

Министерство науки и высшего образования Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

ФАКУЛЬТЕТ ДИСТАНЦИОННОГО ОБУЧЕНИЯ (ФДО)

А. И. Исакова, С. М. Левин

**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА
В СЕМЕСТРЕ**

Учебное методическое пособие

Томск
2020

Корректор: А. Н. Миронова

Исакова А. И., Левин С. М.

Научно-исследовательская работа в семестре : учебное методическое пособие / А. И. Исакова, С. М. Левин. – Томск : ФДО, ТУСУР, 2020. – 153 с.

Пособие содержит теоретические и методические материалы для проведения научно-исследовательской работы студентами направления подготовки 09.03.03 «Прикладная информатика» (уровень бакалавриата), обучающимися с применением дистанционных образовательных технологий.

© Исакова А. И.,
Левин С. М., 2020
© Оформление.
ФДО, ТУСУР, 2020

Оглавление

Введение	4
1 Цели и задачи научной работы в учебном процессе	6
2 Организация проведения научной работы студентов	9
2.1 Общие положения о проведении научно-исследовательской работы.....	9
2.2 Задания на научно-исследовательскую работу.....	10
2.3 Требования к содержанию и оформлению отчетов.....	12
2.4 Порядок выполнения и защиты (рецензирования) научно-исследовательской работы.....	16
2.5 Методологические аспекты научно-исследовательской работы	17
2.5.1 Метод проектов в научной работе.....	17
2.5.2 Научный стиль исследовательских работ.....	18
3 Аналоги программного продукта	20
3.1 Характеристика программного продукта	20
3.2 Основные характеристики программных продуктов	22
3.3 Маркетинговые исследования рынка программных средств	25
3.4 Защита программных продуктов	27
4 Среды разработки информационной системы.....	33
4.1 Состав информационных систем.....	33
4.2 Системы управления базами данных (СУБД).....	35
4.2.1 Классификация СУБД	35
4.2.2 Корпоративные СУБД	38
4.2.3 Обоснование выбора СУБД при проектировании информационной системы	40
4.3 Клиентские приложения, обеспечивающие интерфейс пользователя	46
5 Проектирование информационной системы	51
5.1 Этапы создания информационных систем	51
5.2 Последовательность создания информационной модели данных	54
5.3 Методология IDEF0	56
5.4 Физическая и логическая модели данных	65
5.5 Подходы к концептуальному моделированию	67
5.6 Уровни представления диаграмм	70
5.7 Основные правила стандарта IDEF1X	80
5.8 Дополнения к модели и лексические соглашения стандарта IDEF1X	83
Литература.....	85
Приложение А (справочное) Пример отчета по контрольной работе.....	88
Приложение Б (справочное) Пример итогового отчета	123

Введение

Ведущую роль в повышении качества подготовки специалистов в плане развития творческих способностей призвана сыграть научная работа студентов, так как учебный процесс, сливаясь с научным трудом студентов, все более превращается в реальную профессиональную деятельность, которая в настоящее время составляет основу процесса становления будущего специалиста.

Согласно ФГОС ВО, студент бакалавриата по направлению подготовки 09.03.03 «Прикладная информатика» [1] готовится к следующим видам профессиональной деятельности:

- научно-исследовательской;
- аналитической.

Задача высшей школы состоит в том, чтобы сократить период адаптации студентов к учебно-исследовательской и научной работе. Решение этой задачи возможно в том случае, если с первых дней пребывания в вузе студент будет активно участвовать в разнообразных формах научной работы, проводимых кафедрами, факультетами.

Научно-исследовательская деятельность студентов позволяет наиболее полно проявить индивидуальность, творческие способности, готовность к самореализации личности. Несмотря на обширную нормативно-правовую базу в данной области, развитие методологии и методики исследовательской подготовки в высшей школе, на деле данному виду деятельности уделяется недостаточно внимания. Важно определить готовность студентов к научно-исследовательской деятельности. Процесс исследования индивидуален и имеет ценность как в образовательном, так и в личностном смысле, поэтому необходимо совершенствовать подходы к научно-исследовательской работе, для того чтобы сделать этот процесс наиболее интересным и продуктивным.

Проведение научно-исследовательской работы в течение семестра в вузе обеспечивает непрерывное совершенствование учебно-воспитательного процесса на основе фундаментальных и прикладных исследований по существующим направлениям подготовки, а также внедрение в образовательную деятельность современных методик и педагогических технологий.

Целями данного учебного методического пособия являются определение главных составляющих и структуры научно-исследовательской деятельности,

готовности к ней студентов; рассмотрение понятий, уровней, условий формирования, в том числе средствами проблемного обучения, а также примера практической организации научно-исследовательской деятельности студентов.

Работа включает теоретическую часть, в которой рассматриваются понятие, структура научно-исследовательской деятельности и понятие готовности к этой деятельности, и практическую часть, в которой описывается применение системного подхода к организации научно-исследовательской работы в учебном заведении.

В соответствии с учебным планом направления подготовки 09.03.03 «Прикладная информатика» (уровень бакалавриата) научно-исследовательская работа в семестре и подготовка выпускной квалификационной работы (ВКР) являются обязательными элементами подготовки бакалавров.

В настоящем учебно-методическом пособии изложен порядок выполнения научно-исследовательской работы в течение семестра студентами данного направления подготовки кафедры АСУ, получающими квалификацию «бакалавр» [2].

Соглашения, принятые в учебном методическом пособии

Для улучшения восприятия материала в данном учебном методическом пособии используются пиктограммы и специальное выделение важной информации.



.....
Эта пиктограмма означает определение или новое понятие.



.....
 Эта пиктограмма означает «Внимание!». Здесь выделена важная информация, требующая акцента на ней. Автор может поделиться с читателем опытом, чтобы помочь избежать некоторых ошибок.

1 Цели и задачи научной работы в учебном процессе

В связи с усиливающейся информатизацией и интеллектуализацией производственных технологий быстрыми темпами растет объем специальной информации – научной, технической, технологической и т. д. В этих условиях технология обучения, ориентированная на предоставление и усвоение готовых знаний, не может быть признана рациональной и перспективной. Необходимы новые технологии образования, связанные с формированием интеллектуальной культуры и повышением творческих способностей специалиста. Работа, осуществляемая в данном направлении, должна базироваться на педагогической технологии, основанной на концепции творческой деятельности. Наиболее эффективной формой ее реализации в вузе является непрерывная система работы студентов, максимальное приближение ее к учебному процессу.

Научная работа студентов является одним из важнейших средств повышения качества подготовки специалистов с высшим образованием, способных творчески применять в практической деятельности достижения научно-технического прогресса, а следовательно, быстро адаптироваться к современным условиям развития экономики.



.....

Основные цели научно-исследовательской работы в течение семестра – формирование творческих способностей, развитие и совершенствование форм привлечения молодежи к научной деятельности, обеспечивающей единство учебного, научного, воспитательного процессов для повышения профессионально-технического уровня подготовки работников с высшим образованием, а также привитие навыков работы с научно-технической литературой, оформления отчетной документации.

.....

Основными задачами научной работы студента являются:

- обучение методологии рационального и эффективного изучения и использования знаний;
- совершенствование и поиск новых форм интеграции системы высшего образования с наукой и производственной деятельностью в рамках единой системы «учебно-воспитательного процесс – повышение навыков научной, творческой и исследовательской деятельности»;

- участие студентов в научных исследованиях, реальных разработках и техническом творчестве;
- освоение современных технологий в области науки, техники, производства;
- знакомство с современными научными методами, работа с научной литературой;
- выявление способной молодежи для дальнейшего обучения в аспирантуре, работы на кафедрах и в научных лабораториях [2].

В результате научно-исследовательской работы студенты должны *уметь*:

- 1) составлять литературный обзор математических методов и их программной реализации;
- 2) формализованно ставить задачи;
- 3) проводить анализ полученных результатов и давать рекомендации по их использованию;
- 4) представлять результаты в отчете;
- 5) оформлять работу в соответствии с требованиями [2].

Тематика научно-исследовательской работы должна быть актуальна, соответствовать современному состоянию и перспективам развития электронных информационных систем (ЭИС) на базе различных классов ЭВМ и разнообразных средств сбора, передачи и отображения информации.



.....
 Научно-исследовательская работа в семестре является продолжением учебно-исследовательской работы и связана с выпускной квалификационной работой.

При определении задач научной работы следует исходить из реальной потребности организаций, предприятий, банков, фирм в разработке и из возможности внедрения фрагментов будущей выпускной квалификационной работы на предприятии.

Развитие самостоятельности человека, формирование творческой личности – задача социальная, и решаться она должна всеми звеньями системы «школа – вуз», с использованием всего многообразия методов, с учетом индивидуальных особенностей студентов. Ее решение зависит от квалификации, способностей и возможностей профессорско-преподавательского состава, а также от способностей студента.

Под самостоятельной работой в вузе понимают познавательную деятельность, выполняемую студентом самостоятельно под руководством преподавателя. Разработаны методики самостоятельной работы в виде общей и социально-предметных.

Результаты научно-исследовательской работы студента в течение семестра оформляются в виде *двух отчетов, презентаций к каждому отчету* и рецензируются преподавателем. Отчеты должны содержать разделы, указанные в п. 2.2, кратко изложенную теоретическую или практическую часть, полученные результаты и выводы. В конце работы приводится список использованных источников.

Научно-исследовательская работа в течение семестра – глубокое и объемное исследование избранной проблемы, это первая ступень в овладении методикой исследовательской работы. Именно эта работа поможет расширить, обобщить и систематизировать знания по изучаемой проблеме. Выполнение задания поможет овладеть современными методами поиска, обработки и использования информации, освоить методику проведения научно-исследовательской работы.

Обучающийся по направлению подготовки 09.03.03 «Прикладная информатика» (уровень бакалавриата) должен решать следующие профессиональные задачи

в научно-исследовательской деятельности:

- применение системного подхода к информатизации и автоматизации решения прикладных задач, к построению информационных систем на основе современных информационно-коммуникационных технологий и математических методов;
- подготовка обзоров, аннотаций, составление рефератов, научных докладов, публикаций и библиографии по научно-исследовательской работе в области прикладной информатики;

в аналитической деятельности:

- анализ и выбор проектных решений по созданию и модификации информационных систем;
- анализ и выбор программно-технологических платформ и сервисов информационной системы;
- анализ результатов тестирования информационной системы;
- оценка затрат и рисков проектных решений, эффективности информационной системы [1].

2 Организация проведения научной работы студентов

2.1 Общие положения о проведении научно-исследовательской работы

Научно-исследовательская работа (НИР) студентов в течение семестра является продолжением учебного процесса. Для обучающихся на ФДО задания для НИР соотносятся с деятельностью конкретной профильной организации.

Научно-исследовательская работа студентов, включаемая в учебный процесс, выполняется самостоятельно студентом в течение семестра, но под руководством преподавателя и предусматривает:

- выполнение заданий, содержащих элементы научных исследований;
- изучение теоретических основ методики, постановки, организации выполнения научных исследований по дисциплине «Научно-исследовательская работа в семестре».

Названная дисциплина включена в учебный план данного направления подготовки.

Научно-исследовательская работа в семестре завершается обязательной подготовкой *двух отчетов и презентаций к каждому отчету*, в которых представляются основные результаты работы.

По дисциплине «Научно-исследовательская работа в семестре» обучающиеся должны:

- для получения промежуточной аттестации выполнить контрольную работу и предоставить по ней отчет и презентацию. В данном отчете отражается выбор способа автоматизации предметной области в профильной организации;
- для получения итоговой аттестации по дисциплине предоставить отчет по практической разработке ИС: проектированию информационной системы в профильной организации, включающий презентацию.

Научно-исследовательская работа является важным элементом подготовки обучающихся и имеет целью развитие навыков исследовательской и проектной работы, личное получение автором работы аналитических, практических, научных результатов с использованием знаний, приобретенных как в учебном процессе, так и самостоятельно.

Ответственный за научную работу преподаватель кафедры обязан:

- оказывать консультационную и методическую помощь по вопросам, возникающим в процессе выполнения научно-исследовательской работы;
- периодически (по мере необходимости) отвечать на вопросы студентов по обсуждаемой проблеме;
- принять отчеты по заданным темам и презентации, задать вопросы и по результатам ответов оценить работу обучающегося.



.....

Следует помнить, что само по себе изучение предметной области не может являться конечной целью научной работы – работа должна содержать элементы активного, самостоятельного исследования.

.....

Научно-исследовательская работа выполняется студентом индивидуально в течение семестра. Задания на работу соответствуют данному направлению подготовки бакалавров и уровню учебной подготовки студентов. Работа, выполненная в течение семестра, должна обладать тематической и логической завершенностью и быть направлена на решение теоретической либо практической задачи, результаты которой могут принести пользу для деятельности профильной организаций (предприятия).

2.2 Задания на научно-исследовательскую работу

При выполнении научно-исследовательской работы необходимо определить профильную организацию (предприятие), деятельность которой соответствует направлению и профилю подготовки, изучить ее информационное и программное обеспечение, выявить проблемы, которые нужно решить в плане дальнейшей автоматизации. Желательно, чтобы результаты, полученные во время научно-исследовательской работы студента, были продолжением его учебной и учебно-исследовательской работы и могли быть использованы в выпускной квалификационной работе (ВКР).

Научно-исследовательская работа в семестре является одним из важнейших видов деятельности любого вуза. Для обучающихся по направлению подготовки 09.03.03 «Прикладная информатика» (уровень бакалавриата) научно-ис-

следовательская работа предусматривает анализ предметной области и выполнение проектирования информационной системы на основе современных информационно-коммуникационных технологий.

При создании информационной системы в экономике обучающемуся в первую очередь необходимо познакомиться с аналогами информационных систем (ИС), используемых в различных подразделениях организации (например, в отделе кадров, при складском учете, электронном документообороте и т. д.), чтобы иметь представление о том, что применяется в данной организации из сферы программного обеспечения. Далее необходимо проанализировать все возможные среды разработки ИС и обосновать свой выбор. В завершении, приступая к проектированию, надо построить информационно-логическую модель (SADT-модель) и концептуальную модель всех уровней (ER-, KB-, FA-модели).

Таким образом, по дисциплине «Научно-исследовательская работа в семестре» студенты должны предоставить:

- для промежуточной аттестации (получения зачета по контрольной работе) *отчет 1 по контрольной работе*, включающий следующие разделы:
 - 1 «Обзор аналогов информационных систем», в котором рассмотрены информационные системы профильной организации (предприятия);
 - 2 «Обоснование выбора среды разработки и создаваемой ИС», в котором приведены аргументы в пользу избранного способа автоматизации предметной области, а также самой разработки создаваемой (используемой) ИС профильной организации (предприятия);
- для итоговой аттестации по дисциплине (зачета с оценкой) *отчет 2*, включающий следующие разделы:
 - 3 «Построение инфологической модели создаваемой ИС (SADT-модель)», в котором представлен алгоритм построения SADT-модели;
 - 4 «Построение концептуальной модели всех уровней (ER-, KB-, FA-модели)», в котором представлен алгоритм проектирования ER-, KB-, FA-моделей.

Содержание всех вышеперечисленных разделов оформляется в два отчета, содержащих презентации.

В главах 3–5 данного учебного методического пособия приведена справочная информация по каждому из указанных разделов.

2.3 Требования к содержанию и оформлению отчетов

При оформлении отчетов по научно-исследовательской работе предъявляются следующие общие требования согласно образовательному стандарту вуза ОС ТУСУР 01–2013 [2]:

1. Работа должна быть выполнена печатным способом с использованием компьютера. Формат листа А4 (210 × 297 мм).
2. Общий объем текста каждого отчета без приложений должен составлять *от 15 до 25 стр.*, текст должен быть напечатан с интервалом 1,5 шрифтом Times New Roman, размер шрифта 12–14 пунктов.
3. Все страницы, включая иллюстрации и приложения, нумеруются по порядку. Первым листом является титульный лист. Номер листа проставляется *посередине верхнего поля листа* (страницы). На титульном листе номер не проставляется. Нумерация страниц должна быть сквозной от титульного листа до последнего листа текста, включая иллюстрации, таблицы, графики, диаграммы и т. д., расположенные внутри текста или после него, а также приложения.

Текст работы разделяют на разделы и подразделы (или, соответственно, главы и параграфы). Внутри подразделов выделяют пункты, которые при необходимости могут быть разделены на подпункты.

Каждый раздел рекомендуется начинать с нового листа (страницы).

Разделы (за исключением структурных элементов работы «Оглавление», «Сокращения, обозначения, термины и определения» и «Список использованных источников») должны иметь порядковые номера, обозначенные арабскими цифрами и записанные перед соответствующим заголовком.

Разделы и подразделы (главы и параграфы) должны иметь заголовки. Пункты, как правило, заголовков не имеют. Заголовки должны четко и кратко отражать содержание разделов и подразделов (глав и параграфов).

Оформление заголовков должно соответствовать единому стилю форматирования, принятому в работе. Допускается выделение заголовков размером и (или) жирностью шрифта.

Обязательными элементами отчетов по научно-исследовательской работе являются:

- титульный лист;
- задание на выполнение научной работы (перечень разделов, перечисленных в п. 2.2);
- оглавление;

- введение;
- основная часть документа;
- заключение;
- сокращения, обозначения, термины и определения (при необходимости);
- список использованных источников;
- приложение (при необходимости).

Титульный лист служит обложкой документа. Пример оформления титульного листа отчетов по контрольной и итоговой работе по НИР представлен в приложениях А и Б.

Оглавление включает: введение, наименования всех глав, разделов, подразделов, пунктов (если они имеют наименования), заключение, список использованных источников, приложения (при наличии). Строки оглавления заканчиваются указанием *номеров страниц*, на которых расположено *начало* соответствующей части документа.

Заголовок «*Оглавление*» (с прописной буквы) размещают в центре строки (симметрично тексту). Наименования, включенные в оглавление (содержание), записывают строчными буквами, начиная с прописной буквы. Вместо слова «*Оглавление*» допускается использовать наименование «*Содержание*». Содержание включается в общее количество страниц документа.

В разделе «*Введение*» отчета обязательно должны быть обоснованы актуальность, теоретическая и практическая значимость работы, сформулирована цель работы и перечислены задачи, решаемые для достижения поставленной цели. Объем введения, как правило, не превышает 1–2 страниц.

Заголовок «*Введение*» записывают симметрично тексту с прописной буквы. Перед заголовком, как правило, ставят номер раздела, например: «1 Введение».

Основная часть отчета, как правило, состоит из нескольких самостоятельных разделов, каждый из которых характеризуется логической завершенностью и при необходимости может делиться на подразделы и пункты. *Заголовок «Основная часть» в отчете не указывается!* Рекомендуемая структура основной части отчета по НИР представлена в п. 2.2. В основной части содержится обзор рассматриваемой предметной области со ссылками на источники информации.

Рекомендуется в конце каждого раздела формулировать краткие выводы (1–2 абзаца). Разделы основной части отчета должны быть пронумерованы, начиная с первого раздела. Наибольший раздел не должен более чем в 2–3 раза превышать наименьший.

В разделе «*Заключение*» формулируется основной результат работы и (по пунктам) выводы по результатам выполненной работы (как правило, 3–5 выводов), а также указываются вероятные пути и перспективы продолжения работы. Объем заключения, как правило, не превышает 1–2 страниц.

Заголовок «*Заключение*» записывают симметрично тексту с прописной буквы. Перед заголовком, как правило, ставят номер раздела, например: «9 *Заключение*».

Сокращения, обозначения, термины и определения. Если в работе используется значительное количество (более пяти) сокращений, обозначений и (или) нестандартных терминов, соответствующие пояснения рекомендуется выполнять в виде специального раздела «*Сокращения, обозначения, термины и определения*». Наличие специального раздела не исключает расшифровку сокращения или обозначения после первого упоминания в тексте.

Раздел «*Сокращения, обозначения, термины и определения*» оформляют на отдельном листе, помещают его после заключения и указывают в оглавлении работы. Запись сокращений, обозначений, терминов приводят, как правило, в алфавитном порядке. Каждое сокращение, обозначение, термин указывают на новой строке, с абзацного отступа. Через знак «тире» записывают необходимую расшифровку, определение и/или пояснение и завершают строку точкой с запятой, а последнюю строку – точкой.

Список использованных источников содержит библиографическое описание всех литературных источников, использованных в процессе выполнения научной работы. Заголовок «*Список использованных источников*» записывается симметрично тексту с прописной буквы и не нумеруется.

Список оформляется в виде перечня библиографических записей согласно требованиям к библиографическим записям и библиографическим описаниям (ГОСТ 7.1 и ГОСТ 7.0.11).

При ссылках в тексте работы на библиографические источники рекомендуется руководствоваться требованиями к библиографическим ссылкам (ГОСТ 7.0.5).

В список включают все источники, на которые имеются ссылки в работе. Источники в списке нумеруют, как правило, в порядке их упоминания в тексте работы арабскими цифрами без точки.

При ссылке на литературные источники в тексте, начиная с введения и далее, приводится порядковый номер источника, заключенный в квадратные скобки. При необходимости в дополнение к номеру источника указывается номер его раздела, подраздела, страницы, рисунка или таблицы. Например: [2, раздел 3], [6, приложение Б], [24, с. 66, таблица 2.4].

Оформление библиографических списков осуществляется по системе стандартов по информации, библиотечному и издательскому делу (ГОСТ 7.1–2003). Образцы библиографического описания наиболее важных типов литературных источников (с учетом требований нормативных документов) приведены в приложениях А и Б.

В *приложении* могут быть представлены примеры первичных экономических документов (входные и выходные формы и бланки или реальные заполненные документы); информация об объекте исследования (фрагменты громоздких таблиц с описаниями сущностей и атрибутов, различные справочники предприятий и т. д.).

При оформлении текстового материала необходимо соблюдать требования ОС ТУСУР 01–2013 [2]:

- 1) текст отчета должен иметь поля: левое – 30 мм, правое – 15 мм, верхнее – 20 мм, нижнее – 25 мм;
- 2) абзац должен начинаться на расстоянии 1,25 см от левого края страницы;
- 3) каждая глава отчета должна начинаться с новой страницы.

Названия глав, параграфов, пунктов, подпунктов рекомендуется оформлять *по центру*, допускается оформление более крупным шрифтом, чем текст. При этом цифры, указывающие порядковый номер раздела, не должны выступать за границу абзаца. Точка в конце названия не ставится.

Названия глав и параграфов в оглавлении должны соответствовать их наименованию в тексте. Все страницы работы должны соответствовать оглавлению.



.....
 Подчеркивания наименований глав, параграфов и фрагментов текста не допускаются.

Примеры оформления таблиц, рисунков и формул представлены в ОС ТУСУР 01–2013 (<https://regulations.tusur.ru/documents/70>).

Требования к презентации

К каждому отчету оформляется презентация (12–15 слайдов), выполненная в PowerPoint.

В презентации необходимо отразить основные результаты научно-исследовательской работы студента.

Например, *при выполнении отчета по контрольной работе* можно показать скриншоты аналогов рассматриваемых ИС, привести основные выводы по сравнению этих ИС. В большей степени надо указать недостатки рассмотренных аналогов ИС предметной области. Также в этом отчете можно привести краткое описание рассмотренных сред разработки ИС и главное обоснование о выборе конкретных сред.

При *выполнении итогового отчета по дисциплине* можно привести построенные информационно-логические модели (SADT-модели) предметной области уровня А-0 и его детализацию А0, а также концептуальные модели всех уровней (ER-, KB-, FA-модели).

На заключительных слайдах необходимо перечислить основные результаты работы.

2.4 Порядок выполнения и защиты (рецензирования) научно-исследовательской работы

Для получения *зачета по контрольной работе* обучающийся оформляет первый отчет с иллюстративным материалом (презентацией), включающий разделы, указанные в п. 2.2, согласно структуре, представленной в п. 2.3.

Для получения *зачета с оценкой по дисциплине* обучающийся представляет отчет о результатах выполнения научно-исследовательской работы, содержащий разделы, перечисленные в п. 2.2 данного пособия, включающий иллюстративный материал (презентацию).

Отчеты высылаются для проверки преподавателю. После проверки возможно проведение корректировки студентом в соответствии с замечаниями преподавателя.

Рецензирование научной работы студентов

Преподаватель проверяет отчеты, включающие презентации, может задать уточняющие вопросы; после внесения исправлений обучающимся оформляет рецензию на НИР.

На основе проверки научной работы студента преподаватель представляет рецензию, в которой оцениваются:

- актуальность избранной темы, достоверность результатов, ее новизна;
- степень обоснованности научных выводов, результатов.

Объективность оценки предусматривает отражение как положительных, так и отрицательных сторон работы. Отмеченные выше критерии являются основополагающими и при оценке работы студента.



.....
 В случае использования чужого материала без ссылки на автора и источник работа студента не засчитывается.

Студент уже с первых шагов в науке должен иметь четкую установку на важность библиографических ссылок и грамотное оформление списка используемой литературы.

2.5 Методологические аспекты научно-исследовательской работы

2.5.1 Метод проектов в научной работе

Научно-исследовательская работа направлена на получение аналитических результатов, относящихся к изучению выбранной предметной области, объектов, их характеристики. Особое значение имеют формы и методы научной работы. Остановимся на одном из самых продуктивных методов – методе проектов.

Метод проектов. В основу метода проектов положена идея, составляющая суть понятия «проект», его прагматическую направленность на результат, который можно получить при решении той или иной практически или теоретически значимой проблемы. Метод проектов всегда ориентирован на самостоятельную деятельность учащихся – индивидуальную, парную, групповую, которую учащиеся выполняют в течение определенного отрезка времени. Этот метод органично сочетается с индивидуальным подходом к обучению.

Метод проектов всегда предполагает решение какой-то проблемы. Решение проблемы предусматривает, с одной стороны, использование совокупности

разнообразных методов, средств обучения, а с другой – необходимость интегрирования знаний, умений из различных областей науки, техники, технологии, творческих областей.

По сути дела, проект является своего рода «диссертацией студента». Необходимым условием метода проектов является наличие значимой в исследовательском, творческом плане проблемы/задачи, требующей интегрированного знания, исследовательского поиска для ее решения (например, исследование и проведение предпроектной работы по созданию и разработке информационной системы и пр.).

Особенности метода проектов:

1. Самостоятельная (индивидуальная, парная, групповая) деятельность учащихся.
2. Структурирование содержательной части проекта (с указанием поэтапных результатов).
3. Использование исследовательских методов, предусматривающих определенную последовательность действий:
 - определение проблемы и вытекающих из нее задач исследования (использование в ходе совместного исследования метода «мозговой атаки», «круглого стола»);
 - выдвижение гипотез их решения;
 - обсуждение методов исследования (статистических методов, экспериментальных наблюдений и пр.);
 - обсуждение способов оформления конечных результатов (презентаций, творческих отчетов, просмотров и пр.). Сбор, систематизация и анализ полученных данных;
 - подведение итогов, оформление результатов, их презентация;
 - выводы, выдвижение новых проблем исследования.

2.5.2 Научный стиль исследовательских работ

Научно-исследовательские работы (тезисы докладов, статьи, отчеты по практикам и т. д.) студентов имеют большое значение в формировании профессионализма будущего специалиста. Каждая из них – это самостоятельное научно-прикладное исследование, которое является одной из форм отчетности и контроля знаний студентов, доказательством приобретения знаний по избранной проблеме, творческим осмыслением соответствующей научной мысли.

Результаты научной работы могут быть представлены только в письменной форме и должны характеризоваться наличием научного стиля.

Основные черты научного стиля:

- точность, ясность и полнота высказывания;
- лаконичность в выражении мысли;
- широкое использование абстрактной лексики;
- употребление слов в конкретном значении;
- безличность;
- монологический характер высказывания;
- последовательность;
- тесная связь отдельных частей высказывания (с помощью развернутых синтаксических конструкций с причастными, деепричастными оборотами, перечислениями);
- использование специфической терминологии, условных знаков и обозначений.

3 Аналоги программного продукта

3.1 Характеристика программного продукта

Многие профессиональные программные продукты стоят недешево, хотя в последние годы относительная стоимость массового программного обеспечения постоянно снижается. На рынке для многих типов программных продуктов существуют аналоги, которые, при тех или иных условиях, могут использоваться бесплатно. Разумеется, на адаптацию и освоение таких аналогов приходится тратить время, силы и в ряде случаев финансы. Но все же следует признать, что такой путь куда честнее и безопаснее, чем использование контрафактных программ под надуманным оправданием «очень дорого» [3].

Все программы по характеру использования и категориям пользователей можно разделить на два класса: утилитарные программы и программные продукты (изделия) (рис. 3.1).



Рис. 3.1 – Классификация программ по категориям пользователей

Утилитарные программы («программы для себя») предназначены для удовлетворения нужд их разработчиков. Чаще всего утилитарные программы выполняют роль сервиса в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения.

Программные продукты (изделия) предназначены для удовлетворения потребностей пользователей, широкого распространения и продажи.

В настоящее время существуют и другие варианты легального распространения программных продуктов, которые появились с использованием глобальных или региональных телекоммуникаций:

- *freeware* – бесплатные программы, свободно распространяемые, поддерживаются самим пользователем, который правомочен вносить в них необходимые изменения;

- *shareware* – некоммерческие (условно-бесплатные) программы, которые могут использоваться, как правило, бесплатно. При условии регулярного использования подобных продуктов осуществляется взнос определенной суммы.

Ряд производителей используют *OEM-программы* (*Original Equipment Manufacturer*), т. е. встроенные программы, устанавливаемые на компьютеры или поставляемые вместе с вычислительной техникой.

Программный продукт должен быть соответствующим образом подготовлен к эксплуатации, иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы программы, иметь товарный знак изготовителя, а также желательно наличие кода государственной регистрации. Только при таких условиях созданный программный комплекс может быть назван программным продуктом.

Программный продукт – комплекс взаимосвязанных программ для решения определенной проблемы (задачи) массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Путь от «программ для себя» до программных продуктов достаточно долгий, он связан с изменениями технической и программной среды разработки и эксплуатации программ, с появлением и развитием самостоятельной отрасли – информационного бизнеса, для которой характерны разделение труда фирм – разработчиков программ, их дальнейшая специализация, формирование рынка программных средств и информационных услуг.

Программные продукты могут создаваться:

- как индивидуальная разработка под заказ;
- разработка для массового распространения среди пользователей.

При индивидуальной разработке фирма-разработчик создает оригинальный программный продукт, учитывающий специфику обработки данных для конкретного заказчика.

При разработке для массового распространения фирма-разработчик должна обеспечить, с одной стороны, универсальность выполняемых функций обработки данных, с другой стороны, гибкость и настраиваемость программного продукта на условия конкретного применения. Отличительной особенностью программных продуктов должна быть их системность – функциональная полнота и законченность реализуемых функций обработки, которые применяются в совокупности.

Программный продукт разрабатывается на основе промышленной технологии выполнения проектных работ с применением современных инструментальных средств программирования. Специфика заключается в уникальности процесса разработки алгоритмов и программ, зависящего от характера обработки информации и используемых инструментальных средств. На создание программных продуктов затрачиваются значительные ресурсы: трудовые, материальные, финансовые; требуется высокая квалификация разработчиков.

Как правило, программные продукты требуют сопровождения, которое осуществляется специализированными фирмами – распространителями программ (дистрибьюторами), реже – фирмами-разработчиками. Сопровождение программ массового применения сопряжено с большими трудозатратами – исправление обнаруженных ошибок, создание новых версий программ и т. п.

Сопровождение программного продукта – поддержка работоспособности программного продукта, переход на его новые версии, внесение изменений, исправление обнаруженных ошибок и т. п.

Программные продукты в отличие от традиционных программных изделий не имеют строго регламентированного набора качественных характеристик, задаваемых при создании программ, либо эти характеристики невозможно заранее точно указать или оценить, т. к. одни и те же функции обработки, обеспечиваемые программным средством, могут иметь различную глубину проработки. Даже время и затраты на разработку программных продуктов не могут быть определены с большой степенью точности заранее.

3.2 Основные характеристики программных продуктов

Основные характеристики программ:

- алгоритмическая сложность (логика алгоритмов обработки информации);
- состав и глубина проработки реализованных функций обработки;
- полнота и системность функций обработки;
- объем файлов программ;
- требования к операционной системе и техническим средствам обработки со стороны программного средства;
- объем дисковой памяти;
- размер оперативной памяти для запуска программ;
- тип процессора;

- версия операционной системы;
- наличие вычислительной сети и др. [4].

Программные продукты отличаются многообразием показателей качества, которые отражают следующие аспекты:

- насколько хорошо (просто, надежно, эффективно) можно использовать программный продукт;
- насколько легко эксплуатировать программный продукт;
- можно ли использовать программный продукт при изменении условия его применения и др.

