Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Оренбургский государственный университет»

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

С.А.Щелоков

ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПРЕДПРИЯТИЯ

Рекомендовано к изданию Редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Оренбургский государственный университет» в качестве методических указаний для студентов, обучающихся по программам высшего образования по направлениям подготовки 09.04.01 Информатика и вычислительная техника, 09.04.04 Программная инженерия

Оренбург
2015

УДК 004.75:658(076.5)
ББК 32.973.202я7+65.291я7
Щ46

Рецензент – кандидат технических наук, доцент А.С. Боровский

Щелоков, С.А.

Щ46 Проектирование, разработка и реализация распределенной информационной системы предприятия: методические указания к выполнению курсовой работы по дисциплине «Проектирование распределенных информационных систем» / С.А.Щелоков; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2015. – 123 с.

Методические указания к выполнению курсовой работы по дисциплине «Проектирование распределенных информационных систем» предназначены для учебно-методического обеспечения подготовки магистров по образовательной программе уровня высшего образования: по направлению подготовки 09.04.01. Информатика и вычислительная техника по магистерской программе «Информационное и программное обеспечение автоматизированных систем»; по направлению подготовки 09.04.04. Программная инженерия по магистерской программе «Разработка информационно-телекоммуникационных систем».

Дисциплина «Проектирование распределенных информационных систем» входит в состав обязательных дисциплин вариативной части.

В методических указаниях изложены методические основы по современной технологии ADO.NET для разработки распределенных баз данных в СУБД SQL Server в инструментальной среде Visual Studio с использованием языка программирования C#.

В методических указаниях содержатся практические задания на выполнение курсовой работы, требования по оформлению отчетов, варианты предметной области и примеры к выполнению основных этапов курсовой работы по дисциплине «Проектирование распределенных информационных систем».

УДК 004.75:658(076.5)
ББК 32.973.202я7+65.291я7

Содержание

Введение

Методические указания предназначены для выполнения курсовой работы по дисциплине «Проектирование распределенных информационных систем»

Целью курсовой работы является овладение современными методами и средствами технологии проектирования, разработки и реализации проблемно-ориентированных распределенных информационных систем в корпоративной ЛВС предприятия.

Задачи, которые решаются при выполнении работ соответствуют рабочим программам дисциплины: изучение распределенной обработки информации в автоматизированных информационных системах, архитектуры РИС, технологической базы РИС, распределенных информационных ресурсов и сетей, распределенных баз данных, принципов и технологий управления обменом информацией в РИС, методов и средств доступа к удаленным информационным ресурсам.

Выполнение курсовых работ позволяет формировать элементы следующих компетенций в соответствии с ФГОС ВПО и ООП ВПО по данному направлению подготовки:

ОК-1 способность совершенствовать и развивать свой интеллектуальный и общекультурный уровень;

ОПК-1 способность воспринимать математические, естественнонаучные, социально-экономические и профессиональные знания, умением самостоятельно приобретать, развивать и применять их для решения нестандартных задач, в том числе, в новой или незнакомой среде и в междисциплинарном контексте;

ПК-7 применением перспективных методов исследования и решения профессиональных задач на основе знания мировых тенденций развития вычислительной техники и информационных технологий.

1 Порядок выполнения работ при проектировании РИС

Данные методические указания разработаны на основе наиболее востребованных в настоящее время технологий проектирования информационных систем (ИС).

Работы по данным технологиям базируются на основе комплекса российских и международных стандартов.

Наиболее популярными в этом комплексе являются стандарты:

1) **ГОСТ 34:**

ГОСТ 34.601 – 90 «Стадии создания АС»;

ГОСТ 34.602 – 89 «ТЗ на создание АС»;

ГОСТ 34.603 – 92 «Виды испытаний АС»;

РД 50 – 34.698 – 90 «Требования к содержанию документов».

Данный отечественный комплекс стандартов достаточно устарел, но считается полезным с тех позиций, что в нем детально прописано, «как и что делать», и он имеет высокую степень адаптивности.

2) **ISO/IEC 12207:1995 – 08 – 01** «Организация жизненного цикла продуктов программного обеспечения».

Данный стандарт ориентирован на приобретение и эксплуатацию систем, при этом разработка является процессом, логически вытекающим из приобретения.

3) **Oracle CDM** - методология по разработке прикладных информационных систем под заказ.

Данный стандарт ориентирован на создание и поддержку информационных систем.

Дальнейшее развитие стандарта ISO/IEC 12207 реализовано в двух более поздних стандартах **ISO/IEC 15288:2002** «Системная инженерия – Процессы жизненного цикла систем» и **ГОСТ Р ИСО/МЭК 14764 – 2002** «Информационная технология. Сопровождение программных средств».

Полная прежняя обязательность применения только стандартов ГОСТ 34 в настоящее время отменена. Одним из наиболее распространенных способов адаптации стандартов является разработка и применение **профилей стандартов** (ПС). В ПС используется базовые методические основы нескольких стандартов, приемлемые и заказчиком и разработчиком, не приводящие к конфликтным ситуациям.

Традиционной для российских стандартов является «каскадная» («классическая» или «водопадная») модель ЖЦ ИС (рисунок 1). В общем случае считается, что в процессе развития проекта над ним работают различные команды специалистов и при завершении этапа проекта готовится весь необходимый комплект проектной документации, фиксирующий принятые решения.

Рисунок 1 - Каскадная модель ЖЦ ИС

В ГОСТ 34.601—90 «Автоматизированные системы. Стадии создания» определяются стадии и этапы создания ИС (рисунок 2, а).

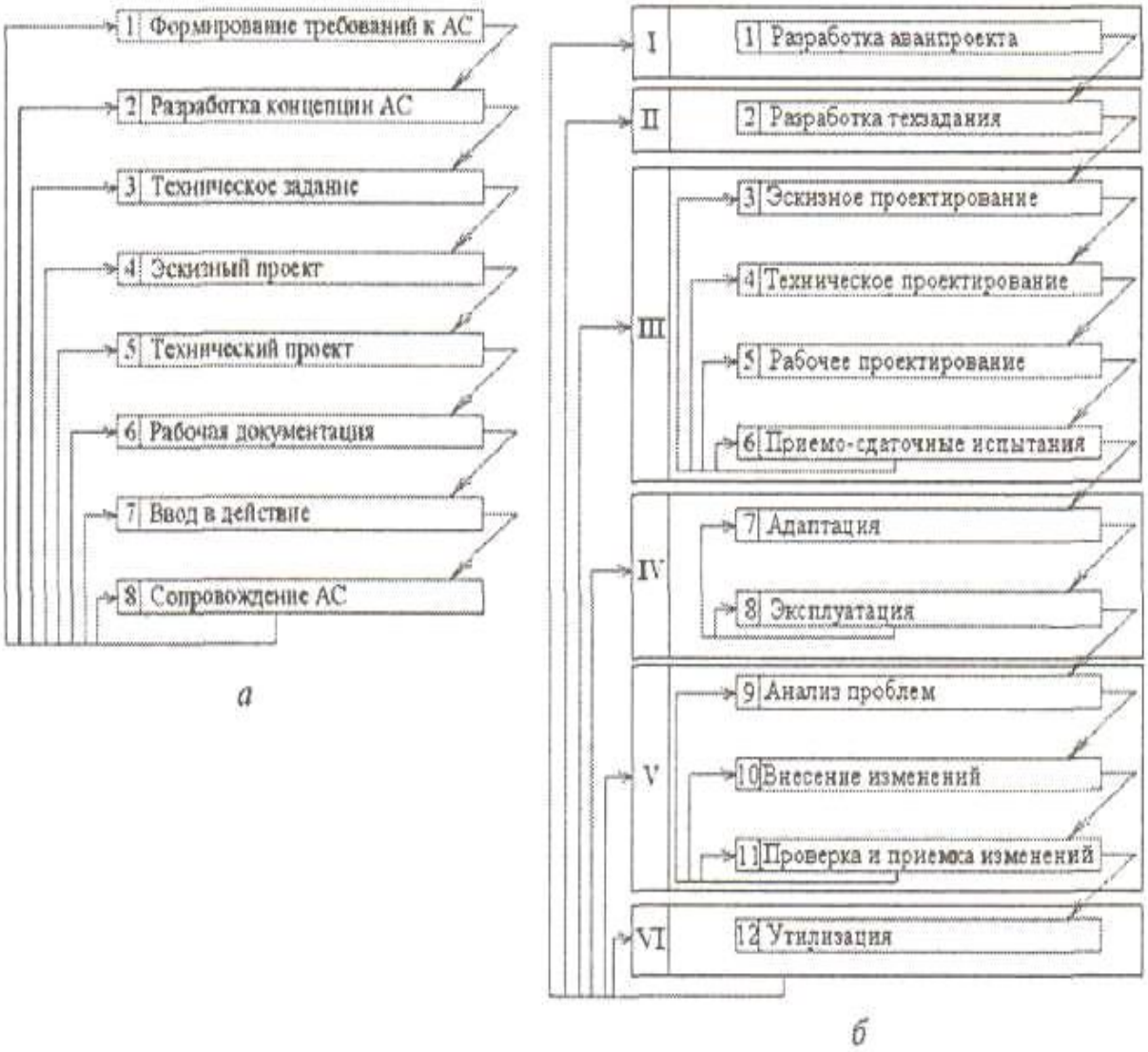


Рисунок 2 - Этапы жизненного цикла ИС: а - ГОСТ 34.601-90; б - ГОСТ Р 53622-2009

1. **Формирование требований**:

обследование объекта и обоснование необходимости создания ИС;

формирование отчета о выполненной работе и заявки на раз работку тактико-технического задания.

2. **Разработка концепции АС**:

изучение объекта (модель «AS IS» («Как есть»));

разработка вариантов концепции ИС и выбор варианта (модели «ТО ВЕ» («Как должно быть»));

оформление отчета о выполненных работах (в частности, технико-экономическое обоснование).

3. **Техническое задание**:

разработка и утверждение ТЗ.

4. **Эскизный проект**:

разработка предварительных проектных решений по системе и ее частям (определяются функции ИС, функции подсистем; состав комплексов задач и интерфейсов; концепция БД, ее укрупненная структура; функции СУБД; состав вычислительной системы; функции и параметры основных аппаратных средств);

разработка документации на ИС.

5. **Технический проект**:

разработка проектных решений по системе и ее частям (разработка общих решений по системе и ее частям, функционально-алгоритмической структуре системы по функциям персонала, спецификаций на программные модули и таблицы БД, организационной структуре, по структуре технических средств, по алгоритмам решаемых задач, применяемым языкам, по организации и ведению информационной базы, по программному обеспечению);

разработка документации на ИС (полное описание совокупности принятых проектных решений и достаточного для дальнейшего выполнения работ по созданию ИС (ГОСТ 34.201—89)).

6**. Рабочая документация**:

разработка программной и эксплуатационной документации на систему и ее части (для обеспечения работы по вводу ИС в эксплуатацию, а также для поддержания уровня эксплуатационных характеристик системы (ГОСТ 34.201—89));

разработка или адаптация (покупных) программ.

7. **Ввод в действие:**

подготовка объекта автоматизации к вводу ИС в действие (организационная подготовка);

подготовка персонала;

строительно-монтажные работы;

пусконаладочные работы (автономная наладка технических и программных средств, загрузка информации в БД; комплексная наладка всех средств системы);

проведение предварительных испытаний;

проведение опытной эксплуатации;

проведение приемочных испытаний.

8. **Сопровождение АС:**

выполнение гарантийных обязательств;

анализ функционирования ИС, ее модификация и выпуск новых версий.

На каждом этапе ЖЦ генерируется определенный набор документов и технических решений. При этом для каждого этапа исходными являются документы и решения, полученные на предыдущем этапе. Каждый этап завершается верификацией созданных документов и решений с целью проверки их соответствия стандартам и ТЗ.

С первого января 2011 г. введен в действие новый стандарт **ГОСТ Р 53622—2009** «Стадии и этапы жизненного цикла, виды и комплектность документов». Он базируется на стандарте ГОСТ 34.601—90, содержит 12 этапов, которые распределены по шести стадиям (рисунок 2, б).

Содержание работ и формируемых документов, выполняемых на этапах стадий I—IV (Разработка аванпроекта, Разработка технического задания, Проектирование, Внедрение), в основном соответствует ГОСТ 34.601—90. Добавляется только этап «Адаптации ИС на конкретные условия применения», на котором производится корректировка конструкторских и эксплуатационных документов.

Стадия V «Сопровождение» содержит три этапа, которые обеспечивают анализ проблем и разработку предложений по изменениям, внесение изменений, проверку и приемку изменений.

Последняя стадия VI «Снятие с эксплуатации» содержит только один этап «Утилизация», на котором формируются протоколы об архивировании программ утилизации аппаратных средств.

Требования к составу, содержанию и порядку согласования и утверждения документов устанавливаются отдельными нормативными документами.

Для решения проблемы взаимного недопонимания системных аналитиков и представителей заказчика на этапе исследования предметной области и последующем проектировании системы применяют структурные методы, чаще всего структурный анализ.

Структурным анализом принято называть исследование, которое начинается с общего обзора системы и затем детализируется, приобретая иерархическую структуру с увеличивающимся числом уровней (на каждом уровне от трех до семи элементов) В основе структурного анализа используются два базовых принципа: принцип «разделяй и властвуй» и принцип иерархического упорядочения.

Первый принцип системного анализа позволяет решение трудных проблем выполнять путем разбиения их на множество более мелких независимых задач, легких для понимания и решения. Второй принцип в дополнение к первому декларирует, что эти независимые задачи должны быть организованы в виде древовидных иерархических структур.

Результаты применения структурного анализа информационных систем наглядно представлены в виде диаграмм с использованием методологий IDEFO, IDEF3 и DFD.

Порядок выполнения работ для первых двух этапов рассмотрим во второй главе.

На этапе «Формирование требований к ИС» проводится обследование предметной области, для которой будет разрабатываться ИС, с последующим обоснованием необходимости ее создания. Построение организационной диаграммы и моде-

ли «AS IS» как в виде IDEFO-модели, так и в виде диаграмм Swim Lane позволяет в удобной форме представить существующие в автоматизируемом предприятии бизнес-процессы.

Результирующим документом этапа формирования требований является технико-экономическое обоснование (ТЭО) (ГОСТ 24.202—80, РД 50-34.698-90).

На этапе «Разработка концепций ИС» создаются несколько моделей «ТО ВЕ», которые отражают необходимые изменения бизнес-процессов в разрабатываемой ИС. Результатом этого этапа является выбор оптимальной модели «ТО ВЕ», определение функциональности ИС на уровне набора программных модулей, построение базы данных логического уровня и дерева меню.

Этап «Техническое задание».

Полученные на предыдущем этапе функциональная модель, база данных логического уровня и пользовательский интерфейс разрабатываемой системы позволяют разработчику и заказчику определить сложность и трудоемкость разработки ИС, ее функциональный состав, структуру информации, условия эксплуатации. Этой информации достаточно для разработки технического задания (ТЗ) на ИС (ГОСТ 34.602—89). ТЗ является основным юридическим документом во взаимоотношениях между разработчиком и заказчиком и определяет цели создания ИС, требования к системе и основные исходные данные, необходимые для ее разработки, а также план-график создания ИС. Обычно ТЗ разрабатывает разработчик, а выдает его разработчику заказчик.

ТЗ содержит титульный лист, на котором помещают подписи заказчика, разработчика и согласующих организаций, которые скрепляют гербовой печатью. Подписи разработчиков ТЗ на ИС и должностных лиц, участвующих в согласовании и рассмотрении проекта ТЗ на ИС, помещают на последнем листе.

При разработке технического задания необходимо решить следующие задачи:

установить общую цель создания ИС, определить состав подсистем и функциональных задач;

разработать и обосновать требования, предъявляемые к подсистемам;

разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств (включая средства связи и передачи данных);

установить общие требования к проектируемой системе;

определить перечень задач создания системы и исполнителей;

определить этапы создания системы и сроки их выполнения;

провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения.

ТЗ на ИС содержит следующие разделы, которые могут быть разделены на подразделы:

1)общие сведения;

2)назначение и цели создания (развития) системы;

3)характеристика объектов автоматизации;

4)требования к системе;

5)состав и содержание работ по созданию системы;

6)порядок контроля и приемки системы;

7)требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8)требования к документированию;

9)источники разработки.

Эти разделы подробно описаны в ГОСТ 34.602—89, но особенно следует обратить внимание на следующие моменты.

В разделе «Требования к системе» в подразделе требования к функциям (задачам), выполняемым системой, следует перечислить все блоки нижнего уровня иерархии «дерева узлов» модели «ТО ВЕ».

В подразделе «Требования к системе в целом» следует описать только те требования, которые имеют смысл для разрабатываемой системы. В обязательном порядке составляются требования к защите информации от несанкционированного доступа, требования по сохранности информации при авариях, отказах технических

средств (в том числе — потери питания) и т. п., при которых должна быть обеспечена сохранность информации в системе.

В подразделе «Требования к видам обеспечения» в зависимости от вида системы приводят требования к математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другим видам обеспечения системы.

Раздел «Состав и содержание работ по созданию (развитию) системы» должен содержать перечень стадий и этапов работ по созданию системы (например, в соответствии с ГОСТ 34.601—90), сроки их выполнения. В данном разделе также приводят:

1)перечень документов по ГОСТ 34.201—89, предъявляемых по окончании соответствующих стадий и этапов работ;

2)вид и порядок проведения экспертизы технической документации (стадия, этап, объем проверяемой документации, организация-эксперт).

В разделе «Порядок контроля и приемки системы» указывают виды (приемочные испытания, опытная эксплуатация и приемочные испытания), состав, объем и методы испытаний системы на основе стандарта ГОСТ 34.603—92. Рекомендуется указать, кто (разработчик или заказчик) разрабатывает программу и методики испытаний.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в опытную эксплуатацию.

Опытную эксплуатацию системы проводят обычно в течение одного месяца на ограниченном количестве рабочих мест, но с полной функциональностью с целью определения фактических значений количественных и качественных характеристик системы и готовности персонала к работе в условиях ее функционирования, а также определения фактической эффективности и корректировки, при необходимости, документации.

Приемочные испытания проводят для определения соответствия системы техническому заданию, оценки качества опытной эксплуатации и решения вопроса о возможности приемки системы в постоянную эксплуатацию.

Следует иметь в виду, что опытная эксплуатация и приемочные испытания проводятся только представителями заказчика.

В разделе «Требования к документированию» приводят согласованный разработчиком и заказчиком системы перечень подлежащих разработке комплектов и видов документов, соответствующих требованиям ГОСТ 34.201-89 и ЕСПД.

В последнее время рекомендуется разрабатывать профиль стандартов на разрабатываемую систему и утверждать его в составе ТЗ.

После утверждения ТЗ разработчик и заказчик заключают контракт на разработку ИС, с указанием этапов выполнения работ, сроков их выполнения и стоимости.

Этап «Технический проект».

Этап технического проекта является самым ответственным и решающим в успешном завершении разработки ИС. Этап выполняется на основе стандарта РД 50-34.698—90.

Технический проект ИС содержит основные проектные решения по системе в целом, ее функциям и всем видам обеспечения ИС, которых должно быть достаточно для разработки программных кодов и рабочей документации. В стандарте ГОСТ 34.201—89 приведены более 20 документов, которые следует разработать на этом этапе (пояснительная записка к техническому проекту, ведомость технического проекта, перечень входных сигналов и данных, перечень выходных сигналов (документов), описание автоматизируемых функций, описание информационного обеспечения системы, описание организации информационной базы, описание комплекса технических средств и др.). Содержание этих документов приведено в стандарте РД 50-34.698—90.

Для упрощения оформления документации для этапа технический проект предлагаются так называемые «Утвержденные спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты» [2]. Эти спецификации требований являются основой для детального планирования процесса разработки программных средств и их компонентов. Под спецификациями требований понимается формальное описание свойств объектов

будущего программного продукта: программных модулей, таблиц БД и элементов пользовательского интерфейса.

Поэтому в упрощенном варианте рекомендуется следующий комплект документов для этапа технического проекта:

пояснительная записка к техническому проекту;

спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты;

описание организации информационной базы;

структурная схема комплекса технических средств.

Содержание пояснительной записки к техническому проекту подробно описано в стандарте РД 50-34.698—90 и не требует дополнительных объяснений.

Она оформляется в виде отдельного документа с титульным листом, содержащим утверждающие и согласующие подписи.

Спецификации требований и алгоритмы на функциональные группы программ, программные и информационные компоненты содержат описание свойств основных компонент ИС: программных модулей, таблиц БД и пользовательского интерфейса.

Спецификации для программного модуля содержат назначение и характеристику каждого программного модуля (постановка задачи, общие требования к входным и выходным данным, описание алгоритма функционирования) и результаты выполнения модуля (выходной документ, экранная форма и т. п.). В качестве программных модулей описываются все функциональные блоки нижнего уровня иерархии модели «ТО ВЕ».

Спецификации для каждого программного модуля выглядят следующим образом:

заголовок модуля - имя модуля, имена и типы формальных параметров, краткое описание назначения модуля и выполняемые им функции;

паспорт модуля - описание всех входных данных, вызываемых модулем; функциональная схема модуля (блок-схема алгоритма или ссылка на реализуемую выходную или экранную формы).

Спецификации для таблиц БД содержат описание каждого поля таблиц (тип и размер поля, его смысловое значение).

Спецификации для пользовательского интерфейса содержат описание всех элементов пользовательского интерфейса (имя элемента, назначение, какой программный модуль запускается, какой результат получают).

Документ «Описание организации информационной базы» содержит разделы:

входная информация;

выходная информация;

логическая структура базы данных;

физическая структура базы данных.

Раздел «Входная информация» должен содержать перечень и описание входных сообщений: наименование, форму представления, сроки и частоту поступления, а также источник информации (документ, видеокадр, устройство, информационная база на машинных носителях и т. д.).

Раздел «Выходная информация» содержит перечень и описание выходных сообщений: наименование, форму представления сообщения (документ, видеокадр, сигнал управления), периодичность выдачи, сроки выдачи и допустимое время задержки решения; получателей и назначение выходной информации.

В разделе «Логическая структура» приводят описание состава данных, их форматов и взаимосвязей между данными (ER-диаграмма).

В разделе «Физическая структура» приводят описание избранного варианта расположения данных в среде конкретного СУБД.

Документ Структурная схема комплекса технических средств содержит схему размещения технических средств (рисунок 3) с краткой аннотацией. Например, в состав комплекса технических средств входят следующие технические средства:

• серверы БД;

• серверы приложений;

• сервер системы формирования отчетности;

• веб-сервер;

• ПК пользователей;

• ПК администраторов.

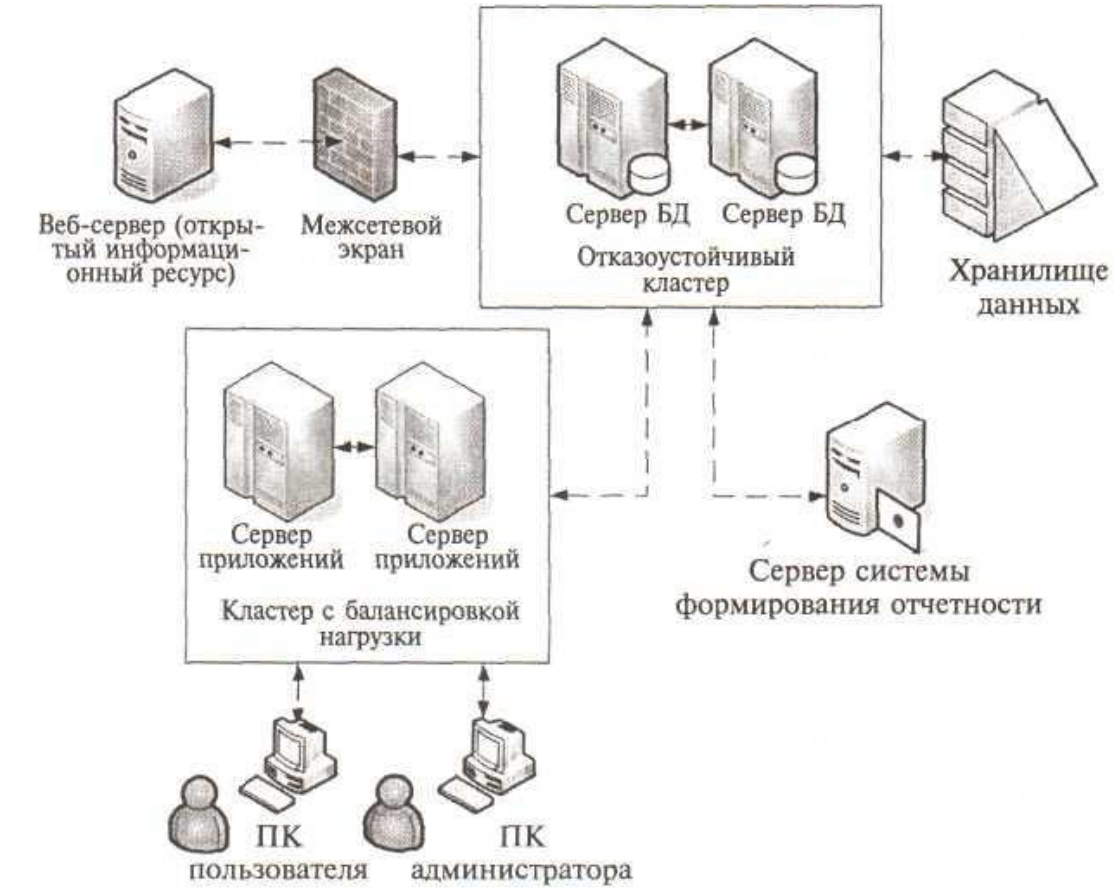


Рисунок 3 - Структурная схема комплекса технических средств

Серверы БД объединены в отказоустойчивый кластер. Связь между серверами БД и хранилищем данных осуществляется по оптическому каналу. Серверы приложений образуют кластер с балансировкой нагрузки. Серверы БД, серверы приложений и сервер системы формирования отчетности объединены одной локальной сетью с пропускной способностью 100 Мбит.

На этапе технического проекта завершается проектирование и начинается разработка ИС с помощью программных инструментариев, предназначенных для этих целей, например с помощью средств быстрой разработки Delphi, Oracle Developer 10g и др.

На этапе «Рабочая документация» каскадной модели ЖЦ разрабатываются программы на основе спецификаций требований для программных модулей и пользовательского интерфейса. В случае приобретения готового программного обеспечения производится его адаптация и интеграция с системой. На всех стадиях создания программных средств осуществляется их тестирование.

На этом же этапе производится разработка программной документации, к которой относятся документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ (руководства операторов, программистов и администраторов, различного рода инструкции). Регламентацию деятельности в процессе подготовки документации определяют стандарты Единой системы программной документации (ЕСПД). В частности, перечень программных документов приводится в стандарте ГОСТ 19.101—77.

Умение создавать программную документацию определяет профессиональный уровень программиста и избавляет разработчиков от необоснованных претензий и большого количества вопросов пользователей.

На этапе «Ввод в действие» выделяются три группы работ: подготовка персонала, пусконаладочные работы и испытания.

Подготовка персонала заключается в обучении различного рода пользователей и обслуживающего персонала по специально созданным учебным материалам обычно с участием разработчиков и в проверке их способности обеспечить функционирование ИС.

Пусконаладочные работы включают автономную наладку технических и программных средств, загрузку информации в базу данных и комплексную наладку всех средств системы.

Испытания системы в полном объеме содержат три стадии, подробно описанные в ГОСТ 34.603—92: предварительные испытания, опытная эксплуатация и приемочные испытания. Общим для них является создание комиссий для проведения испытаний, программы, методик и протоколов испытаний, а также акта об их результатах.

Предварительные испытания следует выполнять после проведения разработчиком отладки и тестирования поставляемых программных и аппаратных средств системы и представления им соответствующих документов об их готовности к испытаниям, а также после ознакомления персонала ИС с эксплуатационной документацией.

Во время предварительных испытаний осуществляют проверку ИС на работоспособность и соответствие техническому заданию в соответствии с программой и методиками и предварительными испытаниями. По результатам предварительных испытаний производят устранение неисправностей в системе и внесение изменений в документацию техническую и эксплуатационную. В акте о результатах проведенных предварительных испытаниях указываются действия по устранению обнаруженных недостатков, после выполнения которых, возможен переход к проведению опытной эксплуатации.

Опытная эксплуатация заключается в работе с реальными данными без участия разработчиков по программе и методикам испытаний с регистрацией всех ошибок, сбоев и нештатных ситуаций. По итогам опытной эксплуатации производится доработка программного обеспечения и дополнительная наладка технических средств. Работы завершаются оформлением акта о результатах опытной эксплуатации.

Проведение приемочных испытаний заключается в комплексной проверке реально функционирующей в полном объеме ИС на соответствие техническому заданию. В случае положительных результатов испытаний оформляется акт о приемке ИС в постоянную эксплуатацию. Этот акт является подтверждением того факта, что разработчик полностью реализовал все положения ТЗ.

Для каждого вида испытания издается приказ руководителя со стороны заказчика о составе комиссии и сроках проведения испытаний, разрабатываются программа и методики испытаний. В процессе проведения испытаний оформляются журналы испытаний, на их основе составляется протокол испытаний (по каждой методике испытаний), который подписывается членами комиссии по проведению испытаний.

На основании протоколов испытаний составляется Акт результатов испытаний, в котором указываются обнаруженные в результате испытаний недостатки и ошибки, а также условия проведения последующих испытаний (опытной эксплуатации, приемочных испытаний). После подписания Акта о завершении приемосдаточных испытаний и результатах выполнения контракта на разработку ИС все обязательства разработчика перед заказчиком считаются выполненными.

На этапе сопровождения ИС анализируется функционирование ИС, выявляются отклонения эксплуатационных характеристик, устраняются причины этих отклонений, своевременно выявляются ошибки и нештатные ситуации.

Кроме этого, этап включает процессы и операции, связанные с регистрацией, диагностикой и локализацией ошибок, внесением изменений и тестированием, проведением доработок, тиражированием и распространением новых версий ИС в места ее эксплуатации, что фактически является повторной итерацией почти всех этапов модели ЖЦ. На каждый вид работы оформляется определенная документация: отчет пользователей о выявленных дефектах и предложениях по корректировке версий ИС; журнал выявленных дефектов и предложений по совершенствованию версий ИС; журнал подготовленных и утвержденных корректировок и др.

2 Обозначение профиля проектирования РИС

Прежде чем перейти непосредственно к проектированию системы, следует отметить, что проектировщик ознакомлен с исходными данными предметной области и требованиями основных регламентирующих документов.

Во-первых, был определен состав информационной системы, необходимый для реализации CALS-технологий. РИС будет проектироваться для предприятия, например, для компьютерного интегрированного полиграфического производства в локальной корпоративной вычислительной сети (рисунок 4).

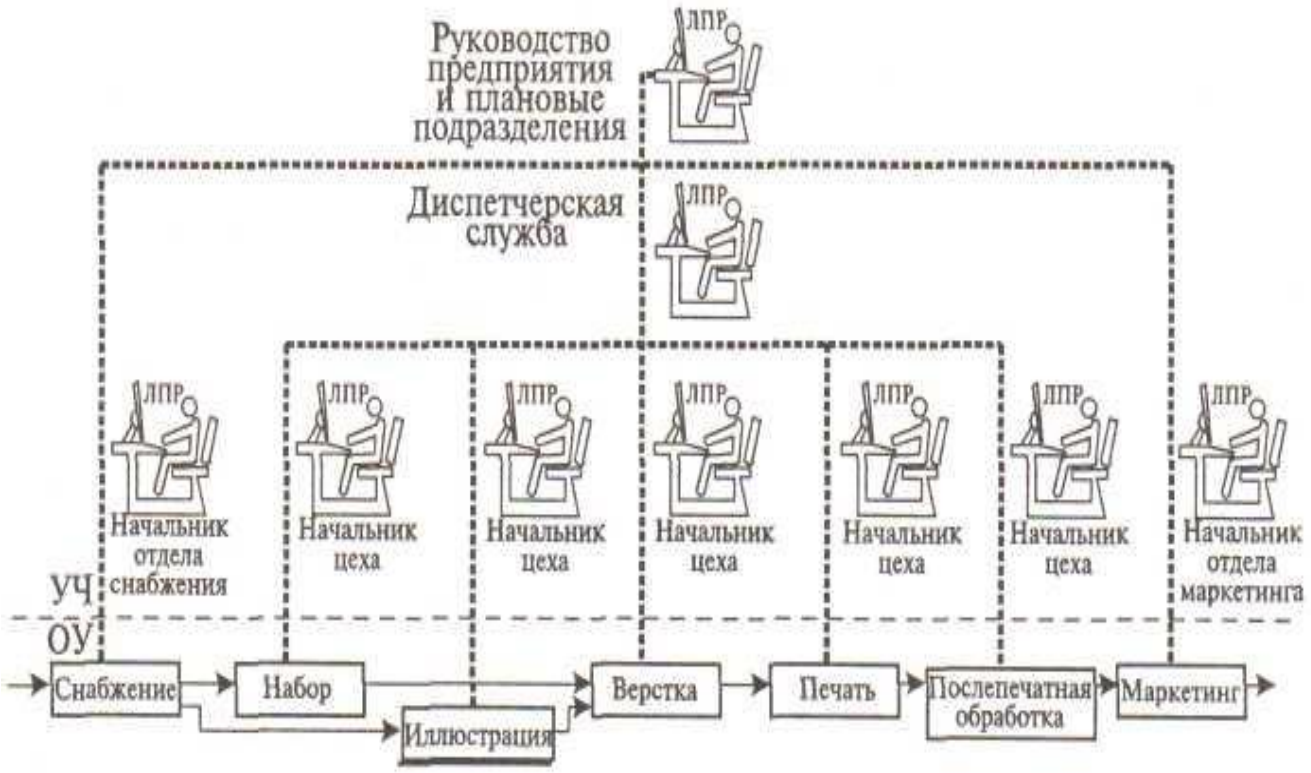


Рисунок 4 - Схема компьютерного интегрированного полиграфического производства: УЧ — управляющая часть; ОУ — объект управления

Во-вторых, проектировщик познакомился с технологиями проектирования баз данных в СУБД SQL Server, с программированием на языке высокого уровня C# в программной среде Visual Studio по технологии DOT.NET.

В-третьих, проектировщик познакомился с современными программными инструментариями CASE-технологий и их правильным выбором для проектирования

конкретной ИС. Особенно тщательно были изучены методологии инструментариев BPwin и ERwin, входящих в пакет All Fusion Modeling Suite, которые чрезвычайно популярны в учебном процессе и в проектировании ИС.

И, в-четвертых, проектировщик изучил наиболее востребованные в настоящее время стандарты (ГОСТ 34, ISO/IEC 12207:1995, ISO/IEC 15288:2002, Oracle CDM и др.), на которых базируются технологии проектирования ИС. Кроме того, проектировщик должен определить профиль стандартов для разрабатываемых систем.

Для обозначения профиля проектирования РИС определим основные компоненты в проектировании ИС, охватывающие четыре основные области:

проектирование процессов;

проектирование объектов данных;

проектирование программ, экранных форм, отчетов;

разработка архитектуры ИС.

Главная цель проектирования процессов заключается в определении функциональности ИС в результате построения функциональной иерархической модели. Для этого с помощью графических моделей переходят от текстового описания деятельности (содержащегося в нормативных документах организации, таких как положения о подразделениях, должностные инструкции, технологические карты производственных процессов) к полному формализованному графическому описанию.

Описание деятельности организации исключительно трудоемкая работа, которая обычно осуществляется постепенно — с описания наиболее значимых процессов верхнего уровня с их последующей детализацией. При этом корректным считается такой подход к моделированию, когда диаграмма любого уровня (кроме верхнего) является детализацией объекта какой-либо диаграммы предыдущего уровня.

Такой подход, в частности, поддерживается инструментом AllFusion Process Modeler (BPwin) компании Computer Associates. Детализация (декомпозиция) — это условный прием, позволяющий представить систему в виде, удобном для восприятия и анализа как для разработчиков, так и для представителей заказчика.

Функции, полученные на нижнем уровне иерархии функциональной модели, представляют собой модули информационной системы, которые определяют интер-

фейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы. Конечным продуктом проектирования процессов является набор спецификаций модулей системы.

Проектирование объектов данных заключается в формировании базы данных логического и физического уровней. Проектировщики в качестве исходной информации получают результаты анализа документооборота в построенной функциональной модели. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных. Конечным продуктом проектирования объектов данных является схема базы данных.

При этом проектирование процессов продолжается параллельно с проектированием схемы базы данных, поскольку часть бизнес-логики (программных модулей) обычно реализуется в базе данных (ограничения, триггеры, хранимые процедуры).

Проектирование программ, экранных форм, отчетов производится на основе спецификаций (описания) всех модулей ИС, таблиц базы данных и элементов пользовательского интерфейса.

Разработка архитектуры ИС включает в себя выбор конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т. п.

В следующей главе рассмотрим технологию проектирования процессов обработки информации на предприятии в среде инструментариев BPwin на базе российских стандартов ГОСТ 34 для первых двух этапов каскадной модели: формирование требований к ИС и разработка концепций ИС.

