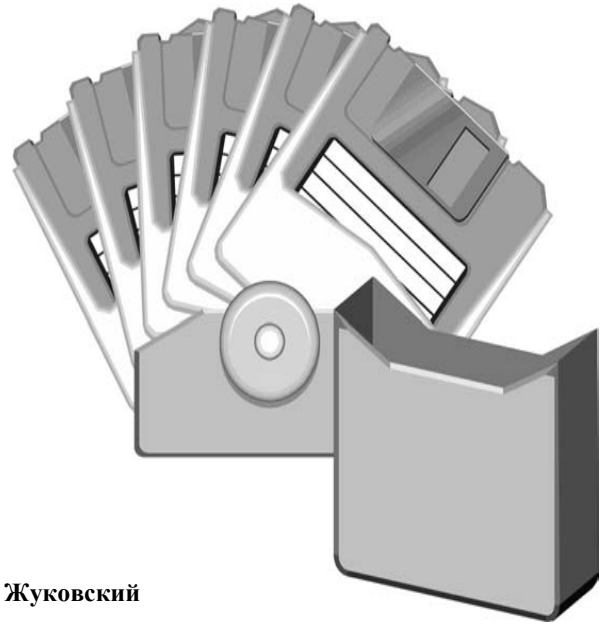


О.И. Жуковский

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Министерство образования Российской Федерации
Томский государственный университет систем управления и радиоэлектроники

О.И. Жуковский

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Рекомендовано Сибирским региональным учебно-методическим
центром высшего профессионального образования
для межвузовского использования в качестве учебного пособия

УДК 681.3:002.6
ББК 32.97
Ж 86

Рецензенты:
кафедра прикладной математики
Томского политехнического университета,
зав. кафедрой профессор, д-р физ.-мат. наук **Григорьев В.П.**

профессор кафедры системного анализа и управления
Томского госуниверситета д-р техн. наук **Тарасенко В.Ф.**

Жуковский О.И.
Ж 86 Информационные технологии: Учебное пособие. — Томск:
Томск. гос. ун-т систем управления и радиоэлектроники, 2003. —
167 с.
ISBN 5-86889-122-8

Учебное пособие отражает современные методы и средства, используемые в новейших информационных технологиях. Рассмотрены основные понятия и концепции таких направлений информационных технологий, как структурная разметка документов, построение систем оперативной аналитической обработки данных и обнаружение знаний в хранилищах данных, CASE-технологии и геоинформационные системы, приведено большое количество иллюстраций и практических при-меров.

Предназначено для студентов, обучающихся по направлению подготовки 654600 «Информатика и вычислительная техника» специальности 220200 «Автоматизированные системы обработки информации и управления», а также представляет интерес для специалистов в области обработки информации.

УДК 681.3:002.6
ББК 32.97

ISBN 5-86889-122-8

Учебное издание

Жуковский Олег Игоревич

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
Учебное пособие

Редактор Кирпиченко Л.И.
Технический редактор Коновалова Н.В.
Корректор Коновалова О.В.

Подписано в печать 31.03.03. Формат 60x84/16. Бумага офисная.
Печать трафаретная. Гарнитура Times New Roman.
Усл. печ. л. 9,77. Учет.-изд. л. 8,28.
Тираж 100. Заказ № 102

Томский государственный университет систем управления
и радиоэлектроники. 634050, Томск, пр. Ленина, 40

ОГЛАВЛЕНИЕ

/

ВВЕДЕНИЕ

Учебная дисциплина «Информационные технологии» входит в состав общеобразовательных дисциплин Государственного образовательного стандарта по специальности 220200 «Автоматизированные системы обработки информации и управления» (АСОИУ).

Информационные технологии как самостоятельное научное направление сформировалось в начале 90-х годов XX века. Под ***информационной технологией*** понимается совокупность методов получения, хранения, преобразования и передачи информации в той или иной сфере деятельности человека на основе компьютерной поддержки. Задача информационной технологии состоит в выявлении наиболее эффективных средств и способов обработки информации.

Учебное пособие подготовлено в соответствии с Государственным образовательным стандартом по данной дисциплине и раскрывает содержание новых информационных технологий, особенно активно используемых в настоящее время. Пособие содержит пять разделов. В пер-вом разделе раскрывается содержание информационной технологии как составной части информатики [1]. Во втором разделе приводится современная технология обработки текстовых сообщений, основанная на принципах структурной разметки документов [2, 3, 4, 5]. В третьем разделе излагаются современные информационные технологии обработки данных, рассматриваются принципы обработки данных в системах оперативной обработки транзакций и в системах оперативной аналитической обработки данных, приводятся наиболее важные моменты технологии обнаружения знаний в хранилищах данных [6, 7, 8, 9]. В четвертом разделе рассматриваются основные положения CASE-технологий. Особое внимание уделено методологии функционального моделирования IDEF0 [10, 11]. Пятый раздел посвящен изложению базовых идей и ме-тодов геоинформационных технологий [12].

1. СООБЩЕНИЕ И ИНФОРМАЦИЯ

1.1. ОСНОВНЫЕ ПОНЯТИЯ

Информатика — научное направление, занимающееся изучением законов, методов и способов накапливания, обработки и передачи информации с помощью ЭВМ и других технических средств. Информационные технологии как система методов и способов сбора, накопления, хранения, поиска, обработки и выдачи информации, являясь составной частью информатики, используют в качестве ключевых основные понятия информатики — ***сообщение и информацию***. Техническое значение приведенных понятий не вполне соответствует употреблению данных двух слов в обиходной речи. Необходимое в связи с этим уточнение содержания указанных понятий не может быть достигнуто с помощью определения, так как последнее лишь сводило бы их к другим не определённым основным понятиям. Поэтому целесообразно ввести сообщение и информацию как неопределяемые основные понятия и рассмотреть их использование на ряде примеров [1].

При разграничении понятий сообщения и информации будем исходить из распространённых оборотов речи типа «это сообщение не даёт мне никакой информации», что приводит к следующему отношению между этими понятиями: (абстрактная) информация передаётся посредством (конкретного) сообщения.

Соответствие между сообщением и информацией не является вза-имно-однозначным. Для одной и той же информации могут существовать различные передающие её сообщения, например сообщения на разных языках или сообщения, которые получаются добавлением неважного сообщения, не несущего никакой дополнительной информации. Сообщения, передающие одну и ту же информацию, образуют класс эквивалентных сообщений. С другой стороны, одно и то же сообщение может передавать совершенно различную информацию: сообщение о падении самолета для близких родственников погибшего имеет совсем иной смысл, нежели для авиакомпании; разные читатели из одной и той же газетной статьи черпают совершенно различную информацию, соответствующую кругу их интересов.

Таким образом, одно и то же сообщение, по-разному интерпретированное, может передавать разную информацию. Обобщая, можно сказать, что решающим для связи между сообщением N и информацией I является некое отображение α, представляющее собой результат договорённости между отправителем и получателем сообщения или предписанное им обоим и называемое ***правилом интерпретации.*** Символически мы будем записывать правило интерпретации в следующей форме: $N \xrightarrow{\alpha} I$.

Правило интерпретации α для данного сообщения часто получается как частный случай некоторого общего правила, применимого к целому множеству $\mathfrak{R}$ сообщений, которые построены по одинаковым законам. Если мы формулируем сообщения на некотором ***языке***, то высказывание «X понимает язык $\mathfrak{R}$» выражает тот факт, что лицо X знает правило интерпретации α для всех (или по крайней мере для большинства) сообщений, формулируемых на данном языке.

Иногда правило интерпретации известно лишь ограниченному кругу лиц; сюда относятся правила интерпретации для специальных языков, в частности для различных профессиональных и научных языков (жаргонов).

Связь между сообщением и информацией особенно отчётливо видна в криптографии: здесь никто посторонний не должен суметь извлечь информацию из передаваемого сообщения, иначе это означало бы, что он располагает «ключом».

Часто встречаются и такие сообщения, которые могут интерпретироваться по-разному, причём различные интерпретации основываются одна на другой. Так, сообщение «идёт дождь» может нести дополнительную информацию «нужно взять с собой зонтик». В этом случае говорят об информации различной ***степени отвлечённости.***

Для сообщений, которыми обмениваются люди, в большинстве случаев имеются соглашения относительно их формы. О таких сообщениях будем говорить, что они передаются в ***языковой форме,*** что они составлены на некотором ***языке.*** При этом слово «язык» используется в существенно более широком смысле, чем в случае связанного с ним понятия «говорить». Мы знаем разговорный и письменный языки, язык глухонемых, построенный на жестах и мимике, печать для слепых, воспринимаемую осязанием. Два последних примера показывают, что высокоразвитое языковое общение не ограничивается устной и письменной речью. Хотя многое говорит в пользу того, что именно разговорный язык знаменует начало истории человека, всё же и в современном обществе остаётся язык жестов, дополненный специфическими звуками, такими как шипение, мычание, свист и щелчки, — хотя и примитивное, но иногда решающее вспомогательное средство, обеспечивающее взаимопонимание.

Когда мы говорим о ***языковых сообщениях,*** мы имеем в виду то общее, что присуще каждому из этих случаев; способ передачи — письменно, устно, посредством осязания или ещё как-то — не имеет здесь никакого значения. При этом следует иметь в виду, что, например, информация, содержащаяся в устном сообщении, не всегда полностью воспроизводится соответствующим письменным сообщением. Такие чувства, как гнев, радость, горечь, искренность, находят своё полное выражение только в устной речи. Ударения и паузы в устной речи также несут информацию.

В случае, когда мы говорим о немецком языке, английском языке и других, лучше использовать термин ***язык-речь***. Для различия между языком и языком-речью характерно, что внутри данного языка-речи можно говорить на высоком языке, на разговорном языке, на блатном языке (сленге) или на профессиональном языке (жаргоне). Существуют также языки, которые не принадлежат и вообще не могут быть причислены ни к какому языку-речи, например искусственные языки вроде эсперанто или некоторые специальные языки, в том числе язык формул математики и языки программирования.

Наконец, понятие языка не ограничивается случаем общения между людьми, оно используется и в случае сравнительно высокоразвитых форм общения между другими живыми существами. Примером может служить открытый К. Фришем язык ориентации пчел.

Для нашего предмета особенно важны языки, в которых для передачи сообщений используются ***долговременные носители информации.*** При этом передача освобождается от гнёта реального времени, и становятся возможными даже сообщения человека самому себе — ***заметки*** на память, чем уменьшается нагрузка на человеческую память за счёт использования «инструмента». Представление сообщений на долговременных носителях будем называть ***письмом,*** а сам долговременный носитель — ***носителем письма*** [1].

Прежде всего следует упомянуть зрительно воспринимаемое письмо, которое создается вручную (рукопись) или механически (машинопись, типографская печать). Письмо, воспринимаемое на слух, стало реальностью после того, как Эдисон изобрел фонограф. Письмом, воспринимаемым осязанием, является письмо слепых, которое пишется вручную (посредством наколов иголкой), а также механически. Фиксация изображений (например, в кино) также представляет собой письмо.

1.2. РОЛЬ ОРГАНОВ ЧУВСТВ В ВОСПРИЯТИИ СООБЩЕНИЙ

Выше уже были упомянуты органы чувств, которые могут служить для передачи языковых сообщений. Некоторые из воспринимающих ор-ганов чувств служат также и для односторонней, неязыковой, связи с окружающим миром. У высших живых существ только слуховые, зрительные и тактильные восприятия достаточно дифференцированы, чтобы естественным образом служить для передачи языковых сообщений. Наряду с языками непосредственного общения — разговорной речью; призывающими и предостерегающими звуками (слуховое восприятие); языком глухонемых (зрительное восприятие); воспринимаемым рукой языком слепых (тактильное восприятие) — существуют языки, в которых используются инструменты (барабанный бой, стук, свистки, звуки рожка или трубы, сирена, световые сигналы, сигналы флажками).

Определенные знания о свойствах и работе органов чувств человека оказываются существенными, когда мы хотим рационально включить человека в цепочку обработки и передачи информации (в её начало или конец). Функциональная способность органов чувств лежит в определенных пределах. Здесь прежде всего следует назвать ***время реакции*** (латентное, или скрытое, время). Для акустических (звуковой импульс) и оптических (загорание лампочки) сигналов оно составляет для человека 140–250 мс до ответа, состоящего в том, что испытуемый нажимает кнопку. Для более сложных заданий время реакции заметно увеличивается (прочитать указанное слово — 350–550 мс, назвать указанный предмет домашнего обихода — 600–800 мс). Это уже говорит о том, что процесс восприятия — не только функция рецепторов. Сюда примыкают проведение раздражения по нервным путям, переработка его в мозге, а также проведение ответа к эффектору. При этом на глаз как на воспринимающий орган приходится около 40 мс, а на срабатывание мышц руки как передающего органа — около 50 мс. Для того чтобы орган вообще смог что-либо воспринять, интенсивность раздражения должна превосходить определённое ***пороговое значение***. Для слуха пороговое значение составляет около $2 \cdot 10^{-7}$ мбар.

Знакомство с физиологией и психологией чувств учит нас многому, прежде всего тому, что обработка информации наряду с ее передачей есть обязательная составная часть нашего чувственного восприятия. Когда мы говорим об искусственной, т.е. изобретённой людьми и выполняемой машинами, обработке информации, то должны помнить о том, что обработка информации не является чем-то принципиально новым.

1.3. УСТРОЙСТВА СВЯЗИ И ПЕРЕДАЧА СООБЩЕНИЙ

Письмо и газета относятся к самым старым и до сих пор не устаревшим способам (случайным или регулярным) передачи сообщений посредством записи на ***долговременном*** носителе сообщений. В случае передачи информации с помощью ***недолговременного*** носителя сообще-ний человек использует также различные физические устройства в соответствии с уровнем развития техники на данный момент. Примерами таких устройств связи служат телефон, радио и телевидение, предназначенные как для случайной, так и для регулярной передачи сообщений.

Внешне ***устройство связи,*** или, точнее, приёмно-передающее устройство, состоит из ***приёмника*** (получателя) и ***передатчика*** (отправителя). О внутреннем строении устройств связи никаких общих утверждений сделать нельзя, разве что при более внимательном рассмотрении многие из них оказываются составленными из нескольких более мелких устройств связи.

Может случиться, что для сообщений на входе и на выходе используется один и тот же носитель. Такие устройства служат лишь для возможного ***усиления*** или ***регенерации*** сообщения, связанной с устранением помех, и называются ***релейными линиями***. Примерами таких устройств являются рупор, слуховая трубка, а также их современные электронные варианты — мегафон и слуховой аппарат. Если для сообщений на входе и выходе устройства используются различные физические носители, то устройство связи называют ***преобразователем.***

Если устройство предназначено для связи между людьми, то сообщения на входе или выходе должны быть производимы или соответственно воспринимаемы людьми, т. е. носители должны соответствовать человеческим эффекторам и рецепторам. В качестве примеров назовём музыкальный инструмент, на котором играют, нажимая на клавиши (физический носитель на входе — давление, на выходе — звуковые волны), и осциллограф, управляемый через микрофон (на входе — звуковые волны, на выходе — световые волны).

Как уже упоминалось, два или более устройств связи можно соединить друг с другом таким образом, что результирующее устройство снова будет устройством связи. Приёмником составного устройства является приёмник первого устройства, участвующего в соединении, а передатчиком — передатчик последнего устройства. В этом случае между передатчиком одного устройства связи и приёмником другого могут использоваться и такие носители, которые недоступны человеческим эффекторам и рецепторам. Примером могут служить телефонная связь по проводам или радио.

Исходя из подобных примеров, протяжённую в пространстве среду, «через» которую носитель сообщения передаётся от передатчика к приёмнику, называют ***каналом связи.***

Помимо бумаги, используемой в письме и чтении человеком, в качестве долговременных носителей текстовых сообщений в современной технике чаще всего используются намагничиваемые и светочувствительные плёнки и диски, а также перфорируемая бумага (перфокарты, перфоленты).

Аналогии между рецепторами и эффекторами живых существ и техническими приёмопередающими устройствами служат ***предметом исследований в кибернетике.*** Кибернетика занимается главным образом аспектами, общими для человека и технических устройств с точки зрения передачи и переработки сообщений.

Следует особо отметить то, что передача сообщений происходит во времени. Поэтому в качестве носителей заслуживают внимания только такие физические процессы, которые могут изменяться во времени. Изменение некоторого физического процесса во времени, обеспечивающее передачу сообщения (а тем самым и информации), называется ***сигналом.*** При этом для воспроизведения сообщения могут использоваться различные свойства сигнала. Характеристика сигнала, которая служит для представления сообщения, называется ***параметром сигнала*** [1].

В качестве примера рассмотрим радио. Сигналами здесь являются электромагнитные колебания. В диапазоне средних волн сообщение воспроизводится амплитудой колебаний, а в диапазоне ультракоротких волн — частотой колебаний ***(амплитудная и частотная модуляции*** соответственно). Таким образом, в первом случае параметром сигнала является амплитуда, а во втором — частота колебаний.

Если для передачи сообщений используются импульсы, то параметром сигнала может быть либо амплитуда импульса, либо интервал между импульсами ***(амплитудно-импульсная и частотно-импуль-сная модуляции*** соответственно).

1.4. ДИСКРЕТНЫЕ СООБЩЕНИЯ И ЗНАКИ

Сигнал называется ***дискретным,*** если параметр сигнала может принимать лишь конечное число значений и существен лишь в конечном числе моментов времени (возможно, периодически повторяющихся). ***Дискретными сообщениями*** называются такие сообщения, которые могут быть переданы с помощью дискретных сигналов.

Языковые сообщения в письменной форме строят обычно, записывая знаки письма ***(графемы)*** друг за другом. Хотя длинные сообщения могут размещаться на многих строчках и страницах, это разбиение не имеет, вообще говоря, никакого значения; оно не несёт важной информации. По существу, такие сообщения являются последовательностями знаков. Это оказывается справедливым и для устных языковых сообщений, если разложить устный текст на элементарные составные части, так называемые ***фонемы,*** и под знаками понимать фонемы. Чтобы можно было воспроизводить фонемы и письменно, принято соглашение о международных письменных знаках для отдельных фонем. Точка зрения, что сообщение есть последовательность знаков, не ограничивается тем случаем, когда знаки — это фонемы или графемы (например, знаки букв и цифр, знаки препинания). Знаки планет или знаки зодиака и даже кивок и покачивание головой также могут пониматься как знаки. В силу этого определим понятие знака следующим образом.

Знак — это элемент некоторого конечного множества, набора знаков, все элементы которого отличны друг от друга.

Набор знаков, в котором определён линейный порядок знаков, называется ***алфавитом*** [1].

Вот некоторые примеры алфавитов (порядок в них — это порядок перечисления):

а) алфавит десятичных цифр — {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

б) алфавит заглавных кириллических букв — {А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я}.

Вот некоторые наборы знаков, для которых нет какого-либо общепринятого порядка знаков: набор знаков клавиатуры пишущей машинки; набор знаков планет.

Особенно важное значение имеют наборы, состоящие всего из двух знаков. Такие наборы называют ***двоичными наборами знаков***, а сами знаки ***— двоичными знаками.*** Вместо термина «двоичный знак» часто употребляют сокращение ***бит*** (от английского binary digit).

1.5. КОДЫ И КОДИРОВАНИЕ

Если N — предложение некоторого естественного языка, то N можно рассматривать как последовательность знаков, по крайней мере, тремя разными способами. Прежде всего N представляет собой последовательность букв, цифр, знаков препинания и т. д.; далее, N — это последовательность слов, которые в другом контексте могут сами рассматриваться как знаки; наконец, и всё предложение целиком можно рассматривать как один знак.

Первый подход используется, например, когда имеется правило для нанесения сообщения N на перфокарты; второй подход лежит в основе стенографических сокращений; третий подход бывает уместным при переводе на другой естественный язык, когда пословица одного языка переводится соответствующей по смыслу пословицей другого языка.

Дискретные сообщения представляют собой конечные или бесконечные ***последовательности знаков***. При этом, исходя из соображений, связанных с физиологией органов чувств, или из чисто технических соображений, их обычно разбивают на конечные последовательности знаков, называемые ***словами***. На более высоком уровне каждое слово можно снова рассматривать как знак, при этом соответствующий набор знаков будет, вообще говоря, шире первоначального. Наоборот, данный набор знаков можно получить с помощью составления слов, исходя из некоторого набора с меньшим числом знаков, в частности из двоичного набора знаков.

Слова над двоичным набором знаков называются ***двоичными словами.*** Они не обязаны иметь постоянную длину (см. азбуку Морзе), если это всё же так, то говорят об n-разрядных двоичных знаках и n-раз-рядных двоичных кодах (например, 2-й международный телеграфный код (5-разрядные двоичные знаки)).

Будем пользоваться следующим определением: ***кодом*** называется правило, описывающее отображение одного набора знаков в другой набор знаков (или слов); кодом также называют и множество образов при этом отображении, помимо основного значения слова «code» — «кодекс», «свод законов», обозначающего начиная с середины XIX в. книгу, в которой словам естественного языка сопоставлены группы цифр или букв. Употребление таких кодов приобрело значение, скорее, в связи со стремлением сэкономить на стоимости телеграмм, чем в связи с соображениями конспиративности [1].

Если каждый образ при кодировании является отдельным знаком, то такое отображение мы называем ***шифровкой,*** а образы — ***шифрами*** (англ. — cipher). Поскольку здесь имеется криптографический аспект, обращение этого отображения, когда оно однозначно, называется ***декодированием*** или ***дешифровкой.***

В коммерческих и криптографических кодах слова, фразы и понятия естественных языков кодируются в большинстве случаев словами над некоторым буквенным или цифровым алфавитом, обычно пятерками. В технических кодах буквы, цифры и другие знаки почти всегда кодируются двоичными словами. У большинства используемых в технике кодов все слова имеют одинаковую длину. Коды со словами разной длины встречаются в технике довольно редко. Исключением является код Морзе. Это двоичный код с набором знаков «точка», «тире» и словами длины не более 5-ти символов для кодирования букв и цифр. Более точно, следует ещё добавить в качестве третьего знака знак «пропуск», который помечает стыки между кодовыми словами (слова нельзя отделить друг от друга по их длине).

В двоичных кодах с постоянной длиной кодовых слов слова могут следовать друг за другом непосредственно (***последовательная передача***), так что получается единая последовательность двоичных знаков. Расположение стыков и тем самым исходная группировка кодовых слов устанавливаются с помощью отсчёта, и, таким образом, сообщения, составленные из кодовых слов, однозначно декодируемы. Правда, при отсчёте кодовой длины нельзя просчитаться, нельзя «сбиться с ритма», а это ведёт к усложнению с технической точки зрения (параллельность, синхронизация).

Напротив, для кодов с переменной длиной кодовых слов расположение стыков восстановить нельзя. При определённых условиях сообщение, состоящее из нескольких кодовых слов, либо вовсе не декодируется, либо декодируется неоднозначно. Однако декодируемость будет обеспечена, если соблюдается ***условие Фано***: никакое кодовое слово не является началом другого кодового слова («свойство префиксности»). Тогда, очевидно, стык между кодовыми словами определяется тем моментом, когда «дальше не читается». Очевидно также, что код удовлетворяет условию Фано тогда и только тогда, когда кодовое дерево не содержит ни одного языка во «внутренних» вершинах (***дерево с размеченными листьями***) [1].

Условие Фано является достаточным, но не необходимым условием однозначной декодируемости. Тривиальная возможность обеспечить выполнение условия Фано состоит в том, что каждое кодовое слово должно начинаться специальным знаком (или группой знаков), называемым ***разделителем.*** Это, очевидно, имеет место в случае кода Морзе, и именно пропуск является разделителем для последовательности точек и тире. С технической точки зрения при передаче по телеграфу также передается разделитель, синхронизирующий «такт разбивки».

При параллельной передаче мы, в отличие от последовательной, ограничены кодами со словами постоянной длины: для n-разрядного двоичного кода используется n параллельных двоичных каналов передачи. В случае оп-

тического, электростатического, электролитического и электромагнитного телеграфа путь технического прогресса шёл прежде всего от параллельной к последовательной передаче.

Вопрос о том, какие коды являются оптимальными с точки зрения передачи, изучается в теории информации.

1.6. СИМВОЛЫ

Следует различать собственно знак и его смысл. Знак вместе с его смыслом называется ***символом.*** В соответствии с целью употребления один и тот же знак часто имеет разный смысл. Знак «♀» применяется в астрономии как символ планеты Венера, а в биологии — как символ женской особи. К несчастью, часто бывает также, что разные знаки имеют одинаковый смысл; например, знаки «•» и «×», а в последнее время и «*» понимаются как символы умножения.

Обычно всякое сообщение имеет смысл, т. е. уже является символом. Очевидно, что этот символ получается в результате присоединения к сообщению той информации, которая им передается.