Мобильность программных продуктов означает их независимость от технического комплекса системы обработки данных, операционной среды, сетевой технологии обработки данных, специфики предметной области и т. п.

Дерево характеристик качества программных продуктов представлено на рисунке 3.2.

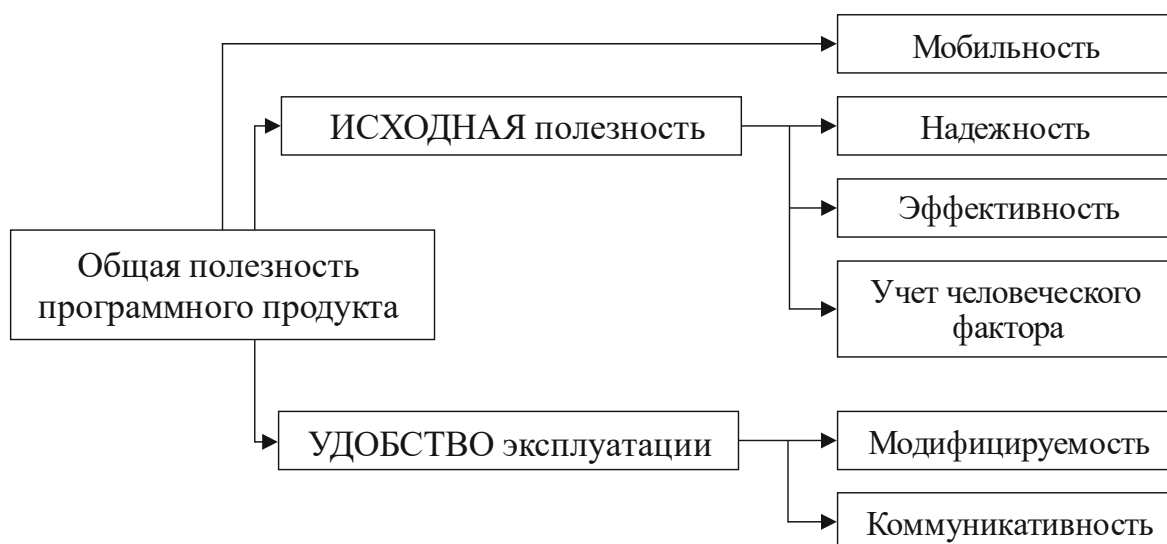


Рис. 3.2 – Дерево характеристик качества программных продуктов

Мобильный (многоплатформный) программный продукт может быть установлен на различных моделях компьютеров и операционных систем, без ограничений на его эксплуатацию в условиях вычислительной сети. Функции обработки такого программного продукта пригодны для массового использования без каких-либо изменений.

Надежность работы программного продукта определяется устойчивостью в работе программ, точностью выполнения предписанных функций обработки, возможностью диагностики возникающих в процессе работы программ ошибок.

Эффективность программного продукта оценивается как с позиций прямого его назначения – требований пользователя, так и с точки зрения расхода вычислительных ресурсов, необходимых для его эксплуатации.

Расход вычислительных ресурсов оценивается через объем внешней памяти для размещения программ и объем оперативной памяти для запуска программ.

Учет человеческого фактора означает обеспечение дружественного интерфейса для работы конечного пользователя, наличие контекстно зависимой подсказки или обучающей системы в составе программного средства, хорошей документации для освоения и использования заложенных в программном средстве функциональных возможностей, анализ и диагностику возникших ошибок и др.

Модифицируемость программных продуктов означает способность к внесению изменений, например расширение функций обработки, переход на другую техническую базу обработки и т. п.

Коммуникативность программных продуктов основана на максимально возможной их интеграции с другими программами, обеспечении обмена данными в общих форматах представления (экспорт/импорт баз данных, внедрение или связывание объектов обработки и др.).

В условиях существования рынка программных продуктов важными характеристиками являются:

- стоимость;
- количество продаж;
- время нахождения на рынке (длительность продаж);
- известность фирмы-разработчика и программы;
- наличие программных продуктов аналогичного назначения.

Программные продукты массового распространения продаются по ценам, которые учитывают спрос и конъюнктуру рынка (наличие и цены программ-конкурентов). Большое значение имеет проводимый фирмой маркетинг, который включает:

- формирование политики цен для завоевания рынка;
- широкую рекламную кампанию программного продукта;
- создание торговой сети для реализации программного продукта (так называемые дилерские и дистрибьюторные центры);

- обеспечение сопровождения и гарантийного обслуживания пользователей программного продукта, создание горячей линии (оперативный ответ на возникающие в процессе эксплуатации программных продуктов вопросы);
- обучение пользователей программного продукта.

Спецификой программных продуктов (в отличие от большинства промышленных изделий) является также и то, что их эксплуатация должна выполняться на правовой основе – лицензионные соглашения между разработчиком и пользователями с соблюдением авторских прав разработчиков программных продуктов.

3.3 Маркетинговые исследования рынка программных средств

Если программный продукт создается под заказ и предполагается выход на рынок программных средств, маркетинг выполняется в полном объеме: изучаются программные продукты-конкуренты и аналоги, обобщаются требования пользователей к программному продукту, устанавливается потенциальная емкость рынка сбыта, дается прогноз цены и объема продаж. Кроме того, важно оценить необходимые для разработки программного продукта материальные, трудовые и финансовые ресурсы, ориентировочную длительность основных этапов жизненного цикла программного продукта [3–4].

Проектирование структуры программного продукта связано с алгоритмизацией процесса обработки данных, детализацией функций обработки, разработкой структуры программного продукта (архитектуры программных модулей) и/или структуры информационной базы (базы данных) задачи, выбором методов и средств создания программ-технологии программирования.

Программирование, тестирование и отладка программ являются технической реализацией проектных решений и выполняются с помощью выбранного инструментария разработчика (алгоритмические языки и системы программирования, инструментальные среды разработчиков и т. п.).

Для больших и сложных программных комплексов, имеющих развитую модульную структуру построения, отдельные работы данного этапа могут выполняться параллельно, обеспечивая сокращение общего времени разработки программного продукта. Важная роль принадлежит используемым при этом инструментальным средствам программирования и отладки программ, поскольку они влияют на трудоемкость выполнения работ, их стоимость, качество создаваемых программ.

Документирование программного продукта является обязательным видом работ, выполняемых, как правило, не самим разработчиком, а лицом, связанным с распространением и внедрением программного продукта. Документация должна содержать необходимые сведения по установке и обеспечению надежной работы программного продукта, поддерживать пользователей при выполнении функций обработки, определять порядок комплексирования программного продукта с другими программами. Успех распространения и эксплуатации программного продукта в значительной степени зависит от качества его документации [4].

На машинном уровне программного продукта, как правило, создаются:

- автоматизированная контекстно зависимая помощь (HELP);
- демонстрационные версии, работающие в активном режиме по типу обучающих систем (электронный учебник) или в пассивном режиме (ролик, мультфильм); они предназначены для представления функциональных возможностей программного продукта и информационной технологии его использования.

Требуется постоянная программа маркетинговых мероприятий и поддержки программных продуктов. Вначале уровень продаж программного продукта идет вверх, затем наступает стабилизация продаж программного продукта. Фирма-разработчик стремится к максимальной длительности периода стабильных продаж на высоком уровне. Далее происходит падение объема продаж, что является сигналом к изменению маркетинговой политики фирмы в отношении данного программного продукта, требуется модификация данного продукта, изменение цены или снятие с продажи.

Эксплуатация программного продукта идет параллельно с его сопровождением, при этом эксплуатация программ может начинаться и в случае отсутствия сопровождения или продолжаться в случае завершения сопровождения еще какое-то время. После снятия программного продукта с продажи его сопровождение может выполняться еще некоторое время. В процессе эксплуатации программного продукта производится устранение обнаруженных ошибок.

Снятие программного продукта с продажи и отказ от сопровождения происходят, как правило, в случае изменения технической политики фирмы-разработчика, неэффективности работы программного продукта, наличия в нем неустраняемых ошибок, отсутствия спроса.

Длительность жизненного цикла для различных программных продуктов неодинакова. Для большинства современных программных продуктов длительность жизненного цикла измеряется в годах (2–3 года). Хотя достаточно часто

встречаются на компьютерах и давно снятые с производства программные продукты [4].

Особенность разработки программного продукта заключается в том, что на начальных этапах принимаются решения, реализуемые на последующих этапах. Допущенные ошибки, например при спецификации требований к программному продукту, приводят к огромным потерям на последующих этапах разработки или эксплуатации программного продукта и даже к неучасу всего проекта. Так, при необходимости внесения изменений в спецификацию программного продукта следует повторить в полном объеме все последующие этапы проектирования и создания программного продукта.

3.4 Защита программных продуктов

Программные продукты и компьютерные базы данных являются предметом интеллектуального труда специалистов высокой квалификации. Процесс проектирования и реализации программных продуктов характеризуется значительными материальными и трудовыми затратами, основан на использовании наукоемких технологий и инструментария, требует применения и соответствующего уровня дорогостоящей вычислительной техники. Это обуславливает необходимость принятия мер по защите интересов разработчика программ и создателей компьютерных баз данных от несанкционированного их использования [5–7].

Программное обеспечение является объектом защиты также и в связи со сложностью и трудоемкостью восстановления его работоспособности, значимостью программного обеспечения для работы информационной системы.

Защита программного обеспечения преследует две цели [5]:

- 1) ограничение несанкционированного доступа к программам или предотвращение их преднамеренного разрушения и хищения;
- 2) исключение несанкционированного копирования (тиражирования) программ.

Программный продукт и базы данных должны быть защищены по нескольким направлениям от воздействия:

- 1) человека – хищение машинных носителей и документации программного обеспечения; нарушение работоспособности программного продукта и др.;
- 2) аппаратуры – подключение к компьютеру аппаратных средств для считывания программ и данных или их физического разрушения;

- 3) специализированных программ – приведение программного продукта или базы данных в неработоспособное состояние (например, вирусное заражение), несанкционированное копирование программ и базы данных и т. д.

Самый простой и доступный способ защиты программных продуктов и базы данных – ограничение доступа.

Контроль доступа к программному продукту и базе данных строится путем [6]:

- парольной защиты программ при их запуске;
- использования ключевого внешнего носителя для запуска программ;
- ограничения программ или данных, функций обработки, доступных пользователям, и др.

Могут также использоваться и криптографические методы защиты информации базы данных или головных программных модулей.

Программные системы защиты от несанкционированного копирования предотвращают нелегальное использование программных продуктов и баз данных. Программа выполняется только при опознании некоторого уникального не копируемого ключевого элемента.

Таким ключевым элементом могут быть:

- внешний носитель, на котором записан не подлежащий копированию ключ;
- определенные характеристики аппаратуры компьютера;
- специальное устройство (электронный ключ), подключаемое к компьютеру и предназначенное для выдачи опознавательного кода.

Программные системы защиты от копирования программных продуктов:

- идентифицируют среду, из которой будет запускаться программа;
- устанавливают соответствие среды, из которой запущена программа, той, для которой разрешен санкционированный запуск;
- вырабатывают реакцию на запуск из несанкционированной среды;
- регистрируют санкционированное копирование;
- противодействуют изучению алгоритмов и программ работы системы [5].

Идентификация среды компьютера обеспечивается за счет:

- 1) закрепления месторасположения программ на жестком магнитном диске (так называемые неперемещаемые программы);

- 2) привязки к номеру BIOS (расчет и запоминание с последующей проверкой при запуске контрольной суммы системы);
- 3) привязки к аппаратному (электронному) ключу, вставляемому в порт ввода-вывода, и др. [7].

На Западе наиболее популярны методы правовой защиты программных продуктов и баз данных, которые включают:

- патентную защиту;
- закон о производственных секретах;
- лицензионные соглашения и контракты;
- закон об авторском праве [5].

Различают две категории прав:

- экономические права, дающие их обладателям право на получение экономических выгод от продажи или использования программных продуктов и баз данных;
- моральные права, обеспечивающие защиту личности автора в его произведении.

Во многих странах несанкционированное копирование программ в целях продажи или бесплатного распространения рассматривается как государственное преступление, карается штрафом или тюремным заключением. Но, к сожалению, само авторское право не обеспечивает защиту новой идеи, концепции, методологии и технологии разработки программ, поэтому требуются дополнительные меры их защиты.

Патентная защита устанавливает приоритет в разработке и использовании нового подхода или метода, примененного при разработке программ, удостоверяет их оригинальность.

Статус производственного секрета для программы ограничивает круг лиц, знакомых или допущенных к ее эксплуатации, а также определяет меру их ответственности за разглашение секретов. Например, используется парольный доступ к программному продукту или базе данных, вплоть до паролей на отдельные режимы (чтение, запись, корректировку и т. п.). Программы, как любой материальный объект большой стоимости, необходимо охранять от кражи и преднамеренных разрушений.

Лицензионные соглашения распространяются на все аспекты правовой охраны программных продуктов, включая авторское право, патентную защиту, производственные секреты. Наиболее часто используются лицензионные соглашения на передачу авторских прав.

Лицензия – договор на передачу одним лицом (лицензиаром) другому лицу (лицензиату) права на использование имени, продукции, технологии или услуги. Лицензиар увеличивает свои доходы сбором лицензионных платежей, расширяет область распространения программного продукта или базы данных; лицензиат извлекает доходы за счет их применения.

В лицензионном соглашении оговариваются все условия эксплуатации программ, в том числе создание копий. На каждой копии программы должны быть те же отметки, что и на оригинале:

- знак авторского права (обычно ©) и название разработчика, года выпуска программы, прочих ее атрибутов;
- знак патентной защиты или производственного секрета;
- торговые марки, соответствующие использованным в программе другим программным изделиям (обычно – ТМ и название фирмы-разработчика программного продукта);
- символ зарегистрированного права на распространение программного продукта (обычно ®).

Существует несколько типов лицензий на программные продукты.

Исключительная лицензия – продажа всех имущественных прав на программный продукт или базу данных. Покупателю лицензии предоставляется исключительное право на их использование, а автор или владелец патента отказывается от самостоятельного их применения или предоставления другим лицам.

Это самый дорогой вид лицензии, к нему прибегают для монопольного владения с целью извлечения дополнительной прибыли либо с целью прекращения существования на рынке программных средств программного продукта.

Простая лицензия – лицензиар предоставляет право лицензиату использовать программный продукт или базу данных, оставляя за собой право применять их и предоставлять на аналогичных условиях неограниченному числу лиц (лицензиат при этом не может сам выдавать сублицензии, может лишь продать копии приобретенного программного продукта или базы данных).

Такой вид лицензии приобретают дилер (торговец) либо фирмы-производители, использующие купленные лицензии как сопутствующий товар к основному виду деятельности. Например, многие производители и фирмы, торгующие компьютерной техникой, осуществляют продажу вычислительной техники с установленным лицензионным программным обеспечением (операционная система, текстовый редактор, электронная таблица, графические пакеты и т. д.).

Этикеточная лицензия – лицензия на одну копию программного продукта или базы данных. Данный тип лицензии применяется при розничной продаже. Каждый официальный покупатель заключает лицензионное соглашение с продавцом на их использование, но при этом сохраняется авторское право разработчика.

Экономические отношения между лицензиаром и лицензиатом могут строиться различным образом. За право пользования программным продуктом или базой данных выплачивается единовременное вознаграждение (паушальный платеж), которое и является фактической ценой лицензии. Возможны и периодические отчисления лицензиару за право пользования в виде *роялти* (фиксированная ставка в определенные интервалы времени в течение действия лицензионного соглашения, как правило, процент от стоимости программных продуктов или баз данных).

Закон об охране программных продуктов и компьютерных баз данных автором признает физическое лицо, в результате творческой деятельности которого они созданы. Автору независимо от его имущественных прав принадлежат личные авторские права: авторство, имя, неприкосновенность (целостность) программ или баз данных.

Авторское право действует с момента создания программного продукта или базы данных в течение всей жизни автора и 50 лет после его смерти. Автор может:

- выпускать в свет; распространять и модифицировать;
- воспроизводить в любой форме, любыми способами;
- осуществлять любое иное использование программного продукта или базы данных [4].

Авторское право не связано с правом собственности на материальный носитель. Имущественные права на программный продукт или базу данных могут быть переданы частично или полностью другим физическим или юридическим лицам по договору. Имущественные права относятся к категории наследуемых. Если программный продукт или база данных созданы в порядке выполнения служебных обязанностей, имущественные права принадлежат работодателю.

Программные продукты и базы данных могут использоваться третьими лицами – пользователями на основании договора с правообладателем.

Лицо, правомерно владеющее экземпляром программы или базы данных, вправе без получения дополнительного разрешения правообладателя осуществлять любые действия, связанные с функционированием программного продукта или базы данных в соответствии с ее назначением, в том числе:

- исправлять явные ошибки;
- адаптировать программный продукт или базу данных;
- изготавливать страховые копии и т. д.

4 Среды разработки информационной системы

4.1 Состав информационных систем



Информационная система – это взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Современное понимание информационной системы предполагает использование в качестве основного технического средства переработки информации компьютера. Кроме того, техническое воплощение информационной системы само по себе ничего не будет значить, если не учтена роль человека, для которого предназначена производимая информация и без которого невозможно ее получение и представление [8].

Необходимо понимать разницу между компьютерами и информационными системами. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для информационных систем. Информационная система немислима без персонала, взаимодействующего с компьютерами и телекоммуникациями.

В нормативно-правовом смысле информационная система определяется как «совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств»¹.

В целом информационные системы определяются следующими свойствами:

- любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем;
- информационная система является динамичной и развивающейся;
- при построении информационной системы необходимо использовать системный подход;
- выходной продукцией информационной системы является информация, на основе которой принимаются решения;

¹ФЗ РФ «Об информации, информационных технологиях и о защите информации» от 27.07.2006 № 149-ФЗ.

- информационную систему следует воспринимать как человеко-машинную систему обработки информации [9].

Внедрение информационных систем может способствовать:

- получению более рациональных вариантов решения управленческих задач за счет внедрения математических методов;
- освобождению работников от рутинной работы за счет ее автоматизации;
- обеспечению достоверности информации;
- совершенствованию структуры информационных потоков (включая систему документооборота);
- предоставлению потребителям уникальных услуг;
- уменьшению затрат на производство продуктов и услуг (включая информационные) [10].

Общую структуру информационной системы можно рассматривать как совокупность подсистем независимо от сферы применения. В этом случае говорят о структурном признаке классификации, а подсистемы называют обеспечивающими. Таким образом, структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем, среди которых обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечение.



.....

***Информационное обеспечение** – совокупность единой системы классификации и кодирования информации, унифицированных систем документации, схем информационных потоков, циркулирующих в организации, а также методология построения баз данных.*

.....

Назначение подсистемы *информационного обеспечения* состоит в своевременном формировании и выдаче достоверной информации для принятия управленческих решений.

Для создания информационного обеспечения необходимо:

- ясное понимание целей, задач, функций всей системы управления организацией;
- выявление движения информации от момента возникновения и до ее использования на различных уровнях управления, представленной для анализа в виде схем информационных потоков;

- совершенствование системы документооборота;
- наличие и использование системы классификации и кодирования;
- владение методологией создания концептуальных информационно-логических моделей, отражающих взаимосвязь информации;
- создание массивов информации на машинных носителях, что требует наличия современного технического обеспечения.

4.2 Системы управления базами данных (СУБД)

4.2.1 Классификация СУБД

По степени универсальности различают два класса СУБД:

- 1) системы общего назначения;
- 2) специализированные системы [11].

СУБД общего назначения не ориентированы на какую-либо предметную область или на информационные потребности какой-либо группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной системе, и поставляется многим пользователям как коммерческое изделие. Такие СУБД обладают средствами настройки на работу с конкретной базой данных. Использование СУБД общего назначения в качестве инструментального средства для создания автоматизированных информационных систем, основанных на технологии баз данных, позволяет существенно сокращать сроки разработки, экономить трудовые ресурсы. Этим СУБД присущи развитые функциональные возможности и даже определенная функциональная избыточность.

Специализированные СУБД создаются в редких случаях при невозможности или нецелесообразности использования СУБД общего назначения [12].

СУБД общего назначения – это сложные программные комплексы, предназначенные для выполнения всей совокупности функций, связанных с созданием и эксплуатацией базы данных информационной системы.

Рынок программного обеспечения ПК располагает большим числом разнообразных по своим функциональным возможностям коммерческих систем управления базами данных общего назначения, а также средствами их окружения практически для всех массовых моделей машин и для различных операционных систем.

Используемые в настоящее время СУБД обладают средствами обеспечения целостности данных и надежной безопасности, что дает возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование. Продукты, функционирующие в среде Windows, выгодно отличаются удобством пользовательского интерфейса и встроенными средствами повышения производительности.

Рассмотрим основные характеристики некоторых СУБД – лидеров на рынке программ, предназначенных как для разработчиков информационных систем, так и для конечных пользователей.

По технологии обработки данных базы данных подразделяются на централизованные и распределенные [11, 12].

Централизованная база данных хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе. Этот способ использования баз данных часто применяют в локальных сетях ПК.

Распределенная база данных состоит из нескольких, возможно пересекающихся или даже дублирующих друг друга частей, хранимых в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По способу доступа к данным базы данных разделяются на базы данных с *локальным доступом* и базы данных с *удаленным (сетевым) доступом*. Системы централизованных баз данных с сетевым доступом предполагают различные *архитектуры* подобных систем:

- файл-сервер;
- клиент-сервер [12].

Файл-сервер. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (сервер файлов). На такой машине хранится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих станций, с помощью которых поддерживается доступ пользовательской системы к централизованной базе данных. Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где в основном и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать на рабочих станциях локальные БД, которые используются ими монополично. Концепция «файл-сервер» условно отображена на рисунке 4.1.

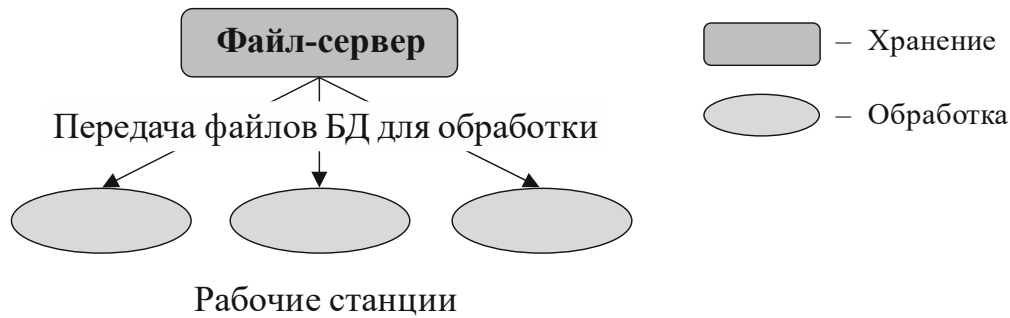


Рис. 4.1 – Схема обработки информации в БД по принципу «файл-сервер»

Клиент-сервер. В этой концепции подразумевается, что помимо хранения централизованной базы данных центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Спецификой архитектуры «клиент-сервер» является использование языка запросов SQL. Концепция «клиент-сервер» условно изображена на рисунке 4.2.

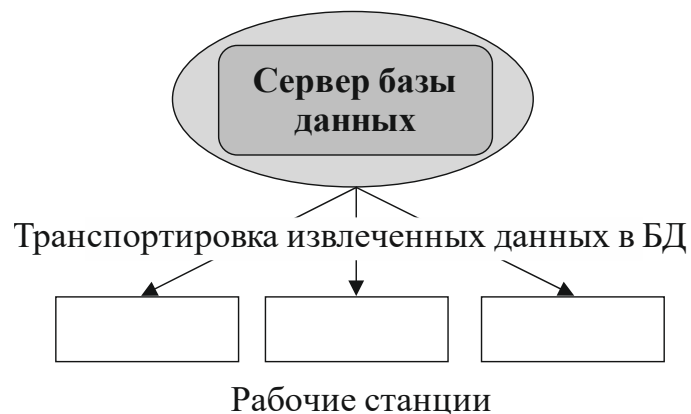


Рис. 4.2 – Схема обработки информации в БД по принципу «клиент-сервер»

Системы управления базами данных разделяются на серверные и клиентские [13].

Примеры серверных СУБД: Caché, DB2, Firebird, Informix, Ingres, InterBase, MSDE, MS SQL Server, MySQL, mSQL, Oracle, Pervasive SQL, PostgreSQL, Sybase ASE, Sybase ASA, Sybase IQ, Teradata, ЛИНТЕР, Mnesia, H2.

Примеры клиентских СУБД: DataFlex, dBase, MS Access, Paradox, OpenOffice.org Base, Sav Zigzag.

4.2.2 Корпоративные СУБД

Функционирование практически любого современного предприятия немыслимо без манипуляции данными, связанными с его производственной деятельностью. Нередко эффективность его деятельности и конкурентоспособность на рынке товаров или услуг непосредственно связаны с тем, актуальны ли эти данные и доступны ли они обращающимся к ним пользователям (причем нередко не только пользователям локальной сети, но и посетителям корпоративного веб-сервера и сотрудникам, обращающимся к ним с помощью мобильных устройств). С этой целью применяются различные архитектуры физического хранения данных, такие как *Storage Area Network (SAN)* или *Network Attached storage (NAS)*, а также системы управления базами данных, предназначенные для логической организации данных и осуществления доступа к ним.

Корпоративные данные большинства компаний сейчас хранятся в реляционных СУБД.

В простейшем случае корпоративная информационная система, использующая СУБД, состоит из двух основных компонентов: сервера баз данных, управляющего данными и выполняющего поступающие от клиентских приложений запросы, и самих клиентских приложений, обеспечивающих интерфейс пользователя и посылающих запросы к серверу. Именно сервер баз данных может манипулировать файлами, в которых хранятся данные, выполнять пользовательские запросы, поддерживать ссылочную целостность данных, обеспечивать доступ к ним, осуществлять резервное копирование данных и протоколировать операции, связанные с их изменением.

К современным реляционным СУБД предъявляются следующие требования:

- *масштабируемость*, то есть способность одновременно обслуживать большее количество пользовательских запросов с той же скоростью при пропорциональном этому количеству увеличении объема предоставляемых ресурсов (процессоров, оперативной памяти и т. д.);
- *доступность*, то есть постоянная возможность получения ответа на запрос;
- *надежность*, то есть минимальная вероятность сбоев, а также наличие средств восстановления данных после сбоев, резервирования и дублирования;
- *управляемость*, то есть простота администрирования и конфигурирования, а нередко и наличие средств автоматического конфигурирования

(обычно набор средств администрирования включает средства создания баз данных и их объектов, инструменты репликации данных между различными серверами, утилиты управления пользователями и группами, средства мониторинга событий, средства просмотра планов выполнения запросов, утилиты миграции из других СУБД);

- *наличие средств защиты* данных от потери и несанкционированного доступа;
- *поддержка стандартных механизмов доступа к данным* (ODBC, JDBC, OLE DB, ADO).

Как правило, отсутствие какого-либо из этих признаков приводит к тому, что даже у неплохой по другим потребительским свойствам СУБД область применения оказывается весьма ограниченной.

Так, СУБД с плохой масштабируемостью, успешно применявшаяся при небольшом обрабатываемом объеме данных, оказывается непригодной при его росте, и нередко ее приходится заменять на другую; при этом неизбежны определенные затраты на переписывание серверного кода. Лишние затраты на администрирование обычно тоже никому не нужны. Плохие масштабируемость и доступность влекут за собой дополнительные затраты рабочего времени сотрудников, простои, а также потерю компанией клиентов, отчаявшихся дожидаться нужных данных на корпоративном сайте и вынужденных обратиться на сайт конкурента.

Именно поэтому лидеры рынка корпоративных СУБД стремятся к производству продуктов, удовлетворяющих всем вышеуказанным требованиям. Кроме того, как правило, подобные продукты существуют для нескольких платформ, а нередко и в разных редакциях, предназначенных для решения разнообразных задач или обслуживания различного количества данных и пользователей. Из последних тенденций развития корпоративных СУБД следует отметить поддержку XML и веб-сервисов XML.

Существует два типа реляционных баз данных:

- 1) оперативные, или OLTP (OLTP – *On-Line Transaction Processing*), базы данных. Обычно в эти базы данных осуществляется интенсивный ввод данных, а вот число адресованных к ним запросов невелико;
- 2) хранилища данных, применяемые, как правило, в аналитических приложениях и системах поддержки принятия решений. К ним обычно адресуется большое число запросов, но ввод данных в них не столь интенсивен [13].

Отметим, что многие современные СУБД с успехом поддерживают создание баз данных обоих типов – все определяется тем, как будет спроектирована структура данных. Однако нередко для создания хранилищ данных применяются специальные СУБД, способ хранения данных в которых особым образом оптимизирован для ускорения выполнения запросов. И, как правило, создание OLAP-хранилищ, основанных на нереляционных многомерных базах данных, требует наличия отдельных серверных продуктов.

В заключение отметим, что существующие на сегодняшний день возможности СУБД ведущих производителей отражают современные тенденции развития информационных систем, такие как использование многопроцессорных систем и распределенной обработки данных, создание распределенных систем, применение средств быстрой разработки приложений, создание систем поддержки принятия решений с использованием аналитической обработки данных, а также все возрастающие требования к надежности и отказоустойчивости.

4.2.3 Обоснование выбора СУБД при проектировании информационной системы

Ниже приведены основные требования к СУБД, которые надо учитывать при проектировании информационной системы.

1. Основные показатели СУБД. Система управления базами данных отвечает за агрегирование данных и их последующее хранение и обработку.

СУБД управляется на языках работы с базами данных (БД), например SQL (*Structured Query Language*). СУБД основаны на реляционной модели данных. Реляционная модель – представление БД в виде таблиц для действий над записями на языке SQL. Реляционные системы – это системы «автоматической навигации». SQL – более абстрактный язык, чем C++, т. к. способ запроса остается на выбор оптимизатора СУБД. «Постреляционная СУБД» – наличие в реляционной СУБД файлов управления данными, не вписывающихся в реляционную модель, т. е. объектов.

Ранее данные хранились только в алфавитно-цифровой форме, классифицировались по стандартным типам (строки, целые числа и т. д.). Теперь сюда включаются и бинарные объекты: изображения, видео и большие фрагменты текста, по которым может происходить поиск.

Другим необходимым элементом СУБД является встроенный язык программирования для автоматизации процедур обслуживания системы и обработ-

ки данных внутри СУБД ее собственными средствами. Пользовательские приложения взаимодействуют в СУБД в рамках двух- или трехуровневой клиент-серверной архитектуры. Следовательно, физический сервер, на который установлена СУБД, называется сервером БД. Администрирование СУБД включает в себя создание БД, управление и обслуживание инфраструктуры сервера [14].



.....
Выбор СУБД зависит от тех приложений, которыми она будет управляться.

Выбор СУБД – прерогатива разработчика, а не пользователя.

Ведущие поставщики СУБД: IBM, Oracle и Microsoft.



.....
При выборе СУБД необходимо руководствоваться такими показателями, как масштабируемость, быстродействие (как в выборе транзакций, так и в построении сложных аналитических выборок), работа с XML и кластерные решения.

В среднем скорости работы IBM, DB/2, MS SQL и Oracle примерно одинаковы. На общем фоне выделяются только Cache из-за новизны подхода и особой идеологии архитектуры [15].

Масштабируемость. Чем больше данных, тем сложнее ими управлять. Например, СУБД Oracle10g существует в нескольких вариантах, с разными схемами лицензирования. Для всех версий существует одно ядро, все версии совместимы.

Мультиплатформенность. Oracle и IBM DB/2 также расширяют возможности масштабирования: можно менять аппаратную платформу и ОС на более соответствующую растущим потребностям бизнеса без потерь данных, смены прикладного ПО и переподготовки администратора БД.

Кластерные технологии в приложении к СУБД, например по технологии Oracle RAC, повышают надежность системы, упрощают масштабируемость и снимают расходы на развитие инфраструктуры.

Различные СУБД отличаются характерными чертами. Например, IBM DB/2 имеет собственную высокопроизводительную кластерную структуру, которая позволяет переходить от больших RISC-серверов в качестве серверов БД к мейнфреймам. Oracle поддерживает XML DB. Oracle и IBM DB/2 поддерживают SQLJ, что особенно важно в телекоммуникации.

2. Требования к «рабочему месту» веб-разработчика. Анализ «рабочего места» разработчика подразумевает разговор о веб-серверах, СУБД и языках написания сценариев, т. е. «жизненно необходимого» пакета веб-программиста.

Веб-сервер – это сервер, принимающий HTTP-запросы от клиентов и вызывающий HTTP-ответы, как правило, вместе с HTML-страницей, изображением или мультимедийным объектом. Говоря о веб-сервере, имеют в виду как программное обеспечение (ПО), так и отдельный компьютер, на котором это ПО установлено.

