

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

Кафедра математики и информатики

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ
«БАЗЫ ДАННЫХ И ИНФОРМАЦИОННЫЕ СИСТЕМЫ»

Направление подготовки (специальность):

01.03.02 Прикладная математика и информатика

Образовательная программа:

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Очная форма обучения

Составители:

Лавров В.В., старший
преподаватель кафедры МиИ

г. Череповец - 2022

Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

Основная литература:

1. Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование : учебник для вузов / В. К. Волк. — 3-е изд., стер. — Санкт-Петербург : Лань, 2022. — 244 с. — ISBN 978-5-8114-9368-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/193373>
2. Крейдер, О. А. Информационные системы и технологии : учебное пособие / О. А. Крейдер. — Дубна : Государственный университет «Дубна», 2019. — 61 с. — ISBN 978-5-89847-577-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/154486>
3. Кривцов, А. Н. Информационные технологии. Основы работы с базами данных : учебное пособие / А. Н. Кривцов, С. В. Хорошенко. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2018. — 107 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/180052>

Дополнительная литература:

1. Петрова, А. Н. Реализация баз данных : учебное пособие / А. Н. Петрова, В. Е. Степаненко. — Комсомольск-на-Амуре : КНАГУ, 2020. — 144 с. — ISBN 978-5-7765-1448-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/151716>
2. Козлова, О. С. Базы данных : методические рекомендации / О. С. Козлова, А. С. Тучкова. — Самара : ПГУТИ, 2019. — 50 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/223232>
3. Селяничев О.Л. Базы данных : учебное пособие для вузов. - Череповец : ФГБОУ ВПО ЧГУ, 2013. - 126 с. - Библиогр.: с.126. - ISBN 978-5-85341-550-8<https://edu.chsu.ru/>
4. Кузин, А.В. Базы данных : учебное пособие для вузов / А.В. Кузин, С.В. Левонисова. - Москва : ИЦ "Академия", 2005. - 315 с. + Приложения. - (Высшее профессиональное образование). - Библиогр.: с.313. - ISBN 5-7695-1796-4.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля), включая перечень информационных справочных систем (при необходимости)

- 1 Интерактивная доска.
- 2 <http://www.ois.org.ua/spravka/mat/index.htm> - электронная библиотека по математике.
- 3 <http://eqworld.ipmnet.ru/ru/library.htm>- учебно-образовательная физико-математическая библиотека.
- 4 <http://www.exponenta.ru/>- образовательный математический сайт.

Учебно-методические указания и рекомендации к изучению тем лекционных и практических занятий, самостоятельной работе студентов

Лекции

№ п/п	Тема лекции	Количество часов
1	Введение в теорию баз данных	4
2	Введение в архитектуру систем баз данных	4
3	Модели данных и модели базы данных	4
4	Реляционная модель данных	4
5	Нормализация реляционных базы данных	4
6	Проектирование баз данных, ER-диаграммы	4
7	Основы языка SQL	2
8	Основные понятия технологии проектирования информационных систем	2
9	Жизненный цикл программного обеспечения ИС	2
10	Проектирование ИС	2
11	Анализ и моделирование функциональной области внедрения ИС	2
12	Спецификация функциональных требований к ИС	2
13	Методологии моделирования предметной области	2
14	Моделирование бизнес-процессов средствами BPwin	4
15	Информационное обеспечение ИС	2
16	Моделирование информационного обеспечения	2
17	Унифицированный язык визуального моделирования Unified Modeling Language (UML)	2
18	Этапы проектирования ИС с применением UML	2
Итого		50

Лабораторные работы

№ п/п	Тема лекции	Количество часов
1	Основы языка SQL	10
2	Основные понятия технологии проектирования информационных систем	8
3	Жизненный цикл программного обеспечения ИС	8
4	Проектирование ИС	8
5	Анализ и моделирование функциональной области внедрения ИС	8
6	Спецификация функциональных требований к ИС	8
7	Методологии моделирования предметной области	8
8	Моделирование бизнес-процессов средствами BPwin	8
9	Информационное обеспечение ИС	8
10	Моделирование информационного обеспечения	8
11	Унифицированный язык визуального моделирования Unified Modeling Language (UML)	10
12	Этапы проектирования ИС с применением UML	8
Итого		100

Лекция 1. Введение в теорию баз данных

План лекции:

1. Основные понятия.
2. Компоненты системы баз данных.
3. Этапы развития систем управления базами данных и ведущие производители.
4. Преимущества и недостатки централизованного подхода в управлении данными.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – М. : Издательство Юрайт, 2017. – 230 с. – Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).
4. Маркин, А. В. Программирование на sql в 2 ч. Часть 1 : учебник и практикум для бакалавриата и магистратуры / А. В. Маркин. — М. : Издательство Юрайт, 2017. — 362 с. — (Бакалавр и магистр. Академический курс).

Основные понятия

Исторически сложились два основных направления использования вычислительной техники, первое из которых связано с проведением сложных преобразований над относительно небольшими объемами данных с простой структурой. Здесь компьютеры позволили быстрее проводить расчеты по сложным вычислительным алгоритмам. Подобные задачи дали толчок к созданию первых ЭВМ, их актуальность не снижается и сейчас.

Другое направление связано с созданием информационных систем. В них необходимо не только обрабатывать, но и хранить большие объемы данных со сложной внутренней структурой, обеспечивать быстрый поиск нужной информации. Создание подобных систем стало возможным после появления надежных, емких и быстродействующих устройств энергонезависимой памяти: в первую очередь речь идет о накопителях на жестких магнитных дисках. Классическим примером систем подобного типа являются системы резервирования железнодорожных и авиабилетов. Последовательность операций, выполняемых при каждом заказе, относительно проста, но для

корректного функционирования всей системы необходимо хранить и постоянно актуализировать большие объемы данных, выполнять в них поиск и т.п.

В отечественных нормативных документах в сфере разработки баз данных даются следующие определения:

База данных (БД) – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ [ГОСТ 20886-85 Организация данных в системах обработки данных, п. 6].

Система управления базами данных (СУБД) – совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами [ГОСТ 20886-85 Организация данных в системах обработки данных, п. 46]

Банк данных – это система специальным образом организованных данных (баз данных), программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

В англоязычной литературе понятие, по сути близкое к банку данных, обозначается термином *система баз данных* (англ. database system).

БД можно рассматривать как электронную картотеку, хранилище для некоторого набора занесенных в компьютер данных. Над БД выполняют следующие операции:

- добавление новых данных;
- изменение существующих данных;
- удаление данных;
- поиск данных и пр.

Базы данных организуют на основе различных моделей данных. Пример фрагмента БД реляционного типа представлен в табл. 1. Данные в этом случае организуются в виде реляционных таблиц, строки таблицы называют записями, столбцы – полями или атрибутами.

Принципиально важной особенностью БД является то, что они содержат дополнительную служебную информацию о своей структуре, иначе говоря, являются самодокументируемыми.

Таблица 1. Фрагмент реляционной БД

StudID	FIO	Group
123	Иванов И.И.	382
124	Петров П.П.	382

Компоненты системы баз данных

Рассмотрим упрощенную схему системы БД (рис. 1). Она включает следующие основные компоненты: данные; аппаратное обеспечение; программное обеспечение; пользователей.

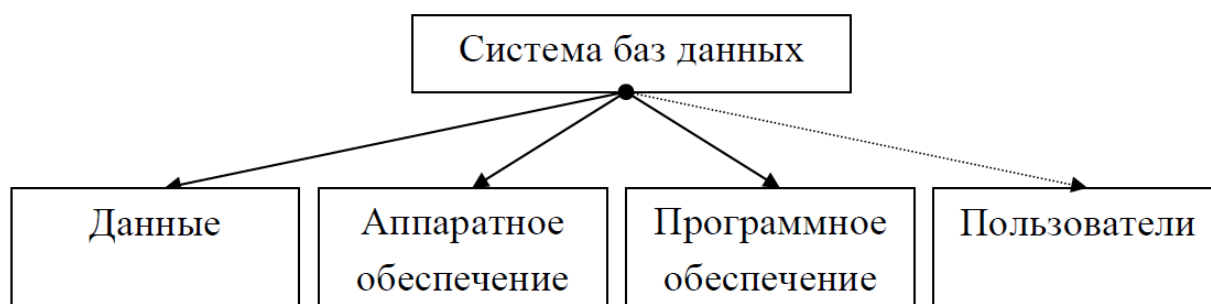


Рисунок 1. Обобщенная схема системы БД

Данные. БД состоят из некоторого набора *постоянных данных*. Выделяют также транзитные данные, такие как промежуточные результаты, входные и выходные данные. *Входные данные* – информация, передаваемая системе (например, вводимая с клавиатуры). Такая информация может стать причиной изменения постоянных данных (она может стать частью постоянных данных), но не является частью БД как таковой. *Выходные данные* – сообщения и результаты, выдаваемые системой. Они, как правило, берутся из постоянных данных, но их нельзя рассматривать, как часть БД. Например, пусть в систему каждые 10 минут поступают данные о температуре воздуха, в базе сохраняется среднее значение за час, а запрос выводит среднесуточную температуру. В этом случае хранимые данные могут отличаться от входных и выходных.

Кроме данных, описывающих предметную область, в БД обычно содержатся данные, описывающие элементы и структуры самой базы. Подобные описания относятся к разряду *метаинформации*, т.е. «информации об информации». Централизованное хранилище метаинформации называется *словарем данных* или *репозиторием*. Именно наличие репозитория позволяет говорить о свойстве самодокументируемости БД. В современных СУБД реляционного типа такое хранилище реализуется в виде системного каталога – набора служебных таблиц, куда заносится информация о структуре объектов (таблиц, представлений и др.), пользователях, разрешениях и др.

По виду отношения «пользователь – данные» выделяют два типа систем БД:

- 1) *однопользовательская система* (англ. single-user system) – система, в которой в одно и то же время к БД может получить доступ только один пользователь;
- 2) *многопользовательская система* (англ. multi-user system) – система, в которой к БД могут получить доступ одновременно несколько пользователей. При этом для конечного пользователя необходимо обеспечить такие условия, чтобы результат его работы не зависел от того, работает он с данными в однопользовательском режиме или совместно с другими пользователями.

Данные в БД должны быть *интегрированными* и *общими*. Когда говорят про *интегрированные* данные, подразумевают, что к данным, собранным из разных источников, предоставляется единый способ доступа. Например, система позволяет получить данные с кафедр университета об успеваемости студентов, из библиотеки – об использовании студентами литературы, и совместно использовать их для решения какой-либо задачи. *Общие* данные подразумевают возможность использования отдельных наборов данных из общей БД разными группами пользователей для решения своих специфических задач. Например, менеджер интернет-магазина может работать с данными о конкретном заказе, а руководитель – с итоговыми данными, характеризующими деятельность магазина за определенный период. Эти два свойства представляют собой наиболее важное преимущество использования систем БД корпоративного уровня, а «интеграция» является преимуществом при использовании настольных (персональных) систем БД.

Аппаратное обеспечение. В наиболее общем виде можно выделить две группы устройств, принципиально важных каждой системе БД:

- 1) устройства хранения данных;
- 2) устройства обработки данных.

Для небольших систем обработка и хранение могут производиться на одном и том же компьютере. Крупная БД может использовать различные системы хранения и множество серверов для обработки данных. Здесь возникает целый класс новых задач, связанных с разработкой и эксплуатацией распределенных систем.

Программное обеспечение. Между физической БД и пользователями системы располагается уровень программного обеспечения (ПО), основной компонент которого – система управления базами данных (СУБД) (англ. database management system). Основная функция СУБД – предоставление пользователю БД возможности работать с ней, не вникая в детали на уровне аппаратного обеспечения.

Кроме СУБД система БД, как правило, включает еще ряд программных компонентов – утилиты, генераторы отчетов, пользовательское прикладное ПО и др.

Пользователи. Пользователей системой БД можно разделить на три класса: прикладные программисты; конечные пользователи; администраторы данных и администраторы баз данных.

Прикладные программисты отвечают за написание прикладных программ, использующих БД. Разрабатываемые ими программы обращаются с запросами к СУБД и получают результаты запросов. Выделяют программы пакетной обработки и оперативные приложения, функция которых – поддержка работы конечного пользователя, имеющего интерактивный доступ к системе.

Конечные пользователи работают с системой БД непосредственно с рабочей станции или терминала. Они могут воспользоваться разработанными для них прикладными программами или встроенными средствами СУБД (графическими или с интерфейсом командной строки). Нужно понимать, что система БД создается для поддержания деятельности конечных пользователей.

Администраторы данных и администраторы баз данных. Администратор данных – специалист, который несет ответственность за данные предприятия. Он принимает решение, какие данные необходимо вносить в БД, кому и к каким данным следует предоставлять доступ и т.д. Таких специалистов называют еще и аналитиками данных. Администратор базы данных – технический специалист, который отвечает за реализацию решений администратора данных. На этапе разработки системы он занимается созданием БД, на этапе эксплуатации – настройкой, обслуживанием, резервным копированием и другими подобными задачами.

Этапы развития систем управления базами данных и ведущие производители

До появления СУБД вопросы хранения данных разработчики каждой программы решали самостоятельно, используя при этом функции операционной системы (ОС) или даже напрямую обращаясь к устройствам ввода/вывода. Однако ОС предоставляет функции по работе с файлами, а вопросы организации хранения записей внутри файла, поиска данных, проверки ограничений для записи средствами ОС не решить. Кроме того, при одновременном доступе нескольких пользователей к одним и тем же данным необходимы дополнительные механизмы, позволяющие централизованно управлять этим процессом. Эти и ряд других причин привели к созданию отдельного класса программного обеспечения – СУБД.

Первый этап развития СУБД связаны с мейнфреймами. Первая коммерческая СУБД называлась IMS (англ. Information Management System, система управления информацией) и была выпущена компанией IBM в 1968 году для платформы IBM System/360. Этот этап характеризуется централизованным хранением данных. СУБД должны были обеспечивать коллективный доступ к БД, а сами они работали на «больших» машинах под управлением сложных и достаточно развитых ОС. На первом этапе исследователями были получены существенные результаты в области теории БД. В частности, это создание иерархической, сетевой и реляционной моделей данных. Реляционную модель предложил работавший в IBM математик Эдгар Франк Кодд (англ. Edgar Frank Codd, 1923-2003). В 1970 г. он опубликовал статью «A Relational Model of Data for Large Shared Data Banks», в которой описал основные идеи реляционного подхода. В дальнейшей работе над моделью принял участие Кристофер Дейт (Christopher J. Date), автор классического учебника «Введение в системы баз данных». Реляционные СУБД на сегодняшний день являются наиболее распространенными.

Следующий этап развития СУБД связан с появлением персональных компьютеров. Их широкое распространение, ограниченные вычислительные возможности и в среднем более низкий (по сравнению с большими ЭВМ) уровень подготовки пользователей привели к возникновению целого класса настольных СУБД. Изначально это были однопользовательские системы с достаточно ограниченными возможностями, простым пользовательским интерфейсом и невысокими требованиями к аппаратуре. Многие из них не выдержали конкуренции и сейчас не поддерживаются. Оставшиеся в процессе развития стали приобретать черты многопользовательски СУБД, например, механизмы совместного использования и защиты данных. В качестве примера популярных сейчас настольных СУБД можно называть Microsoft Access и LibreOffice Base. Параллельно существенные изменения происходили и с СУБД корпоративного уровня. Они были связаны с распространением компьютерных сетей, в результате чего доминирующей стала клиент-серверная технология, в том числе с поддержкой распределенной обработки данных.

Большое влияние на развитие СУБД оказала сеть Интернет. При динамическом формировании веб-страниц в большинстве случаев задействуются СУБД и обслуживаемые ими БД. Это привело к появлению ряда СУБД, чья популярность, в первую очередь, связана с использованием при создании веб-приложений. Наиболее яркий пример – реляционная СУБД MySQL.

В то же время выяснилось, что реляционные СУБД и используемый для работы с ними язык SQL подходят далеко не для всех задач. Появилась и

активно развивается идеология NoSQL (англ. Not only SQL, не только SQL), объединяющая ряд подходов и проектов, связанных с созданием нереляционных БД.

Наиболее именитый производитель серверных СУБД – корпорация Oracle, выпустившая в 1979 г. первую коммерческую реляционную СУБД Oracle v2 и с тех пор являющаяся ключевым производителем в области серверов БД.

Существенное место на рынке занимает корпорация IBM, выпускающая реляционную СУБД DB2 и иерархическую СУБД IMS. Приобретя в 2001 г. подразделение корпорации Informix, IBM добавила в свою линейку продуктов одноименную СУБД.

Заметное место среди разработчиков СУБД занимает корпорация Microsoft с серверной СУБД Microsoft SQL Server и настольной СУБД Access, входящей в пакет Microsoft Office. Несмотря на то, что MS SQL Server выпускается только для ОС семейства Windows, широкие возможности данной СУБД привели к ее широкому распространению.

Основанная в 1984 г. компания Sybase может также быть названа одним из пионеров в области разработки реляционных СУБД. В линейке продуктов Sybase можно назвать реляционный сервер БД Adaptive Server Enterprise, реляционную СУБД SQL Anywhere и нереляционную СУБД Sybase IQ, предназначенную для задач аналитической обработки данных и построения хранилищ данных. В 2010 г. Sybase была приобретена компанией SAP AG, ведущим поставщиком программных решений для управления бизнесом.

Среди приверженцев свободно распространяемого ПО широкую популярность приобрела СУБД MySQL, изначально разрабатываемая созданной в Швеции компанией MySQL AB. В настоящее время у MySQL лидирующие позиции в качестве СУБД, используемой в области веб-разработки. В 2008 г. компания MySQL AB была приобретена Sun Microsystems, а в 2010 г. уже саму Sun Microsystems приобрела компания Oracle. Сейчас выпускается как коммерческая, так и бесплатно распространяемая версия MySQL (MySQL Community Edition). Кроме того, существуют разрабатываемые сообществом свободно распространяемые ответвления MySQL, например, MariaDB.

Также необходимо отметить, что у многих коммерческих разработчиков есть бесплатно распространяемые версии СУБД, такие как Oracle Database Express Edition, IBM DB2 Express-C, Microsoft SQL Server Express Edition.

Если говорить о СУБД, основанных на объектной модели данных, то наиболее известным на сегодняшний день проектом в этой области является система Cache (произносится «кашэ»), разрабатываемая компанией InterSystems.

Особенность данной СУБД заключается в том, что она реализует объектное представление данных, сохраняя в то же время возможность доступа к данным средствами языка SQL как к реляционной БД.

Преимущества и недостатки централизованного подхода в управлении данными

Преимущества централизованного подхода в управлении данными, по сравнению с ситуацией, когда каждая программа независимо реализует хранение своих данных, заключается в следующем:

1. *Сокращение избыточности.* При отсутствии централизованной БД каждое приложение будет отдельно хранить свои данные. Нередко одни и те же данные используются несколькими приложениями. Например, список сотрудников в отделе кадров и список сотрудников, записанных в библиотеку предприятия, содержат имя, адрес, паспортные данные и др. одних и тех же сотрудников. В централизованной БД такие данные можно объединять с полным или частичным устранением избыточности.
2. *Возможность устранения противоречивости.* Зачастую противоречия являются следствием избыточности. Если одинаковые данные об одном человеке представлены в двух различных записях и это «раздвоение» не уточнено, то со временем эти записи могут перестать согласовываться, например, в одну из них внесут изменения, а в другую – нет. В этом случае БД станет противоречивой. Противоречий можно избежать, устраняя избыточность или контролируя ее. В последнем случае может использоваться множественное обновление, когда при внесении изменений в одну из записей автоматически будут изменены и записи, связанные с ней.
3. *Возможность общего доступа к данным.* При наличии централизованной БД сотрудники разных подразделений в соответствии с их предпочтениями могут совместно использовать эти данные.
4. *Возможность соблюдения стандартов.* Внедрение единых стандартов по обработке данных намного проще осуществить в централизованной системе.
5. *Возможность введения ограничений для обеспечения безопасности.* При централизованном хранении и обработке данных проще разработать и внедрить правила разграничения доступа к ним.
6. *Возможность обеспечения целостности данных.* Задача обеспечения целостности данных заключается в обеспечении правильности и точности данных в БД. Возникновение противоречий – это пример нарушения целостности. Другой пример: при централизованном

хранении проще организовать процедуры резервного копирования и восстановления БД.

В то же время, неудачное проектирование и внедрение системы БД может повлечь ряд проблем, основная из них связана с тем, что вся работа информационной системы предприятия будет зависеть от системы БД и сбой в ней могут привести к катастрофическим последствиям.

Вопросы и задания к лекции

1. Познакомьтесь с ГОСТ 20886-85 «Организация данных в системах обработки данных. Термины и определения» (<http://vsegost.com/Catalog/12/12548.shtml>).

Данный стандарт устанавливает применяемые в науке, технике и производстве термины и определения основных понятий в области организации данных в системах обработки данных. Термины, установленные настоящим стандартом, обязательны для применения в документации всех видов, научно-технической, учебной и справочной литературе.

Приведите определения следующих понятий: база данных; набор данных; элемент данных; система управления базами данных. Как соотносятся друг с другом данные понятия? Представьте это соотношение графически.

2. Познакомьтесь со стандартом ISO/IEC 2382-1 (<http://www.morepc.ru/informatisation/iso2381-1.html>). Приведите определения понятий: данные; база данных; банк данных. Как соотносятся данные понятия? Отобразите это соотношение графически.
3. Полностью ли соответствуют друг другу определения понятия «база данных» в стандартах, рассмотренных вами при выполнении заданий 1-2? В чем заключается общность подходов? В чем заключается отличие в подходах, если оно существует?
4. Познакомьтесь с обобщенной схемой системы баз данных. Как соотносятся друг с другом компоненты данной системы? Представьте графически это соотношение.
5. В чем отличие между рассмотрением истории баз данных в «широком» и «узком» смыслах? Какие популярные СУБД (и их разработчики) не упомянуты в соответствующем параграфе лекции? В каких сферах деятельности используются данные системы?
6. В чем заключается суть подхода NoSQL? Перечислите основные типы хранилищ данных при таком подходе.
7. Какие недостатки и достоинства централизованного подхода в управлении данными можно выделить, кроме приведенных в соответствующем параграфе лекции?

Лекция 2. Введение в архитектуру систем баз данных

План лекции:

1. Введение
2. Архитектура систем управления базами данных ANSI/SPARC.
3. Архитектура многопользовательских систем баз данных.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – М. : Издательство Юрайт, 2017. – 230 с. – Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).
4. Швецов, В. Лекция 3: Различные архитектурные решения, используемые при реализации многопользовательских СУБД. Краткий обзор СУБД [электронный ресурс] // URL: <http://www.intuit.ru/studies/courses/508/364/lecture/8643>

Введение

Под архитектурой системы БД понимается совокупность ее функциональных компонентов, а также средств обеспечения их взаимодействия друг с другом, с пользователем и с системным персоналом.

Представленный далее материал обобщенно описывает архитектуру системы БД. Это не означает, что это описание подходит для каждой системы БД. Например, «малые» системы, возможно, не будут поддерживать все аспекты архитектуры.

Ниже в упрощенном виде рассматривается архитектура, предложенная комитетом SPARC (англ. Standards Planning and Requirements Committee, комитет по планированию стандартов) американского национального института стандартов ANSI, так называемая архитектура ANSI/SPARC, впервые представленная в 1975 г.

Архитектура систем управления базами данных ANSI/SPARC

Архитектура СУБД ANSI/SPARC описывает логическую организацию системы с точки зрения представления данных пользователям и включает три уровня: *внешний; концептуальный; внутренний*.

Каждый из них состоит из одного или нескольких представлений (см. рис. 1).
Уровни определяются следующим образом:

- *внешний уровень* наиболее близок к пользователям, он связан со способами представления данных для отдельных пользователей;
- *концептуальный уровень* – промежуточный уровень между двумя первыми, на котором представлены все имеющиеся данные, но в более абстрактном по сравнению с внутренним уровнем виде;
- *внутренний уровень* – уровень наиболее близкий к физическому хранению.

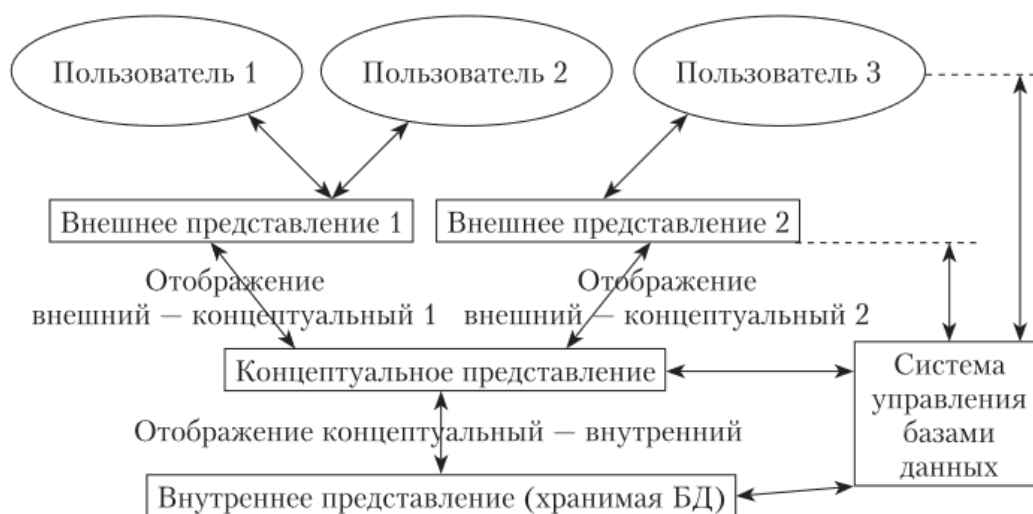


Рисунок 1. Схематичное представление архитектуры ANSI/SPARC

Введем несколько определений:

- *Хранимое поле* – наименьшая единица хранения данных. БД содержит экземпляры каждого из нескольких типов хранимых полей.
- *Хранимая запись* – набор связанных хранимых полей.
- *Хранимый файл* – набор всех экземпляров хранимых записей одного типа.

Внешний уровень. Индивидуальный уровень пользователя называется внешним уровнем. Пользователи могут относиться к различным группам: прикладные программисты; конечные пользователи; администраторы (они работают также с внутренним и с концептуальным уровнями). У каждого пользователя есть свой собственный язык общения с СУБД. У конечного пользователя это может быть язык запросов или специальный язык, основанный на формах и меню. Для прикладных программистов им может быть один из языков высокого уровня. Все эти языки включают подязык данных, т.е. подмножество операторов базового языка, связанное с объектами и операциями БД. Наиболее распространенный подобный язык – SQL, он поддерживается большинством систем реляционного типа.

Любой язык данных является комбинацией, по крайней мере, двух подчиненных языков – *языка определения данных* (англ. data definition language, DDL), который поддерживает определение и объявление объектов БД, и *языка обработки данных* (англ. data manipulation language, DML), который поддерживает операции с объектами БД, их обработку.

Отдельного пользователя обычно интересует только некоторая часть всей БД. Кроме того, пользовательское представление этих данных может существенно отличаться от того, как они хранятся. В соответствии с терминологией ANSI/SPARC представление отдельного пользователя называется *внешним представлением*. *Внешнее представление* – это содержимое БД, каким его видят определенные конечные пользователи или группа пользователей.

Внешних представлений обычно бывает несколько. Например, пользователь из отдела кадров может рассматривать БД как набор записей об отделах и служащих. Он может ничего не знать про записи о деталях и поставщиках, с которыми работают пользователи в отделе обеспечения.

В общем случае, внешнее представление состоит из множества экземпляров внешних записей, которые могут не совпадать с хранимыми записями.

Концептуальный уровень состоит из одного представления. *Концептуальное представление* – это представление всей информации БД в несколько абстрактной форме (как и в случае внешнего представления) по сравнению с физическим способом хранения данных.

Концептуальное представление состоит из множества экземпляров каждого типа концептуальной записи. Концептуальная запись не обязательно должна совпадать с внешней записью, с одной стороны, и с хранимой записью – с другой.

Концептуальное представление – это представление всего содержимого БД, а *концептуальная схема* – это определение такого представления. Определения в концептуальной схеме могут включать определения многих дополнительных средств, таких как средства безопасности или правила для обеспечения целостности. Администратор данных, определяя, какие характеристики предметной области требуется сохранять в системе, фактически определяет основу концептуальной схемы.

Внутренний уровень. Так же, как и концептуальный, внутренний уровень состоит из одного представления. *Внутреннее представление* описывает все подробности, связанные с хранением данных в базе. Он состоит из экземпляров каждого типа внутренней записи. Термин «внутренняя запись» принадлежит терминологии ANSI/SPARC и фактически соответствует хранимой записи. Внутреннее представление, так же как внешнее и

концептуальное, не связано с аппаратным уровнем и не включает подробности, связанные с размещением данных на дисках, таких как номер сектора и др.

Внутреннее представление описывается с помощью внутренней схемы, которая определяет типы хранимых записей, индексы (служебные структуры, упрощающие поиск данных), способы представления хранимых полей, физическую последовательность хранимых записей и др.

Отображения. В представленной архитектуре присутствуют отображения двух уровней.

Отображение *концептуального уровня на внутренний* определяет соответствие между концептуальным представлением и хранимой БД. При изменении структур хранения изменяется и отображение «концептуальный - внутренний» таким образом, чтобы концептуальная схема оставалась неизменной. Это обеспечивает *физическую независимость данных*.

Отображение внешнего уровня на концептуальный определяет соответствие между внешними представлениями и концептуальным. Например, несколько концептуальных полей «индекс», «город», «улица», «дом» для пользователей могут быть объединены в одно внешнее поле «адрес». Появляется возможность менять отдельные внешние представления, дополнительно изменяя только отображения и не затрагивая остальные уровни системы. Отделение внешнего уровня от концептуального обеспечивает *логическую независимость данных*.

Определения представлений каждого из уровней и отображения СУБД должна хранить вместе с прочей метаинформацией и использовать их при обработке запросов.

Архитектура ANSI/SPARC имеет большое теоретическое значение, определяя пути обеспечения логической и физической независимости данных. Однако два уровня отображения приводят к дополнительным расходам при обработке запросов пользователя, поэтому разработчики СУБД, стараясь увеличить быстродействие систем, обычно отходят от строгой реализации этой архитектуры.

Архитектура многопользовательских систем баз данных

Рассмотрим аспекты архитектуры систем БД, связанные с обеспечением совместной работы пользователей. В несколько абстрактной форме можно представить многопользовательскую систему БД как совокупность двух компонентов – создающего запросы клиента и сервера, который выполняет приходящие запросы. В частном случае, и клиентское, и серверное приложения могут выполняться на одном и том же компьютере. Но они могут

находиться и на разных компьютерах, связанных телекоммуникационной сетью. В зависимости от разделения функций между клиентом и сервером выделяются несколько типов архитектур систем БД.

Файл-серверная архитектура. Некоторые СУБД могут работать только в режиме «файл-сервер». Это означает, что СУБД находится на клиентской рабочей станции и даже может быть скомпонована с прикладным ПО. Для осуществления совместного доступа к данным файлы БД размещаются на файловом сервере (см. рис. 2).

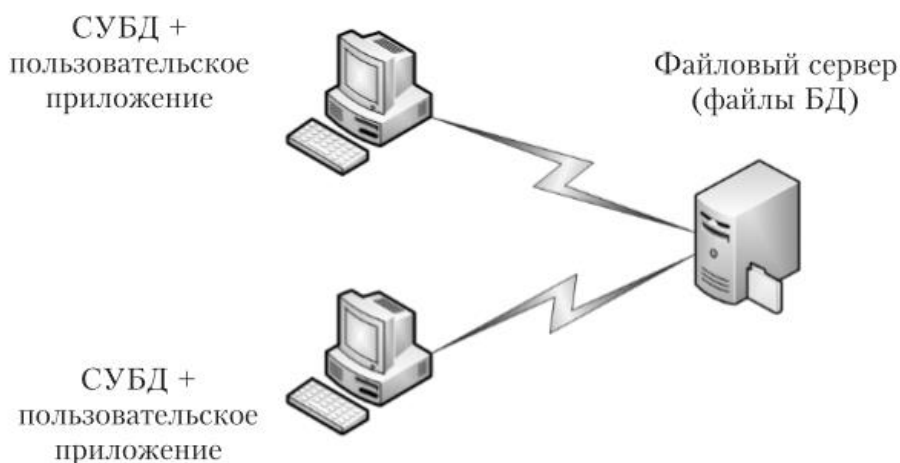


Рисунок 2. Файл-серверная архитектура

Если работа с данными ведется на нескольких компьютерах, то на каждом из них будет запущен экземпляр СУБД. Для обработки задания пользователя с сервера запрашиваются файлы с данными, а их обработка производится локально. В результате в таких системах приходится передавать по сети много данных для того, чтобы на стороне клиента среди них найти те, которые удовлетворяют запросу.

Основным достоинством подобной архитектуры является простота: для того чтобы из локального приложения сделать многопользовательское сетевое приложение следует просто переместить файлы БД на файловый сервер. При этом СУБД должна обеспечить минимально необходимый набор функциональности в области совместного использования файлов БД, например, блокировку изменения записей.

Недостатков у такой архитектуры существенно больше:

1. **Неэффективная загрузка сети передачи данных:** по сети передается во много раз больше данных, чем это необходимо приложению, например, если таблица имеет 100 000 записей, из которых нужны только 10, а передается вся таблица для обработки на клиентский компьютер. Помимо загрузки сети это существенно увеличивает время

выполнения операций по сравнению с обработкой базы, хранящейся локально.

2. **Проблемы при синхронизации совместной работы пользователей:** для поддержания информации об изменяемых в данный момент записях файл-серверные СУБД могут создавать на сервере файлы блокировок или другие подобные структуры, если один из клиентов заблокировал часть записей (СУБД внесла информацию в файл блокировок, что эти записи будут изменяться и их нельзя использовать другим клиентам), после чего потерял связь с сервером, экземпляры СУБД на других клиентах не смогут использовать соответствующие фрагменты БД, пока эту проблему не решит администратор, откорректировав файл блокировок.
3. **Дополнительные требования к производительности клиентских рабочих мест:** все обработка данных производится на клиенте.

Примером многопользовательской системы с файл-серверной архитектурой является совместное использование БД Microsoft Access в локальной сети. В таком режиме с одной базой могут нормально работать несколько пользователей: в зависимости от размеров базы, характеристик сети и интенсивности работы с данными, это может быть 5-10 одновременных сеансов.

Двухзвенная архитектура «клиент-сервер». Данная архитектура предполагает, что СУБД находится на сервере и только она имеет доступ к файлам БД (см. рис. 3).

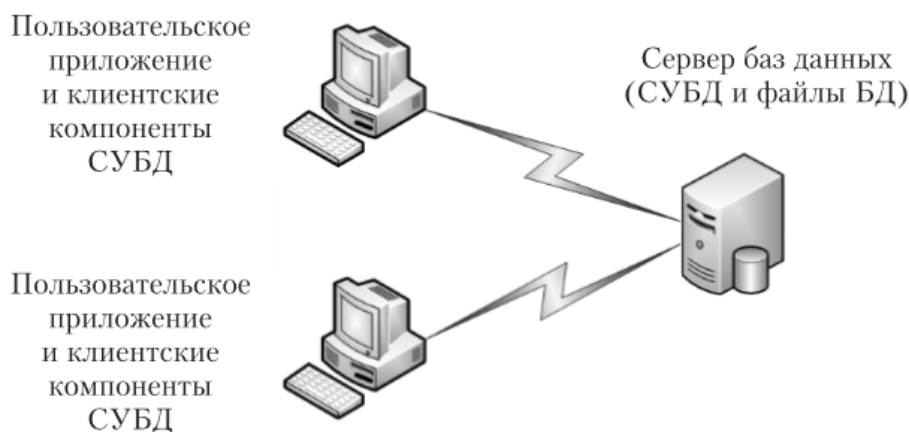


Рисунок 3. Двухзвенная архитектура «клиент-сервер»

На клиентских компьютерах работают пользовательские приложения и клиентские компоненты СУБД, осуществляющие взаимодействие с сервером. От клиента на сервер приходят запросы, которые обрабатываются СУБД, и результат отправляется на клиентский компьютер. По сравнению с файл-серверной архитектурой в этом случае минимизируется сетевой трафик, т.к. по сети передаются только затребованные данные.

Централизованная работа с файлами БД позволяет более эффективно решать вопросы, связанные с совместным доступом к данным и с обеспечением безопасности. В связи с тем, что СУБД работает на сервере, снижаются требования к оборудованию на стороне клиента. Прикладная программа может быть написана на любом языке, который поддерживает взаимодействие с используемой СУБД через имеющиеся клиентские компоненты: могут использоваться такие технологии, как ODBC, ADO, ADO.Net, JDBC др.

Двухзвенная архитектура широко используется для создания информационных систем, но у нее есть ряд недостатков:

1. **«Логика» приложения, вынесенная на сторону клиента:** при большом количестве и географической удаленности клиентских рабочих мест возникают проблемы с обновлением и устранением ошибок в приложении.
2. **Каждый клиент независимо от других в любой момент времени может открыть соединение с СУБД:** сервер поддерживает открытые соединения со всеми активными клиентами, даже если никакой работы нет. При большом числе клиентов это может негативно влиять на производительность сервера БД.

Трехзвенная архитектура. Исправить недостатки двухзвенной архитектуры позволяет трехзвенная, также называемая трехуровневой (см. рис. 4).



Рисунок 4. Трехзвенная архитектура

Данная архитектура предполагает наличие дополнительного сервера приложений, который проводит предварительную обработку запросов клиентов, формирует запросы к серверу БД и обрабатывает полученные результаты перед отправкой их клиенту.

В трехзвенной архитектуре большая часть логики приложения перенесена с клиента на сервер, и задачи клиентского приложения сводятся, в основном, к реализации пользовательского интерфейса и представлению результатов. Появляется возможность централизованного внесения изменений в алгоритмы

бизнес-логики, реализованные в ПО сервера приложений. Также в этой архитектуре только сервер приложений может подключаться к серверу БД. Это снимает проблему поддержания неиспользуемых соединений и более предпочтительно с точки зрения безопасности. Недостатком многоуровневой архитектуры «клиент-сервер» является сложность разработки подобных решений.

Архитектура интернет/интранет-решений. В этом случае обязательным компонентом серверной части является веб-сервер, на котором выполняется веб-приложение, обращающееся к СУБД для доступа к БД (см. рис. 5). В роли универсального клиентского приложения выступает интернет-браузер на устройстве клиента (в данном случае это может быть персональный компьютер, планшетный компьютер или мобильное устройство).

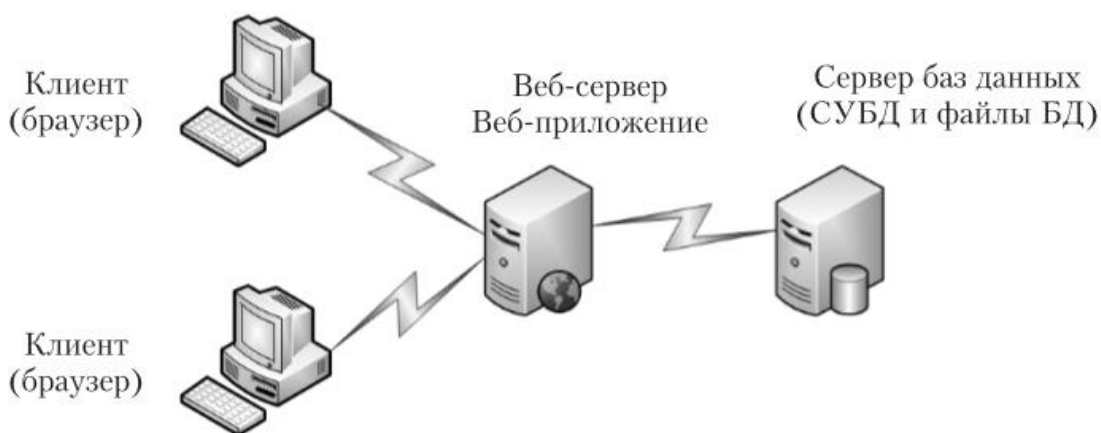


Рисунок 5. Архитектура интернет/интранет-решений

В небольших решениях веб-сервер может находиться на одном физическом компьютере с СУБД. В крупных системах, испытывающих большие нагрузки, задействуется множество веб-серверов и серверов БД.

Достоинства и недостатки подобной архитектуры связаны с использованием браузера в качестве универсального клиента. С одной стороны, отпадает необходимость в разработке, распространении и поддержке клиентского приложения. Появляется универсальность и частичная независимость от клиентской платформы. С другой стороны, в случае ограниченного числа пользователей приложения с помощью специально разработанного клиентского ПО можно добиться большей функциональности по сравнению с использованием веб-технологий.

Задания к лекции

1. Изучите презентацию, рассматривающую архитектуру ANSI/SPARC <https://www.slideshare.net/DamianGordon1/startrek-ansisparc>, и приведите пример внешней, концептуальной и внутренней схем.

2. Опишите алгоритм обработки запроса пользователя к базе данных при централизованной архитектуре. Изобразите графически централизованную архитектуру [4].
3. Опишите алгоритм обработки запроса пользователя к базе данных в случае файл-серверной архитектуры.
4. Опишите алгоритм запроса пользователя к базе данных в случае двухзвенной архитектуры «клиент-сервер».
5. Опишите алгоритм запроса пользователя к базе данных в случае трехзвенной архитектуры.
6. Приведите примеры использования интернет/интранет архитектуры.

Лекция 3. Модели данных и модели базы данных

План лекции:

1. Введение.
2. Иерархическая модель данных.
3. Сетевая модель данных.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – М. : Издательство Юрайт, 2017. – 230 с. – Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).
4. Карпова, Т. Лекция 4: Реляционная модель данных [электронный ресурс] // URL: <http://www.intuit.ru/studies/courses/1001/297/lecture/7405>
5. Database fundamentals / N.Sharma, R.F. Chong [et al.]/ URL: http://public.dhe.ibm.com/software/dw/db2/express-c/wiki/Database_fundamentals.pdf

Введение

Выделяют три уровня моделей БД: инфологическая; даталогическая; физическая.

Инфологическая модель отражает информацию о предметной области без ориентации на конкретную СУБД (или даже на тип предлагаемой к использованию СУБД). В связи с этим некоторые авторы [1] говорят о существовании *инфологической модели предметной области*, а не БД.

Даталогическая модель БД – модель логического уровня, представляющая собой отображение логических связей между элементами данных независимо от их содержания и среды хранения. Эта модель строится в терминах информационных единиц, допустимых в той СУБД, в среде которой будет создаваться БД. Этап создания данной модели называется *даталогическим* или *логическим проектированием*.

Физическая модель БД строится с учетом возможностей по организации и хранению данных, предоставляемых конкретной СУБД и используемой программно-аппаратной платформой. Она, в частности, определяет

используемые запоминающие устройства и способы организации данных в среде хранения.

При проектировании БД первой строится инфологическая модель, после чего – даталогическая, и только после нее – физическая.

Рассмотрим модели данных. Каждая СУБД реализует ту или иную *модель данных*, которая определяет правила порождения допустимых для системы видов структур данных, возможные операции над такими структурами, классы представимых средствами системы ограничений целостности данных. Таким образом, модель данных задает границы множества всех конкретных БД, которые могут быть созданы средствами этой системы.

Разные авторы предлагают несколько различающиеся перечни существующих моделей данных. Например, в [5, с.27-28] предлагается следующий список моделей данных и периодов времени, когда в их разработке были получены основные результаты:

- иерархическая (*англ. hierarchical*), конец 1960-х и 1970-е гг.;
- сетевая (*англ. network*), 1970-е гг.;
- реляционная (*англ. relational*), 1970-е и начало 1980-х гг.;
- «сущность-связь» (*англ. entity – relationship*), 1970-е гг.;
- семантическая (*англ. semantic*), конец 1970-х и 1980-е гг.;
- расширенная реляционная (*англ. extended relational*), 1980-е гг.;
- объектно-ориентированная (*англ. object-oriented*), конец 1980-х – начало 1990-х гг.;
- объектно-реляционная (*англ. object-relational*), конец 1980-х – начало 1990-х гг.;
- полуструктурированная (*англ. semi-structured*) с конца 1990-х гг. и до настоящего времени.

Первыми появились модели данных, основанные на теории графов, - *иерархическая и сетевая*.

Следующей появилась разработанная Э. Коддом (Edgar Codd) *реляционная модель данных*, основанная на математической теории множеств. На сегодняшний день она является самой распространенной.

Модель «сущность-связь» (ER, *entity – relationship*) была предложена П. Ченом (Peter Chen) в 1976 г. в качестве унифицированного способа описания предметной области. Чен предложил думать о БД как о наборе экземпляров сущностей. Сущности – это объекты, которые существуют независимо от любых других объектов в БД. У сущностей есть атрибуты, которые являются элементами данных. Атрибуты характеризуют сущности. Один или несколько из этих атрибутов могут быть обозначены как ключ.

Наконец, между сущностями могут быть связи: 1-to-1, 1-to-n, n-to-1 или m-to-n. Связи могут также иметь атрибуты, описывающие отношения между сущностями. Как самостоятельная модель «сущность-связь» развития не получила, но осталась основой для создания инфологических моделей БД. На рис. 1 приведен пример ER-диаграммы.

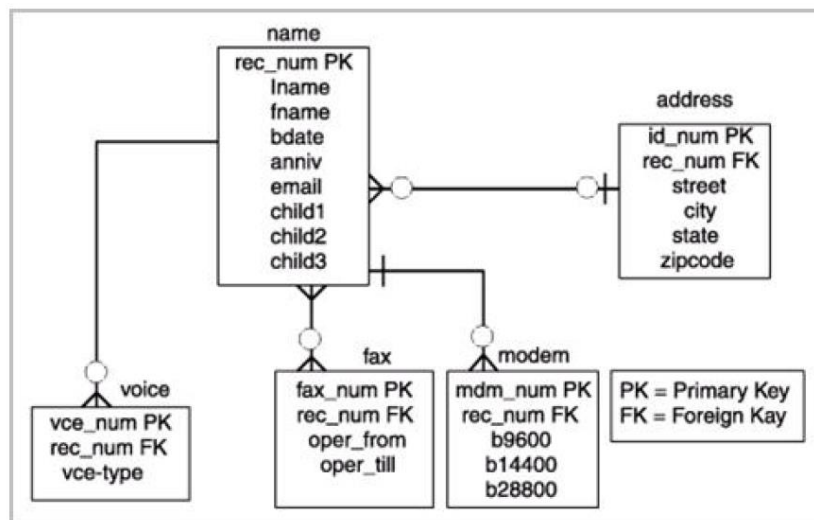


Рисунок 1. ER-диаграмма

В последнее десятилетие наблюдается значительная работа над полуструктурированными, семантическими и объектно-ориентированными моделями данных.

К *полуструктурированным* относятся данные, в которых можно выделить некоторую структуру, но она недостаточно строгая по сравнению с реляционными структурами данных (или структурами других традиционных моделей). Наиболее ярким примером полуструктурированных данных являются XML-документы (от *англ.* eXtensible Markup Language – расширяемый язык разметки). Действительный (*англ.* valid) XML-документ должен соответствовать определенному формату описания (схеме), где заданы структура документа, допустимые названия элементов, атрибутов и др. Формат XML широко используется для обмена данными между приложениями, его поддержка обеспечивается многими СУБД.

Объектно-ориентированные модели данных пользуются популярностью в университетах, но не получили широкого распространения в отрасли. Однако доступны средства объектно-реляционного сопоставления (ORM), которые позволяют интегрировать объектно-ориентированные программы с реляционными БД.

Иерархическая модель данных

Исторически первыми появились СУБД, реализующие иерархическую модель данных: первая коммерческая СУБД IBM IMS (*англ.* Information Management

System, система управления информацией) относится к этому типу. В иерархической системе данные организованы в наборы древовидных структур (иерархий). Основными информационными единицами являются *поле*, *сегмент (запись)*, *связь*, *БД*.

Поле данных (или просто «поле») – минимальная именованная единица данных, доступная пользователю с помощью СУБД.

Сегмент или запись составляют основную единицу обработки БД: записи запоминаются, извлекаются, удаляются. Определяют тип и экземпляр записи (сегмента). *Тип записи* – это именованная совокупность полей данных с указанием их типов, *экземпляр записи* (или просто запись) – некоторая совокупность значений элементов в последовательности, соответствующей определению типа. Иными словами, тип записи задает все множество подобных объектов, а экземпляр – конкретный объект из этого множества. Например, тип записи – стол, тогда экземпляр – это конкретный стол, описываемый набором характеристик (цвет, размер, материал и др.). Для того чтобы однозначно различать записи, каждый тип записи должен иметь ключ – набор полей, однозначно идентифицирующих экземпляр записи. Например, в записи, описывающей человека, таким ключом может быть номер паспорта.

Связь (англ. link) – иерархическое отношение между записями двух типов. Связь при графическом отображении обозначается дугой графа, запись – вершиной графа. Такое изображение структуры БД называется диаграммой Бахмана (американский ученый Charles Bachman). Дуги графа, обозначающие связь, являются направленными. Запись, из которой выходит стрелка, является логически исходной, в которую приходит стрелка, – логически подчиненной.

Тип связи определяется ее именем и задает свойства, общие для всех экземпляров связи данного типа. *Экземпляр связи* задается *исходной записью* («владельцем») и множеством (возможно пустым) *подчиненных записей*. Таким образом, каждой подчиненной записи в иерархической модели может соответствовать только одна исходная, одной исходной записи может соответствовать несколько подчиненных.

В иерархической модели записи и связи между ними создают древовидные структуры (деревья). В каждом дереве существует только одна запись, которая не связана ни с какой исходной записью, – она называется *корневой*. Таким образом, дерево – совокупность корневой записи и множества подчиненных записей.

Схема иерархической БД представляет собой совокупность отдельных деревьев, каждое дерево в рамках модели называется *физической базой данных*.

Каждая физическая БД удовлетворяет следующим иерархическим ограничениям:

- в каждой физической БД существует только одна корневая запись, т.е. запись, у которой нет логически исходной (родительской) записи;
- каждая логически исходная запись может быть связана с произвольным числом логически подчиненных записей;
- каждая логически подчиненная запись может быть связана только с одной логически исходной (родительской) записью.

Набор всех экземпляров записей, подчиненных одному экземпляру корневой записи, называется *физической записью*.

Экземпляры-потомки одного типа, связанные с одним экземпляром записи-предка, называются *близнецами*. В примере на рис. 2 (б) записи близнецы типа «Начальник отдела» - Петров и Сидорова, типа «Служащий» - Николаев и Васильев.

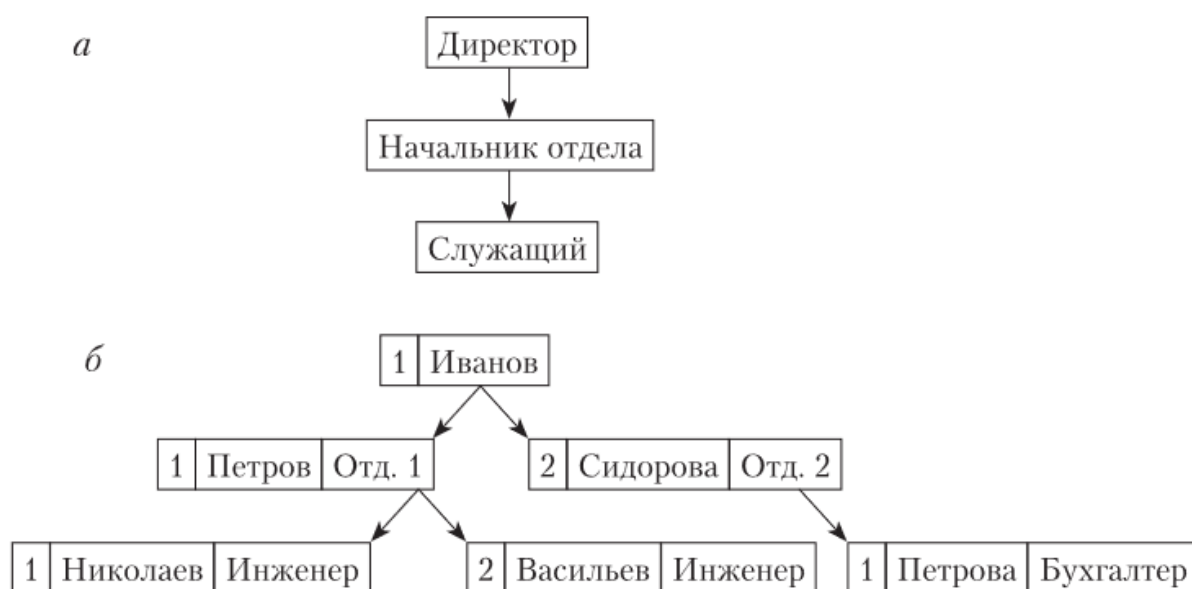


Рисунок 2. Иерархические базы данных:

а – пример структуры дерева; б – пример отдельной физической записи

Чтобы записи могли быть однозначно идентифицированы, применяется следующая схема:

- корневая запись каждого дерева обязательно содержит уникальный ключ с уникальными значениями;
- ключи некорневых записей должны быть уникальны только среди «близнецов»;
- каждая запись идентифицируется полным составным ключом, под которым понимается совокупность ключей записей, начиная от корневой и далее вниз по иерархическому пути до искомой записи.

Для связей данных в иерархической модели обеспечивают *автоматический режим включения* и *фиксированное членство*. Это означает, что для запоминания любой некорневой записи в БД должна существовать соответствующая ей исходная запись. Подчиненная запись жестко закрепляется за исходной, а при удалении исходной записи автоматически удаляются все подчиненные ей.

Основной единицей работы в иерархической модели является запись, над ней могут производиться следующие операции:

- *добавить* (англ. Insert) – позволяет занести в БД новую запись (с учетом ограничений, связанных с требованием относительной уникальности ключа в рамках всей БД для корневой записи и среди «близнецов» - для подчиненных);
- *обновить* (англ. Update) – дает возможность изменять значения данных предварительно извлеченной записи. Ключевые данные записи не могут подвергаться обновлению, в подобных случаях старая запись должна быть удалена, а новая, содержащая измененные данные, - запомнена;
- *удалить* (англ. Delete) – служит для исключения из БД некоторой записи и всех подчиненных ей;
- *извлечь* (англ. Get) – имеет несколько модификаций: извлечь по значению ключа; извлечь следующую запись (в порядке левостороннего обхода дерева); извлечь следующую, удовлетворяющую некоторому условию и др.

Следует отметить, что обработка в любом случае начинается с корневой записи, доступ к некорневой записи осуществляется по иерархическому пути.

Сетевая модель данных

Сетевая модель данных, как и иерархическая, относится к разряду графовых, но она позволяет строить структуры данных, описываемые графом более общего вида, чем предполагает иерархическая модель. Важное влияние на создание и развитие сетевой модели данных оказал проект по разработке СУБД IDS (англ. Integrated Data Store, интегрированное хранилище данных), осуществлявшийся в 1960-х гг. в корпорации General Electric под руководством Ч. Бахмана. Другой известный проект – СУБД IDMS (англ. Integrated Database Management System, интегрированная система управления базами данных), разработка которой началась в 1970-х гг. В настоящее время эта СУБД для мейнфреймов продолжает развиваться, права на нее принадлежат CA Technologies.

Базовые структуры этой модели: *элемент данных*; *агрегат данных*; *запись* (или группа), *набор* (групповое отношение), *БД*.

Элемент данных (или просто «элемент») – минимальная именованная единица данных, доступная пользователям с помощью СУБД.

Агрегат данных – именованная совокупность элементов или других агрегатов данных. Разница между элементом и агрегатом может быть проиллюстрирована следующим примером: пусть в БД вносятся адреса, если разработчик рассматривает адрес как единое целое (и соответствующим образом проектирует БД), то адрес – это элемент данных, если же необходимо разделить адрес на части (страна – город – улица – номер дома – номер квартиры), то адрес уже будет выступать как агрегат, состоящий из соответствующих элементов. При этом пользователь может запросить из БД отдельно город или номер дома, а может запросить адрес целиком, т.к. агрегат – тоже именованный объект.

Различаются агрегаты типов «*вектор*» и «*повторяющаяся группа*». Агрегат, состоящий из простых элементов данных, называется вектором. Примеров агрегата типа «вектор» может служить упомянутый выше адрес, представимый как совокупность элементов. Агрегат, повторяющийся компонент которого представлен совокупностью данных, называется повторяющейся группой. В повторяющуюся группу могут входить элементы данных и другие агрегаты. В качестве примера здесь можно привести агрегат «Оценки студента», в котором будет сохраняться название предмета и оценка, причем столько раз, сколько экзаменов сдаст студент.

Запись – это агрегат, который не входит в состав другого агрегата, обычно описывает некоторый объект реального мира и составляет основную единицу обработки в БД (записи запоминаются, извлекаются, удаляются).

Набор в сетевой модели является иерархическим отношением между двумя типами записей, экземпляр подчиненной записи не может быть участником двух экземпляров набора одного типа. В сетевой модели один и тот же тип записи может участвовать в нескольких наборах. В частности, для любых двух типов записей может быть задано любое количество наборов, которые их связывают. Наличие подобных возможностей позволяет моделировать отношение объектов типа «многие-ко-многим», что выгодно отличает сетевую модель данных от иерархической.

В то же время, ни в иерархической, ни в сетевой модели один и тот же тип записи не может быть одновременно и владельцем, и членом группового отношения (набора).

На рис. 3 с помощью диаграммы Бахмана проиллюстрировано различие между групповыми отношениями иерархической и сетевой модели данных.

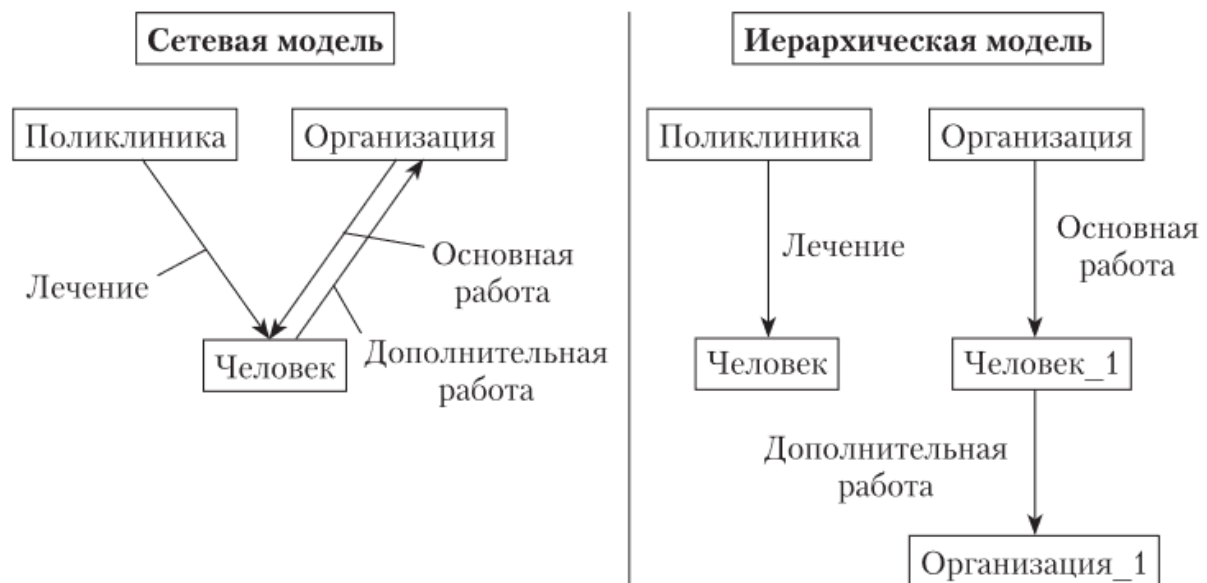


Рисунок 3. Особенности групповых отношений в сетевой и иерархической моделях данных

На диаграмме отображены отношения между типами записей «Человек», «Организация», «Поликлиника». В рамках сетевой модели можно описать, что между типами записей «Человек» и «Организация» имеется связь «многие-ко-многим»: один человек может работать в нескольких организациях, и в одной организации могут работать несколько человек. В иерархической модели в аналогичной ситуации придется ввести два типа записей «Организация» и «Организация_1» и частично продублировать значения.

В сетевой модели отдельно выделяется так называемый *сингулярный набор*, владельцем которого формально определена вся система. Сингулярный набор изображается в виде входящей стрелки, которая имеет имя набора, имя члена набора, но у которой не определен тип записи «владелец набора». Сингулярные наборы позволяют обеспечить произвольный доступ к некоторому типу записи.

Каждый набор характеризуется следующими свойствами:

- способом упорядочения подчиненных записей;
- режимом включения подчиненных записей;
- режимом исключения подчиненных записей.

Каждый *экземпляр набора* можно рассматривать как список записей-членов, поставленных в соответствие некоторой записи-владельцу. Записи-члены могут быть упорядочены в таком списке следующими способами:

- произвольным;
- хронологическим – в последовательности их поступления (очередь);

- обратным хронологическим (стек);
- сортированном – в подчиненной записи выделяется ключ упорядочения, а место новой записи в списке определяется значением этого ключа.

Режим включения записей в набор может быть *автоматическим* или *ручным*. При автоматическом включении подчиненная запись включается в набор одновременно с ее запоминанием в БД. Ручное включение позволяет запомнить в БД подчиненную запись и не включать ее немедленно в экземпляр набора. Эта операция выполняется позднее по команде пользователя.

Выделяют три класса членства подчиненных записей в наборах: *фиксированное*; *обязательное*; *необязательное*. Тип членства определяет и режим исключения записей из набора.

При *фиксированном членстве* подчиненная запись жестко закрепляется за записью-владельцем. Она не может существовать без этого владельца. Подчиненную запись можно исключить из экземпляра группового отношения, только удалив ее из БД. При удалении владельца автоматически удаляются все подчиненные записи.

Обязательное членство означает, что каждая подчиненная запись, будучи однажды включена в групповое отношение, впредь всегда будет связана с некоторой записью-владельцем. Допускается переключение подчиненной записи к другому владельцу, но не допускается ее существование без владельца. Для успешного удаления записи-владельца необходимо, чтобы она не имела подчиненных записей с обязательным членством.

Необязательное членство позволяет исключить подчиненную запись из экземпляра группового отношения, но сохранить в БД, не прикрепляя к другому владельцу. При удалении записи-владельца подчиненные записи сохраняются в БД, не участвуя в соответствующем наборе.

В общем случае сетевая БД представляет собой совокупность взаимосвязанных наборов, которые образуют на концептуальном уровне некоторый граф.

Сетевая модель определяет следующие операции над записями:

- запомнить (*англ. Store*) – позволяет внести в БД новую запись;
- удалить (*англ. Erase*) – удаляет текущий экземпляр записи;
- включить в набор (*англ. Connect*) – связывает существующую подчиненную запись с записью-владельцем;
- исключить из набора (*англ. Disconnect*) – исключает текущую запись из текущего экземпляра набора;

- найти (*англ. Find*) – всегда можно найти запись по значению первичного ключа. Также можно найти запись, используя групповые отношения, в которых она участвует;
- извлечь (*англ. Get*) – позволяет извлечь запись (после того, как она найдена);
- обновить (*англ. Modify*) – позволяет изменить значения элементов существующих в БД записей.

Вопросы и задания к лекции

1. Перечислите и охарактеризуйте три уровня моделей БД.
2. Назовите основные модели данных.
3. В чем отличие между реляционной и расширенной реляционной моделями данных?
4. Какие основные структуры данных определены в иерархической модели данных?
5. Какие операции над данными предусматриваются иерархической моделью данных?
6. Чем в сетевой модели данных агрегат типа «вектор» отличается от агрегата типа «повторяющаяся группа»?
7. Какие типы членства записи в наборе допускает сетевая модель?
8. Перечислите операции, определенные в сетевой модели данных?
9. В чем принципиальное отличие между иерархической и сетевой моделями данных?
10. Сравните между собой операции в иерархической и сетевой моделях данных.

Лекция 4. Реляционная модель данных

План лекции:

1. Допустимые информационные структуры.
2. Ограничения целостности данных.
3. Реляционная алгебра.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. — М. : Издательство Юрайт, 2017. — 230 с. — Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).
4. Карпова, Т. Лекция 4: Реляционная модель данных [электронный ресурс] // URL: <http://www.intuit.ru/studies/courses/1001/297/lecture/7405>

Допустимые информационные структуры

В реляционной модели данных при рассмотрении структур данных вводится ряд специфических понятий.

Наименьшую единицу данных называют *скаляром*. Скалярные значения логически неделимы. Например, значение «Номер группы» в таблице, описывающей студентов, хотя номер группы можно разложить на отдельные символы, но при этом потеряется смысл.

Домен можно определить, как именованное множество скалярных значений одного типа, он является совокупностью значений, из которых берутся значения атрибутов. Каждый атрибут должен быть определён на единственном домене. Значение доменов в реляционной модели состоит в том, что они ограничивают сравнения, т.е. имеет смысл сравнивать только значения атрибутов, определённые в одном домене. Бессмысленно, например, сравнивать вес детали А и размер детали В, чтобы формально определить это, вводятся домены.

В реальной БД домены не сохраняются в виде набора значений. Но формально реляционная модель требует, чтобы домены были определены, а при определении атрибута был указан соответствующий домен.

Переменная отношения – именованный объект, сопоставляемый реляционному отношению, значение которого может изменяться со временем. Значение этой переменной в конкретный момент называется значением отношения.

Отношение R , определённое на множестве доменов D_1, D_2, \dots, D_m (не обязательно различных), содержит две части – заголовок и тело.

Заголовок реляционного отношения, также называемый схемой отношения, содержит фиксированное множество пар вида $\langle \text{имя_атрибута} : \text{имя_домена} \rangle$. Например: $\{ \langle A_1 : D_1 \rangle, \dots, \langle A_n : D_m \rangle \}$, где A_i – уникальное имя атрибута, определённого на одном из доменов. Несколько атрибутов могут быть определены на одном домене, поэтому $n \geq m$.

Тело реляционного отношения содержит множество кортежей, которое может со временем изменяться. Каждый кортеж содержит множество пар вида $\langle \text{имя_атрибута} : \text{значение_атрибута} \rangle$, например: $\{ \langle A_1 : V_{1l} \rangle, \dots, \langle A_n : V_{im} \rangle \}$, где $V_{ij} \in D_j$.

Кардинальное число – это число кортежей в отношении. *Степень отношения* – число атрибутов (в нашем примере – число n).

Для примера ниже представлено отношение «Студенты» (см. табл. 1).

Таблица 1. Отношение «Студенты»

Номер билета	ФИО	Группа
928012	Иванов И.И.	3082/4
928189	Петров П.П.	3082/4

В отношении «Студенты» кардинальное число равно 2, степень – 3. Также подразумевается, что введено три домена – домен номеров студенческих билетов, домен фамилий с инициалами, домен номеров групп. Каждый домен содержит все возможные значения соответствующего атрибута, часть из которых может не использоваться в отношении.

Фундаментальные отличия реляционного отношения от таблицы связаны с тем, что отношение – это множество, в котором:

- не может быть одинаковых элементов (кортежей);
- кортежи не упорядочены (а строки в таблице упорядочены сверху вниз);
- атрибуты не упорядочены (столбцы в таблице упорядочены слева направо);
- все значения атрибутов атомарные.

Выделяют следующие типы реляционных отношения: *именованное*; *базовое*; *производное*.

Именованное отношение – это отношение с именем, т.е. такое, для которого создана переменная отношения.

Базовое отношение – это именованное отношение, которое является автономным. На практике это означает, что разработчик сделал его непосредственной частью БД, в отличие от временных отношений.

Производное отношение определяется через именованные отношения и, в конечном счёте, через базовые отношения.

Выражаемым называется отношение, которое можно получить из набора именованных отношений посредством некоторого реляционного выражения. Множество выражаемых отношений – это множество всех базовых и всех производных отношений.

Представление (англ. view) – именованное производное отношение. Представления виртуальны, они существуют в системе исключительно через определение в терминах именованных отношений.

Снимок (англ. snapshot) – это именованное производное отношение, которое, в отличие от представления, реально, т.е. представлено не только с помощью определения в терминах других именованных отношений, но и своими отдельными данными. Создание снимка похоже на выполнение запроса, за исключением того, что полученные результаты сохраняются в БД под определённым именем (и могут обновляться с заданной периодичностью).

Результат запроса – именованное производное отношение. Результаты запросов не сохраняются в БД.

Промежуточный результат – неименованное производное отношение, являющееся результатом некоторого реляционного выражения, вложенного в другое, большее выражение.

Хранимое отношение – отношение, непосредственно поддерживаемое в физической памяти.

Можно определить *реляционную БД* как БД, воспринимаемую пользователем, как набор реляционных отношений разной степени.

Ограничения целостности данных

Говоря о целостности данных, будем исходить из следующих предположений:

- 1) в любой момент времени БД содержит определённый набор значений данных;

- 2) просто набор значений данных не имеет смысла, если он не отражает действительность, т.е. не является представлением части реального мира.

Ограничения целостности могут быть специфическими для конкретной БД. Например, группа, в которую входит студент, должна быть из определённого списка существующих групп. Однако в реляционной модели существуют и общие правила обеспечения целостности, описываемые потенциальными и внешними ключами.

Пусть R – реляционное отношение. Тогда *потенциальным ключом* (обозначим его K) будет называться подмножество атрибутов R , обладающее следующими свойствами:

- 1) *уникальность* – в отношении R не может быть двух различных кортежей с одинаковым значением K ;
- 2) *неизбыточность* – никакое из подмножеств K , отличных от него, не обладает свойством уникальности.

Потенциальный ключ, состоящий из одного атрибута, называется *простым*. Например, в отношении, представленном в табл. 1, будет простой потенциальный ключ, состоящий из атрибута «Номер билета» (считаем, что номер студенческого билета уникален). Потенциальный ключ, состоящий из нескольких атрибутов, называется *составным*.

Потенциальные ключи важны из-за того, что они обеспечивают основной механизм адресации на уровне кортежей. Иначе говоря, единственный гарантируемый системой способ указать на кортеж – указать значение его потенциального ключа или одного из таких ключей, если их несколько.

В реляционной модели принято один из потенциальных ключей выбирать в качестве *первичного ключа*. Остальные потенциальные ключи будут *альтернативными ключами*. С точки зрения реляционной модели, какой из ключей выбрать в качестве первичного – не принципиально. Однако многие СУБД при хранении данных упорядочивают записи по значению первичного ключа. В этом случае важно, чтобы операция сравнения значений первичного ключа выполнялась максимально быстро. Поэтому предпочтительнее будет использовать простой ключ с целочисленными значениями. Если среди имеющихся атрибутов подходящего на такую роль нет, нередко специально для создания первичного ключа вводят атрибут, которому присваиваются значения целочисленного счётчика. Подобный первичный ключ называют *суррогатным*, в противоположность естественному ключу, который формируется из атрибутов, характеризующих реальный объект.

Некоторые авторы вводят понятие *вторичного ключа* – такого подмножества атрибутов, которое существенным образом характеризует кортеж, но не обладает уникальностью. В табл. 1 это может быть поле «ФИО». В таком понимании вторичный ключ не является ограничением целостности, но при проектировании БД он может учитываться путём создания неуникального индекса.

Отношений в БД может быть достаточно много, причём некоторые из них могут быть связаны друг с другом. При операциях с ними большое значение имеют внешние ключи.

Пусть R_1 – базовое отношение. Подмножество атрибутов FK отношения R_1 будет называться *внешним ключом*, если выполняются следующие условия:

- 1) существует базовое отношение R (причём R_1 и R не обязательно различны) с потенциальным ключом K ;
- 2) каждое значение FK в текущем значении отношения R_1 всегда совпадает со значениями ключа K некоторого кортежа в текущем значении отношения R .

Внешний ключ будет простым тогда и только тогда, когда соответствующий потенциальный ключ простой. Аналогично для составного внешнего ключа. Каждый атрибут, входящий в данный ключ, должен быть определён на том же домене, что и соответствующий атрибут потенциального ключа.

Проблема обеспечения того, что БД не включает неверных значений внешних ключей, называется проблемой обеспечения *ссылочной целостности*. Отношение, содержащее внешний ключ, называется *ссылающимся отношением*. Отношение, содержащее соответствующий потенциальный ключ, называется *ссылочным отношением*.

Правило ссылочной целостности заключается в том, что БД не должна содержать несогласованных значений внешних ключей. При реализации данного правила необходимо ответить, как минимум, на два вопроса:

1. Что должно случиться при попытке удалить объект ссылки внешнего ключа?
2. Что должно случиться при попытке обновить потенциальный ключ, на который ссылается внешний ключ?

В первом случае можно ограничить действие – запретить операцию удаления до того момента, пока существуют ссылающиеся значения внешних ключей. Другой вариант – выполнить каскадное удаление, т.е. удалить кортеж с соответствующим значением потенциального ключа и все кортежи со ссылающимися на него внешними ключами. Аналогично можно поступить и

во втором случае – ограничить обновление или выполнять каскадное изменение.

На практике следует аккуратно использовать каскадное удаление записей. Единичная ошибка пользователя в этом случае может нанести большой вред, т.к. вместе с удаляемой записью можно удалить множество записей в связанных отношениях.

Реляционная алгебра

При разработке реляционной модели Э. Кодд ввёл реляционную алгебру, которая состоит из набора операторов, использующих отношения в качестве операндов в возвращающих отношениях в качестве результата. Она включает восемь операций:

- традиционные операции над множествами – объединение, пересечение, вычитание, декартово произведение;
- специальные реляционные операции – выборку, проекцию, соединение, деление.

Говоря о реляционной алгебре, нельзя обойти вниманием *свойство замкнутости*. Оно заключается в том, что результат реляционной операции над отношением также является отношением. Следовательно, результат одной операции может использоваться в качестве исходных данных для другой. Таким образом, можно использовать вложенные выражения.

Там, где это возможно, выполняется правило наследования имен атрибутов: соответствующие атрибуты результата будут называться так же, как у исходных отношений. В некоторых случаях будет выполняться наследование потенциальных ключей.

Введем еще одно определение: два реляционных отношения *совместимы по типу*, если:

- 1) каждое из них имеет одно и то же множество атрибутов;
- 2) соответствующие атрибуты (т.е. атрибуты с одинаковыми именами) определены на одном и том же домене.

Объединение. Отношение с тем же заголовком, что у исходных, и с телом, состоящим из множества всех кортежей, принадлежащих А, или В, или им обоим, называется объединением двух совместимых по типу отношений А и В.

Поскольку в отношении невозможно наличие двух одинаковых кортежей, операция объединения может сопровождаться удалением дубликатов. Это произойдёт в том случае, если в объединяемых отношениях А и В встречаются полностью совпадающие кортежи: во множестве кортежей,

составляющих тело нового отношения, не может быть полностью совпадающих элементов.

Степень результата будет равна степени исходных отношений, а кардинальное число – не больше, чем сумма кардинальных чисел исходных отношений.

Для записи объединения используют, как правило, одно из двух обозначений: $A \cup B$; $A \text{ UNION } B$.

Пересечение. Отношение с тем же заголовком, что у исходных, и телом, состоящим из множества всех кортежей, принадлежащих обоим отношениям одновременно, называется пересечением двух совместимых по типу отношений A и B . Операция записывается следующим образом: $A \cap B$; $A \text{ INTERSECT } B$. Кардинальное число результата пересечения будет не больше, чем наименьшее из кардинальных чисел отношений A и B , степень – равна степеням исходных отношений.

Вычитание. Отношение с тем же заголовком, что у исходных, и телом, состоящим из множества всех кортежей, принадлежащих A и не принадлежащих B , называется вычитанием двух совместимых по типу отношений A и B . Формы записи: $A \setminus B$; $A \text{ MINUS } B$.

Кардинальное число результата будет не больше кардинального числа A , степень – равна степеням исходных отношений.

Поясним введенные определения на примерах. Пусть отношения A и B заданы, как представлено в табл. 2 и 3 соответственно. Подчеркиванием обозначено, что атрибут *StudID* является первичным ключом. Тогда результаты выполнения объединения, пересечения и вычитания отношений A и B представлены в табл. 4-6 соответственно.

Таблица 2. Отношение A

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382

Таблица 3. Отношение B

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
127	Сидоров С.С.	383

Таблица 4. Объединение $A \cup B$

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383

Таблица 5. Пересечение $A \cap B$

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382

Таблица 6. Вычитание $A \setminus B$

<u>StudID</u>	ФИО	Группа
124	Петров П.П.	382

Необходимо обратить внимание на следующие моменты:

- 1). Наследование ключей и имён атрибутов позволило определить заголовок результирующего отношения.
- 2). Встречающий в обоих отношениях кортеж $\langle 123, \text{Иванов И.И.}, 382 \rangle$ в результате объединения присутствует в единственном экземпляре, в результате пересечения – является единственным кортежем в теле отношения, в результате вычитания – отсутствует.

Декартово произведение двух отношения A и B (где A и B не имеют общих имён атрибутов) определяется как отношение с заголовком, который представляет собой объединение («сцепление») двух заголовков исходных отношения A и B , и телом, состоящим из множества T всех кортежей, таких, что кортеж $t \in T$ представляет собой сцепление кортежа $a \in A$ и кортежа $b \in B$. Кардинальное число результата равно произведению кардинальных чисел исходных отношения, а степень равна сумме их степеней. Формы записи: $A \times B$; $A \text{ TIMES } B$.

Рассмотрим пример декартова произведения отношений A и B , представленных в табл. 7-8 соответственно.

Таблица 7. Отношение A

<u>StudID</u>	ФИО
123	Иванов И.И.
124	Петров П.П.
127	Сидоров С.С.

Таблица 8. Отношение B

<u>Предмет</u>
Математика
Физика

Декартово произведение $A \times B$ представлено в табл. 9.

Таблица 9. Декартово произведение $A \times B$

<u>StudID</u>	ФИО	<u>Предмет</u>
123	Иванов И.И.	Математика
123	Иванов И.И.	Физика
124	Петров П.П.	Математика
124	Петров П.П.	Физика
127	Сидоров С.С.	Математика
127	Сидоров С.С.	Физика

Можно убедиться, что степень произведения равна сумме степеней исходных отношений ($2+1$), кардинальное число равно произведению кардинальных чисел исходных отношений (3×2). В результирующем отношении получен составной первичный ключ $\{StudID, Предмет\}$, тогда как в исходных отношениях первичные ключи были простыми.

Стоит обратить внимание на следующие особенности этой операции:

- 1). Если в отношениях A и B оказались одинаковые атрибуты, чтобы различать их в результирующем отношении, можно условиться формировать имя такого атрибута в виде $\langle \text{Имя отношения} \rangle . \langle \text{Имя атрибута} \rangle$. Например, $A.FIO$ и $B.FIO$.
- 2). Если необходимо декартово произведение отношения с собой, то следует использовать ещё одну переменную отношения, иначе не удастся правильно сформировать названия атрибутов результат: $A1 = A$, $R = A \times A1$. Здесь « $=$ » обозначает оператор присваивания.

Выборка – это сокращённое название θ -выборки, где θ (тета) обозначает любой скалярный оператор сравнения ($=, <, >, \leq, \geq, \neq$). θ -выборкой из

отношения A по атрибутам X и Y (в этом порядке) называется отношение, имеющее тот же заголовок, что и отношение A , и тело, содержащее множество T всех кортежей из отношения A , для которых проверка условия « $X \theta Y$ » дает значение «истина». Формы записи: $A [X \theta Y]$; $A \text{ WHERE } X \theta Y$.

Для операции выборки необходимо, чтобы атрибуты X и Y были определены на одном домене, а операция θ должна иметь смысл для данного домена. Константа может быть указана как вместо одного из атрибутов, так и вместо обоих.

Рассмотрим пример. В табл. 10 представлено отношение *STUDENTS*, а в табл. 11 результат выборки по условию (Группа = 382).

Таблица 10. Отношение *STUDENTS*

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383

Таблица 11. Выборка *STUDENTS* (Группа=382)

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382

Расширим определение операции θ -выборки. Будем считать, что условие для θ -выборки может содержать произвольное число простых сравнений, соединенных логическими связками. При необходимости могут использоваться круглые скобки.

Для логических связок также используются альтернативные формы записи, одна из которых ближе к принятой в математической логике, другая – к форме записи в языках запросов к БД:

- «не» может записываться как \neg или *NOT*;
- «и» - $\&$, \wedge или *AND*;
- «или» - \vee или *OR*.

Теперь можно построить, например, такое выражение:

STUDENTS WHERE ((Группа = 382) AND (StudID < 124))

Результатом станет отношение, содержащее в теле только кортеж $\langle 123, \text{Иванов И. И.}, 382 \rangle$.

Проекцией отношения A по атрибутам V, W, \dots, Z (где каждый из атрибутов принадлежит отношению A) называется отношение с заголовком $\{V, W, \dots, Z\}$ и телом, содержащим множество всех кортежей $\{V:v, W:w, \dots, Z:z\}$, для которых в отношении A существует кортеж со значение атрибута V , равным v , атрибута W – w , ... и атрибута Z – z . Таким образом, в результате проекции часть атрибутов исходного отношения исключается, после чего может потребоваться удаление повторяющихся кортежей.

Проекция отношения A по атрибутам V, W, Z обозначается как $A[V, W, Z]$.

Например, несколько изменим отношение *STUDENTS* (см. табл. 12) и сделаем проекцию по атрибуту «ФИО» (см. табл. 13).

Таблица 12. Измененное отношение *STUDENTS*

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383
128	Иванов И.И.	384

Таблица 13. Проекция *STUDENTS* [ФИО]

<u>ФИО</u>
Иванов И.И.
Петров П.П.
Сидоров С.С.

Как можно заметить, после отбрасывания атрибутов *StudID* и «Группа» был получен дубликат кортежа $\langle \text{«Иванов И.И.»} \rangle$, который в результате оставлен в одном экземпляре. Подобные ситуации могут возникать, если во множество атрибутов, на которое делается проекция, не вошёл ни один потенциальный ключ исходного отношения.

Соединение. Определены две разновидности операции соединения: *естественное соединение* и *θ -соединение*.

Пусть отношения A и B имеют заголовки $\{X, Y\}$ и $\{Y, Z\}$, где атрибуты X, Y, Z могут быть составными. Иными словами, только атрибут Y (или подмножество атрибутов, обозначаемое Y) присутствует как в первом, так и во втором отношении. Атрибуты с одинаковыми именами определены на

одних доменах. Тогда *естественным соединением* отношения A и B называется отношение с заголовком $\{X, Y, Z\}$ и телом, содержащим все кортежи $\{X: x, Y: y, Z: z\}$, для которых в отношении A значение атрибута X равно x , значение атрибута Y – y , а в отношении B значение атрибута Y равно y , а атрибута Z – z . Обозначается естественное соединение A и B как $A \text{ JOIN } B$. Если отношения A и B не имеют общих имён атрибутов, то их естественное соединение эквивалентно декартовому произведению.

В табл. 14-16 приведён пример естественного соединения двух реляционных отношений.

Таблица 14. Отношение STUDENTS

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383

Таблица 15. Отношение GROUPS

Староста	<u>Группа</u>
Петров П.П.	382
Сергеев С.С.	384

Таблица 16. Естественное соединение STUDENTS JOIN GROUPS

<u>StudID</u>	ФИО	Группа	Староста
123	Иванов И.И.	382	Петров П.П.
124	Петров П.П.	382	Петров П.П.

Из-за того, что в первом отношении (см. табл. 14) нет кортежей со значением атрибута «Группа», равным «384», а во втором отношении (см. табл. 15) нет кортежей со значением того же атрибута «383», в результирующем отношении остались только кортежи, содержащие данные о студентах группы «382».

Из требований уникальности и избыточности для первичного ключа следует, что в результирующем отношении первичный ключ – $\{StudID\}$.

Пусть отношения A и B не имеют общих атрибутов, и оператор θ определяется также, как в выборке (т.е. оператор сравнения). Тогда θ -соединением отношения A по атрибуту X с отношением B по атрибуту Y называется отношение с тем же заголовком, что и при декартовом произведении отношений A и B , и с телом, содержащим множество T всех

кортежей, таких, что любой кортеж $t \in T$ принадлежит декартовому произведению A и B , и вычисление для него условия « $X \theta Y$ » дает значение «истина».

Как и в случае выборки, X и Y должны быть определены на одном и том же домене, а операция θ должна иметь смысл для этого домена.

На самом деле, θ -соединение эквивалентно нахождению декартова произведения двух отношений и последующей θ -выборке, произведённой над результатом.

Могут использоваться два варианта записи этой операции: $A [X \theta Y] B$; $(A \text{ TIMES } B) \text{ WHERE } X \theta Y$.

Рассмотрим пример. Пусть отношение A содержит информацию о цене приборов (см. табл. 17), а отношение B – о цене подставок для приборов (см. табл. 18). Необходимо получить те комбинации приборов, подставок и цен на них, где подставка не дороже прибора. Это можно сделать с помощью операции θ -соединения по условию «Цена_прибора \geq Цена_подставки».

Таблица 17. Отношение A

<u>Прибор</u>	<u>Цена_прибора</u>
Прибор1	100
Прибор2	200

Таблица 18. Отношение B

<u>Подставка</u>	<u>Цена_подставки</u>
Подставка1	50
Подставка2	150

Таблица 19. θ -соединение $A [Цена_прибора \geq Цена_подставки] B$

<u>Прибор</u>	<u>Цена_прибора</u>	<u>Подставка</u>	<u>Цена_подставки</u>
Прибор1	100	Подставка1	50
Прибор2	200	Подставка1	50
Прибор2	200	Подставка2	150

Деление. Пусть отношения A и B имеют заголовки $\{X, Y\}$ и $\{Y\}$ соответственно, атрибуты X и Y могут быть составными. Одинаково названные атрибуты двух отношений определены на одних и тех же доменах.

Результатом деления отношения A на B будет отношение с заголовком $\{X\}$ и телом, содержащим множество всех кортежей $\{X:x\}$, таких, что в A существует кортеж $\{X:x, Y:y\}$, для **всех** кортежей $\{Y:y\}$ из B . Обозначается эта операция как A/B или $A \text{ DEVIDEBY } B$.

В табл. 20 приведено отношения A , содержащее информация о том, какие предметы сдавали студенты. Отношение B содержит список предметов (см. табл. 21). Необходимо получить список идентификаторов студентов, сдававших все перечисленные в таблице B предметы.

Задача решается делением A на B .

Таблица 20. Отношение A

<u>StudID</u>	<u>Предмет</u>
123	Физика
123	Математика
124	Математика
127	Физика

Таблица 21. Отношение B

<u>Предмет</u>
Физика
Математика

Таблица 22. Деление A/B

<u>StudID</u>
123

Рассмотрим пример задачи, для решения которой используется несколько реляционных операций. Нужно написать выражение, в результате выполнения которого из отношений $STUDENTS$ (см. табл. 14) и $GROUPS$ (см. табл. 15) будет получено отношение, содержащее данные о фамилии и инициалах студента и старосте его группы.

Искомое выражение: $(STUDENTS \text{ JOIN } GROUPS) [\text{ФИО}, \text{Староста}]$

Пусть имеется реляционное отношение, которое содержит следующие данные о книжных изданиях (см. табл. 23):

- 1) ID – внутренний идентификатор издания (первичный ключ);

- 2) *Title* – название;
- 3) *Author* – фамилия и инициалы автора (полагаем, что к одной книге только один автор, и среди авторов однофамильцев с совпадающими инициалами нет);
- 4) *Publisher* – издательство;
- 5) *Year* – год издания.

Рассмотрим несколько задач, иллюстрирующих применение операторов реляционной алгебры.

Таблица 23. Отношение *Book*

<u>ID</u>	Title	Author	Publisher	Year

Пусть нужно найти книги, изданные после 1999 г., причём интересует вся информация о книге. Эта задача решается с помощью выборки:

$$Res = Book[Year > 1999]$$

Если требуется перечень всех издательств, о которых имеется упоминание в БД, то будем использовать проекцию:

$$Res = Book[Publisher]$$

Пусть необходимо получить список авторов, издававшихся во всех издательствах. Просто список издательств был получен в предыдущем примере, аналогично с помощью проекции можно получить список сочетаний «автор» - «издательство», а деление даст искомый результат:

$$Res = (Book[Author, Publisher]) / (Book[Publisher])$$

Чтобы получить список авторов, издававшихся более одного раза в одном и том же году, понадобится θ -соединение отношения самого с собой. Для этого сначала введем еще одну переменную для отношения *Book*, потом выполним соединение, проверив, чтобы автор и год совпадали, а издания были различны, и в конце выполним проекцию:

$$B = Book$$

$$Res = (B[B.ID \neq Book.ID \ \& \ B.Author = Book.Author \ \& \ B.Year = Book.Year]Book)[B.Author]$$

Следующая задача – получить список авторов, чьи книги ни разу не издавались в издательстве «Азбука». Часто при решении такой задачи делают ошибку, используя выборку по условию *Publisher* \neq "Азбука". Но это условие, с последующей проекцией, позволит выбрать авторов, которые хотя бы раз издавались в издательстве, отличном от «Азбуки», т.е. в общем

случае такое решение неверно. Правильное решение будет получено, если из множества всех авторов вычесть тех, кто издавался в «Азбуке»:

$$Res1 = Book[Author]$$

$$Res2 = (Book[Publisher = \text{Азбука}])[Author]$$

$$Res = Res1 \setminus Res2$$

Вопросы и задания к лекции

1. Как в реляционной модели данных определяется понятие «отношение»? В чем отличие между реляционным отношением и таблицей?
2. Что такое потенциальный ключ? Поясните разницу между естественным и суррогатным ключами.
3. Какие реляционные отношения являются совместимыми по типу?
4. Перечислите и дайте определение реляционных операций, входящих в алгебру Кодда.
5. В табл. 24 и 25 представлены реляционные отношения *St* и *Res*, первичные ключи выделены подчёркиванием. Первое из отношений содержит информацию о студентах – номер студенческого билета (атрибут *StudID*), фамилию и инициалы (атрибут *FIO*), номер группы (атрибут *Group*). В отношении *Res* содержатся данные о номере студенческого билета (*StudID*, внешний ключ), названии учебного курса (атрибут *Subject*), оценке за экзамен (атрибут *Result*).

Таблица 24. Отношение *St*

<u>StudID</u>	FIO	Group

Таблица 25. Отношение *Res*

<u>StudID</u>	<u>Subject</u>	Result

Напишите выражения реляционной алгебры, позволяющие получить следующие данные:

- 1) список студентов (номер студенческого билета и фамилию с инициалами) группы 328;
- 2) список студентов (номер студенческого билета и фамилию с инициалами), сдавших экзамен по физике на «пять»;
- 3) список дисциплин, по которым все студенты, сдававшие эту дисциплину, получили только пятерки;

- 4) список студентов (номер студенческого билета и фамилию с инициалами), сдавших все упомянутые в Res учебные дисциплины на «пять».

Лекция 5. Нормализация реляционных базы данных

План лекции:

1. Основные понятия.
2. Первая нормальная форма (1НФ).
3. Вторая нормальная форма (2НФ).
4. Третья нормальная форма (3НФ).
5. Нормальная форма Бойса-Кодда (НФБК).
6. Четвертая нормальная форма (4НФ).
7. Пятая нормальная форма (5НФ).
8. Доменно-ключевая нормальная форма (ДКНФ). Денормализация.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – М. : Издательство Юрайт, 2017. – 230 с. – Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).

Основные понятия

Рассмотрим подходы к логическому проектированию реляционной БД. На этапе логического проектирования разрабатывается логическая структура БД: определяются реляционные отношения, их атрибуты, потенциальные и внешние ключи. Ошибки, сделанные при разработке структуры БД, могут привести к нежелательным эффектам при сохранении, удалении и изменении данных, называемым *аномалиями модификации*. Среди аномалий модификации выделяют аномалии удаления, аномалии вставки, аномалии обновления.

Аномалия удаления проявляется в том, что, удаляя факты, относящиеся к одной сущности, мы непроизвольно удаляем факты, относящиеся к другой сущности.

Аномалия вставки приводит к тому, что нельзя поместить в БД некоторый факт об одной сущности, не указав дополнительно некоторый факт о другой.

Аномалия обновления заключается в том, что для внесения изменения в данные об одном факте необходимо выполнить множественные изменения в БД.

Одним из существенных преимуществ реляционной модели является наличие формального механизма оценки качества логической структуры БД средствами теории нормализации.

Нормализация – это процесс приведения структуры реляционных отношений к форме, обладающей лучшими свойствами при добавлении, изменении и удалении данных.

Окончательная цель нормализации сводится к получению такого проекта БД, в котором каждый факт проявляется лишь в одном месте, т.е. исключена избыточность информации. Кроме задачи более эффективного использования памяти, нормализация позволяет снизить угрозы нарушения целостности БД из-за появления в ней внутренних противоречий.

Вводится понятие *нормальных форм* (НФ) отношений, каждой из которых соответствует определенный набор ограничений. Отношение находится в данной нормальной форме, если удовлетворяет указанным ограничениям. Каждая следующая НФ включает в себя требования всех предыдущих, т.е. является более строгим ограничением.

Введем определения.

Пусть R – реляционное отношение, а X и Y – некоторые подмножества атрибутов этого отношения. Y *функционально зависит* от X тогда и только тогда, когда для каждого значения множества X существует только одно значение множества Y . Иначе говоря, если два кортежа отношения совпадают по значению X , то они обязательно будут совпадать и по значению Y . Записывается *функциональная зависимость* (ФЗ) как $X \rightarrow Y$, читается как « X функционально определяет Y ». Если существует ФЗ $X \rightarrow Y$, то X называют детерминантом, а Y – зависимой частью.

Из определения ФЗ, в частности, следует, что любое подмножество атрибутов отношения функционально зависит от любого из потенциальных ключей.

Рассмотрим пример. Пусть отношение STUDENTS имеет структуру, представленную в табл. 1, и атрибут StudID является первичным ключом. В представленном отношении есть зависимости:

- $\{\text{StudID}\} \rightarrow \{\text{ФИО}\}$
- $\{\text{StudID}, \text{ФИО}\} \rightarrow \{\text{Группа}\}$
- $\{\text{StudID}, \text{ФИО}\} \rightarrow \{\text{StudID}\}$ и др.

Таблица 1. Отношение STUDENTS

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383

В то же время подмножество атрибутов {Группа} в общем случае не является функционально зависимым от {ФИО}, т.к. в списке студентов могут оказаться однофамильцы с одинаковыми инициалами, но обучающиеся в разных группах.

Как видно из примера, множество ФЗ, даже для отношения с небольшой степенью, может быть достаточно велико. В то же время ФЗ является ограничением целостности, и ее выполнение должно проверяться СУБД при таких операциях, как добавление, удаление или изменение данных. Поэтому желательно исключить из рассмотрения такие ФЗ, которые можно вывести из других ФЗ.

Функциональная зависимость называется *тривиальной*, если ее зависимая часть является подмножеством детерминанта. В рассмотренном примере $\{\text{StudID}, \text{ФИО}\} \rightarrow \{\text{StudID}\}$ – это тривиальная зависимость.

Функциональная зависимость называется *неприводимой*, если выполняются следующие условия:

- зависимая часть является одноэлементным множеством, т.е. справа находится только один атрибут;
- детерминант является неприводимым, т.е. ни один атрибут из детерминанта не может быть исключен без потери ФЗ.

Существуют правила вывода новых ФЗ на основе уже имеющихся. По имени исследователя, впервые их опубликовавшего, их часто называют правилами Армстронга:

- *рефлексивность* – если множество атрибутов B является подмножеством A , то $A \rightarrow B$;
- *дополнение* – если $A \rightarrow B$, то $A \cup C \rightarrow B \cup C$;
- *транзитивность* – если $A \rightarrow B$ и $B \rightarrow C$, то $A \rightarrow C$.

На базе правил Армстронга был выведен ряд других правил:

- *самоопределение* – $A \rightarrow A$;
- *декомпозиция* – если $A \rightarrow B \cup C$, то $A \rightarrow B$ и $A \rightarrow C$;
- *объединение* – если $A \rightarrow B$ и $A \rightarrow C$, то $A \rightarrow B \cup C$;

- *композиция* – если $A \rightarrow B$ и $C \rightarrow D$, то $A \cup C \rightarrow B \cup D$.

Для множества ФЗ S его *замыканием* S^* называется множество всех ФЗ, которые можно вывести из зависимостей S . Множество S всегда будет являться подмножеством своего замыкания.

Первая нормальная форма

Отношение находится в *первой нормальной форме* (1НФ) тогда и только тогда, когда оно содержит только скалярные значения атрибутов и ни один из ключевых атрибутов не имеет значения NULL. *Ключевым* является атрибут, входящий в любой из потенциальных ключей.

NULL – это специальное значение, показывающее, что значение атрибута не определено. Пусть, например, в отношении «Студенты» есть атрибут «Оценка за экзамен». Если оценка студента за какой-то экзамен неизвестна, то соответствующему атрибуту кортежа будет установлено значение NULL. Важно не путать NULL с числом 0.

В табл. 2 приведен пример группированной таблицы, нарушающей требования 1НФ, т.к. значения атрибутов «Предмет» и «Оценка» не являются скалярными.

Таблица 2. Пример группированной таблицы STUD

StudID	ФИО	Группа	Предмет	Оценка
123	Иванов И.И.	382	Математика	4
			Физика	5
124	Петров П.П.	382	Математика	5
			Физика	5
127	Сидоров С.С.	383	Базы данных	4

Чтобы привести табл. 2 к требованиям 1НФ следует выполнить преобразования, приведенные в табл. 3. Обратите внимание, что в табл. 3 первичный ключ является составным {StudID, Предмет}. Хотя данное преобразование и улучшило структуру таблицы, но оно все равно содержит избыточную информацию – повторяющиеся фамилии студентов и номера их групп. Кроме того, данному отношению свойственны *аномалии модификации*. В частности, нельзя внести информацию о студенте, не указав хотя бы один предмет (аномалия вставки) и невозможно удалить информацию обо всех сданных студентом учебных дисциплинах, не удалив информацию о нем самом (аномалия удаления).

Таблица 3. Отношение STUD

<u>StudID</u>	ФИО	Группа	<u>Предмет</u>	Оценка
123	Иванов И.И.	382	Математика	4
123	Иванов И.И.	382	Физика	5
124	Петров П.П.	382	Математика	5
124	Петров П.П.	382	Физика	5
127	Сидоров С.С.	383	Базы данных	4

Вторая нормальная форма

Реляционное отношение находится во *второй нормальной форме (2НФ)*, если оно удовлетворяет определению 1НФ и все его атрибуты, не входящие в первичный ключ, неприводимо зависимы от него.

При описании 2НФ и 3НФ везде, кроме случаев, где это указано явно, предполагается, что реляционные отношения имеют только один потенциальный ключ, который и является первичным.

Отношение, представленное в табл. 3, не находится в 2НФ. Например, существует функциональная зависимость $\{StudID\} \rightarrow \{ФИО\}$. Таким образом, зависимость атрибута «ФИО» от первичного ключа не является неприводимой: первичный ключ $\{StudID, Предмет\}$, а атрибут «Предмет» можно убрать из детерминанта ФЗ $\{StudID, Предмет\} \rightarrow \{ФИО\}$ без потери зависимости.

Можно улучшить структуру отношения, разбив его на два, находящихся в 2НФ. Тут возникает проблема декомпозиции без потерь, т.е. такого разбиения на два, чтобы в результате этой процедуры не произошла потеря информации.

Процесс разбиения отношения $R\{A, B, C\}$ на два отношения $R1\{A, B\}$, $R2\{A, C\}$ называется *проецированием*, а отношения $R1$ и $R2$ – *проекциями*. Здесь A, B, C – это некоторые непересекающиеся подмножества атрибутов исходного отношения, объединение которых даст все множество атрибутов. Если была произведена декомпозиция без потерь, то соединение проекций $R1$ и $R2$ должно дать исходное отношение R .

При проведении декомпозиции часто опираются на теорему Хеза (*англ. I.J. Heath*): Пусть $R\{A, B, C\}$ является реляционным отношением, где A, B, C – атрибуты (возможно, составные) этого отношения. Если R удовлетворяет зависимости $A \rightarrow B$, то R равно соединению своих проекций $\{A, B\}$ и $\{A, C\}$.

