# Реализация законченного приложения, работающего с созданной базой данных

## Разработка и построение интерфейса главной и рабочих форм

Прежде всего, в виду необходимости защиты информации, находящейся в базе, было создана форма авторизации (рис.6.1). Только после успешного ввода логина и пароля, либо удачной Windows-аутентификации пользователь может увидеть главную форму.

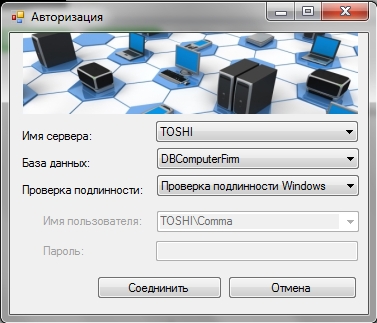


Рисунок 6.2 – Окно авторизации приложения

Главная форма построена на подобие MDI-форм: представляет собой окна, расположенные под одним общим окном. Главная форма содержит меню, разделенное на категории (подробнее в пункте 6.2). Все вторичные окна, за исключением диалоговых, являются вложенными - их нельзя вытащить за пределы формы, но можно развернуть на всю основную форму либо свернуть (при этом на нижней панели появится кнопка с соответствующим заголовком), что позволяет легко ориентироваться между окошками, открывать множество окон и предотвращает их «потерю» неопытным пользователем.

Внешний вид главной формы можно увидеть в приложении B (рис. B.1).

Основные формы приложения построены в основном на компоненте DataGridView. Данный компонент позволяет представлять данные в удобной табличной форме с использованием дополнительных собственных расширений. Например, на нескольких формах использовалось расширение класса DataGridViewColumn - DataGridViewNumericUpDownColumn, что позволило упростить интерфейс и избавиться от множества ошибок пользователей за счет задания нижней и верхней границы чисел, величины приращения. Внешний вид DataGridViewNumericUpDownColumn изображен на рисунке 6.2.

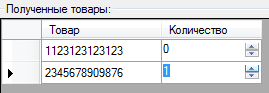


Рисунок 6.2 – Результат использования класса

DataGridViewNumericUpDownColumn

При разработке интерфейса форм главным был принцип предотвращения ошибок, а не их констатации, потому были использованы (по возможности), к примеру, интерактивные обращения к базе еще на этапе заполнения форм (рис. 6.3). Таким образом, частично проверяется уникальность имен, номеров, проходит верификация электронной почты; также для всех полей, в которые должны быть занесены сугубо числовые данные, сделан соответствующий фильтр, что также позволяет предотвратить некорректные данные в базе и пугающие неопытного пользователя сообщения об ошибках.



Рисунок 6.3 – Подсказки пользователю на этапе добавления данных

Также для упрощения восприятия форм пользователем были использованы соответствующие картинки, отображающие тематику данных для этой формы и выполняемое действие, например, добавление или редактирование. В качестве примера можно привести внешний вид диалогового окна редактирования данных работника на рисунке 6.4.

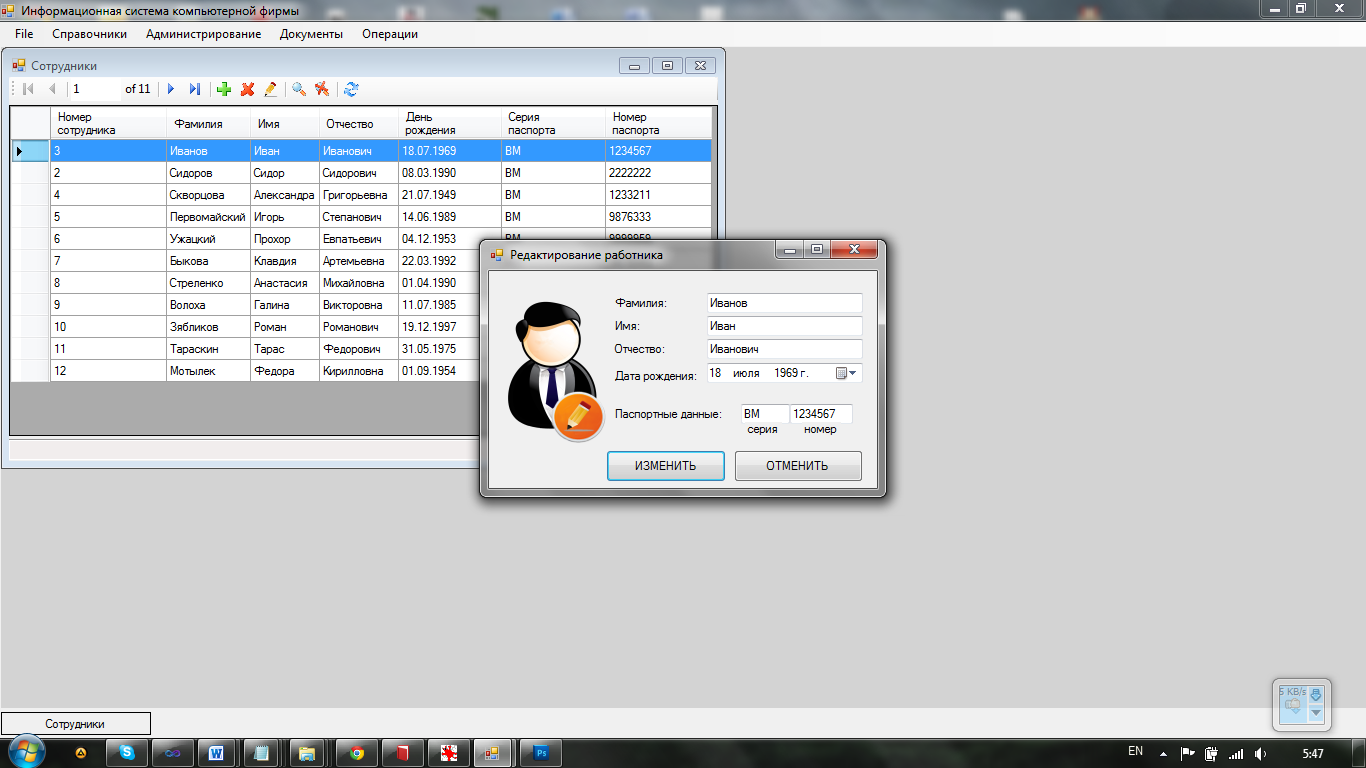


Рисунок 6.4 – Редактирование данных работника

## Построение главного меню и кнопок панели инструментов

Меню реализовано с помощью контейнера MenuStrip, представляющего систему меню для формы. Таким образом, данный класс позволил нам разделить весь функционал по следующим категориям:

