МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

«ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В УПРАВЛЕНИИ»

для студентов направления подготовки
«Государственное и муниципальное управление»

Томск - 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ
Зав. Кафедрой АОИ
Д.т.н., профессор
_____ Ю. П. Ехлаков
«___» _____ 2016 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения лабораторных работ и организации самостоятельной работы
по дисциплине «**Информационные технологии в управлении**»

для студентов направления
Государственное и муниципальное управление

Разработчик
доцент каф. АОИ
канд.техн.наук, с.н.с.
_____О.И. Жуковский

Томск 2016

Содержание

Методические указания по выполнению лабораторных работ

Целью проведения лабораторных работ является:

- развитие навыков использования информационных технологий в процессе выработки управленческих решений;
- закрепление материала, изученного на лекционных и практических занятиях путем решения комплексных задач.

Все занятия двухчасовые. Занятия проводятся в компьютерном зале каф. АОИ.

Лабораторная работа №1. Разметка электронных документов

1. Указания по выполнению работы

Цель данной работы – овладеть основами HTML и создать прообраз личной домашней странички, которая и покажет глубину полученных навыков в создании HTML-документов.

Для успешного выполнения всей работы необходимо прочитать приведенный в методическом описании материал и выполнить все приведенные примеры. При выполнении примеров Вы получаете возможность соотнести полученное представление о работе конструкций HTML с реальным эффектом от их применения.

1.1. Основные этапы выполнения задания

1. Знакомство с базовыми элементами языка HTML

Прочитайте раздел 2, который дает представление о базовых элементах языка. Создайте с помощью выбранного вами текстового редактора документ из примера 1 и просмотрите его с помощью вашего броузера, что позволит вам освоить основную технологическую цепочку – создание документа и его просмотр.

Для получения навыка в формировании тела документа создайте и просмотрите документ из примера 2.

2. Изучение основ форматирования текста

Прочитайте раздел 3. Для получения навыков в форматировании текста создайте и просмотрите документ из примера 3. Для изучения особенностей технологии использования графических изображений в документе выполните пример 4. Все указанные в примете графические файлы находятся в директории sample4.files, местоположение которой

вам укажет преподаватель. Попробуйте использовать в примере свои графические файлы

После знакомства с основами форматирования табличных данных выполните пример 5. Необходимые для его выполнения графические файлы находятся в директории sample5.files.

Для закрепления знаний по созданию списков и меню выполните пример 6. Попробуйте расширить пример, сделав список студентов группы и преподавателей вашей кафедры.

Изучив раздел 3.6., посвященный форматированию с использованием таблицы стилей, выполните пример 7. Необходимый графический файл находится в директории sample7.files.

3. Создание форм и фреймов

Прочитайте раздел 4. Для закрепления навыков создания форм выполните пример 8. Попытайтесь модифицировать форму, изменив содержание текстов предпочитаемой деятельности.

Выполните пример 9, который позволит вам закрепить навыки использования фреймов. Все необходимые для выполнения графические файлы находятся в директории sample9.files.

4. Изучение средств расширения возможностей стандартных HTML-документов

Внимательно прочитайте раздел 5, который потребует от вас хороших навыков в области программирования. Выполните пример 10, который позволит получить вам начальные навыки использования javascript.

Выполнив пример 11, вы получите начальные навыки работы с DHTML. Необходимые для выполнения этого примера графические файлы находятся в директории samples11.files. Выполнив пример 12, вы получите начальные навыки в использовании элементов анимации в HTML-документах. Необходимые графические файлы находятся в директории samples12.files. Выполнение примера 13 позволит вам освоить технологию предварительной обработки данных из ваших форм а пример 14 даст основные навыки управления окнами броузера.

5. Создание WEB-страницы

Используя полученные навыки, создайте собственную WEB-страницу. Страница должна состоять не менее чем из трех документов. Обязательно использование списков, таблиц, изображений. На странице должны быть как внутренние ссылки, так и ссылки на внешние Internet-ресурсы.

В качестве отчета представьте текст всех документов вашей страницы и продемонстрируйте ее преподавателю с помощью броузера.

1.2. Рекомендации по программному обеспечению работы

Прежде всего, вы должны выбрать для себя - вы хотите сами вручную описывать все конструкторы языка - таблицы, тип фонта , изображения и т. д. или вы хотите переложить эти черновые функции на редактор?

Существуют два основных типа программ обработки: обычные текстовые редакторы Editors и Composers. В первом случае этот способ позволит вам использовать абсолютно все возможности HTML. Для работы в этом режиме можно использовать любой редактор текстов или редактор программ. Существуют много Composers, позволяющих упростить процесс разработки страниц.

Можно применить и комбинированный метод: в Composer создать страницу, а в Editor просмотреть ее "изнутри" и внести необходимые дополнения. Можно и наоборот: создать страницу в Editor, а изменения вносить в Composers.

Отметим, что качество исходного кода гораздо лучше при работе с Editor. С Composer, конечно, работать удобнее, но создайте с его помощью раздел и посмотрите исходный текст - там очень много лишнего. А ведь размер исходного кода - один из показателей качества Вашей страницы.

2. Основы HTML

2.1. Базовые элементы языка и структура HTML документа

Базовым понятием языка разметки гипертекста является ***элемент***. Пара меток (тэги), открывающая и закрывающая, используется для выделения элементов в тексте, так же, как разные скобки или кавычки используются в обычной пунктуации. Открывающая метка имеет вид <название>, где открывающая угловая скобка означает начало открывающей метки, «название» - обобщенный идентификатор отмечаемого элемента, и закрывающая угловая скобка означает конец метки. Закрывающая метка имеет аналогичный вид, за исключением того, что за открывающей угловой скобкой стоит символ косой черты, так что соответствующая закрывающая метка будет </название>.

HTML документ представляет собой обычный текстовый файл, содержащий размеченный элементами текст, а так же заданные

специальными элементами ссылки на графические и прочие файлы мультимедиа, ссылки на другие документы HTML и ресурсы Internet.

Документ HTML начинается открывающим тегом <HTML> и заканчивается закрывающим тегом </HTML>. Между данной парой тегов располагаются две другие основные части HTML документа: элемент ***заголовок***, заключенный в тэги <HEAD>...</HEAD> и ***тело*** документа в тэгах <BODY>...</BODY>. Таким образом, структура простого HTML документа выглядит так:

<HTML> - Начало документа
<HEAD>
ЗАГОЛОВОК ДОКУМЕНТА
</HEAD>
<BODY>
ТЕЛО ДОКУМЕНТА
</BODY>
</HTML> - Конец документа

2.2. Ссылки в HTML документах

Как было сказано выше, HTML-документ это обычный текстовый файл. Гипертекстовым его делают содержащиеся в нем ссылки на другие документы и ресурсы Internet. Рассмотрим, что такое ссылка, какие бывают типы ссылок и как их задать в документе.

Ссылка состоит из двух компонентов: так называемого якоря (anchor) или элемента привязки и URL (Universal Resurse Locator) связанного с ним ресурса Internet.

Первый компонент ссылки – якорь. Это текстовый или графический объект, который, как правило, служит органом управления на Web-странице. Каждый раз при просмотре Web-страниц мы видим множество различных элементов-якорей. Это и красочные рекламные баннеры, всевозможные кнопки и иконки, выделенные подчеркнутым курсивом элементы текста, адреса электронной почты.

Второй компонент ссылки не отображается Web-броузером, но служит конкретным указанием, где в Internet найти, и что сделать при активизации пользователем соответствующего ему якоря.

Адреса ресурсов бывают относительные и абсолютные. Относительный адрес это адрес ресурса относительно компьютера и каталога загрузки HTML-документа, если иной базовый адрес не указан в заголовке документа (элемент <BASE>). Относительный адрес задается в сокращенной форме (путь/файл). Например, если ваша начальная страница index.htm загружена браузером с

http://www.site.ru, то использование в ней относительной ссылки resume.htm означает загрузку http://www.site.ru/resume.htm.

Абсолютные адреса используются для привязки к ресурсам других узлов Internet и задаются полным форматом записи (http://компьютер/путь/файл). Например: http://www.sitename.ru/fil.htm.

Ссылки в документах задаются при помощи элемента <A> ...</A>, следующей структуры:

<A HREF="Ресурс" TARGET="имя окна" TITLE="Подсказка">Элемент - якорь</A>

Атрибут HREF в открывающем теге задает ресурс, который необходимо обработать браузеру при выборе на Web-странице, соответствующего ему якоря. Рассмотрим наиболее часто используемые ресурсы:

<A HREF="URL"> - ссылки на другие документы HTML и файлы.

<A HREF="ftp://ftp_server/path/filename"> - ссылки на файлы FTP-сервера.

<A HREF="mailto:e-mail"> - ссылки на адреса электронной почты.

Атрибут TITLE задает текстовую подсказку в стиле ToolTip, отображаемую браузером при позиционировании указателя-курсора в зоне элемента-якоря.

Элемент-якорь выделяется браузером особым образом (текст - цветом и подчеркиванием, графика - рамкой) при отображении на Web-странице. Можно задать свой способ выделения элемента-якоря в атрибутах тега <BODY> - тела документа.

Рассмотрим несколько конкретных примеров использования ссылок в документах:

<A HREF="http://www.site.ru" TITLE="Переход на www.site.ru"> Заходите к нам на огонек </A> - абсолютная ссылка: переход на сайт www.site.ru, текстовый якорь - Заходите к нам на огонек, с подсказкой.

<A HREF="/VW/cars.htm"> Модельный ряд VW </A> - относительная ссылка: открытие станицы cars.htm в подразделе VW относительно раздела основной страницы, текстовый якорь - Модельный ряд VW, без подсказки.

<A HREF="mailto:webmaster@freemail.ru">Связь с вебмастером </A> - загрузка интерфейса к почтовой системе пользователя с автозаполнением реквизитов получателя, текстовый якорь - Связь с вебмастером, без подсказки.

<A HREF="ftp://ftp.site.ru/soft/driver.zip">Новый драйвер здесь </A> - доступ на FTP-сервере к файлу драйвера, текстовый якорь- Новый драйвер здесь, без подсказки.

При использовании графического файла в качестве элемента-якоря необходимо вместо текста в элементе <A>...</A> использовать конструкцию <IMG SRC="файл"> (См. раздел 3.2.). Например:

<A HREF="/VW/passat.htm" > <IMG SRC="/VW/CARS/passat.gif"> </A> - относительная ссылка: открытие станицы passat.htm в подразделе VW относительно раздела основной страницы, графический якорь - passat.gif, без подсказки.

Кроме вышеперечисленных ссылок существуют еще внутренние ссылки или закладки. Этот тип ссылок используется для удобства перемещения в пределах документа. Для использования в HTML-документе закладок необходимо задать имена тех областей документа, на которые необходимо ссылаться.

Имя закладки в теле документа задается использованием атрибута NAME=ИмяЗакладки в элементе <A>...</A>. Причем в данном случае текст, заключенный в элемент, не является элементом-якорем (но выводится). Например, для перехода на начало документа необходимо поместить там закладку:

<A NAME=DocBegin>Начало документа<A>

Внутренняя ссылка на закладку в документе имеет следующий формат:

<A HREF="URL документа#ИМЯ">Элемент - якорь</A>

Например, для размещения в документе ссылки на внутреннюю закладку (содержащуюся в данном документе) необходимо применить:

<A HREF="#DocBegin">Перейти к началу документа</A>

А для размещения в документе ссылки на внешнюю закладку (например содержащуюся в файле Doc1.htm) необходимо применить:

<A HREF="Doc1.htm#DocBegin">Перейти к началу документа Doc1.htm</A>

В заключении надо описать еще один важный атрибут тега ссылки, это атрибут TARGET. Данный атрибут задает окно либо фрейм назначения для документа заданного атрибутом HREF. По умолчанию загрузка происходит в текущее окно браузера, но можно указать имя нового или существующего окна, а так же одно из предопределенных имен объектов браузера: _blank, _self, _parent, _top. Например:

<A HREF="sample/sample.htm" TARGET="new_win">Пример</A> - загрузка документа sample.htm в новое окно браузера с именем "new_win".

2.3 Заголовок HTML документа

Заголовок является необязательной частью структуры HTML документа и служит для определения служебной информации и

названия документа. В случае использования в документе элемента заголовка <HEAD>...</HEAD> единственным обязательным его элементом является элемент <TITLE>...</TITLE>, который задает имя документа. Именно это имя пользователь видит в заголовке окна браузера при просмотре Web-страниц в Internet.

Все остальные элементы заголовка не отображаются браузером и служат для определения различных свойств документа, его взаимосвязи с другими Web-страницами и служебной информации для внешних программ. Попробуем разработать типовой заголовок для документов на примере Web-страницы об автомобилях.

Внимание! В этом и в дальнейших примерах используются выдуманные e-mail адреса!

Пример 1. ***Формирование заголовка документа***.

```
<HTML>
<HEAD>
<TITLE>Автомобили Фольксваген</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1251">
<META NAME="Keywords" LANG=ru CONTENT="автомобили, авто, Фольксваген, Гольф, Бора, Поло, Пассат, Жук">
<META NAME="Description" CONTENT="Модельный ряд автомобилей Фольксваген - 2000 года">
<BASE HREF="http://www.cars.ru/vw">
<LINK REL="author" HREF="mailto:autofan@mail.ru">
</HEAD>
<BODY>
В разработке.
</BODY>
</HTML>
```

В примере первый и обязательный элемент заголовка это элемент <TITLE>...</TITLE>, определяющий имя документа, отображаемое в шапке окна браузера.

Далее следует последовательность <META> тегов, задающих так называемую мета (или внешнюю) информацию о документе. У <META> тегов наиболее часто используются следующие атрибуты:

HTTP-EQUIV - задать имя мета-записи в документе;

NAME - задать имя дополнительной мета-записи (по умолчанию NAME=HTTP-EQUIV);

CONTENT - присвоить значение мета-записи заданной атрибутом NAME или HTTP-EQUIV;

LANG - язык описания значений мета-записи;

В нашем примере первый <META HTTP-EQUIV="Content-Type"> тег описывает тип и кодировку содержимого документа. Два следующих <META> тега служат для передачи информации о содержании документа поисковым службам Internet.

Тег <META NAME="Keywords" LANG=ru CONTENT=" "> задает список ключевых слов, содержащихся в документе, а тег <META NAME="Description" CONTENT=" "> является словесным описанием содержимого документа.

Далее следует тег <BASE HREF="URL">, задающий базовый адрес данного документа. Это необходимо для поддержания работоспособности относительных ссылок, в случае миграции документа в Internet или изменения каталога его загрузки. Как уже говорилось выше, при отсутствии тега <BASE> относительные ссылки в документе определяются от адреса его загрузки.

Завершает наш заголовок тег <LINK>. Данный тег не отображает информацию в окне браузера и предназначен для формирования различных типов отношений между документами и другими объектами. Данные отношения помогают разработчикам ориентироваться в структуре сложного документа и используются поисковыми системами. Рассмотрим, какие бывают отношения и как они задаются. У тега <LINK> наиболее часто используются следующие атрибуты:

REV - отношение текущего документа с другим, заданным HREF (обратное REL);

REL - отношение между документом заданным HREF и текущим документом (обратное REV);

HREF - задает URL документа или объекта;

LANG - языковая версия;

MEDIA - назначение документа (Print/Screen);

TYPE - тип содержимого связанного объекта (листа стилей);

Данный тег довольно редко используется. Как правило, его применение ограничивается привязкой листа стилей (stylesheet) к документу, но в HTML-документах со сложной иерархической структурой иногда встречается множество тегов <LINK> с довольно запутанным синтаксисом. Наиболее понятные из них связи типа: следующий/предыдущий (next/prev), документ/автор (author), документ/оглавление (index). В нашем примере тег <LINK> использован для формирования связи документ/автор.

2.4. Тело HTML документа

Тело HTML документа располагается после заголовка и ограничено элементом <BODY>...</BODY>. Содержимое тела документа отображается в окне Web-браузера и состоит из маркированного тегами форматирования текста, таблиц, графических и прочих мультимедиа-элементов, ссылок на другие ресурсы и различных органов управления.

Так как в элементе <BODY> нами не были заданы атрибуты внешнего вида документа, то данный текст будет отображен в окне браузера в соответствии с пользовательскими установками. Для обеспечения соответствия между внешним видом документа в браузере пользователя и дизайном разработчика необходимо использовать специальные атрибуты тега <BODY>:

BGCOLOR - цвет фона документа;

BACKGROUND - URL графического изображения, для создания фона;

BGPROPERTIES - режим прокрутки фона по отношению к тексту документа (по умолчанию прокрутка вместе с текстом, BGPROPERTIES=FIXED - стационарный фон);

TEXT - цвет текста документа;

LINK - цвет выделения элементов-якорей ссылок;

ALINK - цвет выделения активных элементов-якорей ссылок;

VLINK - цвет выделения элементов-якорей просмотренных ссылок;

Для определения цветов в HTML применяются шестнадцатеричные коды RGB компонентов, но стандартные 16 цветов можно задавать по их общепринятым названиям:

BLACK=#000000	MAROON=#800000
GREEN=#008000	OLIVE=#808000
GRAY=#808080	NAVY=#000080
RED=#FF0000	PURPLE=#800080
YELLOW=#FFFF00	TEAL=#008080
BLUE=#0000FF	LIME=#00FF00

WHITE=#FFFFFF	FUSHSIA=#FF00FF
SILVER=#C0C0C0	AQUA=#00FFFF

Рассмотрим пример HTML документа, в котором вышеперечисленные атрибуты используются для определения внешнего вида элементов тела документа.

Пример 2. ***Формирование тела документа***

```
<HTML>
<HEAD>
<TITLE>Атрибуты тела документа</TITLE>
</HEAD>
<BODY BGCOLOR ="SILVER", TEXT="BLACK",
LINK="BLUE", ALINK="RED", VLINK="NAVY">
<H1>Наш документ такой - как мы задумали !</H1>
<HR>
<H2>Обратите внимание на:</H2>
<P>Цветовое выделение элемента-якоря ссылки в документе
<P>Цвет текста документа
<P>Цвет фона документа</P>
<HR>
<P ALIGN=CENTER><A HREF="mailto:myname@mail.ru">Попробуй связаться со мной</A>
</BODY>
</HTML>
```

Не будем разбирать, что означают неизвестные нам пока теги (просмотрев пример в окне браузера не трудно догадаться), но как задавать внешний вид элементов тела документа мы уже знаем.

3. Основы форматирования

3.1. Форматирование текста

Для форматирования текстовых данных в HTML применяются следующие элементы:

Элемент управления абзацами

<P ALIGN=CENTER/LEFT/RIGHT >...</P> - элемент нового абзаца, используется в формате одиночного тега или элемента. При

использовании в форме одиночного тега концом абзаца считается начало следующего т.е. следующий тег <P>. Атрибут ALIGN задает выравнивание элементов абзаца, значение по умолчанию LEFT

<P>...</P> или <P>	Этот абзац выравнивается по левому краю. И этот абзац тоже.
<P ALIGN=CENTER>	Этот абзац выравнивается по центру. И этот абзац тоже.
<P ALIGN=RIGHT>	Этот абзац выравнивается по правому краю. И этот абзац тоже.

Элемент управления переносом

,<NOBR>...</NOBR>, <WBR> - элементы управления разрывами и переносом строк в тексте документа. При разрыве строки межстрочный интервал не увеличивается.

 	Используется для указания места принудительного разрыва. Пример: <P>ФИО: Иванов С.С.</P> Будет выглядеть так: ФИО: Иванов С.С.
<NOBR>...</NOBR>	Используется для запрета разрыва текста, помещенного в данный элемент. Пример: <NOBR>Это лучше не разрывать</NOBR> при необходимости переноситься на новую строку целиком, а не так: Это лучше не разрывать
<WBR>	Используется для указания рекомендуемого места для разрыва строки. Может быть вложенным в элемент <NOBR>...</NOBR>. Пример: <NOBR>42301<WBR>810600000000001</NOBR> - номер счета заданный таким образом при помещении в поле уже своей ширины, разорвется

	после балансового счета: 42301 810600000000001

Элементы выделения структуры документа

<H1>...</H1>, ... ,<H6>...</H6> - элементные теги шестиуровневых заголовков документа. Имеют атрибут ALIGN (по умолчанию LEFT) для выравнивания заголовка.