1.7. ОБРАБОТКА СООБЩЕНИЙ И ОБРАБОТКА ИНФОРМАЦИИ

Всякое правило обработки сообщений можно понимать как отображение (функцию) ν: $\mathfrak{N} \xrightarrow{\nu} \mathfrak{N}'$, которое сообщениям N из некоторого множества сообщений $\mathfrak{N}$ ставит в соответствие новые сообщения N' из множества сообщений $\mathfrak{N}'$. Каждое из сообщений N и N' — это последовательность знаков.

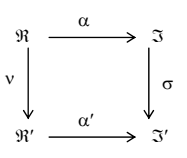
Большая свобода в понимании сообщения как последовательности знаков, просматриваемая в обсуждавшихся выше примерах, позволяет констатировать следующее: ***всякую обработку сообщений можно рассматривать как кодирование***. Конечно, это соображение является важным и для изучения процессов обработки сообщений у живых существ, но прежде всего оно лежит в основе всякой машинной обработки дискретных сообщений.

Чтобы правило обработки $\mathfrak{N} \xrightarrow{\nu} \mathfrak{N}'$ могло служить основой для обработки сообщений, недостаточно того, чтобы правило ν неким акси-оматическим образом задавало условия, которым должно удовлетворять сообщение $N' = \nu(N) \in \mathfrak{N}'$. Правило ν должно задавать некоторый способ построения сообщения $\nu(N) \in \mathfrak{N}'$, исходя из сообщения $N \in \mathfrak{N}$.

Так как обработку дискретных сообщений можно рассматривать как кодирование, то операции, которые следует задать, должны иметь вид преобразований последовательности знаков.

В заключение данного раздела остановимся на том, что множество $\mathfrak{N}$ сообщений N представляет интерес только тогда, когда ему посредством некоторого правила соответствия α сопоставлено, по крайней мере, одно множество $\mathfrak{I}$ сведений J (в нашем случае «сведения» означает «информация во множественном числе»): $\mathfrak{N} \xrightarrow{\alpha} \mathfrak{I}$.

Поскольку множеству сообщений $\mathfrak{N}'$ также соответствует некоторое множество сведений $\mathfrak{I}'$, то правило обработки $\mathfrak{N} \xrightarrow{\nu} \mathfrak{N}'$ можно представить в виде диграммы.



Говорят, что правило обработки ν сохраняет информацию, если соответствие σ явля-ется отображением. Тогда диа-грамма называется $\sigma\alpha = \alpha'\nu$, и коммутативной, а отобра-жение σ называют правилом обработки ин-формации [1].

Обычно сообщения обрабатывают именно для того, чтобы обра-ботать информацию. При этом всегда исходят из определенного правила σ и пытаются определить ν, α и α' таким образом, чтобы получилась ситуация, представленная на диаграмме.

В соответствии с тем, является σ обратимым отображением или нет, различают следующие случаи [1]:

1) если σ обратимое отображение, т. е. информация при обработ-ке не теряется, то соответствующую обработку сообщений называют ***перешифровкой***;

2) если σ и ν обратимы, то имеем простой случай ***перекодировки***, т. е. по сообщению $N' = \alpha(N)$ можно восстановить не только исходную информацию, но и само исходное сообщение N. Наиболее часто встречается случай, когда $\mathfrak{I} = \mathfrak{I}'$, а σ — тождественное отображение. В идеале всякая передача сообщений должна иметь именно такой вид;

3) если σ обратимо, а ν нет, то несколько сообщений $N \in \mathfrak{N}$ будут кодироваться одним и тем же сообщением $N' \in \mathfrak{N}'$. Но так как при этом никакой информации не теряется, то это означает, что исходное множество сообщений $\mathfrak{N}$ было избыточным. Перешифровку ν такого рода называют ***сжимающей.*** Если при этом α' обратимо, то ν называют ***вполне сжимающей*** перешифровкой;

4) если σ необратимое отображение, т. е. разные сведения $J \in \mathfrak{I}$ отображаются в одну и ту же информацию $J' \in \mathfrak{I}'$, то такую обработку сообщений ν называют ***избирательной***. Наиболее часто встречается случай, когда $\mathfrak{I}'$ есть подмножество $\mathfrak{I}$ и σ для сведений из $\mathfrak{I}'$ является тождественным отображением. В этом случае отображение σ, по существу, производит выбор из заданного множества сведений.

Приведем ряд примеров [1]:

- обычный способ чтения газеты избирателен, а изучение разных газетных статей, описывающих одно и то же событие, является сжимающим;
- переход от избыточного кода к менее избыточному или вообще к коду без избыточности, как правило, однозначно обратим. Значит, можно говорить о несжимающей перешифровке, поскольку уменьшается не количество сообщений, а их длина.

Контрольные вопросы

1. Каким образом связаны понятия «сообщение» и «информация»?
2. Что определяет информацию, которая передается конкретным сообщением?
3. Дайте характеристику роли органов чувств в восприятии сообщений человеком.
4. Чем отличается знак от символа?
5. Что называется кодом?
6. Приведите примеры наборов знаков, которые не являются алфавитом.
7. Что понимается под обработкой сообщений?

2. СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ОБРАБОТКИ ТЕКСТОВЫХ СООБЩЕНИЙ

2.1. РАЗМЕТКА ДОКУМЕНТА

Значительную часть сообщений, несущих информацию, человек предпочитает хранить в виде документов. Но хранение документов не является самоцелью — это лишь промежуточный этап работы с информацией. ***Документ*** представляет собой объект, предназначенный для дальнейшего использования читателем, в роли которого может выступать и сам человек, и, что становится все более актуальным, компьютерные программы.

Каждый документ имеет три составляющие — содержание (смысловое наполнение), структуру и внешнее представление. В данной главе не будем останавливаться на анализе содержания, а обратим свое внимание в первую очередь на структуру, а также на внешнее представление. Структура документа позволяет правильно определить составляющие его части и взаимоотношения между ними. Внешнее представление направлено на повышение эффективности восприятия информации читателем, что достигается за счет выделения смысловых частей документа теми или иными средствами, доступными для данной формы представления.

В документе, помимо смыслового наполнения, должна содержаться некоторая метаинформация, позволяющая определить его структуру и внешнее представление. Такая метаинформация называется разметкой документа. Разметка документа преследует следующие цели:

- выделение смысловых частей (логических элементов) документа и связей между ними;
- указание действий, которые должны быть осуществлены с этими элементами.

Для достижения первой цели предназначена ***структурная разметка***. Действия, направленные на получение внешнего представления, задаются ***разметкой представления***. В качестве примеров ниже приведены два возможных способа разметки начала данной главы.

Пример 2.1

```
<div1 type="Section">
 <head>2.1. Разметка документа</head>
  <p>Значительную часть сообщений, несущих информа-
цию,человек предпочитает хранить в виде документов...</p>
</div1>
```

Пример 2.2

```
<font face="Arial Bold" size=16>1. 2.1. Разметка до-
кумента<hspace size=20>
<tab size=5><font face="Times New Roman" size=12>
Значительную часть сообщений, несущих информацию, че-
ловек предпочитает хранить в виде документов...
```

В первом случае мы описываем раздел, который имеет заголовок и текст в виде абзаца, то есть определяем структуру документа. Структурная разметка говорит о том, как текст устроен, то есть из каких он частей состоит и как эти части друг с другом соотносятся.

Во втором случае мы показываем, каким образом данный текст должен быть отображен на бумаге или на мониторе — выделить шрифтом Arial Bold размера 16, отступить по вертикали 20, сделать табуляцию 5, выделить шрифтом Times New Roman размера 12. Здесь мы имеем дело с разметкой представления документа, которая говорит о том, что делать с текстом, как его отображать.

Исторически разметка представления появилась раньше, и в течение длительного времени разметка документа была ориентирована исключительно на внешний (бумажный) вид документа. Но в последнее время ситуация существенно меняется — быстрый рост числа документов, их создание, хранение и использование в электронном виде, автоматизированные обработка и обмен документами предъявляют новые требования к разметке. В числе этих требований — независимость от среды представления, возможность осуществления эффективного поиска, возможность повторного использования документа как целиком, так и отдельных его элементов.

Сейчас существует большое число устройств, с помощью которых можно отображать документы. Среди таких устройств и дисплеи — от компьютерных до мобильных, и принтеры — от формата А1 до миниатюрных, встроенных в кассовые аппараты, и различные синтезаторы речи, и многое другое. Для воспроизведения некоторого документа на всех этих устройствах требуется либо наличие огромного количества вариантов одного и того же документа, только размеченного разными способами, либо существование единой универсальной разметки и программных средств для корректного преобразования в соответствующее внешнее представление.

Быстрый рост количества документов привел к тому, что поиск нужной информации стал занимать все больше и больше времени. Например, если нам необходимо найти в Интернете информацию об авторе статей по фамилии Иванов, то простой контекстный поиск даст нам огромное количество ссылок на те места, где встречается данная фамилия. После чего нам придется либо просмотреть все полученные ссылки, либо задавать дополнительную информацию для сужения области поиска. Если бы мы могли сразу указать, что фамилию следует искать только среди авторов журнальных статей технического плана, это во много раз упростило бы поиск. Но для этого необходимо, чтобы документы, среди которых ведется поиск, были размечены должным образом с явным выделением элементов «автор», «тематика» и т. п.

Возможность повторного использования документов или отдельных его частей приводит к тому, что мы не составляем каждый раз заново отчет или деловое письмо, используем в своей работе шаблоны контрактов, изменяя лишь некоторую существенную для данного случая информацию. Но делаем мы это преимущественно вручную. Если говорить об автоматизированном формировании, связывании, повторном использовании документов, то это становится возможным только тогда, когда документы как информационные объекты являются структурированными, а используемая метаинформация полно и ясно описывает характеристики каждого элемента документа.

Все перечисленные задачи можно решить, используя структурный подход при разметке документов. Именно структурная разметка позволяет выделять смысловые элементы, определять их связи с другими элементами как в рамках одного документа, так и вне этих рамок. Далеко не всякая разметка настолько формализована, что можно говорить о языке разметки. Язык разметки должен определять ряд специальных инструкций, правил и соглашений для описания структуры элементов документа и отношений между элементами этой структуры. Специальные инструкции, их еще называют маркерами или тэгами, в структурированных документах должны определенным образом кодироваться, то есть выделяться среди основного текста. Их главное назначение — служить управляющими инструкциями для программных средств обработки структурированных текстов.

В данной главе мы остановимся на истории возникновения таких языков разметки, как стандартный обобщенный язык разметки SGML (Standard Generalized Markup Language) и язык разметки гипертекстов HTML (Hyper Text Markup Language), а также подробно рассмотрим, что собой представляет расширяемый язык разметки XML (eXtensible Markup Language) [2, 3, 4, 5].

Для эффективной работы с языками разметки необходимо наличие специализированных средств для создания и редактирования размеченных текстов, их просмотра и обработки. Естественно, что развитие языков разметки, принятие стандартов и начало их широкомасштабного использования не могло не сказаться на росте количества предлагаемых программных продуктов. О некоторых из них будет рассказано ниже.

2.2. СТАНДАРТНЫЙ ОБОБЩЕННЫЙ ЯЗЫК РАЗМЕТКИ SGML

2.2.1. Основные положения

Стандартный обобщенный язык разметки SGML был утвержден Международной организацией по стандартизации ISO (International Standards Organisation) в качестве стандарта ISO 8879:1986 в 1986 году.

SGML — это ***метаязык***, то есть средство формального описания прикладных языков разметки, предназначенных для кодирования структурированных документов [3, 4].

Разметка, определяемая в рамках SGML, основывается на двух основных положениях:

- разметка должна описывать структуру документа, а не указывать, что с документом или его частями должно происходить;
- разметка должна быть строгой, чтобы программы и базы данных могли быть использованы для хранения и обработки размеченных документов.

Структура документа с точки зрения SGML представляет собой граф компонентов, вершины которого являются компонентами, а ребра — связями между ними. Основным компонентом структурированного текста является элемент. Таким образом, можно сказать, что каждый структурированный документ состоит из некоторого набора семантических элементов, связанных друг с другом по определенным правилам.

Синтаксическое представление элемента документа показано на рис. 2.1.

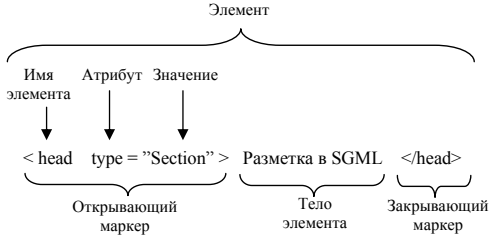


Рис. 2.1. Пример SGML-элемента

Тело элемента (содержательный текст) обрамляется открывающим и закрывающим маркерами. Каждый маркер состоит из имени элемента, уникального для элементов одинаковой семантики, и может иметь некоторое

количество атрибутов. Атрибуты предназначены для более детального описания текста среди семантически однородных элементов.

Важным достоинством SGML является то, что он не определяет заранее имена элементов и их атрибуты. Например, если автор документа считает, что семантически корректнее определить в тексте два типа списков: список фамилий и список компаний, то он может ввести два элемента: listofpeople и listofcompanies. В дальнейшем эти элементы могут обрабатываться как различные семантические единицы.

Чтобы документ являлся синтаксически корректным с точки зрения SGML, необходимо, чтобы его разметка подчинялась некоторому набору правил, определяемых стандартом ISO 8879. Одно из правил состоит в том, что допускается лишь полная вложенность одного элемента в другой. Таким образом, в каждом документе всегда будет один корневой элемент и некоторое количество иерархически вложенных элементов. Вообще говоря, допускается наложение на документ двух независимых разметок, элементы одной из которых могут не являться вложенными в другую, но это предмет отдельного обсуждения. Вложенность является одним из видов связей между вершинами графа компонентов.

Размеченный документ предназначен для дальнейшей обработки различными программами, каждая из которых может применять свои правила обработки к тем или иным элементам документа. Одна программа может преобразовывать текст к виду, пригодному для печати на бумаге, а другая — лишь извлекать некоторые данные (например, названия терминов) и помещать их в таблицу или базу данных.

2.2.2. Типы документов

Структурная разметка не предназначена для обеспечения удобочитаемости документов. Для этого существует разметка представления и соответствующие программные средства, преобразующие структурную разметку в разметку представления. Эти и другие программы, обрабатывающие документ, должны уметь распознавать элементы структуры и атрибуты элементов и применять необходимые операции к определенным элементам. В SGML это достигается с помощью определений ***типов документов*** DTD (Document Type Definition) посредством конструкций языка, называемых декларациями элементов. В то время как разметка документа занимается описанием семантических единиц, DTD определяет набор всех возможных разметок документов описываемого типа [3].

Тип документа формально определяется его составными частями и их структурой. Например, письмо можно определить как документ, имеющий реквизиты отправителя и получателя, заголовок, несколько абзацев и дату отправления. Если документ не имеет реквизитов отправителя, то в соответствии с нашим определением письмом он не яв-ляется.

DTD определяет допустимые элементы для данного типа документа на любом из уровней вложенности, допустимое содержание каждого из элементов и набор допустимых атрибутов. При этом наличие DTD является обязательным для любого документа. Можно сказать, что в рамках SGML имеют право на существование информационные объекты, состоящие из размеченного документа и его DTD.

Декларация элементов в DTD определяет допустимое содержание как тела элемента, так и его атрибутов. Предположим, например, что необходимо дать определение элемента <list>, представляющего собой список. В этом случае декларации могут выглядеть так, как показано в примере 2.3.

Пример 2.3

```
<!-- ELEMENT      MIN    CONTENT (EXEPTIONS) -->
<!ELEMENT list    - -    (head?, item+)>
<!ELEMENT head    - 0    (#PCDATA)>
<!ELEMENT item    - 0    (p+)>
<!ELEMENT p       - 0    (#PCDATA)>
```

Первая декларация (вторая строка листинга, так как первая является комментарием) обозначает, что список может включать необязательный заголовок, но обязательно содержит один или несколько элементов списка. Вторая декларация говорит, что заголовок содержит некоторое количество символов (текст). Третья декларация указывает на то, что каждый элемент списка в свою очередь состоит из одного или более абзацев. И, наконец, последняя декларация, как и вторая, говорит, что абзацы содержат символьный текст.

Символ «0» в колонке MIN обозначает, что закрывающий маркер в данном элементе может быть опущен без нарушения структуры документа. Следующий открывающий маркер такого же элемента или маркер внешнего элемента фактически будет выполнять ту же функцию.

Возможное использование списков приведено в примере 2.4.

Пример 2.4

```
<list>
  <head>Перечень важных дел
  <item>
    <p>В 11-00 переговоры
    <p>Необходимо подготовить полный комплект
       документов
  <item>
    <p>В 14-00 совещание у руководства
</list>
<list>
  <head>Перечень важных дел
  <item>
    <p>В 11-00 переговоры
    <p>Необходимо подготовить полный комплект
       документов
  <item>
    <p>В 14-00 совещание у руководства
</list>
```

2.2.3. Работа с объектами

Одним из достоинств SGML является то, что он позволяет работать не только со структурированными текстами, но и с произвольными информационными объектами. Для этого вводится понятие объекта (entity).

Объектом может быть строка символов или файл (текстовый или бинарный). Для включения его в документ используется конструкция, известная в ряде языков программирования как ссылка на объект. Например, объявление <!ENTITY SGML "Standard Generalized Markup Language"> определяет объект, называющийся SGML, значением которого является строка "Standard Generalized Markup Language". Это пример декларации объекта (entity declaration), которая содержит внутренний объект (internal entity). Следующее объявление, напротив, вводит системный объект (system entity): <!ENTITY picture SYSTEM "picture.gif">.

В этом случае определен объект, являющийся рисунком, а не структурированным текстом. При обработке документа некоторой программой файл picture.gif может быть, например, выведен на экран монитора для иллюстрации соответствующего текста.

2.2.4. Возможности и недостатки SGML

В рамках данной главы мы не ставим цель дать полное описание возможностей SGML и требований, которые в связи с этим накладываются на создаваемый документ. SGML представляет собой достаточно емкий и в то же время сложный метаязык. На его основе создаются языки разметки, используемые в различных областях: подготовка книг, документации, построение систем визуализации данных и т. д. Такие языки, как HTML, XML, MathML, CML и многие другие, созданы на основе SGML и полностью ему соответствуют.

Широта охвата порождает вместе с тем и ряд недостатков. Так, например, создание единого DTD для подготовки документации в рамках одной организации, несомненно, имеет преимущества, такие как унификация исходного кода, возможность автоматического индексирования данных, ведение единого словаря терминов, написание стандартных средств обработки документов, получение стандартного бумажного представления и т. п. Но как только мы выходим за рамки организации, проекта или отрасли, то все упирается в утверждение данного DTD в качестве общего стандарта. Кроме того, как только принимается стандарт на некоторый DTD, сразу начинается борьба за его расширение, и так может продолжаться до бесконечности.

Другой недостаток проявляется при создании программ (например, для редактирования SGML-документов), которые должны позволять работать с любыми возможными DTD и учитывать все возможности, предоставляемые стандартом SGML. К сожалению, это возможно лишь теоретически, так как объем таких программ будет чрезвычайно велик.

Вот почему со временем возникла тенденция создания языков разметки с более простым синтаксисом, которые в то же время подчинялись бы требованиям стандарта SGML.

2.3. ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

Язык разметки HTML был разработан в Лаборатории физики высоких энергий (CERN) в Женеве в 1990 году. Первоначально HTML был предназначен для разметки научных документов и их последующего совместного использования сотрудниками разных институтов и лабораторий. HTML состоял из небольшого фиксированного набора элементов — заголовков нескольких уровней, абзацев, списков и др., но главной его особенностью было

использование гиперссылок и специальных меток (anchors) для указания точек перехода. Все вместе позволяло достаточно легко размечать простые документы и устанавливать связи как между ними, так и между компонентами одного документа. Человек всегда обрабатывает и анализирует информацию нелинейным образом. Поэтому возможности нелинейного хранения информации, простота использования языка разметки и широкая область применения привели к тому, что популярность HTML стала быстро расти. Как это часто бывает с любыми гениальными открытиями, успех превзошел все ожидания создателей.

В 1992 году HTML был формализован в качестве SGML DTD, при этом в его спецификацию была заложена возможность дальнейшего расширения. Простой синтаксис языка, в отличие от SGML, позволял создавать простые программы для анализа размеченного текста и его отображения. Начался бурный рост публикаций в HTML-формате и рост числа приложений, поддерживающих этот формат. Потребности пользователей, а также конкурентная борьба производителей прог-раммного обеспечения привели к тому, что в HTML стали добавляться неспецифицированные элементы разметки. Отсутствие строгих синтаксических правил и использование нестандартных элементов вынудили производителей программного обеспечения допускать использование синтаксически некорректных конструкций. Отметим, что в WWW найдется не так много документов, полностью удовлетворяющих общепринятым спецификациям.

В целях регулирования процесса роста и стандартизации предлагаемых решений для WWW в октябре 1994 года была создана координирующая рабочая группа — World Wide Web Consortium (W3C), которая сегодня объединяет представителей более чем 370 организаций. Основными задачами W3C являются накопление информации о WWW, необходимой как разработчикам, так и пользователям, подготовка и утверждение стандартов (технических спецификаций) на технологии, связанные с WWW, и создание прототипов и образцов приложений для демонстрации использования новых технологий.

Положительная роль W3C в судьбе HTML очевидна — этот язык удалось сохранить от разделения на несколько диалектов, правда, ценой постоянного принятия все новых и новых расширенных спецификаций, которые сменяют друг друга с периодичностью раз в два года.

За время своего существования HTML претерпел множество изменений, что весьма неудобно для создателей документов и разработчиков программ. Но гораздо большей неприятностью стало то, что изначально задуманный как язык структурной разметки в результате своего развития HTML превратился в язык разметки представления. Чего стоит, например, форматирование документа для улучшения его внешнего вида с помощью таблиц. Исходный текст таких документов становится практически нечитаемым, а доля полезной информации составляет лишь несколько процентов [4].

Однако ситуация постепенно начинает улучшаться. К настоящему времени в версии языка HTML 4.0 содержится около 80 элементов. Темп роста их числа заметно уменьшился. Этому способствовало прежде всего введение атрибута CLASS во все элементы. Используя этот атрибут, можно определить новые семантические единицы без изменения синтаксиса языка в целом (рис. 2.2). Кроме того, несомненным шагом вперед по направлению к структуризации языка стало удаление ряда элементов, отвечающих только за внешнее представление, и декларирование строгой необходимости использования таблиц стилей (style sheets) для целей внешнего представления.

```
<div CLASS=”author”>
  <div CLASS=”name”>      Иванов Иван     </div>
  <div CLASS=”email”>  ivan @ mail2000.ru  </div>
</div>
```

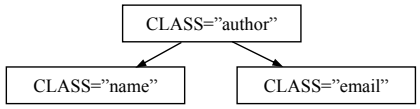


Рис. 2.2. Пример использования атрибута CLASS

Несмотря на массовое признание и использование HTML, а также на ряд разумных шагов, предпринятых W3C, в HTML все еще имеются существенные недостатки.

Отсутствие жесткой иерархии элементов приводит к тому, что один и тот же документ может быть размечен и соответственно будет интерпретироваться программным обеспечением различными способами. Так, например, текст HTML-документа или любая его часть может предваряться заголовком любого уровня, что оставляет автору слишком большую свободу выбора, а читателю создает некоторые трудности при работе с документами разных авторов.

Далеко не всякая метаинформация может быть простым и корректным образом вставлена в документ, поэтому при преобразовании произвольного документа в формат HTML часть информации может быть потеряна. Использование атрибута CLASS только частично решает эту проблему.

Для некоторых областей деятельности HTML не предоставляет возможностей ни структурно размечать требуемые элементы, ни правильным образом выводить их на экран или принтер. Математикам необходима возможность работы с формулами, химикам нужно отображать структуру химических соединений, и вместе с тем всем разработчикам и пользователям WWW необходимо наличие единых принципов разметки документов, универсальность их обработки и отображения.

2.4. РАСШИРЯЕМЫЙ ЯЗЫК РАЗМЕТКИ XML

2.4.1. Основные понятия

Простота и ограниченность HTML, являющегося самым популярным приложением SGML, создавали трудности при описании сложных информационных объектов, поиске необходимой информации, создании приложений, обменивающихся данными через Интернет. Поэтому в 1996 году была сформирована рабочая группа W3C, основной задачей которой являлось создание нового языка разметки. Этот язык должен был включать в себя гораздо больше возможностей SGML, чем HTML, но в то же время оставаться подходящим для использования в WWW. Чуть позже этот язык стал известен как расширяемый язык разметки XML. Разработка нового языка разметки велась около двух лет, и в начале февраля 1998 года W3C утвердила в качестве рекомендации первую спецификацию XML — XML версии 1.0.