1. «Справочники», где происходит просмотр, частичное добавление и редактирование основных базовых таблиц, таких как: помещения, торговые точки, данные сотрудников, КСА и др.
2. «Документы», где происходит добавление и просмотр основных документов, касающихся товарооборота фирмы: накладные, заказы, заявки, чеки, платежные поручения. В данном разделе можно также заполнить заказ (рис. С.2) либо сформировать заявки (рис. С.3)
3. «Операции». Данная категория содержит 2 подкатегории:

* Продажи
* Работа с сотрудниками, в частность, наём и формирование расписания (рис. С.4)

1. «Отчеты». Содержит необходимые отчеты согласно заданию, например, вывод «общего» склада фирмы. Предполагается дальнейшее расширение.
2. «Администрирование». Позволяет добавлять новых пользователей. Предполагается дальнейшее расширение.

## Выполнение программного кода на языке C#

Опишем работу приложения с базой данных. Для подключения к базе будем использовать SqlClient. Строка подключения строится автоматически с помощью класса SqlConnectionStringBuilder во время авторизации пользователя и хранится в классе Session до завершения работы приложения.

SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder();

builder["Server"] = comboBoxServName.Text;

builder.ConnectTimeout = 30;

builder["Trusted\_Connection"] = true;

builder.InitialCatalog = comboBoxBaseName.Text;

success = true;

if (comboBoxAuthenticationType.SelectedIndex == 0)

{

builder.IntegratedSecurity = true;

}

else

{

builder.IntegratedSecurity = false;

builder.UserID = comboBoxUserName.Text;

builder.Password = textBoxPassword.Text;

}

В зависимости от выбранного пользователем режима проверки подлинности: Windows либо SQLServer – строка будет иметь различный формат.

Каждый раз при надобности манипуляций с данными базы происходит подключение посредством объекта SqlConnection. Заполнение таблиц производим с помощью объекта SqlDataAdapter.

using (SqlConnection conn = new SqlConnection(Session.sqlConnString))

{

conn.Open();

using (SqlDataAdapter cmd = new SqlDataAdapter("SELECT IDWarehouse AS [Номер склада],"

+ " BShopFloor AS [Зал магазина], BBase AS База, WarehouseAddress AS [Адрес склада],"

+ " Room AS [Номер помещения] FROM Warehouses", conn))

{

BindingSource bs = new BindingSource();

DataTable dt = new DataTable();

cmd.Fill(dt);

bs.DataSource = dt;

dataGridView1.DataSource = bs;

bindingNavigator1.BindingSource = bs;

dataGridView1.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

}

}

Решением проблемы выполнения серии запросов без ошибок является выполнения запросов в единой транзакции. Демонстрация выполнения серии запросов к базе по добавлению связанных данных приведена в приложении С.

**ЗАКЛЮЧЕНИЕ**

В результате выполненной курсовой работы была создана база данных «Компьютерная фирма» а также программный продукт, позволяющий пользователю взаимодействовать с базой данных. Базы данных была разработана в среде Microsoft SQL Server 2008, приложение – Microsoft Visual Studio 2010 на языке программирования C#.

Приложение позволяет:

* просматривать и редактировать справочные таблицы;
* осуществлять заказ товаров на точки и базы;
* формировать заявки фирмам-поставщикам на основании заказов;
* заполнять накладные по пришедшим товарам;
* вести учет товаров на базах и торговых точках.

В процессе выполнения данной курсовой работы были закреплены навыки в проектировании баз данных и реализации их в СУБД Microsoft SQL Server 2008.

# Список использованных источников

1. http://blog.nguen.net/post33-good\_bad\_plus\_minus\_dot\_net.html - статья «Достоинства и недостатки .Net» Павла Нгуена.
2. http://websideworld.com/main/data-storage/6/2 - статья «Выбор СУБД»
3. Методические указания по курсовому проектированию по дисциплине «Системы управления базами данных» для студентов специальности 1-40 01 01 Программное обеспечение информационных технологий специализации 1-40 01 01 01 Компьютерные системы и ИНТЕРНЕТ технологии/ сост. И. Б. Бураченок, 2009. – 117с.

**ПРИЛОЖЕНИЕ B**

(обязательное)

ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ

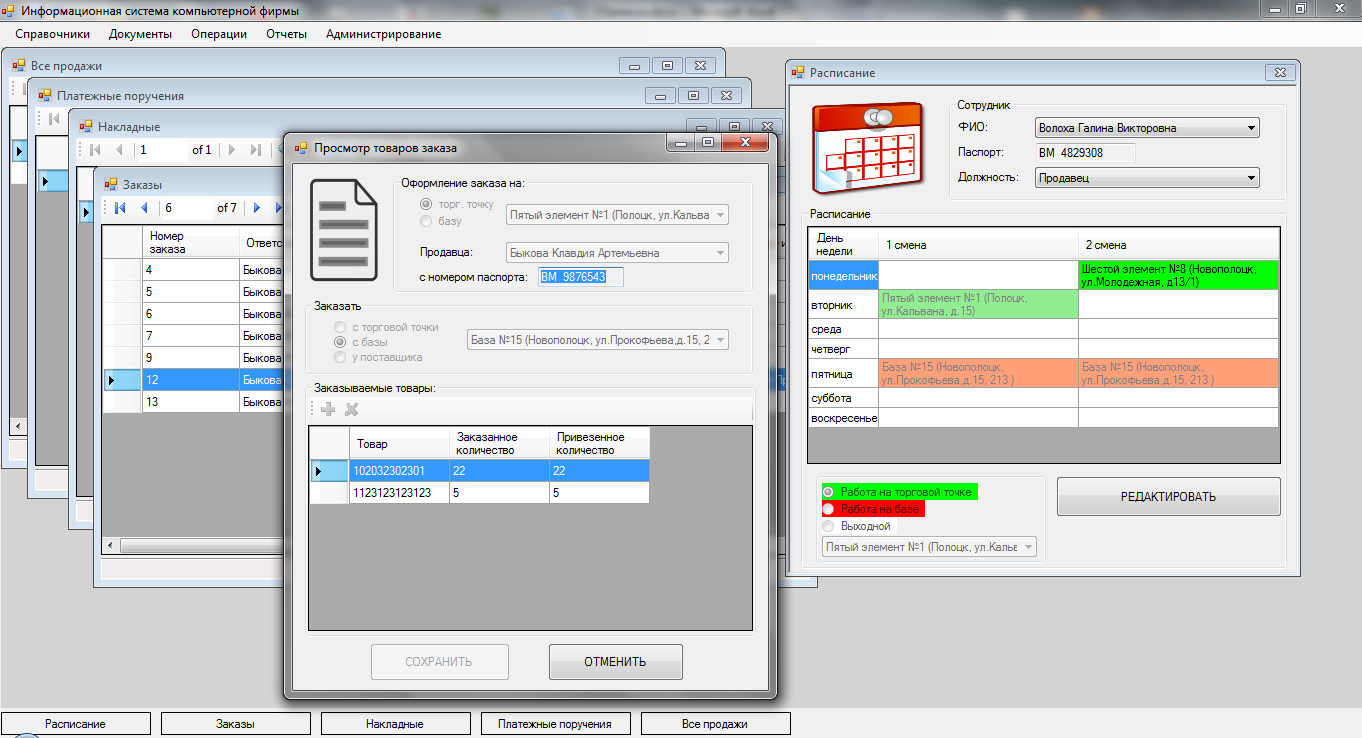


Рисунок B.1 **–** Главная форма приложения

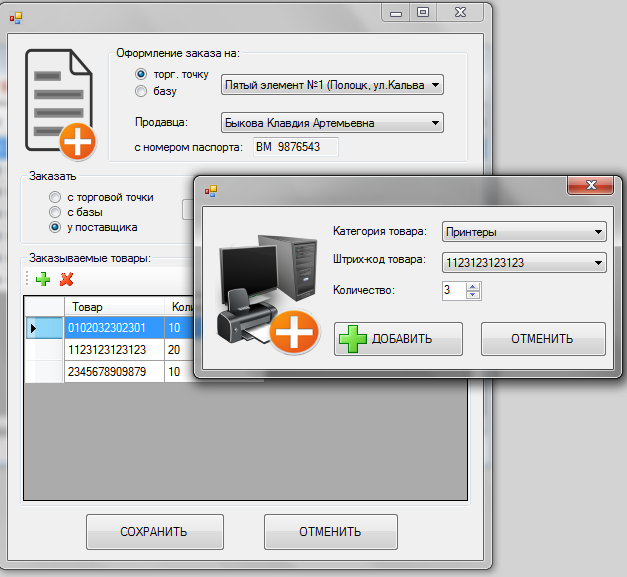


Рисунок B.2 **–** Оформление заказа для поставщика