3 Технология проектирования процессов обработки информации

3.1 Анализ данных предметной области

Целью анализа данных предметной области является установление состояния предприятия в процессе обработки информации «Как есть (AS IS)». Для достижения данной цели изучается организационно-штатная структура предприятия и уясняется регламент существующих информационных процессов. Затем, используя пакет All Fusion Modeling Suite, производится разработка организационной диаграммы и Swim Lane Diagram.

Организационные диаграммы (Organization charts) позволяют описать структуру предприятия и создаются на основе предварительно собранной информации о сотрудниках предприятия. Благодаря организационным диаграммам можно отобразить как структуру предприятия, так и любую другую иерархическую структуру (рисунок 5).

Рисунок 5 - Организационная диаграмма отделения больницы

Для построения этих диаграмм необходимо сформировать четыре словаря: словарь изображений, словарь ресурсов, словарь ролей и словарь групп ролей.

Словарь изображений содержит условные обозначения различных ролей для улучшения внешнего вида диаграмм. Изображения импортируются в словарь, для чего следует выполнить команду Dictionary/Bitmaps, в появившемся диалоговом окне Bitmap Dictionary щелкнуть по кнопке Import и найти файл формата .bmp.

В словаре групп ролей (Role Group Dictionary) создаются и определяются свойства групп ролей (необходимо выполнить команду Dictionary/Role Group) (рисунок 6). В качестве значения группы ролей может быть название предприятия, отдела, цеха и т. д., но удобнее в качестве группы ролей представлять различные виды деятельности: производственной, конструкторской, технологической, финансовой, бухгалтерской, снабженческой и т. п.

Формирование групп ролей ведется системными аналитиками на основе выявленных ролей, которые заносятся в словарь ролей (Role Dictionary) по команде Dictionary/Role (рисунок 7). Ролью может быть должность или профессия конкретного исполнителя.

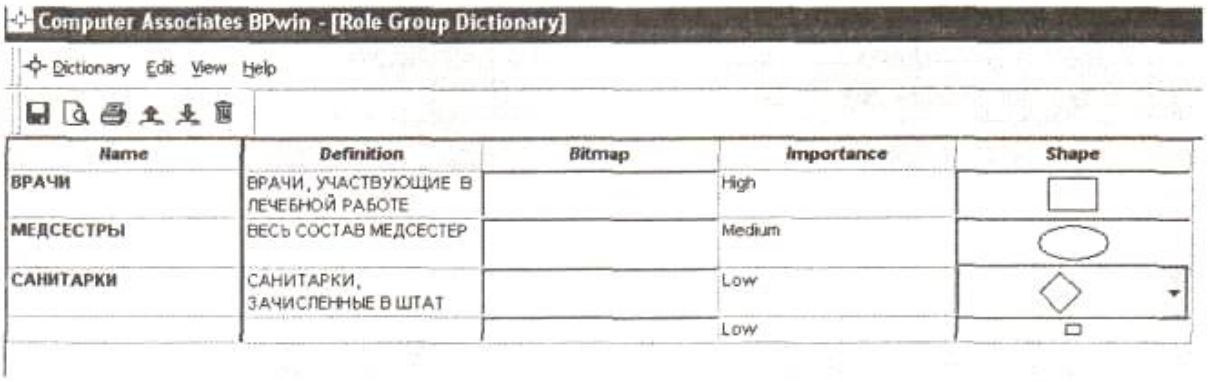


Рисунок 6 - Окно для создания словаря групп ролей

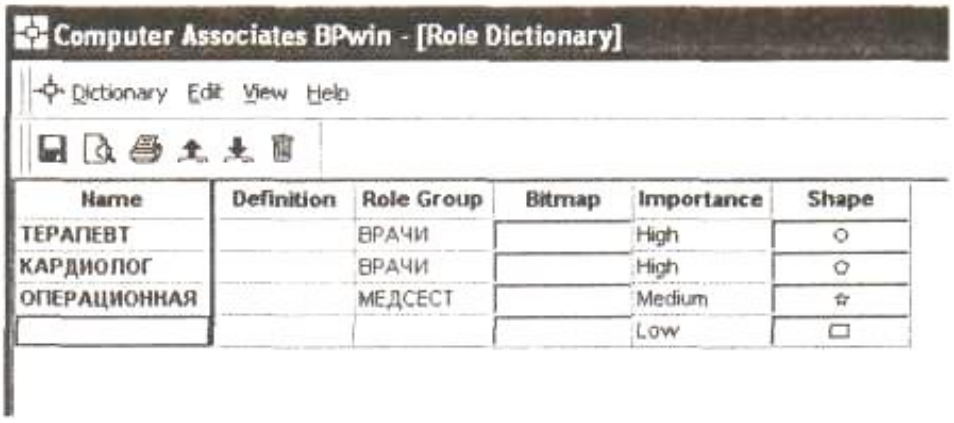


Рисунок 7 - Окно для создания словаря ролей

Каждой роли может соответствовать одна или несколько групп ролей, которые заносятся в словарь ролей. Кроме этого, в словарь ролей для каждой роли заносится определение (Definition), изображение для данной роли (Bitmap) и геометрическая фигура (Shape), а также указывается важность роли (Importance).

Затем формируется словарь ресурсов (Resource Dictionary) по команде Dictionary/Resource. Ресурсом для роли является конкретный исполнитель (фамилия, имя, отчество), который связывается с конкретной комбинацией «группа ролей/роль» (рисунок 8). То есть в словарь ресурсов заносятся имя исполнителя, описание исполнителя и из списка выбираются его группа ролей и роль.

В результате этой работы системные аналитики получают точное представление о структуре и составе сотрудников предприятии.

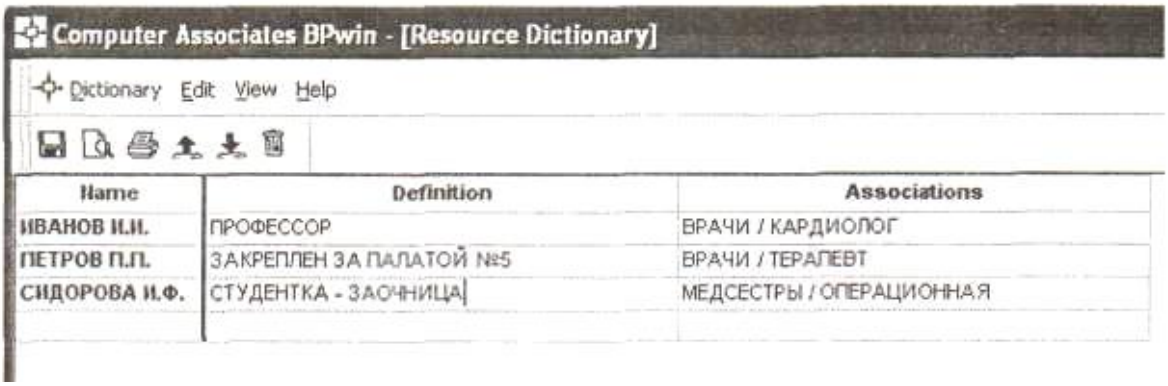


Рисунок 8 - Окно для создания словаря ресурсов

Следующим шагом является построение самой организационной диаграммы, которая не только представит в графическом виде все группы ролей и соответствующие им роли, но и упорядочит их в виде иерархической структуры.

Создание диаграммы выполняется с помощью мастера построения (Organization Chart Wizard), который запускается командой Diagram/ Add Organization Chart (рисунок 9).

На первом шаге вносятся имя диаграммы, имя автора диаграммы, группа ролей и роль для верхнего уровня иерархии диаграммы.

На втором шаге создается второй уровень иерархии с помощью диалогового окна, которое в верхнем окне содержит все доступные роли и ресурсы. Из этого списка переносятся в нижний список с помощью кнопки Add роли и ресурсы второго уровня.

Третий шаг предназначен для изменения свойств организационной диаграммы: в группе drawing можно выбрать ту информацию, которая будет отображаться на блоках диаграммы (наименование блока, имя группы ролей, роль, ресурс). После щелчка по иконе Готово (Finish) создается два уровня организационной диаграммы.

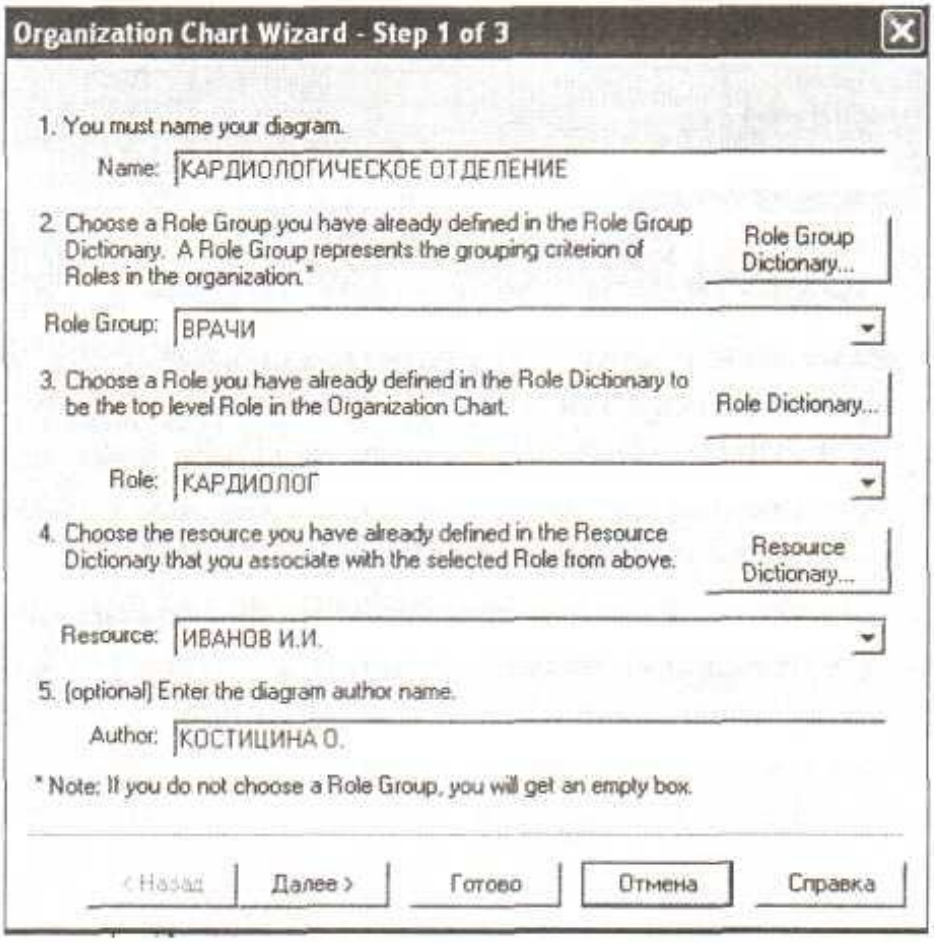


Рисунок 9 - Окно мастера построения организационной диаграммы

Для построения оставшихся уровней иерархического дерева используется контекстное меню, которое вызывается щелчком правой клавиши мыши по соответствующему блоку диаграммы. Оно содержит следующие элементы: edit subroutine

list (редактирование блока), add subordinates (добавить нижний уровень), add sibling on left (добавить блок на текущий уровень слева от редактируемого блока), add sibling on right (добавить блок на текущий уровень справа от редактируемого блока).

Информация для заполнения блоков вызывается из словарей, для чего предварительно в контекстном меню необходимо щелкнуть по строке Role name.

На основе полученной организационной диаграммы и сформированных словарей можно приступить к графическому описанию основных сценариев предметной области с помощью диаграмм плавательных дорожек (Swim Lane Diagram).

В последних версиях пакета появились диаграммы нового типа - Swim Lane, так называемые диаграммы плавательных дорожек. Swim Lane иллюстрируют несколько параллельных потоков, что позволяет отобразить несколько взаимосвязанных процессов в виде плавательных дорожек на одной диаграмме (рисунок 10). Кроме того, на диаграммах Swim Lane можно указать роли исполнителей работ, тем самым более качественно формализовать роли и ответственность.

Эти диаграммы представляют собой горизонтальные полосы (плавательные дорожки) для каждой роли. На этих плавательных дорожках с помощью графических объектов нотации IDEF3 (единиц работы, связей и перекрестков) отражается участие той или иной роли в рассматриваемом сценарии (см. рис. 4.1). Диаграмма может быть построена для одной или нескольких групп ролей, при этом из данной группы выбираются те роли, для которых необходимо сформировать плавательные дорожки. Для создания диаграммы Swim Lane следует выполнить команду Diagram/Add Swim Lane Diagram, которая запускает мастер построения плавательной диаграммы (Swim Lane Diagram Wizard) (рисунок 11).

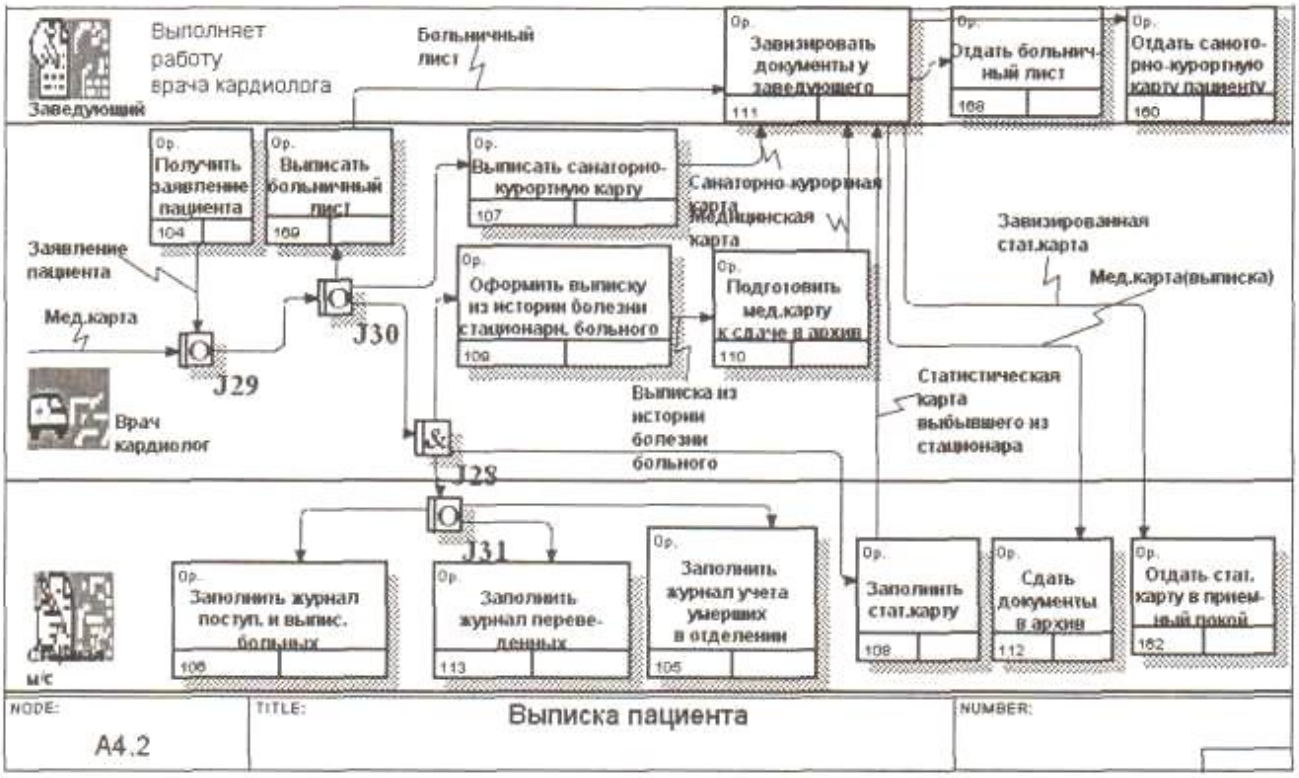


Рисунок 10 - Диаграмма Swim Lane

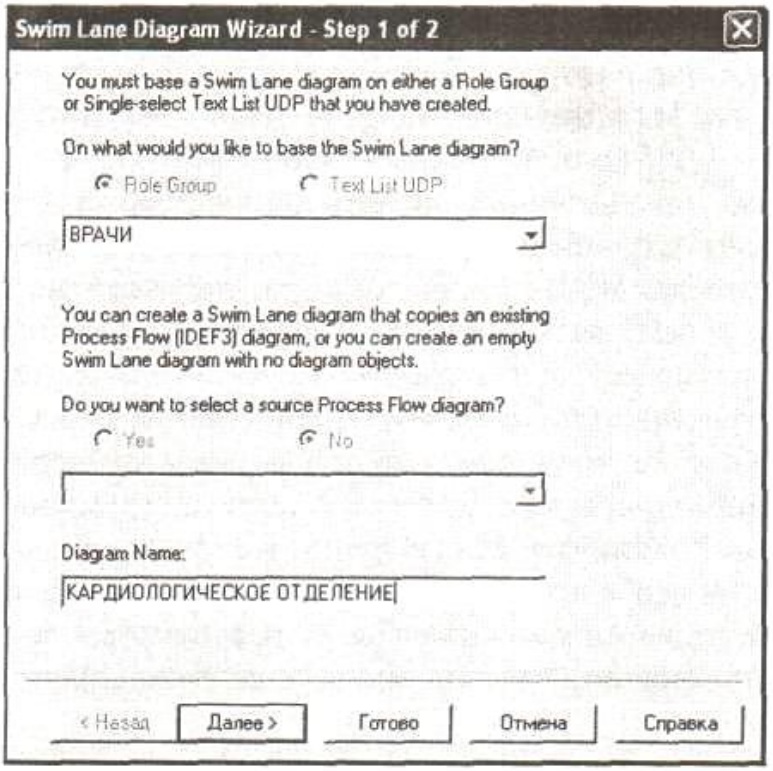


Рисунок 11 - Окно мастера построения диаграммы Swim Lane (1-й шаг)

На первом шаге следует внести название диаграммы, имя автора и группу ролей, из которой можно будет выбрать роли, связанные с диаграммой.

На втором шаге следует выбрать роли, на основе которых будет создана диаграмма (рисунок 12).

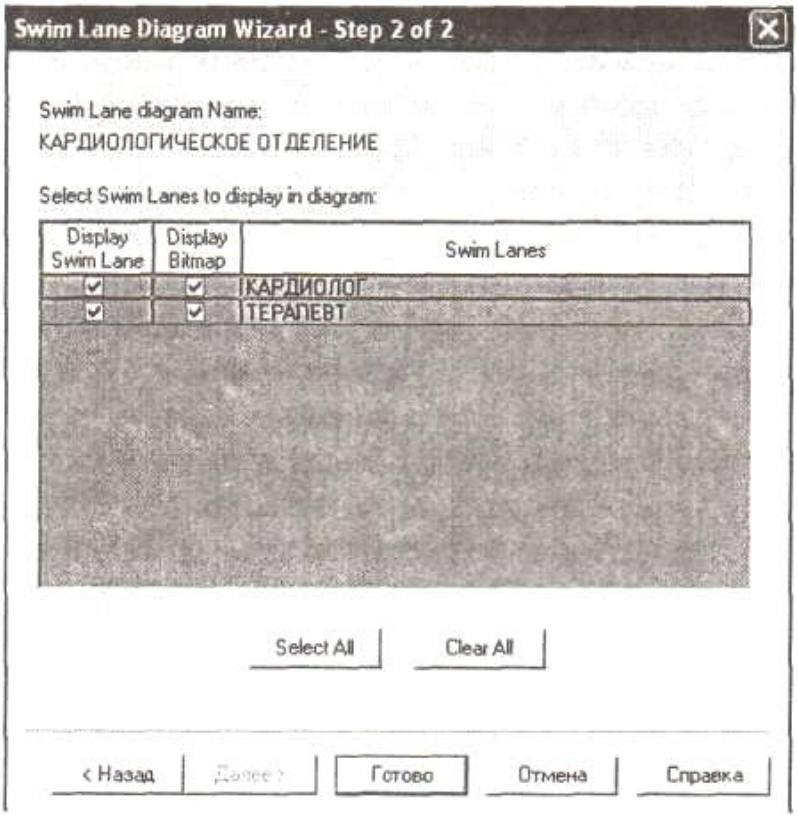


Рисунок 12 -Окно мастера построения диаграммы Swim Lane (2-й шаг)

В диаграмме будет столько полос (плавательных дорожек), сколько ролей будет отмечено «галочкой» в колонке Display Swim Lane.

После щелчка по иконе Готово создается новая диаграмма. Если она создавалась на основе IDEF3-диаграммы, то все объекты этой диаграммы будут располагаться по полосам произвольно и их надо разместить по ним в соответствии с ролями вручную. При создании диаграммы Swim Lane без использования готовой IDEF3-диаграммы она будет содержать только полосы, в которых необходимо вручную построить все необходимые объекты в соответствии со сценарием.

При необходимости можно добавлять новые полосы или удалять ненужные путем выбора опций в столбце Display Swim Lane.

Содержание диаграмм Swim Lane интуитивно понятно неподготовленному пользователю, и они чрезвычайно полезны на начальном этапе формализации предметной области в виде диаграмм и обычно используются в качестве модели «AS IS».

3.2 Построение диаграмм IDEF0, IDEF3 и DFD

3.2.1 Методология IDEFO

Методология структурного моделирования предполагает построение функциональной модели «TO-BE» («Как должно быть»), т. е. модели, которая должна использоваться при построении информационной системы для предприятия. Для построения функциональных моделей обычно используется методология IDEF0, которая хорошо представлена в пакетах Design/IDEF и All Fusion Process Modeler (BPwin).

Построение lDEFO-моделей в среде этих двух пакетов практически не отличается, но в пакете BPwin возможно построение интегрированной функциональной модели, объединяющей диаграммы трех методологий: IDEFO, IDEF3 и DFD (Data Flow Diagramm).

Методология IDEFO, более известная как методология SADT (Structure Analysis and Design Technique), предназначена для представления функций системы и анализа требований к системам. Она является одной из самых известных и широко используемых методологий проектирования РИС.

В IDEFO реализованы идеи системного анализа, под которыми понимают исследования, начинающиеся с общего обзора системы, а затем детализируют ее в виде иерархической структуры с определенным числом уровней, на каждом из которых допускается не более 8 элементов. В результате система разбивается на функциональные части, дается их описание, исследуются информационные потоки и формализуется структура данных.

Нотация IDEF0 позволяет наглядно представить бизнес-процессы и легко выявить такие недостатки, как недостаточно эффективное управление, ненужные, дублирующие, избыточные или неэффективные работы, неправильно использующиеся

ресурсы и т. д. Признаком неэффективной организации работ является, например, отсутствие обратных связей по входу и управлению для многих критически важных работ.

В нотации IDEF0 основными элементами диаграмм являются функциональные блоки и дуги, которые представляются соответственно прямоугольниками и стрелками. Все дуги и функциональные блоки должны быть поименованы. Дуги представляют собой информацию, в которой блоки нуждаются или ее производят. Поэтому имена дуг - существительные, имена блоков должны начинаться с глагола или отглагольного существительного, поскольку блоки на диаграмме представляют функции, которые показывают, что должно выполняться: «выполнить расчет зарплаты», «сформировать накладную», «оформление договора». Каждый блок имеет свой номер, который размещается обычно в нижнем правом углу.

В методологии IDEF0 функциональный блок, который на самом верхнем уровне иерархии представляет систему в качестве единого блока (контекстная диаграмма), детализируется на другой диаграмме с помощью нескольких блоков, соединенных между собой интерфейсными дугами. Эти блоки представляют основные подфункции (подсистемы) единого исходного модуля. Каждый из этих подмодулей может быть декомпозирован подобным образом для более детального представления. Количество уровней иерархии не ограничивается, процесс декомпозиции блоков заканчивается тогда, когда каждый из модулей самого нижнего уровня декомпозиции может быть реализован в проектируемой системе одним программным модулем.

Результатом применения IDEFO-методологии является функциональная модель, которая состоит из диаграмм, фрагментов текста и глоссария, имеющих ссылки друг на друга. Диаграммы являются главными компонентами модели.

В IDEF0 используется четыре типа дуг: входные (INPUT), управления (CONTROL), выходные (OUTPUT) и механизма (MECHANIZM), представляющие собой ICOM-коды (аббревиатура из первых букв английских названий дуг). В качестве иллюстрации приведем контекстную диаграмму функциональной модели управления городом (рисунок 13).

Входы (INPUT) — это информация, которая используется или преобразуется функциональным блоком для получения результата. Они входят в функциональный блок слева (медицинская карта, заявления клиентов, накладные на поступивший товар и т. д.), допускается отсутствие входных дуг у блока.

Управление (CONTROL) — это правила, стратегии, процедуры или стандарты, которые указывают, чем руководствуется исполнитель при выполнении каждого из функциональных блоков (приказы Минздрава России, нормативная документация для выполнения данной работы в виде стандартов, законов, устава предприятия, инструкций, технических условий, должностных обязанностей и т. п.). Дуги управления входят в функциональный блок сверху, должна быть обязательно хотя бы одна дуга.

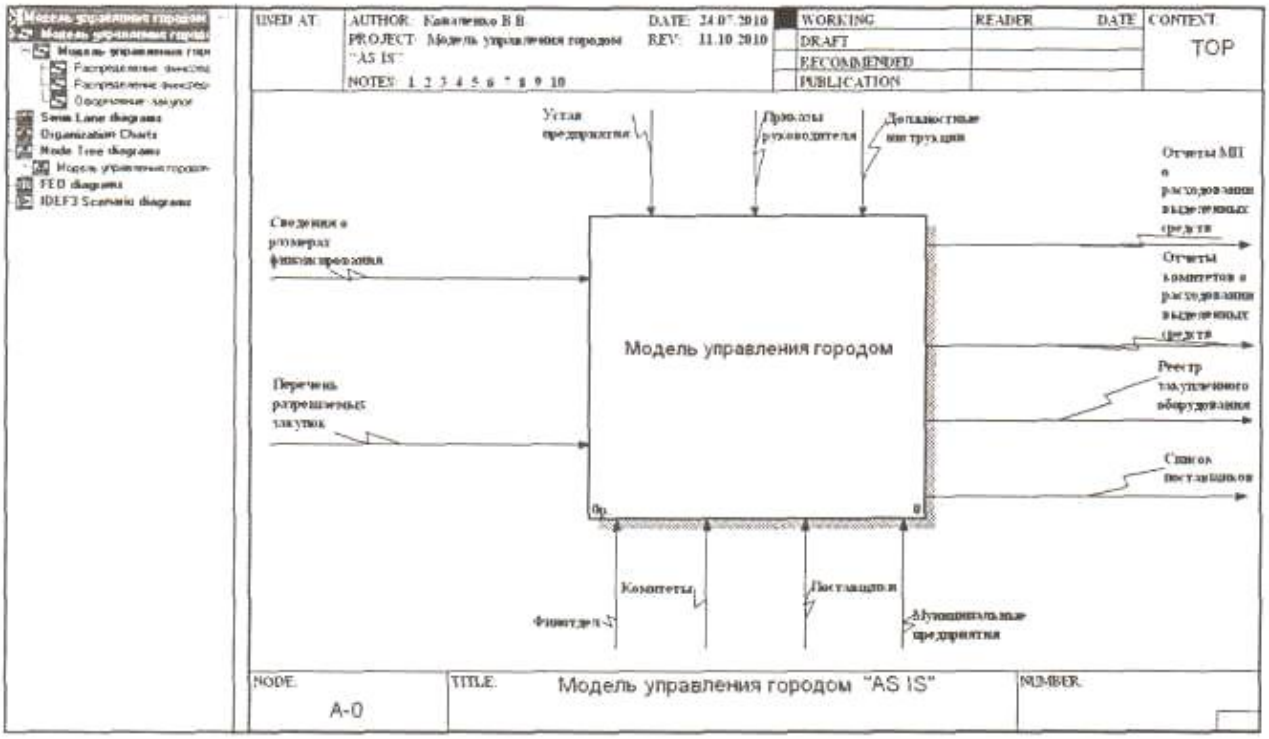


Рисунок 13 - Контекстная диаграмма методологии IDEF0

Механизмы (MECHANIZM) — это ресурсы, которые выполняют работу в функциональном блоке (персонал, подразделения, предприятия и т. п.). Они определяют, кто является ответственным за выполнение каждой из функций. Эти дуги вхо-

дят в функциональный блок снизу. По усмотрению системного аналитика дуги ресурсов могут не изображаться графически, а представляться в виде текстового описания на диаграмме.

Дуги механизмов или управления могут быть входами других блоков.

Выходы (OUTPUT) - это данные, которые получаются в результате выполнения функции. Они указывают, что является результатом выполнения работы (всевозможные выходные формы, отчеты, договоры, больничный лист и т.д.). Выходные дуги выходят справа от функционального блока.

Обязательно должна быть хотя бы одна дуга, так как работа без результата не имеет смысла.

Контекстная диаграмма является верхним уровнем иерархии функциональной модели и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. Для ее декомпозиции необходимо после выделения функционального блока щелкнуть левой клавишей мыши (ЛКМ) по ярлыку в панели инструментов.

В появившемся диалоговом окне (рисунок 14) следует выбрать методологию, в нотации которой будет выполняться его декомпозиция, и количество блоков декомпозиции.

Щелкнув по клавише ОК, получают диаграмму декомпозиции первого уровня, которая содержит: выбранное выше количество блоков и ICOM-коды, т. е. названия всех дуг контекстной модели. Необходимо дать названия блокам, подсоединить к ним по смыслу ICOM-коды и построить требуемые соединительные (внутренние) дуги, присвоив им названия (рисунок 15).

В результате получают представление о том, какие функции и в какой последовательности выполняются в рамках декомпозированной родительской функции.

Если ICOM-объекты не используются на каком-либо уровне иерархии, то дуги, соединяющие эти объекты, можно поместить в туннель. Причем если в туннель помещен конец дуги (конец дуги помещается в квадратные скобки), то дуга и соответствующий ей ICOM-объект отсутствует на диаграмме-родителе.

Если в туннель помещено начало дуги, то дуга и соответствующий ей ICOM-объект отсутствуют на диаграмме-потомке.

Для того чтобы извлечь дугу из туннеля, необходимо курсор поместить между квадратных скобок, а затем вызвать щелчком ПКМ контекстное меню и выполнить команду Arrow Tunnel. В появившемся диалоговом окне извлечь дугу из туннеля.

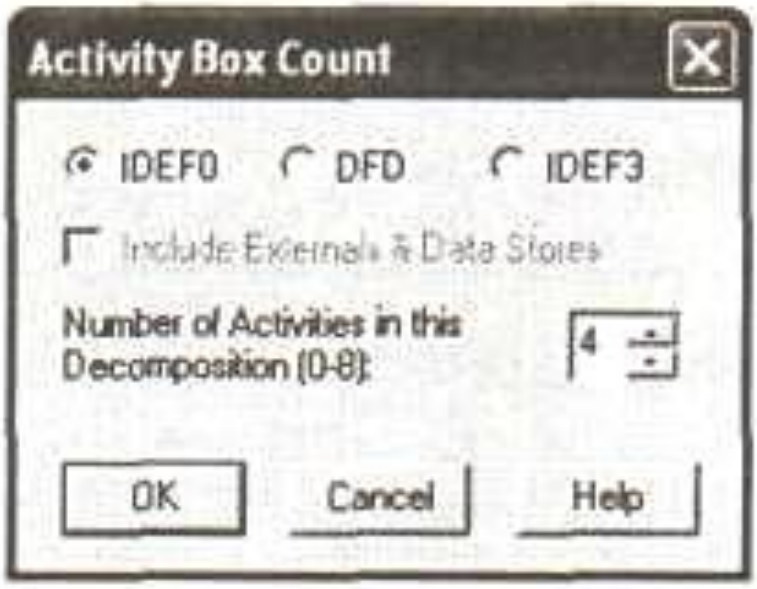


Рисунок 14 – Диалоговое окно определения нотаций

Дуги могут разветвляться и соединяться, образовывать обратные связи, итерации. Обратные связи этого типа на диаграмме отображаются снизу, т. е. обходят функции. С помощью этой обратной связи на диаграмме отображают тот факт, что регулярно анализируется выполнение плана и при необходимости формируют информацию, необходимую для корректировки плана на следующий период.

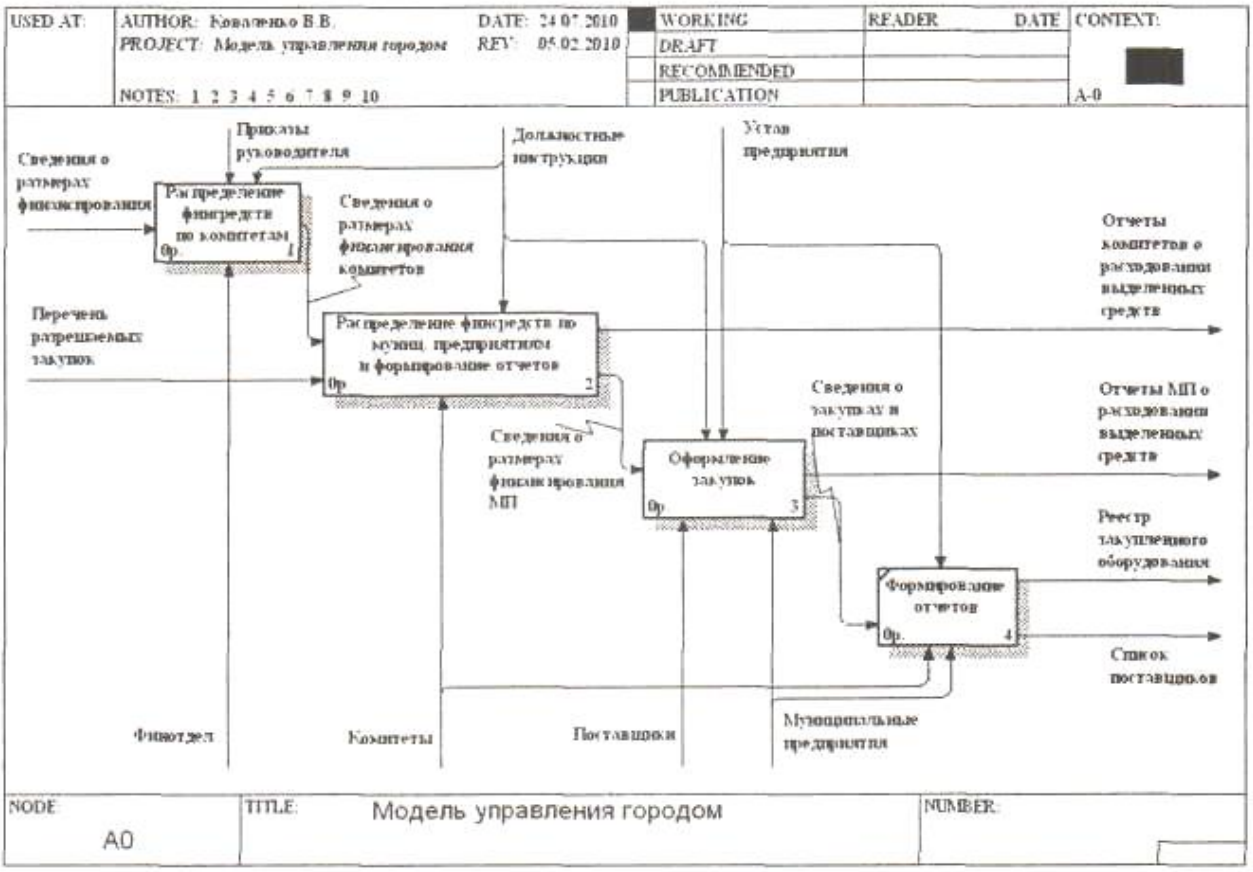


Рисунок 15 - IDEFO-диаграмма первого уровня декомпозиции

Обратная связь по управлению изображается в виде дуги, выходящей из правой стороны блока и входящей в верхнюю сторону блока, огибая диаграмму сверху функций. Обратная связь по управлению означает, что при анализе выполнения плана могут быть приняты оперативные управленческие решения, регулирующие выполнение по производству готовой продукции.

Для перемещения объекта необходимо вначале его выделить. Затем при нажатой Л КМ переместить указатель объекта, который появится вместо курсора, в нужную точку.

При удалении объекта его необходимо выделить, а затем нажать на клавишу <DeIete>. Для редактирования графических параметров объекта его следует выделить. После этого установить указатель мыши на необходимый маленький квадра-

тик и, удерживая ЛКМ, перемещать курсор до получения желаемого графического состояния редактируемого объекта.

У выделенного объекта можно изменять тип линий, толщину и цвет, а также его фон. Это делается с помощью верхней инструментальной панели стандартным для Windows-приложений способом.

Для редактирования имен объектов необходимо с ним совместить курсор, с помощью ПКМ вызвать контекстное меню и воспользоваться командой Name.

Для удаления какой-либо из диаграмм ее необходимо открыть, выполнить команду Edit/Delete diagram и щелкнуть по кнопке Delete. Другой вариант удаления — выполнить команду Diagram/Diagram manager.