Для нормализации представленного в табл. 3 отношения STUD будем использовать ФЗ $\{StudID\} \rightarrow \{ФИО, Группа\}$. Это позволит в соответствии с

теоремой Хеза выполнить декомпозицию без потерь, результаты которой представлены в табл. 4 и табл. 5.

Таблица 4. Отношение STUDENTS

<u>StudID</u>	ФИО	Группа
123	Иванов И.И.	382
124	Петров П.П.	382
127	Сидоров С.С.	383

Таблица 5. Отношение RESULTS

<u>StudID</u>	<u>Предмет</u>	Оценка
123	Математика	4
123	Физика	5
124	Математика	5
124	Физика	5
127	Базы данных	4

Необходимо отметить следующее, если в реляционном отношении только один потенциальный ключ, и он простой, наличие 1НФ автоматически делает отношение соответствующим 2НФ. Таким образом, отношение STUDENTS требованиям 2НФ удовлетворяет. Что касается отношения RESULTS, то в нем есть неприводимая ФЗ $\{\text{StudID, Предмет}\} \rightarrow \{\text{Оценка}\}$, что доказывает соответствие требованиям 2НФ.

Третья нормальная форма

Отношение находится в *третьей нормальной форме* (3РФ), если оно удовлетворяет определению 2НФ и ни один из его неключевых атрибутов не зависит функционально от любого другого неключевого атрибута.

Если существует ФЗ между неключевыми атрибутами, а детерминант этой ФЗ будет, в свою очередь, зависеть от первичного ключа, мы получим транзитивную ФЗ. Иными словами, если в отношении есть только один потенциальный ключ и можно выделить транзитивные ФЗ, то это указывает, что отношение не соответствует 3НФ.

Рассмотрим пример (табл. 6). Реляционное отношение *T1* содержит список сотрудников с указанием их должностей и размеров заработной платы. Первичным ключом является атрибут ID, альтернативных ключей нет, отношение находится во 2НФ. При этом известно, что размер заработной платы сотрудника полностью определяется занимаемой должностью. Иными

словами, существует ФЗ {Должность} → {Зарплата}, что делает отношение не соответствующим требованиям ЗНФ, и в нем есть аномалии модификации.

Таблица 6. Отношение $T1$

<u>ID</u>	ФИО	Должность	Зарплата
1	Иванов И.И.	Инженер	50 000
2	Петров П.П.	Инженер	50 000
3	Сидорова С.С.	Бухгалтер	30 000
4	Васильева В.В.	Бухгалтер	30 000

Воспользуемся теоремой Хеза и выполним декомпозицию, используя приведенную выше ФЗ. Полученные реляционные отношения обозначим $T2$ и $T3$ (см. табл. 7 и табл. 8), в $T2$ должен быть определен внешний ключ {Должность}, ссылающийся на первичный ключ отношения $T3$.

Таблица 7. Отношение $T2$

<u>ID</u>	ФИО	Должность
1	Иванов И.И.	Инженер
2	Петров П.П.	Инженер
3	Сидорова С.С.	Бухгалтер
4	Васильева В.В.	Бухгалтер

Таблица 8. Отношение $T3$

<u>Должность</u>	Зарплата
Инженер	50 000
Бухгалтер	30 000

Легко показать, что требования ЗНФ здесь будут выполняться: в отношении $T2$ между атрибутами «ФИО» и «Должность» функциональных зависимостей нет, а в отношении $T3$, неключевой атрибут – единственный.

Нормальная форма Бойса – Кодда

Представленное выше определение ЗНФ не совсем подходит для анализа структуры отношений, соответствующих перечисленным ниже условиям:

- отношение имеет два или более потенциальных ключей;
- потенциальные ключи являются сложными;
- потенциальные ключи пересекаются (т.е. у сложных ключей есть хотя бы один общий атрибут).

В этом случае обычно используют более «сильную» нормальную форму Бойса – Кодда. Отношение находится в *нормальной форме Бойса – Кодда* тогда и только тогда, когда все детерминанты нетривиальных и неприводимых ФЗ являются потенциальными ключами.

Докажем, что рассмотренные ранее отношения STUDENTS (табл. 4) и RESULTS (табл. 5) соответствуют НФБК. Рассмотрим имеющиеся в них нетривиальные неприводимые ФЗ. Для STUDENTS это будут ФЗ {StudID} → {ФИО} и {StudID} → {Группа}. Для RESULTS: {StudID, Предмет} → {Оценка}. Во всех случаях детерминант является потенциальным ключом, и требования НФБК выполняются.

Четвертая нормальная форма

На практике процесс нормализации часто заканчивается приведением отношений БД в соответствие требованиям НФБК, но в некоторых случаях требуется привести отношение к более старшей НФ.

Рассмотрим пример (табл. 9). Реляционное отношение STUD1 содержит данные о студентах – на какой специальности они учатся и в какой спортивной секции занимаются. Предполагается, что секций и специальностей может быть более одной (параллельно получаемое второе образование и др.), поэтому все три атрибута входят в состав первичного ключа {StudID, Специальность, Спорт}.

Таблица 9. Отношение STUD1

<u>StudID</u>	<u>Специальность</u>	<u>Спорт</u>
123	Информационные системы	Лыжи
123	Менеджмент	Лыжи
123	Информационные системы	Бег
123	Менеджмент	Бег
124	Информационные системы	Плавание

Требования НФБК в данном случае выполняются, но в отношении присутствуют избыточность и аномалии модификации. В частности, невозможно внести данные о специальности студента, не введя данные о спортивной секции (аномалия вставки), так как атрибут «Спорт» является ключом и не может иметь значение NULL.

Введем определение. В отношении $R \{A, B, C\}$ существует *многозначная зависимость* $A \twoheadrightarrow B$ в том и только том случае, если множество значений B , соответствующее паре значений A и C , зависит только от A и не зависит от C .

При этом существует многозначная зависимость $A \twoheadrightarrow C$. Что позволяет ввести запись $A \twoheadrightarrow B|C$.

Если считать, что выбор специальности и спортивной секции определяется только самим студентом, то в табл. 9 присутствуют две многозначные зависимости: $\{StudID\} \twoheadrightarrow \{Специальность\}|\{Спорт\}$.

Ни одна из них не будет являться ФЗ, так как одному студенту может соответствовать несколько специальностей и секций.

Отношение R находится в четвертной нормальной форме (4НФ) тогда и только тогда, когда в случае существования нетривиальной многозначной зависимости $A \twoheadrightarrow B$ все остальные атрибуты R функционально зависят от A .

В рассмотренном примере (табл. 9) отношение не удовлетворяет требованиям 4НФ, но использовать теорему Хеза для проведения декомпозиции не удастся из-за отсутствия подходящих ФЗ: все атрибуты входят в составной ключ и любые ФЗ будут тривиальными. В подобных случаях используется теорема Фейгина (англ. R. Fagin): Пусть A , B и C являются подмножествами атрибутов отношения $R \{A, B, C\}$. Отношение R будет равно соединению своих проекций $\{A, B\}$ и $\{A, C\}$ тогда и только тогда, когда для R выполняется многозначная зависимость $A \twoheadrightarrow B|C$.

Воспользовавшись этой теоремой, проведем нормализацию и получим удовлетворяющие 4НФ отношения STUD2 (табл. 10) и STUD3 (табл. 11), соединение которых даст исходное отношение.

Таблица 10. Отношение STUD2

<u>StudID</u>	<u>Специальность</u>
123	Информационные системы
123	Менеджмент
124	Информационные системы

Таблица 11. Отношение STUD3

<u>StudID</u>	<u>Спорт</u>
123	Льжи
123	Бег
124	Плавание

Пятая нормальная форма

В ряде случаев в реляционных отношениях могут присутствовать более сложные зависимости. Введем определения.

Пусть U, V, \dots, Z – подмножества атрибутов реляционного отношения R . Отношение R удовлетворяет *зависимости соединения*, обозначаемой $*$ (U, V, \dots, Z) , тогда и только тогда, когда R восстанавливается без потерь соединением своих проекций на U, V, \dots, Z .

Реляционное отношение, для которого нельзя провести декомпозицию без потерь на две проекции, но можно провести декомпозицию без потерь на n проекций ($n > 2$), называется *n -декомпозируемым*.

Рассмотрим пример 3-декомпозируемого отношения. Отношение R (табл. 12) содержит информацию о поставках деталей на объекты. Его первичный ключ является составным и включает в себя все три атрибута.

Таблица 12. Отношение R

<u>Поставщик</u>	<u>Деталь</u>	<u>Объект</u>
п1	д1	о2
п1	д2	о1
п2	д1	о1
п1	д1	о1

Проекции R на пары атрибутов {Поставщик, Деталь}, {Поставщик, Объект} и {Деталь, Объект} назовем R_1, R_2, R_3 (табл. 13-15).

Таблица 13. Проекция $R_1 = R[\text{Поставщик, Деталь}]$

<u>Поставщик</u>	<u>Деталь</u>
п1	д1
п1	д2
п2	д1

Таблица 14. $R_2 = R[\text{Поставщик, Объект}]$

<u>Поставщик</u>	<u>Объект</u>
п1	о2
п1	о1
п2	о1

Таблица 15. $R3 = R[\text{Деталь}, \text{Объект}]$

<u>Деталь</u>	<u>Объект</u>
д1	о2
д2	о1
д1	о1

Рассмотрим теперь результаты соединений этих проекций. Результат операции $R1 \text{ JOIN } R2$ не даст исходного отношения, так как в нем будет один дополнительный кортеж (табл. 16, последняя строка, отмеченная стрелкой). И только после соединения с проекцией $R3$ будет получено исходное отношение (табл. 17).

Если отношение R всегда можно будет восстановить из указанных проекций, оно будет 3-декомпозируемым. Также в нем будет зависимость соединения $*$ ($\{\text{Поставщик}, \text{Деталь}\}, \{\text{Поставщик}, \text{Объект}\}, \{\text{Деталь}, \text{Объект}\}$).

Указанная зависимость, в частности, означает, что если поставщик P_n поставляет детали D_m , детали D_m используются на объекте O_k и поставщик P_n обслуживает объект O_k , то он поставляет D_m на O_k .

Таблица 16. Соединение $R4 = R1 \text{ JOIN } R2$

<u>Поставщик</u>	<u>Деталь</u>	<u>Объект</u>
п1	д1	о2
п1	д2	о1
п2	д1	о1
п1	д1	о1
п1	д2	о2

Таблица 17. Соединение $R5 = R4 \text{ JOIN } R3$

<u>Поставщик</u>	<u>Деталь</u>	<u>Объект</u>
п1	д1	о2
п1	д2	о1
п2	д1	о1
п1	д1	о1

Отношение R находится в *пятой нормальной форме (5НФ)* или в *нормальной форме проекции-соединения* в том и только в том случае, когда любая зависимость соединения в R следует из существования некоторого потенциального ключа в R .

Представленное в табл. 12 отношение не находится в 5НФ, и для его нормализации надо выполнить его декомпозицию, как это сделано в примере (табл. 13-15).

Доменно-ключевая нормальная форма. Денормализация

Дальнейшие исследования в области нормализации шли по пути поиска ситуаций, в которых могут проявляться скрытые аномалии, связанные с 5НФ. В 1981 г. Фейгин опубликовал статью, в которой определил *доменно-ключевую нормальную форму* (ДКНФ) и показал, что отношение в ДКНФ не имеет аномалий модификации. Более того, любое отношение, не имеющее аномалий модификации, должно находиться в ДКНФ. После этого в НФ более высоких порядков отпала необходимость, по крайней мере, для устранения аномалий модификации.

Отношение находится в *доменно-ключевой нормальной форме*, если каждое ограничение, накладываемое на это отношение, является логическим следствием определения доменов и ключей. *Ограничение* здесь определяется как любое правило, регулирующее возможные статические значения атрибутов и достаточно точное для того, чтобы можно было установить, выполняется оно или нет. При анализе на соответствие ДКНФ не рассматриваются ограничения, зависящие от времени, например, «зарплата за текущий период не может быть меньше, чем за предыдущий период».

Покажем, что рассмотренное ранее отношение STUDENTS (табл. 4) соответствует ДКНФ. В этом отношении есть ограничение первичного ключа (StudID), и могут присутствовать ограничения на правила формирования идентификаторов студентов, номеров учебных групп, порядок записи имен и фамилий. Все перечисленные ограничения связаны или с ключами, или с определением доменов, таким образом, отношение соответствует ДКНФ. Алгоритм преобразования отношения к ДКНФ автором однозначно не определен, поэтому можно сказать, что ДКНФ показывает, каким требованиям должно отвечать отношение, но не дает ответа, как этого добиться.

В заключении рассмотрим понятие *денормализации*, которое определяется как намеренное приведение структуры реляционных отношений к форме, нарушающей требования НФ. В процессе нормализации реляционные отношения обычно подвергаются декомпозиции, разделяются на несколько. При этом для чтения данных из нескольких отношений приходится проводить операции соединения, которые требуют дополнительного времени. В большинстве случаев потери времени на соединение таблиц относительно невелики, и важнее спроектировать такую структуру БД, в которой не будет аномалий модификации. Но в ряде случаев забота о быстродействии системы вынуждает разработчиков согласиться с некоторой избыточностью хранимых

данных и отсутствием у отдельных реляционных отношений старших нормальных форм. Это можно рассматривать в качестве крайней меры, и решение о денормализации надо принимать осторожно.

Вопросы и задания к лекции

1. В чем заключаются аномалии модификации?
2. Как определяется функциональная зависимость? Приведите пример функциональной зависимости.
3. Какие условия должны выполняться, чтобы реляционное отношение находилось в 1НФ?
4. Как формулируется теорема Хеза?
5. Какой класс ФЗ исключается путем приведения реляционного отношения в соответствие требованиям 3НФ?
6. В чем заключаются требования НФБК?
7. Могут ли быть аномалии модификации у реляционных отношений, находящихся в ДКНФ?
8. Требуется спроектировать БД, в которой все отношения нормализованы до НФБК. Предметная область – учет накопителей на жестких дисках (HDD), используемых в организации. О дисках мы знаем следующее:
 - у жесткого диска есть производитель;
 - у производителя есть веб-сайт (допустим, что только один), где можно получить нужную информацию о диске;
 - для жесткого диска определена модель;
 - модель определяет объем диска (в гигабайтах), скорость вращения, тип используемого интерфейса;
 - по названию модели можно определить производителя;
 - у конкретного экземпляра жесткого диска есть: серийный номер; модель; дата приобретения; дата выхода из строя (если диск вышел из строя, возможность восстановления не рассматривается); комментарии по поводу его работы.

При проектировании БД нужно учитывать, что:

- для любой модели обязательно должен быть указан производитель и объем;
- для любого диска должна быть указана модель;
- фирмы-производители и модели дисков именуются уникальным образом, серийные номера дисков также уникальны.

Лекция 6. Проектирование баз данных, ER-диаграммы

План лекции:

1. Основные понятия.
2. ER-диаграммы в нотации Чена.
3. ER-диаграммы в нотациях Баркера и Мартина.
4. Проектирование баз данных с использованием методологии IDEF1X.
5. CASE-средства. Создание логической и физической моделей базы данных в Toad Data Modeler 6.1 Freeware.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – М. : Издательство Юрайт, 2017. – 230 с. – Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).

Основные понятия

Инфологическое (концептуальное) проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами.

Даталогическое (логическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Описание предметной области в инфологической модели обычно производится в терминах «сущность» – «связь» (*англ. entity-relation, сокр. ER*). Описание модели впервые было опубликовано П. Ченом (*англ. Peter Chen*) в 1976 г. в статье «The Entity-Relationship Model – Towards a Unified View of Data» (модель «сущность-связь» – шаг к единому представлению данных).

Сущность – множество экземпляров реальных или абстрактных однотипных объектов, присутствующих в предметной области. Именованная сущность принята существительным в единственном числе, например, «Рейс», «Заказчик», «Поставщик» и т.д.

Сущности бывают правильными («сильными») и слабыми. *Правильная сущность* соответствует тем объектам, чье существование не зависит от других объектов. *Слабая сущность* находится в зависимости от других сущностей. Например, если заказ должен быть сделан каким-то клиентом, то «Заказ» будет слабой сущностью, зависимой от сущности «Клиент». В этом примере, если не задано дополнительных зависимостей, сущность «Клиент» будет являться правильной.

Сущность имеет набор свойств, называемых *атрибутами*. Каждая группа атрибутов, описывающая одно реальное проявление сущности, представляет собой *экземпляр* (*англ. instance*) сущности.

Чтобы одну сущность можно было отличить от другой, выбирается один атрибут (или несколько атрибутов) с уникальным значением (уникальной комбинацией значений), который называется *идентификатором сущности*. Часто подобные атрибуты называют *ключевыми*.

Атрибуты могут быть:

- 1) *простыми и составными* (составной атрибут может включать в себя набор значений, например, атрибут адрес включает в себя следующий набор значений: индекс; страна; город; улица; дом и др.), при построении инфологической модели составные атрибуты допустимы, при переходе к даталогическим моделям все зависит от выбранной модели данных и возможностей конкретной СУБД, в реляционной модели составных атрибутов нет, их нужно представлять, как множество простых;
- 2) *ключевыми и неключевыми*;
- 3) *обязательными и необязательными* (значения обязательных атрибутов всегда должны быть определены, значения необязательных атрибутов могут быть неопределенными);
- 4) *однозначными и многозначными* (например, в компании может быть только один главный бухгалтер – однозначный атрибут, и может быть много номеров телефонов – многозначный атрибут; при построении

инфологической модели допустимы многозначные атрибуты, но при переходе к даталогической модели от таких атрибутов обычно избавляются).

Атрибуты должны быть заданы на доменах (домен в данном случае – это множество допустимых значений атрибута). На ER-диаграммах обычно домены не указаны. Эта информация помещается в отдельный документ, называемый *словарем данных*. В то же время некоторые варианты нотации Чена допускают указания доменов на диаграммах под изображением атрибута.

Связь при инфологическом проектировании описывает имеющуюся в предметной области логическую связь между двумя или более сущностями. Объекты, которые соединяются, называются *участниками связи*. Число участников определяет *степень связи*. Как и в случае с сущностями, выделяют тип и экземпляр связи. Связь принято именовать с помощью глагола, например, связь между сущностями «Клиент» и «Заказ» можно именовать «Размещает».

Пусть R – тип связи, E – тип сущности. Если каждый экземпляр сущности типа E находится, по крайней мере, в одном экземпляре связи типа R , то участие E в R называется *полным* или *обязательным*, в противном случае – *частичным* или *необязательным*. Например, в предметной области образование выделены сущности «Преподаватель» и «Курс», между которыми существует связь «Читает». Каждый курс читается, хотя бы одним преподавателем, но существуют преподаватели, не читающие ни одного курса, поэтому участие сущности «Курс» в связи «Читает» является полным, а сущности «Преподаватель» – частичным.

Выделяют следующие типы связей:

- 1) один-к-одному;
- 2) один-ко-многим;
- 3) многие-ко-многим.

Приведем формальное определение для данных типов связей.

Пусть имеются две сущности (A и B), участвующие в связи R . Если каждый экземпляр A_i сущности A связан не более чем с одним экземпляром сущности B , а каждый экземпляр B_j сущности B связан не более чем с одним экземпляром сущности A , то связь R имеет тип «один-к-одному».

Связь R имеет тип «один-ко-многим», если экземпляр A_i сущности A может быть связан с нулем, одним или несколькими экземплярами сущности B , а экземпляр B_j сущности B может быть связан не более чем с одним экземпляром

сущности *A*. Обратным по отношению к типу «*один-ко-многим*» является тип связи «*многие-к-одному*».

Связь между сущностями *A* и *B* будет иметь тип «*многие-ко-многим*», если экземпляр A_i сущности *A* может быть связан с нулем, одним или несколькими экземплярами сущности *B* и экземпляр B_j сущности *B* также может быть связан с нулем, одним или несколькими экземплярами сущности *A*. Например, если один курс может читать несколько преподавателей, а один преподаватель может читать несколько курсов, то между сущностями «Преподаватель» и «Курс» есть связь типа «многие-ко-многим». В большинстве случаев в процессе проектирования связи типа «многие-ко-многим» удаляются путем введения составных (ассоциативных) сущностей и определения двух связей типа «один-ко-многим».

Составная (ассоциативная) сущность представляет данные, которые ассоциируются со связью между двумя или более сущностями. Введение данного типа сущностей связано с тем, что связь, по своей сути, не должна обладать собственными свойствами. Но часто случается, что это не так. Например, связь между пассажиром и авиарейсом, которым он летит, описывается набором собственных параметров: номер места; класс; цена; дата приобретения билета и др. Чтобы представить эти данные, можно ввести составную сущность «Билет».

Существует несколько нотаций или правил изображения диаграмм «сущность – связь» (ER-диаграмм).

ER-диаграммы в нотации Чена

В нотации Чена сущности изображаются прямоугольником, внутри которого помещается имя сущности. Прямоугольник, соответствующий слабой сущности, обводится двойной рамкой. Атрибуты изображаются в виде овала, соединенного в соответствующим прямоугольником, но обычно на диаграммах атрибуты вообще не отображаются. Ключевые атрибуты выделяются подчеркиванием или служебным символом (например, #) в начале имени. Связь обозначается ромбом. Ромб окружен двойной линией, если связь задана между слабой сущностью и сущностью, от которой она зависит. Участники связи присоединены к ромбу линией. Для обозначения типа связи используются символы «1» и «М» (иногда вместо «М» применяют символ бесконечности). Двойная линия обозначает полное участи сущности в связи. Ассоциативные сущности изображаются ромбом, заключенным в прямоугольник.

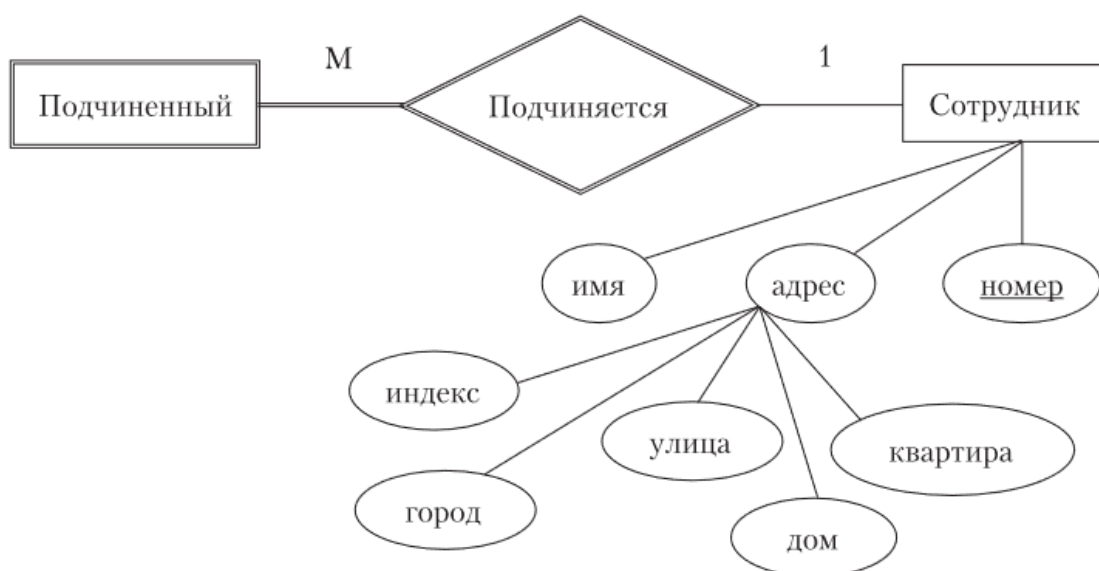


Рисунок 1. ER-диаграмма в нотации Чена

На рис. 1 изображены две сущности «Сотрудник» и «Подчиненный» и задана связь «Подчиняется» типа «один-ко-многим» (одному сотруднику могут подчиняться несколько человек, каждый подчиненный имеет «над собой» только одного прямого начальника). Для сущности «Сотрудник» обозначены ее атрибуты. Здесь атрибут «номер» является ключевым, атрибут «адрес» – составным.

Диаграмма «сущность – связь» в некотором смысле является абстрактным макетом базы данных, поэтому был выработан ряд правил, облегчающих переход от диаграмм к реляционным отношениям:

- 1) каждый правильный (сильный) тип сущности соответствует базовому реляционному отношению;
- 2) каждая бинарная связь типа «многие-ко-многим» соответствует отдельному отношению, которое должно включать в себя два внешних ключа, ссылающихся на потенциальные ключи отношений, соответствующих сущностям – участникам связи;
- 3) связь типа «один-ко-многим» между сильными сущностями может быть представлена с помощью внешнего ключа и не требует отдельного отношения;
- 4) связь слабого объекта с сильным, от которого он зависит, является связью типа «многие-к-одному» и может быть представлена внешним ключом. В некоторых случаях, когда и сильная, и подчиненная ей слабая сущности представлены одним реляционным отношением, внешний ключ может ссылаться на первичный ключ своего же отношения. Так можно поступить в примере «Сотрудник» – «Подчиненный» (рис. 1). Однако в подобных случаях самый главный руководитель станет

«самона начальником» – один и тот же человек будет являться и начальником, и подчиненным;

5) атрибуты сущностей приводятся к атрибутам отношений;

6) в случае n -арной связи ($n > 2$) обычно вводят $n + 1$ отношение: по одному на каждую сущность и одну на связь.

ER-диаграммы в нотациях Баркера и Мартина

В начале 1980-х гг. были предложены новые подходы к инфологическому проектированию БД, в большей степени ориентированные на БД реляционного типа. Среди работавших в этом направлении исследователей можно назвать Р. Баркера (*англ.* Richard Barker) и авторов нотации Information Engineering (сокр. IE) Дж. Мартина (*англ.* James Martin) и К. Финкельштейна (*англ.* Clive Finkelstein).

В предложенных нотациях сущности изображаются сходным образом – в виде прямоугольника, содержащего в заголовке имя сущности, и далее идет перечень атрибутов. Ключевые атрибуты выделяются шрифтом, специальными символами или отделяются чертой от остальных.

Все связи являются бинарными (т.е. только с двумя участниками) и изображаются линией, соединяющей сущности. На рис. 2 представлены правила изображения связей в нотациях Баркера и Мартина.

a

Обозначение	Кардинальность
-----	0,1
_____	1,1
----->	0,N
_____>	1,N

б

Обозначение	Кардинальность
—	нет
—	1,1
—0	0,1
—<	M,N
—0<	0,N
— <	1,N

Рисунок 2. Правила изображения связей: а – нотация Баркера; б – нотация Мартина (IE)

Из-за особенностей изображения связей нотации Баркера и Мартина в литературе иногда называют «crow's foot notation» (дословно – «нотация вороньей лапки»).

На рис. 3 приведен фрагмент диаграммы в нотации Мартина, изображающей две сущности («Клиент» и «Заказ») и связь между ними. Первичные ключи на рисунке выделяются символом «#». Предполагается, что:

- клиент может разместить один, несколько или ни одного заказа (связь от сущности «Клиент» к сущности «Заказ» отмечена —0<);
- заказ может быть размещен одним и только одним клиентом (связь от сущности «Заказ» к сущности «Клиент» отмечена —||).

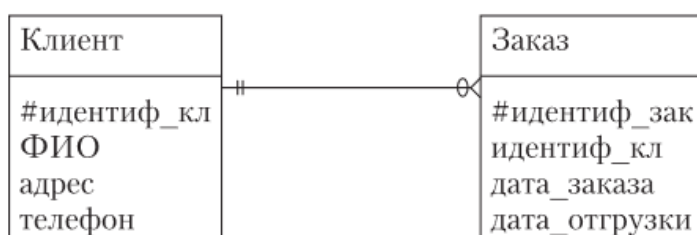


Рисунок 3. ER-диаграмма в нотации Мартина

Проектирование баз данных с использованием методологии IDEF1X

В настоящее время широкое распространение получила нотация, определенная стандартом IDEF1X (*англ.* – Integrated DEFinition).

Стандарт, описывающий методологию проектирования IDEF1X, был разработан в США в 1993 г. IDEF1X (IDEF1 Extended). Он уточняет предшествующую версию IDEF1 и является частью семейства стандартов IDEF.

Методология IDEF1X разрабатывалась для создания информационных моделей, представляющих структуру данных, описывающих предприятие или другую предметную область.

Поддерживающие IDEF1X программные средства позволяют построить независимую от СУБД логическую модель БД и по ней создать физическую модель для реализации в среде конкретной СУБД. При этом появляется возможность уже на ранних этапах проектирования системы согласовать описание предметной области: выделить объекты, их взаимосвязи, атрибуты, описывающие отдельные характеристики. При необходимости по одной логической модели можно создать физические модели для разных СУБД.

На основе физической модели может быть сгенерирован скрипт, выполнение которого в среде выбранной СУБД приведет к созданию необходимых структур БД. Таким образом, при создании физической модели необходимо описать все подробности относительно БД, например, конкретный тип данных, поддерживаемый выбранной СУБД, тогда как при создании логической модели можно ограничиться указанием обобщенного типа (числовой, строковый и др.).

Разделение на логическую и физическую модель БД имеет ряд преимуществ и с точки зрения документирования разрабатываемого решения. В силу ряда причин, таких как накладываемые СУБД ограничения на длину имен таблиц и столбцов, ограниченная поддержка национальных алфавитов в именах объектов БД, многие разработчики предпочитают использовать для таблиц и столбцов короткие имена, отображаемые латиницей. Например, столбец с идентификатором рабочей станции может называться «WstID». Некоторые CASE-средства позволяют в логической модели использовать понятные читаемые названия, а при разработке физической модели изменить их на короткие технические обозначения.

При создании модели БД описываются сущности, их атрибуты и связи между сущностями. Терминология IDEF1X очень близка к терминологии, используемой в других методиках моделирования БД.

Сущность (англ. Entity) является представлением множества реальных или абстрактных объектов (явлений, событий и др.), относимых к одному типу: они обладают одинаковым набором характеристик и могут участвовать в однотипных связях с другими сущностями. Конкретный объект из подобного множества будет называться *экземпляром*. Экземпляры должны отличаться один от другого. Сущности именуются существительными в единственном числе, например, «Компьютер». Также стандарт IDEF1X допускает в имени указания уникального в рамках модели БД номера сущности, например,

«Компьютер/1». Обратите внимание, что в приведенном примере «1» – это номер сущности, а не ее экземпляра.

Атрибут (англ. Attribute) соответствует свойству или характеристике, имеющейся у всех или некоторых экземпляров сущности. Множество возможных значений атрибута ограничивается доменом, на котором он определяется.

Связь (англ. Relationship) описывает логическое отношение между двумя сущностями или между экземплярами одной сущности. Связь принято именовать глаголом или глагольной фразой.

Сущности в IDEF1X изображаются с помощью прямоугольника, над которым указано название сущности, а в верхней части над чертой перечисляются атрибуты, являющиеся уникальным идентификатором сущности. Сущности могут быть *независимыми* (англ. Identifier-Independent Entity) и *зависимыми* (англ. Identifier-Dependent Entity). Отличие заключается в том, что для существования экземпляра зависимой сущности необходима его связь с экземпляром сущности, от которой он зависит. Зависимые сущности изображаются прямоугольником со скругленными углами.

Между независимой сущностью и подчиненной ей зависимой сущностью устанавливается *идентифицирующая связь* (англ. Identifying Relationship) типа «один-ко-многим». Связь отображается непрерывной линией с черным кружком на стороне подчиненной сущности.

Связь между независимыми сущностями называется *не идентифицирующей* (англ. Non-identifying Relationship). Она изображается на диаграмме пунктирной линией с черными кружками на стороне связанных сущностей. Рассмотрим связь между двумя независимыми сущностями «Компьютер» и «Помещение» (компьютеры находятся в некоторых помещениях предприятия). Будем считать, что для компьютера обязательно должно быть определено одно помещение. Создадим *обязательную не идентифицирующую связь* «находится» (англ. mandatory non-identifying relationship) (см. рис. 4).



Рисунок 4. Обязательная не идентифицирующая связь

Если предположить, что для некоторых компьютеров помещение может быть не определено, например, для ноутбуков, то будет создана *опциональная* или *необязательная не идентифицирующая связь* (англ. optional non-identifying relationship) (см. рис. 5).



Рисунок 5. Необязательная не идентифицирующая связь

Третий тип связи, определяемый методологией IDEF1X, – связь «многие-ко-многим». При проектировании реляционной базы данных подобная связь допустима только в логической модели. Например, на компьютере может быть установлено некоторое программное обеспечение, при этом каждый программный продукт может быть установлен на нескольких компьютерах. Тогда между сущностями «Компьютер» и «Программное обеспечение» существует связь «многие-ко-многим». Связь отображается пунктирной линией с черными кружками на концах.

При переходе от логической к физической модели реляционной БД происходит преобразование связи «многие-ко-многим» путем создания зависимой сущности, для которой определяются две связи «один-ко-многим».

Рассматриваемая нотация при необходимости позволяет указать мощность связи (англ. Cardinality) – количество экземпляров подчиненной сущности, которые могут быть связаны с экземпляром родительской сущности. Могут быть определены следующие варианты:

- по умолчанию одному экземпляру родительской сущности может соответствовать 0, 1 или несколько экземпляров дочерней сущности, этот вариант на диаграмме специально не отмечается;
- P (от англ. positive – положительный), указывает, что экземпляру родительской сущности соответствует 1 или более экземпляров дочерней сущности, символ ставится рядом с черной точкой в изображении связи;
- Z (от англ. zero – ноль) указывает, что экземпляру родительской сущности соответствует 0 или 1 экземпляр дочерней сущности;
- число рядом с четной точкой указывает количество экземпляров подчиненной сущности.

Рассмотрим n -арные связи между сущностями. Связь между двумя сущностями называется бинарной. Если участников связи n ($n > 2$), связь будет n -арной. Нотация IDEF1X определяет правила изображения только бинарных связей, а для представления n -арных вводятся дополнительная сущность. Например, чтобы описать связь между сущностями «Компания», «Продукт», «Заказчик» вводится сущность «Контракт», которая описывает заказ товара у поставщика конкретным клиентом в заданную дату (см. рис. 6).

Сущность «Контракт» является *ассоциативной* (англ. associative) – она связана с несколькими родительскими сущностями и содержит дополнительную информацию о связи этих сущностей. Если дополнительных атрибутов нет, а есть только внешние ключи, такая сущность называется *именующей* или *указательной* (англ. designated).

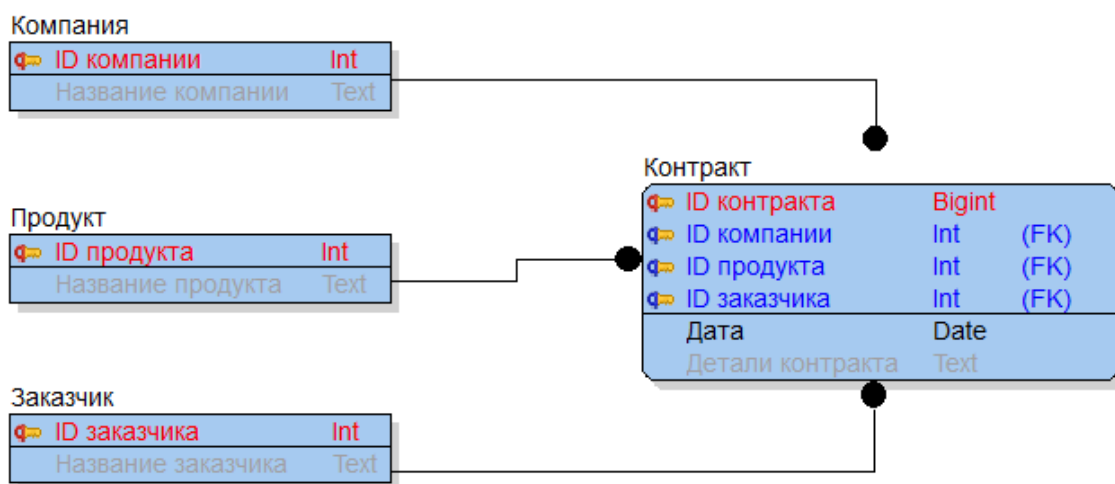


Рисунок 6. Пример n -арной связи

Другой частный случай – это *рекурсивная связь* (англ. Self-Relationship), где и в качестве родительской, и в качестве дочерней выступает одни и та же сущность. Подобные связи часто описывают иерархии однотипных объектов,

имеющихся в предметной области. Например, для описания служебной иерархии в организации можно использовать структуру данных, представленную на рис. 7.

Среди атрибутов сущности «Сотрудник» (рис. 7) присутствует атрибут «ID начальника», являющийся внешним ключом, ссылающимся на первичный ключ «ID сотрудника» этой же сущности. Таким образом, появляется возможность указать, кому подчинен сотрудник, а за счет того, что связь определена как неидентифицирующая и необязательная, в БД можно будет хранить информацию о сотрудниках, которые на данный момент никому не подчинены.

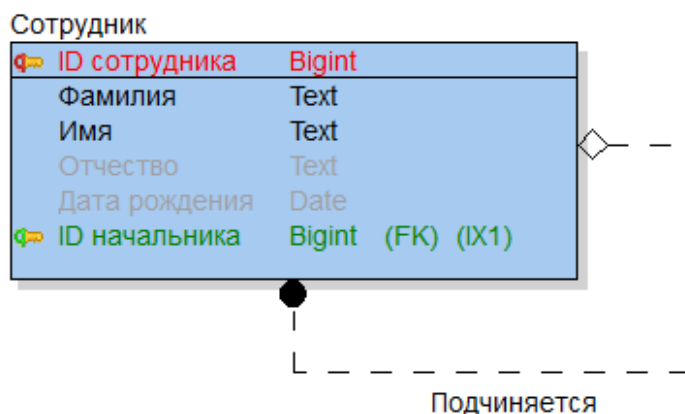


Рисунок 7. Рекурсивная связь

Рассмотренный вариант задания рекурсивной связи удобен при описании иерархии, когда у каждого логически подчиненного экземпляра сущности есть не более одного логически исходного. Например, у каждого сотрудника может быть только один непосредственный начальник, а у начальника может быть несколько подчиненных. Такую связь называют *иерархической рекурсией* (англ. *hierarchical recursion*). Однако на практике может встречаться и *сетевая рекурсия* (англ. *network recursion*), когда между родительскими и дочерними объектами присутствует связь типа «многие-ко-многим». Например, производственная компания может как закупать товары у многих поставщиков, так и поставлять товары многим заказчикам. В подобном случае при проектировании БД нужно использовать дополнительную зависимую сущность и две идентифицирующие связи «один-ко-многим». На рис. 8 для этих целей введена сущность «Поставка», описывающая связь заказчика с поставщиком.

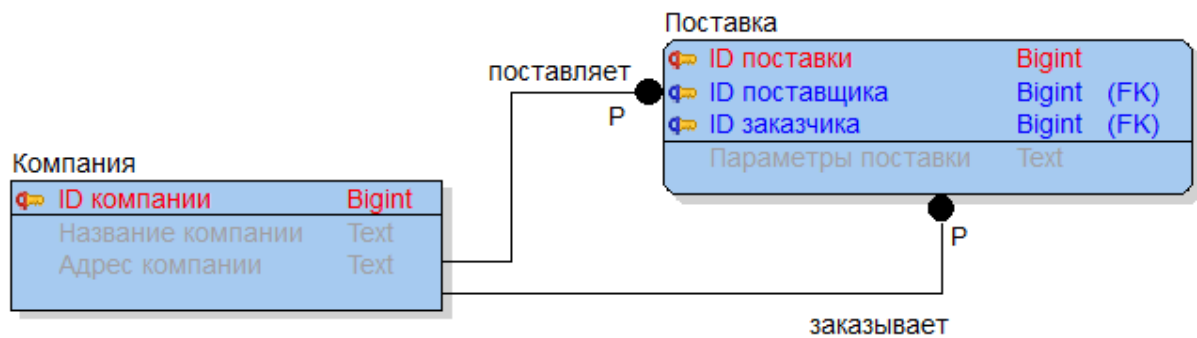


Рисунок 8. Пример задания сетевой рекурсии

Следующая возможность нотации IDEF1X, на которую следует обратить внимание, – это определение категорий. В ряде случаев одна сущность является подтипом или категорией другой. Например, компьютер может быть сервером, рабочей станцией или ноутбуком. Рассматриваемое отношение между сущностями является иерархическим, но в отличие от иерархической рекурсии, в данном случае строится иерархия сущностей, а не экземпляров. Дочерняя сущность в этой иерархии называется *категориальной*. В приведенном примере категориальные сущности – «Сервер», «Рабочая станция», «Ноутбук».

Подобным образом может строиться *иерархия категорий* или *иерархия наследования*, представляющая собой особый тип объединения сущностей, которые разделяют общие характеристики.

Для каждой категории можно указать *дискриминатор* (англ. discriminator) – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой. Например, это может быть атрибут «Тип компьютера», относящийся к сущности «Компьютер».

Множество категорий может быть полным и неполным. В первом случае экземпляр родительской сущности обязательно относится к одной из указанных категорий, во втором случае могут существовать экземпляры, не относящиеся ни к одной из перечисленных категорий.

На рис. 9 представлен фрагмент логической модели с неполным множеством категорий: для родительской сущности «Компьютер» определены категории «Сервер» и «Рабочая станция». На то, что множество категорий неполно, указывает одна черта в значке категории. В качестве дискриминатора выбран атрибут «Тип компьютера».

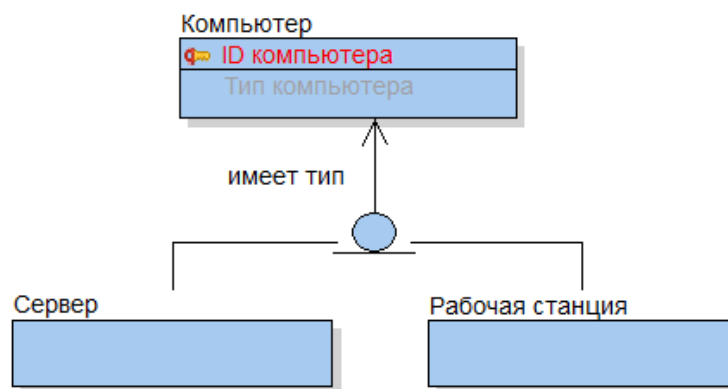


Рисунок 9. Иерархия категорий, неполное множество категорий

На рис. 10 добавлена категориальная сущность «Ноутбук» и указано, что множество категорий теперь является полным (значок категории с двойной чертой).

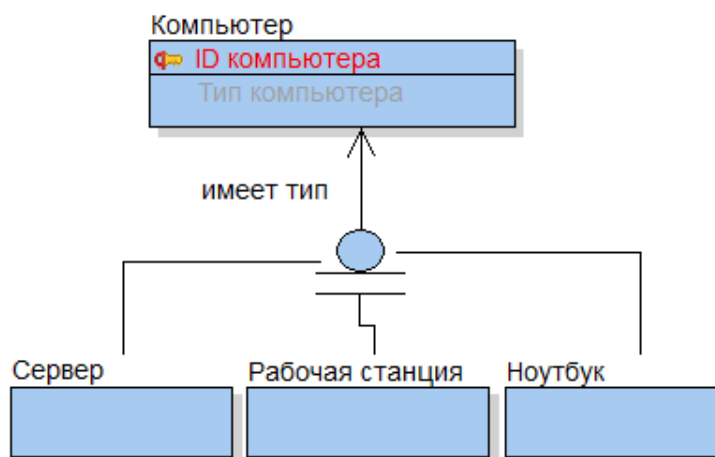


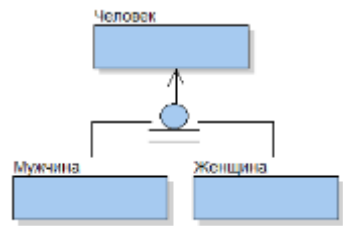
Рисунок 10. Иерархия категорий, полное множество категорий

Категории не поддерживаются физической моделью. При преобразовании логической модели в физическую будут созданы связи дочерних сущностей с родительской сущностью на основе внешних ключей.

В табл. 1 представлена классификация зависимых сущностей, поддерживаемых в нотации IDEF1X.

Таблица 1. Классификация зависимых сущностей

Название типа	Описание	Пример
Характеристика (<i>англ. characteristic</i>)	Используется для представления группы атрибутов, которая может более одного раза встречаться у экземпляра сущности. При этом данные атрибуты не относятся к другой независимой	

	сущности. Например, хобби является характеристикой человека.	
Ассоциативная (<i>англ. associative</i>) или именуемая/указательная (<i>англ. designated</i>)	Используется для описания связи между двумя или более сущностями. Если сущность содержит только атрибуты, входящие во внешние ключи, указывающие на родительские сущности, она называется именуемой. Если сущность имеет также собственные атрибуты, описывающие дополнительные параметры связи, она называется ассоциативной.	
Категория или подтип (<i>англ. subtype</i>)	Категориальная сущность является подтипом родительской сущности.	

CASE-средства. Создание логической и физической моделей базы данных в Toad Data Modeler 6.1 Freeware

Задача проектирования БД для современной информационной системы корпоративного уровня может быть достаточно трудоемкой и требовать совместной работы большой группы специалистов – аналитиков, разработчиков БД, разработчиков прикладного ПО, специалистов в предметной области, для которой разрабатывается БД. Для автоматизации этого процесса широко используются CASE-средства (*англ. computer aided software engineering*) – программные средства, поддерживающие одну или несколько технологий проектирования БД. В качестве примера можно назвать программные продукты ERwin Data Modeler (разработчик – компания CA Technologies), ER/Studio (разработчик – компания Embarcadero Technologies), PowerDesigner (разработчик компания Sybase, в настоящее время приобретена SAP), Toad Data Modeler (разработчик – компания Quest Software). Отчасти подобная функциональность реализована и в популярном офисном программном продукте Microsoft Visio.

Рассмотрим создание логической и физической моделей данных с использованием Toad Data Modeler 6.1 Freeware. Данный программный продукт позволяет отображать логическую и физическую модели данных в нотациях IE и IDEF1X (смена нотации выполняется командой View – Notation - ...).

Пусть требуется разработать БД для хранения информации о компьютерной системе предприятия. Компьютеры имеют уникальные номера и имена, причем для одного компьютера может быть использовано несколько дополнительных имен (псевдонимов), например, компьютер serv1.corp.ru может иметь псевдонимы www.mycorp.net и <ftp.mycorp.net>. Компьютер по типу может быть сервером, рабочей станцией или ноутбуком. Компьютеры размещены в помещениях организации, на них установлено ПО.

Создадим новый проект в Toad Data Modeler 6.1 Freeware. Для этого:

1. Перейдем File – New – Project...
2. В открывшемся окне New Project выберем тип проекта – Toad Data Modeler Project и щелкнем кнопку Next.
3. В поле Project Name зададим имя проекту – Computer_system и определим папку для размещения проекта, используя поле Path to Project. Затем щелкнем кнопку Create.

В окне Application View (если оно не отображается на экране, выберите Window – Application View) вы увидите имя проекта и папки, созданные для размещения основных компонентов проекта: Models; Reports; SQL Scripts; Others.

Создадим логическую модель базы данных. Для этого:

1. В окне Application View щелкнем правой клавишей мыши по папке Models и выберем команду New Model...
2. В открывшемся окне New Model перейдем на вкладку Logical Data Model, в поле Model Name введем имя модели Logical model_Computer system и щелкнем кнопку Ok.

На экране появится вкладка Logical model_Computer system, на которой отображаются основные папки для размещения компонентов логической модели: Workspaces (рабочие пространства); Entities (сущности); Relationships (связи); Inheritances (наследования); Categories (категории); Defaults (значения по умолчанию); Domains (домены); Notes (заметки); Rules (правила); Images (изображения); Inheritances Ancestors (предки); Inheritances Descendents (потомки).

Создадим сущности: Компьютер; Помещение; Псевдоним компьютера; Программное обеспечение (см. табл. 2).

Таблица 2. Сущности логической модели базы данных

Имя сущности	Надпись	Атрибут, уникально идентифицирующий сущность (надпись/имя/тип)	Атрибуты сущности (надпись/имя/тип)
--------------	---------	--	-------------------------------------

Computer	Компьютер	Номер компьютера/ Comp_Number/ Bigint	<ul style="list-style-type: none"> Имя компьютера/ Comp_Name/Text Тип компьютера/ Comp_Type/Text
Room	Помещение	Номер помещения/ Room_Number/ Integer	<ul style="list-style-type: none"> Название помещения/ Room_Name/ Text Описание помещения/ Room_Desc/ Text
Computer Nickname	Псевдоним компьютера	Псевдоним/ Nickname/Text	Комментарий/ Comment/ Text
Software	Программное обеспечение	Идентификатор программы/ ProgramID/ Bigint	Название программы/ Program_Name/ Text

Для *создания сущности* щелкнем по папке Entities в окне Logical model_Computer system правой кнопкой мыши и выберем команду Add Entity. На рабочем пространстве логической модели появится обозначение созданной сущности (прямоугольник), а в папке Entities – описание этой сущности и ее основные компоненты: Shortcuts (ярлыки); Attributes (атрибуты); Unique Identifiers (уникальные идентификаторы).

Для *переименования сущности* выполним двойной щелчок мышью по ее имени в папке Entities. В открывшемся окне Entity Properties на вкладке General зададим Caption (надпись) и Name (имя) сущности.

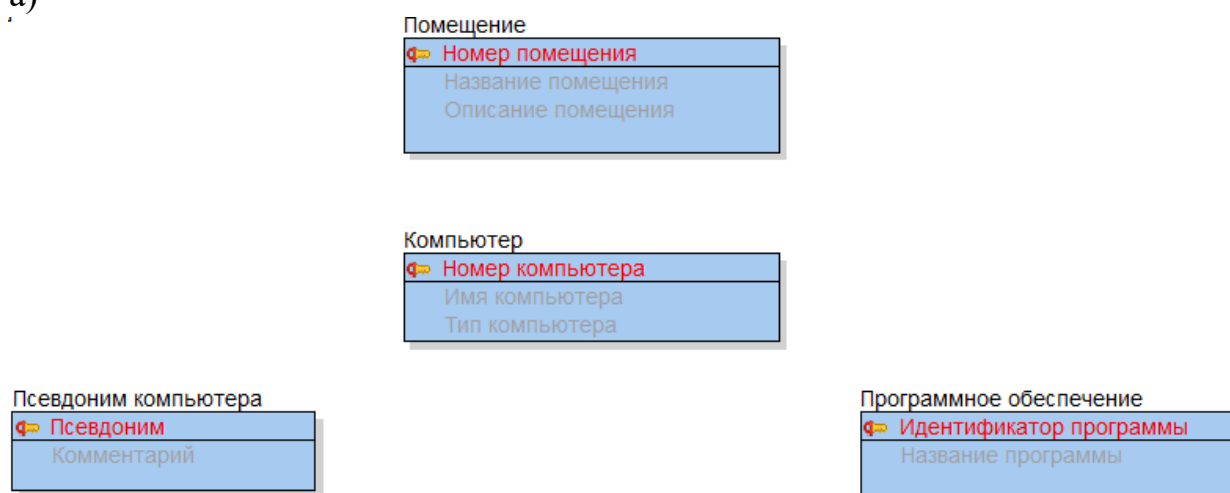
Для *создания атрибута, уникально идентифицирующего сущность*, щелкнем по имени сущности в папке Entities правой клавишей мыши и выберем команду Add – Attribute (UI). В открывшемся окне Attribute Properties зададим: Caption (надпись); Name (имя); Data Type (тип данных).

Для *создания атрибута*, щелкнем по имени сущности в папке Entities правой клавишей мыши и выберем команду Add – Attribute. В открывшемся окне Attribute Properties зададим: Caption (надпись); Name (имя); Data Type (тип данных). Если значение атрибута обязательно должно быть определено, поставим флажок Mandatory (обязательный).

Для *удаления атрибута* щелкнем по его имени в папке Attributes и выберем команду Delete Item.

На рис. 11 представлено описание сущностей табл. 2.

а)



б)

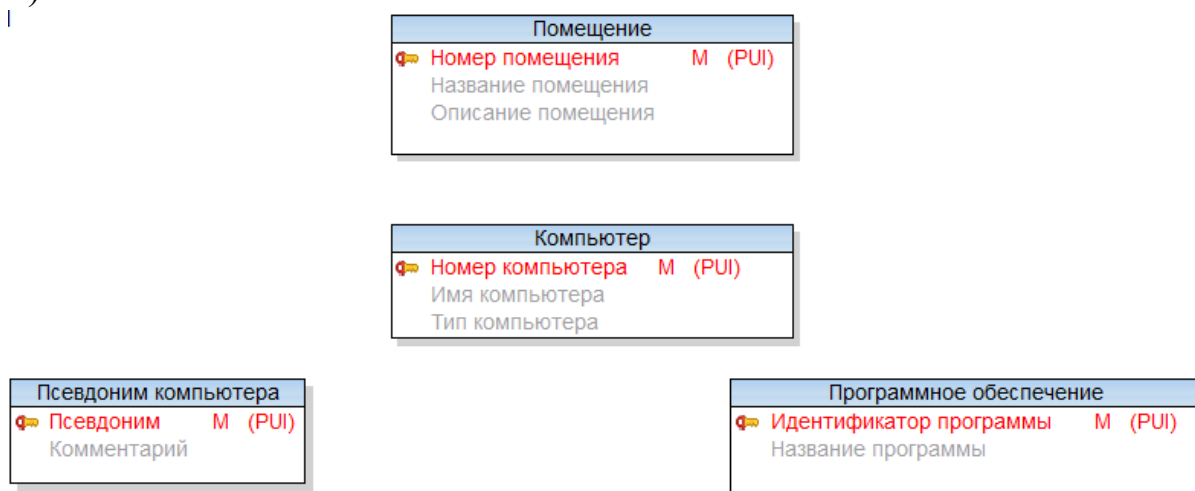


Рисунок 11. Представление сущностей логической модели базы данных
 а) в нотации IDEF1X, б) в нотации IE

Установим связи между сущностями (см. рис. 12), учитывая следующие положения:

1. Компьютер может быть не привязан к помещению или привязан к одному помещению. В одном помещении может находиться ноль или некоторое количество компьютеров. Т.е. между независимыми сущностями «Помещение» и «Компьютер» должна быть установлена необязательная неидентифицирующая связь.
2. Компьютер может не иметь псевдонима, иметь один или несколько псевдонимов. Т.е. между сущностями «Компьютер» и «Псевдоним компьютера» существует идентифицирующая связь «один-ко-многим».
3. На каждом компьютере может быть установлено различное программное обеспечение и при этом один программный продукт может быть установлен на несколько компьютеров. Т.е. между сущностями

«Компьютер» и «Программное обеспечение» существует связь «многие-ко-многим».

Для создания *идентифицирующей* связи между сущностями выберем Objects – Add New – Identifying Relationship, щелкнем по сущности, которая находится в связи на стороне «один», затем щелкнем по сущности, которая находится в связи на стороне «ко-многим».

Для создания *обязательной неидентифицирующей* связи между сущностями выберем Objects – Add New – Non-identifying relationship, щелкнем по сущности, которая находится в связи на стороне «один», затем щелкнем по сущности, которая находится в связи на стороне «ко-многим». В окне Relationship Properties на вкладке Cardinality оставим флажок Mandatory (обязательный).

Для создания *необязательной неидентифицирующей* связи между сущностями выберем Objects – Add New – Non-identifying relationship, щелкнем по сущности, которая находится в связи на стороне «один», затем щелкнем по сущности, которая находится в связи на стороне «ко-многим». В окне Relationship Properties на вкладке Cardinality снимем флажок Mandatory (обязательный).

Для *изменения свойств связи* выполним двойной щелчок мышью по ее изображению, в окне Relationship Properties на вкладке General зададим Caption (надпись) и Name (имя) связи, на вкладке Cardinality – тип связи.

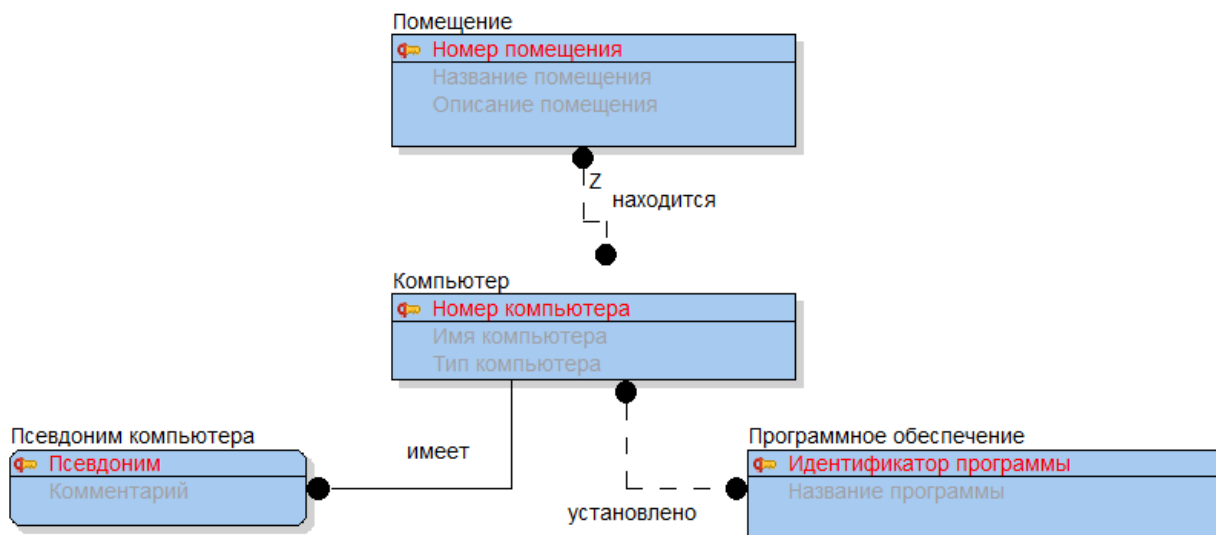


Рисунок 12. Отображение связей между сущностями в нотации IDEF1X

Для сущности «Компьютер» определим категории «Сервер» (атрибут – тип сервера), «Рабочая станция» (атрибут – монитор) и «Ноутбук» (атрибут – вес). Для этого создадим соответствующие сущности, не указывая атрибут, уникально идентифицирующий сущность.

Для создания связи между сущностями выполним действия:

1. Выберем Objects – Add New – Inheritance.
2. Щелкнем мышью по сущности «Компьютер», затем по сущности «Сервер».
3. Выполним двойной щелчок мышью по значку связи в диаграмме и настроим параметры связи в окне Inheritance Properties (на вкладке General поставим флажок Complete, зададим свойство Caption – имеет тип, зададим свойство Name – haveType; на вкладке Generation определим дискриминатор связи – Comp_Type см. рис. 13). Закроем окно свойств связи.

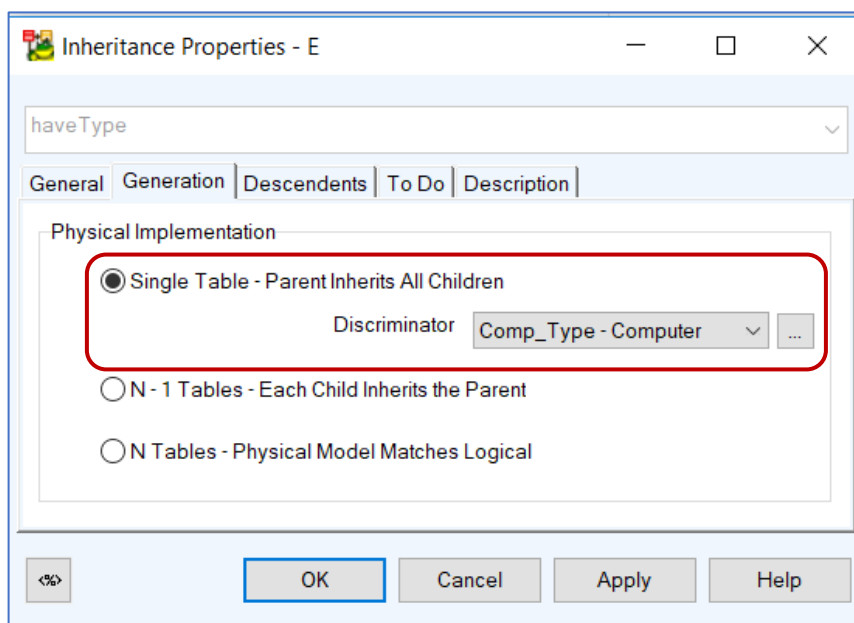


Рисунок 13. Определение дискриминатора связи

4. Снова выберем Objects – Add New – Inheritance, щелкнем по значку созданной связи «имеет тип», затем по сущности «Рабочая станция». Выполним аналогичные действия для включения сущности «Ноутбук» в связь.

Итог выполненной работы представлен на рис. 14.

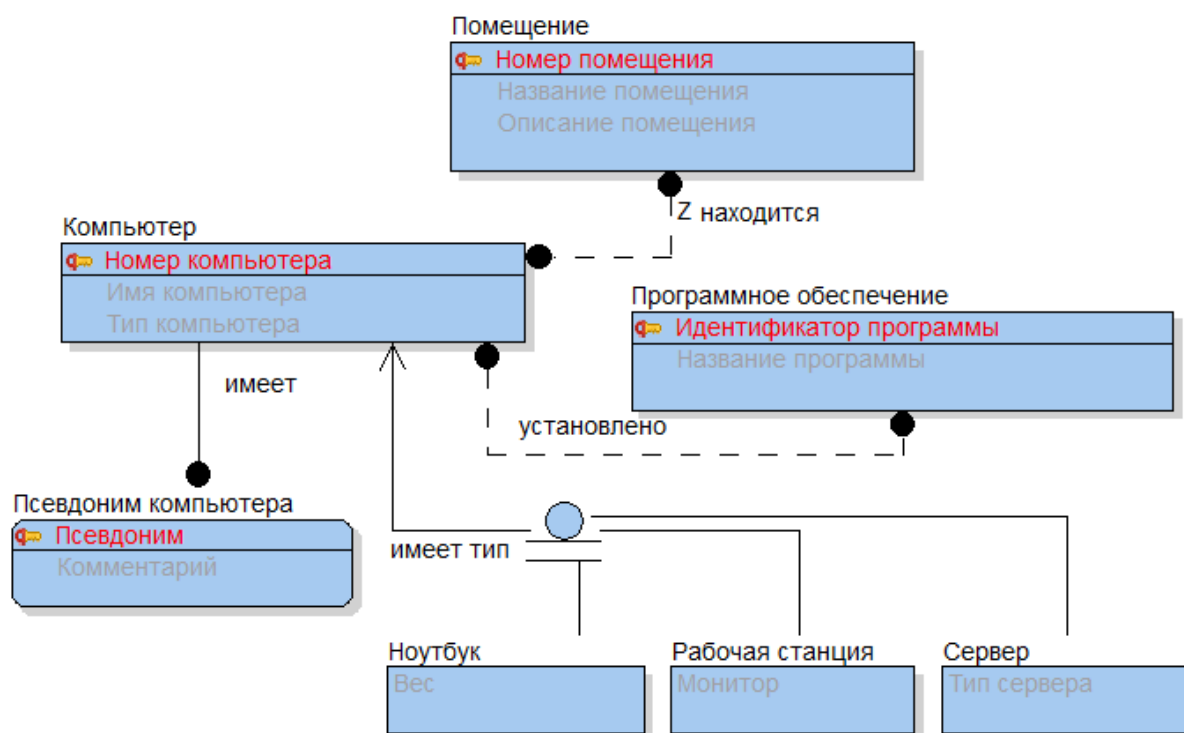


Рисунок 14. Логическая модель базы данных

На основе логической модели построим физическую модель, ориентированную на СУБД Microsoft SQL Server 2016. Для этого в меню выберем Model – Convert Model – Simple Conversion. В открывшемся окне Conversion в поле To Database выберем нужную СУБД, в поле New Model Name зададим имя модели – Physical model_Computer system. Результат преобразования представлен на рис. 15.

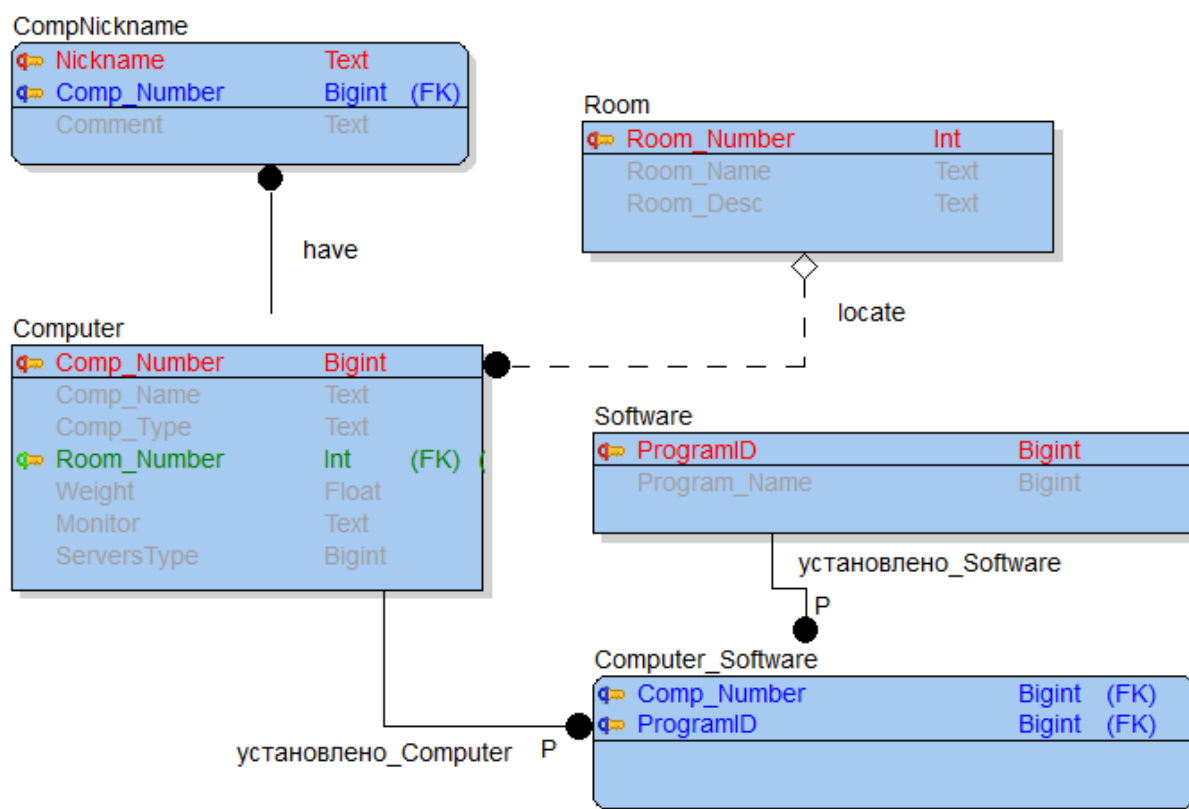


Рисунок 15. Физическая модель базы данных

В физической модели базы данных не могут использоваться связи типа «многие-ко-многим» и категориальные связи, поэтому они были преобразованы.

Кроме уточнения типов данных и приведения названий таблиц и полей в соответствие требованиям СУБД, физическая модель отличается возможностью включения в нее индексов, хранимых процедур, триггеров и прочих объектов БД.

Вопросы и задания к лекции

1. В чем отличие сильных и слабых сущностей?
2. В каком случае участие сущности в связи будет полным, в каком – частичным?
3. Как изображаются сущности и связи в нотации Чена?
4. Как изображаются сущности и связи в нотации Баркера?
5. Каким образом преобразуется связь «многие-ко-многим» при переходе от ER-диаграмм к реляционным отношениям?
6. В среде Toad Data Modeler 6.1 Freeware постройте логическую модель, аналогичную представленной на рис. 14. Поработайте с представлением модели в нотации IDEF1X и IE.
7. Дополните модель новыми сущностями – «Пользователь», «Группа пользователей», «Периферийное устройство». Для периферийных

устройств определите категории (принтер, сканер и др.). Определите необходимые связи между сущностями: пользователи работают за компьютером, пользователи объединены в группы и др.

8. Перейдите к физической модели БД для выбранной СУБД.

Лекция 7. Основы языка SQL

План лекции:

1. Основные понятия.
2. Типы данных.
3. Создание доменов.
4. Создание таблиц.
5. Операции добавления, обновления и удаления данных.

Литература:

1. Нестеров, С. А. Базы данных : учебник и практикум для академического бакалавриата / С. А. Нестеров. – М. : Издательство Юрайт, 2017. – 230 с. – Серия : Бакалавр. Академический курс.
2. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2017. — 311 с. — (Университеты России).
3. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для академического бакалавриата / В. М. Илюшечкин. — М. : Издательство Юрайт, 2017. — 213 с. — (Бакалавр. Академический курс).

Основные понятия

В настоящее время SQL (*англ.* structured query language – язык структурированных запросов) является стандартным языком БД и поддерживается всеми распространенными реляционными и объектно-реляционными СУБД.

Стандарт SQL определяется ANSI (Американским Национальным Институтом Стандартов), а также ISO (Международной организацией по стандартизации).

Язык SQL включает три подязыка:

- DDL (*англ.* Data Definition Language) – язык определения данных, включающий такие операторы, как CREATE, ALTER, DROP;
- DML (*англ.* Data Manipulation Language) – язык обработки данных, который позволяет запрашивать и изменять данные и включает операторы SELECT, INSERT, UPDATE, DELETE;
- DCL (*англ.* Data Control Language) – язык управления данными, позволяющий управлять разрешениями на доступ к данным и включает операторы GRANT и REVOKE.

Стандарт SQL может не в полной мере поддерживаться конкретной СУБД. И наоборот, конкретная СУБД может давать возможность использования в SQL-

выражениях дополнительных функций, не предусмотренных стандартом. Таким образом, большинство компаний разработчиков поддерживают свой собственный диалект языка SQL, который может быть совместим с одной из версий стандарта SQL. Ниже приведен перечень некоторых СУБД и наименований диалектов SQL:

- Microsoft SQL Server – Transact-SQL или T-SQL;
- Microsoft Access – Jet SQL;
- Oracle Database – PL/SQL;
- IBM DB2 – SSQL PL.

Используемые в SQL термины отличаются от принятых в реляционной алгебре. В частности, используются такие названия, как «таблица», «строка» и «столбец» вместо «отношение», «кортеж» и «атрибут». Также необходимо отметить, что результат запроса SQL может содержать повторяющиеся строки. Это отличает его от результата реляционного выражения, где повторяющихся кортежей быть не может.

Типы данных

Стандарт языка SQL определяет ряд типов данных. Ниже перечислены наиболее часто используемые из них. Некоторые СУБД могут не поддерживать часть из перечисленных типов или наоборот – добавлять свои типы данных.

INTEGER, *SMALLINT* – целые числа со знаком. Размерность определяется реализацией. Размерность *SMALLINT* не должна превышать *INTEGER*. Многие СУБД позволяют сократить *INTEGER* до *INT*.

NUMERIC (*p*, *s*), *DECIMAL* (*p*, *s*) – задают в общем виде формат точного числа, целого или с дробной частью. *NUMERIC* представляет число с *точностью p* – число знаков и *масштабом s* – число знаков после запятой. Если опущен масштаб, то он предполагается равным 0 (т.е. число целое), а если опущена точность, то ее значение по умолчанию определяется реализацией. Тип *DECIMAL* требует, чтобы использовалась зависящая от реализации точность *m*, такая, что $m \geq p$. Допустимо сокращение *DEC*.

FLOAT (*p*), *REAL*, *DOUBLE PRECISION* – числа в представлении с плавающей точкой (приближенные числовые типы). Точность *REAL* определяется реализацией, точность *DOUBLE PRECISION* должна быть больше точности *REAL*. Точность *FLOAT* задается параметром *p*, если параметр отсутствует – определяется реализацией.

CHARACTER (n) – строка из n символов, допускающая сокращение *CHAR(n)*. Запись *CHAR* соответствует *CHAR(1)*. Если длина сохраняемого значения меньше n , в конце будет добавлено соответствующее число пробелов.

Тип *NATIONAL CHARACTER (n)* хранит данные в кодировке UNICODE (2 байта на символ для поддержки национальных алфавитов). Допустима запись *NCHAR*. Строковые константы заключаются в одинарные или двойные кавычки (например, "example", 'example2').

CHARACTER VARYING (n) – строка переменной длины не более чем из n символов, допускается сокращение *VARCHAR (n)*. Если длина сохраняемого значения m , где $m < n$, будет сохраняться ровно m символов. Аналогично предыдущему случаю, существует тип *NATIONAL CHARACTER VARYING (n)* с синонимом *NVARCHAR (n)*.

CHARACTER LARGE OBJECT – символьный тип, позволяющий хранить большой объем текста. Допустима запись *CLOB*. Также существует тип *NCLOB*.

XML – документ в формате XML (тип данных введен в версии стандарта SQL:2003).

BIT(n) – битовая строка длиной n бит. Запись *BIT* аналогична *BIT(1)*.

BIT VARYING(n) – битовая строка переменной длины не более n бит.

BINARY LARGE OBJECT – большой двоичный объект, допустима запись *BLOB*.

DATA – дата в формате 'YYYY-MM-DD', например, '2017-08-21' для 21 августа 2017 года.

TIME – тип данных для хранения отметок времени, включающих поля <часы>:<минуты>:<секунды>:<доли секунды>.

TIMESTAMP – тип данных для совместного хранения даты и времени.

Создание доменов

В SQL понятие «домен» несколько отличается от аналогичного понятия реляционной модели:

- использование доменов не обязательно – столбцы в таблицах могут определяться напрямую через встроенные типы данных;
- домен в SQL должен определяться в терминах одного из строенных типов данных;

- домен в SQL играет роль синтаксического сокращения, позволяющего один раз ввести значение по умолчанию или набор ограничений и использовать их для определения множества столбцов;
- домены в SQL не ограничивают сравнение – контролируется, чтобы в сравнении участвовали значения одного базового типа (числовые, строковые и др.), которые не обязательно должны принадлежать одному домену.

Для создания домена используется оператор *CREATE DOMAIN*, формат которого приведен ниже:

```
CREATE DOMAIN < domain name > [AS] < data type >
[default definition]
[domain – constraint – definition – list]
```

где

- < *domain name* > – название создаваемого домена;
- < *data type* > – поддерживаемый SQL тип данных, который является базовым для создаваемого домена;
- *AS* – необязательное ключевое слово, разделяющее название домена и базового типа данных;
- *default definition* – необязательный параметр, определяющий значение по умолчанию, которое применяется к элементам каждого столбца, определенного на данном домене и не имеющего собственного определения значения по умолчанию;
- *domain – constraint – definition – list* – список ограничений целостности, применяемый к каждому столбцу, основанному на данном домене.

Например, нужно описать домены, на которых будут определяться столбцы приведенной ниже таблицы (см. табл. 1).

Таблица 1. Таблица STUDENTS

NUM	NAME	GROUP
123	Иванов И.И.	382
124	Петров П.П.	382

Пусть первый домен будет основан на типе *INTEGER* и будет содержать ограничение на значения (значения > 0):

```
CREATE DOMAIN Dnum INTEGER
CONSTRAINT ck_ValNum CHECK (VALUE > 0)
```

Здесь:

- *Dnum* – имя домена;
- *CONSTRAINT* – ключевое слово, обозначающее «ограничение»;
- *ck_ValNum* – название создаваемого ограничения, которое заключается в проверке значения на условие «> 0».

Если не задавать имя ограничения, то ключевое слово *CONSTRAINT* тоже не пишется.

Второй домен – строки переменной длины до 50 символов со значением по умолчанию «???»:

```
CREATE DOMAIN Dname VARCHAR(50) DEFAULT '???'
```

Третий домен – номер группы, размерностью не более трех десятичных символов:

```
CREATE DOMAIN Dgroup NUMERIC(3)
```

Существующий домен можно изменить с помощью оператора *ALTER DOMAIN*:

```
ALTER DOMAIN < domain name > < alter domain action >
```

При помощи данного оператора можно определить новое значение по умолчанию, удалить существующее, ввести новое ограничение целостности. Например, удалим ограничение на значение домена *Dnum* и добавим значение по умолчанию для домена *Dgroup*:

```
ALTER DOMAIN Dnum DROP CONSTRAINT ck_valNum;
```

```
ALTER DOMAIN Dgroup SET DEFAULT 382
```

Существующий домен можно удалить оператором *DROP DOMAIN*. Его синтаксис:

```
DROP DOMAIN < domain name > {RESTRICT | CASCADE}
```

Где:

- *< domain name >* – имя домена;
- *RESTRICT* – домен не будет уничтожен, если он использован в определении какого-либо столбца, представления или ограничения целостности;
- *CASCADE* – домен будет уничтожен в любом случае.

Столбцы, определенные на этом домене, автоматически переопределяются следующим образом:

- считается, что каждый такой столбец теперь относится к типу данных, на основе которого определялся уничтоженный домен;
- если у столбца не было определено собственное значение по умолчанию, то считается, что теперь у него значение по умолчанию, которое было у уничтожаемого домена;
- каждый столбец наследует все ограничения уничтожаемого домена.

Например, удалим домен Dnum.

DROP DOMAIN < Dnum > RESTRICT

В СУБД Microsoft SQL Server оператор *CREATE DOMAIN* не поддерживается. В данной СУБД подобную задачу можно частично решить с помощью оператора *CREATE TYPE*. Но этот оператор не позволяет задавать значения по умолчанию и ограничения на значения с помощью *CHECK*. Пример использования оператора:

CREATE TYPE Dchar15 FROM CHAR(15) NOT NULL

Создание таблиц

Таблицы SQL имеют следующие особенности по сравнению с отношениями реляционной модели:

- в таблицах SQL допустимы идентичные строки, поэтому нет требования обязательного наличия потенциальных ключей;
- в таблицах SQL столбцы рассматриваются в порядке слева направо, тогда как в отношении порядок атрибутов неважен.

Базовые таблицы создаются с помощью оператора *CREATE TABLE*, формат которого

CREATE TABLE < table name > (table – element – commalist), где:

- *< table name >* – имя создаваемой базовой таблицы;
- *table – element – commalist* – список через запятую определения столбцов или ограничений уровня таблицы. В списке элементов должно быть хотя бы одно определение столбца.

Определение столбца должно включать название столбца и указание на базовый тип или домен, на котором столбец определен. Также оно может включать указание значения по умолчанию и ограничения уровня столбца.

Ограничение *NOT NULL* требует, чтобы столбец не мог содержать значение *NULL* (все значения должны быть определены). По умолчанию или при явном указании *NULL* в определении столбца неопределенные значения допускаются.

Ключевое слово *PRIMARY KEY* позволяет указать первичный ключ, а *UNIQUE* – альтернативный. Оба эти ограничения требуют уникальности значений, но *PRIMARY KEY* дополнительно включает ограничение *NOT NULL*. Таким образом, если для столбца задано ограничение *UNIQUE*, в этом столбце может встречаться значение *NULL* не более одного раза.

Если указано ограничение *CHECK* (<условие>), будет выполняться проверка условия на значение. Попытка создания строки рассматривается как нарушение проверочного условия, если в результате вычисления условного выражения получено значение «ложь».

Ограничение внешнего ключа, если его задавать как ограничение уровня столбца, описывается в соответствии с форматом

REFERENCES < table name > [(column)]
[*ON DELETE* option] [*ON UPDATE* option]

где < table name > – имя таблицы, на которую ссылаются внешний ключ; в скобках может быть указан столбец, если он опущен, то объект ссылки – первичный ключ указанной таблицы; *option* определяет поведение при попытке удалить или изменить строку «родительской» таблицы, на которую ссылается внешний ключ; может принимать значения *NO ACTION* (запретить изменение), *CASCADE* (каскадировать изменение или удаление), *SET DEFAULT* (установить значение по умолчанию), *SET NULL* (установить значение *NULL*).

Рассмотрим пример. Пусть требуется создать таблицу T5 с двумя столбцами – целочисленным первичным ключом ID и вещественным столбцом Sum, значение которого должно быть обязательно определено. Это можно сделать при помощи следующего оператора:

CREATE TABLE T5 (ID INTEGER PRIMARY KEY, Sum REAL Not NULL)

Создадим таблицу Students со столбцами STUdId, Fio, Group. При определении двух последних столбцов используем домены Dname и Dgroup, а столбцу Group определим значение по умолчанию. Код на SQL буде выглядеть следующим образом:

CREATE TABLE Students
(StudId INTEGER PRIMARY KEY,
Fio Dname, Group Dgroup DEFAULT 383)

И для домена *Dname*, и для домена *Dgroup* были определены значения по умолчанию. Однако у столбца *Group* будет значение по умолчанию, явно заданное при создании таблицы, а у *Fio* – унаследованное от определения домена.

В зависимости от СУБД из-за совпадения названия столбца и ключевого слова языка SQL может понадобиться явно указать, что Group – это название столбца. В случае Microsoft SQL Server для этого используются квадратные скобки – [Group].

Рассмотрим теперь задание ограничений уровня таблицы. С их помощью можно, например, определить составной первичный или альтернативный ключ, что не удастся сделать посредством ограничения уровня столбца.

Ограничения уровня таблицы описываются в операторе CREATE TABLE после описания столбцов. Используется следующий формат:

```
[CONSTRAINT <constraint name >]
{PRIMARY KEY | UNIQUE} (column_commlist)
| FOREIGN KEY (column_commlist)
references_definition
| CHECK (conditional_expression)}
```

Рассмотрим пример. Пусть требуется создать таблицу Results с результатами сдачи экзаменов студентами. Столбцы: StudId – идентификатор студента, внешний ключ, ссылающийся на альтернативный столбец в таблице Students, Subj – название предмета, Mark – оценка за предмет. Первичный ключ составной: StudId и Subj. Подобную таблицу можно создать с помощью следующего выражения:

```
CREATE TABLE Results
(StudId INTEGER REFERENCES Students,
Subj CHAR(20),
Mark SMALLINT Not NULL,
PRIMARY KEY (StudId, Subj),
CHECK (Mark >= 2 and Mark <= 5))
```

Здесь в определении внешнего ключа пропущено указание на столбец в таблице, на которую ссылаемся. Это означает, что внешний ключ ссылается на первичный ключ соответствующей таблицы. Такой же результат, но с явным заданием имен первичного и внешнего ключей и ограничения CHECK, получится при вызове выражения:

```
CREATE TABLE Results
(StudId INTEGER,
Subj CHAR(20),
Mark SMALLINT Not NULL,
CONSTRAINT pk_Results PRIMARY KEY (StudId, Subj)
CONSTRAINT fk_Results FOREIGN KEY (StudId)
REFERENCES Students (StudId),
```

CONSTRAINT ck_Results CHECK (Mark >= 2 and Mark <= 5))

Здесь *pk_Results*, *fk_Results*, *ck_Results* – явно заданные названия ограничений. Явное указание имен ограничений может быть полезно, например, при их изменении. В то же время, если ключевое слово *CONSTRAINT* и название ограничения пропустить, СУБД автоматически сгенерирует имя ограничения, которое можно будет узнать с помощью инструментов администрирования.

Базовая таблица может быть изменена с помощью оператора *ALTER TABLE*. Допустимы следующие изменения:

- добавление и удаление столбцов;
- определение для существующего столбца значения по умолчанию и удаление ранее определенного значения по умолчанию;
- создание нового ограничения целостности для таблицы и удаление ранее определенного.

Рассмотрим пример. Ранее была создана таблица *Results*, добавим в нее столбец *Comment* для комментария к оценке, после чего удалим его:

ALTER TABLE Results ADD Comment varchar(100);
ALTER TABLE Results DROP COLUMN Comment

Стандарт SQL допускает в первом выражении наличие необязательного ключевого слова *COLUMN* (получится «...*ADD COLUMN Comment*...»), но в диалекте *Transact-SQL* это не допускается. В большинстве СУБД, чтобы отделить одно выражение (запрос) на SQL от другого, используется разделитель «;», но эта настройка может меняться.

Теперь предположим, что изменился диапазон оценок – стала допустима оценка «1». Удалим старое ограничение целостности *ck_Results* и определим новое:

ALTER TABLE Results DROP CONSTRAINT ck_Results;
ALTER TABLE Results ADD CONSTRAINT ck_Results
CHECK (Mark >= 1 and Mark <= 5)

Базовая таблица может быть удалена оператором *DROP TABLE*, формат которого представлен ниже:

DROP TABLE < table name > {RESTRICT | CASCADE}, где

- *< table name >* – название существующей базовой таблицы;
- опция *RESTRICT* не позволит удалить таблицу, если она используется при определении какого-либо представления или ограничения целостности;

- при использовании опции *CASCADE* оператор выполняется в любом случае, определения представлений и ограничений целостности, включающие указания на данную таблицу, также будут удалены.

В СУБД Microsoft SQL Server опции *RESTRICT* и *CASCADE* не поддерживаются, а оператор *DROP TABLE* всегда работает так, как будто указана опция *RESTRICT*. Таким образом, таблицу T1 можно удалить с помощью следующего выражения:

DROP TABLE T1

Операции добавления, обновления и удаления данных

Язык обработки данных (DML) включает три операции обновления: *INSERT* (вставка), *UPDATE* (изменение) и *DELETE* (удаление).

Вставка строк производится с помощью оператора *INSERT*, формат которого приведен ниже:

INSERT INTO < table name > [(column_list)]
{VALUES (value_list) | < query >}

где *< table name >* – имя базовой таблицы или обновляемого представления, *column_list* – необязательный параметр, позволяющий указать обновляемые столбцы; если он не указан, то порядок столбцов берется таким же, как в определении таблицы.

Форма записи с ключевым словом *VALUES* позволяет добавить в таблицу строку с заданными значениями атрибутов. Например, добавим в таблицу *Students* дополнительную запись (атрибуты *StudID* и *Group* – числовые; *FIO* – строковый, значение в соответствии с требованиями синтаксиса SQL взято в одинарные кавычки):

INSERT INTO Students
VALUES (130, 'Ивановский И. И.', 386)

Аналогичный результат может быть получен выполнением следующего выражения (разница в том, что порядок значений изменен и надо явно указать перечень столбцов):

INSERT INTO Students (FIO, Group, StudID)
VALUES ('Ивановский И. И.', 386, 130)

Таблица 2. Таблица Students

StudID	FIO	Group
123	Иванов И.И.	382
124	Петров П.П.	382
125	Сидоров С.С.	383

Значения отдельных атрибутов при добавлении записи могут быть опущены. В этом случае столбец получит или значение по умолчанию, если оно для него было определено (явно или через определение домена), или значение *NULL*. Если такой столбец находится в середине списка, пропуск значения явно указывается, если в конце – можно ничего не указывать:

```
INSERT INTO Students (StudID, Group, FIO)
VALUES (131, 'Васильев В. В.')
```

Сравните с

```
INSERT INTO Students (StudID, FIO, Group)
VALUES (131, 'Васильев В. В.')
```

С помощью оператора *INSERT* также можно добавить в таблицу набор строк, полученных в результате выполнения запроса *< query >*. В этом случае вместо ключевого слова *VALUES* и перечисления значений должен стоять оператор *SELECT*.

Рассмотрим пример. Добавим в таблицу Students записи из таблицы St1 о студентах группы 387. Предполагается, что заголовки у этих таблиц одинаковые и добавление новых значений не приведет к нарушению ограничения первичного ключа (значения StudID добавляемых записей отличны от имеющихся в Students значений этого поля):

```
INSERT INTO Students (StudID, FIO, Group)
SELECT * FROM St1
WHERE Group = 387
```

Значения полей существующих записей таблицы можно изменить с помощью оператора UPDATE, формат которого следующий:

```
UPDATE < table name >
SET column_1 = value_1 [, column_2 = value_2 ...]
[WHERE < predicate >]
```

Здесь *< table name >* – название обновляемой таблицы; *column_1* – название первого из обновляемых столбцов, *value_1* – присваиваемое ему значение

(константа или результат вычислений). Обновляемых столбцов может быть несколько.

Выражение *< predicate >* обозначает логическое условие. Если необязательная секция *WHERE* пропущена, обновляются все записи таблицы. Если эта секция присутствует, то обновляются только те записи, для которых *< predicate >* будет истинным. Составное условие формируется с помощью логических связей *AND* (логическое «и»), *OR* (логическое «или»), *NOT* (отрицание). Если необходимо обнести только одну конкретную запись, это можно сделать, указать в условии значение первичного ключа. Например:

```
UPDATE Students
SET FIO = 'Петровский П. П.', Group = 384
WHERE StudID = 130
```

В данном примере изменены фамилия и номер группы студента, идущего в БД под идентификатором 130. Рассмотрим другой пример. Пусть в таблице T1 в поле Price хранится информация о цене товара. Следующее выражение увеличивает все цены на 10%:

```
UPDATE T1 SET Price = Price * 1.1
```

Удаление строк из таблицы производится оператором *DELETE*:

```
DELETE FROM < table name >
[WHERE < predicate >]
```

Здесь *< table name >* – название таблицы, из которой удаляются данные; *< predicate >* – логическое условие. Если секция *WHERE* присутствует, удаляются только те записи, для которых *< predicate >* будет истинным. Например, удалим из таблицы Students всех Ивановых И.И. из группы 382:

```
DELETE FROM Students
WHERE Group = 382 AND FIO = 'Иванов И. И.'
```

Выполнение приведенного ниже выражения, удалит все данные из таблицы, но, в отличие от оператора *DROP TABLE*, сама таблица удалена не будет:

```
DELETE FROM Students
```

Нужно учитывать, что определенные для БД ограничения целостности (например, ограничение внешнего ключа) могут не позволить обнести или удалить значения некоторых записей.

Оператор *MERGE* позволяет изменить, добавить или удалить записи в таблице, на основании проверки условий относительно совпадения записей изменяемой таблицы и какой-то другой. Это может понадобиться, например, при синхронизации данных, хранящихся в разных таблицах.

Рассмотрим синтаксис оператора *MERGE*:

```
MERGE [INTO] < target_table > [[AS] table_alias]  
USING < table_source >  
ON < merge_search_condition >  
[WHEN MATCHED [AND < search_condition >]  
THEN < merge_matched >] [...n]  
[WHEN NOT MATCHED [BY TARGET]  
[AND < search_condition >]  
THEN < merge_not_matched >]  
[WHEN NOT MATCHED BY SOURCE  
[AND < search_condition >]  
THEN < merge_matched >] [...n];
```

Здесь [*INTO*] < *target_table* > определяет «целевую» таблицу или представление, данные из которой сравниваются с «исходной» таблицей, представлением или выражением, обозначенным как < *table_source* >. По результатам сравнения изменения происходят в целевой таблице. Условия, выполнение которых проверяется в отношении записей исходной и целевой таблиц, определяются выражением *ON* < *merge_search_condition* >. Целевой таблице может быть задан псевдоним (более короткое или понятное имя, далее используемое в выражении), указанный в формате как *table_alias*.

Действия, выполняемые при соответствии значений записей условиям, определяется необязательный раздел [*WHEN MATCHED* [*AND* < *search_condition* >]]. Одно выражение *MERGE* может иметь не больше двух разделов *WHEN MATCHED*. Если их два, в первом обязательно должны быть дополнительные условия: *AND* < *search_condition* >. Тогда второй *WHEN MATCHED* применяется только в тех случаях, если не применяется первый. Если имеются два *WHEN MATCHED*, один должен указывать действие *UPDATE* (изменить записи), а другой – действие *DELETE* (удалить).

Наличие раздела [*WHEN NOT MATCHED* [*BY TARGET*] [*AND* < *search_condition* >]] говорит, что если в целевой таблице нет строки их «исходной» таблицы (выражения) и такая строка соответствует дополнительным условиям < *search_condition* >, то она должна быть вставлена (*INSERT*) в таблицу *target_table* так, как определяется в *THEN* < *merge_not_matched* >. Инструкция *MERGE* может иметь только один раздел *WHEN NOT MATCHED BY TARGET*.

[*WHEN NOT MATCHED BY SOURCE* [*AND* < *search_condition* >]]
THEN < *merge_matched* >] определяет действия (*UPDATE* или *DELETE*) над строками, которые присутствуют в целевой таблице, но отсутствуют в исходной. Как и в случае с *WHEN MATCHED*, в одном выражении *MERGE*

может быть не больше двух разделов *WHEN NOT MATCHED BY SOURCE*, один из которых определяет изменение данных, другой – удаление.

Рассмотрим пример использования данного оператора. Пусть имеются две таблицы – Subj и NewSubj (табл. 3 и 4). В первой таблице перечислены читаемые учебные курсы, во второй приведены изменения, вводимые в учебные планы. Поле SubjId является первичным ключом в обеих таблицах.

Таблица 3. Таблица Subj

SubjId	SubjTitle	Comment
3_BD	Базы данных	3 курс
4_OS	Операционные системы	4 курс

Таблица 4. Таблица NewSubj

SubjId	SubjTitle	OldSubjTitle
2_Inf	Информатика	
3_BD	Основы баз данных	Базы данных

Пусть требуется согласовать содержимое таблиц таким образом, что если имеющемуся в таблице Subj курсу в таблице NewSubj дано новое название, то мы обновляем только название, а если в NewSubj появляется новый (по значению SubjId) курс, мы его добавляем в Subj. Это можно сделать следующим выражением:

```
MERGE INTO Subj USING NewSubj
ON Subj.SubjID = NewSubj.SubjId
WHEN MATCHED AND
Subj.SubjTitle <> NewSubj.SubjTitle
THEN UPDATE SET Subj.SubjTitle = NewSubj.SubjTitle
WHEN NOT MATCHED BY TARGET
THEN INSERT (SubjId, SubjTitle)
VALUES (NewSubj.SubjId, NewSubj.SubjTitle);
```

После выполнения данного выражения в таблице Subj будут значения, представленные в табл. 5. Нужно отметить, что аналогичный результат можно получить, последовательно выполнив запросы на обновление (UPDATE) и добавление (INSERT) требуемых данных. Однако с точки зрения производительности это будет менее эффективным решением.

Таблица 5.Таблица Subj после преобразований

SubjId	SubjTitle	Comment
2_Inf	Информатика	NULL
3_BD	Основы баз данных	3 курс
4_OS	Операционные системы	4 курс

Вопросы и задания к лекции

1. В чем состоит разница между типами данных CHAR(n), NCHAR(n), VARCHAR(n) и NVARCHAR(n)?
2. В чем заключаются основные отличия реляционных отношений от таблиц в SQL?
3. Какие задачи могут быть решены с помощью оператора ALTER TABLE?
4. Какой оператор SQL используется для добавления записей в таблицу?
5. Какой оператор SQL используется для изменения значений существующих в таблице записей?
6. Какой оператор используется для удаления записей из таблицы?

Лекция: Основные понятия технологии проектирования информационных систем (ИС)

Классификация ИС

Информация в современном мире превратилась в один из наиболее важных ресурсов, а информационные системы (ИС) стали необходимым инструментом практически во всех сферах деятельности.

Разнообразие задач, решаемых с помощью ИС, привело к появлению множества разнотипных систем, отличающихся принципами построения и заложенными в них правилами обработки информации.

Информационные системы можно **классифицировать** по целому ряду различных признаков. В основу рассматриваемой классификации положены наиболее существенные признаки, определяющие функциональные возможности и особенности построения современных систем. В зависимости от объема решаемых задач, используемых технических средств, организации функционирования, информационные системы делятся на ряд групп (классов) ([рис. 1.1](#)).

По типу хранимых данных ИС делятся на фактографические и документальные:

- *Фактографические системы* предназначены для хранения и обработки структурированных данных в виде чисел и текстов. Над такими данными можно выполнять различные операции.
- В *документальных системах* информация представлена в виде документов, состоящих из наименований, описаний, рефератов и текстов. Поиск по неструктурированным данным осуществляется с использованием семантических признаков. Отобранные документы предоставляются пользователю, а обработка данных в таких системах практически не производится.

Основываясь на **степени автоматизации информационных процессов** в системе управления фирмой, информационные системы делятся на ручные, автоматические и автоматизированные:

- *Ручные ИС* характеризуются отсутствием современных технических средств переработки информации и выполнением всех операций человеком.
- В *автоматических ИС* все операции по переработке информации выполняются без участия человека.
- *Автоматизированные ИС* предполагают участие в процессе обработки информации и человека, и технических средств, причем главная роль в выполнении рутинных операций обработки данных отводится компьютеру. Именно этот класс систем соответствует современному представлению понятия "информационная система".



Рис. 1.1. Классификация ИС

В зависимости от **характера обработки** данных ИС делятся на информационно-поисковые и информационно-решающие:

- *Информационно-поисковые* системы производят ввод, систематизацию, хранение, выдачу информации по запросу пользователя без сложных преобразований данных. Например, ИС библиотечного обслуживания, резервирования и продажи билетов на транспорте, бронирования мест в гостиницах и пр.
- *Информационно-решающие* системы осуществляют, кроме того, операции переработки информации по определенному алгоритму. По **характеру использования выходной информации** такие системы принято делить на *управляющие* и *советующие*. Результирующая информация управляющих ИС непосредственно трансформируется в принимаемые человеком решения. Для этих систем характерны задачи расчетного характера и обработка больших объемов данных. Например, ИС планирования производства или заказов, бухгалтерского учета. Советующие ИС вырабатывают информацию, которая принимается человеком к сведению и учитывается при формировании управленческих решений, а не инициирует конкретные действия. Эти системы имитируют интеллектуальные процессы обработки знаний, а не данных. Например, экспертные системы.

В зависимости от **сферы применения** различают следующие классы ИС:

- Информационные системы *организационного управления* – предназначены для автоматизации функций управленческого персонала как промышленных предприятий, так и непромышленных объектов (гостиниц, банков, магазинов и пр.). Основными функциями подобных систем являются: оперативный контроль и регулирование, оперативный учет и анализ, перспективное и оперативное планирование, бухгалтерский учет, управление сбытом, снабжением и другие экономические и организационные задачи.
- ИС *управления технологическими процессами* (ТП) – служат для автоматизации функций производственного персонала по контролю и управлению производственными операциями. В таких системах обычно предусматривается наличие развитых средств измерения параметров технологических процессов (температуры, давления, химического состава и т.п.), процедур контроля допустимости значений параметров и регулирования технологических процессов.
- ИС *автоматизированного проектирования* (САПР) – предназначены для автоматизации функций инженеров-проектировщиков, конструкторов, архитекторов, дизайнеров при создании новой техники или технологии. Основными функциями подобных систем являются: инженерные расчеты, создание графической документации (чертежей, схем, планов), создание проектной документации, моделирование проектируемых объектов.
- *Интегрированные* (корпоративные) ИС - используются для автоматизации всех функций фирмы и охватывают весь цикл работ от планирования деятельности до сбыта продукции. Они включают в себя ряд модулей (подсистем), работающих в едином информационном пространстве и выполняющих функции поддержки соответствующих направлений деятельности. Типовые задачи, решаемые модулями корпоративной системы, приведены в таблице 1.

Таблица 1. Типовые задачи, решаемые модулями корпоративной ИС

Подсистема маркетинга	Производственные подсистемы	Финансовые и учетные подсистемы	Подсистема кадров (человеческих ресурсов)	Прочие подсистемы (например, ИС руководства)
Исследование рынка и прогнозирование продаж	Планирование объемов работ и разработка календарных планов	Управление портфелем заказов	Анализ и прогнозирование потребности в трудовых ресурсах	Контроль за деятельностью фирмы
Управление продажами	Оперативный контроль и управление производством	Управление кредитной политикой	Ведение архивов записей о персонале	Выявление оперативных проблем
Рекомендации по производству новой продукции	Анализ работы оборудования	Разработка финансового плана	Анализ и планирование подготовки кадров	Анализ управленческих и стратегических ситуаций
Анализ и установление цены	Участие в формировании заказов поставщикам	Финансовый анализ и прогнозирование		Обеспечение процесса выработки стратегических

				решений
Учет заказов	Управление запасами	Контроль бюджета, бухгалтерский учет и расчет зарплаты		

Анализ современного состояния рынка ИС показывает устойчивую тенденцию роста спроса на информационные системы организационного управления. Причем спрос продолжает расти именно на интегрированные системы управления. Автоматизация отдельной функции, например, бухгалтерского учета или сбыта готовой продукции, считается уже пройденным этапом для многих предприятий.

В таблице 2 приведен перечень наиболее популярных в настоящее время программных продуктов для реализации ИС организационного управления различных классов.