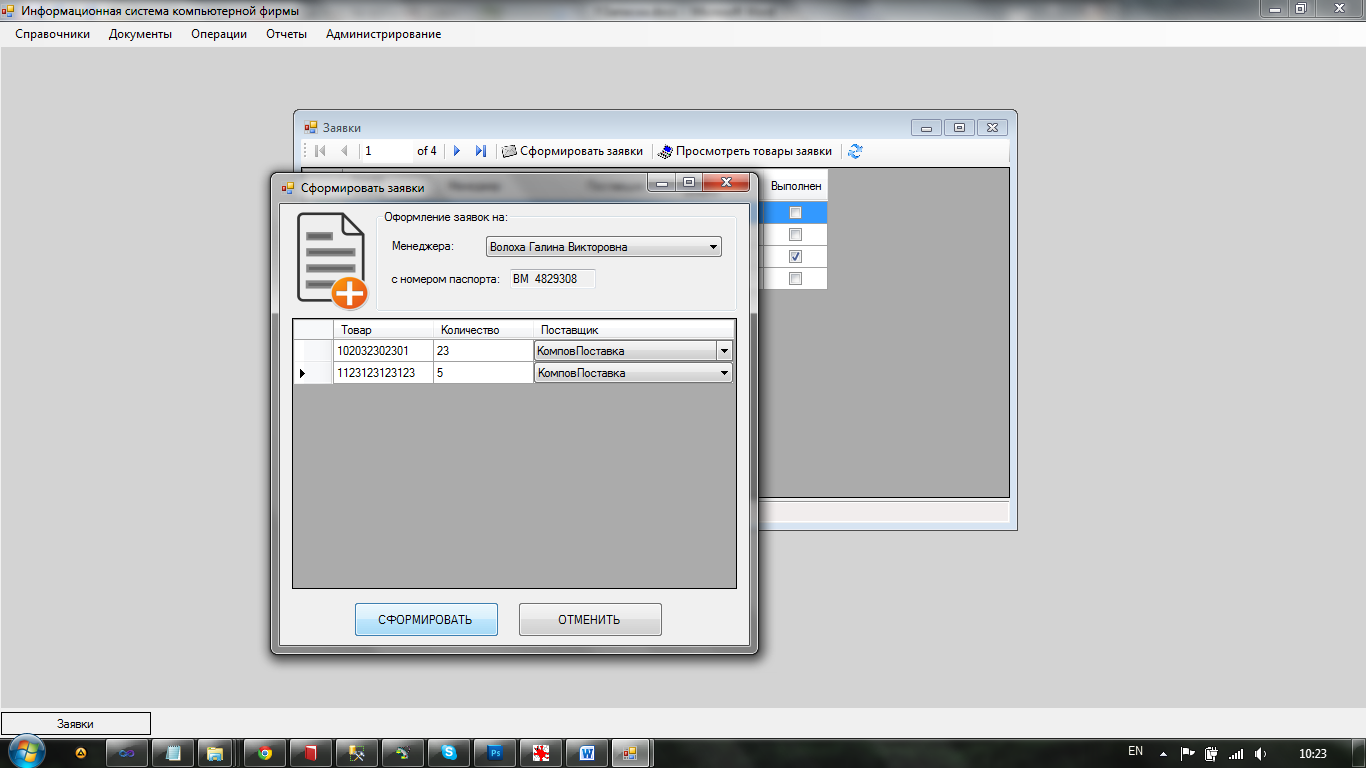


Рисунок B.3 **–** Формирование заявок на товары фирмам-поставщикам

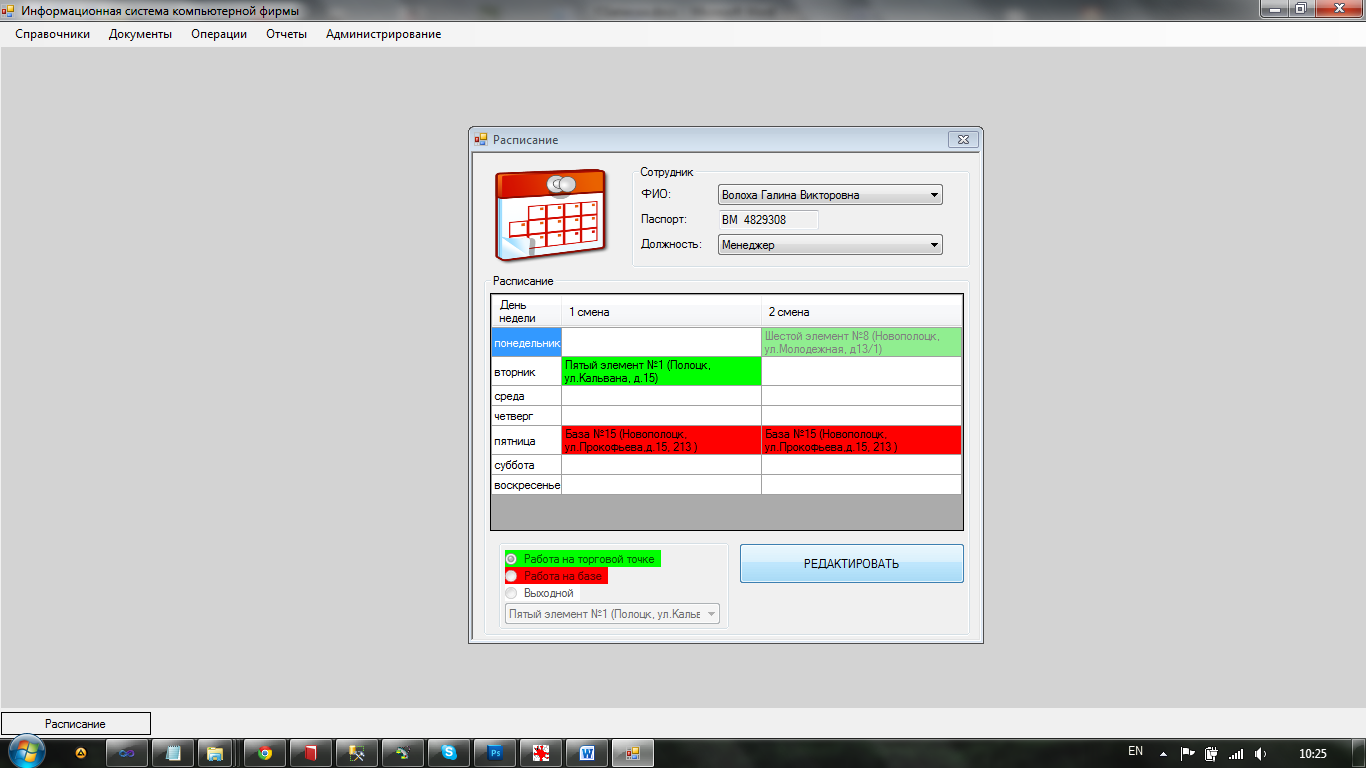


Рисунок B.4 **–** Формирование расписания работника.