Клиент (веб-браузер) подает веб-серверу запросы на получение ресурсов, обозначаемых URL-адресами.

Дополнительные функции веб-сервера:

- ведение журнала обращений пользователя к ресурсам;
- аутентификация и авторизация пользователей;
- поддержка динамически генерируемых страниц;
- поддержка HTTPS защищенных соединений с клиентами.

Существуют также легкие веб-серверы, например *lighttpd*, *litespeed*, *mongrel*.

Они имеют следующие параметры [16]:

- эффективность (быстрота реакции);
- масштабируемость (для многих пользователей);
- безопасность (не выходит ли за дозволенные рамки операций; шифрование; делает ли более уязвимыми соседей);
- работоспособность: проверяется в режимах отказа и аварийности;
- соответствие стандартам (протокол RFC в разновидностях);
- гибкость (можно ли настроить сервер для принятия большого числа запросов или динамических страниц, требующих значительных вычислений);
- требования к платформе: на каких платформах установлен;
- управляемость: легко ли установить и обслуживать сервер.

Легкие веб-серверы могут конкурировать с Apache и IIS, а также соединяться с ними для ускорения работы.

В чем «легкость» веб-серверов?

- Веб-сервер является простым, удобно устанавливаемым, нетребовательным и устойчивым (некоторые стали громоздкими, например Java Web Server, AOLServer и Zeus).

- Весь веб-сервер способен поместиться в одном файле, машины могут иметь встроенные, доступные через Веб управляющие консоли без сложной разработки и накладных расходов.
- Веб-серверы имеют открытый исходный код.
- Сервис-ориентированные архитектуры SOA довольно капризны, легкие серверы быстро устанавливаются SOA для демонстраций.
- Некоторые легкие веб-серверы ускоряют передачу видео и изображений Apache веб-сервером.

Веб-серверы обычно пишутся на C++, Erlang, Java, Lisp, Perl, Python Tcl.

Зачем использовать редкий язык?

- В целях образования: программист получает опыт работы с языком.
- На C++ некоторые файлы довольно громоздкие, в языках высокого уровня удается существенно уменьшить размер файлов.
- Языки высокого уровня облегчают экспериментальные возможности программиста: веб-серверы являются удобным экспериментальным материалом.
- Легкая модификация: добавление HTTP-сервера к существующему приложению может потребовать увеличение исходного кода только на несколько строк.

Все «легкие» веб-серверы являются узконаправленными объектами. В некоторых приложениях упор идет на быстродействие, в некоторых – на безопасность или экономию памяти. Примеры «легких» веб-серверов:

- ультралегкие (Cheetah Server, DustMote, fnord, ihttpd, mattous, Scrinchy, ZWS);
- высокопроизводительные (cghttpd, darkhttpd, Gatling, Kernux, lighthouse, Light Speed Web Server, Miniature JWS, Yaws);
- библиотеки (EHC, Embedded TCL Web Server);
- веб-серверы, написанные на Python (cdServer, edna);
- веб-серверы, написанные на Perl и других, менее известных языках (Camlserver, dhttpd, DNHTTPD, Jellybean, Ins.http, Mongrel, Nanoweb, Naridesh, OpenAngel, Xavante, XSP);
- веб-серверы, написанные на C++ (ABYSS, Anti_Web HTTPD, MHTTPD, mini_httpd, Nullhttpd, Seminode, thhttpd) [16].

3. Требования к аппаратным и программным ресурсам. Рассмотрим минимальные требования к аппаратным и программным ресурсам для работы веб-

сервера Apache, а значит, и всей работы веб-программиста [17]. Как известно, Apache изначально был разработан для работы на платформах Unix, теперь же существуют и конфигурации, работающие под Windows XP или Vista. Поэтому логично отдельно рассматривать системные требования для разных типов платформ.

Согласно исследованию, проведенному netcraft.com, большинство акций разработчиков принадлежали Apache (47,17%) и Microsoft (23,34%). Из всех активных сайтов 51,12% работали на Apache, 23,99% – на Microsoft. Распределение рынка акций топ-серверов таково: Apache принадлежит 66,82% всех акций, Microsoft принадлежит 18,25%. Вышеуказанные цифры довольно ясно демонстрируют, что Apache захватил более 50% рынка [17].

4. Требования к обеспечению информационной безопасности. Хакерским атакам подвержены все серверы. Это может быть как крупная корпорация, так и малая сеть, в которых хакерские программы сканируют весь IP-диапазон и ищут уязвимые места в сети. Даже если работать в локальной сети, а не в Интернете, существует вероятность заражения троянами через флеш-носители или ноутбуки.

Самые распространенные угрозы и проблемы, возникающие вследствие хакерских атак, а также меры, при помощи которых можно ликвидировать эти угрозы, приведены ниже.

- Угроза доступа к файлам .htpasswd и .htaccess. Данные файлы отвечают за авторизацию пользователей. Борьба с этой угрозой можно регулярно проходя авторизацию.
- Угроза доступа через уязвимости приложений. Функциональный уровень приложений может вызывать проблемы из-за ошибок кода и утечек (например, если в программе существуют скрытые «лазейки» быстрого доступа к данным, которые изначально программист писал «под себя» для ускорения процессов отладки приложения и просто забыл убрать). Борьба с данной угрозой можно посредством недопуска к коду программы. Кроме того, можно дописывать ядро, если работа идет под Unix.
- Угрозы межскриптовой уязвимости (например, посредством тегов HTML). С этой угрозой можно бороться посредством фильтрации определенных символов прокси-сервером.

- Проблемы переполнения буфера памяти в Apache. Неконтролируемые потоки данных способны привести к отказу сервера авторизовать реальных пользователей. Бороться с такой проблемой можно при помощи регулярного обновления сервера Apache.
- Сигналы к различным процессам. Хакер может посылать сигналы различным процессам и тем самым «убивать» или «порождать» лишние процессы. В нескольких версиях Apache с этой угрозой можно бороться посредством ограничения прав доступа пользователей к файловой системе и процессам.

Ко всем возможным угрозам применяются следующие контрмеры:

- постоянное обновление Apache; мониторинг активности;
- определение вторжения; понимание конфигурации;
- регулярная проверка посредством антивирусных программ;
- отключение ненужных модулей и ненужных файлов Apache;
- механизмы аутентификации; безопасный доступ администратора [17].

5. Критерии выбора платформы для разработки веб-приложения.

При написании веб-приложения необходимо учитывать особенности самого веб-приложения и те программные пакеты, которыми оно будет дополнено [16].

Необходимо исходить из того, что пишется веб-приложение в основном на HTML, оно имеет PHP-сценарии, обращающиеся к базам данных MySQL, использует JavaScript почтовый сервер, обращается к XML-файлам, ссылается на CSS-файлы. Кроме того, зачастую требуется участие Flash-сценариев, сценариев Java, CGI-сценариев; существует возможность зарегистрировать и авторизовать пользователей.

В настоящее время существуют версии большинства приложений, поддерживаемые на той или иной платформе (например, Flash был изначально разработан под Windows, но в настоящее время существуют версии, работающие под разновидностями Unix). То есть обычно программисты руководствуются набором приложений, поддерживаемым веб-сервером, предоставляющим хостинг. Если веб-приложение имеет небольшое количество Unix-ориентированных скриптов (например, на Perl или на Python), насыщено графикой, видео, Flash-скриптами и т. д., то есть смысл выбрать именно Windows-платформу, а не Unix. Кроме того, если приложение написано именно под Windows и является частью веб-приложения, то есть смысл либо полного, либо частичного перехода на

платформу Windows (здесь есть смысл экономической оценки: стоит ли покупать или переписывать приложение под Unix или дешевле сменить платформу).

Если в приложении присутствует большое число скриптов, написанных на Perl, Python, C++, кроме того, приложение выполняет параллельные процессы с другими приложениями, то есть смысл выбрать платформу Unix.



.....
Другим важным моментом в вопросе выбора платформы является ориентация приложения на браузеры.

Например, если идет ориентация на IE, то он не будет работать на платформе Unix. Если разрабатывается Java-апплет, к примеру J-Ads2, то при загрузке на сервер он должен работать. Для этого сервер должен поддерживать один из соответствующих браузеров (Netscape 3.x, Netscape 4.0x, 4.x, IE4.x, 5.x или выше, Opera с плагином Java, NeoPlanet...), поддерживающих AWT. Например, Netscape 4.61 работал только под Mac OS, которая не поддерживала AWT. При работе с Java-машиной предпочтительнее платформа Windows.

Необходимо учесть, что веб-приложение и сама система становятся более уязвимыми для вирусов, поскольку многие вирусы, например «черви», пишутся как апплеты или сервлеты, т. е. как Java-приложения.

4.3 Клиентские приложения, обеспечивающие интерфейс пользователя

В настоящее время существует большое количество инструментальных средств для разработки интерфейса, поддерживающих различные методы его реализации [18].

Основное назначение тех средств разработки пользовательских графических интерфейсов, которые разрабатываются и поставляются отдельно от оконной системы, – облегчение создания нового графического интерфейса за счет использования существующих параметризованных заготовок. Как видно, в принципе это те же самые идеи, на которых основана объектно-ориентированная библиотека оконной системы X Xt Intrinsics [18].

И действительно, наиболее распространенный пакет, предназначенный для быстрой и качественной разработки графических пользовательских интерфейсов, Motif, который был спроектирован и разработан в североамериканском консорциуме OSF, в основном является развитием идей Xt Intrinsics. Motif – это

сугубо коммерческий продукт. Дело дошло до того, что компания OSF запатентовала внешний интерфейс продуктов, входящих в состав Motif, чтобы не дать кому-нибудь возможность воспроизвести этот интерфейс.

В Калифорнийском университете г. Беркли был создан альтернативный механизм под названием Tcl/Tk. Этот механизм основан на наличии специализированного командного языка, предназначенного для описания графических пользовательских интерфейсов, соответствующего интерпретатора и библиотеки ранее разработанных заготовок интерфейсов. Пакет Tcl/Tk распространяется (вместе с полной документацией) свободно, и многие профессиональные программисты находят его более удобным, чем Motif.

1. Пакет Motif.

Motif (официальное название этого продукта – OSF/Motif) представляет собой программный пакет, включающий оконный менеджер, набор вспомогательных утилит, а также библиотеку классов, построенных на основе Xt Intrinsics. Для конечных пользователей оконных систем, опирающихся на Motif, основной интерес представляет менеджер окон.

Для разработчиков же графических интерфейсов важны все три компонента Motif. Новый интерфейс разрабатывается в графическом же режиме с использованием оконного менеджера. При этом полезно использование утилит Motif и необходимо использование библиотеки классов Motif.

Библиотека классов Motif является расширением библиотеки Xt Intrinsics с целью придания этой библиотеке практического смысла (по-другому можно сказать, что Motif – это то, чем должен был бы быть Xt, если бы при его создании ставились коммерческие цели). Все графические объекты (правильнее сказать, классы) Xt Intrinsics включаются в библиотеку классов Motif, хотя в ней используются другие имена.

Но Motif существенно расширяет возможности Xt Intrinsics. В его библиотеке поддерживается большое число классов, позволяющих создавать меню, «нажимаемые» кнопки и т. д. Основное назначение этих классов – определение новых виджетов, связанных с окнами.

Однако в Motif поддерживается и новый вид графических объектов (их классов) – так называемые гаджеты (gadgets). Гаджет отличается от виджета тем, что соответствующий класс также может использоваться для создания элементов интерфейса, но графический объект не привязывается к определенному окну. При отображении на экран гаджета используется окно объекта, относящегося к суперклассу класса гаджета.

Основная идея Motif: развитая библиотека классов языка Си++, возможности применения этих классов при использовании обычного стиля программирования и поддержка визуального программирования с немедленным отображением получающихся графических объектов.

2. Язык и интерпретатор Tcl/Tk.

Продукт Tcl/Tk в действительности представляет собой два связанных программных пакета, которые совместно обеспечивают возможность разработки и использования приложений с развитым графическим пользовательским интерфейсом [20]. Название Tcl относится к командному языку инструментальных средств – *tool command language*. Это простой командный язык для управления приложениями и расширения их возможностей. Язык Tcl является «встраиваемым»: его интерпретатор реализован в виде библиотеки функций языка Си++, так что интерпретатор может быть легко пристыкован к любой прикладной программе, написанной на языке Си++.

Tk (рекомендуемое произношение – «ти-кей») является библиотекой Си-функций, ориентированной на облегчение создания пользовательских графических интерфейсов в среде оконной системы X (т. е., по сути дела, некоторый аналог Xt Intrinsics). С другой стороны, аналогично тому, как это делается в командных языках семейства shell, функции библиотеки Tk являются командами языка Tcl, так что любой программист может расширить командный репертуар языка Tcl путем написания новой функции на языке Си++.

Совместно Tcl и Tk обеспечивают четыре преимущества для разработчиков приложений и пользователей (мы используем здесь авторские тексты разработчиков). Во-первых, наличие командного языка Tcl дает возможность в каждом приложении использовать мощный командный язык. Все, что требуется от разработчика приложения, чтобы удовлетворить его специфические потребности, – это создать несколько новых команд Tcl, требующихся приложению (и, возможно, другим приложениям – явно традиционный стиль командного программирования в ОС UNIX). После этого нужно связать прикладную программу с интерпретатором Tcl и пользоваться полными возможностями командного языка [19].

Вторым преимуществом использования Tcl/Tk является возможность быстрой разработки графических интерфейсов. Многие интересные оконные приложения могут быть написаны в виде скриптов языка Tcl без привлечения языков Си или Си++ (а Tcl позволяет скрыть многие несущественные детали). Как утверждают разработчики Tcl/Tk, пользователи оказываются способными к

созданию новых графических интерфейсов уже после нескольких часов знакомства с продуктом. Другой особенностью языка Tcl, способствующей быстрой разработке оконных приложений, является то, что язык является интерпретируемым. Можно опробовать новую идею интерфейса, выражающуюся в сотнях или тысячах строк кода на языке Tcl, без потребности вызова новых программных средств, путем простого нажатия на клавишу мыши (не наблюдая существенных задержек при использовании современных рабочих станций).

Третьим преимуществом языка Tcl является то, что его можно применять в качестве языка «склейки» приложений. Например, любое основанное на Tcl и использующее Tk оконное приложение может направить свой скрипт любому другому аналогично ориентированному приложению. С использованием Tcl/Tk можно создавать приложения, работающие в стиле мультимедиа, и опять же они смогут обмениваться скриптами, поскольку пользуются общим интерпретатором командного языка Tcl и общей внешней библиотекой Tk [19].

Наконец, четвертым удобством интегрированного пакета Tcl/Tk является удобство пользователей. Для написания нового приложения в среде Tcl/Tk достаточно выучить несколько совершенно необходимых команд и этого окажется достаточно.

3. Microsoft Expression Blend – инструмент создания интерфейсов.

Появление языка описания пользовательских интерфейсов XAML (произносится «зámмель») и новой среды разработки Expression Blend позволяет заметно ускорить и облегчить проектирование и построение пользовательских интерфейсов как для веб-, так и для настольных приложений [18].

Данный язык позволяет описывать внешний вид и поведение интерфейсных элементов, устанавливая взаимодействие этих элементов с различными данными и событиями. Допускает прямое подключение к Common Language Runtime (CLR), что обеспечивает большую гибкость при проектировании ПО.

Blend обладает разветвленными возможностями для построения качественных интерфейсов и его главной возможностью является создание пользовательских библиотек-стилей, содержащих интерфейсные элементы с заранее заданным внешним видом и поведением. Blend является мощным приложением для создания пользовательских интерфейсов.

Blend – программа для создания уже готовых интерфейсов, т. е. дизайнер выдает программистам готовый интерфейс, не требующий их вмешательства в графическое решение. Программист только подключает интерфейс к процедур-

ному коду. Данная концепция является достаточно новаторской как для дизайнеров интерфейсов, так и для программистов. С точки зрения дизайнера интерфейсов Blend является дополнительным инструментом при проектировании интерфейсов, так как проектирование интерфейсов – это не только и даже не совсем внешний вид, а в первую очередь взаимодействие программы и человека, а инструменты, позволяющие это делать в Blend, малоразвиты или отсутствуют совсем.

Преимущества новой технологии

Основное преимущество – гибкость при создании приложений, которая обеспечивается наличием современных средств визуализации и новых технологий:

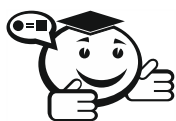
- Векторная графика: теперь интерфейс состоит полностью из векторных объектов (интерфейсные элементы, графика, пиктограммы).
- Новые экранные шрифты и новая технология попиксельного позиционирования изображения на экране.
- Одна программа может содержать несколько интерфейсов (разные разрешения, веб- и настольные приложения и т. д.) [18].

Дизайнер и программист могут одновременно работать над одним проектом, каждый выполняя свою функцию, что обеспечивает гибкость при создании приложений и увеличивает скорость работы.

5 Проектирование информационной системы

5.1 Этапы создания информационных систем

Создание электронных информационных систем (ЭИС) осуществляется на основе требований со стороны предполагаемых пользователей, которые, как правило, изменяются в процессе разработки. С точки зрения теории принятия решений процесс проектирования ЭИС – это процесс принятия проектно-конструкторских решений, направленных на получение описания системы (проекта ЭИС), удовлетворяющего требованиям заказчика [20–22].



.....

*Под **проектом ЭИС** понимают проектно-конструкторскую и технологическую документацию, в которой представлено описание проектных решений по созданию и эксплуатации ЭИС в конкретной программно-технической среде.*

*Под **проектированием ЭИС** понимается процесс преобразования входной информации об объекте проектирования, о методах проектирования и об опыте проектирования объектов аналогичного назначения в соответствии с ГОСТом в проект ЭИС.*

.....

С этой точки зрения проектирование ЭИС сводится к последовательной формализации проектных решений на различных стадиях жизненного цикла ЭИС: планирования и анализа требований, технического и рабочего проектирования, внедрения и эксплуатации ЭИС.

Объектами проектирования ЭИС являются отдельные элементы или их комплексы функциональных и обеспечивающих частей. Так, функциональными элементами в соответствии с традиционной декомпозицией выступают задачи, комплексы задач и функции управления. В составе обеспечивающей части ЭИС объектами проектирования служат элементы и целые комплексы информационного, программного и технического обеспечения системы.

Осуществление проектирования ЭИС предполагает использование проектировщиками определенной технологии проектирования, соответствующей масштабу и особенностям разрабатываемого проекта [20].



.....

Технология проектирования ЭИС – это совокупность методов и средств проектирования ЭИС, а также методов и средств организации проектирования (управления процессом создания и модернизации проекта ЭИС) (рис. 5.1).

.....

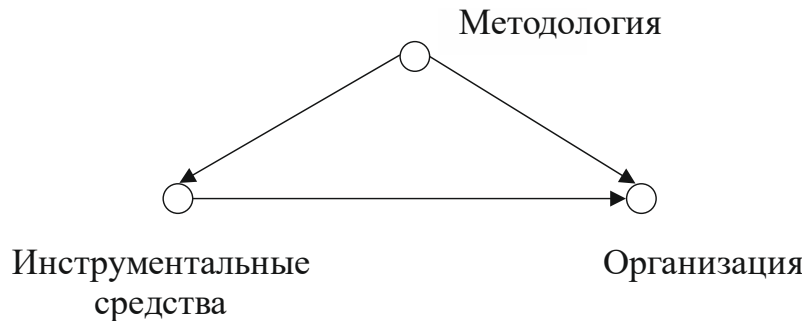


Рис. 5.1 – Состав компонентов технологии проектирования

В основе технологии проектирования лежит технологический процесс, который определяет действия, их последовательность, состав исполнителей, средства и ресурсы, требуемые для выполнения этих действий.

Так, технологический процесс проектирование ЭИС в целом делится на совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий, каждое из которых может иметь свой предмет. Действия, которые выполняются при проектировании ЭИС, могут быть определены как неделимые технологические операции или как подпроцессы технологических операций [21]. Все действия могут быть собственно проектировочными, которые формируют или модифицируют результаты проектирования, и оценочными, которые вырабатывают по установленным критериям оценки результатов проектирования.

Таким образом, технология проектирования задается регламентированной последовательностью технологических операций, выполняемых в процессе создания проекта на основе того или иного метода, в результате чего стало бы ясно, не только *что* должно быть сделано для создания проекта, но и *как*, *кому* и в *какой последовательности* это должно быть сделано.

Принято выделять два уровня представления модели данных – логический и физический [22].

Цель моделирования данных на *логическом уровне* состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели

или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.



.....

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например «Отдел», «Фамилия сотрудника».

.....

Объекты, модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных может быть построена на основе другой логической модели, например модели процессов. Такая модель данных является универсальной и никак не связана с конкретной реализацией СУБД (системы управления базой данных). Построение логической модели ИС до ее программной разработки или до начала проведения архитектурной реконструкции столь же необходимо, как наличие проектных чертежей перед строительством большого здания. Хорошие модели ИС позволяют наладить плодотворное взаимодействие между заказчиками, пользователями и командой разработчиков.



.....

На физическом уровне данные, напротив, зависят от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация обо всех объектах БД.

.....

Поскольку стандартов на объекты БД не существует, физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей.

Логические модели. На логическом уровне проектирования строится так называемая визуальная модель объекта [21].

Визуальные модели обеспечивают ясность представления выбранных архитектурных решений и позволяют понять разрабатываемую систему во всей ее полноте.

Визуальное моделирование не способно раз и навсегда решить все проблемы, однако его использование существенно облегчает достижения таких целей, как:

- повышение качества программного продукта;

- сокращение стоимости проекта;
- поставка системы в запланированные сроки [20].

Существует множество подходов к построению таких моделей: производные, фреймы, графовые модели, семантические сети, модель «сущность – связь» (ERD), UML и т. д.

Физические модели. Логическая модель данных должна быть отображена в компьютерно-ориентированную даталогическую модель, «понятную» СУБД [21]. В процессе развития теории и практического использования баз данных, а также средств вычислительной техники создавались СУБД, поддерживающие различные даталогические модели.

5.2 Последовательность создания информационной модели данных

Процесс создания информационной модели данных начинается с определения концептуальных требований ряда пользователей. Концептуальные требования могут определяться и для некоторых задач (приложений), которые в ближайшее время реализовывать не планируется. Это может несколько повысить трудоемкость работы, однако поможет наиболее полно учесть все нюансы функциональности, требуемой для разрабатываемой системы, и снизит вероятность ее переделки в дальнейшем. Требования отдельных пользователей интегрируются в едином «обобщенном представлении». Последнее называют концептуальной моделью.



.....
Концептуальная модель представляет объекты и их взаимосвязи без указания способов их физического хранения.



.....
 Концептуальная модель является, по существу, моделью предметной области.

При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели основано на анализе решаемых на этом предприятии задач по обработке данных. Кон-

цептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных. Здесь имеются в виду данные, используемые как в уже разработанных прикладных программах, так и в тех, которые только будут реализованы [26].

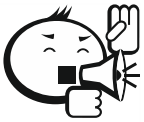


.....

Концептуальная модель транслируется затем в модель данных, совместимую с выбранной СУБД.

.....

Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии не реализуемыми средствами выбранной СУБД. Это потребует изменения концептуальной модели. Версия концептуальной модели, которая может быть обеспечена конкретной СУБД, называется логической моделью.



.....

Логическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения.

Логическая модель данных может быть реляционной, иерархической или сетевой.

.....

Пользователям выделяются подмножества этой логической модели, называемые внешними моделями, отражающие их представления о предметной области.



.....

Внешняя модель соответствует представлениям, которые пользователи получают на основе логической модели, в то время как концептуальные требования отражают представления, которые пользователи первоначально желали иметь и которые легли в основу разработки концептуальной модели.

.....



.....

Логическая модель отображается в физическую память.

.....



.....

Физическая модель, определяющая размещение данных, методы доступа и технику индексирования, называется *внутренней моделью системы*.

.....

Внешние модели никак не связаны с типом физической памяти, в которой будут храниться данные, и с методами доступа к этим данным. С другой стороны, если концептуальная модель способна учитывать расширение требований к системе в будущем, то вносимые в нее изменения не должны оказывать влияния на существующие внешние модели. Основное различие между указанными выше тремя типами моделей данных (концептуальной, логической и физической) состоит в способах представления взаимосвязей между объектами. При проектировании баз данных необходимо различать взаимосвязи между объектами, между атрибутами одного объекта и между атрибутами различных объектов.

Моделирование данных проводится как поуровневый спуск от концептуальной модели к логической, а затем к физической модели.

5.3 Методология IDEF0

Создание современных информационных систем представляет собой сложнейшую задачу, решение которой требует применения специальных методик и инструментов. В последнее время среди системных аналитиков и разработчиков значительно вырос интерес к CASE (*Computer-Aided Software/System Engineering*) – технологиям и инструментальным CASE-средствам, позволяющим максимально систематизировать и автоматизировать все этапы разработки программного обеспечения [22].

Технология создания информационных систем предъявляет особые требования к методикам реализации и программным инструментальным средствам.

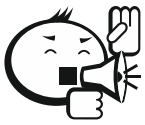
Реализацию проектов по созданию ИС принято разбивать на стадии анализа (прежде чем создавать ИС, необходимо понять и описать бизнес-логику предметной области), проектирования (необходимо определить модули и архитектуру будущей системы), непосредственного кодирования, тестирования и сопровождения.

Для создания ИС жизненно необходим инструмент, значительно (в несколько раз) уменьшающий время разработки ИС.

На современном рынке средств разработки ИС достаточно много CASE-средств, например ERwin и PRwin, которые были разработаны фирмой Logic

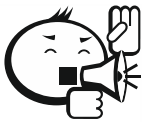
Works. После слияния в 1998 г. Logic Works с PLATINUM technology они выпускаются под логотипом PLATINUM technology.

На начальных этапах создания ИС необходимо понять, как работает организация, которую собираются автоматизировать. Никто в организации не знает, как она работает в той мере подробности, которая необходима для создания ИС. Руководитель хорошо знает работу в целом, но не в состоянии вникнуть в детали работы каждого рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но плохо знает, как работают коллеги. Поэтому для описания работы предприятия необходимо построить модель [22].



Такая модель должна быть адекватна предметной области, следовательно, она должна содержать в себе знания всех участников бизнес-процессов организации.

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, предложенный более 20 лет назад Дугласом Россом (SoftTech, Inc.) и называвшийся первоначально *SADT – Structured Analysis and Design Technique*. В дальнейшем это подмножество SADT было принято в качестве федерального стандарта США под наименованием IDEF0. Подробные спецификации на стандарты IDEF можно найти на сайте <http://www.idef.ru>.



В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Такая чисто функциональная ориентация является принципиальной – функции системы анализируются независимо от объектов, которыми они оперируют. Это позволяет более четко смоделировать логику и взаимодействие процессов организации.

Методология IDEF0 основана на следующих концептуальных положениях [23]:

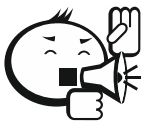
1. Модель – искусственный объект, представляющий собой отображение (образ) системы и ее компонентов. Модель описывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций и что производит.
2. Блочное моделирование и его графическое представление. Основной концептуальный принцип методологии IDEF0 – представление любой

изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия, происходящие в изучаемой системе.

3. Лаконичность и точность. Графический язык позволяет лаконично, однозначно и точно показать все элементы (блоки) системы и все отношения и связи между ними, выявить ошибочные, лишние или дублирующие связи и т. д.
4. Передача информации. Средства IDEF0 облегчают передачу информации от одного участника разработки модели (отдельного разработчика или рабочей группы) к другому.
5. Строгость и формализм. Разработка моделей IDEF0 требует соблюдения ряда строгих формальных правил, обеспечивающих преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей.
6. Итеративное моделирование. Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, после чего цикл повторяется.
7. Отделение «организации» от «функций». При разработке моделей следует избегать изначальной «привязки» функций исследуемой системы к существующей организационной структуре моделируемого объекта (предприятия, фирмы). Это помогает избежать субъективной точки зрения, навязанной организацией и ее руководством.

Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Моделируемая система имеет границу. Взаимодействие системы с окружающим миром описывается как вход (нечто, что перерабатывается системой), выход (результат деятельности системы), управление (стратегии и процедуры, под управлением которых производится работа) и механизм (ресурсы, необходимые для проведения работы). Находясь под управлением, система преобразует входы в выходы, используя механизмы.

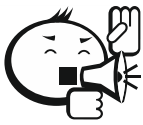


.....
 Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т. е. наиболее абстрактного уровня

описания системы в целом. В контекст входят определение субъекта моделирования, цели и точки зрения на модель.

.....

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что – как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования – вопросы, на которые построенная модель должна дать ответ.



.....
Первоначально необходимо определить область (Scope) моделирования.

Описание области – как системы в целом, так и ее компонентов – является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента – широту и глубину [20].

Широта подразумевает определение границ модели, что будет рассматриваться внутри системы, а что снаружи.

Глубина определяет, на каком уровне детализации модель является завершенной. При определении глубины системы необходимо не забывать об ограничениях времени – трудоемкость построения модели растет в геометрической прогрессии от глубины декомпозиции. После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему; поскольку все объекты модели взаимосвязаны, внесение нового объекта может быть не просто арифметической добавкой, но в состоянии изменить существующие взаимосвязи.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения.

Диаграммы IDEF0. Основу методологии IDEF0 составляет графический язык описания бизнес-процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать четыре типа диаграмм:

- 1) контекстную диаграмму (в каждой модели может быть только одна контекстная диаграмма);
- 2) диаграммы декомпозиции;
- 3) диаграммы дерева узлов;
- 4) диаграммы только для экспозиции (FEO) [22].

1. *Контекстная диаграмма* является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. После описания системы в целом проводится разбиение ее на крупные фрагменты.
2. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются *диаграммами декомпозиции*. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступить к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.
3. *Диаграмма дерева узлов* показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколько угодно много, поскольку дерево может быть построено на произвольную глубину и необязательно с корня.
4. *Диаграммы для экспозиции (FEO)* строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения либо для специальных целей.

Работы обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Работы изображаются в виде прямоугольников. Все работы должны быть названы и определены. Имя работы должно быть выражено отглагольным существительным, обозначающим действие (например, «Изготовление детали»),

«Прием заказа» и т. д.). Работа «Изготовление детали» может иметь, например, следующее определение: «Работа относится к полному циклу изготовления изделия от контроля качества сырья до отгрузки готового упакованного изделия». При создании новой модели (меню File/New) автоматически создается контекстная диаграмма с единственной работой, изображающей систему в целом (рис. 5.2).



Рис. 5.2 – Пример контекстной диаграммы

Диаграммы декомпозиции содержат родственные работы, т. е. дочерние работы, имеющие общую родительскую работу. Декомпозировать работу на одну работу не имеет смысла: диаграммы с количеством работ более восьми получаются перенасыщенными и плохо читаются. Для обеспечения наглядности и лучшего понимания моделируемых процессов рекомендуется использовать от трех до шести блоков на одной диаграмме.

Если оказывается, что количество работ недостаточно, то работу можно добавить в диаграмму, щелкнув сначала по кнопке на палитре инструментов, а затем по свободному месту на диаграмме.

Работы на диаграммах декомпозиции обычно располагаются по диагонали от левого верхнего угла к правому нижнему.

Такой порядок называется порядком доминирования. Согласно этому принципу расположения в левом верхнем углу располагается самая важная ра-

бота или работа, выполняемая по времени первой. Далее вправо вниз располагаются менее важные или выполняемые позже работы. Такое расположение облегчает чтение диаграмм, кроме того, на нем основывается понятие взаимосвязей работ (рис. 5.3).