За сравнительно недолгий срок с момента своего появления на свет XML сумел завоевать огромную популярность среди разработчиков Интернет-технологий. Число созданных и разрабатываемых программных продуктов на основе XML, число компаний, включающих поддержку XML в свои уже готовые продукты, количество публикаций в компьютерной прессе уже весьма велико и продолжает расти.

Как и SGML, XML является метаязыком для формального описания прикладных языков разметки, предназначенных для кодирования структурированных документов. Спецификация XML определяет, как стандартным способом разметить документ, выделяя все семантически значимые компоненты.

При разработке нового языка разметки учитывались достоинства и недостатки уже существующих языков, а также то, что основным местом применения XML является Интернет. Основные требования к создаваемому языку были сформулированы следующим образом:

- XML должен быть годен к непосредственному применению в Интернете;
- XML должен быть совместимым с SGML (XML-документ должен одновременно являться и SGML-документом без внесения каких-либо изменений или дополнений);
- число необязательных свойств в XML должно быть минимальным, в идеале — нулевым (любая XML-программа должна уметь читать любой XML-документ);
- XML-документы должны быть легко читаемыми с помощью простейших текстовых процессоров;
- XML-разметка должна быть простой для понимания.

Формальное описание нового языка разметки состоит из нескольких взаимосвязанных частей:

– спецификации eXtensible Markup Language (XML) 1.0, которая определяет синтаксис языка;

– спецификаций XML Pointer Language (XPointer) и XML Linking Language (XLink), которые определяют стандартные механизмы установления связей между компонентами XML-документов;

– спецификации eXtensible Style Language (XSL), которая определяет механизмы для внешнего представления XML-документов.

2.4.2. Структура XML-документа

По своей структуре XML-документ очень похож на SGML- или HTML-документ. В качестве примера приведем уже знакомый нам текст начала главы 2.

Пример 2.5

```
<?xml version="1.0"?>
<Section>
  <head-of-section>2.1.  Разметка  документа</head-of-
section>
  <paragraph>Значительную  часть  сообщений,  несущих
информацию, человек предпочитает хранить в виде докумен-
тов...</paragraph>
</Section>
```

Существует несколько основных правил составления XML-доку-мента. Каждый документ начинается с пролога. В данном случае — это инструкция `<?xml version="1.0"?>`, которая является XML-декла-рацией. Ее наличие идентифицирует XML-документ и указывает, какой версии XML он соответствует.

В данном примере нет указания на используемое определение типа документа (DTD), так как в отличие от SGML XML не требует обязательного определения DTD для каждого документа. При необходимости описание или указание на месторасположение DTD также помещается в прологе документа. За прологом следует тело до-

кумента, которое представляет собой жесткую структуру элементов, подчиняющихся принципу вложенности. Именование элементов либо соответствует объявленному DTD, либо произвольно.

Обязательным является наличие как открывающего, так и закрывающего маркера в каждом элементе, ибо без этого при отсутствии DTD определить структуру документа невозможно.

Каждый из элементов может по аналогии с SGML содержать атрибуты, предназначенные для более детального описания семантически однородных элементов.

Возможно наличие пустых элементов, то есть элементов без содержимого. Такие элементы обозначаются с помощью символа «/» перед закрывающей угловой скобкой, например <Empty-Marker/>.

В общем случае XML-документ может иметь шесть типов компонентов:

1) элементы;
2) ссылки на текстовые или бинарные объекты (entity references);
3) комментарии;
4) инструкции обработки;
5) отмеченные разделы данных (CDATA sections);
6) декларация типа документов.

Мы не будем подробно останавливаться на всех типах компонентов. Отметим лишь, что инструкции обработки, в соответствии со своим названием, предназначены для предоставления информации программам, которые будут в дальнейшем обрабатывать документ. Тип документа определяется тем же способом, что и в SGML, а отмеченные разделы данных позволяют передавать размещенные в них данные или текст «как есть» без анализа его структуры.

Что можно сказать про структурную и семантическую корректность разметки? Необязательность определения DTD, с одной стороны, существенно облегчает XML-разметку, но, с другой стороны, может значительно усложнить программы обработки. Каким образом определить в данном случае корректность XML-документа?

Чтобы определить класс правильно составленных (с точки зрения XML) документов, вводятся понятия структурной и синтаксической корректности. XML-документ является структурно корректным, если он отвечает следующим требованиям:

- конструкция документа должна отвечать общим правилам составления документа, определенным в спецификации. В частности, некоторые конструкции (например, инструкция `<?xml versi-on="1.0"?>`) могут присутствовать только в определенных местах документа;
- никакой атрибут не используется более одного раза в одном маркере элемента;
- значения атрибутов не ссылаются на внешние объекты;
- все непустые элементы удовлетворяют принципу вложенности;
- все используемые объекты продекларированы;
- нет ссылок на бинарные объекты непосредственно из текста. Такие ссылки возможны лишь в момент декларации объекта;
- текстовые объекты не являются рекурсивными.

По определению, если документ не является структурно корректным, то он не является и XML-документом.

При наличии у документа DTD возможна его проверка на синтаксическую корректность. При этом XML-документ считается синтаксически корректным, если он, во-первых, является структурно корректным, а во-вторых, полностью соответствует всем правилам, изложенным в соответствующем DTD.

2.4.3. Ссылки в XML-документах

Для языка разметки с непредопределенными названиями элементов и даже отсутствующим иногда DTD невозможно определить стандарт на механизм связывания через элементы. Напротив, ссылающиеся и указуемые объекты должны иметь специальные атрибуты, которые идентифицируют их в этом качестве.

Все элементы XML имеют специально зарезервированный атрибут Link. Присутствие этого атрибута в элементе определяет наличие ссылки, а значение атрибута указывает, какой тип ссылки в данном месте используется. В XML в отличие от HTML возможно создание не только однонаправленных гипертекстовых ссылок по типу «один-к-одному», но и двунаправленных ссылок. Используя HTML и переходя по стандартной гипертекстовой ссылке на новую страницу, пользователь имеет только одну возможность перехода назад — нажатием кнопки «Back» в Web-навигаторе. При использовании двунаправленных ссылок пользователь не только может вернуться по ссылке в то место, откуда пришел, но и перейти на те страницы, которые ссылаются на указуемый объект.

При использовании ссылок с возможностью перехода к одному или нескольким объектам («один-ко-многим») у пользователя появляется возможность выбора между несколькими точками назначения при использовании всего одной ссылки. Гипертекстовые ссылки в HTML опираются только на файловую адресацию информации, а XLink предоставляет возможность динамического изменения адресов местонахождения информации с помощью баз данных. То, что произойдет при переходе по ссылке, определяется атрибутом SHOW, который может иметь одно из следующих значений: EMBED, REPLACE, NEW.

В первом случае указуемый объект будет импортирован в то место, откуда идет ссылка. Это произойдет либо при показе документа, либо при его обработке. Такой подход может быть полезен при вставке некоторого текста из другого файла или для вставки картинки внутрь текста. При этом возможна как автоматическая подстановка объекта, так и ручная, требующая от пользователя некоторых действий.

Во втором случае ссылающийся объект будет заменен на указуемый. Это может быть полезным, например, при наличии двух вариантов некоторого компонента. При помощи этого механизма возможен просмотр обеих версий или обработка по выбору в зависимости от наличия тех или иных инструкций обработки.

В последнем случае исходный объект исчезает, и происходит полный переход к указуемому объекту. Такой механизм реализован в обычных гипертекстовых ссылках, когда при переходе по ссылке на экране отображается новая HTML-страница.

Механизмы ссылок и адресации в XML описываются в трех спецификациях W3C: XPath, XPointer и XLink. XLink описывает механизмы связывания: организацию многонаправленных и однонаправленных ссылок между ресурсами, аннотированных ссылок и внешних наборов ссылок.

XPath поддерживает спецификацию месторасположений в XML-документах в терминах элементов и списков элементов и является языком для адресации частей XML-документа. XPath использует компактный не-XML-синтаксис для возможности применения XPath внутри указателей ресурсов и значений атрибутов XML. Он оперирует с абстрактной структурной моделью документа, а не с его синтаксическим выражением и моделирует документ как некое дерево элементов для осуществления навигации по иерархической структуре XML-документа.

XPointer, построенный на основе XPath, определяет язык для поддержки адресации внутренних структур XML-документа. В частности, он позволяет определять ссылки к элементам, символьным строкам и другим объектам вне зависимости от того, имеют ли они универсальный атрибут идентификации (ID). Для этого XPointer использует структуру документа и возможность выбора его частей на основе типов элементов, значений атрибутов, относительного расположения, содержания или порядка следования элементов. Так, например, возможно ссылаться на первый абзац третьей главы. И даже после осуществления другой разбивки на главы и абзацы или добавления нескольких абзацев впереди указуемого эта ссылка все равно приведет нас именно к первому абзацу третьей главы. Это свойство чрезвычайно полезно, если нужно сослаться на некоторое место в чужом документе, к которому нет прав на редактирование, а автор не расставил меток на требуемом участке документа.

XLink определяет конструкции, которые могут быть использованы в XML-документах для описания связей между объектами. XLink может описывать простые ссылки (simple links), которые очень похожи на гипертекстовые ссылки в HTML, но при этом они могут быть более детально классифицированы, например по типу приложения, которое будет использовать данный документ. Кроме того, XLink в комбинации с XPointer может описывать расширенные ссылки (extended links) и указуемые точки объекта, являющиеся многонаправленными по своей природе.

Косвенные ссылки (indirect links) позволяют упростить сопровождение большого количества документов. Рассмотрим случай, когда необходимо изменить месторасположение какого-нибудь документа, скажем, перенести его в другой каталог или на другой сервер. При использовании простых гипертекстовых ссылок это привело бы к необходимости найти все места во всех документах, откуда определены ссылки на перемещаемый документ, и внести соответствующие исправления. При этом разрешения на редактирование ко всем документам получить практически нереально. Потребуются просто титанические усилия, чтобы осуществить задуманное, поэтому документ, на который ссылаются другие документы, становится неперемещаемым по файловой системе. С использованием косвенных ссылок все оказывается гораздо проще. Связывание документов происходит не напрямую, а через некоторый промежуточный файл ссылок. При изменении месторасположения документа необходимо лишь внести изменения в этот промежуточный файл, а все остальные документы при этом останутся нетронутыми.

2.4.4. Отображение документов

Используя XML, автор документа может самостоятельно определять тот набор элементов, который наиболее точным образом будет соответствовать его структурным компонентам. Но свобода выбора имеет свои неудобства — набор используемых элементов не обладает предопределенной семантикой. Для совместной работы с XML-документами необходим стандартный механизм получения внешнего представления. Таким механизмом для XML является расширяемый язык стилей XSL (eXtensible Style Language) [3].

Обычные таблицы стилей, используемые, например, для работы с HTML, содержат набор инструкций, которые говорят программе (Web-навигатору, текстовому редактору или процессору печати), каким образом преобразовывать структуру документа во внешнее представление. При этом таблицы стилей, как правило, содержат такие инструкции:

- отображение гипертекстовых связей синим цветом;
- выполнение условия — каждая новая глава должна начинаться с новой страницы;
- ведение сквозной нумерации рисунков по всему документу.

Необходимо понимать, что использование или наложение стиля — это не что иное, как преобразование исходного документа к требуемому виду. Документ, отображаемый на экране, и документ, написанный и раз-

меченный автором, — это совсем не одно и то же. Степень трансформации может меняться в зависимости от презентационных целей — страница документа для публикации в Интернете и для высококачественной полноцветной полиграфической печати должна обрабатываться по-разному, но в любом случае требуется некоторое преобразование.

Использование языков разметки с предопределенной семантикой позволяет существенно упростить реализацию таблицы стилей. Программа, обрабатывающая, например, размеченную таблицу, может отобразить ее различным способом, но она заранее, даже без использования таблицы стилей, знает, что обрабатываемый объект является таблицей.

В случае использования XML-разметки XSL не только должен определять, каким образом тот или иной элемент будет отображаться, ска-жем, на экране, но и каким объектом он будет в итоге являться. Для того чтобы передать содержание XML-документа наиболее эффективным образом, необходимы два компонента: стандартный язык, описывающий требуемую разметку на выходе (в XSL это форматирующие объекты — formatting objects), и средство для преобразования исходного документа к требуемой разметке (в XSL это язык трансформации — transformation language). XSL включает стандартный словарь форматирующих объектов с хорошо определенными свойствами для осуществления контроля. Эти форматирующие объекты, такие как страница, блок текста, таблица, список и другие, позволяют авторам стилей получать высококачественное внешнее представление.

Работа с XML начинается с обработки исходного текста программой-анализатором (parser). Эта программа проверяет структурную и синтаксическую корректность XML-документа и создает дерево элементов исходного документа. Далее вступает в действие XSL-процессор, который в качестве исходных данных берет построенное дерево и соответствующий стиль. Шаг за шагом, начиная с корневого элемента, XML-процессор по шаблону, определенному в таблице стилей, обрабатывает всю структуру документа. Получающееся в результате дерево элементов может состоять из форматирующих объектов, которые и описывают внешнее представление документа. Форматирующие объекты представляют собой описание, не зависимое от устройства представления, и, следовательно, конечный документ может быть использован различными устройствами вывода.

Возможна и альтернатива форматирующим объектам. Так, в случае необходимости преобразования к HTML-виду вместо форматиру-ющих объектов будут использованы элементы языка разметки HTML. При этом результирующий документ будет выглядеть очень похожим на HTML-документ и обрабатываться стандартными Web-навигато-рами. Однако следует понимать, что любое XSL-преобразование XML-документа в результате даст тоже XML-документ [2].

Основными преимуществами XSL над другими механизмами наложения стилей, помимо возможности работы с элементами непредопределенной семантики, являются:

- возможность изменения порядка следования элементов в результирующем документе;
- возможность сортировки и сравнения элементов текста (список используемых терминов, упомянутых авторов);
- повторная обработка некоторых элементов (например, для печати разными стилями названия главы в начале страницы, в колонтитуле, оглавлении);
- возможность генерации вспомогательного текста («Глава», «Оглавление», «Список иллюстраций» и т. п.);
- подавление вывода некоторого текста (удаление редакторских примечаний или вывод только предисловия, а не полного документа).

2.5. СРЕДСТВА РАБОТЫ С ЯЗЫКАМИ РАЗМЕТКИ

2.5.1. Анализаторы структурированных документов

Основное назначение SGML-документов — хранение структурированной информации. Для извлечения информации из документов и передачи ее прикладной программе используются программы-ана-лизаторы (parsers).

Программа-анализатор по исходному документу строит древовидную структуру с ассоциированной мета-информацией о назначении элементов-узлов. Эта структура используется в дальнейшем прикладными программами. В качестве примера приложения можно привести программу просмотра, которая, применяя к дереву элементов некоторый стиль отображения, выводит документ на экран в удобном для чтения виде. Другим примером является программа преобразования дерева в HTML-формат, понятный обычному Web-навигатору.

Анализаторы используются также для проверки структурной и синтаксической корректности XML-документов. В последнем случае необходимо наличие соответствующего DTD.

Одним из лучших SGML-анализаторов является свободно распространяемый написанный на языке C анализатор SP, который используется как для построения древовидной структуры элементов, так и для проверки синтаксической корректности разметки. Основными его достоинствами являются поддержка практически всех используемых в настоящее время аппаратно-программных платформ, высокое быстродействие и доступность в исходных текстах, что позволяет при необходимости модифицировать и встраивать его в новые приложения.

Для обеспечения мобильности многие анализаторы написаны на языке Java. Это, например, msxml, предназначенный для разбора XML-документов с последующим просмотром их структуры в Internet Exp-lorer 5, а также XML for Java, созданный в корпорации IBM.

2.5.2. Редакторы структурированных документов

Adept Editor

Редактор Adept Editor компании Arbortext предназначен для создания и редактирования SGML- и XML-документов. Adept Editor состоит из собственно редактора текста документа, модуля для редактирования таблиц, редактора математических формул. Дополнительно Adept Editor может включать в себя модули проверки орфографии на 15-ти языках.

В комплект поставки Adept Editor входит стандартный набор DTD для создания деловых документов, файлов в формате HTML и технической документации. При необходимости возможно использование любых имеющихся у пользователя корректных SGML- или XML-DTD или определение собственных типов документов с помощью дополнительного продукта Arbortext Architect.

Внешний вид редактора и его функциональное наполнение соответствуют тому, что пользователи привыкли получать от традиционных текстовых редакторов. Многооконность, многофункциональная кнопочная панель, использование режима «cut-and-paste» дополнено, однако, специфическими особенностями, связанными с работой со структурированными данными. Так, например, Adept Editor позволяет редактировать документ с одновременным предварительным просмотром его внешнего представления. Для этого используется набор стилей и возможность быстрой фиксации произведенных изменений. Существует возможность редактирования как в режиме WYSIWYG (в том числе при работе с таблицами и формулами), так и в режиме полного показа маркеров разметки.

При вставке блока информации в редактируемый текст Adept Editor проводит автоматическую проверку правильности получаемой структуры. В случае, если будет обнаружено несоответствие структуры получаемого текста заданному DTD, то либо недостающая структура будет сформирована автоматически, либо в копировании информации будет отказано.

Предусмотрены специальные функции Quick Tag и Tag Prompt. Quick Tag позволяет в любом месте редактируемого документа получить список допустимых в данном месте элементов в соответствии с используемым DTD. Tag Prompt автоматически выделяет пустые элементы в документе, сообщая о пропуске смысловой информации.

Отдельно стоит отметить возможность использования командной строки для осуществления операций над редактируемым текстом, поддержку Unicode, поддержку таблицы формата CALS, SGML Open Ex-change и ATI, а также наличие встроенного языка Adept Common Lan-guage (ACL). С помощью ACL можно автоматизировать работу пользователя с самим редактором, а также интегрировать Adept Editor с другими офисными приложениями.

Adept Editor позволяет открывать для редактирования файлы с некорректной разметкой для устранения ошибок и приведения текста к виду, соответствующему используемому DTD.

Недостатками Adept Editor являются его высокая стоимость, требовательность к ресурсам памяти компьютера и отсутствие поддержки операционной системы Linux.

XMetaL

Продукт XMetaL компании SoftQuad представляет собой весьма изящный редактор XML-документов, очень простой в использовании. Его внешняя схожесть со стандартными редакторами текстов делает очень простыми процессы обучения пользователей и внедрения нового программного средства в качестве рабочей среды.

Широкие возможности настройки свойств редактора под любой используемый DTD без какого-либо программирования позволяют превратить XMetaL в удобный инструментарий автора. Проверка соответствия используемому DTD происходит «на лету», и в случае обнаружения ошибки в разметке происходит автоматическое переключение в режим плоского текста для предоставления автору возможности коррекции разметки.

Несомненным удобством является возможность работы с тремя видами отображения документа. Стандартным и наиболее принятым для обычных текстовых редакторов является отображение WYSIWYG. Редактирование в режиме плоского текста может быть особенно удобным для авторов, привыкших все править своими руками. И наконец, режим работы с маркированным текстом сочетает возможности стандартных тестовых редакторов и быстроту доступа к метаинформации: элементам и их атрибутам.

Встроенный в редактор Инспектор Атрибутов, доступный при любом отображении текста, позволяет проверять корректность разметки и предоставляет список атрибутов, доступных в точке редактирования. При удалении или вставке некоторого блока информации пользователь автоматически обращается к Инспектору Атрибутов, который проверяет корректность действия с учетом DTD.

Весьма полезными свойствами XMetaL являются возможность проверки орфографии и расширенные механизмы поиска и замены как текста, так и элементов документа.

Менеджер ресурсов предоставляет автору документа расширенные возможности объектной работы с изображениями, текстовыми фрагментами, стилями представления, шаблонами документов вне зависимости от того, где расположены эти объекты — на жестком диске, в локальной сети или Интернете.

Использование Database Import Wizard дает возможность с помощью ODBC получать доступ к базам данных. XMetaL позволяет осуществлять запросы к базам данных, создавать персональную библиотеку таких запросов для их повторного применения. Он поддерживает сложные запросы к объединениям таблиц данных. Вставка запрошенной информации осуществляется в виде CALS- или HTML-таблиц, а также в виде отдельных элементов XML.

Поддержка каскадных таблиц стилей (Cascading Style Sheets, CSS) заключается в возможности не только немедленного отображения текста с использованием стиля во время редактирования, но и при необходимости редактирования самих CSS-файлов с помощью встроенного редактора.

Несомненными достоинствами XMetaL являются нетребовательность к ресурсам компьютера, возможность получения пробной версии редактора по Интернет. Однако XMetaL работает только под Windows (95/98/NT).

2.5.3. Просмотрщики

При создании XML-документов с помощью специализированных редакторов, как правило, удобно пользоваться специальными средствами просмотра, встроенными непосредственно в редактор. Если же по каким-либо причинам подобный подход не годится, то для просмотра можно найти отдельный программный продукт. Правда, выбор тут не особенно велик, что в первую очередь связано с отсутствием ряда утвержденных стандартов на стили отображения (например, спецификация XSL все еще находится в стадии разработки).

Mozilla создается и распространяется по модели Open Source, что означает доступность исходных текстов продукта для всех желающих. В текущую версию M11 включена полная поддержка основных стандартов (HTML 4.0, XML, CSS, DOM). Кроме того, сама программа является небольшой по объему, быстрой и модульной. Mozilla включает XML-анализатор Expat, поддерживает просмотр XML + CSS, а также простые ссылки XLink.

XML Viewer for Java является приложением, целиком написанным на языке Java, которое позволяет просматривать любые XML-документы. Пользователь может перемещаться по документу, представленному в виде дерева, находить атрибуты отдельных элементов и просматривать как исходные тексты XML-документов, так и соответствующие DTD. Возможно также одновременное отображение некоторого элемента исходного текста XML-документа и определение этого элемента, указанного в DTD.

***Internet Explorer* 5.0** (IE 5.0) является программным продуктом, который может отображать XML-документы с использованием стилей CSS. XML-документы могут быть показаны и в виде иерархической структуры элементов, и без использования специальных стилей. Поддержка XSL заключается в возможности осуществления преобразований и пока не включает поддержку словаря форматирующих объектов. IE 5.0 также осуществляет поддержку векторного языка разметки (Vector Markup Language, VML).

2.5.4. Пакетные средства

Одним из достоинств работы с размеченными документами является возможность их пакетной обработки. В качестве примера можно привести программный продукт Jade. Он представляет собой реализацию обработки разобранных структурированных документов с помощью языка описания стилей DSSSL (Document Style Semantics and Specification Language). Это первый (свободно распространяемый с исходными текстами) продукт подобного класса.

Jade доступен для большинства UNIX-платформ, существуют его реализации под Windows. На выходе он позволяет получать файлы в форматах RTF, TeX, HTML и MIF [2].

2.6. ОБЛАСТИ ПРИМЕНЕНИЯ XML

Кратко опишем некоторые проекты, где используются либо XML, либо производные от него языки.

Компания Zedak Corp., являясь агентом The New York Times, разработала систему автоматической подписки на рекламу. Эта система использует специально разработанный на основе XML язык разметки AdMarkup и большое количество стандартизированных DTD. Система позволяет рекламным агентствам размещать заказы на рекламу через Интернет, предоставляя издателю информацию о тексте и изображениях рекламы, месте размещения рекламы, уникальный номер рекламодателя, предварительную дату устаревания рекламы, а также назначение данного заказа (создание нового рекламного модуля, отмена существующего, обновление или предварительное согласование). Кроме того, система позволяет получать информацию о текущем состоянии счета рекламодателя, специальных требованиях по размещению рекламы, географических регионах публикации и т. п. Система была с успехом внедрена в большом количестве рекламных агентств, сотрудничающих с The New York Times.

Одним из направлений использования языков разметки является электронная коммерция. Во-первых, необходимость предоставления потенциальному покупателю информации о товаре в структурированном виде сразу наводит на мысль об использовании языков разметки. Так, например, специально созданная библиотека Commerce One's Com-mon Business Library представляет собой набор стандартизованных описаний заказов на покупку, счетов, описаний товаров и графиков поставки. Во-вторых, осуществление финансовых транзакций через Интернет должно основываться на полной стандартизации используемых механизмов. Чтобы помочь развитию в этом направлении, был учрежден проект Bank Internet Payment System, поддерживаемый консорциумом ведущих банков, направленный на поощрение компаний, использующих платежные транзакции через Интернет. Основная часть финансовой информации в системах, создаваемых в рамках данного проекта, передается с использованием структурно размеченных документов.

J.P. Morgan&Co, PricewaterhouseCoopers и IBM ведут совместные разработки по созданию FpML (XML-based Financial Products Markup Language), основанного на XML языка разметки для финансовых применений. Спецификация FpML позволит осуществлять интеграцию широкого спектра финансовых услуг — от электронной торговли до анализа рисков.

Для предоставления информации об объектах недвижимости и осуществления сделок на рынке недвижимости OpenMLS и 4thWORLD Telecom разработали набор DTD, помогающий стандартизировать информационный обмен между покупателем и продавцом. Этот набор включает Commercial DTD, Vacant Land DTD, Working Land DTD и Residential Listing DTD.