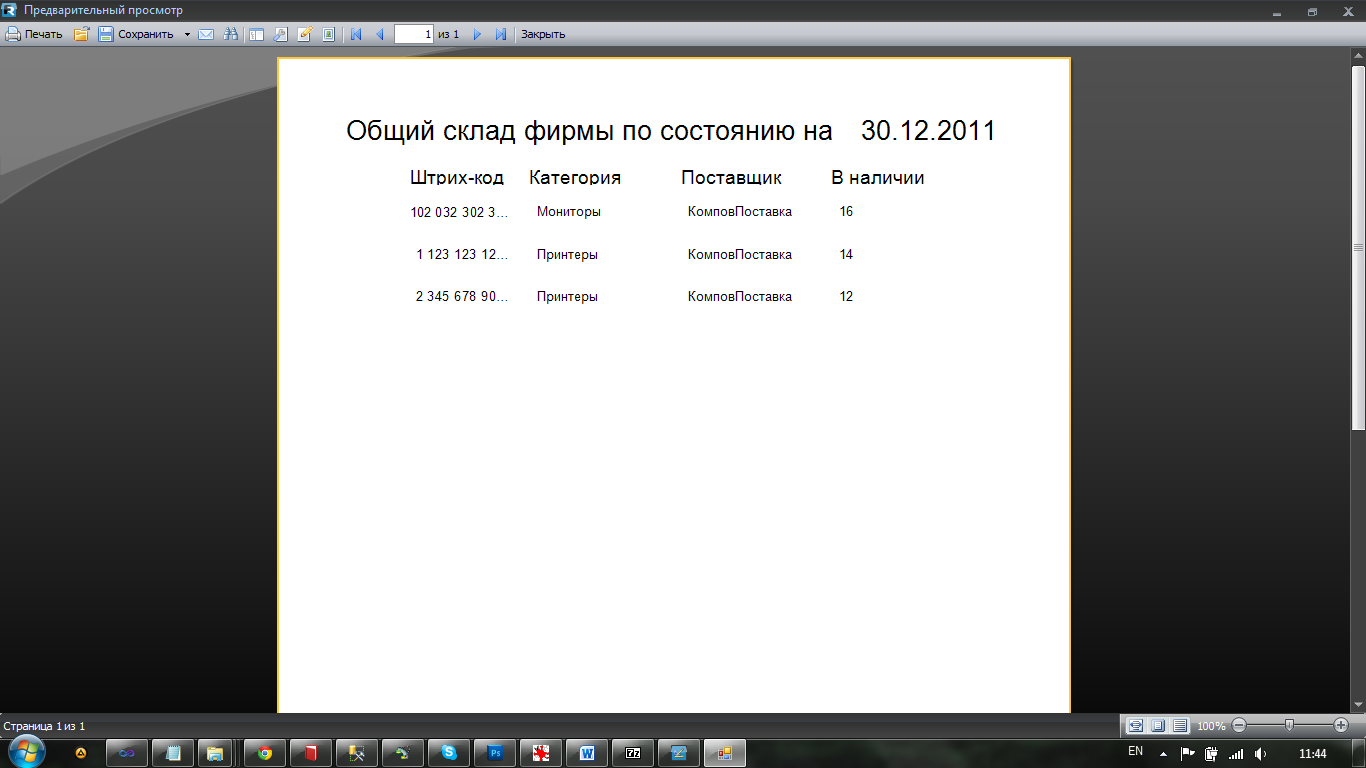


Рисунок B.5 **–** Отчет, отображающий общий склад фирмы

**ПРИЛОЖЕНИЕ C**

(обязательное)

ЛИСТИНГ ПРОГРАММНОГО КОДА

using (SqlConnection conn = new SqlConnection(Session.sqlConnString))

{

conn.Open();

SqlTransaction tran = conn.BeginTransaction();

int index = 0;

bool payment = false;

if (radioButton1.Checked)

{

using (SqlCommand cmd = new SqlCommand("INSERT INTO Receipts VALUES ("

+ comboBox2.SelectedValue.ToString() + "," + comboBox1.SelectedValue.ToString()

+ "," + comboBox3.SelectedValue.ToString() + "," + textBox1.Text + ", @date, "

+ textBox2.Text + ") SELECT scope\_identity()", conn, tran))

{

cmd.Parameters.Add("@date", SqlDbType.Date).Value = dateTimePicker1.Value.Date;

object i = cmd.ExecuteScalar();

index = Convert.ToInt32(i);

cmd.Dispose();

}

}

else

{

payment = true;

using (SqlCommand cmd = new SqlCommand("INSERT INTO PaymentOrders VALUES ("

+ textBox4.Text + ", @date, " + textBox3.Text + ", '" + textBox5.Text

+ "') SELECT scope\_identity()", conn, tran))

{

cmd.Parameters.Add("@date", SqlDbType.Date).Value = dateTimePicker2.Value.Date;

object i = cmd.ExecuteScalar();

index = Convert.ToInt32(i);

cmd.Dispose();

}

}

using (SqlCommand cmd = new SqlCommand("INSERT INTO Selling VALUES("

+ comboBox1.SelectedValue.ToString() + "," + comboBox4.SelectedValue.ToString()

+ "," + (payment ? index.ToString() : "null") + ","

+ (payment ? "null" : index.ToString()) + ", @date, " + textBox6.Text + ")"

+ " SELECT scope\_identity()", conn, tran))

{

cmd.Parameters.Add("@date", SqlDbType.Date).Value = DateTime.Now.Date;

object i = cmd.ExecuteScalar();

index = Convert.ToInt32(i);

cmd.Dispose();

}

foreach (DataGridViewRow dr in dataGridView1.Rows)

{

int supplier = 0;

using (SqlCommand cmd = new SqlCommand("SELECT IDSupplier FROM Suppliers"

+ " WHERE SuplierName = '" + dr.Cells[2].Value.ToString() + "'",conn, tran))

{

object obj = cmd.ExecuteScalar();

supplier = Convert.ToInt32(obj);

}

using (SqlCommand cmd = new SqlCommand("insertInvoiceWares",conn, tran))

{

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@so", SqlDbType.Int).Value = comboBox1.SelectedValue;

cmd.Parameters.Add("@sel", SqlDbType.Int).Value = index;

cmd.Parameters.Add("@ware", SqlDbType.Decimal).Value = dr.Cells[0].Value;

cmd.Parameters.Add("@sup", SqlDbType.Int).Value = supplier;

cmd.Parameters.Add("@q", SqlDbType.Int).Value = dr.Cells[3].Value;

cmd.Parameters.Add("@mu", SqlDbType.Int).Value = dr.Cells[5].Value;

cmd.Parameters.Add("@s", SqlDbType.Bit).Value = (checkBox1.Checked ? 1 : 0);

cmd.ExecuteNonQuery();

}

}

tran.Commit();

}

Листинг C.1 – Осуществление серии связанных запросов по добавлению новой продажи в рамках единой транзакции.