<H1>...</H1>	**Заголовок 1 уровня**
<H2>...</H2>	**Заголовок 2 уровня**
<H3>...</H3>	**Заголовок 3 уровня**
<H4 ALIGN=LEFT>...</H4>	**Заголовок 4 уровня по левому краю**
<H5 ALIGN=CENTER >...</H5>	**Заголовок 5 уровня по центу**
<H6 ALIGN=RIGHT>...</H6>	**Заголовок 6 уровня по правому краю**

Элементы смыслового выделения текста.

Элементы для смыслового выделения заключенного в них текста на Web-страницах. Способ выделения зависит от типа используемого браузера, но главное назначение этих тегов передача читателям логики автора.

<CODE>...</CODE>	Компьютерный код - `Function Sum(a,b);`
<CITE>...</CITE>	Выделение цитат - *Цитата*
<KBD>...</KBD>	Клавиатурный шрифт - `Клавиатура`

<SAMP>...</SAMP>	Выделение примеров - `Пример`
<STRONG>...</STRONG>	Выделение важных фрагментов - **Важно**
<VAR>...</VAR>	Выделение имен (*i, j, k*) переменных
<DFN>...</DFN>	Выделение определений - *Определение*
<EM>...</EM>	Расставить акценты - *Акцент*
<BLOCKQUOTE>...</BLOCKQUOTE>	Выделение фрагмента текста в большом блоке текстовом блоке на странице. Вот фрагмент, который мы хотели выделить из текстового блока в документе. Таким образом выделенные фрагменты текста отображаются броузером.

Элементы стилистического выделения текста

Данная группа элементных тегов применяется для стилистического выделения элементов текста. Допускается любая комбинация ниже перечисленных тегов.

<B>...</B>	**Выделение полужирным шрифтом**
<I>...</I>	*Выделение курсивом*
<TT>...</TT>	`Выделение телетайпным шрифтом`

<U>...</U>	Выделение подчеркиванием
<STRIKE>...</STRIKE>	~~Выделение перечеркиванием~~
^{...}	Шрифт в верхнем индексе
_{...}	Шрифт в нижнем индексе
<SMALL>...</SMALL>	Мелкий шрифт
<BIG>...</BIG>	Крупный шрифт

Дополнительные элементы форматирования

<PRE>...</PRE> - элемент предварительного форматирования текста. Используется для размещения на Web-страницах предварительно отформатированных текстовых фрагментов с сохранением их формата. Внутри элемента можно использовать элементы абзаца, переноса строк, а так же элементы стилистического и логического выделения. Данный элемент в основном применяется для опубликования исходного кода программ, так как браузер своим форматированием может нарушить принятые синтаксические конструкции языка программирования.

<HR> - элемент вставки линии-разделителя. Применяется для визуального разделения текста, при помощи горизонтальных линий (не путайте с графическими изображениями в форме разделителей). При отображении линии-разделителя в документе, до и после нее, браузер добавляет разделение абзацев. Формат линии-разделителя задается при помощи следующих атрибутов:

ALIGN - выравнивание (LEFT / RIGHT / CENTER);

WIDTH - ширина линии (пиксели или проценты к ширине окна WIDTH=50%);

SIZE - высота линии (пиксели);

COLOR - цвет линии;

NOSHADE - отключить эффекты 3-х мерности;

Примеры тега <HR>:

<HR>

<HR ALIGN=LEFT SIZE=5 WIDTH=50% COLOR=RED>

Комментарии и специальные символы

Для добавления комментариев в HTML документы используется элемент <!-- ...-->. Например:

<!-- После праздников (на свежую голову), эту главу надо переработать 31/12/2000 -->.

Кроме комментариев в HTML документах можно использовать специальные символы. Для внедрения специального символа в документ применяется конструкция следующего формата: &ИМЯ СИМВОЛА. Специальные символы используются для отображения элементов математических символов (&divide это ÷, &frac34 это ¾), редких символов национальных алфавитов и общепринятых символов (&copy это ©, &reg это ®).

Например, для отображения на Web-странице HTML тегов необходимо использовать символы заменители угловых скобок (< это &lt) и (> это &gt). Еще один часто используемый при форматировании (например, создание пустых ячеек в таблицах) спецсимвол это неразрывный пробел .

Использование шрифтов в документах

При использовании различных шрифтов для оформления текста следует помнить, что у пользователя может не оказаться шрифта, использованного вами для создания документа. Если вы используете редкие или нестандартные шрифты, то браузер пользователя может не подобрать шрифт для корректного отображения документа.

Существуют технологии внедрения необходимых разработчику шрифтов в Web-страницы. У Microsoft это технология Embedded fonts, а у их конкурентов Netscape, это называется Dinamic fonts. Оба подхода примерно одинаковы, но форматы шрифтов (OpenType и TrueDoc), а так же утилиты для их создания, упаковки и внедрения в документы различаются.

Для определения шрифта текста в HTML документах применяется элемент <FONT>...</FONT> и одиночный тег <BASEFONT>.

Тег <BASEFONT> задает базовые параметры шрифта, общие для всего документа. Действие базовых установок может быть отменено атрибутами нового тега <BASEFONT>.

Элемент <FONT> применяется для изменения параметров шрифта отдельных элементов документа, которые необходимо отобразить шрифтом отличным от базового. Действие его атрибутов ограничивается фрагментом документа, заключенным в данный элемент, и он может быть вложенным по отношению к другим тегам форматирования текста.

Для задания характеристик шрифта в тегах <FONT>...</FONT> и <BASEFONT> используются следующие атрибуты:

FACE - Задает имя шрифта (или перечня шрифтов - по мере убывания предпочтения) на компьютере пользователя. В случае отсутствия текст отображается шрифтом, заданным по умолчанию в браузере пользователя. Например:

<FONT FACE="Arial">Пример Arial</FONT> - Пример Arial

SIZE - абсолютный или относительный размер шрифта. Относительный размер это размер шрифта относительно стиля Normal (SIZE=3) или размера заданного тегом <BASEFONT>. Минимальное абсолютное значение размера шрифта 1, максимальное 7. Например:

<FONT SIZE=4>4 абсолютный шрифт</FONT> - 4 абсолютный шрифт

<FONT SIZE=+1>4 относительный шрифт</FONT> - 4 относительный шрифт

COLOR - цвет шрифта. Например:

<FONT COLOR=RED>Красный шрифт</FONT> - Красный шрифт

<FONT COLOR=#FF0000>Красный шрифт</FONT> - Красный шрифт

Полученные в данном разделе навыки, по форматированию текста, закрепим конкретным примером:

Пример 3. ***Формирование текста***.

```
<HTML>
<HEAD>
<TITLE>Форматирование текстовых данных</TITLE>
</HEAD>
<BODY BGCOLOR ="WHITE", TEXT="BLACK", LINK="BLUE", ALINK="RED", VLINK="NAVY" >
<BASEFONT FACE="Times New Roman","Arial" SIZE=4>
<H1 ALIGN=CENTER>АНЕКДОТЫ</H1>
<HR SIZE=5 COLOR=BLACK>
<FONT SIZE=+2><U>Анекдот 1</U></FONT>
<P>
Программист едет в трамвае, читает книгу.<BR>
Старушка смотрит на программиста, смотрит на книгу, крестится <BR>
и в ужасе выбегает на следующей остановке.<BR>
Программист читал книгу <DFN>"Язык Ада"</DFN>.
</P>
```

```
<FONT SIZE=+2><U>Анекдот 2</U></FONT>
<P>
Программист ставит себе на тумбочку перед сном два стакана. <BR>
Один с водой - на случай, если захочет ночью пить.<BR>
А второй пустой - на случай, если не захочет.
</P>
<FONT SIZE=+2><U>Анекдот 3</U></FONT>
<P>
Программист заходит в лифт, нажимает клавишу с номером этажа<BR>
и мучительно ищет клавишу <KBD>"enter"</KBD>.
</P>
<HR SIZE=5 COLOR=BLACK>
<P ALIGN=CENTER>&copy 2001 Вебмастер
<A HREF="mailto:myname@mail.ru">Попробуй связаться со мной</A>
</BODY>
</HTML>
```

В заключении следует отметить, что теги управления шрифтами, в последних спецификациях HTML, объявлены выведенными из употребления. На смену данным тегам пришли свойства шрифтов (font-family, font-size, font-style, font-variant, font-weight) из листов стилей CSS.

Та же участь постигла и некоторые теги форматирования символов. Они заменены свойствами текста CSS (например тег <U>...</U> заменен свойством text-decoration:underline, а тег <STRIKE>...</STRIKE> заменен свойством text-decoration:line-through).

Вы можете продолжать использовать данные теги, но в современных проектах они поддерживаться не будут.

3.2. Использование графики

Использование графики в документах позволяет повысить привлекательность ваших Web-страниц, делает изложенный материал более доступным для восприятия.

Web-браузеры поддерживают множество графических форматов, но наиболее часто используются GIF и JPEG (некоторые форматы требуют установки дополнительных программных компонентов браузера).

Доминирующими графическими форматами в Internet являются GIF и JPEG. Оба формата обладают специфическими особенностями,

что и определяет область их применения. Формат GIF (поддержка 256 цветов, сжатие без потери качества, чересстрочный формат, анимация, "прозрачность"), широко применяется для создания различных элементов Web-страниц: органов управления (кнопки, иконки, баннеры), анимированных изображений и других быстро загружаемых изображений с низкой цветопередачей. Формат JPEG (поддержка 16,7 миллиона цветов, потери качества при сжатии, высокая контрастность) применяется для публикации высококонтрастных изображений, фотографического качества. Больший размер файлов и следовательно более медленная загрузка.

Вставка изображений в документ

Для вставки изображения в документ используется одиночный тег <IMG>. Местоположение изображения на странице и его выравнивание относительно текста задается следующими атрибутами:

SRC - URL изображения;

ALIGN - выравнивание текста относительно изображения (режимы с обтеканием текста: LEFT - изображение слева, текст обтекает справа / RIGHT-изображение справа, текст обтекает слева; режимы без обтекания текстом: TOP - по верхнему краю изображения / MIDDLE - по центру изображения / BOTTOM - по нижнему краю);

WIDTH - ширина изображения (пикселы);

HEIGHT - высота изображения (пикселы);

ALT - текстовое описание-альтернатива, для тех, кто отключил загрузку изображений;

BORDER - ширина рамки (по умолчанию BORDER=1);

HSPACE - пустое поле от изображения по горизонтали;

VSPACE - пустое поле от изображения по вертикали;

ISMAP - признак карты-ссылок (обработка сервером);

USEMAP - признак карты-ссылок (обработка клиентом);

Примеры тега <IMG>:

<IMG SRC="pic1.gif" ALIGN=MIDDLE>

<IMG SRC="pic2.jpg" HSPACE=20 VSPACE=20 ALT="Здесь изображение офиса нашей компании">

<IMG SRC="pic3.jpg" WIDTH=120 HEIGHT=160 ALIGN=LEFT BORDER=5>

Закрепим на примере использование графики в ваших документах:

Пример 4. ***Использование графики***.

<HTML>

```
<HEAD>
<TITLE>Все графическое: элемент-якорь, линия-разделитель, фон и содержимое</TITLE>
</HEAD>
<BODY BACKGROUND="bgp.gif", BGCOLOR ="WHITE", TEXT="BLACK", LINK="BLUE", ALINK="RED", VLINK="NAVY">
<H1 ALIGN=CENTER>Два вида обезьян</H1>
<P ALIGN=CENTER><IMG SRC="bar.gif" WIDTH="50%">
<P ALIGN=CENTER><IMG SRC="monkey.gif" ALT="GIF"> & <IMG SRC="monkey.jpg" WIDTH=182 HEIGHT=122 ALT="JPG">
<H2 ALIGN=CENTER>GIF обезьяна & JPEG обезьяна</H2>
<P ALIGN=CENTER>
GIF обезьяна похуже качеством, но зато живая.<br>
JPG обезьяна красивая, но глазами хлопать не умеет. <P>
<P ALIGN=CENTER><IMG SRC="bar.gif" WIDTH="50%">
<P ALIGN=CENTER>&copy 2001 <A HREF="mailto:myname@mail.ru">
<IMG SRC="mbox.gif" BORDER=0 ALT="[ почта ]"></A> Вебмастер.
</BODY>
</HTML>
```

Приведем несколько рекомендаций по использованию графики:

старайтесь указывать размер изображения и его текстовую альтернативу, т.к. в случае невозможности загрузить изображение или загрузки в браузер с отключенной графикой не нарушается структура документа (вместо графики будет прямоугольник заданного размера с текстовым описанием);

при использовании изображения в качестве элемента-якоря ссылки отключайте рамку изображения (BORDER=0), дабы не портить внешний вид документа;

при указании размеров изображения больших или меньших от оригинального размера браузер производит их масштабирование, что может нарушить качество изображения некоторых форматов;

3.3. Форматирование табличных данных

Таблицы являются важнейшим элементом HTML-документов, т.к. кроме богатых возможностей по представлению структурированных данных они широко применяются как средство дл создания "каркасов" Web-страниц.

Таблицы в HTML определяются при помощи элемента <TABLE>...</TABLE>, заключающего в себе другие элементы таблицы: название, заголовки ячеек и их содержимое. Атрибутами элемента <TABLE> задается формат линии-сетки и общие правила форматирования (общий формат действуют, если иной формат не задан атрибутами формата конкретных строк и ячеек).

Наименование таблицы определяется при помощи элемента <CAPTION>...</CAPTION>. Выравнивание наименования задается при помощи атрибута ALIGN, который может принимать значения TOP (сверху таблицы) и BOTTOM (снизу таблицы).

Данные в таблице организованы построчно, каждая новая строка таблицы задается тегом <TR>...</TR> (закрывающий тег элемента </TR> можно не использовать). Для каждой строки таблицы при помощи специальных атрибутов тега <TR> можно управлять общим форматированием составляющих строку ячеек.

Строки таблицы разбиваются на ячейки при помощи тегов ячеек-заголовков <TH>...</TH> и тегов ячеек-данных <TD>...</TD> (допускается форма записи без закрывающих тегов). Как и для строк таблицы при помощи специальных атрибутов тегов <TD> и <TH> можно управлять форматированием конкретных ячеек таблицы.

Структура таблицы:

<TABLE> - начало элемента таблицы

<CAPTION> название таблицы </CAPTION>

<TR> <TH> 1 заголовок </TH>…<TH> N заголовок </TH> </TR> - первая строка / шапка

<TR> <TD> ячейка 1/1 </TD>…<TD> ячейка N/1 </TD> </TR> - 1 строка

<TR> <TD> ячейка 1/i </TD>…<TD> ячейка N/i </TD> </TR> - i строка

<TR> <TD> ячейка 1/M </TD>…<TD> ячейка N/M </TD> </TR> - последняя строка

</TABLE> - конец элемента таблицы

Таким образом, простейшая таблица, без линий сетки, в HTML-документе определяется следующим образом:

```
<TABLE>
<CAPTION ALIGN=TOP>Список друзей и подруг</CAPTION>
<TR><TH>ФИО</TH><TH>Телефон</TH></TR>
<TR><TD>Иванов Иван Иваныч</TD><TD>35-35-35</TD></TR>
<TR><TD>Валина Валентина Валентиновна</TD><TD>46-46-46</TD></TR>
```

</TABLE>

А вот, что у нас получится в окне браузера:

Список друзей и подруг

ФИО	**Телефон**
Иванов Иван Иваныч	35-35-35
Валина Валентина Валентиновна	46-46-46

Управление выравниванием элементов таблиц

Для выравнивания элементов таблиц в тегах <TABLE>,<TH> и <TD> используется много различных атрибутов. Рассмотрим наиболее полезные из них:

Атрибут	Теги	**Описание**
ALIGN	<TABLE>	Общее горизонтальное выравнивание таблицы на странице - LEFT/RIGHT/CENTER
	<TR>	Общее выравнивание элементов строки - LEFT/RIGHT/CENTER/CHAR
	<TH>	Выравнивание заголовка - LEFT/RIGHT/CENTER/CHAR (*по умолчанию CENTER*)
	<TD>	Выравнивание данных в ячейке - LEFT/RIGHT/CENTER/CHAR (*по умолчанию LEFT*)
CHAR	ALIGN=CHAR	Задает символ-выравниватель, при использование атрибута выравнивания ALIGN=CHAR.

		Например: <TR ALIGN=CHAR CHAR=","><TD>1,35</TD></TR>
CHAROFF	ALIGN= CHAR	Задает отступ (*в % ширины ячейки или в пикселях*) относительно начала ячейки, символов заданных атрибутом CHAR. Например: <TR ALIGN=CHAR CHAR="," CHAROFF="10%"> <TD>1,35</TD><TD>-1,45</TD></TR>
VALIGN	<TABLE>	Общее вертикальное выравнивание элементов таблицы - BOTTOM/MIDDLE/TOP (*по умолчанию MIDDLE*).
	<TR>	Общее выравнивание элементов строки - BOTTOM/MIDDLE/TOP/BASELINE
	<TH>	Выравнивание заголовка - BOTTOM/MIDDLE/TOP
	<TD>	Выравнивание данных в ячейке - BOTTOM/MIDDLE/TOP
CELLPADDING	<TABLE>	Свободное пространство между содержимым ячеек и их границами
CELLSPACING	<TABLE>	Свободное пространство между границами смежных ячеек
WIDTH	<TABLE>	Ширина таблицы в пикселях или процентах ширины окна браузера.
	<TH>,<TD>	Ширина ячейки таблицы в пикселях или процентах от ширины таблицы.
HEIGHT	<TABLE>	Высота таблицы в пикселях или процентах ширины окна браузера

	<TH>,<TD>	Высота ячейки таблицы в пикселях или процентах от ширины таблицы.

Управление линиями сетки таблиц

Для управления линиями сетки таблиц используется атрибут BORDER элемента <TABLE>, который задает толщину рамки таблицы в пикселях (по умолчанию рамки нет, BORDER=0). В стандарте HTML 4, появились новые атрибуты таблиц, строк и ячеек: FRAME для более гибкого управления линиями сетки таблиц и RULES для создания дополнительных линий разделителей групп в таблицах.

Данные атрибуты могут принимать следующие значения:

Атрибут FRAME:

- VOID - без рамки;
- BOX - с рамкой;
- ABOVE - верхняя сторона рамки;
- BELOW - нижняя сторона рамки;
- LHS - левая сторона рамки;
- RHS - правая сторона рамки;
- VSIDES - вертикальные линии;
- HSIDES - горизонтальные линии;

Атрибут RULES:

- NONE - без разделителя групп;
- GROUPS - вертикальные и горизонтальные линии разделители групп;

Например:

<TABLE >...</TABLE> равнозначно <TABLE FRAME=VOID>...</TABLE>

<TABLE BORDER>...</TABLE> равнозначно <TABLE FRAME=BOX>...</TABLE>

Управление цветом элементов таблиц

Атрибуты управления цветом элементов таблиц появились в HTML как расширения стандарта, предлагаемые основными поставщиками Web-броузеров (Microsoft и Netscape).

Расширения управления цветом: BORDERCOLOR - цвет рамки и BGCOLOR - цвет фона, используются как атрибуты для тегов: <TABLE>, <TR>, <TH>, <TD>. Соответственно областью их действия может быть вся таблица, строка или отдельно взятая ячейка.

Например:

<TABLE BORDER BORDERCOLOR=RED BGCOLOR=YELLOW>...</TABLE> - таблица.

<TR BORDERCOLOR=RED BGCOLOR=YELLOW></TR> - строка.

<TD BORDERCOLOR=RED BGCOLOR=YELLOW></TD> - ячейка.

Объединение элементов таблиц.