Большое количество проектов осуществляется в сфере науки. Например, для осуществления обмена стандартизованными данными на основе XML в астрономии был создан язык разметки AML (Astronomical Markup Language). AML поддерживает работу со следующими объектами: статьями, таблицами, наборами таблиц, изображениями астрономических объектов, персоналиями. Это означает, что все эти объекты могут быть описаны с помощью единого языка разметки, что облегчает установление связей между объектами и создание программных продуктов, поддерживающих все перечисленные объекты в рамках единого пользовательского интерфейса. Другим примером является создание математического языка разметки MathML (Mathematical Markup Language).

2.7. ПРИМЕР РАЗРАБОТКИ ПРОЕКТА С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА СТРУКТУРНОЙ РАЗМЕТКИ

2.7.1. Постановка задачи

В качестве примера использования структурного языка разметки рассмотрим проект создания Web-версии информационного журнала Jet Info — Jet Info Online [www.jetinfo.ru].

Идея создания Web-версии технического издания, разумеется, не нова. Однако реализация этой идеи оказалась не совсем простым делом в связи с задачами, которые были поставлены перед участниками проекта [2].

Во-первых, необходимо было обработать весь накопленный материал (а Jet Info издается с 1993 года) и привести его к единому виду.

Во-вторых, требовалось реализовать функциональные модули электронного издания, отвечающие за контекстный и атрибутный поиск, организацию тематической и хронологической иерархии материала. Большого размера статьи и разделы должны быть представлены в удобном для «пролистывания» виде.

В-третьих, предстояло создать единый дизайн журнала.

И наконец, в проекте должна быть реализована технология, позволяющая минимизировать расходы на сопровождение журнала.

Все поставленные задачи требовалось решить в сжатые сроки, поэтому возникла еще одна организационно-техническая задача — распараллеливание работы участников проекта. Это означало, что стандартный подход к реализации подобных проектов — создание и утверждение дизайна, написание функциональных модулей и лишь затем HTML-разметка документов — не подходил. Наиболее объемную часть проекта — разметку накопленного материала — необходимо было начать как можно раньше.

Было принято естественное решение — разделить вопросы HTML-представления (дизайн издания) и структурную разметку материала, используя для последней SGML-средства. Страницы, передаваемые Web-навигатору, должны создаваться по определенным правилам путем преобразования исходных SGML-документов.

2.7.2. Выбор проектных решений

Выбор средств структурной разметки предопределил построение всей дальнейшей работы над проектом. Все было осуществлено в рамках традиционного использования SGML-технологии:

- выбор DTD, который определяется типом размечаемых документов;
- разметка исходных текстов в соответствии с выбранным DTD;

• контроль корректности осуществляемой разметки с помощью стандартных SGML-анализаторов;

• параллельное определение дизайна издания;

• выбор механизма преобразования (процессора) и стиля преобразования SGML-документов в HTML-формат;

• преобразование структурно размеченных документов к HTML-представлению.

Выбранный подход, опирающийся на SGML-технологию, имеет ряд важных достоинств.

Во-первых, структурная разметка имеет достаточно формальный характер и позволяет описывать все компоненты документов наиболее естественным образом. Статья может быть обозначена маркером article, автор — author, термин — term и т. д. Кроме того, корректность размеченного документа может быть автоматически проверена с помощью программ-анализаторов, что является первым рубежом контроля качества разметки.

Во-вторых, преобразование документов в формат HTML производится программно. Следовательно, качество внешнего представления не зависит от художественных способностей «разметчиков». К то-му же внешним представлением можно управлять с помощью стилей.

В-третьих, в результате структурной разметки получается полноценный электронный архив документов, имеющих формальную структуру и пригодных для любого автоматического преобразования, в том числе и для преобразования к виду, соответствующему требованиям к оригинал-макету издаваемой печатной продукции.

Наконец, решающим соображением в пользу использования SGML-технологии стало то, что при таком подходе можно формализовать всю технологическую цепочку публикации. При этом существенны как доступность средств реализации всех операций, так и наличие опыта работы с SGML у участников проекта.

Поскольку решаемая задача и используемые подходы являются типичными, полезно проследить, как проходила работа по реализации проекта.

Выбор DTD имеет очень большое значение для всей последующей работы. Неудачный выбор DTD может привести к тому, что семантика документов будет неадекватно отражаться в их структуре, разметка будет неочевидной, искусственной, а преобразование в форматы для внешнего представления — некачественным и негибким.

При выборе DTD для Jet Info Online рассматривалось два варианта:

1) создание собственного DTD;

2) выбор из DocBook и TEILite.

В данном случае на создание собственного DTD у авторов проекта не было ни временных, ни людских ресурсов. DTD легко и просто созда-вать тогда, когда документы имеют простую структуру, которая к тому же является фиксированной и в дальнейшем не будет претерпевать изменений. Jet Info — издание развивающееся, поэтому предусмотреть характер будущих публикаций и составляющих их элементов невозможно.

Статьи, публикуемые в Jet Info, имеют весьма развитую структуру. В них присутствуют многоуровневые разделы, приложения, библиографии, эпиграфы, листинги, списки терминов, рисунки, таблицы и многое другое. Естественно, хотелось, чтобы вся эта структура была отражена без потери семантики.

Не стоит забывать и о том, что, если встать на путь создания собственного DTD, придется разрабатывать новые стили для различных способов отображения структурированной информации, что займет немало времени. В силу перечисленных причин авторами проекта было принято решение сориентироваться на один из двух наиболее часто используемых DTD общего назначения — DocBook или TEILite. Они длительное время применяются в компании «Инфосистемы Джет» при подготовке различных документов, для них существуют процессоры и стили, как стандартные, так и специально разработанные для нужд Компании [2].

Было учтено, что DocBook больше ориентирован на документы технического характера, а TEILite — на художественную прозу. К тому же личные симпатии большинства участников проекта были на стороне DocBook. Все это и предопределило сделанный выбор.

2.7.3. Стиль и процессор

Выбрав DocBook в качестве DTD, авторы проекта практически лишились вариантов при выборе стиля и процессора, так как разработка собственных стилей для DocBook — задача весьма трудоемкая. К их счастью существовал готовый, качественный и сопровождаемый автором стиль — Modular DocBook StyleSheet. Этот стиль реализован с помощью языка спецификации стиля и семантики документов (DSSSL). Он позволяет преобразовывать исходные документы как в HTML, так и в TeX или RTF, что дает дополнительную возможность получения качественного представления бумажных копий размеченного материала. В качестве процессора был выбран Jade — свободно распространяемая реализация DSSSL.

2.7.4. Разметка материалов

На первом этапе была выполнена рутинная работа по «извлечению» текста из внутреннего формата существующих электронных версий статей и набору тех материалов, которые сохранились только в печатном виде.

Затем была проведена разметка полученных текстовых файлов в соответствии с DocBook DTD. В качестве примера ниже представлен размеченный текст начала одной из статей данного издания, послужившей основой для данного раздела пособия.

Пример 2.6

```
<article id="article1.1.2000" arch="corp-systems">
  <artheader>
    <title>Современное состояние языков и средств
разметки документов </title>
      <authorgroup>
        <author>
          <surname>Дуров</surname>
          <firstname>Илья</firstname>
        </author>
      </authorgroup>
  </artheader>
 <sect1>
  <title>Введение</title>
  <para>Так уж сложилось, что большую часть информа-
ции человек предпочитает хранить в виде докумен-
тов...</para>
```

При этом оказалось, что некоторые документы, например официальные Руководящие документы Гостехкомиссии России, не укладываются в выбранную модель. Для преодоления этой трудности была проведена модификация DocBook, который имеет соответствующие встроенные возможности. Так, например, был добавлен новый тип элемента «clause», который представляет собой нумерованную статью в разделе.

После осуществления разметки с помощью Jade и Modular Doc-Book StyleSheet было произведено преобразование материалов в HTML. В примере 2.7 представлен преобразованный в HTML текст начала статьи из предыдущего примера 2.6 (для экономии места из примера исключены разметка внешнего представления верхней части HTML-страницы и разметка оглавления).

Пример 2.7

```
<html version="-//W3C//DTD HTML 3.2 Final//EN">
  <head>
    <title>Современное состояние языков и средств
разметки документов</title>
  </head>
  <body text="black" bgcolor="white">
...
  <hr>
    <div class="article">
      <h1 class="title">Современное состояние языков
и средств разметки</h1>
      <i>Дуров Илья</i>
      <br>
  <hr>
...
  </div>
  <h1              class="sect1">              <a
name="aen8">Введение</a></h1>
  <p>Так уж сложилось, что большую часть информации
человек предпочитает хранить в виде документов...</p>
```

На этом этапе авторами проекта выявлялись смысловые ошибки, формальные ошибки разметки (несоответствия DTD) и недостатки во внешнем представлении документов.

Не обошлось и без некоторых трудностей. В первую очередь к ним можно отнести фиксированную структуру получаемого журнала, которую диктует DocBook. Дело в том, что в рамках используемого DTD статьи можно объединить в книги, которые в свою очередь можно объединить в наборы книг (Bookset). Это, с одной стороны, удобно, так как можно оставаться в рамках одного DTD и реализовывать необходимую иерархию материалов. С другой стороны, используемый стиль, хотя и обладает большим количеством настраиваемых параметров (что позволяет управлять внешним видом получаемых HTML-до-кументов), все же недостаточно гибок для того, чтобы, используя упомянутую иерархию, получить традиционный, привычного вида, электронный журнал.

Один из вариантов решения этой проблемы заключался в том, чтобы на верхних уровнях иерархии отказаться от DocBook в пользу HTML, обеспечивая для них традиционный внешний вид при сохранении всех преимуществ структурной разметки и автоматического форматирования содержательных материалов.

Технически это было реализовано в виде набора HTML-шабло-нов, в которые с помощью специальных (не входящих в HTML) элементов и атрибутов вставляются ссылки на размеченные статьи.

Для обработки этих шаблонов был разработан специальный стиль, назначение которого состояло в том, чтобы, оставляя неизменным содержание шаблона, обрабатывать ссылки на статьи и подставлять вместо них правильные ссылки на HTML-версии статей. С точки зрения SGML-технологий это стандартный прием включения в основной документ подчиненных документов, имеющих, возможно, другой DTD.

Опыт осуществления проекта Jet Info Online показывает, что выбранная SGML-технология хорошо подходит как для Web-публикации большого количества накопленного материала, так и для организации работы по подготовке новых статей к публикации и типографским способом, и в WWW [2].

Использование набора HTML-шаблонов позволяет при необходи-мости изменять дизайн Web-издания, заменив лишь несколько файлов и не затрагивая самого сложного — технологию подготовки материалов.

Использование структурной разметки позволяет создать полноценный архив документов — текстовую базу данных. Как следствие, создание подходящей поисковой машины и встраивание ее в Web-издание становится делом несложным. Ее возможности не ограничиваются только контекстным поиском. Можно осуществлять поиск по таким параметрам, как автор, тема, год издания. Причем перечень параметров поиска в дальнейшем может быть легко расширен.

Сопровождение Web-издания из искусства превращается в рутинное занятие, благодаря тому что формирование публикуемой электронной версии по размеченному документу осуществляется автоматически. Также автоматически оказываются «дополнительные услуги»: первые абзацы новых статей выносятся на корневую страницу, соответствующим образом модифицируются тематический и хронологический каталоги.

В заключение еще раз отметим, что мощность и универсальность SGML не только предоставляют широкие возможности для структурной разметки документов, но и создают значительные трудности при реализации приложений и, как следствие, использовании языка разметки в Интернете. Напротив, HTML идеально приспособлен для работы в Интернете, но его простота и легкость не позволяют в полной мере реализовать всю необходимую на сегодняшний день функциональность.

Разработка XML стала важным этапом поиска разумного сочетания полноты структурного описания и широких возможностей создания прикладных программ на его основе.

Контрольные вопросы

1. Что понимается под разметкой документа?
2. Назовите особенности структурной разметки документа.
3. Что необходимо иметь для эффективной работы с языками разметки?
4. На каких основных положениях основывается разметка, определяемая в SGML?
5. Приведите достоинства SGML.
6. Что определяет DTD?
7. Чем отличается HTML от SGML?
8. Что привело к созданию XML?
9. Чем определяется структура XML-документа?
10. Как определяются ссылки в XML-документах?
11. При решении каких задач целесообразно применять XML?

3. ИНФОРМАЦИОННЫЕ СИСТЕМЫ ОБРАБОТКИ ДАННЫХ

3.1. ОСНОВНЫЕ КЛАССЫ ИНФОРМАЦИОННЫХ СИСТЕМ

В 60-х годах XX в. была осознана необходимость применения средств компьютерной обработки хранимой информации там, где были накоплены значительные объемы полезных данных, — в военной промышленности, в бизнесе. Появились автоматизированные информационные системы (АИС) — программно-аппаратные комплексы, предназначенные для хранения, обработки информации и обеспечения ею пользователей. Первые АИС работали преимущественно с информацией фактического характера, например с характеристиками объектов и их связей. По мере «интеллектуализации» АИС появилась возможность обрабатывать текстовые документы на естественном языке, изображения и другие виды и форматы представления данных.

Несмотря на то что принципы хранения данных в системах обработки фактической и документальной (текстовой) информации схожи, алгоритмы обработки в них заметно различаются. Поэтому в зависимости от характера информационных ресурсов, которыми оперируют такие системы, принято различать два крупных класса АИС — документальные и фактографические [6].

Документальные системы служат для работы с документами на естественном языке — монографиями, публикациями в периодике, сообщениями пресс-агентств, текстами законодательных актов. Они обеспечивают их смысловой анализ при неполном, приближенном представлении смысла. Наиболее распространенный тип документальных систем — информационно-поисковые системы (ИПС), предназначенные для накопления и поиска по различным критериям документов на естественном языке.

Фактографические системы оперируют фактическими сведениями, представленными в виде специальным образом организованных совокупностей формализованных записей данных. Центральное функциональное звено фактографических информационных систем — системы управления базами данных (СУБД). Фактографические системы используются не только для реализации справочных функций, но и для решения задач обработки данных. Под обработкой данных понимается специальный класс решаемых на ЭВМ задач, связанных с вводом, хранением, сортировкой, отбором и группировкой записей данных однородной структуры. В большинстве случаев эти задачи предусматривают предоставление пользователям итоговых результатов обработки в виде отчетов табличной формы.

Задачи, связанные с обработкой данных, широко распространены в любой деятельности. На их основе ведут учет товаров в супермаркетах и на складах, начисляют зарплату в бухгалтериях и т.д. Невозможно представить себе деятельность современного предприятия или учреждения без использования АИС. Эти системы составляют фундамент информационной деятельности во всех сферах, начиная с производства, управления финансами и телекоммуникациями и заканчивая управлением семейным бюджетом.

Массивы информации, накопленные в АИС, должны быть оптимальным образом организованы для компьютерного хранения и обработки, должна обеспечиваться их целостность и непротиворечивость. Используя функции стандартных файловых систем, невозможно добиться нужной производительности при решении подобных задач, поэтому все автоматизированные информационные системы опираются на СУБД.

Среди фактографических систем важное место занимают два класса: системы операционной обработки данных и системы, ориентированные на анализ данных и поддержку принятия решений.

Первые рассчитаны на быстрое обслуживание относительно простых запросов большого числа пользователей. Системы операционной обработки работают с данными, которые требуют защиты от несанкционированного доступа, нарушений целостности, аппаратных и программных сбоев. Время ожидания выполнения типичных запросов в них не должно превышать нескольких секунд. Сфера применения таких систем — системы платежей, резервирования мест в поездах, самолетах, гостиницах, банковские и биржевые системы. Логическая единица функционирования систем операционной обработки данных транзакция — это некоторое законченное с точки зрения пользователя действие над базой данных. В современной литературе для обозначения систем операционной обработки часто используют термин OLTP (On-Line Transaction Processing — оперативная обработка транзакций или выполнение транзакций в режиме реального времени) [7, 9].

Системы поддержки принятия решений (аналитические системы) ориентированы на выполнение более сложных запросов, требующих статистической обработки исторических (накопленных за некоторый промежуток времени) данных, моделирования процессов предметной области, прогнозирования развития тех или иных явлений. Аналитические системы также часто включают средства обработки информации на основе методов искусственного интеллекта, средства графического представления данных. Эти системы оперируют большими объемами исторических данных, позволяя выделить из них содержательную информацию — получить знания из данных.

Современные требования к скорости и качеству анализа привели к появлению систем оперативной аналитической обработки OLAP (On-Line Analysis Processing) [8, 9]. Оперативность обработки больших объемов данных в таких системах достигается за счет применения мощной, в том числе многопроцессорной вычислительной техники, сложных методов анализа, а также специальных хранилищ данных, накапливающих информацию из различных источников за большой период времени и обеспечивающих быстрый доступ к ней.

Оба класса систем основаны на СУБД, но типы выполняемых ими запросов сильно различаются. Например, в OLTP-системе продажи железнодорожных билетов допустим такой запрос: «Есть ли свободные места в купе поезда "Томск—Новокузнецк", отправляющегося 21 августа в 18.130?». В аналитической системе запрос может быть таким: «Каким будет объем продажи железнодорожных билетов в денежном выражении в следующие три месяца с учетом сезонных колебаний?». Принципиально отличаются и структуры баз данных для высокопроизводительных OLAP- и OLTP-систем. Эти отличия, а также особенности обработки данных в OLTP- и OLAP-системах будут рассмотрены далее.

3.2. ОСОБЕННОСТИ ОБРАБОТКИ ДАННЫХ В OLTP-СИСТЕМАХ

3.2.1. Обработка транзакций

Основной логической единицей функционирования систем операционной обработки данных является транзакция. Транзакцией называют неделимую с позиции воздействия на базу данных (БД) последовательность операции манипулирования данными. Транзакция может состоять из операции чтения, удаления, вставки, модификации данных. В OLTP-системах транзакция реализует некоторое осмысленное с точки зрения пользователя действие, например перевод денег со счета на счет в платежной системе банка, резервирование места в поезде системой оформления железнодорожных билетов.

Традиционно понятие «обработка транзакций» использовалось применительно к крупномасштабным системам обработки данных — системам, осуществлявшим международные банковские операции и др. Теперь ситуация меняется. Информационные системы в различных областях человеческой деятельности становятся все более распределенными и неоднородными, в них остро стоят проблемы сохранения целостности данных и разграничения доступа. Одно из направлений решения этих проблем — использование средств обслуживания транзакций в информационных системах.

Чтобы использование механизмов обработки транзакций позволило обеспечить целостность данных и изолированность пользователей, транзакция должна обладать четырьмя основными свойствами: атомарности (atomicity), согласованности (consistency), изолированности (isolation), долговечности (darability). Транзакции, обладающие перечисленными свойствами, иногда называют ACID-транзакциями по первым буквам их английских названий.

Свойство атомарности означает, что транзакция должна выполняться как единая операция доступа к БД. Она должна быть выполнена полностью либо не выполнена совсем, то есть выполняются все операции манипулирования данными, которые входят в транзакцию, либо, если по каким-то причинам выполнение части операций невозможно, ни одна из операций не выполняется. Свойство атомарности обычно коротко выражают фразой: «все или ничего».

Свойство согласованности гарантирует взаимную целостность данных, то есть выполнение ограничений целостности БД после окончания обработки транзакции. Следует отметить, что база данных может обладать такими ограничениями целостности, которые сложно не нарушить, выполняя только один оператор ее изменения. Например, если в отношении A хранится число кортежей отношения B, то добавить новый кортеж в отношение B, не нарушив ограничений целостности, невозможно. Поэтому такое нарушение внутри транзакции допускается, но к моменту ее завершения база данных должна быть в целостном состоянии. Несоблюдение в системах со средствами контроля целостности этого условия приводит к отмене всех операций транзакции.

В многопользовательских системах с одной БД одновременно могут работать несколько пользователей или прикладных программ. Поскольку каждая транзакция может изменять разделяемые данные, данные могут временно находиться в несогласованном состоянии. Доступ к этим данным другим транзакциям должен быть запрещен, пока изменения не будут завершены. Свойство изолированности транзакций гарантирует, что они будут выполняться отдельно друг от друга.

Свойство долговечности означает, что если транзакция выполнена успешно, то произведенные в ходе ее выполнения изменения в данных не будут потеряны ни при каких обстоятельствах [7, 8].

Результатом выполнения транзакции может быть ее фиксация или откат. Фиксация транзакции — это действие, обеспечивающее запись в БД всех изменений, которые были произведены в процессе ее выполнения. До того как транзакция зафиксирована, возможна отмена всех сделанных изменений и возврат базы данных в то состояние, в котором она была до начала выполнения транзакции. Фиксация транзакции означает, что все результаты ее выполнения становятся видимыми другим транзакциям. Для фиксации транзакции необходимо успешное выполнение всех ее операторов.

Если нормальное завершение транзакции невозможно, например нарушены ограничения целостности БД или пользователь выдал специальную команду, происходит откат транзакции. База данных возвращается в исходное состояние, все изменения аннулируются.

Механизм корректного отката и фиксации транзакций основан на использовании журнала транзакций. Для того чтобы иметь возможность сделать откат, СУБД должна сохранять все изменения, которые транзакция внесла в БД. Однако необходимости в том, чтобы каждый раз сохранять всю информацию базы данных, нет. Реляционные операции изменяют строки отношений БД, поэтому, чтобы обеспечить возможность отката, СУБД должна хранить те строки, которые были модифицированы. При выполнении любой операции, изменяющей базу данных, СУБД автоматически сохраняет в журнале транзакций состояние модифицируемых строк до опе-

рации и после нее. Только после этого изменения вносятся в БД. Если по окончании обработки транзакция фиксируется, то в журнале делается соответствующая отметка. Если же производится откат транзакции, то СУБД по журналу восстанавливает те строки отношений, которые были модифицированы, отменяя, таким образом, все изменения.

Для того чтобы оперировать транзакцией как единой логической единицей, СУБД должна уметь определять ее границы, то есть первую и последнюю входящую в нее операции. Стандарт языка SQL предусматривает следующий принцип выделения транзакции как некоторой законченной последовательности действий. Предполагается, что транзакция начинается с первого SQL-оператора, вводимого пользователем или содержащегося в прикладной программе. Все следующие далее операторы составляют тело транзакции. Тело транзакции завершается SQL-операторами COMMIT WORK или ROLLBACK WORK. Выполнение транзакции заканчивается также при завершении программы, которая сгенерировала транзакцию. Транзакция фиксируется, если ее тело оканчивается оператором COMMIT WORK либо при успешном завершении программы, сформировавшей транзакцию. Откат транзакции производится при достижении оператора ROLLBACK WORK, либо в случае, когда приложение, сгенерировавшее транзакцию, завершилось с ошибкой. Возможные варианты завершения выполнения транзакции представлены на рис. 3.1.

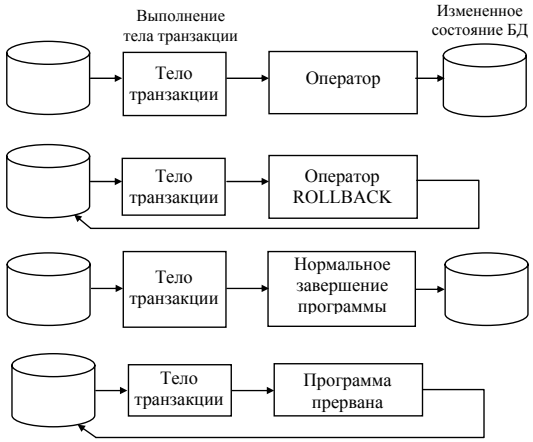


Рис. 3.1. Возможные варианты завершения транзакции

Рассмотрим пример транзакции, модифицирующей телефон (атрибут Phone) сотрудника с фамилией (атрибут Name) «Петров» в отношении Отдел (Department). Транзакция завершается фиксацией по достижении оператора COMMIT WORK: UPDATE Department SET Phone = "414-470" WHERE Name = "Петров" COMMIT WORK.

Некоторые диалекты языка SQL, например диалект, принятый в СУБД Sybase, включают специальные операторы, позволяющие производить промежуточную фиксацию транзакции. В теле транзакции могут быть определены точки, в которых сохраняется состояние базы данных. Откат в этом случае может производиться как к одной из точек промежуточной фиксации, так и к состоянию до начала выполнения транзакции. Точки промежуточной фиксации применяются в «длинных» транзакциях. Они позволяют разделить ее на несколько отдельных фрагментов.

Применение транзакций — эффективный механизм организации многопользовательского доступа к БД. Однако при реализации этого механизма СУБД приходится сталкиваться с целым рядом проблем. Во-первых, необходимо избежать потери изменений БД в ситуации, когда несколько программ читают одни и те же данные, изменяют их и пытаются записать результат на прежнее место. В БД могут быть сохранены изменения, выполненные только одной программой, результаты работы всех остальных программ будут потеряны. Во-вторых, требуется исключить возможность чтения незафиксированных изменений. Это может произойти в случае, когда одна транзакция вносит изменения в БД, они тут же считываются в другой транзакции, но затем другая транзакция прерывается оператором ROLLBACK WORK.