Таблица 2. Наиболее популярные программные продукты для реализации ИС

Локальные системы	Малые интегрированные системы	Средние интегрированные системы	Крупные интегрированные системы (ИС)
<ul style="list-style-type: none"> • БЭСТ • Инотек • Инфософт • Супер-Менеджер • Турбо-Бухгалтер • Инфо-Бухгалтер 	<ul style="list-style-type: none"> • Concorde XAL Exact • NS-2000 Platinum PRO/MIS • Scala SunSystems • БЭСТ-ПРО • 1С-Предприятие • БОСС-Корпорация • Галактика • Парус • Ресурс • Эталон 	<ul style="list-style-type: none"> • Microsoft-Business Solutions - Navision, Microsoft-Business Solutions Axapta • J D Edwards (Robertson & Blums) • MFG-Pro (QAD/BMS) • SyteLine (СОКАП/SYMIX) 	<ul style="list-style-type: none"> • SAP/R3 (SAP AG) • Baan (Baan) • BPCS (ITS/SSA) • OEBS (Oracle E-Business Suite)

Существует *классификация ИС* в зависимости от **уровня управления**, на котором система используется.

Информационная система оперативного уровня - поддерживает исполнителей, обрабатывая данные о сделках и событиях (счета, накладные, зарплата, кредиты, поток сырья и материалов). Информационная система оперативного уровня является связующим звеном между фирмой и внешней средой.

Задачи, цели, источники информации и алгоритмы обработки на оперативном уровне заранее определены и в высокой степени структурированы.

Информационные системы специалистов - поддерживают работу с данными и знаниями, повышают продуктивность и производительность работы инженеров и проектировщиков. Задача подобных информационных систем - интеграция новых сведений в организацию и помощь в обработке бумажных документов.

Информационные системы уровня менеджмента - используются работниками среднего управленческого звена для мониторинга, контроля, принятия решений и администрирования. Основные функции этих информационных систем:

- сравнение текущих показателей с прошлыми;
- составление периодических отчетов за определенное время, а не выдача отчетов по текущим событиям, как на оперативном уровне;
- обеспечение доступа к архивной информации и т.д.

Стратегическая информационная система - компьютерная информационная система, обеспечивающая поддержку принятия решений по реализации стратегических перспективных целей развития организации.

Информационные системы стратегического уровня помогают высшему звену управленцев решать неструктурированные задачи, осуществлять долгосрочное планирование. Основная задача - сравнение происходящих во внешнем окружении изменений с существующим потенциалом фирмы. Они призваны создать общую среду компьютерной телекоммуникационной поддержки решений в неожиданно возникающих ситуациях. Используя самые совершенные программы, эти системы способны в любой момент предоставить информацию из многих источников. Некоторые стратегические системы обладают ограниченными аналитическими возможностями.

С точки зрения **программно-аппаратной реализации** можно выделить ряд типовых архитектур ИС.

Традиционные архитектурные решения основаны на использовании выделенных файлов-серверов или серверов баз данных. Существуют также варианты архитектур корпоративных информационных систем, базирующихся на технологии Internet (Intranet-приложения). Следующая разновидность архитектуры информационной системы основывается на концепции "хранилища данных" (DataWarehouse) - интегрированной информационной среды, включающей разнородные информационные ресурсы. И, наконец, для построения глобальных распределенных информационных приложений используется архитектура интеграции информационно-вычислительных компонентов на основе объектно-ориентированного подхода.

Индустрия разработки автоматизированных информационных систем управления зародилась в 1950-х - 1960-х годах и к концу века приобрела вполне законченные формы.

На первом этапе основным подходом в *проектировании ИС* был метод "снизу-вверх", когда система создавалась как набор приложений, наиболее важных в данный момент для поддержки деятельности предприятия. Основной целью этих проектов было не создание тиражируемых продуктов, а обслуживание текущих потребностей конкретного учреждения. Такой подход отчасти сохраняется и сегодня. В рамках "лоскутной автоматизации" достаточно хорошо обеспечивается поддержка отдельных функций, но практически полностью отсутствует стратегия развития комплексной системы автоматизации, а объединение функциональных подсистем превращается в самостоятельную и достаточно сложную проблему.

Создавая свои отделы и управления автоматизации, предприятия пытались "обустроиться" своими силами. Однако периодические изменения технологий работы и должностных инструкций, сложности, связанные с разными представлениями пользователей об одних и тех же данных, приводили к непрерывным доработкам программных продуктов для

удовлетворения все новых и новых пожеланий отдельных работников. Как следствие - и работа программистов, и создаваемые ИС вызвали недовольство руководителей и пользователей системы.

Следующий этап связан с осознанием того факта, что существует потребность в достаточно стандартных программных средствах автоматизации деятельности различных учреждений и предприятий. Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения бухгалтерского аналитического учета и технологических процессов. Системы начали проектироваться "сверху-вниз", т.е. в предположении, что одна программа должна удовлетворять потребности многих пользователей.

Сама идея использования универсальной программы накладывает существенные ограничения на возможности разработчиков по формированию структуры базы данных, экранных форм, по выбору алгоритмов расчета. Заложенные "сверху" жесткие рамки не дают возможности гибко адаптировать систему к специфике деятельности конкретного предприятия: учесть необходимую глубину аналитического и производственно-технологического учета, включить необходимые процедуры обработки данных, обеспечить интерфейс каждого рабочего места с учетом функций и технологии работы конкретного пользователя. Решение этих задач требует серьезных доработок системы. Таким образом, материальные и временные затраты на внедрение системы и ее доводку под требования заказчика обычно значительно превышают запланированные показатели.

Согласно статистическим данным, собранным Standish Group (США), из 8380 проектов, обследованных в США в 1994 году, неудачными оказались более 30% проектов, общая стоимость которых превышала 80 миллиардов долларов. При этом оказались выполненными в срок лишь 16% от общего числа проектов, а перерасход средств составил 189% от запланированного бюджета.

В то же время, заказчики ИС стали выдвигать все больше требований, направленных на обеспечение возможности комплексного использования корпоративных данных в управлении и планировании своей деятельности.

Таким образом, возникла насущная необходимость формирования новой методологии построения информационных систем.

Цель такой методологии заключается в регламентации процесса проектирования ИС и обеспечении управления этим процессом с тем, чтобы гарантировать выполнение требований как к самой ИС, так и к характеристикам процесса разработки. Основными задачами, решению которых должна способствовать методология проектирования корпоративных ИС, являются следующие:

- обеспечивать создание корпоративных ИС, отвечающих целям и задачам организации, а также предъявляемым требованиям по автоматизации деловых процессов заказчика;
- гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета проекта;
- поддерживать удобную дисциплину сопровождения, модификации и наращивания системы;
- обеспечивать преемственность разработки, т.е. использование в разрабатываемой ИС существующей информационной инфраструктуры организации (задела в области информационных технологий).

Внедрение методологии должно приводить к снижению сложности процесса создания ИС за счет полного и точного описания этого процесса, а также применения современных методов и технологий создания ИС на всем жизненном цикле ИС - от замысла до реализации.

Проектирование ИС охватывает три основные области:

- проектирование объектов данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Проектирование информационных систем всегда начинается с определения цели проекта. В общем виде цель проекта можно определить как решение ряда взаимосвязанных задач, включающих в себя обеспечение на момент запуска системы и в течение всего времени ее эксплуатации:

- требуемой функциональности системы и уровня ее адаптивности к изменяющимся условиям функционирования;
- требуемой пропускной способности системы;
- требуемого времени реакции системы на запрос;
- безотказной работы системы;
- необходимого уровня безопасности;
- простоты эксплуатации и поддержки системы.

Согласно современной методологии, процесс создания ИС представляет собой процесс построения и последовательного преобразования ряда согласованных моделей на всех этапах жизненного цикла (ЖЦ) ИС. На каждом этапе ЖЦ создаются специфичные для него модели - организации, требований к ИС, проекта ИС, требований к приложениям и т.д. Модели формируются рабочими группами команды проекта, сохраняются и накапливаются в репозитории проекта. Создание моделей, их контроль, преобразование и предоставление в коллективное пользование осуществляется с использованием специальных программных инструментов - CASE-средств.

Процесс создания ИС делится на ряд **этапов** (стадий [\[1\]](#)), ограниченных некоторыми временными рамками и заканчивающихся выпуском конкретного продукта (моделей, программных продуктов, документации и пр.).

Обычно выделяют следующие *этапы создания ИС*: формирование требований к системе, проектирование, реализация, тестирование, ввод в действие, эксплуатация и сопровождение [\[1\]](#) [\[2\]](#). (Последние два этапа далее не рассматриваются, поскольку выходят за рамки тематики курса.)

Начальным этапом процесса создания ИС является моделирование бизнес-процессов, протекающих в организации и реализующих ее цели и задачи. Модель организации, описанная в терминах бизнес-процессов и бизнес-функций, позволяет сформулировать основные требования к ИС. Это фундаментальное положение методологии обеспечивает объективность в выработке требований к проектированию системы. Множество моделей описания требований к ИС затем преобразуется в систему моделей, описывающих концептуальный проект ИС. Формируются модели архитектуры ИС, требований к

программному обеспечению (ПО) и информационному обеспечению (ИО). Затем формируется архитектура ПО и ИО, выделяются корпоративные БД и отдельные приложения, формируются модели требований к приложениям и проводится их разработка, тестирование и интеграция.

Целью начальных *этапов создания ИС*, выполняемых на стадии анализа деятельности организации, является формирование требований к ИС, корректно и точно отражающих цели и задачи организации-заказчика. Чтобы специфицировать процесс создания ИС, отвечающей потребностям организации, нужно выяснить и четко сформулировать, в чем заключаются эти потребности. Для этого необходимо определить требования заказчиков к ИС и отобразить их на языке моделей в требования к разработке проекта ИС так, чтобы обеспечить соответствие целям и задачам организации.

Задача формирования требований к ИС является одной из наиболее ответственных, трудно формализуемых и наиболее дорогих и тяжелых для исправления в случае ошибки. Современные инструментальные средства и программные продукты позволяют достаточно быстро создавать ИС по готовым требованиям. Но зачастую эти системы не удовлетворяют заказчиков, требуют многочисленных доработок, что приводит к резкому удорожанию фактической стоимости ИС. Основной причиной такого положения является неправильное, неточное или неполное определение требований к ИС на этапе анализа.

На этапе проектирования прежде всего формируются модели данных. Проектировщики в качестве исходной информации получают результаты анализа. Построение логической и физической моделей данных является основной частью проектирования базы данных. Полученная в процессе анализа информационная модель сначала преобразуется в логическую, а затем в физическую модель данных.

Параллельно с проектированием схемы базы данных выполняется проектирование процессов, чтобы получить спецификации (описания) всех модулей ИС. Оба эти процесса проектирования тесно связаны, поскольку часть бизнес-логики обычно реализуется в базе данных (ограничения, триггеры, хранимые процедуры). Главная цель проектирования процессов заключается в отображении функций, полученных на этапе анализа, в модули информационной системы. При проектировании модулей определяют интерфейсы программ: разметку меню, вид окон, горячие клавиши и связанные с ними вызовы.

Конечными продуктами этапа проектирования являются:

- схема базы данных (на основании ER-модели, разработанной на этапе анализа);
- набор спецификаций модулей системы (они строятся на базе моделей функций).

Кроме того, на этапе проектирования осуществляется также разработка архитектуры ИС, включающая в себя выбор платформы (платформ) и операционной системы (операционных систем). В неоднородной ИС могут работать несколько компьютеров на разных аппаратных платформах и под управлением различных операционных систем. Кроме выбора платформы, на этапе проектирования определяются следующие характеристики архитектуры:

- будет ли это архитектура "файл-сервер" или "клиент-сервер";
- будет ли это 3-уровневая архитектура со следующими слоями: сервер, ПО промежуточного слоя (сервер приложений), клиентское ПО;

- будет ли база данных централизованной или распределенной. Если база данных будет распределенной, то какие механизмы поддержки согласованности и актуальности данных будут использоваться;
- будет ли база данных однородной, то есть, будут ли все серверы баз данных продуктами одного и того же производителя (например, все серверы только Oracle или все серверы только DB2 UDB). Если база данных не будет однородной, то какое ПО будет использовано для обмена данными между СУБД разных производителей (уже существующее или разработанное специально как часть проекта);
- будут ли для достижения должной производительности использоваться параллельные серверы баз данных (например, Oracle Parallel Server, DB2 UDB и т.п.).

Этап проектирования завершается разработкой технического проекта ИС.

На этапе реализации осуществляется создание программного обеспечения системы, установка технических средств, разработка эксплуатационной документации.

Этап тестирования обычно оказывается распределенным во времени.

После завершения разработки отдельного модуля системы выполняют автономный тест, который преследует две основные цели:

- обнаружение отказов модуля (жестких сбоев);
- соответствие модуля спецификации (наличие всех необходимых функций, отсутствие лишних функций).

После того как автономный тест успешно пройден, модуль включается в состав разработанной части системы и группа сгенерированных модулей проходит тесты связей, которые должны отследить их взаимное влияние.

Далее группа модулей тестируется на надежность работы, то есть проходят, во-первых, тесты имитации отказов системы, а во-вторых, тесты наработки на отказ. Первая группа тестов показывает, насколько хорошо система восстанавливается после сбоев программного обеспечения, отказов аппаратного обеспечения. Вторая группа тестов определяет степень устойчивости системы при штатной работе и позволяет оценить время безотказной работы системы. В комплект тестов устойчивости должны входить тесты, имитирующие пиковую нагрузку на систему.

Затем весь комплект модулей проходит системный тест - тест внутренней приемки продукта, показывающий уровень его качества. Сюда входят тесты функциональности и тесты надежности системы.

Последний тест информационной системы - приемо-сдаточные испытания. Такой тест предусматривает показ информационной системы заказчику и должен содержать группу тестов, моделирующих реальные бизнес-процессы, чтобы показать соответствие реализации требованиям заказчика.

Необходимость контролировать процесс создания ИС, гарантировать достижение целей разработки и соблюдение различных ограничений (бюджетных, временных и пр.) привело к широкому использованию в этой сфере методов и средств программной инженерии: структурного анализа, объектно-ориентированного моделирования, CASE-систем.

Лекция: Жизненный цикл программного обеспечения ИС

Методология проектирования информационных систем описывает процесс создания и сопровождения систем в виде *жизненного цикла* (ЖЦ) ИС, представляя его как некоторую последовательность стадий и выполняемых на них процессов. Для каждого этапа определяются состав и последовательность выполняемых работ, получаемые результаты, методы и средства, необходимые для выполнения работ, роли и ответственность участников и т.д. Такое формальное описание ЖЦ ИС позволяет спланировать и организовать процесс коллективной разработки и обеспечить управление этим процессом.

Жизненный цикл ИС можно представить как ряд событий, происходящих с системой в процессе ее создания и использования.

Модель жизненного цикла отражает различные состояния системы, начиная с момента возникновения необходимости в данной ИС и заканчивая моментом ее полного выхода из употребления. *Модель жизненного цикла* - структура, содержащая процессы, действия и задачи, которые осуществляются в ходе разработки, функционирования и сопровождения программного продукта в течение всей жизни системы, от определения требований до завершения ее использования.

В настоящее время известны и используются следующие *модели жизненного цикла*:

- **Каскадная модель** ([рис. 2.1](#)) предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
- **Поэтапная модель с промежуточным контролем** ([рис. 2.2](#)). Разработка ИС ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах; время жизни каждого из этапов растягивается на весь период разработки.
- **Спиральная модель** ([рис. 2.3](#)). На каждом витке спирали выполняется создание очередной версии продукта, уточняются требования проекта, определяется его качество и планируются работы следующего витка. Особое внимание уделяется начальным этапам разработки - анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования).



Рис. 2.1. Каскадная модель ЖЦ ИС



Рис. 2.2. Поэтапная модель с промежуточным контролем



Рис. 2.3. Спиральная модель ЖЦ ИС

На практике наибольшее распространение получили две основные *модели жизненного цикла*:

- *каскадная модель* (характерна для периода 1970-1985 гг.);
- *спиральная модель* (характерна для периода после 1986 г.).

В ранних проектах достаточно простых ИС каждое приложение представляло собой единый, функционально и информационно независимый блок. Для разработки такого типа приложений эффективным оказался каскадный способ. Каждый этап завершался после полного выполнения и документального оформления всех предусмотренных работ.

Можно выделить следующие положительные стороны применения каскадного подхода:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логической последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

Каскадный подход хорошо зарекомендовал себя при построении относительно простых ИС, когда в самом начале разработки можно достаточно точно и полно сформулировать все требования к системе. Основным недостатком этого подхода является то, что

реальный процесс создания системы никогда полностью не укладывается в такую жесткую схему, постоянно возникает потребность в возврате к предыдущим этапам и уточнении или пересмотре ранее принятых решений. В результате реальный процесс создания ИС оказывается соответствующим *поэтапной модели с промежуточным контролем*.

Однако и эта схема не позволяет оперативно учитывать возникающие изменения и уточнения требований к системе. Согласование результатов разработки с пользователями производится только в точках, планируемых после завершения каждого этапа работ, а общие требования к ИС зафиксированы в виде технического задания на все время ее создания. Таким образом, пользователи зачастую получают систему, не удовлетворяющую их реальным потребностям.

Спиральная модель ЖЦ была предложена для преодоления перечисленных проблем. На этапах анализа и проектирования реализуемость технических решений и степень удовлетворения потребностей заказчика проверяется путем создания прототипов. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии системы. Это позволяет уточнить требования, цели и характеристики проекта, определить качество разработки, спланировать работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который удовлетворяет действительным требованиям заказчика и доводится до реализации.

Итеративная разработка отражает объективно существующий спиральный цикл создания сложных систем. Она позволяет переходить на следующий этап, не дожидаясь полного завершения работы на текущем и решить главную задачу - как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения вводятся временные ограничения на каждый из этапов *жизненного цикла*, и переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. Планирование производится на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Несмотря на настойчивые рекомендации компаний - вендоров и экспертов в области проектирования и разработки ИС, многие компании продолжают использовать *каскадную модель* вместо какого-либо варианта итерационной модели. Основные причины, по которым *каскадная модель* сохраняет свою популярность, следующие [3]:

1. **Привычка** - многие ИТ-специалисты получали образование в то время, когда изучалась только *каскадная модель*, поэтому она используется ими и в наши дни.
2. **Иллюзия снижения рисков** участников проекта (заказчика и исполнителя). *Каскадная модель* предполагает разработку законченных продуктов на каждом этапе: технического задания, технического проекта, программного продукта и пользовательской документации. Разработанная документация позволяет не только определить требования к продукту следующего этапа, но и определить обязанности сторон, объем работ и сроки, при этом окончательная оценка сроков и стоимости проекта производится на начальных этапах, после завершения обследования. Очевидно, что если требования к информационной системе меняются в ходе реализации проекта, а качество документов оказывается невысоким (требования неполны и/или противоречивы), то в действительности использование *каскадной модели* создает лишь иллюзию определенности и на деле увеличивает риски, уменьшая лишь ответственность участников проекта. При формальном подходе менеджер проекта реализует только те требования, которые содержатся в спецификации, опирается на документ, а не на реальные потребности бизнеса. Есть два основных типа контрактов на разработку ПО.

Первый тип предполагает выполнение определенного объема работ за определенную сумму в определенные сроки (*fixed price*). Второй тип предполагает повременную оплату работы (*time work*). Выбор того или иного типа контракта зависит от степени определенности задачи. *Каскадная модель* с определенными этапами и их результатами лучше приспособлена для заключения контракта с оплатой по результатам работы, а именно этот тип контрактов позволяет получить полную оценку стоимости проекта до его завершения. Более вероятно заключение контракта с повременной оплатой на небольшую систему, с относительно небольшим весом в структуре затрат предприятия. Разработка и внедрение интегрированной информационной системы требует существенных финансовых затрат, поэтому используются контракты с фиксированной ценой, и, следовательно, *каскадная модель* разработки и внедрения. *Спиральная модель* чаще применяется при разработке информационной системы силами собственного отдела ИТ предприятия.

3. **Проблемы внедрения** при использовании итерационной модели. В некоторых областях *спиральная модель* не может применяться, поскольку невозможно использование/тестирование продукта, обладающего неполной функциональностью (например, военные разработки, атомная энергетика и т.д.). Поэтапное итерационное внедрение информационной системы для бизнеса возможно, но сопряжено с организационными сложностями (перенос данных, интеграция систем, изменение бизнес-процессов, учетной политики, обучение пользователей). Трудозатраты при поэтапном итерационном внедрении оказываются значительно выше, а управление проектом требует настоящего искусства. Предвидя указанные сложности, заказчики выбирают *каскадную модель*, чтобы "внедрять систему один раз".

Каждая из стадий создания системы предусматривает выполнение определенного объема работ, которые представляются в виде *процессов ЖЦ*. *Процесс* определяется как совокупность взаимосвязанных действий, преобразующих входные данные в выходные. Описание каждого процесса включает в себя перечень решаемых задач, исходных данных и результатов.

Существует целый ряд стандартов, регламентирующих ЖЦ ПО, а в некоторых случаях и процессы разработки.

Значительный вклад в теорию проектирования и разработки информационных систем внесла компания IBM, предложив еще в середине 1970-х годов методологию BSP (*Business System Planning* - методология организационного планирования). Метод структурирования информации с использованием матриц пересечения бизнес-процессов, функциональных подразделений, функций систем обработки данных (информационных систем), информационных объектов, документов и баз данных, предложенный в BSP, используется сегодня не только в ИТ-проектах, но и проектах по реинжинирингу бизнес-процессов, изменению организационной структуры. Важнейшие шаги процесса BSP, их последовательность (получить поддержку высшего руководства, определить процессы предприятия, определить классы данных, провести интервью, обработать и организовать данные интервью) можно встретить практически во всех формальных методиках, а также в проектах, реализуемых на практике.

Среди наиболее известных стандартов можно выделить следующие:

- ГОСТ 34.601-90 - распространяется на автоматизированные системы и устанавливает стадии и этапы их создания. Кроме того, в стандарте содержится описание содержания работ на каждом этапе. Стадии и этапы работы, закрепленные в стандарте, в большей степени соответствуют *каскадной модели* жизненного цикла [4].
- ISO/IEC 12207:1995 - стандарт на процессы и организацию *жизненного цикла*. Распространяется на все виды заказного ПО. Стандарт не содержит описания фаз, стадий и этапов [5].

- Custom Development Method (методика Oracle) по разработке прикладных информационных систем - технологический материал, детализированный до уровня заготовок проектных документов, рассчитанных на использование в проектах с применением Oracle. Применяется CDM для классической модели ЖЦ (предусмотрены все работы/задачи и этапы), а также для технологий "быстрой разработки" (Fast Track) или "облегченного подхода", рекомендуемых в случае малых проектов.
- Rational Unified Process (RUP) предлагает итеративную модель разработки, включающую четыре фазы: начало, исследование, построение и внедрение. Каждая фаза может быть разбита на этапы (итерации), в результате которых выпускается версия для внутреннего или внешнего использования. Прохождение через четыре основные фазы называется циклом разработки, каждый цикл завершается генерацией версии системы. Если после этого работа над проектом не прекращается, то полученный продукт продолжает развиваться и снова минует те же фазы. Суть работы в рамках RUP - это создание и сопровождение моделей на базе UML [6].
- Microsoft Solution Framework (MSF) сходна с RUP, так же включает четыре фазы: анализ, проектирование, разработка, стабилизация, является итерационной, предполагает использование объектно-ориентированного моделирования. MSF в сравнении с RUP в большей степени ориентирована на разработку бизнес-приложений.
- Extreme Programming (XP). Экстремальное программирование (самая новая среди рассматриваемых методологий) сформировалось в 1996 году. В основе методологии командная работа, эффективная коммуникация между заказчиком и исполнителем в течение всего проекта по разработке ИС, а разработка ведется с использованием последовательно дорабатываемых прототипов.

В соответствии с базовым международным стандартом ISO/IEC 12207 все *процессы ЖЦ ПО* делятся на три группы:

1. **Основные процессы:**
 - приобретение;
 - поставка;
 - разработка;
 - эксплуатация;
 - сопровождение.
2. **Вспомогательные процессы:**
 - документирование;
 - управление конфигурацией;
 - обеспечение качества;
 - разрешение проблем;
 - аудит;
 - аттестация;
 - совместная оценка;
 - верификация.
3. **Организационные процессы:**
 - создание инфраструктуры;
 - управление;
 - обучение;
 - усовершенствование.

В [таблице 2.1](#) приведены ориентировочные описания основных процессов ЖЦ. Вспомогательные процессы предназначены для поддержки выполнения основных процессов, обеспечения качества проекта, организации верификации, проверки и тестирования ПО. Организационные процессы определяют действия и задачи, выполняемые как заказчиком, так и разработчиком проекта для управления своими процессами.

Для поддержки практического применения стандарта ISO/IEC 12207 разработан ряд технологических документов: Руководство для ISO/IEC 12207 (ISO/IEC TR 15271:1998 Information technology - Guide for ISO/IEC 12207) и Руководство по применению ISO/IEC 12207 к управлению проектами (ISO/IEC TR 16326:1999 Software engineering - Guide for the application of ISO/IEC 12207 to project management).

Таблица 2.1. Содержание основных процессов ЖЦ ПО ИС (ISO/IEC 12207)

Процесс (исполнитель процесса)	Действия	Вход	Результат
Приобретение (заказчик)	<ul style="list-style-type: none"> • Инициирование • Подготовка заявочных предложений • Подготовка договора • Контроль деятельности поставщика • Приемка ИС 	<ul style="list-style-type: none"> • Решение о начале работ по внедрению ИС • Результаты обследования деятельности заказчика • Результаты анализа рынка ИС/ тендера • План поставки/ разработки • Комплексный тест ИС 	<ul style="list-style-type: none"> • Техничко-экономическое обоснование внедрения ИС • Техническое задание на ИС • Договор на поставку/ разработку • Акты приемки этапов работы • Акт приемно-сдаточных испытаний
Поставка (разработчик ИС)	<ul style="list-style-type: none"> • Инициирование • Ответ на заявочные предложения • Подготовка договора • Планирование исполнения • Поставка ИС 	<ul style="list-style-type: none"> • Техническое задание на ИС • Решение руководства об участии в разработке • Результаты тендера • Техническое задание на ИС • План управления проектом • Разработанная ИС и документация 	<ul style="list-style-type: none"> • Решение об участии в разработке • Коммерческие предложения/ конкурсная заявка • Договор на поставку/ разработку • План управления проектом • Реализация/ корректировка • Акт приемно-сдаточных испытаний
Разработка (разработчик ИС)	<ul style="list-style-type: none"> • Подготовка • Анализ требований к ИС • Проектирование архитектуры ИС • Разработка требований к ПО • Проектирование архитектуры ПО • Детальное проектирование ПО • Кодирование и 	<ul style="list-style-type: none"> • Техническое задание на ИС • Техническое задание на ИС, модель ЖЦ • Подсистемы ИС • Спецификации требования к компонентам ПО • Архитектура ПО • Материалы детального проектирования 	<ul style="list-style-type: none"> • Используемая модель ЖЦ, стандарты разработки • План работ • Состав подсистем, компоненты оборудования • Спецификации требования к компонентам ПО • Состав

	<ul style="list-style-type: none"> тестирование ПО • Интеграция ПО и квалификационное тестирование ПО • Интеграция ИС и квалификационное тестирование ИС 	<ul style="list-style-type: none"> ПО • План интеграции ПО, тесты • Архитектура ИС, ПО, документация на ИС, тесты 	<ul style="list-style-type: none"> компонентов ПО, интерфейсы с БД, план интеграции ПО • Проект БД, спецификации интерфейсов между компонентами ПО, требования к тестам • Тексты модулей ПО, акты автономного тестирования • Оценка соответствия комплекса ПО требованиям ТЗ • Оценка соответствия ПО, БД, технического комплекса и комплекта документации требованиям ТЗ
--	---	--	--

Позднее был разработан и в 2002 г. опубликован стандарт на процессы *жизненного цикла* систем (ISO/IEC 15288 System life cycle processes). К разработке стандарта были привлечены специалисты различных областей: системной инженерии, программирования, управления качеством, человеческими ресурсами, безопасностью и пр. Был учтен практический опыт создания систем в правительственных, коммерческих, военных и академических организациях. Стандарт применим для широкого класса систем, но его основное предназначение - поддержка создания компьютеризированных систем.

Согласно стандарту ISO/IEC серии 15288 [7] в структуру ЖЦ следует включать следующие группы процессов:

1. Договорные процессы:

- приобретение (внутренние решения или решения внешнего поставщика);
- поставка (внутренние решения или решения внешнего поставщика).

2. Процессы предприятия:

- управление окружающей средой предприятия;
- инвестиционное управление;
- управление ЖЦ ИС;
- управление ресурсами;
- управление качеством.

3. Проектные процессы:

- планирование проекта;
- оценка проекта;
- контроль проекта;
- управление рисками;
- управление конфигурацией;
- управление информационными потоками;
- принятие решений.

4. Технические процессы:

- определение требований;
- анализ требований;
- разработка архитектуры;
- внедрение;
- интеграция;
- верификация;
- переход;
- аттестация;
- эксплуатация;
- сопровождение;
- утилизация.

5. Специальные процессы:

- определение и установка взаимосвязей исходя из задач и целей.

Стадии создания системы, предусмотренные в стандарте ISO/IEC 15288, несколько отличаются от рассмотренных выше. Перечень стадий и основные результаты, которые должны быть достигнуты к моменту их завершения, приведены в [таблице 2.2](#).

Таблица 2.2. Стадии создания систем (ISO/IEC 15288)

№ п/п	Стадия	Описание
1	Формирование концепции	Анализ потребностей, выбор концепции и проектных решений
2	Разработка	Проектирование системы
3	Реализация	Изготовление системы
4	Эксплуатация	Ввод в эксплуатацию и использование системы
5	Поддержка	Обеспечение функционирования системы
6	Снятие с эксплуатации	Прекращение использования, демонтаж, архивирование системы

Лекция: Проектирование ИС

План:

1. Методология и технология проектирования ИС.
2. Каноническое проектирование ИС.
3. Типовое проектирование ИС.

Литература:

1. Гагарина Л.Г., Киселев Д.В., Федотова Е.Л. Разработка и эксплуатация автоматизированных информационных систем: учеб. пособие / Под ред. Проф. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2007. – 386 с.: ил. – (Профессиональное образование). – с. 41-67.
2. Информационные системы / Петров В.Н. – СПб.: Питер, 2002. – 688 с.: ил. – с. 58-89.

Методология и технология проектирования ИС

Современные методологии и реализующие их технологии проектирования ИС поставляются в электронном виде вместе с CASE-средствами и включают библиотеки процессов, шаблонов, методов, моделей и других компонентов, предназначенных для построения ПО того класса систем, на который ориентирована методология.

Технология проектирования ИС – это совокупность методов и средств проектирования ИС, а также методов и средств организации проектирования (управление процессом создания и модернизации проекта ИС). В основе технологии проектирования лежит технологический процесс (ТП), который определяет действия, их последовательность, состав исполнителей, средства и ресурсы, требуемые для выполнения этих действий.

Проектирование ИС представляет собой совокупность последовательно-параллельных, связанных и соподчиненных цепочек действий, каждое из которых может иметь свой предмет. Действия, которые выполняются при проектировании ИС, могут быть определены как неделимые технологические операции или как подпроцессы технологических операций. Все действия могут быть проектировочными, которые формируют или модифицируют результаты проектирования, и оценочными, которые вырабатывают по установленным критериям оценки результатов проектирования.

Т.о., технология проектирования задается регламентированной последовательностью технологических операций, выполняемых в процессе создания проекта на основе того или иного метода.

Предметом выбираемой технологии проектирования должно служить отражение взаимосвязанных процессов проектирования на всех стадиях жизненного цикла ИС.

Основные требования, предъявляемые к выбираемой технологии проектирования, следующие:

- Созданный с помощью этой технологии проект должен отвечать требованиям заказчиков.

- Технология должна максимально отражать все тапы цикла жизни проекта.
- Технология должна обеспечить минимальные трудовые и стоимостные затраты на проектирование и сопровождение проекта.
- Технология должна способствовать росту производительности труда проектировщика.
- Технология должна обеспечить надежность процесса проектирования и эксплуатации проекта.
- Технология должна способствовать простому ведению проектной документации.

Технология проектирования ИС реализует определенную методологию проектирования. Методология предполагает наличие концепции, принципов проектирования и реализуется набором методов и средств. Методы проектирования ИС можно классифицировать по степени использования средств автоматизации, типовых проектных решений, адаптивности к предполагаемым изменениям.

По *степени автоматизации* различают:

- Ручное проектирование, при котором проектирование компонентов ИС осуществляется без использования специальных инструментальных программных средств; программирование производится на алгоритмических языках.
- Компьютерное проектирование, при котором генерация или конфигурация (настройка) проектных решений производится с использованием специальных инструментальных программных средств.

По *степени использования* типовых проектных решений различают:

- Оригинальное (индивидуальное) проектирование, когда проектные решения разрабатываются «с нуля» в соответствии с требованиями к ИС.
- Типовое проектирование выполняется на основе готовых решений и является обобщением опыта, полученного ранее при создании родственных проектов.

По *степени адаптивности* проектных решений выделяют методы:

- Реконструкция – адаптация проектных решений выполняется путем переработки соответствующих компонентов (перепрограммирование модулей).
- Параметризация – проектные решения настраиваются в соответствии с заданными и изменяемыми параметрами.
- Реструктуризация модели – изменяется модель предметной области, что приводит к автоматическому переформатированию проектных решений.

Сочетание различных признаков классификации методов проектирования обуславливает характер используемой технологии проектирования ИС. Выделяются два основных класса технологии проектирования: **каноническая** и **индустриальная**. Индустриальная технология разбивается на два подкласса: **автоматизированное** (использование CASE-технологий) и **типовое** (параметрически-ориентированное или модельно-ориентированное) проектирование (см.

Таблица 1).

Таблица 1. Характеристика классов технологий проектирования

Класс технологии проектирования	Степень автоматизации	Степень типизации	Степень адаптивности
Каноническое проектирование	Ручное проектирование	Оригинальное проектирование	Реконструкция
Индустриальное автоматизированное проектирование	Компьютерное проектирование	Оригинальное проектирование	Реструктуризация модели
Индустриальное типовое проектирование	Компьютерное проектирование	Типовое сборочное проектирование	Параметризация и реструктуризация модели

Каноническое проектирование ИС

Организация канонического проектирования ИС ориентирована на использование главным образом каскадной модели жизненного цикла ИС. Стадии и этапы работы описаны в стандарте ГОСТ 34.601-90.

В зависимости от сложности объекта автоматизации и набора задач, требующих решения при создании конкретной ИС, стадии и этапы работ могут иметь различную трудоемкость. Допускается объединять последовательные этапы и даже исключать некоторые из них на любой стадии проекта. Допускается также начинать выполнение работ следующей стадии до окончания предыдущей.

Стадии и этапы создания ИС, выполняемые организациями-участниками, прописываются в договорах и *технических заданиях* на выполнение работ:

Стадия 1. Формирование требований к ИС.

На начальной стадии проектирования выделяют следующие этапы работ:

- *обследование* объекта и обоснование необходимости создания ИС;
- формирование требований пользователей к ИС;
- оформление отчета о выполненной работе и *тактико-технического задания* на разработку.

Стадия 2. Разработка концепции ИС.

- изучение объекта автоматизации;
- проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции ИС, удовлетворяющих требованиям пользователей;
- оформление отчета и утверждение концепции.

Стадия 3. Техническое задание.

- разработка и утверждение *технического задания* на создание ИС.

Стадия 4. Эскизный проект.

- разработка предварительных проектных решений по системе и ее частям;
- разработка эскизной документации на ИС и ее части.

Стадия 5. Технический проект.

- разработка проектных решений по системе и ее частям;
- разработка документации на ИС и ее части;
- разработка и оформление документации на поставку комплектующих изделий;
- разработка заданий на проектирование в смежных частях проекта.

Стадия 6. Рабочая документация.

- разработка *рабочей документации* на ИС и ее части;
- разработка и адаптация программ.

Стадия 7. Ввод в действие.

- подготовка объекта автоматизации;
- подготовка персонала;
- комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);
- строительно-монтажные работы;
- пусконаладочные работы;
- проведение *предварительных испытаний*;
- проведение *опытной эксплуатации*;
- проведение *приемочных испытаний*.

Стадия 8. Сопровождение ИС.

- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

Обследование – это изучение и диагностический анализ организационной структуры предприятия, его деятельности и существующей системы обработки информации. Материалы, полученные в результате *обследования*, используются для:

- обоснования разработки и поэтапного внедрения систем;
- составления *технического задания* на разработку систем;
- разработки технического и рабочего проектов систем.

На этапе *обследования* целесообразно выделить две составляющие: определение стратегии внедрения ИС и детальный анализ деятельности организации.

Основная задача первого этапа *обследования* – оценка реального объема проекта, его целей и задач на основе выявленных функций и информационных элементов автоматизируемого объекта высокого уровня. Эти задачи могут быть реализованы или заказчиком ИС самостоятельно, или с привлечением консалтинговых организаций. Этап предполагает тесное взаимодействие с основными потенциальными пользователями системы и бизнес-экспертами. Основная задача взаимодействия – получить полное и однозначное понимание требований заказчика. Как правило, нужная информация может быть получена в результате интервью, бесед или семинаров с руководством, экспертами и пользователями.

По завершении этой стадии *обследования* появляется возможность определить вероятные технические подходы к созданию системы и оценить затраты на ее реализацию (затраты на аппаратное обеспечение, закупаемое программное обеспечение и разработку нового программного обеспечения).

Результатом этапа определения стратегии является документ (*технико-экономическое обоснование проекта*), где четко сформулировано, что получит заказчик, если согласится финансировать проект, когда он получит готовый продукт (график выполнения работ) и сколько это будет стоить (для крупных проектов должен быть составлен график финансирования на разных этапах работ). В документе желательно отразить не только затраты, но и выгоду проекта, например время окупаемости проекта, ожидаемый экономический эффект (если его удастся оценить).

Ориентировочное содержание этого документа:

- ограничения, риски, критические факторы, которые могут повлиять на успешность проекта;
- совокупность условий, при которых предполагается эксплуатировать будущую систему: архитектура системы, аппаратные и программные ресурсы, условия функционирования, обслуживающий персонал и пользователи системы;
- сроки завершения отдельных этапов, форма приемки/сдачи работ, привлекаемые ресурсы, меры по защите информации;
- описание выполняемых системой функций;
- возможности развития системы;
- информационные объекты системы;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам ПО, требования к СУБД;
- что не будет реализовано в рамках проекта.

На этапе детального анализа деятельности организации изучаются задачи, обеспечивающие реализацию функций управления, организационная структура, штаты и содержание работ по управлению предприятием, а также характер подчиненности вышестоящим органам управления. На этом этапе должны быть выявлены:

- инструктивно-методические и директивные материалы, на основании которых определяются состав подсистем и перечень задач;
- возможности применения новых методов решения задач.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

- функции - информация о событиях и процессах, которые происходят в бизнесе;
- сущности - информация о вещах, имеющих значение для организации и о которых что-то известно.

При изучении каждой функциональной задачи управления определяются:

- наименование задачи; сроки и периодичность ее решения;
- степень формализуемости задачи;
- источники информации, необходимые для решения задачи;
- показатели и их количественные характеристики;
- порядок корректировки информации;

- действующие алгоритмы расчета показателей и возможные методы контроля;
- действующие средства сбора, передачи и обработки информации;
- действующие средства связи;
- принятая точность решения задачи;
- трудоемкость решения задачи;
- действующие формы представления исходных данных и результатов их обработки в виде документов;
- потребители результатной информации по задаче.

Одной из наиболее трудоемких, хотя и хорошо формализуемых задач этого этапа является описание документооборота организации. При *обследовании* документооборота составляется схема маршрута движения документов, которая должна отразить:

- количество документов;
- место формирования показателей документа;
- взаимосвязь документов при их формировании;
- маршрут и длительность движения документа;
- место использования и хранения данного документа;
- внутренние и внешние информационные связи;
- объем документа в знаках.

По результатам *обследования* устанавливается перечень задач управления, решение которых целесообразно автоматизировать, и очередность их разработки.

На этапе *обследования* следует классифицировать планируемые функции системы по степени важности. Один из возможных форматов представления такой классификации – MuSCoW (Must have – необходимые функции; Should have – желательные функции; Could have – возможные функции; Won't have – отсутствующие функции).

Функции первой категории обеспечивают критичные для успешной работы системы возможности. Реализация функций второй и третьей категорий ограничивается временными и финансовыми рамками: разрабатывается то, что необходимо, а также максимально возможное в порядке приоритета число функций второй и третьей категорий. Последняя категория функций особенно важна, поскольку необходимо четко представлять границы проекта и набор функций, которые будут отсутствовать в системе.

Модели деятельности организации создаются в двух видах:

- **модель «как есть»** («as-is») – отражает существующие в организации бизнес-процессы;
- **модель «как должно быть»** («to-be») – отражает необходимые изменения бизнес-процессов с учетом внедрения ИС.

На этапе анализа необходимо привлекать к работе группы тестирования для решения следующих задач:

- получения сравнительных характеристик предполагаемых к использованию аппаратных платформ, операционных систем, СУБД, иного окружения;
- разработки плана работ по обеспечению надежности информационной системы и ее тестирования.

Привлечение тестировщиков на ранних этапах разработки является целесообразным для любых проектов. Если проектное решение оказалось неудачным и это обнаружено слишком поздно (на этапе разработки или, что еще хуже, на этапе внедрения в эксплуатацию), то исправление ошибки проектирования обходится очень дорого. Чем раньше группы тестирования выявляют ошибки в информационной системе, тем ниже стоимость сопровождения системы. Время на тестирование системы и на исправление обнаруженных ошибок следует предусматривать не только на этапе разработки, но и на этапе проектирования.

Для автоматизации тестирования следует использовать системы отслеживания ошибок (bug tracking). Это позволяет иметь единое хранилище ошибок, отслеживать их повторное появление, контролировать скорость и эффективность исправления ошибок, видеть наиболее нестабильные компоненты системы, а также поддерживать связь между группой разработчиков и группой тестирования (уведомления об изменениях по e-mail и т.п.). Чем больше проект, тем сильнее потребность в bug tracking.

Результаты *обследования* представляют объективную основу для формирования *технического задания* на информационную систему.

Техническое задание (ТЗ) – это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления.

При разработке *технического задания* необходимо решить следующие задачи:

- установить общую цель создания ИС, определить состав подсистем и функциональных задач;
- разработать и обосновать требования, предъявляемые к подсистемам;
- разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств (включая средства связи и передачи данных);
- установить общие требования к проектируемой системе;
- определить перечень задач создания системы и исполнителей;
- определить этапы создания системы и сроки их выполнения;
- провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения.

Типовые требования к составу и содержанию *технического задания* см. Таблица 2.

Таблица 2. Состав и назначение технического задания (ГОСТ 34.602 - 89)

Раздел	Содержание
Общие сведения	<ul style="list-style-type: none"> • полное наименование системы и ее условное обозначение • шифр темы или шифр (номер) договора; • наименование предприятий разработчика и заказчика системы, их реквизиты • перечень документов, на основании которых создается ИС • плановые сроки начала и окончания работ • сведения об источниках и порядке финансирования работ • порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств
Назначение и цели создания	<ul style="list-style-type: none"> • вид автоматизируемой деятельности • перечень объектов, на которых предполагается использование

(развития) системы	<p>системы</p> <ul style="list-style-type: none"> • наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении ИС
Характеристика объектов автоматизации	<ul style="list-style-type: none"> • краткие сведения об объекте автоматизации • сведения об условиях эксплуатации и характеристиках окружающей среды
Требования к системе	<p>Требования к системе в целом:</p> <ul style="list-style-type: none"> • требования к структуре и функционированию системы (перечень подсистем, уровни иерархии, степень централизации, способы информационного обмена, режимы функционирования, взаимодействие со смежными системами, перспективы развития системы) • требования к персоналу (численность пользователей, квалификация, режим работы, порядок подготовки) • показатели назначения (степень приспособляемости системы к изменениям процессов управления и значений параметров) • требования к надежности, безопасности, эргономике, транспортабельности, эксплуатации, техническому обслуживанию и ремонту, защите и сохранности информации, защите от внешних воздействий, к патентной чистоте, по стандартизации и унификации <p>Требования к функциям (по подсистемам):</p> <ul style="list-style-type: none"> • перечень подлежащих автоматизации задач • временной регламент реализации каждой функции • требования к качеству реализации каждой функции, к форме представления выходной информации, характеристики точности, достоверности выдачи результатов • перечень и критерии отказов <p>Требования к видам обеспечения:</p> <ul style="list-style-type: none"> • математическому (состав и область применения мат. моделей и методов, типовых и разрабатываемых алгоритмов) • информационному (состав, структура и организация данных, обмен данными между компонентами системы, информационная совместимость со смежными системами, используемые классификаторы, СУБД, контроль данных и ведение информационных массивов, процедуры придания юридической силы выходным документам) • лингвистическому (языки программирования, языки взаимодействия пользователей с системой, системы кодирования, языки ввода-вывода) • программному (независимость программных средств от платформы, качество программных средств и способы его контроля, использование фондов алгоритмов и программ) • техническому • метрологическому • организационному (структура и функции эксплуатирующих подразделений, защита от ошибочных действий персонала) • методическому (состав нормативно-технической документации)
Состав и	<ul style="list-style-type: none"> • перечень стадий и этапов работ

содержание работ по созданию системы	<ul style="list-style-type: none"> • сроки исполнения • состав организаций — исполнителей работ • вид и порядок экспертизы технической документации • программа обеспечения надежности • программа метрологического обеспечения
Порядок контроля и приемки системы	<ul style="list-style-type: none"> • виды, состав, объем и методы испытаний системы • общие требования к приемке работ по стадиям • статус приемной комиссии
Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие	<ul style="list-style-type: none"> • преобразование входной информации к машиночитаемому виду • изменения в объекте автоматизации • сроки и порядок комплектования и обучения персонала
Требования к документированию	<ul style="list-style-type: none"> • перечень подлежащих разработке документов • перечень документов на машинных носителях
Источники разработки	<ul style="list-style-type: none"> • документы и информационные материалы, на основании которых разрабатывается ТЗ и система

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям. Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание *эскизного проекта* задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе *технического задания* и *эскизного проекта* разрабатывается *технический проект* ИС. **Технический проект** системы — это техническая документация, содержащая общесистемные проектные решения, алгоритмы решения задач, а также оценку экономической эффективности автоматизированной системы управления и перечень мероприятий по подготовке объекта к внедрению.

На этом этапе осуществляется комплекс научно-исследовательских и экспериментальных работ для выбора основных проектных решений и расчет экономической эффективности системы.

Состав и содержание технического проекта см. Таблица 3.

Таблица 3. Содержание технического проекта

Раздел	Содержание
Пояснительная записка	<ul style="list-style-type: none"> • основания для разработки системы • перечень организаций разработчиков • краткая характеристика объекта с указанием основных технико-экономических показателей его функционирования и связей с другими объектами • краткие сведения об основных проектных решениях по функциональной и обеспечивающим частям системы
Функциональная и организационная структура системы	<ul style="list-style-type: none"> • обоснование выделяемых подсистем, их перечень и назначение • перечень задач, решаемых в каждой подсистеме, с краткой характеристикой их содержания • схема информационных связей между подсистемами и между задачами в рамках каждой подсистемы
Постановка задач и алгоритмы решения	<ul style="list-style-type: none"> • организационно-экономическая сущность задачи (наименование, цель решения, краткое содержание, метод, периодичность и время решения задачи, способы сбора и передачи данных, связь задачи с другими задачами, характер использования результатов решения, в которых они используются) • экономико-математическая модель задачи (структурная и развернутая форма представления) • входная оперативная информация (характеристика показателей, диапазон изменения, формы представления) • нормативно-справочная информация (НСИ) (содержание и формы представления) • информация, хранимая для связи с другими задачами • информация, накапливаемая для последующих решений данной задачи • информация по внесению изменений (система внесения изменений и перечень информации, подвергающейся изменениям) • алгоритм решения задачи (последовательность этапов расчета, схема, расчетные формулы) • контрольный пример (набор заполненных данными форм входных документов, условные документы с накапливаемой и хранимой информацией, формы выходных документов, заполненные по результатам решения экономико-технической задачи и в соответствии с разработанным алгоритмом расчета)
Организация информационной базы	<ul style="list-style-type: none"> • источники поступления информации и способы ее передачи • совокупность показателей, используемых в системе • состав документов, сроки и периодичность их поступления • основные проектные решения по организации фонда НСИ

	<ul style="list-style-type: none"> • состав НСИ, включая перечень реквизитов, их определение, диапазон изменения и перечень документов НСИ • перечень массивов НСИ, их объем, порядок и частота корректировки информации • структура фонда НСИ с описанием связи между его элементами; требования к технологии создания и ведения фонда • методы хранения, поиска, внесения изменений и контроля • определение объемов и потоков информации НСИ • контрольный пример по внесению изменений в НСИ • предложения по унификации документации
Альбом форм документов	Отсутствует
Система математического обеспечения	<ul style="list-style-type: none"> • обоснование структуры математического обеспечения • обоснование выбора системы программирования • перечень стандартных программ
Принцип построения комплекса технических средств	<ul style="list-style-type: none"> • описание и обоснование схемы технологического процесса обработки данных • обоснование и выбор структуры комплекса технических средств и его функциональных групп • обоснование требований к разработке нестандартного оборудования • комплекс мероприятий по обеспечению надежности функционирования технических средств
Расчет экономической эффективности системы	<ul style="list-style-type: none"> • сводная смета затрат, связанных с эксплуатацией систем • расчет годовой экономической эффективности, источниками которой являются оптимизация производственной структуры хозяйства (объединения), снижение себестоимости продукции за счет рационального использования производственных ресурсов и уменьшения потерь, улучшения принимаемых управленческих решений
Мероприятия по подготовке объекта к внедрению системы	<ul style="list-style-type: none"> • перечень организационных мероприятий по совершенствованию бизнес-процессов • перечень работ по внедрению системы, которые необходимо выполнить на стадии рабочего проектирования, с указанием сроков и ответственных лиц
Ведомость документов	Отсутствует

В завершение стадии технического проектирования производится разработка документации на поставку серийно выпускаемых изделий для комплектования ИС, а также определяются технические требования и составляются ТЗ на разработку изделий, не изготавливаемых серийно.

На стадии **«Рабочая документация»** осуществляется создание программного продукта и разработка всей сопровождающей документации. Документация должна содержать все необходимые и достаточные сведения для обеспечения выполнения работ по вводу ИС в действие и ее эксплуатации, а также для поддержания уровня эксплуатационных характеристик (качества) системы. Разработанная документация должна быть соответствующим образом оформлена, согласована и утверждена.

На стадии «**Ввод в действие**» для АИС устанавливают следующие основные виды испытаний: *предварительные испытания, опытная эксплуатация и приемочные испытания*. При необходимости допускается дополнительно проведение других видов испытаний системы и ее частей.

В зависимости от взаимосвязей частей ИС и объекта автоматизации испытания могут быть *автономные* или *комплексные*. Автономные испытания охватывают части системы. Их проводят по мере готовности частей системы к сдаче в *опытную эксплуатацию*. Комплексные испытания проводят для групп взаимосвязанных частей или для системы в целом.

Для планирования проведения всех видов испытаний разрабатывается документ «Программа и методика испытаний». Разработчик документа устанавливается в договоре или ТЗ. В качестве приложения к документу могут включаться тесты или контрольные примеры.

Отладка – наиболее трудоемкий процесс проектирования. Скрытые ошибки иногда проявляются после многолетней эксплуатации системы. Насчитывают 169 типов ошибок, сведенных в 19 классов:

1. Логические;
2. Ошибки манипулирования данными;
3. Ошибки ввода-вывода;
4. Ошибки в вычислениях;
5. Ошибки в пользовательских интерфейсах;
6. Ошибки в ОС и вспомогательных программах;
7. Ошибки компоновки;
8. Ошибки в межпрограммных интерфейсах;
9. Ошибки в интерфейсах «Программа – системное ПО»;
10. Ошибки при обращении к внешним устройствам;
11. Ошибки сопряжения с базой данных (БД);
12. Ошибки инициализации БД;
13. Ошибки изменений по запросу извне;
14. Ошибки, связанные с глобальными переменными;
15. Повторяющиеся ошибки;
16. Ошибки в документации;
17. Нарушение технических требований;
18. Неопознанные ошибки;
19. Ошибки оператора.

Для выявления ошибок создаются тесты и проводятся испытания. Методика отладки учитывает симптомы возможных ошибок:

- Неверная обработка (неправильный ответ, результат) – до 30%;
- Неверная передача управления – 16%;
- Несовместимость программ с используемыми данными – 15%;
- Несовместимость программ с пересылаемыми данными – 9%.

Предварительные испытания проводят для определения работоспособности системы и решения вопроса о возможности ее приемки в *опытную эксплуатацию*. *Предварительные испытания* следует выполнять после проведения разработчиком отладки и тестирования поставляемых программных и технических средств системы и представления им

соответствующих документов об их готовности к испытаниям, а также после ознакомления персонала ИС с эксплуатационной документацией.

Опытную эксплуатацию системы проводят с целью определения фактических значений количественных и качественных характеристик системы и готовности персонала к работе в условиях ее функционирования, а также определения фактической эффективности и корректировки, при необходимости, документации.

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

Типовое проектирование ИС

Типовое проектирование ИС предполагает создание системы из готовых типовых элементов. Основопологающим требованием для применения методов *типового проектирования* является возможность декомпозиции проектируемой ИС на множество составляющих компонентов (подсистем, комплексов задач, программных модулей и т.д.). Для реализации выделенных компонентов выбираются имеющиеся на рынке *типовые проектные решения*, которые настраиваются на особенности конкретного предприятия.

Типовое проектное решение (ТПР) – это тиражируемое (пригодное к многократному использованию) проектное решение.

Принятая классификация ТПР основана на уровне декомпозиции системы. Выделяются следующие классы ТПР:

- элементные ТПР – типовые решения по задаче или по отдельному виду обеспечения задачи (информационному, программному, техническому, математическому, организационному);
- подсистемные ТПР – в качестве элементов типизации выступают отдельные подсистемы, разработанные с учетом функциональной полноты и минимизации внешних информационных связей;
- объектные ТПР – типовые отраслевые проекты, которые включают полный набор функциональных и обеспечивающих подсистем ИС.

Каждое типовое решение предполагает наличие, кроме собственно функциональных элементов (программных или аппаратных), документации с детальным описанием ТПР и процедур настройки в соответствии с требованиями разрабатываемой системы.

Основные особенности различных классов ТПР см. Таблица 4.

Таблица 4. Достоинства и недостатки ТПР

Класс ТПР, реализация ТПР	Достоинства	Недостатки
Элементные ТПР. Библиотеки методо-ориентированных программ	обеспечивается применение модульного подхода к проектированию и документированию ИС	<ul style="list-style-type: none">• большие затраты времени на сопряжение разнородных элементов вследствие информационной, программной и технической

		несовместимости <ul style="list-style-type: none"> • большие затраты времени на доработку ТПР отдельных элементов
Подсистемные ТПР. Пакеты прикладных программ	<ul style="list-style-type: none"> • достигается высокая степень интеграции элементов ИС • позволяют осуществлять: модульное проектирование; параметрическую настройку программных компонентов на различные объекты управления • обеспечивают: сокращение затрат на проектирование и программирование взаимосвязанных компонентов; хорошее документирование отображаемых процессов обработки информации 	<ul style="list-style-type: none"> • адаптивность ТПР недостаточна с позиции непрерывного инжиниринга деловых процессов • возникают проблемы в комплексировании разных функциональных подсистем, особенно в случае использования решений нескольких производителей программного обеспечения
Объектные ТПР. Отраслевые проекты ИС	<ul style="list-style-type: none"> • комплексирование всех компонентов ИС за счет методологического единства и информационной, программной и технической совместимости • открытость архитектуры — позволяет устанавливать ТПР на разных программно-технических платформах • масштабируемость — допускает конфигурацию ИС для переменного числа рабочих мест • конфигурируемость — позволяет выбирать необходимое подмножество компонентов 	<ul style="list-style-type: none"> • проблемы привязки типового проекта к конкретному объекту управления, что вызывает в некоторых случаях даже необходимость изменения организационно-экономической структуры объекта автоматизации

Для реализации *типового проектирования* используются два подхода: *параметрически-ориентированное* и *модельно-ориентированное* проектирование.

Параметрически-ориентированное проектирование включает следующие этапы: определение критериев оценки пригодности пакетов прикладных программ (ППП) для решения поставленных задач, анализ и оценка доступных ППП по сформулированным критериям, выбор и закупка наиболее подходящего пакета, настройка параметров (доработка) закупленного ППП.

Критерии оценки ППП делятся на следующие группы:

- назначение и возможности пакета;

- отличительные признаки и свойства пакета;
- требования к техническим и программным средствам;
- документация пакета;
- факторы финансового порядка;
- особенности установки пакета;
- особенности эксплуатации пакета;
- помощь поставщика по внедрению и поддержанию пакета;
- оценка качества пакета и опыт его использования;
- перспективы развития пакета.

Внутри каждой группы критериев выделяется некоторое подмножество частных показателей, детализирующих каждый из десяти выделенных аспектов анализа выбираемых ППП. Числовые значения показателей для конкретных ППП устанавливаются экспертами по выбранной шкале оценок (например, 10-балльной). На их основе формируются групповые оценки и комплексная оценка пакета (путем вычисления средневзвешенных значений). Нормированные взвешивающие коэффициенты также получают экспертным путем.

Модельно-ориентированное проектирование заключается в адаптации состава и характеристик типовой ИС в соответствии с моделью объекта автоматизации. Технология проектирования в этом случае должна обеспечивать единые средства для работы как с моделью типовой ИС, так и с моделью конкретного предприятия.

Специальная база метаданных – *репозиторий* – содержит модель объекта автоматизации, на основе которой осуществляется конфигурирование ПО. Таким образом, *модельно-ориентированное проектирование ИС* предполагает, прежде всего, построение модели объекта автоматизации с использованием специального программного инструментария (например, SAP Business Engineering Workbench, BAAN Enterprise Modeler и др.). Возможно также создание системы на базе *типовой модели* ИС из репозитория, который поставляется вместе с программным продуктом и расширяется по мере накопления опыта проектирования ИС для различных отраслей и типов производства.

Репозиторий содержит *базовую (ссылочную) модель ИС, типовые (референтные) модели* определенных классов ИС, модели конкретных ИС предприятий.

Базовая модель ИС в репозитории содержит описание бизнес-функций, бизнес-процессов, бизнес-объектов, бизнес-правил, организационной структуры, которые поддерживаются программными модулями типовой ИС.

Типовые модели описывают конфигурации информационной системы для определенных отраслей или типов производства.

Модель конкретного предприятия строится либо путем выбора фрагментов основной или *типовой модели* в соответствии со специфическими особенностями предприятия (BAAN Enterprise Modeler), либо путем автоматизированной адаптации этих моделей в результате экспертного опроса (SAP Business Engineering Workbench).

Построенная модель предприятия в виде метаописания хранится в репозитории и при необходимости может быть откорректирована. На основе этой модели автоматически осуществляется конфигурирование и настройка ИС.

Бизнес-правила определяют условия корректности совместного применения различных компонентов ИС и используются для поддержания целостности создаваемой системы.

Модель бизнес-функций представляет собой иерархическую декомпозицию функциональной деятельности предприятия.

Модель бизнес-процессов отражает выполнение работ для функций самого нижнего уровня модели бизнес-функций. Для отображения процессов используется *модель управления событиями* (EPC – Event-driven Process Chain). Именно модель бизнес-процессов позволяет выполнить настройку программных модулей – приложений ИС в соответствии с характерными особенностями конкретного предприятия.

Модели бизнес-объектов используются для интеграции приложений, поддерживающих исполнение различных бизнес-процессов.

Модель организационной структуры предприятия представляет собой традиционную иерархическую структуру подчинения подразделений и персонала.

Внедрение типовой ИС начинается с анализа требований к конкретной ИС, которые выявляются на основе результатов предпроектного *обследования* объекта автоматизации. Для оценки соответствия этим требованиям программных продуктов может использоваться описанная выше методика оценки ППП. После выбора программного продукта на базе имеющихся в нем референтных моделей строится предварительная модель ИС, в которой отражаются все особенности реализации ИС для конкретного предприятия. Предварительная модель является основой для выбора *типовой модели* системы и определения перечня компонентов, которые будут реализованы с использованием других программных средств или потребуют разработки с помощью имеющихся в составе типовой ИС инструментальных средств (например, ABAP в SAP, Tools в BAAN).

Реализация типового проекта предусматривает выполнение следующих операций:

- установку глобальных параметров системы;
- задание структуры объекта автоматизации;
- определение структуры основных данных;
- задание перечня реализуемых функций и процессов;
- описание интерфейсов;
- описание отчетов;
- настройку авторизации доступа;
- настройку системы архивирования.

Полная бизнес-модель компании

Практика выработала ряд подходов к проведению организационного анализа, но наибольшее распространение получил инжиниринговый подход. Организационный анализ компании при таком подходе проводится по определенной схеме с помощью *полной бизнес-модели компании*. Компания рассматривается как целевая, открытая, социально-экономическая система, принадлежащая иерархической совокупности открытых внешних надсистем (рынок, государственные учреждения и пр.) и внутренних подсистем (отделы, цеха, бригады и пр.). Возможности компании определяются характеристиками ее структурных подразделений и организацией их взаимодействия. На [рис. 4.1](#) представлена обобщенная схема организационного бизнес-моделирования. Построение *бизнес-модели компании* начинается с описания модели взаимодействия с внешней средой по закону единства и борьбы противоположностей, то есть с определения *миссии компании*.

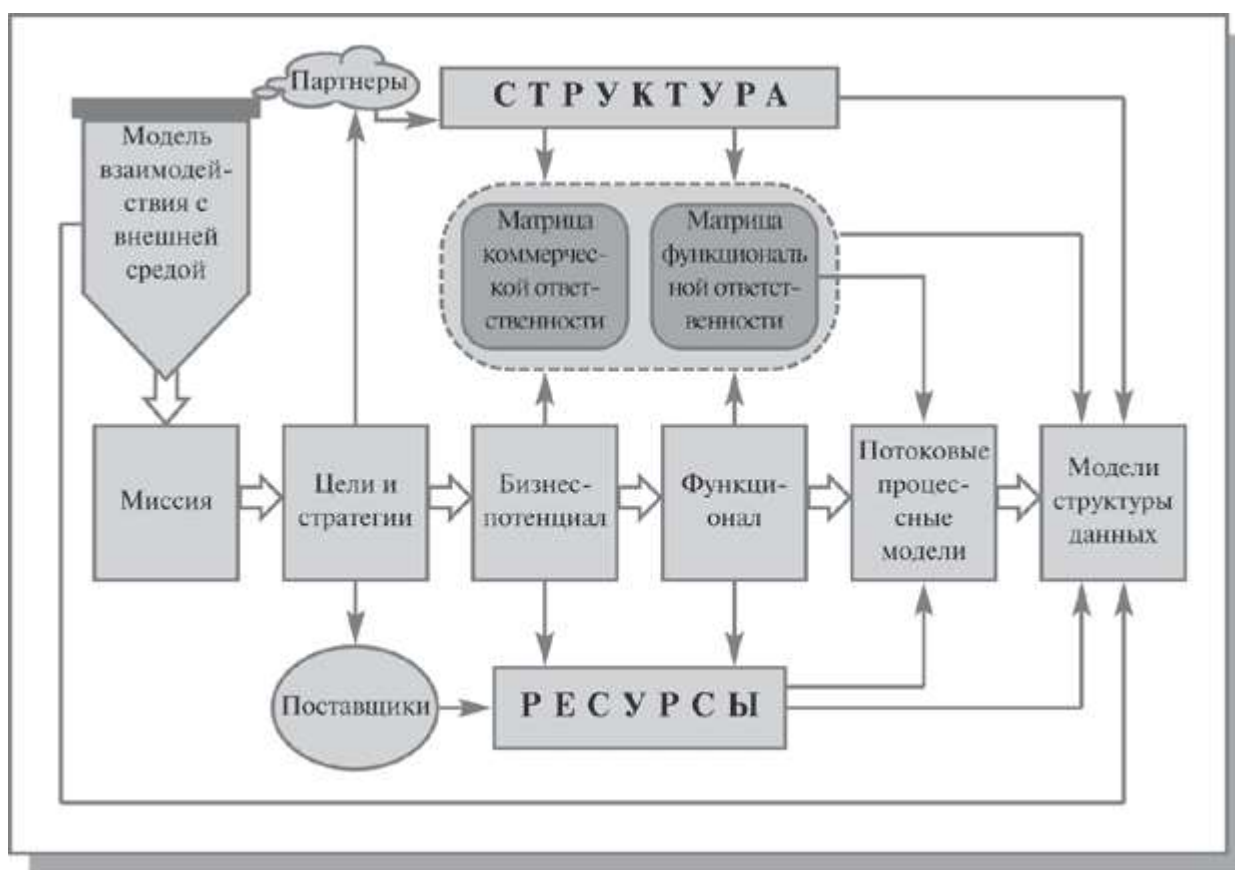


Рис. 4.1. Обобщенная схема организационного бизнес- моделирования

Миссия согласно [ISO-15704] -это

1. деятельность, осуществляемая предприятием для того, чтобы выполнить функцию, для которой оно было учреждено, - предоставления заказчикам продукта или услуги.
2. Механизм, с помощью которого предприятие реализует свои цели и задачи.

Миссия компании по удовлетворению социально-значимых потребностей рынка **определяется как компромисс интересов рынка и компании**. При этом *миссия* как атрибут открытой системы разрабатывается, с одной стороны, исходя из рыночной конъюнктуры и позиционирования компании относительно других участников внешней

среды, а с другой - исходя из объективных возможностей компании и ее субъективных ценностей, ожиданий и принципов. **Миссия является своеобразной мерой** устремлений компании и, в частности, определяет рыночные претензии компании (предмет конкурентной борьбы). Определение **миссии** позволяет сформировать **дерево целей компании** - иерархические списки уточнения и детализации миссии.

Дерево целей формирует **дерево стратегий** - иерархические списки уточнения и детализации способов достижения целей. При этом на корпоративном уровне разрабатываются стратегии роста, интеграции и инвестиции бизнесов. Блок бизнес-стратегий определяет продуктовые и конкурентные стратегии, а также стратегии сегментации и продвижения. Ресурсные стратегии определяют стратегии привлечения материальных, финансовых, человеческих и информационных ресурсов. Функциональные стратегии определяют стратегии в организации компонентов управления и этапов жизненного цикла продукции. Одновременно выясняется потребность и предмет партнерских отношений (субподряд, сервисные услуги, продвижение и пр.). Это позволяет обеспечить заказчикам необходимый продукт требуемого качества, в нужном количестве, в нужном месте, в нужное время и по приемлемой цене. При этом компания может занять в партнерской цепочке создаваемых ценностей оптимальное место, где ее возможности и потенциал будут использоваться наилучшим образом. Это дает возможность сформировать **бизнес-потенциал компании** - набор видов коммерческой деятельности, направленный на удовлетворение потребностей конкретных сегментов рынка. Далее, исходя из специфики каналов сбыта, формируется первоначальное представление об организационной структуре (определяются центры коммерческой ответственности). Возникает понимание основных ресурсов, необходимых для воспроизводства товарной номенклатуры.

Бизнес-потенциал, в свою очередь, определяет **функционал компании** - перечень бизнес-функций, функций менеджмента и функций обеспечения, требуемых для поддержания на регулярной основе указанных видов коммерческой деятельности. Кроме того, уточняются необходимые для этого ресурсы (материальные, человеческие, информационные) и структура компании.

Построение **бизнес-потенциала** и **функционала компании** позволяет с помощью **матрицы проекций** определить **зоны ответственности менеджмента**.

Матрица проекций - модель, представленная в виде матрицы, задающей систему отношений между классификаторами в любой их комбинации.

Матрица коммерческой ответственности закрепляет ответственность структурных подразделений за получение дохода в компании от реализации коммерческой деятельности. Ее дальнейшая детализация (путем выделения центров финансовой ответственности) обеспечивает построение финансовой модели компании, что, в свою очередь, позволяет внедрить систему бюджетного управления. **Матрица функциональной ответственности** закрепляет ответственность структурных звеньев (и отдельных специалистов) за выполнение бизнес-функций при реализации процессов коммерческой деятельности (закупка, производство, сбыт и пр.), а также функций менеджмента, связанных с управлением этими процессами (планирование, учет, контроль в области маркетинга, финансов, управления персоналом и пр.). Дальнейшая детализация матрицы (до уровня ответственности отдельных сотрудников) позволит получить функциональные обязанности персонала, что в совокупности с описанием прав, обязанностей, полномочий обеспечит разработку пакета должностных инструкций.

Описание **бизнес-потенциала**, **функционала** и соответствующих матриц ответственности представляет собой **статическое описание компании**. При этом процессы, протекающие в компании пока в свернутом виде (как функции), идентифицируются, классифицируются и, что особенно важно, закрепляются за исполнителями (будущими хозяевами этих процессов).

На этом этапе бизнес-моделирования формируется общепризнанный набор основополагающих внутрифирменных регламентов:

- базовое Положение об организационно-функциональной структуре компании;
- пакет Положений об отдельных видах деятельности (финансовой, маркетинговой и т.д.);
- пакет Положений о структурных подразделениях (цехах, отделах, секторах, группах и т.п.);
- должностные инструкции.

Это вносит прозрачность в деятельность компании за счет четкого разграничения и документального закрепления зон ответственности менеджеров.

Дальнейшее развитие (детализация) бизнес-модели происходит на этапе динамического описания компании на уровне *процессных потоковых моделей*. **Процессные потоковые модели** - это модели, описывающие процесс последовательного во времени преобразования материальных и информационных потоков компании в ходе реализации какой-либо бизнес-функции или функции менеджмента. Сначала (на верхнем уровне) описывается логика взаимодействия участников процесса, а затем (на нижнем уровне) - технология работы отдельных специалистов на своих рабочих местах.

Завершается организационное бизнес-моделирование разработкой **модели структур данных, которая определяет перечень и форматы документов, сопровождающих процессы в компании, а также задает форматы описания объектов внешней среды, компонентов и регламентов самой компании**. При этом создается система справочников, на основании которых получают пакеты необходимых документов и отчетов.

Такой подход позволяет описать деятельность компании с помощью универсального множества управленческих регистров (цели, стратегии, продукты, функции, организационные звенья и др.).

Управленческие регистры по своей структуре представляют собой иерархические классификаторы. Объединяя классификаторы в функциональные группы и закрепляя между собой элементы различных классификаторов с помощью матричных проекций, можно получить *полную бизнес-модель компании*.

При этом происходит процессно-целевое описание компании, позволяющее получить взаимосвязанные ответы на следующие вопросы: зачем-что-где-кто-как-когда-кому-сколько ([рис. 4.2](#)).

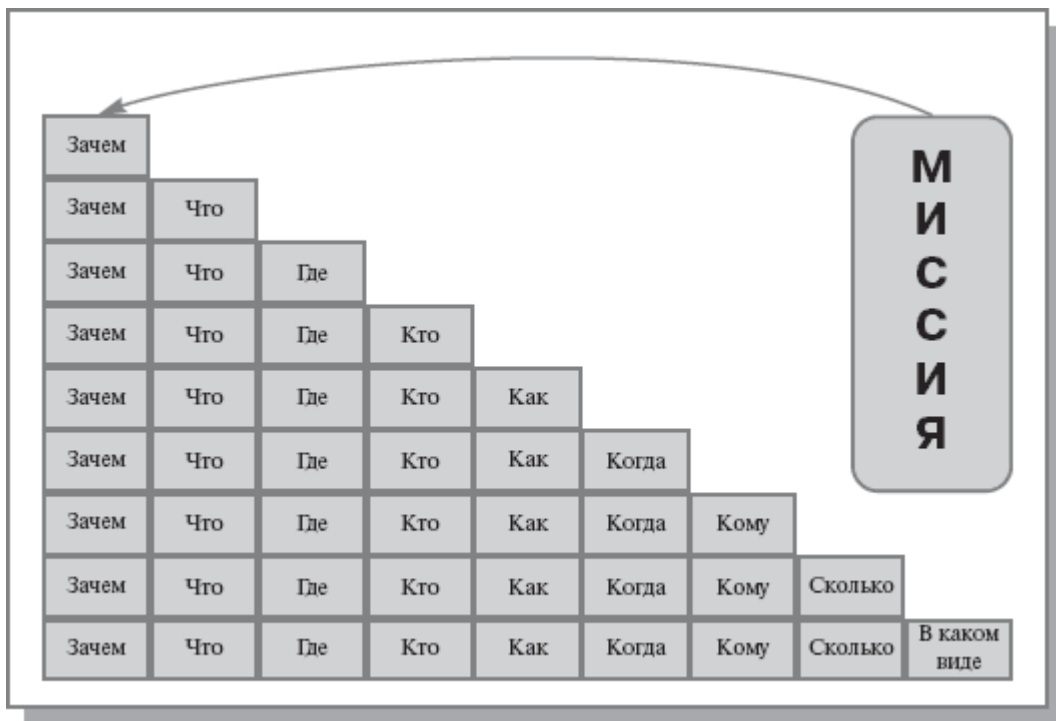


Рис. 4.2. Основные этапы процессно-целевого описания компании

Следовательно **полная бизнес-модель компании** - это совокупность функционально ориентированных информационных моделей, обеспечивающая взаимосвязанные ответы на следующие вопросы: "зачем" - "что" - "где" - "кто" - "сколько" - "как" - "когда" - "кому" ([рис. 4.3](#)).

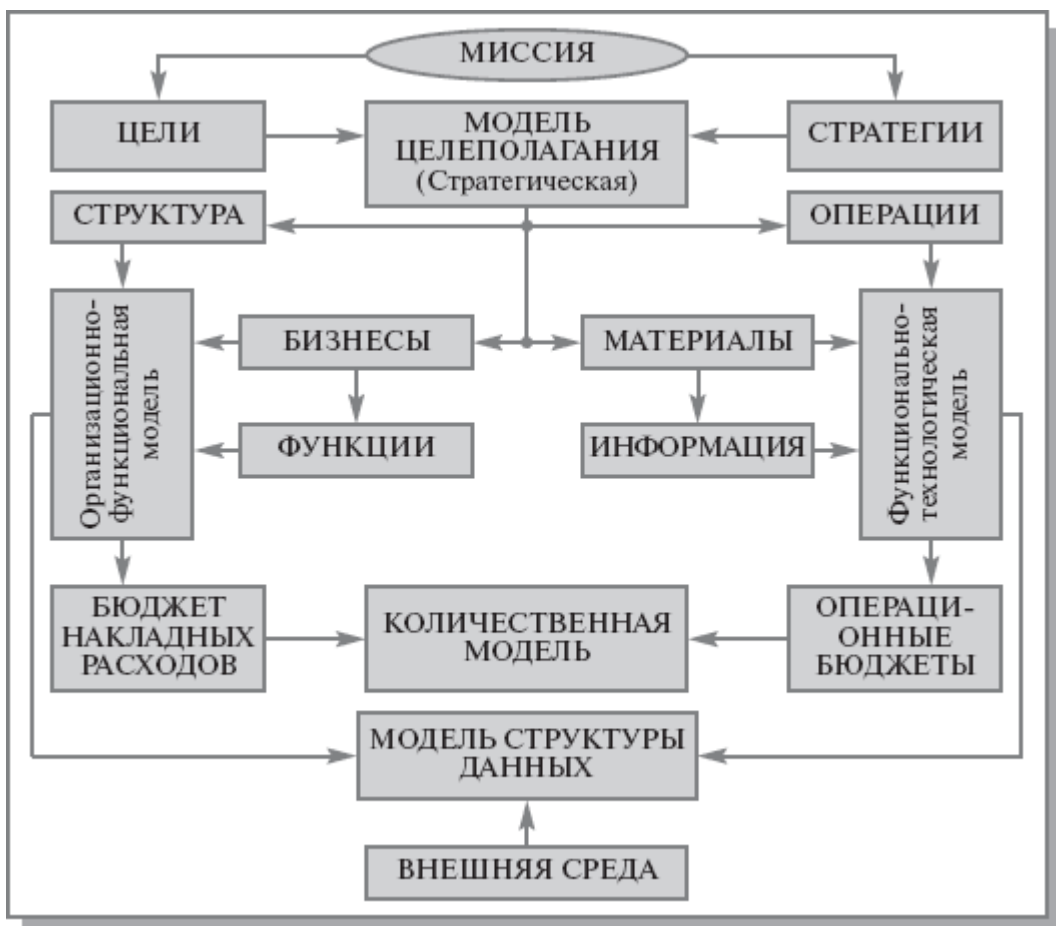


Рис. 4.3. Полная бизнес-модель компании

Таким образом, организационный анализ предполагает построение комплекса взаимосвязанных информационных моделей компании, который включает:

- **Стратегическую модель целеполагания** (отвечает на вопросы: зачем компания занимается именно этим бизнесом, почему предполагает быть конкурентоспособной, какие цели и стратегии для этого необходимо реализовать);
- **Организационно-функциональную модель** (отвечает на вопрос кто-что делает в компании и кто за что отвечает);
- **Функционально-технологическую модель** (отвечает на вопрос что-как реализуется в компании);
- **Процессно-ролевую модель** (отвечает на вопрос кто-что-как-кому);
- **Количественную модель** (отвечает на вопрос сколько необходимо ресурсов);
- **Модель структуры данных** (отвечает на вопрос в каком виде описываются регламенты компании и объекты внешнего окружения).

Представленная совокупность моделей обеспечивает необходимую полноту и точность описания компании и позволяет вырабатывать понятные требования к проектируемой информационной системе.

Шаблоны организационного бизнес-моделирования

Технология организационного бизнес-моделирования предполагает использование типовых шаблонных техник описания компании.

Шаблон разработки миссии

Как было сказано выше, любая компания с ее микро- и макроокружением представляет собой иерархию вложенных друг в друга открытых, субъектно-ориентированных систем. Компания, с одной стороны, является частью рынка, а с другой отстаивает в конкурентной борьбе собственные интересы. **Миссия** представляет собой результат позиционирования компании среди других участников рынка. Поэтому *миссию компании* нельзя описывать путем анализа ее внутреннего устройства. Для построения модели взаимодействия компании с внешней средой (определение *миссии компании* на рынке) необходимо:

- идентифицировать рынок (надсистему), частью которого является компания;
- определить свойства (потребности) рынка;
- определить предназначение (*миссию*) компании, исходя из ее роли на рынке.

Кроме этого, *миссия*, как было сказано выше, это компромисс между потребностями рынка, с одной стороны, и возможностями и желанием компании удовлетворить эти интересы, с другой. Поиск компромисса может быть выполнен по шаблону, представленному на [рис. 4.4](#).

		надо				
		рыночная конъюнк- тура	внешняя среда			
			Политика	Экономика	Социал. сфера	Технология
объект	Уникальность технологий					
	Исключительность ресурсов					
	Знания и умения					
хочу						
	Ценности и ожидания					

МИССИЯ

Рис. 4.4. Шаблон разработки миссии (матрица проекций)

При разработке модели *миссии компании* рекомендуется:

1. Описать базис конкурентоспособности компании - совокупность характеристик компании как социально-экономической системы. Например:
 - для объекта - уникальность освоенных технологий и исключительность имеющихся в компании ресурсов (финансовых, материальных, информационных и др.)
 - для субъекта - знания и умения персонала и опыт менеджеров.

Это определяет уникальность ресурсов и навыков компании и формирует позицию "могу".

2. Выяснить конъюнктуру рынка, т.е. определить наличие платежеспособного спроса на предлагаемые товары или услуги и степень удовлетворения рынка конкурентами. Это позволяет понять потребности рынка и сформировать позицию "надо".
3. Выявить наличие способствующих и противодействующих факторов для выбранного вида деятельности со стороны государственных институтов в области политики и экономики.
4. Оценить перспективу развития технологии в выбранной сфере деятельности.
5. Оценить возможную поддержку или противодействие общественных организаций.
6. Сопоставить результаты вышеперечисленных действий с учетом правовых, моральных, этических и др. ограничений со стороны персонала и сформировать позицию "хочу".
7. Оценить уровень возможных затрат и доходов.
8. Оценить возможность достижения приемлемого для всех сторон компромисса и сформулировать *Миссию компании* в соответствии с шаблоном, приведенным на [рис. 4.5](#).

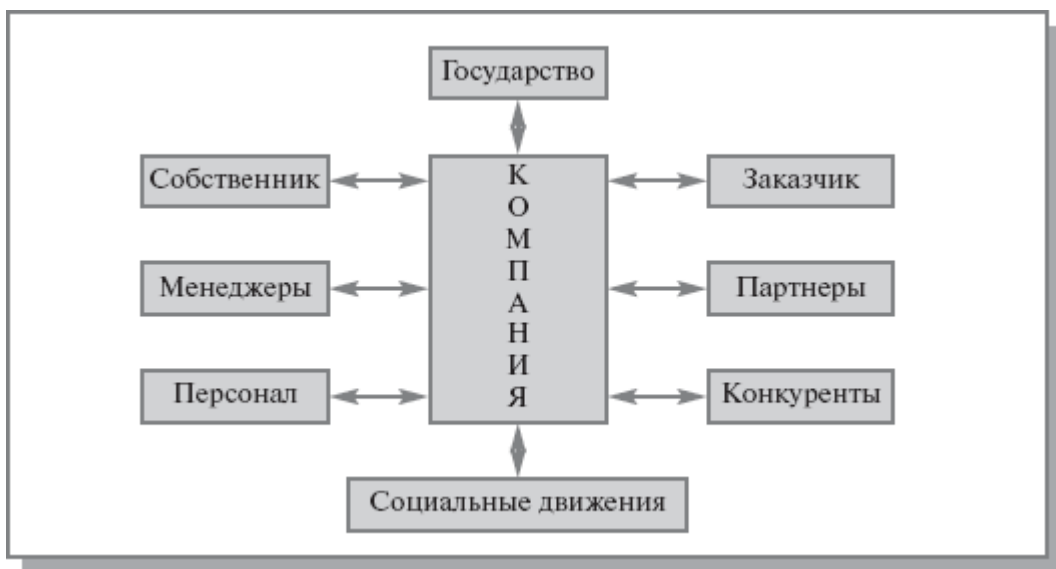


Рис. 4.5. Шаблон разработки миссии

Миссия в широком понимании представляет собой основную деловую концепцию компании, изложенную в виде восьми положений, определяющих взаимоотношения компании с другими субъектами:

- что получит Заказчик в части удовлетворения своих потребностей;
- кто, для чего и как может выступать в качестве партнера компании;
- на какой основе предполагается строить отношения с конкурентами (какова, в частности, готовность пойти на временные компромиссы);
- что получит собственник и акционеры от бизнеса;
- что получают от бизнеса компании менеджеры;
- что получит от компании персонал;
- в чем может заключаться сотрудничество с общественными организациями;
- как будут строиться отношения компании с государством (в частности, возможное участие в поддержке государственных программ).

Шаблон формирования бизнесов

В соответствии с разработанной *Миссией компании* определяются социально значимые потребности, на удовлетворение которых направлен бизнес компании.

Разработка *бизнес-потенциала компании* может быть выполнена по Шаблону формирования бизнесов, представленному на [рис. 4.6](#).

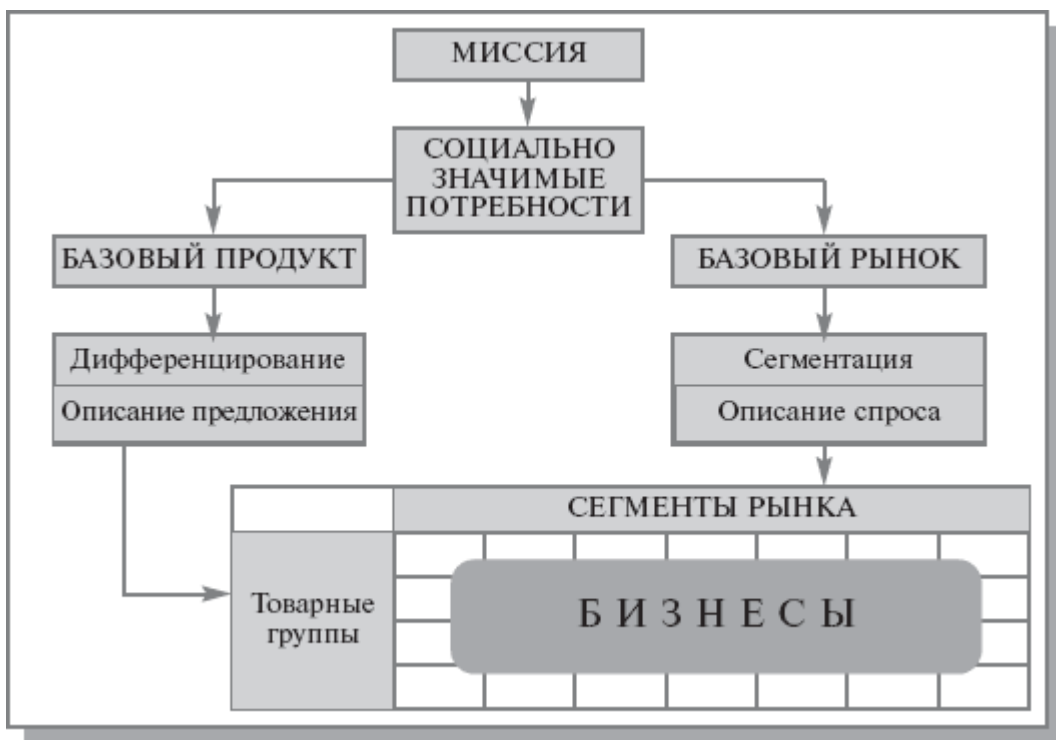


Рис. 4.6. Шаблон формирования бизнесов

В результате формируются базовый рынок и базовый продукт, детализация которых определяет предложения компании глазами покупателей (товарные группы) и однородные по отношению к продуктам компании группы покупателей (сегменты рынка). С помощью матричной проекции (рис. 4.7) устанавливается соответствие между сформированными товарными группами и сегментами рынка и определяется список бизнесов компании (на пересечении строк и столбцов находятся бизнесы компании).

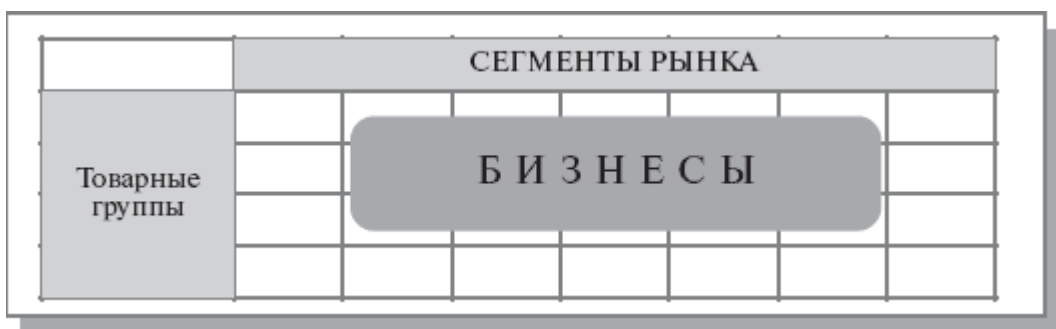


Рис. 4.7. Шаблон формирования бизнесов (матрица проекций)

Шаблон формирования функционала компании (основных бизнес-функций)

На основании списка бизнесов, с помощью матричной проекции (рис. 4.8) формируется классификатор бизнес-функций компании.

		БИЗНЕСЫ		
		№1	№2	№3
ЭТАПЫ ПРОИЗВОДСТВЕННОГО ЦИКЛА	Проектирование	БИЗНЕС-ФУНКЦИИ (ОСНОВНЫЕ)		
	Закупки			
	Производство			
	Распределение			
	Сбыт			
	Сопровождение			

Рис. 4.8. Шаблон формирования основных бизнес-функций

Для формирования основных функций менеджмента компании сначала разрабатываются и утверждаются два базовых классификатора - "Компоненты менеджмента" (перечень используемых на предприятии инструментов/контуров управления) и "Этапы управленческого цикла" (технологическая цепочка операций, последовательно реализуемых менеджерами при организации работ в любом контуре управления). Далее аналогично, с помощью *матрицы проекций*, формируется список основных функций менеджмента. На [рис. 4.9](#) приведены примеры классификаторов, на основании которых построена матрица - генератор основных функций менеджмента.

Этапы управленческого цикла	Компоненты менеджмента	Структуры	Логистика	Финансы	Экономика	Учет	Маркетинг	Персонал
Сбор информации								
Выработка решений								
Реализация								
Учет								
Контроль								
Анализ								
Регулирование								

Рис. 4.9. Шаблон формирования основных функций менеджмента

Представленные матричные проекции ([рис. 4.8](#), [рис. 4.9](#)) позволяют формировать функции любой степени детализации путем более подробного описания как строк, так и столбцов матрицы.

Шаблон формирования зон ответственности за функционал компании

Формирование зон ответственности за функционал компании выполняется с помощью матрицы организационных проекций ([рис. 4.10](#)).

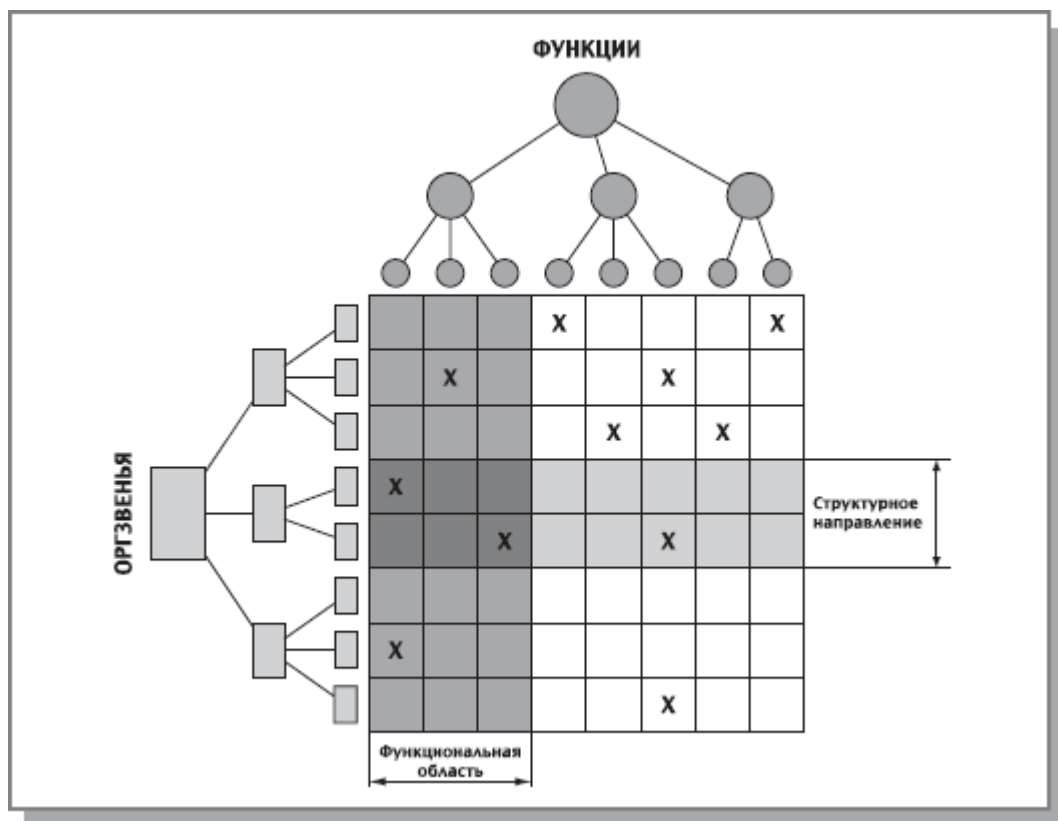


Рис. 4.10. Шаблон распределения функций по организационным звеньям

Матрица организационных проекций представляет собой **таблицу, в строках которой расположен список исполнительных звеньев, в столбцах - список функций, выполняемых в компании.** Для каждой функции определяется исполнительное звено, отвечающее за эту функцию.

Заполнение такой таблицы позволяет по каждой функции найти исполняющие ее подразделения или сотрудника. Анализ заполненной таблицы позволяет увидеть "пробелы" как в исполнении функций, так и в загруженности сотрудников, а также рационально перераспределить все задачи между исполнителями и закрепить как систему в документе "Положение об организационной структуре".

Положение об организационной структуре - это **внутрифирменный документ, фиксирующий: продукты и услуги компании, функции, выполняемые в компании, исполнительные звенья, реализующие функции, распределение функций по звеньям.**

Таблица проекций функций на исполнительные звенья может иметь весьма большую размерность. В средних компаниях это, например, 500 единиц - 20 звеньев на 25 функций. В больших компаниях это может быть 5 000 единиц - 50 звеньев на 100 функций.

Аналогично строится *матрица коммерческой ответственности*.

Шаблон потокового процессного описания

Шаблон потокового процессного описания приведен на [рис. 4.11](#). Такое описание дает представление о процессе последовательного преобразования ресурсов в продукты усилиями различных исполнителей на основании соответствующих регламентов.

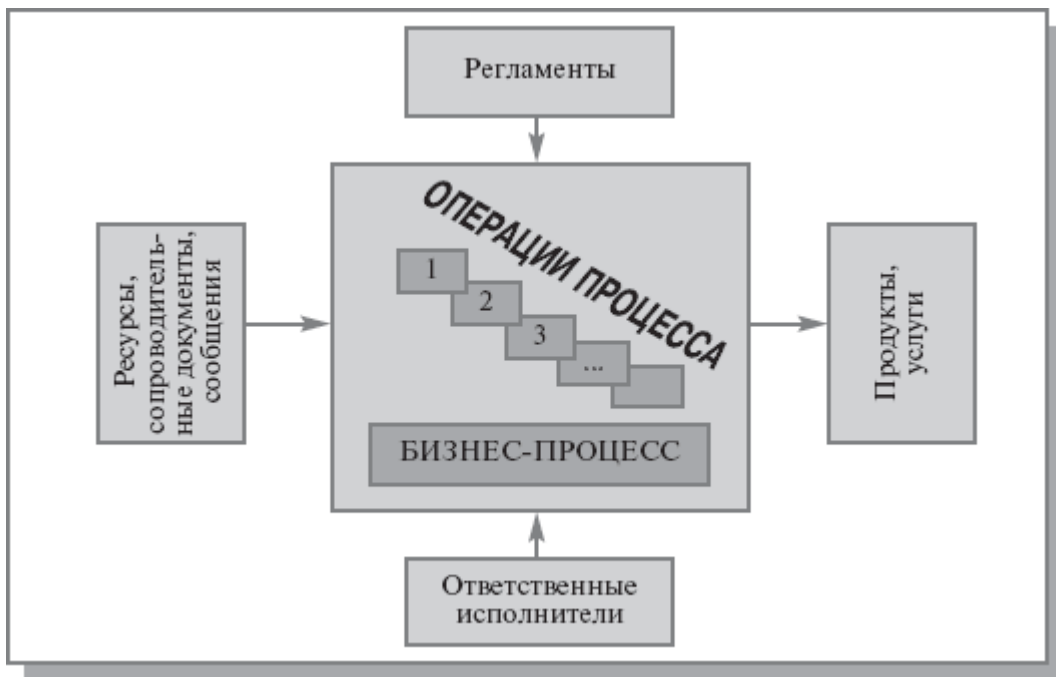


Рис. 4.11. Потоковая процессная модель

Методики построения процессных моделей будут приведены ниже.

Построения организационно-функциональной модели компании

Организационно-функциональная модель компании строится на основе функциональной схемы деятельности компании [рис. 4.12](#).

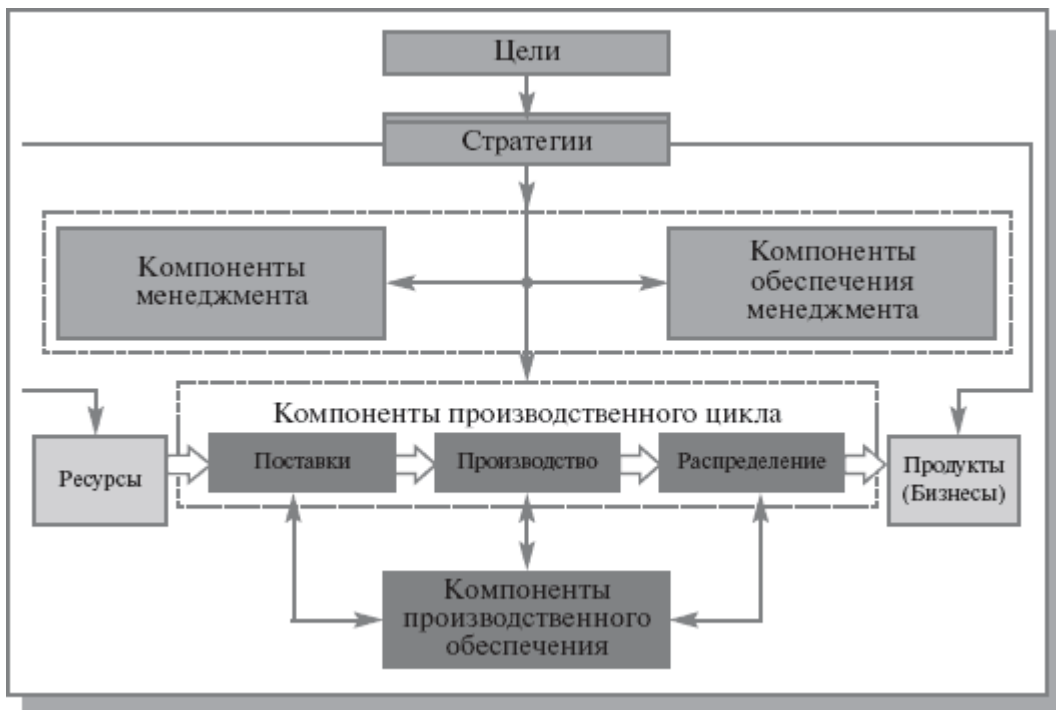


Рис. 4.12. Функциональная схема компании

На основании *миссии* формируются цели и стратегии компании. С их помощью определяется необходимый набор продуктов и, как следствие - требуемые ресурсы. Воспроизводство продукции происходит за счет переработки ресурсов в основном производственном цикле. Его компоненты формируют необходимые бизнес-функции для поставки ресурсов, производства продуктов и их распределения в места реализации. Для управления указанным процессом воспроизводства формируется совокупность компонентов менеджмента, которая порождает набор функций управления. Для поддержания процессов воспроизводства и управления формируются наборы соответствующих функций обеспечения (охраны, технического оснащения, профилактики и ремонта и пр.). Такой подход позволяет описать предприятие с помощью универсального множества управленческих регистров (цели, стратегии, продукты, функции, организационные звенья и пр.). Управленческие регистры представляют собой иерархические классификаторы. Объединяя классификаторы в функциональные группы и закрепляя между собой элементы различных классификаторов с помощью матричных проекций, можно получить модель организационной структуры компании.

Для построения организационно-функциональной модели используется всего два типа элементарных моделей.

Древовидные модели (классификаторы) - точные иерархические списки выделенных объектов управления (организационных звеньев, функций, ресурсов, в том числе исполнительных механизмов для бизнес-процессов, документов и их структуры, и т.п.). Каждый элемент классификатора может быть дополнительно охарактеризован рядом атрибутов: тип, шкала, комментарий и т.п. Фактически, классификаторы представляют собой набор управленческих регистров, содержащих, в основном, неколичественную информацию, совокупность которых задает систему координат для описания деятельности компании. Количество таких списков-классификаторов определяется целью построения модели.

Матричные модели - это проекции, задающие систему отношений между классификаторами в любой их комбинации. Связи могут иметь дополнительные атрибуты (направление, название, индекс, шкала и вес).

В начальной модели применяется всего несколько классификаторов предметной области:

- основные группы продуктов и услуг компании;
- ресурсы, потребляемые компанией в ходе своей деятельности;
- функции (процессы), поддерживаемые в компании;
- организационные звенья компании.

В классификаторе функций обычно выделяют три базовых раздела:

- основные функции - непосредственно связанные с процессом преобразования внешних ресурсов в продукцию и услуги предприятия;
- функции менеджмента - или функции управления предприятием;
- функции обеспечения - поддерживающие производственную, коммерческую и управленческую деятельность.