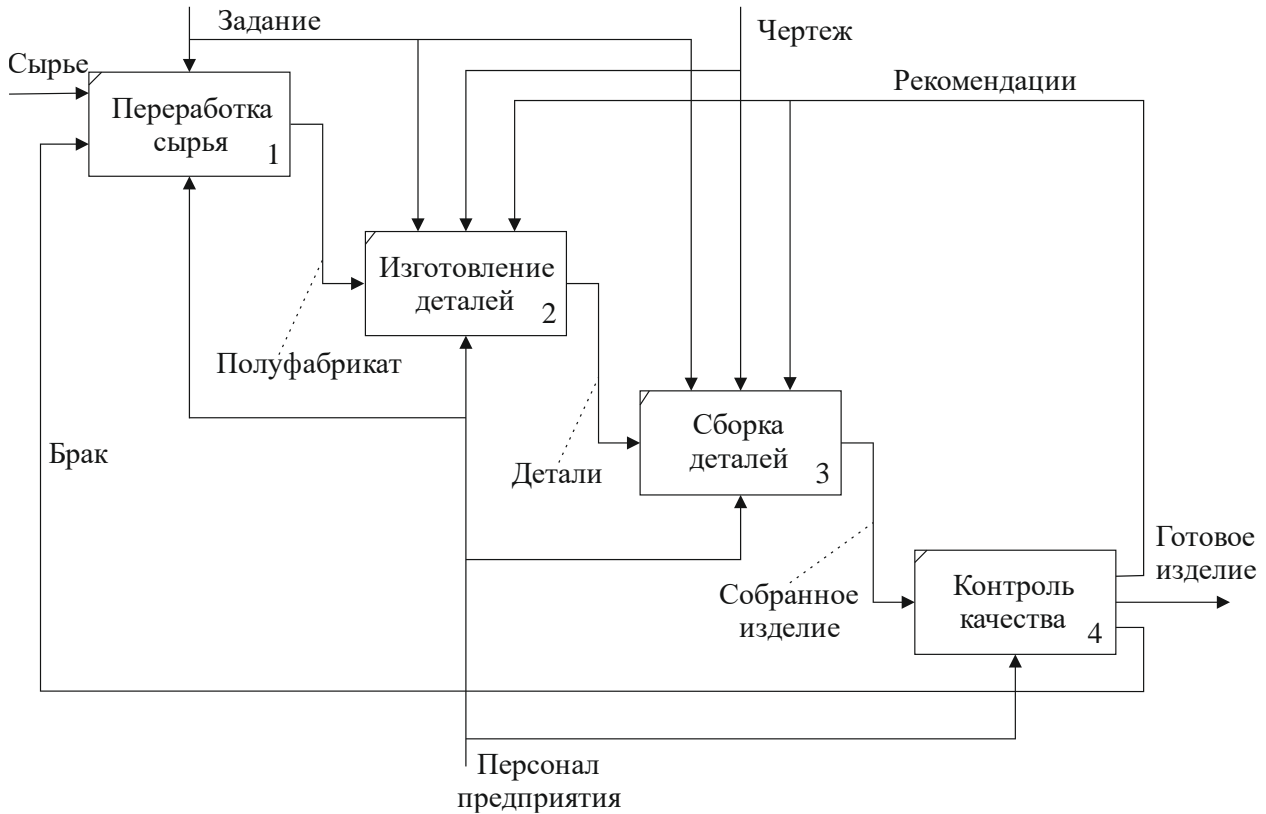


Рис. 5.3 – Пример диаграммы декомпозиции «Изготовление изделия»

Каждая из работ на диаграмме декомпозиции может быть в свою очередь декомпозирована. На диаграмме декомпозиции работы нумеруются автоматически слева направо. Номер работы показывается в правом нижнем углу. В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок [20, 22]. В IDEF0 различают пять типов стрелок (рис. 5.4).

Вход (*Input*) – материал или информация, которые используются или преобразуются работой для получения результата (выхода). Допускается, что работа может не иметь ни одной стрелки входа. Каждый тип стрелок подходит к определенной стороне прямоугольника, изображающего работу, или выходит из нее. Стрелка входа рисуется как входящая в левую грань работы. При описании технологических процессов (для этого и был придуман IDEF0) не возникает проблем определения входов. Действительно, «Сырье» на рисунке 5.3 – это нечто,

что перерабатывается в процессе «Переработка сырья» для получения результата. При моделировании ИС, когда стрелками являются не физические объекты, а данные, не все так очевидно. Например, при «Приеме пациента» карта пациента может быть и на входе, и на выходе, между тем качество этих данных меняется. Другими словами, в нашем примере стрелки входа и выхода должны быть точно определены, чтобы указать на то, что данные действительно были переработаны (например, на выходе – «Готовое изделие»). Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить то, перерабатываются/изменяются ли данные в работе или нет. Если изменяются, то скорее всего это вход, если нет – управление.

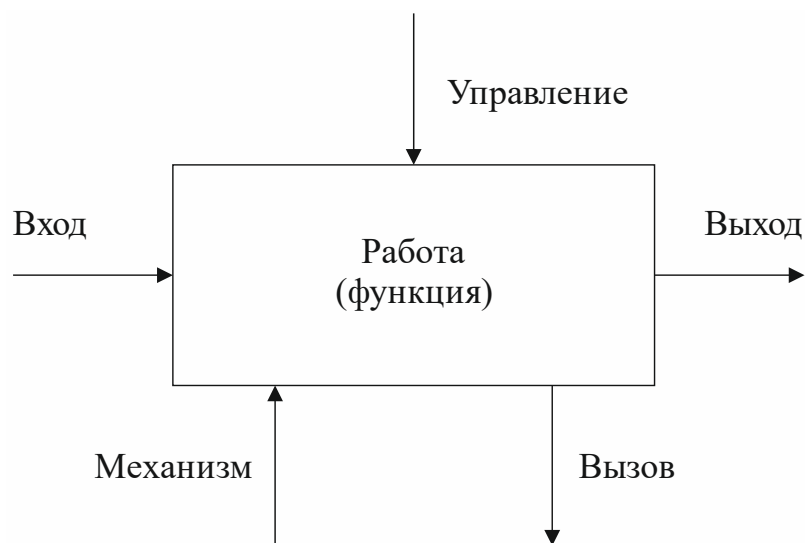


Рис. 5.4 – Основные элементы графической нотации IDEF0

Управление (*Control*) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелка управления рисуется как входящая в верхнюю грань работы. На рисунке 5.3 стрелки «Задание» и «Чертеж» – управление для работы «Изготовление деталей». Управление влияет на работу, но не преобразуется работой. Если цель работы – изменить процедуру или стратегию, то такая процедура или стратегия будет для работы входом. В случае возникновения неопределенности в статусе стрелки (управление или вход) рекомендуется рисовать стрелку управления.

Выход (*Output*) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Стрелка выхода рисуется

как исходящая из правой грани работы. На рисунке 5.3 стрелка «Готовое изделие» является выходом для работы «Изготовление изделия».

Механизм (*Mechanism*) – ресурсы, которые выполняют работу, например, персонал предприятия, станки, устройства и т. д. Стрелка механизма рисуется как входящая в нижнюю грань работы. На рисунке 5.3 стрелка «Персонал предприятия» является механизмом для работы «Изготовление изделия». По усмотрению аналитика стрелки механизма могут не изображаться в модели.

Вызов (*Call*) – специальная стрелка, указывающая на другую модель работы. Стрелка вызова рисуется как исходящая из нижней грани работы. Стрелка вызова используется для указания того, что некоторая работа выполняется за пределами моделируемой системы. В VPwin стрелки вызова используются в механизме слияния и разделения моделей.

Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. Контекстная диаграмма всегда имеет номер А-0, декомпозиция контекстной диаграммы – номер А0, остальные диаграммы декомпозиции – номера по соответствующему узлу (например, А1, А2, А3 и т. д., см. рис. 5.5).

Граничные стрелки. Стрелки на контекстной диаграмме служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у работы, или наоборот. Такие стрелки называются граничными. Имена вновь внесенных стрелок автоматически заносятся в словарь (*Arrow Dictionary*).

VPwin автоматически поддерживает нумерацию по узлам, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер. В результате проведения экспертизы диаграммы могут уточняться и изменяться, следовательно, могут быть созданы различные версии одной и той же (с точки зрения ее расположения в дереве узлов) диаграммы декомпозиции.

Правила IDEF0 включают [4]:

- ограничение количества блоков на каждом уровне декомпозиции (правило 3–6 блоков) (рис. 5.3);
- связность диаграмм (номера блоков);
- уникальность меток и наименований (отсутствие повторяющихся имен);
- синтаксические правила для графики (блоков и дуг);
- разделение входов и управлений (правило определения роли данных).

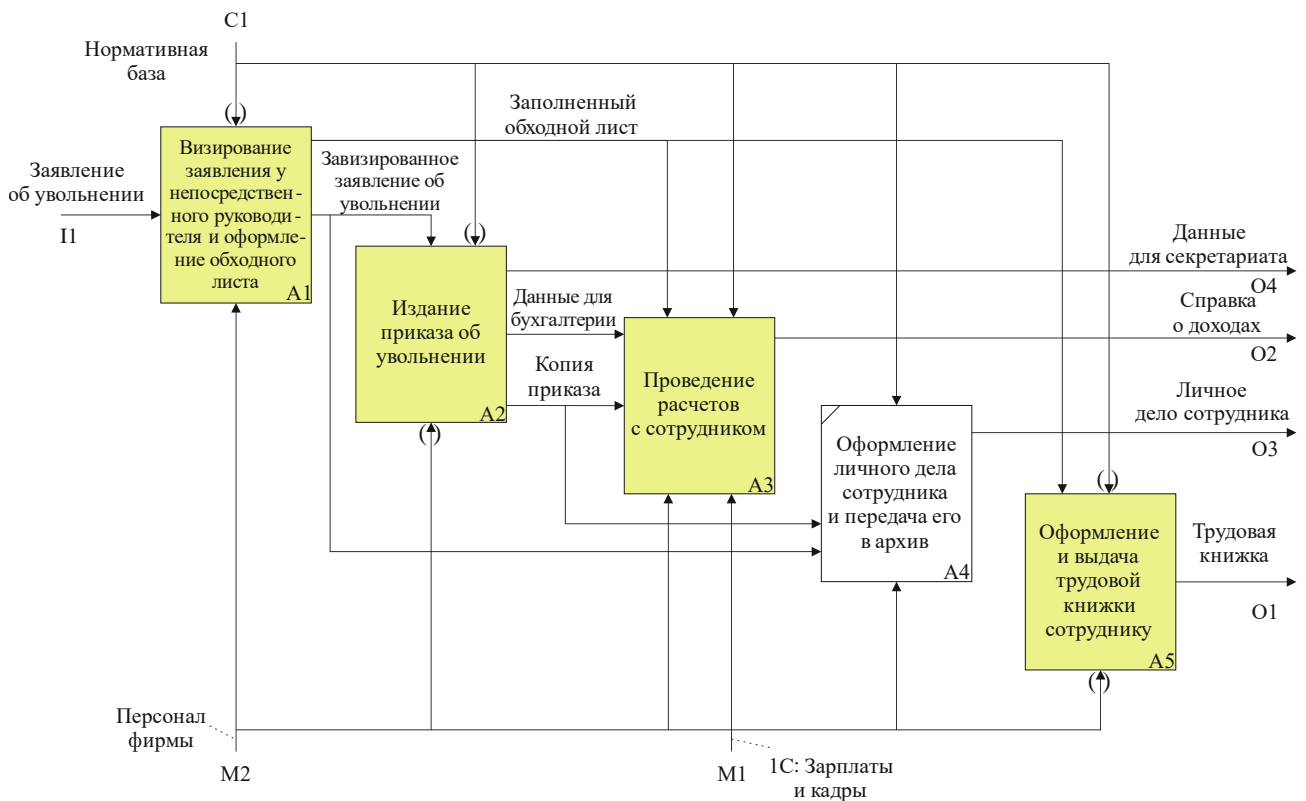


Рис. 5.5 – Пример диаграммы, разработанной с помощью стандарта IDEF0

5.4 Физическая и логическая модели данных

ERwin имеет два уровня представления модели – логический и физический [24].

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например «Постоянный клиент», «Отдел» или «Фамилия сотрудника». Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами (подробнее о сущностях и атрибутах будет рассказано ниже). Логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если

в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т. д. Разделение модели данных на логические и физические позволяет решить несколько важных задач:

1. Документирование модели. Многие СУБД имеют ограничение на именование объектов (например, ограничение на длину имени таблицы или запрет использования специальных символов – пробела и т. п.). Зачастую разработчики ИС имеют дело с нелокализованными версиями СУБД. Это означает, что объекты БД могут называться короткими словами, только латинскими символами и без использования специальных символов (т. е. нельзя назвать таблицу предложением, только одним словом). Кроме того, проектировщики БД нередко злоупотребляют «техническими» наименованиями, в результате таблица и колонки получают наименования типа *RTD_324* или *CUST_A12* и т. д. Полученную в результате структуру могут понять только специалисты (а чаще всего только авторы модели), ее невозможно обсуждать с экспертами предметной области. Разделение модели на логическую и физическую позволяет решить эту проблему. На физическом уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам подобрать синонимы – имена, более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Например, таблице *CUST_A12* может соответствовать сущность *Постоянный клиент*. Такое соответствие позволяет лучше задокументировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

2. Масштабирование. Создание модели данных, как правило, начинается с создания логической модели. После описания логической модели проектировщик может выбрать необходимую СУБД и ERwin автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт. Этот процесс называется прямым проектированием (*Forward Engineering*). Тем самым достигается масштабируемость – создав одну логическую модель данных, можно сгенерировать физические модели под любую поддерживаемую ERwin СУБД. С другой стороны, ERwin способен по содержимому системного каталога или SQL-скрипту воссоздать физическую и логическую модель данных (*Reverse Engineering*). На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД и затем сгенерировать ее

системный каталог. Следовательно, ERwin позволяет решить задачу по переносу структуры данных с одного сервера на другой. Например, можно перенести структуру данных с Oracle на Informix (или наоборот) или перенести структуру dbf-файлов в реляционную СУБД, тем самым облегчив решение по переходу от файл-серверной к клиент-серверной ИС. Заметим, однако, что формальный перенос структуры «плоских» таблиц на реляционную СУБД обычно неэффективен. Для того чтобы извлечь выгоды от перехода на клиент-серверную технологию, структуру данных следует модифицировать. Процессы прямого и обратного проектирования будут рассмотрены ниже.

Для создания моделей данных в ERwin можно использовать две нотации: IDEF1X и IE (*Information Engineering*). Методология IDEF1X была разработана для армии США и широко используется в государственных учреждениях США, финансовых и промышленных корпорациях. Методология IE, разработанная Дж. Мартином (*J. Martin*), К. Финкельштейном (*C. Finkelstein*) и другими авторами, используется преимущественно в промышленности.

ERwin имеет несколько уровней отображения диаграммы: уровень сущностей, уровень атрибутов, уровень определений, уровень первичных ключей и уровень иконок. Переключиться между первыми тремя уровнями можно с использованием кнопок панели инструментов.

5.5 Подходы к концептуальному моделированию

Задача проектировщика БД на этапе концептуального моделирования состоит в том, чтобы на основании локальных представлений пользователей найти обобщенное представление информации, свойственное природе предметной области (ПО) как целого [25].

Для этого необходимо выполнить анализ ПО, то есть:

- уяснить цели предприятия, для которого создается БД;
- выявить функции или действия, направленные на достижение целей, и правила бизнеса, принятые на предприятии;
- выявить данные, необходимые для выполнения функций;
- выделить объекты, вовлеченные в деятельность, и выявить связи между ними [22].

Результатом анализа является концептуальная (информационная) модель ПО. С точки зрения проектировщика, она содержит спецификации логического макета БД – определения базовых отношений и правил целостности данных.

Последовательность шагов при концептуальном проектировании [3]:

1. Выделение сущностей (определение основных объектов, которые могут интересовать пользователя и, следовательно, должны храниться в БД).
2. Определение атрибутов.
3. Определение связей.

Существуют два основных подхода к проектированию концептуальной модели – формальный и семантический.

Формальный подход включает два этапа моделирования:

- 1) анализ ПО с целью выявления полного перечня хранимых атрибутов и правил бизнеса;
- 2) нормализация универсального отношения до требуемого уровня [22].

Результатами анализа являются определения схемы универсального отношения и множества межатрибутных связей, существующих в нем. Результат нормализации – система отношений, связанных по типу «родитель – потомок» или «супертип – категория». Выполняя нормализацию, обычно стремятся строить такие проекции универсального отношения, которые можно интерпретировать как объекты ПО или факты связи объектов.

Основной недостаток формального подхода состоит в том, что он требует проведения детального анализа ПО до начала проектирования логического макета БД. Поэтому методики, основанные на формальном подходе, мало пригодны для решения сложных задач.

Семантический подход, в отличие от формального, предполагает параллельное выполнение анализа ПО и проектирование логического макета БД. В основе подхода лежат понятия ER-модели данных. Процесс проектирования включает три этапа [26].

На первом этапе проектировщик формирует общие представления о компонентах бизнеса и их отношениях и идентифицирует основные сущности и связи. При этом он не стремится получить детальную информацию о свойствах объектов ПО и их взаимосвязях.

На втором этапе представления проектировщика детализируются до уровня идентификаторов экземпляров сущностей и типов связей. Экземпляр сущности – это конкретный представитель данной сущности. Например экземпляром сущности «Сотрудник» может быть сотрудник Иванов.

Здесь окончательно определяется состав сущностей модели и специфицируются ограничения целостности сущности и ссылочной целостности – первичные и альтернативные ключи, внешние ключи, типы сущностей и связей. Связь (*Relationship*) – поименованная ассоциация между сущностями, значимая для рассматриваемой предметной области. Степенью связи называется количество сущностей, участвующих в связи, мощностью связи – число экземпляров сущности, участвующих в связи.

Результат этапа – логический макет БД с точностью до ключей.

На третьем этапе формируются окончательные представления о составе атрибутов сущностей и полностью определяются схемы всех отношений, соответствующих сущностям. Как правило, все отношения схемы находятся в третьей нормальной форме (ЗНФ).



.....

Атрибут (Attribute) – характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для идентификации, классификации, количественной характеристики или выражения состояния сущности.

.....

Выделяя сущности и определяя связи между ними, проектировщик опирается на свои текущие представления о ПО и здравый смысл. На каждом этапе он может согласовать свои представления с представлениями конечных пользователей. Поэтому грубые ошибки моделирования при разумном использовании семантического подхода – редкость. Этот подход к проектированию представляется вполне естественным и понятным. Кроме того, он очень эффективен, если проектировщик придерживается определенной дисциплины проектирования и использует подходящие средства для документирования работ.

Методологии семантического подхода. В настоящее время существуют несколько методологий анализа и проектирования логического макета БД, реализующих семантический подход [21].

Основным компонентом любой из них является графический язык определения данных – система графических нотаций для основных понятий *ER-модели*. Этими средствами проектировщик может точно, ясно и наглядно сформулировать свои текущие представления о ПО в виде диаграмм.

Второй компонент – глоссарий, содержащий однозначные определения имен сущностей и атрибутов. Он раскрывает те аспекты модели, которые невоз-

можно отразить графическими средствами. Методологии различаются, в основном, нотациями графических языков определения данных и обладают рядом общих достоинств.

Во-первых, все они поддерживают *параллельное развитие анализа ПО и проектирования структуры БД и правил целостности данных*. Нет никакой необходимости выявлять весь перечень хранимых данных и правил бизнеса еще до начала проектирования. Для выделения базовых отношений не нужно использовать громоздкие и сложные процедуры нормализации универсального отношения.

Во-вторых, простота нотаций графических языков, строго определенные правила именования сущностей, атрибутов и связей, а также содержащиеся в глоссарии определения имен обеспечивают точную передачу представлений автора модели и однозначную (и одинаковую!) понимаемость модели всеми участниками разработки – аналитиками, экспертами и проектировщиками структур хранения данных.

В-третьих, строго определенная иерархия уровней модели открывает возможность согласования представлений аналитика с представлениями конечных пользователей системы на всех этапах разработки.

В-четвертых, в настоящее время существует несколько десятков товарных CASE-систем, поддерживающих семантические методологии на всех этапах жизненного цикла системы – от формулировки замысла до выпуска проектной документации. Многие из них обеспечивают отображение окончательных диаграмм модели на физические структуры данных распространенных СУБД, а также создание триггеров ссылочной целостности, как стандартных, так и оригинальных.

В силу указанных причин использование семантического подхода значительно снижает трудозатраты на ранних этапах проектирования системы, упрощает и облегчает верификацию моделей и обеспечивает создание высококачественных спецификаций систем баз данных.

5.6 Уровни представления диаграмм

Методология информационного моделирования IDEF1X (Integrated DEFinitions 1 eXpanded) различает три уровня графического представления информации, три уровня диаграмм или три уровня логической модели, отличающихся по глубине представления информации о данных:

- уровень «сущность – связь» (Entity Relationship Level, ER);

- уровень ключей (Key-Based Level, KB);
- уровень атрибутов (Fully Attributed Level, FA) [22, 24].

1. **Уровень «сущность – связь» (ER level).** Это уровень наименее детального представления информации. Он используется на начальной стадии моделирования, когда еще не выяснены или не поняты до конца свойства сущностей и связей. На диаграммах ER-уровня сущности и связи представлены только их именами.

При разработке концептуальной модели прежде всего следует определить сущности. С этой целью нужно сделать следующее [3]:

- понять, какая информация должна храниться и обрабатываться и можно ли это определить как сущность;
- присвоить этой сущности имя;
- выявить атрибуты сущности и присвоить им имя;
- определить уникальный идентификатор сущности.

Выявив сущности, необходимо определить, какие связи имеются между ними.

При определении связей (следует рассмотреть только те связи, которые имеют отношение к решаемым задачам обработки данных) необходимо учитывать следующее:

- то, как экземпляр одной сущности связан с экземпляром другой сущности;
- то, как должны быть установлены связи, чтобы была возможность ответа на все запросы пользователей (исходя из их информационных потребностей).

Далее необходимо присвоить связям имена и определить тип связей.

На втором этапе построенные локальные модели объединяются в обобщенную концептуальную модель.

Диаграмма «сущность – связь» представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма «сущность – связь» может включать связи «многие-ко-многим» и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

На ER-уровне сущности не различаются как зависимые или независимые, а соединения – как идентифицирующие и неидентифицирующие. Сущности не содержат горизонтальных линий, отделяющих область ключей от области данных. Имена сущностей вписываются в обозначающие их прямоугольники.

Диаграмма ER-уровня может показывать категории, но указывать дискриминаторы кластеров необязательно.

На ER-уровне допустимы неспецифические соединения. Для изображения соединений можно использовать как сплошные, так и штриховые линии. Это не специфицирует соединения.

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (*Relationship Verb Phrases*).



.....
Под сущностью в IDEF1X понимается отношение. Сущности изображаются на диаграммах именованными прямоугольниками, в которые вписываются имена атрибутов.

Сущность – это объект, событие или концепция, информация о которых должна сохраняться.

Каждая сущность должна обладать следующими свойствами [1, 4]: иметь уникальное имя; обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь; обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности. Каждая сущность может обладать любым количеством связей с другими сущностями.

Между сущностями возможны четыре типа связей:

- один-к-одному ($1 \leftrightarrow 1$);
- один-ко-многим ($1 \leftrightarrow \infty$);
- многие-к-одному ($\infty \leftrightarrow 1$);
- многие-ко-многим ($\infty \leftrightarrow \infty$).

Связь $1 \leftrightarrow 1$: в любой момент времени каждому экземпляру первого информационного объекта (ИО) соответствует 1 или 0 экземпляров другого ИО, и наоборот.

Связь $1 \leftrightarrow \infty$: одному экземпляру первого ИО соответствует 0, 1, 2, ... экземпляров другого, и наоборот, каждому экземпляру второго ИО соответствует 0 или 1 экземпляр первого ИО. Аналогично определяется тип связи $\infty \leftrightarrow 1$.

Связь $\infty \leftrightarrow \infty$: одному экземпляру первого ИО соответствует 0, 1, 2, ... экземпляров другого ИО, и наоборот.

На ER-уровне независимые и зависимые сущности не различаются, атрибуты не указываются, а имена сущностей вписываются в обозначающие их прямоугольники.

Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы, например:

- каждый ТОВАР <хранится на> СКЛАДе;
- каждая НАКЛАДНАЯ <содержит> ТОВАР;
- каждый ПОКУПАТЕЛЬ <получает> НАКЛАДНую (рис. 5.6).



Рис. 5.6 – ER-диаграмма со связями между сущностями

Связь показывает, какие именно товары содержатся в накладной и какой именно покупатель получает конкретную накладную с товаром, который хранится на складе. По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт *Display Options/Relationship* и затем включить опцию *Verb Phrase*.

На логическом уровне можно установить идентифицирующую связь «один-ко-многим», связь «многие-ко-многим» и неидентифицирующую связь «один-ко-многим» (соответственно это кнопки слева направо в палитре инструментов).

2. Уровень ключей (Key-Based Level, KB). На этом уровне в диаграммах отражаются имена первичных и внешних ключей сущностей и спецификации связей. Диаграмма KB-уровня объявляет уникальные идентификаторы экземпляров сущностей и ограничения ссылочной целостности.

Модель данных (КВ), основанная на ключах, – более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Ключи. Каждый экземпляр сущности должен быть уникален и отличаться от других атрибутов.



.....

Первичный ключ (*primary key*) – это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности.

.....

Атрибуты первичного ключа на диаграмме не требуют специального обозначения – это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии. При внесении нового атрибута в диалоге *Attribute Editor*, для того чтобы сделать его атрибутом первичного ключа, нужно включить флажок *Primary Key* в нижней части закладки *General*. На диаграмме неключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка в палитре инструментов).

ERwin автоматически производит миграцию ключевых атрибутов родительской сущности в дочернюю сущность, где они становятся внешними ключами.

Атрибуты ключа не должны содержать нулевых значений.

Каждая сущность должна иметь по крайней мере один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные – альтернативными ключами.



.....

Альтернативный ключ (*Alternate Key*) – это потенциальный ключ, не ставший первичным. *ERwin* позволяет выделить атрибуты альтернативных ключей, и по умолчанию в дальнейшем при генерации схемы БД по этим атрибутам будет генерироваться уникальный индекс.

.....

Ключи могут быть *сложными*, т. е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения, это список атрибутов выше горизонтальной линии.

На диаграмме атрибуты альтернативных ключей обозначаются как (АК n . m), где n – порядковый номер ключа, m – порядковый номер атрибута в ключе. Когда альтернативный ключ содержит несколько атрибутов, (АК n . m) ставится после каждого. Например, атрибуты *Фамилия*, *Имя*, *Отчество* и *Дата рождения* входят в альтернативный ключ № 1 (АК1), *Номер паспорта* составляет альтернативный ключ № 2 (АК2).

Для того чтобы стать первичным, потенциальный ключ должен удовлетворять следующим требованиям.

Уникальность. Два экземпляра не должны иметь одинаковых значений возможного ключа.

Компактность. Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности. Тогда второй ключ, претендующий на роль первичного, не является компактным, так как удаление энергичности не приведет к утрате уникальности.

Диаграммы КВ-уровня должны удовлетворять следующим правилам для ключей.

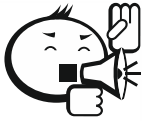
1. На диаграммах КВ- и FA-уровней каждая сущность должна иметь первичный ключ.
2. Сущность может иметь несколько альтернативных ключей.
3. Как первичный, так и альтернативный ключ может быть либо одиночным атрибутом, либо группой атрибутов.
4. Отдельный атрибут может быть частью более чем одного ключа, первичного или альтернативного.
5. Атрибуты, входящие в первичный и альтернативный ключи, могут быть как собственными атрибутами сущности, так и присоединенными через связь с другой сущностью.
6. Первичный и альтернативные ключи должны содержать только те атрибуты, которые необходимы для уникальной идентификации экземпляров сущности.
7. Каждый неключевой атрибут должен неприводимо зависеть от первичного ключа, если он составной.
8. Каждый атрибут, не являющийся частью первичного ключа или какого-либо из альтернативных, должен функционально зависеть только от первичного ключа и каждого из альтернативных [26].

Правила для внешних ключей приведены ниже.

- Каждая сущность, являющаяся потомком в специфической связи или категорией в категоризационной связи, должна содержать множество атрибутов – внешних ключей, переданных связью. Конкретный атрибут может быть элементом нескольких таких множеств. Число атрибутов в каждом множестве внешних ключей должно совпадать с числом атрибутов первичного ключа родительской или родовой сущности.
- Первичный ключ родовой сущности должен передаваться как первичный ключ каждой категории.
- Потомок не может содержать двух полных внешних ключей, которые соотносят с каждым его экземпляром один и тот же экземпляр одного и того же предка, если эти внешние ключи не переданы через различные пути связей, включающие по крайней мере одну промежуточную сущность между этим предком и потомком.
- Каждый присоединенный атрибут потомка или категории должен быть атрибутом первичного ключа связанной с ним родительской или родовой сущности. Обратное, каждый атрибут первичного ключа родительской или родовой сущности должен быть присоединенным атрибутом связанного с нею потомка или категории.
- Каждое имя роли, назначенное присоединенному атрибуту, должно быть уникальным, и в одно и то же имя всегда должен вкладываться один и тот же смысл. Один и тот же смысл не может вкладываться в разные имена, если они не являются псевдонимами.
- Присоединенный атрибут может быть частью более чем одного множества внешних ключей при условии, что он имеет одно и то же значение в этих множествах в некотором фиксированном экземпляре сущности. Такому присоединенному атрибуту может быть назначено имя роли.
- Каждый внешний ключ должен ссылаться на один и только один атрибут первичного ключа родителя [26].

3. **Уровень атрибутов (Fully Attributed Level, FA).** Диаграмма FA-уровня показывает имена всех атрибутов сущностей и связей и полностью определяет структуру и взаимосвязи данных.

Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи [26].



.....

Цель моделирования – создание *FA*-диаграммы. Она является графическим представлением структуры реляционной базы данных с полностью определенными схемами отношений.

.....

Диаграмма *FA*-уровня должна содержать все, что содержит диаграмма *KB*-уровня, и, кроме того, все неключевые атрибуты. На *KB*- и *FA*-уровнях в полной мере действуют все правила синтаксиса, изложенные выше.

Основные компоненты диаграммы *ERwin* – это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте.

Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущность *Заказчик* (но не *Заказчики*!) с атрибутами *Номер заказчика*, *Фамилия заказчика* и *Адрес заказчика*. На уровне физической модели ей может соответствовать таблица *Customer* с колонками *Customer_number*, *Customer_name* и *Customer_address*.

Для внесения сущности в модель необходимо (убедившись предварительно, что вы находитесь на уровне логической модели – переключателем между логической и физической моделью служит раскрывающийся список в правой части панели инструментов) «кликнуть» по кнопке сущности на панели инструментов (*ERwin Toolbox*), затем «кликнуть» по тому месту на диаграмме, где необходимо расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт *Entity Editor*, можно вызвать диалог *Entity Editor*, в котором определяются имя, описание и комментарии сущности.

Каждая сущность должна быть полностью определена с помощью текстового описания в закладке *Definition*. Закладки *Note*, *Note 2*, *Note 3*, *UDP (User Defined Properties – Свойства, определенные пользователем)* служат для внесения дополнительных комментариев и определений к сущности. В прежних версиях ERwin закладкам *Note 2* и *Note 3* соответствовали окна *Query* и *Sample*.

Закладка *Definition* используется для ввода определения сущности. Эти определения полезны как на логическом уровне, поскольку позволяют понять, что это за объект, так и на физическом уровне, поскольку их можно экспортировать как часть схемы и использовать в реальной БД (`CREATE COMMENT on entity_name`).

Закладка *Note* позволяет добавлять дополнительные замечания о сущности, которые не были отражены в определении, введенном в закладке *Definition*. Здесь можно ввести полезное замечание, описывающее какое-либо бизнес-правило или соглашение по организации диаграммы.

В закладке *Note 2* можно задокументировать некоторые возможные запросы, которые, как ожидается, будут использоваться по отношению к сущности в БД. При переходе к физическому проектированию записанные запросы помогут принимать такие решения в отношении проектирования, которые сделают БД более эффективной.

Закладка *Note 3* позволяет вводить примеры данных для сущности (в произвольной форме).

В закладке *Icon* каждой сущности можно поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

Часто приходится создавать производные атрибуты, т. е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить *Возраст сотрудника*, который может быть вычислен из атрибута *Дата рождения сотрудника*. Такой атрибут может привести к конфликтам; действительно, если вовремя не обновить значение атрибута *Возраст сотрудника*, он может противоречить значению атрибута *Дата рождения сотрудника*. Производные атрибуты – ошибка нормализации, однако их вводят для повышения производительности системы – если необходимо узнать возраст сотрудника, можно обратиться к соответствующему атрибуту, а не проводить вычисления (которые на практике могут быть значительно более сложными, чем в приведенном примере) по дате рождения.

В IDEF1X различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (сущность ПОЗИЦИЯ ЗАКАЗА на рисунке 5.7). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т. е. в структуре на рисунке 5.7 информация о позиции заказа не может быть внесена и не имеет смысла без информации о заказе, который его размещает. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ (FK).