Чтобы избежать этих проблем, должна быть использована специальная дисциплина совместной обработки (сериализации) транзакций. В ее основе лежат следующие принципы:

1) транзакция не может получить доступ к незафиксированным данным, то есть к данным, в которых произведены изменения, но эти изменения еще не зафиксированы;

2) результат совместного выполнения транзакций должен быть эквивалентен результату их последовательного выполнения, то есть, если две транзакции выполняются параллельно, то полагается, что результат будет такой же, как если бы сначала выполнялась первая, а затем вторая транзакция или сначала вторая, а потом первая. В современных СУБД сериализация транзакций реализуется через механизм блокировок.

На время выполнения транзакции СУБД блокирует часть БД (отношение, строку или группу строк), к которой транзакция обращается. Блокировка сохраняется до момента фиксации транзакции. Если в процессе параллельной обработки другой транзакции делается попытка обратиться к блокированным данным, обработка транзакции приостанавливается и возобновляется только после завершения транзакции, заблокировавшей данные, и снятия блокировки.

При выполнении транзакции современные СУБД могут блокировать всю БД, отношение, группу строк или отдельную строку. Очевидно, что чем больше блокируемый элемент данных, тем медленнее СУБД обрабатывает транзакции — велико время ожидания снятия блокировок. При работе в режиме оперативного доступа к БД, как правило, реализуется блокировка на уровне отдельных строк. В этом случае можно добиться максимальной производительности за счет того, что блокируемый объект — минимальная структурная единица БД.

Транзакции могут попасть в ситуацию взаимоблокировки. Для предотвращения таких ситуаций СУБД периодически проверяет блокировки, установленные выполняющимися транзакциями. Если обнаруживается ситуация взаимоблокировки, то одна из транзакций, вызвавших эту ситуацию, прерывается. Это разрешает тупиковые ситуации. Программа, которая сгенерировала прерванную транзакцию, получает сообщение об ошибке. Для того чтобы избежать взаимоблокировок, стараются в каждой транзакции обновление отношений делать в одной и той же последовательности.

Современные информационные системы работают с распределенными БД, поэтому в одной транзакции могут модифицироваться отношения, физически хранящиеся на удаленных вычислительных системах. Транзакция, обновляющая данные на нескольких узлах сети, называется распределенной. Если транзакция работает с БД, расположенной на одном узле, то ее называют локальной. Таким образом, логически распределенная транзакция состоит из нескольких локальных.

С точки зрения пользователя, локальные и глобальные транзакции должны обрабатываться одинаково, то есть СУБД должна организовать процесс выполнения распределенной транзакции так, чтобы все локальные транзакции, входящие в нее, синхронно фиксировались на затрагиваемых ими узлах распределенной системы. Однако распределенная транзакция должна фиксироваться только в случае, когда зафиксированы все локальные транзакции, ее составляющие. Если прерывается хотя бы одна из локальных транзакций, должна быть прервана и распределенная транзакция.

Для практической реализации этих требований в СУБД используют механизм двухстадийной фиксации транзакций (two phase commit). При его использовании фиксация распределенных транзакций осуществляется в два этапа (стадии). На первой стадии сервер БД, фиксирующий распределенную транзакцию, посылает команду «приготовиться к фиксации» всем узлам сети (серверам БД), задействованным для выполнения локальных транзакций, инициированных распределенной транзакцией. Все серверы локальных БД должны в ответ сообщить, что они готовы к фиксации. Если хотя бы от одного из серверов ответ не получен, например, если имела место программная или аппаратная ошибка при выполнении локальной транзакции, то сервер распределенной БД производит откат локальных транзакций на всех узлах, включая те, которые прислали оповещение о готовности к фиксации.

Вторая стадия начинается, когда все локальные СУБД готовы к фиксации. Сервер, обрабатывающий распределенную транзакцию, заканчивает ее фиксацию, посылая команду «зафиксировать транзакцию» всем локальным серверам.

Описанный механизм фиксации гарантирует синхронную фиксацию распределенной транзакции на всех узлах сети.

3.2.2. Тиражирование данных

Описанный подход выполнения транзакций в распределенных системах не единственно возможный. Альтернатива ему — ***технология тиражирования данных***. Эта технология предполагает отказ от распределенности данных: во всех узлах вычислительной системы должна находиться своя копия БД. Средства тиражирования автоматически поддерживают согласованное состояние информации в нескольких БД посредством копирования изменений, вносимых в любую из них. Любая транзакция в такой системе выполняется локально, поэтому нет необходимости в сложной процедуре фиксации.

Узкое место такого подхода — обеспечение тождественности данных в узлах сети. Процесс переноса изменений исходной БД в базы, принадлежащие различным узлам распределенной системы, принято называть тиражированием данных. Функции тиражирования данных выполняет специальный модуль СУБД — сервер тиражирования данных (репликатор). При любых изменениях в тиражируемых данных репликатор копирует их на все остальные узлы системы. Схема тиражирования может быть построена на полном обновлении содержимого таблицы на удаленных серверах (схема с полным обновлением) или же на обновлении только изме-

нившихся записей (быстрое обновление). Если в системе нет необходимости поддерживать постоянную идентичность данных и БД узлов должны согласовываться лишь периодически, репликатор накапливает изменения и в нужные моменты времени копирует их на другие узлы. Процесс тиражирования данных скрыт от прикладных программ пользователей, репликатор автоматически поддерживает БД в согласованном состоянии.

При использовании технологии тиражирования уменьшается тра-фик, так как все запросы обрабатываются локальной СУБД, а на другие узлы передаются только изменения в данных; увеличивается скорость доступа к данным. Кроме того, обрыв связи между узлами не останавливает обработку данных. Однако эта технология не лишена недостатков. Так, невозможно полностью исключить конфликты, возникающие при одновременном изменении одних и тех же данных на разных узлах. При переносе этих изменений в узлах вычислительной системы могут оказаться несогласованные копии БД, в результате чего пользователи различных узлов распределенной БД будут получать разные ответы на одни и те же запросы [8].

3.2.3. Надежность хранения данных

Одно из основных требований к современным OLTP-системам — надежность хранения данных. СУБД должна уметь восстанавливать согласованное состояние базы данных после любых аппаратных и программных сбоев. Для восстановления после сбоев СУБД использует журнал транзакций, который содержит последовательность записей, описывающих изменения в БД.

Общий принцип восстановления после сбоя таков: результаты выполнения транзакций, зафиксированных до сбоя, должны присутствовать в восстановленной БД; результаты незафиксированных транзакций в ней должны отсутствовать, то есть восстанавливается последнее до сбоя согласованное состояние базы данных. Процесс восстановления основан на механизме отката незавершенных транзакций, который описан ранее.

Конечно, журнал транзакций не поможет, если содержимое внеш-ней памяти системы физически уничтожено, утеряна вся информация БД. Для того чтобы избежать подобных ситуаций, реализуют дублированное хранение данных, например зеркалирование дискового прост-ранства — запись данных одновременно на несколько устройств. После сбоя копируется содержимое БД, а затем, как и в первом случае, на основе журнала откатываются все незавершенные транзакции.

3.2.4. Мониторы транзакций

С ростом сложности распределенных вычислительных систем возникают проблемы эффективного использования их ресурсов. Для решения этих проблем в состав распределенных OLTP-систем вводят дополнительный компонент — монитор транзакций TPM (Transaction Processing Monitor). Мониторы транзакций выполняют две основные функции: динамическое распределение запросов в системе (выравнивание нагрузки) и оптимизацию числа выполняющихся серверных приложений. Кратко рассмотрим эти функции.

Если в системе функционирует несколько серверов, предоставляющих одинаковый сервис, например, доступ к БД, то для оптимизации пропускной способности системы (числа обрабатываемых запросов в единицу времени) необходимо добиться сбалансированной их загрузки, то есть необходимо обеспечить поступление примерно равного числа пользовательских запросов на каждый из них. При распределении запросов может учитываться также удаленность серверов, их готовность, содержимое запроса. Реализуется функция выравнивания нагрузки следующим образом (рис. 3.2).

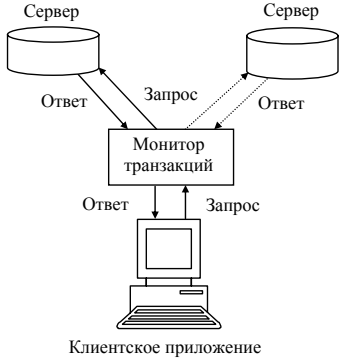


Рис. 3.2. Упрощенная схема работы монитора транзакций

Запрос клиентского приложения поступает монитору транзакций, который, действуя от имени клиентского приложения, определяет получателя запроса. Для этого он обращается к динамической маршрутной таблице, по которой определяет систему, предоставляющую соответствующий сервис. Если нужный сервис предлагают несколько систем, то в зависимости от используемого алгоритма маршрутизации выбирается одна из них, после чего ей перенаправляется запрос клиентского приложения. Маршрутизация может быть произвольной, когда система выбирается случайным образом; циклической, когда запросы посылаются системам по очереди; либо определяться содержимым запроса, если, например, серверы БД обслуживают разные подмножества данных. Результат выполнения запроса через монитор транзакций перенаправляется приложению, пославшему запрос. Клиентские приложения не знают о том, какой системе будут направлены их запросы, предлагается ли нужный им сервис одним или несколькими серверами, расположен ли нужный сервер локально, удаленно или одновременно локально и удаленно. В любом случае их запрос будет обработан оптимальным образом. Подобную схему обработки запросов называют «прозрачность местонахождения серверов» (service location transparency).

Скорость обработки транзакций прямо зависит от числа запущенных серверных приложений. Чем больше приложений одновременно обслуживает запросы, тем выше пропускная способность вычислительной системы. Это увеличение наиболее заметно на многопроцессорных системах, где каждое приложение может работать на отдельном процессоре. В идеале для эффективного использования системных ресурсов нужно по мере необходимости увеличивать или уменьшать число серверных приложений в зависимости от числа обрабатываемых запросов. Для решения этой задачи мониторы транзакций периодически измеряют отношение числа запросов в очереди к числу работающих серверных приложений. Если это отношение превышает некоторое максимальное пороговое значение (maximum watermark), то запускается дополнительная копия серверного приложения. Если это отношение падает ниже минимального порогового значения (minimum watermark), то одна из копий завершается [8].

На рынке мониторов транзакций доступно довольно много продуктов. В числе наиболее известных: TUXEDO фирмы USL, TOP END фирмы NCR, CICS фирмы IBM, ENCINA фирмы Transarc, ACMS фирмы DEC.

3.3. ОСОБЕННОСТИ ПРЕДСТАВЛЕНИЯ ДАННЫХ ДЛЯ АНАЛИЗА OLAP-СИСТЕМЫ

3.3.1. Хранилища данных

Под OLAP-системой принято понимать СППР, основанную на кон-цепции хранилища данных и обеспечивающую малое время выполнения аналитических запросов [9]. Рассмотрим более подробно истоки возникновения этой концепции и основные ее положения.

К середине 80-х годов XX в. в развитых странах мира завершился первый этап оснащения бизнеса и органов государственного управления средствами вычислительной техники. Военные ведомства и крупные корпорации установили распределенные вычислительные системы, состоявшие из мощных мейнфреймов. С появлением персональных компьютеров ЭВМ стали доступны множеству средних фирм и организаций. Исторически эти системы в первую очередь реализовывали потребности в операционной обработке данных — обслуживание информационных архивов, телефонных сетей, систем резервирования билетов, сбора метеоданных и др. Использование мощных средств вычислительной техники позволило накапливать большие объемы информации: документы, сведения о банковских операциях, клиентах, предоставленных услугах. Период хранения этой информации был относительно невелик — текущий календарный период.

Однако сбор данных — не самоцель, и накопленные информационные массивы могут быть полезны. Системы операционной обработки способны выполнять тривиальный анализ данных — вычислять максимальные, минимальные и средние значения атрибутов. Но из накопленных данных можно почерпнуть намного более глубокие сведения: попытаться выявить скрытые, на первый взгляд, закономерности и вывести из них правила, которым подчиняется предметная область информационной системы. Впоследствии эти правила можно использовать для стратегического планирования, принятия решений и прогнозирования их последствий.

Осознание пользы накапливаемой информации и возможности использовать ее для решения аналитических задач привело к появлению нового класса вычислительных систем — систем поддержки принятия решений (СППР), ориентированных на аналитическую обработку данных. Под системой поддержки принятия решений понимают человеко-машинный вычислительный комплекс, ориентированный на анализ данных и обеспечивающий получение информации, необходимой для разработки решений в сфере управления. Следует заметить, что аналитические системы существовали и ранее, но именно возможность обработки больших объемов накапливаемых данных дала новый толчок их развитию и приходу на рынок. Также этому способствовали снижение стоимости высокопроизводительных компьютеров и расходов на хранение больших объемов данных, развитие математических методов обработки информации. К числу задач, которые традиционно решают системы поддержки принятия решений, относятся оценка альтернатив решений, прогнозирование, классификация, кластеризация, выявление ассоциаций и др.

Для получения интересующей их информации лица, принимающие решения (ЛПР), или аналитики, обращаются к СППР с запросами. Эти запросы в большинстве случаев более сложные, чем те, которые применяются в системах операционной обработки данных. Например, в OLTP-системе банка запрос может сводиться к получению сведений о сумме на счету конкретного клиента. В аналитической системе запрос может быть таким: «Найти среднее значение промежутка времени между выставлением счета и оплатой его клиентом в текущем и прошедшем году отдельно для разных групп клиентов».

В большинстве случаев сложный аналитический запрос невозможно сформулировать в терминах языка SQL, поэтому для получения информации приходится применять специализированные языки, ориентированные на аналитическую обработку данных. К их числу можно отнести, например, язык Express 4GL фирмы Oracle. Также для выполнения аналитических запросов могут быть использованы приложения, написанные специально для решения тех или иных аналитических задач [8, 9].

Для того чтобы можно было извлекать полезную информацию из данных, они должны быть организованы особым, отличным от принятого в OLTP-системах образом. Связано это со следующими факторами. Во-первых, для выполнения аналитических запросов необходима обработка больших информационных массивов. Чем выше степень нормализации базы данных и чем больше в ней таблиц, тем медленнее выполняется анализ. Происходит это прежде всего потому, что увеличивается число операций соединения отношений. В системах обработки транзакций нормализация таблиц БД позволяет устранить избыточность данных, уменьшив тем самым объем действий, необходимых при обновлении информации. Поэтому в нормализованных БД нет необходимости менять одни и те же значения в различных отношениях. В аналитических системах данные практически не обновляются — в системе производится лишь их накопление и чтение. Поэтому проблема нормализации БД в них не столь актуальна, как в системах обработки транзакций. Во-вторых, выполнение некоторых аналитических запросов, например анализ тенденций и прогнозирование, требует хронологической упорядоченности данных. Реляционная модель не предполагает существования порядка записей в таблице. В-третьих, данные, используемые для целей анализа, как правило, отличаются от данных систем обработки транзакций. При обслуживании аналитических запросов чаще используются не детальные, а обобщенные (агрегированные) данные. Так, например, для прогнозирования объема продаж сети универмагов будет излишним иметь информацию о каждой сделанной покупке, достаточно знать значение прогнозируемой величины за несколько предыдущих лет.

Принципы, лежащие в основе систем поддержки принятия решений, не позволяют эффективно обрабатывать транзакции, поэтому данные, применяемые для анализа, стали выделять в отдельные базы данных. Впоследствии эти базы данных стали называть хранилищами данных (ХД) или информационными хранилищами. В литературе используется также англоязычный термин «Data Warehouse».

Автором концепции использования хранилищ данных в аналитических системах считают Билла Инмона (Bill Inmon), технического ди-ректора компании «Призм Сольюшнс» (Prism Solutions). В начале 90-х годов прошлого века он опубликовал ряд работ, которые стали отправ-ной точкой для последующих исследований в области аналитических систем. Большое влияние на разработку концепции хранилищ данных оказала также американская корпорация «Ай Би Эм» (IBM) [8].

Концепция хранилищ данных — это концепция подготовки данных для последующего анализа. Она предполагает выполнение следующих положений:

1) интеграция и согласование данных из различных источников: традиционных систем операционной обработки данных, внутренних и внешних по отношению к организации электронных архивов;

2) разделение наборов данных, используемых системами обработки транзакций и системами поддержки принятия решений.

В работе «Создание хранилища данных» («Building the Data Wa-rehouse») Билл Инмон определил хранилище данных как «предметно-ориентированный, интегрированный, неизменяемый и поддерживающий хронологию набор данных, предназначенный для обеспечения принятия управленческих решений». Позднее мы вернемся к этому определению и подробнее рассмотрим черты ХД, указанные Инмоном. А пока попытаемся уяснить схему функционирования СППР, основанной на концепции хранилища данных, проведя аналогии с процессами производства и реализации промышленной продукции.

Производство и реализация товаров имеют много общего с анализом данных: на предприятии из сырья получается готовая продукция, которая затем доставляется потребителю; в процессе анализа из накопленных данных выделяется и предоставляется полезная специалистам информация, используемая для разработки решений. Любая продукция, прежде чем быть доставленной потребителю, должна быть изготовлена. Этим занимаются специализированные промышленные предприятия — фабрики, заводы, комбинаты. Произведенная продукция отправляется на склад, откуда поступает в магазины. Именно там она находит своего потребителя.

Подобная схема обработки и снабжения справедлива и для аналитической системы. Исходные данные для анализа производятся системами операционной обработки, поступают из электронных архивов и от поставщиков информации, например онлайновых информационных агентств. Эти источники слабо связаны между собой, поэтому и данные, которые они предоставляют, имеют различную структуру и форматы представления. Необходимо произвести согласование данных разных источников, чтобы ими было удобно оперировать при анализе. Это подразумевает приведение их к единому формату, устранение дублирующихся и некорректных значений.

Подготовленные данные загружаются в хранилище. Пользователи-аналитики осуществляют доступ к нему через клиентские приложения. Эти приложения могут осуществлять трансляцию запросов потребителей информации либо производить аналитическую обработку данных хранилища. В отличие от систем операционной обработки данных в СППР, использующих концепцию ХД, критерии поиска и состав выдаваемой в виде отчета информации не фиксируются при ее разработке, пользователи оперируют в основном заранее не регламентированными запросами (ad-hoc query).

Использование концепции хранилища данных в системе поддержки принятия решений преследует следующие цели:

1) своевременное обеспечение аналитиков всей информацией, необходимой для выработки решений;

2) создание единой модели данных организации;

3) создание интегрированного источника данных, предоставляющего удобный доступ к разнородной информации и гарантирующего получение одинаковых ответов на одинаковые запросы из различных аналитических подсистем (единый «источник истины»).

Сейчас хранилища данных рассматриваются как панацея, которая может обеспечить новое качество информационной системы. Рост интереса к ним объясняется также и умелой рекламной политикой поставщиков аппаратно-программных решений на основе этой концепции.

В соответствии с определением ХД, предложенным Инмоном, рассмотрим подробнее свойства, им присущие.

Ориентация на предметную область. Хранилище должно разрабатываться с учетом специфики предметной области, а не приложений, оперирующих данными. Структура хранилища должна отражать представления аналитика об информации, с которой ему приходится работать. Например, если система операционной обработки поставщика товаров работает с понятиями «сделка» и «заявка», то хранилище должно использовать понятия «клиенты», «товары» и «производители».

Интегрированность. Информация загружается в хранилище из приложений, созданных разными разработчиками. Необходимо объединить данные этих приложений, приведя их к единому синтаксическому и семантическому виду. Например, в таблицах БД, полученных из разных источников, могут встречаться атрибуты, которые определены на разных доменах, но обозначают те же понятия. Например, месяц года может быть задан полным наименованием (январь, февраль и т. д.), сокращенным наименованием (янв, фев и т. д.) и номером (1, 2 и т. д.). В процессе загрузки хранилища требуется преобразовать эти атрибуты к единому представлению. Важно также провести проверку поступающих данных на целостность и непротиворечивость. Характерный для информационных хранилищ прием — хранение агрегированных данных. Аналитика редко интересует информация о конкретных днях и часах, ему более важны данные о месяцах, кварталах и даже годах. Чтобы при выполнении аналитических запросов избежать выполнения операций группирования, данные должны обобщаться (агрегироваться) при загрузке хранилища. Объем накопленных данных должен быть достаточным для решения аналитических задач с требуемым качеством. Используемые в настоящее время ХД содержат информацию, накопленную за годы и даже десятилетия.

Неизменяемость данных. Важное отличие аналитических систем от систем операционной обработки данных состоит в том, что дан-ные после загрузки в них остаются неизменными, внесение каких-либо изменений, кроме добавления записей, не предполагается. Разработчики подобных систем сосредоточивают основные усилия на достижении высокой скорости доступа к данным. Важное условие неизменности информации в хранилище — использование для его реализации надежного оборудования, которое обеспечивает защиту от сбоев.

Поддержка хронологии. Для выполнения большинства аналитических запросов необходим анализ тенденций развития явлений или характера изменения значений переменных во времени. Учет хронологии достигается введением ключевых атрибутов типа «ДАТА» и/или «ВРЕМЯ» в структуры хранилища данных. Время выполнения аналитических запросов можно уменьшить, если физически упорядочить записи по времени, то есть расположить записи по возрастанию значений атрибута «ДАТА/ВРЕМЯ».

К числу основных задач, которые требуется решать при создании ХД, относятся:

1) выбор оптимальной структуры хранения данных с точки зрения обеспечения приемлемого времени отклика на аналитические запросы и требуемого объема памяти;

2) первоначальное заполнение и последующее пополнение хранилища данными;

3) обеспечение удобства доступа пользователей к данным.

Рассмотрим пути решения этих задач более детально.

3.3.2. Модели данных, используемые для построения хранилищ

Задачи, решаемые OLTP и аналитическими системами, существенно различаются, поэтому их БД тоже построены на разных принципах. Критерием эффективности для систем операционной обработки данных служит число транзакций, которое они способны выполнить в единицу времени. Для аналитических систем важнее скорость выполнения сложных запросов и прозрачность структуры хранения информации для пользователей. Важная особенность СППР на основе ХД состоит в том, что загрузка данных выполняется сравнительно редко, но большими порциями (до нескольких миллионов записей за один раз), поэтому в таких системах обычно не предусматриваются развитые средства обеспечения целостности, восстановления, устранения взаимных блокировок. Это не только существенно облегчает и упрощает сами средства реализации, но и значительно снижает

внутренние накладные расходы при доступе к информации и, следовательно, повышает производительность анализа.

В настоящее время существуют два в чем-то конкурирующих, а в чем-то взаимодополняющих друг друга подхода к построению хранилищ данных: подход, основанный на использовании многомерной модели БД MOLAP (Multidimedsional OLAP), и подход, использующий реляционную модель БД ROLAP (Relational OLAP). Прежде чем рассказать о каждом из них, попытаемся разобраться, какие данные могут находиться в хранилище и как они могут быть представлены. Чаще всего там содержатся сведения о значении некоторых параметров, характеризующих предметную область в определенные моменты или за определенные промежутки времени. Пусть, например, требуется создать хранилище, накапливающее информацию об изменении социально-экономической обстановки в России. Эта обстановка характеризуется многими параметрами, в числе которых объем промышленного производства, индекс потребительских цен и др. Госкомстат России собирает их значения для различных субъектов Российской Федерации помесячно, поквартально или за год.

В хранилище должны попадать факты вида: название параметра в субъекте Российской Федерации в момент времени был равен {значение}.

Например, индекс потребительских цен в городе Москве в декабре 1996 года составил 101 %. В рассматриваемом примере каждое значение связано с точкой в трехмерном пространстве (*N*, *S*, *T*) с измерениями: *N* — название параметра; *S* — субъект Федерации; *T* — момент времени. Число возможных параметров, субъектов РФ, а также рассматриваемых моментов времени конечно, поэтому все возможные значения можно представить в виде гиперкуба (рис. 3.3). В этом гиперкубе каждое значение находится в строго определенной ячейке (см. на рис. 3.3: 1 — значение «Параметра *M*» для «Субъекта РФ 1» в январе 1991 года), что значительно упрощает обращение к ней.

Представленный пример, конечно, упрощен, но он позволяет понять, что такое многомерный взгляд на данные. В реальной задаче число измерений может быть больше трех. Представление данных в виде гиперкуба более наглядно, чем совокупность нормализованных таблиц, оно понятно не только администратору БД, но и рядовым сотрудникам. Это дает им дополнительные возможности построения аналитических запросов к системе, использующей хранилище данных. Кроме того, использование многомерной модели данных позволяет резко уменьшить время поиска в ХД, обеспечивая выполнение аналитических запросов в реальном времени.