Для создания сложных таблиц не обойтись без объединения строк и столбцов. Для объединения ячеек соседних строк и столбцов таблицы, в HTML используются атрибуты ROWSPAN и COLSPAN тегов <TH> и <TD>. Данные атрибуты задают количество объединяемых ячеек в строке (ROWSPAN=N) или столбце (COLSPAN=N). Рассмотрим использование данных атрибутов на примере таблицы.

HTML код таблицы:

```
<TABLE BORDER ALIGN=CENTER>
<TR ALIGN=CENTER>
<TH COLSPAN=3>ДРУЗЬЯ И ПОДРУГИ</TH>
</TR>
<TR ALIGN=CENTER>
<TD ROWSPAN=2>ДРУЗЬЯ</TD><TD>Коля</TD><TD>44-44-44</TD>
</TR>
<TR ALIGN=CENTER>
<TD>Вася</TD><TD>33-33-33</TD>
</TR>
<TR ALIGN=CENTER>
<TD ROWSPAN=2>ПОДРУГИ</TD><TD>Маша</TD><TD>11-11-11</TD>
</TR>
<TR ALIGN=CENTER>
<TD>Глаша</TD><TD>22-22-22</TD>
</TR>
</TABLE>
```

Внешний вид таблицы:

ДРУЗЬЯ И ПОДРУГИ

ДРУЗЬЯ	Коля	44-44-44
	Вася	33-33-33
ПОДРУГИ	Маша	11-11-11
	Глаша	22-22-22

В стандарте HTML 4 появились новые теги для группировки (не объединения, а именно группировки) строк и столбцов таблицы в группы с общими свойствами. Это теги <COLGROUP> и <COL> - группировка и описание свойств столбцов, <THEAD> - определение шапки таблицы, <TBODY> - определение группы тело-таблицы, <TFOOT> - определение нижней строки. Полезным атрибутом тегов <COLGROUP> и <COL> является атрибут SPAN=N, который распространяет свойства, заданные данными тегами на N-столбцов в группе.

Использование данных тегов существенно облегчает компоновку и форматирование таблиц. Схема их применения следующая (в данном примере применяется одиночная форма записи тегов <THEAD>, <TBODY> и <TFOOD>, но в случае использования одного из них, необходимо применять элементную форму):

```
<TABLE атрибуты таблицы>
<COLGROUP общие атрибуты 1-ой группы>
<COL SPAN=10 доп. атрибуты первых 10 столбцов таблицы>
<COL доп. атрибуты 11 столбца таблицы>
...
<COLGROUP общие атрибуты последней группы>
<COL доп. атрибуты j столбца таблицы>...<COL доп. атрибуты N
столбца таблицы>
<THEAD>
<TH><TH>1 заголовок</TH>...<TH>N заголовок</TH><TR>
<TBODY>
<TR><TD>1 столбец</TD>...<TD>N столбец</TD><TR>
...
<TR><TD>1 столбец</TD>...<TD>N столбец</TD><TR>
<TFOOT>
<TR><TD>1 итоговый столбец</TD>...<TD>N итоговый
столбец</TD><TR>
</TABLE>
```

Для демонстрации возможностей новых тегов рассмотрим два варианта, какой из них проще - вам решать, но результат получится один и тот же.

Старый подход

```
<table border align=center>
<tr>
<th width=80>тип
<th width=120>название
<th width=50>1998
<th width=50>1999
<th width=50>2000
</tr>
<tr align=center>
<td width=80>тип1
<td width=120>название1
<td width=50 align=right>1,2
<td width=50 align=right>1,3
<td width=50 align=right>1,4
</tr>
<tr align=center>
<td width=80>тип2
<td width=120>название2
<td width=50 align=right>2,2
<td width=50 align=right>2,3
<td width=50 align=right>2,4
</tr>
</table>
```

Новый подход

```
<table border align=center>
<col width=80 align=center>
<col width=120 align=center>
<col width=50 align=right span=3>
<tr>
<th>тип<th>название
<th align=center>1998<th
align=center>1999<th
align=center>2000
</tr>
<tr>
<td>тип1<td>название1
<td>1,2<td>1,3<td>1,4
</tr>
<tr>
<td>тип2<td>название2
<td>2,2<td>2,3<td>2,4
</tr>
</table>
```

А вот результат обоих наших деяний:

тип	название	1998	1999	2000
тип1	название1	1,2	1,3	1,4
тип2	название2	2,2	2,3	2,4

Для закрепления материала, рассмотрим пример HTML документа, использующего таблицы (обратите внимание, что размер ячеек, содержащих изображение, задан соответствующим размеру изображения).

Пример 5. ***Форматирование таблиц.***

```
<HTML>
<HEAD>
<TITLE>Использование таблиц в документах</TITLE>
</HEAD>
<BODY BGCOLOR="WHITE", TEXT="BLACK", LINK="BLUE",
ALINK="RED", VLINK="NAVY">
<H1 align=center>Список моих друзей и подруг</H1>
</P>
<TABLE BORDER ALIGN=CENTER VALIGN=MIDDLE
WIDTH="50%">
<TR BGCOLOR=#FAD503>
<TH>Категория</TH><TH>ФИО</TH><TH>Фото</TH><TH>Телефон
</TH>
</TR>
<TR ALIGN=CENTER>
<TD ROWSPAN=2
BGCOLOR=#05C9FA>ДРУЗЬЯ</TD><TD>Коликов Коля</TD>
<TD WIDTH=50 HEIGHT=50><IMG SRC="boy1.gif"></TD><TD>44-
44-44</TD>
</TR>
<TR ALIGN=CENTER>
<TD>Васюков Вася</TD>
<TD WIDTH=50 HEIGHT=50><IMG SRC="boy2.gif"></TD><TD>33-
33-33</TD>
</TR>
<TR ALIGN=CENTER>
<TD ROWSPAN=2
BGCOLOR=#F9ACDE>ПОДРУГИ</TD><TD>Машина Маша</TD>
<TD WIDTH=50 HEIGHT=50><IMG SRC="girl1.gif"></TD><TD>11-
11-11</TD>
</TR>
<TR ALIGN=CENTER>
<TD>Глашева Глаша</TD>
```

```
<TD WIDTH=50 HEIGHT=50><IMG SRC="girl2.gif"></TD><TD>22-
22-22</TD>
</TR>
</TABLE>
</P>
<HR>
<P ALIGN=CENTER>&copy 2001 Вебмастер
<A HREF="mailto:myname@mail.ru">Попробуй связаться со мной</A>
</BODY>
</HTML>
```

В заключении следует отметить, что таблицы очень удобный инструмент для компоновки и форматирования элементов Web-страниц. Использование таблиц без линий сетки позволяет жестко связать текстовые блоки документа с графикой и другими объектами.

Например, кнопки управления, которыми вы пользуетесь для навигации по моим страницам - это то же таблица (проверьте посмотрев источник HTML).

3.4. Списки и меню

В HTML существует несколько разновидностей списков и меню, для их определения используются следующие теги:

Упорядоченный список (Ordered list):

<OL> - начало элемента списка
<LH> - заголовок списка
<LI> - первый элемент списка
<LI> - i элемент списка
<LI> - последний элемент списка
</OL> - конец элемента списка

Способ отображения упорядоченного списка на Web-странице зависит от используемого браузера. По умолчанию список автоматически нумеруется с 1, допускается использование вложенных списков и внутренних тегов форматирования. Для изменения способа нумерации и отображения списка используются следующие атрибуты:

TYPE	Тип нумерации элементов. **A/a**-нумерация прописными/строчными буквами **I/i**- нумерация прописными/строчными римскими цифрами

	1 - нумерация числами с единицы, **N**-нумерация числами с N
COMPACT	Компактный список

Неупорядоченный список (Unordered list):

<UL> - начало элемента списка
<LH> - заголовок списка
<LI> - первый элемент списка
<LI> - i элемент списка
<LI> - последний элемент списка
</UL> - конец элемента списка

Способ отображения неупорядоченного списка на Web-странице зависит от используемого браузера. По умолчанию списки разных уровней выделяются разными отступами и маркерами. Допускается использование вложенных списков, внутренних тегов форматирования и других элементов. Для изменения способа отображения списка используются следующие атрибуты:

TYPE	Тип маркировки элементов.DISC/SQUARE/CIRCLE
COMPACT	Компактный список (сжатый формат)

Списки каталогов (Directory list):

<DIR> - начало элемента каталога
<LI> - первый элемент списка
<LI> - i элемент списка
<LI> - последний элемент списка
</DIR> - конец элемента каталога

Способ отображения списка каталога на Web-странице зависит от используемого браузера. По умолчанию каталоги разных уровней выделяются отступами и маркерами. Допускается использование вложенных списков каталогов и тегов форматирования. Для элементов списка действуют ограничения, накладываемые на длину имени файла.

Списки определений (Definition list):

<DL> - начало элемента определений
<DT> - термин 1
<DD> - определение термина 1

<DT> - термин N
<DD> - определение термина N
</DL> - конец элемента определений

Способ отображения списка терминов на Web-странице зависит от используемого броузера. По умолчанию списки терминов выделяются разными отступами. Допускается использование атрибута COMPACT, вложенных списков, тегов форматирования и других элементов.

Списки меню (Menu list):

<MENU> - начало элемента меню
<LI> - пункт 1
<LI> - пункт N
</MENU>- конец элемента меню

Способ отображения списка меню на Web-странице зависит от используемого браузера. По умолчанию списки меню выделяются отступами разных уровней меню, иногда выделяются маркерами. Допускается использование вложенных меню, тегов форматирования и других элементов.

Рассмотрим пример использования списков в HTML-документах, заодно освежим в памяти, что такое внутренние ссылки - закладки.

Пример 6. ***Списки и меню***

```
<HTML>
<HEAD>
<TITLE>Использование списков и меню в документах</TITLE>
</HEAD>
<BODY BGCOLOR ="WHITE", TEXT="BLACK", LINK="BLUE",
ALINK="RED", VLINK="NAVY">
<H1 ALIGN=CENTER>Списки и меню в HTML документах</H1>
<HR>
<A NAME=MENU>Главное меню<A/>
<MENU>
<LH><B>Виды списков в HTML</B>
<LI><A HREF="#OL">Упорядоченный список</A>
<LI><A HREF="#UL">Неупорядоченный список</A>
<LI><A HREF="#DIR">Список каталогов</A>
<LI><A HREF="#DL">Список определений</A>
</MENU>
<HR>
<A NAME=OL>Упорядоченный список<A/>
```

```
<OL TYPE=1>
<LH><B>Сотрудники отдела</B>
<LI>Иванов
<LI>Петров
<LI>Сидоров
<LI>Зайцев
</OL>
<A HREF="#MENU">Вернуться к меню</A>
<p><A NAME=UL>Неупорядоченный список<A/>
<UL TYPE=SQUARE>
<LH><B>Сотрудницы отдела</B>
<LI>Иванова
<LI>Петрова
<LI>Сидорова
<LI>Зайцева
</UL>
<A HREF="#MENU">Вернуться к меню</A>
<p><A NAME=DIR>Список каталогов<A/>
<DIR>
<LI>VSTUDIO
<DIR>
<LI>Project1
<LI>Project2
<LI>Project3
<LI>Project4
</DIR>
</DIR>
<A HREF="#MENU">Вернуться к меню</A>
<p><A NAME=DL>Список определений<A/>
<DL>
<DT>JavaScript
<DD>Язык разработки сценариев интерактивного управления для
Web-страниц, разработанный фирмой Netscape на основе языка Java
(Sun). Поддерживается всеми современными браузерами.
<DT>VBScript
<DD>Язык разработки сценариев интерактивного управления для
Web-страниц, разработанный фирмой Microsoft на основе языка
VBasic. Поддерживается Internet Explorer.
</DL>
<A HREF="#MENU">Вернуться к меню</A>
<HR>
```

```
<P ALIGN=CENTER>&copy 2001 Вебмастер
<A HREF="mailto:myname@mail.ru">Попробуй связаться со мной</A>
</BODY>
</HTML>
```

В заключении отметим, что теги <MENU> и <DIR> исключены из последней спецификации HTML, т.к. они по сути дублировали возможности тегов <UL>, <OL> и <DT>. Старайтесь избегать использования данных тегов, что бы ваши документы соответствовали последним стандартам HTML.

Лабораторная работа №2.
Разработка модели деятельности автоматизируемого процесса

Первая часть

1.Разработка и создание функциональной модели IDEF0

Описание системы с помощью IDEF0 называется моделью. В IDEF0-моделях используются как естественный, так и графический языки.

Одна и та же схема моделирования может быть использована для моделирования любого выбранного объекта. Универсальной единицей для неограниченного строго структурного анализа является блок (рис. 1.1):

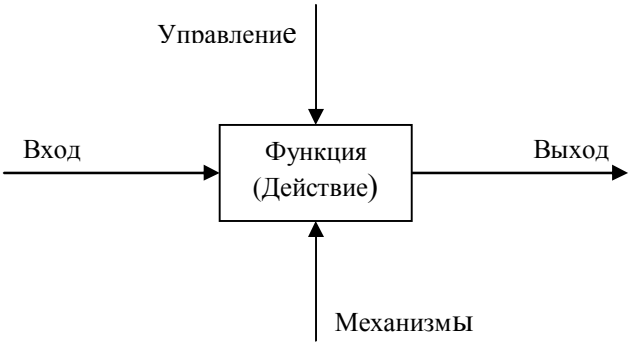

Рис. 1.1. Пример блока

Вход при наличии **управления** преобразуется в **выход** с помощью **механизма**.

Выходы одного блока могут быть входами или управлениями (или механизмами) для других блоков. Блоки именуются, а дуги помечаются с использованием естественного языка. Дуги могут разветвляться и соединяться, а каждый блок может быть подвергнут декомпозиции, т. е. разделен как целое на свои составляющие на более детальной диаграмме. Входы, управления и выходы определяют интерфейсы между блоками, а механизмы позволяют при необходимости в определенной степени объединять объекты. Границы блоков и диаграмм должны быть согласованы, а возникающая иерархически взаимосвязанная совокупность диаграмм является моделью.

Диаграмма ограничивается 3-6 блоками. Вместо одной громоздкой модели используется несколько небольших взаимосвязанных моделей, значения которых взаимно дополняют друг друга, делая понятной структуризацию сложного объекта.

2. Создание функциональных моделей и диаграмм

2.1. Начало моделирования

Начало моделирования означает создание диаграмм A0 и A-0, которые полностью рассказывают все об изучаемой системе с минимальной степенью детализации. Декомпозиция (диаграмма A0) освещает наиболее важные функции и объекты системы. Объединение (диаграмма A-0) трактует систему как «черный ящик», дает ей название и определяет наиболее важные входы, управления, выходы и механизмы.

Декомпозируя объект нужно, прежде всего, обратить внимание на входные и выходные данные для всей системы. Декомпозиция всей системы начинается с составления списка основных типов данных и основных функций. Потом эти списки снабжаются комментариями для указания основных типов, как данных, так и функций системы или их различных сочетаний. Наконец, списки с комментариями используются для создания диаграммы A0, которая затем обобщается с помощью диаграммы A-0.

Цель и точка зрения модели определяются на самой ранней стадии создания модели. Выбор цели осуществляется с учетом вопросов, на которые должна ответить модель, а выбор точки зрения – в соответствии с выбором позиции, с которой описывается система. Иногда цель и точку зрения можно выбрать до того, как будет сделана первая диаграмма. Например, цель модели разработки нового программного продукта можно определить заранее, потому что она очевидна в постановке задачи: «понять обязанности всех задействованных лиц так, чтобы организовать процесс разработки программного обеспечения». Настоятельно рекомендуется, как можно раньше определять цель и выбирать точку зрения новой модели. Но вначале попробуйте сформулировать ряд специфических вопросов, на которые модель должна ответить, чтобы убедиться, что цель сформулирована точно, и рассмотрите систему с нескольких различных точек зрения, прежде чем выбрать одну из них.

Иногда оказывается, что определить цель и точку зрения в самом начале моделирования чрезвычайно трудно. В таком случае следует составить списки данных и функций и, может быть, нарисовать диаграмму А-1, прообраз диаграммы А0. Сделав это, вы начнете чувствовать систему и установите, описывает ли ее диаграмма А0 с нужной точки зрения. Может быть, вам придется нарисовать несколько альтернативных А0-диаграмм, прежде чем появится достаточная уверенность для того, чтобы осуществить выбор правильной цели и точки зрения.

Списки объектов системы, создаваемые в ходе моделирования, в IDEF0 принято называть ***списками данных***. Термин «данное» здесь употребляется как синоним слова «объект». Составление списка данных является начальным этапом создания каждой диаграммы функциональной IDEF0-модели. Правило заключается в том, чтобы вначале составить список данных, а потом список функций. IDEF0-диаграммы представляют границы функций и ограничения, накладываемые на них, причем ограничения должны присутствовать во всех системах. Указывая вначале ограничения, выявляем естественную структуру системы. Без ограничений функциональная IDEF0-диаграмма представляет собой не более чем схему потоков данных. Без ограничительных дуг диаграммы не смогут рассказать читателю, почему аналитик выбрал именно данную декомпозицию. Благодаря тому, что в IDEF0 различаются входные дуги и дуги управления (информация, необходимая для пояснения декомпозиции), IDEF0-диаграммы ясно объясняют изучаемую систему и причину такой декомпозиции.

Закончив список данных, приступайте с его помощью к составлению ***списка функций***. Для этого представьте себе функции системы, использующие тот или иной класс (тип) или набор данных. Помните, что несколько различных типов данных может использоваться одной функцией. Обозначьте, какие типы или наборы данных необходимы для каждой конкретной функции. Это позволит выделить данные сходных типов, которые затем можно объединить в метатипы. Список функций должен находиться на одной странице со списком данных. При составлении исходного списка не пытайтесь объединять функции между собой. Вместо этого постарайтесь вначале сосредоточиться на каждой конкретной функции и ее отношении к группам данных. Кроме того, старайтесь подбирать такие функции, которые могли бы работать с наиболее общими типами данных из вашего списка.

Затем объединяйте функции в «агрегаты». Стремитесь к организации 3-6 функциональных группировок. Старайтесь, чтобы эти группировки имели один и тот же уровень сложности, содержали примерно одинаковый объем функциональности и функции в каждой из них имели сходные операции и цели.

Исходное содержание диаграммы А0 обеспечивают списки данных и функций. Для правильного описания системы содержанию надо придать форму. В IDEF0 это делается посредством построения диаграммы. Начинающим авторам необходимо придерживаться определенного порядка: (1) расположите блоки на странице, (2) нарисуйте основные дуги, представляющие ограничения, (3) нарисуйте внешние дуги и (4) нарисуйте все оставшиеся дуги. Со временем накопленный опыт позволит вам отойти от этой процедуры и изображать блоки и дуги в соответствии с той идеей, которую вы хотите воплотить в диаграмме.

Правильное расположение блоков является самым важным этапом построения диаграммы. Блоки располагаются в соответствии с их доминированием (по степени важности или по порядку следования). Самый доминантный блок обычно располагается в верхнем левом углу, а наименее доминантный – в нижнем правом. Это приводит к расположению, при котором более доминантные блоки ограничивают менее доминантные, образуя ступенчатую схему. Доминирование имеет важнейшее значение для ясного представления процесса. Например, не имеет смысла говорить о контроле над выполнением задания до изготовления детали.

Затем изображают основные дуги, представляющие ограничения. Это является второй важной частью построения диаграммы А0. Они дают основание для разбиения объекта диаграммы на 3 системные функции, изображаемые блоками. Рисуя эти дуги, проверяйте, действительно ли каждая из них оказывает влияние, соответствующее декомпозиции объекта. Проследите по списку данных, не отсутствуют ли какие-то дуги, представляющие ограничения. Если это так, вы, возможно, захотите проверить правильность декомпозиции.

Основными дугами, представляющими ограничения, всегда являются внешние дуги, т.е. дуги, представляющие данные, поступающие из непосредственного окружения диаграммы.