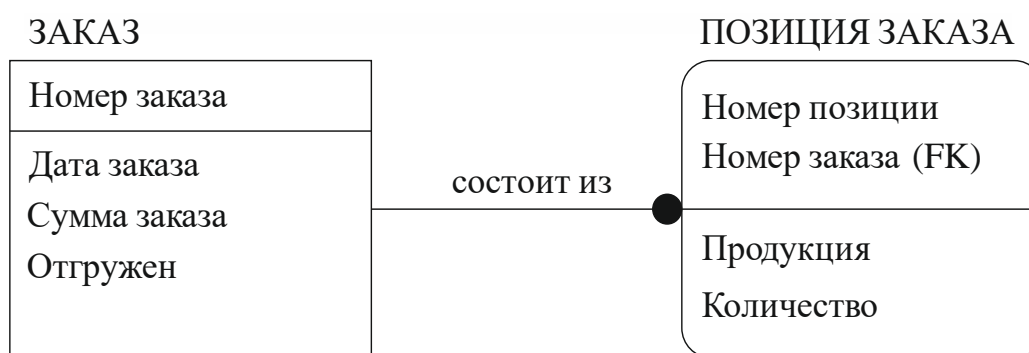


Рис. 5.7 – Идентифицирующая связь между независимой и зависимой таблицей

Синтаксис графического языка IDEF1X обеспечивает однозначное представление ограничений ссылочной целостности в диаграммах KB- и FA-уровня. Правила именования и определения сущностей, доменов и атрибутов дают возможность задать ограничения на значения в текстовых документах, сопровождающих диаграмму. В силу этого трансляция FA-диаграммы в тексты описания таблиц БД на языке конкретной СУБД оказывается чисто формальной процедурой и может выполняться автоматически.

5.7 Основные правила стандарта IDEF1X

Основные правила стандарта IDEF1X приведены ниже.

1. Сущности, атрибуты и домены обязательно именуются. Именем может быть только имя существительное, возможно с определениями. В качестве имен допускаются аббревиатуры и акронимы.

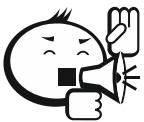
2. Имя должно быть уникальным и осмысленным. Формальное определение имени обязательно включается в глоссарий модели.

Создавая модель, проектировщик стремится сформировать ясное представление о ПО, в частности о том, какие сведения будут храниться в БД. Этого можно добиться, только сформулировав точные и однозначные определения смысла каждого имени, введенного в модель [26].

Например, что такое ДЕТАЛЬ? Это неделимая часть изделия, или она сама может собираться из других деталей? Может ли ИЗДЕЛИЕ быть ДЕТАЛЬю? Наша фирма только закупает ДЕТАЛи или может их производить?

Или что такое ФИЛЬМ? Это лента, лежащая в нашей фильмотеке, или это произведение киноискусства? Нас интересуют любые фильмы или только фильмы определенного жанра?

От ответов на эти вопросы зависит организация данных и их взаимосвязи. Поэтому прежде чем строить диаграммы, необходимо получить точные ответы на все подобные вопросы и зафиксировать документально определения смысла имен. Без этого модель будет неоднозначной и противоречивой.



.....

Имя сущности, атрибута или домена должно иметь единственный смысл, и этот смысл всегда должен выражаться этим именем. Тот же смысл не может вкладываться в другое имя, если оно не является псевдонимом или синонимом основного.

.....

Например, атрибут *дата* не может иметь смысл даты *начала ИЛИ окончания* отчетного периода. Совершенно непонятно, как интерпретировать значения этого атрибута в различных кортежах отношения.



.....

Сущности и атрибуты всегда именуются в единственном числе. Имя должно относиться к одному экземпляру сущности или значению атрибута.

.....

Соблюдение этих правил обеспечивает интерпретацию диаграмм фразами естественного языка и точную передачу смысла, вложенного в имена автором модели.

3. Атрибут есть свойство или характеристика, общая для некоторых или всех экземпляров сущности. Атрибут является конкретизацией домена в контексте сущности.

4. На диаграммах KB- и FA-уровней каждая сущность имеет не менее одного атрибута. Каждый атрибут может быть собственным атрибутом сущности или присоединенным (мигрировавшим), полученным от другой сущности через связь.

5. Каждый атрибут является собственным атрибутом точно одной сущности.

6. Присоединенный атрибут должен быть частью первичного ключа передавшей его сущности (родительской или родовой). Присоединенный атрибут помечается символом (FK), следующим за именем атрибута.

В совокупности эти требования означают, что никакие две сущности не могут иметь одноименных атрибутов, если они не связаны каким-либо отношением. В последнем случае одноименными могут быть атрибуты, которые являются частью первичного ключа родителя и частью внешнего ключа потомка.

7. Каждый экземпляр сущности должен иметь определенное значение каждого атрибута, являющегося частью первичного ключа.

8. Не может быть экземпляра сущности, имеющего более чем одно значение какого-либо из атрибутов.

9. Атрибуты, не являющиеся частью первичного ключа, могут иметь неопределенные значения. Такие атрибуты помечаются символом (O), следующим за именем атрибута (*Optional* – необязательный).

10. Если атрибут является собственным атрибутом одной сущности и присоединенным атрибутом другой, то либо он имеет одинаковые имена в обеих сущностях, либо помечается именем роли как присоединенный.

11. Определение атрибута должно содержать ссылку на имя домена.

12. На KB-диаграммах показываются только атрибуты, входящие в состав первичных, альтернативных и внешних ключей. Атрибуты альтернативных ключей обязательно помечаются символом (AK n), где n – номер альтернативного ключа. Все атрибуты, входящие в состав одного и того же альтернативного ключа, помечаются одним и тем же значением n . На FA-диаграмме показываются все атрибуты каждой сущности.

13. Соединение – это один из двух видов связей, используемых в языке IDEF1X. Стандарт определяет соединение как ассоциацию между двумя сущностями или между экземплярами одной и той же сущности.

14. В диаграммах IDEF1X представляются только бинарные и унарные связи (соединения), а также иерархии (связи категоризации). Отсутствие обозначений для представления связей высшей арности не является ограничением модели. Неспецифические соединения могут быть показаны только на диаграммах ER-уровня.

15. Соединению присваивается имя, выражаемое глагольным оборотом. Имя зрительно привязывается к дуге, изображающей соединение. Имена соединений могут быть неуникальными в пределах диаграммы.

16. Имя каждого соединения одной и той же пары сущностей должно быть уникальным во множестве имен связей этих сущностей.

17. Имена специфических соединений выбираются так, чтобы можно было построить осмысленную фразу, составленную из имени родительской сущности, имени связи, выражения кардинальности и имени потомка.

18. Связь может быть поименована «от родителя» и «от потомка». Имя «от родителя» обязательно.

19. Если связь не именуется со стороны потомка, то имя «от родителя» должно выбираться так, чтобы связь легко читалась и со стороны потомка.

20. Неспецифические соединения обязательно именуются с обеих сторон.

21. Сущность может иметь несколько альтернативных ключей.

22. Как первичный, так и альтернативный ключ может быть либо одиночным атрибутом, либо группой атрибутов.

23. Отдельный атрибут может быть частью более чем одного ключа, первичного или альтернативного.

24. Атрибуты, входящие в первичный и альтернативный ключи, могут быть как собственными атрибутами сущности, так и присоединенными через связь с другой сущностью.

25. Первичный и альтернативные ключи должны содержать только те атрибуты, которые необходимы для уникальной идентификации экземпляров сущности.

26. Каждый неключевой атрибут должен неприводимо зависеть от первичного ключа, если он составной.

27. Каждый атрибут, не являющийся частью первичного ключа или какого-либо из альтернативных, должен функционально зависеть только от первичного ключа и каждого из альтернативных [24].

5.8 Дополнения к модели и лексические соглашения стандарта IDEF1X

Диаграммы модели сопровождаются дополнительными материалами, уточняющими и поясняющими ее смысл. Имеются два вида дополнений: *гlossарий* и *примечания* [15, 24–27].

Гlossарий является обязательным дополнением. Он содержит описания отдельных диаграмм модели и определения сущностей, атрибутов и доменов.

Обязательные компоненты гlossария:

- имена – уникальные, осмысленные и соответствующие природе ПО наименования сущностей, атрибутов и доменов;
- определения – краткие, точные, однозначные тексты, обеспечивающие правильное понимание смысла имен, одинаковое для любого читателя.



.....
 Определение должно быть одинаково применимо в любом контексте, в котором встречается имя.

Кроме того, гlossарий может содержать список псевдонимов. Каждому имени могут соответствовать в различных контекстах псевдонимы. Необходимость их использования обусловлена тем, что в различных частях моделируемой ПО может использоваться разная терминология. Поэтому одни и те же вещи могут называться разными именами. Полный список псевдонимов должен сопровождать каждое имя.

Примечания – это необязательный вид дополнений. Они могут пояснять смысл диаграмм и их элементов, содержать описания путей связей, сообщать о функциях, связанных с тем или иным объектом и т. п.

Примечания нумеруются числами натурального ряда. На диаграмме ссылка на примечание помещается около соответствующего объекта в круглых скобках.

Для того чтобы обеспечить наглядность и читаемость диаграмм, стандарт IDEF1X рекомендует придерживаться ряда *соглашений* относительно построения имен и фраз на диаграммах.

Имена. В именах сущностей и атрибутов используются только буквы, цифры, а также знаки-разделители: «дефис», «подчерк» и «пробел». Имя должно начинаться с *буквы*. Части составного имени отделяются дефисом, подчеркиком или пробелом. Эти разделители не различаются.

Рекомендуется имена сущностей на диаграммах и в текстах глоссариев и замечаний всегда писать ПРОПИСНЫМИ буквами.

Например, глоссарий может содержать такое определение имени ПОСТАВЩИК: «Юридическое лицо, заключившее с фирмой договор на поставку одного или нескольких видов ДЕТАЛЕЙ для одного или нескольких видов ИЗДЕЛИЙ».

Это определение содержит сведения не только о смысле, но и о связях определяемой сущности с другими сущностями ПО. Выделение имен этих сущностей регистром облегчает зрительное восприятие фактов связи.

Помимо вышеперечисленных разделителей в именах используются еще «•» и «/». Точка отделяет имя роли от основного имени атрибута. Имя роли предшествует точке, основное имя следует за точкой. Ни перед, ни после точки не допускаются другие разделители. Слеш разделяет разнонаправленные имена связи. Кроме того, он разделяет имя сущности или связи и его *идентификатор*.

Литература

1. Федеральный государственный образовательный стандарт высшего образования (ФГОС ВО) по направлению подготовки 09.03.03 «Прикладная информатика» (уровень бакалавриата), утвержден Приказом Министерства образования и науки Российской Федерации от 12.03.2015 № 207 (зарегистрирован в Минюсте России 27.03.2015 № 36589).
2. Образовательный стандарт вуза ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления [Электронный ресурс]. – Томск : ТУСУР, 2013. – 57 с. – Режим доступа: <https://regulations.tusur.ru/documents/70> (дата обращения: 09.09.2020).
3. Грибова, В. В. Методы и средства разработки пользовательского интерфейса: современное состояние [Электронный ресурс] / В. В. Грибова, А. С. Клещев // Программные продукты и системы. – 2001. – № 1. – С. 2–6. – Режим доступа: <http://www.swsys.ru/index.php?page=article&id=765> (дата обращения: 09.09.2020).
4. Основы информационной безопасности : учеб. пособие для вузов / Е. Б. Белов, В. П. Лось, Р. В. Мещеряков, А. А. Шелупанов. – М. : НТИ «Горячая линия-Телеком», 2011. – 544 с.
5. Нечаев, Д. В. Методика защиты персональных данных : доклад / Д. В. Нечаев // Научная сессия ТУСУР – 2010. – Томск : В-Спектр, 2010. – Ч. 3. – С. 176–178.
6. Требования о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах : метод. материал / Федеральная служба по техническому и экспортному контролю. – М. : ГНИИИ ПТЗИ ФСТЭК России, 2013. – 38 с.
7. Избачков, Ю. С. Информационные системы : учеб. пособие для вузов / Ю. С. Избачков, В. Н. Петров. – 2-е изд. – СПб. : Питер, 2005. – 655 [1] с.
8. Информационные технологии в экономике и управлении : учебник для бакалавров / Санкт-Петербургский государственный университет экономики и финансов ; ред. В. В. Трофимов. – М. : Юрайт, 2016. – 482 с.
9. Исакова, А. И. Информационные системы : учеб. пособие / А. И. Исакова. – Томск : ФДО, ТУСУР, 2010. – 202 с.

10. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 4-е изд., стереотип. – М. : Академия, 2010. – 320 с.
11. Полонская, Е. В. Сравнительный анализ возможностей СУБД PostgreSQL, MySQL и MS ACCESS : материалы доклада / Е. В. Полонская // Научная сессия ТУСУР – 2009. – Томск : В-Спектр, 2009. – Ч. 1. – С. 237–240.
12. Сенченко, П. В. Организация баз данных : учеб. пособие / П. В. Сенченко. – Томск : ФДО, ТУСУР, 2015. – 170 с.
13. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 5-е изд., испр. – М. : Академия, 2012. – 320 с.
14. Маккоу, Алекс. Веб-приложения на JavaScript : практ. руководство / А. Маккоу ; пер. Н. Вильчинский. – СПб. : ПИТЕР, 2012. – 288 с.
15. Ли, Джеймс. Использование Linux, Apache, MySQL и PHP для разработки WEB-приложений : пер. с англ. / Джеймс Ли, Brent Уэр ; пер. А. Н. Узниченко. – М. : Вильямс, 2004. – 429 с.
16. Тидвелл, Дженифер. Разработка пользовательских интерфейсов : пер. с англ. / Дж. Тидвелл ; пер. Е. Шикарева. – СПб. : Питер, 2008. – 416 с.
17. Колисниченко, Д. Н. Разработка Linux-приложений : научно-популярное издание / Д. Н. Колисниченко. – СПб. : БХВ-Петербург, 2012. – 431 с.
18. Туманов, В. Е. Проектирование реляционных хранилищ данных : справочное издание / В. Е. Туманов, С. В. Маклаков. – М. : ДИАЛОГ-МИФИ, 2007. – 336 с.
19. Мытник, С. А. Проектирование информационных систем : учеб. пособие / С. А. Мытник. – Томск : ТМЦДО, 2008. – 163 с.
20. Золотов, С. Ю. Проектирование информационных систем : учеб. пособие / С. Ю. Золотов. – Томск : Эль Контент, 2013. – 87 с.
21. Коннолли, Томас. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Томас Коннолли, Каролин Бегг ; пер. с англ. – 3-е изд. – М. : ИД «Вильямс», 2003. – 1440 с.
22. Диго, С. М. Базы данных : Проектирование и использование : учебник для вузов / С. М. Диго. – М. : Финансы и статистика, 2005. – 590 с.
23. Методология функционального моделирования IDEF0. Руководящий документ. РД IDEF0. Госстандарт России. – М. : ИПК Издательство стандартов, 2000. – 75 с.

24. Сибилев, В. Д. Базы данных : учеб. пособие / В. Д. Сибилев. – Томск : ТУСУР, 2007. – 279 с.
25. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2012. – 464 с.
26. Давыдова, Е. М. Базы данных : учеб. пособие / Е. М. Давыдова, Н. А. Новгородова. – 3-е изд., перераб. и доп. – Томск : В-Спектр, 2012. – 128 с.
27. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 5-е изд., испр. – М. : Академия, 2012. – 320 с.

Приложение А
(справочное)
Пример отчета по контрольной работе

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

**ВЫБОР СПОСОБА АВТОМАТИЗАЦИИ УЧЕТА ДОГОВОРОВ
НА ПРЕДПРИЯТИИ ООО «ЛЕМА ГРУПП» г. МОСКВЫ**

Отчет по контрольной работе
по дисциплине «Научно-исследовательская работа в семестре»

Студент гр. ____
____ И. О. Фамилия
«__» _____ 20__ г.

Руководитель
доцент каф. АСУ,
канд. техн. наук
____ А. И. Исакова
«__» _____ 20__ г.

Томск 20__

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

ЗАДАНИЕ

на выполнение контрольной работы по дисциплине
«Научно-исследовательская работа в семестре»:
студенту гр. _____ факультета

(Ф. И. О. студента)

1. Тема НИР:

Выбор способа автоматизации учета договоров в профильной организации ООО «ЛЕМА групп» г. Москвы.

2. Цель НИР:

Исследовать рынок программного обеспечения на наличие готовых информационных систем учета договоров в профильной организации.

3. Задачи НИР:

- выбор критериев (требований) к информационной системе;
- сравнительный обзор программ-аналогов;
- выбор среды реализации ИС.

4. Исходные данные для НИР:

- документы профильной организации (предприятия), необходимые для решения требуемых задач;
- результаты исследований, полученные при изучении дисциплины «Учебно-исследовательская работа».

5. Технические требования к отчету по НИР: обязательно соблюдение образовательного стандарта ТУСУР 01–2013.

Дата выдачи: « ___ » _____ 20__ г.

Руководитель НИР от университета

Доцент каф. АСУ, к.т.н.

(должность)

(подпись)

А. И. Исакова

(Ф. И. О.)

Студент гр. _____

(подпись)

(Ф. И. О.)

Оглавление

1 Введение.....	92
2 Особенность учета и контроля договорных отношений сделок	
ООО «ЛЕМА групп» с контрагентами	94
2.1 Виды договоров в ООО «ЛЕМА групп».....	94
2.2 Система классификации задач управления договорами	96
2.3 Договор подряда/субподряда	97
2.4 Функциональное моделирование	100
3 Автоматизация учета договоров	101
3.1 Аналоги программного обеспечения	101
3.1.1 ИС «Ведение договоров» от ARAX GROUP.....	101
3.1.2 ИС «АстроСофт: Учет договоров».....	104
3.1.3 ИС «Респект: Учет договоров».....	105
3.1.4 «Система управления договорами».....	108
3.2 Обоснование выбора программных средств для реализации проекта ...	113
3.3 Требования, предъявляемые к БД в ИС	117
4 Заключение.....	120
Список использованных источников	121

1 Введение

Во время выполнения данной работы по дисциплине «Научно-исследовательская работа в семестре» необходимо было провести исследовательскую работу по сравнению аналогов информационных систем учета договоров на предприятии по функциональному назначению, основным техническим и эксплуатационным параметрам, областям применения, для того чтобы убедиться в том, что все-таки необходимо разрабатывать собственную информационную систему (ИС) [1].

Следующий этап работы – исследовательская работа по изучению возможных сред разработки собственной ИС. Необходимо обосновать выбранную среду разработки.

Цель работы:

– поиск и анализ аналогов проектируемой ИС, сравнение с внедряемой и оценка стоимости разработки и внедрения.

Задачи работы:

- подробное описание необходимых требований к аналогам;
- определение минимальных функций;
- определение стоимости внедрения аналогов;
- проверка масштабируемости и совместимости аналогов;
- оценка пользовательского интерфейса аналогов.

Объектом исследования в данной работе является организация ООО «ЛЕМА групп», занятая в области строительства и оптовой торговли. Заключение сделок ООО «ЛЕМА групп» с разного рода заказчиками происходит путем подписания договоров – соглашений двух или нескольких сторон о взаимных обязательствах [2].

Руководство ООО «ЛЕМА групп» заинтересовано в автоматизации учета и контроля договоров, согласно которым осуществляется хозяйственная деятельность. Данная автоматизация была бы удобна для службы заказчика-застройщика, бухгалтерии, юридической службы, менеджеров проектов, облегчила бы контроль

исполнения ключевых обязательств договоров по выполнению работ/услуг и их оплаты.

Поиском и анализом аналогов разрабатываемой ИС и обоснованием сред разработки ИС возможно достигнуть уменьшение стоимости разработки и внедрения программного продукта.

2 Особенности учета и контроля договорных отношений сделок ООО «ЛЕМА групп» с контрагентами

2.1 Виды договоров в ООО «ЛЕМА групп»

Заключаемые организацией ООО «ЛЕМА групп» договоры относятся к следующим категориям [2]:

- договоры генерального подряда;
- договоры подряда/субподряда;
- договоры поставки;
- договоры купли-продажи;
- договоры об оказании услуг;
- агентские договоры;
- договоры о совместной деятельности.

Из списка категорий договоров видно, что виды работ по договорам различные, есть краткосрочные договоры, а есть договоры, выполнение работ по которым продолжается длительное время. Также нужно контролировать различные виды нарушений: по срокам оплаты, по срокам выполнения договора, по срокам выполнения этапов договора, по предоплате, по суммам.

Основная деятельность компании ООО «ЛЕМА групп» – деятельность в области управления проектами. Управление проектами как управление изменениями является на сегодняшний день интенсивно развивающейся областью теории управления, результаты исследований в которой находят широкое применение на практике. В крупных проектах, как правило, участвует значительное число исполнителей (агентов), взаимодействие которых с заказчиками (центрами) регламентируется договорами [2].

Можно выделить три общих аспекта описания договорных отношений (рис. 2.1) [3].

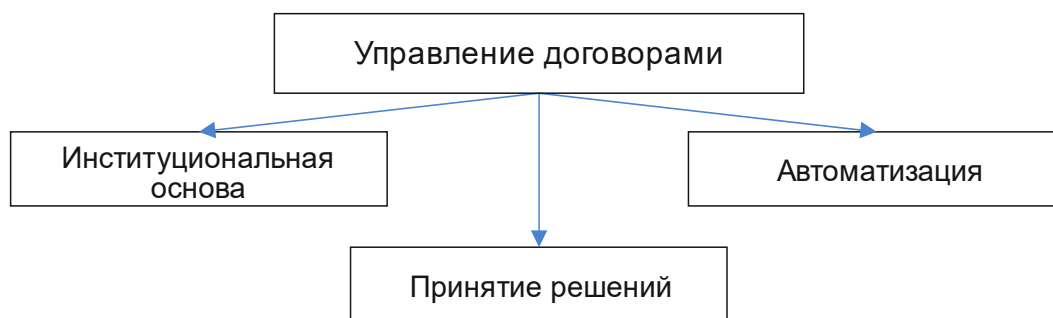


Рисунок 2.1 – Аспекты рассмотрения договорных отношений

Первый соответствует правовым нормам, регламентирующим взаимодействие договаривающихся сторон, то есть институциональным ограничениям, которые не являются предметом настоящего исследования. Второй аспект – аспект принятия решений, с точки зрения которого существуют механизмы управления договорами, то есть модели и методы (процедуры) принятия решений участниками договорных отношений.

И, наконец, третий аспект – автоматизация управления договорами (регистрация, хранение, обработка и т. д. соответствующей информации) [3]. В настоящей работе затрагивается именно этот аспект.

Существуют типовые схемы взаимодействия участников проекта (представлены на рисунках 2.2 и 2.3).

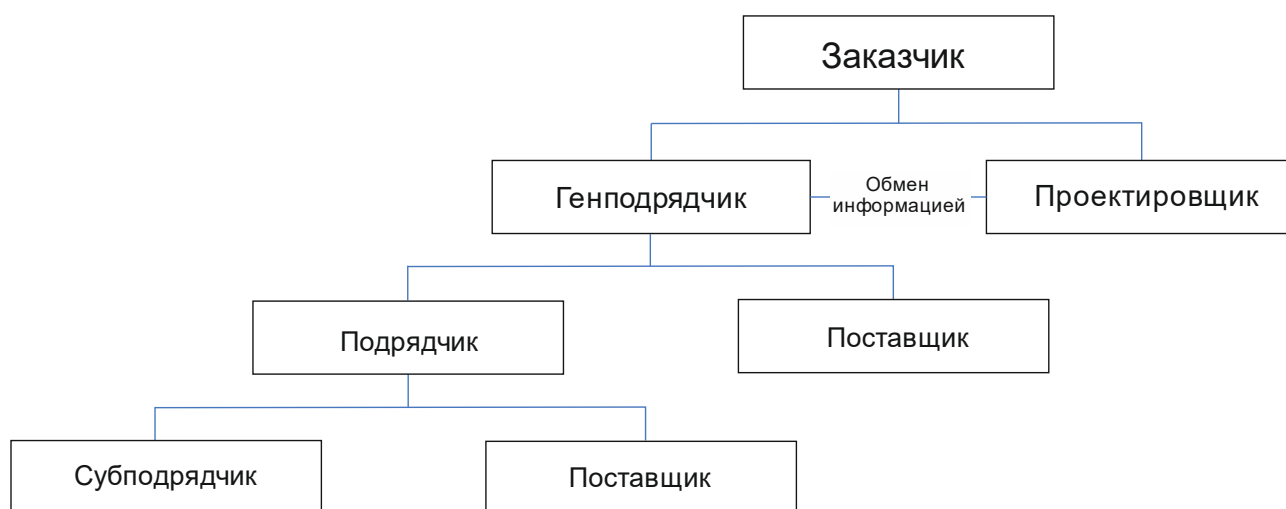


Рисунок 2.2 – Традиционная схема договорных отношений

Договором в гражданском праве называется «соглашение двух или более лиц об установлении, изменении или прекращении гражданских прав и обязанностей (заем, купля-продажа, подряд и др.)» или «соглашение, обычно письменное, о взаимных обязательствах».



Рисунок 2.3 – Схема договорных отношений в управлении проектами

Договорные отношения – распространенный в практике управления тип отношений между экономическими субъектами – заказчиком и исполнителем, отражающий содержание и условия их обоюдовыгодного взаимодействия [3].

2.2 Система классификации задач управления договорами

Договор – это юридический и операционный документ, фиксирующий достигнутые между сторонами соглашения и условия их выполнения [3].

Работа с договорами является составной частью процесса управления, где договоры используются как источник, определяющий действия, права и обязанности сторон.

Под термином «договор» понимают также гражданское правоотношение, возникшее из договора, и документ, в котором изложено содержание договора, заключенного в письменной форме.

Выделяют следующие общие этапы договорных отношений:

- подготовка договора;
- заключение договора;
- выполнение работ по договору;
- завершение договора.

Соответствие между этапами договорных отношений и задачами управления устанавливается таблицей 2.1.

Таблица 2.1 – Соответствие между задачами управления и этапами договорных отношений

Этапы договорных отношений	Задачи управления
Подготовка договора	Планирование и выбор контрагентов
Заключение договора	Определение параметров договора
Выполнение работ по договору	Оперативное управление договорами
Завершение договора	Контроль за исполнением и завершением договоров

Отметим, что задачи контроля за исполнением и завершением договора могут быть решены при наличии, во-первых, механизмов планирования, выбора контрагентов, оперативного управления и определения параметров договора и, во-вторых, при наличии аспекта автоматизации [3], т. е. при наличии информационной системы управления договорами, содержащей в себе всю необходимую информацию не только о заключенных договорах, но и о ходе реализации проекта.

2.3 Договор подряда/субподряда

Основным правовым документом, регулирующим отношения между субъектами договорной деятельности в управлении проектами, в особенности инвестиционными проектами в ООО «ЛЕМА групп», является договор подряда/субподряда, при котором подрядчик (исполнитель) обязуется на свой риск выполнить в конкретный срок определенную работу по заданию заказчика с использованием его или своих материалов, а заказчик обязуется принять работу и оплатить ее по обусловленной цене в установленный срок.

В зависимости от предмета договора различают договоры подряда: на строительство, на производство проектных и изыскательских работ, на выполнение научно-исследовательских и опытно-конструкторских работ [2].

Наиболее распространенный набор услуг, которые стремится получить заказчик при размещении подряда на условиях «под ключ», составляют [2]:

- 1) разработка проекта и рабочей документации, включая изыскательские работы;
- 2) передача сопутствующих лицензий и ноу-хау, связанных с предоставлением заказчику технологических знаний и производственного опыта: поставка строительных машин, технологического оборудования и материалов для сооружения объекта;
- 3) обеспечение кадрами с целью производства всего комплекса работ по сооружению объекта и сдачи его в эксплуатацию;
- 4) выполнение строительно-монтажных и пусконаладочных работ по сооружению объекта согласно утвержденному проекту; обеспечение гарантийной эксплуатации объекта;
- 5) осуществление производственно-технического обучения эксплуатационного персонала заказчика.

Договоры подряда/субподряда в ООО «ЛЕМА групп» могут заключаться:

- 1) на конкурсной основе, когда стоимость строительства определена по законченному проекту и имеет твердую цену. Условия конкурса определяются заказчиком. Этот тип договора выгоден заказчику, поскольку он из нескольких предложений-заявок подрядчиков может выбрать наиболее эффективное по совокупности двух важнейших для него параметров – стоимость и продолжительность строительства (разумеется, при оговоренном качестве). Подрядчику этот тип договора также экономически выгоден: поскольку его прибыль заложена в цене, он заинтересован в снижении издержек;
- 2) на основании переговоров заказчика и подрядчика (генерального подрядчика) возможны два варианта свободной (договорной) цены.

Первый вариант – расходы генподрядчика возмещаются по сметной стоимости плюс оплата оправданных перерасходов, плюс сумма гарантированной прибыли. При этом варианте подрядчик застрахован на случай повышения цен

на материалы, транспортные услуги и т. п. Однако на заказчика возлагается обязанность контроля всего хода работ, так как он обязан оплачивать оправданные перерасходы и отклонять оплату перерасходов, возникших исключительно по вине подрядчика. Этот вариант применяется, как правило, при строительстве крупных сложных объектов, при продолжительных сроках их сооружения или когда невозможно с достаточной точностью определить стоимость строительства, а также в условиях экономического спада, когда возрастает хозяйственный риск.

Второй вариант – возмещение расходов подрядчика по фактической стоимости плюс гарантированная прибыль. Вариант наиболее выгоден подрядчику и предлагается им при заключении договора на строительство «под ключ».

В период переговоров с заказчиком о типе договора и стоимости выполнения работ подрядчик представляет заказчику (по его желанию) список предлагаемых им субподрядчиков, которых заказчик вправе отклонить без обсуждения или пригласить для участия в переговорах на обычных условиях.

Для доказательства профессионализма, квалификации, достаточного производственного потенциала и надежности от исполнителей могут быть затребованы следующие сведения:

1) оборот претендента за три последних завершённых финансовых года, если он касается строительных и других работ, сравниваемых по их характеру с работой по предлагаемому заказу, включая данные об участии в долевом сотрудничестве в других совместных предприятиях;

2) выполнение работ за три последних истекших финансовых года, которые по своему характеру сравнимы с работой по предлагаемому заказу;

3) среднегодовое количество использованных за три последних истекших финансовых года трудовых ресурсов, с распределением при необходимости на профессиональные группы (по специальности и квалификации);

4) характеристика машинного парка и технологического оборудования, находящихся в распоряжении претендента для выполнения работ по предлагаемому заказу;

5) подтверждение регистрации данного претендента по месту его нахождения.

2.4 Функциональное моделирование

Учет – функция, направленная на получение полной и достоверной информации о финансово-хозяйственной деятельности предприятия. На высшем уровне учет отсутствует, однако на основе результатов учета в полной мере выполняется анализ деятельности предприятия и регулирование его производства [4].

Контроль и регулирование – это функции, обеспечивающие сравнение планируемых и фактических показателей по качеству работы и объекта управления и реализацию необходимых управляющих воздействий при помощи отклонений от принятой программы работы [4].

Основная деятельность компании в ООО «ЛЕМА групп» – деятельность в области управления проектами. Предприятие является посредником, выполняет заказы (проекты) одних организаций с помощью других. Ищет надежных исполнителей того или иного проекта и контролирует выполнение. Для одних организаций предприятие является заказчиком, а для других – исполнителем. В связи с такими видами деятельности возникает множество заключенных договоров, которые необходимо контролировать и к которым необходимо иметь быстрый доступ.

Таким образом, очень важным является такой аспект описания договорных отношений, как автоматизация управления договорами. Этот аспект включает в себя регистрацию, хранение, обработку и т. д. соответствующей информации по договорам.

3 Автоматизация учета договоров

3.1 Аналоги программного обеспечения

Во время выполнения учебно-исследовательской работы необходимо было провести исследовательскую работу по сравнению аналогов информационных систем учета договоров на предприятии по функциональному назначению, основным техническим и эксплуатационным параметрам, областям применения, для того чтобы убедиться в том, что все-таки необходимо разрабатывать собственную информационную систему (ИС) [5].

Рассмотрим каждую из выбранных систем-аналогов более подробно.

3.1.1 ИС «Ведение договоров» от ARAX GROUP

ИС «Ведение договоров» предназначена для учета и ведения договоров в организациях.