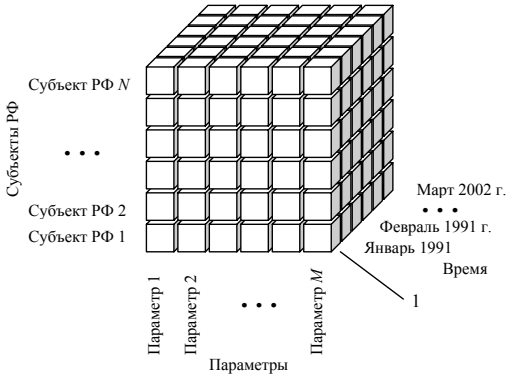


Рис. 3.3. Представление данных в виде гиперкуба

Гиперкуб может быть реализован в рамках реляционной модели или существовать как отдельная БД специальной многомерной структуры. В зависимости от этого и принято различать реляционный (ROLAP) и многомерный (MOLAP) подходы к построению ХД [8].

3.3.3. Многомерная модель хранилища

Многомерная модель БД появилась довольно давно, однако в силу присущих ей ограничений применение получила лишь в последнее время. При использовании этой модели данные хранятся не в виде плоских таблиц, как в реляционных БД, а в виде гиперкубов — упорядоченных многомерных массивов, то есть многомерное представление данных здесь реализуется физически. Конечно, такой подход требует большего объема памяти для хранения данных, при его использовании сложно модифицировать структуру данных. Например, добавление еще одного измерения приводит к необходимости полной перестройки гиперкуба. Однако многомерные СУБД обеспечивают более быстрый по сравнению с реляционными системами поиск и чтение данных, избавляют от необходимости многократно соединять таблицы. Среднее время ответа на сложный аналитический запрос при использовании многомерных СУБД обычно в 10–100 раз меньше, чем в случае реляционной СУБД с нормализованной структурой.

Основные понятия многомерной модели — измерение и значение (ячейка). Измерение — это множество, образующее одну из граней гиперкуба (аналог домена в реляционной модели). Измерения играют роль индексов, используемых для идентификации конкретных значений в ячейках гиперкуба. Значения — это подвергаемые анализу количественные или качественные данные, которые находятся в ячейках гиперкуба (см. рис. 3.3).

В многомерной модели вводятся следующие основные операции манипулирования измерениями:

1) срез (slice);
2) вращение (relate);
3) детализация (drill down);
4) агрегация (drill up).

При выполнении операции среза формируется подмножество гиперкуба, в котором значение одного или более измерений фиксировано. Например, если на рис. 3.3 зафиксировать значение измерения «Время» равным «Январь 1991 года», то мы получим двухмерную таблицу с информацией о значениях всех параметров для всех субъектов РФ в январе 1991 года.

Операция вращения изменяет порядок представления измерений. Она обычно применяется к двухмерным таблицам, обеспечивая представление их в более удобной для восприятия форме. Если в исходной таблице по горизонтали были расположены субъекты РФ, а по вертикали — параметры социально-экономической сферы, то после операции вращения параметры будут размещены по горизонтали, а названия субъектов РФ — по вертикали.

Для выполнения операций агрегации и детализации должна существовать иерархия значений измерения, то есть некоторая подчиненность одних значений другим. Например, 12 месяцев образуют год, субъекты РФ образуют регионы. При выполнении операции агрегации одно из значений измерения заменяется значением более высокого уровня иерархии. Например, аналитик, узнав значения параметров для января 1991 года, желает получить их значения за весь 1991 год. Чтобы это сделать, необходимо выполнить операцию агрегации. Операция детализации — это операция, обратная агрегации. Она обеспечивает переход от обобщенных к детализированным данным.

Основное назначение СУБД, поддерживающих многомерную модель, — реализация систем, ориентированных на аналитическую обработку. Многомерные СУБД лучше других справляются с задачами выполнения сложных нерегламентированных запросов.

Однако у многомерных БД имеются серьезные недостатки, сдерживающие их применение. Многомерные СУБД неэффективно по сравнению с реляционными используют память. В многомерной СУБД заранее резервируется место для всех значений, даже если часть из них заведомо будет отсутствовать. Другой недостаток состоит в том, что выбор высокого уровня детализации при реализации гиперкуба может очень сильно увеличить размер многомерной БД. В силу этих, а также некоторых других причин доступные на рынке многомерные СУБД не в состоянии оперировать данными большого объема. Объем, доступный им для хранения, ограничен 10–20 гигабайтами [8].

Целесообразно использовать многомерную модель, если объем БД невелик и гиперкуб имеет стабильный во времени набор измерений.

3.3.4. Реляционная модель хранилища данных

Основой при построении хранилища данных может служить и традиционная реляционная модель данных. В этом случае гиперкуб эмулируется СУБД на логическом уровне. В отличие от многомерных реляционные СУБД способны хранить огромные объемы данных, однако они проигрывают по скорости выполнения аналитических запросов.

При использовании реляционных СУБД данные для хранилища организуются специальным образом. Чаще всего используется так называемая радиальная схема. Другое ее название — «звезда» (star). В этой схеме используются два типа таблиц: таблица фактов (фактологическая таблица) и несколько справочных таблиц (таблицы измерений).

В таблице фактов обычно содержатся данные, наиболее интенсивно используемые для анализа. Если проводить аналогию с многомерной моделью, то запись фактологической таблицы соответствует ячейке гиперкуба. В справочной таблице перечислены возможные значения одного из измерений гиперкуба. Каждое измерение описывается своей собственной справочной таблицей. Фактологическая таблица индексируется по сложному ключу, скомпонованному из индивидуальных ключей справочных таблиц. Это обеспечивает связь справочных таблиц с фактологической по ключевым атрибутам. В качестве примера на рис. 3.4 приведена упрощенная схема структуры хранилища данных, используемого для накопления информации из рассмотренного ранее примера (см. рис. 3.3).

В реальных системах количество строк в фактологической таблице может составлять десятки и сотни миллионов. Число справочных таблиц обычно не превышает двух десятков. Для увеличения производительности анализа в фактологической таблице могут храниться не только детализированные, но и предварительно вычисленные агрегированные данные.

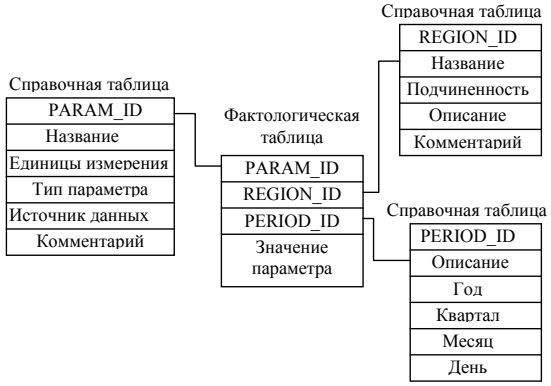


Рис. 3.4. Пример базы данных со схемой «звезда»

Если БД включает большое число измерений, можно использовать схему «снежинка» (snowflake). В этой схеме атрибуты справочных таблиц могут быть детализированы в дополнительных справочных таблицах (рис. 3.5).

Для сокращения времени отклика аналитической системы, можно использовать некоторые специальные средства. В состав мощных реляционных СУБД обычно входят оптимизаторы запросов. При создании хранилищ данных на основе реляционных СУБД их наличие приобретает особую важность. Оптимизаторы анализируют запрос и определяют лучшую с позиции некоторого критерия последовательность операций обращения к БД для его выполнения. Например, может минимизироваться число физических обращений к дискам при выполнении запроса. Оптимизаторы запросов используют сложные алгоритмы статистической обработки, которые оперируют числом записей в таблицах, диапазонами ключей и т. д.

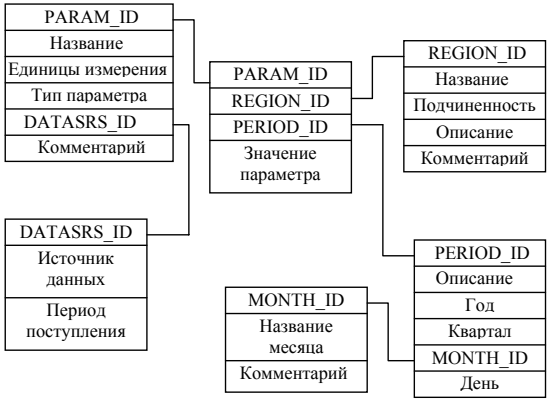


Рис. 3.5. Пример базы данных со схемой «снежинка»

Каждая из описанных моделей имеет как преимущества, так и недостатки. Многомерная модель позволяет производить быстрый анализ данных, но не позволяет хранить большие объемы информации. Реляционная модель, напротив, практически не имеет ограничений по объему накапливаемых данных, однако СУБД на ее основе не обеспечивают такой скорости выполнения аналитических запросов, как многомерная СУБД. Нельзя ли совместить два этих подхода так, чтобы скрыть их недостатки и сделать более заметными их достоинства? Удачные проекты реализации хранилищ данных, появившиеся в последнее время, показывают, что это возможно.

Ситуация, когда для анализа необходима вся информация, находящаяся в хранилище, возникает довольно редко. Обычно каждый аналитик или аналитический отдел обслуживает одно из направлений деятельности организации, поэтому в первую очередь ему необходимы данные, характеризующие именно это направление. Реальный объем этих данных не превосходит ограничений, присущих многомерным СУБД. Возникает идея выделить данные, которые реально нужны конкретным аналитическим приложениям, в отдельный набор. Такой набор мог бы быть реализован в многомерной БД. Источником данных для него должно быть центральное хранилище организации.

Если проводить аналогии с производством и реализацией продукции, то многомерные БД выполняют роль мелких складов. В концепции ХД их принято именовать киосками данных (Data Marts). ***Киоск данных*** — это специализированное тематическое хранилище, обслуживающее одно их направлений деятельности организации [8].

3.3.5. Обнаружение знаний в хранилищах данных

Data mining

При использовании хранилищ остро встает проблема обнаружения в них знаний KDD (knowledge discovery in databases). Основным шагом этого процесса является data mining (исследование данных или, дословно, «***добыча данных***») [9]. После применения традиционных методов увеличения доходов (маркетинговых исследований и действий на рынке, работы с конкурентами) или уменьшения расходов вследствие изменения технологии, работы с поставщиками, перед менеджерами высшего звена встает задача по дальнейшему увеличению прибыли как основной цели деятельности любого коммерческого предприятия.

Для этого в последнее время был разработан ряд технологий, которые призваны извлекать из хранилищ данных большого объема новую информацию путем построения различных моделей. Они и получили название data mining. Простой доступ пользователя к хранилищу данных обеспечивает только получение ответов на задаваемые вопросы, в то время как технология data mining позволяет увидеть («добыть») такие интересные взаимоотношения между данными, о которых пользователь прежде мог и не догадываться, а их применение может способствовать увеличению прибыли предприятия.

Как известно, большинство организаций накапливают за время своей деятельности огромные объемы данных, но единственное, что они хотят от них получить, — это информация. Как можно получить из данных информацию о том, что нужно наиболее предпочтительным для организации клиентам, как разместить ресурсы наиболее эффективным образом или как минимизировать потери? Новейшая технология решения этих проблем — это технология data mining. Она использует сложный статистический анализ и моделирование для нахождения моделей и отношений, скрытых в базе данных, т. е. таких моделей, которые не могут быть найдены обычными методами.

Модель, как и карта, — это абстрактное представление реальности. Карта может указывать на путь от аэропорта до дома, но она не может показать аварию, которая создала «пробку», или ремонтные работы, которые ведутся в настоящий момент и требуют объезда. До тех пор пока модель не соответствует существующим реально отношениям, невозможно получить успешные результаты.

Существуют два вида моделей: предсказательные и описательные. Первые используют один набор данных с известными результатами для построения моделей, которые явно предсказывают результаты для других наборов, а вторые описывают зависимости в существующих данных, которые в свою очередь используются для принятия управленческих решений или действий.

Конечно же, компания, которая долго находится на рынке и знает своих клиентов, уже осведомлена о многих моделях, которые наблюдались в течение нескольких последних периодов. Но технологии data mining могут не только подтвердить эти эмпирические наблюдения, но и найти новые, неизвестные ранее, модели. Сначала это может дать пользователю лишь небольшое преимущество. Но если объединить получаемые преимущества по всем видам товаров и каждому клиенту, то возникает существенный отрыв от тех пользователей, которые не применяют технологии data mining. С другой стороны, с помощью методов data mining можно найти такую модель, которая приведет к радикальному улучшению в финансовом и рыночном положении компании.

В чем же разница между средствами data mining и средствами OLAP?

OLAP — это часть технологий, направленных на поддержку принятия решения. Обычные средства формирования запросов и отчетов описывают саму базу данных. Технология OLAP используется для ответа на вопрос, почему некоторые вещи являются такими, какими они предстают в действительности. При этом пользователь сам формирует гипотезу о данных или отношениях между данными и после этого использует серию запросов к базе данных для подтверждения или отклонения этих гипотез. Средства data mining отличаются от средств OLAP тем, что вместо проверки предполагаемых взаимозависимостей, они на основе имеющихся данных могут производить модели, позволяющие количественно оценить степень влияния исследуемых факторов. Кроме того, средства data mining позволяют создавать новые гипотезы о характере неизвестных, но реально существующих отношений в данных.

Средства OLAP обычно применяются на ранних стадиях процесса KDD потому, что они помогают нам понять данные, фокусируя внимание аналитика на важных переменных. Это приводит к лучшему пониманию данных, что в свою очередь ведет к более эффективному результату процесса KDD.

Наличие хранилища данных является необходимым условием для успешного проведения всего процесса KDD. Вспомним, что хранилище данных — это предметно-ориентированное, интегрированное, привязанное ко времени, неизменяемое собрание данных для поддержки процесса принятия управленческих решений. Предметная ориентация означает, что данные объединены в категории и хранятся в соответствии с теми областями, которые они описывают, а не с приложениями, которые их используют. Интегрированность означает, что данные удовлетворяют требованиям к деятельности всего предприятия в целом, а не какой-либо одной функции бизнеса. Тем самым хранилище данных гарантирует, что одинаковые отчеты, сгенерированные для различных аналитиков, будут содержать одинаковые результаты. Привязанность ко времени означает, что хранилище можно рассматривать как совокупность «исторических» данных: можно восстановить картину на любой момент времени. Атрибут времени всегда явно присутствует в структурах хранилища данных. Неизменяемость означает, что, попав однажды в хранилище, данные уже не изменяются, в отличие от оперативных систем, где данные обязаны присутствовать только в последней версии, поэтому постоянно меняются. В хранилище данные только добавляются.

Для решения перечисленного ряда задач, неизбежно возникающих при организации и эксплуатации информационного хранилища, существует специализированное программное обеспечение. Современ-ные средства администрирования хранилища данных обеспечивают эффективное взаимодействие с инструментарием data mining. В качестве примера можно привести два продукта компании SAS Institute: SAS Warehouse Administrator и SAS Enterprise Miner, степень взаимной интеграции которых позволяет использовать при реализации проекта data mining также и метаданные из информационного хранилища.

Виды моделей

Рассмотрим основные виды моделей, которые используются для нахождения нового знания на основе данных хранилища. Целью технологии data mining является производство нового знания, которое пользователь может в дальнейшем применить для улучшения результатов своей деятельности. Результат моделирования — это выявленные отношения в данных. Можно выделить, по крайней мере, шесть методов выявления и анализа знаний:

1) классификацию;
2) регрессию;
3) прогнозирование временных последовательностей (рядов);
4) кластеризацию;
5) ассоциацию;
6) последовательность.

Первые три используются главным образом для прогнозирования, в то время как последние удобны для описания существующих закономерностей в данных.

Вероятно, наиболее распространенной сегодня операцией интеллектуального анализа данных является ***классификация***. С ее помощью выявляются признаки, характеризующие группу, к которой принадлежит тот или иной объект. Это делается посредством анализа уже классифицированных объектов и формулирования некоторого набора правил. Во многих видах бизнеса болезненной проблемой считается потеря постоянных клиентов. В разных сферах (например, в сотовой телефонной связи, фармацевтическом бизнесе или деятельности, связанной с кредитными карточками) ее обозначают различными терминами — «перемена моды», «истощение спроса» или «покупательская измена», но суть при этом одна. Классификация поможет вам выявить характеристики «неустойчивых» покупателей и создать модель, способную предсказать, кто именно склонен уйти к другому поставщику. Используя ее, можно определить самые эффективные виды скидок и другие выгодные предложения, которые будут наиболее действенны для тех или иных типов покупателей. Благодаря этому вам удастся удержать клиентов, потратив ровно столько денег, сколько необходимо. Однажды определенный эффективный классификатор используется для идентификации новых записей в базе данных в соответствии с уже существующими классами, и в этом случае он приобретает характер прогноза. Например, классификатор, который умеет идентифицировать риск выдачи займа, может быть использован для целей принятия решения по поводу того, насколько велик риск предоставления займа определенному клиенту, то есть для прогнозирования возможности возврата займа.

Регрессионный анализ используется в том случае, если отношения между переменными могут быть выражены количественно в виде некоторой комбинации этих переменных. Полученная комбинация далее используется для предсказания значения, которое может принимать целевая (зависимая) переменная, вычисляемая на заданном наборе значений входных (независимых) переменных. В простейшем случае для этого используются стандартные статистические методы, такие как линейная регрессия. К сожалению, большинство реальных моделей не укладываются в рамки линейной регрессии. Например, размеры продаж или фондовые цены очень сложны для предсказания, потому что могут зависеть от комплекса взаимоотношений множества переменных. Таким образом, необходимы комплексные методы для предсказания будущих значений.

Прогнозирование временных последовательностей позволяет на основе анализа поведения временных рядов оценить будущие значения прогнозируемых переменных. Конечно, эти модели должны включать в себя особые свойства времени: иерархию периодов (декада-месяц-год или месяц-квартал-год), особые отрезки времени (пяти- шести- или семидневная рабочая неделя, тринадцатый месяц), сезонность, праздники и др.

Кластеризация относится к проблеме сегментации. Этот подход распределяет записи в различные группы или сегменты. Кластеризация в чем-то аналогична классификации, но отличается от нее тем, что для проведения анализа не требуется иметь выделенную целевую переменную.

Ассоциация адресована, главным образом, к классу проблем, типичным примером которых является анализ структуры покупок. Классический анализ структуры покупок относится к представлению процесса приобретения какого-либо количества товаров как к одиночной экономической операции (транзакции). Так как большое количество покупок совершается в супермаркетах, а покупатели для удобства используют корзины или тележки, куда и складывается весь товар, то наиболее известным примером нахождения ассоциаций является анализ структуры покупки (market-basket analysis). Целью этого подхода является нахождение трендов среди большого числа транзакций, которые можно использовать для объяснения поведения покупателей. Эта информация может быть использована для регулирования запасов, внесения изменений в размещение товаров на территории магазина и принятия решения по проведению рекламной кампании для увеличения всех продаж или для продвижения определенного вида продукции. Хотя этот подход пришел исключительно из розничной торговли, он может также хорошо применяться и в финансовой сфере для анализа портфеля ценных бумаг и нахождения наборов финансовых услуг, которые клиенты часто приобретают вместе. Это, например, может использоваться для создания некоторого набора услуг как части кампании по стимулированию продаж. Другими словами, ассоциация имеет место в том случае, если несколько событий связаны друг с другом. Например, исследование, проведенное в супермаркете, может показать, что 65 % купивших картофельные чипсы берут также и кока-колу, а при наличии скидки за такой комплект колу приобретают в 85 % случаев. Располагая этими сведениями, менеджерам легко оценить, насколько действенна предоставляемая скидка.

Последовательность. Традиционный анализ структуры покупок имеет дело с набором товаров, представляющим одну транзакцию. Ва-риант такого анализа встречается, когда существует дополнительная информация (номер кредитной карты клиента или номер его банковского счета) для связи различных покупок в единую временную серию. В такой ситуации важно не только сосуществование данных внутри одной транзакции, но и порядок, в котором эти данные появляются в различных транзакциях и время между этими транзакциями. Правила, устанавливающие эти отношения, могут быть использованы для определения типичного набора предшествующих продаж, которые могут повести за собой последующие продажи определенного товара, то есть если существует цепочка связанных во времени событий, то говорят о последовательности. После покупки дома в 45 % случаев в течение месяца приобретается и новая кухонная плита, а в пределах двух недель 60 % новоселов обзаводятся холодильником.

Эти основные типы моделей используются для нахождения нового знания в хранилище данных. Обратимся теперь к методам, которые используются для проведения интеллектуального анализа данных.

3.4. МЕТОДЫ АНАЛИЗА ДАННЫХ

Интеллектуальные средства анализа данных используют следующие основные методы:
нейронные сети;
деревья решений;
индукцию правил.
Кроме названных, существует еще несколько дополнительных методов:
системы рассуждения на основе аналогичных случаев;
нечеткая логика;
генетические алгоритмы;
алгоритмы определения ассоциаций и последовательностей;
анализ с избирательным действием;
логическая регрессия;
эволюционное программирование.
Иногда применяется комбинация перечисленных методов.

Нейронные сети относятся к классу нелинейных адаптивных систем с архитектурой, условно имитирующей нервную ткань из нейронов. Математическая модель нейрона представляет собой некоторый универсальный нелинейный элемент с возможностью широкого изменения и настройки его характеристик. В одной из наиболее распространенных нейросетевых архитектур — многослойном персептроне с обратным распространением ошибки — эмулируется работа нейронов в составе иерархической сети, где каждый нейрон более высокого уровня соединен своими входами с выходами нейронов нижележащего слоя. На нейроны самого нижнего слоя подаются значения входных параметров, на основе которых производятся вычисления, необходимые для принятия решений, прогнозирования развития ситуации и т. д. Эти значения рассматриваются как сигналы, передающиеся в вышележащий слой, ослабляясь или усиливаясь в зависимости от числовых значений (весов), приписываемых межнейронным связям. В результате этого на выходе нейрона самого верхнего слоя вырабатывается некоторое значение, рассматриваемое как ответ (реакция всей сети на введенные значения входных параметров). Для того чтобы сеть можно было применять в дальнейшем, ее прежде надо «натренировать» на полученных ранее данных (примерах), для которых известны и значения входных параметров, и правильные ответы на них. Процесс «тренировки» состоит в подборе весов межнейронных связей и модификации внутренних параметров активационной функции нейронов. Для каждого сочетания обучающих данных на входе выходные

значения сравниваются с известным результатом. Если они различаются, то вычисляется корректирующее воздействие, учитываемое при обработке в узлах сети. Указанные шаги повторяются, пока не выполнится условие останова, например необходимая коррекция не будет превышать заданной величины.

Нейронные сети, по существу, представляют собой совокупность связанных друг с другом узлов, получающих входные данные, осуществляющих их обработку и генерирующих на выходе некий результат. Между узлами видимых входного и выходного уровней может находиться какое-то число скрытых уровней обработки. Нейронные сети реализуют непрозрачный процесс. Это означает, что построенная модель, как правило, не имеет четкой интерпретации. Некоторые алгоритмы могут транслировать модель нейронной сети в набор правил, помогающих уяснить, что именно она делает. Такую возможность предлагают некоторые оригинальные продукты, использующие технологию нейронной сети. Многие пакеты, реализующие принципы нейронных сетей, применяются не только в сфере обработки коммерческой информации. Нередко без них трудно обойтись при решении более общих задач распознавания образов, скажем расшифровки рукописного текста или интерпретации кардиограмм.

Деревья решений — это метод, который пригоден не только для решения задач классификации, но и для вычислений, и поэтому довольно широко применяется в области финансов и бизнеса, где чаще встречаются задачи численного прогноза. В результате применения этого метода к обучающей выборке данных создается иерархическая структура классифицирующих правил типа «ЕСЛИ... ТО...», имеющая вид дерева (это похоже на определитель видов из ботаники или зоологии). Для того чтобы решить, к какому классу отнести некоторый объект или ситуацию, мы отвечаем на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы могут иметь вид «значение параметра *A* больше *x*?» для случая измеряемых переменных или вида «значение переменной *B* принадлежит подмножеству признаков *C*». Если ответ положительный, мы переходим к правому узлу следующего уровня, если отрицательный — то к левому узлу; затем снова отвечаем на вопрос, связанный с соответствующим узлом. Таким образом, мы в конце концов доходим до одного из оконечных узлов — листьев, где стоит указание, к какому классу (сочетанию признаков) надо отнести рассматриваемый объект. Этот метод хорош тем, что такое представление правил наглядно и его легко понять.