Приложение D

(необязательное)

ДЕТАЛИЗАЦИЯ ПРОЦЕССА ТОВАРООБОРОТА (РИС.1.2) ПРИ ПОМОЩИ DFD (DATA FLOW DIAGRAM)

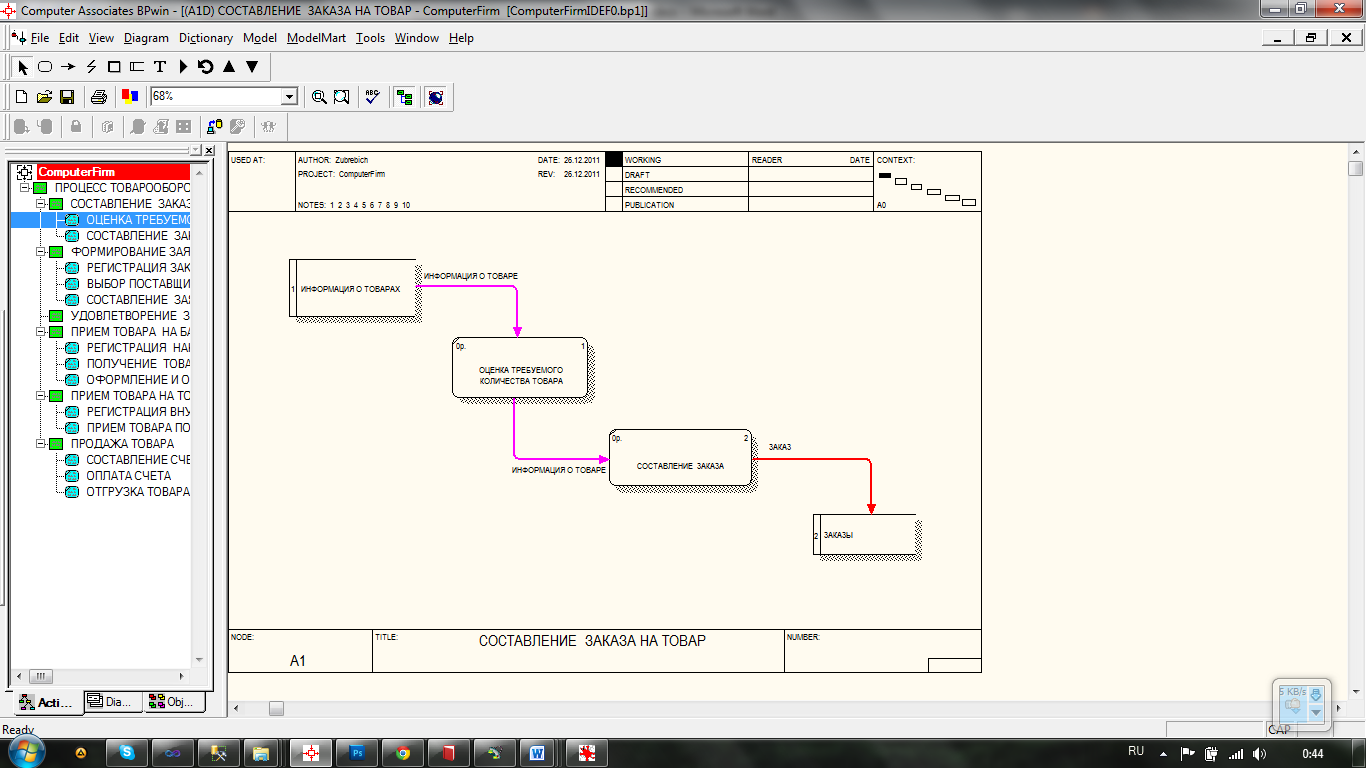


Рисунок E.1 – Детализация блока «Составление заказа на товар» (DFD).