Следующим шагом в построении диаграммы является размещение остальных внешних дуг и назначение им соответствующих ICOM-кодов. Таким образом, все данные, входящие в систему или выходящие из нее, оказываются учтенными на рисунке. ***Потеря внешней дуги – это ошибка интерфейса, одна из самых***

распространенных в системном анализе. Занимаясь декомпозицией объекта, можно забыть об интерфейсных данных, потому что очень легко сосредоточиться на деталях. Начиная с изображения всех внешних дуг, вы повысите точность диаграммы, включив все интерфейсные данные. И, наконец, нарисуйте все остальные дуги, отражающие детали работы системы в целом. Во-первых, нарисуйте оставшиеся ограничения, действующие между блоками. Во-вторых, нарисуйте основной поток данных. В-третьих, уточните обратные связи в потоках данных. В заключение изобразите все обратные связи, вызываемые ошибочными ситуациями.

Здесь следует обратиться к одному очень важному моменту моделирования. На практике оказывается невозможным нарисовать диаграмму сразу набело. Поэтому рекомендуется вначале делать набросок диаграммы, а потом перерисовывать диаграмму набело, чтобы уточнить свое понимание, прояснить ситуацию и создать описание, которое могут посмотреть другие.

Обобщение является последним важным шагом начального этапа моделирования. Вспомните, что для любой IDEF0-диаграммы есть родительская диаграмма, содержащая ее контекст, где под контекстом понимается блок с набором входных дуг, дуг управления и выходных дуг. Верхняя диаграмма модели (т.е. диаграмма A0) не составляет исключения. Контекстом для нее служит диаграмма A-0, представляющая собой обобщение всей модели. Диаграмма A-0 имеет несколько предназначений. Во-первых, она объявляет общую функцию всей системы. Во-вторых, она дает множество основных типов или наборов данных, которые использует или производит система. В-третьих, A-0-диаграмма указывает взаимоотношения между основными типами данных, проводя их разграничение. Таким образом, A-0-диаграмма представляет собой общий вид изучаемой системы.

При создании диаграммы A-0 используется информация, уже зафиксированная на диаграмме A0. Вначале в центре IDEF0-бланка рисуют один большой блок, название которого совпадает с названием диаграммы A0. В этот момент следует проверить, адекватно ли название отражает то, что делает система. Все внешние дуги диаграммы A0 изображаются на диаграмме A-0 входящими в соответствующую сторону блока. При этом проверьте, что название каждой дуги описывает то, чем обмениваются система и ее среда. После того как вы изобразите входные и выходные дуги, остановитесь на минуту, чтобы проверить точность описания потока данных. Нарисовав дуги управления, убедитесь, что именно они управляют

тем, как система преобразует входные данные в выходные. Наконец, напишите цель и точку зрения модели под основным блоком и сверьте их с тем, что представляется блоком и его дугами.
Построение диаграммы A-0 свидетельствует об окончании начального этапа моделирования. Несмотря на ограниченное число описанных деталей, диаграммы A-0 и A0 представляют законченную картину, потому что они отражают все основные входы, управления, выходы и функции системы. Общий вид системы, полученный с помощью диаграмм A-0 и A0, – основная цель аналитика на начальном этапе построения IDEF0-модели.

2.3. Продолжение моделирования

Начальный этап моделирования включает определение объекта, цели и точки зрения модели, ограничения, накладываемые на объект, построение диаграммы верхнего уровня и ее обобщение, составление списков данных и функций, объединение функций в блоки, формирование с использованием списка данных взаимоотношений между блоками. Продолжение моделирования основывается на тех же методах и выводит модель на следующий уровень детализации. Для этого требуется создать отдельную диаграмму для, возможно, каждого блока диаграммы верхнего уровня, затем построить диаграммы для всех блоков новых диаграмм, и так до тех пор, пока модель не будет описывать объект с нужной для достижения цели степенью детализации. Таким образом, продолжение моделирования является рекурсивным процессом.
Декомпозиция модели похожа на начальный этап моделирования, но проще его. При декомпозиции модели аналитик всегда находится в контексте, определенном блоком со своими дугами одной из диаграмм. Эта граница, называемая границей объекта, определена двумя способами. Во-первых, объект, цель и точка зрения каждой новой диаграммы уже определены на диаграмме A0. Во-вторых, каждый блок, декомпозируемый в новую диаграмму, уже является ограниченным объектом. Другими словами, он идентифицирует конкретную функцию и все данные, которые для нее требуются или ею порождаются. Строить диаграмму, исходя из этой информации, проще, потому что список данных создается на основе дуг, входящих в блок и выходящих из него, а также потому, что список функций подробно раскрывает название блока. Процесс декомпозиции ограниченного объекта состоит из следующих шагов: выбор блока диаграммы; рассмотрение объекта, определенного этим блоком; создание новой

диаграммы; выявление недостатков новой диаграммы; создание альтернативных декомпозиций; корректировка новой диаграммы; корректировка всех связанных с ней диаграмм.
Декомпозиция начинается с чтения диаграммы А0 и определения самого содержательного блока. Это такой блок, декомпозиция которого выявит многие аспекты диаграммы А0 и будет оказывать большое влияние на будущие декомпозиции других блоков этой диаграммы. Новая диаграмма строится аналогично диаграммам А0 и А-0. Блоки размещаются в соответствии с доминированием (т.е. согласно взаимным ограничениям блоков), затем создаются основные дуги, представляющие ограничения, потом внешние и, наконец, внутренние дуги.
Практика показывает, что существенная доля ошибок приходится на интерфейсы. Для IDEF0 интерфейсными являются места соединения диаграмм со своими родителями. Вот почему каждую декомпозицию необходимо аккуратно соединять со своим родителем. Дуги выражают связи между блоками. Их вычерчивают не для показа последовательности действий. Они отражают отношения между блоками, независящие от потенциального следования. Такой механизм приводит к реализации различных сценариев, активизируя блоки в различные моменты времени в зависимости от ситуации.

2.4. Проверка диаграммы автором

Построив диаграмму, попытайтесь самостоятельно выявить ее недостатки. Процесс авторской проверки дает новое направление работе – определение ее качества. На этапе декомпозиции возникает диаграмма, которая декомпозирует блок и его дуги. Аналитик пытается объяснить объект самому себе. Неудивительно, что результат может оказаться малодоступным для других. В работе, естественно, появляются жаргон и неявно подразумеваемые факты. При критической оценке аналитик абстрагируется от своей работы. Это позволяет взглянуть свежим взглядом на диаграмму с тем, чтобы информация, которую она несет, стала доступной не только ее автору, но и другим людям.
Вначале следует критически оценить блоки диаграммы. Определим функциональные аспекты диаграммы, задавая вопросы типа:

- Представляют ли блоки содержательную декомпозицию функции?
- Не выглядит ли диаграмма запутанной?
- Все ли блоки соответствуют точке зрения модели?

- Несут ли блоки достаточный объем новой информации?

Теперь зададим вопросы о связи диаграммы с ее родителем. При этом проверим, как диаграмма вписывается в модель.

- Все ли внешние дуги имеют ICOM-коды?
- Все ли ICOM-коды соединяют дуги с одним и тем же значением?

Вопросами о внутренних дугах обычно заканчивают поиск ошибок в диаграмме. Теперь, после разрешения основных вопросов, следует проанализировать детали диаграммы. Мы можем задать вопросы типа:

- Не слишком ли много внутренних дуг?
- Нет ли блоков без дуг управления?
- Нет ли блоков без выходных дуг?
- Все ли важные обратные связи отражены?

Корректировка новой диаграммы. Обычно, потратив время на вопросы к диаграмме, тестирование, выполнение альтернативных набросков, автор корректирует диаграмму. В ходе корректировки следите за правильным доминированием, выбором названий блоков, информативностью дуг и делайте пояснения. Помните, что теперь вы рисуете диаграмму, чтобы донести информацию в точном и понятном виде до читательской аудитории.

Для блоков обычно стараются выбрать содержательные названия. Однако в IDEF0 нет необходимости выражать все с помощью названий блоков, потому что о работе блока многое сообщают метки окружающих его дуг.

Рисуя дуги, старайтесь располагать их аккуратно, минимизируя число пересечений и максимизируя пространство между ними. Правильное графическое расположение вносит большой вклад в повышение наглядности и понятности диаграммы. Помечайте дуги ясно и точно. Хотя определенное количество слов передает информацию лучше. Вычерчивая дуги в порядке их значимости, вы сможете оценивать их в процессе рисования и избежите стремления механически присоединять все дуги ко всем блокам.

Закончив построение диаграммы, поясните ее важные аспекты с помощью замечаний или дополнительного материала. Проясняйте только те понятия, которые нельзя изобразить в виде блоков и дуг.

Исправление взаимосвязанных диаграмм. Создание диаграммы, ответы на связанные с ней вопросы и переделка ее обеспечивают более глубокое понимание родительской диаграммы и диаграмм – потомков вновь построенной диаграммы. Зафиксируйте свое понимание во время исправления диаграммы. Вам придется

переносить информацию на другие диаграммы в трех ситуациях: при изменении меток внешних дуг, при появлении новых внешних дуг и при перераспределении функций. Перенесение необходимо, если изменилось название внешней дуги. Перенесение измененных меток внешних дуг немедленно обеспечивает предоставление родительской диаграммой всех данных, необходимых диаграмме-потомку. Перенесение необходимо также, когда на новой диаграмме появляются новые входные или выходные дуги. Эти новые дуги должны, так или иначе, возникнуть на родительской диаграмме. Есть два пути сделать это: нарисовать новые дуги на родительской диаграмме или объединить дуги новой диаграммы в одну и изменить соответствующим образом метку дуги на родительской диаграмме. Делая это, соблюдайте правила соединения и разветвления дуг. Перемещение блоков представляет самую сложную ситуацию. Оно происходит, когда функция (обычно на низком уровне модели) должна появиться, но не появляется на диаграмме, которую вы рисуете, а появляется на другой диаграмме модели, или наоборот. Перенести такую функцию, представленную блоком и всеми его дугами, с одной диаграммы на другую – нелегкое дело. Обычно это приводит к большим изменениям в метках дуг, появлению множества новых и исчезновению некоторых старых дуг. Иногда перемещение одного блока ведет к перемещению других блоков на различные диаграммы, вызывая целую волну изменений. Как правило, перемещение блока влечет за собой обилие технически сложной работы и может привести к ошибкам, если изменения не отслеживаются достаточно тщательно.

2.6. Завершение моделирования

Одна из наиболее частых проблем, возникающих в процессе реализации IDEF0-проектов, – когда же следует завершить построение конкретной модели? На этот вопрос не всегда легко ответить, хотя существуют некоторые эвристики для определения разумной степени полноты. В этом параграфе представлены правила, которыми пользуются опытные IDEF0-авторы для определения момента завершения моделирования. Однако хотелось бы предупредить, что приведенные здесь правила носят характер рекомендаций. Иногда даже опытные IDEF0-авторы, применив эти правила, обнаруживают в следствии, что приняли неверное решение. Только длительная практика позволит вам приобрести знания, необходимые для принятия правильного решения об окончании моделирования.

Прекращение декомпозиции. Как правило, большинством IDEF0-авторов рекомендуется прекращать моделирование, когда уровень детализации модели удовлетворяет ее цель. Другими словами, вы должны закончить моделирование, когда почувствуете, что дальнейшее продвижение не будет удовлетворять информационные потребности проекта или вступит с ними в противоречие. Хотя интуитивно это правило понятно, ему трудно следовать, не оценив модель. В первое десятилетие использования IDEF0 для создания моделей в различных прикладных областях были разработаны некоторые критерии для определения момента завершения моделирования. Этот опыт показал, что для отдельной модели, которая создается независимо от какой-либо другой модели, декомпозиция одного из ее блоков должна прекращаться, если:

1. блок содержит достаточно деталей;
2. необходимо изменить уровень абстракции, чтобы достичь большей детализации, блока;
3. необходимо изменить точку зрения, чтобы детализировать блок;
4. блок очень похож на другой блок той же модели;
5. блок очень похож на блок другой модели;
6. блок представляет тривиальную функцию.

Эти правила подчеркивают практические аспекты применения IDEF0 для описания систем реального мира с конкретной целью (например, понять работу телефонной станции, чтобы определить требования к ее программному обеспечению).

Одна из типичных ситуаций, встречающихся в конце моделирования – это блок, который описывает систему с нужным уровнем подробности. Проверить достаточность деталей обычно совсем легко. Просто спросите себя, отвечает ли блок на все или на часть вопросов, составляющих цель модели. Если блок помогает ответить на один или более вопросов, то дальнейшая декомпозиция может не понадобиться.

Принятие решения о завершении моделирования. Вероятность принятия неправильного решения о завершении моделирования может быть уменьшена, если вы оцените каждый блок, который хотите детализировать, в соответствии с приведенными выше правилами. Поскольку большинство IDEF0-моделей обычно содержат от 10 до 320 диаграмм, лучшее время для начала оценки блоков, когда модель достигает этих размеров. Если какая-то часть модели достигла уровня, не требующего дальнейшей декомпозиции, обращайтесь к своему собственному критерию для определения момента завершения моделирования, прежде чем декомпозировать блок, который еще не был детализирован.

Помните, что критерии, приведенные в этой главе, носят характер рекомендаций, следовательно, их нельзя применять механически. Иногда может случиться так, что в какой-то конкретной ситуации два правила будут противоречить друг Другу. Разрешение таких конфликтов требует рассудительности и осмотрительности. А это результат накопленного опыта. Кроме того, мы советуем вести записи по сложным вопросам, по тому, как они были решены, и что этому способствовало. Такие записи помогут вам впоследствии, если окажется, что моделирование следовало закончить раньше. Они помогут также при определении и выборе объектов будущего моделирования.

2.7. Дополнения к диаграммам и моделям

Одно из достоинств IDEF0-модели заключается в способе организации и представлении информации. Диаграмма, находящаяся на вершине модели, обобщает всю рассматриваемую систему. Диаграммы первого уровня представляют важнейшие подсистемы с их взаимосвязями, а диаграммы самого нижнего уровня представляют детализированные функции, с помощью которых, собственно, и работает система. Диаграммы законченной IDEF0-модели упорядочение организуют все важные компоненты и детали системы. Опытные аналитики, используя преимущества организации, создают различные дополнения к ней. Дополнения и уточнения, которые не входят в сами диаграммы, обогащают информационное содержание модели. Поскольку дополнительная информация формально не является частью модели, IDEF0 рекомендует помещать такие материалы на отдельных страницах и соединять их с диаграммами модели.

Дополнения к диаграммам. Квалифицированные IDEF0-аналитики придают конкретную направленность дополнительной информации, чтобы выделять некоторый аспект или часть отдельной диаграммы с помощью их дополнительного описания. Таким образом, они повышают отдачу текстовых записей и графики, причем для этого требуется лишь небольшое количество деталей (модель обеспечивает контекст, необходимый для связи дополнительной информации с системой).

IDEF0-диаграммы могут быть дополнены информацией в виде текстов, рисунков и глоссариев. Текст обычно представляет собой рассказ об одной из частей диаграммы. Рисунки – это картинки, поясняющие отдельные моменты. Глоссарий – набор определений объектов и функций, представленных на диаграмме.

Дополнительная информация записывается или представляется на стандартных IDEF0-бланках. Поскольку дополнения уточняют конкретную диаграмму модели, для идентификации и связывания дополнительной страницы с диаграммой, к которой она относится, используется принятая в IDEF0 схема нумерации узлов. К номеру узла диаграммы добавляется буква и целое число. Буква определяет тип дополнения (Т – текст, Р – рисунок и Г – глоссарий), а число означает порядковый номер этой текстовой страницы среди других дополнительных страниц данной диаграммы.

Дополнение моделей. Иерархические наборы IDEF0-диаграмм, называемые «моделями», вводят объект описания и уточняют его регулярным, управляемым и понятным образом. Это обеспечивается тем, что диаграммы модели всегда организованы в соответствии с порядком нумерации узлов. Это означает, что первым идет узел А-0, вторым – узел А0, третьим – узел А1, четвертым – узел А11 и т. д. Такой порядок расположения IDEF0-диаграмм является копией «древовидной» структуры, часто встречающейся в математике и информатике. Полная IDEF0-модель обычно читается двумя способами. Первый способ – обзорное чтение, когда читаются все диаграммы верхнего уровня, прежде чем перейти к диаграммам следующего. Это называется чтением «в ширину». Второй способ – детальное чтение, когда читают отдельную ветвь дерева вплоть до диаграмм самого нижнего уровня. Это называется чтением «в глубину».

Чтобы помочь читателям правильно двигаться по древовидному набору диаграмм, разработаны дополнительные средства. Примерами таких средств могут служить указатель диаграмм и указатель узлов. Они представляют собой составленные с отступами списки узлов или диаграмм (по типу оглавления), определяющих содержание модели. В указателе диаграмм перечисляются все диаграммы модели с приведенными (в отдельной строке) названием и номером узла каждой диаграммы. В указателе узлов перечисляются все блоки модели, причем в каждой отдельной строке записываются название и номер узла соответствующего блока. Таким образом, указатель узлов является просто более подробным, чем указатель диаграмм, списком.

3. Автоматизация построения модели

Пакет Design/IDEF (Meta Software Corp.) – графическая среда для проектирования и моделирования сложных систем широкого назначения, поддерживающая методологии описания и моделирования

системных функций (IDEFO/ IDEF0), структур и потоков данных в системе (IDEF1, IDEF1X, E-R) и поведения системы (IDEF/CPN). Рассмотрим более подробно основные возможности пакета Design/IDEF.

Представление графики. Design/IDEF имеет быструю и высококачественную графику, включающую создание стандартных и пользовательских объектов, выравнивание и манипулирование объектами, выбор атрибутов графических объектов и текста. Дополнительно в Design/IDEF реализованы возможности, требуемые для редактирования и моделирования данных: построение связывающих линий типа «резинка», маршрутизация и сглаживание дуг т.д.

Обеспечение непротиворечивости модели. Design/IDEF имеет встроенные возможности, дающие уверенность разработчику, что IDEF-модель будет точной, целостной и непротиворечивой на протяжении всего цикла ее создания. Например, при модификации текста, принадлежащему функциональному блоку или дуге в какой-то одной части модели, текст будет динамически скорректирован на всех страницах модели.

Поддержка Словаря Данных. Design/IDEF имеет встроенный Словарь Данных, который позволяет хранить информацию и создавать отчеты о функциях и потоках данных в IDEF-модели. Словарь дает возможность определять начальную информацию об объектах и предоставляет разнообразный набор функций сопровождения, восстановления и сохранения целостности файлов данных. Возможности словаря отличаются большой гибкостью и позволяют пользователю вводить неограниченное число параметров для каждого объекта. В сочетании с высококачественной печатью на лазерном принтере, это позволяет разработчику создавать документацию проекта, отвечающую самым высоким требованиям.

Генерация отчетов. Design/IDEF предоставляет возможность использовать пять видов отчетов для поддержки и анализа моделей:

- Отчет о контроле полноты модели;
- Отчет о функциях;
- Отчет о дугах;
- Отчет о ссылках;
- IDEF-отчет.

Все отчеты могут быть показаны на экране компьютера, отредактированы и распечатаны с помощью текстового редактора. Design/IDEF анализирует и отбирает данные для генерации текстового файла, содержащего информацию о диаграммах и Словаре.

Информация, содержащаяся в отчетах, может быть экспортирована для использования в других программах, таких как, например, электронные таблицы, настольные издательские системы и текстовые редакторы.

Организация коллективной работы. Design/IDEF поддерживает работу многочисленной группы разработчиков, создающих одновременно большую и сложную IDEF-модель. Подмодели легко интегрируются в одну большую модель.

Моделирование данных (IDEF1, IDEF1X и E-R-методологии). Design/IDEF дает также возможность создавать информационные модели, которые представляют как собственно данные, так и связи между ними в системе.

Информация, содержащаяся в IDEF-моделях, экспортируется в любую базу данных, а сами модели могут быть экспортированы в Design/CPN – пакет динамического моделирования и анализа сложных систем.