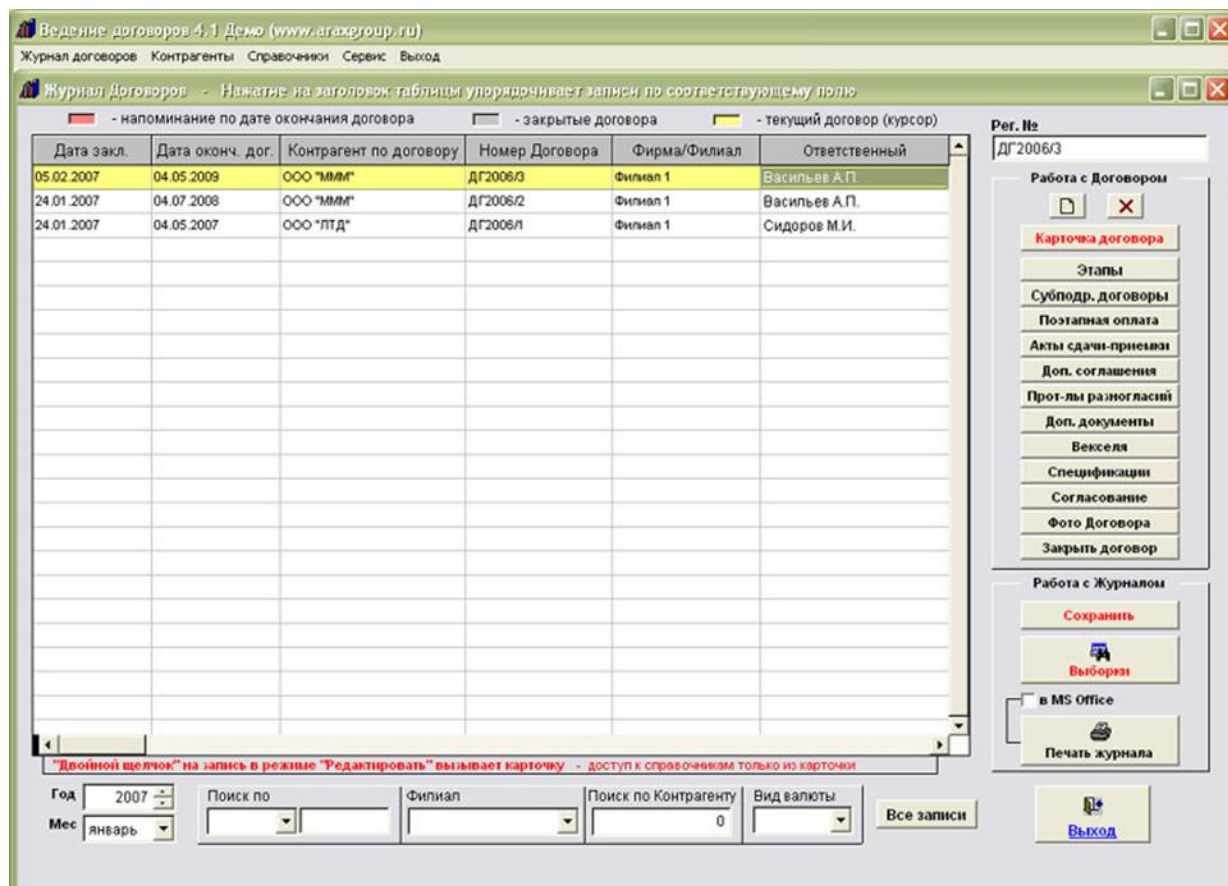


Рисунок 3.1 – Журнал договоров

В программе реализовано (рис. 3.1):

- четыре интерфейса: администратор, пользователь, только ввод и только просмотр;
- журнал договоров, поиски по журналу;
- возможность настройки правила формирования номера договора (в том числе для каждого филиала);
- оповещение по незакрытым договорам, по которым скоро истекает срок действия;
- карточка договора;
- лист согласования с бумажной формой;
- возможность вставки «оригинала» договора в любом формате;
- печать журнала договоров, вывод в MS Office, возможность настройки отчета в MS Excel;
- учет этапов договора и субподрядных договоров;
- согласование договора, оповещение о задержке согласования;
- акты сдачи-приемки, дополнительные соглашения к договору;
- поэтапная оплата по договору, оповещение по поэтапной оплате, возможность настройки оповещений для каждого пользователя;
- выборки по разным критериям с расчетом итоговых сумм;
- экспорт карточки и журнала в MS Word, формирование договора в MS Word по шаблону (как внутреннему в программе, так и на базе шаблонов MS Word *.dot);
- ведение учета и аналитика по трем видам валют (рубль, евро, доллар), возможность ведения договоров любого количества фирм, в каждой из которых несколько филиалов;
- спецификации к договору, векселя к договору с формированием документов в MS Word, экспорт в MS Excel выбранных полей результата поиска по журналам договоров и контрагентов [6].

Для нормальной работы базы данных необходимо выполнение следующих условий:

- операционная система MS Windows 97 (NT 4.0) и выше;
- MS Word 97 и выше;
- библиотеки MS Visual Fox Pro 9.0;
- желательно наличие принтера.

Стоимость лицензий: базовая версия – 1 рабочее место – 5000 руб.

Дополнительное рабочее место: от 2 до 5 включительно – 2000 руб. за 1 рабочее место, от 6 до 10 включительно – 1500 руб. за 1 рабочее место, от 11 и больше – 1000 руб. за 1 рабочее место. Пример расчета стоимости для 6 рабочих мест: $5000 + 4 \times 2000 + 1 \times 1500 = 14\,500$ руб.

Основным недостатком данной системы является то, что необходимо отдельно устанавливать библиотеки MS Visual Fox Pro 9.0.

The screenshot shows the 'Ведение договоров 4.2' application window. The main area is titled 'Карточка договора' and contains the following fields and controls:

- Registration Number:** ДГ2006/184
- Date of Conclusion:** 20.03.2007
- Classification:** поставки валютный
- Income/Expense:** Расходный
- Contract Amount:** 1000000,00 (Rubles)
- Advance:** 30,00%
- Contractor:** Контрагент 1
- Contract Officer:** Директор Петров
- Contract Type:** Трехсторонний договор
- Contract Dates:** 05.05.2005 to 07.07.2007

The right sidebar contains a vertical list of buttons for contract management:

- Печать карточки
- Этапы
- Субподр. договоры
- Поэтапная оплата
- Акты сдачи-приемки
- Доп. соглашения
- Прот-лы разногласий
- Доп. документы
- Спецификации
- Векселя
- Фото Договора
- Файл MS Office
- Согласование
- Фото листа соглас.
- ден Фото ЛС
- Печать договора
- Договор закрыт
- Выход

Рисунок 3.2 – Карточка договора

Из рисунков 3.1 и 3.2 виден неудобный пользовательский интерфейс – не совсем удачное расположение кнопок справа (обычно располагаются слева) – и на рисунке 3.2 расположение строки поиска снизу.

3.1.2 ИС «АстроСофт: Учет договоров»

ИС «АстроСофт: Учет договоров» автоматизирует следующие процессы работы с договорами [7]:

- подготовка;
- согласование;
- регистрация;
- планирование и контроль исполнения обязательств;
- расчеты и учет документов;
- учет штрафных санкций по неисполняемым обязательствам;
- гибкая система для составления отчетности;
- управление взаимоотношениями с контрагентами;
- распределение уровня доступа к информации, в зависимости от пользовательских настроек.

ИС «АстроСофт: Учет договоров» систематизирует и оптимизирует работу с договорами и предоставляет все необходимые инструменты для управления договорами.

ИС «АстроСофт: Учет договоров» автоматизирует управление и работу с договорами. С ее помощью систематизируется и упрощается подготовка, согласование, исполнение и хранение договоров. Появляется возможность контролировать и анализировать договорную деятельность и работу с контрагентами во всех аспектах, избежать издержек, обусловленных несистемной работой, разрозненным хранением документов и отсутствием актуальной информации [7].

ИС обеспечивает быстроту исполнения операций и ограждает от совершения ошибок и несанкционированных действий. Разные группы пользователей найдут в системе решение специфических для них задач. Наиболее тесно связаны

с договорной деятельностью такие структуры, как дирекция, финансовая служба, отдел продаж, юридическая служба.

Можно выделить следующие преимущества системы: 1) использование современной и широко распространенной учетной системы в качестве платформы программы позволяет избежать обособленности системы учета договоров и разрыва в корпоративной информационной системе; 2) возможность организации удаленного доступа к данным через веб-интерфейс существенно снижает затраты предприятия на эксплуатацию системы.

Программа предусматривает не только регистрацию и хранение договоров, но и управление процессами их подготовки, согласования и исполнения. В дополнение к этому в системе реализован CRM-блок, позволяющий оптимизировать работу с клиентами.

Комфорт при работе с договорами обеспечивается удобным графическим пользовательским интерфейсом. Все данные разделены на логические группы, поиск нужного договора, его свойства или экономического показателя не представляет труда. Каждый пользователь получает только необходимую для него информацию.

Для работы с системой не нужно получать специфических знаний, достаточно владеть навыками работы с персональным компьютером и прочитать руководство пользователя.

Технологически программа реализована на платформе «1С: Предприятие 8.0». Существует возможность интеграции и обмена данными с другими win-приложениями.

Программа может быть использована как на небольших, так и на крупных предприятиях без ограничения отраслевой специфики.

3.1.3 ИС «Респект: Учет договоров»

ИС «Респект: Учет договоров» от ООО «РеспектСофт Прикладные Решения» выполнена на платформе 1С [8].

Основные возможности ИС «Респект: Учет договоров» перечислены ниже (рис. 3.3–3.5):

1. Подготовка и согласование договоров.
2. Формирование бланков документов.
3. Ведение журнала договоров предприятия.
4. Планирование и контроль исполнения договоров.
5. Формирование универсальных отчетов.
6. Хранение договоров.
7. Интеграция в 1С:Бухгалтерию 8.
8. Интеграция в 1С:Управление торговлей 8.
9. Интеграция в 1С:УПП 8.
10. Интеграция в 1С:Бухгалтерию 7.7.
11. Совместима с 1С:Бухгалтерией 3.0.

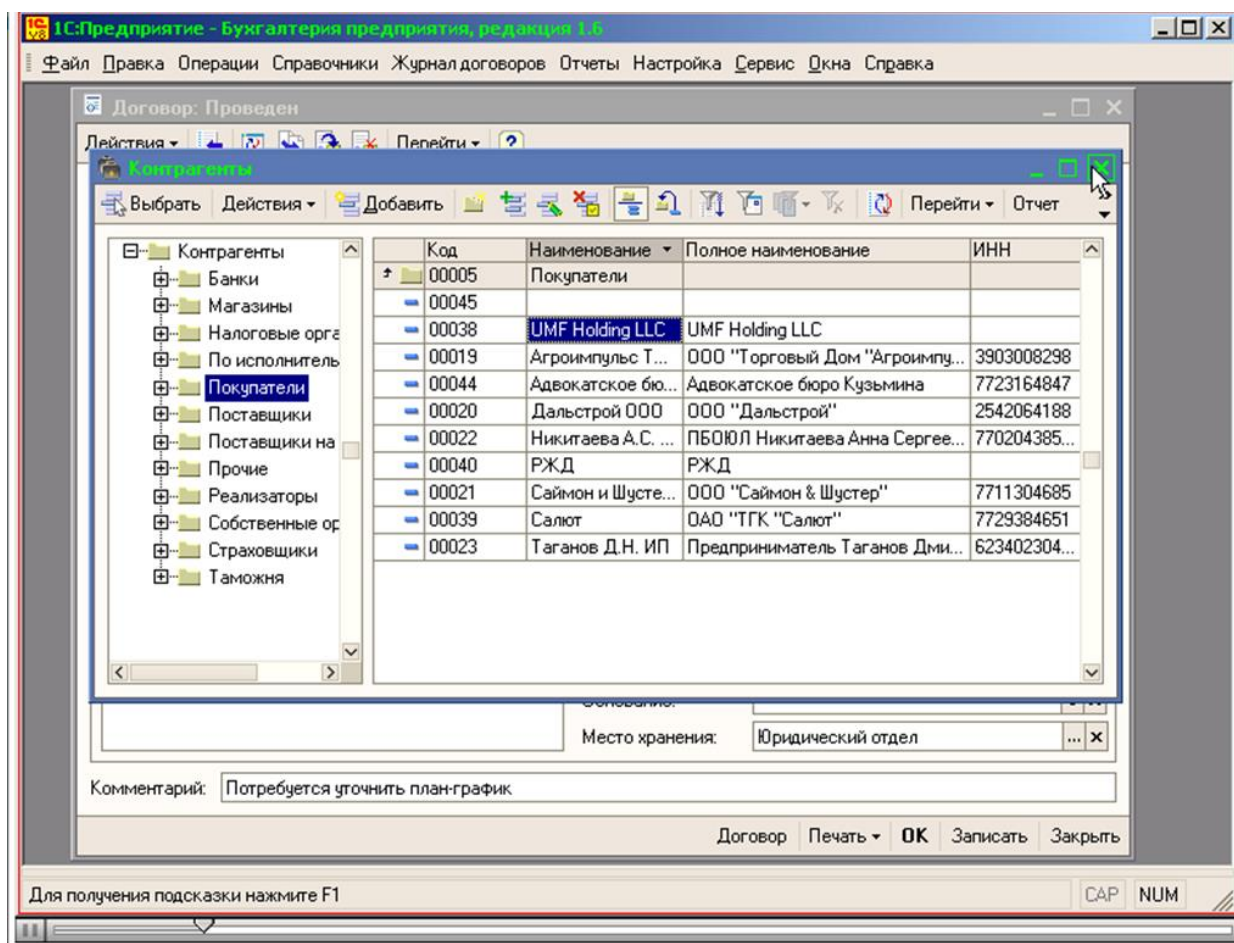


Рисунок 3.3 – Пример интерфейса ИС «Респект: Учет договоров»
(Контрагенты)

ИС «Респект: Учет договоров» предназначена для следующих подразделений [8]:

- руководителей предприятия;
- финансовой службы и бухгалтерии;
- производства;
- договорного отдела;
- юридической службы;
- отдела продаж и снабжения;
- службы ИТ.

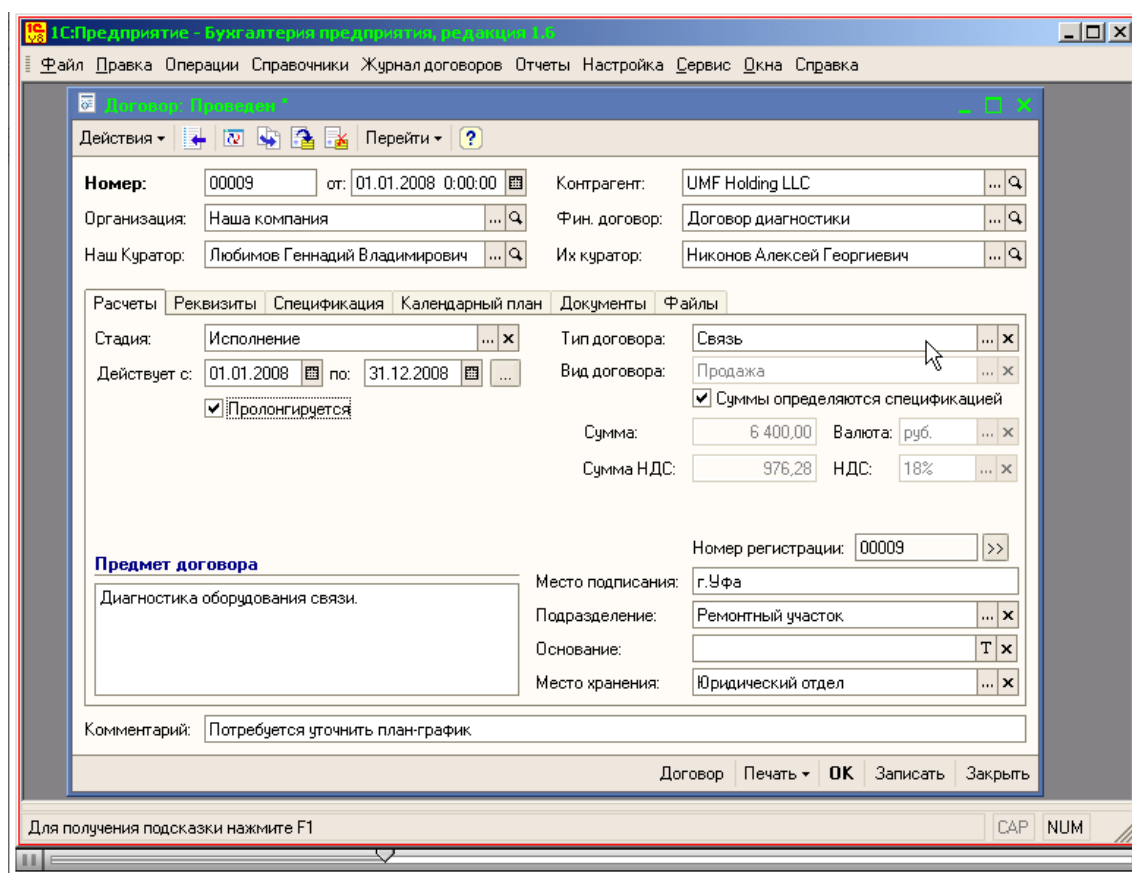


Рисунок 3.4 – Пример интерфейса ИС «Респект: Учет договоров»
(Договор проведен)

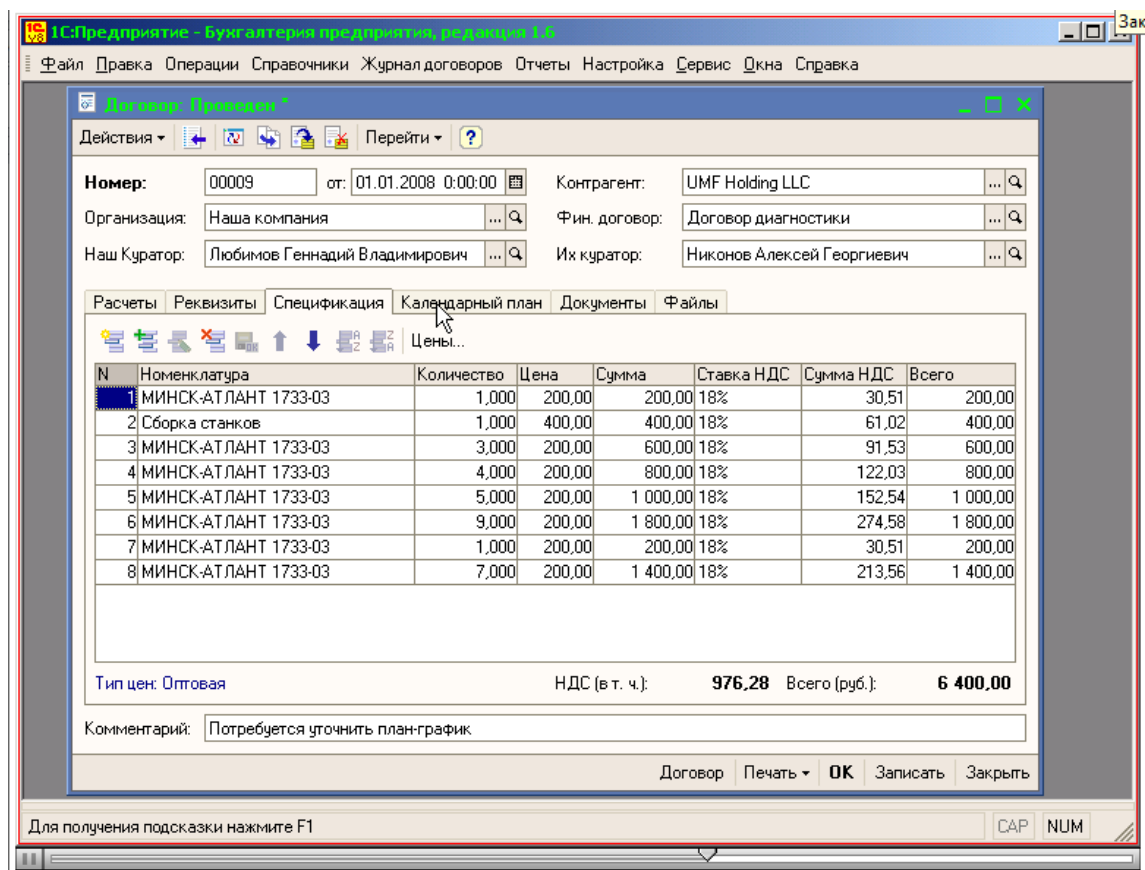


Рисунок 3.5 – Пример интерфейса ИС «Респект: Учет договоров»
(Спецификация)

ИС «Респект: Учет договоров» от ООО «РеспектСофт Прикладные Решения» выполнена на платформе 1С, что ограничивает многие предприятия в ее использовании.

3.1.4 «Система управления договорами»

«Система управления договорами» в СИТЕСК ECOS. Система электронного документооборота на Alfresco ECM написана на Java с использованием Spring Framework и работает под управлением сервера приложений (такого, как Apache Tomcat или JBoss) в операционных системах Windows, Linux или MacOS. Решение для учета договоров, контроля поручений, создания счетов и других сопроводительных документов, автоматизации процесса согласования [9].

Функциональные возможности «Системы управления договорами»:

1. Реестр договоров.

Единый реестр договоров формируется из всех данных, созданных или загруженных в систему.

Для каждого из типов документов (договоры, дополнительные соглашения, платежные поручения и др.) в системе создаются журналы.

Журнальная структура позволяет наглядно отображать весь перечень документов с датой, статусом, наименованием контрагента, суммой и другими настроенными полями. Это удобно и сокращает время на поиск необходимого документа или выборочной информации.

В журнале введена фильтрация договоров согласно выбранному критерию (юрлицо, валюта, предмет договора).

2. Контроль срока действия.

С помощью «Системы управления договорами» можно автоматизировать процесс контроля сроков действия договоров и их подписание.

Система сама уведомит контрагента о новом статусе документа. Исполнителю на e-mail поступит напоминание о необходимости продлить или расторгнуть договор, чтобы не возникло никаких нарушений.

3. Контроль исполнения договоров.

Система поможет отследить результат исполнения договоров и автоматизировать весь процесс.

Исполнение заданий по договору можно разбить на несколько этапов и система сама будет автоматически создавать счета и закрывающие документы в конце каждого цикла.

Для ежемесячных договоров можно настроить автоматический контроль – система «Управление договорами» напомнит «исполнителю» о необходимых действиях: подписании или продлении документа, отправив уведомление на электронную почту (рис. 3.6).

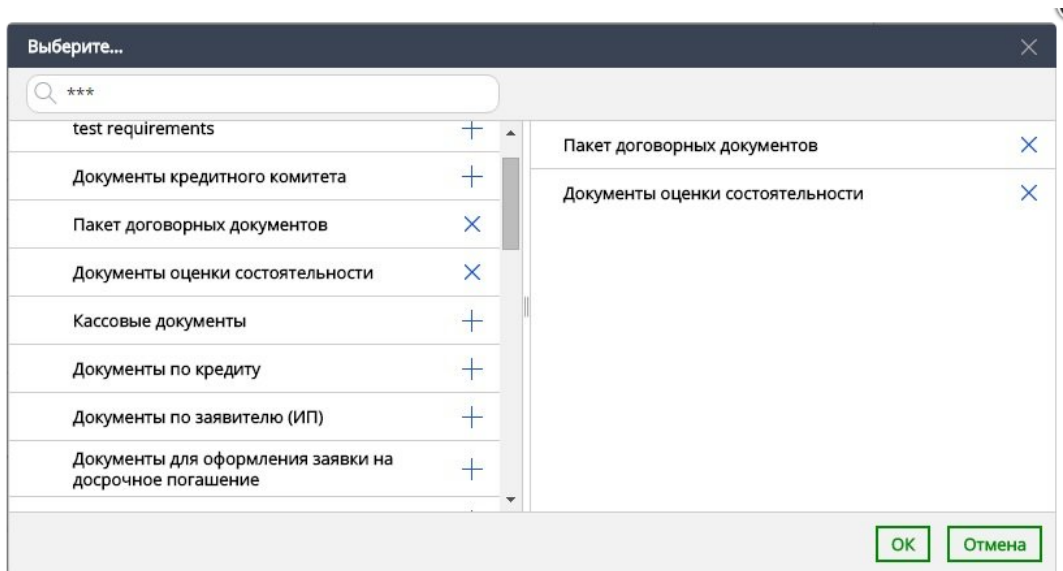


Рисунок 3.6 – Пример интерфейса «Системы управления договорами»
(Контроль исполнения договоров)

4. Справочник продуктов и услуг. Генерация счетов и актов.

Система поможет сформировать сопровождающие документы: счета, акты, накладные. Это происходит автоматически, по заданным шаблонам, которые настраиваются при настройке системы. Стандартизация процесса позволит генерировать счета и закрывающие документы, подставляя выбранные позиции из справочника (рис. 3.7).

▼ ПРОДУКТЫ И УСЛУГИ					
Номер	Название	Цена за единицу	Количество	Всего	Ед. измерения
1	Компьютер в сборе	20000	5	100000	шт
2	Монитор	12500	3	37500	шт

Рисунок 3.7 – Пример интерфейса «Системы управления договорами»
(Генерация счетов и актов)

5. Быстрый поиск договоров, счетов и закрывающих документов.

В системе находится поисковой механизм.

Вы можете найти нужный договор, используя регистрационные данные карточки или фрагмент текста/страницы.

6. Гибкий процесс согласования договоров.

Можно не только откорректировать раздел «Управление договорами» для формализованных документов, но и автоматизировать процесс согласования или подписания для неформализованных договоров.

Регламентные договоры можно согласовывать по заранее разработанному сценарию – с возможностью контроля сроков и действий участников процесса на каждом этапе.

Есть возможность корректировать сроки контроля, списки ответственных лиц или запуск дополнительного согласования.

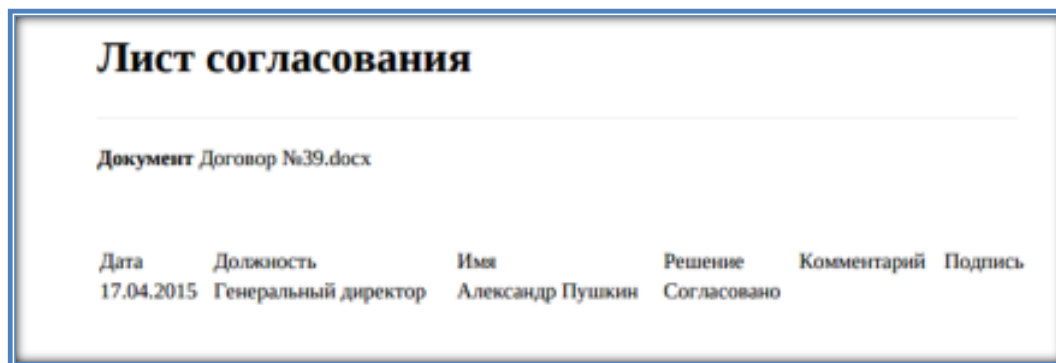
Предусмотрена опция согласования актов и счетов на оплату. Это полезно для минимизации рисков, связанных с неисполнением или ненадлежащим исполнением договорных обязательств в части оплат.

Любой пользователь, согласно заданным правам, может создавать платежные документы, редактировать, согласовывать, отслеживать ход выполнения работы, контролировать проведение оплаты по договорам.

7. Подключение контрагента к процессу согласования договора.

Можно подключить к работе с договором контрагента непосредственно из самой системы, в этом случае редактирование текста будет осуществляться в онлайн-режиме, с сохранением всех версий документа. Действия контрагента в системе регламентируются согласно заданным правам.

8. Печать листа согласования происходит по заранее утвержденной форме (рис. 3.8).



Дата	Должность	Имя	Решение	Комментарий	Подпись
17.04.2015	Генеральный директор	Александр Пушкин	Согласовано		

Рисунок 3.8 – Пример интерфейса «Системы управления договорами»
(Лист согласования)

9. Создание договоров по шаблонам.

Для создания и подготовки типовых договоров в системе используются шаблоны, которые позволяют создать документ в формате MS Word или PDF и подставлять в него данные из карточки договора (данные контрагента, подписанта, реквизиты компании).

Возможна комплексная автоматизация бизнес-процессов компании, а также «частные» решения, основанные на внедрении специализированного программного обеспечения в деятельность конкретного отдела (отдел закупок, бухгалтерия, отдел логистики, служба кадров и т. д.).

10. Контроль прав доступа к договорам.

Имеется возможность настроить права доступа каждому пользователю системы, в зависимости от должности и роли пользователя: по видам договорных документов и по иерархии в рамках процесса согласования.

11. Стоимость.

Стоимость модуля «Управление договорами» зависит от количества пользователей и модели использования: лицензионная или облачная.

Лицензионная версия:

- от 53 000 руб. (единоразово), минимальный пакет до 30 пользователей.

Облачная версия:

- от 19 900 руб. в месяц, минимальный пакет до 60 пользователей.

В пакет также входит модуль «Электронный архив».

Citeck ECOS – это единая бизнес-платформа, на базе которой можно развернуть любое количество модульных решений. Все зависит от потребностей бизнеса, готовности настроить эффективную модель управления документооборотом и бизнес-процессов компании.

Основной недостаток данной ИС – высокая цена.

Таким образом, было принято решение разработать собственную информационную систему для учета и контроля договорных отношений на предприятии.

3.2 Обоснование выбора программных средств для реализации проекта

Рассмотрим различные средства реализации, которые могут быть использованы при создании информационной системы учета и контроля договоров в ООО «ЛЕМА групп». Определим, насколько тот или иной язык подходит для изучения и разработки информационной системы.

Visual C++ – высокопрофессиональный язык. Программы, написанные на нем, имеют превосходное быстродействие. В среде программистов C++ распространен очень широко. Но он слишком сложен для восприятия новичком, программы на нем пишутся с трудом и с него лучше не начинать.

Достоинства:

- наличие объектно-ориентированной библиотеки Microsoft Foundation Classes, использование которой существенно облегчает разработку пользовательского интерфейса;
- возможность использования всей физической и виртуальной памяти, что снимает ограничения на алгоритмы обработки данных.

Недостатки:

- отсутствие стандартных библиотек для работы с базами данных;
- введение новых запросов к системе требует привлечения к работе программиста.

Visual Basic. По мнению профессионалов, наиболее легок и прост из упомянутых языков как раз Visual Basic. Профессиональные программисты любят его за то, что для создания на нем приличного продукта требуется гораздо меньше времени, чем, скажем, на C++. Поэтому популярность у него огромная. Конечно, за все приходится платить, и у Visual Basic есть свой недостаток – программы, написанные на нем, не такие быстрые, как на C++, однако новичок почувствует эту разницу очень не скоро.

Embarcadero Delphi XE3 редакции Architect – это довольно мощная среда разработки интерфейса ИС. Особенности:

- 1) есть пробная версия;

2) платформа создания бизнес-приложений FireMonkey в Embarcadero Delphi XE3 позволяет быстро создавать корпоративные и коммерческие интерактивные приложения;

3) имеются встроенные возможности доступа к данным (технология ADO), что является средством связи с выбранным Microsoft Access 2007.

Новая функция Visual LiveBindings в Embarcadero Delphi XE3 беспрецедентно упрощает создание связей и подключений благодаря графической области, на которой можно связывать элементы, просто протягивая линии между ними. Visual LiveBindings также предусматривает управление слоями, которое позволяет упорядочить динамические связи LiveBindings и новые компоненты. Все это невероятно упрощает и ускоряет создание прототипов [10].

Базы данных считаются основным достоинством *Delphi*. Это действительно так. Хотя язык и не создавался специально под эту предметную область программирования, но реализация работы с данными здесь просто поражает. Даже специализированные языки, которые предназначены для работы с базами данных (такие как MS Visual FoxPro), явно уступают Delphi по простоте и мощи программирования [10].

Delphi скрывает все сложности и в то же время предоставляет широчайшие возможности при создании баз данных. Практически любую задачу в этой предметной области можно реализовать средствами этого языка, причем за довольно короткий промежуток времени. Главное здесь то, что реализация приложения очень удобна и проста в понимании [10].

Достоинства:

- возможность создания .EXE файлов, не требующих присутствия на диске программной среды Delphi;
- легкость разработки гибкого пользовательского интерфейса;
- поддержка основных форматов представления данных;
- высокая скорость обработки данных.

Недостатки:

– для модернизации системы требуется наличие на диске программной среды Delphi.

Для работы с базами в Delphi есть несколько наборов компонентов. Каждый набор очень хорошо подходит для решения определенного круга задач. Почему такое разнообразие компонентов? Все они используют разные технологии доступа к данным и отличаются по своим возможностям. Microsoft встроила в свои продукты разработки только технологию доступа к данным ADO, собственной разработки. Это является еще одним достоинством. При создании приложения будет использоваться именно эта технология.

ADO (Active Data Objects) – технология доступа к данным, разработанная корпорацией Microsoft. Очень хорошая библиотека, но использовать ее желательно только с базами данных Microsoft, а именно MS Access или MS SQL Server.

Рассмотрим СУБД.