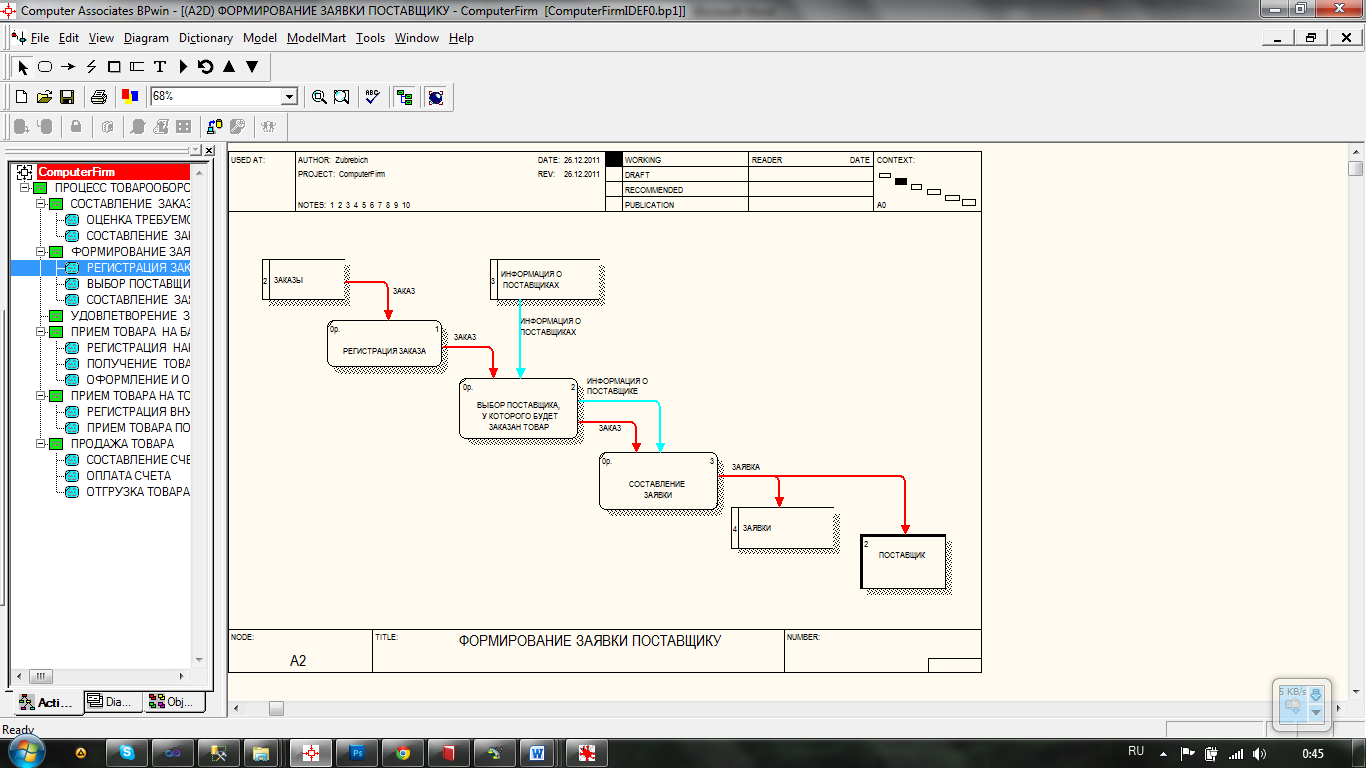


Рисунок E.2 – Детализация блока «Формирование заявки поставщику» (DFD).

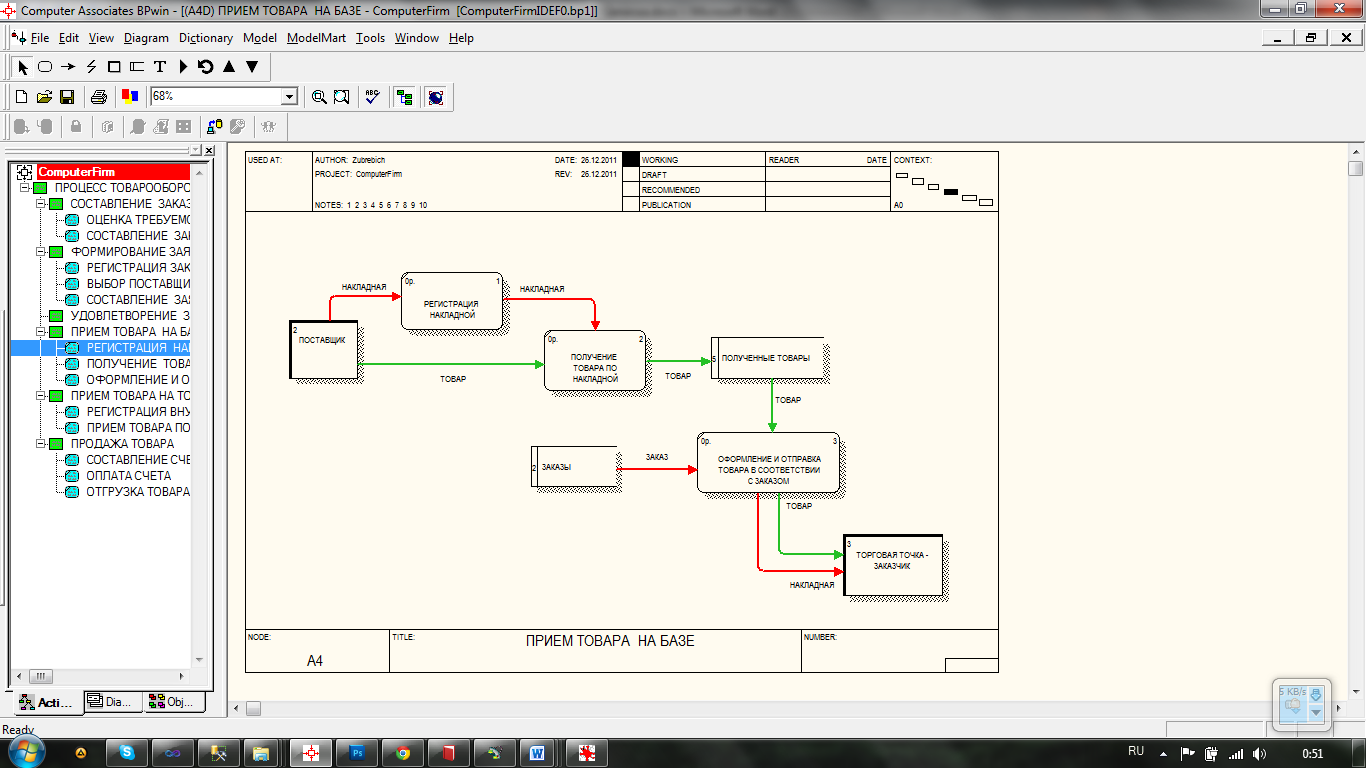


Рисунок E.3 – Детализация блока «Прием товара на базе» (DFD).