Как CASE-пакет по разработке программ много обеспечения Design/IDEF поддерживает первые стадии создания программного продукта:

- Формулировка требований и целей проекта – определение того, что проектируемая система будет делать.
- Разработка спецификаций – формализованное описание требований.
- Создание проекта – определение подсистем и взаимодействий между ними.
- Документирование проекта – создание базы данных проекта, текстуальное описание составных частей проекта.
- Анализ проекта – проверка проекта на полноту и непротиворечивость.

Результатом работы пакета Design/IDEF является проект программной системы, состоящий из двух частей:

1. Проекта функциональной структуры системы, содержащий иерархически связанные страницы с IDEF0-диаграммами и описывающий все модули (вплоть до элементарных функций) системы, их взаимосвязи, входные и выходные параметры.
2. Проекта информационной структуры системы – логической модели ее базы данных, – описывающей все структуры и взаимосвязи данных.

Оба проекта проверяются на полноту и непротиворечивость, сопровождаются базой данных проекта и документацией.

4. Варианты заданий на моделирование

Постройте функциональную модель одного из приведенных ниже процессов. Выполнение задания рекомендуется проводить согласно следующим этапам:

1. Сбор информации

Наиболее подходящая стратегия выполнения данного этапа – самому придумать описание моделируемого процесса (см. раздел 2.1.). Экспертом в данном случае будет выступать ваш преподаватель. Предложите список лиц и документов, которые на ваш взгляд могли бы служить источниками информации при моделировании реального процесса.

2. Начало моделирования

Создайте диаграммы А0 и А-0 и отрецензируйте их у преподавателя. Обратите внимание (см. раздел 2.2), что эти две диаграммы полностью рассказывают все о моделируемой системе с минимальной степенью детализации. Диаграмма А-0, часто называемая контекстной диаграммой, определяет все необходимые связи моделируемого процесса с окружающим миром.

В первую очередь создайте диаграмму А0 и обобщив ее создайте диаграмму А-0.

Особое внимание обратите на то, что каждая модель должна иметь определенную цель и создаваться с конкретной точки зрения. Выбор цели осуществляется с учетом вопросов, на которые должна ответить модель, а выбор точки зрения – в соответствии с выбором позиции, с которой описывается система.

Подготовьте список основных типов данных и функций, необходимый для дальнейшей декомпозиции системы. Начните заполнять словарь данных. Помните, что несколько различных типов данных могут использоваться одной функцией.

При создании диаграммы А0 строго следуйте рекомендациям раздела 2.2. Правильное расположение блоков является самым важным этапом построения диаграммы.

Построенные вами на данном этапе диаграммы А-0 и А0 должны представлять законченную картину, поскольку они отражают все основные входы, управления, выходы и функции системы.

3. Продолжение моделирования

Продолжение моделирования основывается на тех же методах, что и начальный этап и выводит модель на следующий уровень детализации. Создайте отдельную диаграмму для, возможно, каждого блока диаграммы верхнего уровня, затем постройте для всех блоков новых диаграмм и так до тех пор, пока модель не будет описывать объект с нужной для достижения вашей цели степенью детализации. Попытайтесь создать модель не менее чем с четырьмя уровнями детализации.
Не забывайте пополнять и корректировать словарь данных.

4. Завершение моделирования
Для определения момента завершения моделирования воспользуйтесь указаниями раздела 2.6.
По завершении моделирования подготовьте отчеты, предусмотренные программой Design/IDEF: отчет о функциях, отчет о дугах, отчет о ссылках и IDEF-отчет.

Список процессов для моделирования:

1. Питание семьи.
2. Изготовление нестандартной детали.
3. Ликвидация аварийной ситуации на нефтепроводе.
4. Ликвидация пожаров.
5. Производство молока.
6. Заселение в общежитие.
7. Издание книги.
8. Работа библиотеки.
9. Управление гостиницей.
10. Управление автобусным парком.
11. Обеспечение питания в столовой.
12. Продажа компьютеров.
13. Съемка кинокартин.
14. Организация телевизионных программ.
15. Организация грузоперевозками.
16. Организация учебного процесса.
17. Выращивание сельскохозяйственной продукции.
18. Посещение больницы.
19. Лечение больного.
20. Строительство здания.
21. Организация туристической поездки.
22. Проведение выборов.

Лабораторная работа №2.

Вторая часть.
«Разработка и создание концептуальной модели данных IDEF1x»

Целью данной лабораторной работы является получение и развитие навыков в построении концептуальных моделей данных, отвечающих методологии IDEF1x.
Подробно рассматриваются синтаксис и семантика модели IDEF1X. Особое внимание уделено процедуре практического моделирования. Описываются основные стадии создания модели и даются необходимые рекомендации. Приводится процедура сквозного анализа IDEF1X-модели.
Дается описание процесса создания модели в среде программы Design/IDEF.
Приводятся варианты процессов для самостоятельного моделирования.
В основу теоретической части положен стандарт методологии IDEF1x (Information modeling manual IDEF1-extended. ICAM Project Priority 6201. Subcontract #013-078846. USAF Prime Contract #F33615-80-C-5155).
IDEF1 применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы или среды;
IDEF1-методология создана компаниями Hughes Aircraft и D.Appleton Company (DACOM). Она опирается как на собственные разработки обеих компаний, так и на реляционную теорию Т.Кодда и диаграммы «сущности-отношения» П.Ченна.
В настоящем пособии рассматривается расширенная версия IDEF1 (называемая IDEF1X). Улучшение методологии заключается в повышении качества графического представления, развитии семантики и упрощении процедур разработки модели.

1. Моделирование данных

1.1. Цели моделирования данных

Логическая структура данных СУБД, иерархическая, сетевая или реляционная, не может полностью удовлетворять требованиям к концептуальному определению данных, поскольку она имеет ограниченные рамки и обуславливается стратегией реализации СУБД. Необходимость определения данных с концептуальной точки зрения привела к разработке методологии моделирования данных, основанной на семантике, то есть к трактовке данных в контексте их взаимосвязей с другими данными. Семантическая модель данных является

абстрактной схемой, доказывающей, как хранящиеся символы соотносятся с реальным миром. То есть такая модель должна быть верным отражением реального мира.

Семантическая модель данных может применяться в различных целях. Укажем важнейшие из них:

1. **Планирование ресурсов данных.** Предварительная модель данных помогает при выработке широкого взгляда на данные, необходимые для деятельности предприятия. Затем эта модель может быть исследована для построения совместно используемых ресурсов данных.
2. **Построение совместно используемых баз данных.** Полностью разработанная модель может применяться для представления данных независимо от их конкретного использования. Это представление может быть проверено пользователями и затем преобразовано в физический проект базы данных для любой из различных технологий СУБД. Помимо того, что разработанные базы данных будут непротиворечивыми и совместно используемыми, моделирование данных существенно сократит затраты на разработку.
3. **Оценка покупаемого программного обеспечения.** Модель данных, отражающая действительную инфраструктуру организации, позволяет оценить, насколько покупаемое программное обеспечение соответствует модели данных компании и не противоречит ли инфраструктура, налагаемая программным обеспечением, способу ведения дел компании.
4. **Объединение существующих, баз данных.** Определив содержание существующих баз данных через семантические модели данных можно получить интегрированное определение данных. Концептуальная схема может использоваться для управления обработкой запросов в среде распределенной базы данных. Проект «Поддержка информационных интегрированных систем» ВВС США (IISS) является экспериментальной разработкой, в которой демонстрируется применение технологий такого типа к неоднородной среде СУБД.

1.2. IDEF1X-подход

IDEF1X – это методология семантического моделирования данных. Она разработана с учетом следующих требований:

1. **Поддерживает разработку концептуальных схем.** Синтаксис IDEF1X поддерживает семантические конструкции, необходимые для разработки концептуальной схемы. Окончательная версия IDEF1X-модели обладает желаемыми характеристиками – непротиворечивостью, расширяемостью и адаптируемостью.
2. **Обеспечивает ясный язык.** IDEF1X имеет простую, ясную, непротиворечивую структуру и четкие семантические понятия. Синтаксис и семантика IDEF1X сравнительно легки для понимания, хотя и являются достаточно мощным средством.
3. **Проста для изучения.** Семантическое моделирование данных – новое понятие для многих пользователей IDEF1X. Проблема обучаемости этому языку является важным фактором. Язык рассчитан на понимание и использование как профессиональными бизнесменами и системными аналитиками, так и администраторами данных и разработчиками баз данных. Он может служить эффективным средством коммуникации в коллективах, состоящих из различных специалистов.
4. **Надежно проверена на практике.** IDEF1X базируется на многолетнем опыте предшествующих методологий и тщательно проверена как в проектах ВВС, так и в промышленности.
5. **Возможность автоматизации.** IDEF1X-диаграммы могут создаваться большим числом графических программных пакетов. ВВС США на основе концептуальной схемы разработали активный трехсхемный словарь для построения прикладных программ и обработки запросов в распределенной неоднородной среде. Существует также коммерческое программное обеспечение, поддерживающее детализацию, анализ и управление конфигурацией IDEF1X-моделей.

IDEF1X использует подход сущностей-отношений к семантическому моделированию данных. Исходная разработка IDEF1 заключалась в расширении понятий сущности-отношения по методу П. Ченна, объединенных с понятиями реляционной теории Т. Кодда. Кроме того, для улучшения графического представления и процедур моделирования IDEF1X-методология семантически обогащена введением отношений категоризации (называемых также

отношениями обобщения). Язык IDEF1X включает коммерческие разработки D.Appleton Company и The Database Design Group.

3. Процедуры моделирования.

3.1. Начало работы над проектом.

IDEF1X-модель данных должна быть описана и определена в терминах как ее ограничений так и целей.

Определение цели моделирования. Установление цели моделирования охватывает два аспекта:

- Определение направленности – утверждение охватываемых моделью вопросов, т.е. контекстуальных рамок.
- Определение области действия – утверждение функциональных границ модели.

Одним из основных вопросов, на который при установлении цели моделирования должен быть дан ответ, является вопрос о временном интервале для модели: будет ли это модель существующей ситуации (т.е. «как_есть» – модель) или модель того, что произойдет в результате задуманных изменений (т.е. «что_должно_быть» – модель). Формальное описание проблемной области IDEF1X-проекта может включать обзор, построение, модификацию и выработку одной или нескольких (функциональных) IDEF0-моделей. Обычно IDEF0-модель уже существует и может служить основой для описания проблемной области.

Хотя целью моделирования данных является установление объективной картины основной инфраструктуры данных всего предприятия, для каждой модели важно определить конкретную область действия. Это поможет выявить данные, представляющие особый интерес. Область действия может быть связана с типом пользователя (например, покупатель или проектировщик), деловой функцией (например, выпуск технического чертежа или планирование организации цеха), типом данных (например, параметры конфигурации продукции или финансовая информация). Утверждение области действия и направленности модели определяет цель моделирования. Приведем пример цели моделирования:

«*Целью данной модели является определение текущих данных, используемых начальником производственною подразделения для производства и проверки сложных авиационных деталей*».

Разработка плана моделирования. В плане моделирования указываются задания для выполнения и последовательность, в которой они должны выполняться. Задания распределяют в соответствии с общими задачами моделирования:

- Планирование проекта.
- Сбор данных.
- Определение сущностей.
- Определение отношений.
- Определение ключевых атрибутов.
- Заполнение неключевых атрибутов.
- Проверка правильности модели.
- Приемка модели.

План моделирования служит основой для распределения заданий, определения этапов и оценки расходов на моделирование.

3.2. Стадия определения сущностей

Целью данной стадии является выявление и определение сущностей, находящихся в пределах моделируемой проблемной области. Первым шагом в этом процессе является идентификация сущностей.

Идентификация сущностей. «Сущность» представляет в контексте IDEF1X-модели множество «предметов», обладающих связанными с ними данными. Здесь «предмет» может быть отдельной физической субстанцией, событием, состоянием, действием, идеей, понятием, точкой, местом и т.д. Элементы представляемого сущностью множества обладают общим набором атрибутов или характеристик. Например, все элементы множества служащих обладают номером служащего, фамилией и другими общими атрибутами. Отдельный элемент множества сущности называется экземпляром сущности. Например, служащий с именем Джерри и номером 789 является экземпляром сущности СЛУЖАЩИЙ. Сущности всегда именуются общими существительными в единственном числе. Они должны иметь атрибут (ключ), однозначно идентифицирующий каждый из их экземпляров.

Большинство сущностей могут быть прямо или косвенно определены на основе исходного материала, собранного на стадии 0. Если при моделировании расширяется или детализируется предшествующая модель данных, то соответствующие сущности должны быть выделены из прежней модели. Для предварительно не определенных сущностей разработчик должен выявить в списке имен исходного материала предметы, представляющие потенциально возможные сущности. Для простоты можно выбрать все существительные из этого списка. Например, такие термины, как деталь, транспортное средство, машина, чертеж и т.д., могут на этой стадии рассматриваться в качестве потенциальных сущностей.

Другой метод состоит в отборе терминов, перед которыми используется слово «код» или «номер» (например, номер детали, номер заказа на покупку, номер маршрута и т.д.). Предложение, Начинающееся словом «код» или «номер», может также рассматриваться на этой стадии в качестве потенциальной сущности. Что касается оставшихся слов в списке, то разработчик модели должен задать вопрос, представляет ли слово объект или предмет, о котором есть информация, или оно дает информацию о каком-то объекте или предмете. Те слова из списка, которые попадают в категорию объектов, о которых известна информация, потенциально являются сущностями.

Сущность образуется в результате объединения основных экземпляров сущности, становящихся элементами этой сущности. Это означает, что некоторое количество экземпляров сущности, у которых все характеристики однотипны, представляется в качестве сущности. Каждый экземпляр сущности является элементом сущности, обладающим однотипной определяющей информацией.

Для облегчения отделения сущностей от несущностей разработчик модели должен задать себе следующие вопросы, касающиеся каждой возможной сущности:

- Может ли она быть описана? (Обладает ли она характерными особенностями?)
- Существует ли более одного экземпляра этой сущности?
- Может ли один экземпляр быть отделен от другого (идентифицирован)?
- Называет или описывает это что-либо? (Из положительного ответа следует, что это скорее атрибут, чем сущность).

В конце такого анализа разработчик определяет начальный пул (накопитель) сущностей. Данный пул содержит все известные на данный момент имена сущностей в контексте модели.

При построении пула сущностей разработчик присваивает каждой записи свой идентифицирующий номер и записывает ссылку на ее источник. Таким образом, поддерживается возможность отслеживания информации. Целостность пула остается ненарушенной, а управление пула – сравнительно легким. Пример пула сущностей показан в таблице 3.1.

Таблица 3.1. Пул Сущностей

Номер	**Имя сущности**	**Номер источника**
E-1	Платежное требование	2

E-2	Расходный ордер	2
E-3	Приходный ордер	2
E-4	Реестр платежных поручений	3
E-5	Смета	6
E-6	Личная карточка сотрудника	4
E-7	Ведомость на выдачу зарплаты	8
E-8	Журнал учета больничных листов	8
E-9	Сотрудник	10
E-10	Квалификация сотрудника	10
E-11	Отдел	6
E-12	Филиал	6
E-13	Журнал дежурного	12
E-14	Счет	11
E-15	Рабочая карта	12
E-16	График мастера	14
E-17	Материалы	15
E-18	Доступность материалов	15
E-19	Оборудование для обработки материалов	15
E-20	Требования к материалам	15

По всей вероятности, не все имена в списке останутся сущностями к концу последней стадии. Кроме того, к этому списку добавится ряд новых сущностей, которые станут частью информационной модели по мере ее развития и улучшения понимания информации. Обнаруженные на последующих стадиях имена сущностей должны добавляться в пул сущностей и приобретать уникальные идентифицирующие номера. Одним из результатов деятельности на данной стадии является пул сущностей. Он должен обновляться, чтобы оставаться жизнеспособным.

Определение сущностей. Еще одним результатом данной стадии является начало работы над глоссарием сущностей. На протяжении этой стадии глоссарий представляет собой просто набор определений сущностей.

Определение сущности включает следующие компоненты:

1. ***Имя сущности*.** Имя сущности является уникальным именем, с помощью которого сущность будет распознаваться в IDEF1X-модели. Оно должно быть описательным. Хотя допускаются аббревиатуры и акронимы, имя сущности должно быть осмысленным.

2. *Определение сущности*. Это то определение сущности, которое обычно используется на предприятии. Оно не задумано как часть словаря. Поскольку смысл отражаемой в модели информации зависит от точки зрения модели и от определенного на начальной стадии контекста модели, то бессмысленно (а может быть, и вредно) включать сюда определения, не входящие в область действия на начальной стадии. Однако могут быть небольшие различия смысловых оттенков в способе определения сущности, зависящие, главным образом, от контекстуального использования. В этих случаях или при наличии альтернативных определений (которые могут не быть наиболее общеупотребительными с точки зрения модели) они должны быть также записаны. Рецензенты по своему усмотрению устанавливают, какое определение должно быть связано с термином, употребляемым для определения сущности. Процесс определения на стадии определения сущности является средством ускорения формирования общепринятого определения.

3. ***Синонимы сущности.*** Это список других имен, под которыми сущность может быть известна. Единственным относящимся к этому правилом является то, что определение, связанное с именем сущности, должно в точности применяться к каждому из синонимов в списке.

Определения сущностей формулировать легче, если начинать с сущностей, требующих наименьшего количества исследований. Таким образом, объем глоссария увеличится за кратчайшее время. Затем разработчик сможет проводить исследования для полного определения оставшихся имен в пуле. Четкое планирование времени и усилий на сбор и определение информации обеспечит разумный темп процесса моделирования.

3.3. Стадия определение отношений.

Целью стадии определения отношений является выявление и определение основных отношений между сущностями. На этой стадии моделирования некоторые отношения могут быть неспецифическими и потребуют дополнительной детализации на последующих стадиях. Главными результатами стадии являются:

- матрица отношений;
- определения отношений;
- диаграммы уровней сущностей.

Установление связанных сущностей. Отношение может быть определено просто как ассоциация или связь между двумя сущностями Точнее, это называется бинарным отношением. IDEF1X

ограничивается бинарными отношениями, поскольку исследовать и понимать их легче, чем n-арные отношения. Кроме того, они имеют непосредственное графическое представление. Недостатком является некоторое неудобство при представлении n-арных отношений. Но в этом нет ограничения общности, поскольку любое n-арное отношение может быть выражено через n бинарных отношений.

Экземпляр отношения – это имеющая смысл ассоциация или связь между двумя экземплярами сущностей. Например, экземпляр сущности ОПЕРАТОР, имя которого – Джон Дол, а номер оператора - 862, приписан к экземпляру сущности СТАНОК, тип которого – сверлильный станок, а номер станка – 12678. IDEF1X-отношение представляет множество однотипных образцов отношений между двумя специфическими сущностями. При этом одна и та же пара сущностей может обладать отношениями нескольких типов.

Целью IDEF1X является не изображение всех возможных отношений, а определение взаимосвязей между сущностями в терминах, отношений зависимости существования (отношений родитель-потомок). Такое отношение – это ассоциация между типом родительской сущности и типом сущности-потомка, при которой каждый экземпляр родительской сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров сущности-потомка, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром родительской сущности. Это означает, что существование сущности-потомка зависит от существования родительской сущности. Например, ПОКУПАТЕЛЬ делает ноль, один или несколько ЗАКАЗОВ_НА_ПОКУПКУ, а ЗАКАЗ_НА_ПОКУПКУ производится одним ПОКУПАТЕЛЕМ.