Для создания в автоматическом режиме отчета по IDEFO-модели необходимо находиться на любой IDEFO-странице и выполнить команду Tools/Report. После этого открывается меню, в котором следует выбрать один из семи отчетов, а также тип, стиль и размер шрифта.

Иерархия IDEFO-модели может быть представлена в виде «дерева узлов» (рисунок 16). который позволяет задать необходимые опции для выбора нужного типа диаграммы.

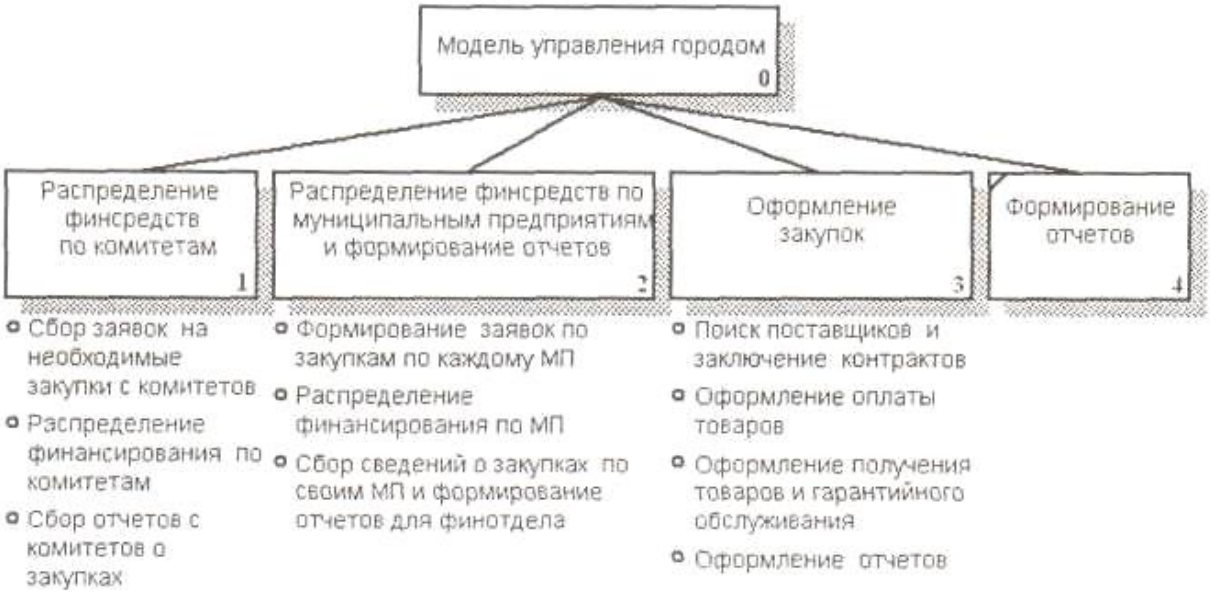


Рисунок 16 - «Дерево узлов»

Для этого необходимо воспользоваться командой Node Tree diagrams в навигаторе, запускающей мастер построения диаграммы (Node Tree Wizard).

IDEFO-методология в большей степени ориентируется на описание процессов верхних уровней. Ее применение характеризуется полнотой описания предметной области, которая достигается за счет наличия средств, отображающих управляющие воздействия, обратные связи по управлению и информации. Эта методология обеспечивает связанность диаграмм различных уровней декомпозиции в составе единой модели предметной области за счет использования механизма мигрирования (ICOM-кодов) и туннелирования дуг. Такой механизм обеспечивает связанность создаваемых диаграмм между собой и делает модель предметной области наглядной.

На более низких уровнях иерархии можно использовать применение методологий IDEF3 и DFD для создания диаграмм в составе единой модели. Основными признаками для применения методологии IDEF3 является наличие логики в описании процессов, для применения методологии DFD — наличие хранилищ для промежуточного хранения данных.

3.2.2 Методология IDEF3

Нотация IDEF3 (Workflow diagramming) является второй важнейшей категорией после IDEF0 и ориентирована на описание логики взаимодействия информационных потоков. Особенно удобно применять IDEF3 на нижних уровнях иерархии функциональных моделей при описании работ, выполняемых в подразделениях и на рабочих местах (рисунок 17). С помощью диаграмм IDEF3 удобно описывать сценарии действий работников подразделения, содержащих логику: когда процессы выполняются в определенной последовательности, задаваемой соответствующими логическими условиями (есть ли товар на складе, подписан ли документ, заключен ли договор с поставщиком и т. п.).

Центральным компонентом модели является единица работы, близкая по смыслу к работе IDEF0, которая изображается прямоугольником и имеет имя в виде

отглагольного существительного в составе фразы, обозначающего процесс действия (изготовление изделия, генерация выходной формы и т. п.).

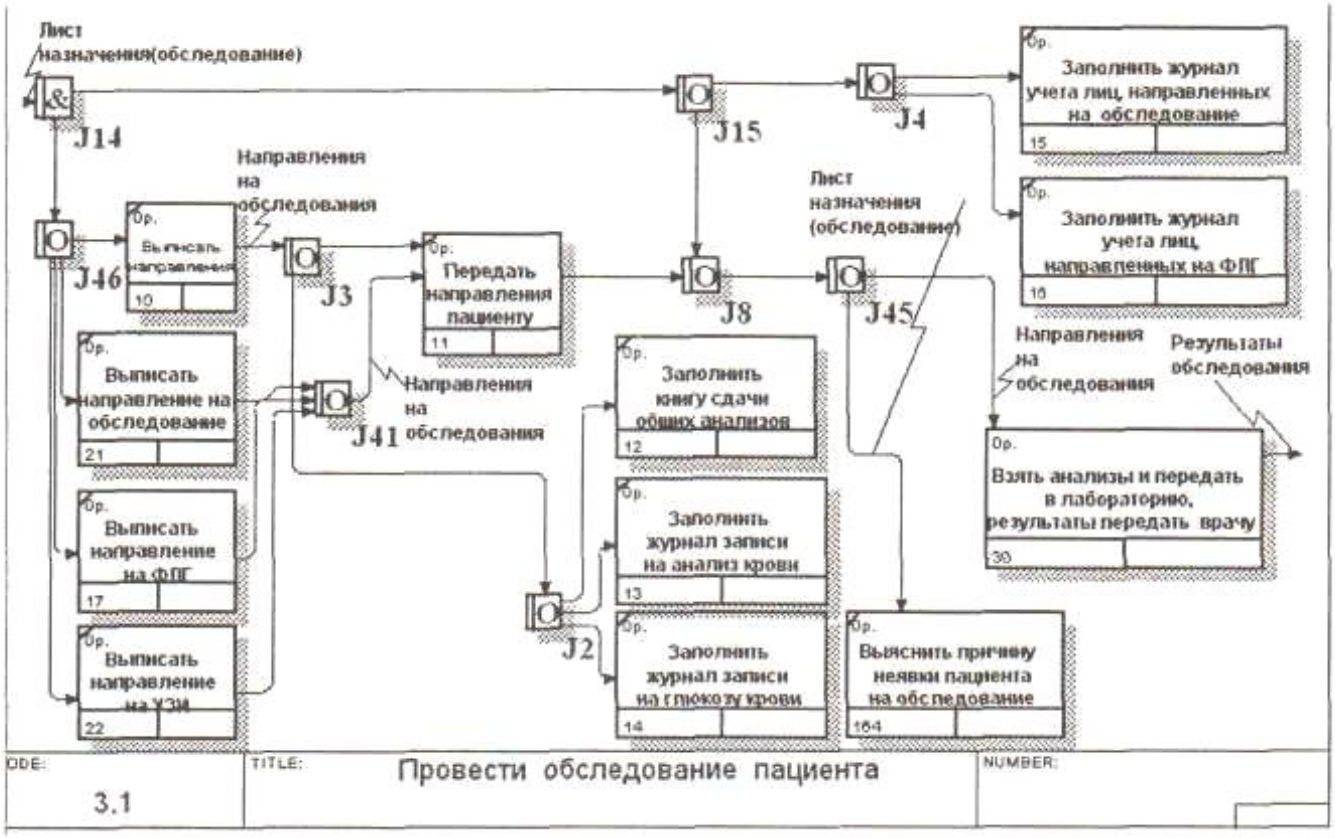


Рисунок 17 – IDEF3-диаграмма сценария «Провести обследование пациента»

Взаимоотношения между работами реализуются с помощью стрелок, которые рисуются слева направо или сверху вниз.

Старшая стрелка (связь предшествования) изображается сплошной линией и означает, что работа-источник должна закончиться прежде, чем работа-цель начнется.

Связь отношения обозначается пунктирной линией и показывает связь между двумя работами или между работой и объектом ссылки.

Объект Ссылка (References) в IDEF3 выражает данные или определенную идею, которые нельзя связать со стрелкой, перекрестком или работой (клиент, заказы клиента, склад и т. п.). Объекты ссылок должны быть связаны с единицами работ или перекрестками пунктирными линиями.

Характерным объектом IDEF3 является Перекресток (Junction), который отображает не только логику взаимодействия стрелок при слиянии и разветвлении, но и для отображения множества событий, которые могут быть завершены перед началом следующей работы.

В диаграммах IDEF3 любое разветвление или объединение стрелок происходит только с помощью перекрестков для разветвления (Fan-out Junction) или слияния (Fan-in Junction) соответственно. Имеется пять наименований перекрестков, которые обеспечивают любую логику в сценариях (таблица 1).

Таблица 1 - Описание перекрестков IDEF3

Обозначение	Наименование	Смысл в случае слияния стрелок	Смысл в случае разветвления стрелок
	Asynchronus AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronus AND	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronus OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronus OR	Один или несколько предшествующих процессов завершены одновременно	Один или несколько процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Для внесения перекрестка в диаграмму служит кнопка в палитре инструментов (добавить в диаграмму перекресток — Junction), тип перекрестка выбирается из диалогового окна Junction Type Editor. Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J.

Работы в диаграммах IDEF3 также декомпозируются, как и в диаграммах IDEF0, но при этом не происходит мигрирования и туннелирования стрелок. Поэтому системный аналитик должен сам заботиться о связанности моделирования процесса и корректности декомпозиции.

Эту особенность необходимо учитывать при декомпозиции работы из диаграмм IDEF0 в диаграмму IDEF3. Как уже было отмечено, согласно нотации IDEF3-диаграмма не должна иметь граничных стрелок, все стрелки должны заканчиваться на работах, перекрестках или объектах ссылки. Поэтому все стрелки на родительском блоке IDEFO-диаграммы автоматически помещаются в тоннель, а в дочерней диаграмме вместо граничных стрелок следует создать объекты ссылки и затем внутренними стрелками соединить их с соответствующими работами.

3.2.3 Методология диаграммы потоков данных (DFD)

Диаграммы потоков данных (Data Flow Diagramming, DFD) применяются для документирования механизма передачи и обработки информации в проектируемой системе, они удобны для наглядного изображения текущей работы системы документооборота организации. Обычно DFD используют в качестве дополнения IDEFO-модели или в ее составе на нижних уровнях иерархии для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации.

DFD строится на основе четырех элементов: работа, хранилище данных, внешние ссылки, дуги. Наличие в диаграммах DFD элементов для описания источников, приемников и хранилищ данных позволяет более эффективно и наглядно описать процесс документооборота.

Работы (функции обработки информации) обозначают процессы, которые обрабатывают и изменяют ситуацию. Они представляются на диаграммах в виде прямоугольников со скругленными углами. Стрелки идут от объекта-источника к объекту-приемнику, обозначая информационные потоки в системе документооборота. В DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, поэтому стрелки могут подходить и выходить из любой грани прямоугольника работы, они могут сливаться и разветвляться.

Хранилища данных изображают объекты в статике, например в очереди на обработку, представляя собой данные, к которым осуществляется доступ и которые могут быть созданы или изменены работами (бумажный документ, файл, база данных и т. п.).

Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом. Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее получения, при этом данные могут выбираться в любом порядке.

В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое. Имя хранилища должно определять его содержимое и быть существительным. На одной диаграмме может присутствовать несколько копий одного и того же хранилища, чтобы исключить наличие длинных и запутанных стрелок.

Внешние ссылки (сущности) указывают на организацию или человека, которые участвуют в процессе обмена информацией с системой, но располагаются за пределами данной диаграммы. То есть они изображают входы в систему и/или выходы из нее. Внешние ссылки изображаются в виде прямоугольника с тенью и располагаются по краям диаграммы. Одна внешняя ссылка может быть использована многократно на одной или нескольких диаграммах, чтобы исключить наличие длинных и запутанных стрелок.

В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы — движение объектов, хранение объектов, поставка и распространение объектов.

В DFD также применяются двунаправленные стрелки для описания диалогов типа команды-ответа между работами, между работой и внешней сущностью и между внешними сущностями.

Процесс построения DFD начинается с создания так называемой основной диаграммы типа «звезда», на которой представлен моделируемый процесс и все внешние сущности, с которыми он взаимодействует.

С помощью нотации DFD можно создать описание реально существующих в организации потоков данных, как по процессному, так и по функциональному признаку. В первом случае получают модели бизнес-процессов в формате DFD, во втором — схему обмена данными между подразделениями.

Диаграмма потоков данных при функциональном подходе для кардиологического терапевтического отделения представлена на рисунке 18. Пациент при поступлении в терапевтическое отделение представляет медицинскую и статистическую карты, проходит обследование — все это является входным потоком документов.

На выходе диаграммы данного уровня пациент может получить следующий перечень выходных документов: медицинскую и статистическую карты с результатами обследования и лечения, санаторно-курортную карту и больничный лист. На диаграмме присутствует хранилище данных, в котором будут размешаться копии документов и отчетные журналы. В тех случаях, когда организационная структура терапевтического отделения не представляет интереса, более удобно использовать «объектную» диаграмму потоков данных (рисунок 19). Основным объектом на диаграмме является терапевтическое отделение, а пациент представляется диаграмме как внешняя ссылка.

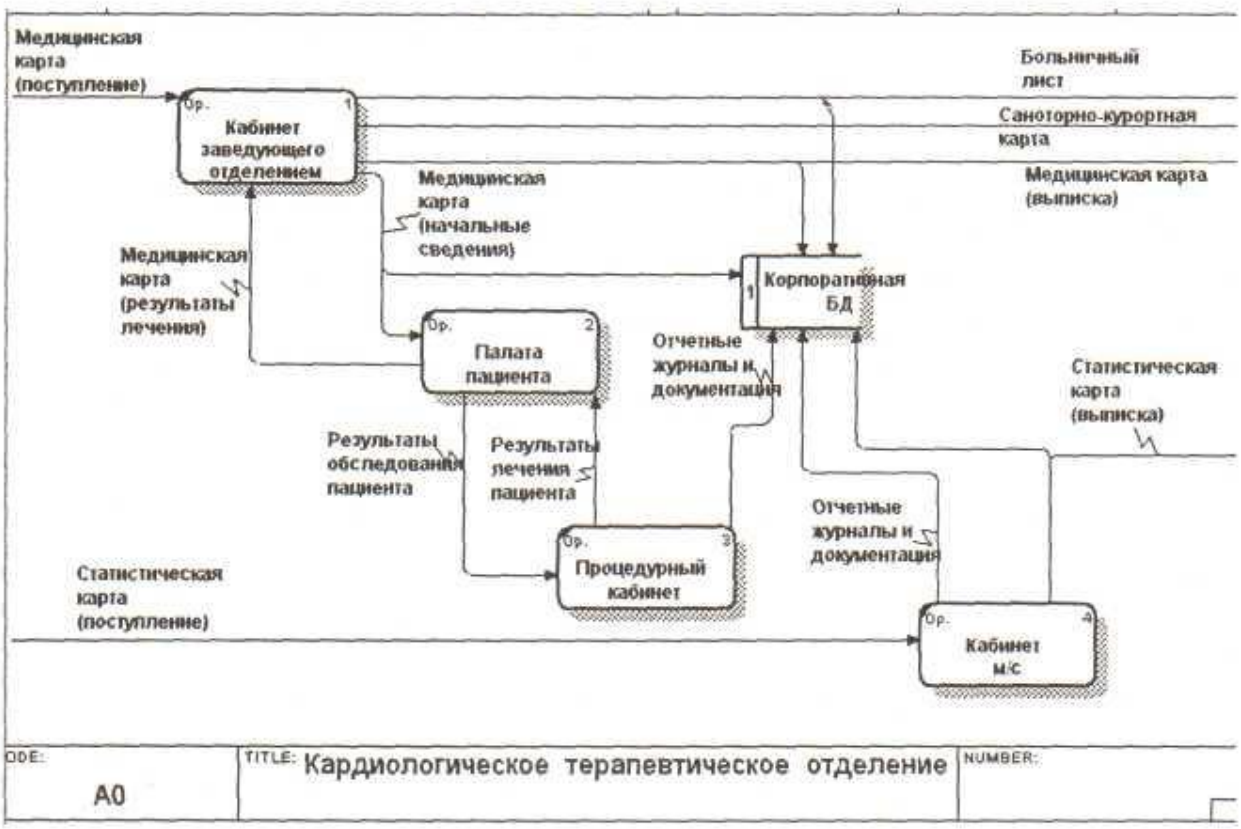


Рисунок 18 - Диаграмма потоков данных (DFD-методология)

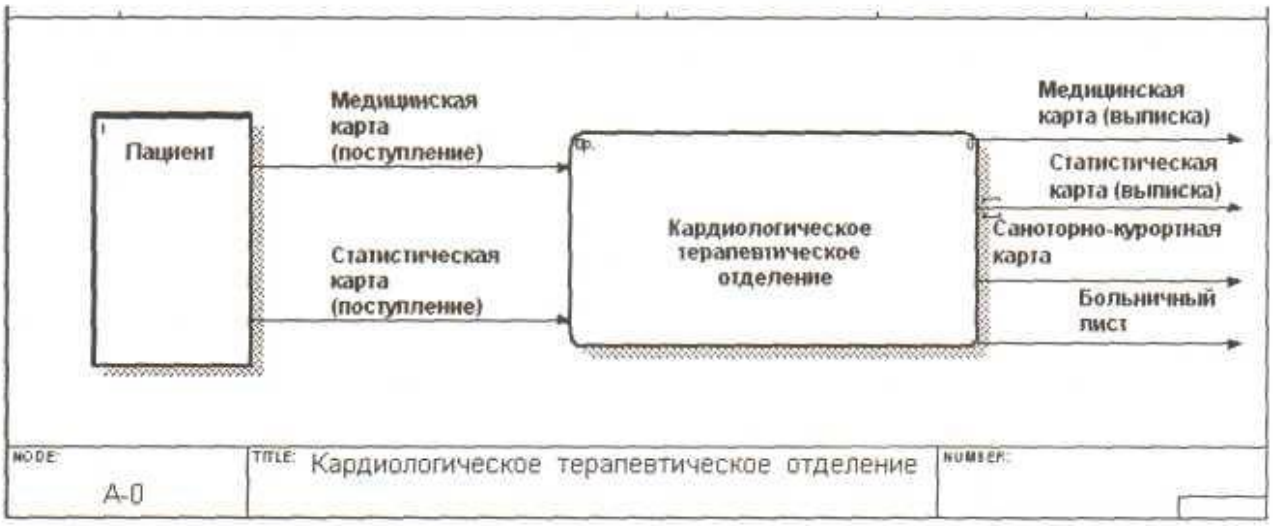


Рисунок 19 - Диаграмма потоков данных при объектном подходе (DFD-методология)

Подобная реализация диаграммы потоков данных позволяет более наглядно представить потоки документов между различными объектами.

На практике обычно применяют смешанные модели: на верхних уровнях иерархии используют нотацию IDEF0 для выявления функций, входящих в процесс. Затем при достижении некоторого уровня иерархии становится целесообразно сформировать диаграммы в формах DFD (для отображения потоков данных и материалов) и IDEF3 (для отображения логики в сложных сценариях).

Применение универсальных графических языков моделирования IDEFO, IDEF3 и DFD обеспечивает логическую целостность и полноту описания предметных областей.

Пакет BPwin обладает удобным инструментом для навигации по уровням иерархии модели Model Explorer, который по организации подобен обычному проводнику Windows. Функциональные IDEFO-блоки в Model Explorer показываются зеленым цветом, DFD — желтым и IDEF3 — синим. Щелкнув мышкой по любой из работ, представленных в навигаторе, пользователь может переходить на диаграмму, которая содержит данную работу. Однако следует помнить об определенных правилах декомпозиции работы одной нотации в диаграмму другой. В пакете BPwin допускаются только следующие переходы с одной нотации на другую: IDEFO -> DFD; IDEF0 -> IDEF3; DFD ->IDEF3.

Согласно нотации DFD-диаграмма не должна иметь граничных стрелок, т. е. все стрелки должны начинаться и заканчиваться на работах, хранилищах данных или внешних ссылках. Поэтому при строгом следовании правилам нотации необходимо на дочерней DFD-диаграмме удалить все граничные стрелки и вместо них создать соответствующие внешние ссылки и хранилища данных. Затем вместо удаленных граничных стрелок создать внутренние стрелки, начинающиеся с внешних ссылок или хранилищ данных.

В тех случаях, когда на смешанных моделях сложно придерживаться этих правил нотации, BPwin позволяет создавать граничные дуги на DFD-диаграммах, не идентифицируя их как синтаксическую ошибку.

При декомпозиции работы IDEFO или DFD в диаграмму IDEF3 дуги не мигрируют на дочернюю диаграмму, поскольку дуги на диаграммах IDEF3 могут показывать только последовательность выполнения работ и, следовательно, имеют другой смысл, чем в IDEFO и DFD.

Пакет BPwin позволяет во всех нотациях использовать для работы не прямоугольники, а практически любые геометрические фигуры. При этом на работах можно поместить изображение, импортированное в словарь Bitmap Dictionary (рисунок 20).

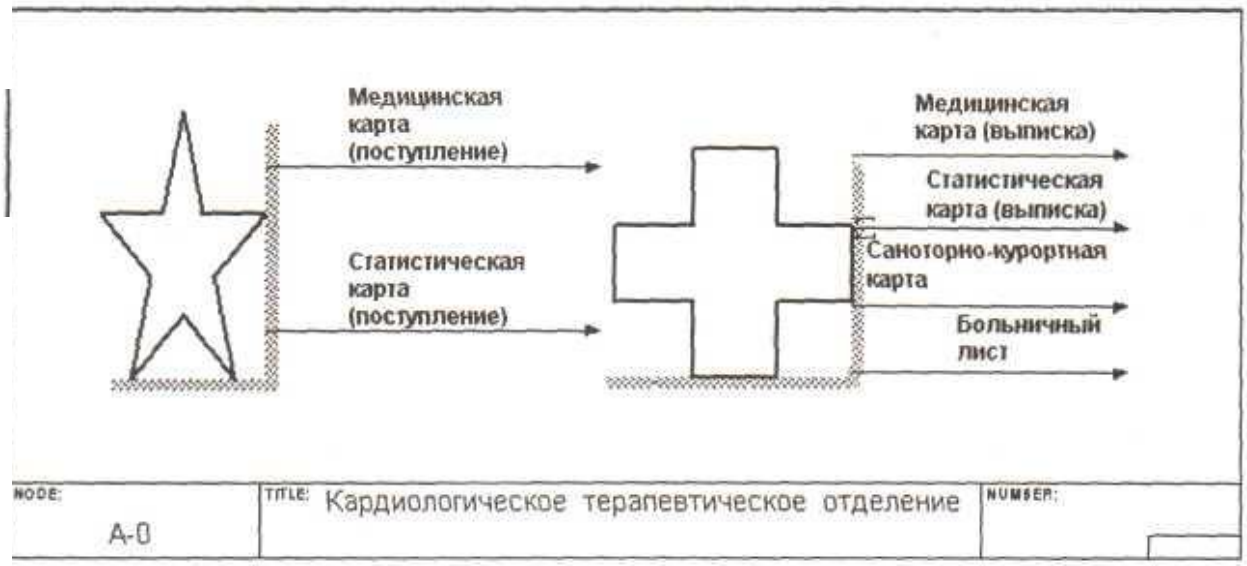


Рисунок 20 - Диаграмма потоков данных в нетрадиционном синтаксисе

Для использования нетрадиционного синтаксиса необходимо щелкнуть по работе и выбрать в контекстном меню пункт Box Style. Во вкладке Box Style следует выбрать опцию Custom и указать геометрическую фигуру (Shape) для работы и изображение (Bitmap). После щелчка по кнопке ОК на диаграмме работа отображается в нетрадиционном синтаксисе.

Использование нетрадиционного синтаксиса удобно не только для визуального эффекта, но и при преобразовании диаграммы IDEF3 в имитационную модель для пакета Arena.

3.2.4 Стоимостный анализ (Activity Based Costing, ABC)

Пакет BPwin предоставляет аналитику инструмент для оценки модели — стоимостный анализ, основанный на работах (Activity Based Costing, ABC). Исходными данными для функционального оценивания являются затраты на ресурсы, необходимые при эксплуатации данной информационной системы (зарплата персонала, стоимость арендуемых помещений и компьютерной техники, расходных материалов, коммунальных услуг и т. д.).

С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости эксплуатации системы, определение действительной стоимости поддержки клиента, идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь), обеспечение менеджеров финансовой мерой предлагаемых изменений и др.

При проведении стоимостного анализа в BPwin сначала задаются единицы измерения времени и денег. Для задания единиц измерения следует вызвать диалог Model Properties (меню Model/Model Properties), вкладка ABC Units (рисунок 21).

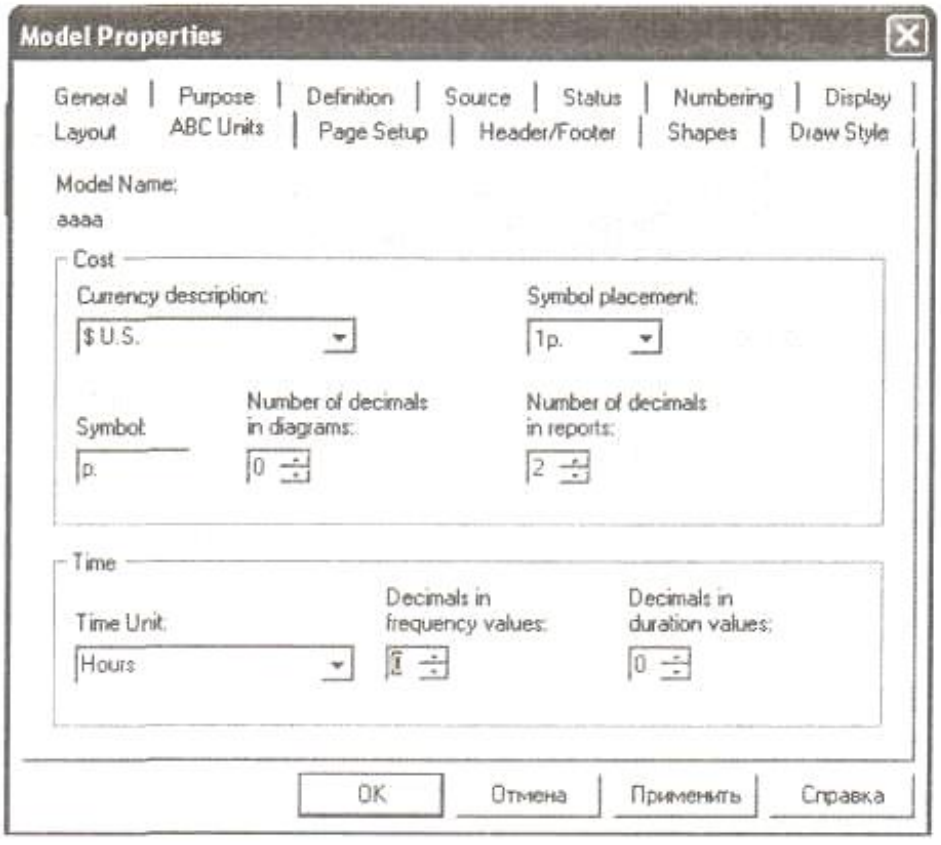


Рисунок 21 - Настройка единиц измерения валюты и времени

Затем описываются центры затрат (cost centers). Для внесения центров затрат необходимо вызвать диалог Cost Center Dictionary из меню Dictionary/Cost Center (рисунок 22). Каждому центру затрат следует дать подробное описание в окне Definition. Для отдельной модели задается один набор функциональных центров.

Для задания стоимости работы (для каждой работы на диаграмме декомпозиции нижнего уровня) следует щелкнуть ПКМ по работе и на всплывающем меню выбрать Costs (рисунок 23).

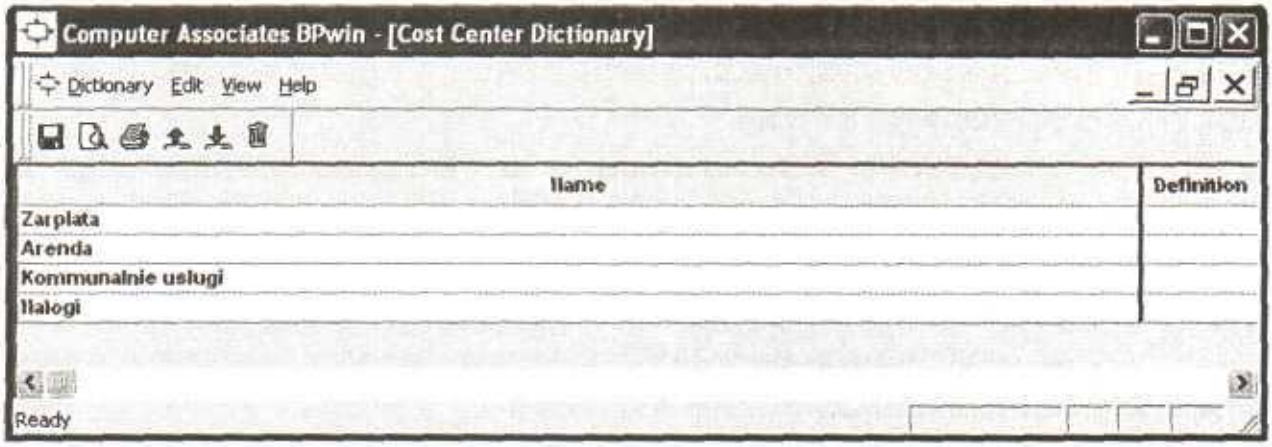


Рисунок 22 - Диалог Cost Center Dictionary

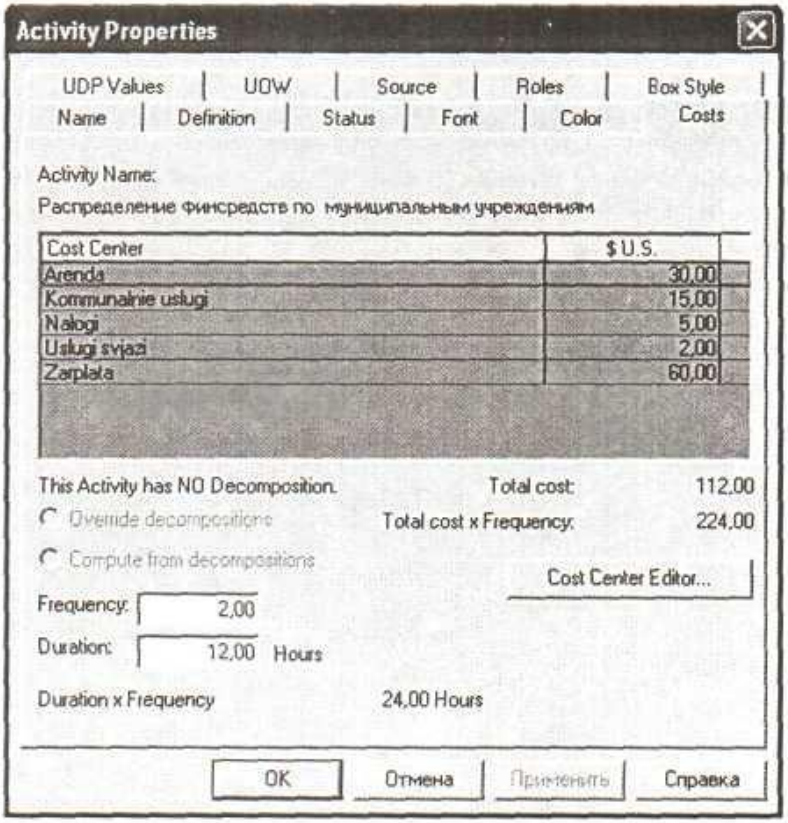


Рисунок 23 - Задание стоимости работ в диалоге Activity Cost

Во вкладке Costs диалога Activity Properties указывается частота проведения данной работы в рамках общего процесса (окно Frequency) и продолжительность (Duration). Затем следует выбрать в списке один из центров затрат и в окне Cost задать его стоимость. Аналогично назначаются суммы по каждому центру затрат, т. е. задается стоимость каждой работы по каждой статье расхода.

Общие затраты по работе рассчитываются как сумма по всем центрам затрат. При вычислении затрат вышестоящей (родительской) работы сначала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим Compute from Decompositions (в окне Activity Properties), подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх. Результат расчета стоимости размещается внутри каждого функционального блока (рисунок 24).

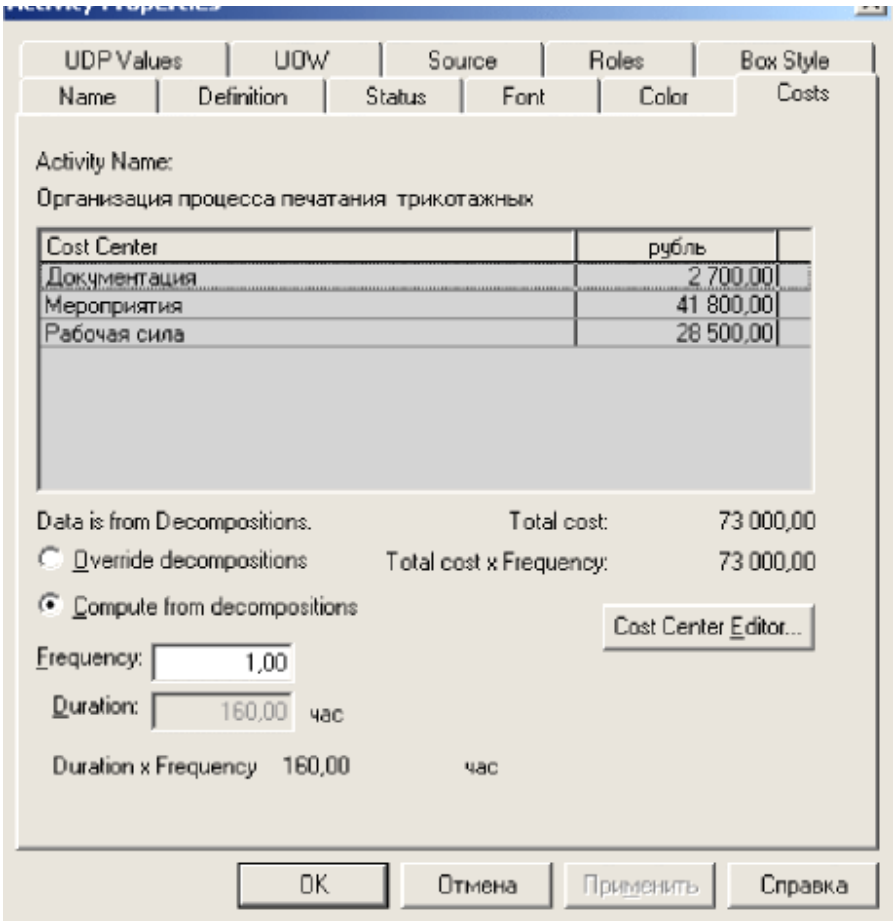


Рисунок 24 - Результат расчета стоимости

4 Проектирование объектов базы данных

4.1 Технология разработки базы данных

4.1.1 Описание внешнего уровня архитектуры базы данных

4.1.1.1 Схема иерархии функций

В ходе построение информационной модели исследуемой предметной области возникает необходимость определения информационных объектов, непосредственно участвующих в видах деятельности, выявленных на этапе функционального моделирования. С этой целью производится построение схемы иерархии функций проектируемой РИС. Вариант такой схемы показан на рисунке 25. На схеме определены основные информационные объекты и выполняемые над ними операции - функции в рамках функциональных подсистем проектируемой РИС. Исследование выявленных информационных объектов позволяет определить уровни доступа пользователей к информационным объектам и функциям РИС.

4.1.1.2 Пользователи РИС и уровни доступа

Каждый из предполагаемых пользователей базы данных будет иметь различные права доступа к информационным объектам. Данное разграничение предусмотрено, чтобы не допустить удаление или редактирование информации некомпетентными пользователями. Выделены следующие пользователи проектируемой РИС: администратор БД, прикладной программист, маркетолог который будет выполнять основную работу с РИС, директор.

Администратор базы данных имеет возможность производить добавление, удаление, редактирование и просмотр любой информации хранящейся в базе данных. Администратор может вносить изменения в структуру базы данных и имеет самый широкий круг прав доступа.

Прикладной программист имеет все привилегии кроме возможности вносить изменения в структуру базы данных и удалять таблицы, а пользователь-руководитель имеет права только на просмотр. Описание прав доступа возможных пользователей представлено в таблице 2.