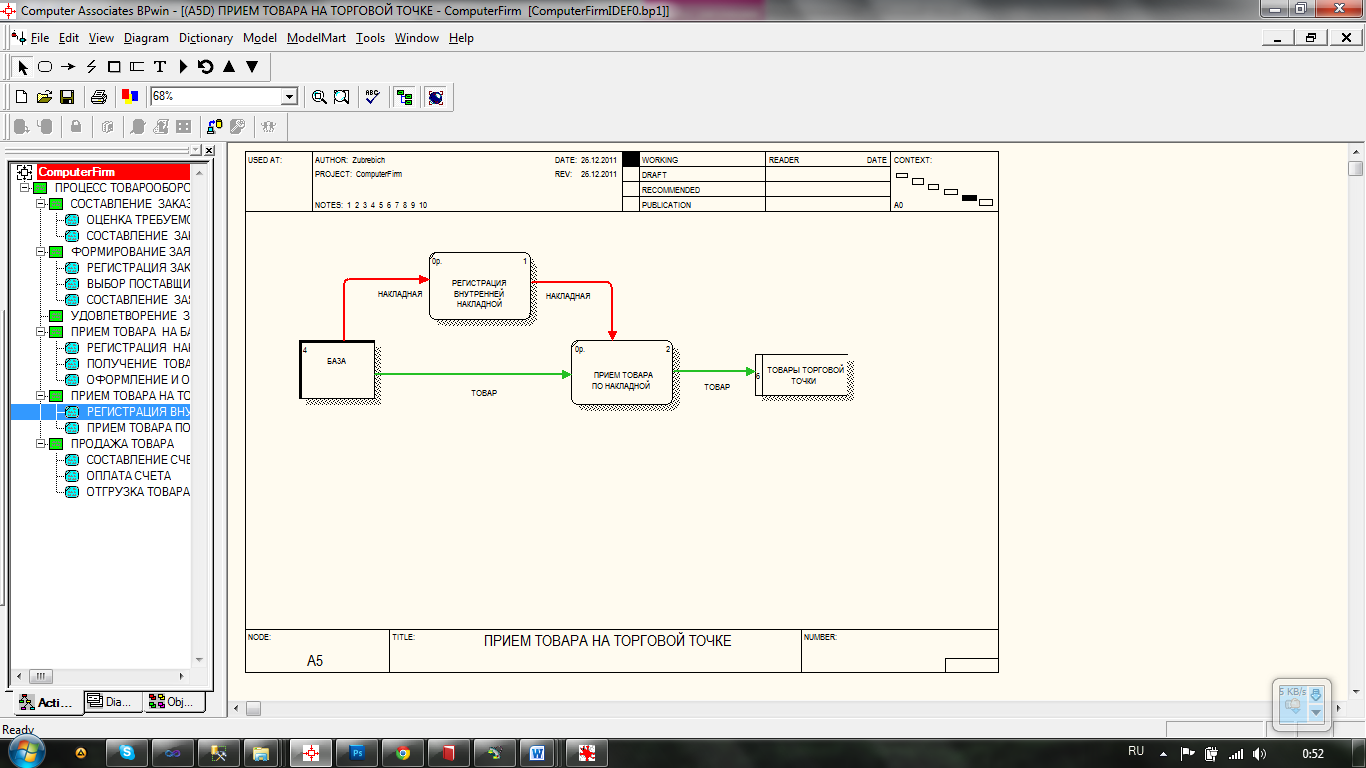


Рисунок E.4 – Детализация блока «Прием товара на торговой точке» (DFD).

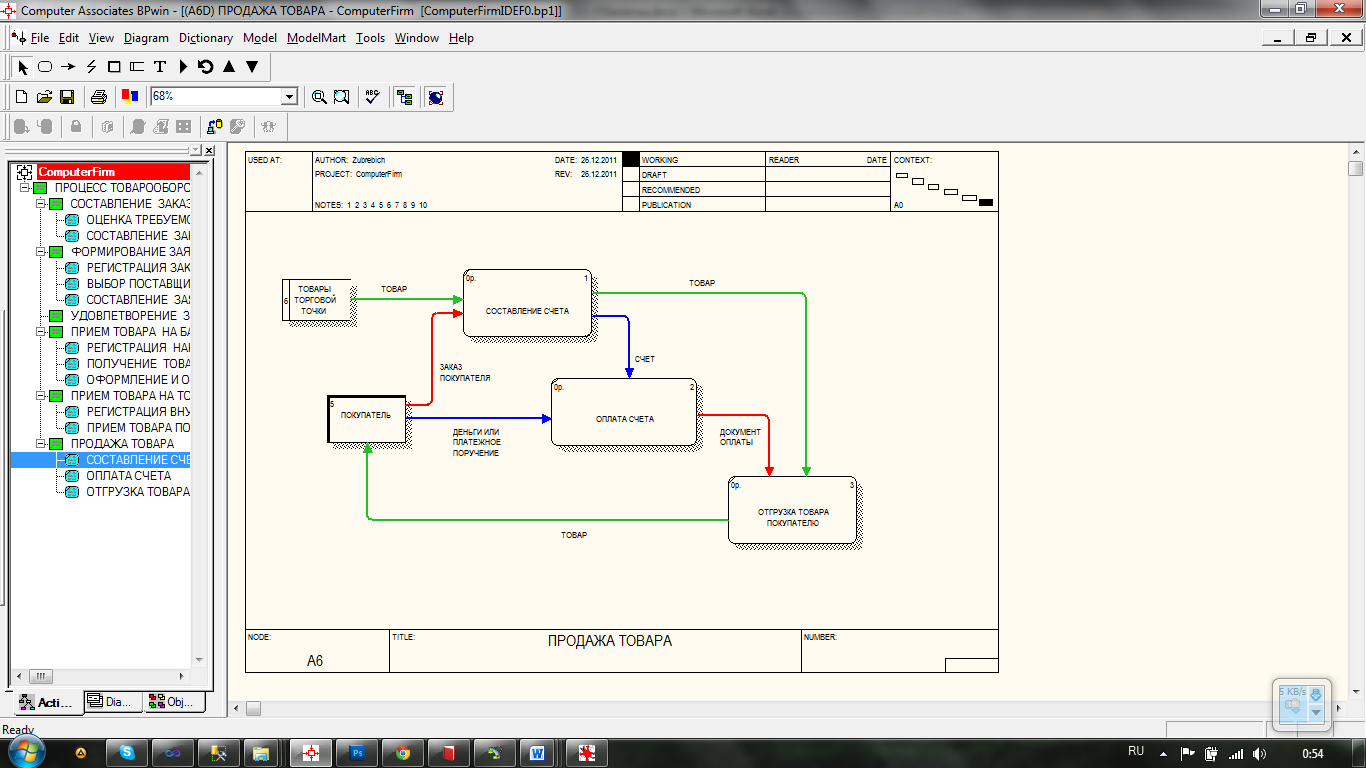


Рисунок E.5 – Детализация блока «Продажа товара» (DFD).