Если сущность-родитель и сущность-потомок представляют один и тот же объект реального мира, то родительская сущность является общей сущностью, а сущность-потомок является сущностью-категорией. Для каждого экземпляра сущности-категории всегда имеется один экземпляр общей сущности. Для каждого экземпляра общей сущности может существовать ноль или один экземпляр сущности-категории. Например, ШТАТНЫЙ_СЛУЖАЩИЙ является СЛУЖАЩИМ. СЛУЖАЩИЙ может быть или не быть ШТАТНЫМ_СЛУЖАЩИМ. Если несколько сущностей-категорий ассоциируются с общей сущностью в отношении категоризации, то только одна категория может соответствовать данному экземпляру общей сущности. Например, отношение категоризации может использоваться для представления того факта, что СЛУЖАЩИЙ может быть либо

ШТАТНЫМ_СЛУЖАЩИМ, либо СЛУЖАЩИМ_ПОЧАСОВИКОМ, но не тем и другим одновременно.

В начале разработки модели часто невозможно представить все отношения как отношения родитель-потомок или отношения категоризации. Поэтому неспецифические отношения на стадии 2 должны быть преобразованы в специфические. Неспецифические отношения имеют общую форму – «ноль, один или много – к – ноль, один или много». Существование любой сущности не зависит от существования другой.

Первым шагом на данной стадии является выявление отношении между элементами различных сущностей. Эта задача может потребовать разработки матрицы отношений, пример которой приведен в таблице 3.2. Матрица отношений – это двумерный массив, обладающий горизонтальной и вертикальной осями. Множество предопределенных факторов (в данном случае – все сущности) записывается вдоль одной из осей, а другое множество факторов (в данном случае – также все сущности) записывается вдоль другой оси. Для указания на возможное отношение между двумя сущностями в точке пересечения соответствующих осей помещается знак «V». В этот момент суть отношения не важна: достаточно того, что отношение может существовать.

Разработчики-новички обычно устанавливают чрезмерное количество отношений между сущностями. Помните, что целью в конечном итоге является определение модели в терминах отношений родитель-потомок. Избегайте косвенных отношений. Например, если ОТДЕЛ ответствен за один или несколько ПРОЕКТОВ, а каждый ПРОЕКТ инициирует одно или несколько ПРОЕКТНЫХ_ЗАДАНИЙ, то нет необходимости в отношении между ОТДЕЛОМ и ПРОЕКТНЫМ_ЗАДАНИЕМ, поскольку все ПРОЕКТНЫЕ_ЗАДАНИЯ связаны с ПРОЕКТОМ, а все ПРОЕКТЫ связаны с ОТДЕЛОМ.

Более опытные разработчики предпочитают наброски диаграммы уровней сущностей, а не составление матрицы отношений. Однако важно определять отношения в процессе их выявления.

Таблица 3.2. Матрица Сущность-Отношение

	Студент	Предмет	Лектор	Аудитория	Классн. занятия
Студент				V	
Предмет			V	V	
Лектор		V			V
Аудитория	V	V			V

Класс. занятия			V	V	

Определение отношений. Следующим шагом является определение выявленных отношений. Эти определения включают:

- указание зависимостей;
- имя отношения;
- комментарии к отношениям.

В ходе определения отношений некоторые из них могут отбрасываться, а новые добавляться.

Для установления зависимости отношение между двумя сущностями должно быть проверено в обоих направлениях. Это делается посредством определения мощности на каждом конце отношения. Для определения мощности предположите существование экземпляра одной из сущностей. Затем определите, сколько различных экземпляров второй сущности может быть связано с первой. Повторите анализ, поменяв сущности ролями.

Рассмотрим отношение между сущностями ГРУППА и СТУДЕНТ. Отдельный студент может быть записан в ноль, одну или много ГРУПП. Анализируя в другом направлении, видим, что отдельная группа может иметь ноль, одного или много студентов. Поэтому между сущностями ГРУППА и СТУДЕНТ существует отношение типа «многие ко многим» с мощностью «ноль, один или много» на каждом конце отношения. Это отношение является неспецифическим, так как на каждом конце отношения не существует мощности «ровно один». Такое неспецифическое отношение позже в процессе моделирования должно каким-то образом разрешиться.

В качестве другого примера возьмем отношение между сущностями ПОКУПАТЕЛЬ и ЗАКАЗ_НА_ПОКУПКУ. Отдельный ПОКУПАТЕЛЬ может сделать ноль, один или много ЗАКАЗОВ_НА_ПОКУПКУ. Отдельный ЗАКАЗ_НА_ПОКУПКУ всегда делается одним ПОКУПАТЕЛЕМ. Поэтому между сущностями ПОКУПАТЕЛЬ и ЗАКАЗ_НА_ПОКУПКУ существует отношение типа «один ко многим» с мощностью «один» на конце отношения у сущности ПОКУПАТЕЛЬ и с мощностью «ноль, один или много» на конце ЗАКАЗ_НА_ПОКУПКУ. (Здесь специфическое отношение, поскольку у конца ПОКУПАТЕЛЬ этого отношения имеется мощность «ровно один», т е. ПОКУПАТЕЛЬ является родительской сущностью для сущности ЗАКАЗ_НА_ПОКУПКУ.)

Установив зависимость отношения, разработчик должен выбрать имя и начать описание отношения. Имя отношения является кратким

выражением, обычно глаголом с союзом, присоединяющим вторую упомянутую сущность. Это выражение отражает смысл представляемого отношения. Часто имя отношения состоит из одного глагола, хотя наречия и предлоги также появляются в именах отношений. После выбора имени отношения разработчик должен иметь возможность, читая отношения, получать осмысленное предложение, определяющее или описывающее отношение между двумя сущностями.

При специфической форме отношения всегда имеются сущность-родитель и сущность-потомок; имя отношения интерпретируется сначала, со стороны сущности-родителя, а затем от сущности-потомка к сущности-родителю. Если между этими сущностями существует отношение категоризации, то отсюда следует, что обе сущности относятся к одному и тому же объекту реального мира и мощностью на конце сущности-потомка (или сущности-категории) всегда является «ноль или один». Имя отношения в таком случае опускается, поскольку имя «может быть» подразумевается. Например, СЛУЖАЩИЙ может быть ШТАТНЫМ_СЛУЖАЩИМ.

При неспецифической форме отношения существует два имени отношения, по одному для каждой сущности, разделенные знаком «/». В этом случае имена сущностей интерпретируются сверху вниз или слева направо (в зависимости от относительных положений сущностей на диаграмме), а затем в обратном направлении. Имена отношений должны быть осмысленными. Под их именами должна быть реальная основа. Полное значение, т.е. причина выбора разработчиком данного имени отношения, может быть отражено в определении отношения. Определение отношения – это текст, объясняющий смысл отношения. Правила для определения сущностей применяются и к определениям отношений.

Определения отношений должны быть:

- специфическими;
- краткими;
- осмысленными.

Например, если отношение «один к нулю или к одному» определено между такими двумя сущностями, как ОПЕРАТОР и РАБОЧАЯ_СТАНЦИЯ, то имя отношения может читаться «в настоящий момент назначен для обслуживания». Это отношение может сопровождаться следующим определением: «Каждый оператор на протяжении каждого рабочего дня может быть назначен для обслуживания нескольких рабочих станций, но это отношение указывает на ту, которую оператор обслуживает в данный момент».

Построение диаграмм уровней сущностей. После определения отношений разработчик может начать строить диаграммы уровней сущностей, изображая графически эти отношения. Пример диаграммы уровней сущностей приведен на рис. 3.1.

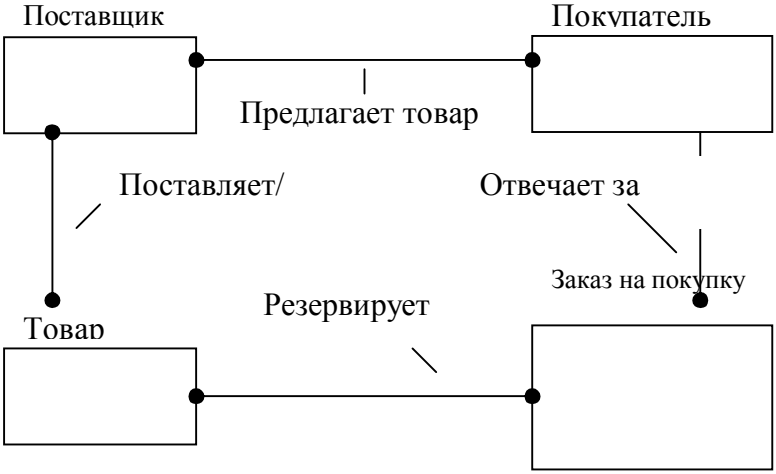

Рис. 3.1. Диаграмма уровней сущностей

На этой стадии моделирования все сущности представляются прямоугольными блоками и допускаются неспецифические отношения. Количество и направленность диаграмм уровней сущностей может меняться в зависимости от размеров модели и точки зрения отдельного наблюдателя. Для установления контекста и проверки непротиворечивости полезно, по возможности, изобразить все сущности и их отношения на единой диаграмме. Если создается несколько диаграмм, разработчик должен позаботиться, чтобы диаграммы не противоречили как определениям сущностей и отношений, так и друг другу. Совокупность диаграмм уровней сущностей должна изображать все определенные отношения.
В качестве частного случая диаграммы уровней сущностей можно выделить диаграмму, сосредоточенную на одной сущности и называемую просто диаграммой сущности. На рис. 3.2. приведен пример такой диаграммы.

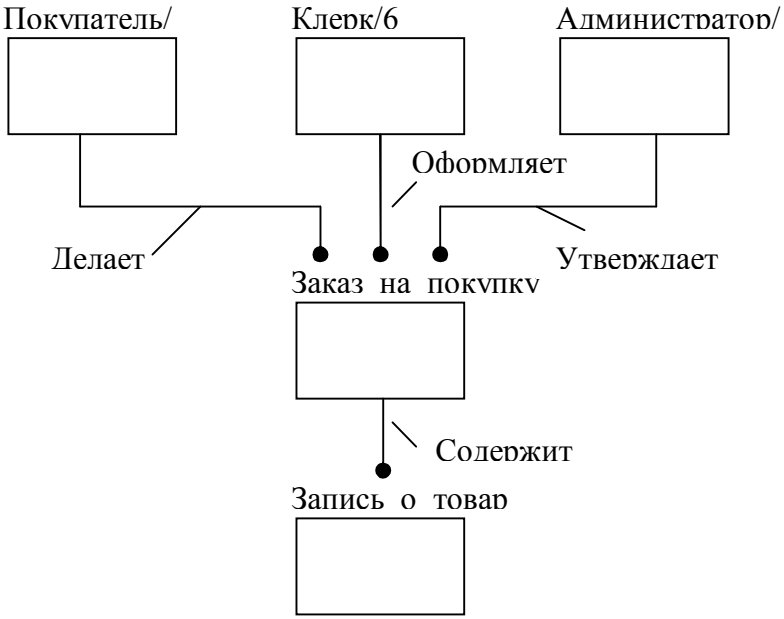

Рис. 3.2. Диаграмма сущностей

Создание отдельной диаграммы сущности для каждой сущности является допустимым, но при этом необходимо придерживаться следующих положений:

- Основная сущность должна располагаться приблизительно в центре страницы.
- Родительские или общие сущности должны размещаться выше основной сущности.
- Сущность-потомок или сущность-категория должны размещаться ниже основной сущности.
- Формы неспецифических отношений часто указываются сбоку от блока основной сущности.
- Линии отношений лучами расходятся от блока основной сущности к связанным сущностям. На диаграмме показываются только ассоциации между основной сущностью и связанными сущностями.
- Каждая линия, представляющая отношение, обладает меткой. В случае неспецифического отношения линия обладает двумя метками, разделенными знаком «/».

Доступная в данный момент информация о каждой сущности включает следующее:

- Определение сущности.
- Имена отношений и возможные определения (для отношений как с родителями, так и с потомками).
- Изображение на одной или более диаграмм уровней сущностей.

Информация о сущности может быть расширена с помощью добавления по усмотрению разработчика справочных диаграмм. К справочным диаграммам (диаграммам «для экспозиции только», иногда называемым FEO-диаграммами) разработчик может обращаться по желанию и вводить для них собственные соглашения. Эти диаграммы являются основой для дискуссий между разработчиком и рецензентами. Они дают разработчику уникальную возможность фиксировать логические обоснования, обсуждать проблемы, анализировать альтернативные варианты и рассматривать различные аспекты разработки модели.

3.4. Стадия определения ключей

Целями данной стадии являются:

- Детализация неспецифических отношений из стадии определения отношений.
- Определение ключевых атрибутов для каждой сущности.
- Перемещение первичных ключей для установления внешних ключей.
- Проверка правильности отношений и ключей.

Результаты стадии изображаются на одной или нескольких диаграммах (диаграммах ключевого уровня). Помимо определения ключевых атрибутов на данной стадии расширяются и детализируются определения сущностей и отношений.

Разрешение неспецифических отношений. Первым шагом на этой стадии является детализация всех неспецифических отношений, выявленных на стадии определения отношений. На стадии определения ключей требуется использовать только специфическую форму отношений: либо специфическое отношение связи (родитель-потомок), либо отношение категоризации. Чтобы выполнить это требование, разработчик предлагает варианты детализации. Диаграммы вариантов детализации обычно делятся на две части: левая часть посвящена субъекту (детализируемому неспецифическому отношению), а в правой части – вариант детализации. На рис. 3.3 показан вариант детализации, относящейся к разрешению отношений типа «многое ко многому».

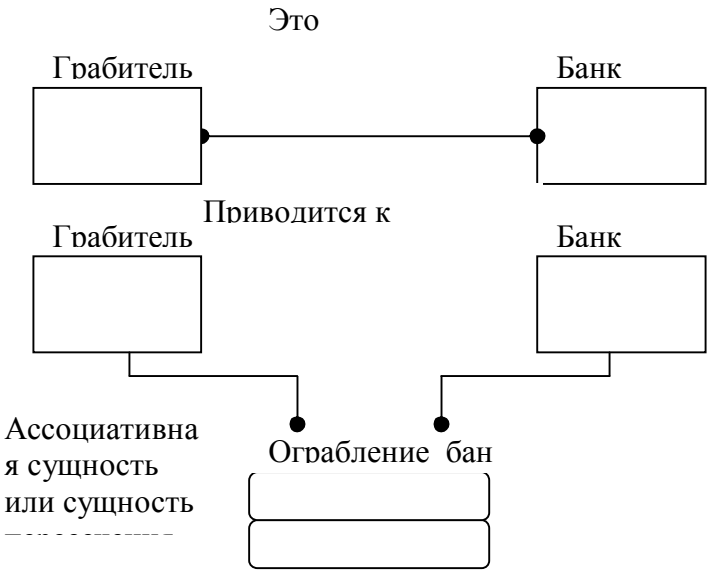

Рис. 3.3. Детализация не специфического отношения

Процесс детализации отношений приводит, или конвертирует, каждое неспецифическое отношение в два специфических отношения. В этом процессе возникают новые сущности. Неспецифическое отношение на рис. 3.3 указывает, что ГРАБИТЕЛЬ может ограбить много БАНКОВ, а БАНК может быть ограблен многими ГРАБИТЕЛЯМИ. Однако мы не можем определить, какой ГРАБИТЕЛЬ грабил какой БАНК, пока не введем для разрешения этого неспецифического отношения третью сущность: ОГРАБЛЕНИЕ_БАНКА. Каждый экземпляр сущности ОГРАБЛЕНИЕ_БАНКА связан с одним БАНКОМ и с одним ГРАБИТЕЛЕМ.

На более ранних стадиях мы имели дело с сущностями, которые могли бы неформально назвать естественными. Естественная сущность – это сущность, которую мы, вероятно, будем считать очевидной в списке исходных данных или протоколе исходных материалов. Естественная сущность будет включать имена, подобные следующим:

- Заказ на покупку.
- Служащий.
- Покупатель.

И только на стадии определения ключей начинают появляться ассоциативные сущности, которые могут быть неформально названы сущностями пересечения. Сущности пересечения используются для

разрешения неспецифических отношений и обычно представляют упорядоченные пары предметов с теми же основными характеристиками (уникальный идентификатор, атрибуты и т.д.), что и естественные сущности. Хотя в предыдущем примере сущность ОГРАБЛЕНИЕ-БАНКА могла бы рассматриваться как естественная сущность, она в действительности представляет объединение сущностей ГРАБИТЕЛИ и БАНКИ. Одно из небольших различий между естественной сущностью и сущностью пересечения состоит в именах сущностей. Именем естественной сущности служит обычно единичное нарицательное существительное; имена сущностей пересечения могут быть составными.

Сущности пересечения являются по своей природе более абстрактными и обычно появляются в результате впервые примененных на стадии определения ключей правил определения правильности сущностей. Первым из этих правил является правило, требующее детализации всех неспецифических отношений. Этот процесс детализации является первым главным этапом детализации объединенной структуры данных.

Процесс детализации включает:

- Разработку для каждого неспецифического отношения одного или нескольких вариантов детализации.
- Выбор разработчиком предпочтительного варианта, который и будет отражен в модели на данной стадии.
- Обновление информации стадии определения сущностей с целью включения возникших при детализации новых сущностей.
- Обновление информации стадии определения отношений с целью определения отношений, связанных с новыми сущностями.

Изображение функциональных точек зрения. К этому моменту объем и уровень сложности модели данных могут быть уже значительными. На стадии определения отношений было довольно естественно анализировать каждую сущность независимо от остальных. В такой ситуации сущности являются просто определениями слов. На стадии определения отношений возможно изобразить практически все отношения на одной диаграмме, поскольку общий объем сущностей и отношений не слишком велик. Однако на стадии определения ключей объем сущностей и сложность отношений обычно таковы, что человек не способен мысленно охватить в целом смысл модели. По этой причине модель может рассматриваться и проверяться с нескольких перспектив. Перспективы дают возможность исследовать модель способом, более непосредственно связанным с функциональными аспектами

моделируемой системы. Эти перспективы представляются функциональными точками зрения. Каждая функциональная точка зрения изображается на отдельной диаграмме с целью установления ограниченного контекста, в котором части модели могут быть исследованы за один прием.
Функциональные точки зрения полезны при исследовании и проверке правильности модели данных. Разработчик должен быть внимателен при выборе того, что будет иллюстрировать функциональная точка зрения. Для этого необходимо:

- Выбрать исходный материал в качестве предмета функциональной точки зрения (например, заказ на покупку).
- Связать функциональные точки зрения с категориями заданий или специфическими процессами, данные о которых представлены организационными отделами или функциональными областями, установленными на начальной стадии в качестве источников информации.

Определение ключевых, атрибутов. На стадии определения ключей методологии IDEF1X идентифицируются и определяются элементы данных об экземплярах сущностей, называемых возможными ключами, первичными ключами, альтернативными ключами и внешними ключами. Цель этого этапа – установить значения атрибутов, однозначно определяющих каждый экземпляр сущности. Важно подчеркнуть определение и смысл терминов «экземпляр атрибута» и «атрибут». Экземпляр атрибута является свойством или характеристикой экземпляра сущности. Экземпляры атрибутов составляются из имени и значения. Другими словами, экземпляр атрибута является элементом информации, известной об отдельном экземпляре сущности. Экземпляры атрибутов являются описателями, т е. они по сути скорее подобны прилагательным.
В таблице 3.3 приведены некоторые экземпляры атрибутов и их соответствующие экземпляры сущностей. Заметим, что первый экземпляр сущности (или индивидуум) идентифицируется номером служащего 1, ассоциированное с этим экземпляром сущности имя – Иванов, а профессия этого экземпляра сущности – оператор. Эти экземпляры атрибутов, взятые вместе, однозначно описывают экземпляр сущности и отделяют его от других аналогичных экземпляров сущностей. Каждый экземпляр атрибута обладает как типом, так и значением. Каждый конкретный экземпляр сущности описывает уникальную комбинацию экземпляров атрибутов.

Таблица 3.3. Примеры атрибутов

Экземпляры атрибутов					
Имя	Иванов	**Имя**	Петров	**Имя**	Сидоров
Номер	1	**Номер**	2	**Номер**	3
Должность	Оператор	**Должность**	Зав. Отд.	**Должность**	Пилот
Сущность – это множество, к которому принадлежит Иванов, Петров, Сидоров. В этом случае сущность получает имя СЛУЖАЩИЙ					
Атрибут – это признаки, которые описывают в общем виде характеристики сущности, например СЛУЖАЩИЙ. В данном случае каждого служащего описывают атрибуты: имя, номер, должность.					