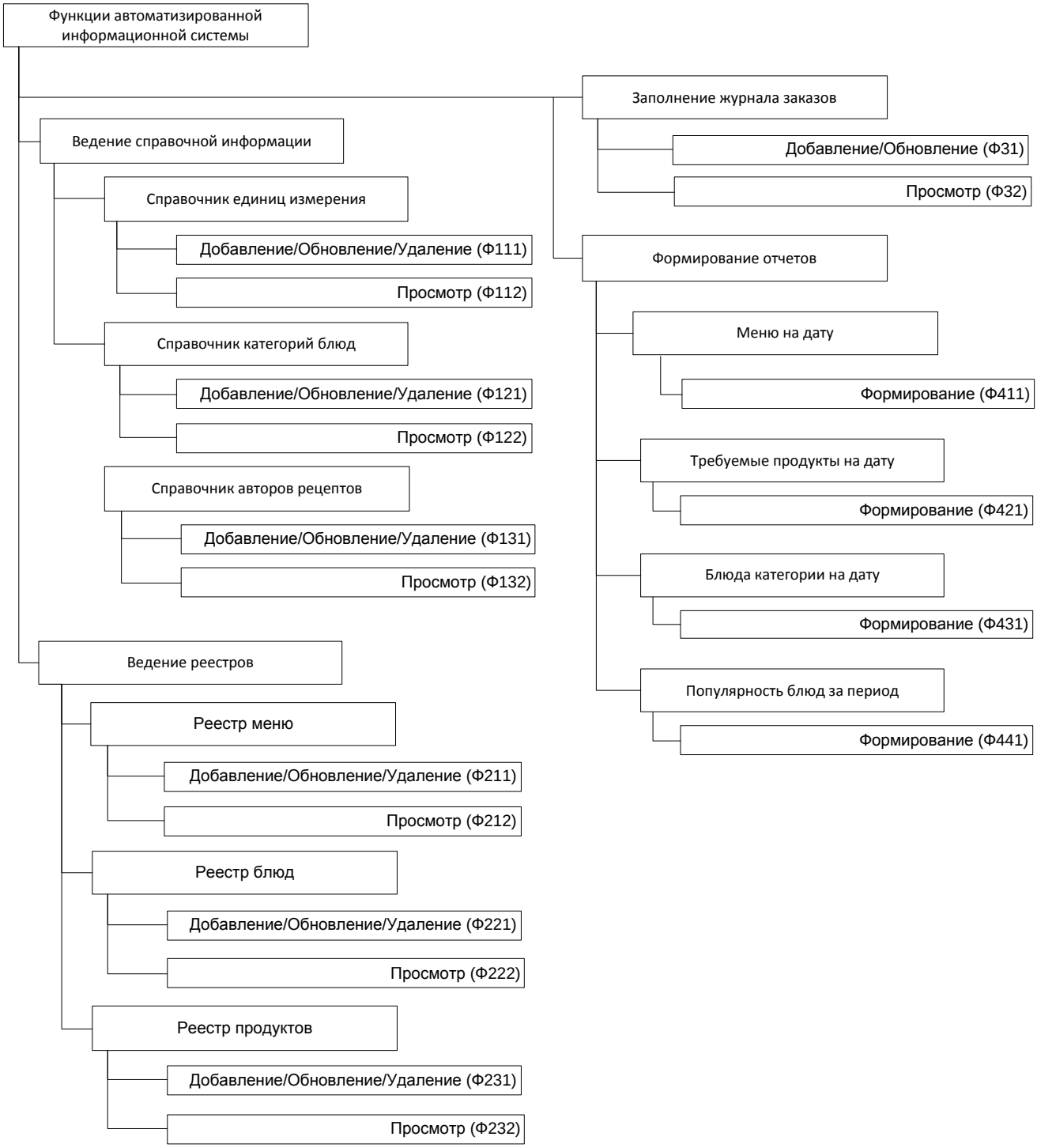


Рисунок 25 – Иерархия функций РИС

Таблица 2 – Уровни доступа пользователей системы

Объекты доступа	Пользователи			
	Администратор БД	Прикладной программист	Конечный пользователь - маркетолог	Конечный пользователь – директор
1	2	3	4	5
Справочник единиц измерения				
Код единицы	RIUD	RIUD	ID	-
Краткое наименование	RIUD	RIUD	RIUD	R
Справочник авторов рецептов				
Код автора	RIUD	RIUD	ID	-
Фамилия	RIUD	RIUD	RIUD	R
Имя	RIUD	RIUD	RIUD	R
Отчество	RIUD	RIUD	RIUD	R
Дата рождения	RIUD	RIUD	RIUD	R
Справочник категорий блюд				
Код категории	RIUD	RIUD	ID	-
Наименование	RIUD	RIUD	RIUD	R
Реестр продуктов				
Код продукта	RIUD	RIUD	ID	-
Наименование	RIUD	RIUD	RIUD	R
Краткое наименование	RIUD	RIUD	RIUD	R
Калорийность	RIUD	RIUD	RIUD	R
Ингредиенты				
Код ингредиента	RIUD	RIUD	ID	-
Количество	RIUD	RIUD	RIUD	R
Реестр блюд				
Код блюда	RIUD	RIUD	RID	R
Краткое наименование	RIUD	RIUD	RIUD	R
Наименование	RIUD	RIUD	RIUD	R
Дата создания	RIUD	RIUD	RIUD	R
Строки меню				
Код строки	RIUD	RIUD	ID	-
Заказы				
Код заказа	RIUD	RIUD	RID	R
Количество	RIUD	RIUD	RIUD	R
Меню				
Код меню	RIUD	RIUD	ID	-
Дата	RIUD	RIUD	RIUD	R

4.1.1.3 Формализованное описание предметной области

В результате анализа предметной области было получено её формализованное описание, которое представляется в виде таблицы. В таблице помечаются ключи, определяется тип, длина, опциональность и т.д. В таблицах 3 и 4 представлено формализованное описание предметной области.

Таблица 3 - Классы объектов, свойства

Класс объектов/ Свойство	Ключ (уникальный, первичный)	Физические характеристики (тип, длина)	Обязательность значения (м.б., д.б.)	Логические ограничения (диапазон значений)	Процессы (генерация, ввод, обновление, просмотр)
1	2	3	4	5	6
Справочник единиц измерения					
Код единицы	У1,ПК	число, 18	должен быть	> 0	Г.
Краткое наименование		симв., 50	должен быть		Вв, Об, Пр
Справочник авторов рецептов					
Код автора	У1,ПК	число, 18	должен быть	> 0	Г.
Фамилия		симв., 50	должен быть		Вв, Об, Пр
Имя		симв., 50	должен быть		Вв, Об, Пр
Отчество		симв., 50	может быть		Вв, Об, Пр
Дата рождения		дата	должен быть		Вв, Об, Пр
Справочник категорий блюд					
Код категории	У1,ПК	число, 18	должен быть	> 0	Г.
Наименование		симв., 50	должен быть		Вв, Об, Пр
Реестр продуктов					
Код продукта	У1,ПК	число, 18	должен быть	> 0	Г.
Наименование		симв., 50	должен быть		Вв, Об, Пр
Краткое наименование		симв., 50	должен быть		Вв, Об, Пр
Калорийность		число, 18	должен быть	> =0	Вв, Об, Пр
Ингредиенты					
Код ингредиента	У1,ПК	число, 18	должен быть	> 0	Г.
Количество		число, 18	должен быть	> 0	Вв, Об, Пр
Реестр блюд					

Продолжение таблицы 3

1	2	3	4	5	6
Код блюда	У1,ПК	число, 18	должен быть	> 0	Г., ПР.
Краткое наименование		симв., 50	должен быть		Вв, Об, Пр
Наименование		симв., 50	должен быть		Вв, Об, Пр
Дата создания		дата	должен быть		Вв, Об, Пр
Строки меню					
Код строки	У1,ПК	число, 18	должен быть	> 0	Г.
Заказы					
Код заказа	У1,ПК	число, 18	должен быть	> 0	Г., ПР.
Количество		число, 18	должен быть	> 0	Вв, Об, Пр
Меню					
Код меню	У1,ПК	число, 18	должен быть	> 0	Г.
Дата		дата	должен быть		Вв, Об, Пр

В таблице 3 использованы сокращения: У – уникальный ключ, П – первичный ключ, Г – генерация данных, Вв – ввод данных, Пр – просмотр данных, Об – обновление.

В таблице 4 представлено распределение данных по классам объектов с целью выделения главных и подчиненных для определения обязательности связей между ними.

Таблица 4 - Связи между классами объектов

Классы объектов		Опциональность связи		Имя связи со стороны		Тип связи со стороны	
главн. КО	подч. КО	главн. КО	подч. КО	главн. КО	подч. КО	главн. КО	подч. КО
1	2	3	4	5	6	7	8
ПРОДУКТ	ЕД.ИЗМЕРЕН.	имеет	относиться	М	1	Д.Б.	М.Б.
БЛЮДО	АВТОР	иметь	относится	М	1	Д.Б.	М.Б.
БЛЮДО	КАТЕГОРИЯ	имеет	относится	М	1	Д.Б.	М.Б.
ПРОДУКТ	ИНГРЕДИЕНТ	относится	соответствует	1	М	М.Б	Д.Б.
БЛЮДО	ИНГРЕДИЕНТ	имеет	соответствует	1	М	Д.Б.	Д.Б.
СТРОКА МЕНЮ	БЛЮДО	имеет	соответствует	М	1	Д.Б.	М.Б.
МЕНЮ	СТРОКА МЕНЮ	имеет	относятся	1	М	Д.Б.	Д.Б.
ЗАКАЗ	СТРОКА МЕНЮ	относится	соответствует	М	1	Д.Б.	М.Б.

Словесно-смысловая проверка связей между классами объектов:

Каждый ПРОДУКТ должен иметь только одну ЕДИНИЦУ ИЗМЕРЕНИЯ.

Одна ЕДИНИЦА ИЗМЕРЕНИЯ может относиться ко многим ПРОДУКТАМ.

Каждое БЛЮДО должно иметь только одного АВТОРА.

Один АВТОР может относиться ко многим БЛЮДАМ.

Каждое БЛЮДО должно иметь только одну КАТЕГОРИЮ.

Одна КАТЕГОРИЯ может соответствовать многим БЛЮДАМ.

Один ПРОДУКТ может относиться ко многим ИНГРЕДИЕНТАМ.

Каждый ИНГРЕДИЕНТ должен соответствовать только одному ПРОДУКТУ.

Одно БЛЮДО должно иметь много ИНГРЕДИЕНТОВ.

Каждый ИНГРЕДИЕНТ должен соответствовать только одному БЛЮДУ.

Каждая СТРОКА МЕНЮ должна иметь только одно БЛЮДО.

Одно БЛЮДО может соответствовать многим СТРОКАМ МЕНЮ.

Одно МЕНЮ должно иметь много СТРОК МЕНЮ.

Каждая СТРОКА МЕНЮ должна соответствовать только одному МЕНЮ.

Каждый ЗАКАЗ должен относиться только к одной СТРОКЕ МЕНЮ.

Одна СТРОКА МЕНЮ может соответствовать многим ЗАКАЗАМ.

4.1.2 Концептуальный уровень БД

4.1.2.1 Инфологическая модель предметной области

На рисунке 26 представлена пример ER-диаграммы предметной области, построенной по методологии Ричарда Баркера.

В методологии используются следующие соглашения:

- класс объектов отображается в виде четырехугольника с закругленными углами, а имя класса объектов указывается внутри четырехугольника, это имя существительное в единственном числе, отображенное заглавными буквами;
- свойства записываются внутри четырехугольника, отображающего класс объектов строчными буквами, это имя существительное в единственном числе;

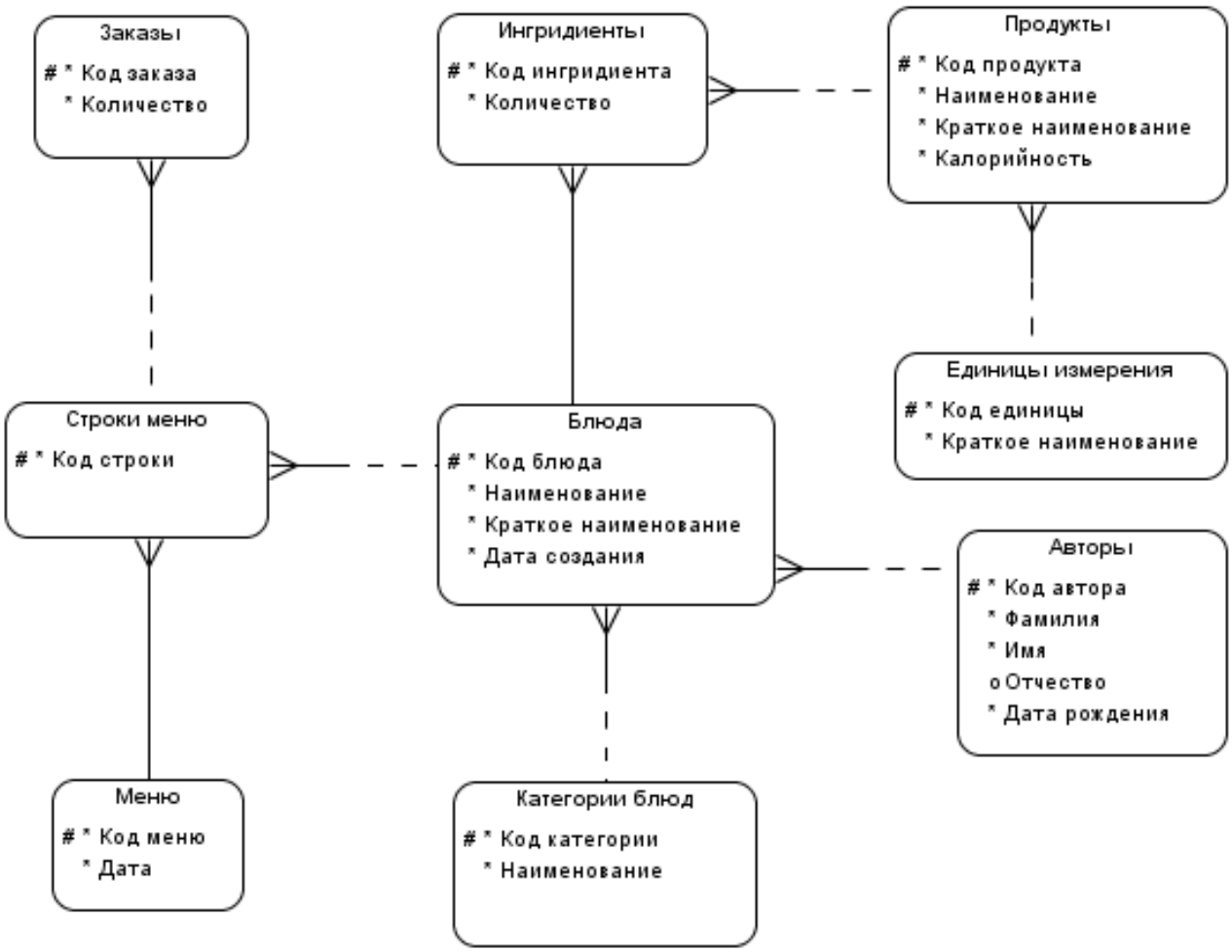


Рисунок 26 – ER-диаграмма предметной области

- четырехугольник, отображающий класс объектов, можно увеличивать до любых размеров, четырехугольники могут быть разных размеров;
- опциональность свойств помечается: обязательное свойство – звездочкой (*), необязательное – кружочком (o);
- уникальный идентификатор помечается #, если уникальных идентификаторов несколько, тогда каждый помечается номером, указанным в скобках, например, # (1), #(2);
- обязательная связь помечается сплошной линией, необязательная связь пунктирной линией;
- тип (мощность) связи “один” помечается линией, “много” – “вороньей лапой”.

4.1.2.2 Перекрестная проверка иерархии функций и модели предметной области

Выполняется перекрестная проверка между моделью данных и иерархией функций. Вариант результатов проверки представлены в таблице 5.

Таблица 5 - Перекрестная проверка иерархии функций и модели предметной области

Функция	Классы объектов								
	Справочник единиц измерения	Справочник категорий блюд	Справочник авторов рецептов	Реестр меню	Строки меню	Реестр блюд	Ингредиенты	Реестр продуктов	Журнал заказов
1	2	3	4	5	6	7	8	9	10
Ф(111)	I,U,D								
Ф(112)	R								
Ф(121)		I,U,D							
Ф(122)		R							
Ф(131)			I,U,D						
Ф(132)			R						
Ф(211)				I,U,D	I,U,D				
Ф(212)				R	R				
Ф(221)						I,U,D	I,U,D		
Ф(222)						R	R		
Ф(231)								I,U,D	
Ф(232)								R	
Ф(31)									I,U,D
Ф(32)									R
Ф(411)		R		R	R	R			
Ф(421)	R			R	R	R	R	R	
Ф(431)		R		R	R	R			
Ф(441)				R	R	R			R

В таблице использованы сокращения: I – операция добавления; U – операция обновления, R – операция чтения (выборки), Ф – конечная функция «ветки» иерархии функций.

Полученная таблица не имеет пустых строк, следовательно, модель предметной области является достаточной для реализации выделенных функций.

Модель не является избыточной, так как пустые столбцы в таблице также отсутствуют.

4.1.2.3 Даталогическая модель

Даталогическая модель данных представляется в виде логической структуры с указанием первичных и внешних ключей, отображением связей и указанием типа связи (рисунок 27).

По правилам метода нисходящего проектирования даталогическая модель строится на основе реляционной модели путем преобразования схемы отношений и связей между классами объектов по некоторым правилам. Название класса объектов становится названием таблицы. Название свойств становится названием полей таблицы.

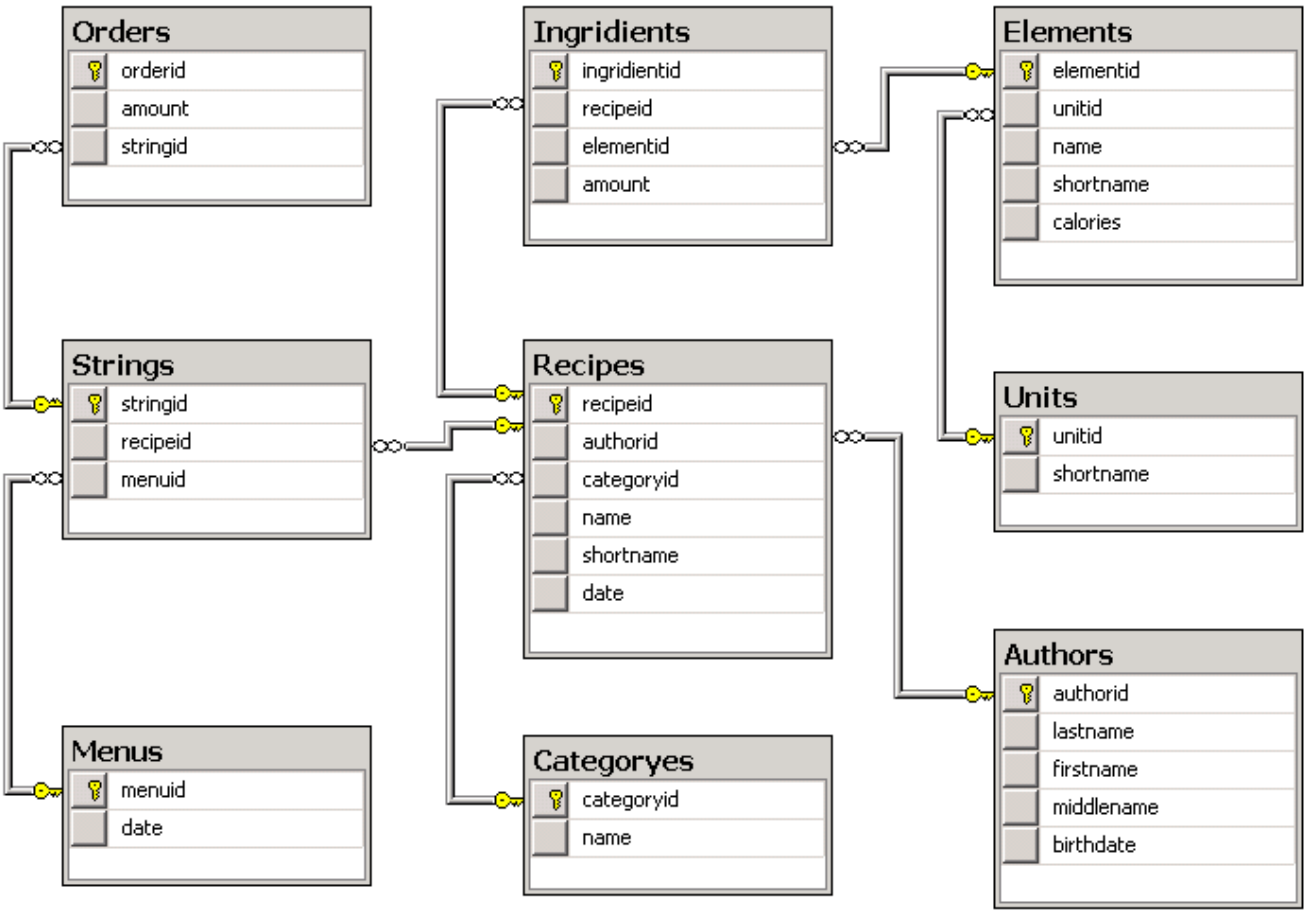


Рисунок 27 – Даталогическая модель предметной области

Преобразование связей «один ко многим» реализуется путем копирования первичного ключа из реляционного отношения на стороне «один» в реляционное отношение на стороне «много», а копия становится внешним ключом. Для моделирования иерархии данных используют рекурсивную связь «один ко многим». Преобразование рекурсивной связи реализуется путем копирования первичного ключа в эту же таблицу.

Для реализации связи «многие ко многим» создается суррогатная дополнительная таблица. В неё включаются копии первичных ключей из таблиц разрываемой связи. Эти копии становятся внешними ключами.

2.1.2.4 Анализ схем отношений на соответствие 3НФ

Необходимо проверить полученную схему на соответствие её нормальной форме Бойса-Кодда.

Нормализация реляционных таблиц подразумевает процесс последовательного преобразования таких таблиц к одной из нормальных форм – к одному из канонических представлений, принятых в реляционной модели данных.

Требования нормализации направлены на то, чтобы последовательно устранить так называемые «аномалии» в базе данных: неоправданное дублирование хранящихся данных или неоправданное усложнение алгоритмов работы с базой данных.

Полученная схема может быть не нормализована по следующим причинам:

- отсутствие опыта проектировщика;

- не были выделены все классы объектов присущие данной предметной области;

- не правильно распределены качественные и количественные характеристики по классам объектов.

Поэтому после построения даталогической модели необходимо провести анализ полученных схем отношений на соответствие нормальной форме Бойса-Кодда (НФБК).

Проанализируем полученные схемы реляционных отношений разработанной базы данных на соответствие НФБК.

Ввиду того, что все атрибуты отношений атомарные и нет повторяющихся групп, можно говорить, что достигнута первая нормальная форма (1НФ).

В каждой из полученных схем использованы не составные ключи, что делает невозможными частичные зависимости, а значит, достигнута вторая нормальная форма (2НФ).

В полученных схемах отношений отсутствуют транзитивные зависимости, что подтверждает их соответствие третьей нормальной форме (3НФ).

Кроме того, каждый детерминант отношения является потенциальным ключом, что свидетельствует о достижении НФБК.

В большинстве случаев достаточно привести схему отношений к нормальной форме Бойса-Кодда и на этом завершить нормализацию.

Проведя проверку на соответствие полученной схемы отношений НФБК, пришли к выводу, что схема отношений соответствует НФБК.

4.1.3 Физическая модель БД на основе выбранной СУБД

На физическом уровне производится описание проектируемых объектов БД средствами выбранной СУБД.

СУБД MS SQL Server 2008 R2 поддерживает ряд объектов реляционной базы данных: таблица, представление, триггер, домен, роль, индекс.

Таблица является базовой структурой реляционной базы данных, которая состоит из строк и столбцов с данными.

Представление – это поименованная динамически поддерживаемая СУБД выборка данных из одной или нескольких таблиц.

Триггер – хранимая процедура, автоматически выполняемая СУБД, как реакция на какое-либо действие в таблице.

Индекс – последовательность упорядоченных данных в столбце или столбцах таблицы, индексы содержат ссылки на записи в столбцах.

Домен – это объект, использующийся как альтернатива типу данных для столбца таблицы, дополнительное множество значений, на котором могут быть определены один или более столбцов одной или нескольких таблиц.

Роль представляет собой именованных набор прав и привилегий на объект БД.

Описанные объекты реализованы в разрабатываемой БД.

Есть несколько способов, с помощью которых структура базы данных описывается для СУБД. Язык, используемый для определения такой структуры, называется языком определения данных (ЯОД).

Техническое описание проектируемых объектов БД на языке определения данных СУБД MS SQL Server 2008 R2 представлено в таблицах 6 – 14.

В таблице 6 приведено техническое описание реляционной таблицы «Тип товара» на ЯОД СУБД MS SQL Server 2008 R2.

Таблица 6 – Реляционная таблица «Orders»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
orderid	Primary key	Numeric(18)	Not null	4
amount		Numeric(18)	Not null	67
stringid		Numeric(18)	Not null	3

Таблица 7 – Реляционная таблица «Strings»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
stringid	Primary key	Numeric(18)	Not null	4
recipeid		Numeric(18)	Not null	3
menuid		Numeric(18)	Not null	3

Таблица 8 – Реляционная таблица «Menus»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
menuid	Primary key	Numeric(18)	Not null	4
date		Datetime	Not null	01/12/2014

Таблица 9 – Реляционная таблица «Ingridients»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
ingridientid	Primary key	Numeric(18)	Not null	4
recipeid		Numeric(18)	Not null	3
elementid		Numeric(18)	Not null	3
amount		Numeric(18)	Not null	6

Таблица 10 – Реляционная таблица «Recipes»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
recipeid	Primary key	Numeric(18)	Not null	4
authorid		Numeric(18)	Not null	3
categoryid		Numeric(18)	Not null	3
name		Nvarchar(50)		Салат Мимоза
shortname		Nvarchar(50)	Not null	Мимоза
date		Datetime	Not null	01/12/2014

Таблица 11 – Реляционная таблица «Categoryes»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
categoryid	Primary key	Numeric(18)	Not null	3
name		Nvarchar(50)	Not null	Первые блюда

Таблица 12 – Реляционная таблица «Elements»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
elementid	Primary key	Numeric(18)	Not null	4
unitid		Numeric(18)	Not null	3
name		Nvarchar(50)		Капуста белокочанная
shortname		Nvarchar(50)	Not null	Капуста
calories		Numeric(18)	Not null	245

Таблица 13 – Реляционная таблица «Units»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
unitid	Primary key	Numeric(18)	Not null	4
shortname		Nvarchar(50)	Not null	Гр.

Таблица 14 – Реляционная таблица «Authors»

Имя поля	Ключ	Тип, длина	Обязательность значения	Примеры данных
authorid	Primary key	Numeric(18)	Not null	4
lastname		Nvarchar(50)	Not null	Иванов
firstname		Nvarchar(50)	Not null	Иван
middlename		Nvarchar(50)		Иванович
birthdate		Datetime	Not null	01/12/1982

Для технического описания таблиц на ЯОД использовались следующие обозначения: Primary key – обозначение первичного ключа; Foreign key – обозначение внешнего ключа; Integer – соответствует целому числу; Not null – значение не должно быть нулевым; Null – поле может быть пустым; Varchar (25) – символьный тип, в скобках указывается количество символов; Date – тип дата.

Объекты базы данных созданы путем написания SQL-скриптов. Таблицы, как и любые другие объекты, создаются с помощью команды create, затем указывается тип создаваемого объекта, в данном случае таблица (table). После слова table указывается имя создаваемой таблицы. Названия и типы полей создаваемой таблицы прописываются в круглых скобках, описания отделяются друг от друга запятой. После имени поля описывается его тип. Для создания таблиц использовались следующие типы:

- numeric() – целое число, в скобках указывается количество разрядов;
- datetime – дата/время;
- varchar() – строка символов, в скобках указывается количество символов.

Для полей, значения которых должны быть целого типа ставится условие check(value>0), означающее, что значение должно быть больше нуля.

Если поле таблицы должно быть обязательно заполнено, необходимо указать условие not null.

Чтобы указать, какое поле таблицы будет ключевым необходимо прописать primary key (), в скобках указывается имя ключевого поля. Внешние ключи выделяются с помощью конструкции foreign key (<имя поля>) references <имя внешней таблицы>(<имя поля>). Параметры <имя поля> должны совпадать по типу и названию.

4.2 Реализация требований по безопасности и целостности

Вопросы безопасности и целостности! - еще одна из важнейших сторон работы СУБД. Под безопасностью понимают защиту БД от несанкционированного разрушения, изменения и модификации. Систему можно считать безопасной только в том случае, если пользователю допускается выполнять только разрешенные действия. Целостность БД связана с корректным выполнением действий пользователя в ходе добавления, удаления и изменения данных (не должны появиться аномалии в данных).

4.2.1 Разграничение прав доступа пользователей

Для разграничения прав доступа пользователей БД описываются объекты, называемые ролями. В приведенном выше примере с целью организации разграниченного доступа, в базе данных созданы пользовательские роли для менеджера – маркетолога и директора ресторана (рисунки 28 и 29).

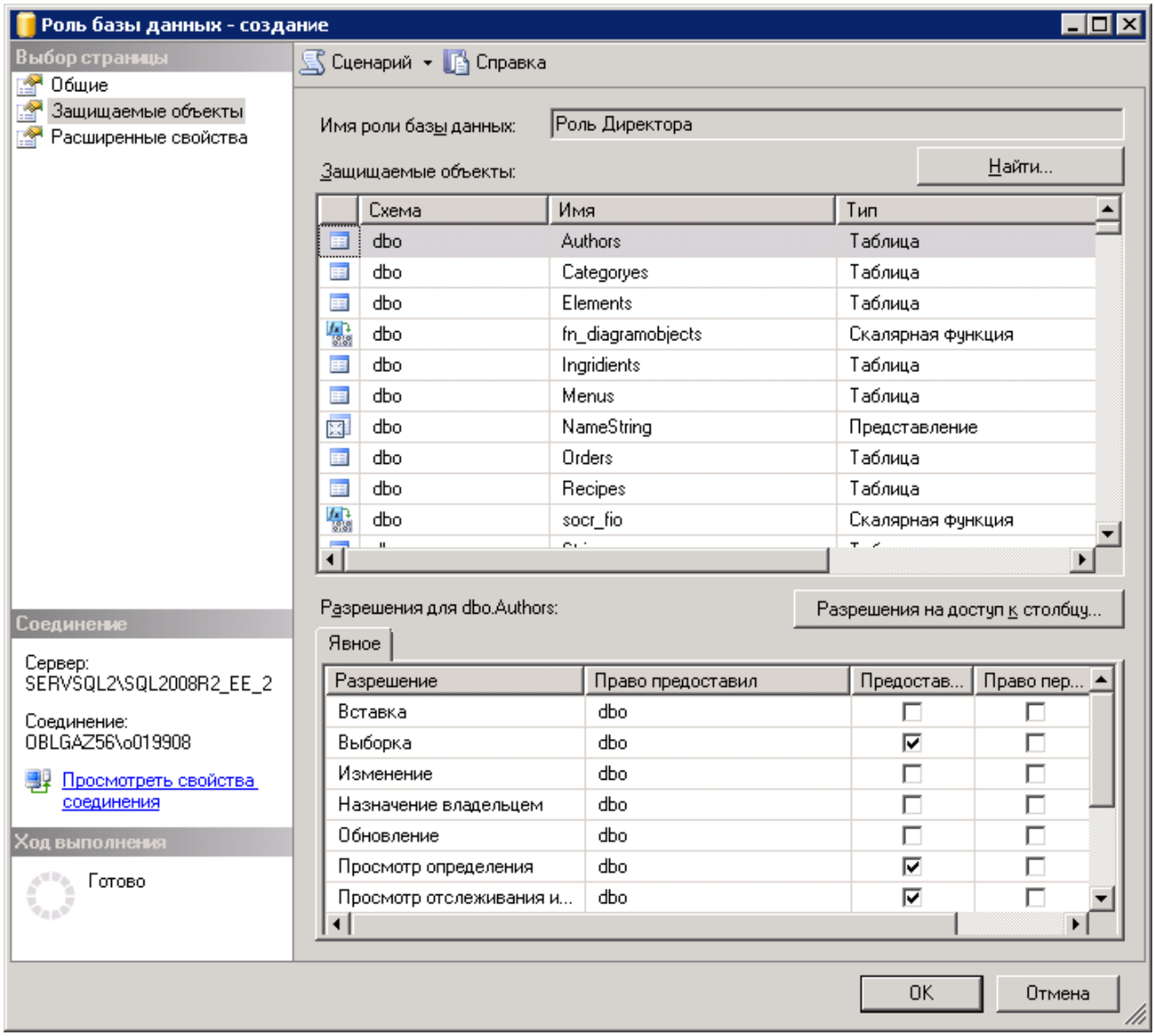


Рисунок 28 – Диалоговое окно добавление роли директора

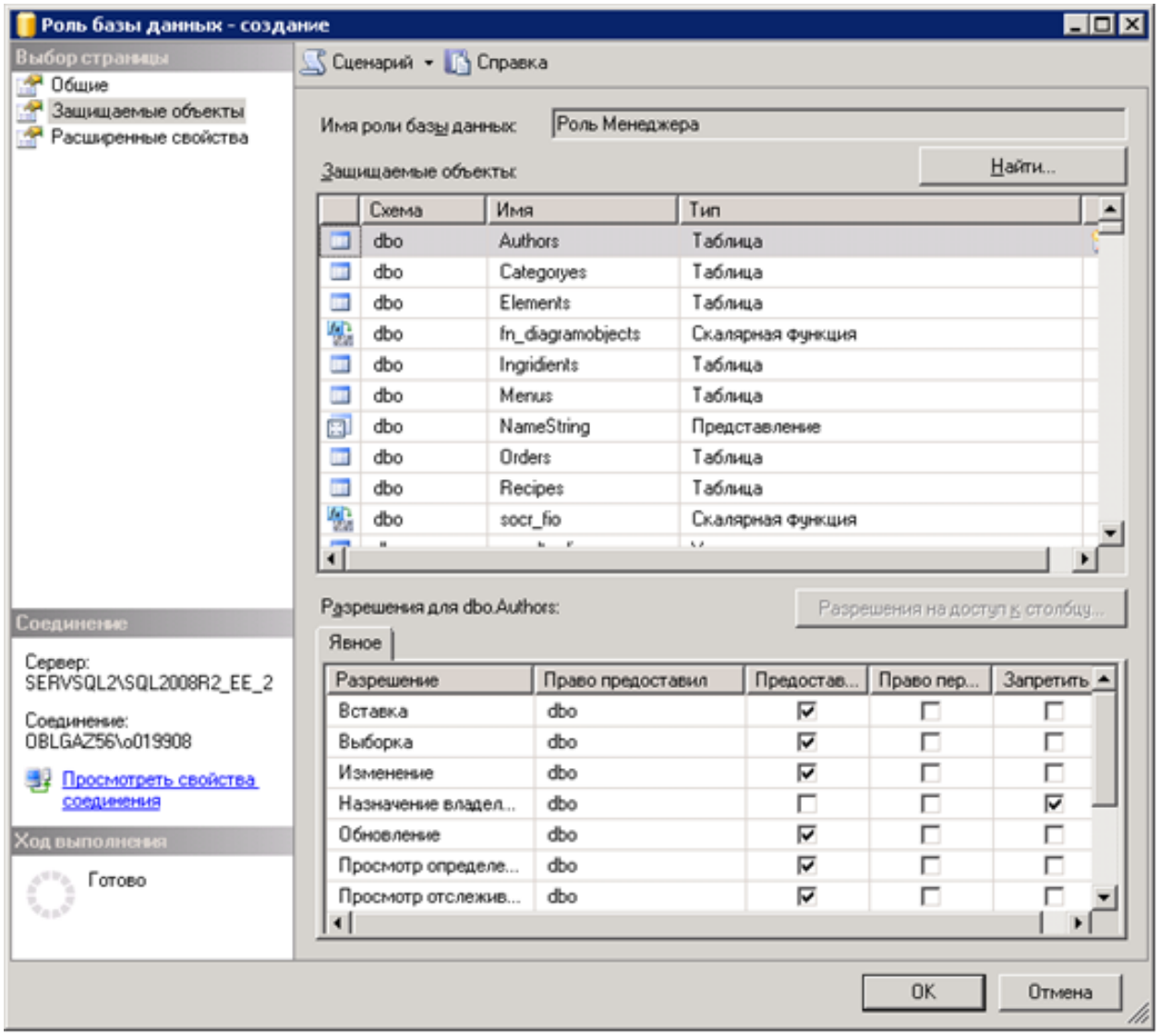


Рисунок 29 – Диалоговое окно добавление роли директора

После добавления ролей, в базе данных были созданы учетные записи пользователей с назначением соответствующих ролей (рисунок 30).