Главной функцией компании является предоставление продуктов и услуг, поэтому сначала производится формальное описание, согласование и утверждение руководством предприятия перечня его бизнесов (направлений коммерческой деятельности), продукции и услуг. Из этого классификатора внешним контрагентам должно быть понятно, чем предприятие интересно рынку, а для внутренних целей - для чего нужен тот или иной *функционал компании*.

В результате этих операций производится идентификация *функционала* и создается единая терминология описания функций предприятия, которая должна быть согласована всеми ведущими менеджерами. При составлении классификатора оргзвеньев важно, чтобы уровень детализации функций соответствовал уровню детализации звеньев. После формирования всех базовых классификаторов с помощью матричных проекций производится их закрепление за оргзвеньями предприятия:

Процесс формирования *матрицы проекций* функций на оргзвенья на практике напоминает игру в крестики-нолики ([рис. 4.10](#)).

По строчкам таблицы указываются подразделения, по столбцам - функции, составляющие содержание процесса управления или бизнес-процесса в данной компании. На пересечениях функций и подразделений, которые ответственны за выполнение функции, ставится крестик. Для проекций большой размерности используется механизм расстановки связей между двумя классификаторами, представленных списками.

Стандартная практика построения моделей организационно-функциональной структуры компаний поддерживает два уровня детализации:

1. агрегированную модель;
2. детализированную модель.

Агрегированная модель - модель организационной структуры, учетные регистры которой имеют ограничение по степени детализации до 2-3 уровней.

Целью построения данной модели является предоставление информации об организационной структуре высшим руководителям компании для проведения стратегического анализа, анализа соответствия данной структуры стратегии и внешнему окружению компании. Модель может также предоставляться внешним пользователям (например, потенциальным инвесторам как иллюстрация к бизнес-плану, крупным клиентам и др.).

Детализированная модель - модель организационной структуры, детализация учетных регистров которой производится на более глубоких уровнях, чем в агрегированной модели. Степень детализации в модели обусловлена конкретными потребностями компании (создание определенных организационных регламентов).

Целью построения данной модели является предоставление информации о распределении функциональных обязанностей между подразделениями компании, а также об организации бизнес-процессов в компании. Построение детализированной модели позволяет создавать различные внутрифирменные регламенты: Положения об организационной структуре [рис. 4.13](#).

Ниже приведен пример описания фрагментов организационно-функциональной модели производственного предприятия [рис. 4.14](#) и торгового предприятия [рис. 4.15](#). Приведенные *матрицы проекций* являются основой для выделения бизнес-процессов предприятия и их владельцев на последующих этапах создания ИС.



Рис. 4.13. Схема создания Положения об организационно- функциональной структуре компании

Рис. 4.14. Распределение функций по подразделениям производственного предприятия

Функциональная область	Клиентский сервис	Корпоративное управление	Финансы	Маркетинг	Заказы	Снабжение/закупки	Сбыт/Продажи
	CS	EM	FM	MK	OF	PR	SL
Генеральный директор	X	X	X	X	X		X
Зам. Ген. директора по сбыту (продажи)	X	X	X	X	X		X
Зам. Ген. директора по коммерческим вопросам (закупки)		X	X	X		X	
Экономист			X		X		
Помощник по правовым вопросам	X	X			X		X
Начальник отдела сбыта	X	X	X	X	X		X
Группа менеджеров	X			X	X		
Отдел оформления заказов			X		X		
НТЦ							
Секретариат						X	X
Бухгалтерия			X		X	X	

Функции выполняемые отделом, отмечены “X”.

Рис. 4.15. Распределение функций по подразделениям торгового предприятия

Функции подразделений производственного предприятия рассматриваются в рамках следующих функциональных областей:

- корпоративное управление;
- финансы;
- персонал;
- материальные ресурсы;
- заказы;
- производство;
- разработка продуктов;
- планирование;
- снабжение/закупки;
- качество;
- сбыт/продажи.

Распределение функций по структурным подразделениям в разрезе отдельных функциональных областей деятельности по управлению производственным предприятием представлено на [рис. 4.14](#).

Функции подразделений торгового предприятия рассматриваются в рамках иных функциональных областей (см. [рис. 4.15](#)).

Инструментальные средства организационного моделирования

Применение современных технологий для организационного моделирования позволяет значительно ускорить организационное проектирование. В начале 1990-х годов на Западе появились первые программы для решения задач, связанных с организационными проблемами управления предприятием. Orgware - новый класс программ - был ориентирован на решение задач систематизации, хранения и обработки "неколичественной" информации об организации бизнеса, которые раньше не имели адекватной компьютерной поддержки.

Первый российский продукт - БИГ-Мастер - был создан как компьютерный инструмент для поддержки определенной концепции управления предприятием, получившей название регулярного менеджмента. Главной задачей orgware был переход к строго документированным процедурам и регламентам деятельности. В основу компьютерной парадигмы регулярного менеджмента был положен следующий подход: "Надо создавать не систему взаимосвязанных документов, а систему взаимосвязанных информационных моделей предприятия, которые и будут порождать требуемые документы".

Концептуальной основой БИГ-Мастера стал современный процессный подход к организации деятельности компании. На верхнем уровне система процессов обычно описывается деревом функций - для его обозначения часто используется термин *функционал*. Функции здесь рассматриваются в качестве "свернутых" процессов. Все процессы-функции, как минимум, должны быть определены (т.е. идентифицированы как вид деятельности, имеющий некую цель и результаты) и классифицированы по видам (основные, обеспечивающие, процессы управления). Также должны быть распределены ответственность и полномочия для управления процессами на регулярной основе. На этом уровне для описания компании в БИГ-Мастере применяются два типа моделей: *древовидные модели (классификаторы)* и *матричные модели (проекции)*.

На нижнем уровне выделенные ("ключевые") процессы могут быть описаны как технологическая последовательность операций (для получения требуемых результатов). Для этого применяются *потокковые модели* бизнес-процессов, назначение которых - описание горизонтальных отношений в организации, связывающих между собой описанные ранее объекты посредством информационных и материальных потоков. Для структурного анализа и проектирования процессов, описываемых *потокковыми моделями*, БИГ-Мастер поддерживает методологию SADT (IDEF). Наличие механизма матричных проекций позволяет определить и описать процессы компании как целостную взаимосвязанную систему.

За счет иерархической структуры классификаторов бизнес-модель одновременно содержит отношения "функция-исполнитель" всех степеней детализации, что позволяет с помощью встроенного генератора отчетов настраивать "разрешение" взгляда на компанию применительно к конкретной управленческой задаче. Система проекций позволяет отразить в отчете любые дополнительные свойства, относящиеся к данному объекту (например, квалификационные требования для персонала, задействованного в процессе). Кроме того, взгляд на компанию может быть связан с любой "координатой отсчета" - например, от документа или сотрудника - в каких процессах и как они участвуют и т.п.

Классификаторы, проекции и *потоковые модели* бизнес-процессов поддерживаются различными способами их визуализации. Для классификаторов - в виде списков и деревьев (оргафов), для проекции - в виде связанных списков и транспонируемых матриц, а для *потоковых моделей* бизнес-процессов - в виде диаграмм IDEF0 (IDEF3) и текстового описания, что облегчает понимание задач участниками процессов. При этом конструирование самих *потоковых моделей* происходит в привычных табличных формах.

В модели возможно формирование неограниченного количества новых классификаторов, проекций и *потоковых моделей*, а следовательно, отчетов и документов для описания и, что особенно важно, создания регламентов деятельности компании.

Наличие в БИГ-Мастере нескольких инструментов моделирования является чрезвычайно полезным. *Матричные модели* поддерживают вертикальную интеграцию - подробное системно-целевое описание компании, выстроенное по иерархии управления и исполняемым функциям. В процессной модели преобладает функционально-технологический подход - горизонтальная интеграция бизнес-операций по процедурам. Все вышеперечисленные возможности БИГ-Мастера делают его удобным инструментальным средством организационного моделирования.

Процессные потоковые модели

Разработка требований к проектируемой ИС строится на основе статического и динамического описания компании. Статическое описание компании, рассмотренное в лекции 4, проводится на уровне функциональных моделей и включает описание бизнес-потенциала, функционала и соответствующих матриц ответственности.

Дальнейшее развитие (детализация) бизнес-модели происходит на этапе динамического описания компании на уровне *процессных потоковых моделей*.

Процессные потоковые модели — это модели, описывающие процесс последовательного во времени преобразования материальных и информационных потоков компании в ходе реализации какой-либо бизнес-функции или функции менеджмента. На верхнем уровне описывается логика взаимодействия участников процесса, на нижнем — технология работы отдельных специалистов на своих рабочих местах. *Процессные потоковые модели* отвечают на вопросы **кто—что—как—кому** (см. лекцию 4 [рис. 4.3](#)).

Современное состояние экономики характеризуется переходом от традиционной функциональной модели деятельности компании, построенной на принципах разделения труда, узкой специализации и жестких иерархических структурах, к модели процессной, основанной на интеграции работ вокруг бизнес-процессов.

Главными недостатками функционального подхода являются:

- разбиение технологий выполнения работы на отдельные фрагменты, иногда между собой несвязанные, которые выполняются различными структурными подразделениями;
- отсутствие целостного описания технологий выполнения работы;
- сложность увязывания простейших задач в технологию, производящую реальный товар или услугу;
- отсутствие ответственности за конечный результат;
- высокие затраты на согласование, налаживание взаимодействия, контроль и т. д.;
- отсутствие ориентации на клиента.

Процессный подход предполагает смещение акцентов от управления отдельными структурными элементами на управление сквозными бизнес-процессами, связывающими деятельность всех структурных элементов. Каждый деловой процесс проходит через ряд подразделений, т. е. в его выполнении участвуют специалисты различных отделов компании. Чаще всего приходится сталкиваться с ситуацией, когда собственно процессами никто не управляет, а управляют лишь подразделениями. Более того, структура компаний строится без учета возможностей оптимизации деловых процессов, обеспечивающих необходимые функции. *Процессный подход* позволяет устранить фрагментарность в работе, организационные и информационные разрывы, дублирование, нерациональное использование финансовых, материальных и кадровых ресурсов.

Процессный подход к организации деятельности предприятия предполагает:

- широкое делегирование полномочий и ответственности исполнителям;
- сокращение уровней принятия решений;
- сочетание принципа целевого управления с групповой организацией труда;
- повышенное внимание к вопросам обеспечения качества;
- автоматизация технологий выполнения бизнес-процессов.

Согласно стандарту "Основные Положения и Словарь — ИСО/ОПМС 9000:2000" (п. 2.4) понятие "**Процессный подход**" определяется как:

"Любая деятельность, или комплекс деятельности, в которой используются ресурсы для преобразования входов в выходы, может рассматриваться как процесс. Чтобы результативно функционировать, организации должны определять и управлять многочисленными взаимосвязанными и взаимодействующими процессами. Часто выход одного процесса образует непосредственно вход следующего. Систематическая идентификация и менеджмент применяемых организацией процессов, и особенно взаимодействия таких процессов, могут считаться "процессным подходом".

Основной принцип процессного подхода определяет структурирование бизнес-системы в соответствии с деятельностью и бизнес-процессами предприятия, а не в соответствии с его организационно-штатной структурой. Именно бизнес-процессы, обеспечивающие значимый для потребителя результат, представляют ценность и для специалистов, проектирующих ИС. Процессная модель компании должна строиться с учетом следующих положений:

1. Верхний уровень модели должен отражать только контекст диаграммы – взаимодействие моделируемого единственным контекстным процессом предприятия с внешним миром.
2. На втором уровне должны быть отражены тематически сгруппированные бизнес-процессы предприятия и их взаимосвязи.
3. Каждая из деятельности должна быть детализирована на бизнес-процессы.
4. Детализация бизнес-процессов осуществляется посредством бизнес – функций.
5. Описание элементарной бизнес-операции осуществляется с помощью миниспецификации.

Процессный подход требует комплексного изучения различных сторон жизни организации — правовых основ и правил деятельности, организационной структуры, функций и показателей результатов их исполнения, интерфейсов, ресурсного обеспечения, организационной культуры. В результате анализа создается модель деятельности "как есть". Обработка этой модели с помощью различных аналитических методов позволяет проверить, насколько деловые процессы рациональны, а также определить, является ли та или иная операция ориентированной на общественно значимый конечный результат или излишней бюрократической процедурой.

В ходе анализа деловых процессов детально исследуются сферы ответственности подразделений ведомства, его руководителей и сотрудников. Это позволяет установить адреса владельцев деловых процессов, в результате чего процессы перестают быть бесхозными, создаются условия для разработки и внедрения систем стимулирования и ответственности за конечные результаты, определяются моменты и процедуры передачи ответственности. Анализ и оценка деловых процессов позволяют подойти к обоснованию стандартов их выполнения, допустимых рисков и диапазонов свободы принятия решений исполнителями, предельных нормативов затрат ресурсов на единицу эффекта.

Однако чисто "процессная компания" является скорее иллюстрацией правильной организации работ. В действительности все бизнес-процессы компании протекают в рамках организационной структуры предприятия, описывающей функциональные компетентности и отношения.

Управление всей текущей деятельностью компании ведется по двум направлениям — управление функциональными областями, которые поддерживают множество унифицированных бизнес-процессов, разделенных на операции, и управление интегрированными бизнес-процессами, задачей которого является маршрутизация и координация унифицированных процессов для выполнения как оперативных заказов потребителей, так и глобальных проектов самой организации (рис. 5.1).

	Функциональная область 1	Функциональная область 2	Функциональная область 3	Функциональная область N
Процесс 1 (⇒ ↑↑ операции, ↑↑ исполнитель)	↑↑ ⇒	→	↑↑ ⇒	↑↑ ⇒
Процесс 2				↑↑ ⇒
Процесс 2		↑↑ ⇒	↗	
Процесс 1 (⇒ ↑↑ операции, ↑↑ исполнитель)	↑↑ ⇒			

Рис. 5.1. Схема управления деятельностью компании

Фактически основной задачей организационного проектирования является выбор оптимального соотношения между эффективностью использования ресурсов и эффективностью процессов. Жесткая специализация подразделений экономит ресурсы организации, но снижает качество реализации процессов. Создание "процессных" команд, включающих собственных специалистов по всем ключевым операциям, обходится достаточно дорого, но при этом значительно сокращается время и повышается точность выполнения процесса. Иногда организации могут позволить себе выбрать этот путь, особенно в тех случаях, когда создается высокая ценность процесса, за которую потребитель согласен платить. Но, как правило, ищется какой-то компромисс на основе процессно-матричных структур. Когда компания начинает ориентироваться на процессы, исключительно важной становится роль *владельцев* интегрированных межфункциональных процессов, касающихся многих функциональных областей. Кроме того, новая парадигма деятельности предприятия вызывает появление большого числа процессов управления, распределенных по всему предприятию, а не сосредоточенных в специализированных организационных единицах: это системы качества, бюджетирования, маркетинга и т.п. Поэтому постановка бюджетирования как организационной, а не только финансовой задачи предполагает делегирование полномочий, т.е. власти (с которой нелегко расстаться). На более низкие уровни делегируется ответственность за принятие финансовых решений: о заключении сделки-договора, об оплате, о закупке, о скидках и отпуске в кредит и т.п. Это позволяет упростить связи между подразделениями и снизить количество уровней вертикального прохождения документов, т.е. является необходимым условием реализации классической схемы реинжиниринга. Таким образом, процессная ориентация ведет к перестройке организационной структуры, делает организационную структуру компании более "плоской", что иллюстрирует тесную связь между "вертикальным" описанием организации (как структуры распределения ответственности, полномочий и взаимоотношений) и ее "горизонтальным" описанием, как системы процессов.

Основные элементы процессного подхода

В рамках *процессного подхода* любое предприятие рассматривается как бизнес-система – система, которая представляет собой связанное множество бизнес-процессов, конечными целями которых является выпуск продукции или услуг.

Под бизнес-процессом понимают совокупность различных видов деятельности, которые создают результат, имеющий ценность для потребителя. Бизнес-процесс – это цепочка работ (функций), результатом которой является какой-либо продукт или услуга.

Каждый бизнес-процесс имеет свои границы (подробнее см. лекции 6, 7) и роли.

В *процессном подходе* используются следующие ключевые роли:

Владелец процесса – человек, отвечающий за ход и результаты процесса в целом. Он должен знать бизнес-процесс, следить за его выполнением и совершенствовать его эффективность. *Владельцу бизнес-процесса* необходимо обладать коммуникативностью, энтузиазмом, способностью влиять на людей и производить изменения.

Лидер команды — работник, обладающий знаниями о бизнес-процессе и имеющий позитивные личные качества.

Коммуникатор – работник, обучающий команду различным методам работы, подготавливающий совместно с *лидером* совещания и анализирующий их результат.

Координатор процесса – работник, отвечающий за согласованную работу всех частей бизнеса и обеспечивающий связь с другими бизнес-процессами. *Координатор* должен обладать административными способностями и пониманием стратегических целей предприятия.

Участники команды – специалисты различных уровней иерархии. *Участники команды* получают поддержку и методическое обеспечение от консультанта и *коммуникатора*, вместе с *лидером* проводят моделирование, анализ и оценку бизнес-процесса.

Одним из основных элементов *процессного подхода* является команда. Существует несколько типов процессных команд:

Ситуационная команда – обычно работает на постоянной основе и выполняет периодически повторяющуюся работу.

Виртуальная команда – создается для разработки нового продукта или услуги.

Ситуационный менеджер – высококвалифицированный специалист, способный самостоятельно выполнить до 90% объема работ.

Важной задачей *процессного подхода* является формирование процессных команд. Подготовка и формирование команды включает:

- учебные курсы;
- практический тренинг по освоению методов, методик и др.;
- психологическое тестирование;
- тестирование рабочих навыков.

Достижение определенной совокупности целей за счет выполнения бизнес-процессов называется **деревом целей**. *Дерево целей* имеет, как правило, иерархический вид. Каждая цель имеет свой вес и критерий (количественный или качественный) достижимости.

Бизнес-процессы реализуют бизнес-функции предприятия. Под бизнес-функцией понимают вид деятельности предприятия. Множество бизнес-функций представляет иерархическую декомпозицию функциональной деятельности и называется *деревом функций*.

Бизнес-функции связаны с показателями деятельности предприятия, образующими **дерево показателей**. На основании показателей строится система показателей оценки эффективности выполнения процессов. *Владельцы процессов* контролируют свои бизнес-процессы с помощью данной системы показателей. Наиболее общими показателями оценки эффективности бизнес-процессов являются:

- количество производимой продукции заданного качества за определенный интервал времени;
- количество потребляемой продукции;
- длительность выполнения типовых операций и др.

Выделение и классификация процессов

При процессном описании должны решаться, как минимум, две задачи:

1. Идентификация всей системы "функциональных областей" и процессов компании и их взаимосвязей.
2. Выделение "ключевых" интегрированных процессов и их описание на потоковом уровне.

Каждая деятельность компании реализуется как процесс, который имеет своего потребителя: внешнего — клиента или внутреннего — сотрудников или подразделения компании, реализующих другие процессы. На стадии системного описания процессов и выявляется значимость каждого процесса — в том числе происходит очищение от малопонятной деятельности. На этом этапе выбираются **ключевые процессы** для потокового описания, которое необходимо, например, для создания информационной системы предприятия.

Наиболее распространены следующие четыре вида бизнес-процессов:

1. Процессы, создающие наибольшую добавленную стоимость (экономическую стоимость, которая определяется издержками компании, относимыми на продукцию).
2. Процессы, создающие наибольшую ценность для клиентов (маркетинговую стоимость за счет дифференциации продукции).
3. Процессы с наиболее интенсивным межзвенным взаимодействием, создающие транзакционные издержки.
4. Процессы, определенные стандартами ИСО 9000, как обязательные к описанию при постановке системы менеджмента качества.

Важнейшим шагом при структуризации любой компании является выделение и классификация бизнес-процессов. Целесообразно основываться на следующих классах процессов:

- основные;
- *процессы управления*;
- *процессы обеспечения*;
- сопутствующие;
- вспомогательные;
- процессы развития.

Рассмотрим модель деятельности компании ([рис. 5.2](#)), при описании которой используют *процессы управления*, основные бизнес-процессы и *процессы обеспечения*.

Основные бизнес-процессы — это процессы, ориентированные на производство товаров и услуг, представляющие ценность для клиента и обеспечивающие получение дохода.



Рис. 5.2. Упрощенная модель деятельности компании

Основные процессы образуют "жизненный цикл" продукции компании. Критериями эффективности таких процессов являются обычно качество, точность и своевременность выполнения каждого заказа. Многие потребители рассматривают увеличение качества как нечто более важное, чем уменьшение цены. Искусный продавец может получить заказ на выполнение работ в условиях конкуренции с другими фирмами, однако только качество товара или услуги определяет в большей степени, повторит ли потребитель свой заказ у этого продавца еще раз. Таких процессов, при развитой деятельности компании, может быть много. Все они описываются по производственно-коммерческим цепочкам: "первичное взаимодействие с клиентом и определение его потребностей → реализация запроса (заявки, заказа, контракта и т.п.) → послепродажное сопровождение и мониторинг удовлетворения потребностей". Процесс "реализации (запроса клиента)" может быть декомпозирован на следующие подпроцессы — процессы более низкого уровня:

- разработка (проектирование) продукции;
- закупка (товаров, материалов, комплектующих изделий);
- транспортировка (закупленного);
- разгрузка, приемка на склад и хранение (закупленного);
- производство (со своим технологическим циклом и внутренней логистикой);
- приемка на склад и хранение (готовой продукции);
- отгрузка (консервация и упаковка, погрузка, доставка);
- пуско-наладка;
- оказание услуг (предусмотренных контрактом на поставку или имеющих самостоятельное значение) и т.п.

Эти этапы цепочки также достаточно стандартны (например, в стандарте ИСО редакции 1994 г. приведены многие из этих процессов в качестве обязательных и подлежащих сертификации). Проверить, какие бизнес-цепочки существуют на предприятии, можно с помощью проекции каждого из выделенных "бизнесов, продукции и услуг" на вышеуказанный (стандартный) библиотечный классификатор жизненного или уже производственного цикла.

Для оценки этапов работы с любым документом можно использовать также анализ "жизненного цикла документа", который может выглядеть следующим образом:

- предоставляет исходные данные;
- подготавливает, разрабатывает;
- заполняет;
- корректирует;
- оформляет;
- подписывает;
- контролирует соответствие установленным требованиям;
- визирует;
- согласует;
- утверждает;

- акцентирует (принимает к сведению, использует);
- хранит;
- снимает копию.

Здесь тоже может быть применена своя матрица-генератор, как средство проверки полноты, — идентификация цикла.

Можно также воспользоваться *референтными моделями* деятельности аналогичных компаний — они могут сопоставляться с процессами конкурентов, лидеров отрасли, а также совершенствоваться.

Процессы управления – это процессы, охватывающие весь комплекс функций управления на уровне каждого бизнес-процесса и бизнес-системы в целом. *Процессы управления* имеют своей целью выработку и принятие управленческого решения. Данные управленческие решения могут приниматься относительно всей организации в целом, отдельной функциональной области или отдельных процессов, например:

- стратегическое управление;
- организационное проектирование (структуризация);
- маркетинг;
- финансово-экономическое управление;
- логистика и организация процессов;
- менеджмент качества;
- персонал.

Другая возможная **систематизация функций управления** связана с понятием управленческого цикла и базируется на пяти исходных функциях управления: планирование, организация, распоряительство, координация, контроль. Самая распространенная ошибка — это смешение этих принципов.

Для реализации процессного описания исключительно важным является то, что любая управленческая деятельность разворачивается по так называемому "управленческому циклу", который включает:

- сбор информации;
- выработку решения;
- реализацию;
- учет;
- контроль;
- анализ;
- регулирование.

Например, наиболее часто встречающиеся варианты детализации:

- сбор информации;
- определение состава собираемой информации;
- определение форм отчетности.
- выработка решения;
- анализ альтернатив;
- подготовка вариантов решения;
- принятие решения;
- выработка критериев оценки;
- реализация;
- планирование;
- организация;
- мотивация;
- координация;

- **контроль исполнения**
- учет результатов;
- сравнение по принятым критериям;
- **анализ**
- анализ дополнительной информации;
- диагностика возможных причин отклонений;
- **регулирование**
- регулирование на уровне реализации (возврат к п.3);
- регулирование на уровне выработки решения (возврат к п.1,2)

Каждый из этих этапов имеет своих характерных для него исполнителей — управленцев, которых можно отнести к трем основным категориям:

- руководитель (ответственный за принятие и организацию выполнения решений);
- специалист-аналитик (ответственный за подготовку решения и анализ отклонений);
- технические исполнители (сбор информации, учет, коммуникации).

Согласно некоторым подходам, в *процессах управления* выделяются два типа процессов, относящихся, соответственно, к двум типам менеджмента, условно обозначаемым как "менеджмент ресурсов" и "менеджмент организации", которые отличаются по объекту управления, базовым моделям и, что важно для описания процессов, — своими управленческими циклами. Тогда модель деятельности предприятия становится двухуровневой ([рис .5.3](#))



Рис. 5.3. Двухуровневая модель деятельности предприятия

Из этой модели следует, что сами циклы ресурсного планирования нуждаются в регламентации — то есть ресурсное управление может осуществляться только по специально разработанным организационным регламентам.

В основе **цикла управления ресурсами** лежит расчет или имитационное моделирование и контроль результатов:

- выбор (или получение от системы верхнего уровня) целевого критерия оценки качества решения;
- сбор информации о ресурсах предприятия или возможностях внешней среды;
- просчет вариантов (с различными предположениями о возможных значениях параметров);
- выбор оптимального варианта — принятие решения (= ресурсного плана);
- учет результатов (и отчетность);
- сравнение с принятым критерием оценки (= контроль результатов);
- анализ причин отклонений и регулирование (возврат к 1, 2 или 3).

В основе **цикла организационного менеджмента** лежит структурное или процессное моделирование и процедурный контроль:

- определение состава задач (обособленных функций, операций);
- выбор исполнителей (- распределение зон и степени ответственности);
- проектирование процедур (последовательности и порядка исполнения);
- согласование и утверждение регламента исполнения (- процесса, плана мероприятий);
- отчетность об исполнении;
- контроль исполнения (- процедурный контроль);
- анализ причин отклонений и регулирование (возврат к 1, 2 или 3).

Таким образом, на определенных шагах декомпозиции предприятию надо определить, какие стадии управленческого цикла реализуются по каждой из ранее выделенных задач управления. Это можно проверить с помощью матрицы-генератора, которая раскладывает компоненты менеджмента по этапам управленческого цикла.

Процессы обеспечения – это процессы, предназначенные для жизнеобеспечения основных и сопутствующих процессов и ориентированные на поддержку их универсальных средств. Например, процесс финансового обеспечения, *процесс обеспечения* кадрами, процесс юридического обеспечения — это вторичные процессы. Они создают и поддерживают необходимые условия для выполнения основных функций и функций менеджмента. Клиенты обеспечивающих процессов находятся внутри компании.

На верхнем уровне детализации можно выделить примерно следующие стандартные *процессы обеспечения*:

- обеспечение производства;
- техобслуживание и ремонт оборудования;
- обеспечение теплоэнергоресурсами;
- обслуживание и ремонт зданий и сооружений;
- технологическое обеспечение;
- метрологическое;
- техника безопасности;
- экологический контроль и т.п.
- обеспечение управления;
- информационное обеспечение;
- обеспечение документооборота;
- коммуникационное обеспечение;
- юридическое обеспечение;
- обеспечение безопасности;
- материально-техническое обеспечение управления;
- хозяйственное обеспечение;
- обеспечение коммунальными услугами;
- транспортное обслуживание и т.п.

Для каждого из выделенных выше подпроцессов также следует определить, какой основной или управленческий процесс является потребителем этих "внутренних" услуг. Для этого существуют свои матрицы-генераторы. Их можно построить отдельно для основных процессов ([рис. 5.4](#)) и процессов управления ([рис. 5.5](#)).

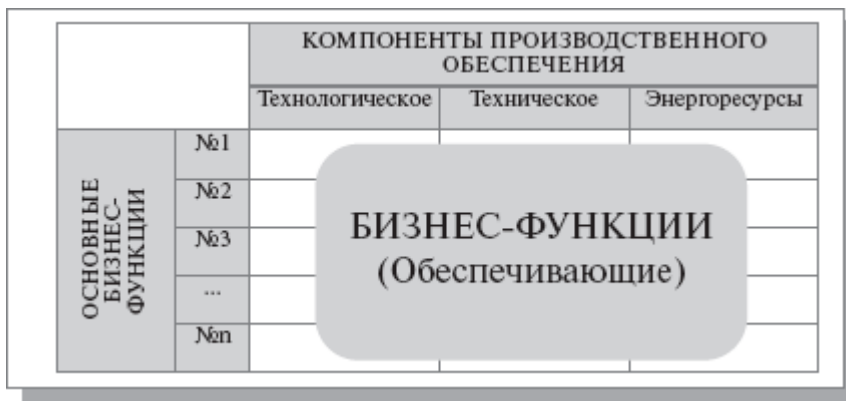


Рис. 5.4. Упрощенная матрица-генератор обеспечивающих бизнес-функций



Рис. 5.5. Матрица-генератор обеспечивающих бизнес-функций

Разбиение данных процессов производится по индивидуальным технологическим цепочкам. Многие из обеспечивающих процессов стандартны для всех компаний или определенных видов деятельности: промышленность, торговля, предоставление услуг и т.п. Однако, как правило, данный класс функций в меньшей степени "подвергается" потоковому процессному описанию. Большинство из них достаточно хорошо регламентируются должностными и специальными инструкциями.

Референтная модель бизнес-процесса

В качестве основного каркаса, объединяющего и систематизирующего все знания по бизнес-модели, можно использовать *референтную модель*. **Референтная модель** — это модель эффективного бизнес-процесса, созданная для предприятия конкретной отрасли, внедренная на практике и предназначенная для использования при разработке/реорганизации бизнес-процессов на других предприятиях. По сути, *референтные модели* представляют собой эталонные схемы организации бизнеса, разработанные для конкретных бизнес-процессов на основе реального опыта внедрения в различных компаниях по всему миру. Они включают в себя проверенные на практике процедуры и методы организации управления. *Референтные модели* позволяют предприятиям начать разработку собственных моделей на базе уже готового набора функций и процессов.

Референтная модель бизнес-процесса представляет собой совокупность логически взаимосвязанных функций. Для каждой функции указывается исполнитель, входные и выходные документы или информационные объекты. Элементы (функции и документы) *референтной модели* бизнес-процесса содержат ссылки на соответствующие объекты ИС, а также документы и другую информацию (пользовательские инструкции,

ответственных разработчиков), расположенную в репозитории проекта. Отсюда и название — *референтная модель* (в переводе с английского ссылочная модель).

Проведение предпроектного обследования предприятий

Обследование предприятия является важным и определяющим этапом проектирования ИС. Длительность обследования обычно составляет 1-2 недели. В течение этого времени системный аналитик должен обследовать не более 2-3 видов деятельности (учет кадров, бухгалтерия, перевозки, маркетинг и др.).

Сбор информации для построения полной бизнес-модели организации часто сводится к изучению документированных информационных потоков и функций подразделений, а также производится путем интервьюирования и анкетирования.

К началу работ по обследованию организация обычно предоставляет комплект документов, в состав которого обычно входят:

1. Сводная информация о деятельности предприятия.
 1. Информация об управленческой, финансово-экономической, производственной деятельности предприятия.
 2. Сведения об учетной политике и отчетности.
2. Регулярный документооборот предприятия.
 1. Реестр входящей информации.
 2. Реестр внутренней информации.
 3. Реестр исходящей информации.
3. Сведения об информационно-вычислительной инфраструктуре предприятия.
4. Сведения об ответственных лицах.

Таблица 5.1. РЕЕСТР ВХОДЯЩЕЙ ИНФОРМАЦИИ

(Наименование предприятия)	(Наименование подразделения)		Характеристики обработки документов		
№ Наименование и назначение документа	Кто обрабатывает	Откуда поступает	Трудоемкость	Периодичность, регламент	Способ получения

Таблица 5.2. РЕЕСТР ВНУТРЕННЕЙ ИНФОРМАЦИИ

(Наименование предприятия)	(Наименование подразделения)		Характеристики обработки документов		
№ Наименование и назначение документа	Кто обрабатывает	Кому передает	Трудоемкость	Периодичность, регламент	Способ получения

Таблица 5.3. РЕЕСТР ИСХОДЯЩЕЙ ИНФОРМАЦИИ

(Наименование предприятия)	(Наименование подразделения)		Характеристики обработки документов		
№ Наименование и назначение документа	Кто обрабатывает	Куда поступает	Трудоемкость	Периодичность, регламент	Способ получения

Списки вопросов для интервьюирования и анкетирования составляются по каждому обследуемому подразделению и утверждаются руководителем компании. Это делается с целью:

- предотвращения доступа к конфиденциальной информации;

- усиления целевой направленности обследования;
- минимизации отвлечения сотрудников предприятий от выполнения должностных обязанностей.

Общий перечень вопросов (с их последующей детализацией) включает следующие пункты:

- основные задачи подразделений;
- собираемая и регистрируемая информация;
- отчетность;
- взаимодействие с другими подразделениями.

Анкететы для руководителей и специалистов могут содержать следующие вопросы:

- Каковы (с позиций вашего подразделения) должны быть цели создания интегрированной системы управления предприятием?
- Организационная структура подразделения.
- Задачи подразделения.
- Последовательность действий при выполнении задач.
- С какими типами внешних организаций (банк, заказчик, поставщик и т.п.) взаимодействует подразделение и какой информацией обменивается?
- Каким справочным материалом вы пользуетесь?
- Сколько времени (в минутах) вы тратите на исполнение основных операций? На какие даты приходятся "пиковые нагрузки"? (периодичность в месяц, квартал, год и т.д.) Техническое оснащение подразделения (компьютеры, сеть, модем и т.п.). Используемые программные продукты для автоматизации бизнес-процессов.
- Какие отчеты и как часто вы готовите для руководства? Ключевые специалисты подразделения, способные ответить на любые вопросы по бизнес-процессам, применяемым в подразделении.
- Характеристики удаленных объектов управления.
- Документооборот на рабочем месте.

Собранные таким образом данные, как правило, не охватывают всех существенных сторон организационной деятельности и обладают высокой степенью субъективности. И самое главное, что такого рода обследования не выявляют устойчивых факторов, связанных со специфическими особенностями организации, воздействовать на которые можно исключительно методами функциональной настройки организационной системы.

Анализ опросов руководителей обследуемых организаций и предприятий показывает, что их представления о структуре организации, общих и локальных целях функционирования, задачах и функциях подразделений, а также подчиненности работников иногда имеют противоречивый характер. Кроме того, эти представления подчас расходятся с официально декларируемыми целями и правилами или противоречат фактической деятельности.

Если структуру информационных потоков можно выявить по образцам документов и конфигурациям компьютерных сетей и баз данных, то структура реальных микропроцессов, осуществляемых персоналом в информационных контактах (в значительной мере недокументированных) остается неизвестной. Ответы на эти вопросы может дать **структурно-функциональная диагностика**, основанная на методах сплошной (или выборочной) фотографии рабочего времени персонала. Цель диагностики — получение достоверного знания об организации и организационных отношениях ее функциональных элементов. В связи с этим к важнейшим задачам функциональной диагностики организационных структур относятся:

- классификация субъектов функционирования (категорий и групп работников);
- классификация элементов процесса функционирования (действий, процедур);
- классификация направлений (решаемых проблем), целей функционирования;

- классификация элементов информационных потоков;
- проведение обследования деятельности персонала организации;
- исследование распределения (по времени и частоте) организационных характеристик: процедур, контактов персонала, направлений деятельности, элементов информационных потоков — по отдельности и в комбинациях друг с другом по категориям работников, видам процедур и их направлениям (согласно результатам и логике исследований);
- выявление реальной структуры функциональных, информационных, иерархических, временных, проблемных отношений между руководителями, сотрудниками и подразделениями;
- установление структуры распределения рабочего времени руководителей и персонала относительно функций, проблем и целей организации;
- выявление основных технологий функционирования организации (информационных процессов, включая и недокументированные), их целеполагания в сравнении с декларируемыми целями организации;
- выявление однородных по специфике деятельности, целевой ориентации и реальной подчиненности групп работников, формирование реальной модели организационной структуры и сравнение ее с декларируемой;
- определение причин рассогласования декларируемой и реальной структуры организационных отношений.

Сплошной "фотографией" рабочего времени называется **непрерывное наблюдение и регистрация характеристик работников в процессе функционирования в течение всего рабочего дня**. При этом индицируемые параметры последовательно вносятся в заранее заготовленную рабочую таблицу. Ниже представлена форма рабочей таблицы системного аналитика;

№	Агент	Время	Процедура	Содержание	Информация	Инициатива	Контрагент	Отношение	Проблем	Примечание
1	2*	3*	4	5*	6	7*	8*	9	10	11

Сразу по окончании процедуры обследования таблица пополняется дополнительными характеристиками: технологическая ветвь, системная функция, предмет, аспект, эмоциональный фон и др.

Часть показателей, те, что помечены звездочкой, заполняются в процессе обследования, остальные — после. Содержание записей следующее:

- номер (по порядку);
- агент (должность обследуемого работника);
- время, в течение которого выполнялась процедура;
- процедура (наименование содержания совокупности элементарных действий, объединенных общностью решаемой частной задачи);
- содержание (суть процедуры, которая должна быть классифицирована);
- информация (направление движения информации между агентом и контрагентом);
- инициатива (инициатор начала выполнения данной процедуры);
- контрагент (должность работника, который находится с обследуемым в контакте);
- отношение (отражающая субординацию агента и контрагента форма взаимодействия в данной процедуре);
- проблема (словесная характеристика решаемой проблемы).

Результаты предпроектного обследования

Результатом предпроектного обследования является "Отчет об экспресс-обследовании предприятия", структура которого приведена ниже.

1. Краткое схематичное описание бизнес-процессов:
 - управление закупками и запасами;
 - управление производством;
 - управление продажами;
 - управление финансовыми ресурсами.
2. Основные требования и приоритеты автоматизации.
3. Оценка необходимых для обеспечения проекта ресурсов заказчика.
4. Оценка возможности автоматизации, предложения по созданию автоматизированной системы с оценкой примерных сроков и стоимости.

Документы, входящие в отчет об обследовании, могут быть представлены в виде текстового описания или таблиц, примерная форма которых приведена ниже.

№	Б-П	Наименование бизнес-процесса
1.		Продажи: сеть, опт
2.		План закупок
3.		Размещение заказа на производство
4.		Производство собственное
5.		Закупка сырья
6.		Платежи
7.		Другие

Операции бизнес-процесса

Операция	Исполнитель	Как часто	Входящие документы (документы-основания)	Исходящий документ (составляемый документ)
----------	-------------	-----------	--	--

Описание документов бизнес-процесса

Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)
--	----------	------------------------------	-----------	--

Проведение предпроектного обследования позволяет решить следующие задачи:

- предварительное выявление требований к будущей системе;
- определение структуры организации;
- определение перечня целевых функций организации;
- анализ распределения функций по подразделениям и сотрудникам;
- выявление функциональных взаимодействий между подразделениями, информационных потоков внутри подразделений и между ними, внешних информационных воздействий;
- анализ существующих средств автоматизации организации.

Информация, полученная в результате предпроектного обследования, анализируется с помощью методов структурного и/или объектного анализа, о которых будет сказано ниже, и используется для построения моделей деятельности организации. Модель организации предполагает построение двух видов моделей:

- модели "как есть", отражающей существующее на момент обследования положение дел в организации и позволяющей понять, каким образом

функционирует данная организация, а также выявить узкие места и сформулировать предложения по улучшению;

- модели "как должно быть", отражающей представление о новых технологиях работы организации. Каждая из моделей включает в себя полную функциональную и информационную модель деятельности организации, а также модель, описывающую динамику поведения организации (в случае необходимости).

Структурная модель предметной области

В основе проектирования ИС лежит моделирование предметной области. Для того чтобы получить адекватный предметной области проект ИС в виде системы правильно работающих программ, необходимо иметь целостное, системное представление модели, которое отражает все аспекты функционирования будущей информационной системы. При этом под **моделью предметной области** понимается некоторая система, имитирующая структуру или функционирование исследуемой предметной области и отвечающая основному требованию – быть адекватной этой области.

Предварительное моделирование предметной области позволяет сократить время и сроки проведения проектировочных работ и получить более эффективный и качественный проект. Без проведения моделирования предметной области велика вероятность допущения большого количества ошибок в решении стратегических вопросов, приводящих к экономическим потерям и высоким затратам на последующее перепроектирование системы. Вследствие этого все современные технологии проектирования ИС основываются на использовании методологии моделирования предметной области.

К *моделям предметных областей* предъявляются следующие требования:

- формализация, обеспечивающая однозначное описание структуры предметной области;
- понятность для заказчиков и разработчиков на основе применения графических средств отображения модели;
- реализуемость, подразумевающая наличие средств физической реализации *модели предметной области* в ИС;
- обеспечение оценки эффективности реализации *модели предметной области* на основе определенных методов и вычисляемых показателей.

Для реализации перечисленных требований, как правило, строится **система моделей**, которая отражает структурный и оценочный аспекты функционирования предметной области.

Структурный аспект предполагает построение:

- объектной структуры, отражающей состав взаимодействующих в процессах материальных и информационных объектов предметной области;
- функциональной структуры, отражающей взаимосвязь *функций* (действий) по преобразованию объектов в процессах;
- структуры управления, отражающей события и бизнес-правила, которые воздействуют на выполнение процессов;
- организационной структуры, отражающей взаимодействие организационных единиц предприятия и персонала в процессах;
- технической структуры, описывающей топологию расположения и способы коммуникации комплекса технических средств.

Для отображения структурного аспекта *моделей предметных областей* в основном используются графические методы, которые должны гарантировать представление информации о компонентах системы. Главное требование к графическим методам документирования — простота. Графические методы должны обеспечивать возможность структурной декомпозиции спецификаций системы с максимальной степенью детализации и согласований описаний на смежных уровнях декомпозиции.

С моделированием непосредственно связана проблема **выбора языка** представления проектных решений, позволяющего как можно больше привлекать будущих пользователей системы к ее разработке. **Язык моделирования** – это *нотация*, в основном графическая, которая используется для описания проектов. **Нотация** представляет собой совокупность графических объектов, используемых в модели. *Нотация является синтаксисом языка моделирования. Язык моделирования, с одной стороны, должен делать решения проектировщиков понятными пользователю, с другой стороны, предоставлять проектировщикам средства достаточно формализованного и однозначного определения проектных решений, подлежащих реализации в виде программных комплексов, образующих целостную систему программного обеспечения.*

Графическое изображение нередко оказывается наиболее емкой формой представления информации. При этом проектировщики должны учитывать, что графические методы документирования не могут полностью обеспечить декомпозицию проектных решений от постановки задачи проектирования до реализации программ ЭВМ. Трудности возникают при переходе от этапа анализа системы к этапу проектирования и в особенности к программированию.

Главный **критерий адекватности структурной модели предметной области** заключается в функциональной полноте разрабатываемой ИС.

Оценочные аспекты моделирования предметной области связаны с разрабатываемыми показателями эффективности автоматизируемых процессов, к которым относятся:

- время решения задач;
- стоимостные затраты на обработку данных;
- надежность процессов;
- косвенные показатели эффективности, такие, как объемы производства, производительность труда, оборачиваемость капитала, рентабельность и т.д.

Для расчета показателей эффективности, как правило, используются статические методы функционально-стоимостного анализа (АВС) и динамические методы имитационного моделирования.

В основе различных методологий моделирования предметной области ИС лежат принципы последовательной детализации абстрактных категорий. Обычно модели строятся на трех уровнях: на внешнем уровне (**определении требований**), на концептуальном уровне (**спецификации требований**) и внутреннем уровне (**реализации требований**). Так, на внешнем уровне модель отвечает на вопрос, что должна делать система, то есть определяется состав основных компонентов системы: объектов, функций, событий, организационных единиц, технических средств. **На концептуальном уровне** модель отвечает на вопрос, как должна функционировать система? Иначе говоря, определяется характер взаимодействия компонентов системы одного и разных типов. На внутреннем уровне модель отвечает на вопрос: с помощью каких программно-технических средств реализуются требования к системе? С позиции жизненного цикла ИС описанные уровни моделей соответственно строятся на этапах анализа требований, логического (технического) и физического (рабочего) проектирования. Рассмотрим особенности построения *моделей предметной области* на трех уровнях детализации.

Объектная структура

Объект — это сущность, которая используется при выполнении некоторой *функции* или *операции* (преобразования, обработки, формирования и т.д.). Объекты могут иметь динамическую или статическую природу: динамические объекты используются в одном цикле воспроизводства, например заказы на продукцию, счета на оплату, платежи;

статические объекты используются во многих циклах воспроизводства, например, оборудование, персонал, запасы материалов.

На внешнем уровне детализации модели выделяются основные виды материальных объектов (например, сырье и материалы, полуфабрикаты, готовые изделия, услуги) и основные виды информационных объектов или документов (например, заказы, накладные, счета и т.д.).

На концептуальном уровне построения *модели предметной области* уточняется состав классов объектов, определяются их атрибуты и взаимосвязи. Таким образом строится обобщенное представление структуры предметной области.

Далее концептуальная модель на внутреннем уровне отображается в виде файлов базы данных, входных и выходных документов ЭИС. Причем динамические объекты представляются единицами переменной информации или документами, а статические объекты — единицами условно-постоянной информации в виде списков, номенклатур, ценников, справочников, классификаторов. Модель базы данных как постоянно поддерживаемого информационного ресурса отображает хранение условно-постоянной и накапливаемой переменной информации, используемой в повторяющихся информационных процессах.

Функциональная структура

Функция (*операция*) представляет собой некоторый преобразователь входных объектов в выходные. Последовательность взаимосвязанных по входам и выходам *функций* составляет *бизнес-процесс*. *Функция бизнес-процесса* может порождать объекты любой природы (материальные, денежные, информационные). Причем *бизнес-процессы* и информационные процессы, как правило, неразрывны, то есть *функции* материального процесса не могут осуществляться без информационной поддержки. Например, отгрузка готовой продукции осуществляется на основе документа "Заказ", который, в свою очередь, порождает документ "Накладная", сопровождающий партию отгруженного товара.

Функция может быть представлена одним действием или некоторой совокупностью действий. В последнем случае каждой *функции* может соответствовать некоторый процесс, в котором могут существовать свои *подпроцессы*, и т.д., пока каждая из подфункций не будет представлять некоторую недекомпозируемую последовательность действий.

На внешнем уровне моделирования определяется список основных бизнес-функций или видов *бизнес-процессов*. Обычно таких *функций* насчитывается 15–20.

На концептуальном уровне выделенные *функции* декомпозируются и строятся иерархии взаимосвязанных *функций*.

На внутреннем уровне отображается структура информационного процесса в компьютере: определяются иерархические структуры программных модулей, реализующих автоматизируемые *функции*.

Структура управления

В совокупности *функций бизнес-процесса* возможны альтернативные или циклические последовательности в зависимости от различных условий протекания процесса. Эти условия связаны с происходящими событиями во внешней среде или в самих процессах и с образованием определенных состояний объектов (например, заказ принят, отвергнут, отправлен на корректировку). **События** вызывают выполнение *функций*, которые, в свою очередь, изменяют состояния объектов и формируют новые события, и

т.д., пока не будет завершен некоторый *бизнес-процесс*. Тогда последовательность событий составляет конкретную реализацию *бизнес-процесса*.

Каждое событие описывается с двух точек зрения: **информационной** и **процедурной**. Информационно событие отражается в виде некоторого сообщения, фиксирующего факт выполнения некоторой *функции* изменения состояния или появления нового. Процедурно событие вызывает выполнение новой *функции*, и поэтому для каждого состояния объекта должны быть заданы описания этих вызовов. Таким образом, события выступают в связующей роли для выполнения *функций бизнес-процессов*.

На внешнем уровне определяются список внешних событий, вызываемых взаимодействием предприятия с внешней средой (платежи налогов, процентов по кредитам, поставки по контрактам и т.д.), и список целевых установок, которым должны соответствовать *бизнес-процессы* (регламент выполнения процессов, поддержка уровня материальных запасов, уровень качества продукции и т.д.).

На концептуальном уровне устанавливаются бизнес-правила, определяющие условия вызова *функций* при возникновении событий и достижении состояний объектов.

На внутреннем уровне выполняется формализация бизнес-правил в виде триггеров или вызовов программных модулей.

Организационная структура

Организационная структура представляет собой совокупность организационных единиц, как правило, связанных иерархическими и процессными отношениями. Организационная единица — это подразделение, представляющее собой объединение людей (персонала) для выполнения совокупности общих *функций* или *бизнес-процессов*. В функционально-ориентированной организационной структуре организационная единица выполняет набор *функций*, относящихся к одной *функции* управления и входящих в различные процессы. В процессно-ориентированной структуре организационная единица выполняет набор *функций*, входящих в один тип процесса и относящихся к разным *функциям* управления.

На внешнем уровне строится структурная модель предприятия в виде иерархии подчинения организационных единиц или списков взаимодействующих подразделений.

На концептуальном уровне для каждого подразделения задается организационно-штатная структура должностей (ролей персонала).

На внутреннем уровне определяются требования к правам доступа персонала к автоматизируемым *функциям* информационной системы.

Техническая структура

Топология определяет территориальное размещение технических средств по структурным подразделениям предприятия, а коммуникация — технический способ реализации взаимодействия структурных подразделений.

На внешнем уровне модели определяются типы технических средств обработки данных и их размещение по структурным подразделениям.

На концептуальном уровне определяются способы коммуникаций между техническими комплексами структурных подразделений: физическое перемещение документов, машинных носителей, обмен информацией по каналам связи и т.д.

На внутреннем уровне строится модель "клиент-серверной" архитектуры вычислительной сети.

Описанные *модели предметной области* нацелены на проектирование отдельных компонентов ИС: данных, функциональных программных модулей, управляющих программных модулей, программных модулей интерфейсов пользователей, структуры технического комплекса. Для более качественного проектирования указанных компонентов требуется построение моделей, увязывающих различные компоненты ИС между собой. В простейшем случае в качестве таких моделей взаимодействия могут использоваться матрицы перекрестных ссылок: "объекты-функции", "функции-события", "организационные единицы — функции", "организационные единицы — объекты", "организационные единицы — технические средства" и т.д. Такие матрицы не наглядны и не отражают особенности реализации взаимодействий.

Для правильного отображения взаимодействий компонентов ИС важно осуществлять совместное моделирование таких компонентов, особенно с содержательной точки зрения объектов и *функций*. Методология структурного системного анализа существенно помогает в решении таких задач.

Структурным анализом принято называть метод исследования системы, который начинается с ее общего обзора, а затем детализируется, приобретая иерархическую структуру с все большим числом уровней. Для таких методов характерно: разбиение на уровни абстракции с ограниченным числом элементов (от 3 до 7); ограниченный контекст, включающий только существенные детали каждого уровня; использование строгих формальных правил записи; последовательное приближение к результату. *Структурный анализ* основан на двух базовых принципах – "разделяй и властвуй" и принципе иерархической упорядоченности. Решение трудных проблем путем их разбиения на множество меньших независимых задач (так называемых "черных ящиков") и организация этих задач в древовидные иерархические структуры значительно повышают понимание сложных систем. Определим ключевые понятия *структурного анализа*.

Операция – элементарное (неделимое) действие, выполняемое на одном рабочем месте.

Функция – совокупность *операций*, сгруппированных по определенному признаку.

Бизнес-процесс — связанная совокупность *функций*, в ходе выполнения которой потребляются определенные ресурсы и создается продукт (предмет, услуга, научное открытие, идея), представляющая ценность для потребителя.

Подпроцесс – это *бизнес-процесс*, являющийся структурным элементом некоторого *бизнес-процесса* и представляющий ценность для потребителя.

Бизнес-модель – структурированное графическое описание сети процессов и *операций*, связанных с данными, документами, организационными единицами и прочими объектами, отражающими существующую или предполагаемую деятельность предприятия.

Существуют различные методологии структурного моделирования предметной области, среди которых следует выделить **функционально-ориентированные и объектно-ориентированные методологии**.

Функционально-ориентированные и объектно-ориентированные методологии описания предметной области

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделируемая организация. В соответствии с различными представлениями об организации методики принято делить на объектные и функциональные (структурные).

Объектные методики рассматривают моделируемую организацию как набор взаимодействующих объектов – производственных единиц. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия.

Функциональные методики, наиболее известной из которых является методика IDEF, рассматривают организацию как набор *функций*, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от *объектной методики* заключается в четком отделении *функций* (методов обработки данных) от самих данных.

С точки зрения бизнес-моделирования каждый из представленных подходов обладает своими преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует существующим структурам организации. Функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых *функций* интуитивно лучше понимается исполнителями при получении от них информации об их текущей работе.

Функциональная методика IDEF0

Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique). Исторически IDEF0 как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing). Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=Icam DEFinition), и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST).

Целью методики является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.

В основе методологии лежат четыре основных понятия: **функциональный блок, интерфейсная дуга, декомпозиция, глоссарий**.

Функциональный блок (Activity Box) представляет собой некоторую конкретную *функцию* в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, "производить услуги"). На диаграмме функциональный блок изображается прямоугольником ([рис. 6.1](#)). Каждая из четырех сторон функционального блока имеет свое определенное значение (роль), при этом:

- верхняя сторона имеет значение "Управление" (Control);
- левая сторона имеет значение "Вход" (Input);
- правая сторона имеет значение "Выход" (Output);
- нижняя сторона имеет значение "Механизм" (Mechanism).



Рис. 6.1. Функциональный блок

Интерфейсная дуга (Arrow) отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на *функцию*, представленную данным функциональным блоком. Интерфейсные дуги часто называют потоками или стрелками.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.).

В зависимости от того, к какой из сторон функционального блока подходит данная интерфейсная дуга, она носит название "входящей", "исходящей" или "управляющей".

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь, по крайней мере, одну управляющую интерфейсную дугу и одну исходящую. Это и понятно – каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Декомпозиция (Decomposition) является основным понятием стандарта IDEF0. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его *функции*. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Последним из понятий IDEF0 является **гlossарий (Glossary)**. Для каждого из элементов IDEF0 — диаграмм, функциональных блоков, интерфейсных дуг — существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Этот набор называется гlossарием и является описанием сущности данного элемента. Гlossарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Модель IDEF0 всегда начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется **контекстной диаграммой**.

В пояснительном тексте к контекстной диаграмме должна быть указана **цель** (Purpose) построения диаграммы в виде краткого описания и зафиксирована **точка зрения** (Viewpoint).

Определение и формализация цели разработки IDEF0-модели является крайне важным моментом. Фактически цель определяет соответствующие области в исследуемой системе, на которых необходимо фокусироваться в первую очередь.

Точка зрения определяет основное направление развития модели и уровень необходимой детализации. Четкое фиксирование точки зрения позволяет разгрузить модель, отказавшись от детализации и исследования отдельных элементов, не являющихся необходимыми, исходя из выбранной точки зрения на систему. Правильный выбор точки зрения существенно сокращает временные затраты на построение конечной модели.

Выделение подпроцессов. В процессе декомпозиции функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы, и называется дочерней (Child Diagram) по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется дочерним блоком – Child Box). В свою очередь, функциональный блок — предок называется родительским блоком по отношению к дочерней диаграмме (Parent Box), а диаграмма, к которой он принадлежит – родительской диаграммой (Parent Diagram). Каждая из подфункций дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции соответствующего ей функционального блока. В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0-модели.

Иногда отдельные интерфейсные дуги высшего уровня не имеет смысла продолжать рассматривать на диаграммах нижнего уровня, или наоборот — отдельные дуги нижнего уровня отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие туннелирования. Обозначение "туннеля" (Arrow Tunnel) в виде двух круглых скобок вокруг начала интерфейсной дуги обозначает, что эта дуга не была унаследована от функционального родительского блока и появилась (из "туннеля") только на этой диаграмме. В свою очередь, такое же обозначение вокруг конца (стрелки) интерфейсной дуги в непосредственной близости от блока-приемника означает тот факт, что в дочерней по отношению к этому блоку диаграмме эта дуга отображаться и рассматриваться не будет. Чаще всего бывает, что отдельные объекты и соответствующие им интерфейсные дуги не рассматриваются на некоторых промежуточных уровнях иерархии, – в таком случае они сначала "погружаются в туннель", а затем при необходимости "возвращаются из туннеля".

Обычно IDEF0-модели несут в себе сложную и концентрированную информацию, и для того, чтобы ограничить их перегруженность и сделать удобочитаемыми, в стандарте приняты соответствующие ограничения сложности.

Рекомендуется представлять на диаграмме от трех до шести функциональных блоков, при этом количество подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг предполагается не более четырех.

Стандарт IDEF0 содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой группой людей, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов:

- Создание модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах IDEF0 называется авторами (Authors). Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов, создавая модели деятельности подразделений. При этом их интересуют ответы на следующие вопросы:

Что поступает в подразделение "на входе"?

- Какие *функции* и в какой последовательности выполняются в рамках подразделения?
- Кто является ответственным за выполнение каждой из *функций*?
- Чем руководствуется исполнитель при выполнении каждой из *функций*?
- Что является результатом работы подразделения (на выходе)?

На основе имеющихся положений, документов и результатов опросов создается черновик (Model Draft) модели.

- Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким кругом компетентных лиц (в терминах IDEF0 — читателей) на предприятии. При этом каждая из диаграмм черновой модели письменно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает ее с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.
- Официальное утверждение модели. Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у авторов модели и читателей отсутствуют разногласия по поводу ее адекватности. Окончательная модель представляет собой согласованное представление о предприятии (системе) с заданной точки зрения и для заданной цели.

Наглядность графического языка IDEF0 делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений в модели.

Функциональная методика потоков данных

Целью методики является построение модели рассматриваемой системы в виде диаграммы потоков данных (Data Flow Diagram — DFD), обеспечивающей правильное описание выходов (отклика системы в виде данных) при заданном воздействии на вход системы (подаче сигналов через внешние интерфейсы). Диаграммы потоков данных являются основным средством моделирования функциональных требований к проектируемой системе.

При создании диаграммы потоков данных используются четыре основных понятия: **потоки данных, процессы (работы) преобразования входных потоков данных в выходные, внешние сущности, накопители данных (хранилища).**

Потоки данных являются абстракциями, используемыми для моделирования передачи информации (или физических компонент) из одной части системы в другую. Потоки на диаграммах изображаются именованными стрелками, ориентация которых указывает направление движения информации.

Назначение **процесса** (работы) состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым именем процесса. Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, "получить документы по отгрузке продукции"). Каждый процесс имеет уникальный номер для ссылок на него внутри диаграммы, который может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Хранилище (накопитель) данных позволяет на указанных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет "срезы" потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее получения, при этом данные могут выбираться в любом порядке. Имя хранилища должно определять его содержимое и быть существительным.

Внешняя сущность представляет собой материальный объект вне контекста системы, являющейся источником или приемником системных данных. Ее имя должно содержать существительное, например, "склад товаров". Предполагается, что объекты, представленные как внешние сущности, не должны участвовать ни в какой обработке.

Кроме основных элементов, в состав DFD входят словари данных и миниспецификации.

Словари данных являются каталогами всех элементов данных, присутствующих в DFD, включая групповые и индивидуальные потоки данных, хранилища и процессы, а также все их атрибуты.

Миниспецификации обработки — описывают DFD-процессы нижнего уровня. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы.

Процесс построения DFD начинается с создания так называемой основной диаграммы типа "звезда", на которой представлен моделируемый процесс и все внешние сущности, с которыми он взаимодействует. В случае сложного основного процесса он сразу представляется в виде декомпозиции на ряд взаимодействующих процессов. Критериями сложности в данном случае являются: наличие большого числа внешних сущностей, многофункциональность системы, ее распределенный характер. Внешние сущности выделяются по отношению к основному процессу. Для их определения необходимо выделить поставщиков и потребителей основного процесса, т.е. все объекты, которые взаимодействуют с основным процессом. На этом этапе описание взаимодействия заключается в выборе глагола, дающего представление о том, как внешняя сущность использует основной процесс или используется им. Например, основной процесс – "учет обращений граждан", внешняя сущность – "граждане", описание взаимодействия – "подает заявления и получает ответы". Этот этап является принципиально важным, поскольку именно он определяет границы моделируемой системы.

Для всех внешних сущностей строится таблица событий, описывающая их взаимодействие с основным потоком. Таблица событий включает в себя наименование внешней сущности, событие, его тип (типичный для системы или исключительный, реализующийся при определенных условиях) и реакцию системы.

На следующем шаге происходит декомпозиция основного процесса на набор взаимосвязанных процессов, обменивающихся потоками данных. Сами потоки не конкретизируются, определяется лишь характер взаимодействия. Декомпозиция завершается, когда процесс становится простым, т.е.:

1. процесс имеет два-три входных и выходных потока;

2. процесс может быть описан в виде преобразования входных данных в выходные;
3. процесс может быть описан в виде последовательного алгоритма.

Для простых процессов строится миниспецификация – формальное описание алгоритма преобразования входных данных в выходные.

Миниспецификация удовлетворяет следующим требованиям: для каждого процесса строится одна спецификация; спецификация однозначно определяет входные и выходные потоки для данного процесса; спецификация не определяет способ преобразования входных потоков в выходные; спецификация ссылается на имеющиеся элементы, не вводя новые; спецификация по возможности использует стандартные подходы и *операции*.

После декомпозиции основного процесса для каждого *подпроцесса* строится аналогичная таблица внутренних событий.

Следующим шагом после определения полной таблицы событий выделяются **потоки данных**, которыми обмениваются процессы и внешние сущности. Простейший способ их выделения заключается в анализе таблиц событий. События преобразуются в потоки данных от инициатора события к запрашиваемому процессу, а реакции – в обратный поток событий. После построения входных и выходных потоков аналогичным образом строятся внутренние потоки. Для их выделения для каждого из внутренних процессов выделяются поставщики и потребители информации. Если поставщик или потребитель информации представляет процесс сохранения или запроса информации, то вводится хранилище данных, для которого данный процесс является интерфейсом.

После построения потоков данных диаграмма должна быть проверена на полноту и непротиворечивость. Полнота диаграммы обеспечивается, если в системе нет "повисших" процессов, не используемых в процессе преобразования входных потоков в выходные. Непротиворечивость системы обеспечивается выполнением наборов формальных правил о возможных типах процессов: на диаграмме не может быть потока, связывающего две внешние сущности – это взаимодействие удаляется из рассмотрения; ни одна сущность не может непосредственно получать или отдавать информацию в хранилище данных – хранилище данных является пассивным элементом, управляемым с помощью интерфейсного процесса; два хранилища данных не могут непосредственно обмениваться информацией – эти хранилища должны быть объединены.

К преимуществам методики DFD относятся:

- возможность однозначно определить внешние сущности, анализируя потоки информации внутри и вне системы;
- возможность проектирования сверху вниз, что облегчает построение модели "как должно быть";
- наличие спецификаций процессов нижнего уровня, что позволяет преодолеть логическую незавершенность функциональной модели и построить полную функциональную спецификацию разрабатываемой системы.

К недостаткам модели отнесем: необходимость искусственного ввода управляющих процессов, поскольку управляющие воздействия (потоки) и управляющие процессы с точки зрения DFD ничем не отличаются от обычных; отсутствие понятия времени, т.е. отсутствие анализа временных промежутков при преобразовании данных (все ограничения по времени должны быть введены в спецификациях процессов).

Объектно-ориентированная методика

Принципиальное отличие между функциональным и объектным подходом заключается в способе декомпозиции системы. Объектно-ориентированный подход использует

объектную декомпозицию, при этом статическая структура описывается в терминах **объектов и связей** между ними, а поведение системы описывается в терминах **обмена сообщениями** между объектами. Целью методики является построение *бизнес-модели* организации, позволяющей перейти от модели сценариев использования к модели, определяющей отдельные объекты, участвующие в реализации бизнес-функций.

Концептуальной основой объектно-ориентированного подхода является объектная модель, которая строится с учетом следующих принципов:

- абстрагирование;
- инкапсуляция;
- модульность;
- иерархия;
- типизация;
- параллелизм;
- устойчивость.

Основными понятиями объектно-ориентированного подхода являются объект и класс.

Объект — предмет или явление, имеющее четко определенное поведение и обладающие состоянием, поведением и индивидуальностью. Структура и поведение схожих объектов определяют общий для них класс. **Класс – это множество объектов, связанных общностью структуры и поведения.** Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм. Понятие **полиморфизм** может быть интерпретировано как способность класса принадлежать более чем одному типу. **Наследование** означает построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой информационной системы от стадии формирования требований до стадии реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

Большинство существующих методов объектно-ориентированного подхода включают *язык моделирования* и описание процесса моделирования. **Процесс** – это описание шагов, которые необходимо выполнить при разработке проекта. В качестве *языка моделирования* объектного подхода используется унифицированный *язык моделирования* UML, который содержит стандартный набор диаграмм для моделирования.

Диаграмма (Diagram) — это графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями) и представляет собой некоторую проекцию системы.

Объектно-ориентированный подход обладает следующими преимуществами:

- Объектная декомпозиция дает возможность создавать модели меньшего размера путем использования общих механизмов, обеспечивающих необходимую экономию выразительных средств. Использование объектного подхода существенно повышает уровень унификации разработки и пригодность для повторного использования, что ведет к созданию среды разработки и переходу к сборочному созданию моделей.
- Объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем.

- Объектная модель естественна, поскольку ориентированна на человеческое восприятие мира.

К недостаткам объектно-ориентированного подхода относятся высокие начальные затраты. Этот подход не дает немедленной отдачи. Эффект от его применения сказывается после разработки двух–трех проектов и накопления повторно используемых компонентов. Диаграммы, отражающие специфику объектного подхода, менее наглядны.

Сравнение существующих методик

В **функциональных моделях** (DFD-диаграммах потоков данных, SADT-диаграммах) главными структурными компонентами являются *функции* (*операции*, действия, работы), которые на диаграммах связываются между собой потоками объектов.

Несомненным достоинством функциональных моделей является реализация структурного подхода к проектированию ИС по принципу "сверху-вниз", когда каждый функциональный блок может быть декомпозирован на множество подфункций и т.д., выполняя, таким образом, модульное проектирование ИС. Для функциональных моделей характерны процедурная строгость декомпозиции ИС и наглядность представления.

При функциональном подходе объектные модели данных в виде ER-диаграмм "объект — свойство — связь" разрабатываются отдельно. Для проверки корректности моделирования предметной области между функциональными и объектными моделями устанавливаются взаимно однозначные связи.

Главный недостаток функциональных моделей заключается в том, что процессы и данные существуют отдельно друг от друга — помимо функциональной декомпозиции существует структура данных, находящаяся на втором плане. Кроме того, не ясны условия выполнения процессов обработки информации, которые динамически могут изменяться.

Перечисленные недостатки функциональных моделей снимаются в **объектно-ориентированных моделях**, в которых главным структурообразующим компонентом выступает класс объектов с набором *функций*, которые могут обращаться к атрибутам этого класса.

Для классов объектов характерна иерархия обобщения, позволяющая осуществлять **наследование** не только атрибутов (свойств) объектов от вышестоящего класса объектов к нижестоящему классу, но и *функций* (методов).

В случае наследования *функций* можно абстрагироваться от конкретной реализации процедур (**абстрактные типы данных**), которые отличаются для определенных подклассов ситуаций. Это дает возможность обращаться к подобным программным модулям по общим именам (**полиморфизм**) и осуществлять повторное использование программного кода при модификации программного обеспечения. Таким образом, адаптивность объектно-ориентированных систем к изменению предметной области по сравнению с функциональным подходом значительно выше.

При объектно-ориентированном подходе изменяется и принцип проектирования ИС. Сначала выделяются классы объектов, а далее в зависимости от возможных состояний объектов (жизненного цикла объектов) определяются методы обработки (функциональные процедуры), что обеспечивает наилучшую реализацию динамического поведения информационной системы.

Для объектно-ориентированного подхода разработаны графические методы моделирования предметной области, обобщенные в языке унифицированного

моделирования UML. Однако по наглядности представления модели пользователю-заказчику объектно-ориентированные модели явно уступают функциональным моделям.

При выборе методики моделирования предметной области обычно в качестве критерия выступает степень ее динамичности. Для более регламентированных задач больше подходят функциональные модели, для более адаптивных *бизнес-процессов* (управления рабочими потоками, реализации динамических запросов к информационным хранилищам) — объектно-ориентированные модели. Однако в рамках одной и той же ИС для различных классов задач могут требоваться различные виды моделей, описывающих одну и ту же проблемную область. В таком случае должны использоваться комбинированные *модели предметной области*.

Синтетическая методика

Как можно видеть из представленного обзора, каждая из рассмотренных методик позволяет решить задачу построения формального описания рабочих процедур исследуемой системы. Все методики позволяют построить модель "как есть" и "как должно быть". С другой стороны, каждая из этих методик обладает существенными недостатками. Их можно суммировать следующим образом: недостатки применения отдельной методики лежат не в области описания реальных процессов, а в неполноте методического подхода.

Функциональные методики в целом лучше дают представление о существующих *функциях* в организации, о методах их реализации, причем чем выше степень детализации исследуемого процесса, тем лучше они позволяют описать систему. Под лучшим описанием в данном случае понимается наименьшая ошибка при попытке по полученной модели предсказать поведение реальной системы. На уровне отдельных рабочих процедур их описание практически однозначно совпадает с фактической реализацией в потоке работ.

На уровне общего описания системы *функциональные методики* допускают значительную степень произвола в выборе общих интерфейсов системы, ее механизмов и т.д., то есть в определении границ системы. Хорошо описать систему на этом уровне позволяет объектный подход, основанный на понятии сценария использования. Ключевым является понятие о сценарии использования как о сеансе взаимодействия действующего лица с системой, в результате которого действующее лицо получает нечто, имеющее для него ценность. Использование критерия ценности для пользователя дает возможность отбросить не имеющие значения детали потоков работ и сосредоточиться на тех *функциях* системы, которые оправдывают ее существование. Однако и в этом случае задача определения границ системы, выделения внешних пользователей является сложной.

Технология потоков данных, исторически возникшая первой, легко решает проблему границ системы, поскольку позволяет за счет анализа информационных потоков выделить внешние сущности и определить основной внутренний процесс. Однако отсутствие выделенных управляющих процессов, потоков и событийной ориентированности не позволяет предложить эту методику в качестве единственной.

Наилучшим способом преодоления недостатков рассмотренных методик является формирование **синтетической методики**, объединяющей различные этапы отдельных методик. При этом из каждой методики необходимо взять часть методологии, наиболее полно и формально изложенную, и обеспечить возможность обмена результатами на различных этапах применения синергетической методики. В бизнес-моделировании неявным образом идет формирование подобной синергетической методики.

Идея **синтетической методики** заключается в последовательном применении функционального и объектного подхода с учетом возможности реинжиниринга существующей ситуации.

Рассмотрим применение синтетической методики на примере разработки административного регламента.

При построении административных регламентов выделяются следующие стадии:

1. Определение границ системы. На этой стадии при помощи **анализа потоков данных выделяют внешние сущности** и собственно моделируемую систему.
2. Выделение сценариев использования системы. На этой стадии **при помощи критерия полезности строят** для каждой внешней сущности **набор сценариев использования системы**.
3. Добавление системных сценариев использования. На этой стадии **определяют сценарии, необходимые для реализации целей системы**, отличных от целей пользователей.
4. Построение диаграммы активностей по сценариям использования. На этой стадии строят **набор действий системы**, приводящих к реализации сценариев использования;
5. Функциональная **декомпозиция диаграмм активностей** как контекстных диаграмм методики IDEF0.
6. Формальное описание отдельных функциональных активностей в виде административного регламента (с применением различных *нотаций*).

Лекция: Моделирование бизнес-процессов средствами BPwin

Моделирование деловых процессов, как правило, выполняется с помощью case-средств. К таким средствам относятся BPwin (PLATINUM technology), Silverrun (Silverrun technology), Oracle Designer (Oracle), Rational Rose (Rational Software) и др. Функциональные возможности инструментальных средств структурного моделирования деловых процессов будут рассмотрены на примере case-средства BPwin.

BPwin поддерживает три методологии моделирования: функциональное моделирование (IDEF0); описание бизнес-процессов (IDEF3); диаграммы потоков данных (DFD).

Инструментальная среда BPwin

BPwin имеет достаточно простой и интуитивно понятный интерфейс пользователя. При запуске BPwin по умолчанию появляется основная панель инструментов, палитра инструментов (вид которой зависит от выбранной нотации) и, в левой части, навигатор модели — Model Explorer (рис. 7.1).

При создании новой модели возникает диалог, в котором следует указать, будет ли создана модель заново или она будет открыта из файла либо из репозитория ModelMart, затем внести имя модели и выбрать методологию, в которой будет построена модель (рис. 7.2).

Как было указано выше, BPwin поддерживает три методологии — IDEF0, IDEF3 и DFD, каждая из которых решает свои специфические задачи. В BPwin возможно построение смешанных моделей, т. е. модель может содержать одновременно диаграммы как IDEF0, так и IDEF3 и DFD. Состав палитры инструментов изменяется автоматически, когда происходит переключение с одной нотации на другую.

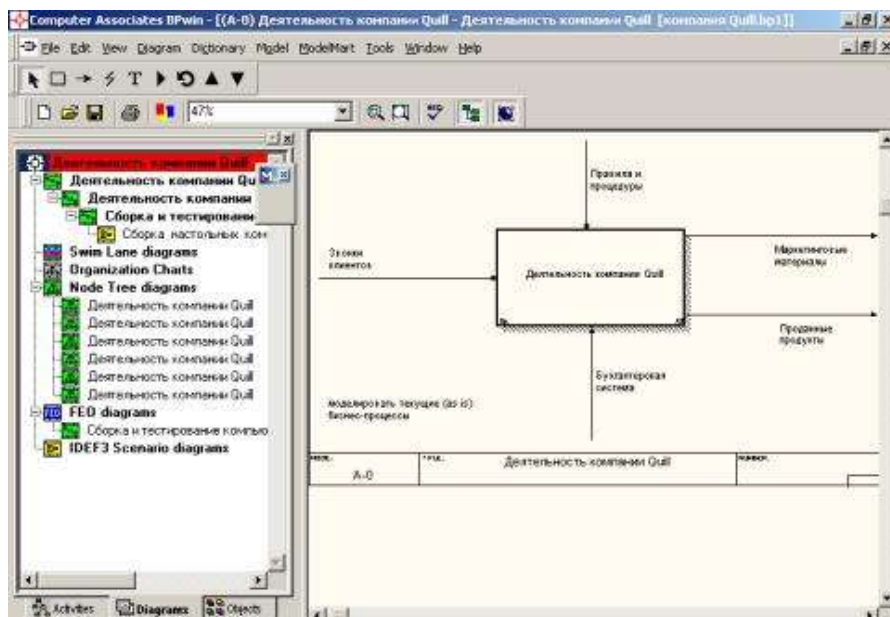


Рис. 7.1. Интегрированная среда разработки модели BPwin

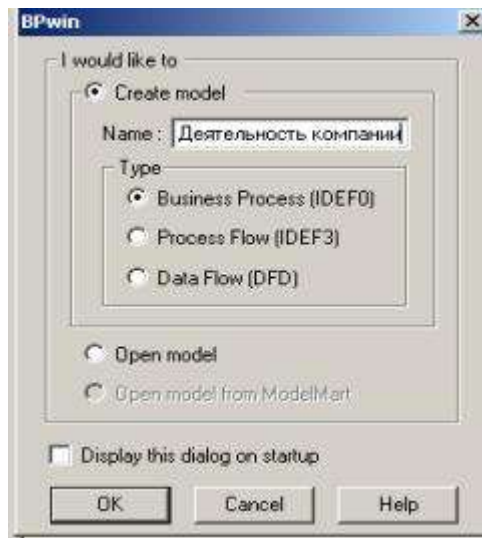


Рис. 7.2. Диалог создания модели

Модель в BPwin рассматривается как совокупность *работ*, каждая из которых оперирует с некоторым набором данных. *Работа* изображается в виде прямоугольников, данные — в виде стрелок. Если щелкнуть по любому объекту модели левой кнопкой мыши, появляется контекстное меню, каждый пункт которого соответствует редактору какого-либо свойства объекта.

Построение модели IDEF0

На начальных этапах создания ИС необходимо понять, как работает организация, которую собираются автоматизировать. 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 может не знать, как работают коллеги. Поэтому для описания *работы* предприятия необходимо построить модель, которая будет адекватна предметной области и содержать в себе знания всех участников бизнес-процессов организации.

Наиболее удобным языком моделирования бизнес-процессов является IDEF0, где 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.

Процесс моделирования системы в IDEF0 начинается с создания *контекстной диаграммы* — диаграммы наиболее абстрактного уровня описания системы в целом, содержащей 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, определить, что будет в дальнейшем рассматриваться как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будут существенно влиять позиция, с которой рассматривается система, и цель моделирования — вопросы, на которые построенная модель должна дать ответ. Другими словами, в начале необходимо определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в ходе моделирования область

может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования. При формулировании области необходимо учитывать два компонента — широту и глубину. Широта подразумевает определение границ модели — что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком уровне детализации модель является завершённой. При определении глубины системы необходимо помнить об ограничениях времени — трудоёмкость построения модели растёт в геометрической прогрессии с увеличением глубины декомпозиции. После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему.

Цель моделирования

Цель моделирования определяется из ответов на следующие вопросы:

- Почему этот процесс должен быть смоделирован?
- Что должна показывать модель?
- Что может получить клиент?

Точка зрения (Viewpoint).

Под точкой зрения понимается перспектива, с которой наблюдалась система при построении модели. Хотя при построении модели учитываются мнения различных людей, все они должны придерживаться единой точки зрения на модель. Точка зрения должна соответствовать цели и границам моделирования. Как правило, выбирается точка зрения человека, ответственного за моделируемую *работу* в целом.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в BPwin следует выбрать пункт меню Model/Model Properties, вызывающий диалог Model Properties ([рис. 7.3](#)). В закладке Purpose следует внести цель и точку зрения, а в закладку Definition — определение модели и описание области.

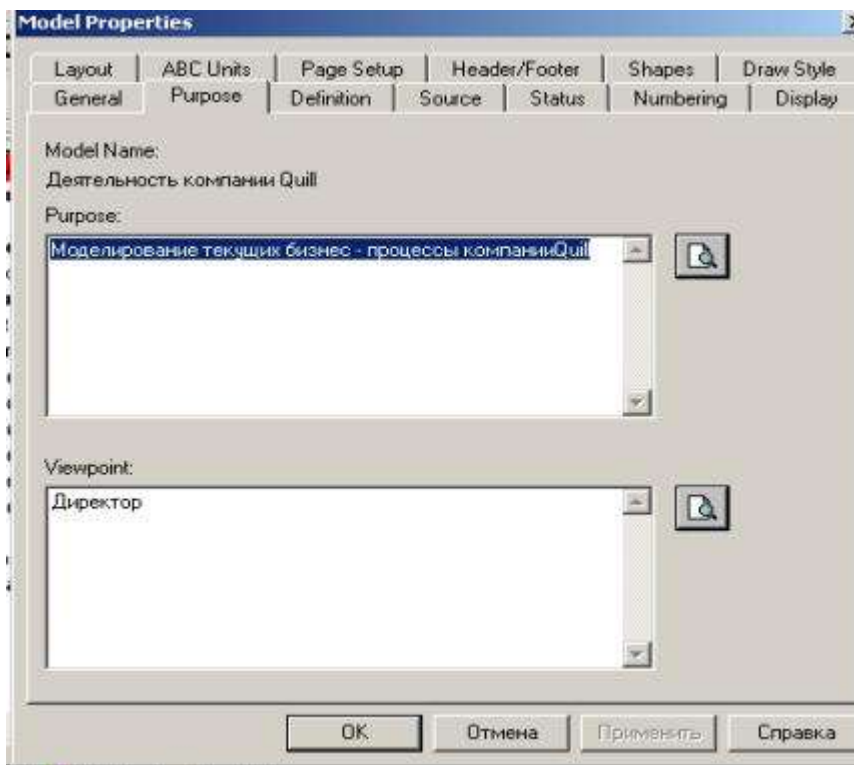


Рис. 7.3. Диалог задания свойств модели

В закладке Status того же диалога можно описать статус модели (черновой вариант, рабочий, окончательный и т. д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате). В закладке Source описываются источники информации для построения модели (например, "Опрос экспертов предметной области и анализ документации"). Закладка General служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок модели — AS-IS и TO-BE.

Модели AS-IS и TO-BE. Обычно сначала строится модель существующей организации *работы* — AS-IS (как есть). Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес-процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса. Детализация бизнес-процессов позволяет выявить недостатки организации даже там, где функциональность на первый взгляд кажется очевидной. Найденные в модели AS-IS недостатки можно исправить при создании модели TO-BE (как будет) — модели новой организации бизнес-процессов.

Технология проектирования ИС подразумевает сначала создание модели AS-IS, ее анализ и улучшение бизнес-процессов, то есть создание модели TO-BE, и только на основе модели TO-BE строится модель данных, прототип и затем окончательный вариант ИС.

Иногда текущая AS-IS и будущая TO-BE модели различаются очень сильно, так что переход от начального к конечному состоянию становится неочевидным. В этом случае необходима третья модель, описывающая процесс перехода от начального к конечному состоянию системы, поскольку такой переход — это тоже бизнес-процесс.

Результат описания модели можно получить в отчете Model Report. Диалог настройки отчета по модели вызывается из пункта меню Tools/Reports/Model Report.

В диалоге настройки следует выбрать необходимые поля, при этом автоматически отображается очередность вывода информации в отчет ([рис. 7.4](#)).

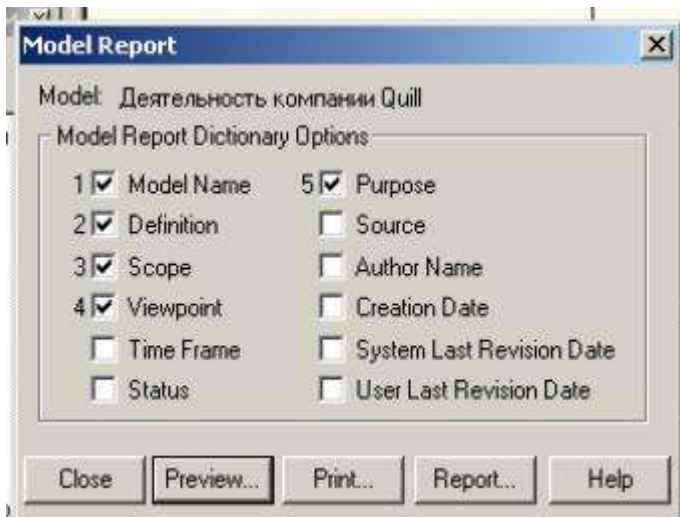


Рис. 7.4. Диалоговое окно для формирования отчета по модели

На [рис. 7.5](#) представлен отчет, сформированный по вышеуказанным полям.

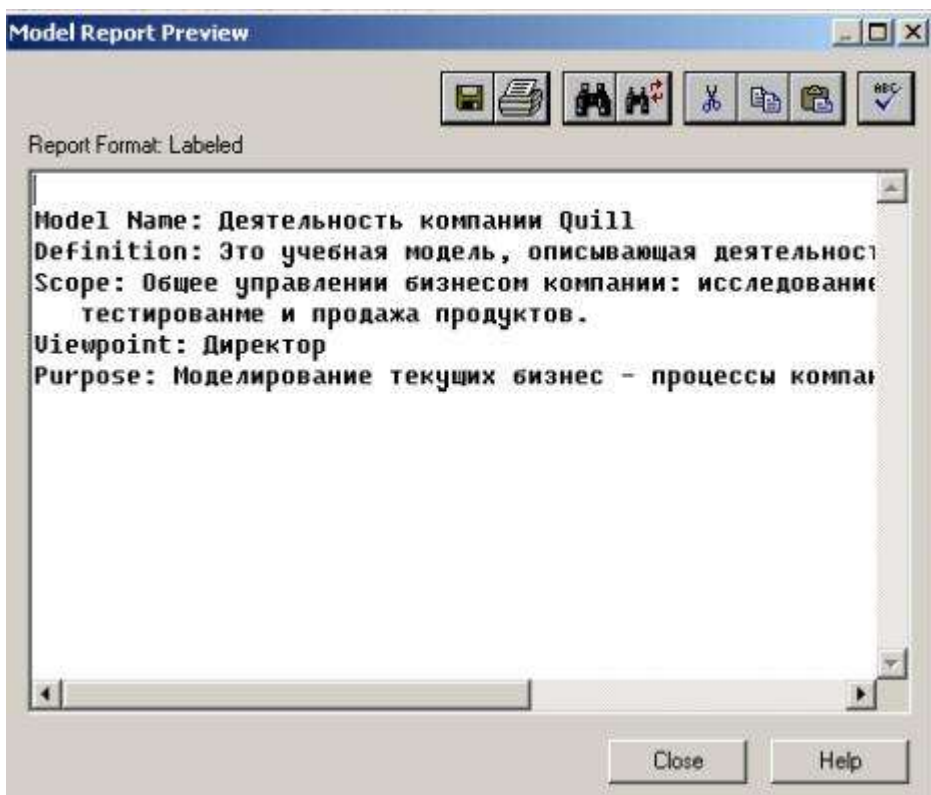


Рис. 7.5. Предварительный просмотр отчета

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:

- *контекстную диаграмму* (в каждой модели может быть только одна *контекстная диаграмма*);
- диаграммы декомпозиции;
- *диаграммы дерева узлов*;
- *диаграммы только для экспозиции (FEO)*.

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. 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.

Диаграмма дерева узлов показывает иерархическую зависимость *работ*, но не взаимосвязи между *работами*. *Диаграмм деревьев узлов* может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения, либо для специальных целей.

Работы (Activity) 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, "Деятельность компании", "Прием заказа" и т.д.). *Работа* "Деятельность компании" может иметь, например, следующее определение: "Это учебная модель, описывающая деятельность компании". При создании новой модели (меню File/New) автоматически создается *контекстная диаграмма* с единственной *работой*, изображающей систему в целом ([рис. 7.6](#)).



Рис. 7.6. Пример контекстной диаграммы

Для внесения имени *работы* следует щелкнуть по *работе* правой кнопкой мыши, выбрать в меню Name Editor и в появившемся диалоге внести имя *работы*. Для описания других свойств *работы* служит диалог Activity Properties ([рис. 7.7](#)).

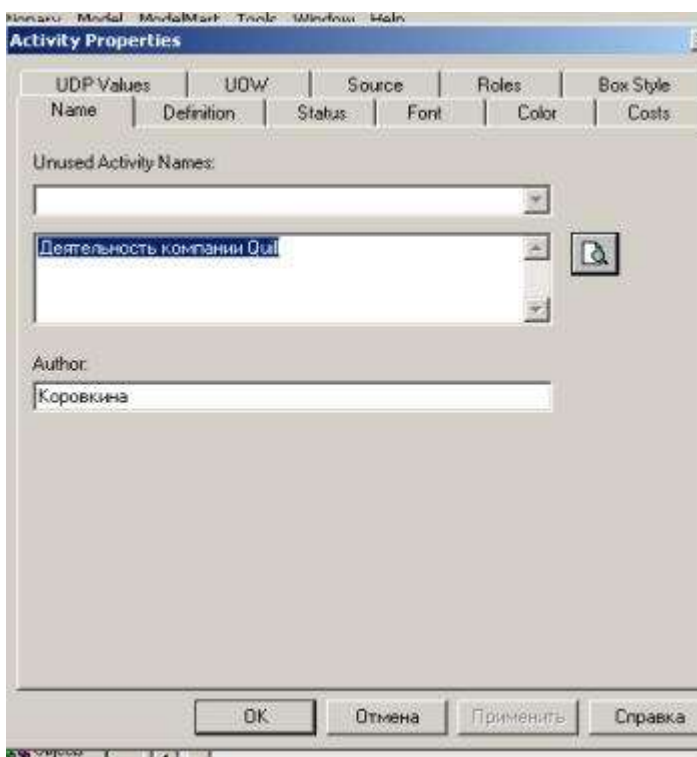


Рис. 7.7. Редактор задания свойств работы

Диаграммы декомпозиции содержат родственные *работы*, т. е. дочерние *работы*, имеющие общую родительскую *работу*. Для создания диаграммы декомпозиции следует щелкнуть по кнопке

на панели инструментов.

Возникает диалог Activity Box Count (рис. 7.8), в котором следует указать нотацию новой диаграммы и количество *работ* на ней. Остановимся пока на нотации IDEF0 и щелкнем на OK. Появляется диаграмма декомпозиции (рис. 7.9). Допустимый интервал числа *работ* — 2-8. 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.



Рис. 7.8. Диалог Activity Box Count

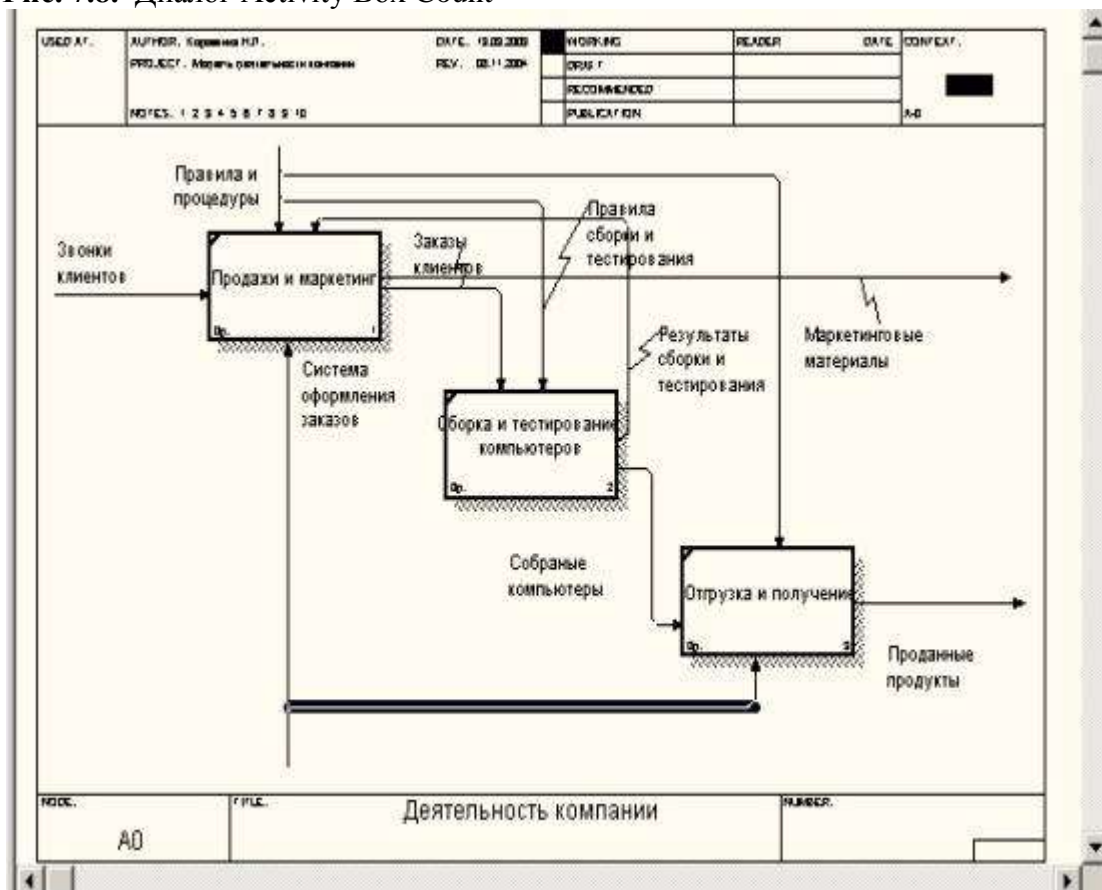


Рис. 7.9. Пример диаграммы декомпозиции

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 помещается самая важная *работа* или *работа*, выполняемая по времени первой. Далее вправо вниз располагаются менее важные или выполняемые позже *работы*. Такое размещение облегчает чтение диаграмм, кроме того, на нем основывается понятие взаимосвязей *работ* (см. ниже).

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. Так, на [рис. 7.9](#) все *работы* еще не были декомпозированы.

Стрелки(Arrow) описывают взаимодействие *работ* и представляют собой некую информацию, выраженную существительными.(Например, "Звонки клиентов", "Правила и процедуры", "Бухгалтерская система".)

В IDEF0 различают пять типов стрелок:

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, "Звонки клиентов" на [рис. 7.6](#) — это нечто, что перерабатывается в процессе "Деятельность компании" 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 для того, чтобы оправдать свое назначение, *стрелки* входа и выхода должны быть точно определены с тем, чтобы указать на то, что данные действительно были переработаны (например, на выходе — "Заполненная карта пациента"). Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить информация о том, перерабатываются/изменяются ли данные в *работе* или нет. Если изменяются, то, скорее всего, это вход, если нет — управление.

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 [рис. 7.6](#) *стрелка* "Правила и процедуры" — управление для *работы* "Деятельность компании". 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 [рис. 7.6](#) *стрелки* "Маркетинговые материалы" и "Проданные продукты" являются выходом для *работы* "Деятельность компании".

Механизм(Mechanism) — ресурсы, которые выполняют *работу*, например персонал предприятия, станки, устройства и т. д. *Стрелка* механизма рисуется как входящая в нижнюю грань *работы*. На [рис. 7.6](#) *стрелка* "Бухгалтерская система" является механизмом для *работы* "Деятельность компании". 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*. На [рис. 7.10](#) *стрелка* "Другая модель работы" является вызовом для *работы* "Изготовление изделия". *Стрелка* вызова используется для указания того, что некоторая *работа* выполняется за пределами моделируемой системы. В BPwin *стрелки* вызова используются в механизме слияния и разделения моделей.

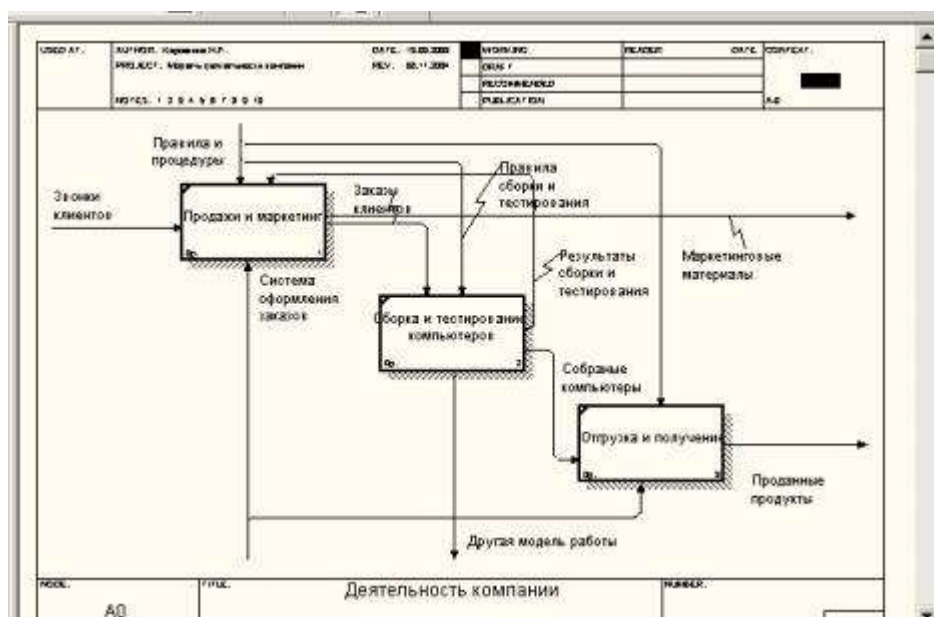




Рис. 7.10. Стрелка вызова, появляющаяся при расщеплении модели

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.

Для внесения граничной *стрелки* входа следует:

- щелкнуть по кнопке с символом *стрелки* ;
- в палитре инструментов перенести курсор к левой стороне экрана, пока не появится начальная штриховая полоска;
- щелкнуть один раз по полоске (откуда выходит *стрелка*) и еще раз в левой части *работы* со стороны входа (где заканчивается *стрелка*);
- вернуться в палитру инструментов и выбрать опцию редактирования *стрелки* .

- щелкнуть правой кнопкой мыши на линии *стрелки*, во всплывающем меню выбрать Name и добавить имя *стрелки* в закладке Name диалога IDEF0 Arrow Properties.

Стрелки управления, входа, механизма и выхода изображаются аналогично. Имена вновь внесенных стрелок (рис. 7.11) автоматически заносятся в словарь Arrow Dictionary.

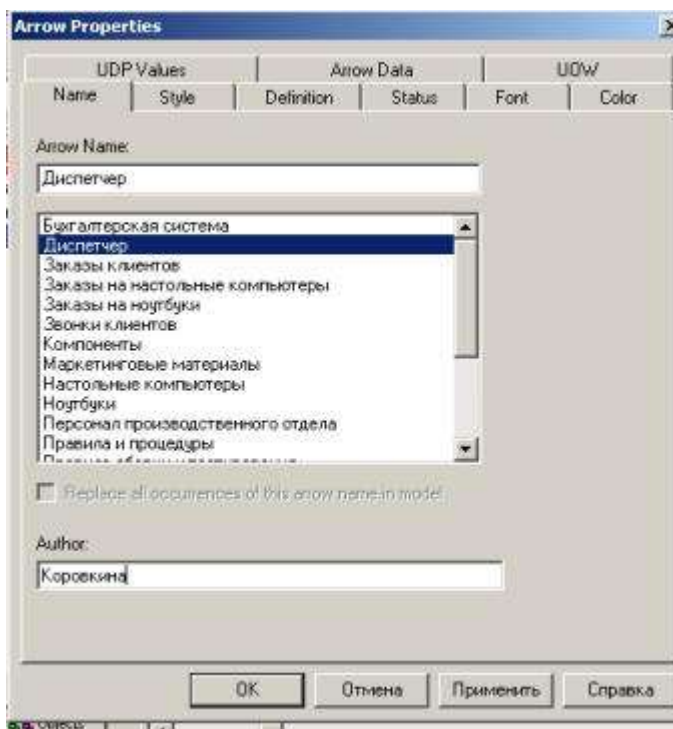


Рис. 7.11. Диалог IDEF0 Arrow Properties

ICOM-коды. Диаграмма декомпозиции предназначена для детализации *работы*. В отличие от моделей, отображающих структуру организации, *работа* на диаграмме верхнего уровня в IDEF0 — это не элемент управления нижестоящими *работами*. *Работы* нижнего уровня — это то же самое, что *работы* верхнего уровня, но в более детальном изложении. Как следствие этого границы *работы* верхнего уровня — это то же самое, что границы диаграммы декомпозиции. ICOM (аббревиатура от Input, Control, Output и Mechanism) — коды, предназначенные для идентификации граничных стрелок. Код ICOM содержит префикс, соответствующий типу *стрелки* (I, C, O или M), и порядковый номер.

BPwin вносит ICOM-коды автоматически. Для отображения ICOM-кодов следует включить опцию ICOM codes на закладке Display диалога Model Properties (меню Model/Model Properties) (рис. 7.12).

Словарь **стрелок** редактируется при помощи специального редактора Arrow Dictionary Editor, в котором определяется *стрелка* и вносится относящийся к ней комментарий (рис. 7.13). Словарь стрелок решает очень важную задачу. Диаграммы создаются аналитиком для того, чтобы провести сеанс экспертизы, т. е. обсудить диаграмму со специалистом предметной области. В любой предметной области формируется профессиональный жаргон, причем очень часто жаргонные выражения имеют нечеткий смысл и воспринимаются разными специалистами по-разному. В то же время аналитик — автор диаграмм должен употреблять те выражения, которые наиболее понятны экспертам. Поскольку формальные определения часто сложны для восприятия, аналитик вынужден употреблять профессиональный жаргон, а чтобы не возникло неоднозначных трактовок, в

словаре стрелок каждому понятию можно дать расширенное и, если это необходимо, формальное определение.