Атрибут представляет множество экземпляров атрибутов одного типа, относящихся к разным экземплярам одной и той же сущности. Имена атрибутов обычно являются описательными существительными в единственном числе. В примере с сущностью СЛУЖАЩИЙ имеется несколько атрибутов:

- Номер служащего.
- Имя служащего.
- Профессия /должность служащего.

В таблице 3.3 показано, как экземпляры атрибутов представляются в качестве атрибутов. Экземпляры атрибутов принадлежат экземплярам сущностей. Но и сами атрибуты принадлежат сущности. Таким образом, между сущностью и некоторым числом атрибутов устанавливается ассоциация собственности.

У атрибутов есть только один владелец. Владелец – это сущность, которой атрибут принадлежит. В нашем примере владельцем атрибута НОМЕР-СЛУЖАЩЕГО будет сущность СЛУЖАЩИЙ. Хотя атрибут имеет только одного владельца, владелец может делить его с другими сущностями. Ниже будет детально рассмотрено, как это происходит.

Атрибут представляет использование экземпляра атрибута для описания отдельного специфического свойства отдельного экземпляра сущности. Кроме того, некоторые атрибуты представляют использование экземпляра атрибута для однозначного установления специфического экземпляра сущности. Эти атрибуты неформально называются ключевыми атрибутами.

На стадии определения ключей производится идентификация ключевых атрибутов в контексте нашей модели. На следующей стадии – определения атрибутов устанавливаются и определяются неключевые атрибуты.

Один или несколько атрибутов образуют возможный ключ сущности. Возможный ключ определяется как один или несколько ключевых атрибутов для однозначной идентификации каждого экземпляра сущности. Примером атрибута, используемого в качестве возможного ключа сущности, является номер служащего. Каждый служащий идентифицируется среди всех других служащих с помощью номера служащего. Поэтому атрибут НОМЕР-СЛУЖАЩЕГО является возможным ключом, который, однозначно определяет каждый элемент сущности СЛУЖАЩИЙ. Некоторые сущности обладают более чем одной группой атрибутов, которые могут применяться для различения одного экземпляра сущности от других. Рассмотрим сущность СЛУЖАЩИЙ с атрибутами НОМЕР-СЛУЖАЩЕГО и НОМЕР_СТРАХОВОГО_ПОЛИСА, каждый из которых сам по себе является возможным ключом. Для такой сущности выбирается один возможный ключ для использования в миграции ключей. Этот ключ называется первичным ключом, а остальные возможные ключи – альтернативными ключами. Если сущность обладает только одним возможным ключом, то он автоматически является первичным ключом. Таким образом, каждая сущность обладает первичным ключом, а некоторые сущности обладают также альтернативными ключами. Каждый тип возможных ключей может применяться для идентификации экземпляров сущностей, но только первичный ключ используется в миграции ключей.

На диаграмме модели в блоке основной сущности проводится горизонтальная линия, и внутри блока, выше этой линии, указывается первичный ключ. Если в первичном ключе имеется более одного атрибута (например, для идентификации проектных заданий требуются и номер проекта, и номер задания), то они все указываются выше горизонтальной линии. Если сущность обладает альтернативным ключом, то ему присваивается уникальный номер альтернативного ключа. На диаграмме этот номер указывается в скобках вслед за каждым атрибутом, являющимся частью этого альтернативного ключа. Если атрибут принадлежит нескольким альтернативным ключам, то каждый из номеров этих ключей указывается в скобках. Если атрибут принадлежит и альтернативному ключу, и первичному ключу, то он указывается выше горизонтальной линии вместе со следующим после него номером альтернативного ключа. Если атрибут не принадлежит первичному ключу, то он указывается ниже горизонтальной линии. На рис. 3.4 приведены различные формы ключей.

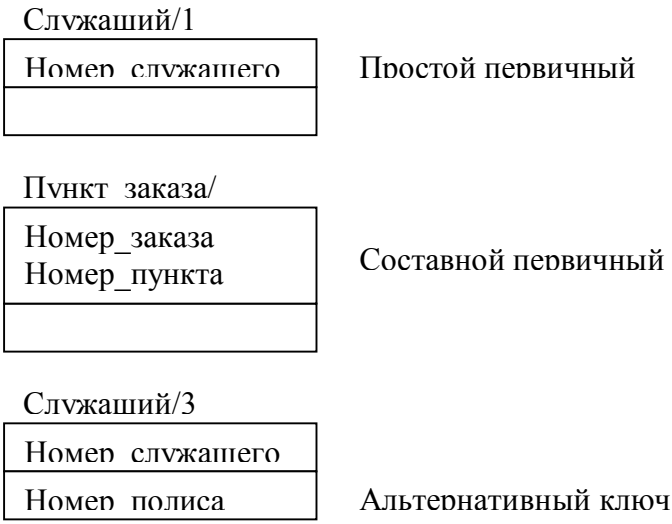

Рис. 3.4. Формы ключей

Процесс идентификации ключей включает:

- Идентификацию возможных ключей сущности
- Выбор одного из них в качестве первичного ключа сущности.

Поскольку некоторые возможные ключи могут возникнуть в результате миграции, идентификация ключей – итеративный процесс. Начинайте с тех сущностей, которые не являются ни в каком отношении сущностями-потомками или сущностями-категориями. Это обычно те сущности, чьи возможные ключи наиболее очевидны. Они являются также начальными точками для миграции ключей, поскольку не содержат внешних ключей.

Миграция ключей. Миграция ключей – это процесс копирования первичного ключа одной сущности в другую, связанную с ней сущность. Эта копия называется внешним ключом. Значение внешнего ключа в каждом экземпляре второй сущности совпадает со значением связанного экземпляра первой сущности. Таким образом, атрибут, принадлежащий одной сущности, разделяется с другой сущностью. Миграция ключей подчиняется следующим трем правилам:

- Миграция всегда происходит в отношении от родительской или общей сущности к сущности-потомку или сущности-категории.

- Весь первичный ключ (т.е. все атрибуты, являющиеся элементами первичного ключа) должен мигрировать по одному разу для каждого отношения, разделяемого парой сущностей.
- Альтернативный ключ и неключевые атрибуты никогда не мигрируют.

Каждый атрибут внешнего ключа соответствует атрибуту первичного ключа родительской или общей сущности. Первичный ключ сущности-категории в категориальном отношении должен совпадать с первичным ключом общей сущности. В других отношениях атрибут внешнего ключа может, но не обязан быть частью первичного ключа сущности-потомка. Атрибуты внешних ключей не считаются принадлежащими сущностям, в которых они появляются, поскольку они отражают атрибуты родительских сущностей. Таким образом, каждый атрибут в сущности либо принадлежит этой сущности, либо принадлежит внешнему ключу этой сущности.

В диаграммах модели внешние ключи обозначаются примерно так же, как альтернативные ключи, т.е. после каждого атрибута, принадлежащего внешнему ключу, следует (FK). Если атрибут принадлежит также первичному ключу, то он располагается выше горизонтальной линии, а если нет, то – ниже. Если первичный ключ сущности-потомка содержит все атрибуты внешнего ключа, то сущность-потомок называется зависимой от идентификатора относительно родительской сущности, а отношение называется идентифицирующим отношением. Если какие-либо атрибуты внешнего ключа не принадлежат первичному ключу сущности-потомка, то сущность-потомок не является независимой от идентификатора относительно родительской сущности, а отношение называется неидентифицирующим. На диаграммах IDEF1X сплошными линиями изображаются только идентифицирующие отношения, а неидентифицирующие отношения изображаются пунктирными линиями.

Сущность, являющаяся сущностью-потомком в одном или нескольких идентифицирующих отношениях, называется зависимой от идентификатора. Сущность, являющаяся сущностью-потомком только в неидентифицирующих отношениях (или не являющаяся сущностью-потомком ни в одном из отношений), называется независимой от идентификатора. В диаграммах блоками с прямыми углами изображаются только идентификаторно-независимые сущности, а идентификаторно-зависимые сущности изображаются блоками с закругленными углами.

Проверка правильности ключей и отношений. Идентификация и миграция ключей подчиняется следующим основным правилам:

- Нельзя использовать синтаксис неспецифических отношений.
- Миграция ключей от родительских (или общих) сущностей к сущностям-потомкам (или сущностям-категориям) является обязательной.
- Запрещается использовать атрибуты, которые могут принимать более одного значения для данного экземпляра сущности в одно и то же время (правило неповторяемости).
- Нельзя использовать атрибуты, обращающиеся в ноль (т.е. не принимающие никакого значения) для некоторого экземпляра сущности (правило необращения в ноль).
- Сущности с составными ключами не могут быть разбиты на несколько сущностей с более простыми ключами (правило наименьшего ключа).
- Необходимо объявлять об имеющихся между двумя сущностями двойных путях отношений.

В предыдущих разделах мы уже рассмотрели первые два правила, поэтому остановимся на оставшихся. На рис. 3.5 приведена диаграмма, относящаяся к применению правила неповторяемости. Обратите внимание, что субъект диаграммы содержит в качестве элементов первичного ключа сущности ЗАКАЗ атрибуты НОМЕР_ЗАКАЗА и НОМЕР_ПУНКТА_ЗАКАЗА. Однако, рассмотрев использование НОМЕРА_ПУНКТА_ЗАКАЗА, мы увидим, что один и тот же экземпляр сущности ЗАКАЗ может быть ассоциирован со многими НОМЕРАМИ_ПУНКТА_ЗАКАЗА, по одному на каждый заказываемый пункт. Для правильного отражения этого факта в модели данных должна быть создана новая сущность, называющаяся ПУНКТ_ЗАКАЗА с добавлением дуги и метки отношения, синтаксиса и определения. Таким образом, начинают выясняться истинные характеристики связи между заказами на покупку и пунктами заказов на покупку.

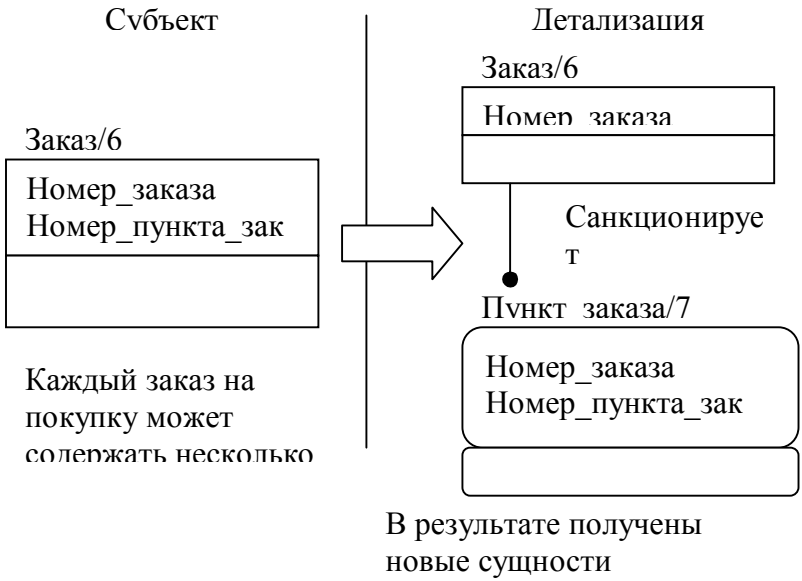

Рис. 3.5. Детализация правила неповторяемости

На рис. 3.6. приведена диаграмма вариантов детализации, относящаяся к применению правила необращения в ноль. Заметим, что НОМЕР_ДЕТАЛИ промигрировал к ПУНКТУ_ЗАКАЗА. Эта ассоциация установилась в связи с тем, что пункты заказа связаны некоторым образом с деталями. Однако показано, что диаграмма утверждает ассоциированность каждого пункта заказа на покупку в точности с одним номером детали. Исследование (или, возможно, комментарий рецензента) показывает, что не все пункты заказа на покупку ассоциируются с деталями. Некоторые из них могут быть связаны с услугами или другими товарами, не имеющими номеров деталей. Это запрещает миграцию НОМЕРА_ДЕТАЛИ непосредственно в сущность ПУНКТА_ЗАКАЗА и требует в нашем примере установления новой сущности ЗАКАЗАННАЯ_ДЕТАЛЬ.

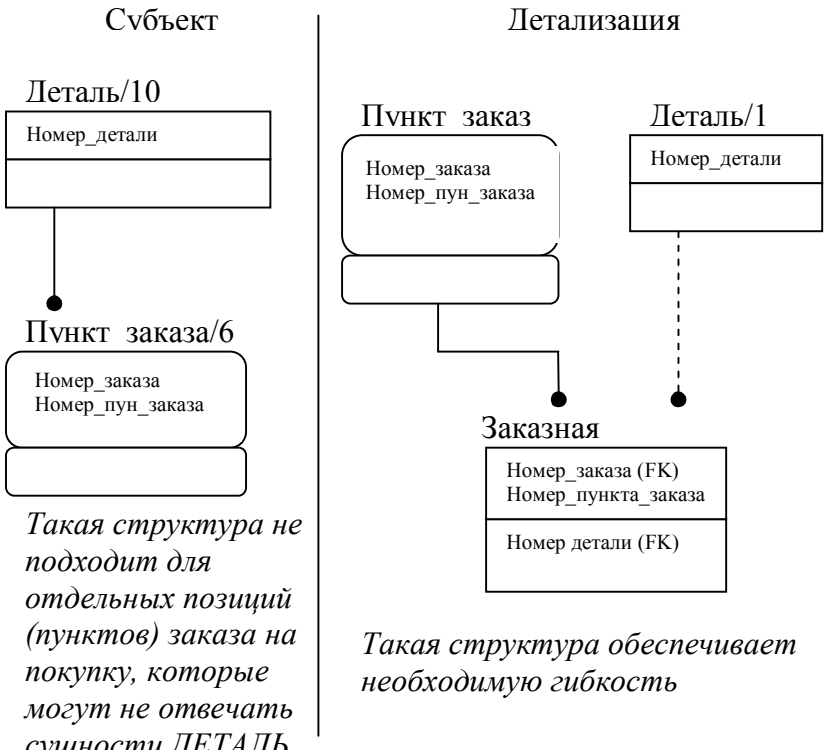

Рис. 3.6. Детализация правила необращения в ноль.

Как только новая сущность установлена, в соответствии с правилом миграции обязательно должна произойти миграция ключа, и разработчик будет снова проверять правильность соответствия структуры сущность-отношение правилам необращения в ноль и неповторяемости.

Каждый составной ключ должен проверяться на соответствие правилу наименьшего ключа. Это правило требует, чтобы любая сущность с составным ключом не могла разделяться на несколько сущностей с более простыми ключами (на меньшие компоненты) без потери некоторой информации. Это правило является комбинацией и расширением четвертой и пятой нормальных форм в реляционной теории. Другие правила нормализации, такие, как правило, полной функциональной зависимости или правило отсутствия транзитивной

зависимости, не могут применяться до тех пор, пока на стадии 4 неключевые атрибуты не будут отражены в модели.

В связи со стадией определения отношений упоминалась тенденция выявления избыточных отношений. Однако на той стадии анализ, в основном, проводится разработчиком по его усмотрению. Установив ключи, разработчик может быть более точным в анализе. Двойной путь отношений существует тогда, когда существует сущность-потомок с двумя отношениями, ведущими, в конечном итоге, обратно (через одно или несколько отношений) к общей «корневой» сущности-родителю. В случае существования двойных путей требуется утверждение пути, чтобы определить, являются ли пути равными, неравными или неопределенными. Пути равны, если для каждого экземпляра сущности-потомка оба пути отношений всегда ведут к одному и тому же экземпляру корневой родительской сущности. Пути являются неравными, если для каждого экземпляра сущности-потомка оба пути отношений всегда ведут к различным экземплярам корневой родительской сущности. Пути являются неопределенными, если они равны для некоторых экземпляров сущности-потомка и не равны для остальных экземпляров. Если один из путей состоит только из одного отношения и пути равны, то путь из одного отношения является излишним и должен быть удален.

Простейшим случаем отношения, имеющего двойственные пути, является отношение, в котором оба пути состоят из единственного отношения. Например, каждый экземпляр сущности СХЕМА_СБОРКИ может быть связан с двумя различными экземплярами сущности ДЕТАЛЬ, избыточности нет. В этом случае утверждение пути будет требовать, чтобы пути были неравны, поскольку ДЕТАЛЬ не может быть вмонтирована в себя.

Если один из путей содержит несколько отношений, а другой путь содержит одно отношение, то такая структура называется триадой. На рис. 3.7 приведен пример триады. В этом случае СЛУЖАЩИЙ связан с ОТДЕЛЕНИЯМИ как прямо, так и косвенно, через ОТДЕЛ. Если утверждение пути состоит в том, что ОТДЕЛЕНИЕ, которому принадлежит СЛУЖАЩИЙ, содержит ОТДЕЛ, которому принадлежит СЛУЖАЩИЙ, то отношение между сущностями ОТДЕЛЕНИЯ и СЛУЖАЩИЙ является избыточным и должно быть удалено. Заметим, что если некоторые, но не все, СЛУЖАЩИЕ могут в действительности принадлежать двум различным отделениям, то должна быть добавлена еще одна сущность, такая, как ВРЕМЕННЫЙ_СЛУЖАЩИЙ, чтобы НОМЕР_ОТДЕЛЕНИЯ

удовлетворял правилу необращения в ноль в качестве внешнего ключа сущности СЛУЖАЩИЙ.

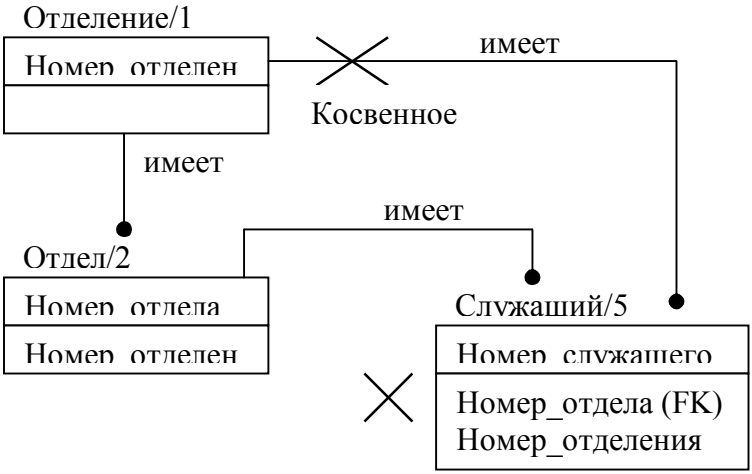

Рис. 3.7. Пример триады

Утверждения могут также применяться к отношениям двойственных путей, когда оба пути раскрывают более одного отношения. В приведенном на рис. 3.8 примере между сущностями ОТДЕЛ и ИСПОЛНИТЕЛЬ_ЗАДАНИЯ существуют два пути отношений. Если СЛУЖАЩИЙ может быть приписан только к тому ПРОЕКТУ, которым руководит его ОТДЕЛ, то пути равны. Если СЛУЖАЩИЙ может быть приписан только к тому ПРОЕКТУ, которым руководит не его отдел, то пути не равны. Если СЛУЖАЩИЙ может быть приписан к ПРОЕКТУ независимо от того, руководит его ОТДЕЛ ПРОЕКТОМ или нет, то пути являются неопределенными. Обычно, пути предполагаются неопределенными, если только утверждения точно не определены. Утверждения должны быть добавлены в качестве примечаний к диаграммам стадии определения ключей и включены в определение сущности-потомка.

После идентификации элементов первичного ключа производятся записи в пул атрибутов. Для идентификации распределения и использования атрибутов в модели может применяться матрица сущность/атрибут. Эта матрица обладает следующими свойствами:

1. Все имена сущностей изображаются с краю.

2. Все имена атрибутов изображаются наверху.

3. Использование атрибутов сущностями изображается в соответствующей строке с помощью следующей кодировки:

- «O» = владелец,
- «K» = первичный ключ,
- «I» = наследуемый.