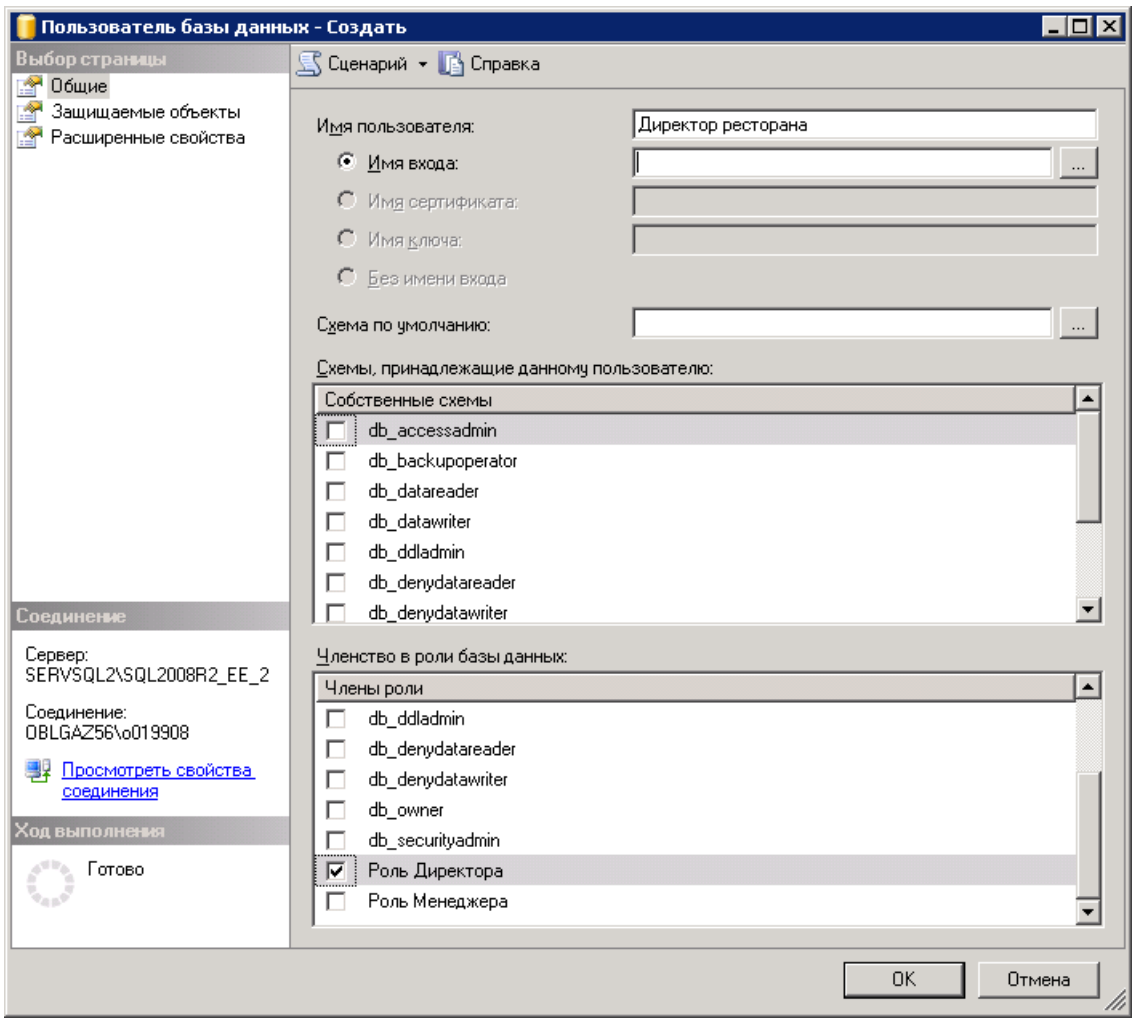


Рисунок 30 – Создание пользователя и назначение роли

4.2.2 Обеспечение целостности данных

Для поддержания целостности данных в базе используется особый инструмент SQL-сервера, называемый триггером. На практике триггеры представляют собой SQL-команды, которые могут выполняться при определенных действиях в таблице базы данных, например, добавлении или удалении записей.

Рассмотрим методику разработки триггеров.

Триггер – это набор инструкций SQLServer, который выглядит и действует подобно хранимой процедуре, но триггер нельзя вызвать с помощью команды EXEC. Триггеры активизируются при выполнении пользователем определенной инструкции Transact-SQL.

Команда ROLLBACK указывает серверу остановить обработку модификации и запретить транзакцию. Существует также команда RAISEERROR с помощью которой вы можете отправить сообщение об ошибке пользователю.

Триггеры INSERT запускаются при каждой попытке создать новую запись в таблице с помощью команды INSERT. При попытке вставить новую запись в таблицу SQLServer копирует эту запись в таблицу триггеров БД и специальную таблицу inserted.

Пример: допустим у нас есть таблица «Должники» БД “Оплата за междугородние разговоры…”. Необходимо запретить вставлять запись о должнике размер задолжности которого, менее 200 рублей.

Решение:

1. Открыть SQL Server Management Studio. В окне Object Explorer развернуть папку БД «Оплата за международ. разговоры…” - Программирование – Триггеры баз данных.

2. Щелкаем правой кнопкой мыши на папке Триггеры баз данных и выбираем команду NewTriggers.

3. В нашем случае триггер будет выглядеть следующим образом (рисунок 31):

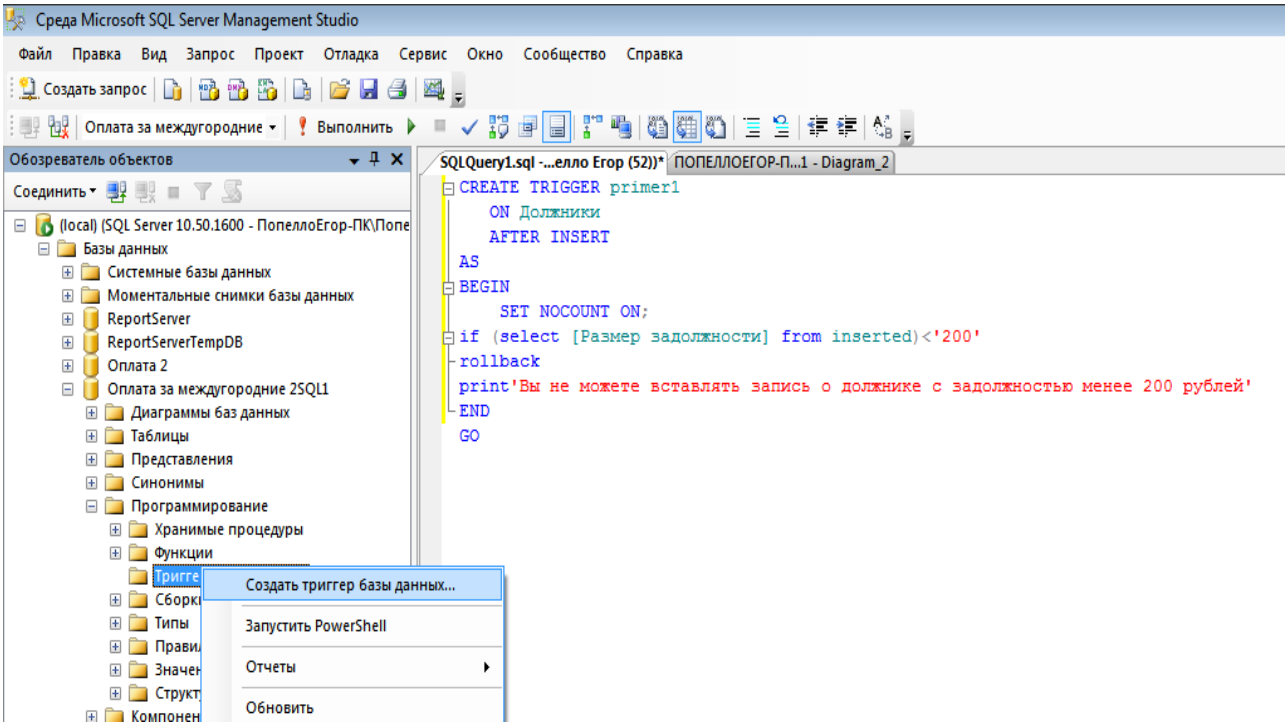


Рисунок 31 – Создание нового триггера

```
CREATE TRIGGER primer1
  ON Должники
  AFTER INSERT
AS
BEGIN
   SET NOCOUNT ON;
if (select размер задолжности from inserted)<'200'
rollback
print'Вы не можете вставлять запись о должнике с размером задолжности менее 200 рублей'
END
GO
```

4. Выполняем триггер, нажав на кнопку  на панели инструментов. В результате в нижней части экрана мы увидим строчку «Выполнение команд успешно завершено».

5. Далее сохраняем триггер (рисунок 32), щелкнув по  кнопке Save.

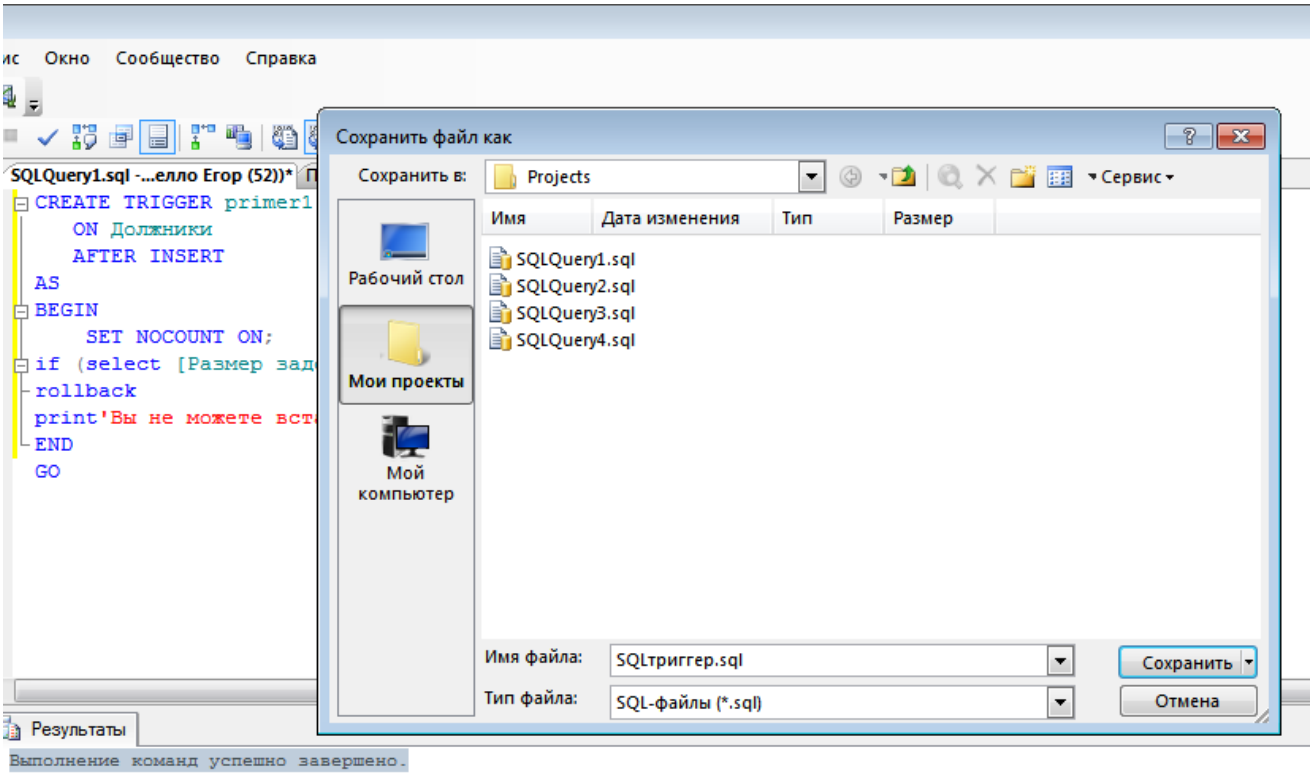
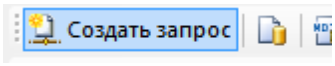
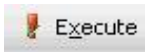


Рисунок 32 – Сохранение триггера

Чтобы протестировать новый триггер, необходимо создать запрос, нажав на кнопку «Создать запрос» 

следующего содержания:

```
USE Оплата за междугородни….
INSERT Должники
VALUES (‘S28’,’Петров’,’Валерий’,’Михайлович’,’200’2’,’1’)
```

Выполняем запрос, щелкнув по кнопке 

В нижней части окна увидим сообщение об ошибке, что такую запись вставлять нельзя.

Итак, мы создали триггер, который при вставке новой записи, проверяет вводимый размер задолжности.

Триггеры DELETE используются для предотвращения удаления данных пользователями из БД. При наличии триггера Delete SQLServer переносит удаляемую запись в логическую таблицу deleted, т.е. записи не исчезают полностью, и вы можете ссылаться на них в коде.

Пример: допустим, у нас есть таблица «Должники» БД “Оплата за междугородн…”. Необходимо запретить удалять запись о должнике задолжность которого более 5000 рублей.

Решение:

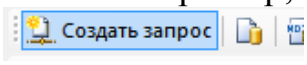
1. Открыть SQL Server Management Studio. В окне Object Explorer развернуть папку БД «Оплата за международ. разговоры…” - Программирование – Триггеры баз данных.

2. Щелкаем правой кнопкой мыши на папке Триггеры баз данных и выбираем команду NewTriggers.

3. В нашем случае триггер будет выглядеть следующим образом:

4. Выполняем триггер, нажав на кнопку  на панели инструментов. В результате в нижней части экрана мы увидим строчку «Выполнение команд успешно завершено».

5. Далее сохраняем триггер, щелкнув по кнопке Save 

Чтобы протестировать новый триггер, необходимо создать запрос, нажав на кнопку «Создать запрос» 

следующего содержания:

USE Оплата за междугородние…..

DELETE Должник
WHERE Размер задолжности=’55000’

Выполняем запрос, щелкнув по кнопке 

В нижней части окна увидим сообщение об ошибке (рисунок 33), что такую запись вставлять нельзя.

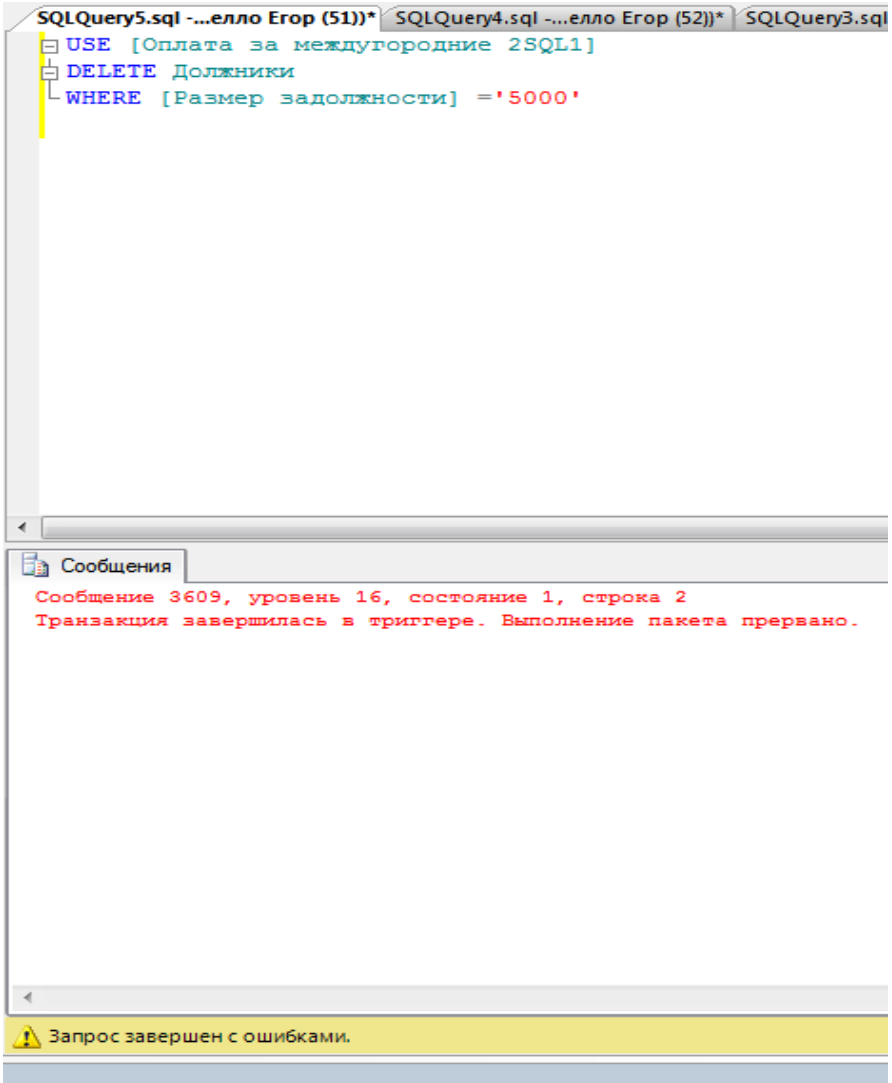


Рисунок 33 – Сообщение об ошибке

Таким образом, мы создали триггер DELETE, который использует логическую таблицу deleted.

Триггер UPDATE используется для ограничения инструкций обновления данных (триггер UPDATE – для перехвата и сверки модификации данных). Триггер UPDATE использует обе таблицы – inserted и deleted. Это объясняется тем, что триггер UPDATE использует два действия - удаление и вставку.

Пример: допустим, у нас есть таблица «Должники» БД “Оплата за междунар. разговоры…”. Необходимо предотвратить вставлять запись с отрицательным размером задолженности.

Решение:

1. Открыть SQL Server Management Studio. В окне Object Explorer развернуть папку БД «Оплата за междунар. разговоры…” - Программирование – Триггеры баз данных.

2. Щелкаем правой кнопкой мыши на папке Триггеры баз данных и выбираем команду NewTriggers.

3. В нашем случае триггер будет выглядеть следующим образом (рисунок 34):

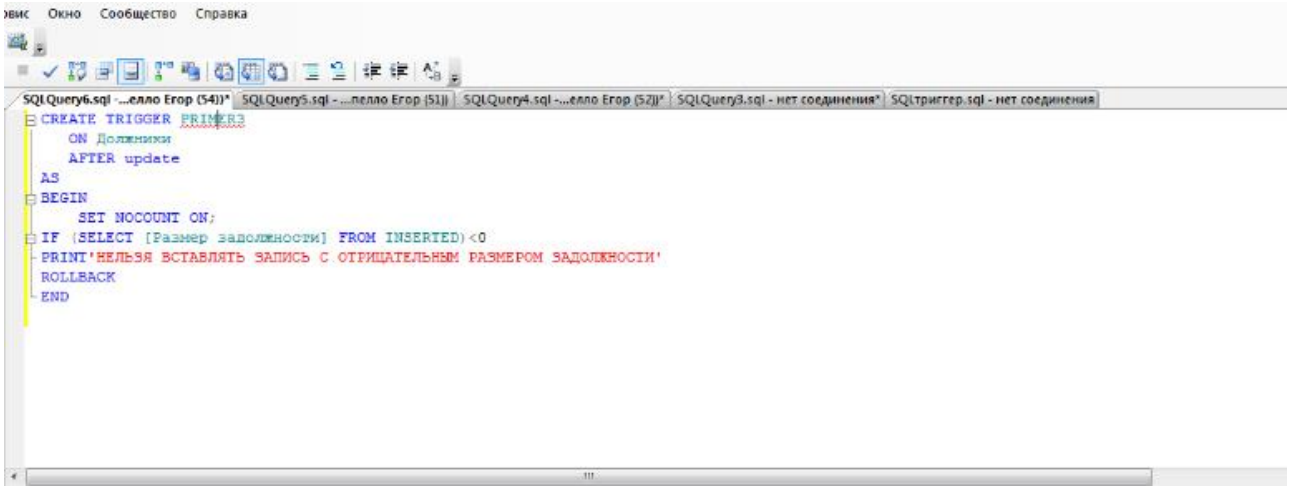
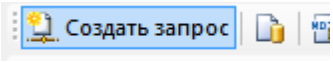


Рисунок 34 – Создание триггера

```
CREATE TRIGGER PRIMER3
  ON Должники
  AFTER update
AS
BEGIN
   SET NOCOUNT ON;
IF (SELECT Размер задолжности FROM INSERTED)<0
PRINT'НЕЛЬЗЯ ВСТАВЛЯТЬ ЗАПИСЬ С ОТРИЦАТЕЛЬНЫМ РАЗМЕРОМ ЗАДОЛЖЕННОСТИ'
ROLLBACK
END
```

4. Выполняем триггер, нажав на кнопку  на панели инструментов. В результате в нижней части экрана мы увидим строчку «Выполнение команд успешно завершено».

5. Далее сохраняем триггер, щелкнув по кнопке Save 

Чтобы протестировать новый триггер, необходимо создать запрос, нажав на кнопку «Создать запрос»  следующего содержания

```
USE Оплата за между…
UPDATE Должники
SET Размер задолжности=’-12000’
WHERE ФИО клиента=’Петров’
```

Выполняем запрос, щелкнув по кнопке  . В нижней части окна увидим сообщение об ошибке (рисунок 35).

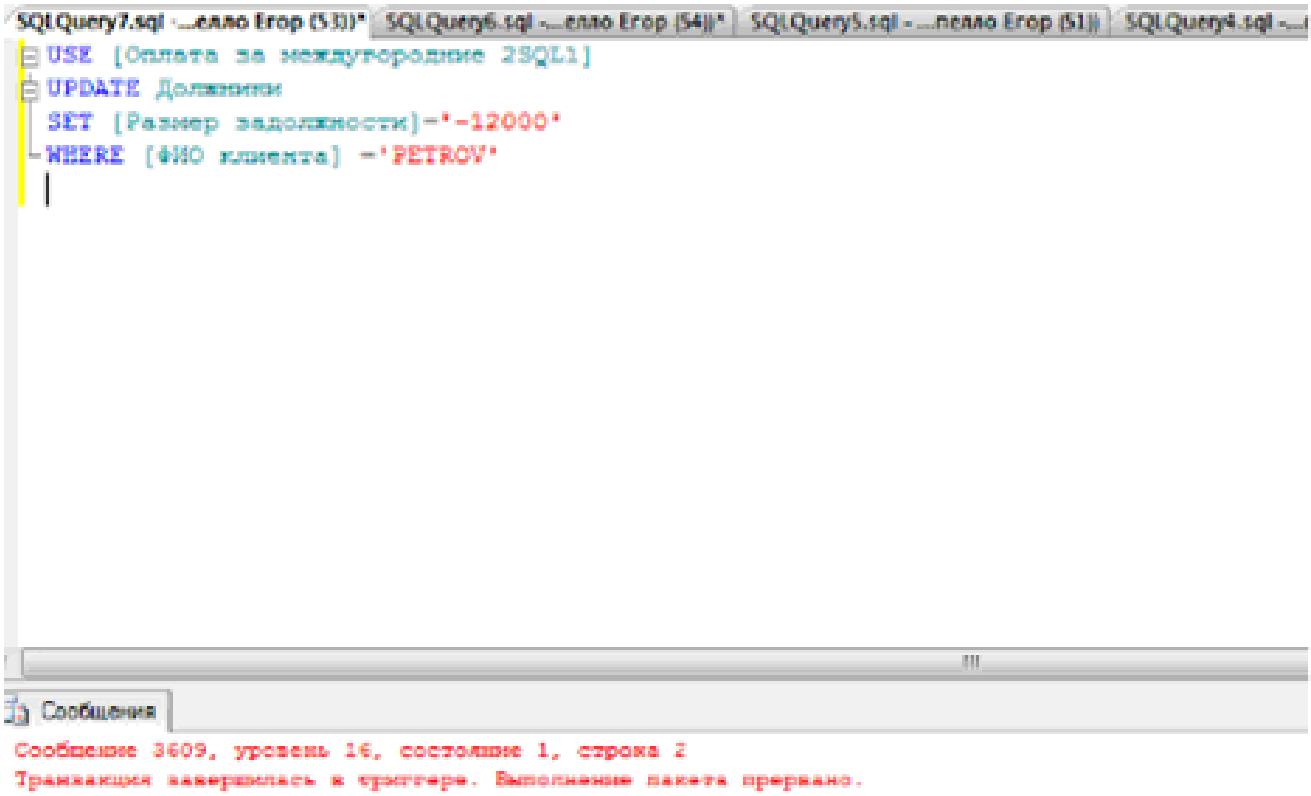


Рисунок 35 – Сообщение об ошибке

Итак, данный триггер UPDAT E запрещает вставлять запись о должнике с отрицательным размером задолжности

Часто возникают ситуации, когда обновление большинства столбцов не интересует, но существует один столбец, который нельзя изменять ни в коем случае.

Пример: допустим, у нас есть таблица «Клиенты» БД “Оплата за междугор…”. Необходимо запретить изменять столбец, содержащий адрес.

Решение:

1. Открыть SQL Server Management Studio. В окне Object Explorer развернуть папку БД «Оплата за международ. разговоры…” - Программирование – Триггеры баз данных.

2. Щелкаем правой кнопкой мыши на папке Триггеры баз данных и выбираем команду NewTriggers.

3. В нашем случае триггер будет выглядеть следующим образом (рисунок 36):

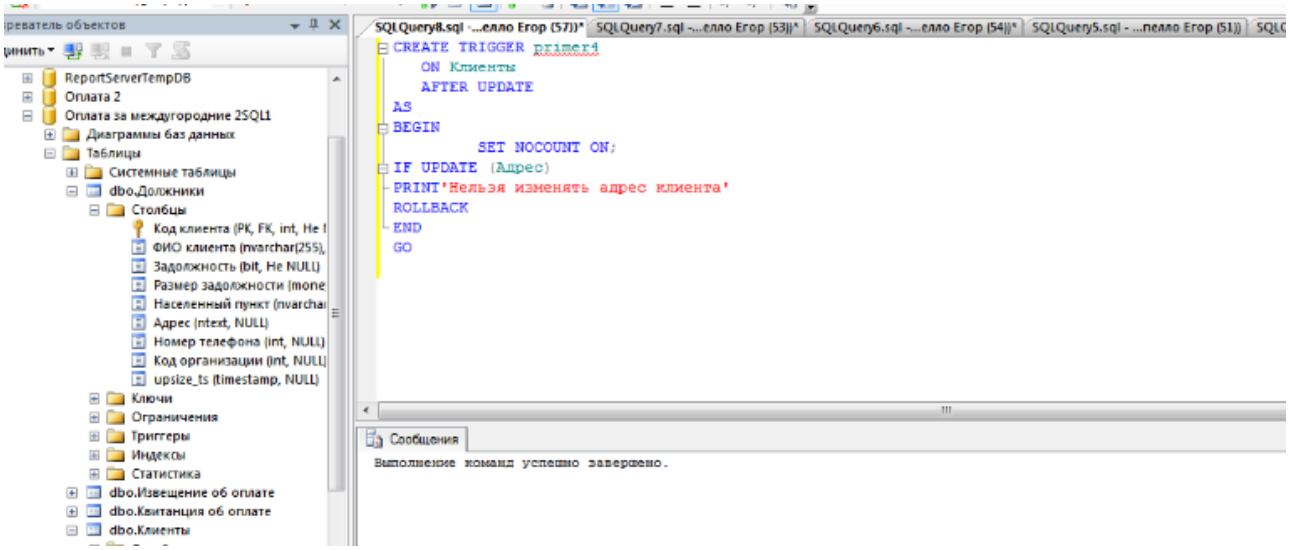
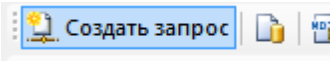


Рисунок 36 – Создание триггера

```
CREATE TRIGGER primer4
  ON Клиенты
  AFTER UPDATE
AS
BEGIN
      SET NOCOUNT ON;
IFUPDATE(Адрес)
PRINT'Нельзя изменять адрес клиента'
ROLLBACK
END
GO
```

4. Выполняем триггер, нажав на кнопку  на панели инструментов. В результате в нижней части экрана мы увидим строчку «Выполнение команд успешно завершено».

5. Далее сохраняем триггер, щелкнув по кнопке Save 

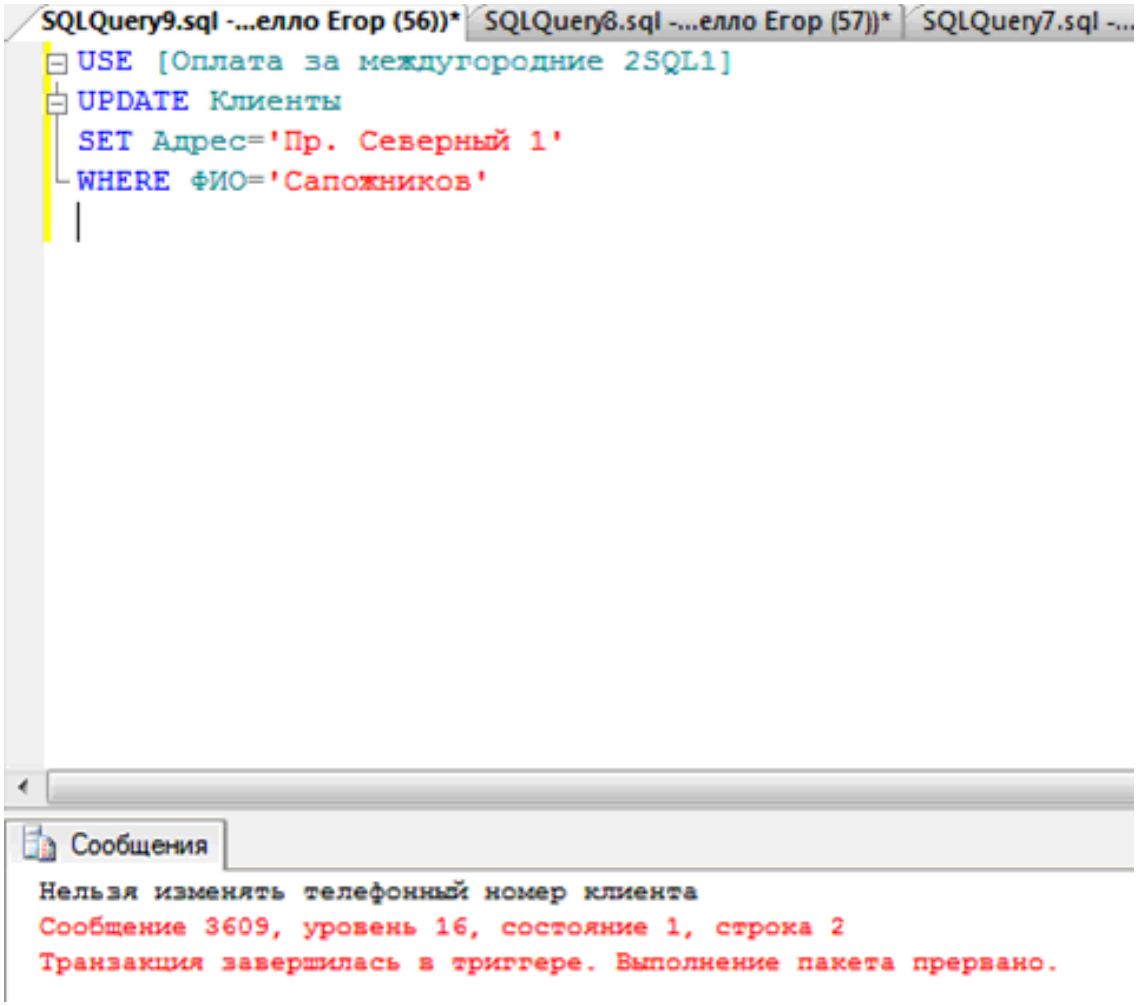
Чтобы протестировать новый триггер, необходимо создать запрос, нажав на кнопку «Создать запрос» 

следующего содержания

```
USE Оплата за междуго….
UPDATE Клиенты
SET Адрес='Пр.Северный 1'
WHERE ФИО='Сапожников'
```

Выполняем запрос, щелкнув по кнопке 

, в нижней части окна увидим сообщение об ошибке.



Нельзя изменять адрес клиенту.

Триггеры DDL применяются для инструкций CREATE, ALTER, DROP и т.д.

Пример: создадим триггер, который предотвратит удаление или изменение таблиц в БД.

Решение:

1. Создадим запрос, нажав на кнопку New Query ,

следующего содержания (рисунок 37):

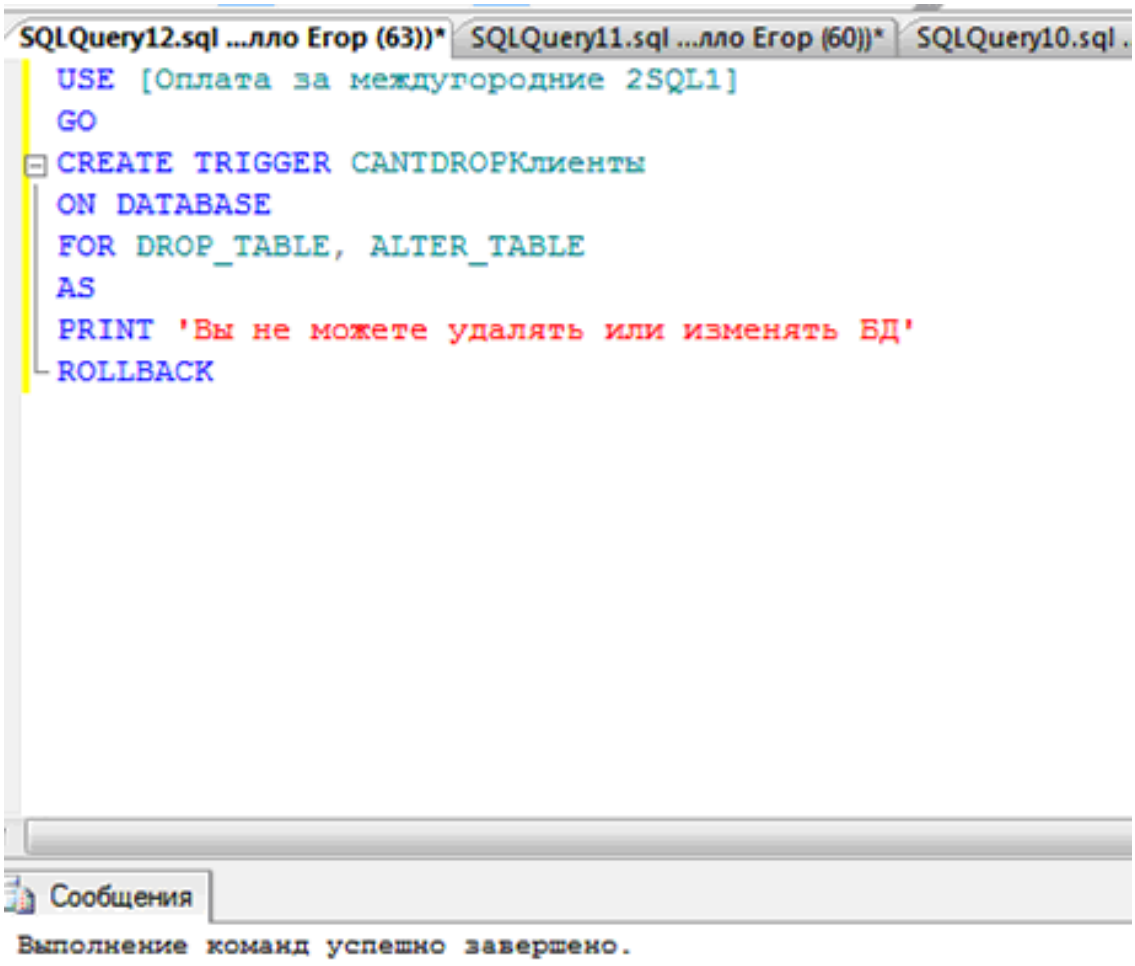


Рисунок 37 – Создание триггера

```
USE Оплата за междугор….
GO
CREATE TRIGGER CANTDROPSOTRYDNIK
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
PRINT ‘Вы не можете удалять или изменять БД’
ROLLBACK
```

2. Проверим созданный триггер. Создадим запрос, нажав на кнопку New Query следующего содержания (рисунок 38):

DROP_TABLES Клиент

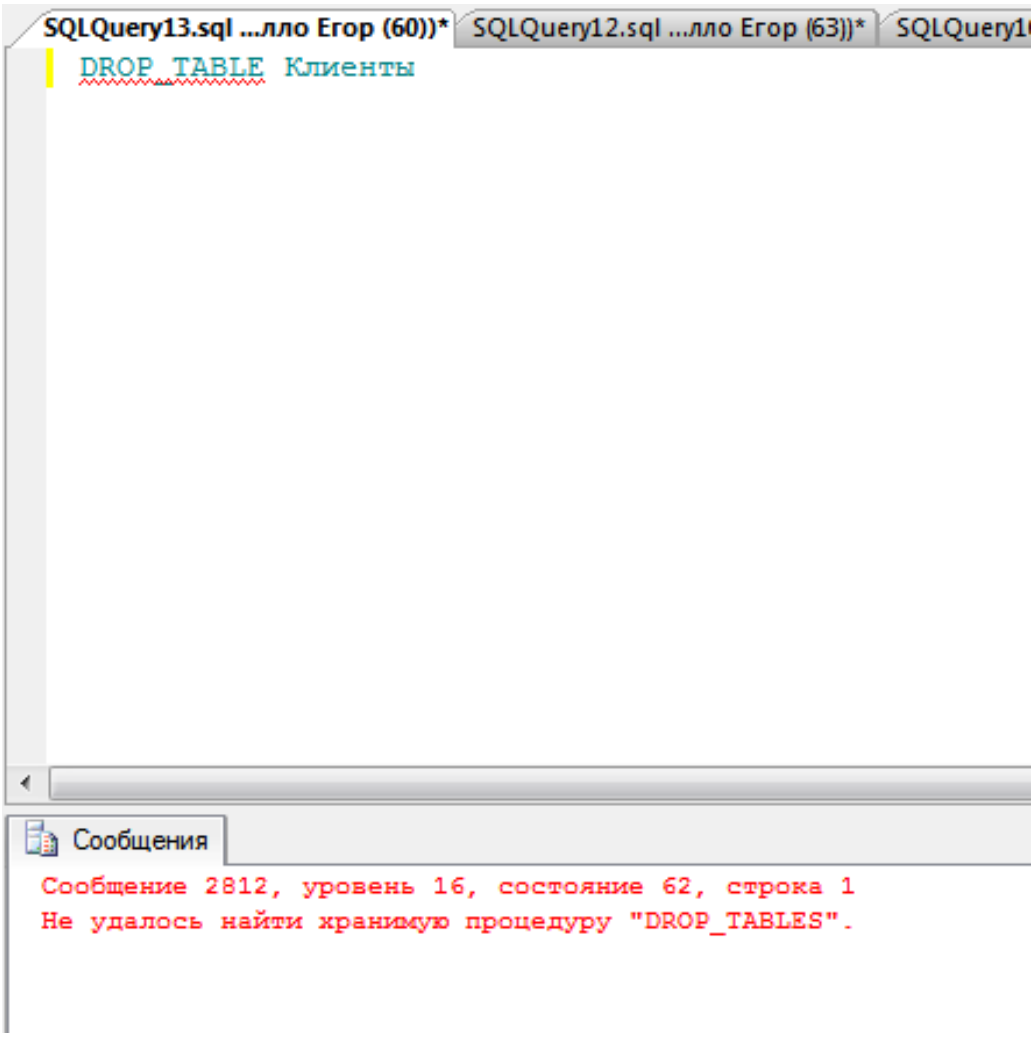


Рисунок 38 – Сообщение об ошибке

3. Вы увидите сообщение об ошибке:

Вы не можете удалять или изменять БД

5 Разработка серверного приложения

Для создания серверного приложения к базе данных в среде разработки Visual Studio 2010 создается решение, в котором необходимо инициализировать новый проект и выбрать тип проекта приложения (рисунки 39 и 40).