Сегодня наблюдается всплеск интереса к продуктам, применяющим деревья решений. В основном это объясняется тем, что многие коммерческие проблемы решаются ими быстрее, чем алгоритмами нейронных сетей. К тому же они более просты и понятны для пользователей. В то же время нельзя сказать, что деревья решений всегда действуют безотказно: для определенных типов данных они могут оказаться неприемлемыми. В частности, методы дерева решений не очень эффективны, если целевая переменная зависит линейным образом от входных переменных, так как в этом случае дерево должно иметь большое число листьев. Иногда возникают проблемы при обработке непрерывных величин, скажем данных о возрасте покупателей или объеме продаж. В этом случае их необходимо группировать и ранжировать. Однако выбранный для ранжирования метод способен случайно скрыть выявляемую закономерность. Например, если группа объединяет людей в возрасте от 25-ти до 34-х лет, то тот факт, что на рубеже 30-ти лет некий параметр испытывает существенный разрыв, может оказаться скрытым. Этого недостатка не имеет продукт SAS En-terprise Miner в силу того, что реализованные в нем методы построения дерева решений могут автоматически выявлять границу (численный критерий) разделения данных на более однородные подгруппы.

Для деревьев решений очень остро стоит проблема значимости. Дело в том, что отдельным узлам на каждом новом построенном уровне дерева соответствует все меньшее и меньшее число записей данных — дерево может сегментировать данные на большое количество частных случаев. Чем больше этих частных случаев, чем меньше обучающих примеров попадает в каждый такой частный случай, тем менее надежной становится их классификация. Если построенное дерево слишком «кустистое» — состоит из неоправданно большого числа мелких веточек, оно не будет давать статистически обоснованных ответов. Как показывает практика, в большинстве систем, использующих деревья решений, эта проблема не находит удовлетворительного решения. Исключением из этого ряда является упомянутый выше SAS Enterprise Miner, включающий в себя широкий спектр диагностических инструментов, с помощью которых аналитик может выбрать статистически наиболее обоснованную модель из производимого множества деревьев решений и, более того, сравнить полученную модель дерева с принципиально другими типами моделей (регрессионной и нейросетевой). В данном продукте в качестве целевой переменной можно использовать как измеряемые, так и дискретные, не измеряемые, переменные или признаки.

Индукция правил создает неиерархическое множество условий, которые могут перекрываться. Индукция правил осуществляется путем генерации неполных деревьев решений, а для того чтобы выбрать, какое из них будет применено к входным данным, используются статистические методы.

Идея ***систем рассуждения на основе аналогичных случаев*** крайне проста. Для того чтобы сделать прогноз на будущее или выбрать правильное решение, эти системы находят в прошлом близкие аналоги наличной ситуации и выбирают тот же ответ, который был для них правильным. Поэтому этот метод еще называют методом «ближайшего соседа» (nearest neighbour). Системы рассуждения на основе аналогичных случаев показывают очень хорошие результаты в самых разнообразных задачах. Главный их минус заключается в том, что они вообще не создают каких-либо моделей или правил, обобщающих предыдущий опыт. В выборе решения они основываются на всем массиве доступных исторических данных, поэтому невозможно сказать, на основе каких конкретно факторов эти системы строят свои ответы.

Нечеткая логика применяется для таких наборов данных, где причисление данных к какой-либо группе является вероятностью, находящейся в интервале от 0 до 1, но не принимающей крайние значения. Четкая логика манипулирует результатами, которые могут быть либо истиной, либо ложью. Нечеткая логика применяется в тех случаях, когда необходимо манипулировать степенью «может быть» в дополнении к «да» и «нет».

Строго говоря, интеллектуальный анализ данных — далеко не основная область применения ***генетических алгоритмов***, которые, скорее, нужно рассматривать как мощное средство решения разнообразных комбинаторных задач и задач оптимизации. Тем не менее генетические алгоритмы вошли сейчас в стандартный инструментарий методов data mining. Этот метод назван так потому, что в какой-то степени имитирует процесс естественного отбора в природе. Пусть нам надо найти решение задачи, наиболее оптимальное с точки зрения некоторого критерия. Пусть каждое решение полностью описывается некоторым набором чисел или величин нечисловой природы. Скажем, если нам надо выбрать совокупность фиксированного числа параметров рынка, наиболее выраженно влияющих на его динамику, мы будем иметь дело с набором имен данных параметров. О взятом наборе можно говорить как о совокупности хромосом, определяющих качества индивида — данного решения поставленной задачи. Значения параметров, определяющих решение, будут тогда называться генами. Поиск оптимального решения при этом похож на эволюцию популяции индивидов, представленных их наборами хромосом. В этой эволюции действуют три механизма: во-первых, отбор сильнейших — наборов хромосом, которым соответствуют наиболее оптимальные решения; во-вторых, скрещивание — производство новых индивидов при помощи смешивания хромосомных наборов отобранных индивидов; в-третьих, мутации — случайные изменения генов у некоторых индивидов популяции. В результате смены поколений вырабатывается такое решение поставленной задачи, которое уже не может быть далее улучшено.

Генетические алгоритмы имеют два слабых места. Во-первых, сама постановка задачи в их терминах не дает возможности проанализировать статистическую значимость получаемого с их помощью решения и, во-вторых, эффективно сформулировать задачу, определить критерий отбора хромосом под силу только специалисту. Поэтому сегодня генетические алгоритмы надо рассматривать, скорее, как инструмент научного исследования, чем как средство анализа данных для практического применения в бизнесе и финансах.

Алгоритмы выявления ассоциаций находят правила об отдельных предметах, которые появляются вместе в одной экономической операции, например в одной покупке. Последовательность — это тоже ассоциация, но зависящая от времени. Ассоциация записывается как *A*/*B*, где *A* называется левой частью или предпосылкой, *B* — правой частью или следствием.

Частота появления каждого отдельного предмета или группы предметов определяется очень просто — считается количество появления этого предмета во всех событиях (покупках) и делится на общее количество событий. Эта величина измеряется в процентах и носит название «распространенность». Низкий уровень распространенности (менее одной тысячной процента) говорит о том, что такая ассоциация не существенна.

Для определения важности каждого полученного ассоциативного правила необходимо получить величину, которая носит название «доверительность *A* к *B*» (или взаимосвязь *A* и *B*). Эта величина показывает, как часто при появлении *A* появляется *B*, и рассчитывается как отношение частоты появления (распространенности) *A* и *B* вместе к распространенности *A*; то есть если доверительность *A* к *B* равна 20 %, то это значит, что при покупке товара *A* в каждом пятом случае приобретается и товар *B*. Необходимо заметить, что если распространенность *A* не равна распространенности *B*, то и доверительность *A* к *B* не равна доверительности *B* к *A*. В самом деле, покупка компьютера чаще ведет к покупке дискет, чем покупка дискеты к покупке компьютера.

Ещё одной важной характеристикой ассоциации является мощность ассоциации. Чем больше мощность, тем сильнее влияние, которое появление *A* оказывает на появление *B*. Мощность рассчитывается по формуле: (доверительность *A* к *B*) / (распространенность *B*).

Некоторые алгоритмы поиска ассоциаций сначала сортируют данные и только после этого определяют взаимосвязь и распространенность. Единственным различием таких алгоритмов является скорость или эффективность нахождения ассоциаций. Это особенно важно из-за огромного количества комбинаций, которые необходимо перебрать для нахождения наиболее значимых правил. Алгоритмы поиска ассоциаций могут создавать свои базы данных распространенности, доверительности и мощности, к которым можно обращаться по запросу. Например: «Найти все ассоциации, в которых для товара *X* доверительность более 50 % и распространенность не менее 2,5 %». При нахождении последовательностей добавляется переменная времени, которая позволяет работать с серией событий для нахождения последовательных ассоциаций на протяжении некоторого периода времени.

Подводя итоги описания этого метода анализа, необходимо сказать, что случайно может возникнуть такая ситуация, когда товары в супермаркете будут сгруппированы при помощи найденных моделей, но ожидаемая прибыль не будет получена. Это может произойти из-за того, что клиент не будет долго ходить по магазину в поисках желаемого товара, приобретая при этом ещё что-то, что попадается на глаза, и то, что он изначально не планировал приобрести.

Анализ с избирательным действием — одна из самых старых математических технологий классификации, которая известна по работам Р. Фишера с 1936 года. Она находит плоскости (или линии для двумерного пространства), которые разделяют группы. Полученные модели очень просты для объяснения, так как все, что пользователь должен сделать с полученной моделью, — это определить, по какую сторону плоскости или линии попадает точка. Такая технология часто применяется в медицине, социальных науках и биологии.

Однако следует признать, что анализ с избирательным действием не очень часто применяется при анализе данных с помощью технологии data mining по трем причинам. Во-первых, предполагается, что переменные, которые необходимо предсказать, нормально распределены (то есть их гистограммы напоминают колоколообразные кривые). Во-вторых, невозможно предсказать категориальные переменные, которые нельзя упорядочить (красный-желтый-зеленый). В-третьих, границы, которые разделяют классы, — это линии или плоскости (в их классическом, евклидовом, понимании), но данные большей частью не могут быть разделены таким способом.

Новые методы анализа с избирательным действием решают некоторые из этих проблем, позволяя границам быть не линейными, а квадратичными. Также можно заменить предположение нормальности оценкой реального распределения, то есть теоретическую колоколообразную кривую — параметрами реального распределения предсказываемой переменной.

Логическая (логистическая) регрессия имеет много общего с обычной линейной регрессией, но используется для предсказания вероятности появления того или иного значения дискретной целевой переменной, например (0 или 1), («да» или «нет»), («отлично», «хорошо», «плохо»). Так как эта переменная дискретна, она не может быть смоделирована методами обычной многофакторной линейной регрессии. Тем не менее результат (вероятность) может быть выражен в виде линейной комбинации входных переменных, что позволяет получить количественные оценки влияния этих параметров на зависимую переменную. Полученные вероятности могут использоваться и для оценки шансов. Шанс — это отношение вероятности появления события к вероятности того, что событие не произойдет. Это отношение значит то же самое, что и шанс в играх или спортивных соревнованиях. Когда мы говорим о том, что шансы команды выиграть футбольный матч 3 к 1, это значит, что вероятность победы в три раза больше, чем вероятность поражения; то есть мы имеем 75-процентную вероятность победы и 25-процентную вероятность поражения. Аналогичная терминология применяется и к шансам определенного типа клиента (клиента с заданным полом, доходом, семейным положением и др.) ответить на целевую рекламу.

Логическая регрессия — это, с одной стороны, классификационный инструмент, который используется для предсказания значений категориальных переменных (сделает ли определенный клиент покупку или нет), и, с другой стороны, регрессионный инструмент, который используется для оценки степени влияния входных факторов (в данном случае индивидуальных характеристик клиентов).

Эволюционное программирование — сегодня самая молодая и наиболее перспективная ветвь data mining. Суть метода в том, что гипотезы о виде зависимости целевой переменной от других переменных формулируются системой в виде программ на некотором внутреннем языке программирования. Если это универсальный язык, то теоретически на нем можно выразить зависимость любого вида. Процесс построения этих программ выглядит как эволюция в мире программ (этим метод немного похож на генетические алгоритмы). Когда система находит программу, достаточно точно выражающую искомую зависимость, она начинает вносить в нее небольшие модификации и отбирает среди построенных таким образом дочерних программ те, которые повышают точность. Таким образом, система «выращивает» несколько генетических линий программ, которые конкурируют между собой в точности выражения искомой зависимости. Специальный транслирующий модуль переводит найденные зависимости с внутреннего языка системы на понятный пользователю язык (математические формулы, таблицы и пр.), делая их легкодоступными. Для того чтобы сделать полученные результаты еще понятнее для пользователя-нематематика, имеется богатый арсенал разнообразных средств визуализации обнаруживаемых зависимостей.

Поиск зависимости целевых переменных от остальных ведется в форме функций какого-то определенного вида. Например, в одном из наиболее удачных алгоритмов этого типа — методе группового учета аргументов (МГУА) — зависимость ищут в форме полиномов. Причем сложные полиномы заменяются несколькими более простыми, учитывающими только некоторые признаки (групп аргументов). Обычно используются попарные объединения признаков. По всей видимости, этот метод не имеет существенных преимуществ по сравнению с нейронными сетями с их готовым набором стандартных нелинейных функций, несмотря на то что полученная формула зависимости, в принципе, поддается анализу и интерпретации (хотя на практике все же бывает слишком сложна для этого).

Комбинированные методы. Часто производители сочетают разные методы. Объединение средств нейронных сетей и технологии деревьев решений должно способствовать построению более точной модели и повышению ее быстродействия. Программы визуализации данных в каком-то смысле не являются средством анализа информации, поскольку они только представляют ее пользователю. Тем не менее визуальное представление, скажем, сразу четырех переменных достаточно выразительно обобщает очень большие объемы данных. Некоторые производители понимают, что для решения каждой проблемы следует применять оптимальный метод. Например, продукт SAS Enterprise Miner 3.0 включает в себя модуль автоматического построения результирующей гибридной модели, определенной на множестве моделей, созданных предварительно принципиально различными методами — методами дерева решений, нейронных сетей, обобщенной многофакторной регрессии. Другой продукт под названием Darwin, готовящийся к выпуску компанией Thinking Machines (Бедфорд, штат Массачусетс), позволит не только строить модели на основе нейронных сетей или деревьев решений, но также использовать визуализацию и системы рассуждения на основе аналогичных случаев. Кроме того, продукт включает в себя своеобразный генетический алгоритм для оптимизации моделей. Чрезвычайно активно работает в области анализа и интерпретации информации хранилищ данных и компания IBM. Многие из полученных в ее лабораториях результатов нашли применение в выпускаемых компанией инструментальных пакетах, которые можно отнести к четырем из пяти стандартных типов приложений «глубокой переработки» информации: классификации, кластеризации, выявлению последовательностей и ассоциаций.

Одной из наиболее серьезных проблем анализа и интерпретации информации является необходимость ***выделения подмножества данных*** (из соображений производительности). При построении своей модели вы можете искать компромисс между числом записей (строк) в выборке данных и количеством оцениваемых переменных. В SAS Enterprise Miner для преодоления такого рода трудностей имеется специальный модуль, позволяющий легко настроить процесс выборки из генеральной совокупности.

Контрольные вопросы

1. Приведите характеристики основных классов ИС.
2. Что понимается под термином «АИС»?
3. Приведите основные особенности и назначение OLTP-систем.
4. Раскройте сущность процесса управления транзакциями.
5. При решении каких задач применяется двухстадийная фиксация транзакций?
6. Что такое «тиражирование данных»?
7. Для чего нужны «хранилища данных»?
8. Какие основные операции присущи многомерной базе данных?
9. Приведите основные характеристики процесса обнаружения знаний в хранилище данных.
10. Какие существуют основные методы анализа данных?

4. CASE-ТЕХНОЛОГИИ

4.1. ИСТОКИ ВОЗНИКНОВЕНИЯ CASE-ТЕХНОЛОГИЙ

Закономерности развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС), создаваемых в различных областях жизнедеятельности. Современные крупные проекты ИС характеризуются, как правило, следующими особенностями [10]:

- сложностью описания (достаточно большим количеством функций, процессов, элементов данных и сложными взаимосвязями между ними), требующей тщательного моделирования и анализа данных и процессов;
- наличием совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач; приложений аналитической обработки — поддержки принятия решений, использующих нерегламентированные запросы к данным большого объема);
- отсутствием прямых аналогов, ограничивающим возможность использования каких-либо типовых проектных решений и прикладных систем;
- необходимостью интеграции существующих и вновь разрабатываемых приложений;
- функционированием в неоднородной среде на нескольких аппаратных платформах;
- разобщенностью и разнородностью отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств;
- существенной временной протяженностью проекта, обусловлен-ной, с одной стороны, ограниченными возможностями коллектива разработчиков и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИС.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

В 70-х и 80-х годах XX в. при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима. Ручная разработка обычно порождала следующие проблемы:

- неадекватную спецификацию требований;
- неспособность обнаруживать ошибки в проектных решениях;
- низкое качество документации, снижающее эксплуатационные качества;
- затяжной цикл и неудовлетворительные результаты тестирования.

С другой стороны, разработчики ИС исторически всегда стояли последними в ряду тех, кто использовал компьютерные технологии для повышения качества, надежности и производительности в своей собственной работе.

Перечисленные факторы способствовали появлению программно-технологических средств специального класса — CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (Computer Aided Software Engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управле-ние и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС [10].

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т. д. Кроме того, появлению CASE-технологии способствовали следующие факторы:

- подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;
- широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;
- внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

CASE-технология представляет собой методологию проектирования ИС, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения ИС и разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентиро-ванного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

Согласно обзору передовых технологий (Survey of Advanced Technology), составленному фирмой Systems Development Inc. в 1996 г. по результатам анкетирования более 1000 американских фирм, CASE-технология в настоящее время попала в разряд наиболее стабильных информационных технологий (ее использовали половина всех опрошенных пользователей более чем в трети своих проектов, из них 85 % завершились успешно). Однако, несмотря на все потенциальные возможности CASE-средств, существует множество примеров их неудачного внедрения, в результате чего CASE-средства становятся «полочным» ПО (shelfware). В связи с этим необходимо отметить следующее:

- CASE-средства не обязательно дают немедленный эффект, он может быть получен только спустя какое-то время;
- реальные затраты на внедрение CASE-средств обычно намного превышают затраты на их приобретение;
- CASE-средства обеспечивают возможности для получения существенной выгоды только после успешного завершения процесса их внедрения.

Ввиду разнообразной природы CASE-средств было бы ошибочно делать какие-либо безоговорочные утверждения относительно реального удовлетворения тех или иных ожиданий от их внедрения. Можно перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

- большое разнообразие качества и возможностей CASE-средств;
- относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
- отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
- широкий диапазон предметных областей проектов;
- различная степень интеграции CASE-средств в различных проектах.

Вследствие этих сложностей доступная информация о реальных внедрениях крайне ограниченна и противоречива. Она зависит от типа средств, характеристик проектов, уровня сопровождения и опыта пользователей. Некоторые аналитики полагают, что реальная выгода от использования некоторых типов CASE-средств может быть получена только после одно- или двухлетнего опыта. Другие полагают, что воздействие может реально проявиться в фазе эксплуатации жизненного цикла ИС, когда технологические улучшения могут привести к снижению эксплуатационных затрат.

Для успешного внедрения CASE-средств организация должна обладать определенными качествами в следующих областях:

- технологии (понимание ограниченности существующих возможностей и способность принять новую технологию);
- культуры (готовность к внедрению новых процессов и взаимоотношений между разработчиками и пользователями);
- управления (четкое руководство и организованность по отношению к наиболее важным этапам и процессам внедрения).

Если организация не обладает хотя бы одним из перечисленных качеств, то внедрение CASE-средств может закончиться неудачей независимо от степени тщательности следования различным рекомендациям по внедрению.

Для того чтобы принять взвешенное решение относительно инвестиций в CASE-технологию, пользователи вынуждены производить оценку отдельных CASE-средств, опираясь на неполные и противоречивые данные. Эта проблема зачастую усугубляется недостаточным знанием всех возможных особенностей использования CASE-средств. Среди наиболее важных проблем выделяются следующие [10]:

• достоверная оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки ПО;

• внедрение CASE-средств представляет собой достаточно длительный процесс и может не принести немедленной отдачи. Возможно даже краткосрочное снижение продуктивности в результате усилий, затрачиваемых на внедрение. Вследствие этого руководство организации-пользователя может утратить интерес к CASE-средствам и прекратить поддержку их внедрения;

• отсутствие полного соответствия между теми процессами и методами, которые поддерживаются CASE-средствами, и теми, которые используются в данной организации, приводит к дополнительным трудностям;

• CASE-средства зачастую трудно использовать в комплексе с другими подобными средствами. Это объясняется как различными парадигмами, поддерживаемыми различными средствами, так и проблемами передачи данных и управления от одного средства к другому;

• некоторые CASE-средства требуют слишком много усилий для того, чтобы оправдать их использование в небольшом проекте, однако и при этом можно извлечь выгоду из той дисциплины, к которой обязывает их применение;

• негативное отношение персонала к внедрению новой CASE-технологии может быть главной причиной провала проекта.

Пользователи CASE-средств должны быть готовы к необходимости долгосрочных затрат на эксплуатацию, частому появлению новых версий и возможному быстрому моральному старению средств, а также постоянным затратам на обучение и повышение квалификации персонала.

Несмотря на все высказанные предостережения и некоторый пессимизм, грамотный и разумный подход к использованию CASE-средств позволяет преодолеть все перечисленные трудности. Успешное внедрение CASE-средств должно обеспечить:

• высокий уровень технологической поддержки процессов разработки и сопровождения ПО;

• положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;

• приемлемый уровень отдачи от инвестиций в CASE-средства.

4.2. СТРУКТУРНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ИС

4.2.1. Сущность структурного подхода

Сущность структурного подхода к разработке ИС заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи, и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы снизу вверх от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов.

Все наиболее распространенные методологии структурного подхода базируются на ряде общих принципов. В качестве двух базовых принципов используются следующие:

1) «разделяй и властвуй» — принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;

2) иерархическое упорядочивание — принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к провалу всего проекта). Основными из этих принципов являются следующие [11]:

принцип абстрагирования — выделение существенных аспектов системы и отвлечение от несущественных;

принцип формализации — строгий методический подход к решению проблемы;

принцип непротиворечивости — обоснованность и согласованность элементов;

принцип структурирования данных — данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой, и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

SADT (Structured Analysis and Design Technique) — модели и соответствующие функциональные диаграммы;

DFD (Data Flow Diagrams) — диаграммы потоков данных;

ERD (Entity-Relationship Diagrams) — диаграммы «сущность-связь».

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

4.2.2. Методология функционального моделирования SADT

Методология SADT создана Дугласом Россом в 1981 году. На ее основе разработана, в частности, известная методология IDEF0 (Icam DEFinition), которая является основной частью программы ICAM (Integrated Computer-Aided Manufacturing), проводимой по инициативе ВВС США и направленной на увеличение производительности в промышленности посредством повсеместного внедрения компьютерных технологий.

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т. е. производимые им действия и связи между этими действиями. Основные элементы этой методологии основываются на следующих принципах:

• графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих «ограничения», которые в свою очередь определяют, когда и каким образом функции выполняются и управляются;

• строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика.

Правила SADT включают:

• ограничение количества блоков на каждом уровне декомпозиции (правило 3–6 блоков);

• связность диаграмм (номера блоков);

• уникальность меток и наименований (отсутствие повторяющихся имен);

• синтаксические правила для графики (блоков и дуг);

• разделение входов и управлений (правило определения роли данных).

• отделение организации от функции, т. е. исключение влияния организационной структуры на функциональную модель.

Методология SADT может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем SADT может быть использована в ходе анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

4.2.3. Методология IDEF0

В основе методологии IDEF0 лежат следующие правила:

• функциональный блок (или функция) преобразует входы в выходы (т. е. входную информацию в выходную); управление определяет, когда и как это преобразование может или должно произойти; исполнители непосредственно осуществляют это преобразование;

• с дугами связаны надписи (или метки) на естественном языке, описывающие данные, которые они представляют;

• дуги показывают, как функции между собой взаимосвязаны, как они обмениваются данными и осуществляют управление друг другом;

• выходы одной функции могут быть входами, управлением или исполнителями для другой;

• дуги могут разветвляться и соединяться;

• функциональный блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных между собой интерфейсными дугами;

• полученные в ходе декомпозиции блоки представляют основные подфункции (подмодули) единого исходного модуля;

• каждая декомпозиция выявляет полный набор подмодулей, каждый из которых представлен как блок, границы которого определены интерфейсными дугами;

• каждый из этих подмодулей может быть декомпозирован подобным же образом для более детального представления.

В IDEF0 реализованы три базовых принципа моделирования процессов:

1) принцип функциональной декомпозиции;

2) принцип ограничения сложности;

3) принцип контекста.

Принцип функциональной декомпозиции представляет собой спо-соб моделирования типовой ситуации, когда любое действие, операция, функция могут быть разбиты (декомпозированы) на более простые действия, операции, функции. Другими словами, сложная бизнес-функция может быть представлена в виде совокупности элементарных функций.

Принцип ограничения сложности. При работе с IDEF0-диаг-раммами существенным является условие их разборчивости и удобочитаемости. Суть принципа ограничения сложности состоит в том, что количество блоков на диаграмме должно быть не менее двух и не более шести. Практика показывает, что соблюдение этого принципа приводит к тому, что функциональные процессы, представленные в виде IDEF0-модели, хорошо структурированы, понятны и легко поддаются анализу.