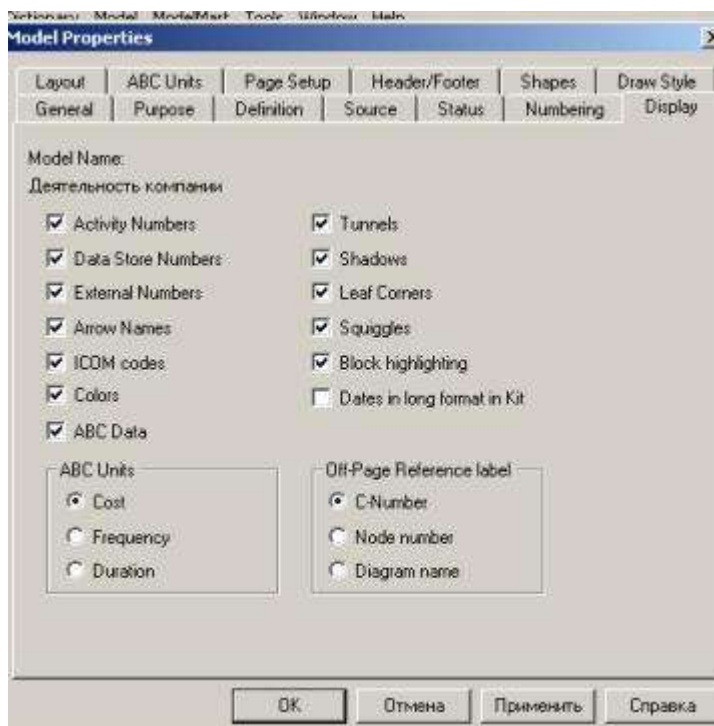


Рис. 7.12. Включение опции ICOM codes на закладке Display

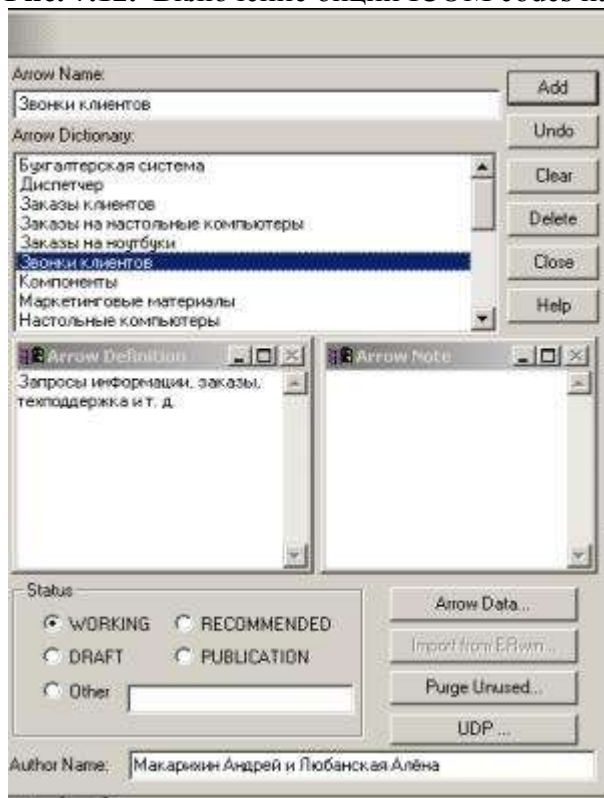


Рис. 7.13. Редактирование словаря стрелок

Содержимое словаря стрелок можно распечатать в виде отчета (меню Tools/ Report /Arrow Report...) и получить толковый словарь терминов предметной области, использующихся в модели.

Несвязанные граничные стрелки (unconnected border arrow). При декомпозиции *работы* входящие в нее и исходящие из нее *стрелки* (кроме *стрелки* вызова) автоматически появляются на диаграмме декомпозиции (миграция стрелок), но при этом не касаются *работ*. Такие *стрелки* называются несвязанными и воспринимаются в VPwin как синтаксическая ошибка.

На [рис. 7.14](#) приведен фрагмент диаграммы декомпозиции с несвязанными *стрелками*, генерирующийся VPwin при декомпозиции *работы* "Сборка настольных компьютеров" (см. [рис. 7.9](#)). Для связывания стрелок входа, управления или механизма необходимо перейти в режим редактирования стрелок, щелкнуть по кончику *стрелки* и потом по соответствующему сегменту *работы*. Для связывания *стрелки* выхода необходимо перейти в режим редактирования стрелок, щелкнуть по сегменту выхода *работы* и затем по *стрелке*.

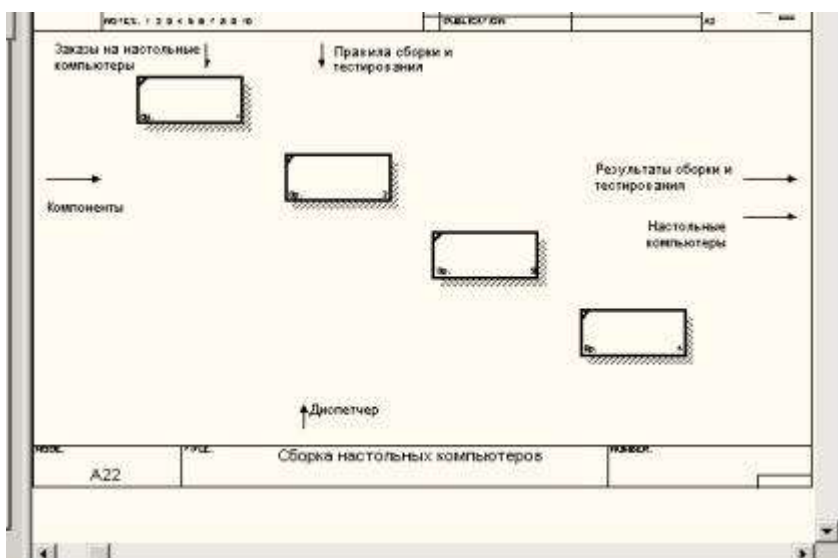


Рис. 7.14. Пример несвязанных стрелок

Внутренние стрелки. Для связи *работ* между собой используются внутренние *стрелки*, то есть *стрелки*, которые не касаются границы диаграммы, начинаются у одной и кончаются у другой *работы*.

Для рисования внутренней *стрелки* необходимо в режиме рисования стрелок щелкнуть по сегменту (например, выхода) одной *работы* и затем по сегменту (например, входа) другой. В IDEF0 различают пять типов связей *работ*.

Связь по входу (output-input), когда *стрелка* выхода вышестоящей *работы* (далее — просто выход) направляется на вход нижестоящей (например, на [рис. 7.15](#) *стрелка* "Собранные компьютеры" связывает *работы* "Сборка и тестирование компьютеров" и "Отгрузка и получение").



Рис. 7.15. Связь по входу

Связь по управлению (output-control), когда выход вышестоящей *работы* направляется на управление нижестоящей. Связь по управлению показывает доминирование вышестоящей *работы*. Данные или объекты выхода вышестоящей *работы* не меняются в нижестоящей. На [рис. 7.16](#) стрелка "Заказы клиентов" связывает *работы* "Продажи и маркетинг" и "Сборка и тестирование компьютеров".

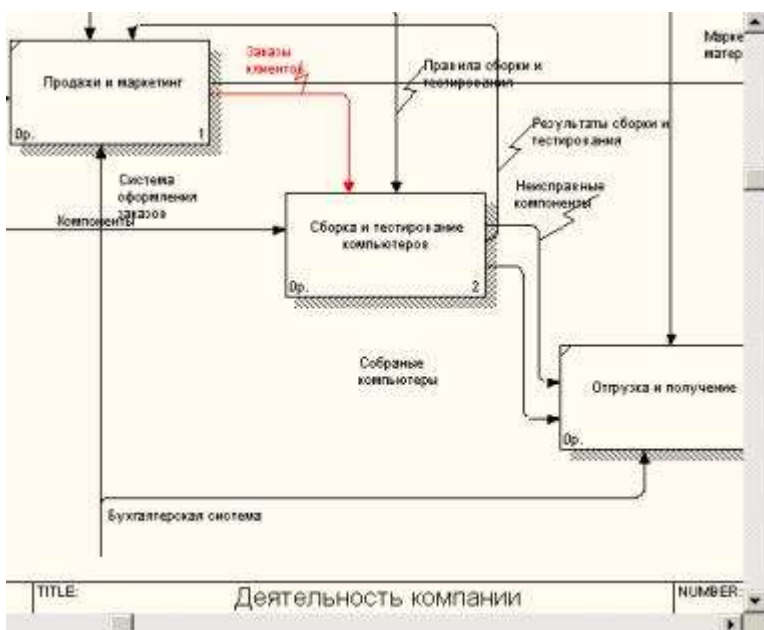


Рис. 7.16. Связь по управлению

Обратная связь по входу (output-input feedback), когда выход нижестоящей *работы* направляется на вход вышестоящей. Такая связь, как правило, используется для описания циклов. На [рис. 7.17](#) стрелка "Результаты тестирования" связывает *работы* "Тестирование компьютеров" и "Отслеживание расписания и управление сборкой и тестированием".

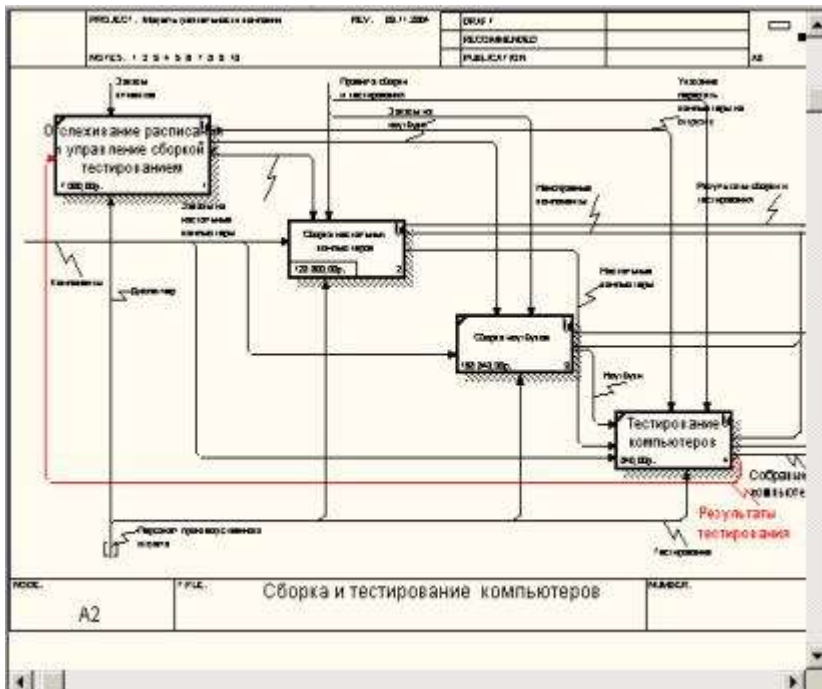


Рис. 7.17. Обратная связь по входу

Обратная связь по управлению (output-control feedback), когда выход нижестоящей работы направляется на управление вышестоящей (стрелка "Результаты сборки и тестирования", [рис. 7.18](#)). Обратная связь по управлению часто свидетельствует об эффективности бизнес-процесса. На [рис. 7.18](#) объем продаж может быть повышен путем непосредственного регулирования процессов сборки и тестирования компьютеров (выхода) работы "Сборка и тестирование компьютеров".

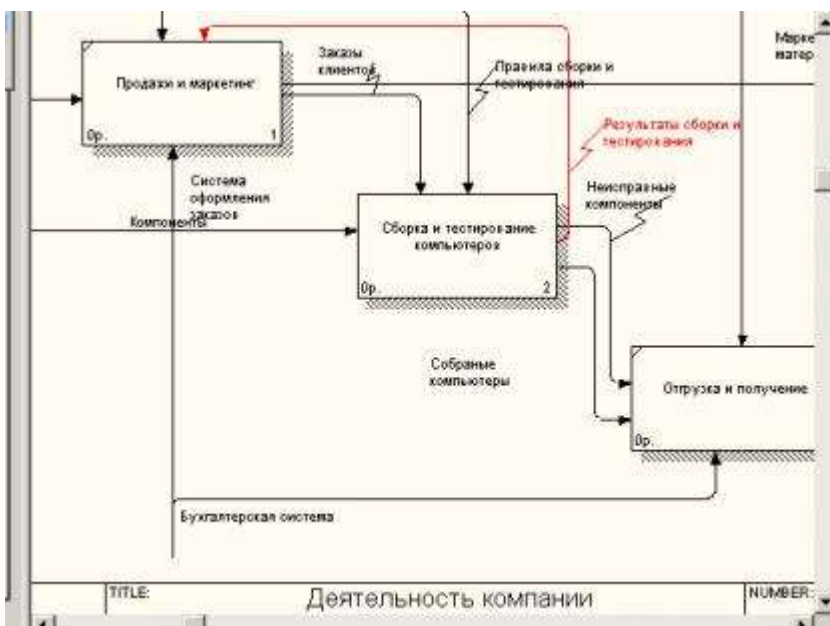


Рис. 7.18. Обратная связь по управлению

Связь выход-механизм (output-mechanism), когда выход одной работы направляется на механизм другой. Эта взаимосвязь используется реже остальных и показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы ([рис. 7.19](#)).

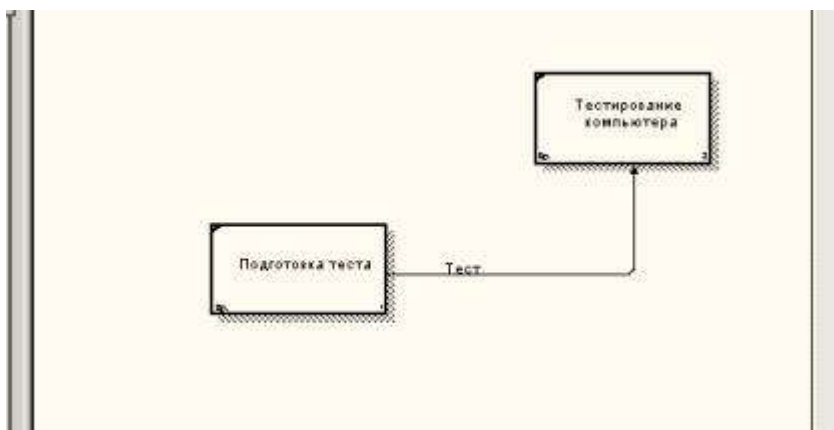


Рис. 7.19. Связь выход-механизм

Явные стрелки. Явная *стрелка* имеет источником одну-единственную *работу* и назначением тоже одну-единственную *работу*.

Разветвляющиеся и сливающиеся стрелки. Одни и те же данные или объекты, порожденные одной *работой*, могут использоваться сразу в нескольких других *работах*. С другой стороны, *стрелки*, порожденные в разных *работах*, могут представлять собой одинаковые или однородные данные или объекты, которые в дальнейшем используются или перерабатываются в одном месте. Для моделирования таких ситуаций в IDEF0 используются разветвляющиеся и сливающиеся *стрелки*. Для разветвления *стрелки* нужно в режиме редактирования *стрелки* щелкнуть по фрагменту *стрелки* и по соответствующему сегменту *работы*. Для слияния двух стрелок выхода нужно в режиме редактирования *стрелки* сначала щелкнуть по сегменту выхода *работы*, а затем по соответствующему фрагменту *стрелки*.

Смысл разветвляющихся и сливающихся стрелок передается именованием каждой ветви стрелок. Существуют определенные правила именования таких стрелок. Рассмотрим их на примере разветвляющихся стрелок. Если *стрелка* именована до разветвления, а после разветвления ни одна из ветвей не именована, то подразумевается, что каждая ветвь моделирует те же данные или объекты, что и ветвь до разветвления ([рис. 7.20](#)).

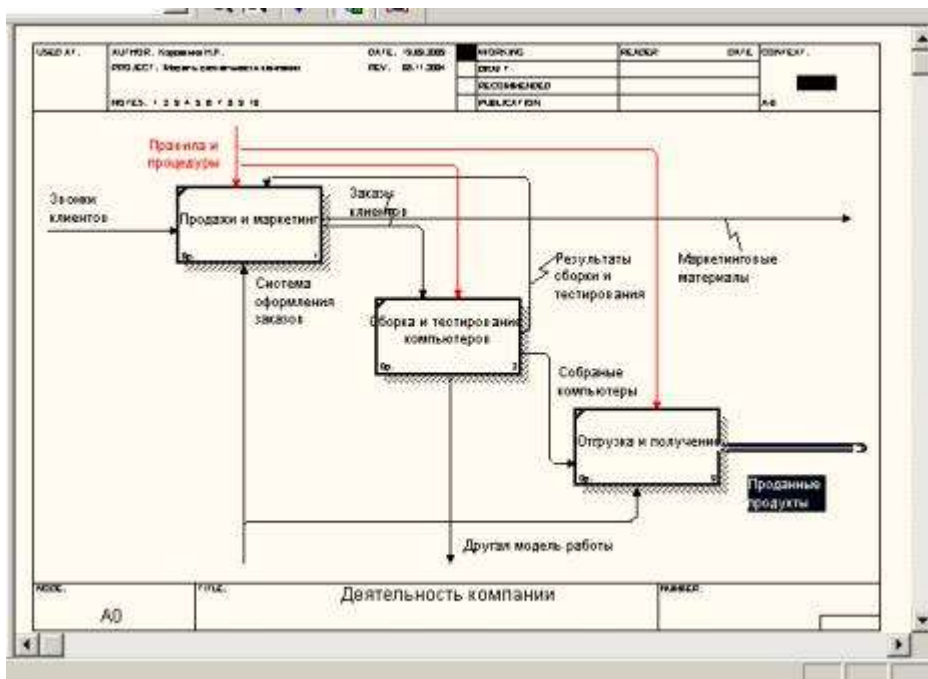


Рис. 7.20. Пример именования разветвляющейся стрелки

Если *стрелка* именована до разветвления, а после разветвления какая-либо из ветвей тоже именована, то подразумевается, что эти ветви соответствуют именованию. Если при этом какая-либо ветвь после разветвления осталась неименованной, то подразумевается, что она моделирует те же данные или объекты, что и ветвь до разветвления ([рис. 7.21](#)).

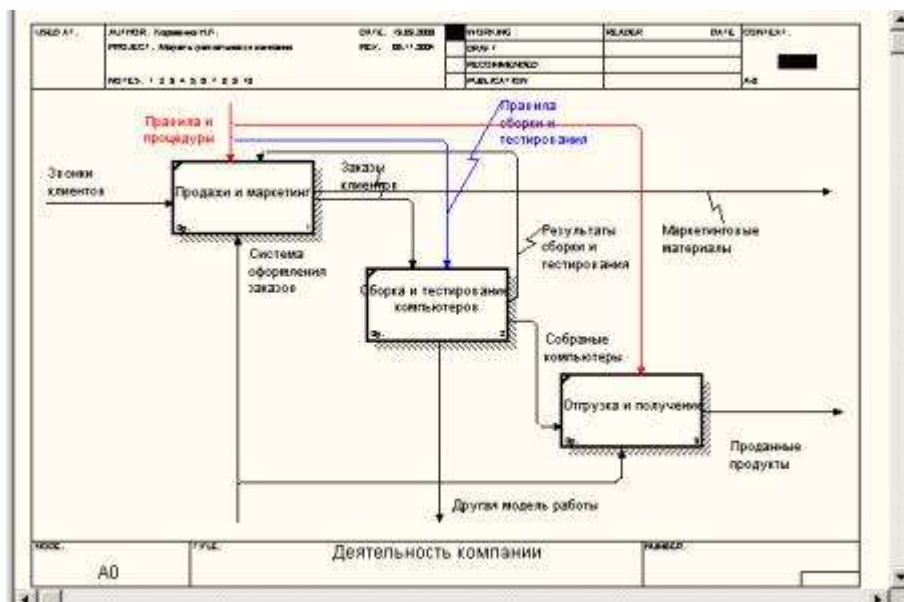


Рис. 7.21. Пример именования разветвляющейся стрелки

Недопустима ситуация, когда *стрелка* до разветвления не именована, а после разветвления не именована какая-либо из ветвей. ВРwin определяет такую *стрелку* как синтаксическую ошибку.

Правила именования сливающихся стрелок полностью аналогичны — ошибкой будет считаться *стрелка*, которая после слияния не именована, а до слияния не именована какая-

Туннелирование стрелок. Вновь внесенные граничные *стрелки* на диаграмме декомпозиции нижнего уровня изображаются в квадратных скобках и автоматически не появляются на диаграмме верхнего уровня (рис. 7.22).



Появляется диалог Border Arrow Editor (рис. 7.24).

Если щелкнуть по кнопке Resolve Border Arrow, *стрелка* мигрирует на диаграмму верхнего уровня, если по кнопке Change To Tunnel — *стрелка* будет туннелирована и не попадет на другую диаграмму. Туннельная *стрелка* изображается с круглыми скобками на конце ([рис. 7.25](#)).

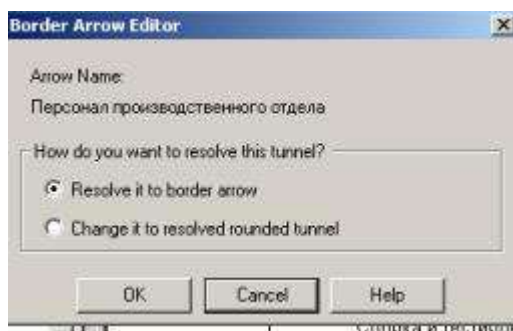


Рис. 7.24. Диалог Border Arrow Editor

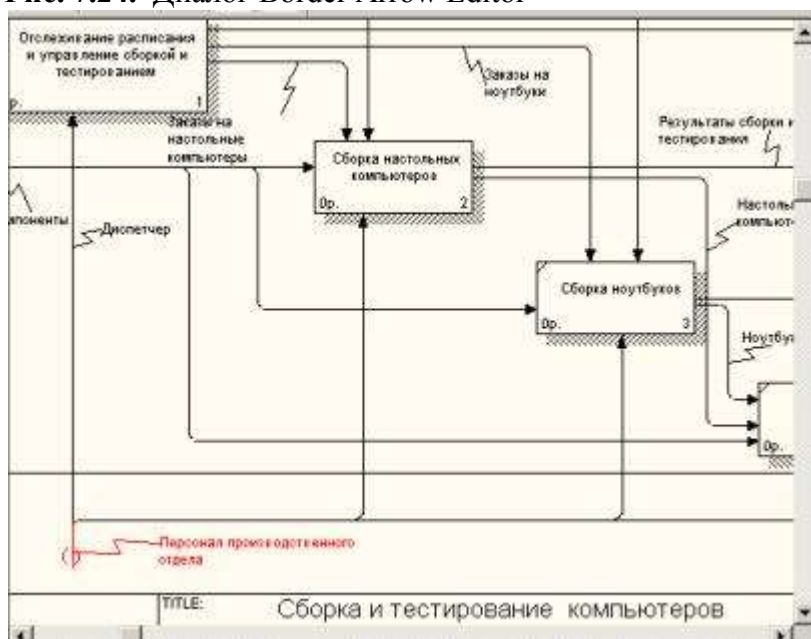


Рис. 7.25. Типы туннелирования стрелок

Туннелирование может быть применено для изображения малозначимых стрелок. Если на какой-либо диаграмме нижнего уровня необходимо изобразить малозначимые данные или объекты, которые не обрабатываются или не используются *работами* на текущем уровне, то их необходимо направить на вышестоящий уровень (на родительскую диаграмму). Если эти данные не используются на родительской диаграмме, их нужно направить еще выше, и т. д. В результате малозначимая *стрелка* будет изображена на всех уровнях и затруднит чтение всех диаграмм, на которых она присутствует. Выходом является туннелирование *стрелки* на самом нижнем уровне. Такое туннелирование называется "не-в-родительской-диаграмме".

Другим примером туннелирования может быть ситуация, когда *стрелка* механизма мигрирует с верхнего уровня на нижний, причем на нижнем уровне этот механизм используется одинаково во всех *работах* без исключения. (Предполагается, что не нужно детализировать *стрелку* механизма, т. е. *стрелка* механизма на дочерней *работе* именована до разветвления, а после разветвления ветви не имеют собственного имени). В этом случае *стрелка* механизма на нижнем уровне может быть удалена, после чего на

родительской диаграмме она может быть туннелирована, а в комментарии к *стрелке* или в словаре можно указать, что механизм будет использоваться во всех *работах* дочерней диаграммы декомпозиции. Такое туннелирование называется "не-в-дочерней-работе" ([рис. 7.25](#)).

Нумерация работ и диаграмм. Все *работы* модели нумеруются. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс **A**. Контекстная (корневая) *работа* дерева имеет номер **A0**. *Работы* **i** декомпозиции **A0** имеют номера **A1**, **A2**, **A3** и т. д. *Работы* декомпозиции нижнего уровня имеют номер родительской *работы* и очередной порядковый номер, например *работы* декомпозиции **A3** будут иметь номера **A31**, **A32**, **A33**, **A34** и т. д. *Работы* образуют иерархию, где каждая *работа* может иметь одну родительскую и несколько дочерних *работ*, образуя дерево. Такое дерево называют деревом узлов, а вышеописанную нумерацию — нумерацией по узлам. Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. *Контекстная диаграмма* всегда имеет номер **A-0**, декомпозиция *контекстной диаграммы* — номер **A0**, остальные диаграммы декомпозиции — номера по соответствующему узлу (например, **A1**, **A2**, **A21**, **A213** и т. д.). 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. VPwin позволяет иметь в модели только одну диаграмму декомпозиции в данном узле. Прежние версии диаграммы можно хранить в виде бумажной копии либо как FEO-диаграмму. (К сожалению, при создании FEO-диаграмм отсутствует возможность отката, т. е. из диаграммы можно получить декомпозиции FEO, но не наоборот.) В любом случае следует отличать различные версии одной и той же диаграммы. Для этого существует специальный номер — C-number, который должен присваиваться автором модели вручную. C-number — это произвольная строка, но рекомендуется придерживаться стандарта, когда номер состоит из буквенного префикса и порядкового номера, причем в качестве префикса используются инициалы автора диаграммы, а порядковый номер отслеживается автором вручную, например **MSB00021**.

Диаграммы дерева узлов и FEO

Диаграмма деревьев узлов показывает иерархию *работ* в модели и позволяет рассмотреть всю модель целиком, но не показывает взаимосвязи между *работами* ([рис. 7.26](#)). Процесс создания модели *работ* является итерационным, следовательно, *работы* могут менять свое расположение в дереве узлов многократно. Чтобы не запутаться и проверить способ декомпозиции, следует после каждого изменения создавать *диаграмму дерева узлов*. Впрочем, VPwin имеет мощный инструмент навигации по модели — Model Explorer, который позволяет представить иерархию *работ* и диаграмм в удобном и компактном виде, однако составляющей стандарта IDEF0.

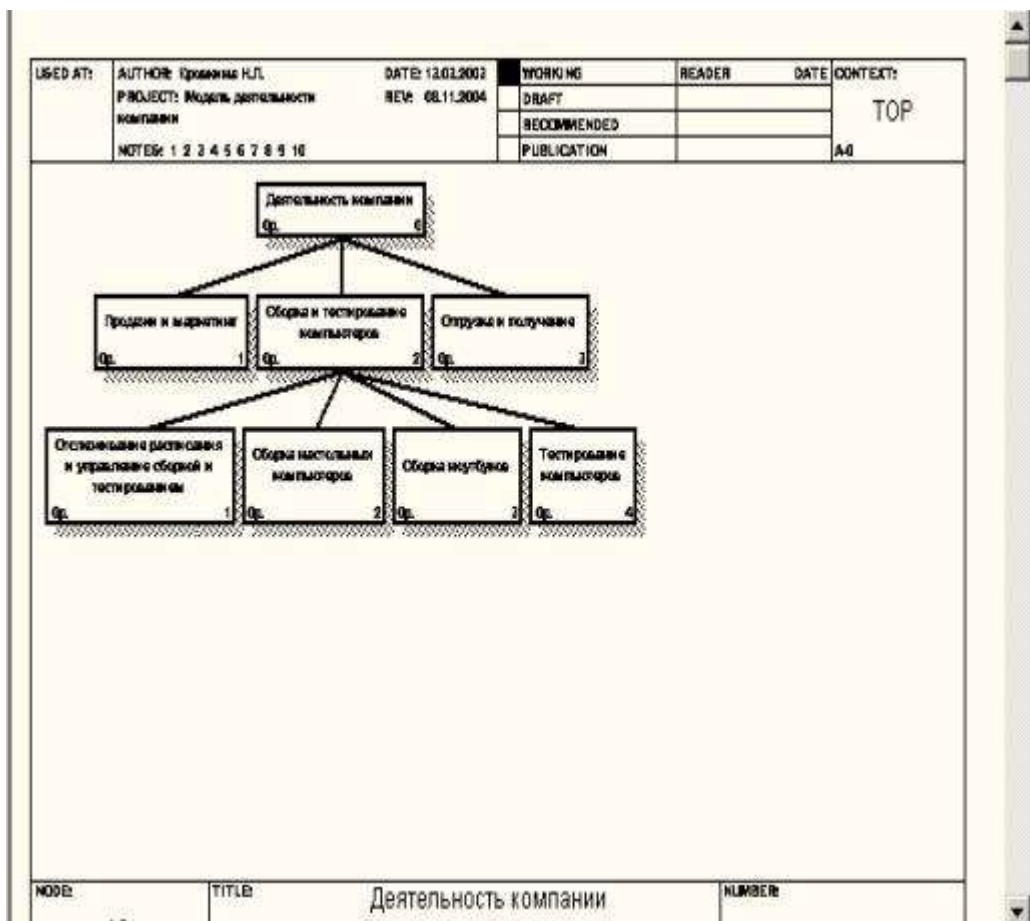


Рис. 7.26. Диаграмма дерева узлов

Для создания *диаграммы дерева узлов* следует выбрать в меню пункт Diagram/Add Node Tree (рис. 7.27). Возникает диалог формирования *диаграммы дерева узлов* Node Tree Definition (рис. 7.28, 7.29).



Рис. 7.27. Выбор команды для формирования диаграммы дерева узлов

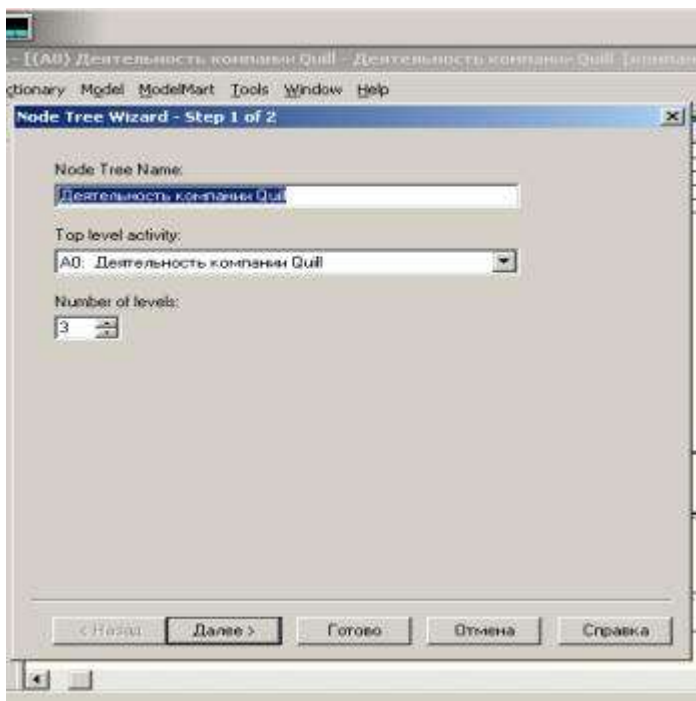


Рис. 7.28. Диалог настройки диаграммы дерева узлов (шаг 1)



Рис. 7.29. Диалог настройки диаграммы дерева узлов (шаг 2)

В диалоге Node Tree Definition следует указать глубину дерева — Number of Levels (по умолчанию — 3) и корень дерева (по умолчанию — родительская *работа* текущей

диаграммы). По умолчанию нижний уровень декомпозиции показывается в виде списка, остальные *работы* — в виде прямоугольников. Для отображения всего дерева в виде прямоугольников следует выключить опцию **Bullet Last Level**. При создании дерева узлов следует указать имя диаграммы, поскольку, если в нескольких диаграммах в качестве корня на дереве узлов использовать одну и ту же *работу*, все эти диаграммы получат одинаковый номер (номер узла + постфикс N, например AON) и в списке открытых диаграмм (пункт меню **Window**) их можно будет различить только по имени.

Диаграммы "*только для экспозиции*" (FEO) часто используются в модели для иллюстрации других точек зрения, для отображения отдельных деталей, которые не поддерживаются явно синтаксисом IDEF0. Диаграммы FEO позволяют нарушить любое синтаксическое правило, поскольку по сути являются просто картинками — копиями стандартных диаграмм и не включаются в анализ синтаксиса. Для создания диаграммы FEO следует выбрать пункт меню **Diagram/Add FEO Diagram**. В возникающем диалоге **Add New FEO Diagram** следует указать имя диаграммы FEO и тип родительской диаграммы ([рис. 7.30](#)).



Рис. 7.30. Диалог создания FEO-диаграммы

Новая диаграмма получает номер, который генерируется автоматически (номер родительской диаграммы по узлу + постфикс **F**, например **A1F**).

Каркас диаграммы

На [рис. 7.31](#) показан типичный пример диаграммы декомпозиции с граничными рамками, которые называются каркасом диаграммы.

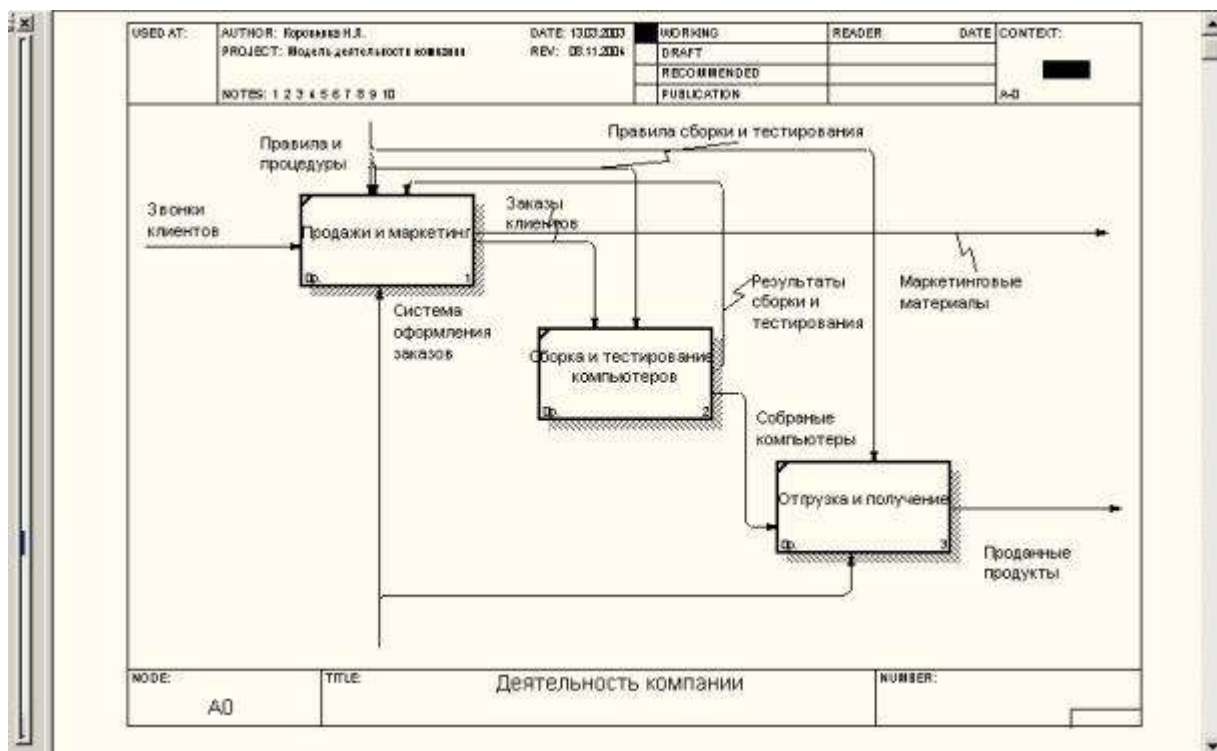


Рис. 7.31. Пример диаграммы декомпозиции с каркасом

Каркас содержит заголовок (верхняя часть рамки) и подвал (нижняя часть). Заголовок каркаса используется для отслеживания диаграммы в процессе моделирования. Нижняя часть используется для идентификации и позиционирования в иерархии диаграммы.

Смысл элементов каркаса приведен в [табл. 7.1](#) и [7.2](#).

Значения полей каркаса задаются в диалоге Diagram Properties (меню Diagram /Diagram Properties) — [рис. 7.32](#).

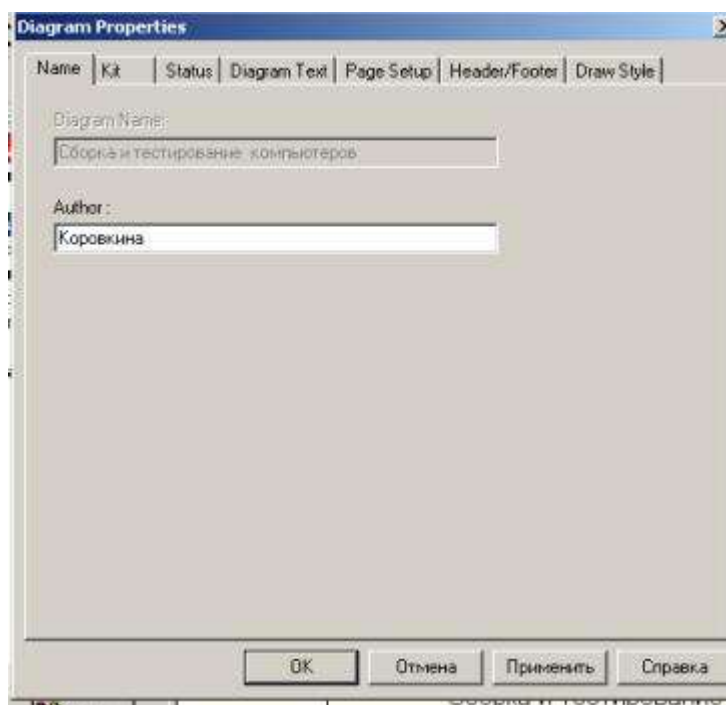
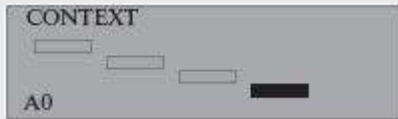


Рис. 7.32. Диалог Diagram Properties

Таблица 7.1. Поля заголовка каркаса (слева направо)

Поле	Смысл
Used At	Используется для указания на родительскую <i>работу</i> в случае, если на текущую диаграмму ссылались посредством <i>стрелки</i> вызова
Autor, Date, Rev, Project	Имя создателя диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма. REV-дата последнего редактирования диаграммы
Notes 123456789 10	Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания
Status	Статус отображает стадию создания диаграммы, отображая все этапы публикации
Working	Новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы
Draft	Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению
Recommended	Диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается

Publication	Диаграмма готова к окончательной печати и публикации
Reader	Имя читателя (эксперта)
Date	Дата прочтения (экспертизы)
Context	<p>Схема расположения <i>работ</i> в диаграмме верхнего уровня. <i>Работа</i>, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На <i>контекстной диаграмме</i> (A-0) показана надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы:</p> 

Слияние и расщепление моделей

Возможность слияния и расщепления моделей обеспечивает коллективную *работу* над проектом. Так, руководитель проекта может создать декомпозицию верхнего уровня и дать задание аналитикам продолжить декомпозицию каждой ветви дерева в виде отдельных моделей. После окончания *работы* над отдельными ветвями все подмодели могут быть слиты в единую модель. С другой стороны, отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели, для доработки или архивирования.

Таблица 7.2. Поля подвала каркаса (слева направо)

Поле	Смысл
Node	Номер узла диаграммы (номер родительской <i>работы</i>)
Title	Имя диаграммы. По умолчанию — имя родительской <i>работы</i>
Number	C-Number, уникальный номер версии диаграммы
Page	Номер страницы, может использоваться как номер страницы при формировании папки

BPwin использует для слияния и разветвления моделей *стрелки* вызова. Для слияния необходимо выполнить следующие условия:

- Обе сливаемые модели должны быть открыты в BPwin.
- Имя модели-источника, которое присоединяют к модели-цели, должно совпадать с именем *стрелки* вызова *работы* в модели-цели.
- *Стрелка* вызова должна исходить из недекомпозируемой *работы* (*работа* должна иметь диагональную черту в левом верхнем углу) ([рис. 7.33](#)).

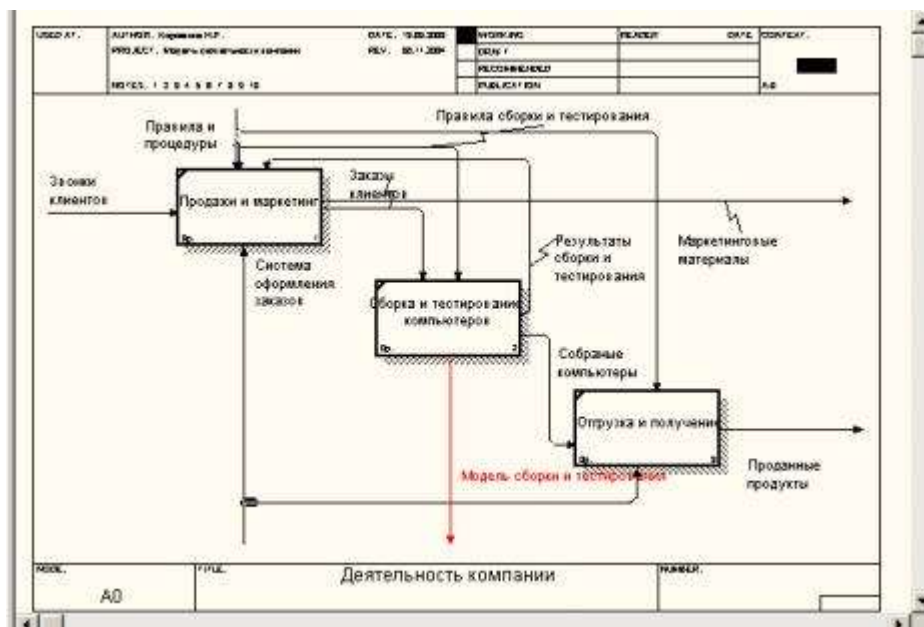


Рис. 7.33. Стрелка вызова работы "Сборка и тестирование компьютеров" модели-цели

- Имена контекстной *работы* подсоединяемой модели-источника и *работы* на модели-цели, к которой мы подсоединяем модель-источник, должны совпадать.
- Модель-источник должна иметь, по крайней мере, одну диаграмму декомпозиции.

Для слияния моделей нужно щелкнуть правой кнопкой мыши по *работе* со *стрелкой* вызова в модели-цели и во всплывающем меню выбрать пункт Merge Model.

Появляется диалог, в котором следует указать опции слияния модели (рис. 7.34). При слиянии моделей объединяются и словари стрелок и *работ*. В случае одинаковых определений возможна перезапись определений или принятие определений из модели-источника. То же относится к именам стрелок, хранилищ данных и внешним ссылкам. (Хранилища данных и внешние ссылки — объекты диаграмм потоков данных, DFD, будут рассмотрены ниже.)

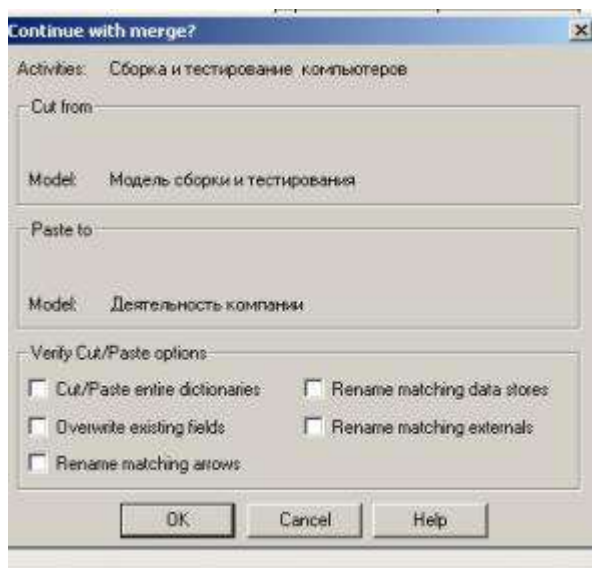


Рис. 7.34. Диалог Continue with merge

После подтверждения слияния (кнопка ОК) модель-источник подсоединяется к модели-цели, *стрелка* вызова исчезает, а *работа*, от которой отходила *стрелка* вызова, становится декомпозируемой — к ней подсоединяется диаграмма декомпозиции первого уровня модели-источника. *Стрелки*, касающиеся *работы* на диаграмме модели-цели, автоматически не мигрируют в декомпозицию, а отображаются как неразрешенные. Их следует туннелировать вручную.

В процессе слияния модель-источник остается неизменной, и к модели-цели подключается фактически ее копия. Не нужно путать слияние моделей с синхронизацией. Если в дальнейшем модель-источник будет редактироваться, эти изменения автоматически не попадут в соответствующую ветвь модели-цели.

Разделение моделей производится аналогично. Для отщепления ветви от модели следует щелкнуть правой кнопкой мыши по декомпозированной *работе* (*работа* не должна иметь диагональной черты в левом верхнем углу) и выбрать во всплывающем меню пункт Split Model. В появившемся диалоге Split Options следует указать имя создаваемой модели. После подтверждения расщепления в старой модели *работа* станет недекомпозированной (признак — диагональная черта в левом верхнем углу), будет создана *стрелка* вызова, ее имя будет совпадать с именем новой модели, и, наконец, будет создана новая модель, причем имя контекстной *работы* будет совпадать с именем *работы*, от которой была "оторвана" декомпозиция.

Создание отчетов в BPwin

BPwin имеет мощный инструмент генерации отчетов. Отчеты по модели вызываются из пункта меню Report. Всего имеется семь типов отчетов:

1. Model Report. Включает информацию о контексте модели — имя модели, точку зрения, область, цель, имя автора, дату создания и др.
2. Diagram Report. Отчет по конкретной диаграмме. Включает список объектов (*работ*, *стрелок*, хранилищ данных, внешних ссылок и т. д.).
3. Diagram Object Report. Наиболее полный отчет по модели. Может включать полный список объектов модели (*работ*, *стрелок* с указанием их типа и др.) и свойства, определяемые пользователем.
4. Activity Cost Report. Отчет о результатах стоимостного анализа. Будет рассмотрен ниже.
5. Arrow Report. Отчет по *стрелкам*. Может содержать информацию из словаря *стрелок*, информацию о работе-источнике, работе-назначении *стрелки* и информацию о разветвлении и слиянии *стрелок*.
6. Data Usage Report. Отчет о результатах связывания модели процессов и модели данных. (Будет рассмотрен ниже.)
7. Model Consistency Report. Отчет, содержащий список синтаксических ошибок модели.

Стоимостный анализ

Как было указано ранее, обычно сначала строится **функциональная** модель существующей организации работы — AS-IS (как есть). После построения модели AS-IS проводится анализ бизнес-процессов, потоки данных и объектов перенаправляются и улучшаются, в результате строится модель TO-BE. Как правило, строится несколько моделей TO-BE, из которых по какому-либо критерию выбирается наилучшая. Проблема состоит в том, что таких критериев много и непросто определить важнейший. Для того чтобы определить **качество** созданной модели с точки зрения эффективности бизнес-процессов, необходима система метрики, т. е. качество следует оценивать количественно.

BPwin предоставляет аналитику два инструмента для оценки модели — **стоимостный анализ**, основанный на работах (Activity Based Costing, ABC), и свойства, определяемые пользователем (User Defined Properties, UDP). **Функциональное** оценивание – ABC – это технология выявления и исследования стоимости выполнения той или иной функции (действия). Исходными данными для **функционального** оценивания являются затраты на ресурсы (материалы, персонал и т.д.). В сравнении с традиционными способами оценки затрат, при применении которых часто недооценивается продукция, производимая в незначительном объеме, и переоценивается массовый выпуск, ABC обеспечивает более точный метод расчета стоимости производства продукции, основанный на стоимости выполнения всех технологических операций, выполняемых при ее выпуске.

Стоимостный анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания **функциональности** предприятия. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR). С помощью **стоимостного анализа** можно решить такие задачи, как определение действительной стоимости производства продукта, определение действительной стоимости поддержки клиента, идентификация наиболее дорогостоящих работ (тех, которые должны быть улучшены в первую очередь), обеспечение менеджеров финансовой мерой предлагаемых изменений и т.д.

ABC-анализ может проводиться только тогда, когда модель работы последовательная (следует синтаксическим правилам IDEF0), корректная (отражает бизнес), полная (охватывает всю рассматриваемую область) и стабильная (проходит цикл экспертизы без изменений), другими словами, когда создание модели работы закончено.

ABC включает следующие основные понятия:

- **Объект затрат** — причина, по которой работа выполняется, обычно **основной выход работы**. Стоимость работ есть суммарная стоимость **объектов затрат** ("**Сборка и тестирование компьютеров**", [рис. 8.1](#));
- **Двигатель затрат** — характеристики входов и управлений работы ("**Заказы клиентов**", "**Правила сборки и тестирования**", "**Персонал производственного отдела**" [рис. 8.1](#)), которые влияют на то, как выполняется и как долго длится работа;
- **Центры затрат**, которые можно трактовать как статьи расхода.

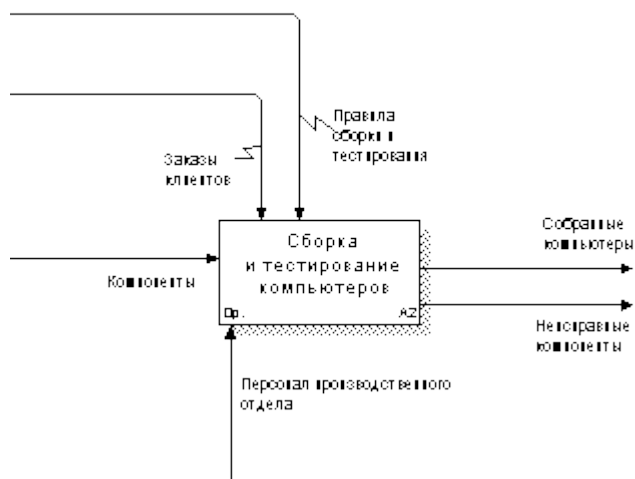


Рис. 8.1. Иллюстрация терминов ABC

При проведении *стоимостного анализа* в BPwin сначала задаются единицы измерения времени и денег. Для задания единиц измерения следует вызвать диалог Model Properties (меню Model), закладка ABC Units ([рис. 8.2](#)).

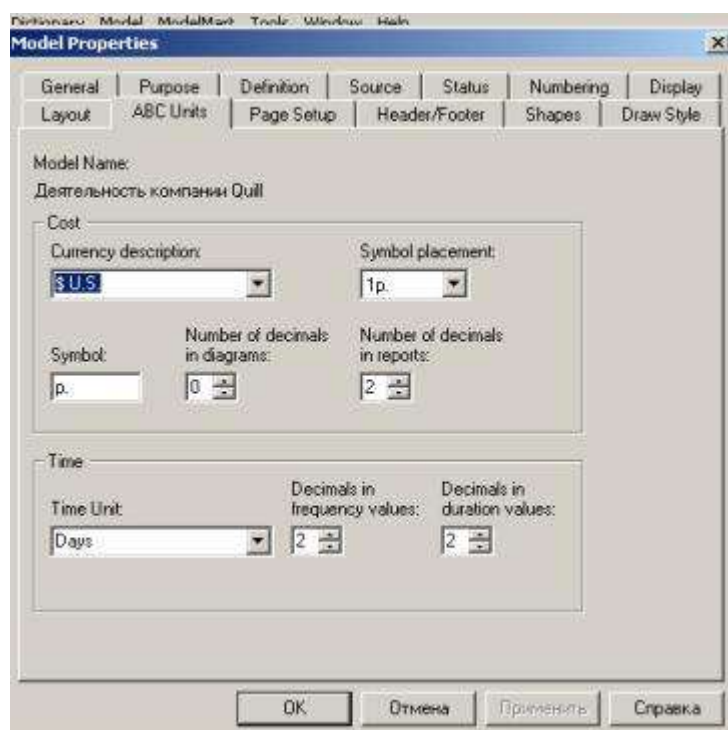


Рис. 8.2. Настройка единиц измерения валюты и времени

Если в списке выбора отсутствует необходимая валюта (например, рубль), ее можно добавить. Диапазон измерения времени в списке Unit of measurement достаточен для большинства случаев — от секунд до лет.

Затем описываются *центры затрат* (cost centers). Для внесения *центров затрат* необходимо вызвать диалог Cost Center Editor из меню Model ([рис. 8.3](#)).

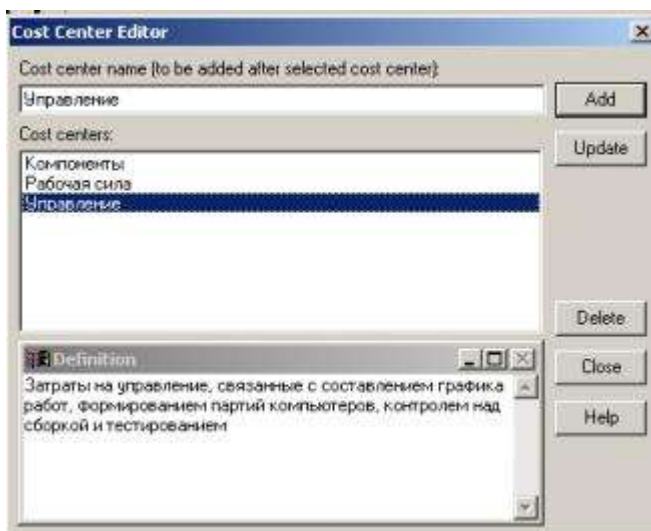


Рис. 8.3. Диалог Cost Center Editor

Каждому *центру затрат* следует дать подробное описание в окне Definition. Список *центров затрат* упорядочен. Порядок в списке можно менять при помощи стрелок, расположенных справа от списка. Задание определенной последовательности *центров затрат* в списке, во-первых, облегчает последующую работу при присвоении стоимости работам, а во-вторых, имеет значение при использовании единых стандартных отчетов в разных моделях. Хотя BPwin сохраняет информацию о стандартном отчете в файле BPWINRPT.INI, информация о *центрах затрат* и UDP сохраняется в виде указателей, т. е. хранятся не названия *центров затрат*, а их номера. Поэтому, если нужно использовать один и тот же стандартный отчет в разных моделях, списки *центров затрат* должны быть в них одинаковы.

Для задания стоимости работы (для каждой работы на диаграмме декомпозиции) следует щелкнуть правой кнопкой мыши по работе и на всплывающем меню выбрать Cost ([рис. 8.4](#)). В диалоге Activity Cost указывается частота проведения данной работы в рамках общего *процесса* (окно Frequency) и продолжительность (Duration). Затем следует выбрать в списке один из *центров затрат* и в окне Cost задать его стоимость. Аналогично назначаются суммы по каждому *центру затрат*, т. е. задается стоимость каждой работы по каждой статье расхода. Если в *процессе* назначения стоимости возникает необходимость внесения дополнительных *центров затрат*, диалог Cost Center Editor вызывается прямо из диалога Activity Properties/Cost соответствующей кнопкой.

Activity Properties

UDP Values | UOW | Source | Roles | Box Style

Name | Definition | Status | Font | Color | Costs

Activity Name: Сборка настольных компьютеров

Cost Center	Рубль
Компоненты	16 000,00
Рабочая сила	100,00
Управление	0,00

Data is from this level. Total cost: 16 100,00

☒ Override decompositions Total cost x Frequency: 128 800,00

☐ Compute from decompositions

Frequency: 8,00

Duration: 2,00 час

Duration x Frequency 16,0000 час

Cost Center Editor...

OK Отмена Применить Справка

Рис. 8.4. Задание стоимости работ в диалоге Activity Properties/Cost

Общие затраты по работе рассчитываются как сумма по всем *центрам затрат*. При вычислении затрат вышестоящей (родительской) работы сначала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим Compute from Decompositions ([рис. 8.4](#)), подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх ([рис. 8.5](#)).



Рис. 8.5. Вычисление затрат родительской работы

Этот достаточно упрощенный принцип подсчета справедлив, если работы выполняются последовательно. Встроенные возможности BPwin позволяют разрабатывать упрощенные модели стоимости, которые, тем не менее, оказываются чрезвычайно полезными при предварительной оценке затрат. Если схема выполнения более сложная (например, работы производятся альтернативно), можно отказаться от подсчета и задать итоговые суммы для каждой работы вручную (Override Decompositions). В этом случае результаты расчетов с нижних уровней декомпозиции будут игнорироваться, и при расчетах на

верхних уровнях будет учитываться сумма, заданная вручную. На любом уровне результаты расчетов сохраняются независимо от выбранного режима, поэтому при выключении опции Override Decompositions расчет снизу вверх производится обычным образом.

Для проведения более тонкого анализа можно воспользоваться специализированным средством *стоимостного анализа* EasyABC (ABC Technology, Inc.). BPwin имеет двунаправленный интерфейс с EasyABC. Для экспорта данных в EasyABC следует выбрать пункт меню File/Export/Node Tree, задать в диалоге Export Node Tree необходимые настройки и экспортировать дерево узлов в текстовый файл (.txt). Файл экспорта можно импортировать в EasyABC. После проведения необходимых расчетов результирующие данные можно импортировать из EasyABC в BPwin. Для импорта нужно выбрать меню File/Import/Costs и в диалоге Import Activity Costs выбрать необходимые установки.

Результаты *стоимостного анализа* могут существенно повлиять на очередность выполнения работ. Предположим, что для оценки качества изделия необходимо провести три работы:

- внешний осмотр — стоимость 50 руб.;
- пробное включение — стоимость 150 руб.;
- испытание на стенде — стоимость 300 руб.

Предположим также, что с точки зрения технологии очередность проведения работ несущественна, а вероятность выявления брака одинакова (50%). Пусть необходимо проверить восемь изделий. Если проводить работы в убывающем по стоимости порядке, то затраты на получение готового изделия составят:

300 руб. (испытание на стенде)*8 +150 руб. (пробное включение) *4 + 50 руб. (внешний осмотр) *2 = 3100 руб.

Если проводить работы в возрастающем по стоимости порядке, то на получение готового изделия будет затрачено:

50 руб. (внешний осмотр) *8 +150 руб. (пробное включение) *4 + 300 руб. (испытание на стенде) *2 = 1600 руб.

Следовательно, с целью минимизации затрат первой должна быть выполнена наиболее дешевая работа, затем — средняя по стоимости и в конце — наиболее дорогая.

Результаты *стоимостного анализа* наглядно представляются на специальном отчете BPwin, настройка которого производится в диалоговом окне Activity Cost Report (меню Tools/Reports/Activity Cost Report) ([рис. 8.6](#)). Отчет позволяет документировать имя, номер, определение и стоимость работ, как суммарную, так и отдельно по *центрам затрат*.



Рис. 8.6. Диалог настройки отчета по стоимости работ

Результаты отображаются и непосредственно на диаграммах. В левом нижнем углу прямоугольника работы может показываться либо стоимость (по умолчанию), либо продолжительность, либо частота проведения работы. Настройка отображения осуществляется в диалоге Model Properties (меню Model/Model Properties), закладка Display (ABC Data, ABC Units).

Свойства, определяемые пользователем (UDP)

ABC позволяет оценить стоимостные и временные характеристики системы. Если стоимостных показателей недостаточно, имеется возможность внесения собственных метрик — свойств, определенных пользователем — (User Defined Properties, UDP). UDP позволяют провести дополнительный анализ, хотя и без суммирующих подсчетов.

Для описания UDP служит диалог User-Defined Property Editor (меню Model/UDP Definition Editor) (рис. 8.7). В верхнем окне диалога вносится имя UDP, в списке выбора Datatype описывается тип свойства. Имеется возможность задания 18 различных типов UDP, в том числе управляющих команд и массивов, объединенных по категориям. Для внесения категории следует задать имя категории в окне New Keyword и щелкнуть по кнопке Add Category. Для присвоения свойства категории необходимо выбрать UDP из списка, затем категорию из списка категорий и щелкнуть по кнопке Update. Одна категория может объединять несколько свойств, в то же время одно свойство может входить в несколько категорий. Свойство типа List может содержать массив предварительно определенных значений. Для определения области значений UDP типа List следует задать значение свойства в окне New Keyword и щелкнуть по кнопке Add Member. Значения из списка можно редактировать и удалять.



Рис. 8.7. Диалог описания UDP

Каждой работе можно поставить в соответствие набор UDP. Для этого следует щелкнуть правой кнопкой мыши по работе и выбрать пункт меню UDP. В закладке UDP Values диалога IDEF0 Activity Properties можно задать значения UDP. Результат задания можно проанализировать в отчете Diagram Object Report (меню Tools/Report/Diagram Object Report) ([рис. 8.8](#)).

Диаграммы потоков данных

Диаграммы потоков данных (Data Flow Diagramming) являются основным средством моделирования функциональных требований к проектируемой системе. Требования представляются в виде иерархии *процессов*, связанных потоками данных. Диаграммы потоков данных показывают, как каждый *процесс* преобразует свои входные данные в выходные, и выявляют отношения между этими *процессами*. DFD-диаграммы успешно используются как дополнение к модели IDEF0 для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет моделируемую систему как сеть связанных работ. Основные компоненты DFD (как было сказано выше) – *процессы* или работы, *внешние сущности*, потоки данных, накопители данных (хранилища).



Рис. 8.8. Диалог настройки отчета Diagram Object Report

В BPwin для построения диаграмм потоков данных используется нотация Гейна-Карсона.

Для того чтобы дополнить модель IDEF0 диаграммой DFD, нужно в *процессе* декомпозиции в диалоге Activity Box Count "кликнуть" по радио-кнопке DFD. В палитре инструментов на новой диаграмме DFD появляются новые кнопки:

- **(External Reference)** — добавить в диаграмму внешнюю ссылку;
- **(Data store)** — добавить в диаграмму *хранилище данных*;
- **Diagram Dictionary Editor** – ссылка на другую страницу. В отличие от IDEF0 этот инструмент позволяет направить стрелку на любую диаграмму (а не только на верхний уровень).

В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) двигаются от одной работы к другой. Это представление потоков совместно с *хранилищами данных* и *внешними сущностями* делает модели DFD более похожими на физические характеристики системы — движение объектов, хранение объектов, поставка и распространение объектов ([рис. 8.9](#)).

В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, DFD рассматривает систему как совокупность предметов. Контекстная диаграмма часто включает работы и внешние ссылки. Работы обычно именуются по названию системы, например **"Система обработки информации"**. Включение внешних ссылок в контекстную диаграмму не отменяет требования методологии четко определить цель, область и единую точку зрения на моделируемую систему.



Рис. 8.9. Пример диаграммы DFD

В DFD **работы** (*процессы*) представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками со скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же, как *процессы* IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0 (рис. 8.9) (блоки "Проверка и внесение клиентов", "Внесение заказов").

Внешние сущности изображают входы в систему и/или выходы из системы.

Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы (рис. 8.9, блок "Звонки клиентов"). Одна *внешняя сущность* может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

Потоки работ изображаются стрелками и описывают движение объектов из одной части системы в другую. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы. В DFD также применяются двунаправленные стрелки для описания диалогов типа "команда-ответ" между работами, между работой и *внешней сущностью* и между *внешними сущностями* (рис. 8.9).

В отличие от стрелок, описывающих объекты в движении, *хранилища данных* изображают объекты в покое (рис. 8.10).

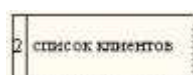


Рис. 8.10. Хранилище данных

В материальных системах *хранилища данных* изображаются там, где объекты ожидают обработки, например в *очереди*. В системах обработки информации *хранилища данных* являются **механизмом, который позволяет сохранить данные для последующих процессов.**

В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя.

Диаграммы DFD могут быть построены с использованием традиционного структурного анализа, подобно тому, как строятся диаграммы IDEF0.

В DFD номер каждой работы может включать префикс (A), номер родительской работы и номер объекта. Номер объекта — это уникальный номер работы на диаграмме. Например, работа может иметь номер A.12.4. Уникальный номер имеют *хранилища данных* и *внешние сущности* независимо от их расположения на диаграмме. Каждое *хранилище данных* имеет префикс D и уникальный номер, например D5. Каждая *внешняя сущность* имеет префикс E и уникальный номер, например E5.

Метод описания процессов IDEF3

Наличие в диаграммах DFD элементов для обозначения *источников*, приемников и *хранилищ данных* позволяет более эффективно и наглядно описать *процесс* документооборота. Однако для описания логики взаимодействия информационных потоков более подходит IDEF3, называемая также *workflow diagramming*, — методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между *процессами* обработки информации и объектов, являющихся частью этих *процессов*. Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием *процесса* и может быть использован для документирования каждой функции.

IDEF3 — это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда *процессы* выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном *процессе*.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний *процессов* IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания *процессов*. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес-процесса и может являться составляющей другой работы. Поскольку сценарий описывает цель и рамки модели, важно, чтобы работы именовались отглагольным существительным, обозначающим *процесс* действия, или фразой, содержащей такое существительное.

Точка зрения на модель должна быть документирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо документировать цель модели — те вопросы, на которые призвана ответить модель.

Диаграмма является основной единицей описания в IDEF3. Важно правильно построить диаграммы, поскольку они предназначены для чтения другими людьми (а не только автором).

Единицы работы — Unit of Work (UOW) — также называемые **работами** (activity), являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным **существительным**, обозначающим *процесс* действия, одиночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы (например, "Изготовление изделия"). Часто имя существительное в имени работы меняется в *процессе* моделирования, поскольку модель может уточняться и редактироваться. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, ее

идентификатор не будет вновь использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Связи показывают взаимоотношения работ. Все *связи* в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы *связи* были направлены слева направо. В IDEF3 различают **три типа стрелок, изображающих связи**, стиль которых устанавливается через меню Edit/Arrow Style:

Старшая (Precedence)



сплошная линия, связывающая единицы работ (UOW). Рисуеться слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется.

Отношения (Relational Link)



пунктирная линия, используемая для изображения *связей* между единицами работ (UOW) а также между единицами работ и объектами ссылок.

Потоки объектов (Object Flow)



стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Старшая связь показывает, что работа-источник заканчивается ранее, чем начинается работа-цель. Часто результатом работы-источника становится объект, необходимый для запуска работы-цели. В этом случае стрелку, обозначающую объект, изображают с двойным наконечником. Имя стрелки должно ясно идентифицировать отображаемый объект. Поток объектов имеет ту же семантику, что и старшая стрелка.

Отношение показывает, что стрелка является альтернативой старшей стрелке или потоку объектов в смысле задания последовательности выполнения работ — работа-источник не обязательно должна закончиться, прежде чем работа-цель начнется. Более того, работа-цель может закончиться прежде, чем закончится работа-источник.

Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. **Для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы, используются перекрестки (Junction).** Различают *перекрестки* для слияния (Fan-in Junction) и разветвления стрелок (Fan-out Junction). Перекресток не может использоваться одновременно для слияния и для разветвления. Для внесения *перекрестка* служит кнопка



— (добавить в диаграмму перекресток — Junction) в палитре инструментов. В диалоге Select Junction Type необходимо указать тип *перекрестка*.

Смысл каждого типа приведен в [таблице 8.1](#).

Все *перекрестки* на диаграмме нумеруются, каждый номер имеет префикс J. Можно редактировать свойства *перекрестка* при помощи диалога Junction Properties, который вызывается в контекстном меню *перекрестка* командой Definition/Note. В отличие от IDEF0 и DFD в IDEF3 стрелки могут сливаться и разветвляться только через *перекрестки*.

Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, *перекрестком* или работой. Для внесения объекта ссылки служит кнопка



— (добавить в диаграмму объект ссылки — Referent) в палитре инструментов. Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы



. Имя объекта ссылки задается в диалоге Referent (пункт Name контекстного меню), в качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или *перекрестками* пунктирными линиями. Официальная спецификация IDEF3 различает **три стиля объектов ссылок** — **безусловные** (unconditional), **синхронные** (synchronous) и **асинхронные** (asynchronous). BPwin поддерживает только безусловные объекты ссылок. Синхронные и асинхронные объекты ссылок, используемые в диаграммах переходов состояний объектов, не поддерживаются.

Таблица 8.1. Типы перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
	Asynchronous AND	Все предшествующие <i>процессы</i> должны быть завершены	Все следующие <i>процессы</i> должны быть запущены
	Synchronous AND	Все предшествующие <i>процессы</i> завершены одновременно	Все следующие <i>процессы</i> запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих <i>процессов</i> должны быть завершены	Один или несколько следующих <i>процессов</i> должны быть запущены
	Synchronous OR	Один или несколько предшествующих <i>процессов</i> завершены одновременно	Один или несколько следующих <i>процессов</i> запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий <i>процесс</i> завершен	Только один следующий <i>процесс</i> запускается

При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки. Типы объектов ссылок приведены в [таблице 8.2](#).

В IDEF3 **декомпозиция** используется для детализации работ. Методология IDEF3 позволяет декомпозировать работу многократно, т. е. работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки. Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ. Так, номер работы состоит из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме.

Рассмотрим *процесс* декомпозиции диаграмм IDEF3, включающий взаимодействие автора (аналитика) и одного или нескольких экспертов предметной области.

Перед проведением сеанса экспертизы у экспертов предметной области должны быть документированные сценарии и рамки модели, для того чтобы понять цели декомпозиции. Обычно эксперт предметной области передает аналитику текстовое описание сценария. В дополнение к этому может существовать документация, описывающая интересующие *процессы*. Из этой информации аналитик должен составить предварительный список работ (отглагольные существительные, обозначающие *процесс*) и объектов (существительные, обозначающие результат выполнения работы), которые необходимы для перечисленных работ. В некоторых случаях целесообразно создать графическую модель для представления ее эксперту предметной области.

Таблица 8.2. Типы объектов ссылок

Тип объекта ссылки	Цель описания
OBJECT	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме, но не обязательно. Если все работы цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекресток
UOB (Unit of behaviour)	Применяется, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа "Контроль качества" может быть использована в <i>процессе</i> "Изготовление изделия" несколько раз, после каждой единичной операции. Обычно этот тип ссылки не используется для моделирования автоматически запускающихся работ
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. NOTE является альтернативой внесению текстового объекта в диаграмму
ELAB (Elaboration)	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на <i>перекрестках</i>

На [рисунке 8.11](#) представлено описание *процесса* "Сборка настольных компьютеров" в методологии IDEF3.

Поскольку разные фрагменты модели IDEF3 могут быть созданы разными группами аналитиков в разное время, IDEF3 поддерживает простую схему нумерации работ в рамках всей модели. Разные аналитики оперируют разными диапазонами номеров, работая при этом независимо. Пример выделения диапазона приведен в [табл. 8.3](#).

В результате дополнения диаграмм IDEF0 диаграммами DFD и IDEF3 может быть создана смешанная модель, которая наилучшим образом описывает все стороны деятельности предприятия. Иерархию работ в смешанной модели можно увидеть в окне Model Explorer ([рис. 8.12](#)). Модели в нотации IDEF0 изображаются зеленым цветом, в IDEF3 — желтым, в DFD — голубым.

Таблица 8.3. Диапазоны номеров работ

Аналитик	Диапазон номеров IDEF3
Иванов	1-999
Петров	1000-1999
Сидоров	2000-2999

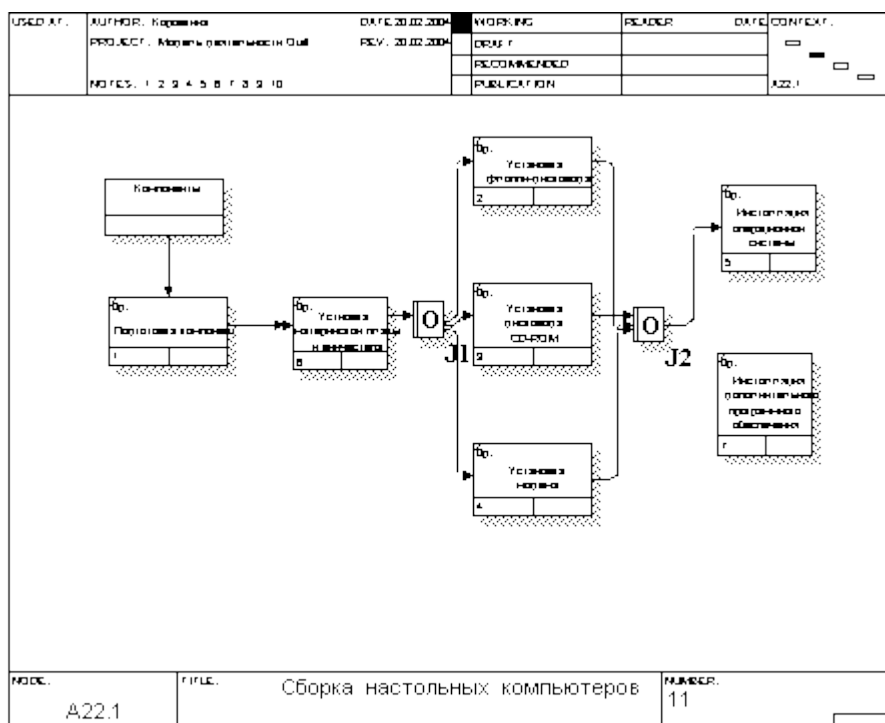


Рис. 8.11. Описание процесса в методологии IDEF3

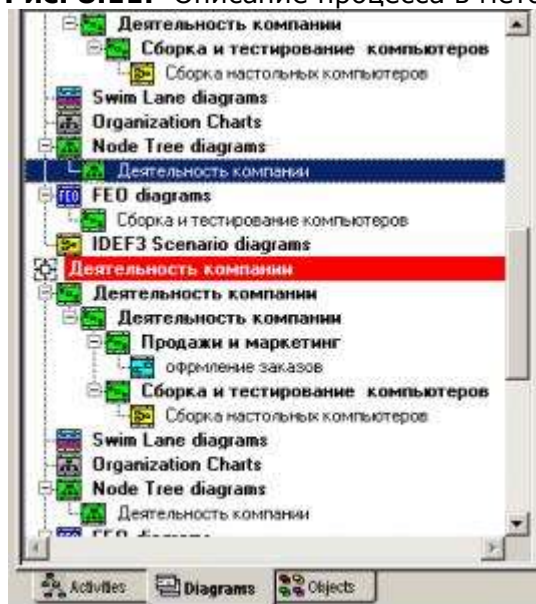


Рис. 8.12. Представление смешанной модели в окне Model Explorer

Имитационное моделирование

Оценочные аспекты моделирования предметной области связаны с разрабатываемыми показателями эффективности автоматизируемых *процессов*.

Метод функционального моделирования позволяет оптимизировать существующие на предприятии бизнес-процессы, однако для оптимизации конкретных технологических операций функциональной модели может быть недостаточно. В этом случае целесообразно использовать имитационное моделирование.

Имитационное моделирование – это метод, позволяющий строить модели, учитывающие время выполнения операций, и обеспечивающий наиболее полные средства анализа динамики бизнес-процессов. Имитационные модели описывают не только потоки сущностей, информации и управления, но и различные метрики. Полученную модель можно "проиграть" во времени и получить статистику происходящих *процессов* так, как это было бы в реальности. В имитационной модели изменения *процессов* и данных ассоциируются с событиями. "Проигрывание" модели заключается в последовательном переходе от одного события к другому.

Связь между имитационными моделями и моделями *процессов* заключается в возможности преобразования модели *процессов* в имитационную модель. Имитационная модель дает больше информации для анализа системы, в свою очередь результаты такого анализа могут быть причиной модификации модели *процессов*.

Одним из наиболее эффективных инструментов имитационного моделирования является система ARENA, разработанная фирмой System Modeling Corporation. Система позволяет строить имитационные модели, проигрывать их и анализировать результаты.

Имитационная модель включает следующие основные элементы:

- *Источники* и *стоки* (Create и Dispose). **Источники** — это элементы, от которых в модель поступает информация или объекты. По смыслу они близки к понятиям "внешняя ссылка" на DFD-диаграммах или "объект ссылки" на диаграммах IDEF3. Скорость поступления данных или объектов от *источника* обычно задается статистической функцией. **Сток** — это устройство для приема информации или объектов.
- *Очереди* (Queues). Понятие *очереди* близко к понятию *хранилища данных* на DFD-диаграммах — это место, где объекты ожидают обработки. Время обработки объектов в разных работах может быть разным. В результате перед некоторыми работами могут накапливаться объекты, ожидающие своей *очереди*. Часто целью имитационного моделирования является минимизация количества объектов в *очередях*.
- *Процессы* (Process) — это аналог работ в модели *процессов*. В имитационной модели может быть задана производительность *процессов*.

Построение модели производится путем переноса из панели инструментов в рабочее пространство модулей **Create, Dispose и Process**. *Связи* между модулями устанавливаются автоматически, но могут быть переопределены вручную. Далее модулям назначаются свойства. Для контроля проигрывания модели необходимо в модель добавить модуль Simulate и задать для него параметры. Результаты проигрывания модели отображаются в автоматически генерируемых отчетах.

BPwin не имеет собственных инструментов, позволяющих создавать имитационные модели, однако дает возможность экспортировать модель IDEF3 в специализированное средство создания таких моделей. Для экспорта модели необходимо настроить свойства, определяемые пользователем UDP, специально включенные в BPwin для целей экспорта.

Функциональные и имитационные модели тесно взаимосвязаны и эффективно дополняют друг друга. Имитационные модели дают больше информации для анализа системы, результаты которого могут быть причиной модификации модели *процессов*. Целесообразно сначала строить функциональную модель, а на ее основе — имитационную.

Лекция: Информационное обеспечение ИС

Информационное обеспечение ИС является средством для решения следующих задач:

- однозначного и экономичного представления информации в системе (на основе кодирования объектов);
- организации процедур анализа и обработки информации с учетом характера связей между объектами (на основе *классификации* объектов);
- организации взаимодействия пользователей с системой (на основе экранных форм ввода-вывода данных);
- обеспечения эффективного использования информации в контуре управления деятельностью объекта автоматизации (на основе унифицированной *системы документации*).

Информационное обеспечение ИС включает два комплекса: немашинное информационное обеспечение (*классификаторы* технико-экономической информации, документы, методические инструктивные материалы) и внутримашинное информационное обеспечение (макеты/экранные формы для ввода первичных данных в ЭВМ или вывода результатной информации, структуры *информационной базы*: входных, выходных файлов, базы данных).

К информационному обеспечению предъявляются следующие общие требования:

- информационное обеспечение должно быть достаточным для поддержания всех автоматизируемых функций объекта;
- для кодирования информации должны использоваться принятые у заказчика *классификаторы*;
- для кодирования входной и выходной информации, которая используется на высшем уровне управления, должны быть использованы *классификаторы* этого уровня;
- должна быть обеспечена совместимость с информационным обеспечением систем, взаимодействующих с разрабатываемой системой;
- формы документов должны отвечать требованиям корпоративных стандартов заказчика (или унифицированной *системы документации*);
- структура документов и экранных форм должна соответствовать характеристиками терминалов на рабочих местах конечных пользователей;
- графики формирования и содержание информационных сообщений, а также используемые аббревиатуры должны быть общеприняты в этой предметной области и согласованы с заказчиком;
- в ИС должны быть предусмотрены средства контроля входной и результатной информации, обновления данных в информационных массивах, контроля целостности *информационной базы*, защиты от несанкционированного доступа.

Информационное обеспечение ИС можно определить как совокупность единой *системы классификации*, унифицированной *системы документации* и *информационной базы* [\[21\]](#).

Немашинное информационное обеспечение

Основные понятия классификации технико-экономической информации

Для того чтобы обеспечить эффективный поиск, обработку на ЭВМ и передачу по каналам связи технико-экономической информации, ее необходимо представить в цифровом виде. С этой целью ее нужно сначала упорядочить (классифицировать), а затем формализовать (закодировать) с использованием *классификатора*.

Классификация – это разделение множества объектов на подмножества по их сходству или различию в соответствии с принятыми методами. *Классификация* фиксирует закономерные связи между классами объектов. Под объектом понимается любой предмет, процесс, явление материального или нематериального свойства. *Система классификации* позволяет сгруппировать объекты и выделить определенные классы, которые будут характеризоваться рядом общих свойств. Таким образом, совокупность правил распределения объектов множества на подмножества называется *системой классификации*.

Свойство или характеристика объекта *классификации*, которое позволяет установить его сходство или различие с другими объектами *классификации*, называется **признаком классификации**. Например, признак "роль предприятия-партнера в отношении деятельности объекта автоматизации" позволяет разделить все предприятия на две группы (на два подмножества): "поставщики" и "потребители". Множество или подмножество, объединяющее часть объектов *классификации* по одному или нескольким признакам, носит название **классификационной группировки**.

Классификатор — это документ, с помощью которого осуществляется формализованное описание информации в ИС, содержащей наименования объектов, наименования классификационных группировок и их кодовые обозначения [21].

По сфере действия выделяют следующие виды *классификаторов*: международные, общегосударственные (общесистемные), отраслевые и локальные *классификаторы*.

Международные *классификаторы* входят в состав Системы международных экономических стандартов (СМЭС) и обязательны для передачи информации между организациями разных стран мирового сообщества.

Общегосударственные (общесистемные) *классификаторы*, обязательны для организации процессов передачи и обработки информации между экономическими системами государственного уровня внутри страны.

Отраслевые *классификаторы* используют для выполнения процедур обработки информации и передачи ее между организациями внутри отрасли.

Локальные *классификаторы* используют в пределах отдельных предприятий.

Каждая *система классификации* характеризуется следующими свойствами:

- гибкостью системы;
- емкостью системы;
- степенью заполненности системы.

Гибкость системы — это способность допускать включение новых признаков, объектов без разрушения структуры *классификатора*. Необходимая гибкость определяется временем жизни системы.

Емкость системы — это наибольшее количество классификационных группировок, допускаемое в данной *системе классификации*.

Степень заполненности системы определяется как частное от деления фактического количества группировок на величину емкости системы.

В настоящее время чаще всего применяются два типа *систем классификации*: иерархическая и многоаспектная.

При использовании иерархического метода *классификации* происходит "последовательное разделение множества объектов на подчиненные, зависимые классификационные группировки" [22]. Получаемая на основе этого процесса классификационная схема имеет иерархическую структуру. В ней первоначальный объем классифицируемых объектов разбивается на подмножества по какому-либо признаку и детализируется на каждой следующей ступени *классификации*. Обобщенное изображение иерархической классификационной схемы представлено на [рис. 9.1](#).

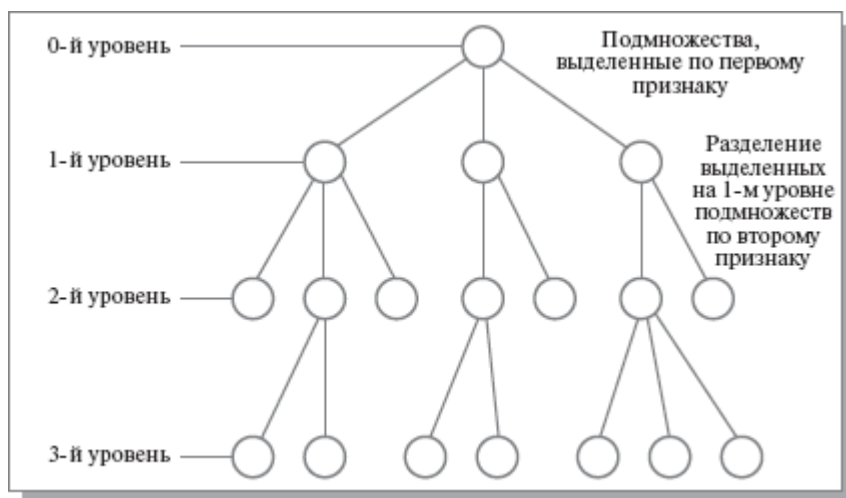


Рис. 9.1. Иерархическая классификационная схема

Характерными особенностями иерархической системы являются:

- возможность использования неограниченного количества признаков *классификации*;
- соподчиненность признаков *классификации*, что выражается разбиением каждой классификационной группировки, образованной по одному признаку, на множество классификационных группировок по нижестоящему (подчиненному) признаку.

Таким образом, классификационные схемы, построенные на основе иерархического принципа, имеют неограниченную емкость, величина которой зависит от глубины *классификации* (числа ступеней деления) и количества объектов *классификации*, которое можно расположить на каждой ступени. Количество же объектов на каждой ступени *классификации* определяется основанием кода, то есть числом знаков в выбранном алфавите кода. (Например, если алфавит – двузначные десятичные числа, то можно на одном уровне разместить 100 объектов). Выбор необходимой глубины *классификации* и структуры кода зависит от характера объектов *классификации* и характера задач, для решения которых предназначен *классификатор*.

При построении иерархической *системы классификации* сначала выделяется некоторое множество объектов, подлежащее классифицированию, для которого определяются полное множество признаков *классификации* и их соподчиненность друг другу, затем производится разбиение исходного множества объектов на классификационные группировки на каждой ступени *классификации*.

К положительным сторонам данной системы следует отнести логичность, простоту ее построения и удобство логической и арифметической обработки.

Серьезным недостатком иерархического метода *классификации* является жесткость классификационной схемы. Она обусловлена заранее установленным выбором признаков *классификации* и порядком их использования по ступеням *классификации*.

Это ведет к тому, что при изменении состава объектов *классификации*, их характеристик или характера решаемых при помощи *классификатора* задач требуется коренная переработка классификационной схемы. Гибкость этой системы обеспечивается только за счет ввода большой избыточности в ветвях, что приводит к слабой заполненности структуры *классификатора*. Поэтому при разработке *классификаторов* следует учитывать, что иерархический метод классификации более предпочтителен для объектов с относительно стабильными признаками и для решения стабильного комплекса задач.

Примеры применения иерархической *классификации* объектов в корпоративной ИС приведены на [рис 9.2](#) и [9.3](#). Использование приведенных моделей позволяет выполнить кодирование информации о соответствующих объектах, а также использовать процедуры обобщения при обработке данных (при анализе затрат на заработную плату — по принадлежности работника к определенной службе, при анализе затрат на производство — по группам материалов: по металлу, по покупным комплектующим и пр.).

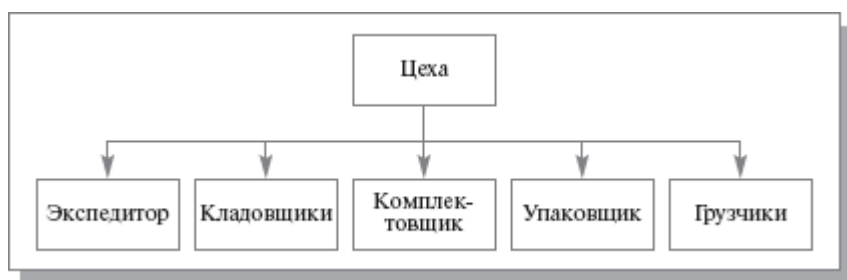


Рис. 9.2. Организационная структура подразделения предприятия-цеха отгрузки

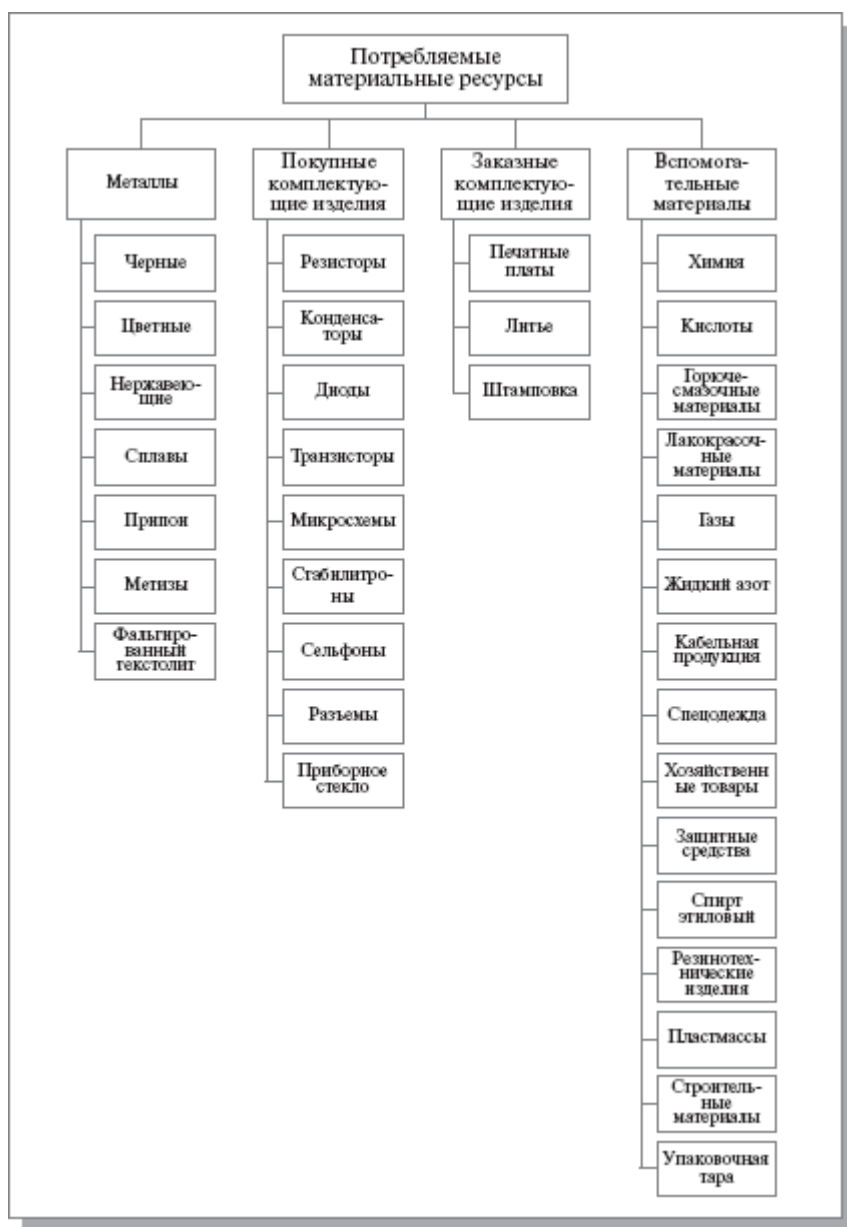


Рис. 9.3. Классификатор материальных ресурсов для обеспечения производства

Недостатки, отмеченные в иерархической системе, отсутствуют в других системах, которые относятся к классу многоаспектных систем классификации.

Аспект — точка зрения на объект классификации, который характеризуется одним или несколькими признаками. **Многоаспектная система** — это система классификации, которая использует параллельно несколько независимых признаков (аспектов) в качестве основания классификации. Существуют два типа многоаспектных систем: фасетная и дескрипторная. **Фасет** — это аспект классификации, который используется для образования независимых классификационных группировок. **Дескриптор** — ключевое слово, определяющее некоторое понятие, которое формирует описание объекта и дает принадлежность этого объекта к классу, группе и т.д.

Под фасетным методом классификации понимается "параллельное разделение множества объектов на независимые классификационные группировки" [22]. При этом методе классификации заранее жесткой классификационной схемы и конечных группировок не создается. Разрабатывается лишь система таблиц признаков объектов классификации, называемых фасетами. При необходимости создания классификационной группировки для решения конкретной задачи осуществляется

выборка необходимых признаков из фасетов и их объединение в определенной последовательности. Общий вид фасетной классификационной схемы представлен на [рис. 9.4](#).

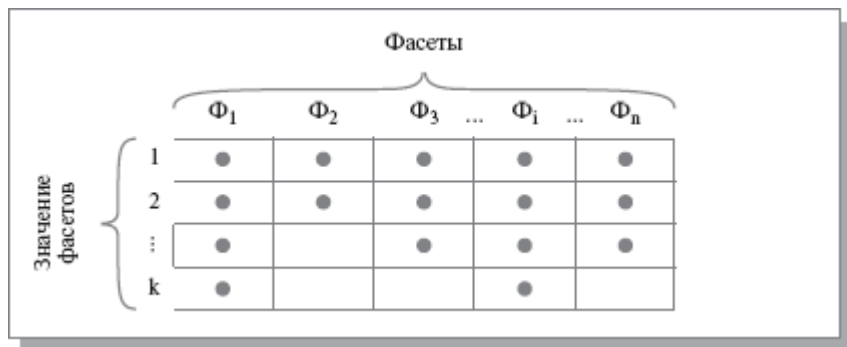


Рис. 9.4. Схема признаков фасетной классификации

Внутри фасета значения признаков могут просто перечисляться по некоторому порядку или образовывать сложную иерархическую структуру, если существует соподчиненность выделенных признаков.

К преимуществам данной системы следует отнести большую емкость системы и высокую степень гибкости, поскольку при необходимости можно вводить дополнительные фасеты и изменять их место в формуле. При изменении характера задач или характеристик объектов *классификации* разрабатываются новые фасеты или дополняются новыми признаками уже существующие фасеты без коренной перестройки структуры всего классификатора.

К недостаткам, характерным для данной системы, можно отнести сложность структуры и низкую степень заполненности системы.

В современных классификационных схемах часто одновременно используются оба метода *классификации*. Это снижает влияние недостатков методов *классификации* и расширяет возможность использования *классификаторов* в информационном обеспечении управления.

В качестве примера использования комбинированных схем *классификации* в корпоративных ИС можно привести следующую модель описания продукции предприятия.

Правила классификации продукции

Принята *классификация* выпускаемой продукции по следующему ряду уровней (Иерархическая *классификация*):

- семейство продуктов;
- группа продуктов;
- серия продуктов.

Однако эта *система классификации* не обеспечивает идентификацию любого выпускаемого изделия. Для каждой единицы продукта должны указываться следующие атрибуты (Фасеты):

- код серии продукта;
- конфигурационные параметры;

- свойства.

Код серии продукта – алфавитно-цифровой код, однозначно идентифицирующий отдельный продукт. Конфигурационные параметры – свойства, значения которых могут быть различными в зависимости от потребностей пользователей. Свойства – predetermined characteristics of individual products, which cannot change for one and the same product.

Допустимые варианты записи кода серии для различных продуктов показаны на [рис. 9.5](#).

Признаки фасета "Конфигурационные параметры" для одного семейства продуктов приведены в [таблице 9.1](#).

Рассмотренные выше *системы классификации* хорошо приспособлены для организации поиска с целью последующей логической и арифметической обработки информации на ЭВМ, но лишь частично решают проблему содержательного поиска информации при принятии управленческих решений.

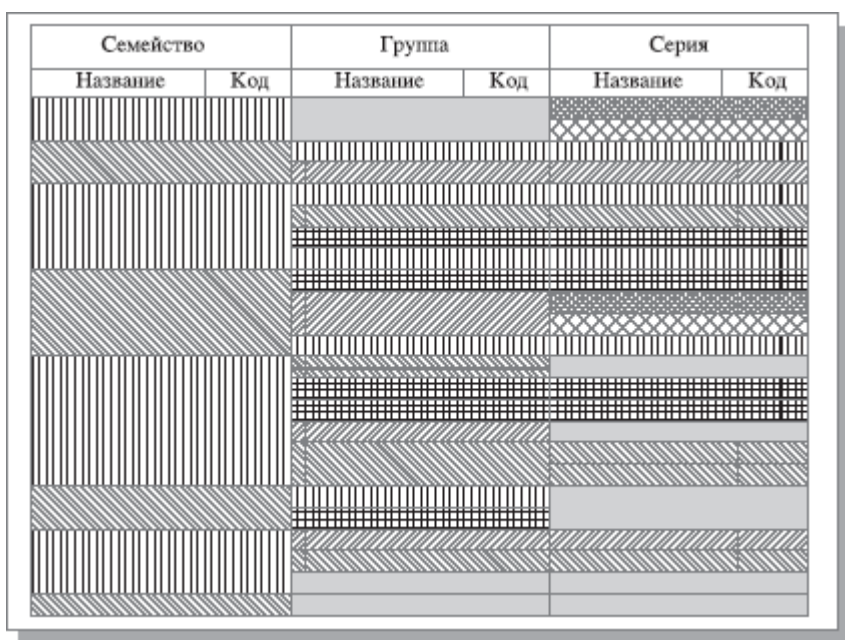


Рис. 9.5. Варианты записи кода серии продукта (серым цветом отмечены неиспользуемые элементы кода)

Таблица 9.1. Признаки фасета "Конфигурационные параметры" для одного семейства продуктов

Продукты и модификации	Характеристики	
	Общие для семейства	Специальные для отдельных моделей
Датчики разности давлений	<ul style="list-style-type: none"> Искробезопасное исполнение Взрывозащищенное исполнение 	<ul style="list-style-type: none"> Предельно допустимое рабочее избыточное давление

Датчики абсолютного давления, избыточного давления, разрежения, давления-разрежения	<ul style="list-style-type: none"> • Исполнение по материалам • Климатическое исполнение • Предел допускаемой основной погрешности • Верхний предел измерений • Код выходного сигнала • Состав комплекта монтажных частей 	<ul style="list-style-type: none"> • Измеряемый параметр
---	---	---

Для поиска показателей и документов по набору содержательных признаков используется информационный язык дескрипторного типа, который характеризуется совокупностью терминов (дескрипторов) и набором отношений между терминами.

Содержание документов или показателей можно достаточно полно и точно отразить с помощью списка ключевых слов — дескрипторов. **Дескриптор** — это термин естественного языка (слово или словосочетание), используемый при описании документов или показателей, который имеет самостоятельный смысл и неделим без изменения своего значения.

Для того чтобы обеспечить точность и однозначность поиска с помощью дескрипторного языка, необходимо предварительно определить все постоянные отношения между терминами: родовидовые, отношения синонимии, омонимии и полисемии, а также ассоциативные отношения.

Все выделенные отношения явно описываются в систематическом словаре понятий — **тезаурусе**, который разрабатывается с целью проведения индексирования документов, показателей и информационных запросов.

Кодирование технико-экономической информации

Для полной формализации информации недостаточно простой *классификации*, поэтому проводят следующую процедуру — кодирование. **Кодирование** — это процесс присвоения условных обозначений объектам и классификационным группам по соответствующей системе кодирования. Кодирование реализует перевод информации, выраженной одной системой знаков, в другую систему, то есть перевод записи на естественном языке в запись с помощью кодов. **Система кодирования** — это совокупность правил обозначения объектов и группировок с использованием кодов. **Код** — это условное обозначение объектов или группировок в виде знака или группы знаков в соответствии с принятой системой. Код базируется на определенном алфавите (некоторое множество знаков). Число знаков этого множества называется основанием кода. Различают следующие типы алфавитов: цифровой, буквенный и смешанный.

Код характеризуется следующими параметрами:

- длиной;
- основанием кодирования;
- структурой кода, под которой понимают распределение знаков по признакам и объектам *классификации*;
- степенью информативности, рассчитываемой как частное от деления общего количества признаков на длину кода;
- коэффициентом избыточности, который определяется как отношение максимального количества объектов к фактическому количеству объектов.

К методам кодирования предъявляются определенные требования:

- код должен осуществлять идентификацию объекта в пределах заданного множества объектов *классификации*;
- желательно предусматривать использование в качестве алфавита кода десятичных цифр и букв;
- необходимо обеспечивать по возможности минимальную длину кода и достаточный резерв незанятых позиций для кодирования новых объектов без нарушения структуры *классификатора*.

Методы кодирования могут носить самостоятельный характер – регистрационные методы кодирования, или быть основанными на предварительной *классификации* объектов – классификационные методы кодирования.

Регистрационные методы кодирования бывают двух видов: порядковый и серийно-порядковый. В первом случае кодами служат числа натурального ряда. Каждый из объектов классифицируемого множества кодируется путем присвоения ему текущего порядкового номера. Данный метод кодирования обеспечивает довольно большую долговечность *классификатора* при незначительной избыточности кода. Этот метод обладает наибольшей простотой, использует наиболее короткие коды и лучше обеспечивает однозначность каждого объекта *классификации*. Кроме того, он обеспечивает наиболее простое присвоение кодов новым объектам, появляющимся в процессе ведения *классификатора*. Существенным недостатком порядкового метода кодирования является отсутствие в коде какой-либо конкретной информации о свойствах объекта, а также сложность машинной обработки информации при получении итогов по группе объектов *классификации* с одинаковыми признаками.