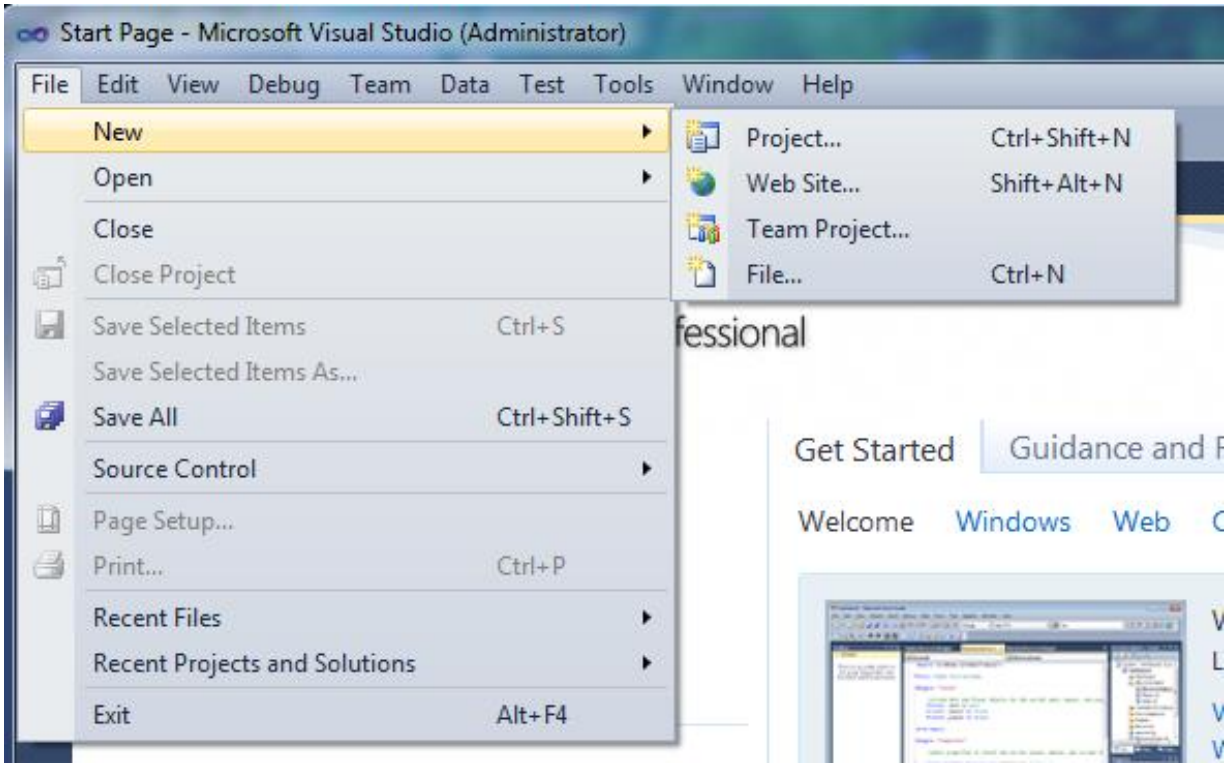


Рисунок 31 – Создание нового проекта

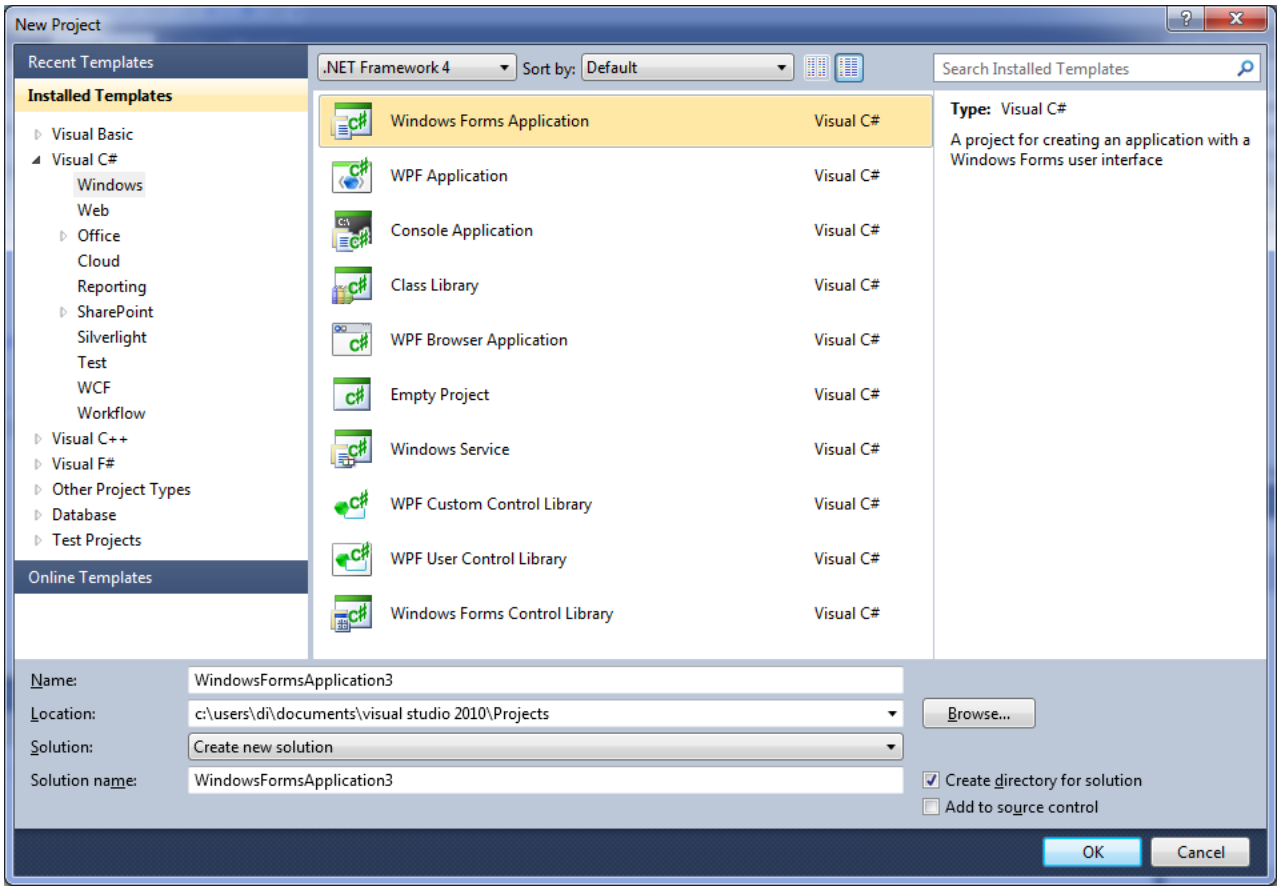


Рисунок 40 – Выбор типа проекта

В новом проекте создаются наборы данных, классы экранных форм и отчетов. Экранные формы приложения разрабатываются в концепции MDI, где в основном рабочем окне содержатся дочерние окна, реализующие взаимодействие пользователя с различными функциональными подсистемами. Общее рабочее окно приложения представлено на рисунке 41.

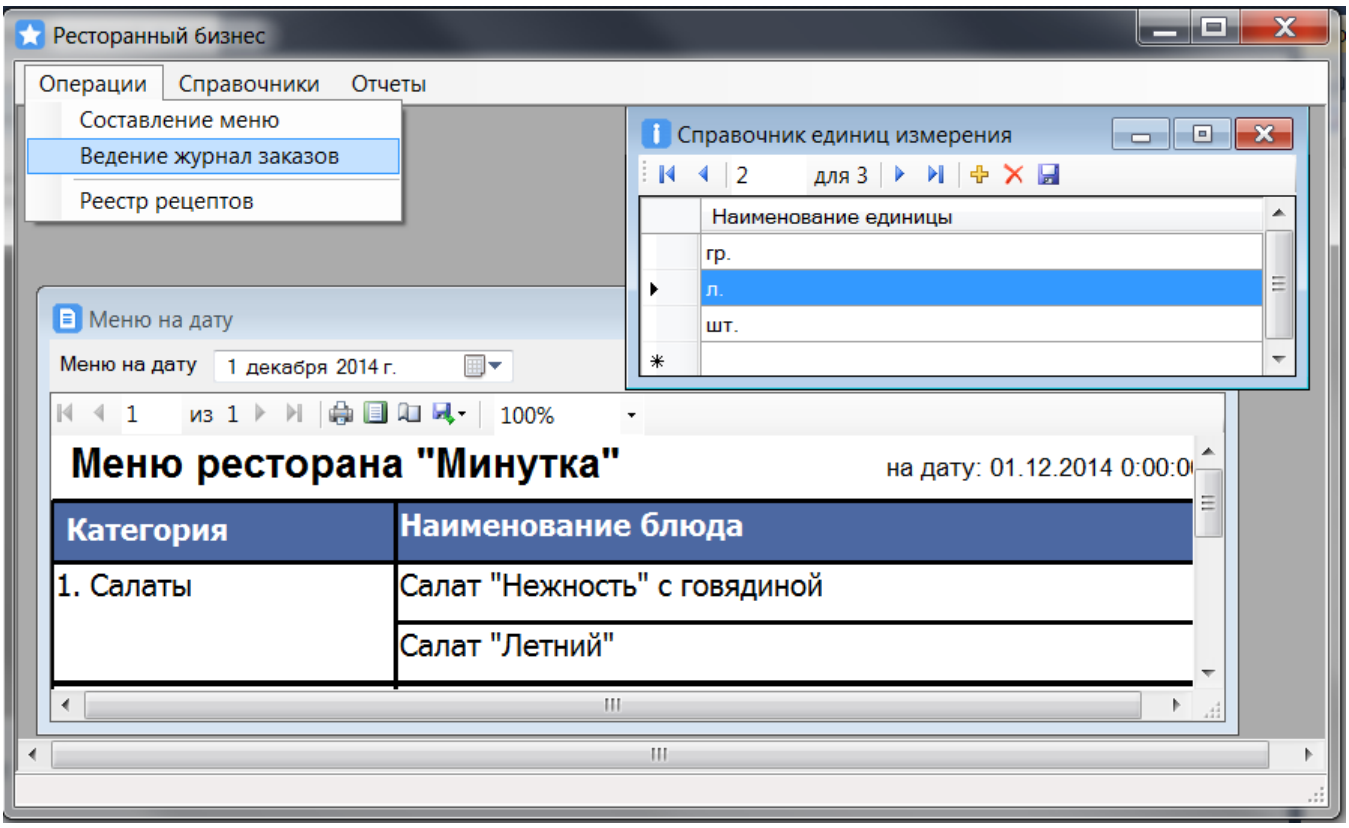


Рисунок 41 – Основное рабочее окно приложения

При начале работы с системой происходит процедура авторизации пользователя. Экранная форма окна входа в систему представлена на рисунке 42.

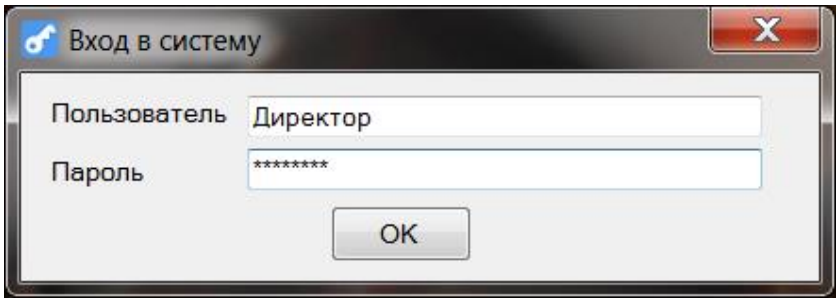


Рисунок 42 – Экранная форма входа в систему

Разработанное приложение содержит функционал работы со справочниками единиц измерения, категорий блюд, авторов рецептов.

Экранные формы справочников представлены на рисунках 43 - 45.

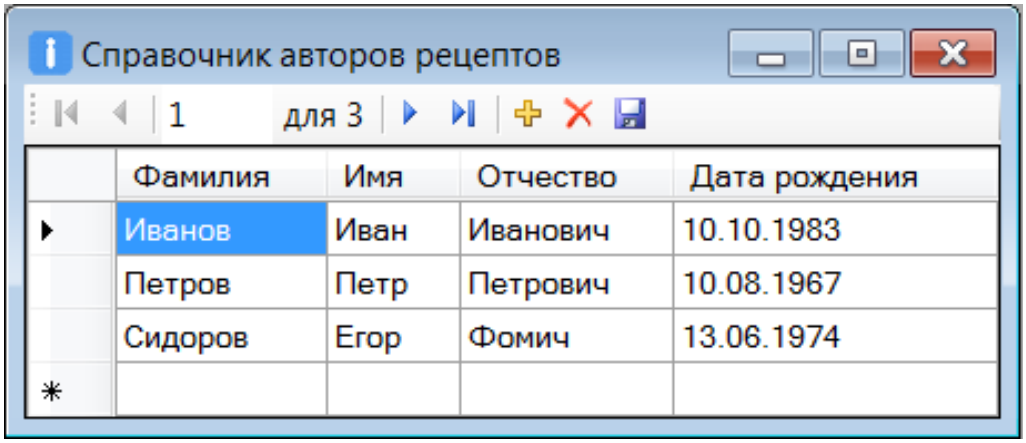


Рисунок 43 – Справочник авторов

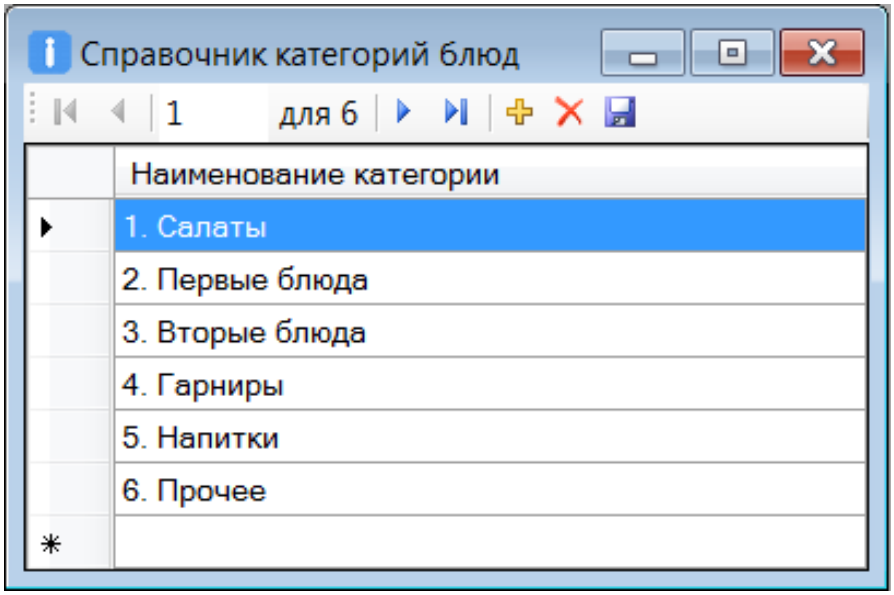


Рисунок 44 – Справочник категорий блюд

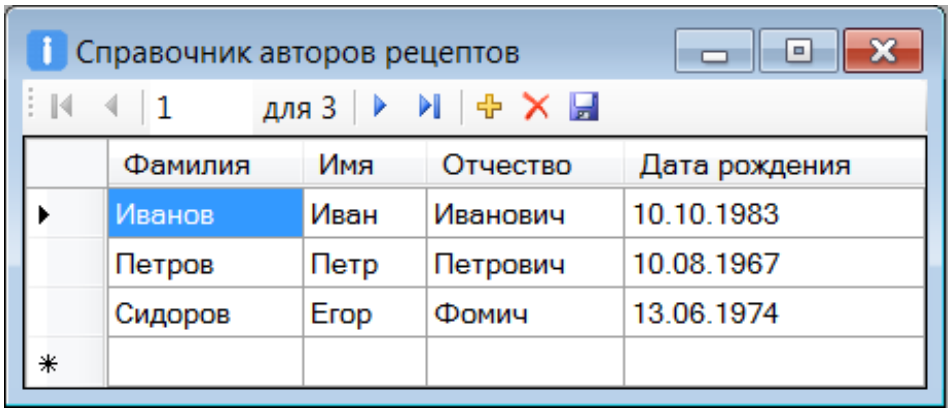


Рисунок 45 – Справочник категорий блюд

Разработанные экранные формы реестров и журнала заказов представлены на рисунках 46 - 48.

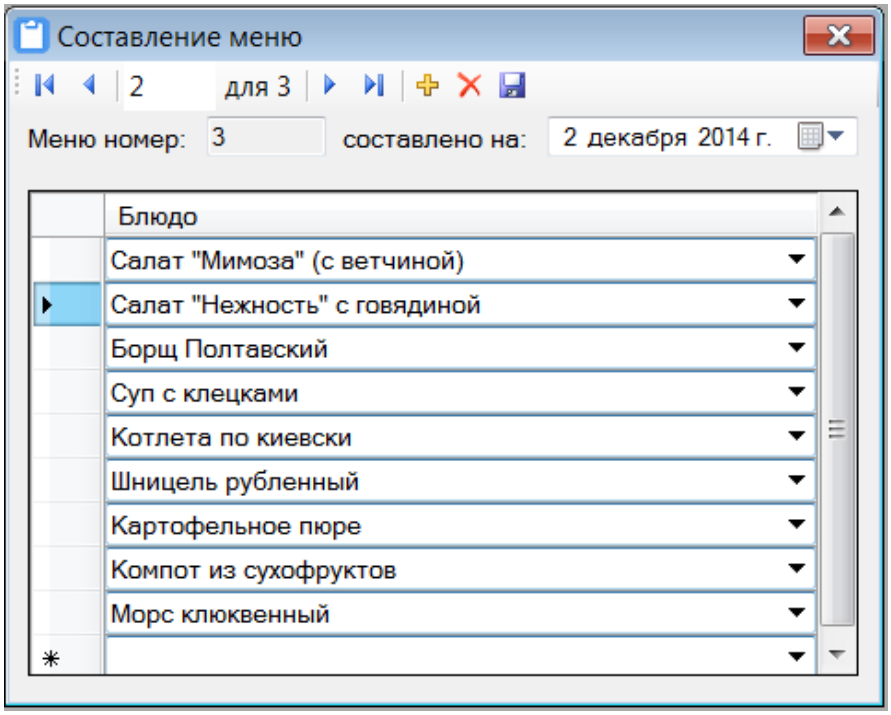


Рисунок 46 – Реестр меню

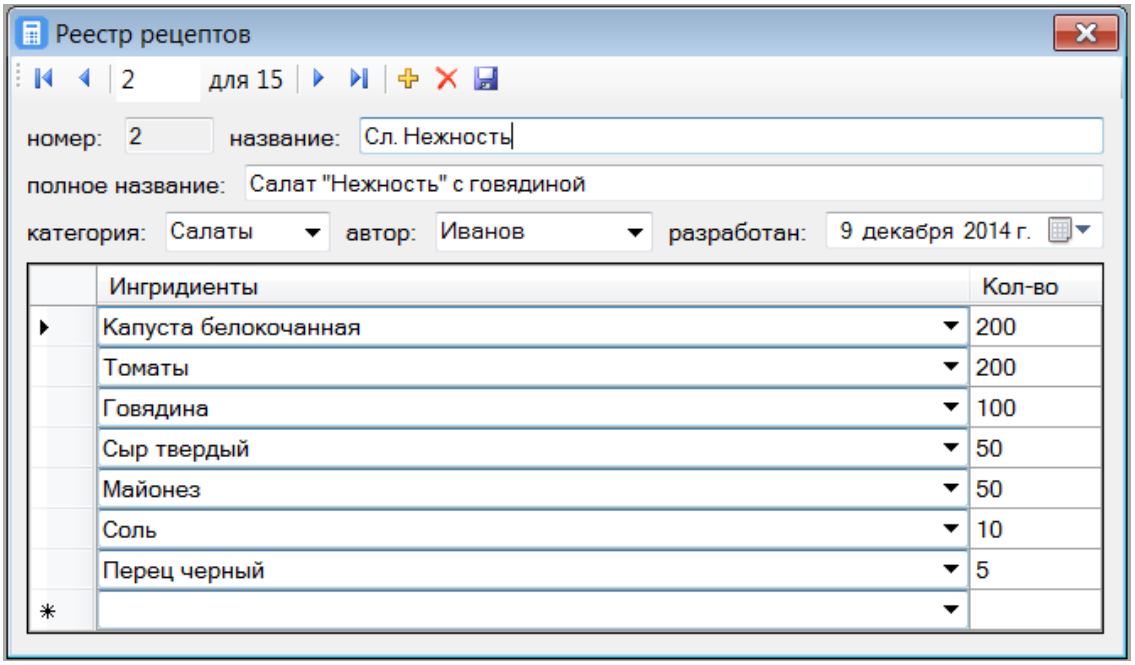


Рисунок 47 – Реестр рецептов

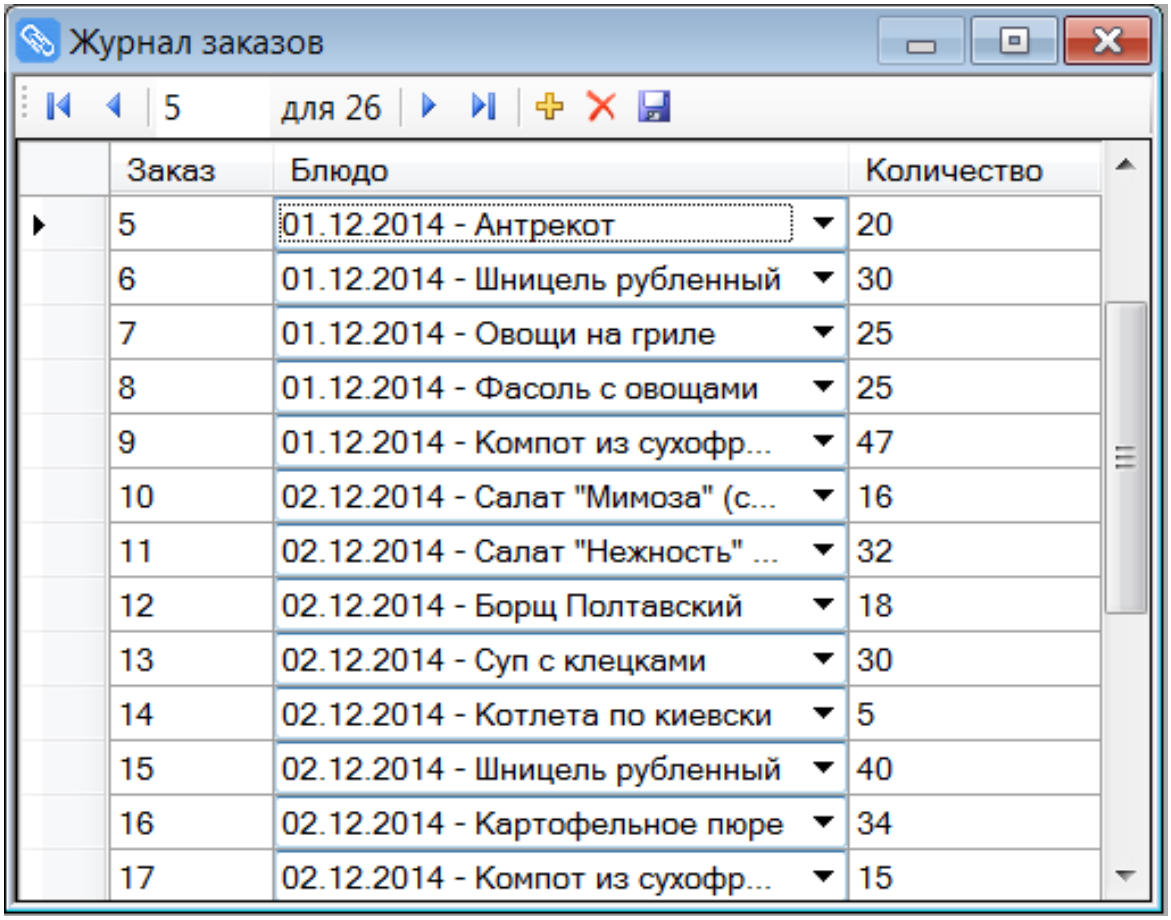


Рисунок 48 – Журнал заказов

Итак, мы рассмотрели пример разработки серверного приложения РИС, которое в основном предназначено для автоматизации документооборота и ведения справочной информации. Кроме того, серверное приложение должно обеспечивать для пользователей выполнение запросов информации по интересующим параметрам.

Рассмотрим методику разработки параметрических запросов с помощью хранимых процедур.

Хранимые процедуры— это группа из одной или нескольких инструкций Transact-SQL или ссылка на метод Microsoft .NET Framework среды CLR. Процедуры аналогичны конструкциям в других языках программирования, поскольку обеспечивают следующее: обрабатывают входные параметры и возвращают вызывающей программе значения в виде выходных параметров; содержат программные инструкции, которые выполняют операции в базе данных, включая вызов других про-

цедур; возвращают значение состояния вызывающей программе, таким образом передавая сведения об успешном или неуспешном завершении (и причины последнего).

Рассмотрим методику создания хранимой процедуры в среде SQL Server Management Studio.

1) В обозревателе объектов подключитесь к экземпляру компонента Компонент Database Engine и разверните его.

2) Последовательно разверните узел Базы данных, базу данных AdventureWorks2012 и узел Программирование.

3) Щелкните правой кнопкой мыши элемент Хранимые процедуры и выберите пункт Создать хранимую процедуру (рисунок 49).

4) В меню Запрос выберите пункт Задание значений для параметров шаблона (рисунок 50).

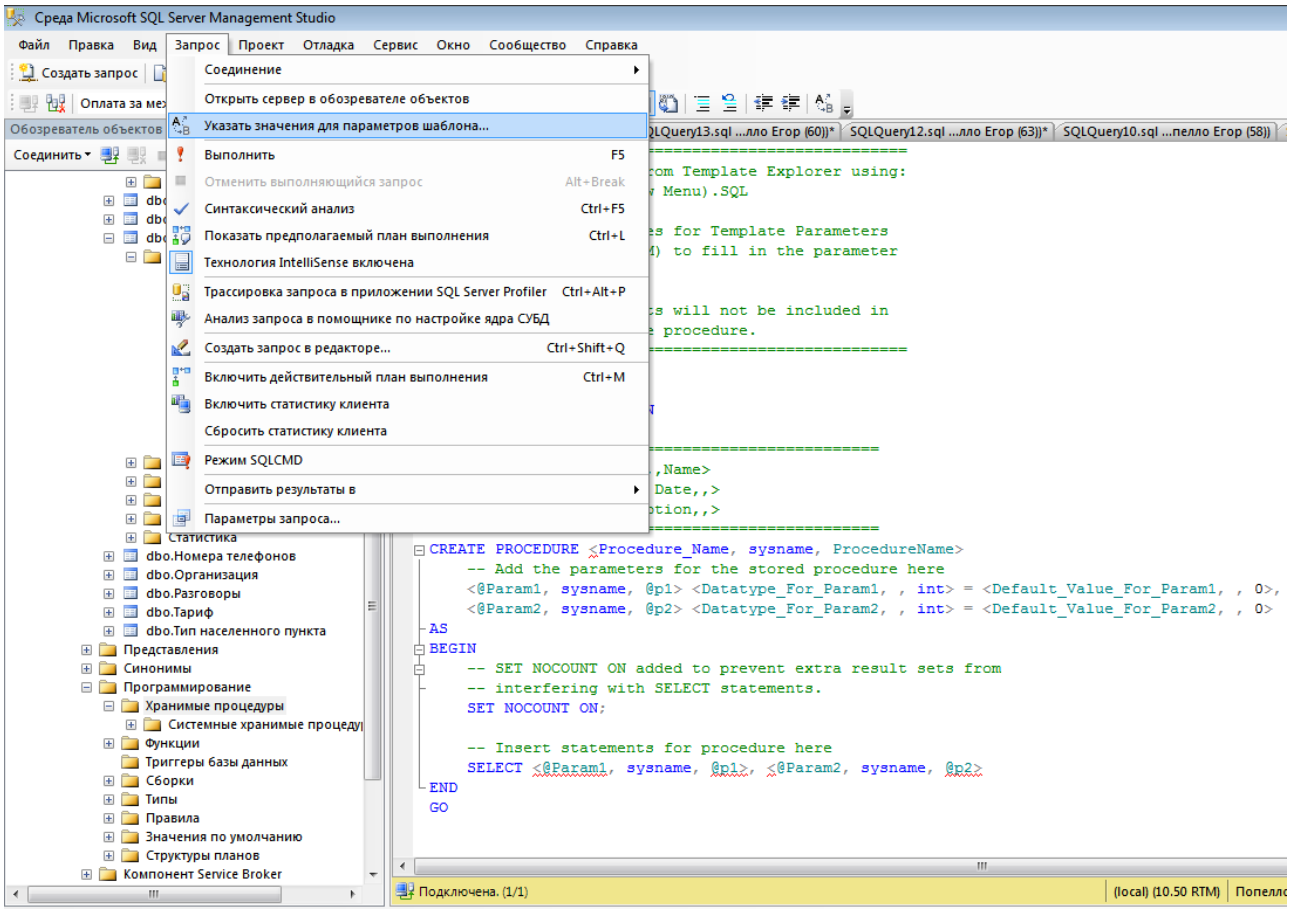


Рисунок 49 - Задание параметров шаблона

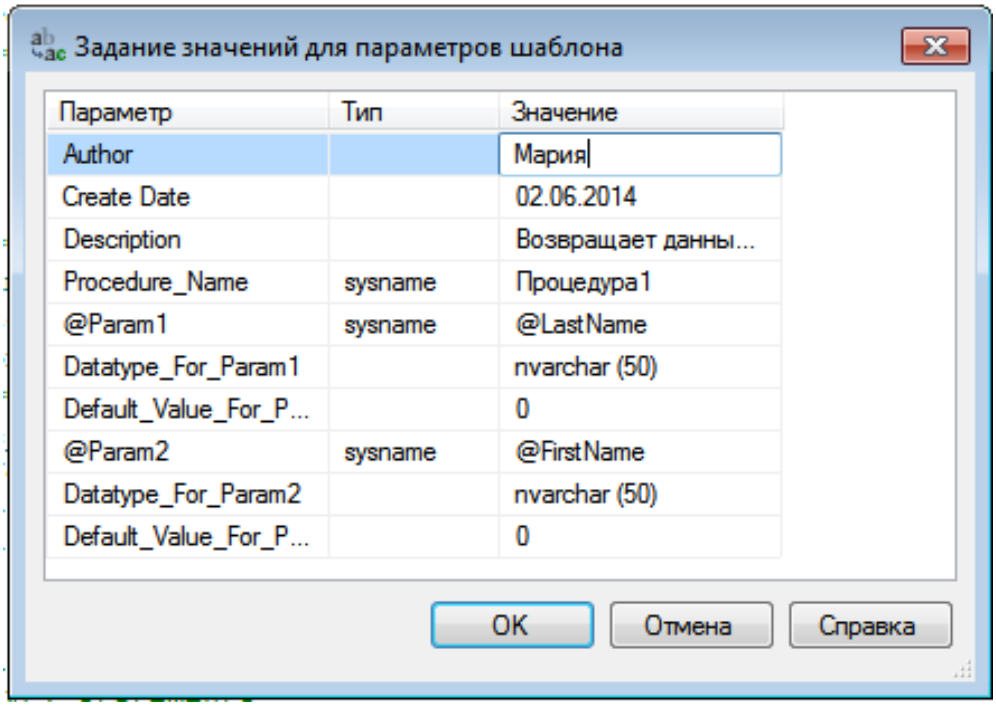


Рисунок 50 - Задание параметров шаблона

5) В диалоговом окне Задание значений для параметров шаблона введите для показанных параметров следующие значения.

6) Нажмите кнопку ОК.

7) В редакторе запросов замените инструкцию SELECT следующей инструкцией (рисунок 51):

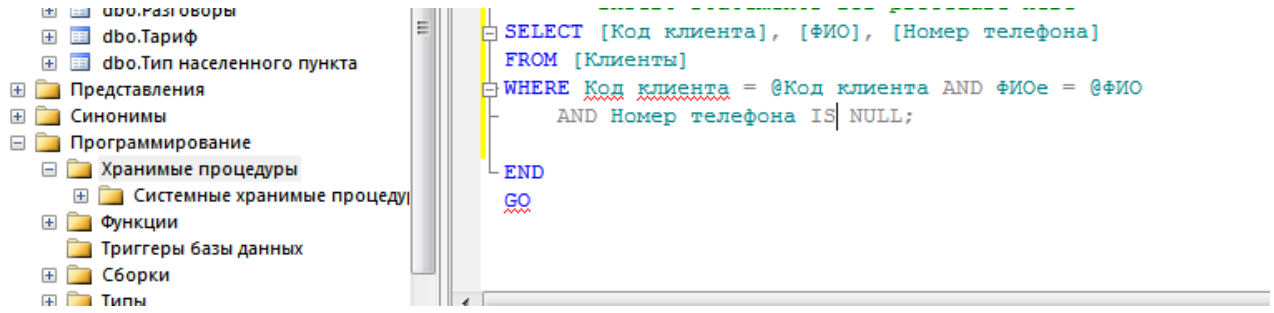
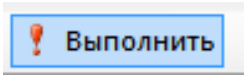


Рисунок 51 – Создание инструкции

```
Transact-SQL
SELECT Код клиента,ФИО, Номер телефона
FROM Клиенты
WHERE Код клиента = @Код клиентаAND ФИО= @ФИО
   AND Номер телефона IS NULL;
```

8) Для проверки синтаксиса выберите пункт Синтаксический анализ в меню Запрос. Если возвращается сообщение об ошибке, сравните инструкции с приведенными выше и при необходимости внесите исправления.

9) Чтобы создать процедуру, в меню Запрос выберите пункт Выполнить. Процедура создается как объект в базе данных.



10) Чтобы увидеть процедуру в обозревателе объектов, щелкните правой кнопкой мыши элемент Хранимые процедуры и выберите пункт Обновить (рисунок 52).