СУБД Oracle – весьма сложная система [5]. Одной из основных характеристик СУБД Oracle является функционирование системы на большинстве платформ, и в том числе на больших ЭВМ, UNIX-серверах, персональных компьютерах и т. д. Другой важной характеристикой является поддержка Oracle всех возможных вариантов архитектур, в том числе симметричных многопроцессорных систем, кластеров, систем с массовым параллелизмом и т. д. Очевидна значимость этих характеристик для крупномасштабных организаций, где эксплуатируется множество компьютеров различных моделей и производителей. В таких условиях фактором успеха является максимально возможная типизация предлагаемых решений, ставящая своей целью существенное снижение стоимости владения программным обеспечением. Унификация систем управления базами данных – один из наиболее значимых шагов на пути достижения этой цели.

Поддержка Oracle большинства популярных компьютерных платформ и архитектур достигается за счет жесткой технологической схемы разработки кода

СУБД. Разработку серверных продуктов выполняет единое подразделение корпорации Oracle, изменения вносятся централизованно, после этого все версии подвергаются тщательному тестированию в базовом варианте, а затем переносятся на все платформы, где также детально проверяются.

SQL Server – это реляционная СУБД, которая отвечает за поддержку структуры базы данных и решает следующие задачи:

- поддерживает связи между данными в базе;
- гарантирует корректное хранение данных и выполнение правил, регламентирующих связи между ними;
- восстанавливает данные после аварии системы, переводя их в согласованное состояние, зафиксированное до сбоя.

Microsoft Access – это полнофункциональная реляционная СУБД.

По существу, Microsoft Access является лишь одним из стратегических направлений корпорации Microsoft в области обработки данных. Подобно всем системам управления реляционными базами данных, Access позволяет легко объединять связанную информацию [11].

Microsoft Access имеет понятный интерфейс приложения. Удобные инструменты и наличие множества мастеров позволяют создать базы данных даже не очень опытным пользователям. К числу наиболее мощных средств Microsoft Access относятся мастера, которые можно использовать для создания таблиц, запросов, различных типов форм и отчетов.

Преимущества Microsoft Access:

1. Microsoft Access имеет стандартный интерфейс пакета Microsoft Office Professional, что позволит сократить время на разработку базы данных и ускорит процесс создания информационной системы.

2. С помощью Microsoft Access можно создавать приложения, работающие в среде Windows XP Professional.

3. Используя запросы, можно отбирать нужные данные и вычислять итоги.

4. Можно проводить анализ структуры базы данных, импорт таблиц и текстовых данных.

5. Существует возможность работы приложений, созданных Microsoft Access, с локальной сетью, а также на удаленном компьютере, подключающемся к локальной сети через Интернет или модем.

6. Все данные вместе с запросами и формами физически хранятся в одном файле (файле с расширением MDB).

7. Широкие возможности импорта и экспорта данных из различных СУБД, присоединение данных.

8. Любой пользователь может освоить и даже доработать БД, добавить новые формы, связи, таблицы, создать новые запросы.

Таким образом, было принято решение о выборе среды разработки ИС учета и контроля договоров в Embarcadero Delphi XE3 и СУБД Microsoft Access.

3.3 Требования, предъявляемые к БД в ИС

База данных есть динамическая информационная модель предметной области, адекватно отражающая ее состояние в любой момент времени [12].

База данных не может существовать вне некоторой совокупности средств поддержки. Она является информационным ядром человеко-машинной системы – системы базы данных (СБД) [13–16].

Цель этой системы состоит в накоплении и хранении сведений о ПО и предоставлении их заинтересованным пользователям по их требованиям в виде, пригодном для решения их задач. Система предоставляет доступ к сведениям одновременно и независимо многим пользователям, создавая для каждого из них иллюзию индивидуальной работы.

Система базы данных – это система специальным образом организованных данных, программных, технических, языковых и организационно-методических средств, предназначенная для поддержания динамической информационной модели ПО и коллективного многоцелевого использования данных.

Требования к БД [13]:

1. *Принцип интегрированного хранения информации.*

На предприятии должен поддерживаться только общий архив, в который поступают все сведения по мере их появления. Любой сотрудник получает нужную ему информацию только из этого архива.

2. *Целостность данных.*

Целостности данных – это обусловленные требованиями конкретной ПО правила, соблюдение которых обеспечивает интерпретируемость хранимых данных. Выше мы определили базу данных как модель некоторой части реального мира. Она отражает реальное состояние ПО в любой фиксированный момент времени в виде текущей конфигурации хранимых данных. Очевидны два основных требования к этой конфигурации:

- любое хранимое в БД значение любого семантически значимого атрибута в любой момент времени должно быть истинным значением характеристики соответствующего объекта ПО;
- состояние БД в любой момент времени должно иметь осмысленную интерпретацию в терминах ПО. Текущее состояние БД является целостным, если возможна осмысленная интерпретация его в терминах ПО.

3. *Нормализация отношений.*

Цель нормализации – преобразовать универсальное отношение в систему отношений, удовлетворяющих определенным формальным требованиям. При обновлении данных, хранящихся в такой системе отношений, не возникают аномальные ситуации.

Требования к базовым отношениям и алгоритмы нормализации без потерь информации составляют предмет математической теории нормальных форм отношений:

- первая нормальная форма (1НФ): отношение находится в первой нормальной форме, если и только если все домены его атрибутов содержат скалярные значения;

- вторая нормальная форма (2НФ): отношение находится во второй нормальной форме, если и только если каждый его неключевой атрибут неприводимо зависит от первичного ключа;
- третья нормальная форма (3НФ): отношение находится в третьей нормальной форме, если и только если оно находится в 2НФ и нет транзитивных зависимостей.

Можно дать альтернативное определение 3НФ, используя понятия неприводимой ФЗ и взаимной независимости атрибутов: отношение находится в третьей нормальной форме, если и только если каждый его неключевой атрибут неприводимо зависит только от первичного ключа и все неключевые атрибуты взаимно независимы.

Второй шаг процедуры нормализации направлен на устранение транзитивных зависимостей. Он, как и первый шаг, всегда может быть выполнен без потерь информации.

Определения 1НФ и 2НФ предполагают, что отношение имеет единственный потенциальный ключ.

Для того чтобы в БД не было аномалий обновления данных, следует хранить данные не в одном отношении, а в нескольких взаимосвязанных. Эти отношения должны быть проекциями универсального отношения, находящимися по крайней мере в 3НФ. Естественное соединение всех проекций должно восстанавливать универсальное отношение без потерь информации [16].

4 Заключение

В процессе выполнения задания по контрольной работе по дисциплине «НИР в семестре» была проведена следующая работа:

1) изучены особенности ведения договорных отношений в ООО «ЛЕМА групп»;

2) произведен анализ аналогов учета договоров («Ведение договоров» 4.4, ИС «АстроСофт: Учет договоров», ИС «Респект: Учет договоров», «Система управления договорами»). Выявлены их достоинства и недостатки. Было принято решение о разработке собственной ИС;

3) проведен анализ возможных сред реализации необходимой ИС;

4) выбрано средство реализации ИС – MS Access.

Поставленные цели и задачи в работе были выполнены.

При подготовке отчета по контрольной работе использовалась методическая литература, специальная литература предметной области, литература по проектированию информационных систем.

Отчет по дисциплине подготовлен в соответствии с образовательным стандартом ТУСУР [17].

Цели работы достигнуты, все поставленные задачи выполнены.

В дальнейшем предполагается продолжить проектирование и разработку информационной системы по учету и контролю договоров на предприятии ООО «ЛЕМА групп».

Список использованных источников

1. Фаронов, В. В. Программирование баз данных в Delphi 7 : учеб. курс / В. В. Фаронов. – СПб. : Питер, 2006. – 459 с.
2. Устав Общества с ограниченной ответственностью «ЛЕМА групп». Утвержден Протоколом общего собрания учредителей № 1 от 18.04.2018.
3. Лысаков, А. В. Договорные отношения в управлении проектами / А. В. Лысаков, Д. А. Новиков. – М. : ИПУ РАН, 2004. – 100 с.
4. Исакова, А. И. Информационные системы : учеб. пособие / А. И. Исакова. – Томск, 2010. – 132 с.
5. Исакова, А. И. Научная работа 1 : учеб. пособие / А. И. Исакова. – Томск : ТУСУР, 2017. – 141 с.
6. Руководство пользователя программы «Ведение договоров» [Электронный ресурс] // Официальный сайт компании AraxGroup. – Режим доступа: <http://www.araxgroup.ru/opdog/Glava2/vozmozn.htm> (дата обращения: 01.06.2020).
7. ИС «АстроСофт: Учет договоров» [Электронный ресурс] // Официальный сайт фирмы «АстроСофт». – Режим доступа: <http://www.astrosoft.ru/> (дата обращения: 02.06.2020).
8. ИС «Респект: Учет договоров» [Электронный ресурс] // Официальный сайт фирмы ООО «РеспектСофт Прикладные Решения». – Режим доступа: <https://dogovorun.ru/product> (дата обращения: 02.05.2020).
9. Система управления договорами [Электронный ресурс] // Официальный сайт разработчика СІТЕСК. – Режим доступа: <https://www.citeck.ru/> (дата обращения: 02.06.2020).
10. Delphi XE3 [Электронный ресурс]. – Режим доступа: <http://www.embarcadero.com/ru/products/delphi> (дата обращения: 03.06.2020).
11. 10 основных причин для использования Access в Excel [Электронный ресурс] // Сайт «Поддержка Microsoft». – Режим доступа: <https://support.microsoft.com/ru-ru/office/10-основных-причин-для-использования-access-в-excel-2a454445-13cc-4b39-bc2f-d27fd12ca414> (дата обращения: 03.06.2020).
12. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 4-е изд., стереотип. – М. : Академия, 2010. – 320 с.
13. Сибилев, В. Д. Базы данных : учеб. пособие / В. Д. Сибилев. – Томск : ТУСУР, 2007. – 279 с.

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

15. Давыдова, Е. М. Базы данных : учеб. пособие / Е. М. Давыдова, Н. А. Новгородова. – 3-е изд., перераб. и доп. – Томск : В-Спектр, 2012. – 128 с.

16. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 5-е изд., испр. – М. : Академия, 2012. – 320 с.

17. Образовательный стандарт вуза ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления [Электронный ресурс]. – Томск : ТУСУР, 2013. – 57 с. – Режим доступа: <https://regulations.tusur.ru/documents/70> (дата обращения: 27.03.2020).

Приложение Б
(справочное)
Пример итогового отчета

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

**ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ИНФОРМАЦИОННОЙ
СИСТЕМЫ УЧЕТА ДОГОВОРОВ ООО «ЛЕМА ГРУПП» г. МОСКВЫ**

Итоговый отчет по дисциплине
«Научно-исследовательская работа в семестре»

Студент гр. ____
____ И. О. Фамилия
« ____ » _____ 20__ г.

Руководитель
доцент каф. АСУ,
канд. техн. наук
____ А. И. Исакова
« ____ » _____ 20__ г.

Томск 20__

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

Кафедра автоматизированных систем управления (АСУ)

ЗАДАНИЕ

на выполнение итогового отчета по дисциплине
«Научно-исследовательская работа в семестре»:
студенту гр. _____ факультета

(Ф. И. О. студента)

1. **Тема НИР:**

Проектирование базы данных информационной системы учета договоров ООО «ЛЕМА групп» г. Москвы.

2. **Цель НИР:**

Исследовать уровень автоматизации учета договоров с целью выявления бизнес-процесса, требующего автоматизации и приступить к проектированию БД создаваемой ИС.

3. **Задачи НИР:**

- выявление (выделение) бизнес-процесса (вида деятельности), требующего автоматизации (доработки, модернизации и т. п.);
- разработка SADT-модели «As-Is»;
- формализованная постановка задачи на автоматизацию (пользователь, функции, входы, выходы);
- разработка структуры базы данных логического и физического уровня;
- концептуальное проектирование ER-, KB-, FA-моделей;
- разработка глоссария модели, определение доменов;
- разработка правил целостности данных;
- разработка делового регламента;
- определение транзакций пользователя.

4. Исходные данные для НИР:

- документы профильной организации (предприятия), необходимые для решения требуемых задач;
- результаты исследований, полученные при изучении дисциплины «Учебно-исследовательская работа».

5. Технические требования к отчету по НИР: обязательно соблюдение образовательного стандарта ТУСУР 01–2013.

Дата выдачи: «__» _____ 20__ г.

Руководитель НИР от университета

Доцент каф. АСУ, к.т.н. _____

(должность)

(подпись)

А. И. Исакова _____

(Ф. И. О.)

Студент гр. _____

(подпись)

(Ф. И. О.)

Оглавление

1 Введение.....	127
2 Краткое описание предприятия ООО «ЛЕМА групп».....	129
3 Проектирование базы данных информационной системы	133
3.1 Описание бизнес-процесса, требующего автоматизации	133
3.2 Формализованная постановка задачи.....	136
3.3 Логический уровень модели с использованием методологии IDEF1X	137
3.4 Этапы построения информационной модели средствами ERWIN	137
3.5 Методология информационного моделирования IDEF1X (INTEGRATED DEFINITIONS 1 EXPANDED)	138
3.6 ER-модель «сущность – связь»	139
3.7 КВ – модель данных, основанная на ключах	140
3.8 FA – полная атрибутивная модель	142
3.9 Физическая модель БД.....	142
3.10 Глоссарий модели.....	144
3.11 Определение доменов	148
3.12 Правила целостности данных	149
3.13 Деловой регламент	149
3.14 Транзакции пользователя	151
4 Заключение.....	152
Список использованных источников	153

1 Введение

Основным заданием во время выполнения итогового задания дисциплины «НИР в семестре» является изучение бизнес-процессов деятельности ООО «ЛЕМА групп» и их автоматизации, их информационного обеспечения [1].

Целью работы является знакомство с деятельностью предприятия, исследование уровня ее автоматизации с целью выявления бизнес-процесса, требующего автоматизации.

Задачи данной работы:

- знакомство с предприятием, уровнем автоматизации деятельности;
- выявление (выделение) бизнес-процесса (вида деятельности), требующего автоматизации (доработки, модернизации и т. п.);
- разработка SADT-моделей «As-Is», «As-To-Be»;
- формализованная постановка задачи на автоматизацию (пользователь, функции, входы, выходы);
- разработка концептуальных моделей БД (ER-, KB-, FA-уровней) информационной системы учета договоров на предприятии ООО «ЛЕМА групп» (г. Москва);
- разработка структуры базы данных логического и физического уровня;
- разработка глоссария модели, определение доменов;
- разработка правил целостности данных;
- разработка делового регламента;
- определение транзакций пользователя.

Объектом исследования в данной работе является организация, занятая в области строительства и оптовой торговли – ООО «ЛЕМА групп» (г. Москва) [2]. Основной задачей предприятия является обеспечение предпроектных разработок, проектирования и строительства зданий и сооружений, выполнение функций заказчика-застройщика, генерального подрядчика, а также осуществление агентской деятельности в области оптовой торговли.

Предметом исследования является автоматизация учета договоров на предприятии ООО «ЛЕМА групп».

Во время исследования деятельности предприятия применялась методология структурного анализа и проектирования SADT (*Structured Analysis and Design Technique*) и методология проектирование структуры данных в нотации IDEF1x.

Руководство ООО «ЛЕМА групп» заинтересовано в автоматизации учета и контроля договоров, согласно которым осуществляется хозяйственная деятельность предприятия. Данная автоматизация была бы удобна для службы заказчика-застройщика, бухгалтерии, юридической службы, менеджеров проектов, облегчила бы контроль исполнения ключевых обязательств договоров по выполнению работ/услуг и их оплаты.

2 Краткое описание предприятия ООО «ЛЕМА групп»

В настоящее время предприятие ООО «ЛЕМА групп» (г. Москва) входит в состав торгово-строительного холдинга, обеспечивая выполнение следующих функций [2]:

- деятельность заказчика-застройщика, генерального подрядчика;
- деятельность агента по оптовой торговле топливом, рудами, металлами и химическими веществами;
- обеспечение инженерных изысканий, инженерно-технического проектирования, управления проектами строительства, выполнения строительного контроля и авторского надзора, предоставление технических консультаций в этих областях.

Также организация выполняет и другие работы:

- разработка проектов тепло-, водо-, газоснабжения;
- управление эксплуатацией нежилого фонда;
- аренда и лизинг строительных машин и оборудования.

Компания работает в составе холдинга с 2018 г., активно развивается и увеличивает количество проектов, как собственных, так и обслуживаемых в порядке оказания услуг. В составе проектов можно выделить реновацию жилого фонда в порядке реализации государственно-частного партнерства, обеспечение различных этапов строительства в рамках генерального подряда, обеспечение эксплуатации жилого фонда. Также компания обеспечивает проведение торговых операций по поставке ультрадисперсного медного порошка и нефтепродуктов.

Отношения организации с контрагентами подпадают под понятие сделки, т. е. классифицируются как действия, направленные на «установление, изменение или прекращение гражданских прав и обязанностей» [1]. Действующее законодательство РФ обязывает совершать сделки в письменной форме, если они совершаются юридическими лицами между собой или с гражданами [1].

Одна сделка длительного характера может породить множество договоров, направленных на ее исполнение. Например, договор подряда, генерального

подряда, на выполнение функций заказчика и или иной подобный договор влекут за собой заключение ряда последующих договоров, направленных на выполнение основного. Договор генерального подряда для своей реализации подразумевает заключение ряда договоров подряда или субподряда, договоров на производство рабочей документации, подведение коммуникаций, подготовки строительной площадки и многих других. Помимо основных документов – договоров – каждые вновь возникающие правоотношения сопровождаются приложениями и дополнениями, содержащими отдельные обязательства и/или изменяющие уже действующие обязательства, оформленные ранее.

Подобные комплексные сделки выполняются, как правило, по частям или этапам, имеющим свой срок исполнения, денежную оценку и иногда – порядок оплаты. Во избежание просрочки исполнения обязательств, влекущей общую просрочку выполнения договора и штрафные санкции для виновной стороны, требуется постоянный контроль соблюдения сроков исполнения, платежей, а также действия документов (договоров и приложений).

Учитывая рост числа проектов предприятия, соразмерно увеличивается и количество договоров, подлежащих контролю со стороны сотрудников организации.

По состоянию на январь 2020 г. число собственных проектов ООО «ЛЕМА групп» составило 7, а количество обслуживаемых – 34. В связи с увеличением хозяйственной активности за прошедшие два года увеличился и объем действующих договоров (рис. 2.1).

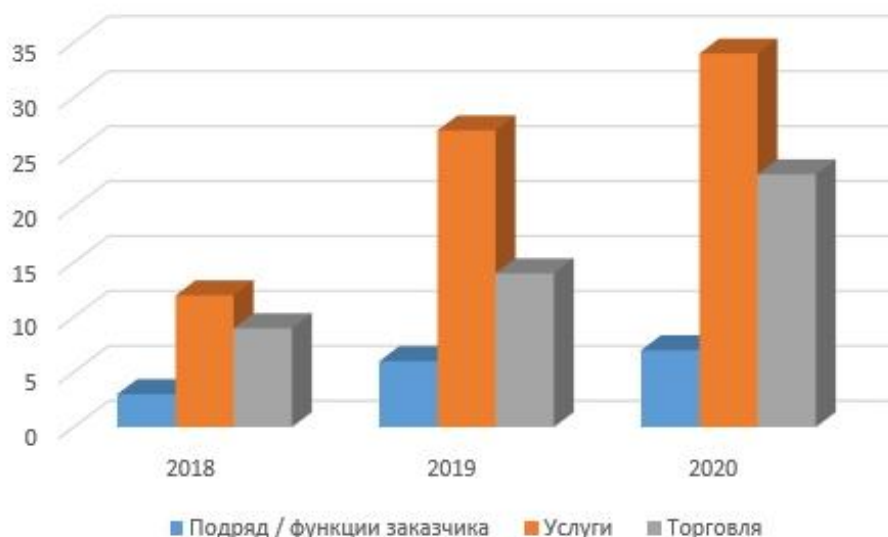


Рисунок 2.1 – Динамика количества договоров

В локальной компьютерной сети ООО «ЛЕМА групп» используется клиент-серверная архитектура.

В отдельной комнате находится сервер, доступ к которому имеет только администратор сети. Сервер обеспечен системой защиты Cisco 1605. Через маршрутизаторы Cisco 2500 информация поступает на сервер отдела информационных технологий и автоматизации типа ML 110 G5 и рабочие станции, установленные у сотрудников ООО «ЛЕМА групп» Dell Precision TM T3500.

Для выхода в сеть Интернет и поддержки связи с удаленными сотрудниками или другими организациями используются модем-роутер D-Link DSL-2500U/BRU/D.

Архитектура аппаратного и программного обеспечения ООО «ЛЕМА групп» имеет четырехуровневую архитектуру.

На первом уровне выполнена организация рабочих мест сотрудников, где установлено локальное программное и аппаратное обеспечение для организации рабочих мест и используется клиент-серверная архитектура.

На втором уровне выполняется поддержка Центров обработки данных, представленных серверами СХД, San и Colocation. Для обеспечения потоков данных по различным бизнес-процессам, планирования деятельности в ООО

«ЛЕМА групп» используются системы управления базами данных (СУБД) и бизнес-приложения.

Третий уровень включает программное и аппаратное обеспечение для обеспечения интеграционных решений, IP-телефонию, беспроводной Интернет, Wi-Fi, средства для обеспечения видео-конференц-связи и бесперебойной работы локальной сети LAN.

Последний уровень представляет собой систему защиты данных, организованный на уровнях обеспечения физической безопасности, диспетчеризации, пожарной системы и поддержки обеспечивающих систем.

3 Проектирование базы данных информационной системы

3.1 Описание бизнес-процесса, требующего автоматизации

Предприятие заключает различные договоры в связи с осуществлением хозяйственной деятельности. Организация активно работает с договорами подряда, генерального подряда, субподряда, на проектирование, изготовление проектной документации, договорами поставки, купли-продажи и пр. После заключения договора его копия направляется в бухгалтерию, а также в подразделение, отвечающее за реализацию договора. Оригинал договора направляется в юридическую службу.

Сотрудник юридической службы вносит сведения в журнал учета договоров. Вносимые данные [4–5]:

- дата договора;
- номер договора;
- вид/предмет договора;
- срок действия договора;
- стоимость работ по договору;
- проект, к которому относится договор.

После внесения данных в журнал учета копия договора рассылается по подразделениям, ответственным за его выполнение.

Руководитель и/или сотрудник подразделения, ответственного за договор, а также юридической службы контролируют исполнение существенных обязательств.

Бухгалтерия производит сверку с договорами при осуществлении платежей, обращая внимание на сроки и размер при их исполнении.

При завершении исполнения договора последний закрывается актом приема-сдачи, подтверждающим исполнение обязательств. На основании этого документа в журнал учета договоров вносится соответствующая отметка, сам договор сдается в архив, копия – в бухгалтерию.

Для анализа процесса, требующего автоматизации [5], применялась методология SADT – совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области [6, 7]. Разработанная Дугласом Россом во второй половине XX в. методология стала основой, в частности для методологии функционального моделирования IDEF0, графической нотацией, формализующей и описывающей бизнес-процессы [7, 8].

В результате анализа полученных сведений о модернизируемом бизнес-процессе построена SADT-модель «As-Is» учета и контроля договоров уровня А-0 (рис. 3.1) и детализация А0 (рис. 3.2) в графической нотации IDEF0. В качестве инструмента визуализации модели использовался программный продукт BPWin (версия 4.1.4), относящийся к категории CASE-средств верхнего уровня, ориентированных на начальные этапы построения информационной системы и связанных с анализом и планированием.

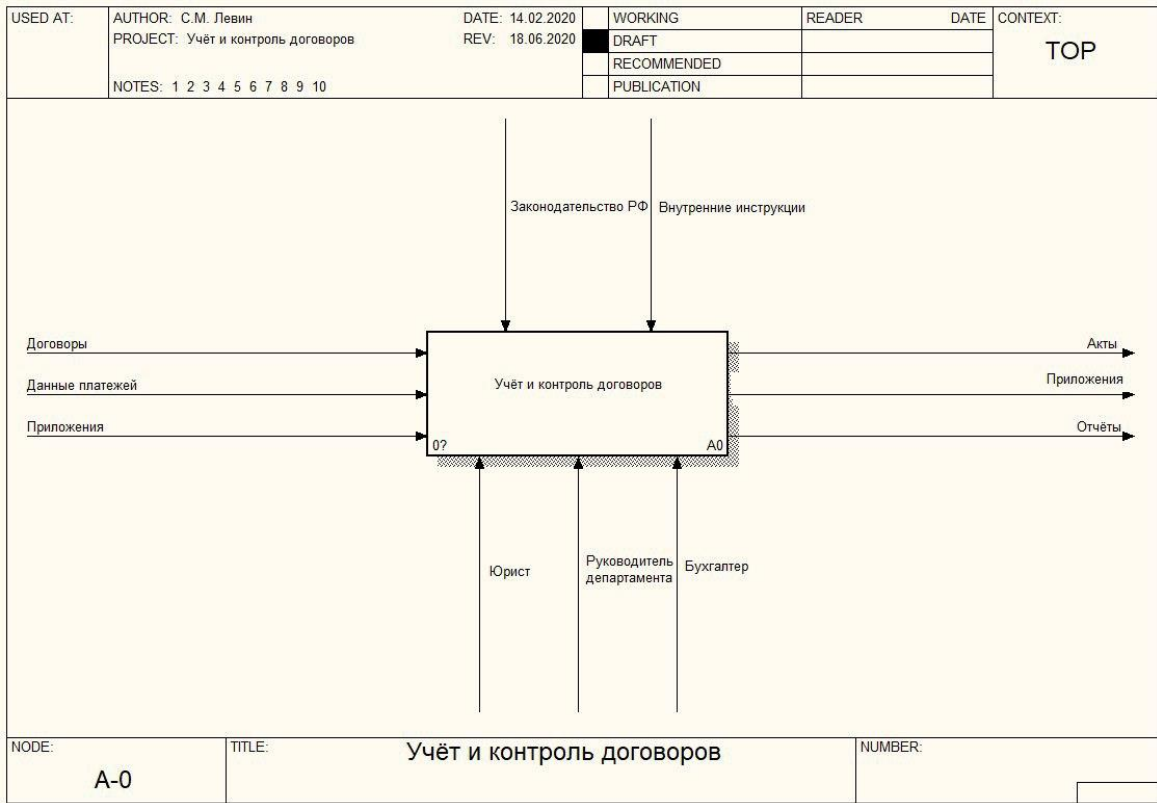


Рисунок 3.1 – SADT-модель «As-Is»

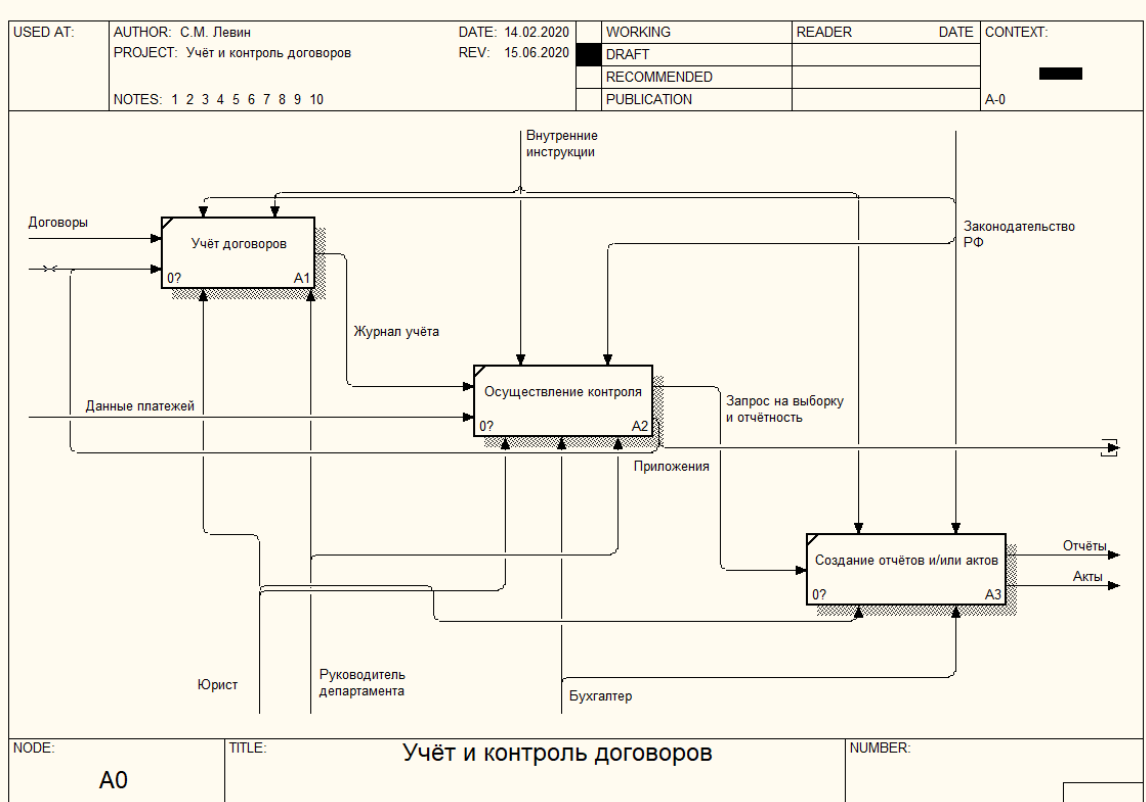


Рисунок 3.2 – Детализация A0

3.2 Формализованная постановка задачи

Необходимо создание информационной системы для учета и контроля исполнения договоров.

Информационная система должна обеспечивать обработку и хранение данных о договорах, их статусе, выполнении существенных обязательств, суммах и сроках. Необходимо предусмотреть систему уведомления об истечении срока какого-либо обязательства или договора в целом.

Автоматизации подлежат следующие действия:

- учет заключенных договоров;
- контроль исполнения сроков договоров или составляющих этапов;
- учет платежей по договорам;
- формирование отчетов и актов.

Подобная система предоставит возможность экономии рабочего времени, уменьшения финансовых потерь и обеспечения учета и контроля своевременного исполнения обязательств, как собственных, так и контрагентов. Иными словами, экономическим результатом автоматизации будет повышение эффективности хозяйственной деятельности предприятия [2].

Входная информация:

- договоры;
- приложения к договорам (дополнительные соглашения к договорам).

Выходная информация:

- отчеты;
- акты;
- приложения.

Пользователями системы будут являться работники организации, занятые в области создания, исполнения договоров, а также лица, выполняющие функции контроля – сотрудники юридической службы и соответствующих департаментов, руководители департаментов, бухгалтер.

3.3 Логический уровень модели с использованием методологии IDEF1X

Для проектирования концептуальной модели БД использовалась методология информационного моделирования IDEF1X (*Integrated DEFinitions 1 eXpanded*) [9, 10].

Методология IDEF1X определяет стандарты терминологии, используемой при информационном моделировании, и графического изображения типовых элементов на диаграммах. IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для построения концептуальной схемы структуры данных предприятия, независимой от конечной реализации базы данных и аппаратной платформы [11].

Логический (первый) уровень модели подразумевает, что мы мыслим в понятиях реального мира и непосредственно из него берем объекты для моделирования. Логический уровень отражает точку зрения пользователя и базируется на прямом отображении фактов из реальной жизни.

Объекты, на которые ссылаются на логическом уровне, получают имена на естественном языке, с использованием любых разделителей слов (пробелов, черточек, запятых и т. п.), которые имеют смысл. Например, *Договор, Заказчик, Исполнитель, Документ, Банк*.

Логическая модель является основой разработки физической модели БД. Термин *логический уровень* в ERwin соответствует концептуальной модели.

ERwin поддерживает построение и организацию работы с этими двумя различными уровнями представления одной модели и позволяет создавать множество вариантов отображения модели на каждом уровне.

3.4 Этапы построения информационной модели средствами ERwin

Построение информационной модели средствами ERwin предполагает определенную последовательность действий [12]:

- 1) определение сущностей;
- 2) определение зависимостей между сущностями;

- 3) задание первичных и альтернативных ключей;
- 4) определение атрибутов сущностей;
- 5) приведение модели к требуемому уровню нормальной формы;
- 6) переход к физическому описанию модели, назначение соответствий:

имя сущности – имя таблицы, атрибут сущности – атрибут таблицы;

- 7) задание процедур и ограничений;
- 8) генерация базы данных.

3.5 Методология информационного моделирования IDEF1X (Integrated DEFinitions 1 eXpanded)

CASE-средство ERwin поддерживает методологию IDEF1X и стандарт *IE (Information engineering)* [9].

Методология IDEF1X определяет стандарты терминологии, используемой при информационном моделировании, и графического изображения типовых элементов на диаграммах. IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для построения концептуальной схемы структуры данных предприятия, независимой от конечной реализации базы данных и аппаратной платформы.

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, является строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели [11].

Методология IDEF1X подразделяется на уровни, соответствующие проектируемой модели данных системы. Каждый такой уровень соответствует определенной фазе проекта.