В серийно-порядковом методе кодирования кодами служат числа натурального ряда с закреплением отдельных серий этих чисел (интервалов натурального ряда) за объектами *классификации* с одинаковыми признаками. В каждой серии, кроме кодов имеющихся объектов *классификации*, предусматривается определенное количество кодов для резерва.

Классификационные коды используют для отражения классификационных взаимосвязей объектов и группировок и применяются в основном для сложной логической обработки экономической информации. Группу классификационных систем кодирования можно разделить на две подгруппы в зависимости от того, какую *систему классификации* используют для упорядочения объектов: системы последовательного кодирования и параллельного кодирования.

Последовательные системы кодирования характеризуются тем, что они базируются на предварительной *классификации* по иерархической системе. Код объекта *классификации* образуется с использованием кодов последовательно расположенных подчиненных группировок, полученных при иерархическом методе кодирования. В этом случае код нижестоящей группировки образуется путем добавления соответствующего количества разрядов к коду вышестоящей группировки.

Параллельные системы кодирования характеризуются тем, что они строятся на основе использования фасетной *системы классификации* и коды группировок по фасетам формируются независимо друг от друга.

В параллельной системе кодирования возможны два варианта записи кодов объекта:

1. Каждый фасет и признак внутри фасета имеют свои коды, которые включаются в состав кода объекта. Такой способ записи удобно применять тогда, когда объекты характеризуются неодинаковым набором признаков. При формировании кода какого-либо объекта берутся только необходимые признаки.

2. Для определения групп объектов выделяется фиксированный набор признаков и устанавливается стабильный порядок их следования, то есть устанавливается фасетная формула. В этом случае не надо каждый раз указывать, значение какого из признаков приведено в определенных разрядах кода объекта.

Параллельный метод кодирования имеет ряд преимуществ. К достоинствам рассматриваемого метода следует отнести гибкость структуры кода, обусловленную независимостью признаков, из кодов которых строится код объекта *классификации*. Метод позволяет использовать при решении конкретных технико-экономических и социальных задач коды только тех признаков объектов, которые необходимы, что дает возможность работать в каждом отдельном случае с кодами небольшой длины. При этом методе кодирования можно осуществлять группировку объектов по любому сочетанию признаков. Параллельный метод кодирования хорошо приспособлен для машинной обработки информации. По конкретной кодовой комбинации легко узнать, набором каких характеристик обладает рассматриваемый объект. При этом из небольшого числа признаков можно образовать большое число кодовых комбинаций. Набор признаков при необходимости может легко пополняться присоединением кода нового признака. Это свойство параллельного метода кодирования особенно важно при решении технико-экономических задач, состав которых часто меняется.

Наиболее сложными вопросами, которые приходится решать при разработке *классификатора*, являются выбор методов *классификации* и кодирования и выбор системы признаков *классификации*. Основой *классификатора* должны быть наиболее существенные признаки *классификации*, соответствующие характеру решаемых с помощью *классификатора* задач. При этом данные признаки могут быть или соподчиненными, или несоподчиненными. При соподчиненных признаках *классификации* и стабильном комплексе задач, для решения которых предназначен *классификатор*, целесообразно использовать иерархический метод *классификации*, который представляет собой последовательное разделение множества объектов на подчиненные классификационные группировки. При несоподчиненных признаках *классификации* и при большой динамичности решаемых задач целесообразно использовать фасетный метод *классификации*.

Важным вопросом является также правильный выбор последовательности использования признаков *классификации* по ступеням *классификации* при иерархическом методе *классификации*. Критерием при этом является статистика запросов к *классификатору*. В соответствии с этим критерием на верхних ступенях *классификации* в *классификаторе* должны использоваться признаки, к которым будут наиболее частые запросы. По этой же причине на верхних ступенях *классификации* выбирают наименьшее основание кода.

Понятие унифицированной системы документации

Основной компонентой *внемашинного информационного обеспечения ИС* является *система документации*, применяемая в процессе управления экономическим объектом. Под документом понимается определенная совокупность сведений, используемая при решении технико-экономических задач, расположенная на материальном носителе в соответствии с установленной формой.

Система документации — это совокупность взаимосвязанных форм документов, регулярно используемых в процессе управления экономическим объектом. Отличительной особенностью системы экономической документации является большое разнообразие видов документов.

Существующие *системы документации*, характерные для неавтоматизированных ИС, отличаются большим количеством разных типов форм документов, большим объемом потоков документов и их запутанностью, дублированием информации в документах и работ по их обработке и, как следствие, низкой достоверностью получаемых

результатов. Для того чтобы упростить *систему документации*, используют следующие два подхода:

- проведение унификации и стандартизации документов;
- введение безбумажной технологии, основанной на использовании электронных документов и новых информационных технологий их обработки.

Унификация документов выполняется путем введения единых форм документов. Таким образом, вводится единообразие в наименования показателей, единиц измерения и терминов, в результате чего получается унифицированная *система документации*.

Унифицированная *система документации* (УСД) — это рационально организованный комплекс взаимосвязанных документов, который отвечает единым правилам и требованиям и содержит информацию, необходимую для управления некоторым экономическим объектом. По уровням управления, они делятся на межотраслевые *системы документации*, отраслевые и *системы документации* локального уровня, т. е. обязательные для использования в рамках предприятий или организаций.

Любой тип УСД должен удовлетворять следующим **требованиям**:

- документы, входящие в состав УСД, должны разрабатываться с учетом их использования в системе взаимосвязанных ЭИС;
- УСД должна содержать полную информацию, необходимую для оптимального управления тем объектом, для которого разрабатывается эта система;
- УСД должна быть ориентирована на использование средств вычислительной техники для сбора, обработки и передачи информации;
- УСД должна обеспечить информационную совместимость ЭИС различных уровней;
- все документы, входящие в состав разрабатываемой УСД, и все реквизиты-признаки в них должны быть закодированы с использованием международных, общесистемных или локальных *классификаторов*.

Внутримашинное информационное обеспечение

Внутримашинное информационное обеспечение включает макеты (экранные формы) для ввода первичных данных в ЭВМ или вывода результатной информации, и структуры *информационной базы*: входных, выходных файлов, базы данных.

Проектирование экранных форм электронных документов

Под *электронными формами документов* понимается не изображение бумажного документа, а изначально электронная (безбумажная) технология работы; она предполагает появление бумажной формы только в качестве твердой копии документа.

Электронная форма документа (ЭД) — это страница с пустыми полями, оставленными для заполнения пользователем. Формы могут допускать различный тип входной информации и содержать командные кнопки, переключатели, выпадающие меню или списки для выбора.

Создание форм электронных документов требует использования специального программного обеспечения. На [рис. 9.6](#) приведены основные типовые элементы электронного документа, использование которых предусмотрено в большинстве программ автоматизации проектирования электронных документов.

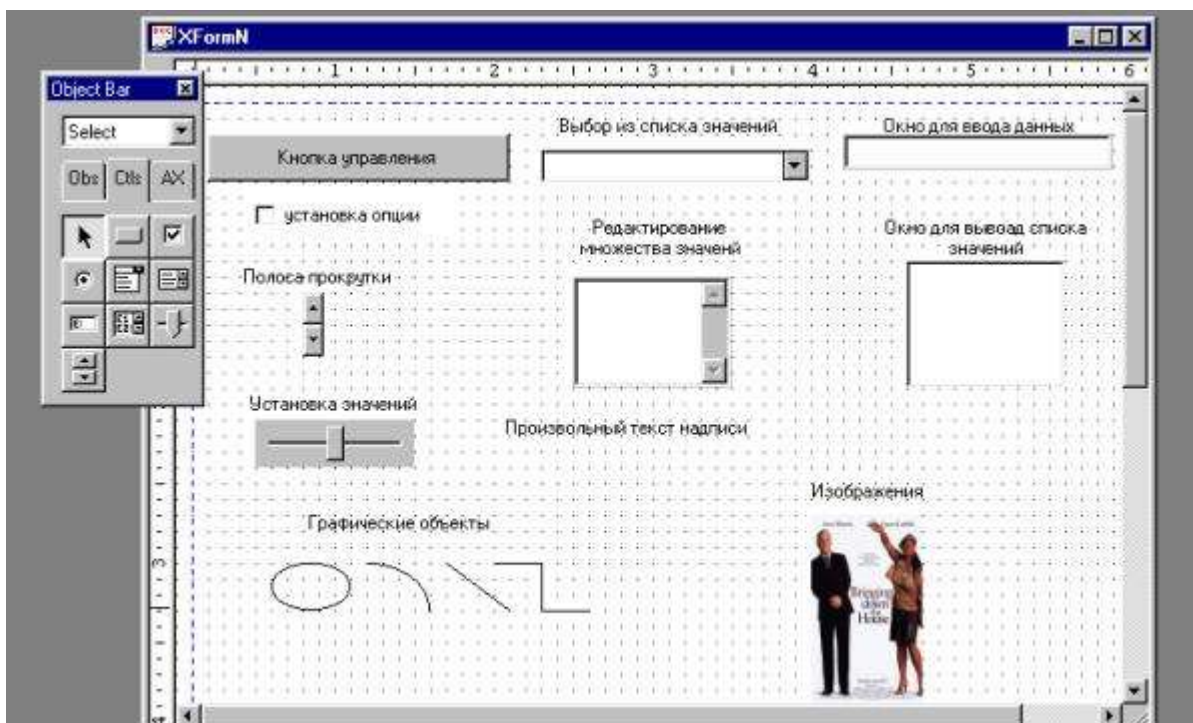


Рис. 9.6. Элементы электронного документа

К недостаткам электронных документов можно отнести неполную юридическую проработку процесса их утверждения или подписания.

Технология обработки электронных документов требует использования специализированного программного обеспечения — программ управления документооборотом, которые зачастую встраиваются в корпоративные ИС.

Проектирование форм электронных документов, т.е. создание шаблона формы с помощью программного обеспечения проектирования форм, обычно включает в себя выполнение следующих шагов:

- **создание структуры ЭД** — подготовка внешнего вида с помощью графических средств проектирования;
- **определение содержания формы ЭД**, т.е. выбор способов, которыми будут заполняться поля. Поля могут быть заполнены вручную или посредством выбора значений из какого-либо списка, меню, базы данных;
- **определения перечня макетов экранных форм** — по каждой задаче проектировщик анализирует "постановку" каждой задачи, в которой приводятся перечни используемых входных документов с оперативной и постоянной информацией и документов с результатной информацией;
- **определение содержания макетов** — выполняется на основе анализа состава реквизитов первичных документов с постоянной и оперативной информацией и результатных документов.

Работа заканчивается программированием разработанных макетов экранных форм и их апробацией.

Информационная база и способы ее организации

Основной частью внутримашинного информационного обеспечения является *информационная база*. Информационная база (ИБ) — это совокупность данных, организованная определенным способом и хранимая в памяти вычислительной системы

в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач.

Все файлы ИБ можно классифицировать по следующим признакам:

- **по этапам обработки** (входные, базовые, результатные);
- **по типу носителя** (на промежуточных носителях — гибких магнитных дисках и магнитных лентах и на основных носителях — жестких магнитных дисках, магнитооптических дисках и др.);
- **по составу информации** (файлы с оперативной информацией и файлы с постоянной информацией);
- **по назначению** (по типу функциональных подсистем);
- **по типу логической организации** (файлы с линейной и иерархической структурой записи, реляционные, табличные);
- **по способу физической организации** (файлы с последовательным, индексным и прямым способом доступа).

Входные файлы создаются с первичных документов для ввода данных или обновления базовых файлов.

Файлы с **результатной информацией** предназначены для вывода ее на печать или передачи по каналам связи и не подлежат долговременному хранению.

К числу **базовых файлов**, хранящихся в *информационной базе*, относят основные, рабочие, промежуточные, служебные и архивные файлы.

Основные файлы должны иметь однородную структуру записей и могут содержать записи с оперативной и условно-постоянной информацией. **Оперативные файлы** могут создаваться на базе одного или нескольких входных файлов и отражать информацию одного или нескольких первичных документов. **Файлы с условно-постоянной информацией** могут содержать справочную, расценочную, табличную и другие виды информации, изменяющейся в течение года не более чем на 40%, а следовательно, имеющие коэффициент стабильности не менее 0,6.

Файлы **со справочной информацией** должны отражать все характеристики элементов материального производства (материалы, сырье, основные фонды, трудовые ресурсы и т.п.). Как правило, справочники содержат информацию *классификаторов* и дополнительные сведения об элементах Материальной сферы, например о ценах. Нормативно-расценочные файлы должны содержать данные о нормах расхода и расценках на выполнение операций и услуг. Табличные файлы содержат сведения об экономических показателях, считающихся постоянными в течение длительного времени (например, процент удержания, отчисления и пр.). Плановые файлы содержат плановые показатели, хранящиеся весь плановый период.

Рабочие файлы создаются для решения конкретных задач на базе основных файлов путем выборки части информации из нескольких основных файлов с целью сокращения времени обработки данных.

Промежуточные файлы отличаются от рабочих файлов тем, что они образуются в результате решения экономических задач, подвергаются хранению с целью дальнейшего использования для решения других задач. Эти файлы, так же как и рабочие файлы, при высокой частоте обращений могут быть также переведены в категорию основных файлов.

Служебные файлы предназначены для ускорения поиска информации в основных файлах и включают в себя справочники, индексные файлы и каталоги.

Архивные файлы содержат ретроспективные данные из основных файлов, которые используются для решения аналитических, например прогнозных, задач. Архивные данные могут также использоваться для восстановления *информационной базы* при разрушениях.

Организация хранения файлов в *информационной базе* должна отвечать следующим требованиям:

- полнота хранимой информации для выполнения всех функций управления и решения экономических задач;
- целостность хранимой информации, т. е. обеспечение непротиворечивости данных при вводе информации в ИБ;
- своевременность и одновременность обновления данных во всех копиях данных;
- гибкость системы, т.е. адаптируемость ИБ к изменяющимся информационным потребностям;
- реализуемость системы, обеспечивающая требуемую степень сложности структуры ИБ;
- релевантность ИБ, под которой подразумевается способность системы осуществлять поиск и выдавать информацию, точно соответствующую запросам пользователей;
- удобство языкового интерфейса, позволяющее быстро формулировать запрос к ИБ;
- разграничение прав доступа, т.е. определение для каждого пользователя доступных типов записей, полей, файлов и видов операций над ними.

Существуют следующие **способы организации ИБ**: совокупность локальных файлов, поддерживаемых функциональными пакетами прикладных программ, и интегрированная база данных, основывающаяся на использовании универсальных программных средств загрузки, хранения, поиска и ведения данных, т.е. системы управления базами данных (СУБД).

Локальные файлы вследствие специализации структуры данных под задачи обеспечивают, как правило, более быстрое время обработки данных. Однако недостатки организации локальных файлов, связанные с большим дублированием данных в информационной системе и, как следствие, несогласованностью данных в разных приложениях, а также негибкостью доступа к информации, перекрывают указанные преимущества. Поэтому организация локальных файлов может применяться только в специализированных приложениях, требующих очень высокой скорости реакции при импорте необходимых данных.

Интегрированная ИБ, т.е. база данных (БД) — это совокупность взаимосвязанных, хранящихся вместе данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для множества приложений.

Централизация управления данными с помощью СУБД обеспечивает совместимость этих данных, уменьшение синтаксической и семантической избыточности, соответствие данных реальному состоянию объекта, разделение хранения данных между пользователями и возможность подключения новых пользователей. Но централизация управления и интеграция данных приводят к проблемам другого характера: необходимости усиления контроля вводимых данных, необходимости обеспечения соглашения между пользователями по поводу состава и структуры данных, разграничения доступа и секретности данных.

Основными способами организации БД являются создание централизованных и распределенных БД. Основным критерием выбора способа организации ИБ является достижение минимальных трудовых и стоимостных затрат на проектирование структуры ИБ, программного обеспечения системы ведения файлов, а также на перепроектирование ИБ при возникновении новых задач.

Моделирование данных

Одной из основных частей **информационного обеспечения** является **информационная база**. Как было определено выше (см. лекцию 9), информационная база (ИБ) представляет собой совокупность данных, организованную определенным способом и хранимую в памяти вычислительной системы в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач. Разработка БД выполняется с помощью моделирования данных. **Цель моделирования данных** 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. Наиболее распространенным **средством моделирования** данных являются **диаграммы "сущность-связь"** (ERD). С помощью ERD осуществляется детализация накопителей данных DFD – диаграммы, а также документируются информационные аспекты бизнес-системы, включая идентификацию объектов, важных для предметной области (*сущностей*), свойств этих объектов (*атрибутов*) и их *связей* с другими объектами (отношений).

Базовые понятия ERD

Сущность (Entity) — множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, идей, предметов и др.), обладающих общими *атрибутами* или характеристиками. Любой объект системы может быть представлен только одной *сущностью*, которая должна быть уникально идентифицирована. При этом имя *сущности* должно отражать тип или класс объекта, а не его конкретный экземпляр (например, АЭРОПОРТ, а не ВНУКОВО).

Каждая *сущность* должна обладать уникальным **идентификатором**. Каждый экземпляр *сущности* должен однозначно идентифицироваться и отличаться от всех других экземпляров данного *типа сущности*. Каждая *сущность* должна обладать некоторыми свойствами:

- иметь уникальное имя; к одному и тому же имени должна всегда применяться одна и та же интерпретация; одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- иметь один или несколько *атрибутов*, которые либо принадлежат *сущности*, либо наследуются через *связь*;
- иметь один или несколько *атрибутов*, которые однозначно идентифицируют каждый экземпляр *сущности*.

Каждая *сущность* может обладать любым количеством *связей* с другими *сущностями* модели.

Связь (Relationship) — поименованная ассоциация между двумя *сущностями*, значимая для рассматриваемой предметной области. *Связь* — это ассоциация между *сущностями*, при которой каждый экземпляр одной *сущности* ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй *сущности*, и наоборот.

Атрибут (Attribute) — любая характеристика *сущности*, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния *сущности*. *Атрибут* представляет тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, предметов и т.д.). Экземпляр *атрибута* — это определенная характеристика отдельного элемента множества. **Экземпляр атрибута** определяется типом характеристики и ее значением,

называемым **значением атрибута**. На диаграмме "сущность-связь" *атрибуты* ассоциируются с конкретными *сущностями*. Таким образом, экземпляр *сущности* должен обладать единственным определенным значением для ассоциированного *атрибута*.

Метод IDEFI

Наиболее распространенными методами для построения ERD-диаграмм являются метод Баркера и метод IDEFI.

Метод Баркера основан на нотации, предложенной автором, и используется в case-средстве Oracle Designer.

Метод IDEFI основан на подходе Чена и позволяет построить *модель данных*, эквивалентную реляционной модели в третьей нормальной форме. На основе совершенствования метода IDEFI создана его новая версия — метод IDEFIX, разработанный с учетом таких требований, как простота для изучения и возможность автоматизации. IDEFIX-диаграммы используются в ряде распространенных CASE-средств (в частности, ERwin, Design/IDEF).

В методе IDEFIX *сущность* является независимой от идентификаторов или просто независимой, если каждый экземпляр *сущности* может быть однозначно идентифицирован без определения его отношений с другими *сущностями*. *Сущность* называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра *сущности* зависит от его отношения к другой *сущности* ([рис. 10.1](#), [10.2](#)).

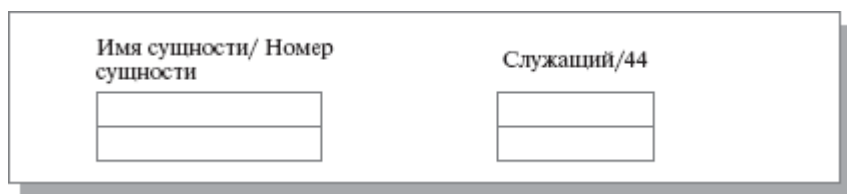


Рис. 10.1. Независимые от идентификации сущности

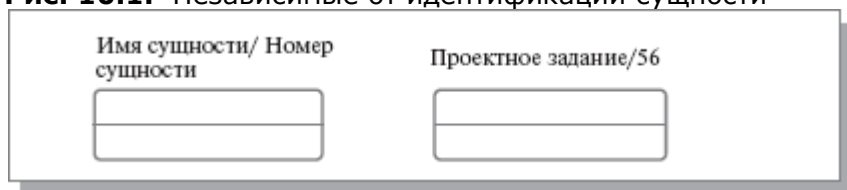


Рис. 10.2. Зависимые от идентификации сущности

Каждой *сущности* присваиваются уникальные имя и номер, разделяемые косой чертой "/" и помещаемые над блоком.

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может порождать каждый экземпляр сущности-родителя). В IDEFIX могут быть выражены следующие *мощности связей*:

- каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;

- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей *связью* с сущностью-родителем, то *связь* называется идентифицирующей, в противном случае — неидентифицирующей.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка ([рис. 10.3](#)). *Мощность связей* может принимать следующие значения: N — ноль, один или более, Z — ноль или один, P — один или более. По умолчанию *мощность связей* принимается равной N.

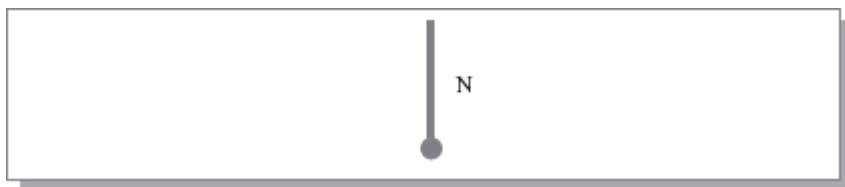


Рис. 10.3. Графическое изображение мощности связи

Идентифицирующая *связь* между сущностью-родителем и сущностью-потомком изображается сплошной линией. Сущность-потомок в идентифицирующей *связи* является зависимой от идентификатора *сущностью*. Сущность-родитель в идентифицирующей *связи* может быть как независимой, так и зависимой от идентификатора *сущностью* (это определяется ее *связями* с другими *сущностями*).

Пунктирная линия изображает неидентифицирующую *связь* ([рис. 10.4](#)). Сущность-потомок в неидентифицирующей *связи* будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей *связи*.

Атрибуты изображаются в виде списка имен внутри блока *сущности*. *Атрибуты*, определяющие *первичный ключ*, размещаются наверху списка и отделяются от других *атрибутов* горизонтальной чертой ([рис. 10.4](#)).

Сущности могут иметь также **внешние ключи** (Foreign Key), которые могут использоваться в качестве части или целого *первичного ключа* или неключевого *атрибута*. Для обозначения внешнего ключа внутрь блока *сущности* помещают имена *атрибутов*, после которых следуют буквы FK в скобках ([рис. 10.4](#)).

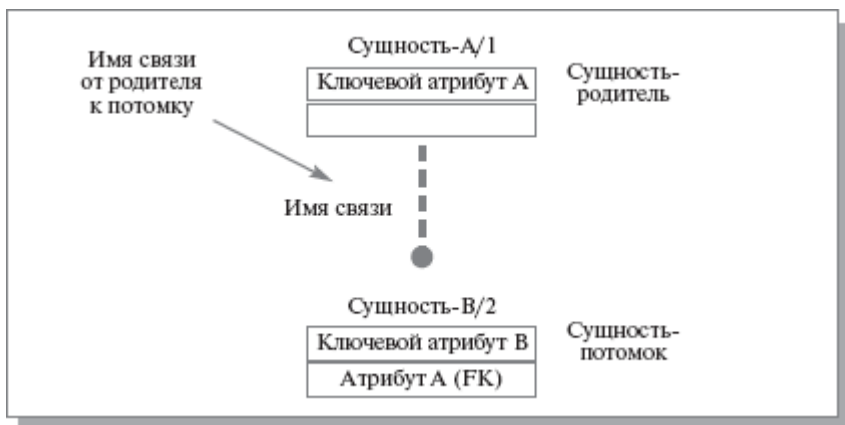


Рис. 10.4. Неидентифицирующая связь

Отображение модели данных в инструментальном средстве ERwin

ERwin имеет два уровня *представления* модели — логический и физический.

Логический уровень — это абстрактный взгляд на данные, когда 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*. 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 обо 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.

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, используя предложение — ее можно назвать 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 дать 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 документировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

Масштабирование

Создание *модели данных*, как правило, начинается с разработки 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 создать 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 переход от файл-серверной к клиент-серверной ИС. Однако, 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.

Для переключения между логической и *физической моделью данных* служит список выбора в центральной части панели инструментов ERwin ([рис. 10.5](#)).

Если при переключении физической модели еще не существует, она будет создана автоматически.

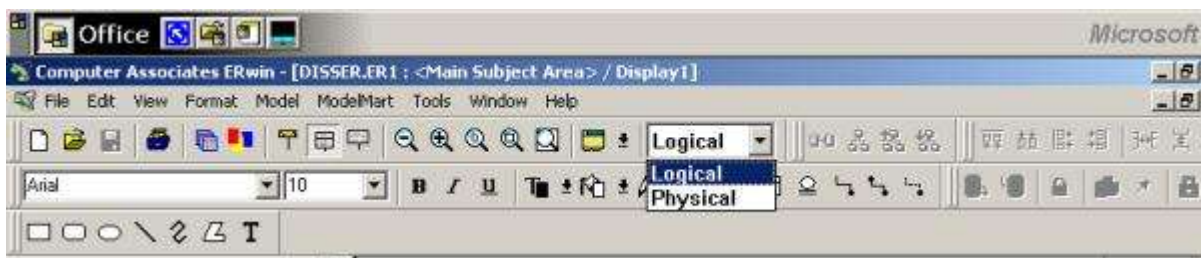


Рис. 10.5. Переключение между логической и физической моделью

Интерфейс ERwin. Уровни отображения модели

Интерфейс выполнен в стиле Windows-приложений, достаточно прост и интуитивно понятен. Рассмотрим кратко основные функции ERwin по отображению модели.

Каждому уровню отображения модели соответствует своя палитра инструментов. На **логическом уровне** палитра инструментов имеет следующие кнопки:

- кнопку указателя (режим мыши) — в этом режиме можно установить фокус на каком-либо объекте модели;
- кнопку внесения *сущности*;
- кнопку категории (категория, или категориальная *связь*, — специальный тип *связи* между *сущностями*, которая будет рассмотрена ниже);
- кнопку внесения текстового блока;
- кнопку перенесения *атрибутов* внутри *сущностей* и между ними;
- кнопки создания *связей*: идентифицирующую, "многие-ко-многим" и неидентифицирующую.

На **физическом уровне** палитра инструментов имеет:

- вместо кнопки категорий — кнопку внесения *представлений* (view);
- вместо кнопки *связи* "многие-ко-многим" — кнопку *связей представлений*.

Для создания *моделей данных* в ERwin можно использовать две нотации: IDEFIX и IE (Information Engineering). В дальнейшем будет рассматриваться нотация IDEFIX.

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. Переключиться на другие уровни отображения можно при помощи контекстного меню, которое появляется, если "кликнуть" по любому месту диаграммы, не занятому объектами модели. В контекстном меню следует выбрать пункт Display Level ([рис. 10.6](#)) и затем — необходимый уровень отображения.

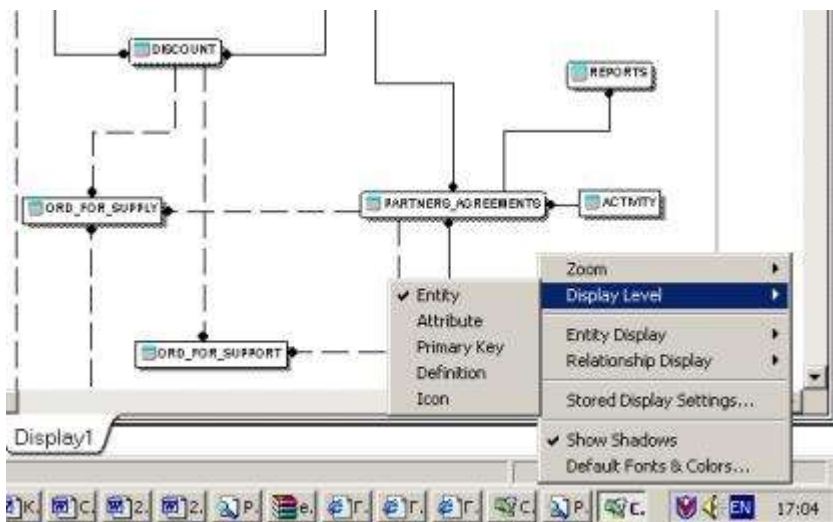


Рис. 10.6. Выбор уровней отображения диаграммы

Создание логической модели данных

Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- *диаграмма сущность-связь* (Entity Relationship Diagram, ERD);
- *модель данных, основанная на ключах* (Key Based model, KB);
- *полная атрибутивная модель* (Fully Attributed model, FA).

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.

Модель данных, 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.

Полная атрибутивная модель — наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все *сущности*, *атрибуты* и *связи*.

Сущности и атрибуты

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*. *Сущность* можно определить **как объект, событие или концепцию, информация о которых должна сохраняться**. *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. Именованная *сущность* в единственном числе облегчает в дальнейшем чтение модели. Фактически имя *сущности* дается по имени ее экземпляра. Примером может быть *сущности 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*. Каждая *сущность* должна быть полностью определена с помощью текстового описания. Для внесения дополнительных комментариев и определений к *сущности* служат свойства, определенные пользователем (UDP). Использование (UDP) аналогично их использованию в BPwin.

Как было указано выше, каждый *атрибут* хранит информацию об **определенном свойстве сущности**, а каждый экземпляр *сущности* должен быть уникальным. *Атрибут* или группа *атрибутов*, которые идентифицируют *сущность*, называется **первичным ключом**.

Очень важно дать *атрибуту* правильное имя. *Атрибуты* должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения *атрибутов*. Например, создание в *сущности* *Сотрудник* *атрибута* *Телефоны сотрудника* противоречит требованиям нормализации, поскольку *атрибут* должен быть атомарным, т. е. не содержать множественных значений. Согласно синтаксису IDEFIX имя *атрибута* должно быть уникально в рамках модели (а не только в рамках *сущности*!). По умолчанию при попытке внесения уже существующего имени *атрибута* ERwin переименовывает его.

Каждый *атрибут* должен быть определен, при этом следует избегать циклических определений, например, когда термин 1 определяется через термин 2, термин 2 — через термин 3, а термин 3 в свою очередь — через термин 1. 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, чтобы не проводить вычисления, которые на практике могут быть сложными.

Связи

Связь является логическим соотношением между *сущностями*. Каждая *связь* должна именоваться глаголом или глагольной фразой. *Имя связи* выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. По умолчанию *имя связи* на диаграмме не показывается. На логическом уровне можно установить идентифицирующую *связь* "один-ко-многим", *связь* "многие-ко-многим" и неидентифицирующую *связь* "один-ко-многим".

В IDEFIX 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. Экземпляр зависимой *сущности* определяется только через отношение к родительской *сущности*. 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.

При установлении неидентифицирующей *связи* дочерняя *сущность* остается независимой, а *атрибуты первичного ключа* родительской *сущности* мигрируют в состав неключевых компонентов родительской *сущности*. Неидентифицирующая *связь* служит для связывания независимых *сущностей*.

Идентифицирующая *связь* показывается на диаграмме сплошной линией с жирной точкой на дочернем конце *связи*, неидентифицирующая – пунктирной (см. [рис. 10.6](#)).

Мощность связей (Cardinality) — служит для обозначения отношения числа экземпляров родительской *сущности* к числу экземпляров дочерней.

Различают четыре *типа сущности*:

- общий случай, когда одному экземпляру родительской *сущности* соответствуют 0, 1 или много экземпляров дочерней *сущности*; не помечается каким-либо символом;
- символом P помечается случай, когда одному экземпляру родительской *сущности* соответствуют 1 или много экземпляров дочерней *сущности* (исключено нулевое значение);
- символом Z помечается случай, когда одному экземпляру родительской *сущности* соответствуют 0 или 1 экземпляр дочерней *сущности* (исключены множественные значения);
- цифрой помечается случай точного соответствия, когда одному экземпляру родительской *сущности* соответствует заранее заданное число экземпляров дочерней *сущности*.

Имя связи (Verb Phrase) — фраза, характеризующая отношение между родительской и дочерней *сущностями*. Для *связи* "один-ко-многим", идентифицирующей или неидентифицирующей, достаточно указать имя, характеризующее отношение от родительской к дочерней *сущности* (Parent-to-Child). Для *связи* многие-ко-многим следует указывать имена как Parent-to-Child, так и Child-to-Parent.

Типы сущностей и иерархия наследования

Как было указано выше, *связи* определяют, является ли *сущность* независимой или зависимой. Различают несколько **типов зависимых сущностей**.

Характеристическая — зависимая дочерняя *сущность*, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской *сущности* ([рис. 10.7](#)).



Рис. 10.7. Пример характеристической сущности "Хобби"

Ассоциативная — *сущность*, связанная с несколькими родительскими *сущностями*. Такая *сущность* содержит информацию о *связях сущностей*.

Именуемая — частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

Категориальная — дочерняя сущность в иерархии наследования.

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день (постоянные служащие), и совместители. Из их общих свойств можно сформировать обобщенную сущность (родовой предок) **Сотрудник** (рис. 10.8), чтобы представить информацию, общую для всех типов служащих. Специфическая для каждого типа информация может быть расположена в категориальных сущностях (потомках) **Постоянный сотрудник** и **Совместитель**.

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи (например, если бы **Постоянный сотрудник** и **Совместитель** имели сходную по смыслу связь "работает в" с сущностью **Организация**), либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать дискриминатор — атрибут родového предка, который показывает, как отличить одну категориальную сущность от другой (атрибут **Тип** на рис. 10.8).

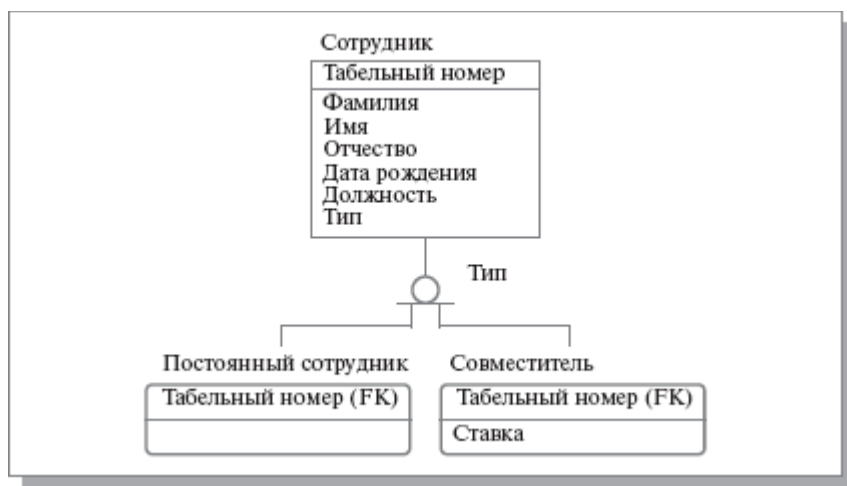


Рис. 10.8. Иерархия наследования. Неполная категория

Иерархии категорий делятся на два типа — полные и неполные. В полной категории одному экземпляру родového предка (сущность **Служащий**, рис. 10.9) обязательно соответствует экземпляр в каком-либо потомке, т. е. в примере служащий обязательно является либо совместителем, либо консультантом, либо постоянным сотрудником.



Рис. 10.9. Иерархия наследования. Полная категория

Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. На [рис. 10.8](#) показана неполная категория — сотрудник может быть не только постоянным или совместителем, но и консультантом, однако сущность **Консультант** еще не внесена в иерархию наследования.

Ключи

Как было сказано выше, каждый экземпляр сущности должен быть уникален и должен 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 (см., например, [рис. 10.9](#)).

В одной сущности могут оказаться несколько атрибутов или наборов атрибутов, претендующих на роль первичного ключа. Такие претенденты называются **потенциальными ключами** (candidate key).

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

Рассмотрим кандидатов на роль первичного ключа сущности **Сотрудник** ([рис. 10.10](#)).

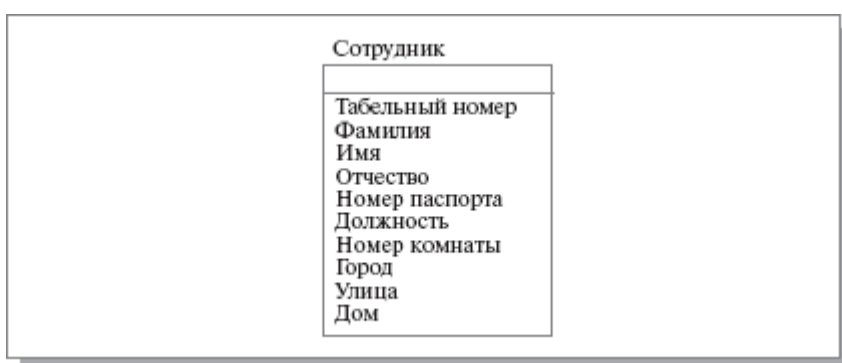


Рис. 10.10. Определение первичного ключа для сущности "Сотрудник"

Здесь можно выделить следующие *потенциальные ключи*:

1. Табельный номер;
2. Номер паспорта;
3. Фамилия + Имя + Отчество.

Для того чтобы стать первичным, *потенциальный ключ* должен удовлетворять ряду требований:

Уникальность. Два экземпляра не должны иметь одинаковых значений возможного ключа. *потенциальный ключ* № 3 (Фамилия + Имя + Отчество) является плохим кандидатом, поскольку в организации могут работать полные тезки.

Компактность. Сложный возможный ключ не должен содержать ни одного *атрибута*, удаление которого не приводило бы к утрате уникальности. Для обеспечения уникальности ключа № 3 дополним его *атрибутами* Дата рождения и Цвет волос. Если бизнес-правила говорят, что сочетания *атрибутов* Фамилия + Имя + Отчество + Дата рождения достаточно для однозначной идентификации сотрудника, то Цвет волос оказывается лишним, т. е. ключ Фамилия + Имя + Отчество + Дата рождения + Цвет волос не является компактным.

При выборе *первичного ключа* предпочтение должно отдаваться более простым ключам, т. е. ключам, содержащим меньшее количество *атрибутов*. В приведенном примере ключи № 1 и 2 предпочтительней ключа № 3.

Атрибуты ключа не должны содержать нулевых значений. Значение *атрибутов* ключа не должно меняться в течение всего времени существования экземпляра *сущности*. Сотрудница организации может выйти замуж и сменить как фамилию, так и паспорт. Поэтому ключи № 2 и 3 не подходят на роль *первичного ключа*.

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.

Нормализация данных

Нормализация данных — **процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных**. Нормализация позволяет быть уверенным, что каждый *атрибут* определен для своей *сущности*, а также значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам — формализованным требованиям к организации данных. Известны шесть нормальных форм.

На практике обычно ограничиваются приведением данных к третьей нормальной форме. Для углубленного изучения нормализации рекомендуется книга К. Дж. Дейта "Введение в системы баз данных" (Киев; М.: Диалектика, 1998).

ERwin не содержит полного алгоритма нормализации и не может проводить нормализацию автоматически, однако его возможности облегчают создание

нормализованной *модели данных*. Запрет на присвоение неуникальных имен *атрибутов* в рамках модели (при соответствующей установке опции Unique Name) облегчает соблюдение правила "один факт — в одном месте". Имена ролей *атрибутов* внешних ключей и унификация *атрибутов* также облегчают построение нормализованной модели.

Домены

Домен можно определить как совокупность значений, из которых берутся значения *атрибутов*. Каждый *атрибут* может быть определен только на одном *домене*, но на каждом *домене* может быть определено множество *атрибутов*. В понятие *домена* входит не только тип данных, но и область значений данных. Например, можно определить *домен* "Возраст" как положительное целое число и определить *атрибут* **Возраст** **сотрудника** как принадлежащий этому *домену*.

В ERwin *домен* может быть определен только один раз и использоваться как в логической, так и в физической модели.

Домены позволяют облегчить работу с данными как разработчикам на этапе проектирования, так и администраторам БД на этапе эксплуатации системы. На логическом уровне *домены* можно описать без конкретных физических свойств. На физическом уровне они автоматически получают специфические свойства, которые можно изменить вручную. Так, *домен* "Возраст" может иметь на логическом уровне тип **Number**, на физическом уровне колонкам *домена* будет присвоен тип **INTEGER**.

Каждый *домен* может быть описан, снабжен комментарием или свойством, определенным пользователем (UDP).

Создание физической модели данных

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД. Различают два уровня физической модели:

- трансформационную модель;
- модель СУБД.

Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. Данная модель позволяет проектировщикам и администраторам БД лучше представить, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель удовлетворяет требованиям к ИС.

Модель СУБД автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД.

Физический уровень *представления* модели зависит от выбранного сервера. ERwin поддерживает более 20 реляционных и нереляционных БД.

По умолчанию ERwin генерирует имена таблиц и *индексов* по шаблону на основе имен соответствующих *сущностей* и ключей логической модели, которые в дальнейшем могут быть откорректированы вручную. Имена таблиц и колонок будут сгенерированы по умолчанию на основе имен *сущностей* и *атрибутов* логической модели.

Правила валидации и значения по умолчанию

ERwin поддерживает *правила валидации* для колонок, а также значение, присваиваемое колонкам по умолчанию.

Правило валидации задает список допустимых значений для конкретной колонки и/или правила проверки допустимых значений. В список допустимых значений можно вносить новые значения. ERwin позволяет сгенерировать *правила валидации* соответственно синтаксису выбранной СУБД с учетом границ диапазона или списка значений.

Значение по умолчанию – значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. С каждой колонкой или *доменом* можно связать значение по умолчанию. Список значений можно редактировать.

После создания *правила валидации* и значения по умолчанию их можно присвоить одной или нескольким колонкам или *доменами*.

Индексы

В БД данные обычно хранятся в том порядке, в котором их ввели в таблицу. Многие реляционные СУБД имеют страничную организацию, при которой таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно. Такой способ позволяет быстро вводить новые данные, но затрудняет поиск данных.

Чтобы решить проблему поиска, СУБД используют объекты, называемые *индексами*.

Индекс содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки. Поскольку значения в *индексе* хранятся в определенном порядке, при поиске просматривать нужно значительно меньший объем данных, что существенно уменьшает время выполнения запроса. *Индекс* рекомендуется создавать для тех колонок, по которым часто производится поиск.

При генерации схемы физической БД ERwin автоматически создает *индекс* на основе *первичного ключа* каждой таблицы, а также на основе всех *альтернативных ключей* и внешних ключей, поскольку эти колонки наиболее часто используются для поиска данных. Можно отказаться от генерации *индексов* по умолчанию и создать собственные *индексы*. Для увеличения эффективности поиска администратор БД должен анализировать часто выполняемые запросы и на основе анализа создавать собственные *индексы*.

Триггеры и хранимые процедуры

Триггеры и хранимые процедуры – это именованные блоки кода SQL, которые **заранее откомпилированы и хранятся на сервере для того, чтобы быстро производить обработку запросов, валидацию данных и другие часто выполняемые функции**. Хранение и выполнение кода на сервере позволяет создавать код только один раз, а не в каждом приложении, работающем с БД. Это экономит время при написании и сопровождении программ. При этом гарантируется, что целостность данных и бизнес-правила поддерживаются независимо от того, какое именно клиентское приложение обращается к данным. *Триггеры и хранимые процедуры* не требуется пересылать по сети из клиентского приложения, что значительно снижает сетевой трафик.

Хранимой процедурой называется именованный набор предварительно откомпилированных команд SQL, который может вызываться из клиентского приложения или из другой *хранимой процедуры*.

Триггером называется процедура, которая выполняется автоматически как реакция на событие. Таким событием может быть вставка, изменение или удаление строки в существующей таблице. *Триггер* сообщает СУБД, какие действия нужно выполнить при

выполнении команд SQL INSERT, UPDATE или DELETE для обеспечения дополнительной функциональности, выполняемой на сервере.

Триггер ссылочной целостности – это особый вид *триггера*, используемый для поддержания целостности между двумя таблицами, которые связаны между собой. Если строка в одной таблице вставляется, изменяется или удаляется, то *триггер* ссылочной целостности сообщает СУБД, что нужно делать с теми строками в других таблицах, у которых значение внешнего ключа совпадает со значением *первичного ключа* вставленной строки (измененной или удаленной строки).

Для генерации *триггеров* ERwin использует механизм шаблонов – специальных скриптов, использующих макрокоманды. При генерации кода *триггера* вместо макрокоманд подставляются имена таблиц, колонок, переменные и другие фрагменты кода, соответствующие синтаксису выбранной СУБД. Шаблоны *триггеров* ссылочной целостности, генерируемые ERwin по умолчанию, можно изменять.

Для создания и редактирования *хранимых процедур* ERwin располагает специальными редакторами, аналогичными редакторам, используемым для создания *триггеров*. В отличие от *триггера* *хранимая процедура* не выполняется в ответ на какое-то событие, а вызывается из другой программы, которая передает на сервер имя процедуры. *Хранимая процедура* более гибкая, чем *триггер*, поскольку может вызывать другие *хранимые процедуры*. Ей можно передавать параметры, и она может возвращать параметры, значения и сообщения.

Проектирование хранилищ данных

В хранилища данных помещают данные, которые редко меняются. Хранилища ориентированы на выполнение аналитических запросов, обеспечивающих поддержку принятия решений для руководителей и менеджеров. При проектировании хранилищ данных необходимо выполнять следующие требования:

- хранилище должно иметь понятную для пользователей структуру данных;
- должны быть выделены статические данные, которые модифицируются по расписанию (ежедневно, еженедельно, ежеквартально);
- должны быть упрощены требования к запросам для исключения запросов, требующих множественных утверждений SQL в традиционных реляционных СУБД;
- должна обеспечиваться поддержка сложных запросов SQL, требующих обработки миллионов записей.

Как видно из этих требований, по своей структуре реляционные СУБД существенно отличаются от хранилищ данных. Нормализация данных в реляционных СУБД приводит к созданию множества связанных между собой таблиц. Выполнение сложных запросов неизбежно приводит к объединению многих таблиц, что значительно увеличивает время отклика. Проектирование хранилища данных подразумевает создание денормализованной структуры данных, ориентированных в первую очередь на высокую производительность при выполнении аналитических запросов. Нормализация делает модель хранилища слишком сложной, затрудняет ее понимание и снижает скорость выполнения запроса. Для эффективного проектирования хранилищ данных ERwin использует размерную модель – методологию проектирования, предназначенную специально для разработки хранилищ данных. Размерное моделирование сходно с моделированием *связей* и *сущностей* для реляционной модели, но имеет другую цель. Реляционная модель акцентируется на целостности и эффективности ввода данных. Размерная модель ориентирована в первую очередь на выполнение сложных запросов

В размерном моделировании принят стандарт модели, называемый схемой "звезда", которая обеспечивает высокую скорость выполнения запроса посредством денормализации и разделения данных. Невозможно создать универсальную структуру

данных, обеспечивающую высокую скорость обработки любого запроса, поэтому схема "звезда" строится для обеспечения наивысшей производительности при выполнении самого важного запроса (или группы запросов).

Схема "звезда" обычно содержит одну большую таблицу, называемую таблицей факта, помещенную в центре. Ее окружают меньшие таблицы, называемые таблицами размерности, которые связаны с таблицей факта радиальными *связями*.

Для создания БД со схемой "звезда" необходимо проанализировать бизнес-правила предметной области для выяснения центрального запроса. Данные, обеспечивающие выполнение этого запроса, должны быть помещены в центральную таблицу. При проектировании хранилища важно определить источник данных, метод, которым данные извлекаются, преобразуются и фильтруются, прежде чем они импортируются в хранилище. Знания об источнике данных позволяют поддерживать регулярное обновление и проверку качества данных.

Вычисление размера БД

ERwin позволяет рассчитать приблизительный размер БД в целом, а также таблиц, *индексов* и других объектов через определенный период времени после начала эксплуатации ИС. Расчет строится на основе следующих параметров: начальное количество строк; максимальное количество строк; прирост количества строк в месяц. Результаты расчетов сводятся в отчет.

Прямое и обратное проектирование

Прямым проектированием называется процесс генерации физической схемы БД из логической модели. При генерации физической схемы ERwin включает *триггеры* ссылочной целостности, *хранимые процедуры*, *индексы*, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД.

Обратным проектированием называется процесс генерации логической модели из физической БД. *Обратное проектирование* позволяет конвертировать БД из одной СУБД в другую. После создания логической модели БД путем *обратного проектирования* можно переключиться на другой сервер и произвести *прямое проектирование*.

Кроме режима прямого и *обратного проектирования* программа обеспечивает синхронизацию между логической моделью и системным каталогом СУБД на протяжении всего жизненного цикла создания ИС.

Генерация кода клиентской части с помощью ERwin

Расширенные атрибуты

ERwin поддерживает не только проектирование сервера БД, но и автоматическую генерацию клиентского приложения в средах разработки MS Visual Basic и Power Builder. Технология генерации состоит в том, что на этапе разработки *физической модели данных* каждой колонке присваиваются расширенные *атрибуты*, содержащие информацию о свойствах объектов клиентского приложения (в том числе и визуальных), которые будут отображать информацию, хранящуюся в соответствующей колонке. Эта информация записывается в файл модели. На основе информации, содержащейся в расширенных *атрибутах*, генерируются экранные формы. Полученный код может быть откомпилирован и выполнен без дополнительного ручного кодирования.

Каждой колонке в модели ERwin можно задать предварительно описанные и именованные свойства:

- *правила валидации* (проверка значений);
- начальные значения, устанавливаемые по умолчанию;
- стиль визуального объекта (например, радиокнопка, поле ввода и др.);
- формат изображения.

Для описания каждого свойства ERwin содержит соответствующие редакторы.

Генерация кода в Visual Basic

ERwin поддерживает генерацию кода в Visual Basic версий 4.0 и 5.0. В качестве источника информации при генерации форм служит модель ERwin. С помощью ERwin можно одновременно описывать как клиентскую часть (объекты, отображающие данные на экране), так и сервер БД (процедуры и *триггеры*), тем самым оптимально распределяя функциональность ИС между клиентской и серверной частью. Компонент ERwin Form Wizard автоматически проектирует формы с дочерними объектами – кнопками, списками, полями, радиокнопками и т. д., используя расширенные *атрибуты*. Совместное использование ERwin и Visual Basic позволяет сократить жизненный цикл разработки ИС путем употребления для каждой задачи наиболее эффективного инструмента. Visual Basic может быть использован для проектирования визуального интерфейса, а ERwin – для разработки физической и *логической модели данных* с последующей генерацией системного каталога сервера. Если БД уже существует, то с помощью ERwin можно провести *обратное проектирование*, полученную модель дополнить расширенными *атрибутами* и сгенерировать клиентское приложение.

Создание отчетов

Для генерации отчетов в ERwin имеется простой и эффективный инструмент – Report Browser. По умолчанию Report Browser содержит предварительно определенные отчеты, позволяющие наглядно представить информацию об основных объектах *модели данных* – как логической, так и физической. С помощью специального редактора существующие отчеты можно изменить или создать собственный отчет. Каждый отчет может быть настроен индивидуально, данные в нем могут быть отсортированы и отфильтрованы. Browser Report позволяет сохранять результаты выполнения отчетов, печатать и экспортировать их в распространенные форматы.

Генерация словарей

Для управления большими проектами ERwin имеет специальный инструмент – ERwin Dictionary, который обеспечивает коллективную работу над диаграммами и позволяет сохранять и документировать различные версии *моделей данных*. ERwin Dictionary представляет собой специальную БД, которая позволяет решить проблемы документирования и хранения моделей, однако не полностью отвечает требованиям многопользовательской работы.

Лекция: Унифицированный язык визуального моделирования Unified Modeling Language (UML)

Существует множество технологий и инструментальных средств, с помощью которых можно реализовать в некотором смысле оптимальный проект ИС, начиная с этапа анализа и заканчивая созданием программного кода системы. В большинстве случаев эти технологии предъявляют весьма жесткие требования к процессу разработки и используемым ресурсам, а попытки трансформировать их под конкретные проекты оказываются безуспешными. Эти технологии представлены CASE-средствами верхнего уровня или CASE-средствами полного жизненного цикла (upper CASE tools или full life-cycle CASE tools). Они не позволяют оптимизировать деятельность на уровне отдельных элементов проекта, и, как следствие, многие разработчики перешли на так называемые CASE-средства нижнего уровня (lower CASE tools). Однако они столкнулись с новой проблемой — проблемой организации взаимодействия между различными командами, реализующими проект.

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью *UML* жизненный цикл информационных систем, и, одновременно, *UML* является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

Мощный толчок к разработке этого направления информационных технологий дало распространение *объектно-ориентированных* языков программирования в конце 1980-х — начале 1990-х годов. Пользователям хотелось получить единый язык моделирования, который объединил бы в себе всю мощь *объектно-ориентированного* подхода и давал бы четкую модель системы, отражающую все ее значимые стороны. К середине девяностых явными лидерами в этой области стали методы Booch (Grady Booch), OMT-2 (Jim Rumbaugh), OOSE — Object-Oriented Software Engineering (Ivar Jacobson). Однако эти три метода имели свои сильные и слабые стороны: OOSE был лучшим на стадии анализа проблемной области и анализа требований к системе, OMT-2 был наиболее предпочтителен на стадиях анализа и разработки информационных систем, Booch лучше всего подходил для стадий дизайна и разработки.

Все шло к созданию единого языка, который объединял бы сильные стороны известных методов и обеспечивал наилучшую поддержку моделирования. Таким языком оказался *UML*.

Создание *UML* началось в октябре 1994 г., когда Джим Рамбо и Гради Буч из Rational Software Corporation стали работать над объединением своих методов OMT и Booch. Осенью 1995 г. увидела свет первая черновая версия объединенной методологии, которую они называли Unified Method 0.8. После присоединения в конце 1995 г. к Rational Software Corporation Айвара Якобсона и его фирмы Objectory, усилия трех создателей наиболее распространенных *объектно-ориентированных* методологий были объединены и направлены на создание *UML*.

В настоящее время консорциум пользователей *UML* Partners включает в себя представителей таких грандов информационных технологий, как Rational Software, Microsoft, IBM, Hewlett-Packard, Oracle, DEC, Unisys, IntelliCorp, Platinum Technology.

UML представляет собой *объектно-ориентированный* язык моделирования, обладающий следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика ИС, различных групп разработчиков ИС;

- содержит механизмы расширения и специализации базовых концепций языка.

UML — это стандартная нотация визуального моделирования программных систем, принятая консорциумом Object Managing Group (OMG) осенью 1997 г., и на сегодняшний день она поддерживается многими *объектно-ориентированными CASE-продуктами*.

UML включает внутренний набор средств моделирования (модулей?) ("ядро"), которые сейчас приняты во многих методах и средствах моделирования. Эти концепции необходимы в большинстве прикладных задач, хотя не каждая концепция необходима в каждой части каждого приложения. Пользователям языка предоставлены возможности:

- строить модели на основе средств ядра, без использования механизмов расширения для большинства типовых приложений;
- добавлять при необходимости новые элементы и условные обозначения, если они не входят в ядро, или специализировать компоненты, систему условных обозначений (нотацию) и ограничения для конкретных предметных областей.

Синтаксис и семантика основных объектов UML

Классы

Классы — это базовые элементы любой *объектно-ориентированной системы*. *Классы* представляют собой описание совокупностей однородных объектов с присущими им свойствами — атрибутами, операциями, отношениями и семантикой.

В рамках модели каждому *классу* присваивается уникальное имя, отличающее его от других *классов*. Если используется составное имя (в начале имени добавляется имя пакета, куда входит *класс*), то имя *класса* должно быть уникальным в пакете.

Атрибут — это свойство *класса*, которое может принимать множество значений. Множество допустимых значений атрибута образует домен. Атрибут имеет имя и отражает некоторое свойство моделируемой сущности, общее для всех объектов данного *класса*. *Класс* может иметь произвольное количество атрибутов.

Операция — реализация функции, которую можно запросить у любого объекта *класса*. Операция показывает, что можно сделать с объектом. Исполнение операции часто связано с обработкой и изменением значений атрибутов объекта, а также изменением состояния объекта.

На [рис. 11.1](#) приведено графическое изображение *класса "Заказ"* в нотации *UML*.



Рис. 11.1. Изображение класса в UML

Синтаксис *UML* для свойств *классов* (в отдельных программных средствах, например, в IBM *UML Modeler*, порядок записи параметров может быть иным):

<признак видимости> <имя атрибута> : <тип данных>
= <значение по умолчанию>
<признак видимости> <имя операции> <(список аргументов)>

Видимость свойства указывает на возможность его использования другими *классами*. Один *класс* может "видеть" другой, если тот находится в области действия первого и между ними существует явное или неявное отношение. В языке *UML* определены три уровня видимости:

- **public** (общий) — любой внешний *класс*, который "видит" данный, может пользоваться его общими свойствами. Обозначаются знаком "+" перед именем атрибута или операции;
- **protected** (защищенный) — только любой потомок данного *класса* может пользоваться его защищенными свойствами. Обозначаются знаком "#";
- **private** (закрытый) — только данный *класс* может пользоваться этими свойствами. Обозначаются символом "-" .

Еще одной важной характеристикой атрибутов и операций *классов* является область действия. Область действия свойства указывает, будет ли оно проявлять себя по-разному в каждом экземпляре *класса*, или одно и то же значение свойства будет совместно использоваться всеми экземплярами:

- **instance** (экземпляр) — у каждого экземпляра *класса* есть собственное значение данного свойства;
- **classifier** (классификатор) — все экземпляры совместно используют общее значение данного свойства (выделяется на диаграммах подчеркиванием).

Возможное количество экземпляров *класса* называется его кратностью. В *UML* можно определять следующие разновидности *классов*:

- не содержащие ни одного экземпляра — тогда *класс* становится служебным (**Abstract**);
- содержащие ровно один экземпляр (**Singleton**);
- содержащие заданное число экземпляров;
- содержащие произвольное число экземпляров.

Принципиальное назначение *классов* характеризуют стереотипы. Это, фактически, классификация объектов на высоком уровне, позволяющая определить некоторые основные свойства объекта (пример стереотипа — *класс* "действующее лицо"). Механизм стереотипов является также средством расширения словаря *UML* за счет создания на основе существующих блоков языка новых, специфичных для решения конкретной проблемы.

Диаграммы классов

Классы в *UML* изображаются на **диаграммах классов**, которые позволяют описать систему в статическом состоянии — определить типы объектов системы и различного рода статические связи между ними.

Классы отображают типы объектов системы.

Между *классами* возможны различные отношения, представленные на [рис. 11.2](#):

- зависимости, которые описывают существующие между *классами* отношения использования;
- обобщения, связывающие обобщенные *классы* со специализированными;

- ассоциации, отражающие структурные отношения между объектами *классов*.



Рис. 11.2. Отображение связей между классами

Зависимостью называется отношение использования, согласно которому изменение в спецификации одного элемента (например, класса "товар") может повлиять на использующий его элемент (класс "строка заказа"). Часто зависимости показывают, что один класс использует другой в качестве аргумента.

Обобщение — это отношение между общей сущностью (родителем — класс "клиент") и ее конкретным воплощением (потомком — классы "корпоративный клиент" или "частный клиент"). Объекты класса-потомка могут использоваться всюду, где встречаются объекты класса-родителя, но не наоборот. При этом он наследует свойства родителя (его атрибуты и операции). Операция потомка с той же сигнатурой, что и у родителя, замещает операцию родителя; это свойство называют полиморфизмом. Класс, у которого нет родителей, но есть потомки, называется корневым. Класс, у которого нет потомков, называется листовым.

Ассоциация — это отношение, показывающее, что объекты одного типа неким образом связаны с объектами другого типа ("клиент" может сделать "заказ"). Если между двумя классами определена ассоциация, то можно перемещаться от объектов одного класса к объектам другого. При необходимости направление навигации может задаваться стрелкой. Допускается задание ассоциаций на одном классе. В этом случае оба конца ассоциации относятся к одному и тому же классу. Это означает, что с объектом некоторого класса можно связать другие объекты из того же класса. Ассоциации может быть присвоено имя, описывающее семантику отношений. Каждая ассоциация имеет две роли, которые могут быть отражены на диаграмме (рис. 11.3). Роль ассоциации обладает свойством множественности, которое показывает, сколько соответствующих объектов может участвовать в данной связи.

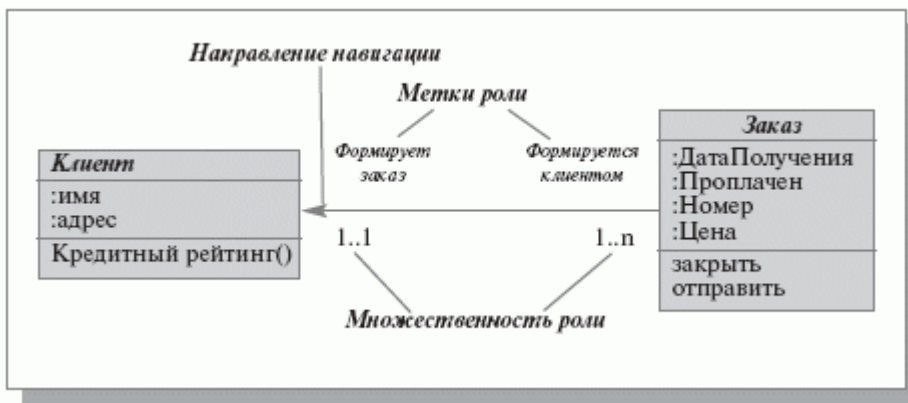


Рис. 11.3. Свойства ассоциации

[рис. 11.3](#) иллюстрирует модель формирования заказа. Каждый заказ может быть создан единственным клиентом (множественность роли 1..1). Каждый клиент может создать один и более заказов (множественность роли 1..n). Направление навигации показывает, что каждый заказ должен быть **"привязан"** к определенному клиенту.

Такого рода ассоциация является простой и отражает отношение между равноправными сущностями, когда оба *класса* находятся на одном концептуальном уровне и ни один не является более важным, чем другой. Если приходится моделировать отношение типа "часть-целое", то используется специальный тип ассоциации — агрегирование. В такой ассоциации один из *классов* имеет более высокий ранг (целое — *класс* "**заказ**", [рис. 11.2](#)) и состоит из нескольких меньших по рангу *классов* (частей — *класс* "**строка заказа**"). В UML используется и более сильная разновидность агрегации — композиция, в которой объект-часть может принадлежать только единственному целому. В композиции жизненный цикл частей и целого совпадают, любое удаление целого обязательно захватывает и его части.

Для ассоциаций можно задавать атрибуты и операции, создавая по обычным правилам UML *классы* ассоциаций.

Диаграммы использования

Диаграммы использования описывают функциональность ИС, которая будет видна пользователям системы. "Каждая функциональность" изображается в виде "прецедентов использования" (use case) или просто прецедентов. Прецедент — это типичное взаимодействие пользователя с системой, которое при этом:

- описывает видимую пользователем функцию,
- может представлять различные уровни детализации,
- обеспечивает достижение конкретной цели, важной для пользователя.

Прецедент обозначается на диаграмме овалом, связанным с пользователями, которых принято называть действующими лицами (актеры, actors). Действующие лица используют систему (или используются системой) в данном прецеденте. Действующее лицо выполняет некоторую роль в данном прецеденте. На диаграмме изображается только одно действующее лицо, однако реальных пользователей, выступающих в данной роли по отношению к ИС, может быть много. Список всех прецедентов фактически определяет функциональные требования к ИС, которые лежат в основе разработки технического задания на создание системы.

На **диаграммах прецедентов**, кроме связей между действующими лицами и прецедентами, возможно использование еще двух видов связей между прецедентами: "использование" и "расширение" ([рис. 11.4](#)). Связь типа "расширение" применяется,

когда один прецедент подобен другому, но несет несколько большую функциональную нагрузку. Ее следует применять при описании изменений в нормальном поведении системы. Связь типа "использование" позволяет выделить некий фрагмент поведения системы и включать его в различные прецеденты без повторного описания.

На [рис. 11.4](#) показано, что при исполнении прецедента "формирование заказа" возможно использование информации из предыдущего заказа, что позволит не вводить все необходимые данные. А при исполнении прецедентов "оценить риск сделки" и "согласовать цену" необходимо выполнить одно и то же действие — рассчитать стоимость заказа.

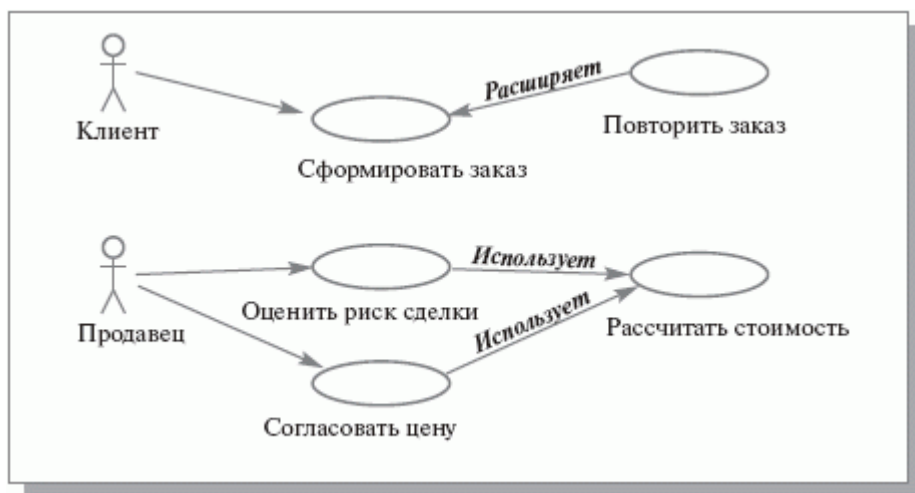


Рис. 11.4. Связи на диаграммах прецедентов

Динамические аспекты поведения системы отражаются приведенными ниже диаграммами.

В отличие от некоторых подходов объектного моделирования, когда и состояние, и поведение системы отображаются на *диаграммах классов*, UML отделяет описание поведения в **диаграммы взаимодействия**. В UML диаграммы классов не содержат сообщений, которые усложняют их чтение. Поток сообщений между объектами выносится на *диаграммы взаимодействия*. Как правило, *диаграмма взаимодействия* охватывает поведение объектов в рамках одного варианта использования.

Прямоугольники на диаграмме представляют различные объекты и роли, которые они имеют в системе, а линии между *классами* отображают отношения (или ассоциации) между ними. Сообщения обозначаются ярлыками возле стрелок, они могут иметь нумерацию и показывать возвращаемые значения.

Существуют два вида *диаграмм взаимодействия*: диаграммы последовательностей и кооперативные диаграммы.

Диаграммы последовательностей

Этот вид диаграмм используется для точного определения логики сценария выполнения прецедента. Диаграммы последовательностей отображают типы объектов, взаимодействующих при исполнении прецедентов, сообщения, которые они посылают друг другу, и любые возвращаемые значения, ассоциированные с этими сообщениями. Прямоугольники на вертикальных линиях показывают "время жизни" объекта. Линии со стрелками и надписями названий методов означают вызов метода у объекта.



Рис. 11.5. Диаграмма последовательности обработки заказа

- вводятся строки заказа;
- по каждой строке проверяется наличие товара;
- если запас достаточен — инициируется поставка;
- если запас недостаточен — инициируется дозаказ (повторный заказ).

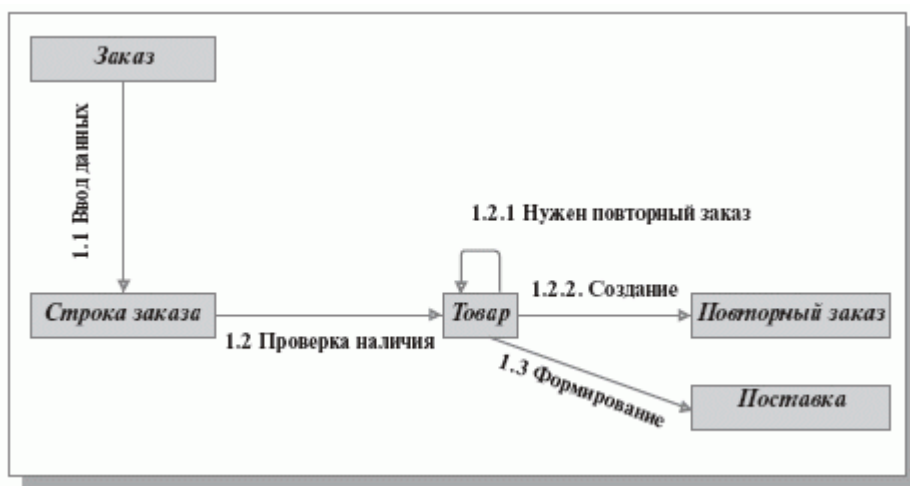


Рис. 11.6. Кооперативная диаграмма прохождения заказа

Сообщения появляются в той последовательности, как они показаны на диаграмме — сверху вниз. Если предусматривается отправка сообщения объектом самому себе (самodelегирование), то стрелка начинается и заканчивается на одной "линии жизни".

На диаграммы может быть добавлена управляющая информация: описание условий, при которых посылается сообщение; признак многократной отправки сообщения (маркер итерации); признак возврата сообщения.

Кооперативные диаграммы

На кооперативных диаграммах объекты (или *классы*) показываются в виде прямоугольников, а стрелками обозначаются сообщения, которыми они обмениваются в рамках одного варианта использования. Временная последовательность сообщений отражается их нумерацией.

Диаграммы состояний

Диаграммы состояний используются для описания поведения сложных систем. Они определяют все возможные состояния, в которых может находиться объект, а также процесс смены состояний объекта в результате некоторых событий. Эти диаграммы обычно используются для описания поведения одного объекта в нескольких прецедентах.

Прямоугольниками представляются состояния, через которые проходит объект во время своего поведения. Состояниям соответствуют определенные значения атрибутов объектов. Стрелки представляют переходы от одного состояния к другому, которые вызываются выполнением некоторых функций объекта. Имеется также два вида псевдо-состояний: начальное состояние, в котором находится только что созданный объект, и конечное состояние, которое объект не покидает, как только туда перешел.

Переходы имеют метки, которые синтаксически состоят из трех необязательных частей (см. [рис. 11.7](#)):

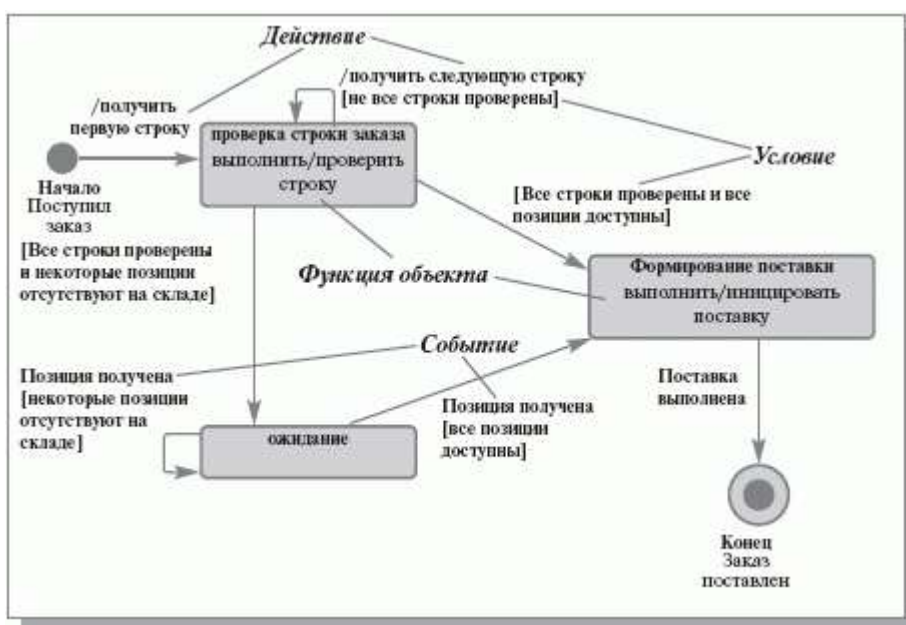


Рис. 11.7. Диаграмма состояний объекта «заказ»

<
Событие> <[Условие]> < / Действие>.

На диаграммах также отображаются функции, которые выполняются объектом в определенном состоянии. Синтаксис метки деятельности:

выполнить/< деятельность >.

Диаграммы деятельности

Диаграмма деятельности — это частный случай *диаграммы состояний*. На диаграмме деятельности представлены переходы потока управления от одной деятельности к другой внутри системы. Этот вид диаграмм обычно используется для описания поведения, включающего в себя множество параллельных процессов.

Основными элементами диаграмм деятельности являются ([рис. 11.8](#)):



Рис. 11.8. Диаграмма деятельности — обработка заказа

- овалы, изображающие действия объекта;
- линейки синхронизации, указывающие на необходимость завершить или начать несколько действий (модель логического условия "И");
- ромбы, отражающие принятие решений по выбору одного из маршрутов выполнения процесса (модель логического условия "ИЛИ");
- стрелки — отражают последовательность действий, могут иметь метки условий.

На диаграмме деятельности могут быть представлены действия, соответствующие нескольким вариантам использования. На таких диаграммах появляется множество начальных точек, поскольку они отражают теперь реакцию системы на множество внешних событий. Таким образом, диаграммы деятельности позволяют получить полную картину поведения системы и легко оценивать влияние изменений в отдельных вариантах использования на конечное поведение системы.

Любая деятельность может быть подвергнута дальнейшей декомпозиции и представлена в виде отдельной диаграммы деятельности или спецификации (словесного описания).

Диаграммы компонентов

Диаграммы компонентов позволяют изобразить модель системы на физическом уровне.

Элементами диаграммы являются компоненты — физические замещаемые модули системы. Каждый компонент является полностью независимым элементом системы. Разновидностью компонентов являются узлы. Узел — это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки. Узлы делятся на два типа:

- устройства — узлы системы, в которых данные не обрабатываются.
- процессоры — узлы системы, осуществляющие обработку данных.

Для различных типов компонентов предусмотрены соответствующие стереотипы в языке UML.

Компонентом может быть любой достаточно крупный модульный объект, такой как таблица или экстенд базы данных, подсистема, бинарный исполняемый файл, готовая к использованию система или приложение. Таким образом, *диаграмму компонентов* можно

рассматривать как *диаграмму классов* в более крупном (менее детальном) масштабе. Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как *классы*, интерфейсы и кооперации.

Основное назначение *диаграмм компонентов* — разделение системы на элементы, которые имеют стабильный интерфейс и образуют единое целое. Это позволяет создать ядро системы, которое не будет меняться в ответ на изменения, происходящие на уровне подсистем.

На [рис. 11.9](#) показана упрощенная схема элементов фрагмента корпоративной системы. "Коробки" представляют собой компоненты — приложения или внутренние подсистемы. Пунктирные линии отражают зависимости между компонентами.

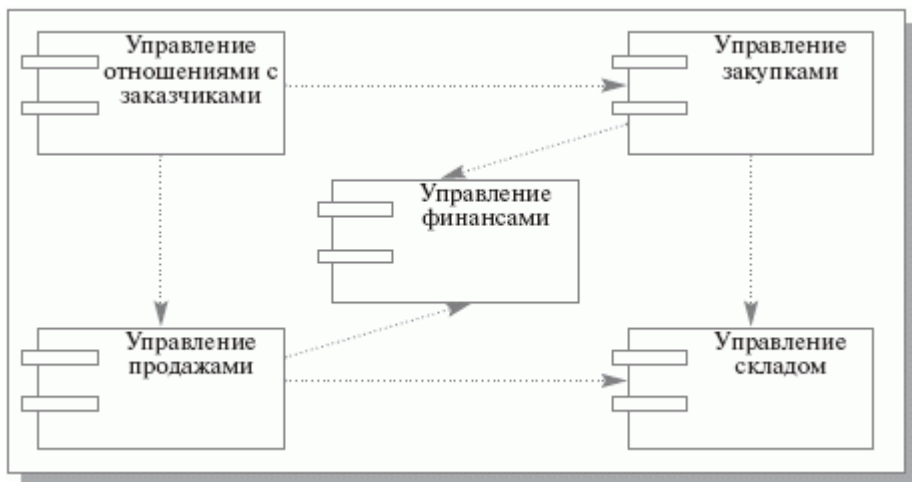


Рис. 11.9. Диаграмма компонентов фрагмента КИС

Каждый компонент диаграммы при необходимости документируется с помощью более детальной *диаграммы компонентов*, *диаграммы сценариев* или *диаграммы классов*.

Пакеты UML

Пакеты представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить диаграммы различного типа и назначения. В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки. Изображается пакет в виде папки с закладкой, содержащей, как правило, только имя и иногда — описание содержимого.

Диаграмма пакетов содержит пакеты *классов* и зависимости между ними. Зависимость между двумя пакетами имеет место в том случае, если изменения в определении одного элемента влекут за собой изменения в другом. По отношению к пакетам можно использовать механизм обобщения (см. выше раздел "*Диаграммы классов*").

Лекция: Этапы проектирования ИС с применением UML

UML обеспечивает поддержку всех этапов жизненного цикла ИС и предоставляет для этих целей ряд графических средств – диаграмм.

На этапе создания *концептуальной модели* для описания бизнес-деятельности используются *модели бизнес-прецедентов* и диаграммы видов деятельности, для описания бизнес-объектов – *модели бизнес-объектов* и диаграммы последовательностей.

На этапе создания логической модели ИС описание требований к системе задается в виде модели и описания системных прецедентов, а предварительное проектирование осуществляется с использованием диаграмм классов, диаграмм последовательностей и диаграмм состояний.

На этапе создания физической модели детальное проектирование выполняется с использованием диаграмм классов, диаграмм компонентов, диаграмм развертывания.

Ниже приводятся определения и описывается назначение перечисленных диаграмм и моделей применительно к задачам проектирования ИС (в скобках приведены альтернативные названия диаграмм, используемые в современной литературе).

Диаграммы прецедентов (диаграммы вариантов использования, use case diagrams) – это обобщенная модель функционирования системы в окружающей среде.

Диаграммы видов деятельности (диаграммы деятельностей, activity diagrams) – модель бизнес-процесса или поведения системы в рамках прецедента.

Диаграммы взаимодействия (interaction diagrams) – модель процесса обмена сообщениями между объектами, представляется в виде диаграмм последовательностей (sequence diagrams) или кооперативных диаграмм (collaboration diagrams).

Диаграммы состояний (statechart diagrams) – модель динамического поведения системы и ее компонентов при переходе из одного состояния в другое.

Диаграммы классов (class diagrams) – логическая модель базовой структуры системы, отражает статическую структуру системы и связи между ее элементами.

Диаграммы базы данных (database diagrams) — модель структуры базы данных, отображает таблицы, столбцы, ограничения и т.п.

Диаграммы компонентов (component diagrams) – модель иерархии подсистем, отражает физическое размещение баз данных, приложений и интерфейсов ИС.

Диаграммы развертывания (диаграммы размещения, deployment diagrams) – модель физической архитектуры системы, отображает аппаратную конфигурацию ИС.

На [рис. 12.1](#) показаны отношения между различными видами диаграмм UML. Указатели стрелок можно интерпретировать как отношение "является источником входных данных для..." (например, диаграмма прецедентов является источником данных для диаграмм видов деятельности и последовательности). Приведенная схема является наглядной иллюстрацией итеративного характера разработки моделей с использованием UML.

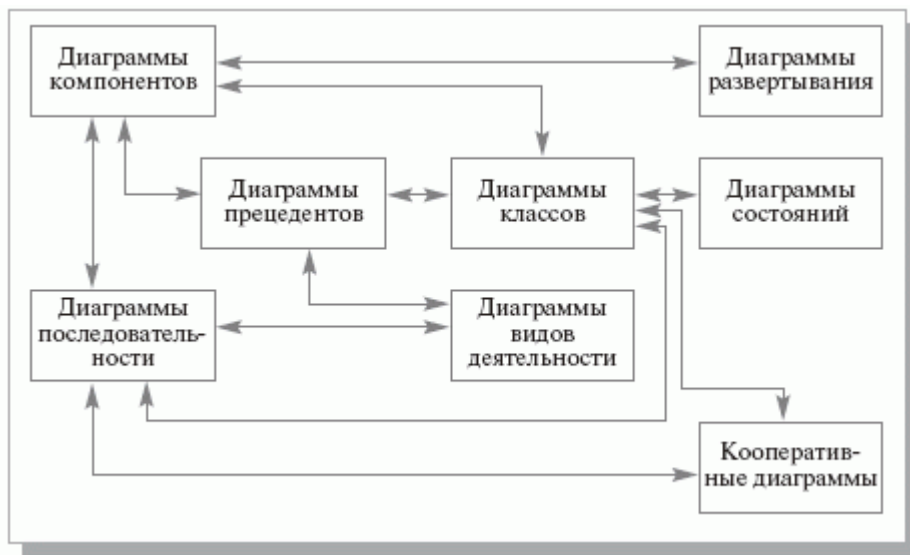


Рис. 12.1. Взаимосвязи между диаграммами UML

Ниже приводятся описания последовательных этапов проектирования ИС с использованием UML.

Разработка модели бизнес-прецедентов

Модель бизнес-прецедентов описывает бизнес-процессы с точки зрения внешнего пользователя, т.е. отражает взгляд на деятельность организации из вне.

Проектирование системы начинается с изучения и моделирования бизнес-деятельности организации. На этом этапе вводится и отображается в модели ряд понятий, свойственных объектно-ориентированному подходу:

Исполнитель (Действующее лицо, Actor) – личность, организация или система, взаимодействующая с ИС; различают внешнего исполнителя (который использует или используется системой, т.е. порождает прецеденты деятельности) и внутреннего исполнителя (который обеспечивает реализацию прецедентов деятельности внутри системы). На диаграмме исполнитель представляется стилизованной фигуркой человека.

Прецедент – законченная последовательность действий, инициированная внешним объектом (личностью или системой), которая взаимодействует с ИС и получает в результате некоторое сообщение от ИС. На диаграмме представляется овалом с надписью, отражающей содержание действия.

Класс — описание совокупности однородных объектов с их атрибутами, операциями, отношениями и семантикой. На диаграмме представляется прямоугольником, содержащим описания атрибутов и операций класса.

Ассоциация – связь между двумя элементами модели. На диаграмме представляется линией.

Обобщение – связь между двумя элементами модели, когда один элемент (подкласс) является частным случаем другого элемента (суперкласса). На диаграмме представляется стрелкой.

Агрегация – отношение между элементами модели, когда один элемент является частью другого элемента (агрегата). На диаграмме представляется стрелкой с ромбовидным концом.

Для иллюстрации этапов разработки проекта использованы адаптированные материалы проекта ИС медицинского центра [[рис. 12.2](#)]. Назначение ИС – автоматизация ведения и использования клинических записей о пациентах. В настоящее время эта работа производится вручную персоналом центра. На [рис. 12.2](#) представлена общая модель деятельности центра в виде диаграммы прецедентов. Прецедент "Обслуживание пациента" реализуется через множество других, более ограниченных прецедентов ([рис. 12.3](#)), отражающих детализацию представления функционирования центра.



Рис. 12.2. Общая диаграмма деятельности медицинского центра по обслуживанию пациента

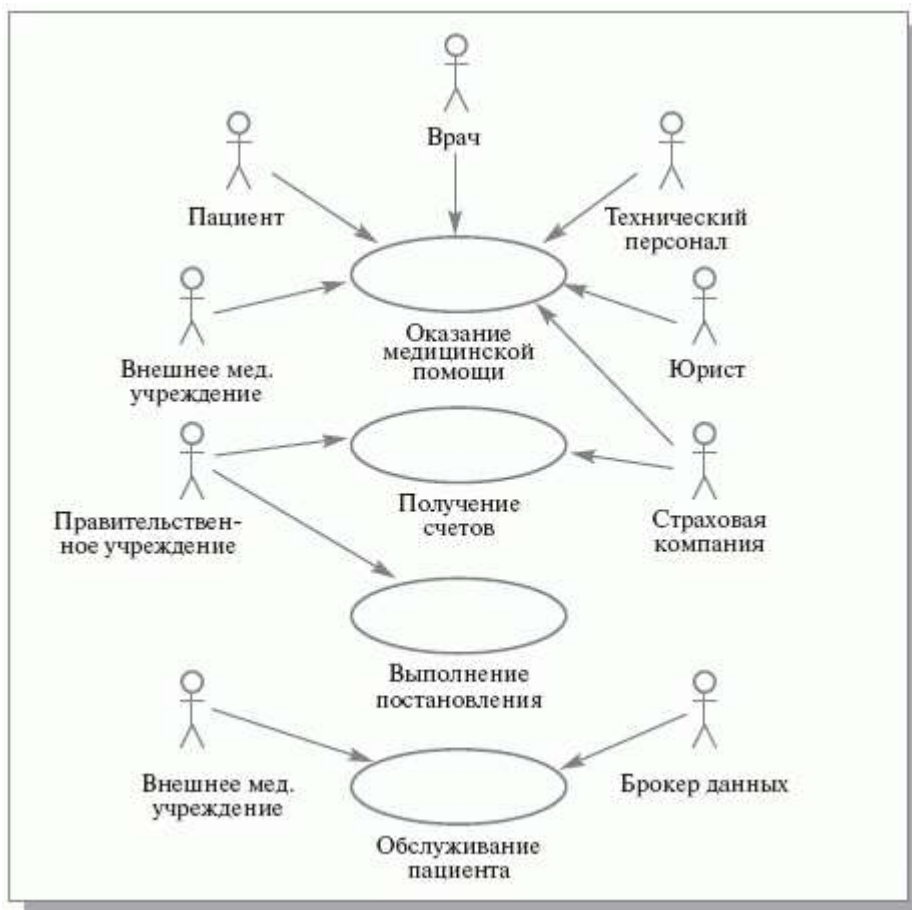


Рис. 12.3. Модель бизнес-прецедентов, составляющих обслуживание пациента

Для включения в диаграмму выбранные прецеденты должны удовлетворять следующим критериям:

- прецедент должен описывать, **ЧТО** нужно делать, а не **КАК**;
- прецедент должен описывать действия с точки зрения **ИСПОЛНИТЕЛЯ**;
- прецедент должен возвращать исполнителю некоторое **СООБЩЕНИЕ**;
- последовательность действий внутри прецедента должна представлять собой одну **НЕДЕЛИМУЮ** цепочку.

Исходя из цели создания системы, для дальнейшего исследования и моделирования отбираются только те бизнес-прецеденты, которые связаны с использованием клинических записей.

Выполнение прецедента описывается с помощью диаграмм видов деятельности, которые отображают исполнителей и последовательность выполнения соответствующих бизнес-процессов ([рис. 12.4](#)).

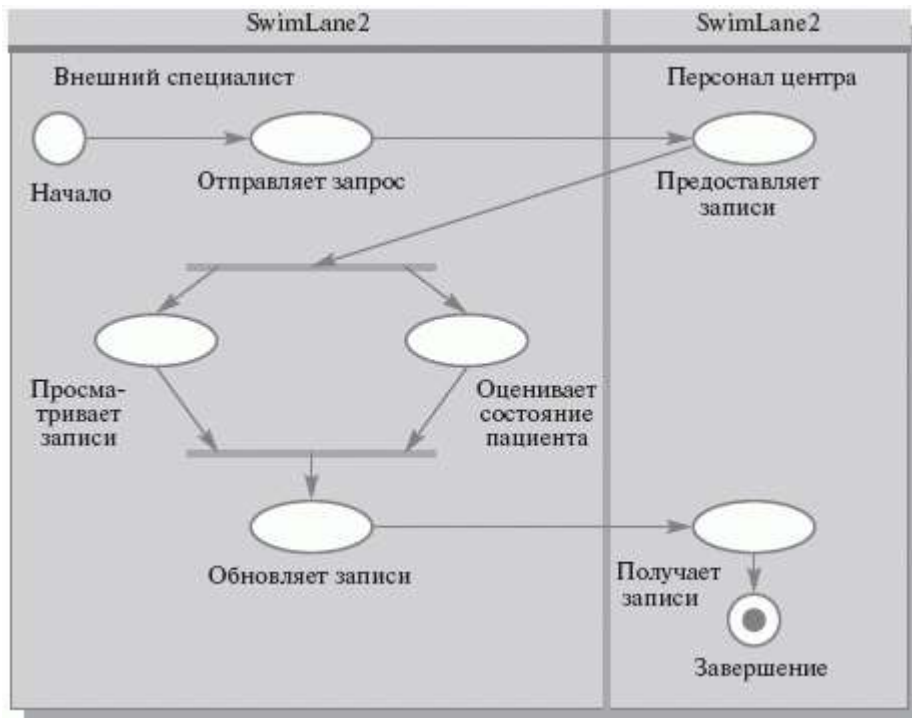


Рис. 12.4. Диаграмма видов деятельности для прецедента "Оказание медицинской помощи"

Несмотря на то, что оказание медицинской помощи предусматривает множество разнообразных действий исполнителей, для нашей задачи существенными являются только процессы обмена информацией между этими исполнителями, и именно они отображаются в создаваемых моделях. Поэтому на диаграмме отражен процесс оценки состояния пациента на основании имеющейся в центре информации о нем.

Общее поле диаграммы деятельности делится на несколько "плавающих дорожек", каждая из которых содержит описание действий одного из исполнителей. Основными элементами диаграмм видов деятельности являются обозначения состояния ("начало", "конец"), действия (овал) и момента синхронизации действий (линейка синхронизации, на которой сходятся или разветвляются несколько стрелок).

Диаграмма подходит для описания действий как внешнего, так и внутреннего специалиста центра.

Этап завершается после разработки диаграмм видов деятельности для всех выделенных в модели бизнес-прецедентов. Естественно, на последующих этапах анализа и проектирования будут выявлены какие-то важные подробности в описании деятельности объекта автоматизации. Поэтому разработанные на данном этапе модели будут еще неоднократно корректироваться.

Разработка модели бизнес-объектов

Следующим этапом проектирования ИС является разработка модели бизнес-объектов, которая показывает выполнение бизнес-процессов организации ее внутренними исполнителями. Основными компонентами **моделей бизнес-объектов** являются внешние и внутренние исполнители, а также бизнес-сущности, отображающие все, что используют внутренние исполнители для реализации бизнес-процессов. Пример модели бизнес-объектов для прецедента "Ответ на запрос" приведен на [рис. 12.5](#).



Рис. 12.5. Модель бизнес-объектов прецедента "Ответ на запрос"

В этой диаграмме появилось новое действующее лицо – отправитель запроса. На самом деле с запросом о состоянии пациента могут обращаться в систему многие из действующих лиц: юрист, страховая компания, технический персонал и даже сам пациент. Таким образом, понятие "Отправитель запроса" служит для обобщенного представления всех этих действующих лиц при описании прецедента "Ответ на запрос" (рис. 12.6). "Отправитель запроса" становится суперклассом по отношению к обобщаемым понятиям (подклассам).

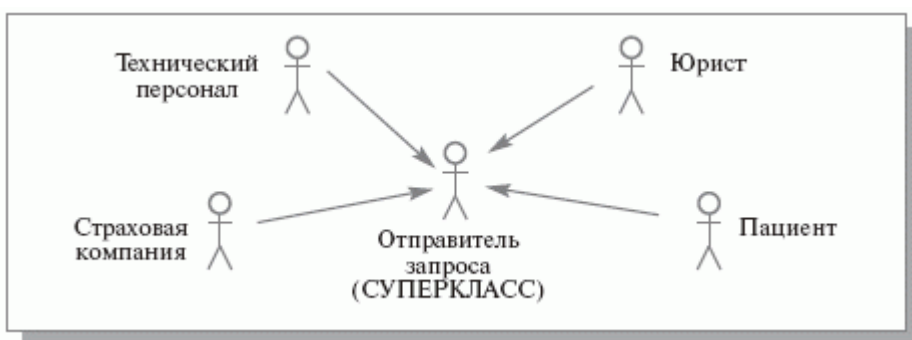


Рис. 12.6. Обобщение классов

Для детального описания выполнения бизнес-процессов обычно используются диаграммы последовательностей (рис. 12.7).



Рис. 12.7. Диаграмма последовательностей для прецедента "Ответ на запрос"

Основными элементами диаграммы последовательностей являются обозначения объектов (прямоугольники), вертикальные линии, отображающие течение времени при деятельности объекта, и стрелки, показывающие выполнение действий объектами.

Результатом этого этапа являются согласованные с заказчиком и достаточно подробные описания действий специалистов организации, внедряющей ИС, необходимые для обеспечения исполнения ее функций.

Разработка концептуальной модели данных

Затем на основе информации, выявленной на этапах бизнес-моделирования, выполняется разработка **концептуальной модели данных**, которые будут использоваться в разрабатываемой системе. На [рис. 12.8](#) представлена в виде диаграммы классов модель данных для объекта "Клинические записи".

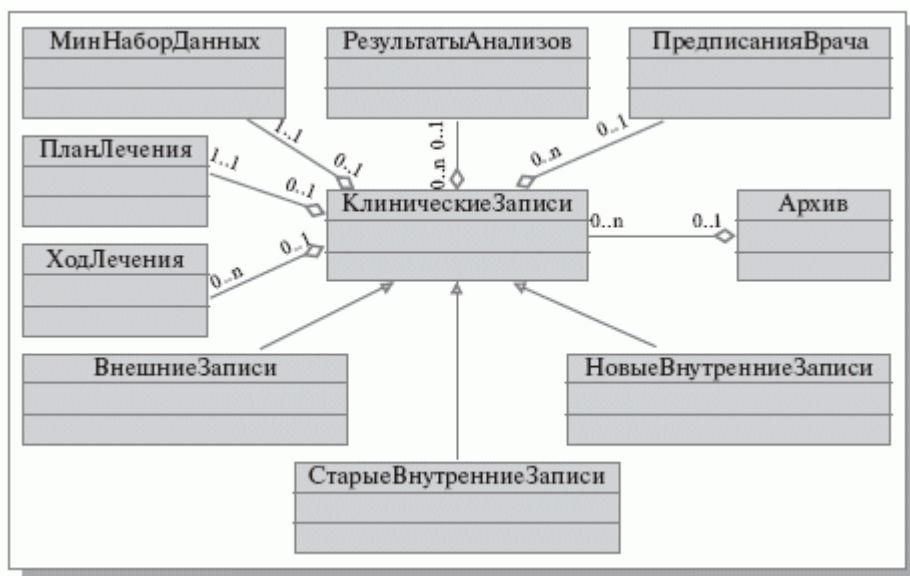


Рис. 12.8. Концептуальная модель данных

Модель показывает, что клинические записи включают (агрегируют) ряд блоков. При этом "минимальный набор данных" и "план лечения" могут быть включены в каждую клиническую запись в единственном экземпляре, а блоки "результаты анализов", "предписания врача", "ход лечения" могут повторяться неограниченное число раз.

Архив состоит из множества клинических записей (агрегирует клинические записи), но может быть и пустым.

Поскольку пациент может предварительно проходить лечение в других учреждениях, или несколько раз проходить лечение в центре, появляются дополнительные разновидности (подклассы) клинических записей: внешние, старые внутренние, новые внутренние.

Этот этап завершает процедуры бизнес-моделирования и позволяет представить команде проектировщиков в едином формате ту информацию, которая будет необходима для создания системы. Разработанные диаграммы являются отправной точкой в процессах проектирования баз данных и приложений системы, обеспечивают согласованность действий бизнес-аналитиков и разработчиков в процессе дальнейшей работы над системой. Эти диаграммы, конечно же, будут претерпевать изменения в процессе последующего проектирования, однако эти изменения будут фиксироваться в формате, уже привычном для всей команды разработчиков, и будут автоматически отражаться в последующих моделях.

Разработка требований к системе

На этапе формирования требований, прежде всего, необходимо определить область действия разрабатываемой системы и получить точное представление о желаемых возможностях системы.

Основой разработки требований является **модель системных прецедентов**, отражающая выполнение конкретных обязанностей внутренними и внешними исполнителями с использованием ИС.

Источником данных для создания *модели системных прецедентов* являются разработанные на предыдущем этапе бизнес-модели. Однако при создании модели полезно предварительно составить детальные описания прецедентов, содержащие определения используемых данных и точную последовательность их выполнения. Описание осуществляется в соответствии с принятым в организации шаблоном, который обычно включает следующие разделы:

- заголовок (название прецедента, ответственный за исполнение, дата создания шаблона/внесения изменений);
- краткое описание прецедента;
- ограничения;
- предусловия (необходимое состояние системы или условия, при которых должен выполняться прецедент);
- постусловия (возможные состояния системы после выполнения прецедента);
- предположения;
- основная последовательность действий;
- альтернативные последовательности действий и условия, их инициирующие;
- точки расширения и включения прецедентов.

В процессе создания *модели системных прецедентов* осуществляется преобразование и перенос компонентов бизнес-моделей на новые диаграммы. Типовые преобразования по технологии Rational Unified Process приведены в [таблица 12.1](#).

Таблица 12.1.	
Элементы бизнес-модели	Элементы <i>модели системных прецедентов</i>
Бизнес-прецеденты	Подсистемы
Внешние исполнители	Исполнители
Внутренние исполнители	Исполнители или прецеденты
Процессы, выполняемые внутренними исполнителями	Прецеденты

На [рис. 12.9](#) представлена *модель системных прецедентов* для бизнес-прецедента "Оказание медицинской помощи". Исходя из цели создания системы, в *модели системных прецедентов* отражены только те действия исполнителей, которые связаны с предоставлением доступа и обновлением клинических записей.



Рис. 12.9. Модель системных прецедентов

Описываемые моделью функции характерны только для одного вида деятельности – оказания медицинской помощи, и в основном не используются в других видах деятельности Центра. Это позволяет объединить выделенные функции в некую единую подсистему проектируемой ИС.

Внутренний исполнитель "Персонал центра" (см. [рис. 12.4](#), [рис. 12.7](#)) и выполняемый им ручной процесс преобразован в системный прецедент "Предоставление доступа к клиническим записям".

Внешние исполнители (например, "Производитель медицинского оборудования") непосредственно взаимодействуют с проектируемой системой, т.е. превращаются в исполнителей.

В модели отражены два специальных типа связи между прецедентами (на [рис. 12.9](#) соответствующие прецеденты выделены тенью):

- "включает" — один прецедент в процессе своего исполнения обязательно выполняет некий блок действий, составляющих другой прецедент;
- "расширяет" — когда прецеденты подобны по своим действиям, но один несет несколько большую функциональную нагрузку.

Прецедент "Проверка прав доступа" впервые появился на диаграммах и реализуется средствами разрабатываемой ИС. Поэтому для него приходится разрабатывать диаграмму последовательностей, описывающую его исполнение ([рис. 12.10](#)). В результате в проектируемой ИС появляются два новых объекта – программный модуль "Менеджер защиты" и информационный блок "Набор прав".

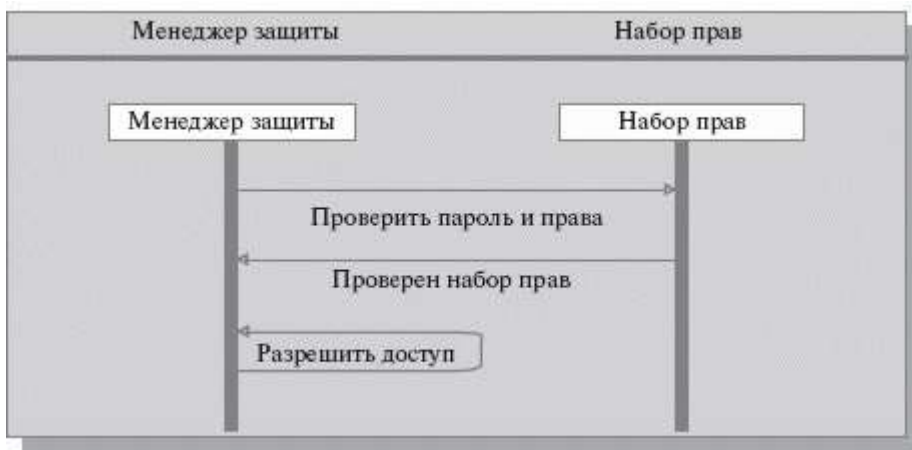


Рис. 12.10. Диаграмма последовательностей для прецедента "Проверка прав"

Таким образом, результатом разработки *модели системных прецедентов* является не только исчерпывающий перечень функций, которые должны быть реализованы в проектируемой системе, но и подробное описание необходимой реализации этих функций.