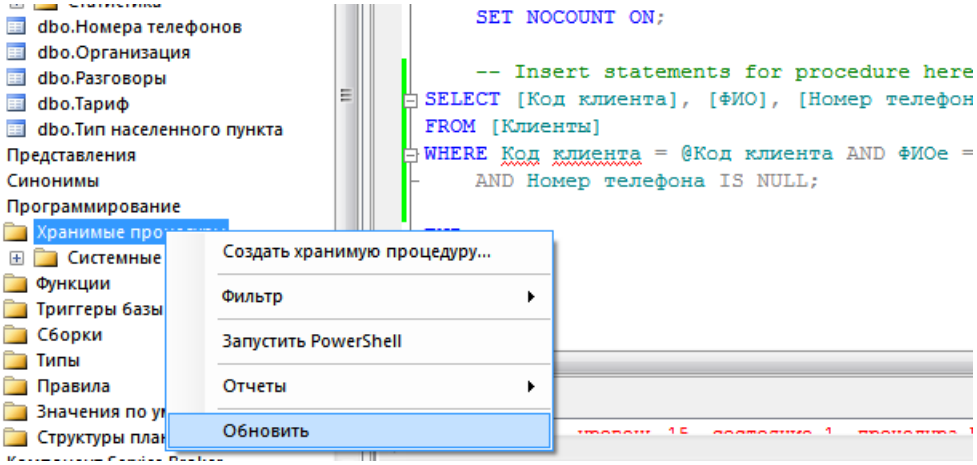


Рисунок 52 – Обновление хранимой процедуры

Чтобы выполнить процедуру, в обозревателе объектов щелкните правой кнопкой мыши имя хранимой процедуры Процедура 1 и выберите пункт Выполнение хранимой процедуры.

Внимание!

Проверяйте все данные, вводимые пользователем. Не включайте их в сценарий, не выполнив проверку. Никогда не выполняйте команду, построенную на основании непроверенных пользовательских входных данных.

6 Разработка клиентских приложений

Выходная информация на клиентских приложениях может быть представлена отчетами. Экранные формы отчетов представлены на рисунках 53 - 56.

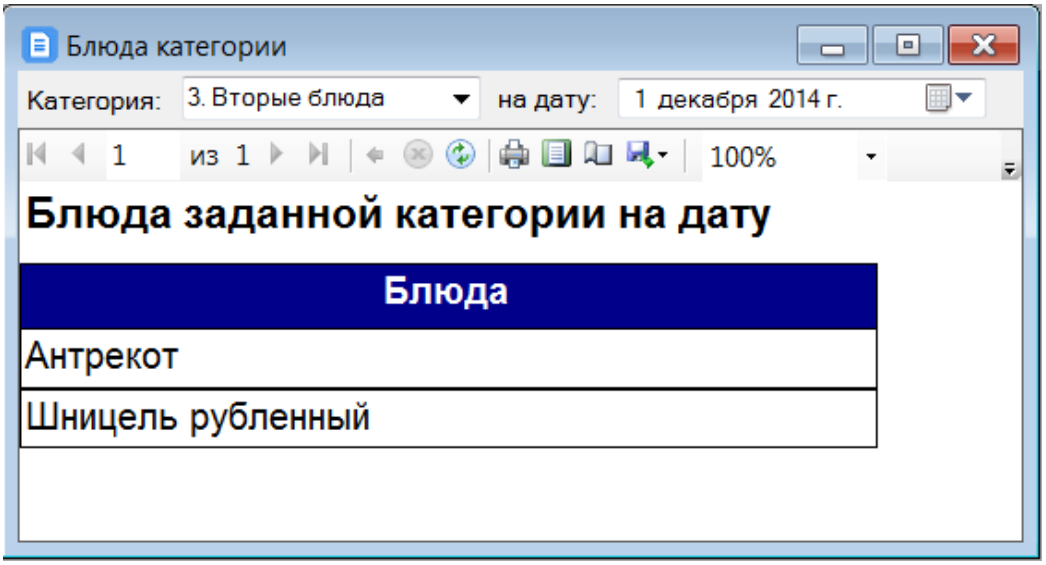
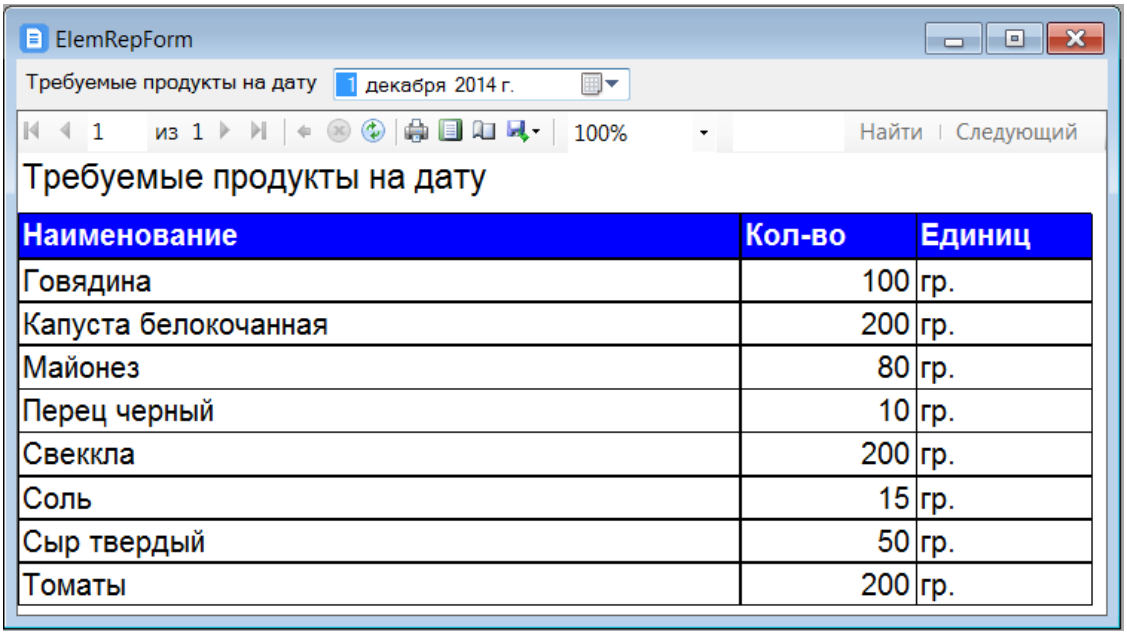
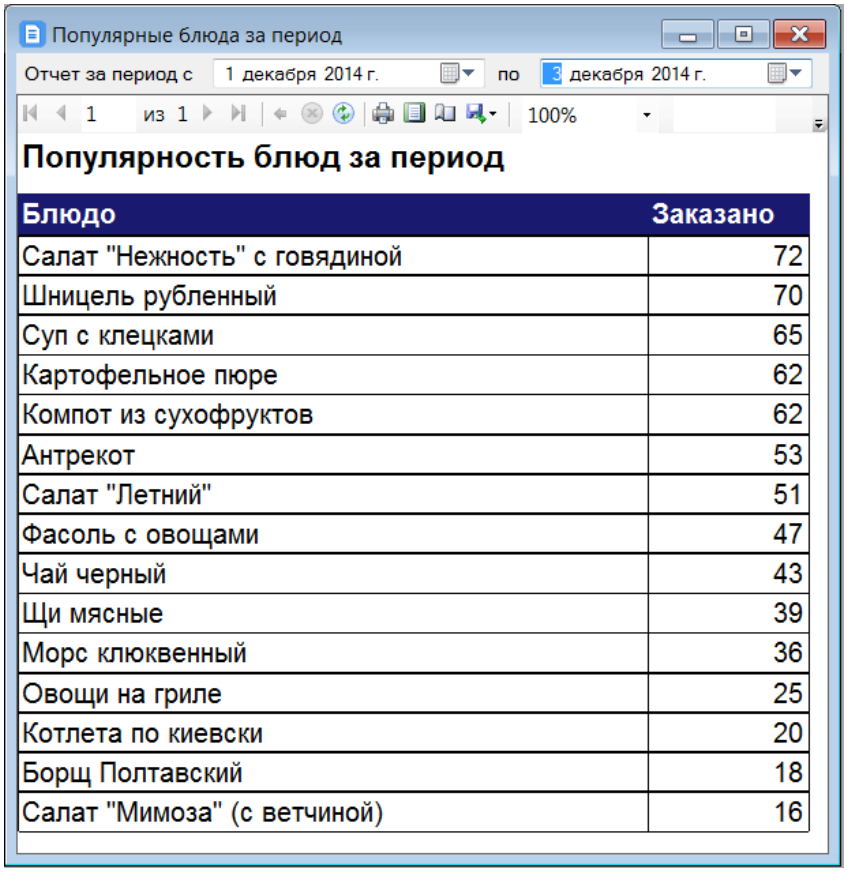


Рисунок 53 – Отчет «Блюда заданной категории на дату»



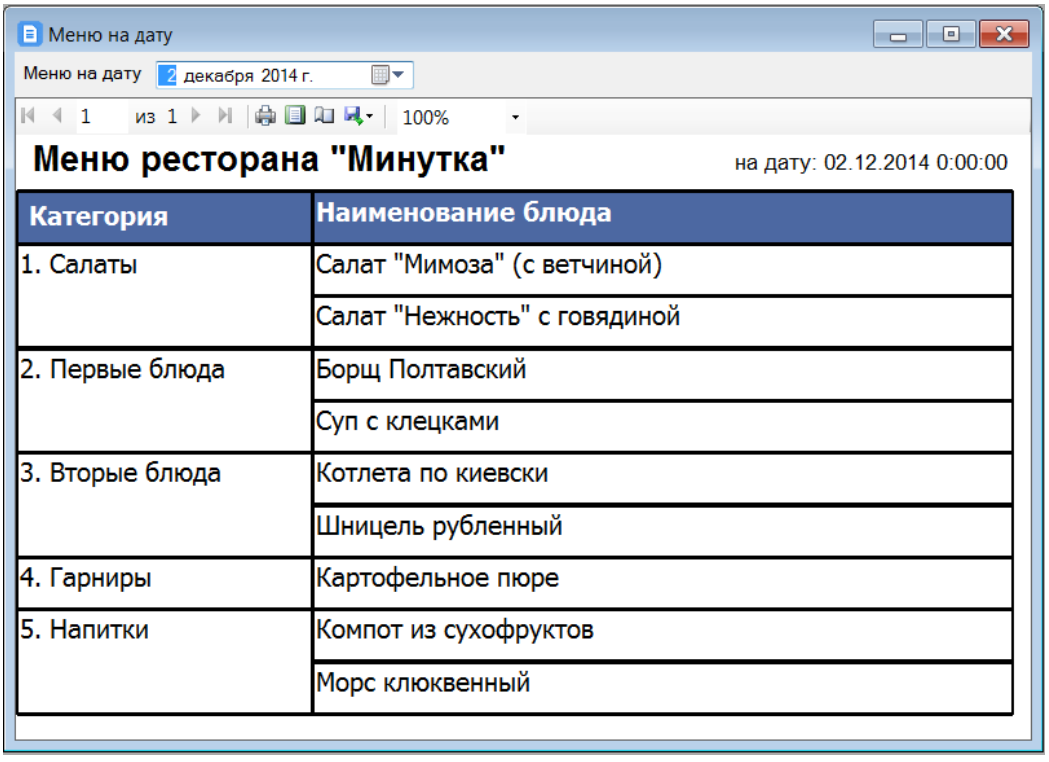
Наименование	Кол-во	Единиц
Говядина	100	гр.
Капуста белокочанная	200	гр.
Майонез	80	гр.
Перец черный	10	гр.
Свеккла	200	гр.
Соль	15	гр.
Сыр твердый	50	гр.
Томаты	200	гр.

Рисунок 54 – Отчет «Требуемые продукты на дату»



Блюдо	Заказано
Салат "Нежность" с говядиной	72
Шницель рубленный	70
Суп с клецками	65
Картофельное пюре	62
Компот из сухофруктов	62
Антрекот	53
Салат "Летний"	51
Фасоль с овощами	47
Чай черный	43
Щи мясные	39
Морс клюквенный	36
Овощи на гриле	25
Котлета по киевски	20
Борщ Полтавский	18
Салат "Мимоза" (с ветчиной)	16

Рисунок 55 – Отчет «Популярность блюд за период»



Категория	Наименование блюда
1. Салаты	Салат "Мимоза" (с ветчиной)
	Салат "Нежность" с говядиной
2. Первые блюда	Борщ Полтавский
	Суп с клецками
3. Вторые блюда	Котлета по киевски
	Шницель рубленный
4. Гарниры	Картофельное пюре
5. Напитки	Компот из сухофруктов
	Морс клюквенный

Рисунок 56 – Отчет «Меню на заданную дату»

Рассмотрим методику разработки SQL-запросов с помощью функций пользователей.

Для создания запроса необходимо в обозревателе серверов кликнуть правой кнопкой мыши по пункту «Функции» и выбрать Добавить новый - Подставляемая функция (рисунок 57).

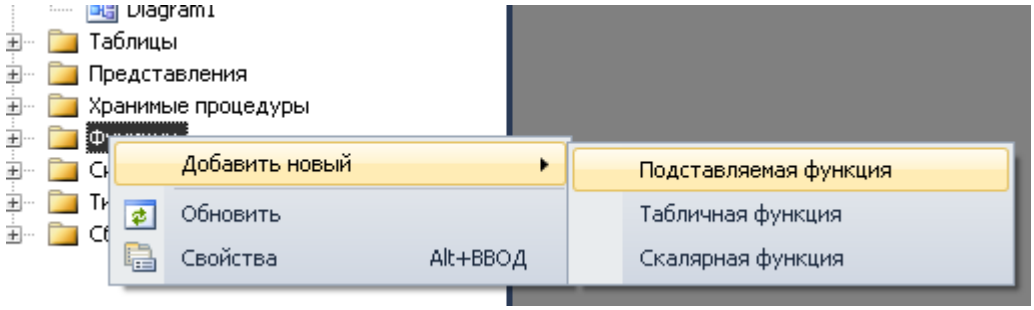


Рисунок 57 – Добавление подставляемой функции

В появившемся окне необходимо написать текст запроса (рисунок 58).

```
CREATE FUNCTION dbo.Function2
    (
    /*
    @parameter1 int = 5,
    @parameter2 datatype
    */
    )
RETURNS TABLE
AS
    RETURN /* SELECT ... FROM ... */
```

Рисунок 58 – Окно для создания запроса

В скобках после названия функции указываются параметры запроса. Параметру может быть присвоено значение по умолчанию.

Строка RETURNS TABLE означает, что функция возвращает таблицу.

После оператора RETURNS следует инструкция SELECT, с помощью которой производится отбор необходимых строк и столбцов из таблиц БД и которая имеет следующий формат:

SELECT <список полей>

FROM <список таблиц>

[WHERE <спецификация выбора строк>]

[GROUP BY <спецификация группировки>]

[HAVING <спецификация выбора групп>]

[ORDER BY <спецификация сортировки>]

Например, имеется таблица Студенты: Номер группы (ключевое поле), Номер студента (ключевое поле), ФИО, Год рождения, Адрес, Стипендия, Курс.

Пример: выбрать студентов определённой группы, обучающихся на определённом курсе (рисунок 59).

```
CREATE FUNCTION dbo.forStat1
    (
    @group int, @kurs int
    )
RETURNS TABLE
AS
    RETURN
            SELECT [Номер группы], [Номер студента], ФИО
            FROM dbo.Студенты
            WHERE @group=[Номер группы] AND @kurs=Курс
```

Рисунок 59 – Текст функции forStat1

Параметрами данной функции являются: @group, @kurs, которые имеют тип int. Тип должен быть указан для каждого параметра отдельно.

В предложении WHERE можно указать диапазоны с помощью оператора BETWEEN.

Пример: выбрать студентов с номерами в определённом диапазоне (рисунок 60).

```
ALTER FUNCTION dbo.forStat2
    (
    @number1 int, @number2 int
    )
RETURNS TABLE
AS
    RETURN
            SELECT *
            FROM dbo.Студенты
            WHERE [Номер студента] BETWEEN @number1 AND @number2
```

Рисунок 60 - Текст функции forStat2

Символ * после оператора SELECT означает, что выбираются все столбцы исходной таблицы. Строки выбираются, исходя из условия: Номер студента>=@number1 и Номер студента<=@number2.

В SQL предусмотрено пять встроенных функций: COUNT, SUM, AVG, MAX и MIN.

Функция COUNT вычисляет количество строк в таблице.

Пример: вычислить количество студентов определённой группы, обучающихся на определённом курсе (рисунок 61).

```
ALTER FUNCTION dbo.forStat4
    (
    @kurs int, @group int
    )
RETURNS TABLE
AS
    RETURN
            SELECT COUNT(*) AS [Количество студентов]
            FROM dbo.Студенты
            WHERE Курс=@kurs AND [Номер группы]=@group
```

Рисунок 61 - Текст функции forStat4

Данная функция возвращает отношение, у которого один столбец и одна строка. Оператор AS задаёт название столбца.

Функция SUM подсчитывает сумму числовых столбцов.

Пример: вычислить сумму стипендий, получаемых студентами определённой группы, учащихся на определённом курсе (рисунок 62).

```
ALTER FUNCTION dbo.forStat5
    (
    @kurs int, @group int
    )
RETURNS TABLE
AS
    RETURN
          SELECT SUM(Стипендия) AS [Сумма стипендий]
          FROM dbo.Студенты
          WHERE Курс=@kurs AND [Номер группы]=@group
```

Рисунок 62 - Текст функции forStat5

Функции AVG, MAX и MIN также работают с числовыми столбцами: AVG вычисляет среднее значение, а MAX и MIN находят соответственно максимальное и минимальное значение столбца в таблице.

В SQL предусмотрены встроенные функции, которые можно применять к группам строк внутри таблицы.

Пример: подсчитать количество студентов каждой группы определённого курса (рисунок 63).

```
ALTER FUNCTION dbo.forStat6
    (
    @kurs int
    )
RETURNS TABLE
AS
    RETURN
          SELECT [Номер группы], COUNT(*) AS [Количество студентов]
          FROM dbo.Студенты
          WHERE Курс=@kurs
          GROUP BY [Номер группы]
```

Рисунок 63 - Текст функции forStat6

В некоторых случаях возникает потребность рассматривать не все группы. Чтобы указать нужное подмножество групп, можно воспользоваться SQL-предложением HAVING.

Пример: получить группы, количество студентов в которых больше определённого числа (рисунок 64).

```
ALTER FUNCTION dbo.forStat7
    (
    @kol int
    )
RETURNS TABLE
AS
    RETURN
            SELECT [Номер группы], COUNT(*) AS [Количество студентов]
            FROM dbo.Студенты
            GROUP BY [Номер группы]
            HAVING COUNT(*)>@kol
```

Рисунок 64 - Текст функции forStat7

Для того, чтобы связать запрос с формой, необходимо переместить его из источника данных на форму (рисунок 65).

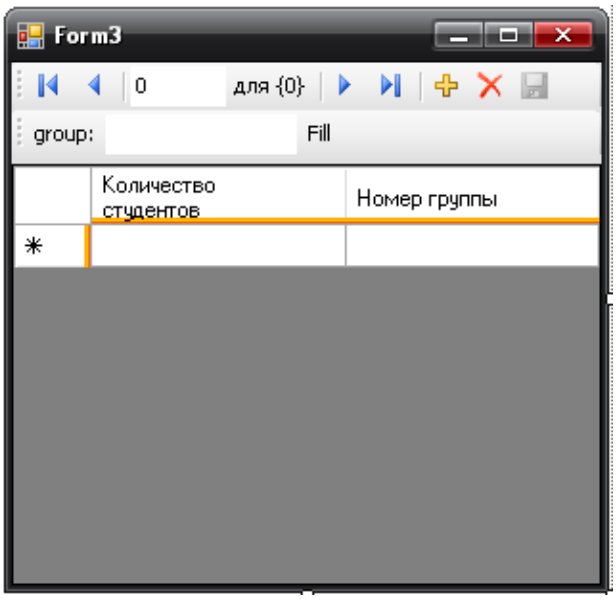


Рисунок 65 – Вид формы для запроса

На форме автоматически создаются поле для ввода параметров и кнопка для выполнения запроса, которые могут быть изменены (например, поле для ввода заменить на элемент DateTimePicker для ввода дат и т.д.).

Если поле для ввода параметров было изменено, то в обработчике события нажатия на кнопку выполнения запроса необходимо изменить название этого поля (рисунок 66).

```
try
{
    this.forStat6TableAdapter.Fill(this._1DataSet.forStat6,
        ((int)(System.Convert.ChangeType(groupToolStripTextBox.Text, typeof(int)))));
}
catch (System.Exception ex)
{
    System.Windows.Forms.MessageBox.Show(ex.Message);
}
```

Рисунок 66 – Обработчик выполнения запроса

Рассмотрим методические указания по созданию параметризированных запросов к базе данных MS SQL Server на основе языка C# в среде разработки MS Visual Studio 2010 на основе технологии ODBC.

Запросы с параметрами используются в тех случаях, когда необходимо вывести часть содержимого таблицы, основываясь на некоторых данных вводимых пользователем. Примеры: список заказов на некоторую дату, список сотрудников отдела предприятия, количество проданного товара и т.п. Существует несколько способов их создания и здесь будет рассмотрен, дающий самые широкие возможности для программиста.

Предположим, что в нашем распоряжении имеется база данных состоящая из двух сущностей: «Товары» и «Продажа» (рисунок 67).

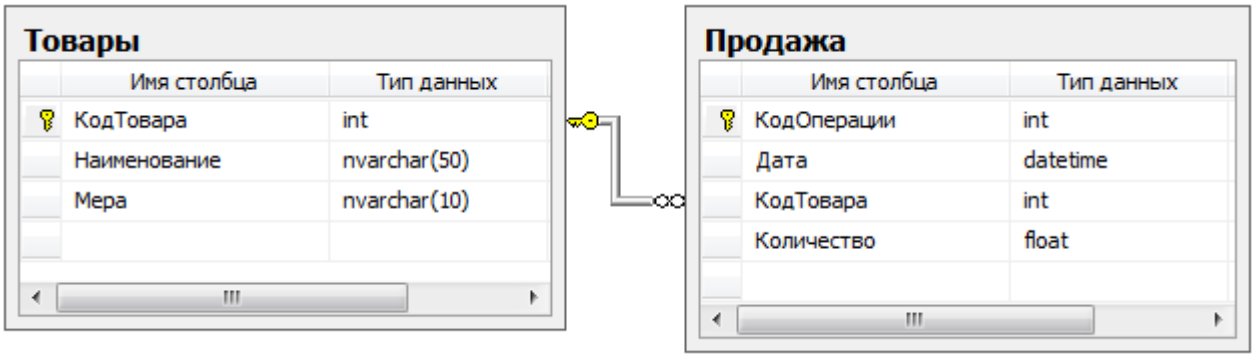


Рисунок 67 – База данных

В среде разработки Visual Studio (рисунок 68) создадим новый проект Windows Forms.

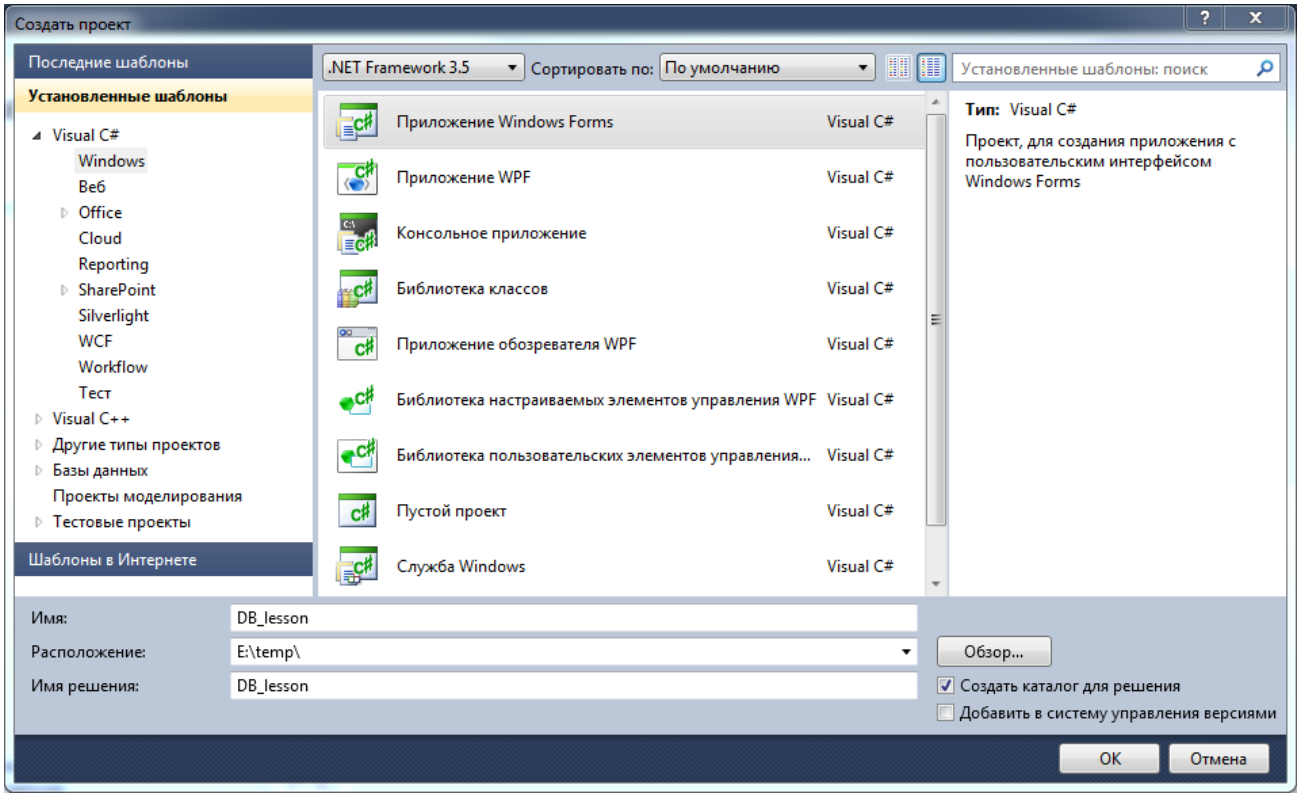


Рисунок 68 – Создание нового проекта

Будем производить запрос на выборку количества проданного товара за определенный период времени.

Для этого необходимо создать форму (рисунок 69) содержащую следующие элементы:

– DateTimePicker с именем «dateTimePickerStart»;
– DateTimePicker с именем «dateTimePickerEnd»;
– Button с именем «Show»;
– DataGridView с именем «dataGridViewList».

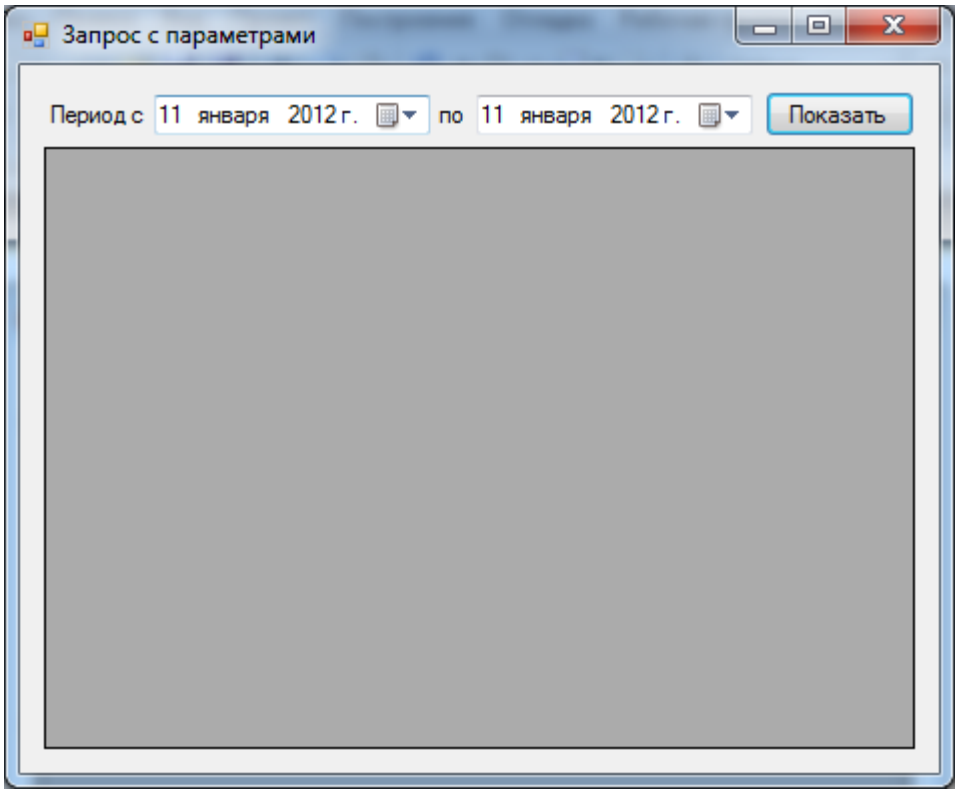


Рисунок 69 – Форма приложения

Создадим обработчик события нажатия кнопки «Показать» («Show»), для этого достаточно в конструкторе форм дважды на нее щелкнуть, откроется редактор кода с пустой функцией обработки («buttonShow_Click»).

Для того чтобы удобно пользоваться классами по работе с сервером баз данных SQL добавим пространство имен «System.Data.SqlClient».

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

Первым действием необходимо создать подключение к базе данных. Делается это с помощью класса SqlConnection. В качестве параметров конструктора класса передается строка подключения. В нашем случае она является таковой:

```
Data Source=(local);Initial Catalog=lesson;Integrated Security=True
```

Такой формат строки подключения подходит для большинства случаев. Здесь имя сервера указано как «(local)», это означает, что подключение будет производиться к серверу, установленному на текущем компьютере. Кроме того идентификация на сервере осуществляется на основе учетных записей Windows.

Чтобы правильно написать строку подключения воспользуемся мастером «Добавить новый источник данных» в меню «Данные» (рисунок 70).

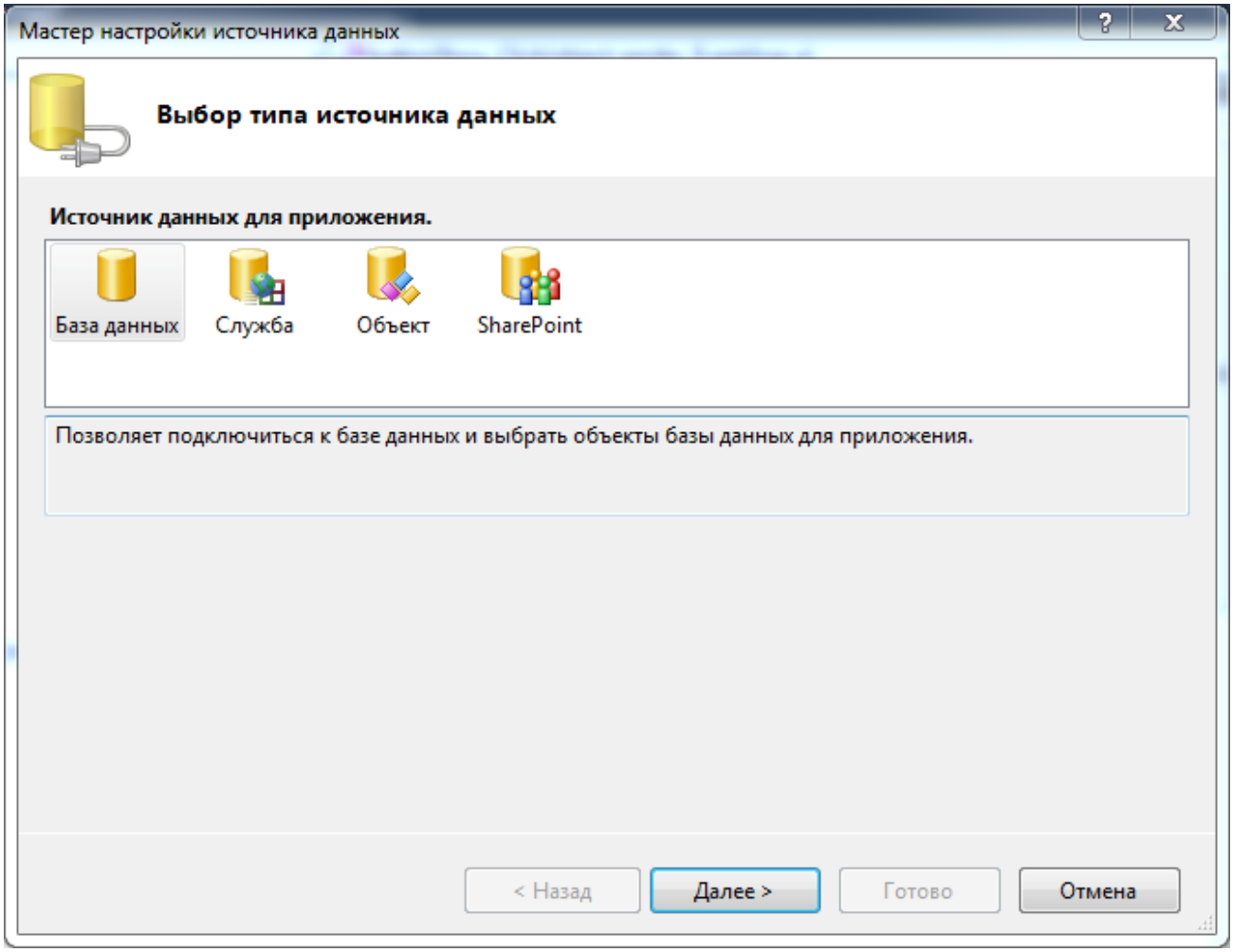


Рисунок 70 – Соединение с базой данных

В качестве источника выбираем «База данных». На следующем шаге тип модели указываем как «Набор данных» (рисунок 71).

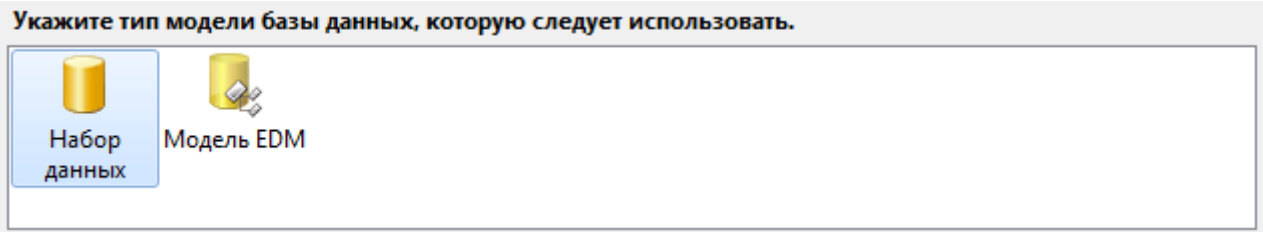


Рисунок 71 – Выбор типа модели

На следующем шаге нажимаем клавишу «Создать подключение…». Указывается необходимый сервер, имя база данных и в случае необходимости пароль (рисунок 72)

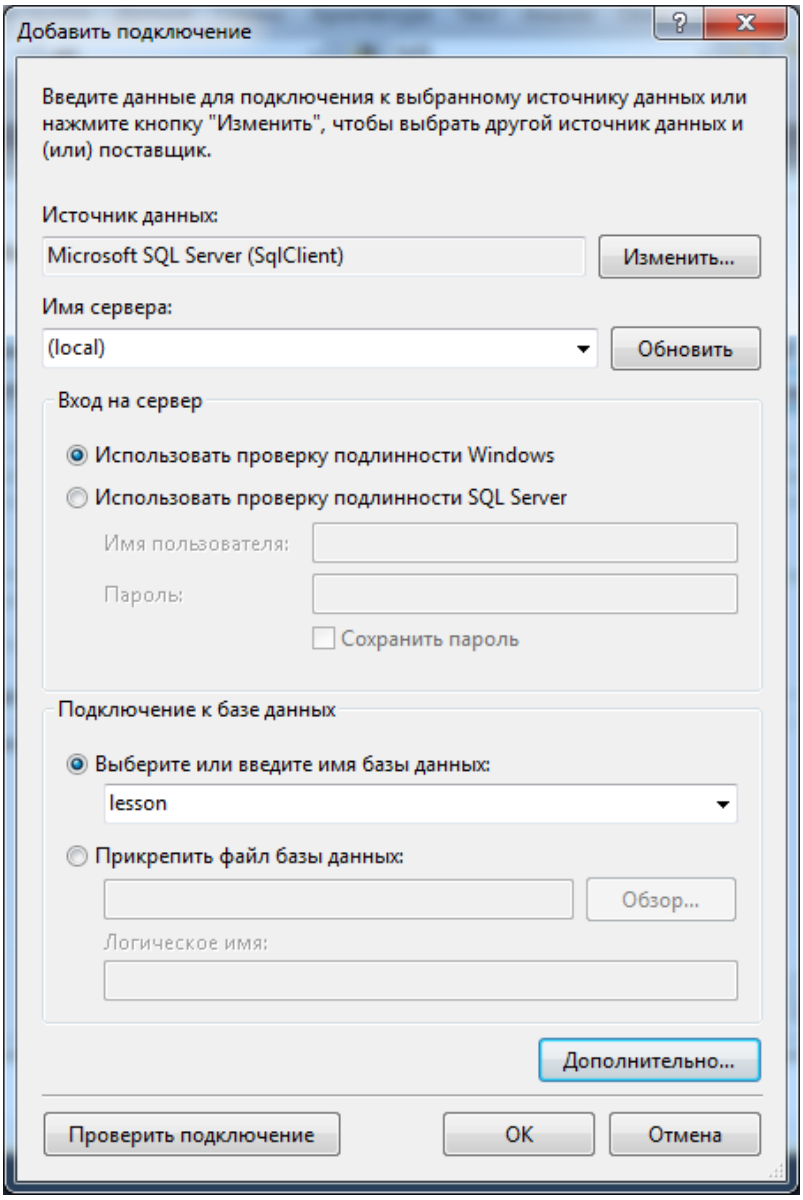


Рисунок 72 – Новое подключение

Для того чтобы увидеть строку подключения (рисунок 73) нажимем кнопку «Дополнительно…».