Верхний уровень состоит из *Entity Relation Diagram (ER, Диаграмма «сущность – связь»)* и *Key-Based model (KB, Модель данных, основанная на ключах)*. Диаграмма «сущность – связь» определяет сущности и их отношения. Модель данных, основанная на ключах, дает более подробное представление данных.

Она включает описание всех сущностей и первичных ключей, которые соответствуют предметной области.

Нижний уровень состоит из *Fully Attributed Model (FA, Полная атрибутивная модель)* содержит всю информацию для реализации проекта, который может быть частью общей информационной системы и описывать предметную область. FA-модель позволяет проектировщикам и администраторам БД представлять, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель данных удовлетворяет требованиям информационной системы. Фактически из этой модели автоматически можно получить модель СУБД, которая является точным отображением системного каталога СУБД.

3.6 ER-модель «сущность – связь»

Модель «сущность – связь» (*Entity Relationship Diagram (ERD)*) является самым высоким уровнем в модели данных. Она определяет набор сущностей, атрибутов и взаимосвязей проектируемой системы. Модель отражает основные бизнес-правила предметной области. ER-модель для проектируемой базы данных приведена на рисунке 3.3.

Модель этого уровня может включать связи «многие-ко-многим» и не включать описание ключей. Целью этой модели является формирование общего взгляда на систему для ее дальнейшей детализации. Обычно *ER-модель* используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Уровень определений – это уровень наименее детального представления информации. Он используется на начальной стадии моделирования.

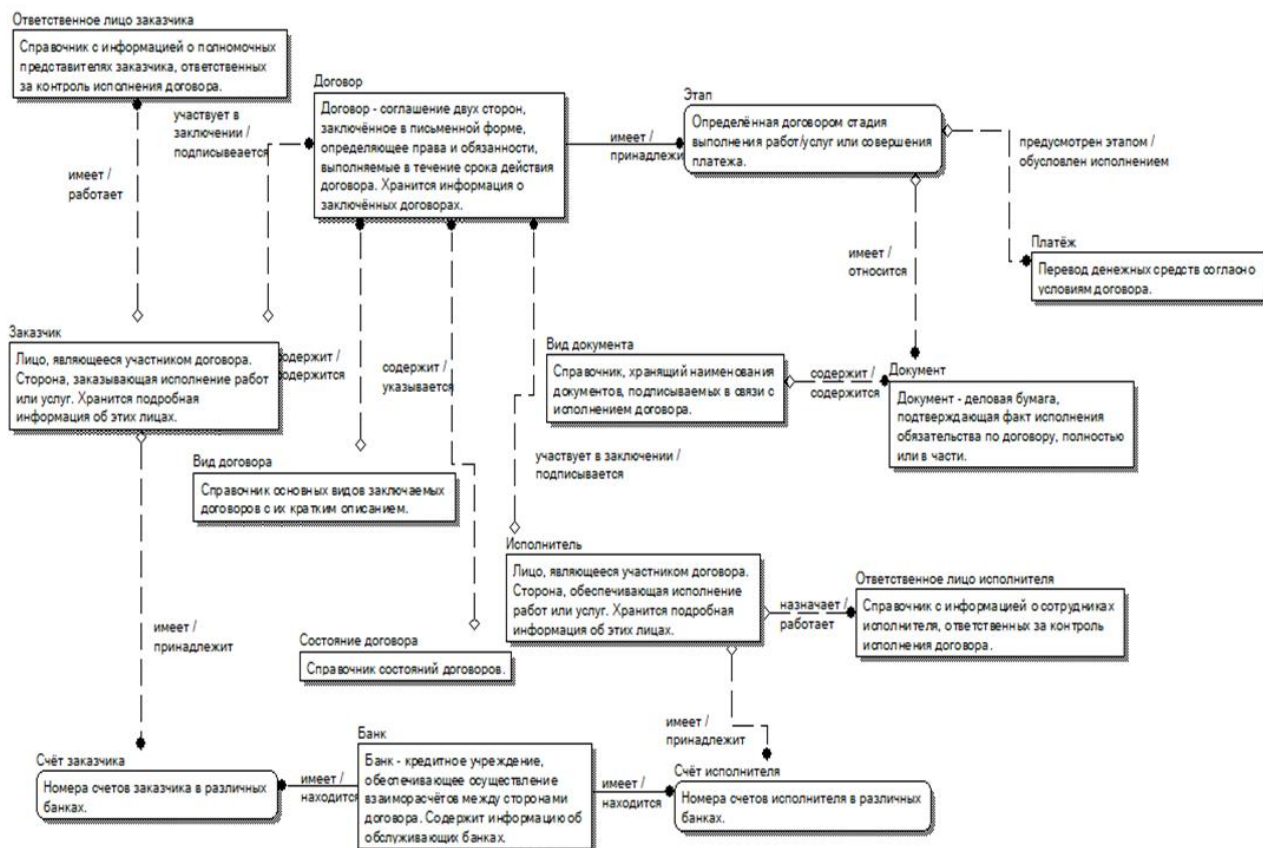


Рисунок 3.3 – Уровень определений (ER-модель)

3.7 КВ – модель данных, основанная на ключах

КВ-модель (*Key Based model (KB)*) описывает структуру данных системы, в которую включены все сущности, атрибуты и первичные ключи. Модель используется для более подробного представления структуры данных и ключей, которые соответствуют предметной области [12]. Целью этой модели является детализация модели «сущность – связь», после чего модель данных может начать реализовываться. Для проектируемой базы данных КВ-уровень изображен на рисунке 3.4.

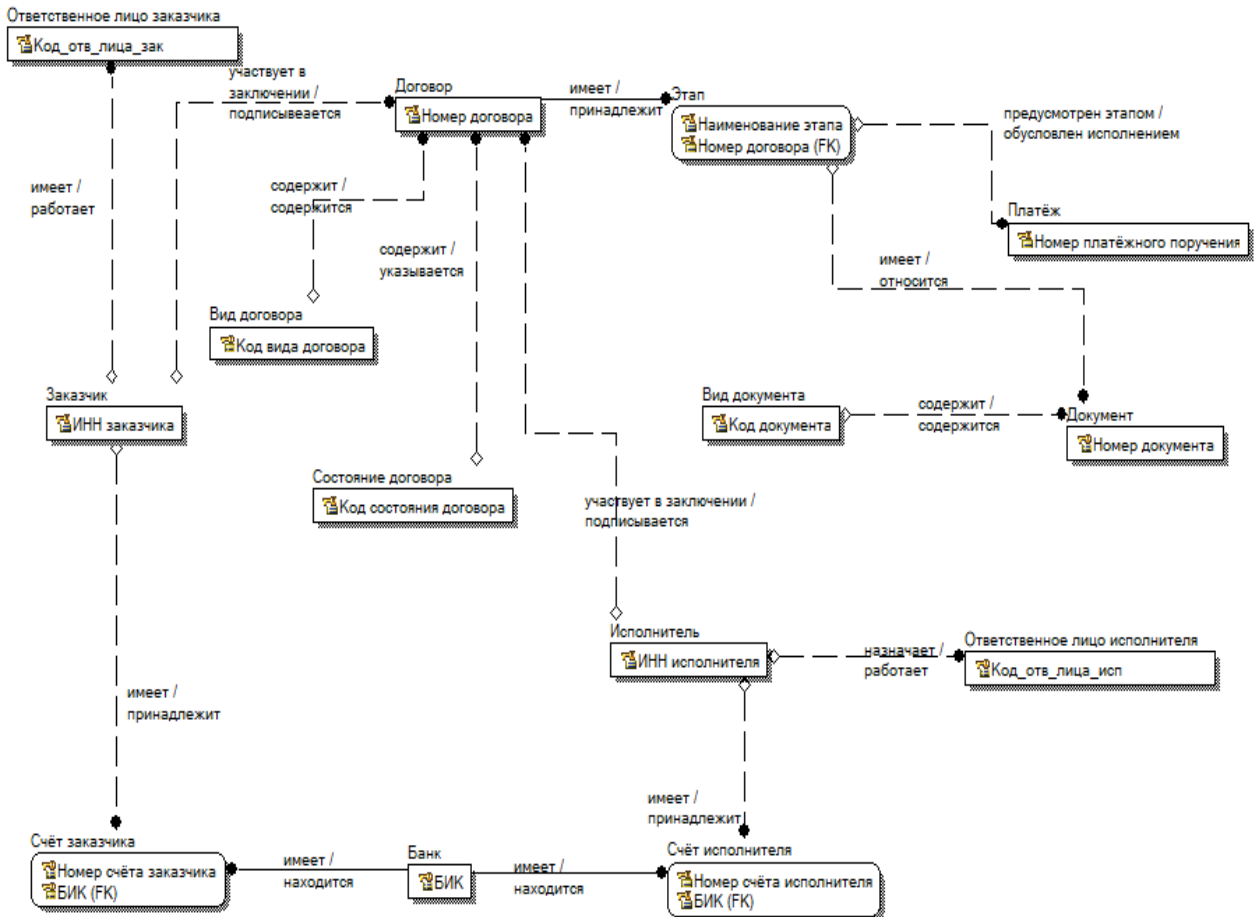


Рисунок 3.4 – КВ-уровень

Определение возможных ключей сущностей (уникальных идентификаторов экземпляров), выделение первичных ключей и специфицирование соединений. Сначала выявляются ключи сущностей, не являющихся потомками в каких-либо соединениях. Первичным ключам этих сущностей будут соответствовать внешние ключи их потомков. Затем определяются возможные ключи потомков и выделяются их первичные ключи. После этого определяются типы соединений. При этом используется правило: «если полный внешний ключ, переданный соединением, входит в состав первичного ключа потомка в этом соединении, то соединение является идентифицирующим, а потомок – (идентификационно) зависимой сущностью. В противном случае соединение неидентифицирующее» [11, 12].

На КВ-уровне в диаграммах отражаются имена первичных и внешних ключей сущностей и спецификации связей.

3.8 FA – полная атрибутивная модель

FA-модель (*Fully Attributed model (FA)*) обеспечивает наиболее детальное представление структуры данных. Полная атрибутивная модель представляет данные в третьей нормальной форме и включает в себя все сущности, атрибуты и связи [10]. FA-диаграмма для базы данных приведена на рисунке 3.5.

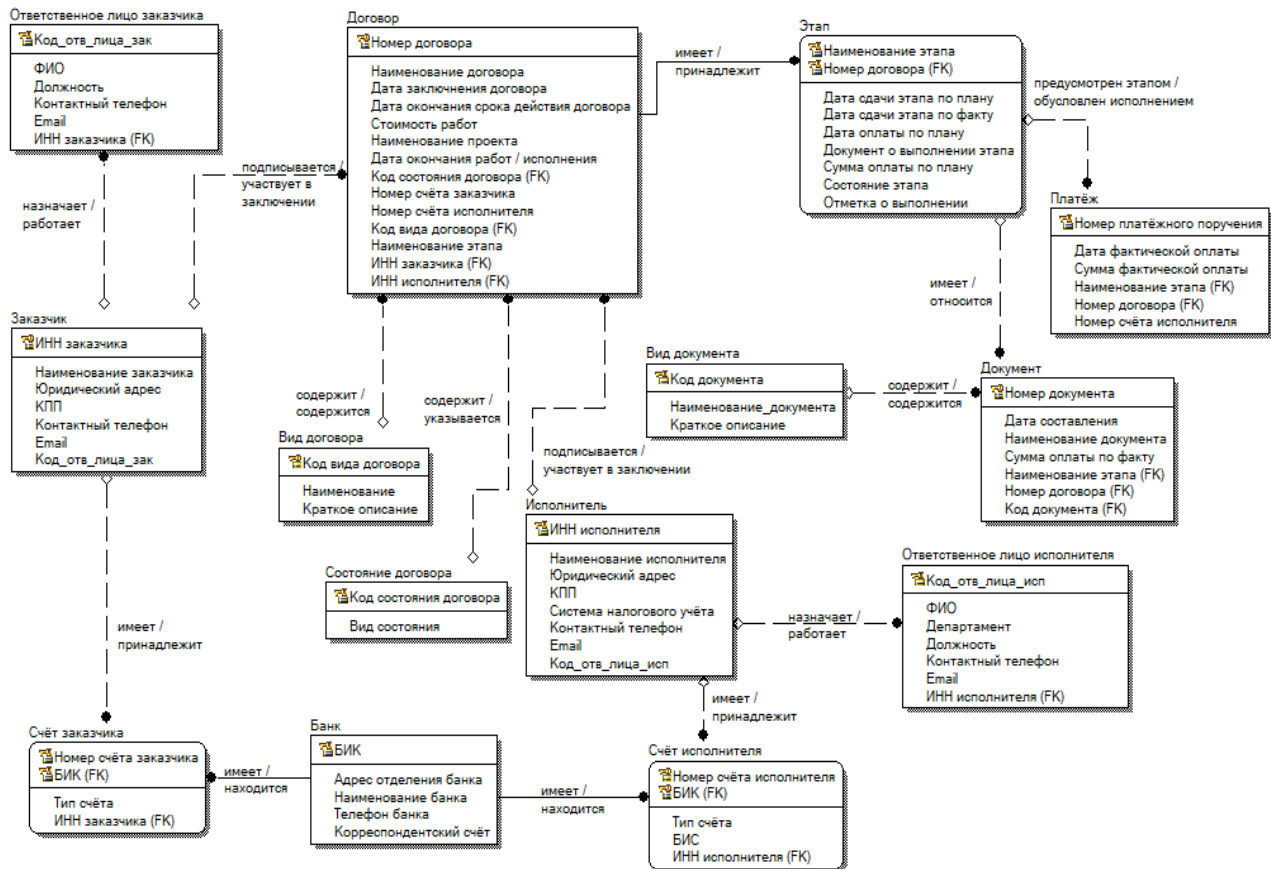


Рисунок 3.5 – FA-модель

Диаграмма FA-уровня показывает имена всех атрибутов сущностей и связей и полностью определяет структуру и взаимосвязи данных.

3.9 Физическая модель БД

Физический (второй) уровень модели жестко опирается на использование конкретной СУБД. В физической модели должна содержаться информация обо всех объектах БД. Если в логической модели не имеет значения, какой конкретно

тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т. д.

Физический уровень представления модели зависит от выбранного сервера СУБД. ERwin поддерживает практически все распространенные СУБД, всего более 20 реляционных и нереляционных СУБД.

Имена таблиц и колонок по умолчанию будут сгенерированы на основе сущностей и атрибутов логической модели, учитывая максимальную длину имени и другие синтаксические ограничения, накладываемые СУБД. Если в имени сущности или атрибута встречается пробел, он будет заменен на символ «_». Все сделанные изменения не отражаются на именах сущностей и атрибутов, поскольку информация на логическом и физическом уровнях в ERwin хранится отдельно (табл. 3.1).

Таблица 3.1 – Сопоставление компонентов логической и физической модели

Логическая модель	Физическая модель
Сущность	Таблица
Атрибут	Столбец
Логический тип (текст, число, дата)	Физический тип (корректный тип, зависящий от выбранной СУБД)
Первичный ключ	Первичный ключ (индекс РК)
Внешний ключ	Внешний ключ (индекс FK)
Альтернативный ключ	АК-индекс – уникальный непервичный индекс
Правило бизнес-логики	Триггер или сохраненная процедура
Взаимосвязи	Взаимосвязи, определяемые использованием FK-атрибутов

Целью создания физической модели является обеспечение администратора соответствующей информацией для переноса логической модели данных в СУБД.

ERwin поддерживает автоматическую генерацию физической модели данных для конкретной СУБД. При этом логическая модель (FA-уровень, рис. 3.5)

трансформируется в физическую по следующему принципу: сущности становятся таблицами, атрибуты становятся столбцами, а ключи становятся индексами.

Далее будут представлены глоссарий созданной модели, определение доменов и правила целостности данных.

3.10 Глоссарий модели

Определения сущностей модели приведены в таблице 3.2. Определения атрибутов приведены в таблице 3.3.

Таблица 3.2 – Определения сущностей

Имя	Определение
Договор	Письменное соглашение двух сторон, фиксирующее их определенные обязательства
Заказчик	Заказчик – тот, кто осуществляет заказ, является заказчиком по заключенному договору
Исполнитель	Исполнитель – тот, кто отвечает за исполнение заказа (выполнение работ) по данному договору
Этап	Это стадия выполнения договора, определенный временной период, за который должна быть выполнена часть работ, описанная в договоре
Документ	Деловые бумаги, служащие, например, подтверждением выполнения этапа договора (либо выполнения всех работ по договору)
Вид документов	Справочник, в котором хранятся наименования всех возможных документов, необходимых для заключения и выполнения договоров
Платеж	Оплата за выполнение работ по этапу (либо за выполнение работ по всему договору)
Банк	Информация об обслуживающем банке
Вид договоров	Справочник, в котором хранятся все необходимые виды договоров
Ответственное лицо заказчика	Список сотрудников от заказчика, которые периодически подписывают различные договоры от организации
Счет исполнителя	Номера всех известных счетов в банках исполнителя по договорам

Счет заказчика	Номера всех известных счетов в банках заказчика по договорам
Состояние договора	Справочники видов состояний договоров
Ответственное лицо исполнителя	Список сотрудников от исполнителя, которые периодически подписывают различные договоры от организации

Таблица 3.3 – Определения атрибутов сущностей

Имя	Описание	Сущность-владелец	Домен
ИНН исполнителя	Уникальный индивидуальный налоговый номер исполнителя	ИСПОЛНИТЕЛЬ	D_Code
Наименование исполнителя	Полное наименование организации, которая берет на себя функции исполнителя по договору	ИСПОЛНИТЕЛЬ	D_Text
Юридический адрес	Юридический адрес организации	ИСПОЛНИТЕЛЬ	D_Adress
КПП	КПП (код причины постановки на учет)	ИСПОЛНИТЕЛЬ	D_string
Система налогового учета	Имеющаяся на предприятии исполнителя система налогового учета	ИСПОЛНИТЕЛЬ	D_Text
Контактный телефон	Контактный телефон, по которому можно связаться с исполнителем по договору	ИСПОЛНИТЕЛЬ	D_Phone
Email	Адрес электронной почты	ИСПОЛНИТЕЛЬ	D_Mail
Код_отв_лица_исп	Код лица от исполнителя, ответственного за выполнение работ по данному договору	ИСПОЛНИТЕЛЬ	D_Code
ИНН заказчика	Уникальный индивидуальный номер налогоплательщика заказчика	ЗАКАЗЧИК	D_string
Юридический адрес	Юридический адрес организации	ЗАКАЗЧИК	D_Adress
Наименование заказчика	Полное наименование организации, которая является заказчиком по договору	ЗАКАЗЧИК	D_Text
КПП	КПП (код причины постановки на учет)	ЗАКАЗЧИК	D_string
Контактный телефон	Контактный телефон, по которому можно связаться с организацией заказчика	ЗАКАЗЧИК	D_Phone
Email	Адрес электронной почты	ЗАКАЗЧИК	D_Mail

Код_отв_лица_зак	Код лица от заказчика, ответственного за выполнение работ по данному договору	ЗАКАЗЧИК	D_Code
Номер договора	Номер договора, который является уникальным	ДОГОВОР	D_D
Наименование договора	Полное наименование договора	ДОГОВОР	D_Text
Дата заключения договора	Дата заключения договора	ДОГОВОР	D_Date
Дата окончания срока действия договора	Дата окончания действия договора, после которой договор теряет свою силу	ДОГОВОР	D_Date
Стоимость работ	Стоимость всех работ, которые будут производиться по договору	ДОГОВОР	D_Number
Наименование проекта	Наименование проекта по договору	ДОГОВОР	D_Text
Дата окончания работ	Дата окончания всех работ по договору	ДОГОВОР	D_Date
Код вида договора	Вид договора	ДОГОВОР	D_Code
ИНН заказчика	Уникальный индивидуальный налоговый номер заказчика по данному договору	ДОГОВОР	D_Code
ИНН исполнителя	Уникальный индивидуальный налоговый номер исполнителя по данному договору	ДОГОВОР	D_Code
Код состояния договора	Код текущего состояния работ по договору	ДОГОВОР	D_Code
Основание договора	Номер договора, который является основанием для заключения текущего	ДОГОВОР	D_D
Код ответственного лица	Код лица, ответственного за выполнение работ по данному договору	ДОГОВОР	D_Code
Счет (заказчика и исполнителя)	Номер счета в банке, указанный в договоре	ДОГОВОР	D_string
Наименование этапа	Наименование этапа, по которому выполняется договор	ДОГОВОР	D_Text
БИК	Банковский идентификационный код, который входит в число обязательных реквизитов	БАНК	D_Code
Адрес отделения банка	Адрес, по которому расположен данный банк	БАНК	D_Adress
Наименование банка	Полное наименование банка	БАНК	D_Text
Телефон банка	Телефон, по которому можно связаться с банком	БАНК	D_Phone

Корреспондентский счет	Корреспондентский счет	БАНК	D_string
Номер платежного поручения	Номер платежного поручения	ПЛАТЕЖ	D_string
Дата фактической оплаты	Дата совершения платежа фактическая, из платежного поручения	ПЛАТЕЖ	D_Date
Сумма фактической оплаты	Сумма оплаты фактическая	ПЛАТЕЖ	D_Number
Наименование этапа	Полное наименование этапа	ПЛАТЕЖ, ЭТАП	D_Text
Номер договора	Номер договора, по которому совершен платеж	ПЛАТЕЖ	D_D
Номер счета исполнителя	Номера счетов, указанные в платежном поручении	ПЛАТЕЖ	D_Code
Наименование этапа	Наименование этапа договора	ЭТАП	D_Text
Номер договора	Номер договора	ЭТАП	D_D
Дата сдачи этапа по плану	Дата сдачи этапа, которая оговорена в договоре, плановая	ЭТАП	D_Date
Дата сдачи этапа по факту	Дата сдачи этапа фактическая	ЭТАП	D_Date
Дата оплаты по плану	Дата оплаты по этапу, оговоренная в договоре	ЭТАП	D_Date
Сумма оплаты по плану	Сумма оплаты по этапу, оговоренная в договоре	ЭТАП	D_Number
Документ о выполнении этапа	Документ, подтверждающий выполнение этапа	ЭТАП	D_Text
Состояние этапа	Текущее состояние работ по этапу	ЭТАП	D_Text
Отметка о выполнении	Отметка о выполнении всех работ по этапу	ЭТАП	D_realization
Код вида договора	Уникальный код вида договора, который присваивается автоматически	ВИД ДОГОВОРОВ	D_Code
Наименование	Наименование вида договора	ВИД ДОГОВОРОВ	D_Text
Краткое описание	Краткое описание, где применяется данный вид	ВИД ДОГОВОРОВ	D_DText
Код документа	Уникальный код документа, который присваивается автоматически	ВИД ДОКУМЕНТОВ	D_Code
Наименование документа	Полное наименование документа	ВИД ДОКУМЕНТОВ	D_Text
Краткое описание	Краткое описание вида документа	ВИД ДОКУМЕНТОВ	D_Dtext
Номер документа	Уникальный номер документа	ДОКУМЕНТ	D_D
Дата составления	Дата составления документа	ДОКУМЕНТ	D_Date

Наименование документа	Полное наименование документа	ДОКУМЕНТ	D_Text
Сумма оплаты по факту	Сумма оплаты по этому документу	ДОКУМЕНТ	D_Number
Наименование этапа	Наименование этапа, к которому относится документ	ДОКУМЕНТ	D_Text
Номер договора	Номер договора, к которому относится данный документ	ДОКУМЕНТ	D_D
Код документа	Определенный вид документа	ДОКУМЕНТ	D_Code
Номер счета исполнителя	Номер счета в банке исполнителя	СЧЕТ ИСПОЛНИТЕЛЯ	D_string
Тип счета	Тип счета исполнителя	СЧЕТ ИСПОЛНИТЕЛЯ	D_Text
БИС	Банковская информационная система, которую использует исполнитель	СЧЕТ ИСПОЛНИТЕЛЯ	D_Text
Номер счета заказчика	Номер счета в банке	СЧЕТ ЗАКАЗЧИКА	D_string
Тип счета	Тип счета заказчика	СЧЕТ ЗАКАЗЧИКА	D_Text
Ф. И. О.	Ф. И. О. сотрудника, ответственного за исполнение договора	ОТВЕТСТВЕННОЕ ЛИЦО ИСПОЛНИТЕЛЯ	D_Text
Департамент	Наименование департамента, в котором работает исполнитель	ОТВЕТСТВЕННОЕ ЛИЦО ИСПОЛНИТЕЛЯ	D_Text
Должность	Должность сотрудника	ОТВЕТСТВЕННОЕ ЛИЦО ИСПОЛНИТЕЛЯ	D_Text
Ф. И. О.	Ф. И. О. сотрудника, представителя заказчика	ОТВЕТСТВЕННОЕ ЛИЦО ЗАКАЗЧИКА	D_Text
Должность	Должность сотрудника, представителя заказчика	ОТВЕТСТВЕННОЕ ЛИЦО ЗАКАЗЧИКА	D_Text

3.11 Определение доменов

Таблица 3.4 – Определение доменов

Имя	Тип	Описание
4D_Code	TEXT(4)	Строка из четырех символов. Значения символов – цифры 0..9. Условие на значение LIKE '****'
D_Text	TEXT(50)	Цифробуквенная информация
D_Date	DATE(8)	Специальный числовой тип, интерпретируемый как <день><месяц><год>
D_Number	INTEGER	Специальный числовой тип. Положительное число
D_Phone	TEXT(14)	Цифробуквенная информация
D_DText	TEXT(500)	Цифробуквенная информация

D_Mail	TEXT(30)	Цифробуквенная информация, обязательно наличие @. Условие на значение LIKE '%@%'
D_Adress	TEXT(50)	Строка. Состоит <улица>, <дом>, [<строение>], [<квартира>][<номер кабинета>], [<номер офиса>]
D_FIO	TEXT(60)	Строка. Состоит из трех слов
D_realiza- tion	STRING	Строка. Принимает значения «да» или «нет». Усло- вие на значения IN ('ДА', 'НЕТ')
D_string	TEXT(20)	Строка. Значения символов – цифры
D_stage	TEXT(2)	Строка из двух символов. Значения символов – цифры 0..9
D_D	TEXT(10)	Цифробуквенная информация
D_object	OLE object	Файл Microsoft Word

3.12 Правила целостности данных

В данной модели данных предусмотрены следующие ограничения целостности:

1. Явно определены все необходимые типы данных (домены).
2. Значения атрибутов выбираются только из его домена.
3. Для каждого базового отношения определен первичный ключ.
4. База данных содержит значения внешнего ключа, не существующие среди значений первичного ключа родительского отношения.
5. Спецификации допустимости и недопустимости неопределенных значений.
6. Деловой регламент.

3.13 Деловой регламент

Деловой регламент – это множество правил, ограничивающих допустимые значения элементов данных и свойства их отношений.

Деловой регламент представлен пунктами, перечисленными ниже.

1. Каждый договор имеет уникальный номер.
2. Дата заключения договора, дата окончания действия договора, дата сдачи этапа и другие даты хранятся в виде: ДД.ММ.ГГГГ.
3. Сумма оплаты, стоимость работ по умолчанию – в рублях.

4. Наименования договоров могут повторяться.
5. Могут повторяться наименования этапов договоров.
6. Если договор выполняется одновременно, значит, он имеет единственный этап, факт выполнения которого фиксируется.
7. Фиксируется фактически поступивший платеж по этапу договора.
8. Отметка о выполнении этапа может принимать значения «ДА» или «НЕТ» (т. е. выполнены работы по этапу или не выполнены).
9. Каждый договор может иметь один определенный вид договора.
10. Не существует двух банков с одинаковыми значениями кода.
11. Каждый договор может иметь несколько этапов.
12. Выполнение каждого этапа подтверждается документом.
13. Каждый договор относится только к одному определенному виду договора.
14. Каждый документ может иметь один определенный вид документа.
15. Дата заключения договора должна быть более ранней датой по сравнению с датой окончания действия договора.
16. Дата сдачи этапа по плану и дата сдачи этапа по факту могут быть одинаковы.
17. Код вида договора имеет уникальное значение.
18. Код документа имеет уникальное значение.
19. ИНН исполнителей и ИНН заказчиков имеют уникальные значения.
20. Обязательно, что заказчик и исполнитель имеют фактический, юридический и почтовый адрес одновременно.
21. Значения расчетных счетов в банке уникальны.
22. Договор может иметь несколько дополнительных документов.
23. Состояние выполнения работ по договору или по этапу договора регистрируется.
24. Код ответственного лица (исполнителя, заказчика) имеет уникальное значение.

25. Фиксируется состояние договора. Состояние договора может принимать значения «на подготовке», «на исполнении», «закрит».

3.14 Транзакции пользователя

Транзакция (англ. *transaction*) – группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакции по последовательности операций учета договоров представлены ниже.

1. Регистрация новых заключенных договоров.
2. Обновление сведений о выполнении договора и этапов по договору.
3. Ввод и обновление сведений о заказчиках и исполнителях по договорам.
4. Регистрация фактов выполнения этапов договоров.
5. Обновление данных о состоянии договора.
6. Регистрация документов, дополнительно необходимых при заключении договоров, а также подтверждающих выполнение этапов договоров.
7. Обновление справочника обслуживающих банков.
8. Ввод данных о фактически поступивших платежах по договорам.
9. Генерация различных отчетов.

4 Заключение

В ходе подготовки итогового отчета по дисциплине «НИР в семестре» проведены виды работ, представленные ниже:

- Изучена деятельность предприятия ООО «ЛЕМА групп», определен уровень автоматизации используемого на предприятии программного обеспечения.

- При изучении автоматизации был выделен и описан бизнес-процесс, который необходимо автоматизировать, построена SADT-модель «As-Is».

- Изучены договорные отношения в ООО «ЛЕМА групп», особенности учета и процесс управления договорами, рассмотрена литература по данной теме.

- Сформулирована постановка задачи, определена входная и выходная информация, описан алгоритм ее решения.

- Построена концептуальная модель базы данных информационной системы (ER-, KB-, FA-модели).

- Разработан глоссарий модели, определены домены.

- Разработаны правила целостности данных.

- Разработан деловой регламент.

- Определены транзакции пользователя.

В дальнейшем предполагается разработка информационной системы, включая реализацию базы данных средствами разработки интерфейса Delphi Architect XE3 и разработки базы данных Microsoft Office Access 2016.

Список использованных источников

1. Исакова, А. И. Информационные системы : учеб. пособие / А. И. Исакова. – Томск, 2010. – 132 с.
2. Устав Общества с ограниченной ответственностью «ЛЕМА групп». Утвержден Протоколом общего собрания учредителей № 1 от 18.04.2018.
3. Гражданский кодекс Российской Федерации. Части первая, вторая, третья, четвертая. По состоянию на 25 января 2020 г. – Новосибирск : Норматика, 2020. – 623 с.
4. Серочудинов, Е. С., Мизова, А. И. Эффективность организационной структуры управления предприятием [Электронный ресурс] // Экономика и менеджмент инновационных технологий. – 2014. – № 6. – Режим доступа: <http://ekonomika.snauka.ru/2014/06/5343> (дата обращения: 16.05.2020).
5. Лысаков, А. В. Договорные отношения в управлении проектами / А. В. Лысаков, Д. А. Новиков. – М. : ИПУ РАН, 2004. – 100 с.
6. Исакова, А. И. Научная работа 1 : учеб. пособие / А. И. Исакова. – Томск : ТУСУР, 2017. – 141 с.
7. Золотов, С. Ю. Проектирование информационных систем : учеб. пособие / С. Ю. Золотов. – Томск, 2016. – 117 с.
8. Золотов, С. Ю. Проектирование информационных систем : учеб. метод. пособие / С. Ю. Золотов. – Томск : ФДО, ТУСУР, 2013. – 37 с.
9. Кузин, А. В. Базы данных : учеб. пособие для вузов / А. В. Кузин, С. В. Левонисова. – 4-е изд., стереотип. – М. : Академия, 2010. – 320 с.
10. Сибилев, В. Д. Базы данных : учеб. пособие / В. Д. Сибилев. – Томск : ТУСУР, 2007. – 279 с.
11. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – 2-е изд. – М. : Юрайт, 2012. – 464 с.
12. Давыдова, Е. М. Базы данных : учеб. пособие / Е. М. Давыдова, Н. А. Новгородова. – 3-е изд., перераб. и доп. – Томск : В-Спектр, 2012. – 128 с.
13. Образовательный стандарт вуза ОС ТУСУР 01–2013. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления [Электронный ресурс]. – Томск : ТУСУР, 2013. – 57 с. – Режим доступа: <https://regulations.tusur.ru/documents/70> (дата обращения: 27.03.2020).