Анализ требований и предварительное проектирование системы.

Основные задачи этапа:

- определить проект системы, который будет отвечать всем бизнес-требованиям;
- разработать общий предварительный проект для всех команд разработчиков (проектировщиков баз данных, разработчиков приложений, системных архитекторов и пр.)

Основным инструментом на данном этапе являются диаграммы классов системы, которые строятся на основе разработанной ***модели системных прецедентов***. Одновременно на этом этапе уточняются диаграммы последовательностей выполнения отдельных прецедентов, что приводит к изменениям в составе объектов и диаграммах классов. Это естественное отражение средствами UML итеративного процесса разработки системы.

Диаграммы классов системы заполняются объектами из *модели системных прецедентов*. Они содержат описание этих объектов в виде классов и описание взаимодействия между классами.

Диаграмма классов, описывающая процедуры защиты доступа к данным, приведена на [рис. 12.11](#).

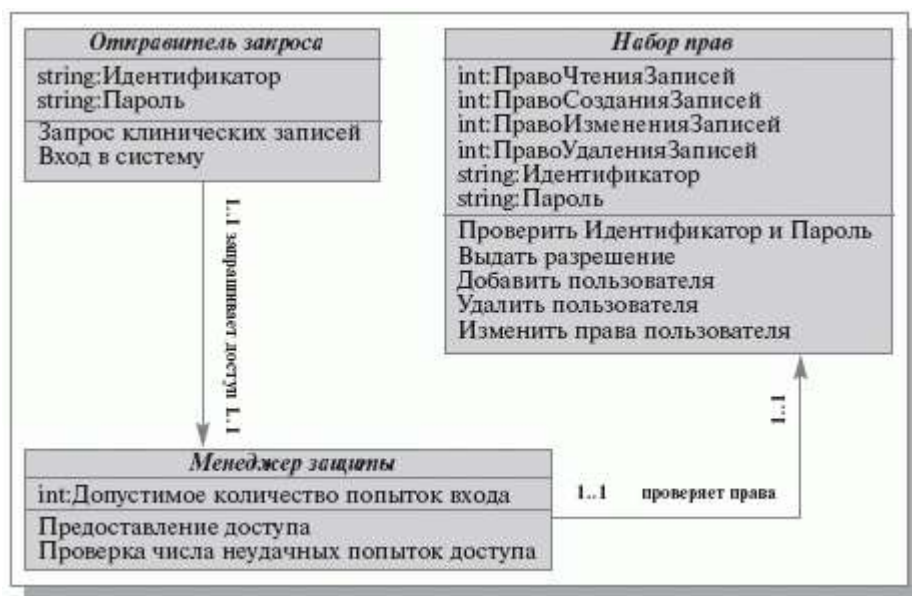


Рис. 12.11. Диаграмма классов "Защита доступа"

Таким образом, в результате этого этапа проектирования появляется достаточно подробное описание состава и функций проектируемой системы, а также информации, которую необходимо использовать в базе данных и в приложениях.

Поскольку диаграммы классов строятся на основе разработанных ранее бизнес-моделей, появляется уверенность в том, что разрабатываемая система будет действительно удовлетворять исходным требованиям заказчика.

В то же время, благодаря своему синтаксису, диаграммы классов оказываются хорошим средством структурирования и представления требований к функциональности, интерфейсам и данным для элементов проектируемой системы.

Разработка моделей базы данных и приложений

На этом этапе осуществляется отображение элементов полученных ранее моделей классов в элементы моделей базы данных и приложений:

- классы отображаются в таблицы;
- атрибуты – в столбцы;
- типы – в типы данных используемой СУБД;
- ассоциации – в связи между таблицами (ассоциации "многие-ко-многим" преобразуются в ассоциации "один-ко-многим" посредством создания дополнительных таблиц связей);
- приложения – в отдельные классы с окончательно определенными и связанными с данными в базе методами и атрибутами.

Поскольку модели базы данных и приложений строятся на основе единой логической модели, автоматически обеспечивается связность этих проектов ([рис. 12.12](#)).

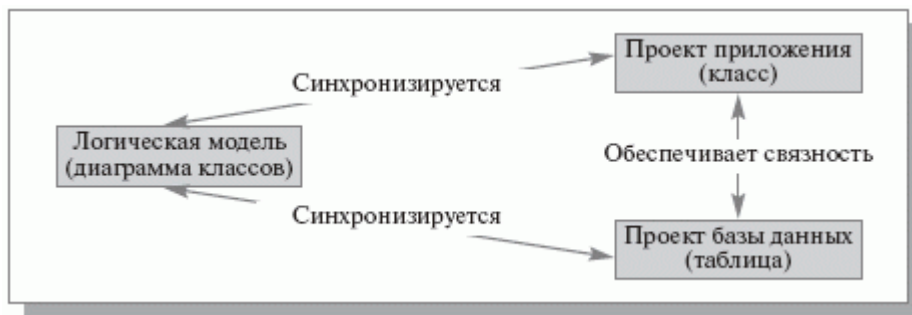


Рис. 12.12. Связь между проектами базы данных и приложений

В модель базы данных отображаются только перманентные классы, из которых удаляются атрибуты, не отображаемые в столбцах (например, атрибут типа "Общий объем продаж", который получается суммированием содержимого множества полей базы данных). Некоторые атрибуты (например, АДРЕС) могут отображаться в множество столбцов (СТРАНА, ГОРОД, УЛИЦА, ДОМ, ПОЧТОВЫЙ ИНДЕКС).

Для каждого простого класса в модели базы данных формируется отдельная таблица, включающая столбцы, соответствующие атрибутам класса.

Отображение классов подтипов в таблицы осуществляется одним из стандартных способов:

- одна таблица на класс;
- одна таблица на суперкласс;
- одна таблица на иерархию.

В первом случае для каждого из классов создается отдельная таблица, между которыми затем устанавливаются необходимые связи. Во втором случае создается таблица для суперкласса, а затем в каждую таблицу подклассов включаются столбцы для каждого из атрибутов суперкласса. В третьем – создается единая таблица, содержащая атрибуты как суперкласса, так и всех подклассов (рис. 12.13). При этом для выделения исходных таблиц подклассов в результирующую таблицу добавляется один или более дополнительных столбцов (на рисунке показан курсивом).



Рис. 12.13. Преобразование иерархии в таблицу

Разработка проекта базы данных осуществляется с использованием специального UML-профиля (Profile for Database Design), который включает следующие основные компоненты диаграмм:

- таблица – набор записей базы данных по определенному объекту;
- столбец – элемент таблицы, содержащий значения одного из атрибутов таблицы;
- первичный ключ (PK) – атрибут, однозначно идентифицирующий строку таблицы;
- внешний ключ (FK) – один или группа атрибутов одной таблицы, которые могут использоваться как первичный ключ другой таблицы;
- обязательная связь – связь между двумя таблицами, при которой дочерняя таблица существует только вместе с родительской;
- необязательная связь – связь между таблицами, при которой каждая из таблиц может существовать независимо от другой;
- представление – виртуальная таблица, которая обладает всеми свойствами обычной таблицы, но не хранится постоянно в базе данных;
- хранимая процедура – функция обработки данных, выполняемая на сервере;
- домен – множество допустимых значений для столбца таблицы.

На [рис. 12.14](#) представлен фрагмент модели базы данных — две таблицы, соответствующие классам "пациент" ([рис. 12.3](#), [рис. 12.6](#)) и "минимальный набор данных" ([рис. 12.8](#)). Связь между ними обязательная, поскольку "минимальный набор данных" не может существовать без "пациента".



Рис. 12.14. Фрагмент модели базы данных

На диаграммах указываются дополнительные характеристики таблиц и столбцов:

- ограничения – определяют допустимые значения данных в столбце или операции над данными (ключ (PK,FK) – ограничение, определяющее тип ключа и его столбец; проверка (Check) – ограничение, определяющее правило контроля данных; уникальность (Unique) – ограничение, определяющее, что в столбце содержатся неповторяющиеся данные);
- триггер – программа, выполняющая при определенных условиях предписанные действия с базой данных;
- тип данных и пр.

Результатом этапа является детальное описание проекта базы данных и приложений системы.

Проектирование физической реализации системы

На этом этапе проектирования модели баз данных и приложений дополняются обозначениями их размещения на технических средствах разрабатываемой системы. На [рис. 12.15](#) приведено изображение разделения таблицы "пациент" на три экстента (<<Tablespace>>) в соответствии с первой буквой фамилии пациента.

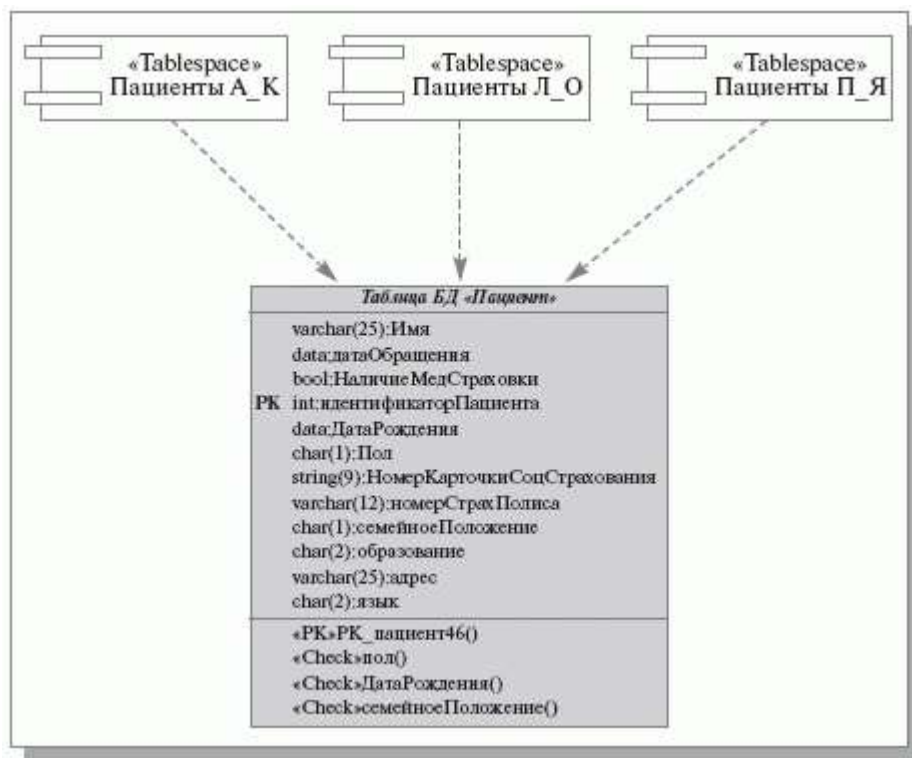


Рис. 12.15. Экстенты таблицы "Пациент"

Основными понятиями UML, которые используются на данном этапе, являются следующие:

- компонент – самостоятельный физический модуль системы;
- зависимость – связь между двумя элементами, при которой изменения в одном элементе вызывают изменения другого элемента;
- устройство – узел, не обрабатывающий данные;
- процессор – узел, выполняющий обработку данных;
- соединение – связь между устройствами и процессорами.

Диаграммы развертывания позволяют отобразить на единой схеме различные компоненты системы (программные и информационные) и их распределение по комплексу технических средств ([рис. 12.16](#)).

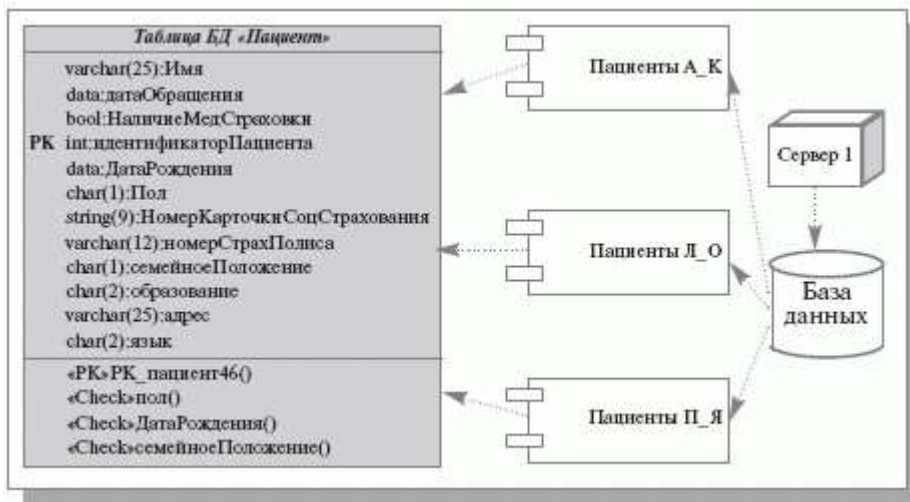


Рис. 12.16. Фрагмент диаграммы развертывания ИС

Таким образом, при проектировании сложной ИС она разделяется на части, и каждая из них затем исследуется и создается отдельно. В настоящее время используются два различных способа такого разбиения ИС на подсистемы: структурное (или функциональное) разбиение и объектная (компонентная) декомпозиция.

С позиций проектирования ИС суть функционального разбиения может быть выражена известной формулой: "Программа = Данные + Алгоритмы". При функциональной декомпозиции программной системы ее структура описывается блок-схемами, узлы которых представляют собой "обрабатывающие центры" (функции), а связи между узлами описывают движение данных.

При объектном разбиении в системе выделяются "активные сущности" – объекты (или компоненты), которые взаимодействуют друг с другом, обмениваясь сообщениями и выполняя соответствующие функции (методы) объекта.

Если при проектировании ИС разбивается на объекты, то для ее визуального моделирования следует использовать UML. Если в основу проектирования положена функциональная декомпозиция ИС, то UML не нужен и следует использовать рассмотренные ранее структурные нотации.

В то же время, при выборе подхода к разработке ИС следует учитывать, что визуальные модели все более широко используются в существующих технологиях управления проектированием систем, сложность, масштабы и функциональность которых постоянно возрастают. Они хорошо приспособлены для решения таких часто возникающих при создании систем задач как: физическое перераспределение вычислений и данных, обеспечение параллелизма вычислений, репликация БД, обеспечение безопасности доступа к ИС, оптимизация балансировки нагрузки ИС, устойчивость к сбоям и т.п. Визуализированные средствами UML модели ИС позволяют наладить плодотворное взаимодействие между заказчиками, пользователями и командой разработчиков. Они обеспечивают ясность представления выбранных архитектурных решений и позволяют понять разрабатываемую систему во всей ее полноте.

Лабораторная работа

Тема: Разработка базы данных с использованием SQL

Теоретические сведения

Родоначальником серии SQL Server и его основой является язык запросов SQL. Данный язык был предложен сотрудником компании IBM Эдгаром Коддом в начале 1970-х гг. Изначально он назывался SEQUEL (*англ.* Structured English Query Language, структурированный английский язык для запросов), который впоследствии по юридическим соображениям был переименован в SQL (*англ.* Structured Query Language, структурированный язык запросов). Официальным произношением стало [es kju:' el] — *эс-кью-эл*. Несмотря на это, даже англоязычные специалисты по-прежнему часто называют SQL *сиквел*, вместо *эс-кью-эл* (по-русски также часто говорят «эс-ку-эль»). Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования.

К началу 1980-х годов SQL завоевал популярность как язык реляционных систем управления базами данных (СУБД) и привлек внимание Американского национального института по стандартизации (American National Standards Institute, ANSI), который в 1986, 1989, 1992, 1999 и 2003 годах выпустил стандарты языка SQL. В 1989 году SQL был включен в стандарты международной организации по стандартизации ISO (SQL:1989), а затем были приняты и опубликованы стандарты SQL:1992, SQL:1999 и SQL:2003. В настоящее время все производители распространенных реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL:2003. В основу языка SQL, используемого в SQL Server, легла разновидность языка T-SQL (Transact-SQL).

В начале 1980-х гг. фирма IBM и в частности в то время ее подразделениями Microsoft и Sybase была создана первая версия сетевой СУБД, которая называлась SQL Server версия 1.0, для операционной системы

IBM OS/2. После этого под эту операционную систему было выпущено еще 3 версии SQL Server. В середине 1980-х г.г. компании Microsoft и Sybase отделяются от фирмы IBM, и Microsoft начинает работу над своей операционной системой Windows, и вместе с компанией Sybase продолжает развитие SQL Server. В середине 1990-х г.г. Microsoft создала операционную систему Windows NT и вместе с компанией Sybase выпустила первую версию SQL Server для Windows версии 4.1.

В дальнейшем компания Sybase разорвала свои отношения с Microsoft, и Microsoft сама продолжила работу над Microsoft SQL Server 6.0. Данная версия была предназначена для работы в операционной системе Windows NT, 95 и 98. Затем были разработаны версии Microsoft SQL Server 6.5 (1995 – 1996 гг.), 7.0 (1996 – 1998 гг.), 2000 (1998 – 2000 гг.), 2005 (2000 – 2005 гг.), 2008 (2005 – 2008 гг.), 2008 R2 (2008 – 2010 гг.), 2012 (2010 – 2012 гг.), 2014, 2016.

В Microsoft SQL Server БД состоит из двух частей:

- файл данных – файл, имеющий расширение mdf и где находятся все таблицы и запросы;
- файл журнала транзакций – файл, имеющий расширение ldf, содержит журнал, где фиксируются все действия с БД. Данный файл предназначен для восстановления БД в случае её выхода из строя.

В СУБД Microsoft SQL Server новую базу данных (БД) можно создать, используя стандартные команды языка T-SQL. Для создания нового файла данных используется SQL-команда `CREATE DATABASE`, которая имеет следующий синтаксис (полное описание см. <https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-database-sql-server-transact-sql>):

```
CREATE DATABASE <Имя БД>  
(Name=<Логическое имя>,  
FileName=<Имя файла>  
[Size=<Нач. размер>],  
[Maxsize=<Макс. размер>],  
[FileGrowth=<Шаг>])  
[LOG ON
```

```
(Name=<Логическое имя>,  
FileName=<Имя файла>  
[Size=<Нач. размер>,  
[Maxsize=<Макс. размер >,  
[FileGrowth=<Шаг>])
```

Здесь Имя БД – имя создаваемой БД; Логическое имя – определяет логическое имя файла данных БД, по которому происходит обращение к файлу данных; Имя файла – определяет полный путь к файлу данных; Нач. размер – начальный размер файла данных в МБ; Макс. размер – максимальный размер файла данных в МБ; Шаг – шаг увеличения файла данных, либо в МБ либо в %.

Параметры в разделе LOG ON аналогичны параметрам в разделе CREATE DATABASE. Однако они определяют параметры журнала транзакций.

Создадим БД Artworks, расположенную в файле «D:\Artworks.mdf» и имеющую начальный размер файла данных – 3 МБ, максимальный размер файла данных – 100 МБ и шаг увеличения файла данных, равный 1 МБ. Файл журнала транзакций данной БД имеет имя «ArtworksLog» и расположен в файле «D:\Artworks.ldf». Данный файл имеет начальный размер, равный 1 МБ, максимальный размер, равный 20 МБ и шаг увеличения, равный 1 МБ.

```
CREATE DATABASE Artworks  
ON  
(Name = Artworks,  
FileName = 'D:\ Artworks.mdf',  
Size = 3MB,  
Maxsize = 100MB,  
FileGrowth= 1MB)  
LOG ON  
(Name = Artworks Log,  
FileName = 'D:\ Artworks.ldf',  
Size = 1MB,  
Maxsize = 20MB,  
FileGrowth = 1MB)
```

В языке запросов T-SQL с БД возможны следующие действия:

1. Отображение сведений о БД: EXEC SP_HELPDB <Имя БД>;
2. Изменение параметров БД: EXEC SP_DBOPTION <Имя БД>, <Параметр>, <Значение>;
3. Добавление новых файлов, удаление файлов и переименование файлов, входящих в БД:

```
ALTER DATABASE <Имя БД>  
ADD FILE (<Параметры>)|  
REMOVE FILE <Логическое имя файла>|  
MODIFY FILE (<Параметры>)
```

где, раздел ADD FILE – добавляет файл, REMOVE FILE – удаляет, а раздел MODIFY FILE – изменяет параметры файла;

4. Сжатие всей БД: DBCC SHRINKDATABASE <Имя БД>;
5. Сжатие конкретного файла БД: DBCC SHRINKFILE <Логическое имя файла>;
6. Переименование БД: EXEC SP_RENAMEDB <Имя БД>,<Новое имя БД>;
7. Удаление БД: DROP DATABASE <Имя БД>.

Вышеперечисленные команды используют следующие параметры:

- <Имя БД> – имя БД с которой производится действие;
- <Параметр> – изменяемый параметр;
- <Значение> – новое значение изменяемого параметра;
- <Параметры> – параметры файла БД, аналогичные параметрам, используемым в команде CREATE DATABASE;
- <Логическое имя файла> – логическое имя файла, входящего в БД;
- <Новое имя БД> – новое имя БД.

Задания

1. Запустим среду разработки «SQL Server Management Studio».

После запуска среды разработки появится окно подключения к серверу «Соединение с сервером» (рис. 1).

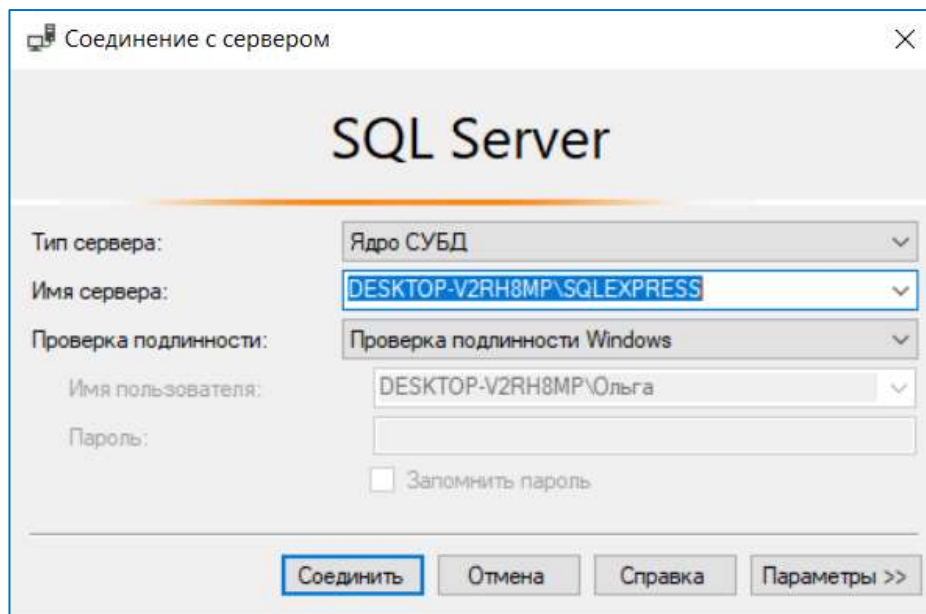


Рис. 1. Окно «Соединение с сервером»

В этом окне необходимо нажать кнопку «Соединить».

После нажатия кнопки «Соединить» появится окно среды разработки «SQL Server Management Studio» (рис. 2).

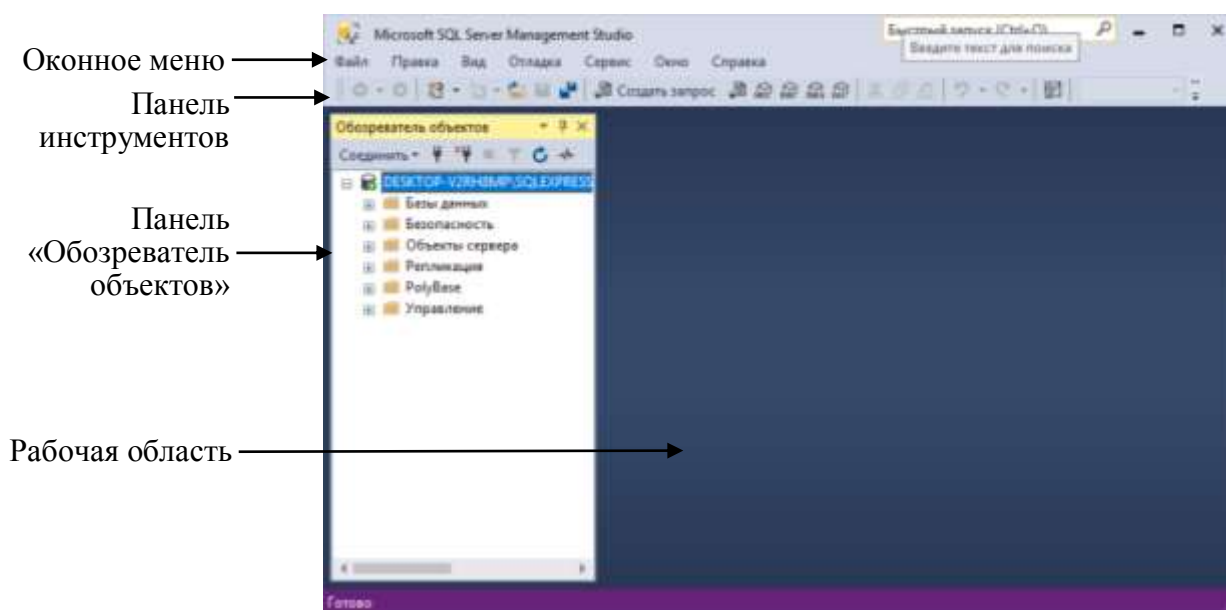



Рис. 2. Окно среды разработки «SQL Server Management Studio»

Данное окно имеет следующую структуру:

- 1). Оконное меню – содержит полный набор команд для управления сервером и выполнения различных операций.
- 2). Панель инструментов – содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.
- 3). Панель «Обозреватель объектов» – обозреватель объектов. Обозреватель объектов – это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим сервером, так и с БД. Обозреватель объектов является основным инструментом для разработки БД.
- 4). Рабочая область. В рабочей области производятся все действия с БД, а также отображается её содержимое.

В обозревателе объектов сами объекты находятся в папках. Чтобы открыть папку необходимо щёлкнуть по знаку «+» слева от изображения папки.

Нажатие кнопки  **Создать запрос** приводит к открытию окна запросов (рис. 3).

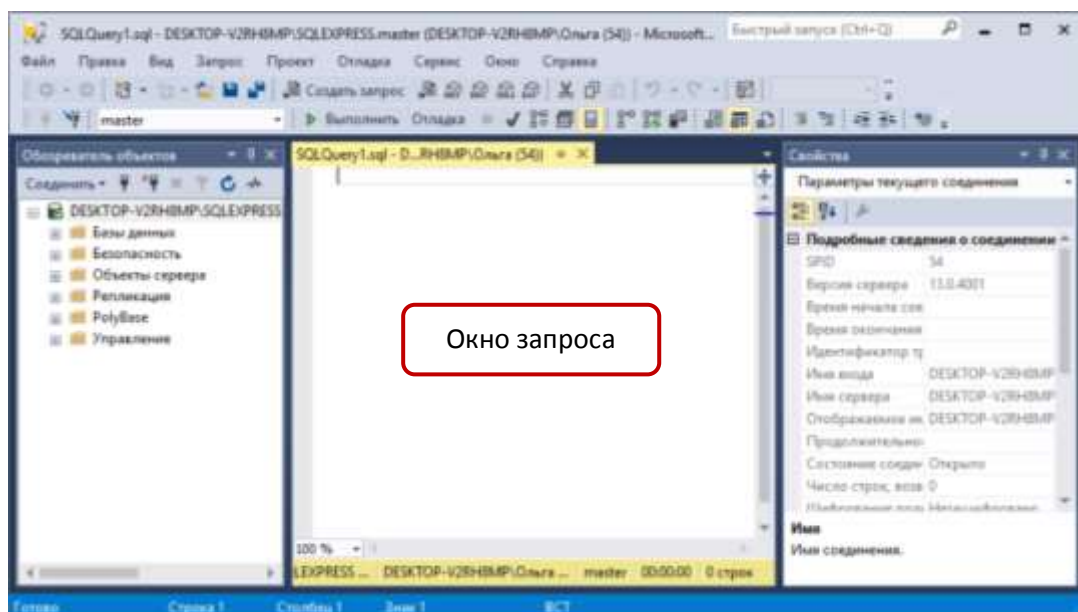



Рис. 3. Окно запросов

Для выполнения запроса, введенного в этом окне, нужно нажать на кнопку  **Выполнить** или в меню Запрос выбрать команду Выполнить.

2. В среде разработки SQL Server Management Studio создадим базу данных Artworks.

В новом окне запросов введем запрос на создание БД:

```
IF DB_ID('Artworks') IS NULL  
CREATE DATABASE Artworks;
```

Если базы данных с именем Artworks не существует, приведенный код создаст новую БД. Функция DB_ID принимает имя базы данных в качестве входного параметра и возвращает внутренний ID базы данных. Если БД с именем входного параметра не существует, функция возвращает значение NULL. Это простой способ проверить наличие заданной БД.

В инструкции CREATE DATABASE используются установочные параметры файла для задания его местоположения и начального размера.

После выполнения данного запроса в окне среды разработки SQL Server Management Studio на панели обозревателя объектов в папке «Базы данных» появится новая БД Artworks (рис. 4).

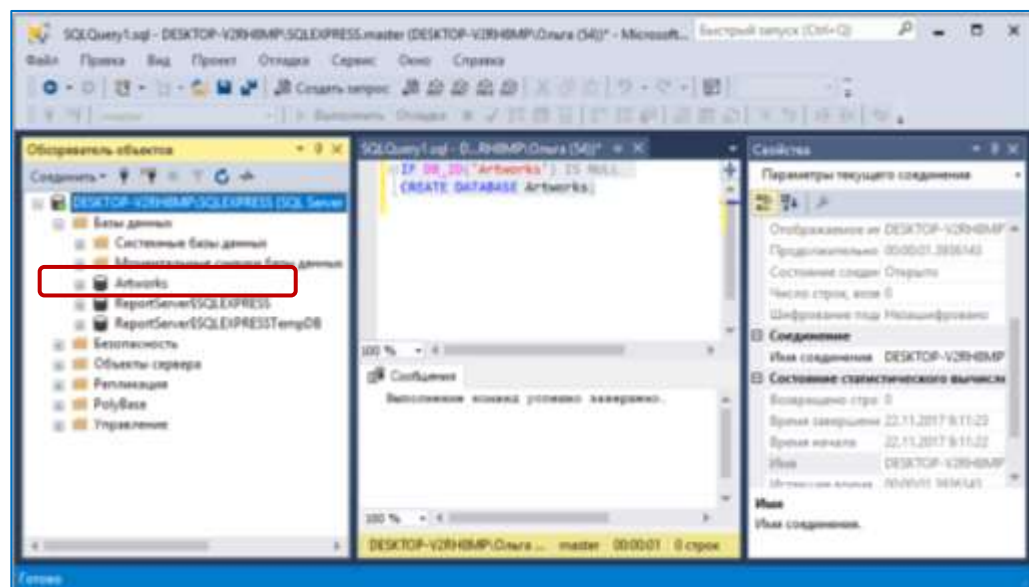


Рис. 4. Информация о создании базы данных Artworks

3. Выполним запрос информации по созданной базе данных.

```
EXEC SP_HELPDB Artworks;
```

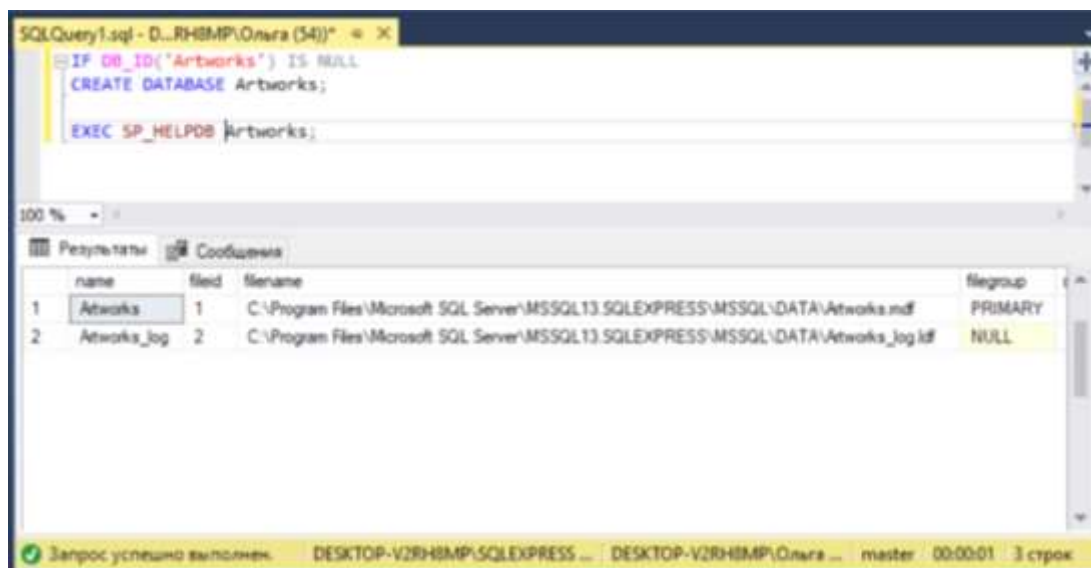


Рис. 5. Результаты запроса информации о базе данных

Выполнив запрос информации по созданной базе данных, мы получили информацию о текущем размере базы данных, ее размещении, файлах базы данных, их текущем размере, максимальном размере, шаге увеличения файла данных и др.

4. Создадим таблицы базы данных.

Вся информация в базе данных хранится в таблицах, которые представляют собой объекты хранения данных. Таблицы состоят из строк или записей и столбцов или полей. Каждое поле имеет три характеристики:

- 1). Имя поля – используется для обращения к полю.
- 2). Значение поля – определяет информацию, хранимую в поле.
- 3). Тип данных поля – определяет, какой вид информации можно хранить в поле.

В SQL сервер используются следующие типы данных (<https://docs.microsoft.com/ru-ru/sql/t-sql/data-types/data-types-transact-sql>):

Двоичные типы данных, которые содержат последовательности нулей и единиц: 1) $\text{binary}(n)$ – двоичный тип фиксированной длины размером в n байт, где n — значение от 1 до 8000; размер при хранении составляет n байт; 2) $\text{varbinary}(n)$ – двоичный тип с переменной длиной, n может иметь значение от 1 до 8000, **max** указывает, что максимальный размер при хранении составляет $2^{31}-1$ байт;

Целочисленные типы данных – типы данных для хранения целых чисел (в скобках указан диапазон значений типа данных): `tinyint` (0..255), `smallint` (-32 768..+32 767), `int` ($-2^{31}..+(2^{31}-1)$), `bigint` ($-2^{63}..+(2^{63}-1)$);

Типы данных для хранения чисел с плавающей запятой: `real` занимает в памяти 4 байта; `float(n)`, где *n* — это количество битов, используемых для хранения мантиссы числа в формате **float** при экспоненциальном представлении, определяет точность данных и размер для хранения; значение параметра *n* должно лежать в пределах от 1 до 53; значением по умолчанию для параметра *n* является 53;

Типы данных для хранения чисел с фиксированной точностью и масштабом: `decimal(p, s)` и `numeric(p, s)`, где *p* (точность) – максимальное количество десятичных разрядов числа (как слева, так и справа от десятичной запятой). Точность должна быть значением в диапазоне от 1 до 38. По умолчанию это значение равно 18. *s* (масштаб) – максимальное количество десятичных разрядов числа справа от десятичной запятой. Масштаб может принимать значение от 0 до *p* и может быть указан только совместно с точностью. По умолчанию масштаб принимает значение 0; поэтому $0 \leq s \leq p$;

Символьные типы данных: `char(n)` – строковые данные фиксированной длины не в Юникоде, аргумент *n* определяет длину строки и должен иметь значение от 1 до 8000, размер при хранении составляет *n* байт; `varchar(n | max)` – строковые данные переменной длины не в Юникоде, аргумент *n* определяет длину строки и должен иметь значение от 1 до 8000, значение **max** указывает, что максимальный размер при хранении составляет $2^{31}-1$ байт (2 ГБ); `text` – данные переменной длины не в Юникоде в кодовой странице сервера и с максимальной длиной строки $2^{31}-1$;

Специальные типы данных: `bit` – целочисленный тип данных, который может принимать значения 1, 0 или NULL; `image` – тип данных для хранения рисунка размером до 2ГБ;

Типы данных даты и времени: date (от 01.01.0001 до 31.12.9999); datetime (диапазон даты – от 01.01.1753 до 31.12.1999, диапазон времени – от 00:00:00 до 23:59:59,997); smalldatetime (диапазон даты – от 01.01.1900 до 6.06.2079, диапазон времени – от 00:00:00 до 23:59:59); time (от 00:00:00.0000000 до 23:59:59.9999999);

Денежные типы данных для хранения финансовой информации: money (8 байт) и smallmoney (4 байта) – типы данных, представляющие денежные (валютные) значения.

Для создания таблиц в SQL Server в первую очередь необходимо сделать активной ту БД, в которой создается таблица. Для этого в новом запросе можно набрать команду: USE <Имя БД>, либо на панели инструментов необходимо выбрать рабочую БД. После выбора БД можно создавать таблицы.

Таблицы создаются командой:

```
CREATE TABLE  
  
[ database_name . [ schema_name ] . | schema_name . ] table_name  
  
( { <column_definition> } [ ,...n ] )  
  
[ ; ]
```

Здесь:

- Database_name – имя базы данных.
- Schema_name – имя схемы, схема — это контейнер, который содержит таблицы, представления, процедуры и др. Они находятся в базе данных, которая расположена на сервере. Эти сущности (сервер и схема) соответствуют друг другу, как вложенные ячейки. Сервер — это внешняя ячейка, а схема — внутренняя. Она хранит все защищаемые объекты: таблицы; представления; процедуры; функции; ограничения и др. Но схема не может содержать другую ячейку. Каждый защищаемый объект в конкретной схеме должен иметь уникальное имя. Полностью указанное имя защищаемого объекта, содержавшегося в схеме, включает имя этой схемы. Таким образом, каждая схема — отдельное пространство имен, существующее независимо от пользователя базы данных, создавшего

ее. Для создания схемы используется команды CREATE SCHEMA (<https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-schema-transact-sql>)

- Table_name – имя таблицы.
- Column_definition – описание столбца.

```
<column_definition> ::=  
column_name <data_type>  
[ CONSTRAINT constraint_name [ DEFAULT constant_expression ] ]  
[ IDENTITY [ ( seed, increment ) ] ]  
[ NULL | NOT NULL ]  
[ <column_constraint> [ ...n ] ]  
[ <column_index> ]
```

Здесь:

column_name – имя столбца в таблице;

data_type – тип данных для столбца;

CONSTRAINT – необязательное ключевое слово, указывающее на начало определения ограничения, constraint_name – имя ограничения;

IDENTITY указывает, что новый столбец является столбцом идентификаторов, при этом seed – значение, используемое для самой первой строки, загружаемой в таблицу, increment – значение приращения, добавляемое к значению идентификатора предыдущей загруженной строки;

NULL | NOT NULL определяет, допустимы ли для столбца значения NULL.

Если имя поля содержит пробел, то оно заключается в квадратные скобки.

Создадим таблицу Artworks, содержащую поля:

- а) код произведения (ArtworkId);
- б) название произведения (Title);
- в) жанр (Genre);
- г) средства создания (Tools);
- д) код автора (AuthorId);

- е) дата создания (CreatDate);
- ж) цена (Price);
- з) отдел хранения (DepId).

SQL-запрос для создания этой таблицы имеет следующий вид:

```
USE Artworks;  
IF OBJECT_ID('dbo.Artworks', 'U') IS NOT NULL  
DROP TABLE dbo.Artworks;  
CREATE TABLE dbo.Artworks  
(  
    ArtworkId BIGINT IDENTITY(1,1) CONSTRAINT PK_Artworks PRIMARY KEY,  
    Title VARCHAR(100) NULL,  
    Genre VARCHAR (50) NULL,  
    Tools VARCHAR (50) NULL,  
    AuthorId BIGINT NULL,  
    CreatDate DATE NULL,  
    Price MONEY NULL,  
    DepId INT NOT NULL  
);
```

Инструкция `USE` изменяет текущую связь с БД на связь с `Artworks`. Включение этой инструкции в сценарии создания объектов очень важно, т.к. гарантирует создание объектов в требуемой БД.

Инструкция `IF` запускает функцию `OBJECT_ID`, которая в качестве входных параметров принимает имя объекта и его тип. Тип `'U'` представляет пользовательские таблицы. Данная функция возвращает внутренний ID объекта, если объект с заданным именем и типом уже существует, и значение `NULL` в противном случае.

При создании таблицы используется схема с именем `dbo`, которая создается автоматически в каждой базе данных.

Для каждого атрибута сущности `Artworks` задается его имя, тип и допустимость значений `NULL`.

Для столбца `ArtworkId` определено ограничение в виде первичного ключа (`PK_Artworks`), при этом значения `ArtworkId` будут начинаться с 1 и увеличиваться при каждом добавлении новых строк в таблицу тоже на 1 (`IDENTITY(1,1)`).

Аналогично создайте еще три таблицы БД Artworks:

- **Authors (Авторы):**
 - AuthorId (Автор_ID, первичный ключ PK_Authors, нумерация с 1),
 - Lastname (Фамилия, NOT NULL),
 - Firstname (Имя, NOT NULL),
 - Middlename (Отчество, NULL),
 - DateOfBirth (Дата рождения, тип DATE, NULL),
 - DateOfDeath (Дата смерти, тип DATE, NULL),
 - Country (Страна, NULL).
- **Employees (Сотрудники):**
 - EmpId (Сотрудник_ID, первичный ключ PK_Employees, нумерация с 1),
 - Lastname (Фамилия, NOT NULL),
 - Firstname (Имя, NOT NULL),
 - Middlename (Отчество, NULL),
 - Position (Должность, NULL),
 - Salary (Зарплата, тип MONEY, NULL),
 - BeginDate (Дата зачисления, тип DATE, NOT NULL),
 - EndDate (Дата увольнения, тип DATE, NULL),
 - DepId (Индекс отдела, NULL).
- **Departments:**
 - DepId (Отдел_ID, первичный ключ PK_Dep, нумерация с 1),
 - Name (Название отдела, NOT NULL).

5. Свяжем таблицы базы данных.

Чтобы обеспечить ссылочную целостность в БД Artworks нужно добавить в созданные таблицы ограничение по внешним ключам.

Таблицы Artworks и Authors нужно связать по столбцу AuthorId, а таблицы Artworks и Departments – по столбцу DepId. Аналогично должны быть связаны между собой таблицы Employees и Departments.

Создадим новый запрос, связывающий таблицы Artworks и Authors по столбцу AuthorId:

```
USE Artworks;  
ALTER TABLE dbo.Artworks  
ADD CONSTRAINT FK_Artw_Auth FOREIGN KEY (AuthorId)  
REFERENCES Authors (AuthorId);
```

Свяжите по столбцу DepId таблицы Artworks и Departments, Employees и Departments.

6. Теперь, когда установлены связи между всеми таблицами, можно создать диаграмму, описывающую эти взаимосвязи.

Для создания диаграммы нужно щелкнуть правой кнопкой мыши на элементе БД «Диаграммы баз данных» и в открывшемся контекстном меню выбрать пункт «Создать диаграмму базы данных» (рис. 6).

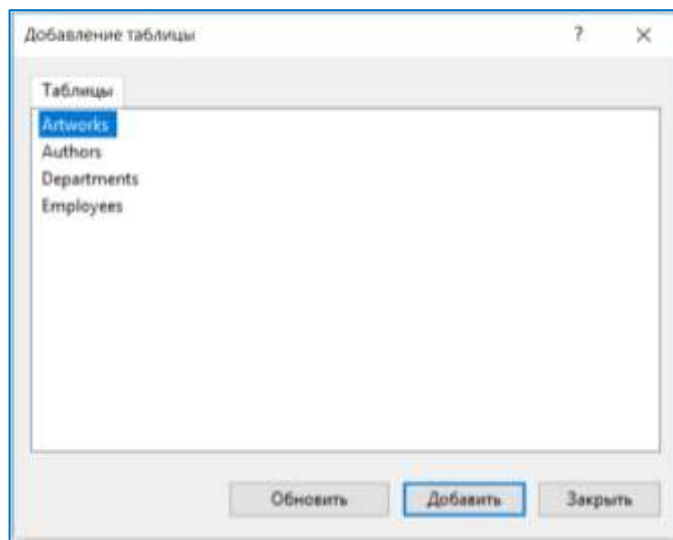


Рис. 6. Окно «Добавление таблицы»

В открывшемся диалоговом окне нужно выбрать таблицы, которые будут добавлены на диаграмму. Выберем все таблицы БД Artworks. В результате будет получена диаграмма следующего вида (рис. 7):

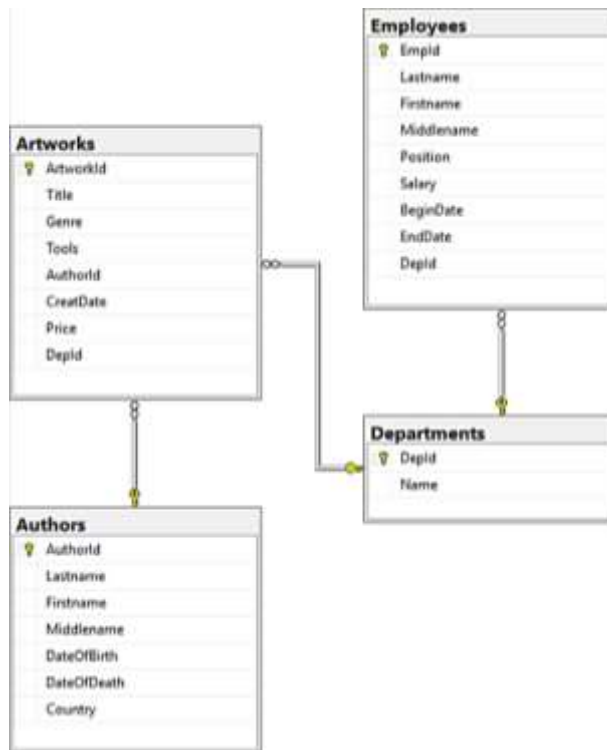


Рис. 7. Диаграмма базы данных Artworks

7. Добавим данные в таблицы, используя инструкцию INSERT.

```

INSERT INTO [ database_name . [ schema_name ] . | schema_name . ]
table_name
[ ( column_name [ ,...n ] ) ]
{
VALUES ( { NULL | expression } )
| SELECT <select_criteria>
}
[ OPTION ( <query_option> [ ,...n ] ) ]
[;]

```

Добавим данные в таблицу Departments:

DepID	Name
1	Imaginative literature
2	Poesy
3	Fantasy

Создадим новый запрос для добавления записей, добавим записи в таблицу Departments:

USE Artworks;

```

INSERT dbo.Departments (Name)
VALUES ('Imaginative literature');
INSERT dbo.Departments (Name)
VALUES ('Poesy');
INSERT dbo.Departments (Name)
VALUES ('Fantasy');

```

Создайте запросы для добавления данных в таблицы: Employees; Authors; Artworks (см. Приложение 1, Таблица 1, Таблица 2, Таблица 3).

Пример 1. Ввод строки в таблицу Employees:

```

USE Artworks;
INSERT dbo.Employees (Lastname, Firstname, Middlename, Position,
Salary, BeginDate, EndDate, DepId)
VALUES ('Петрова', 'Анна', 'Ивановна', 'ведущий специалист', 40000,
'20.12.2016', Null, 1);

```

8. Используя инструкцию UPDATE, изменим значения некоторых данных в таблицах БД Artworks.

```

UPDATE [ database_name . [ schema_name ] . | schema_name . ] table_name
SET { column_name = { expression | NULL } } [ ,...n ]
[ FROM from_clause ]
[ WHERE <search_condition> ]
[ OPTION ( LABEL = label_name ) ]
[:]

```

Заменим имя отдела 'Poesy' на 'Russian poetry' в таблице Departments, создав новый запрос:

```

USE Artworks;
UPDATE dbo.Departments
SET Name = 'Russian poetry'
WHERE DepID = 2;

```

Замените в таблице Artworks для романа «Три мушкетера» жанр «Историко-приключенческий роман» на «Роман».

9. Используя инструкцию ALTER TABLE из таблицы Artworks удалим столбец Tools.

Создадим новый запрос:

```

USE Artworks;
ALTER TABLE dbo.Artworks DROP COLUMN Tools;

```

10. Выведите на экран записи таблиц базы данных Artworks. Создайте новый запрос:

```
USE Artworks;  
SELECT * FROM dbo.Artworks;  
SELECT * FROM dbo.Authors;  
SELECT * FROM dbo.Employees;  
SELECT * FROM dbo.Employees;
```

11. Покажите выполненную работу преподавателю.

Таблица 1. Employees

EmpID	Lastname	Firstname	Middlename	Position	Salary	BeginDate	EndDate	DepId
1	Иванов	Антон	Сергеевич	начальник отдела	50000	21.01.2010	15.07.2017	1
2	Петрова	Анна	Ивановна	ведущий специалист	40000	20.12.2016		1
3	Сидоров	Николай	Павлович	специалист	30000	15.07.2017		1
4	Андреева	Любовь	Николаевна	начальник отдела	50000	01.05.2011		2
5	Прохоров	Евгений	Алексеевич	ведущий специалист	45000	30.12.2014		2
6	Левичев	Даниил	Иванович	начальник отдела	50000	18.04.2012		3
7	Сухарев	Антон	Андреевич	ведущий специалист	40000	24.09.2015		3

Таблица 2. Authors

AuthorID	Lastname	Firstname	Middlename	DateOfBirth	DateOfDeath	Country
1	Ахматова	Анна	Андреевна	23.06.1889	5.03.1966	Россия
2	Бродский	Иосиф	Александрович	24.05.1940	28.01.1996	Россия
3	Горбовский	Глеб	Яковлевич	4.10.1931		Россия
4	Пушкин	Александр	Сергеевич	6.06.1799	10.02.1837	Россия
5	Верн	Жюль		8.02.1828	24.03.1905	Франция
6	Дюма	Александр		24.07.1802	5.12.1870	Франция
7	Толкин	Джон Рональд Руэл		3.01.1892	2.09.1973	Англия

Таблица 3. Artworks

ArtworkID	Title	Genre	Tools	AuthorID	CreatDate	Price	DepID
1	Властелин колец	Фэнтези		7	29.07.1954	772	3
2	Руслан и Людмила	Поэма		4	5.03.1820	400	1
3	Три мушкетера	Историко-приключенческий роман		6	15.02.1844	1000	1
4	Вокруг света за 80 дней	Роман		5	7.06.1872	800	3

Лабораторная работа

Тема: Чтение данных базы данных

Цель работы: Научиться использовать инструкцию SELECT для чтения данных из базы данных.

Теоретические сведения

Для формирования запросов на выборку данных в SQL используется оператор SELECT. Его формат представлен ниже:

```
SELECT [ALL | DISTINCT] select_item_commalist  
FROM table_reference_commalist  
[WHERE conditional_expression]  
[GROUP BY column_name_commalis]  
[HAVING conditional_expression]  
[ORDER BY order_item_commalist]
```

SELECT позволяет выбирать данные из одной или нескольких таблиц, выполнять группировку, обработку данных с помощью агрегатных функций, формировать вложенные запросы и др.

Инструкция SELECT выполняется целиком, а не построчно. В несколько обобщенном виде схема выполнения оператора SELECT следующая:

- 1). Выполняется раздел FROM.
- 2). Выполняется раздел WHERE, если он присутствует в инструкции.
- 3). Выполняются GROUP BY, если он присутствует в инструкции.
- 4). Выполняется раздел HAVING, если он присутствует в инструкции.
- 5). Выполняются определения в разделе SELECT.
- 6). Выполняются ORDER BY, если он присутствует в инструкции.

Задания

Запустите среду разработки «SQL Server Management Studio».

1. Чтение данных из одной таблицы (включите в отчет по лабораторной работе скриншоты выполненных заданий):
 - a. Из таблицы Artworks БД Artworks выберите все романы.

```
USE Artworks;  
SELECT * FROM dbo.Artworks  
WHERE Genre= 'Роман';
```
 - b. Из таблицы Authors БД Artworks выведите данные о русских и французских авторах (фамилия, имя, отчество):

```
USE Artworks;
SELECT Lastname, Firstname, Middlename
FROM dbo.Authors
WHERE Country<>'Англия';
```

- c. Из таблицы Employees БД Artworks выведите фамилии начальников отделов.
- d. Из таблицы Employees БД Artworks выведите фамилии и имена специалистов и ведущих специалистов в алфавитном порядке.
- e. Из таблицы Employees БД Artworks выведите фамилии, имена и оклады сотрудников, зарабатывающих меньше 50 000 рублей в месяц. Отсортируйте список отобранных сотрудников по возрастанию оклада.
- f. Из таблицы Employees БД Artworks выведите фамилии, имена и оклады сотрудников, зарабатывающих меньше 50 000, но больше 30 000 рублей в месяц. Отсортируйте список отобранных сотрудников по убыванию оклада.
- g. Из таблицы Artworks БД Artworks выберите все произведения, написанные в 19 веке.

2. Чтение данных из нескольких таблиц (<http://www.kpress.ru/cs/2009/3/join/join.asp>):

- a. Из таблицы Artworks БД Artworks выведите список произведений (название, жанр) с указанием автора произведения (фамилия, имя, отчество), отсортируйте список в алфавитном порядке названий:

```
USE Artworks;
SELECT Artworks.Title, Artworks.Genre, Authors.Lastname,
Authors.Firstname, Authors.Middlename
FROM dbo.Artworks, dbo.Authors
WHERE Artworks.AuthorId=Authors.AuthorId
ORDER BY Title;
```

ИЛИ

```
USE Artworks;
SELECT Artworks.Title, Artworks.Genre, Authors.Lastname,
Authors.Firstname, Authors.Middlename
FROM dbo.Artworks JOIN dbo.Authors
ON Artworks.AuthorId=Authors.AuthorId
ORDER BY Title;
```

- b. Из таблицы Artworks БД Artworks выведите список произведений (название, жанр) с указанием названий отделов, по которым проходят эти произведения.
- c. Используя внешнее соединение (LEFT JOIN или RIGHT JOIN), выведите список всех авторов с указанием произведений, написанных ими (если произведения указаны в таблице Artworks), иначе с пустым перечнем произведений. Отсортируйте список в алфавитном порядке фамилий авторов.
- d. Используя внешнее соединение, выведите список названий всех отделов с указанием произведений, зарегистрированных в этих отделах (при наличии таковых). Отсортируйте список по названиям отделов.
- e. Напишите запрос, отбирающий следующие данные (см. рис. 1):

Результаты		Сообщения	
	Title	Lastname	Name
1	Властелин колец	Толкин	Fantasy
2	Вокруг света за 80 дней	Верн	Fantasy
3	Руслан и Людмила	Пушкин	Imaginative literature
4	Три мушкетера	Дюма	Imaginative literature

Рис. 1. Результат запроса

В запросе используется соединение трех таблиц: Artworks (поле Title); Authors (поле Lastname); Department (поле Name). Описание соединения можно посмотреть по ссылке: [https://technet.microsoft.com/ru-ru/library/ms191430\(v=sql.105\).aspx](https://technet.microsoft.com/ru-ru/library/ms191430(v=sql.105).aspx)

3. Группировка данных

Добавьте в таблицу Artworks еще одно произведение Пушкина – «Дубровский», DepID – 2.

```
USE Artworks;
```

```
INSERT Artworks (Title, AuthorId, DepId) VALUES ('Дубровский', 4, 2);
```

- a. Напишите запрос, который считает количество произведений по каждому из авторов (см. рис. 2).

The screenshot shows a SQL query in a text editor and its results in a table. The query is as follows:

```
USE Artworks;
SELECT Authors.Lastname, Authors.Firstname, Authors.Middlename, COUNT(Artworks.Title) As Count
FROM Artworks
JOIN Authors
ON Artworks.AuthorId=Authors.AuthorId
GROUP BY Authors.Lastname, Authors.Firstname, Authors.Middlename
ORDER BY Authors.Lastname;
```

The results are displayed in a table with the following data:

	Lastname	Firstname	Middlename	Count
1	Верн	Жюль	NULL	1
2	Дюма	Александр	NULL	1
3	Пушкин	Александр	Сергеевич	2
4	Толкин	Джон Рональд Руэл	NULL	1

Рис. 2. Запрос с групповой операцией

- b. Измените предыдущий запрос, отобрав только тех авторов, у которых в базе больше одного произведения (используется раздел HAVING).
- c. Напишите запрос, подсчитывающий количество авторов из каждой страны.
- d. Напишите запрос, который считает суммарную зарплату (агрегатная функция SUM) сотрудников по каждому из отделов.
- e. Измените предыдущий запрос, показав зарплату сотрудников отдела Fantasy.
- f. Измените предыдущий запрос, показав зарплаты отделов большие 90000.
- g. Измените предыдущий запрос, показав максимальные зарплаты (агрегатная функция MAX) в каждом отделе.
- h. Измените предыдущий запрос, показав минимальную зарплату (агрегатная функция MIN) в каждом отделе.
- i. Измените предыдущий запрос, показав минимальную зарплату (агрегатная функция AVG) в каждом отделе.

Лабораторная работа

Тема: Создание и настройка разрешений на объекты базы данных

Цель работы: научиться создавать объекты базы данных (представления, хранимые процедуры) и настраивать для них разрешения.

Задания

Запустите среду разработки «SQL Server Management Studio».

1. Создание представлений ([https://msdn.microsoft.com/ru-ru/library/ms187956\(v=sql.120\).aspx](https://msdn.microsoft.com/ru-ru/library/ms187956(v=sql.120).aspx)):

- a. Создайте в базе данных Artworks представление с именем vw_Employees, которое содержит следующие поля из таблиц Employees и Departments: Employees.Lastname; Employees.Firstname; Employees.Middlename; Employees.Position; Departments.Name.

```
USE Artworks;  
GO
```

```
IF OBJECT_ID ('vw_Employees', 'V') IS NOT NULL  
DROP VIEW vw_Employees;  
GO
```

```
CREATE VIEW dbo.vw_Employees  
AS  
SELECT Employees.Lastname, Employees.Firstname, Employees.Middlename,  
Employees.Position, Departments.Name  
FROM dbo.Employees  
JOIN dbo.Departments  
ON Employees.DepId=Departments.DepId;  
GO
```

Посмотрите, какие данные отбираются при помощи данного представления:

```
USE Artworks;  
GO
```

```
SELECT * FROM dbo.vw_Employees;  
GO
```

- b. Создайте представление vw_Books, выводящее из БД Artworks список произведений (название, жанр) с указанием названий отделов, по которым проходят эти произведения. Список должен быть отсортирован в алфавитном порядке названий произведений (отсортировать список можно

с использованием оператора TOP, <http://2sql.ru/novosti/sql-top/>). Посмотрите, какие данные отбираются при помощи созданного представления (см. рис. 1).

	Title	Genre	Name
1	Властелин колец	Фэнтези	Fantasy
2	Вокруг света за 80 дней	Роман	Fantasy
3	Дубровский	NULL	Russian poetry
4	Руслан и Людмила	Позма	Imaginative literature
5	Три мушкетера	Роман	Imaginative literature

Рис. 1. Представление vw_Books

- с. Создайте представление vw_Salaries, выводящее из БД Artworks список суммарных зарплат в каждом из отделов. Список должен быть отсортирован в порядке убывания суммарных зарплат. Посмотрите, какие данные отбираются при помощи созданного представления (см. рис. 2).

	Departments Name	SUM Salaries
1	Imaginative literature	120000,00
2	Russian poetry	95000,00
3	Fantasy	90000,00

Рис. 2 Представление vw_Salaries

2. Создание хранимых процедур (<https://docs.microsoft.com/ru-ru/sql/relational-databases/stored-procedures/create-a-stored-procedure#TsqlProcedure>):

- а. Создайте хранимую процедуру pr_Salaries с входным параметром @VarSalary типа money, которая из представления vw_Salaries будет отбирать названия отделов с суммарной заработной платой большей входного параметра процедуры:

```
USE Artworks;
GO

-- @VarSalary - входной параметр процедуры типа money
CREATE PROCEDURE pr_Salaries @VarSalary money
AS
BEGIN
    SELECT [Departments Name] FROM vw_Salaries
        WHERE [SUM Salaries] > @VarSalary;
END
GO

--Запуск процедуры с параметром 90000
EXECUTE pr_Salaries 90000;
GO
```

- b. Создайте хранимую процедуру `pr_Artworks` с входным параметром `@VarAuthors` типа `varchar(25)`, которая из БД `Artworks` будет отбирать все произведения автора, указанного во входном параметре.

3. Создание имени входа для проверки подлинности SQL Server

(<https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-login-transact-sql>):

- a. Создайте имя входа `Mary` для проверки подлинности SQL Server с паролем «`Password1`».

```
CREATE LOGIN Mary WITH PASSWORD = 'Password1';
```

- b. Создайте имя входа `Ivan` для проверки подлинности SQL Server с паролем «`Password2`».

4. Предоставление и отзыв доступа к объектам базы данных

(<https://www.intuit.ru/studies/courses/1141/263/lecture/6710>)

- a. Предоставьте `Mary` доступ к следующим объектам БД `Artworks`:
 - к таблице `Atrworks` на выборку, изменение строк таблицы, вставку новых строк и удаление строк таблицы;
 - к представлению `vw_Books`;
 - к хранимой процедуре `pr_Salaries`.

```
USE Artworks;  
GO
```

```
CREATE USER Mary FOR LOGIN Mary;
```

```
GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Artworks TO Mary;  
GO
```

```
GRANT SELECT ON vw_Books TO Mary;  
GO
```

```
GRANT EXECUTE ON pr_Salaries TO Mary;  
GO
```

Зарегистрируйтесь на сервере под именем `Mary` и проверьте наличие доступа.

- b. Предоставьте для пользователя `Ivan` доступ к следующим объектам БД `Artworks`:

- к таблицам `Artworks`, `Authors`, `Departments` на выборку, изменение строк, вставку новых строк и удаление строк;
- к представлениям `vw_Books`, `vm_Employees`, `vw_Salaries`;
- к хранимым процедурам `pr_Salaries`, `pr_Artworks`.

Зарегистрируйтесь на сервере под именем `Ivan` и проверьте наличие доступа.

c. Отзовите у `Mary` доступ:

- на удаление данных из таблицы `Artworks`;
- на доступ к хранимой процедуре `pr_Salaries`.

```
USE Artworks;
GO
```

```
REVOKE DELETE ON dbo.Artworks TO Mary;
GO
```

```
REVOKE EXECUTE ON pr_Salaries TO Mary;
GO
```

Зарегистрируйтесь на сервере под именем `Mary` и проверьте наличие доступа.

d. Отзовите у пользователя `Ivan` доступ:

- к таблице `Departments`;
- к представлению `vw_Books`;
- к хранимой процедуре `pr_Salaries`;

Зарегистрируйтесь на сервере под именем `Ivan` и проверьте наличие доступа.

Практикум: Учебный проект: "Разработка ИС предприятия оптовой торговли лекарственными препаратами"

Порядок выполнения практического задания

В процессе выполнения практического задания проводится анализ и оформление результатов обследования деятельности гипотетического предприятия "МЕД", и на его основе разрабатываются документы, необходимые для настройки типовой ИС.

По итогам проведения обследования обычно формируются следующие документы:

- Предварительная информация.
- Видение выполнения проекта и границы проекта.
- Отчет об обследовании.

Предварительная информация

Предполагается, что в начале обследования проведен предварительный сбор информации о компании, по итогам которого получены следующие данные:

- Краткая информация о компании (профиль клиента).
- Цели проекта.
- Подразделения и пользователи системы.

На основе предварительной информации сформировано и согласовано с заказчиком общее представление о проекте:

Видение выполнения проекта и границы проекта - документ, который кратко описывает, в каких подразделениях и в какой функциональности будет внедряться ИС.

Затем выполняется детальное обследование предприятия, результаты которого оформляются в виде отдельного документа - отчета об обследовании.

Отчет об обследовании содержит следующие разделы:

- Анализ существующего уровня автоматизации.
- Составляется список программного обеспечения, используемого в компании, и приводятся данные об использовании этих пакетов в каждом из подразделений организации.
- Общие требования к ИС
- Формулируются общие требования к функциональности разрабатываемой системы.
- Формы документов
- Устанавливается перечень и структура документов, которые должны формироваться системой.
- Описание системы учета
- Описание системы учета включает в себя следующие документы:
 - Учетная политика компании
 - План счетов и используемых аналитик
 - Список типовых хозяйственных операций и их отражение в проводках
- Описание справочников

- По каждому справочнику, проектируемому в системе, дается описание необходимой иерархической структуры.
- Организационная диаграмма
- Организационная диаграмма используется для отражения организационной структуры подразделений предприятия и их зон ответственности.
- Описание состава автоматизируемых бизнес-процессов
- Все бизнес-процессы компании должны быть перечислены в общем списке и каждый должен иметь свой уникальный номер.
- Диаграммы прецедентов
- Для выделения автоматизируемых бизнес-процессов и их основных исполнителей используются диаграммы прецедентов.
- Физическая диаграмма
- Физическая диаграмма служит для того, чтобы описать взаимодействие организации на верхнем уровне с внешними контрагентами.
- Описания бизнес-процессов (книга бизнес-процессов).

Далее в отчет об обследовании включается книга бизнес-процессов, содержащая подробное описание автоматизируемых бизнес-процессов. Модели бизнес-процессов позволяют выделить отдельные операции, выполнение которых должно поддерживаться разрабатываемой ИС.

На последнем этапе осуществляется отображение модели предметной области на функциональность типовой системы - выбираются модули системы для поддержки выделенных операций, определяются особенности их настройки, выявляется необходимость разработки дополнительных программных элементов.

Краткая информация о компании "МЕД"

Компания - дистрибьютор "МЕД" закупает медицинские препараты отечественных и зарубежных производителей и реализует их через собственную дистрибьюторскую сеть и сеть аптек. Компания осуществляет доставку товаров как собственным транспортом, так и с помощью услуг сторонних организаций.

Основные бизнес-процессы компании - закупки, складирование запасов, продажи, взаиморасчеты с поставщиками и клиентами.

Уровень конкуренции для компании в последнее время возрос, так как на рынок вышли два новых конкурента, к которым перешла часть клиентов и ряд наиболее квалифицированных сотрудников ЗАО "МЕД". ЗАО "МЕД" имеет два филиала - в Курске и Санкт-Петербурге. Каждый филиал функционирует как самостоятельное юридическое лицо, являясь полностью принадлежащей ЗАО "МЕД" дочерней компанией.

По предварительным планам, Компания намерена открыть также дочернее предприятие для организации производства в непосредственной близости к своим заказчикам.

Адреса и телефоны

Москва, К-123 Центральная улица, д. 20, стр. 7, офис 709

Телефон: (095) 345-6789, факс: (095) 345-9876

Контактные лица

Борис Нефедьев - Генеральный директор

Дмитрий Кононов - Исполнительный директор

Артур Иванченко - Директор по маркетингу

Сотрудники

На момент проведения Диагностики штат компании составляет 110 сотрудников.

Основными целями проекта автоматизации компании "МЕД" являются:

- Разработка и внедрение комплексной автоматизированной системы поддержки логистических процессов компании.
- Повышение эффективности работы всех подразделений компании и обеспечение ведения учета в единой информационной системе.

Видение выполнения проекта и границы проекта

В рамках проекта развертывание новой системы предполагается осуществить только в следующих подразделениях ЗАО "МЕД":

- Отдел закупок;
- Отдел приемки;
- Отдел продаж;
- Отдел маркетинга;
- Группа планирования и маркетинга;
- Группа логистики;
- Учетно-операционный отдел;
- Учетный отдел;
- Отдел сертификации (в части учета сертификатов на медикаменты);
- Бухгалтерия (только в части учета закупок, продаж, поступлений и платежей).

Не рассматривается в границах проекта автоматизация учета основных средств, расчета и начисления заработной платы, управления кадрами. Выходит за рамки проекта автоматизация процессов взаимоотношений с клиентами.

Количество рабочих мест пользователей - 50.

Отчет об обследовании

Список программного обеспечения, используемого компанией на момент обследования

1. "1С: Предприятие 7.7" ("Бухгалтерия", "Торговля", "Зарплата", "Кадры", "Касса", "Банк") для работы бухгалтерии.
2. Две собственные разработки на базе конфигуратора "1С" - "Закупки" и "Продажи".
3. Собственная разработка на базе FOXPRO для финансового отдела.
4. Excel для планирования продаж.

Существующий уровень автоматизации

Количество рабочих станций, всего:	90
Количество сотрудников отдела ИТ	2
Количество ПК, одновременно работающих в сети	50
Наличие и форма связи с удаленными объектами	Терминальная связь со складом
Количество рабочих станций на удаленном объекте	8
<u>Характеристики</u> компьютеров	От Celeron 600 и выше
Операционная система	Windows 98, XP
Системы, которые представляется возможным оставить без изменения	"1С: Предприятия 7.7" в модульном составе "Бухгалтерия", "Зарплата", "Кадры", для работы бухгалтерии

Общие требования к информационной системе

Одно из основных требований компании "МЕД" к будущему решению состоит в том, чтобы оно было построено на фундаменте единой интегрированной системы, а работа всех сотрудников велась в одном информационном пространстве.

Ключевые функциональные требования к информационной системе:

1. Средства защиты данных от несанкционированного доступа. Разграничения доступа к данным в соответствии с должностными обязанностями.
2. Возможность удаленного доступа.
3. Управление запасами. Оперативное получение информации об остатках на складе.
4. Управление закупками. Планирование закупок в разрезе поставщиков.
5. Управление продажами. Контроль лимита задолженности с возможностью блокировки формирования отгрузочных документов.
6. Полный контроль взаиморасчетов с поставщиками и клиентами.
7. Получение управленческих отчетов в необходимых аналитических срезах - как детальных для менеджеров, так и агрегированных для руководителей подразделений и директоров фирмы.

Примеры форм отчетных документов

Отчет о дебиторской задолженности							
Регистрацион	Клие	Догово	Дата договор	Сумма по	Сумма задолженнос	Ожидаемы й срок	Коммента

ный номер	нт	р	а	договору	ти	платежа	рий
Итого							
Отчет о кредиторской задолженности							
Информация о материалах/комплектующих, услугах, работах	Поставщик	№ договора	Сумма по договору	Срок оплаты по договору	Дата оплаты	Сумма задолженности	Комментарий
Отчет о требуемых закупках							
Инвентарный код	Название материала/товара	Ед. измерения	Требуется закупить	Предыдущая дата приобретения			Стоимость приобретения
				Название поставщика	Дата последнего приобретения		

Описание системы учета

ЗАО "МЕД" использует типовой российский план счетов, три аналитики (контрагенты, договора, регионы).

Фрагмент плана счетов компании

Номер бухг. счета	Наименование счета
01.000	Основные средства
02.000	Амортизация основных средств
03.000	Доходные вложения в материальные ценности
04.000	Нематериальные активы

05.000	Амортизация нематериальных активов
08.000	Вложения во внеоборотные активы
10.000	Материалы
10.100	Сырье и материалы
10.200	Прочие материалы
10.300	Инвентарь и хозяйственные принадлежности
14.000	Резервы под снижение стоимости МЦ
16.000	Отклонение в стоимости МЦ
19.000	НДС по приобретениям
...	...

Фрагмент учетной политики

Выручка и прибыль. Выручка от реализации продукции и оказания услуг определяется по мере отгрузки реализованной продукции, оказания услуг и отражается в финансовой отчетности по методу начисления.

Запасы. Компания с целью определения фактической себестоимости товаров, реализованных в отчетном периоде, использует вариант их оценки по себестоимости первых по времени приобретения материалов (ФИФО).

Описание справочников

Фрагмент описания справочников, используемых для автоматизации компании "МЕД", приведен в таблице.

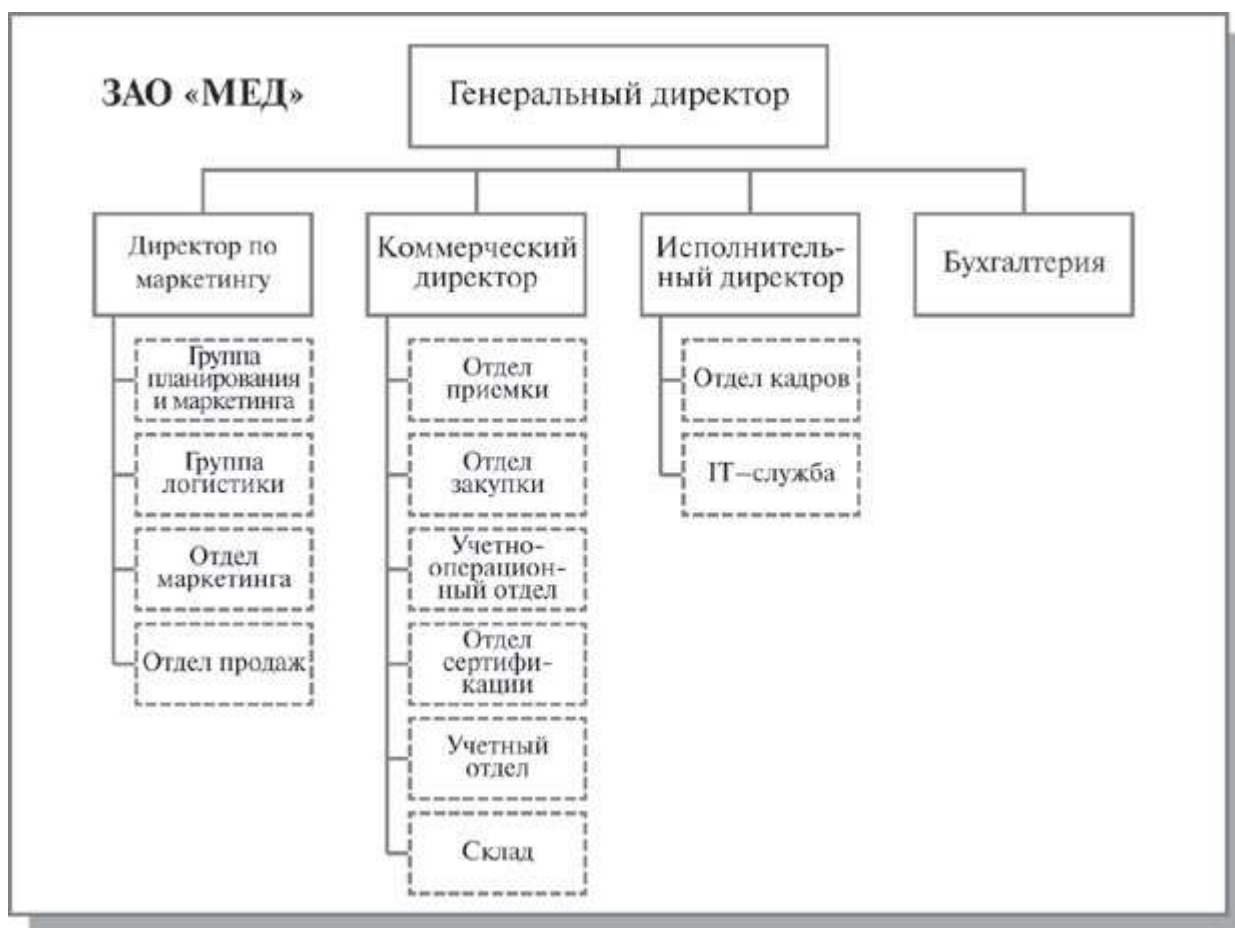
№	Наименование справочника	Код	Наименование
	Клиенты		
		AC_Ap_00001	Покупатель_АПТЕКИ
		AC_Ds_00001	Покупатель_Дистрибьютеры
		OTHER_00001	Прочие
	Поставщики/подрядчики		

Договора		B_00001	Банки
		L_00001	Частные лица
		I_0001	Страховые организации
		OTHER_00001	Прочие
	1 - наши услуги	1_COM_D/M/E	Договор комиссии_Д/М/Г
		1_SERV_D/M/E	Договор на оказание наших услуг_Д/М/Г
2 - услуги нам		2_COM_D/M/E	Договор комиссии_Д/М/Г по услугам нам
		2_SERV_D/M/E	Договор на указание Г услуг нам_Д/М
		2_COM_D/M/E	Договор комиссии_Д/М/Г по услугам нам

Код справочника отражает уровни иерархии. Справочники клиентов и договоров имеют трехуровневую структуру. Справочник поставщиков - двухуровневую структуру. В коде справочника для отображения уровня применен символ подчеркивания. Например, в коде справочника клиентов первый уровень обозначен символами "АС"-покупатель; второй уровень - "Ар"-аптеки, "Ds"-дистрибьюторы; для обозначения третьего уровня предусмотрены пятизначные порядковые номера 00001, 00002 и т.д.

Организационная диаграмма

Оргструктура предприятия оптовой торговли ЗАО "МЕД" имеет следующий вид:



Описание состава автоматизируемых бизнес-процессов

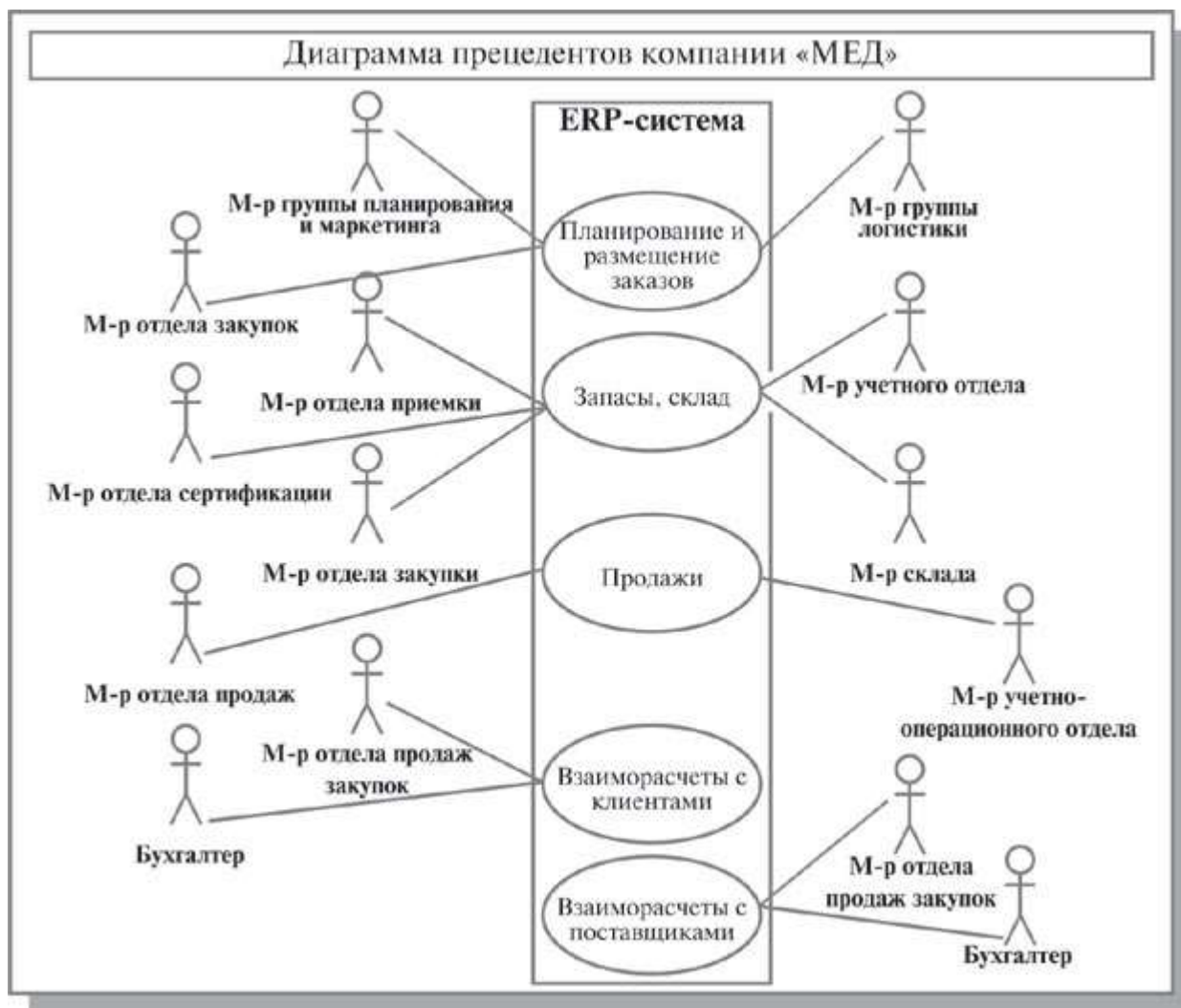
Бизнес-процессы компании, подлежащие автоматизации, приведены в следующей таблице:

№ п.п	Код бизнес-процесса	Наименование бизнес-процесса
1.	Закуп-1	Закупки
2.	Склад-2	Запасы-Склад
3.	Прод-3	Продажи
4.	Врасч-4	Взаиморасчеты с поставщиками и клиентами

Каждый бизнес-процесс имеет свой уникальный номер. Нумерация бизнес-процессов построена по следующему принципу: "префикс-номер", где префикс обозначает группу описываемых бизнес-процессов, а номер - порядковый номер бизнес-процесса в списке.

Диаграмма прецедентов компании "МЕД"

На Диаграмме прецедентов представлены автоматизируемые бизнес-процессы компании и их исполнители.



Разработка моделей бизнес-процессов предприятия оптовой торговли лекарственными препаратами

Термины

Внешняя статистика продаж - статистика по продажам, получаемая из сети аптек;

Внутренняя статистика продаж - статистика по продажам, получаемая из отчетов продаж клиентам компании;

Номенклатурная единица - наименование медикамента, завода-изготовителя;

ABC - классификация товара по выручке от продаж клиентам;

XYZ - классификация товара по рейтингу популярности;

Учетная цена - цена товара у поставщика с учетом скидок;

Действующие контракты - контракты, по которым имеются обязательства сторон на определенный период времени;

График поставок - очередность обращения к поставщикам, необходимая для поддержания деловых отношений;

Страховой запас - минимальный запас товара, необходимый для покрытия потребностей до момента поставки новой партии товара.

Разработка информационных систем включает в себя несколько этапов. Однако всегда начальным этапом создания системы является изучение, анализ и моделирование деятельности заказчика.

Для того чтобы описать взаимодействие компании на верхнем уровне с внешними контрагентами, составляется физическая диаграмма. Для составления физической диаграммы в ходе первого интервью необходимо выяснить, кто является внешними контрагентами и какие у них основные функции.

Задание 1. Формирование физической диаграммы

Составьте физическую диаграмму в соответствии с описанием деятельности компании дистрибьютора МЕД.

Компания дистрибьютор "МЕД" закупает медицинские препараты отечественных и зарубежных производителей и реализует их через собственную дистрибьюторскую сеть и сеть аптек. Планирование закупок компания осуществляет на основании статистики продаж, которую предоставляют сеть аптек и дистрибьюторы. Компания осуществляет доставку медикаментов как собственным транспортом, так и с помощью услуг сторонних организаций. Компания имеет собственный склад для хранения медикаментов.

Выполнение задания 1

Компания осуществляет закупки у отечественных и зарубежных производителей, следовательно, контрагентами компании являются отечественные и зарубежные поставщики медикаментов. Компания пользуется услугами транспортных компаний для доставки медикаментов. Следовательно, транспортные компании являются внешними контрагентами. Кроме того, компания реализует медикаменты через дистрибьюторскую сеть и сеть аптек. Следовательно, контрагентами компании являются покупатели (дистрибьюторы, аптеки). Таким образом, внешними контрагентами компании "МЕД" являются поставщики (отечественные, зарубежные), покупатели (дистрибьюторы, аптеки), транспортные компании.

На физической диаграмме компания изображается прямоугольником, для отображения контрагентов используются графический символ Actor (фигурка человечка). Для изображения связей между компанией и контрагентами используются линии (Communications). Взаимодействия компании и внешних контрагентов должны быть поименованы, чтобы были понятны функции контрагентов по отношению к компании при знакомстве с физической диаграммой.

Создание физической диаграммы в MS Visio:

1. Запустите MS Visio. (Кнопка "Пуск"/ "Программы" / MS Visio).
2. Появится окно, в котором необходимо выбрать папку **Software/ UML Model Diagram**. В открывшемся списке форм (Shapes) для построения физической

диаграммы следует выбрать пункт **UML Use Case**. В результате проделанных действий на экране появится окно, в левой части которого будет отображен набор графических символов, а в правой части - лист для рисования диаграммы ([рис.1](#)).

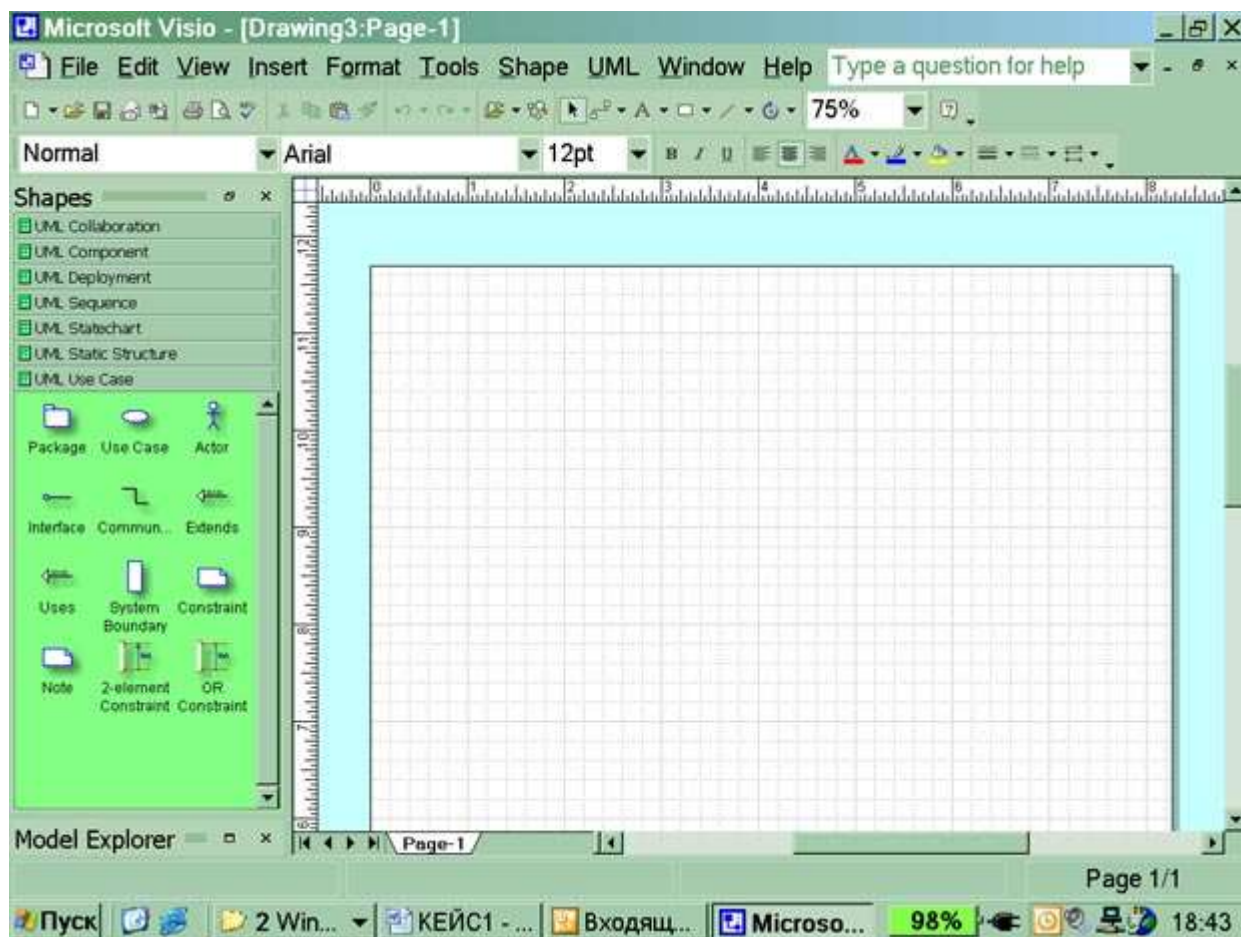



Рис. 1. Общий вид окна MS Visio

3. Для изображения прямоугольника на панели инструментов "Стандартная" найдите и зафиксируйте щелчком мыши пиктограмму с изображением прямоугольника. Затем, при нажатой правой клавиши мыши вы сможете нарисовать произвольного размера прямоугольник.
4. Для изображения на диаграмме контрагентов следует воспользоваться графическим символом с изображением человечка . Графический символ переносится на рабочее поле мышкой при нажатой правой клавише.

Примечание. Для последующего перемещения графических символов по рабочему полю необходимо зафиксировать пиктограмму Pointer Tool с изображением стрелки, размещенную на панели инструментов "Стандартная". Только после этого графический символ будет доступен для перемещения его мышкой.

5. Соедините линиями изображение каждого контрагента с прямоугольником. Для этого на панели инструментов "Стандартная" щелчком мыши зафиксируйте пиктограмму с изображением линии Line Tool и при нажатой левой клавише мышки осуществите соединение фигур.

6. Внесите наименования контрагентов "Покупатели (аптеки)", "Покупатели (дистрибьюторы)", "Поставщики (Россия)", "Поставщики (импорт)", "Транспортные компании". Для того чтобы внести надписи на диаграмме, необходимо на панели инструментов "Форматирование" зафиксировать пиктограмму Text Tool (символ буквы "А"). Щелкните мышкой на изображении человечка, курсор установится на поле с надписью Actor. Введите в это поле наименование контрагента.
7. Введите наименование компании "МЕД" в нарисованный прямоугольник, щелкнув мышкой по прямоугольнику. Обратите внимание на то, что при этом должна быть активна пиктограмма Text Tool (символ буквы "А").
8. Аналогичным образом внесите надписи к линиям соединения фирмы и контрагентов.

Физическая диаграмма ЗАО "МЕД" представлена на [рисунке 2](#).

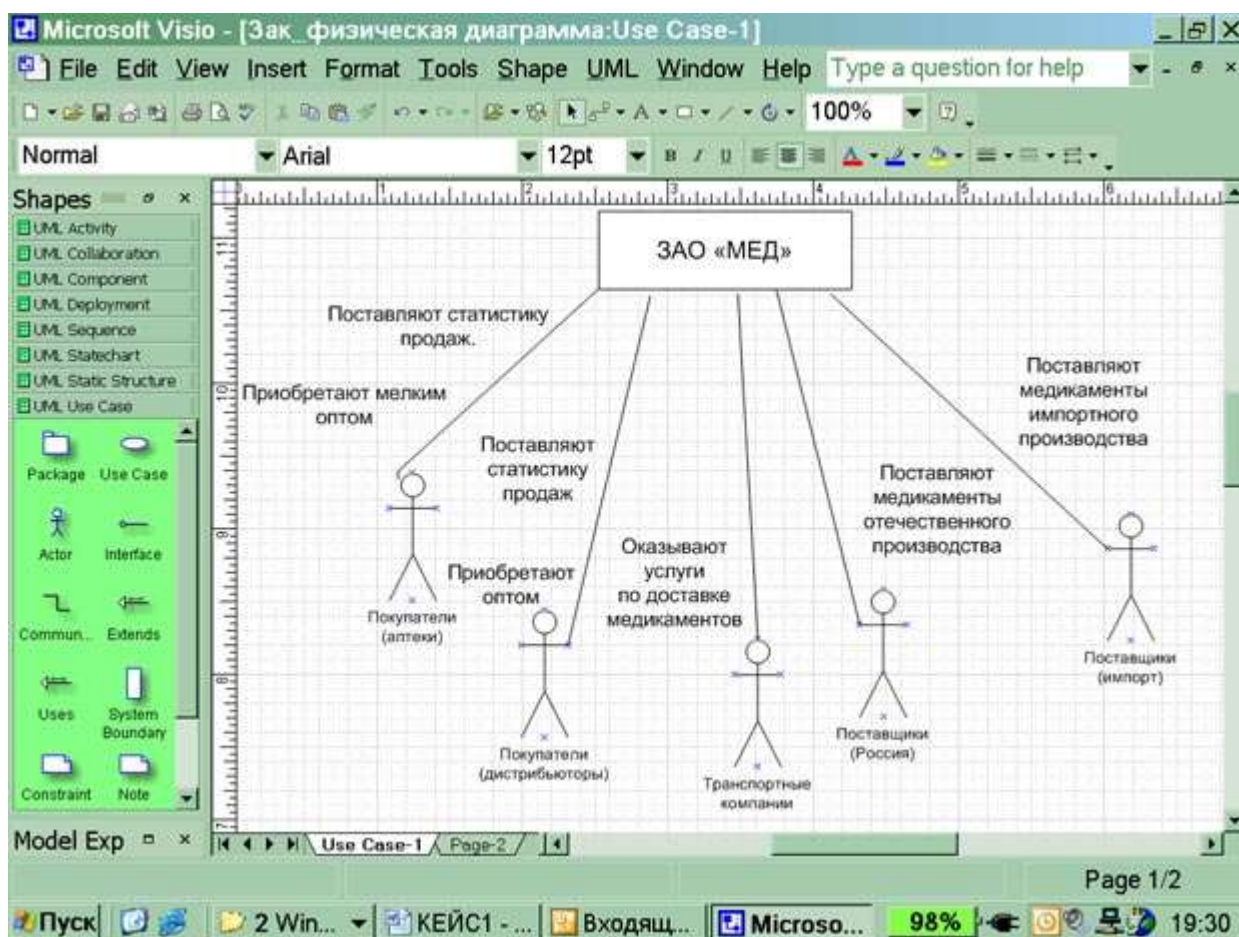


Рис. 2. Физическая диаграмма ЗАО "МЕД"

Задание 2. Формирование списка бизнес-процессов

На основании описания деятельности компании, изложенного в Задании №1, выделите основные бизнес-процессы и занесите их краткое наименование в таблицу со следующим содержанием:

Номер бизнес-процесса	Наименование бизнес-процесса
-----------------------	------------------------------

Номер бизнес-процесса составьте из букв и цифр так, чтобы по номеру был интуитивно понятен смысл бизнес-процесса.

Выполнение задания 2

Для того чтобы выделить бизнес-процессы, необходимо выделить действия, которые совершает компания. В рассматриваемом случае компания планирует закупки, закупает медикаменты, доставляет медикаменты на склад, приходит медикаменты на склад, продает медикаменты. Пример заполнения таблицы бизнес-процессов:

Номер бизнес-процесса	Название бизнес-процесса
1Пл_Зак	Планирование закупок
2-Запкк	Закупки
3-Доствк	Доставка
4-Склад	Запасы-Склад

Примечание. В целях упрощения задачи в дальнейшем объединим описание бизнес-процессов "Закупки" и "Планирование закупок" в один бизнес-процесс под названием "Планирование закупок и размещение заказов" и присвоим ему номер 1Пл_Зак.

Бизнес-процесс "Планирование закупок и размещение заказов поставщикам"

Общее описание бизнес-процесса

Предприятие планирует закупки медикаментов. Планирование закупок осуществляется в Департаменте маркетинга, в группе маркетинга и планирования. Планирование закупок осуществляется следующим образом:

1. Менеджер группы планирования и маркетинга ежедневно получает от контрагентов данные внешней и внутренней статистики продаж медикаментов в виде отчетов продаж.
2. Для планирования закупок медикаментов менеджер группы планирования и маркетинга еженедельно на основании статистики продаж производит расчет потребности в товаре. В результате расчета формируется Таблица потребностей в товаре.
3. Определив количество и номенклатуру заказываемых товаров, менеджер отдела закупок приступает к анализу предложений поставщиков. Данный процесс осуществляется ежемесячно или по мере необходимости. Выбираются наиболее выгодные условия поставки. Для этого сравниваются цены поставщиков. Данные сведения берутся из прайс-листа для закупок. При **выборе** поставщика важно учесть предоставляемую отсрочку платежа. Эта информация берется из контрактов, отмеченных как приоритетные (действующие). В результате формируется список

- поставщиков, каждой позиции присваивается признак основного и запасных поставщиков в порядке убывания приоритета.
4. Менеджер отдела закупок ежемесячно на основании Таблицы потребностей в товаре и списка выбранных поставщиков формирует графики поставок с указанием сроков и периодичности, но без количества поставки.
 5. Ежемесячно после определения потребности в товаре менеджер группы логистики рассчитывает необходимое количество закупок. Необходимое количество закупок рассчитывается на основании фактических запасов на складе, необходимого минимального и максимального уровня запасов. Нормы минимального и максимального количества запасов устанавливаются в днях. При расчете необходимого количества закупки учитывается также время товара в пути. Таким образом, данный расчет должен обеспечить возможность бесперебойного отпуска товара со склада. По результату расчетов формируется план заявок на месяц.
 6. Затем в группе логистики ежедневно по плану заявок, графику поставок, прайс-листам поставщиков формируются заказы поставщикам.
 7. Если предстоит сделать заказ импортному поставщику, то менеджер группы логистики рассчитывает затраты на сертификацию, создается отчет о затратах на сертификацию. Затраты на сертификацию проверяются на соответствие внутрифирменным нормам. Данная операция производится по мере необходимости.
 8. Если затраты на сертификацию превышают внутрифирменные нормы, то менеджер группы логистики повторяет процесс формирования заказов поставщикам. Формируются новые заказы.
 9. Ежедневно подготовленный заказ поставщику акцептуется, заказ должен подписать менеджер по логистике и директор Департамента маркетинга и **управления** товарными запасами.
 10. Ежедневно менеджер группы логистики направляет заказ в отдел закупок. Менеджер отдела закупок направляет заказ поставщику.

Задание 3. Построение диаграммы действий

На основании общего описания бизнес-процесса "Планирование закупок и размещение заказов поставщикам" составьте диаграмму действий, которая показывает участников процесса, выполняемые каждым участником операции и взаимосвязь между ними. Операции на диаграмме должны следовать в хронологическом порядке, который определен в приведенном описании бизнес-процесса.

Выполнение задания 3

1. Изучите общее описание бизнес-процесса, выделите его участников. В пунктах №1, 2 приведенного описания участник процесса - "Менеджер группы планирования и маркетинга", в пунктах № 3, 4 - "Менеджер отдела закупок", с 5 по 9 пункт участник бизнес-процесса - "Менеджер группы логистики". Таким образом, в бизнес-процессе "Закупки" три участника - менеджер группы планирования и маркетинга, менеджер отдела закупок, менеджер группы логистики.
2. Приступите к формированию диаграммы действий. Для этого необходимо разделить поле на 3 части, каждая часть поля отводится для отображения действий участника процесса.
3. Для формирования диаграммы средствами MS Visio необходимо открыть в папке **Software / UML Model Diagram** форму UML Activity.
4. Для **удобства** построения диаграммы на листе расположите его горизонтально (File / Page Setup / Landscape).

5. На панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением линии Line Tool. Удерживая левую клавишу мыши, разделите лист на три части.
6. На панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением буквы "А". Внесите в качестве заголовка полное наименование бизнес-процесса, сокращенное наименование (1Пл_Зак) и участников бизнес-процесса в соответствии с рисунком 3.