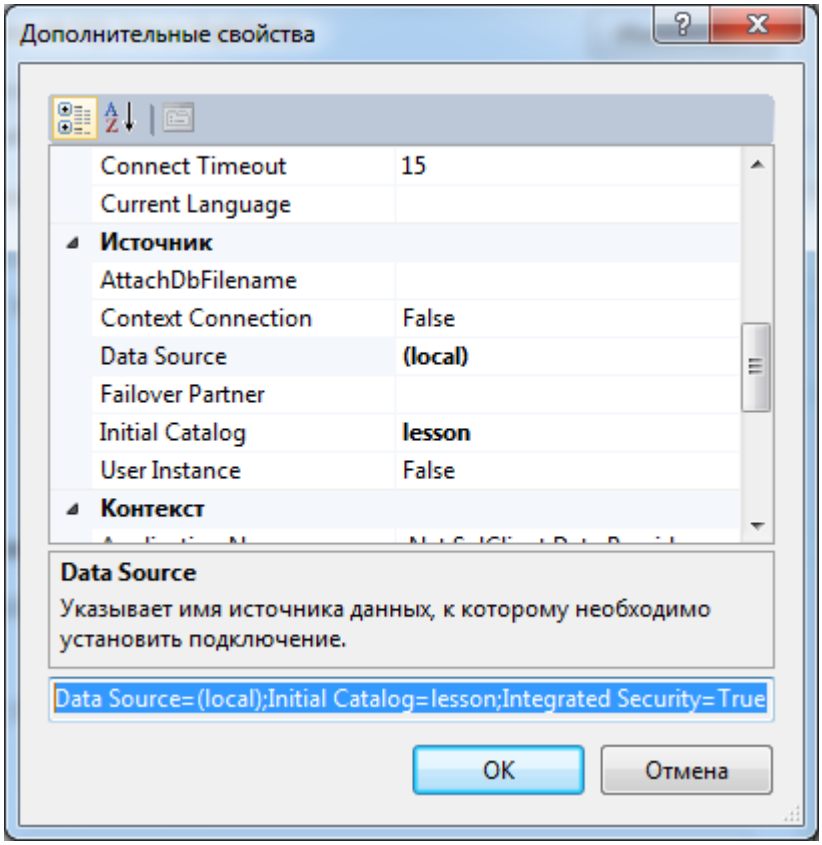


Рисунок 73 – Строка подключения

Работу мастера можно не продолжать.

Теперь создадим экземпляр класса SqlConnection.

```
SqlConnection conn = new SqlConnection("Data Source=(local);Initial Catalog=lesson;Integrated Security=True");
```

Для выборки интересующей нас информации необходимо выполнить команду на языке SQL:

```
SELECT [Товары].[Наименование], SUM([Продажа].[Количество]) AS Продано, [Товары].[Мера]
FROM [Продажа]
INNER JOIN [Товары] ON [Товары].[КодТовара] = [Продажа].[КодТовара]
WHERE [Продажа].[Дата] BETWEEN @Start AND @End
GROUP BY [Товары].[КодТовара], [Товары].[Наименование], [Товары].[Мера]
```

В данном запросе значения переменных @Start и @End определяются пользователем. Все эти данные необходимо указать в классе SqlCommand (смотрите листинг ниже).

```
// создание запроса
SqlCommand myCom = conn.CreateCommand();
// текст запроса
myCom.CommandText =
@"SELECT [Товары].[Наименование], SUM([Продажа].[Количество]) AS Продано, [Товары].[Мера]
FROM [Продажа]
INNER JOIN [Товары] ON [Товары].[КодТовара] = [Продажа].[КодТовара]
WHERE [Продажа].[Дата] BETWEEN @Start AND @End
GROUP BY [Товары].[КодТовара], [Товары].[Наименование], [Товары].[Мера]";
// переменные используемые в запросе
myCom.Parameters.Add("@Start", SqlDbType.DateTime);
myCom.Parameters.Add("@End", SqlDbType.DateTime);
// инициализация значениями
myCom.Parameters["@Start"].Value = dateTimePickerStart.Value.Date;
myCom.Parameters["@End"].Value =
dateTimePickerEnd.Value.Date;
```

И наконец, последний этап: выполнение запроса и пересылка данных на DataGridView. Здесь создаются объекты DataSet (представляет кеш данных в памяти) и DataAdapter (производит манипуляции с данными, инициализацию подключения и формирование конечного запроса):

```
// непосредственно объект манипулирующий запросами
SqlDataAdapter adapter = new SqlDataAdapter(myCom);
// кеш таблиц
DataSet ds = new DataSet();
// заполнение таблиц DataSet
adapter.Fill(ds);
// подключение необходимо закрыть. Открытие происходит неявно при заполнении DataSet
conn.Close();
// результат одна таблица, которую передаем в качестве источника DataGridView
dataGridViewList.DataSource = ds.Tables[0].DefaultView;
```

Создание запроса закончено, результат на рисунке 74.

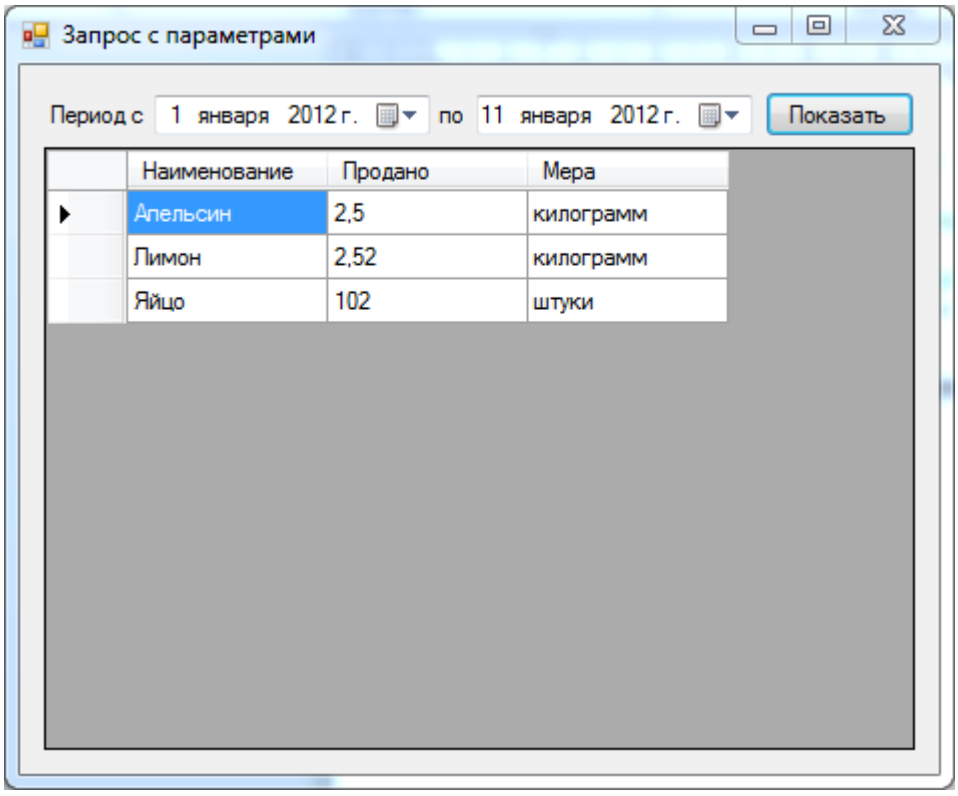


Рисунок 74 – Результат работы запроса

Исходные таблицы содержут следующие данные

Товары

	КодТовара	Наименование	Мера
1	1	Апельсин	килограмм
2	2	Лимон	килограмм
3	3	Яйцо	штуки

Продажа

	КодОперации	Дата	КодТовара	Количество
1	1	2012-01-04 00:00:00.000	1	0,3
2	2	2012-01-05 00:00:00.000	1	1
3	3	2012-01-03 00:00:00.000	1	1,2
4	4	2012-01-06 00:00:00.000	2	0,5
5	5	2012-01-11 00:00:00.000	2	1,22
6	6	2012-01-09 00:00:00.000	2	0,8
7	7	2012-01-09 00:00:00.000	3	34
8	8	2012-01-07 00:00:00.000	3	12
9	9	2012-01-08 00:00:00.000	3	56

Код программы целиком:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace DB_lesson
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonShow_Click(object sender, EventArgs e)
        {
            // создание подключения
            SqlConnection conn = new SqlConnection("Data
Source=(local);Initial Catalog=lesson;Integrated Security=True");
            // создание запроса
            SqlCommand myCom = conn.CreateCommand();
            // текст запроса
            myCom.CommandText =
            @"SELECT [Товары].[Наименование], SUM([Продажа].[Количество]) AS
Продано, [Товары].[Мера]
```

```
            FROM [Продажа]
            INNER JOIN [Товары] ON [Товары].[КодТовара] = [Прода-
жа].[КодТовара]
            WHERE [Продажа].[Дата] BETWEEN @Start AND @End
            GROUP BY [Товары].[КодТовара], [Товары].[Наименование], [Това-
ры].[Мера]";
            // переменные используемые в запросе
            myCom.Parameters.Add("@Start", SqlDbType.DateTime);
            myCom.Parameters.Add("@End", SqlDbType.DateTime);
            // инициализация значениями
            myCom.Parameters["@Start"].Value = dateTimePick-
erStart.Value.Date;
            myCom.Parameters["@End"].Value = dateTimePickerEnd.Value.Date;
            // непосредственно объект манипулирующий запросами
            SqlDataAdapter adapter = new SqlDataAdapter(myCom);
            // кеш таблиц
            DataSet ds = new DataSet();
            // заполнение таблиц DataSet
            adapter.Fill(ds);
            // подключение необходимо закрыть. Открытие происходит неявно при
заполнении DataSet
            conn.Close();
            // результат одна таблица, которую передаем в качестве источника
DataGridView
            dataGridViewList.DataSource = ds.Tables[0].DefaultView;
        }
    }
}
```

Контрольные вопросы

1 Дать определение базы данных.

2 Перечислить типы баз данных и кратко описать особенности каждой.

3 Чем отличаются реляционные базы данных от баз данных других типов баз данных?

4 Описать модели клиент-сервер.

5 В чем сущность принципа распределенной обработки данных?

6 Назвать пять групп функций стандартного интерактивного приложения. Кратко охарактеризуйте каждую группу.

7 Что такое SQL? В чем отличие TRANSACT-SQL?

8 Привести пример простого запроса на выборку данных SQL.

9 Что такое “инфологическая модель предметной области”, по каким правилам она строится в нотации Ричарда Баркера?

10 Что такое “транзакция”?

11 Для чего используются хранимые функции и процедуры?

12 Что такое “триггер”?

13 Какие проблемы могут возникнуть при параллельной обработке данных? Как они могут быть решены?

14 Перечислить модели блокировок. Описать механизмы их действия

15 Описать процесс создания базы данных MS SQL Server с использованием инструмента “Management Studio”.

16 Описать модель защиты данных в MS SQL Server.

17 Перечислить и коротко описать типы данных используемые в MS SQL Server.

18 Описать процесс создания таблиц и добавления связей между ними в MS SQL Server при помощи Visual Studio.

19 Описать процесс создания триггеров MS SQL Server при помощи Visual Studio.

20 Описать процесс создания представлений MS SQL Server при помощи Visual Studio.

21 Описать процесс создания хранимых процедур MS SQL Server при помощи Visual Studio.

22 В чем преимущества использования языка программирования C#?

23 Описать стандартные типы пользователей базы данных и уровни их доступа.

24 Перечислить разрешения, которые можно предоставить на уровне роли базы данных.

Заключение

Методические указания предназначены для выполнения курсовой работы по разработке базы данных и приложения, работающего с ней.

В данные методические указания был включен краткий обзор клиент-серверной технологии баз данных, технологии ADO.NET, приведен процесс создания централизованной базы данных с помощью СУБД Microsoft SQL Server. Также детально рассмотрен процесс создания приложения, созданного при помощи среды разработки Microsoft Visual Studio, которое работает с базой данных, хранимой на сервере.

Целью данных методических указаний является привитие практических навыков студентам разработке приложений на основе архитектуры “клиент-сервер” средствами СУБД Microsoft SQL Server при помощи среды Microsoft Visual Studio на языке C#.

Изложение методических указаний построено на конкретном варианте предметной области. По описанию предметной области была построена инфологическая модель и приведена пошаговая инструкция создания по ней базы данных.

В методических указаниях дан краткий обор новейшей технологии создания приложений – .NET и нового языка программирования, созданного в рамках этой технологии – C#. После этого рассматривалась модель отсоединенного доступа к данным, используемая в ADO.NET, а также приводился обзор объектной модели, составляющей основу ADO.NET. На основе приведенных теоретических предпосылок, была изложена пошаговая инструкция по созданию приложения при помощи Microsoft Visual Studio на языке C#. В методических указаниях изложен вариант руководства администратору, а также пример подробной инструкции пользователя.

В приложениях содержатся варианты заданий на курсовую работу, привила оформления отчета по курсовой работе.

Список использованных источников

1 Советов, Б. Я. Базы данных: теория и практика: учеб. для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской.- 2-е изд. - М. : Юрайт, 2012. - 464 с.

2 Кузин, А. В. Базы данных: учеб. пособие / А. В. Кузин, С. В. Левонисова.- 2-е изд., стер. - М. : Академия, 2008. - 316 с.

3 Хомоненко, А. Д. Базы данных: учеб. для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; под ред. А. Д. Хомоненко.- 6-е изд. - СПб. : КОРОНА-Век, 2010. - 736 с. : ил.

4 Фуфаев, Э. В. Разработка и эксплуатация удаленных баз данных: учеб. для сред. проф. образования / Э. В. Фуфаев, Д. Э. Фуфаев. - М. : Академия, 2008. - 251 с.

5 Пирогов, В.Ю. SQL Server 2005: Программирование клиент-серверных приложений/ В.Ю. Пирогов. – СПб.: БХВ-Петербург, 2006. – 336 с.: ил.

6 Советов, Б.Я. Базы данных: теория и практика: учеб. для втузов/ Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской – 2-е изд., стер. – М.: Высш. шк., 2007. -463 с.: ил.

7 Кузин, А.В. Базы данных: учеб. пособие для студ. высш. учеб. заведений / А.В.Кузин, С.В. Левонисова. – М.: Издательский центр «Академия», 2005. – 320с.

8 Уткин, В.Б. Информационные системы в экономике: учебник для студ. высш. учеб. заведений / В.Б. Уткин, К.В. Балдин. – М.: Издательский центр «Академия», 2009, 2010. – 288 с.

9 Золотова, С.И. Практикум по Access/ С.И. Золотова – М.: Финансы и статистика, 2008. – 144с.: ил.

10 Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: учебник/ А.М. Вендров – М.: Финансы и статистика, 2005.

11 Вендров, А.М. Практикум по проектированию программного обеспечения экономических информационных систем: учеб. пособие/ А.М. Вендров – М.: Финансы и статистика, 2006.

12 Волкова, Т. В. Проектирование и создание баз данных: учебное пособие/ Т.В. Волкова – Оренбург: ПИК ГОУ ОГУ, 2006 – 140с.ISBN 5-02-011452-9

13 Полякова, Л.Н. Основы SQL. Интернет-университет информационных технологий/ Л.Н. Полякова - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2007, 224 с

14 Пирогов, В.Ю. MS SQL Server 2005. Программирование клиент-серверных приложений/ В.Ю. Пирогов – СПб.: БХВ-Петербург, 2006. – 336 с.: ил.

15 Пирогов, В.Ю. MS SQL Server 2000: управление и программирование/ В.Ю. Пирогов – СПб.: БХВ-Петербург, 2005. – 608 с.: ил.– 288 с. : ил. – Парал. тит. англ. SBN 5-8459-0450-1 (рус.)

16 Нильсен, П. Microsoft SQL Server 2005. Библия пользователя: пер. с англ. / П.Нильсен – М.: Вильямс – 2007 – 123 с.

17 Станек, У.Р: Microsoft SQL Server 2005. Справочник администратора: пер. с англ./ У.Р. Станек – М.: Русская редакция – 2006 – 544 с.

18 Троелсен, Э. Язык программирования C# 2005 и платформа .NET 2.0.: пер. с анг./ Э. Троелсен – М.-СПб.-Киев.: Вильямс.– 2007 г. – 168 с.

19 Марченко, А.Л. Основы программирования на C# 2.0: / А.Л. Марченко – М.: Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний – 2007 – 552 с.

20 Пирогов, В.Ю. MS SQL Server 2000: управление и программирование/ В.Ю. Пирогов – СПб.: БХВ-Петербург, 2005. – 608 с.: ил.

21 Кренке Д. Теория и практика построения баз данных. 8-е изд./ Д. Кренке. – СПб.: Питер, 2003. – 800с.: ил.

Приложение А

Варианты предметной области

Вариант № 1

Автоматизированная информационная система «Ремонт оборудования»

Предприятие – высшее учебное заведение. В структуре есть вычислительный центр, в составе которого, в свою очередь, есть ремонтная мастерская.

Словесное описание предметной области: Предприятие, в состав которого входит ряд подразделений, имеет ремонтную мастерскую, в которой производится различного вида ремонт средств вычислительной и оргтехники. Необходимо вести учет, сданной в ремонт техники. Для каждой единицы техники необходимо хранить её инвентарный номер, название устройства, его модель, год выпуска, какому подразделению устройство принадлежит на текущую дату, т.е. необходимо хранить историю перемещения техники между подразделениями организации. Перемещение единицы техники осуществляется на основе документа, в котором указано с какой даты данная единица перемещается в указанное подразделение. Эта дата одновременно является датой окончания нахождения единицы техники в предыдущем подразделении. Необходимо также хранить дату сдачи единицы техники в ремонт, вид ремонта, срок ремонта, ФИО, номер сотрудника, сдавшего технику в ремонт, ФИО, номер сотрудника, который принял технику в ремонт, ФИО, номер, должность сотрудника, выполняющего ремонт, перечень необходимых для ремонта запасных частей, их стоимость на текущую дату. Стоимость запасных частей определяется из документа – накладной, по которой они получены, в ней и указана дата получения, стоимость. О каждом сотруднике необходимо хранить информацию о всех перемещениях за время работы на данном предприятии: дата начала работы, дата окончания, подразделение, должность.

Необходимо реализовать следующую обработку данных:

Построить диаграмму: количество техники определенного наименования в заданном подразделении по годам (за три года).

Список сотрудников (ФИО, год рождения) по подразделениям, мужчин, достигших пенсионного возраста в текущем году (60 лет).

Название подразделения, у которого была наибольшее количество техники, переданной в ремонт.

Вариант № 2

Автоматизированная информационная система
«Технический осмотр автомобилей»

Предприятие – учреждение ГИБДД

Словесное описание предметной области: При проведении технического осмотра автомобиля необходимо фиксировать следующие данные: госномер автомобиля, проходящего технический осмотр, номер двигателя, цвет, марка, номер технического паспорта, номер водительского удостоверения, ФИО владельца, адрес прописки, год рождения, пол. Данные фиксируются на дату прохождения текущего осмотра, необходимо хранить историю осмотров – дата прохождения, результат. Необходимо также фиксировать ФИО, должность, звание сотрудника ГАИ, проводившего осмотр, заключение осмотра. Каждый день технический осмотр могут проходить много автомобилей, проводить осмотр могут разные сотрудники, но каждый сотрудник проводит за день не более 10 осмотров.

Необходимо реализовать следующую обработку данных:

Построить диаграмму: количество автомобилей, прошедших техосмотр за заданный месяц с разбивкой по дням.

Список сотрудников ГАИ, проводивших осмотр на заданную дату: ФИО, звание сотрудника, госномера автомобилей, которые он осматривал.

История прохождения осмотров заданным автомобилем (номер двигателя) – дата прохождения, результат.

Вариант № 3

Автоматизированная информационная система «Комплектующие к станкам»

Предприятие, имеющее в своей структуре производственные участки и склады.

Словесное описание предметной области: необходимо облегчить работу по учету комплектующих деталей, необходимых для ремонта и нормального функционирования станков предприятия. Каждый станок имеет номер, название (модель) и относится к определенному типу (токарные, фрезерные и т.п.). Необходимо фиксировать дату начала работы станка, эксплутационный срок и дату его списания. Каждому станку могут соответствовать разные комплектующие детали, каждая также имеет номер, название. Деталь получают со склада по накладной, в которой указано – с какого склада деталь получена, дата получения, цена детали на дату получения. На накладной расписывается ремонтник, производящий наладку и ремонт станка. Складов на предприятии м.б. несколько, каждый имеет номер, адрес (улица, номер дома), количество метров занимаемой площади.

Необходимо реализовать следующую обработку данных:

Построить диаграмму: количество деталей заданного наименования, отпускаемых с заданного склада за заданный месяц с разбивкой по дням.

Список станков заданного наименования, находящихся на балансе предприятия на текущую дату – название организации, дата, номер станка, дата ввода его в эксплуатацию.

Список складов – название организации, дата, номер склада, адрес.

Вариант № 4

Автоматизированная информационная система «Кадры предприятия»

Государственное предприятие.

Словесное описание предметной области: На предприятии существует ряд подразделений. Каждое подразделение имеет штатное расписание, в котором имеется перечень должностей. Каждая должность имеет название, краткое название, шифр, нижнюю и верхнюю границы разрядов единой тарифной сетки (от 1 до 18). Также известно, сколько единиц каждой должности выделено подразделению. О сотрудниках, работающих на предприятии, необходимо знать всю историю их перемещения – где, в каком подразделении работал сотрудник, на какой должности, какой имел разряд, дату начала и дату окончания работы. Также о сотруднике необходимо хранить личные данные: ФИО, возраст, пол, семейное положение.

Необходимо реализовать следующие запросы:

Штатное расписание заданного подразделения –должность, разряд – нижняя, верхняя граница, количество ставок.

Список сотрудников на заданную дату (ФИО, дата рождения), женского пола, достигших пенсионного возраста (55 лет) в текущем году по подразделениям.

Построить диаграмму: Список сотрудников моложе 20, работающих на заданной должности на предприятии.

Вариант № 5

Автоматизированная информационная система

«Банк данных товаров, производимых различными предприятиями»

Рекламное предприятие.

Словесное описание предметной области: Необходимо хранить информацию о товаре, который производится предприятиями области – каждый товар имеет название, номер, относится к какой-либо группе товаров (канцелярские принадлежности, бумага, скобяные товары и т.п.). Цена товара меняется во времени и определяется позицией прайс-листа, выпускаемого периодически на предприятии, производящем товар. Предприятие характеризуется названием, имеет статистический код, адрес, телефон. Каждое предприятие может производит много товаров, и в тоже время один и тот же товар могут производить несколько предприятий. Также необходимо знать ФИО и должность руководителя предприятия, телефон отдела маркетинга предприятия, руководителя отдела маркетинга, ФИО контактного лица.

Необходимо реализовать следующие запросы:

Список товаров, выпускаемых заданным предприятием (название предприятия, дата, название товара, последняя цена, данные сгруппировать по группе товара).

Список предприятий, выпускающих заданный товар – название товара, дата, название предприятия, название населенного пункта.

Список всех предприятий – дата, название предприятия, телефон, ФИО контактного лица.

Вариант № 6

Автоматизированная информационная система
«Учет договоров страхования»

Предприятие – страховая организация.

Словесное описание предметной области: Страховая организация заключает договора с физическими лицами и юридическими организациями. Для организации оформляется коллективный договор, в котором перечислены страхуемые сотрудники: ФИО, возраст, категория риска (первая, вторая, высшая и т.п.). О предприятии хранится следующая информация: код, полное наименование, краткое наименование, адрес, банковские реквизиты (номер банка), специализация предприятия (медицинское учреждение, автотранспортное предприятие, учебное заведение и т.п.). В заключаемом коллективном договоре указывается дата заключения, срок договора (конец действия договора), сумма выплат по каждой категории сотрудников, выплаты по страховым случаям. Выплаты зависят от категории сотрудника. Необходимо также хранит информацию о страховом агенте, заключившем договор (ФИО, паспортные данные). Каждый агент может заключить много договоров, в каждом договоре м.б. оформлено несколько сотрудников. А каждый конкретный договор м.б. заключен только одним агентом.

Необходимо реализовать следующие запросы:

Список действующих договоров, заключенных заданным предприятием на текущую дату – название предприятия, адрес, дата, список договоров – номер, дата начала, дата окончания.

Список страховых агентов, работающих с заданным предприятием на текущую дату (ведущим действующие договора) – дата, название страховой фирмы, ФИО страхового агента, контактный телефон;

Список страховых выплат для каждой категории работников на текущую дату – дата, категория работника, страховая выплата.

Вариант № 7

Автоматизированная информационная система
«Учёт спроса и предложения»

Производственное предприятие, имеющее в структуре отдел маркетинга.

Словесное описание предметной области: Отдел маркетинга предприятия занимается спросом выпускаемого товара. Каждый товар характеризуется кодом, названием, категорией (промышленные, бытовые, торговое оборудование и т.п.). Продажа товара на предприятии осуществляется по накладным, в которых указано кому отправлен товар (юридическое или физическое лицо, название, имя, адрес, номер, серия документа, банковские реквизиты (номер и название банка). В накладной также указывается отпускная цена на текущую дату, количество отпущенного товара. Необходимо отслеживать название населенных пунктов, название региона России и страны ближнего или дальнего зарубежья куда отправлен товар. Каждая накладная соответствует одному пункту назначения и одному покупателю.

Необходимо реализовать следующие запросы:

Список покупателей, сделавших покупку на максимальную сумму на текущую дату – дата, название покупателя, адрес, сумма покупки.

Изменение стоимости заданного товара за весь учитываемый период времени (название предприятия, производящего товар, наименование товара, дата, стоимость);

Список существующих категорий товаров.

Вариант № 8

Автоматизированная информационная система
«Учет приказов и распоряжений»

Предприятие, имеющее в своей структуре ряд иерархически подчиняющихся подразделений.

Словесное описание предметной области: Необходимо хранить информацию о внутренних приказах, распоряжениях или других подобных документах, сроках их выполнения и исполнителях. Например: название документа - "Указание о введении в действие плана по совершенствованию охраны труда", мероприятие, описываемое в документе – "Установить порядок выпуска приказов по случаю нарушения охраны труда", - исходящий корреспондент (человек, готовивший приказ) – гл. инженер.

Информация о исходящем корреспонденте: название подразделения, должность корреспондента, ФИО. Приказ, распоряжение характеризуются номером, датой, содержанием, мероприятием, ответственным за выполнение (название подразделения, должность, ФИО), датой выполнения мероприятия, пометкой о выполнении мероприятия. В одном документе может быть перечислено несколько мероприятий. Фиксируется дата выхода документа. Распоряжение подписывает сотрудник ранга руководителя предприятия, необходимо фиксировать должность и ФИО.

Необходимо реализовать следующие запросы:

Список исходящих корреспондентов, мероприятия по которым на заданную дату не были выполнены – текущая дата, дата выхода документа, название мероприятия, дата выполнения, ФИО, должность, название подразделения ответственного.

Список мероприятий за заданный период – дата начала периода, дата окончания периода, название мероприятия, выполнено или нет;

Список всех зафиксированных документов заданного типа (например, приказов) на текущую дату, отсортированных по дате документа – название предприятия, текущая дата, название вида документа, список документов – номер, дата документа, пометка о выполнении.

Вариант № 9

Автоматизированная информационная система «Расчет заработной платы»

Государственное предприятие, имеющее в своей структуре ряд иерархически подчиняющихся подразделений.

Словесное описание предметной области: расчет заработной платы производится для сотрудников предприятия, работающих на бюджетной основе. Для сотрудника определена должность и разряд. В соответствии с единой тарифной сеткой каждый разряд имеет свой коэффициент. Оклад рассчитывается как произведение минимального размера оплаты труда (он может меняться со временем) на коэффициент, соответствующий разряду. Также сотрудник имеет доплату – так называемый уральский коэффициент в размере 15 процентов от оклада. Удержания – 13 процентов подоходный налог, 1 процент – пенсионный фонд, 1 процент – профсоюзный взнос, если сотрудник член профсоюза. Необходимо вести расчет заработной платы и накапливать информацию по сотруднику на протяжении всего периода его работы. Необходимо за каждый месяц рассчитывать и хранить сколько всего начислено, удержано и к выплате. Расчет больничных и отпускных дней не вести.

Необходимо реализовать следующие запросы:

Список коэффициентов, соответствующих разрядам на текущую дату – дата, список коэффициентов – разряд, коэффициент (разрядов 18).

Список сотрудников, имеющих минимальную ЗП за текущий месяц – дата, список сотрудников - ФИО, должность сотрудника, сумма к выдаче.

Список всех начислений, удержаний, выплат для заданного сотрудника на текущий месяц – дата, ФИО, должность сотрудника, все начисления, удержания, выплата.

Вариант № 10

Автоматизированная информационная система
«Учет средств вычислительной и оргтехники»

Предприятие, имеющее в своей структуре ряд иерархически подчиняющихся подразделений.

Словесное описание предметной области: Необходимо вести учет средств ВТ и оргтехники на предприятии. Каждое средство имеет инвентарный номер, название, модель, дату приобретения, стоимость. На предприятии средства могут передаваться из подразделения в подразделение, при этом необходимо знать дату передачи и новое материально ответственное лицо (ФИО, должность). Материально ответственный должен работать в том подразделении куда передается техника. Также необходимо знать номер комнаты где находится техника на текущий момент. О каждом подразделении фиксируется номер, полное и краткое название. Также необходимо фиксировать кто по должности в подразделении является руководителем. а кто материально ответственным лицом.

Необходимо реализовать следующие запросы:

Список средств ВТ и оргтехники, закрепленной за данным подразделением в настоящий момент времени – название предприятия, название подразделения, дата, список средств ВТ и оргтехники – инвентарный номер, название, модель.

Список всех помещений подразделения на текущий момент времени, их площадь – название предприятия, название подразделения, дата, список помещений – номер, количество кв.м.

Наименование техники, закрепленной за заданным материально ответственным лицом на текущий момент времени – дата, ФИО, должность материально ответственного лица, список техники – инвентарный номер, название, модель.

Вариант № 11

Автоматизированная информационная система
«Учет инвентаря на складах предприятия»

Предприятие, имеющее в своей структуре ряд иерархически подчиняющихся подразделений.

Словесное описание предметной области: Предприятие имеет несколько складов, в которых находится рабочий инвентарь (тип инвентаря - спецодежда, инструменты, подсобные средства и т.п.). Каждый склад имеет номер, название, телефон. Инвентарь завозят на склад в соответствии с приходной накладной, которая имеет дату, перечень (наименование) инвентаря, количество единиц каждого инвентаря. Также указывается ФИО и должность сотрудника склада, принявшего инвентарь. Расход инвентаря со склада осуществляется по расходной накладной, которая имеет ту же структуру, что и приходная, только учитывает расход инвентаря со склада. Поступление инвентаря на склад отражается в карточке складского учета, заводимой для каждого наименования инвентаря. В карточке учитываются все приходы и расходы.

Необходимо реализовать следующие запросы:

Список инвентаря заданного типа на заданном складе и его количество на текущую дату.

Список всех приходов и расходов инвентаря заданного наименования на всех складах – дата, название инвентаря, список - дата прихода, количество, дата расхода, количество.

Список всех складов, отсортированных по названию на текущую дату.

Вариант № 12

Автоматизированная информационная система
«Учет пациентов клинической больницы»

Клиническая больница.

Словесное описание предметной области: На каждого вновь поступившего больного заводится карточка медицинской статистики: ФИО больного, пол, возраст, предварительный диагноз, как поступил больной (направление поликлиники, доставлен скорой помощью и т.п.), дата поступления, прочее описание: примерный рост, цвет волос, особые приметы, примерный возраст, номер палаты, в которую положен больной. Информация о больном м.б. неполной, если он не может ответить на вопросы. За время лечения в больнице больной м.б. переведен в разные палаты, необходимо знать дату перевода, номер и телефон палаты. После окончания лечения фиксируется дата выписки и причина выписки либо другой исход (полное излечение, направлен в санаторий и т.п.)

Необходимо реализовать следующие запросы:

Телефон и номер палаты заданного больного (больной м.б. определен по ФИО, если однофамильцев несколько, указать всех, при этом выводить возраст больного).

Список всех больных на заданное число – дата, номер палаты, ФИО больного, возраст, дата поступления.

Список больных, достигших заданного возраста женского пола на текущую дату– дата, ФИО больного, возраст.

Вариант № 13

Автоматизированная информационная система
«Места проведения досуга граждан»

Организация, занимающиеся организацией досуга населения.

Словесное описание предметной области: необходимо вести учет всех объектов города, где могут проходит развлекательные мероприятия, отдых граждан. Объект характеризуется названием, типом (танцзал, спортзал, бильярдный клуб и.т.), адресом, количеством мест, адресом, информацией о владельце – частное, юридическое лицо, название, имя, ФИО руководителя, контактный телефон, дата открытия. Объект может закрыться, а потом снова начать работу (сезонные объекты), это необходимо учитывать. Необходимо собирать информацию о популярности среди населения данного объекта (дата, количество посетивших объект). Также необходимо иметь информацию о мероприятиях, заявленных на проведение. В заявке должны учитываться: название объекта, дата проведения, название мероприятия, вид мероприятия (концерт, клубное первенство, просмотр фильма и т.п.).

Необходимо реализовать следующие запросы:

Список объектов города на текущую дату –дата, название действующих объектов – тип, название, адрес.

Список мероприятий, которые будут проводится в ближайшие 2 недели – дата проведения, название мероприятия, название объекта, адрес.

Список объектов заданного типа на текущую дату.

Вариант № 14

Автоматизированная информационная система
«Выставочные залы города»

Предприятие – областной союз художников.

Словесное описание предметной области: необходимо иметь информацию о выставочных залах города, выставках, проводимых в них, участниках выставок. Каждый выставочный зал характеризуется название, площадью, адресом, телефоном. Зал может принадлежать какому-либо владельцу – это м.б. городская организация. областная, общественная, частное лицо. Необходимо иметь сведения о владельцах (название или имя, адрес, телефон). Также необходимо хранить информацию о видах выставок, проводимых в выставочных залах – это могут быть выставки изобразительного искусства, прикладного, скульптура и т.п., датах проведения выставок О художниках, которые принимают участие в выставках, необходимо хранить: имя, место и дату рождения, краткую биографическую справку, сведения об образовании. Каждый художник на выставке может представлять несколько работ, необходимо хранить название работы, её исполнение (краски, акварель, скульптура и т.п.), дату создания, размеры: высота, ширина, если это скульптура – объем.

Необходимо реализовать следующие запросы:

Список участников и их работы заданной выставки – название выставки, дата проведения, список работ – название. исполнение, ФИО автора, возраст автора, дата создания.

Список всех выставочных залов города – дата, название, адрес, занимаемая площадь, владелец.

Перечень выставок, проходящих в городе с адресами залов на текущий момент – дата, список – название выставки, адрес зала.

Вариант № 15

Автоматизированная информационная система
«Учет отказа оборудования»

Предприятие, имеющее в своей структуре ряд иерархически подчиняющихся подразделений.

Словесное описание предметной области: необходимо вести учет отказа оборудования на каком-либо участке предприятия. Производственные участки на предприятии имеют номер, название (гальванический участок, участок переработки сырья и т.п.). На каждом участке может работать разное оборудование: газовое, сварочное, электрическое. Оборудование имеет номер и название (манометр, газосварочный аппарат и т.п.). Любое оборудование проходит технический осмотр, за год их м.б. несколько. Фиксируется дата осмотра и результат – годен к работе, передать в ремонт, списать и т.п., а также причину нерабочего состояния оборудования (механическая поломка, электропроводка и т.п. – причины отказа). Каждый осмотр проводит какой-либо сотрудник технического отдела, необходимо хранить о нем следующие данные: табельный номер, ФИО, занимаемую на время осмотра должность. Оборудование может отказать в работе и между техническими осмотрами, в таком случае также фиксируется дата, причина, ФИО сотрудника, проведшего соответствующий осмотр на момент отказа.

Необходимо реализовать следующие запросы:

Список отказавшего оборудования на участках предприятия – дата, список - название оборудования, тип оборудования, название участка, причина отказа, дата отказа.

История технических осмотров оборудования с заданным инвентарным номером на предприятии – дата формирования отчета, инвентарный номер, название, тип оборудования результат осмотра.

Фамилии и должности сотрудников технического отдела на заданную дату – дата, список – ФИО, должность.