Пример матрицы сущность/атрибут приведен в таблице 3.4. Эта матрица является основным средством поддержки целостности модели.

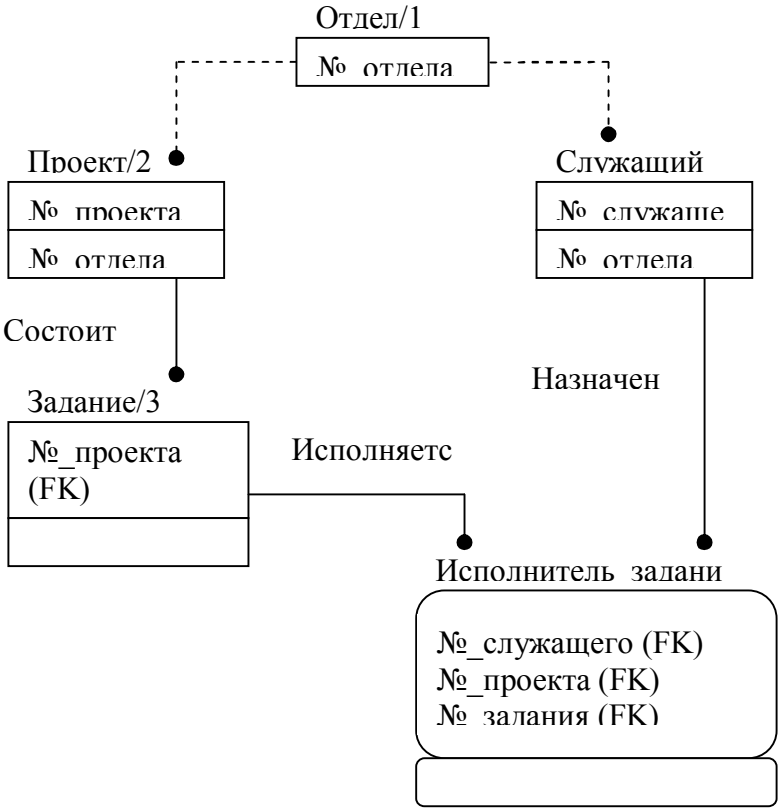

Рис. 3.8. Пример триады

Таблица 3.4. Пример матрицы сущность/атрибут

Сущности		Имена соответствующих атрибутов приведены в таблице 3.5									
		1	2	3	4	5	…	…	20	21	22
Условия покупки	1	Ок									
Покупатель	2		Ок								
Продавец	3			Ок							
Заказ	4										
Оформитель	6										
Деталь	9										
Пункт заказа	10	Ок									
Строка заказа	12	Ок									
Администратор	21			Ок							
Поставщик	22										

Таблица 3.5. Имена атрибутов

№	Атрибуты	№	Атрибуты
1	Номер условий покупки	12	Имя оформителя
2	Код покупателя	13	Код отдела
3	Код продавца	14	Послать через …
4	Код заказа	15	Имя покупателя
5	Номер измерения	16	Номер заказа
6	Пункт назначения	17	Дата отпуска заказа
7	Имя продавца	18	Код получателя
8	Адрес продавца	19	Код налога
9	Код подтверждения	20	Код дилера
10	Имя подтверждающего лица	21	Номер бланка
11	Код дополнительных копий	22	Условия платежа

Определение ключевых атрибутов. После установления ключей наступает момент для определения атрибутов, которые были использованы в качестве ключей. Для этих определений будут

использоваться те же основные принципы: определения должны быть точными, специфическими, полными и универсально понимаемыми. Определения атрибутов всегда ассоциированы с сущностями, владеющими этими атрибутам, т.е. они всегда являются элементами набора документов сущностей-владельцев. Поэтому достаточно просто определить атрибуты, которые принадлежат каждой сущности и используются в этом первичном или альтернативном ключе сущности. В примере из таблицы 3.4 эти атрибуты кодируются ОК в матрице сущность/атрибут.

Определение атрибута включает:

- имя атрибута;
- определение атрибута;
- синонимы атрибута.

Изображение результатов стадии определения ключей. В результате идентификации и миграции ключей диаграммы функционального представления могут теперь быть обновлены для отражения и детализации отношений. Диаграммы функционального представления должны изображать:

- атрибуты первичных, альтернативных и внешних ключей;
- независимые от идентификатора (с прямыми углами) и зависимые от идентификатора (с закругленными углами) сущности;
- идентифицирующие (сплошная линия) и неидентифицирующие (штриховая линия) отношения.

Большую часть информации, полученной в результате анализа на этой стадии, содержат сами сущности. Каждый набор документов сущности содержит:

- определение сущности;
- список атрибутов первичных, альтернативных и внешних ключей;
- определения принадлежащих сущности ключевых атрибутов;
- список отношений, в которых сущность является общей;
- список отношений, в которых сущность является сущностью-категорией;
- список идентифицирующих отношений, в которых сущность является сущностью-родителем;
- список идентифицирующих отношений, в которых сущность является сущностью-потомком;
- утверждения двойных путей (в случае необходимости).

Если нужно, разработчик может построить для сущности отдельную диаграмму, следуя тому же подходу, что и при построении необязательной диаграммы сущности на стадии определения отношений.

Помимо табличных списков определений отношений, полезны перекрестные обратные ссылки на ассоциированные сущности. В документах, разработанных на стадии определения ключей, необходимо перекрестно ссылаться на те атрибуты, которыми сущности обладают, и на те, которые они разделяют.

3.5. Стадия определения атрибутов

Данная стадия является завершающей стадией разработки модели. Она включает:

- разработку пула атрибутов;
- установление принадлежности атрибутов;
- определение неключевых атрибутов;
- проверку правильности и детализацию структуры данных.

Результаты стадии определения атрибутов изображаются на одной или нескольких диаграммах (диаграммах уровня атрибутов). В конце стадии модель данных полностью детализируется (в соответствии с пятой нормальной формой в реляционной теории). Модель снабжается полным множеством определений и перекрестных ссылок для всех сущностей, (ключевых и неключевых) атрибутов и отношений. Пример пула атрибутов приведен в таблице 3.6.

Таблица 3.6.

Номер	Имя атрибута	Номер исх. данных
1	Номер заказа на покупку	1
2	Код покупателя	2
3	Код продавца	3
4	Код заказа	4
5	Номер замены	5
6	Куда отгружен	6
7	Имя продавца	8
8	Адрес продавца	8
9	Код упаковщика	9
10	Имя упаковщика	9
11	Имя заказчика	11,42
12	Код отдела	12
13	Отгружено через	13
14	Имя покупателя	14
15	Номер заказа на покупку	15
16	Дата подписи документа на отгрузку	16
17	Код контроля качества	17
...	...	...

На стадии определения сущностей в пул сущностей в качестве потенциальных сущностей было введено много имен из списка исходных данных начальной стадии. Некоторые из этих имен, однако, могли не быть признаны на стадии определения ключей в качестве сущностей. По всей вероятности они являются атрибутами. Кроме того, многие из имен, не выбранные из списка исходных данных сначала, являются, возможно, атрибутами. Этот список, в сочетании со сведениями, полученными на стадиях определения сущностей и отношений, является основой для установления пула атрибутов. Пул атрибутов является списком потенциально жизнеспособных атрибутов, замеченных в контексте модели. Этот список будет, по всей вероятности, заметно больше пула сущностей.
Пул атрибутов является источником имен, используемых в модели. Атрибуты, появившиеся на более поздних стадиях моделирования, добавляются в пул атрибутов и им приписываются уникальные идентифицирующие номера. Затем они развиваются для дальнейшего использования в модели.

Определение владельцев атрибутов. На следующем этапе каждый неключевой атрибут должен быть приписан к одной сущности-владельцу. Для многих атрибутов сущности-владельцы очевидны. Например, разработчик без заминки свяжет атрибут ИМЯ-ПРОДАВЦА с сущностью ПРОДАВЕЦ. Однако некоторые атрибуты могут вызвать у разработчика трудности при определении их сущностей-владельцев.
Если разработчик не уверен в сущности-владельце для атрибута, он может обратиться к исходному материалу, откуда был выбран атрибут. Это поможет в определении владельца. На начальной стадии был сформирован список исходных данных, ставший основой пула атрибутов. Этот список указывает разработчику места, где значения представленных атрибутов использовались в исходном материале. Анализ примеров использования атрибута в исходном материале упрощает поиск сущности-владельца в модели. Он должен иметь в виду, что главенствующим фактором в определении владельцев атрибутов является вхождение экземпляров атрибутов, представленных отражаемыми в исходном материале значениями атрибутов. После того, как каждому атрибуту будет назначена сущность-владелец, это назначение должно быть зафиксировано.

Определение атрибутов. Для всех выявленных на данной стадии атрибутов должны быть разработаны определения. Здесь также применимы основные принципы построения определений, уже используемых в модели данных (в особенности из стадии определения ключей). Разработанные определения должны быть точными, специфическими, полными и универсально понимаемыми. Эти определения атрибутов записываются в том же формате, что и определения атрибутов из стадии 3.
Определение атрибута включает:
- имя атрибута;
- определение атрибута;
- синонимы/псевдонимы атрибута.

Каждому атрибуту должно быть присвоено уникальное имя, поскольку в IDEF1X-модели и к сущностям, и к атрибутам применяется правило «одно и то же имя – один и тот же смысл». Поэтому разработчик может воспользоваться стандартным подходом к именам атрибутов. Однако для облегчения чтения при проверке правильности лучше использовать привычные для пользователя естественные названия. Если встречаются имена атрибутов, которые должны удовлетворять строгим правилам языка программирования (например, ограниченность имен переменных в Фортране семью символами), то они должны всегда идентифицироваться как псевдонимы или не включаться вовсе.
В определении атрибута разработчик может пожелать установить формат атрибута, например, буквенно-цифровой код, текст, денежные единицы, дату и т.д. В определении может быть также установлена область допустимых значений в формате списка (например, понедельник, вторник, среда, четверг, пятница) или диапазона (например, больше нуля, но меньше десяти). В определении могут быть также указаны утверждения, включающие несколько атрибутов. Например, атрибут ОКЛАД_СЛУЖАЩЕГО должен быть больше 20000 долларов или СЛУЖЕБНЫЙ_КОД_СЛУЖАЩЕГО равен двенадцати.

Детализация модели. Теперь разработчик готов начать детализацию отношений на стадии определения атрибутов. Здесь используются те же основные правила, что и на стадии определения ключей. Правила необращения в ноль и неповторяемости теперь применяются и к ключевым, и к неключевым атрибутам. В результате могут возникнуть некоторые новые сущности. После идентификации этих сущностей

должно применяться правило миграции ключей точно так же, как на стадии определения ключей.

Единственное отличие в применении на данной стадии правил необращения в ноль и неповторяемости заключается в том, что эти правила относятся преимущественно к неключевым атрибутам. На рис. 3.10 проиллюстрировано применение к неключевому атрибуту правила необращения в ноль. На рис. 3.11 приведен пример применения к неключевому атрибуту правила неповторяемости.

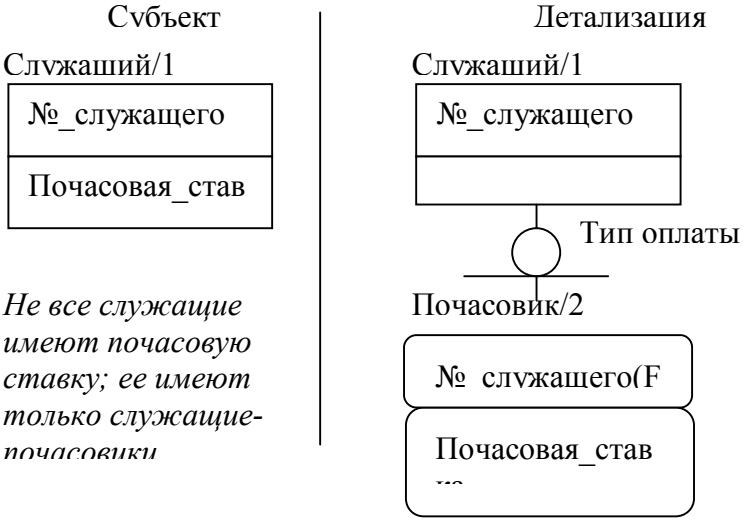

Рис. 3.10. Пример правила необращения в ноль

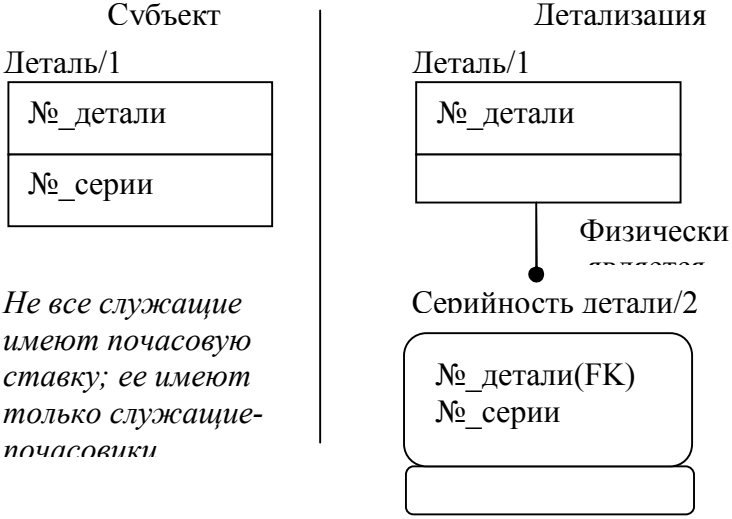

Рис. 3.11. Пример правила неповторяемости

Вместо того чтобы немедленно создавать новые сущности для атрибутов, нарушающих правила детализации, можно отмечать такие нарушения по мере их выявления с тем, чтобы позднее создать новые сущности. Рядом с именами атрибутов-нарушителей в диаграммах атрибутов могут быть сделаны пометки в скобках (буква N для нарушителей правила необращения в ноль и буква R для нарушителей правила неповторяемости).

После выявления новых сущностей они должны быть введены в пул сущностей, определены, отражены в матрице отношений и т.д. Короче говоря, новые сущности должны удовлетворять всем требованиям к документации, созданной на более ранних стадиях, с тем, чтобы их можно было включить в материал данной стадии.

Должна быть также определена принадлежность каждого атрибута в соответствии с правилом полной функциональной зависимости. Это правило утверждает, что ни одно значение неключевого атрибута, принадлежащего экземпляру сущности, не может быть идентифицировано лишь частью значения ключа данного экземпляра сущности. Это правило применимо только к сущностям с составными ключами и эквивалентно второй нормальной форме в реляционной теории.

Все атрибуты модели на данной стадии должны также удовлетворять правилу отсутствия транзитивной зависимости. Это правило требует, чтобы значение принадлежащего экземпляру сущности неключевого

атрибута не могло идентифицироваться значением другого принадлежащего экземпляру сущности или наследуемого ею неключевого атрибута. Это правило эквивалентно третьей нормальной форме в реляционной теории.
Простой способ для запоминания правил полной функциональной зависимости и отсутствия транзитивной зависимости можно сформулировать так: неключевой атрибут должен зависеть от ключа, всего ключа и ни от чего другого, кроме ключа.

Представление результатов стадии определения атрибутов. После определения атрибутов диаграммы функционального представления могут быть теперь обновлены так, чтобы отразить детали модели, и расширены с тем, чтобы показать неключевые атрибуты. Неключевые атрибуты перечисляются ниже линии внутри каждого блока сущности. Для того чтобы внутри блока сущности хватило места, размеры блока могут быть увеличены. На рис. 3.12 приведен пример функционального представления данной стадии.
Соответствующие определения и информация для модели должны быть обновлены, чтобы отразить определение неключевых атрибутов и их принадлежности. Эта дополнительная информация может быть представлена сущностью вместе с ранее определенной информацией. Теперь каждый набор документов сущности будет содержать:

- определение каждой сущности;
- список первичных, альтернативных и внешних ключевых атрибутов;
- список принадлежащих сущности неключевых атрибутов;
- определение каждого принадлежащего сущности атрибута (как ключевого, так и неключевого);
- список отношений, в которых данная сущность является родительской;
- отношение категоризации;
- идентифицирующие отношения указанного выше типа;
- неидентифицирующие отношения указанного выше типа;
- список отношений, в которых данная сущность является сущностью-потомком;
- отношение категоризации;
- идентифицирующие отношения указанного выше типа;
- неидентифицирующие отношения указанного выше типа;
- Утверждения всех двойных путей.

Необязательные диаграммы сущностей также могут быть расширены для указания неключевых атрибутов.

Определения отношений могут быть повторены в комплекте документов для каждой сущности или перечислены отдельно с перекрестными ссылками на эти сущности. Ключевые и неключевые атрибуты должны быть также перечислены и снабжены перекрестными ссылками на эти сущности.

Список литературы

1. Методология IDEF1X. Стандарт. Русская версия – М.:МетаТехнология, 1993.
2. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998.
3. Калянов Г.Н. Консалтинг при автоматизации предприятий: Научно-практическое издание. Серия «Информатизация России на пороге XXI века». – М.:СИНТЕГ, 1997.
4. Калянов Г.Н. CASE-технологии. Консалтинг при автоматизации бизнес-процессов. – М.:Горячая линия, 2000.

Методические указания для организации самостоятельной работы

Цели самостоятельной работы

Целью самостоятельной работы является получение знаний о новых направлениях в развитии информационных технологий и применении их в сфере государственного и муниципального управления.

В рамках самостоятельной работы студентам предлагается более глубоко рассмотреть ряд вопросов. Это направлено на расширение кругозора и уяснение роли информатизации в управленческой деятельности. Студент либо сам предлагает для изучения интересующий его вопрос либо ему дается конкретная тема, по которой он представляет рефераты и (по желанию студента) можно выступить с докладом на практических занятиях. Основой данного вида работ является сбор материала, анализ собранного материала, консультации с преподавателем, сообщение по итогам изучения материала.

В течение данного курса самостоятельная работа предусмотрена в виде ряда форм, в числе которых:

1. Проработка лекционного материала – 6 часов;
2. Подготовка к лабораторным работам 20 часов;
3. Изучение тем (вопросов) теоретической части курса, отводимых на самостоятельную проработку – 38 часов;

Список тем, вынесенных на самостоятельную проработку, приводится ниже.

1. Организация и средства ИТ обеспечения управленческой деятельности.
2. XML-документы.
3. ГИС и Интернет.
4. Законы Зипфа.
5. OLAP-технологии в управлении.
6. Стандарты описания системных проектов.
7. Методология онтологического исследования IDEF5.
8. Пространственный анализ в управлении территорией.

9. Языки разметки географической информации.
10. Веб-ориентированные информационные системы в управлении.

При подготовке к лабораторным вопросам особое внимание рекомендуется уделить следующим вопросам.

1 Разметка электронных документов (списки, таблицы, гипертекст)

Вспомните виденные Вами ранее документы и обратите внимание на вид и структуру имеющихся в них списков и таблиц. Проанализируйте общие черты этих элементов в разных документах и сопоставьте их с описанием элементов для разметки документов в языке разметки гипертекста HTML, на базе которого Вы будете выполнять лабораторную работ.

2 Разработка модели деятельности автоматизируемого процесса

Проработайте детали выбранного для моделирования процесса на базе данных сети Интернет. Обратите особое внимание на предлагаемые в сети варианты автоматизации подобных процессов.

3 Основы языка разметки географической информации KML

Изучите на основе данных сети интернет особенности территорий, по которым будет пролегать размечаемый Вами маршрут. Соберите данные для создания качественного описания понравившихся Вам мест и объектов культурного наследия в ходе дальнейшего выполнения лабораторной работы.

4 Разработка электронной карты в среде QGIS

Изучите наличие растровых и векторных данных в сети интернет на участок территории, выбранный Вами для разработки карты в среде QGIS. Выберите и проанализируйте найденные данные в плане рассматриваемого процесса, для автоматизации управления которым будет возможно использоваться создаваемая электронная карта.