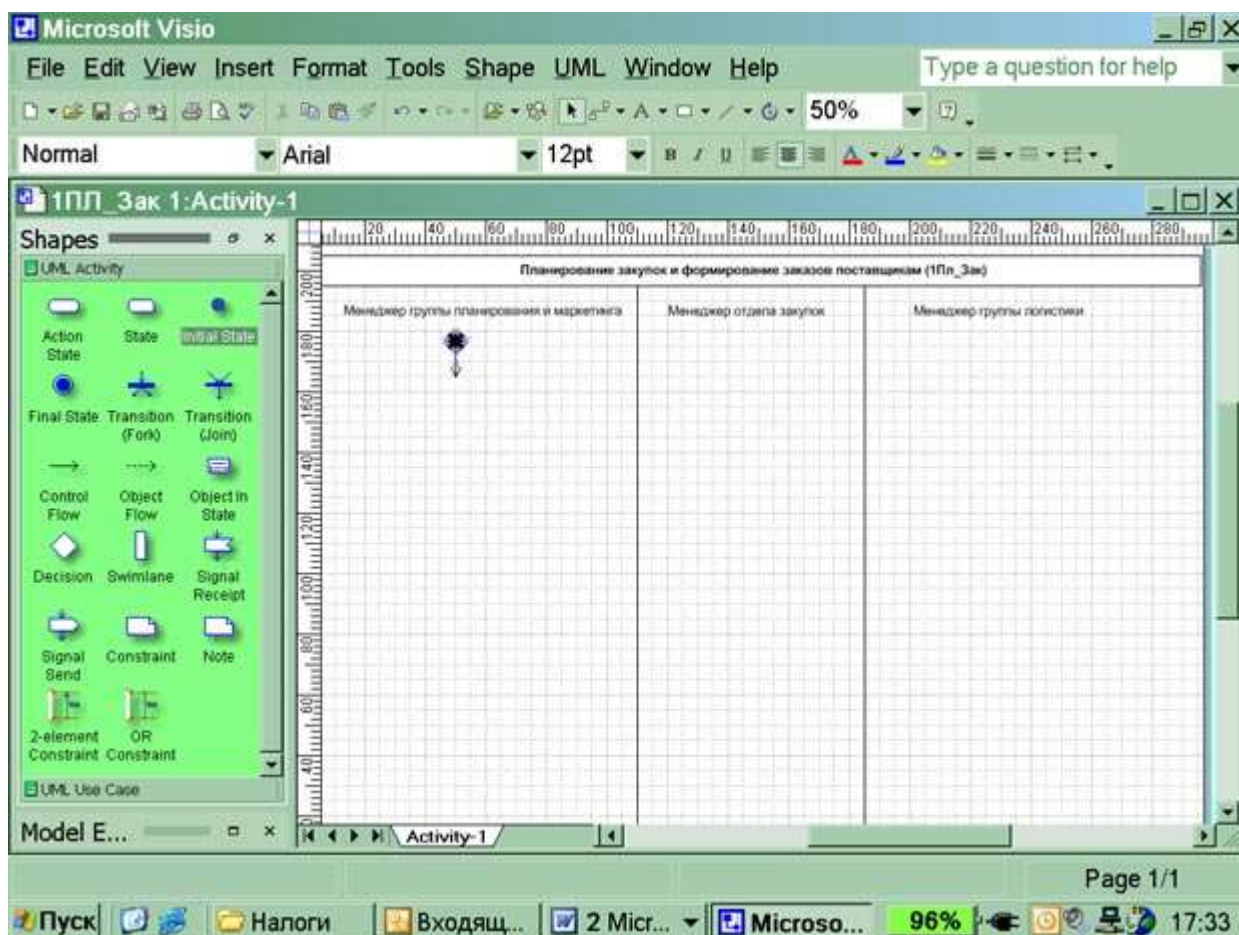






Рис. 3. Подготовительная стадия для изображения диаграммы действий

7. Проанализируйте общее описание бизнес-процесса и выделите участника процесса, с которого начинается процесс. Очевидно, что это менеджер группы планирования и маркетинга. Действительно, процесс закупок должен начинаться только после того, как определена потребность компании в товаре (медикаментах).
8. Обозначьте на диаграмме начало процесса символом  "Initial state" и опустите стрелку вниз (рис. 3). Работу с графическими формами можно осуществлять только при активированной пиктограмме с изображением стрелки на панели "Форматирование".
9. Пользуясь текстовым описанием, выделите действия, выполняемые менеджером группы планирования и маркетинга. Действия (операции), выполняемые менеджером группы планирования и маркетинга: "Получение внутренней статистики продаж", "Получение внешней статистики продаж", "Расчет потребности в товаре".

10. Отобразите на диаграмме действия, выполняемые менеджером группы планирования и маркетинга. Обратите внимание, что процессы получения внутренней и внешней статистики происходят независимо друг от друга. Неважно, в какой последовательности будут получены данные статистики, поэтому действия (операции) по получению внутренней и внешней статистики отобразите на схеме параллельно.
11. Для изображения действия на диаграмме используйте фигуру . Впишите внутри фигуры наименование и порядковый номер действия (операции). Пусть параллельные операции имеют номера 1а), 1б). Для ввода текста на панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением буквы "А".
12. Действия соедините на диаграмме стрелками, переноса их мышкой с формы. Стрелки присоединяйте к отмеченным крестиком местам на фигурах.
13. Для изображения параллельных процессов получения внутренней и внешней статистики примените  (Transition|Fork).
14. Расчет потребностей в товаре менеджер выполняет только после того, как получит и внутреннюю, и внешнюю статистику, следовательно, необходимо объединить параллельные процессы получения статистики в один. Для объединения независимых, параллельных процессов используйте  (Transition|Join).
15. В результате операции по расчету потребностей в товаре (операция № 2) (п. 2 общего описания) менеджер формирует документ - таблицу потребностей в товаре. Для отображения документа на диаграмме используйте изображение прямоугольника. Нарисуйте прямоугольник мышкой, зафиксировав на панели инструментов "Стандартная" соответствующую пиктограмму Rectangle Tool.
16. Операция и получаемый в результате ее выполнения документ на диаграмме соединяются пунктирной линией. Для изображения пунктирной линии зафиксируйте пиктограмму Line Tool на панели инструментов "Стандартная" и выберите пунктирную линию на панели инструментов "Форматирование", используя меню пиктограммы (Line Patter).
17. В результате на диаграмме ([рис. 4](#)) получите изображение действий (операций), осуществляемых менеджером группы планирования и маркетинга.

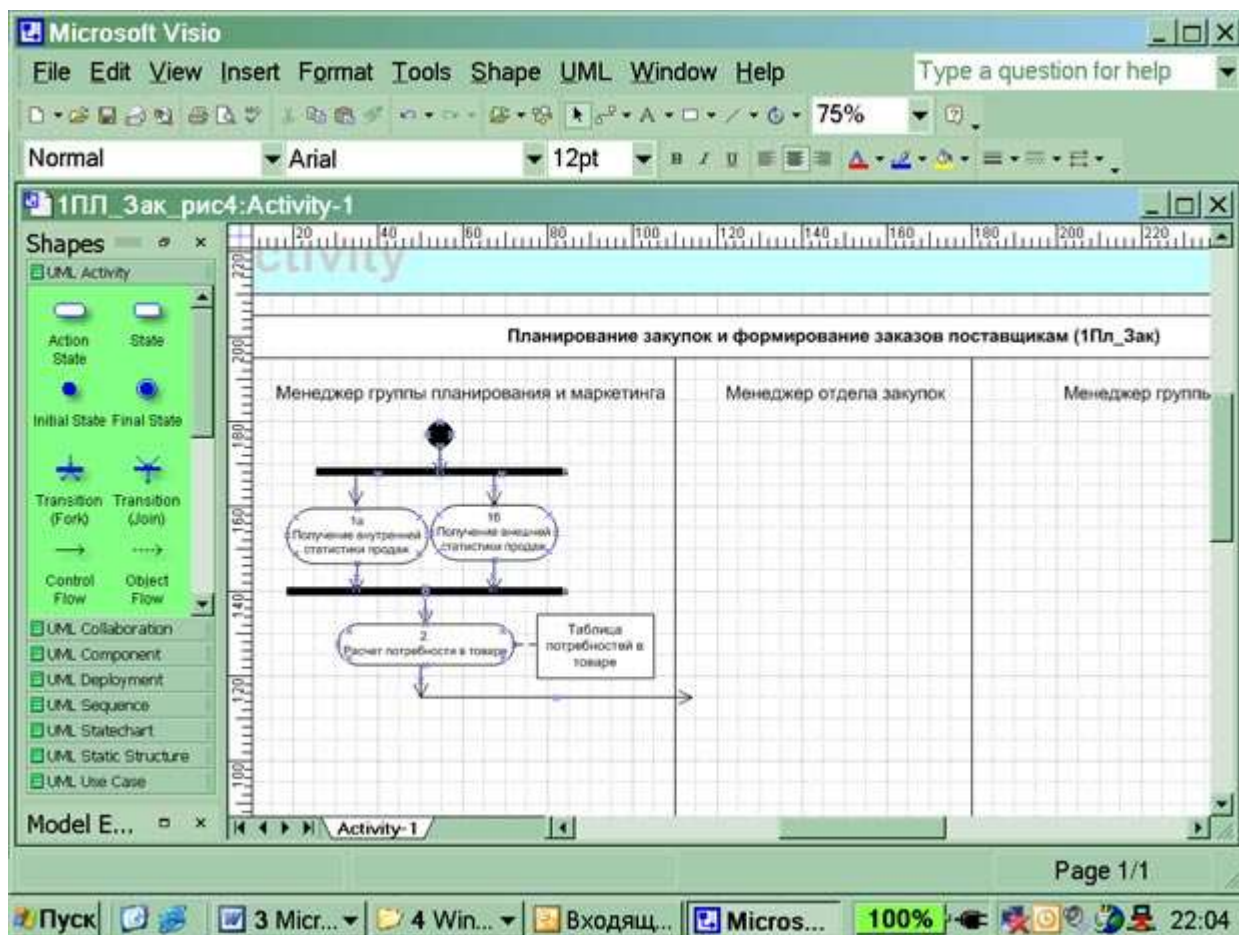


Рис. 4. Диаграмма действий менеджера группы планирования и маркетинга





18. После того как менеджер группы планирования и маркетинга сформировал таблицу потребностей в товаре, в работу включается менеджер отдела закупок, поэтому направьте стрелку от операции "Расчет потребности в товаре" в поле деятельности менеджера закупок, как показано на [рисунке 4](#).
19. Прочитайте общее описание бизнес-процесса и выделите действия (операции), выполняемые менеджером отдела закупок. Определите также действия, которые менеджер отдела закупок выполняет после действий менеджера группы логистики.
20. На диаграмме последовательно отобразите следующие действия менеджера отдела закупок:
 - Ввод в систему прайс-листов поставщиков (операция № 3)
 - Анализ предложений поставщиков (операция № 4)
 - Выбор поставщиков (операция № 5)
 - Формирование графика поставок без указания количества (операция № 6)


Осуществите графическое построение диаграммы аналогично описанному в п. 11.

21. Соедините действия менеджера отдела закупок стрелками аналогично описанию, приведенному в п. 12.
22. Поставьте в соответствие действиям менеджера отдела закупок документы, формируемые в системе. В данном случае это прайс-листы и контракты, список поставщиков с расстановкой приоритетов, график поставок. Выполните работу по рисованию диаграммы в соответствии с описанием в п. 15-16.

23. После формирования менеджером отдела закупок графика поставок в работу включается менеджер группы логистики.
24. На диаграмме предстоит отобразить следующие действия менеджера группы логистики:
 - Расчет необходимого количества закупок (операция № 7);
 - Формирование заказов поставщикам (операция № 8);
 - Расчет затрат на сертификацию импортных товаров, если медикаменты импортные.*) (операция № 9);
 - Проверка суммы затрат на сертификацию на непревышение внутрифирменной нормы*);
 - Формирование заказов поставщикам при превышении затрат на сертификацию (операция № 10);
 - Подпись заказа (операция № 11);
 - Направление заказа менеджеру отдела закупок (операция № 12).

Изучая общее описание бизнес-процесса, обратите внимание на то, что менеджер группы логистики дважды производит проверку условий и в зависимости от результата выполняет то или иное действие. В приведенном выше списке операций символом *) отмечены операции по проверке условий. В этом состоит особенность диаграммирования действий менеджера группы логистики.

25. Отобразите действие "Расчет необходимого количества закупок" и опустите стрелку вниз.
26. Ввиду того, что формирование заказов поставщикам может происходить неоднократно при превышении затрат на сертификацию, предусмотрите эту ситуацию и используйте графику для объединения параллельных потоков (Transition|Join). 
27. Отобразите действие "Формирование заказов поставщикам" после символа объединения потоков.
28. Отобразите ромб-символ проверки условия . Проведите из него две стрелки и надпишите их "Импорт", "Россия".
29. Стрелку "Россия" направьте к операции № 11 "Подпись заказа".
30. По направлению стрелки "Импорт" диаграммируйте последовательно два действия "Расчет затрат на сертификацию импортных товаров", "Проверка суммы затрат на сертификацию на непревышение внутрифирменной нормы".
31. За операцией "Проверка суммы затрат на сертификацию на непревышение внутрифирменной нормы" вновь отобразите ромб-символ проверки условия . Проведите из него две стрелки и надпишите их "больше x%", "меньше x%". Здесь x% - норма затрат на сертификацию.
32. Стрелку с надписью "больше x%" соедините с операцией № 8 "Формирование заказов поставщикам" через символ объединения потоков.
33. Стрелку с надписью "меньше x%" направьте к операции № 11 "Подпись заказа".
34. Поскольку к операции № 11 "Подпись заказа" направлено два потока действий (п. 29 и п. 33), необходимо воспользоваться обозначением объединения независимых (параллельных) потоков  (Transition|Join). В операцию №11 "Подпись заказа", как и в любую другую, должна входить только одна стрелка. Для выполнения этого правила и используют символ объединения потоков.
35. Поставьте в соответствие операции "Подпись заказа" документ - акцептованный заказ поставщику аналогично тому, как написано в п. 15-16.

36. В качестве следующей операции отобразите операцию № 12 "Направление заказа менеджеру отдела закупок". На этом действия, выполняемые менеджером группы логистики, завершаются. Вновь работа переключается на менеджера отдела закупок, поэтому направьте стрелку от 12 операции в поле действий менеджера закупок.
37. Отобразите на диаграмме переход документа "Заказ поставщику" от менеджера группы логистики к менеджеру отдела закупок. Для этого сначала поставьте в соответствие операции № 12 "Направление заказа менеджеру отдела закупок" документ "Заказ поставщику" так, как это описано в п. 15-16. После этого изображение документа с надписью "Заказ поставщику" путем копирования разместите в поле действий менеджера отдела закупок. Затем направьте пунктирную стрелку $--\Rightarrow$ (Object Flow) между двумя изображениями документа "Заказ поставщику" в направлении поля действий менеджера отдела закупок.
38. Соедините операцию № 12 "Направление заказа менеджеру отдела закупок" с операцией № 13 "Направление заказа поставщику", выполняемой менеджером отдела закупок. Это последняя операция в соответствии с заданием.
39. Укажите на диаграмме конец процесса. Для этого используйте символ  (Final State). Соедините стрелкой операцию № 13 "Направление заказа поставщику" с символом Final State.

Общий вид диаграммы действий бизнес-процесса "Планирование закупок, формирование заказов поставщикам" представлен на [рис. 5](#).

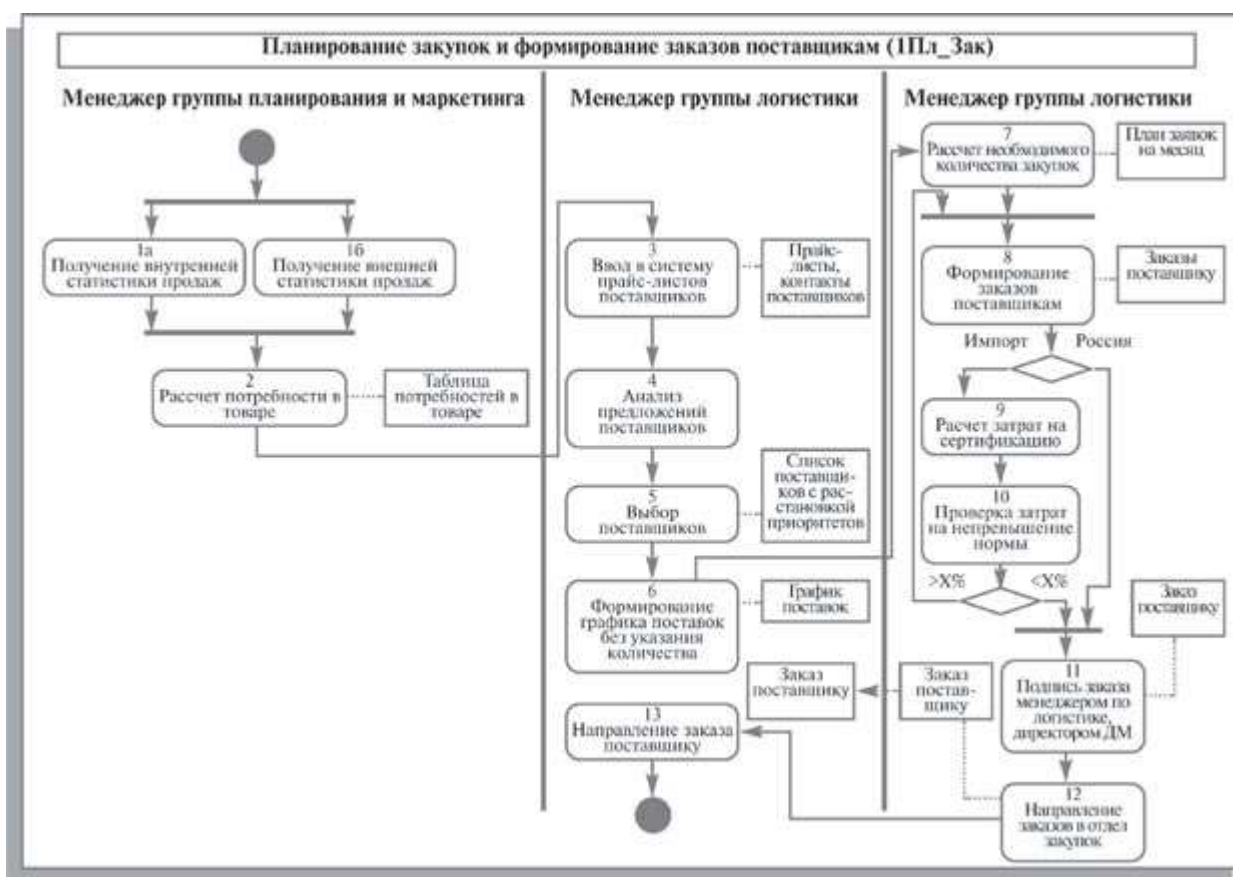


Рис. 5. Диаграмма действий бизнес-процесса "Планирование закупок, формирование заказов поставщикам"

Задание 4. Формирование таблицы операций

Все операции, участвующие в процессе "Планирование закупок, формирование заказов поставщикам", отразите в Таблице описания операций, имеющей следующий формат:

Диаграмма и номер на диаграмме	Операция	Исполнитель	Как часто	Входящие документы (документы-основания)	Исходящий документ (составляемый документ)	Проводка (дебет, кредит, сумма, аналитика)	Комментарий
1	2	3	4	5	6	7	8

Примечание. Далее заполненная форма таблицы описания операций будет использоваться для проектирования перечисленных в ней операций на информационную систему.

Выполнение задания 4

В таблицу последовательно внесите операции бизнес-процесса в соответствии с общим описанием и диаграммой действий.

1. В графе 1 проставьте краткое наименование диаграммы - 1Пл_Зак. Кроме того, в этой графе укажите номер операции, соответствующий изображению на диаграмме действий (рис. 5).
2. В графу 2 путем копирования перенесите из диаграммы действий наименование операции.
3. В графе 3 укажите исполнителя операции. В рассматриваемом бизнес-процессе исполнителями операций являются менеджер группы планирования и маркетинга, менеджер отдела закупок, менеджер группы логистики. Графа 3 заполняется на основании диаграммы действий.
4. В графе 4 укажите, с какой частотой выполняется каждая операция. Проставьте частоту выполнения операций в соответствии с общим описанием бизнес-процесса. Данная информация фиксируется в ходе обследования компании. Например, это может быть "еженедельно", "ежесуточно", 1 раз в месяц, 200 раз в день и т. п. Если операция выполняется с неопределенной периодичностью, то в графе указывают "по мере необходимости". При проектировании или выборе системы данные из графы "Как часто" определяют требования к быстродействию системы, к параметрам сетевого варианта системы.
5. В графу 5 занесите наименования документов, на основании которых осуществляется выполнение операции (входящие документы).
6. В графе 6 укажите наименования документов, которые создаются в результате выполнения операции (исходящие документы). В отдельных случаях входящие и исходящие документы могут совпадать. Например, для операции "Направление заказа поставщику" входящим и исходящим документом будет заказ поставщику.
7. Если на основании операции формируется бухгалтерская проводка, то она указывается в графе 7. В рассматриваемом примере нет операций, по которым бы формировались проводки.

8. Графа 8 предназначена для произвольной дополнительной информации.

На следующих страницах представлена таблица описания операций бизнес-процесса "Планирование закупок и размещение заказов поставщикам".

Операции бизнес-процесса "Планирование закупок и размещение заказов поставщикам"							
Диаграмма и номер операции и на диаграмме	Операция	Исполнитель	Как часто	Входящие документы (документы-основания)	Исходящий документ (составляемый документ)	Проводка (дебет, кредит, сумма, аналитика)	Комментарий
1	2	3	4	5	6	7	8
1Пл_Заказ 1а	1. Получение внутренней статистики продаж	Менеджер гр. планирования и маркетинга	Ежесуточно	Отчет-таблица собственных продаж	Нет	Нет	
1Пл_Заказ 1б	2. Получение внешней статистики продаж	Менеджер гр. планирования и маркетинга	Ежесуточно	Отчет-таблица продаж внешних источников	Нет	Нет	
1Пл_Заказ 2	3. Расчет потребности в товаре	Менеджер гр. планирования и маркетинга	Еженедельно	Отчет-таблица собственных продаж Отчет-таблица продаж внешних источников	Таблица потребностей в товаре	Нет	
1Пл_Заказ 3	4. Ввод в систему прайс-листов	Менеджер отдела закупок	Ежемесячно	Прайс-листы поставщиков	Прайс-листы поставщиков	Нет	

	поставщик ов			ов	в		
1Пл_Зак 4	5. Анализ предложен ий поставщик ов и действующ их контрактов	Менеджер отдела закупок	Ежемесячн о и по мере необходим ости	Прайс- листы поставщик ов Контракты действующ ие	Список поставщико в	Нет	
1Пл_Зак 5	6. Выбор поставщик ов	Менеджер отдела закупок	Ежемесячн о и по мере необходим ости	Список поставщик ов	Список поставщико в с расстановк ой приоритето в	Нет	
1Пл_Зак 6	7. Формиров ание графика поставок без указания количества	Менеджер отдела закупок	Ежемесячн о и по мере необходим ости	Список поставщик ов с расстановк ой приоритето в Таблица потребност ей в товаре	График поставок	Нет	
1Пл_Зак 7	8. Расчет необходим ого количества закупок с учетом остатка на складе и страхового запаса	Менеджер группы логистики	Ежемесячн о и по мере необходим ости	Таблица потребност ей в товаре, график поставок	План заявок на месяц	Нет	
1Пл_Зак 8	9. Формиров ание заказов	Менеджер группы логистики	Ежедневно по плану заявок	План заявок на месяц, график	Заказы поставщику	Нет	

	поставщик ам с учетом складских остатков, товара в пути и резервного запаса			поставок, прайс- листы поставщик ов			
1Пл_Зак 9	10. Расчет затрат на сертифика цию	Менеджер группы логистики	По мере необходим ости	Заказы поставщик у	Отчет о затратах на сертификац ию	Нет	
1Пл_Зак 10	11. Проверка затрат на непревыше ние нормы	Менеджер группы логистики	По мере необходим ости	Отчет о затратах на сертифика цию	Отчет о затратах на сертификац ию	Нет	
1Пл_Зак 11	12. Подпись заказа менеджеро м по логистике, директоро м ДМ	Менеджер группы логистики	Ежедневно	Заказы поставщик у	Заказы поставщику акцептован ные	Нет	
1Пл_Зак 12	13. Направлен ие заказа в отдел закупок	Менеджер группы логистики	Ежедневно	Заказы поставщик у акцептован ные	Заказы поставщику акцептован ные	Нет	
1Пл_Зак 13	14. Направлен ие заказа поставщик у	Менеджер отдела закупок	Ежедневно	Заказы поставщик у акцептован ные	Заказы поставщику акцептован ные	Нет	

Задание 5. Формирование таблицы описания документов

Все документы, участвующие в бизнес-процессе, отразите в Таблице описания документов, имеющей следующий формат:

Диаграмма и номер на диаграмме	Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8

Примечание. После того, как документы будут описаны, приступают к их разработке в ИС. Формы документов в учебном пособии не представлены, в практической же деятельности создается альбом форм, который является приложением к таблице описания документов.

Выполнение задания 5

Таблица описания документов получается путем переформирования (перестановки столбцов и объединении строк) таблицы описания операций. Особенности таблицы описания документов заключаются в следующем. В Графе 2 не должно быть повторяющихся наименований документов. Если один и тот же документ является исходящим на различных операциях, то он один раз указывается в графе 2 "Составляемый документ", а в графе 3 ему в соответствие ставятся несколько операций. Также по наименованию документа следует объединить записи и в других графах.

В графе 7 указывается наименование реестра, в котором регистрируется создаваемый документ. Наименование реестру присваивается, как правило, по наименованию документа. Например, если документ "Заказ", то "Реестр заказов"; документ "прайс-лист", тогда "реестр прайс-листов" и т.д.

На следующих страницах приведена таблица описания документов бизнес-процесса "Планирование закупок и размещение заказов поставщикам".

Бизнес-процесс "Запасы-склад (приходование товара)"

Общее описание бизнес-процесса

ЗАО "МЕД" располагает 10 складами, из которых один, Центральный, расположен в Москве, а другие в филиалах. Количество хранимой номенклатуры медикаментов - от 1000 до 2000.

Документы бизнес-процесса "Планирование закупок и размещение заказов поставщикам"

Диаграмма и номер операции на диаграмме	Составляемый документ (Исходящий документ)	Операция	Исполнитель	Как часто	Документы-основания (Входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8
1Пл_Заказ 2	1. Таблица потребностей в товаре	Расчет потребностей в товаре	Менеджер гр. планирования и маркетинга	Еженедельно	Отчет-таблица собственных продаж	Реестр статистических отчетов	
1Пл_Заказ 3	2. Список поставщиков	Анализ предложений поставщиков и действующих контрактов	Менеджер отдела закупок	Ежемесячно и по мере необходимости	Прайс-листы поставщиков Контракты действующие	Реестр прайс-листов	
1Пл_Заказ 4	3. Список поставщиков с расстановкой приоритетов	Выбор поставщиков	Менеджер отдела закупок	Ежемесячно и по мере необходимости	Список поставщиков	Нет	
1Пл_Заказ 5	4. График поставок	Формирование графика поставок без указания количества	Менеджер отдела закупок	Ежемесячно и по мере необходимости	Список поставщиков с расстановкой приоритетов Таблица потребностей в товаре	Нет	

1Пл_Зак 6	5. План заявок на месяц	Расчет необходимого количества закупок с учетом остатка на складе и страхового запаса	Менеджер группы логистики	Ежемесячно и по мере необходимости	Таблица потребностей в товаре, прайс-листы поставщиков, график поставок	Нет	
1Пл_Зак 7	6. Заказы поставщику	Формирование заказов поставщикам с учетом складских остатков, товара в пути и резервного запаса	Менеджер группы логистики	Ежедневно по плану заявок	План заявок на месяц, график поставок, прайс-листы поставщиков	Реестр заказов	
1Пл_Зак 9, 10	7. Отчет о затратах на сертификацию	Расчет затрат на сертификацию Проверка затрат на превышение нормы	Менеджер группы логистики	По мере необходимости	Заказы поставщику	Нет	
1Пл_Зак 11, 12, 13	8. Заказы поставщику у акцептованные	Подпись заказа менеджером по логистике, директором ДМ Направление заказа в отдел закупок	Менеджер группы логистики	Ежедневно	Заказы поставщику Заказы поставщику у акцептованные	Реестр заказов	

		Направлен ие заказа поставщик у					
--	--	--	--	--	--	--	--

Склад фактически работает не с номенклатурой, а с сериями. Одной позиции номенклатуры может соответствовать несколько серий медикаментов.

Склад разбит на несколько зон хранения. Зоны хранения соответствуют правилам хранения тех или иных медикаментов.

Используются вложенные единицы измерения - упаковка (минимальная единица), блок (несколько упаковок), заводская коробка (несколько блоков).

На складе хранится товар зарезервированный (недоступный для продажи).

Учет ТМЦ ведется в двух валютах - в рублях, валюте прихода.

Процесс приходования медикаментов на склад выглядит следующим образом:

1. Менеджер приемного отдела принимает товар по товарной накладной поставщика, проверяя номенклатуру, количество, посерийное соответствие, срок годности.
2. При полном соответствии фактически поступившего товара товару, указанному в товарно-транспортной накладной и заказе поставщику, менеджер приемного отдела передает документы менеджеру отдела закупок. В противном случае осуществляется процесс выявления виновных лиц и предъявление претензий.
3. Менеджер отдела закупок проверяет соответствие поставки заказу по номенклатуре, количествам и ценам и на основании товарной накладной поставщика формирует приходную накладную, отражая в базе данных количество и учетную цену поступившего товара. При формировании приходной накладной создается проводка Д41-К60. Далее в работу включаются менеджеры отделов сертификации и маркетинга.
4. Менеджер отдела сертификации по товарно-транспортной накладной проверяет наличие серий в справочнике. При необходимости справочник серий пополняется.
5. Менеджером отдела сертификации осуществляется процесс сертификации. Процесс сертификации в данном случае рассматривается и как процесс приходования сертификатов-документов на медикаменты, и как процедура сертификации с целью получения документов-сертификатов.
6. Менеджер учетного отдела при приходовании ТМЦ по товарно-транспортной накладной разбивает каждую номенклатурную позицию по сериям с указанием срока годности.
7. Параллельно с работой менеджера по сертификации, после отражения в базе данных количества товара менеджером отдела закупок, менеджер отдела маркетинга, используя товарно-транспортную накладную, определяет базовую цену продажи и указывает ее в карточке товара.
8. Размещение товара по местам хранения осуществляется менеджером склада в соответствии с Планом расстановки продукции по местам хранения. Место хранения заносится в карточку товара.

Задание 6. Построение диаграммы действий

На основании общего описания бизнес-процесса "Запасы-склад (приходование)" составьте диаграмму действий, которая показывает участников процесса, выполняемые каждым участником операции и взаимосвязь между ними. Операции на диаграмме должны следовать в хронологическом порядке, который определен в приведенном описании бизнес-процесса.

Выполнение задания 6

1. Изучите общее описание бизнес-процесса, выделите его участников. В пунктах № 1, 2 приведенного описания участник процесса - "Менеджер отдела приемки", в пункте № 3 - участник "Менеджер отдела закупок", в пункте № 4, 7 - участник "Менеджер отдела сертификации", в пункте № 5 - участник бизнес-процесса "Менеджер учетного отдела", в пункте № 6 - "Менеджер отдела маркетинга", в пункте № 8 - "Менеджер склада".

Таким образом, в бизнес-процессе "Запасы-Склад" шесть участников процесса - менеджеры отделов приемки, закупок, сертификации, учетного отдела, отдела маркетинга и склада.

2. Приступите к формированию диаграммы действий. Для этого необходимо разделить поле на 6 частей, каждая из которых отводится для отображения действий участника процесса.
3. Для формирования диаграммы средствами MS Visio необходимо открыть в папке **Software / UML Model Diagram** форму UML Activity.
4. Для удобства размещения диаграммы на листе расположите его горизонтально (File / Page Setup / Landscape).
5. На панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением линии Line Tool. Удерживая левую клавишу мышки, разделите лист на шесть частей.
6. На панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением буквы "А". Внесите в качестве заголовка полное наименование бизнес-процесса "Запасы-склад (приходование)", сокращенное наименование (4Склад) и участников бизнес-процесса в соответствии с [рисунком 6](#).

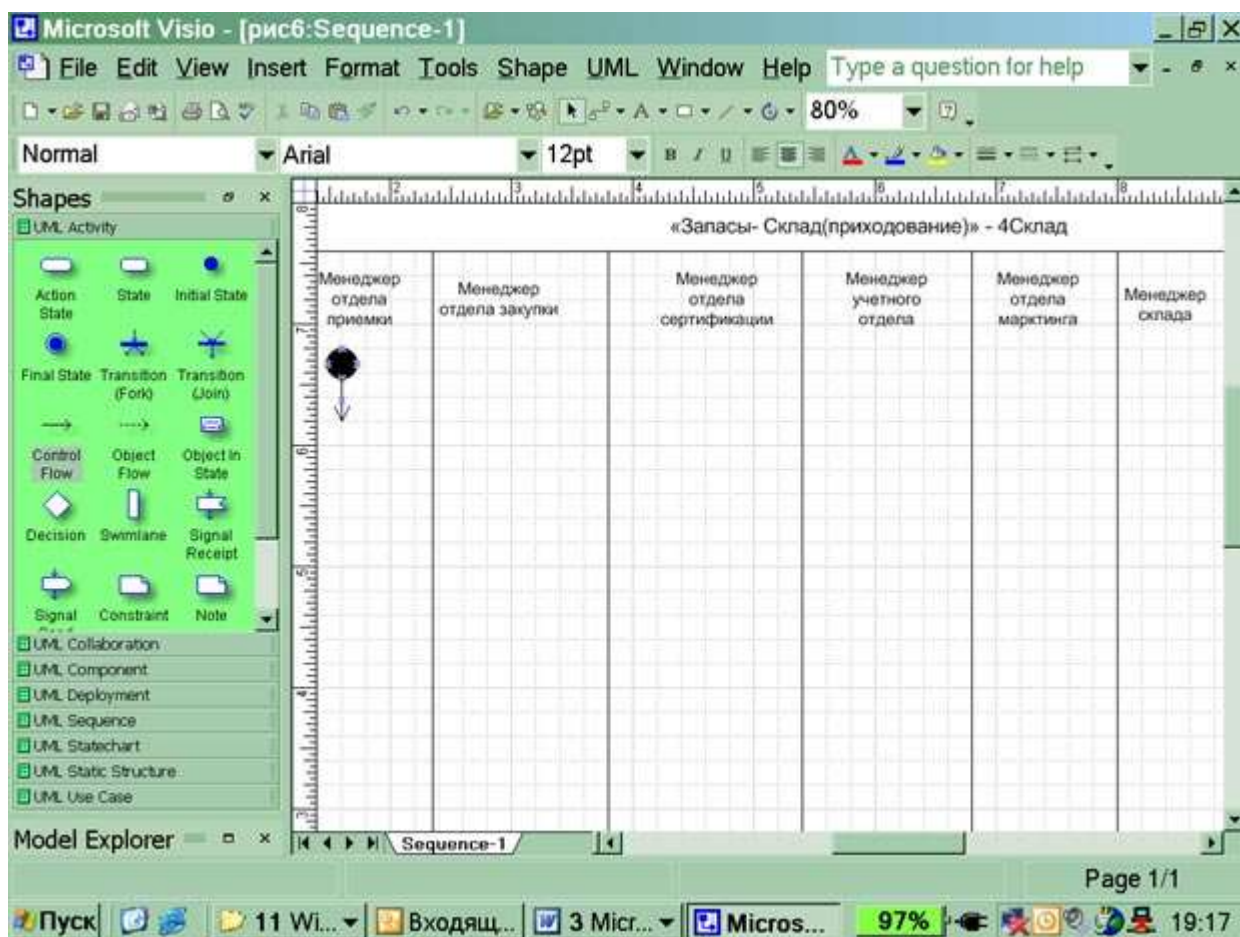

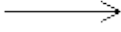


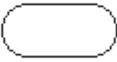




Рис. 6. Подготовительная стадия для изображения диаграммы действий

7. Проанализируйте общее описание бизнес-процесса и выделите участника процесса, с которого начинается процесс. Очевидно, что это менеджер отдела приемки.
8. Обозначьте на диаграмме начало процесса символом  "Initial state" в графе, отведенной для изображения действий менеджера отдела приемки (рис. 6). Не забывайте, что работу с графическими формами можно осуществлять только при активированной пиктограмме с изображением стрелки на панели "Форматирование".
9. Направьте стрелку вниз от изображения начала процесса. Для этого перенесите с формы UML Activity изображение стрелки  (Control Flow) (рис. 6).
10. Пользуясь текстовым описанием, выделите действия, выполняемые менеджером отдела приемки. Действия (операции), выполняемые менеджером отдела приемки - "проверка товара по количеству, серийному соответствию, сроку годности".
11. Отобразите на диаграмме действие, выполняемое менеджером отдела приемки. Для изображения действия на диаграмме используйте фигуру  . Впишите внутрь фигуры наименование и порядковый номер (№ 1) операции. Для ввода текста на панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением буквы "А".
12. Отобразите ромб-символ проверки условия  . Проведите из него две стрелки и надпишите их "Несоответствие документам", "Полное соответствие документов".

13. Стрелку "Несоответствие документам" соедините с подпроцессом выявления виновных лиц и предъявления претензий. Для отображения подпроцесса используйте тот же символ, что и для отображения действия  .
14. При несоответствии документов заданным требованиям прихода товара на склад не происходит, процесс заканчивается. Проведите стрелку из подпроцесса выявления виновных лиц и предъявления претензий и соедините ее с символом завершения процесса  (Final State)
15. Стрелку с надписью "Полное соответствие документов" соедините с операцией № 2 "Отражение в базе данных количества товара", выполняемой менеджером отдела закупки.
16. Отражение в базе данных количества товара осуществляется путем создания в базе приходной накладной. Для отображения документа на диаграмме используйте изображение прямоугольника. Нарисуйте прямоугольник мышкой, зафиксировав на панели инструментов "Стандартная" соответствующую пиктограмму Rectangle Tool.
17. Операция № 2 "Отражение в базе данных количества товара" и получаемый в результате ее выполнения документ "Приходная накладная" на диаграмме соединяются пунктирной линией. Для изображения пунктирной линии зафиксируйте пиктограмму Line Tool на панели инструментов "Стандартная" и выберите пунктирную линию на панели инструментов "Форматирование", используя меню пиктограммы (Line Patter).
18. В соответствии с общим описанием бизнес-процесса (пункты № 3, 7), после выполнения операции № 2 "Отражение в базе данных количества товара", выполняемой менеджером отдела закупки, происходит параллельная работа менеджера отдела сертификации и менеджера отдела маркетинга.
19. Для изображения параллельных процессов примените  (Transition|Fork).
20. Сравните полученное изображение с [рис. 7](#).

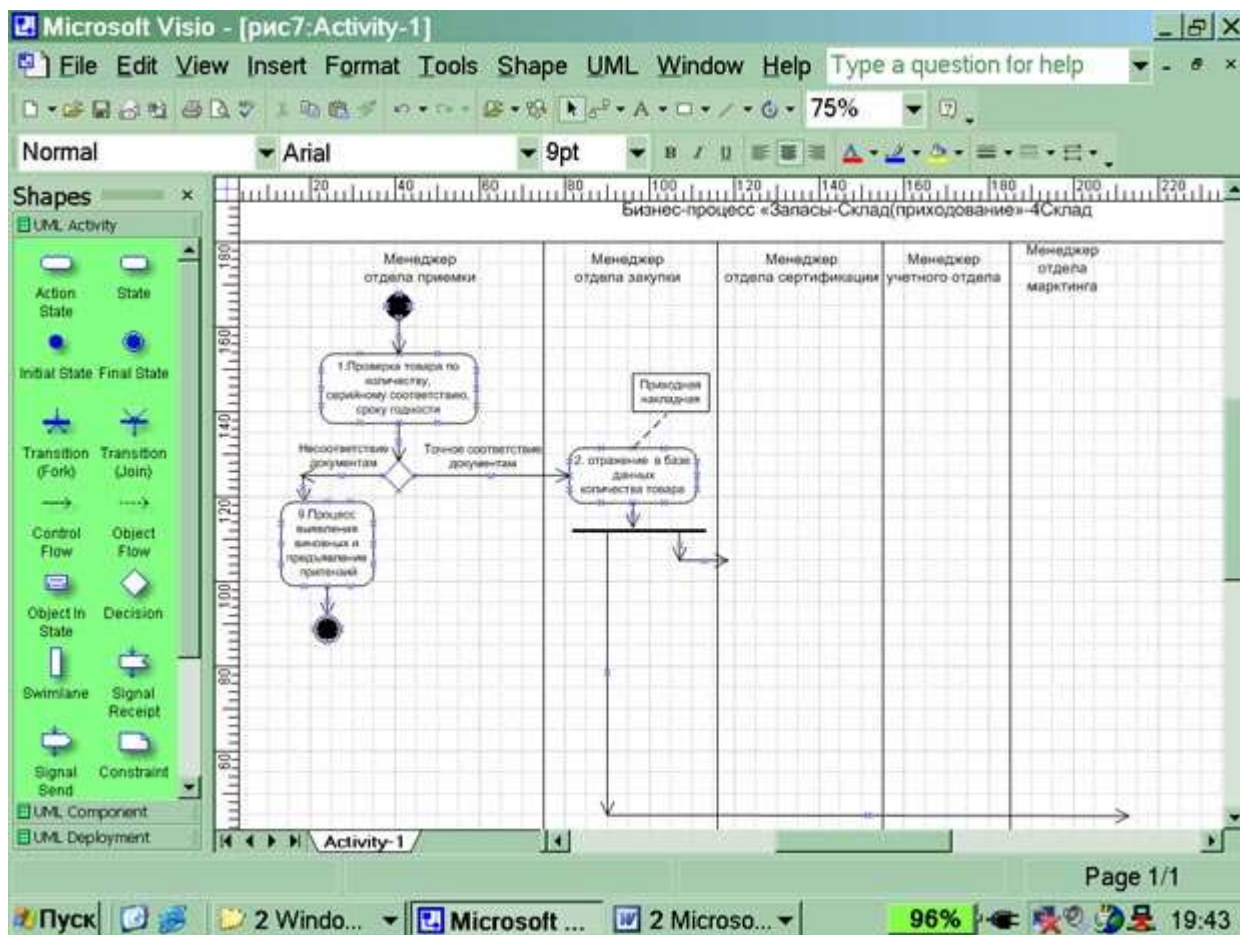
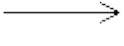






Рис. 7. Диаграмма действий менеджера отдела приемки и менеджера отдела закупок

21. Отобразите последовательно действия (операции) менеджера отдела сертификации и соедините их стрелками  (Control Flow). Менеджер выполняет операции № 3 "Поиск серии в справочнике", № 4 "Добавление серии в справочник", а также № 5 "Процесс сертификации".
22. После выполнения процесса сертификации к работе подключается менеджер учетного отдела. Он выполняет операцию № 6 "Разбиение каждой позиции номенклатуры по сериям". Отобразите эту операцию в зоне действий менеджера  учетного отдела, используя, как всегда, символ .
23. Изобразите на диаграмме операцию № 7 "Процесс размещения серии товаров", выполняемую менеджером склада.
24. Соедините стрелкой операцию № 6, выполняемую менеджером учетного отдела, и № 7, выполняемую менеджером склада.
25. Разместите на диаграмме операцию № 8 "Определение и ввод базовой цены", выполняемую менеджером отдела маркетинга.
26. Соедините операцию № 2 с операцией № 8, используя ранее подготовленную стрелку (рис. 7).
27. Поставьте в соответствие операции №8 документ "Карточка товара" аналогично описанию, приведенному в пунктах 16, 17.

28. Операции, выполняемые участниками рассматриваемого процесса, завершены. Отобразите завершение процесса в поле действия менеджера склада. Для этого прежде всего необходимо объединить параллельные операции. Для объединения независимых параллельных процессов используйте  (Transition/Join). После объединения процессов укажите на диаграмме конец процесса. Для этого используйте символ  (Final State).

Общий вид диаграммы действий бизнес-процесса "Запасы - Склад (приходование)" представлен на [рис. 8](#).

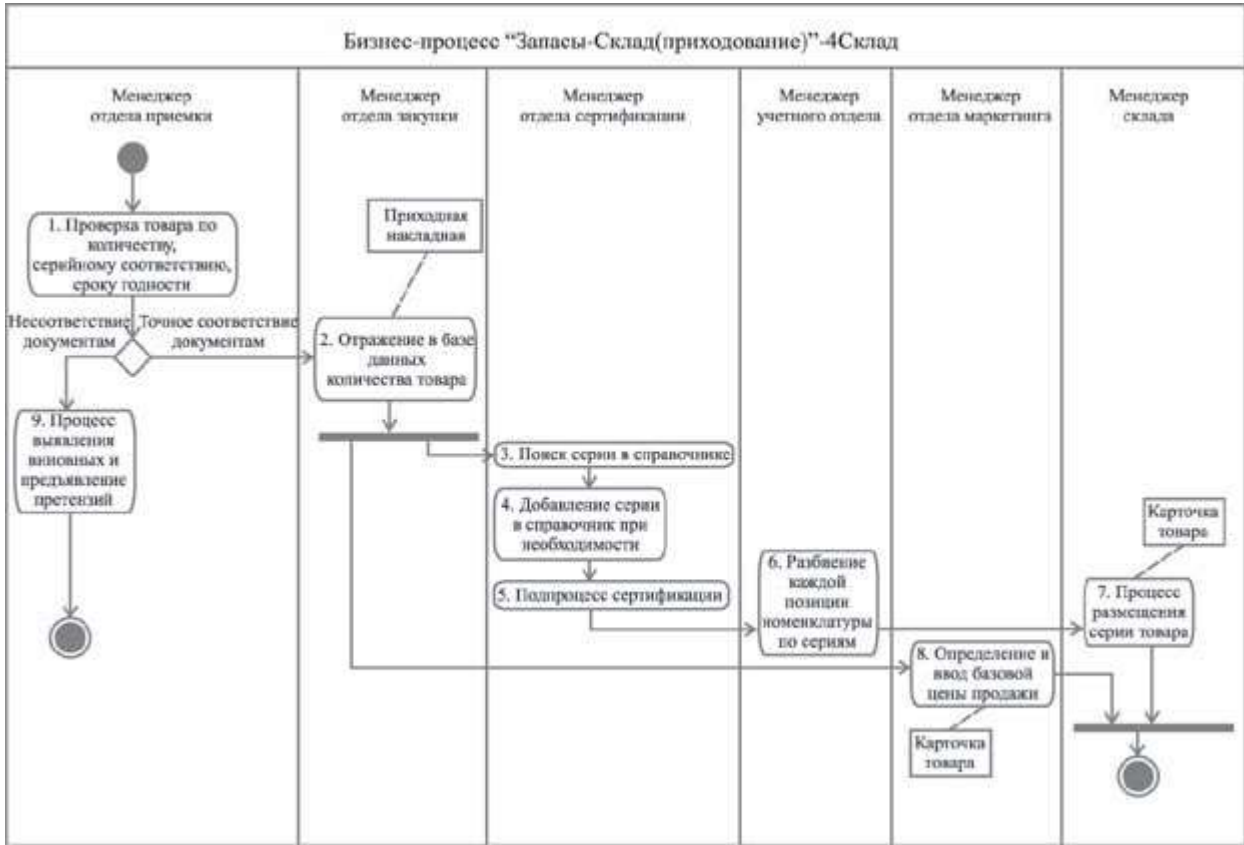


Рис. 8. Диаграмма действий бизнес-процесса "Запасы - Склад (приходование)"

Задание 7. Формирование таблицы операций

Все операции, участвующие в процессе "Запасы - Склад (приходование)", отразите в Таблице описания операций, имеющей следующий формат:

Диаграм ма и номер на диаграмм е	Операци я	Исполните ль	Как част о	Входящие документ ы (документ ы- основания	Исходящий документ (составляем ый документ)	Проводка (дебет, кредит, сумма, аналитик а)	Комментар ий
--	--------------	-----------------	------------------	---	---	--	-----------------

)			
1	2	3	4	5	6	7	8

Примечание. Далее заполненная форма таблицы описания операций будет использоваться для проектирования перечисленных в ней операций в информационной системе.

Выполнение задания 7

В таблицу последовательно внесите операции бизнес-процесса в соответствии с общим описанием и диаграммой действий.

1. В графе 1 проставьте краткое наименование диаграммы - 4Склад. Кроме того, в этой графе укажите номер операции, соответствующий изображению на диаграмме действий ([рис. 8](#)).
2. В графу 2 путем копирования перенесите из диаграммы действий наименование операции.
3. В графе 3 укажите исполнителя операции. В рассматриваемом бизнес-процессе исполнителями операций являются менеджер отдела приемки, отдела закупки, сертификации, учетного отдела, отдела маркетинга, склада. Графа 3 заполняется на основании диаграммы действий.
4. В графе 4 укажите, с какой частотой выполняется каждая операция. Проставьте частоту выполнения операций в соответствии с общим описанием бизнес-процесса. В данном примере все операции за исключением одной выполняются ежедневно. Только операция № 4 выполняется нерегулярно - по мере необходимости. Данная информация фиксируется в ходе обследования компании. При проектировании или выборе системы данные из графы "Как часто" определяют требования к быстродействию системы, к параметрам сетевого варианта системы.
5. В графу 5 занесите наименования документов, на основании которых осуществляется выполнение операции (входящие документы). В данном примере основанием выполнения почти всех операций является товарно-транспортная накладная. Кроме того, основанием операции № 2 является также Заявка поставщику, при выполнении операции № 3 используется справочник серий, а основанием процесса сертификации (операция № 5) служит Сертификат.
6. В графе 6 укажите наименования документов, которые создаются в результате выполнения операции (исходящие документы). В данном бизнес-процессе только два исходящих (формируемых) документа. Это приходная накладная и карточка товара.
7. Если на основании операции формируется бухгалтерская проводка, то она указывается в графе 7. В рассматриваемом примере проводка формируется по операции с номером 2.
8. Графа 8 предназначена для произвольной дополнительной информации.

Ниже представлена таблица описания операций бизнес-процесса.

Операции бизнес-процесса "Запасы - склад (приходование)"							
Диаграмма и	Операция	Исполнитель	Как часто	Входящие	Исходящий документ	Проводка (дебет,	Комментарий

номер на диаграм ме				документ ы (докумен ты- основани я)	(составляе мый документ)	кредит, сумма, аналити ка)	
1	2	3	4	5	6	7	8
4Склад 1	1. Проверка товара по количеств у, серийному соответств ию, сроку годности	Менеджер приемного отдела	Ежедневно	Товарная накладная поставщи ка			
4Склад 2	2. Отражение в базе данных количества и цены приходова ния товара	Менеджер отдела закупок	Ежедневно	Товарная накладная поставщи ка Заказ поставщи ку	Приходная накладная	Д41-К60	
4Склад 3	3. Поиск серии в справочни ке	Менеджер отдела сертифика ции	Ежедневно	Справочн ик серий Товарная накладная поставщи ка			
4Склад 4	4. Добавлени е серии в справочни к	Менеджер отдела сертифика ции	По мере необходим ости	Товарная накладная поставщи ка	Запись в справочник е серий	Нет	
4Склад 5	5. Подпроцес с сертифика ции	Менеджер отдела сертифика ции	Ежедневно	Сертифик ат			

	ции						
4Склад 6	6. Разбиение каждой позиции номенклатуры по сериям	Менеджер учетного отдела	Ежедневно	Товарная накладная поставщика			
4Склад 8	7. Определение и ввод базовой цены продажи	Менеджер отдела маркетинга	Ежедневно	Товарная накладная поставщика	Карточка товара		
4Склад 7	8. Процесс размещения серии товара	Менеджер склада	Ежедневно		Карточка товара		

Задание 8. Формирование таблицы описания документов

Все документы, участвующие в бизнес-процессе, отразите в Таблице описания документов, имеющей следующий формат:

Диаграмма и номер на диаграмме	Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8

Примечание. После того как будут описаны документы, приступают к их разработке в ИС. Формы документов в учебном пособии не представлены, в практической же деятельности создается альбом форм, который является приложением к таблице описания документов.

Выполнение задания 8

Таблица описания документов создается путем переформирования (перестановки столбцов и объединении строк) таблицы описания операций. Особенности таблицы описания документов заключаются в следующем. В Графе 2 не должно быть повторяющихся наименований документов. Если один и тот же документ является исходящим на различных операциях, то он один раз указывается в графе 2 "Составляемый документ", а в графе 3 ему в соответствие ставятся несколько операций. Также по наименованию документа следует объединить записи и в других графах.

В графе 7 указывается наименование реестра, в котором регистрируется создаваемый документ. Наименование реестру присваивается, как правило, по наименованию документа. В данном бизнес-процессе приходная накладная регистрируется в реестре приходных накладных, карточка товара регистрируется в реестре товаров.

Ниже приведена таблица описания документов бизнес-процесса "Запасы - Склад (приходование)".

Документы бизнес-процесса "Запасы - Склад (приходование)"							
Диаграмма и номер на диаграмме	Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8
4Склад 2	1. Приходная накладная	1. Отражение в базе данных количества товара	Менеджер отдела закупок	Ежедневно	Товарная накладная поставщика Заказ поставщику	Реестр приходных накладных	
4Склад 4	2. Запись в справочнике серий	2. Добавление серии в справочник	Менеджер отдела сертификации	По мере необходимости	Товарная накладная поставщика	Нет	

4Склад 7, 8	3. Карточка товара	3. Определе ние и ввод базовой цены продажи 4. Процесс размещен ия серии товара	Менеджер отдела маркетинга Менеджер склада	Ежедневно	Товарная накладна я поставщи ка	Реестр товара	
----------------	-----------------------	--	--	-----------	---	------------------	--

Бизнес-процесс "Продажи"

Общее описание бизнес-процесса

Бизнес-процесс выглядит следующим образом:

1. Менеджер отдела продаж ежедневно получает от клиента Заказ на конкретную номенклатурную единицу медикаментов. В Заказе номенклатурных единиц клиент указывает желаемую отсрочку платежа.
2. При получении Заказа менеджер отдела продаж по справочнику лицензий проверяет наличие у клиента действующей лицензии на право реализации медикаментов. При отсутствии лицензии продажа медикаментов клиенту не производится. Наличие лицензии проверяется по мере необходимости.
3. Менеджер отдела продаж ежедневно проверяет наличие необходимого количества заказанных медикаментов на складе.
4. Если медикаментов недостаточно для выполнения заказа, то менеджер отдела продаж размещает Заказ в реестре "неудовлетворенный спрос". Затем менеджер ежедневно проверяет возможность выполнения Заказа, размещенного в реестре "неудовлетворенный спрос".
5. При наличии у клиента необходимой лицензии и достаточном количестве товара на складе в отделе продаж на основании Заказа и договора формируется Заявка на номенклатурные единицы. Заявки формируются ежедневно.
6. Ежедневно на основании Заявки менеджер отдела продаж осуществляет резервирование товара.
7. Менеджер отдела продаж ежедневно контролирует кредитный лимит и дебиторскую задолженность потенциальных покупателей.
8. Если кредитный лимит и дебиторская задолженность не превышают допустимых значений, то Заявка передается на склад в Учетно-операционный отдел.
9. При превышении кредитного лимита или наличии просроченной дебиторской задолженности свыше допустимого количества дней менеджер отдела продаж заявку в Учетно-операционный отдел не передает, процесс продаж приостанавливается, осуществляются переговоры с клиентом.
10. Менеджер учетно-операционного отдела, получив Заявку, ежедневно производит подборку номенклатурных единиц.
11. Менеджер учетно-операционного отдела ежедневно формирует упаковочные листы для вложения их в каждый ящик.
12. Менеджером учетно-операционного отдела ежедневно формируются для клиента следующие документы: счет, расходная накладная, счет-фактура.

13. При фактической отгрузке товара со склада осуществляется его списание. Списание медикаментов осуществляется по расходной накладной и сопровождается формированием проводки Д62-К41.

Задание 9. Построение диаграммы действий

На основании общего описания бизнес-процесса "Продажи" составьте диаграмму действий, которая показывает участников процесса, выполняемые каждым участником операции и взаимосвязь между ними. Операции на диаграмме должны следовать в хронологическом порядке, который определен в приведенном описании бизнес-процесса.

Выполнение задания 9

1. Изучите общее описание бизнес-процесса, выделите его участников. Участниками бизнес-процесса "Продажи" являются менеджер отдела продаж и менеджер учетно-операционного отдела.
2. Приступите к формированию диаграммы действий. Для этого необходимо разделить поле на 2 части, каждая из которых отводится для отображения действий участника процесса.
3. Для формирования диаграммы средствами MS Visio необходимо открыть в папке **Software / UML Model Diagram** форму UML Activity.
4. Для удобства построения диаграммы на листе расположите его вертикально (File/Page Setup/Portrait).
5. На панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением линии Line Tool. Удерживая левую клавишу мыши, разделите лист на две части.
6. На панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением буквы "А". Внесите в качестве заголовка полное наименование бизнес-процесса "Продажи клиентам", сокращенное наименование (5ПродКл) и участников бизнес-процесса в соответствии с [рисунком 9](#).

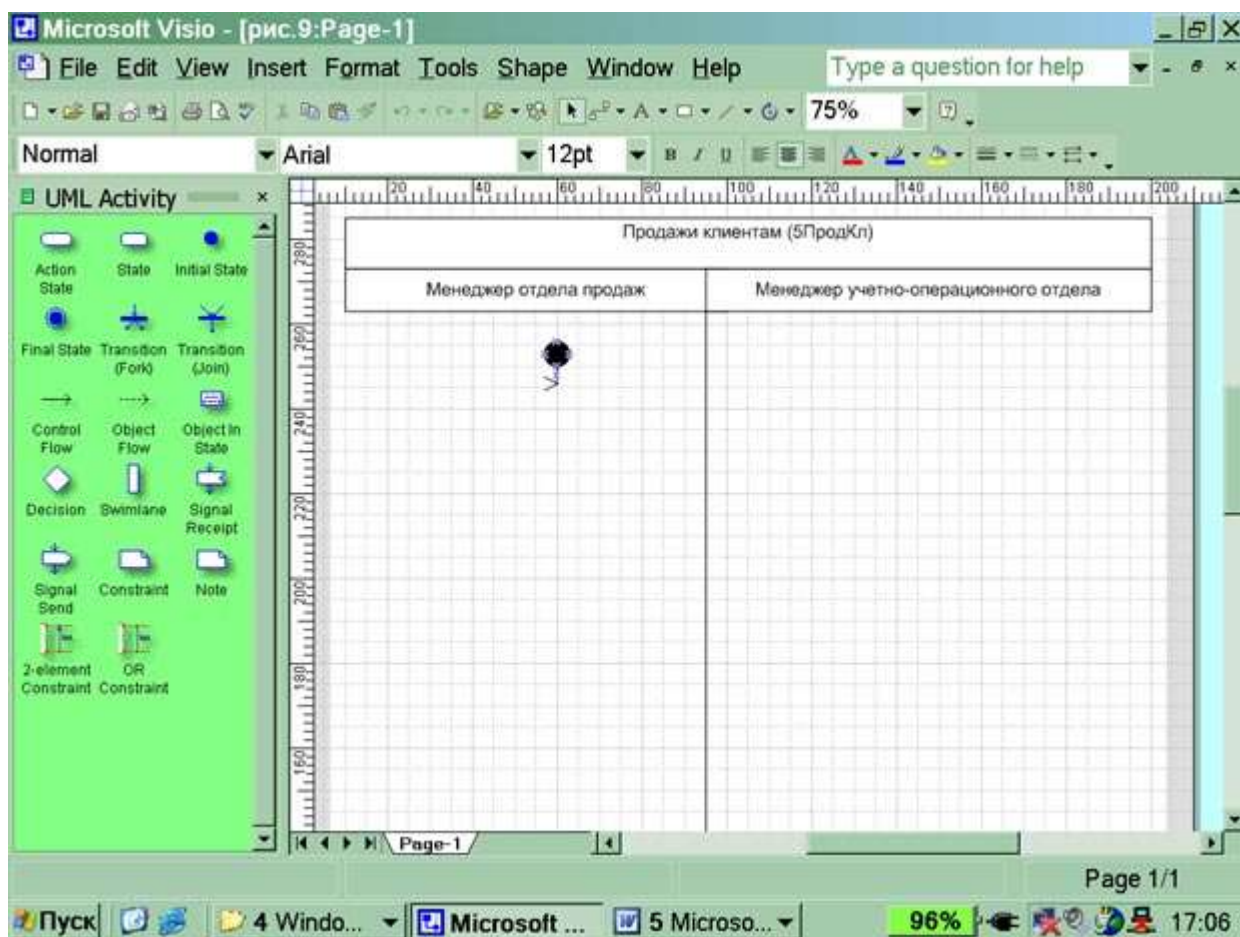


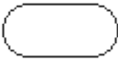
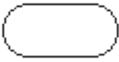







Рис. 9. Подготовительная стадия для изображения диаграммы действий

7. Проанализируйте общее описание бизнес-процесса и выделите участника процесса, с которого начинается процесс. Очевидно, что это менеджер отдела продаж.
8. Обозначьте на диаграмме начало процесса символом  "Initial state" в графе, отведенной для изображения действий менеджера отдела продаж (рис. 9). Не забывайте, что работу с графическими формами можно осуществлять только при активированной пиктограмме с изображением стрелки на панели "Форматирование".
9. Направьте стрелку вниз от изображения начала процесса. Для этого перенесите с формы UML Activity изображение стрелки  (Control Flow) (рис. 9).
10. Пользуясь текстовым описанием, выделите действия, выполняемые менеджером отдела продаж. Действия (операции), выполняемые менеджером отдела продаж:
 - Операция № 1 "Получение от клиента заказа с указанием номенклатурных единиц товара по количеству, серийному соответствию, сроку годности"
 - Операция №2 "Проверка наличия у клиента лицензии на заказанные медикаменты";
 - Операция № 3 "Проверка наличия товарных запасов на складе";
 - Операция № 4 "Размещение заказа в реестре неудовлетворенный спрос";
 - Операция № 5 "Процесс формирования заявки на основании заказа и договора";
 - Операция № 6 "Резервирование товара";
 - Операция № 7 "Контроль кредитного лимита и дебиторской задолженности";

- Операция № 8 "Отклонение заявки".
11. Отобразите на диаграмме первые две операции, выполняемые менеджером отдела продаж. Для изображения действия на диаграмме используйте фигуру 
 12. Впишите внутри двух фигур  наименования и порядковые номера операций. Для ввода текста на панели инструментов "Стандартная" зафиксируйте пиктограмму с изображением буквы "А".
 13. Соедините операции в порядке их следования стрелками  (Control Flow).
 14. Отобразите ромб-символ проверки условия . Проведите из него две стрелки и надпишите их "Лицензия есть", "Лицензии нет". Стрелку с надписью "Лицензии нет" предстоит позже соединить с символом конца бизнес-процесса.
 15. Стрелку "Лицензия есть" соедините с операцией № 3 "Проверка наличия товарных запасов на складе", для изображения которой примените символ 
 16. Отобразите ромб-символ проверки условия . Проведите из него две стрелки и надпишите их "Достаточно запасов", "Недостаточно запасов".
 17. Стрелку с надписью "Недостаточно запасов" соедините с операцией № 4 "Размещение заказа в реестре "неудовлетворенный спрос".
 18. От операции № 4 направьте стрелку к операции № 3, для того чтобы отразить циклический процесс контроля выполнения отложенных заявок. Обратите внимание на то, что к операции № 3 уже направлена стрелка с надписью "Лицензия есть". Поскольку по правилам построения диаграмм в операцию может входить только одна стрелка, воспользуйтесь символом объединения независимых потоков  (Transition|Join).
 19. Сверьте созданный фрагмент диаграммы действий бизнес-процесса "Продажи" с [рисунком 10](#).

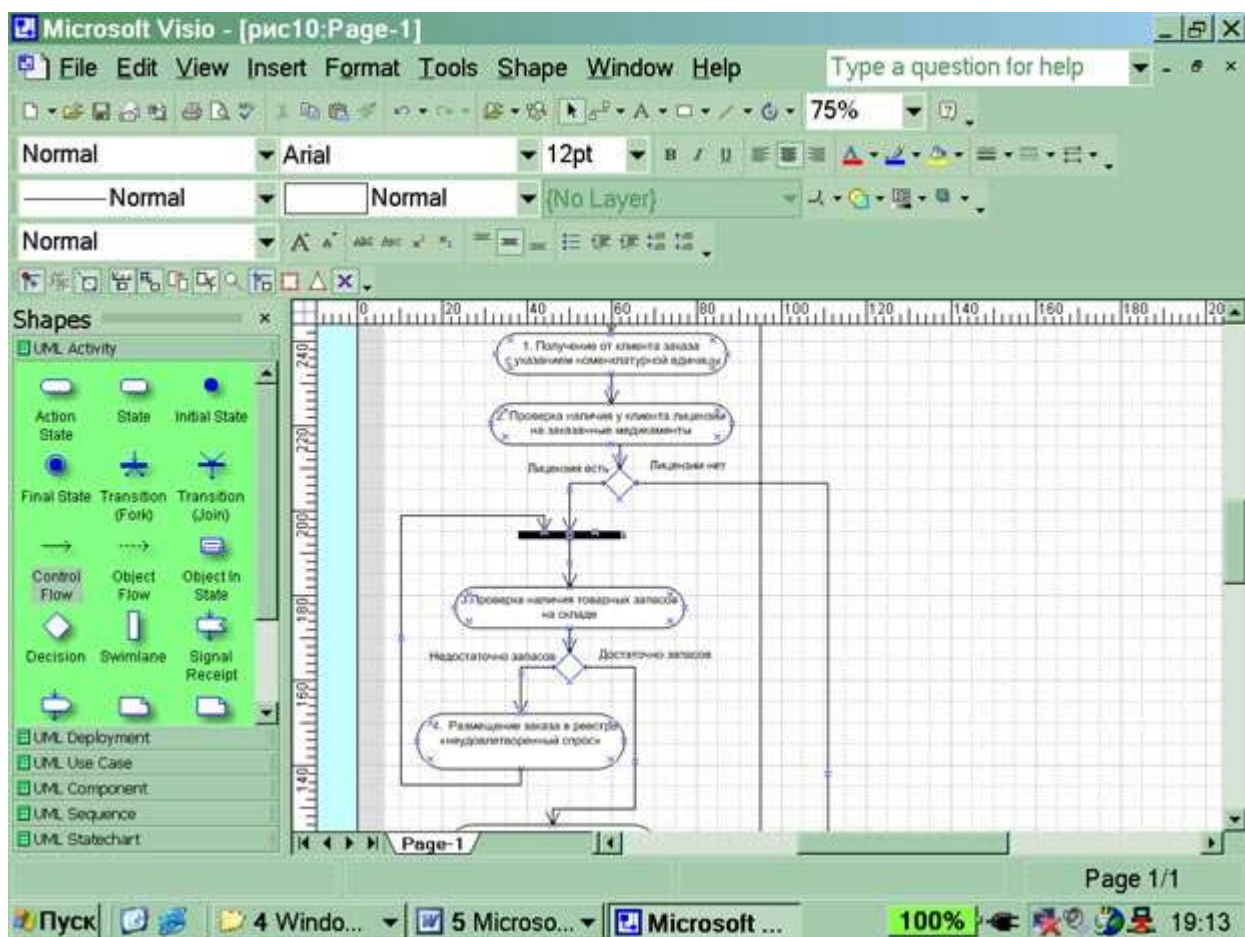






Рис. 10. Фрагмент диаграммы действий бизнес-процесса "Продажи"

20. Стрелку с надписью "Достаточно запасов" направьте к операции № 5 "Процесс формирования заявки на основании заказа и договора", для изображения которой используйте, как всегда, символ .
21. Поставьте в соответствие операции № 5 документ "Заявка". Для отображения документа на диаграмме используйте изображение прямоугольника. Нарисуйте прямоугольник мышкой, зафиксировав на панели инструментов "Стандартная" соответствующую пиктограмму Rectangle Tool.
22. Операция № 5 и получаемый в результате ее выполнения документ "Заявка" соединяются на диаграмме пунктирной линией. Для изображения пунктирной линии зафиксируйте пиктограмму Line Tool на панели инструментов "Стандартная" и выберите пунктирную линию на панели инструментов "Форматирование", используя меню пиктограммы (Line Patter). Затем мышкой нарисуйте соединение операции с документом.
23. За операцией № 5 последовательно отобразите операции № 6 "Резервирование товара" и № 7 "Контроль кредитного лимита и дебиторской задолженности".
24. Операцию № 7 соедините с ромбом  - символом проверки условия. Проведите из него две стрелки и надпишите их "Есть превышения", "Нет превышений".
25. Стрелку с надписью "Есть превышения" соедините с операцией № 8 "Отклонение заявки". В последующем операцию № 8 предстоит соединить с символом завершения бизнес-процесса. Заметьте, что на завершение бизнес-процесса уже направлена стрелка с надписью "Лицензии нет".

26. На этом действия менеджера отдела продаж завершаются, и к работе подключается менеджер учетно-операционного отдела. В поле действий менеджера учетно-операционного отдела последовательно отобразите выполняемые им операции: № 9 "Подбор номенклатурных единиц", № 10 "Формирование упаковочных листов", № 11 "Формирование счета, расходной накладной, счета-фактуры", № 12 "Отгрузка и списание медикаментов".
27. Соедините стрелками операции № 9, 10, 11, 12.
28. К операции № 9 подведите стрелку с надписью "Нет превышений".
29. Поставьте в соответствие операции № 10 документ "Упаковочный лист", операции № 11 - три документа: "Счет", "Расходная накладная", "Счет-фактура". Для отображения документа на диаграмме используйте изображение прямоугольника. Нарисуйте прямоугольник мышкой, зафиксировав на панели инструментов "Стандартная" соответствующую пиктограмму Rectangle Tool.
30. Операции № 10 и № 11 и получаемые в результате их выполнения документы соедините пунктирной линией. Для изображения пунктирной линии зафиксируйте пиктограмму Line Tool на панели инструментов "Стандартная" и выберите пунктирную линию на панели инструментов "Форматирование", используя меню пиктограммы (Line Patter). Затем мышкой нарисуйте соединение операции с документом.
31. Операция № 12 была последней в бизнес-процессе "Продажи", поэтому остается отобразить на диаграмме действий символ завершения процесса. На символ завершения бизнес-процесса направлено три стрелки, отображающие независимые процессы. Примените обозначение для объединения независимых процессов  (Transition|Join). После объединения трех процессов укажите на диаграмме конец процесса. Для этого используйте символ  (Final State).

Общий вид диаграммы действий бизнес-процесса "Продажи" представлен [рис. 11](#).

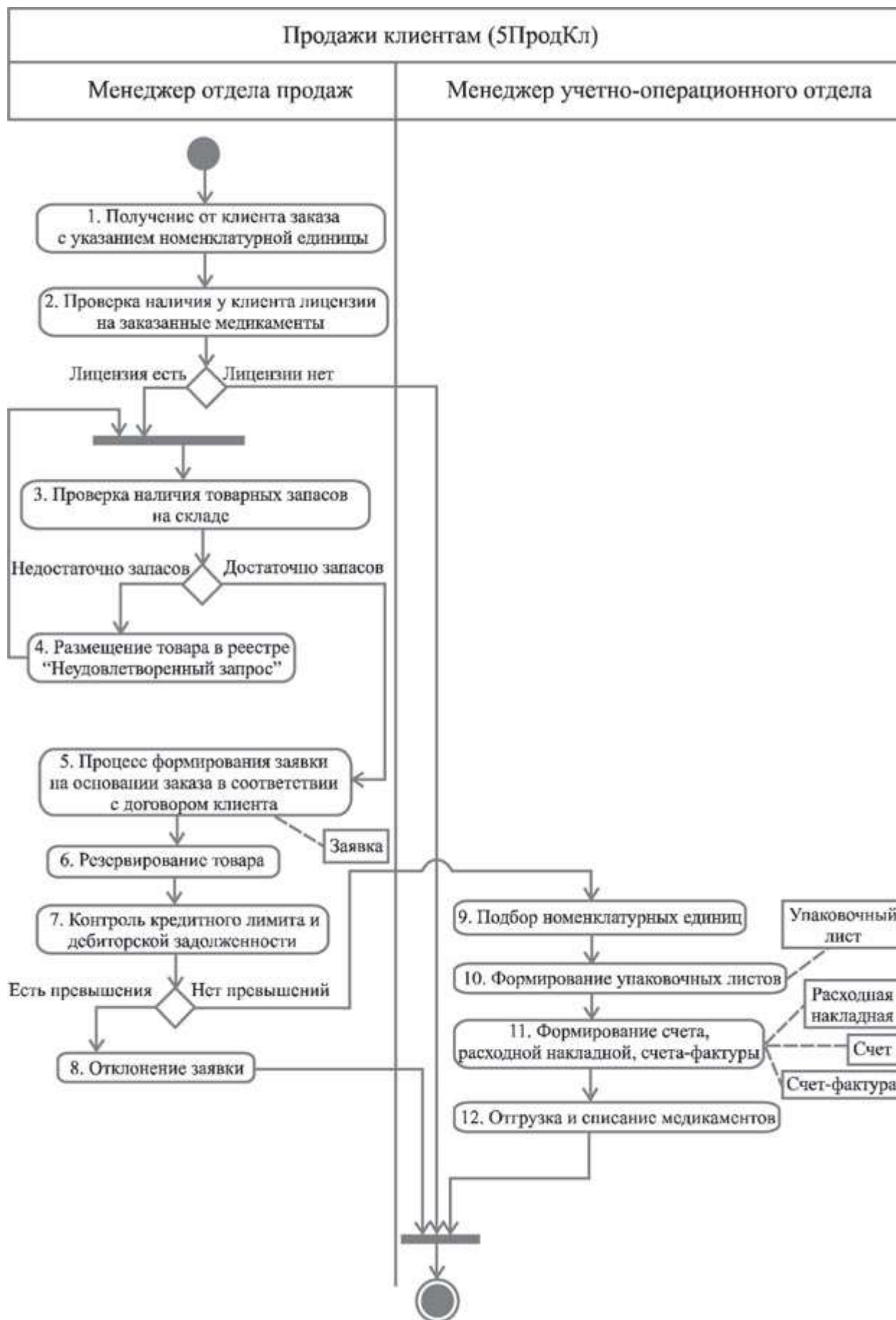


Рис. 11. Диаграмма действий бизнес-процесса "Продажи"

Задание 10. Формирование таблицы операций

Все операции, участвующие в процессе "Продажи", отразите в Таблице описания операций, имеющей следующий формат:

Диаграмма и номер на диаграмме	Операция	Исполнитель	Как часто	Входящие документы (документы-основания)	Исходящий документ (составляемый документ)	Проводка (дебет, кредит, сумма, аналитика)	Комментарий
1	2	3	4	5	6	7	8

Примечание. Далее заполненная форма таблицы описания операций будет использоваться для проектирования перечисленных в ней операций в информационной системе.

Выполнение задания 10

В таблицу последовательно внесите операции бизнес-процесса в соответствии с общим описанием и диаграммой действий.

1. В графе 1 проставьте краткое наименование диаграммы - 5Прод_Кл. Кроме того, в этой графе в каждой строке укажите номер операции, соответствующий изображению на диаграмме действий ([рис. 11](#)).
2. В графу 2 путем копирования перенесите из диаграммы действий наименование операций.
3. В графе 3 укажите исполнителя операции. В бизнес-процессе "Продажи" исполнителями операций являются менеджеры отдела продаж и учетно-операционного отдела. Графа 3 заполняется на основании диаграммы действий.
4. В графе 4 укажите, с какой частотой выполняется каждая операция. Проставьте частоту выполнения операций в соответствии с общим описанием бизнес-процесса. В данном примере все операции, кроме операций № 2 и № 4, выполняются ежедневно. Не регулярно, по мере необходимости выполняются операции № 2, 4. Данная информация фиксируется в ходе обследования компании. При проектировании или выборе системы данные из графы "Как часто" определяют требования к быстродействию системы и к параметрам сетевого варианта системы.
5. В графу 5 занесите наименования документов, на основании которых осуществляется выполнение операции (входящие документы). В данном примере основанием выполнения операций № 1-5 является Заказ номенклатурных единиц. Кроме того, основанием операции № 2 является также Справочник лицензий клиента, для операции № 3 - Картотека склада. Документ "Заявка на номенклатурные единицы" является основанием операций № 6, 7, 9, 11, 12. Для выполнения операции № 7 используются также Приказ по компании о кредитном

лимите и отчет о дебиторской задолженности. Операция № 12 требует также наличия Расходной накладной.

6. В графе 6 в соответствии с общим описанием укажите наименования документов, которые создаются в результате выполнения операций (исходящие документы). В данном бизнес-процессе исходящими (формируемыми) документами являются Заявка номенклатурных единиц, Реестр "неудовлетворенный спрос", Упаковочный лист, Счет, Расходная накладная, Счет-фактура.
7. Если на основании операции формируется бухгалтерская проводка, то она указывается в графе 7. В рассматриваемом примере проводка формируется по операции с номером 12.
8. Графа 8 предназначена для произвольной дополнительной информации.

Ниже представлена таблица описания операций бизнес-процесса "Продажи".

Описание операций бизнес-процесса "Продажи"							
Диаграмма и номер на диаграмме	Операция	Исполнитель	Как часто	Входящие документы (документы-основания)	Исходящий документ (составляемый документ)	Проводка (дебет, кредит, сумма, аналитика)	Комментарий
1	2	3	4	5	6	7	8
1Прод Кл 1	1. Получение от клиента заказа с указанной номенклатурной единицей	Отдел продаж	Ежедневно	Заказ номенклатурных единиц			
1Прод Кл 2	2. Проверка наличия у клиента лицензии на заказанные медикаменты	Отдел продаж	По мере необходимости	Заказ номенклатурных единиц Справочник лицензий клиента			
1Прод Кл 3	3. Проверка наличия товарных запасов на	Отдел продаж	Ежедневно	Картотека склада Заказ номенклат			

	складе			урных единиц			
1Прод Кл 4	4. Размещение заказа в реестре "неудовлетворенный спрос" при частичном или полном не выполнении заказа	Отдел продаж	По мере необходимости	Заказ номенклатурных единиц	Реестр "Неудовлетворенный спрос"		В реестре указывается заказ с неудовлетворенным спросом
1Прод Кл 5	5. Процесс формирования заявки на основании заказа в соответствии с договором клиента	Отдел продаж	Ежедневно	Заказ номенклатурных единиц	Заявка на номенклатурные единицы		В заявку копируются из договора скидки, наценки
1Прод Кл 6	6. Резервирование товара	Отдел продаж	Ежедневно	Заявка на номенклатурные единицы			
1Прод Кл 7	7. Проверка кредитного лимита и дебиторской задолженности	Отдел продаж	Ежедневно	Заявка на номенклатурные единицы Приказ о кредитном лимите Отчет по дебиторской задолженности			
1Прод Кл 9	8. Подбор заказанных номенклатур	Учетно-операционный	Ежедневно	Картотека складского учета			

	ных единиц	отдел		Заявка на номенклатурные единицы			
1Прод Кл 10	10. Формирование упаковочных листов	Учетно-операционный отдел	Ежедневно	Заявка на номенклатурные единицы	Упаковочный лист		
1Прод Кл 11	11. Формирование счета, расходной накладной, счета-фактуры	Учетно-операционный отдел	Ежедневно	Заявка на номенклатурные единицы	Счет Счет-фактура Расходная накладная		
1Прод Кл 12	11. Отгрузка и списание медикаментов	Учетно-операционный отдел	Ежедневно	Заявка на номенклатурные единицы	Расходная накладная	Д62-К41	

Задание 11. Формирование таблицы описания документов

Все документы, участвующие в бизнес-процессе, отразите в Таблице описания документов, имеющей следующий формат:

Диаграмма и номер на диаграмме	Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8

Примечание. После того как будут описаны документы, приступают к их разработке в ИС. Формы документов в учебном пособии не представлены, в практической же деятельности создается альбом форм, который является приложением к таблице описания документов.

Выполнение задания 11

Таблица описания документов создается путем переформирования (перестановки столбцов и объединения строк) таблицы описания операций. Особенности таблицы описания документов заключаются в следующем. В Графе 2 не должно быть повторяющихся наименований документов. Если один и тот же документ является исходящим на различных операциях, то он один раз указывается в графе 2 "Составляемый документ", а в графе 3 ему в соответствие ставятся несколько операций. Так же по наименованию документа следует объединить записи и в других графах.

В графе 7 указывается наименование реестра, в котором регистрируется создаваемый документ. Наименование реестру присваивается, как правило, по наименованию документа. В данном бизнес-процессе расходная накладная регистрируется в реестре расходных накладных, счет - в реестре счетов, счет-фактура - в реестре счетов-фактур, заявка - в реестре заявок.

На следующих страницах приведена таблица описания документов бизнес-процесса "Продажи".

Описание документов бизнес-процесса "Продажи".							
Диаграмма и номер операции на диаграмме	Составляемый документ (исходящий документ)	Входящие документы (документы-основания)	Операция	Исполнитель	Как часто	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8
5Прод Кл 4	1. Реестр "Неудовлетворенный спрос"	Заказ номенклатурных единиц	Размещение заказа в реестре "неудовлетворенный спрос" при частичном или полном невыполнении заказа	Отдел продаж	По мере необходимости	Реестр приходных накладных	
5Прод Кл 5	2. Заявка на номенклатурные единицы	Заказ номенклатурных единиц	Процесс формирования заявки на основании заказа в	Отдел продаж	Ежедневно	Реестр заявок	

			соответствии с договором клиента				
5Прод Кл 10	3. Упаковочный лист	Заявка на номенклатурные единицы	Формирование упаковочных листов	Учетно-операционный отдел	Ежедневно	Нет	
5Прод Кл 11	4. Счет	Заявка на номенклатурные единицы	Формирование счета	Учетно-операционный отдел	Ежедневно	Реестр счетов	
5Прод Кл 11	5. Счет-фактура	Заявка на номенклатурные единицы	Формирование счета-фактуры	Учетно-операционный отдел	Ежедневно	Реестр счетов-фактур	
5Прод Кл 11	6. Расходная номенклатура	Заявка на номенклатурные единицы	Формирование расходной номенклатуры	Учетно-операционный отдел	Ежедневно	Реестр расходных накладных	

Бизнес-процесс "Взаиморасчеты с клиентами"

Общее описание бизнес-процесса

Бизнес-процесс выглядит следующим образом:

1. Менеджер отдела продаж до 10 раз в день отгружает товары клиентам в соответствии с договорами и Приказом по кредитной линии. Одновременно с отгрузкой товара менеджер отдела продаж выставляет счет клиенту. Счет регистрируется в реестре счетов.
2. По факту произведенной отгрузки менеджер отдела продаж делает запись в журнале отгрузок и оплат, тем самым фиксируя задолженность клиента.
3. Бухгалтер компании ежедневно получает и обрабатывает выписки с расчетных счетов банков. Бухгалтер на основании банковской выписки определяет оплаченные счета и делает отметку об оплате счета в реестре счетов.
4. Менеджер отдела продаж ежедневно контролирует поступление платежей от клиентов, проверяя допустимый срок оплаты счета.
5. Если платежи по счету на расчетный счет компании не поступили и срок оплаты счета истек, то менеджер отдела продаж блокирует отгрузку товара клиенту. Если клиент оплатил счет, то менеджер вносит сведения об оплате в Журнал отгрузок и оплат.
6. Бухгалтер в конце каждого месяца выводит сальдо взаиморасчетов с клиентами.

Задание 12. Построение диаграммы действий

На основании общего описания бизнес-процесса "Взаиморасчеты с клиентами" составьте диаграмму действий, которая показывает участников процесса, выполняемые каждым участником операции и взаимосвязь между ними. Операции на диаграмме должны следовать в хронологическом порядке, который определен в приведенном описании бизнес-процесса.

Задание 13. Формирование таблицы операций

Все операции, участвующие в процессе "Взаиморасчеты с клиентами" отразите в Таблице описания операций, имеющей следующий формат:

Диаграмма и номер на диаграмме	Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8

Задание 14. Формирование таблицы описания документов

Все документы, участвующие в бизнес-процессе, отразите в Таблице описания документов, имеющей следующий формат:

Диаграмма и номер на диаграмме	Операция	Исполнитель	Как часто	Входящие документы (документы-основания)	Исходящий документ (составляемый документ)	Проводка (дебет, кредит, сумма, аналитика)	Комментарий
1	2	3	4	5	6	7	8

Бизнес-процесс "Взаиморасчеты с поставщиками"

Общее описание бизнес-процесса

Бизнес-процесс выглядит следующим образом:

1. Менеджер отдела закупок ежедневно получает от поставщика медикаментов счет на оплату, регистрирует его в реестре счетов поставщиков и передает счет поставщика бухгалтеру.
2. Бухгалтер на основании счета поставщика ежедневно формирует платежное поручение на оплату и передает платежное поручение в банк.
3. Бухгалтер на основании выписки с расчетного счета банка делает отметку об оплате счета в реестре счетов поставщика.
4. Менеджер отдела закупок при поступлении товара и (или) при оплате делает запись в Журнале поступлений и оплат.
5. Бухгалтер в конце каждого месяца выводит сальдо взаиморасчетов с клиентами.

Задание 15. Построение диаграммы действий

На основании общего описания бизнес-процесса "Взаиморасчеты с поставщиками" составьте диаграмму действий, которая показывает участников процесса, выполняемые каждым участником операции и взаимосвязь между ними. Операции на диаграмме должны следовать в хронологическом порядке, который определен в приведенном описании бизнес-процесса.

Задание 16. Формирование таблицы операций

Все операции, участвующие в процессе "Продажи", отразите в Таблице описания операций, имеющей следующий формат:

Диаграмма и номер на диаграмме	Составляемый документ (исходящий документ)	Операция	Кто составляет (исполнитель)	Как часто	Документы-основания (входящие документы)	Реестр, в котором регистрируется документ	Комментарий
1	2	3	4	5	6	7	8

Задание 17. Формирование таблицы описания документов

Спецификации настроек типовой ИС

Функциональность модулей ERP-системы MBS Ахарт.

Логистика

Прогноз закупок, продаж, запасов.

Описание номенклатуры с использованием трех аналитик.

Специальные цены, скидки для номенклатуры со специальной группой аналитики.

Описание хранения с использованием склада, палет и размещения.

Отслеживание номенклатур по серийному номеру и номеру партии

ABC-анализ по заданным пользователем критериям ABC-анализа по реализации, себестоимости и марже.

Управление карантинном. Просмотр номенклатуры на карантинном складе на любом этапе контроля качества.

Поддержка штрих-кодов.

Сводное планирование

Расчет потребности в материалах и мощностях.

Прогнозы закупок и продаж. Возможность обзора долгосрочных потребностей по закупке, производству и ресурсам.

Возможность расчета краткосрочных потребностей на основе существующих заказов и/или прогнозного планирования.

Получение сводного плана по заказам.

Система может предложить внести следующие изменения к существующим и спланированным заказам: Увеличение количества заказа, Уменьшение количества заказа, Отложить выполнение заказа или закупки

Управление продажами

Определение планов продаж для менеджеров по продажам и групп менеджеров (отделов продаж)

Управление процессом продаж

Прогнозирование продаж

Отслеживание статуса продаж, включая рассматриваемые предложения

Отслеживание действий и прогресса в работе отдельных сотрудников отдела продаж

Графическое представление данных по продажам

Создание отчетов по предложениям, деятельности менеджеров по продажам и отдельным сотрудникам

Торговля

Учет и размещение номенклатуры на складе

Создание закупок напрямую из заказа

Пересчет единиц измерения по закупке в единице учета на складе

Обработка недопоставок

Автоматическая замена товаров, которых нет в наличии на складе, на другие альтернативные товары

Управление складом

Регистрация и размещение товара, возможность хранения товара в соответствии со структурой склада

Идентификация физического размещения: склад, ячейка и палеты

Идентификация истории происхождения номенклатуры: серийный номер и номер партии

Характеристики товара: конфигурация, цвет и размер

Возможность маркировки как отдельной номенклатуры, так и группы номенклатур с целью дальнейшего отслеживания

Ведение журналов приемки

Возможность перехода из заказов на отгрузку в ячейки комплектации через журналы отгрузки

Учет договоров

Ведение юридической информации о договорах с клиентами и поставщиками, условиях оплаты, контактах и ответственных

Привязка накладных и оплат к конкретному договору (указание договора в строках журналов ГК, заказах, закупках, накладных и оплатах с последующим переносом в проводку по клиенту/поставщику)

Включение атрибутов договоров в предложения по оплате

Автоматическое/периодическое сопоставление проводок по контрагентам и договорам

Форма ручного сопоставления в рамках договоров

Сальдо расчетов в рамках отдельного договора

Номер договора в проводках по курсовой разнице

Переход от моделей предметной области к функциональной модели системы

Бизнес-процесс "Планирование закупок и размещение заказов поставщикам"

Задание 18. Проектирование реализации операций бизнес-процесса в информационной системе (ИС)

Все операции, участвующие в процессе, отразите в Таблице проектирования операций, имеющей следующий формат:

Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5

Выполнение задания 18

1. В графе 1 укажите номер операции и краткое наименование диаграммы действий проектируемого бизнес-процесса. Данные в графу введите в соответствии с таблицей описаний операций.
2. В графу 2 перенесите наименования операций из таблицы описания операций.
3. В графе 3 перечислите необходимые разработки для реализации операций.
4. В графе 4 сформулируйте специфику настройки функционала системы.
5. В графе 5 укажите наименование модуля или функции, необходимые для реализации операции бизнес-процесса.

Пример проектирования операций бизнес-процесса "Планирование закупок и размещение заказов поставщикам" в ИС приведен в таблице.

Проектирование реализации операций бизнес-процесса в ИС				
Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5
1a (1Пл_Зак)	1. Получение внутренней статистики	Разработка узла хранения данных статистики продаж Разработка механизма импорта статистики	Коды клиентов в файле соответствуют кодировке в Системе Единицы измерения номенклатуры соответствуют	Продажи, клиенты

			единицам измерения в Системе	
			Коды номенклатуры статистики соответствуют кодам номенклатуры Системы	
16 (1Пл_Зак)	2. Получение внешней статистики продаж	Разработка узла хранения данных статистики продаж	Коды клиентов в файле соответствуют кодировке в Системе	Продажи, клиенты
		Разработка механизма импорта статистики	Единицы измерения номенклатуры соответствуют единицам измерения в Системе	
			Коды номенклатуры статистики соответствуют кодам номенклатуры Системы	
2 (1Пл_Зак)	3. Расчет потребностей в товаре	Разработка механизма автоматического формирования минимального и максимального запаса препаратов (ассортиментный план на период планирования) эффективности закупок (ABC и XYZ		Сводное планирование, логистика, торговля

3 (1Пл_Зак)	4. Регистрация прайс-листов поставщиков в системе	классификации) Разработка механизма импорта электронной версии прайс-листа в форму "коммерческого соглашения"	В системе регистрируется один базовый прайс-лист, на его основе формируются все другие прайс-листы	Коммерческие соглашения
4, 5 (1Пл_Зак)	5. Анализ прайс-листов поставщиков и действующих контрактов. Выбор поставщиков приоритетных и запасных по каждой позиции	Разработка механизма реализации в системе оценки эффективности закупки на основании полученных прайс-листов, с учетом условий поставки (скидки, отсрочка платежа)		Продажи, клиенты
6 (1Пл_Зак)	6. Формирование (регистрация) графика поставок (сроки, периодичность) без указания количеств	Разработка графика поставок (календарь рабочего времени) для каждого поставщика	В качестве графика поставок (график обращений) используется календарь рабочего времени для каждого поставщика	Календарь рабочего времени
7, 8 (1Пл_Зак)	7. Формирование заказов поставщикам с учетом складских остатков, товара в пути и резервного запаса	Разработка взаимосвязанных данных таблиц Заказов, Складских остатков, Товара в пути, Резервных заказов	При заполнении в заказе поля "Количество" система в первую очередь "просматривает" количество товаров на складе. При недостаточном количестве товаров на складе	Сводное планирование, логистика, торговля

			система обращается к таблице с данными о резервных запасах. При недостаточном количестве резервных запасов система осуществляет поиск заданной в заказе номенклатуры в таблице Товары в пути	
9 (1Пл_Зак)	8. Расчет затрат на сертификацию	Разработать механизм расчета затрат на сертификацию при формировании рабочего прайс-листа	Расчет затрат на сертификацию производится перед формированием прайс-листа поставщика. При формировании рабочего прайс-листа помимо учета скидок, отсрочек платежа, времени движения товара в пути (в денежном выражении), также необходимо учесть стоимость сертификации	Коммерческие соглашения
10 (1Пл_Зак)	9. Проверка суммы затрат на сертификацию на неперевышение		Разработка алгоритма проверки суммы затрат на сертификацию на неперевышение	Сводное планирование, логистика, торговля

	нормы		внутрифирменно й нормы	
11 (1Пл_Зак)	10. Подпись заказа менеджером по логистике, директором	Разработать процедуру утверждения строк спланированных заказов	Результатом процедуры "Сводное планирование" в форме "Спланированные заказы" являются строки. После оценки строк в форме спланированные заказы необходимо провести процедуру одобрения (утверждения) строк спланированных заказов	Сводное планирование, логистика, торговля
12 (1Пл_Зак)	11. Направление заказа в отдел закупок	Разработка многопользовательско й системы, прав доступа к документам		Сводное планирование, логистика, торговля

Бизнес-процесс "Запасы-склад (приходование товара)"

Задание 19. Проектирование реализации операций бизнес-процесса в информационной системе (ИС)

Все операции, участвующие в процессе, отразите в Таблице проектирования операций, имеющей следующий формат:

Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5

Выполнение задания 19

1. В графе 1 укажите номер операции и краткое наименование диаграммы действий проектируемого бизнес-процесса. Данные в графу введите в соответствии с таблицей описаний операций.
2. В графу 2 перенесите операции из таблицы описания операций.
3. В графе 3 перечислите необходимые разработки для реализации операций.
4. В графе 4 сформулируйте специфику настройки функционала системы.
5. В графе 5 укажите наименование модуля или функции, необходимые для реализации операции бизнес-процесса.

Пример проектирования операций бизнес-процесса "Планирование закупок и размещение заказов поставщикам" в ИС приведен в таблице.

Бизнес-процесс "Продажи"

Задание 20. Проектирование реализации операций бизнес-процесса в информационной системе (ИС)

Все операции, участвующие в процессе, отразите в Таблице проектирования операций, имеющей следующий формат:

Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5

Выполнение задания 20

1. В графе 1 укажите номер операции и краткое наименование диаграммы действий проектируемого бизнес-процесса. Данные в графу введите в соответствии с таблицей описаний операций.
2. В графу 2 перенесите операции из таблицы описания операций.
3. В графе 3 перечислите необходимые разработки для реализации операций.
4. В графе 4 сформулируйте специфику настройки функционала системы.
5. В графе 5 укажите наименование модуля или функции, необходимые для реализации операции бизнес-процесса.

Пример проектирования операций бизнес-процесса "Продажи" в ИС приведен в таблице.

Планирование закупок и размещение заказов поставщиками				
Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5

4Склад 1	1. Проверка товара по количеству, серийному соответствию, сроку годности	Не требуется		Не требуется
4Склад 2	2. Отражение в базе данных количества и учетной цены товара	Разработать электронную и печатную формы приходной накладной и взаимосвязь с финансовым блоком для формирования проводки	В системе формируются проводки, а также финансовый приход товара на склад	Логистика, торговля
4Склад 3	3. Поиск серии в справочнике	Разработать справочник серий и систему поиска в нем	Перед тем, как создать новый серийный номер, необходимо произвести поиск серийного номера в базе данных	Логистика
4Склад 4	4. Добавление серии в справочник	Разработать функцию пополнения справочника		Логистика
4Склад 6	5. Разбиение каждой позиции номенклатуры по сериям	Разработать иерархический номенклатурный справочник		Логистика
4Склад 7	6. Процесс размещения серии товара на складе	Разработать разбиение мест хранения по зонам Разработать функцию автоматического подбора зоны хранения Разработать форму	Настроить аналитику серийный номер обязательную для заполнения при реализации	Логистика

4Склад 8	7. Определение и ввод базовой цены товара	карточки товара с возможностью указания в ней места хранения Разработать базовый прайс-лист	Базовый прайс- лист формируется на основании данных отдела маркетинга	Коммерческие соглашения
-------------	---	--	--	----------------------------

Бизнес-процесс "Взаиморасчеты с клиентами и поставщиками"

Самостоятельно выполните следующее задание.

Задание 21. Проектирование реализации операций бизнес-процесса в информационной системе

Все операции, участвующие в процессе, отразите в Таблице проектирования операций, имеющей следующий формат:

Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5

Дальнейшая настройка типовой ИС (выбор экранных форм, формирование отчетов и пр.) осуществляется с использованием специфических средств, предусмотренных в каждой системе.

Пример проектирования операций бизнес-процесса "Продажи"				
Номер операции на диаграмме	Операция	Необходимые разработки	Специфика настройки	Функциональность (модуль) системы
1	2	3	4	5
5ПродКл-1	Получение от клиента заказа с указанной номенклатурной единицей	Разработать шаблон импорта заявки клиента. Разработать функционал импорта заявки	Шаблон может быть разработан на основе "MS Excel", "MS Word"	Логистика, расчеты с клиентами

5ПродКл-2	Проверка наличия у клиента лицензии на заказанные медикаменты	клиента в заказы Разработать функционал, позволяющий при импорте заявки клиента в систему проверять наличие у клиента лицензии	Каждому клиенту ставится в соответствие лицензия. В лицензии указывается срок ее действия	Логистика
5ПродКл-3	Проверка наличия товарных запасов на складе для полного или частичного выполнения заказа	Разработать функционал, проверяющий наличие товарных запасов при импорте заявки клиента	При импорте заявки система проверяет наличие товара на складе	Логистика, сводное планирование
5ПродКл-4	Размещение заказа в реестре "неудовлетворенный спрос"	Разработать функционал, который при импорте заказа клиента и отсутствии товара на складе формирует заказ с меткой "отложен"	При импорте заявки, в случае отсутствия товара на складе, система формирует заказ со специальной меткой и размещает его в специальном реестре	Логистика
5ПродКл-5	Процесс формирования заявки на основании заказа в соответствии с договором клиента	Разработать механизм копирования в Заявку строк из Заказа и Договора	Шаблон файла импорта должен содержать номер договора, по которому формируется заявка	Логистика
5ПродКл-6	Резервирование товара	Разработать функционал, позволяющий	Резервирование товара может происходить в	Логистика

		резервировать заказанный товар	системе двумя способами: автоматически и вручную	
5ПродКл-7	Проверка кредитного лимита и дебиторской задолженности	Разработать алгоритм проверки условий	При импорте заявки клиента системы проверяет кредитный лимит клиента. В случае превышения кредитного лимита система выдает сообщение о превышении кредитного лимита и блокирует дальнейшую обработку	Торговля
5ПродКл-8	Подбор номенклатурных единиц	Автоматизации не подлежит		Функциональность не требуется
5ПродКл-10	Формирование упаковочных листов	Разработать функционал, позволяющий резервировать заказанный товар		Логистика, управление складом
5ПродКл-11	Формирование счета, расходной накладной, счета-фактуры	Разработать электронную и печатную форму документов		Торговля
5ПродКл-12	Отгрузка, списание медикаментов	Разработать взаимосвязь с финансовым блоком	Формирование проводки по выбытию товара со склада	Логистика, торговля

Средства контроля качества обучения

Вопросы к экзамену 7 семестр:

1. Понятия «база данных», «система управления базами данных» (СУБД), «модель данных».
2. Базовые понятия реляционной модели данных.
3. Связанные отношения.
4. Основные свойства отношений.
5. Нормализация данных.
6. Основные функции СУБД.
7. Эволюция СУБД.
8. Концептуальная, логическая и физическая модели данных.
9. Концептуальное моделирование структуры данных, модель «сущность – связь».
10. Создание физической модели данных.
11. Модификация структуры данных.
12. Типы команд SQL.
13. Типы данных SQL.
14. Создание, модификация и удаление таблиц в SQL.
15. Задание ограничений в SQL.
16. Определение значений по умолчанию в SQL.
17. Создание и удаление индексов в SQL.
18. Хранимые процедуры.
19. Триггеры.
20. Добавление в таблицу новой информации с использованием SQL.
21. Изменение данных, хранящихся в таблице при помощи SQL.
22. Оператор SELECT.

Вопросы к экзамену 8 семестр:

1. Задание условий при выборке данных с использованием SQL.
2. Сортировка данных.
3. Выборка данных из нескольких таблиц.
4. Использование вычисляемых полей.
5. Функции агрегирования.
6. Группировка данных.
7. Объединение запросов.
8. Управление безопасностью базы данных, привилегии пользователей.
9. Управление доступом к базе данных.
10. Возможности системы программирования для работы с базами данных.
11. Технология ADO.NET.
12. ServerExplorer.
13. Соединение с базой данных.
14. Хранимые процедуры.
15. Командная строка SQL-запроса CommandText.
16. Параметризованные запросы.
17. Транзакции.

18. Вызов хранимой процедуры.
19. Объекты DataSet, DataTable и DataColumn.
20. Объект DataRow.
21. Объект DataGridView.
22. Объект DataView.