Принцип контекста. Моделирование делового процесса начинается с построения контекстной диаграммы. На этой диаграмме отображается только один блок — главная бизнес-функция моделируемой системы. Если речь идет о моделировании целого предприятия или даже крупного подразделения, главная бизнес-функция не может быть сформулирована как, например, «продавать продукцию». Главная бизнес-функция системы — это «миссия» системы, ее значение в окружающем мире. Нельзя правильно сформулировать главную функцию предприятия, не имея представления о его стратегии. При определении главной бизнес-функции необходимо всегда иметь в виду цель моделирования и точку зрения на модель. Одно и то же предприятие может быть описано по-разному в зависимости от того, с какой точки зрения его рассматривают: директор предприятия и налоговой инспектор видят организацию совершенно по-разному. Контекстная диаграмма играет еще одну роль в функциональной модели. Она «фиксирует» границы моделируемой бизнес-системы, определяя то, как моделируемая система взаимодействует со своим окружением. Это достигается за счет описания дуг, соединенных с блоком, представляющим главную бизнес-функцию.

Основные элементы и понятия IDEF0

Модель IDEF0 состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. ***Диаграммы*** — главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса.

Одной из наиболее важных особенностей методологии IDEF0 является постепенное введение все больших уровней детализации по мере создания диаграмм, отображающих модель.

Графический язык IDEF0 достаточно прост. В основе методологии лежат три основных понятия.

Первым из них является понятие функционального блока (Acti-vity Box). Функциональный блок графически изображается в виде прямоугольника (рис. 4.1) и представляет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «производить услуги», а не «производство услуг»).

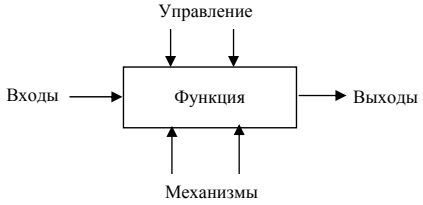


Рис. 4.1. Функциональный блок и интерфейсные дуги

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль):

• верхняя сторона имеет значение «Управление» (Control) (стрелки сверху определяют, на основании чего выполняется данный процесс — законы, стандарты, приказы и т.д.);

• левая сторона имеет значение «Вход» (Input) (стрелки слева отражают данные или объекты, потребляемые или изменяемые процессом);

• правая сторона имеет значение «Выход» (Output); (стрелки справа — основные результаты деятельности процесса, конечные продукты);

• нижняя сторона имеет значение «Механизм» (Mechanism) (стрелки снизу определяют, посредством чего или с помощью кого реализуется данный процесс — материальные и/или кадровые ресурсы, необходимые для процесса).

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер.

Вторым основным понятием методологии IDEF0 являются интерфейсные дуги (Arrow). Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную данным функциональным блоком.

Графическим отображением интерфейсной дуги является однонаправленная стрелка. Каждая интерфейсная дуга должна иметь свое уникальное наименование (Arrow Label). По требованию стандарта наименование должно быть выражено существительным.

С помощью интерфейсных дуг отображают различные объекты, в той или иной степени определяющие процессы, происходящие в системе. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т. д.) или потоки данных и информации (документы, данные, инструкции и т. д.).

В зависимости от того, к какой из сторон подходит данная интерфейсная дуга, она носит название «входящей», «исходящей» или «управляющей». Кроме того, «источником» (началом) и «приемником» (концом) каждой функциональной дуги могут быть только функциональные блоки, при этом «источником» может быть только выходная сторона блока, а «приемником» — любая из трех оставшихся.

Необходимо отметить, что любой функциональный блок по требованиям стандарта должен иметь, по крайней мере, одну управляющую интерфейсную дугу и одну исходящую, поскольку каждый процесс должен происходить по каким-то правилам, отображаемым управля-ющей дугой, и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

Отметим, что обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Существует два вида дуг:

1) внутренние (присоединяющиеся) — концы соединяются с блоками диаграмм;

2) граничные — один конец дуги является внешним:

Модель должна быть непротиворечивой по граничным дугам, т. е. при декомпозиции дуги с главной диаграммы должны точно соответствовать дугам на декомпозируемых диаграммах.

Граничные дуги кодируются с помощью ICOM-кода (Input, Output, Control, Mechanism).

Входы нумеруются сверху вниз в порядке присоединения (I1, …, Ik).

Управление кодируется слева направо в порядке присоединения (C1, …, C*l*).

Выходные дуги кодируются сверху вниз в порядке присоединения (O1, …, Om).

Механизм кодируется слева направо в порядке присоединения (M, …, Mn).

С целью повышения прозрачности графического представления вводятся туннельные дуги. Эти дуги нарушают правило декомпозиции. Данные, которые выражают туннельные дуги, не рассматриваются на соответствующем уровне детализации; обозначаются эти дуги скобками ( ).

Существует два вида туннельных дуг, выражающие:

1) данные, не обязательные на следующем уровне детализации (отсутствие на декомпозируемой диаграмме);

2) данные, не относящиеся к исходной диаграмме или на ней не описываемые (отсутствие на родительской диаграмме).

Туннельные дуги не кодируются ICOM-кодом.

Третьим основным понятием стандарта IDEF0 является декомпозиция (Decomposition). Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Декомпозиция позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Модель IDEF0 всегда начинается с представления системы как единого целого — одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой и обозначается идентификатором «A-0».

В пояснительном тексте к контекстной диаграмме должна быть указана цель (Purpose) построения диаграммы в виде краткого описания и зафиксирована точка зрения (Viewpoint).

Для глоссариев в конце диаграммы приписывается буква G, для текстовых узлов — буква T.

В модели допустимо присутствие специальных FEO-диаграмм. Данный тип диаграмм носит пояснительный характер и позволяет пояснить, что означает модель. Этот вид диаграмм тоже представляет из себя блок. Ему присваивается номер в соответствии с иерархией, а в конце приписывается буква F.

Для одной и той же диаграммы может быть несколько FEO-диа-грамм, несколько глоссариев и несколько страниц текста. Модель и диаграммы должны иметь свои имена — некоторое краткое описание. При этом диаграмма также получает имя. Например, если имя проекта ПРОЕКТ, то диаграмма, которая декомпозирует, например, 3 блок, будет обозначена как ПРОЕКТ/A3.

Декомпозиция IDEF0

Построение IDEF0-модели начинается с представления всей системы в виде простейшего компонента — одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг — они также представляют полный набор внешних интерфейсов системы в целом.

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых

представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т. е., как уже отмечалось, родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

Модель IDEF0 представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других диаграммах. Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы.

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются точно теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой.

На IDEF0-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т. д.

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию.

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая в свою очередь может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм. Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм.

Принципы ограничения сложности IDEF0-диаграмм

Обычно IDEF0-модели несут в себе сложную и концентрированную информацию, и для того чтобы ограничить их перегруженность и сделать удобочитаемыми, в соответствующем стандарте приняты соответствующие ограничения сложности:

- ограничение количества функциональных блоков на диаграмме тремя-шестью. Верхний предел (шесть) заставляет разработчика использовать иерархии при описании сложных предметов, а нижний предел (три) гарантирует, что на соответствующей диаграмме достаточно деталей, чтобы оправдать ее создание;
- ограничение количества подходящих к одному функциональному блоку (выходящих из одного функционального блока) интерфейсных дуг четырьмя.

Разумеется, строго следовать этим ограничениям вовсе необязательно, однако, как показывает опыт, они являются весьма практичными в реальной работе.

Основные этапы в процессе чтения диаграмм:

прочитать название блоков (функции), чтобы получить представление о том, что делает диаграмма;
вернуться к родительской диаграмме и понять смысл графических дуг;
попробовать определить главный путь на диаграмме (самый важный вход и выход);
мысленно пройти по главному пути;
рассмотреть побочные пути и понять их смысл;
рассмотреть все FEO-диаграммы, связанные с этими диаграммами;
рассмотреть текст и глоссарий, если они имеются.

Дисциплина групповой работы над разработкой IDEF0-модели

Стандарт IDEF0 содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой группой людей, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов.

Создание модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах IDEF0 называется авторами (Authors). Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов. На основе имеющихся положений, документов и результатов опросов создается черновик (Model Draft) модели.

Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким спектром компетентных лиц (в терминах IDEF0 — читателей) на предприятии. При этом каждая из диаграмм черновой модели письменно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает её с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.

Официальное утверждение модели. Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у авторов модели и читателей отсутствуют разногласия по поводу ее адекватности. Окончательная модель представляет собой согласованное представление о предприятии (системе) с заданной точки зрения и для заданной цели.

Наглядность графического языка IDEF0 делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений на предприятии (в системе).

4.3. МОДЕЛИРОВАНИЕ ПОТОКОВ ДАННЫХ (ПРОЦЕССОВ)

В основе ***методологии Гейна-Сарсона*** лежит построение модели анализируемой ИС — проектируемой или реально существующей. В со-ответствии с методологией модель системы определяется как иерархия диаграмм потоков данных (ДПД или DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям — потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

внешние сущности;
системы/подсистемы;
процессы;
накопители данных;
потоки данных.

Внешние сущности

Внешняя сущность представляет собой материальный предмет или физическое лицо, представляющие собой источник или приемник информации, например заказчики, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой ИС. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь диаграммы анализируемой ИС, если это необходимо, или, наоборот, часть процессов ИС может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

Рис. 4.2. Внешняя сущность

Внешняя сущность обозначается квадратом (рис. 4.2), расположенным как бы над диаграммой и бросающим на нее тень, для того чтобы можно было выделить этот символ среди других обозначений.

Системы и подсистемы

При построении модели сложной ИС она может быть представлена в самом общем виде на так называемой контекстной диаграмме в виде одной системы как единого целого либо может быть декомпозирована на ряд подсистем. Подсистема (или система) на контекстной диаграмме изображается следующим образом (рис. 4.3).

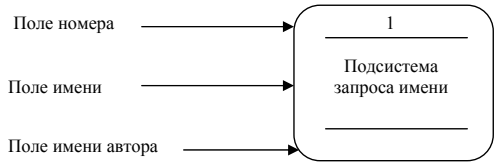


Рис. 4.3. Подсистема

Номер подсистемы служит для ее идентификации. В поле имени вводится наименование подсистемы в виде предложения с подлежащим и соответствующими определениями и дополнениями.

Процессы

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов; программа; аппаратно реализованное логическое устройство и т. д. Процесс на диаграмме потоков данных изображается, как показано на рис. 4.4.

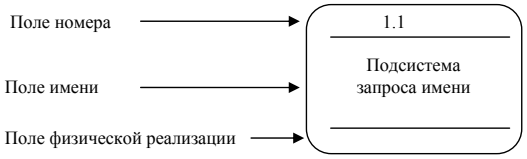


Рис. 4.4. Процесс

Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например:

- «Ввести сведения о клиентах»;
- «Выдать информацию о текущих расходах»;
- «Проверить кредитоспособность клиента».

Использование таких глаголов, как «обработать», «модернизировать» или «отредактировать» означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа.

Информация в поле физической реализации показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

Накопители данных

Накопитель данных представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

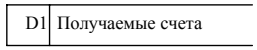


Рис. 4.5. Накопитель данных

Накопитель данных может быть реализован физически в виде микрофиши, ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т. д. Накопитель данных на диаграмме потоков данных изображается, как показано на рис. 4.5. Он идентифицируется буквой «D» и произвольным числом. Имя накопителя выбирается из соображения наибольшей информативности для проектировщика. Накопитель данных в общем случае является прообразом будущей базы данных, и описание хранящихся в нем данных должно быть увязано с информационной моделью.

Потоки данных

Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т. д.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление потока (рис. 4.6). Каждый поток данных имеет имя, отражающее его содержание.

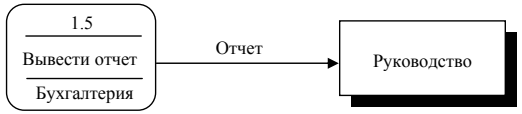


Рис. 4.6. Поток данных

Построение иерархии диаграмм потоков данных

Первым шагом при построении иерархии ДПД является построение контекстных диаграмм. Обычно при проектировании относительно простых ИС строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и кроме того, единственный главный процесс не раскрывает структуру распределенной системы. Признаками сложности (в смысле контекста) могут быть:

- наличие большого количества внешних сущностей (десять и более);
- распределенная природа системы;
- многофункциональность системы с уже сложившейся или выявленной группировкой функций в отдельные подсистемы.

Для сложных ИС строится иерархия контекстных диаграмм. Контекстная диаграмма верхнего уровня при этом содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем. Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой ИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (источниками и приемниками информации), с которыми взаимодействует ИС.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры ИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков. После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами).

Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи ДПД. Каждый процесс на ДПД в свою очередь может быть детализирован при помощи ДПД или мини-спецификации. При детализации должны выполняться следующие правила:

- правило балансировки, означающее, что при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме;
- правило нумерации, означающее, что при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т. д.

Мини-спецификация — описание логики процесса — должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу. Мини-специфи-кация является конечной вершиной иерархии ДПД. Решение о завершении детализации процесса и использовании мини-спецификации принимается аналитиком исходя из следующих критериев:

- наличия у процесса относительно небольшого количества входных и выходных потоков данных (2–3 потока);
- возможности описания преобразования данных процессом в виде последовательного алгоритма;
- выполнения процессом единственной логической функции преобразования входной информации в выходную;
- возможности описания логики процесса при помощи мини-спецификации небольшого объема (не более 20–30 строк).

При построении иерархии ДПД переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне. Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные). Для непрерывных данных может указываться единица измерения (кг, см и т. п.), диапазон значений, точность представления и форма физического кодирования. Для дискретных данных может указываться таблица допустимых значений.

После построения законченной модели системы ее необходимо верифицировать (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные недетализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки. В согласованной модели для всех потоков данных и накопителей данных должно выполняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

4.4. МОДЕЛИРОВАНИЕ ДАННЫХ

4.4.1. Case-метод Баркера

Цель моделирования данных состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются модели «сущность-связь» или ER-модели (Entity-Relation-ship), в основе которых лежат диаграммы ERD (Entity-Relationship Diagrams). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Нотация ERD была впервые введена П. Ченом и получила дальнейшее развитие в работах Баркера. Метод Баркера будет излагаться на примере моделирования деятельности компании по торговле автомобилями. Ниже приведены выдержки из интервью, проведенного с персоналом компании.

Главный менеджер: одна из основных обязанностей — содержание автомобильного имущества. Он должен знать, сколько заплачено за машины и каковы накладные расходы. Обладая этой информацией, он может установить нижнюю цену, за которую мог бы продать данный экземпляр. Кроме того, он несет ответственность за продавцов и ему нужно знать, кто что продает и сколько машин продал каждый из них.

Продавец: ему нужно знать, какую цену запрашивать и какова ниж-няя цена, за которую можно совершить сделку. Кроме того, ему нужна основная информация о машинах: год выпуска, марка, модель и прочее.

Администратор: его задача сводится к составлению контрактов, для чего нужна информация о покупателе, автомашине и продавце, поскольку именно контракты приносят продавцам вознаграждения за продажи.

Первый шаг моделирования — извлечение информации из интервью и выделение сущностей.

Сущность (Entity) — реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению (рис. 4.7).

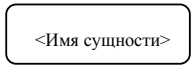


Рис. 4.7. Графическое изображение сущности

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- каждая сущность должна иметь уникальное имя, и к одному и тому же имени должна всегда применяться одна и та же интерпретация. Одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности;
- каждая сущность может обладать любым количеством связей с другими сущностями модели.

Обращаясь к приведенным выше выдержкам из интервью, можно увидеть, что сущности, которые могут быть идентифицированы с главным менеджером – это автомашины и продавцы. Продавцу важны автомашины и связанные с их продажей данные. Для администратора важны покупатели, автомашины, продавцы и контракты. Исходя из этого, выделяются 4 сущности (автомашина, продавец, покупатель, контракт), которые изображаются на диаграмме следующим образом (рис. 4.8).

Рис. 4.8. Выделенные сущности

Следующим шагом моделирования является идентификация связей. Связь (Relationship) — поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Связь — это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потом-ком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущ-ности-потомка может существовать только при существовании сущ-ности-родителя.

Связи может даваться имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя, так что предложение может быть образовано соединением имени сущности-родителя, имени связи, выражения степени и имени сущности-потомка.

Например, связь продавца с контрактом может быть выражена следующим образом:

продавец может получить вознаграждение за 1 или более контрактов;

контракт должен быть инициирован только одним продавцом.

Степень связи и обязательность графически изображаются следующим образом (рис. 4.9).

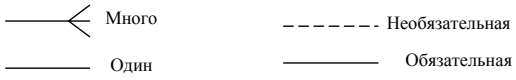


Рис. 4.9. Изображение степени связности и обязательности

Таким образом, 2 предложения, описывающие связь продавца с контрактом, графически будут выражены следующим образом (рис. 4.10).

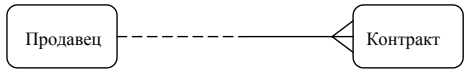


Рис. 4.10. Связь продавца с контрактом

Описав также связи остальных сущностей, получим следующую схему (рис. 4.11).

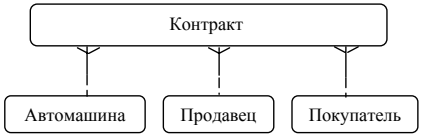


Рис. 4.11. Связи сущностей

Последним шагом моделирования является идентификация атрибутов. Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или свойств, ассоциированных с множеством реальных или аб-страктных объектов (людей, мест, событий, состояний, идей, пар предметов и т. д.). Экземпляр атрибута — это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. В ER-модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут может быть либо обязательным, либо необязательным. Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т. е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор — это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя (рис. 4.12).

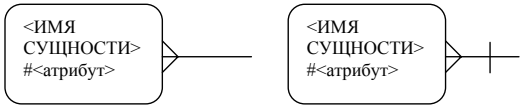


Рис. 4.12. Нотация полной и неполной идентификации

Каждый атрибут идентифицируется уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком «#».

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности — это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные — как альтернативные ключи.

С учетом имеющейся информации дополним построенную ранее диаграмму (рис. 4.13).

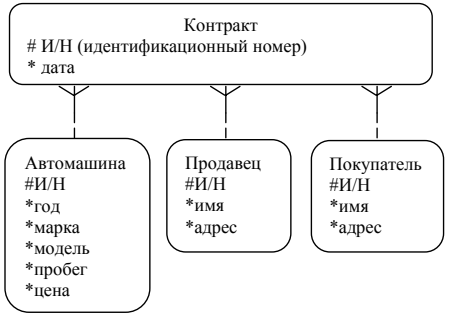


Рис. 4.13. Пример указания первичных ключей

Помимо перечисленных основных конструкций модель данных может содержать ряд дополнительных.

Подтипы и супертипы: одна сущность является обобщающим понятием для группы подобных сущностей (рис. 4.14).

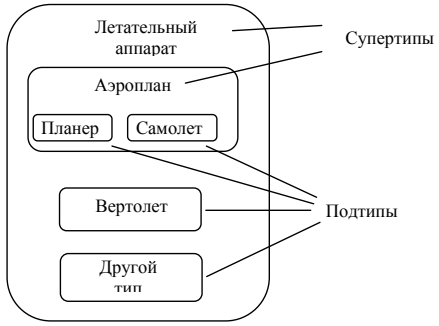


Рис. 4.14. Подтипы и супертипы

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей (рис. 4.15).

Рекурсивная связь: сущность может быть связана сама с собой (рис. 4.16).

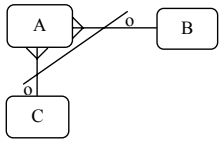


Рис. 4.15. Взаимно исключающие связи

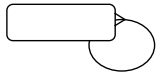


Рис. 4.16. Рекурсивная связь

Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой (рис. 4.17).

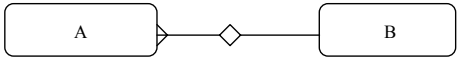


Рис. 4.17. Неперемещаемая связь

4.4.2. Методология IDEF1X

Метод IDEF1 основан на подходе П. Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия — методология IDEF1X, разработанная с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF) [10].

Сущность в методологии IDEF1X является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности (рис. 4.18).

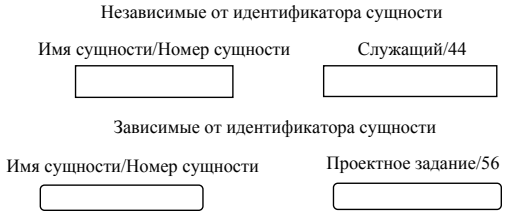


Рис. 4.18. Сущности

Каждой сущности присваивается уникальное имя и номер, разделяемые косой чертой « / » и помещаемые над блоком.

Связь может дополнительно определяться с помощью указания степени, или мощности (количества экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-роди-теля). В IDEF1X могут быть выражены следующие мощности связей:

каждый экземпляр сущности-родителя имеет ноль, один или более связанных с ним экземпляров сущности-потомка;

каждый экземпляр сущности-родителя имеет не менее одного связанного с ним экземпляра сущности-потомка;

каждый экземпляр сущности-родителя имеет не более одного связанного с ним экземпляра сущности-потомка;

каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае — неидентифицирующей.

Связь изображается линией, проводимой между сущностью-ро-дителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. Мощность связи обозначается так, как показано на рис. 4.19 (мощность по умолчанию — N).

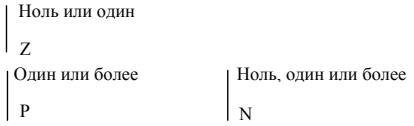


Рис. 4.19. Мощность связи

Идентифицирующая связь между сущностью-родителем и сущно-стью-потомком изображается сплошной линией (рис. 4.20). Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

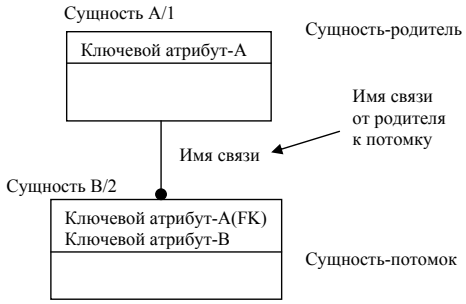


Рис. 4.20. Идентифицирующая связь

Неидентифицирующая связь отображена пунктирной линией (рис. 4.21). Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

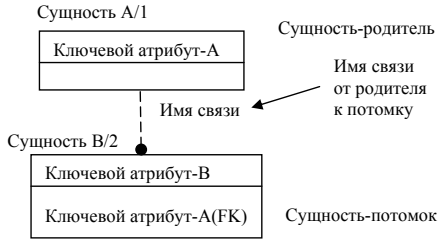


Рис. 4.21. Неидентифицирующая связь

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой (рис. 4.22).

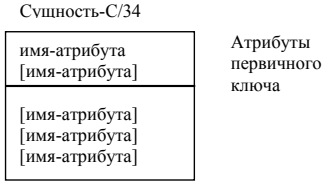


Рис. 4.22. Атрибуты и первичные ключи

Сущности могут иметь также внешние ключи (Foreign Key), которые могут использоваться в качестве части или целого первичного ключа или неключевого атрибута. Внешний ключ изображается с помощью помещения внутрь блока сущности имен атрибутов, после которых следуют буквы FK в скобках (рис. 4.23).

Рис. 4.23. Примеры внешних ключей

4.4.3. Подход, основанный на нотации П. Чена

В CASE-средствах, основанных на нотации П. Чена [10], сущность обозначается прямоугольником, содержащим имя сущности (рис. 4.24), а связь — ромбом, связанным линией с каждой из взаимодействующих сущностей. Числа над линиями означают степень связи.

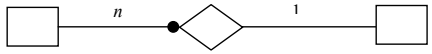


Рис. 4.24. Обозначение сущностей и связей

Связи являются многонаправленными и могут иметь атрибуты (за исключением ключевых). Выделяют два вида связей:

1) необязательная связь (optional);
2) полная связь (total).

В необязательной связи (рис. 4.25) могут участвовать не все экземпляры сущности.

Рис. 4.25. Необязательная связь

В отличие от необязательной связи в полной (total) связи участвуют все экземпляры хотя бы одной из сущностей. Это означает, что экземпляры такой связи существуют только при условии существования экземпляров другой сущности. Полная связь может иметь один из 4-х типов: обязательная связь, слабая связь, связь «супертип — подтип» и ассоциативная связь.

Обязательная (mandatory) связь описывает связь между независимой и зависимой сущностями. Все экземпляры зависимой (обязательной) сущности могут существовать только при наличии экземпляров независимой (необязательной) сущности, т. е. экземпляр обязательной сущности может существовать только при условии существования определенного экземпляра необязательной сущности.

В примере (рис. 4.26) подразумевается, что каждый автомобиль имеет, по крайней мере, одного водителя, но не каждый служащий управляет машиной.

Рис. 4.26. Обязательная связь

В слабой связи существование одной из сущностей, принадлежащей некоторому множеству (слабой) зависит от существования определенной сущности, принадлежащей другому множеству (сильной), т. е. экземпляр слабой сущности может быть идентифицирован только посредством экземпляра сильной сущности. Ключ сильной сущности является частью составного ключа слабой сущности.

Слабая связь всегда является бинарной и подразумевает обязательную связь для слабой сущности. Сущность может быть слабой в одной связи и сильной в другой, но не может быть слабой более, чем в одной связи. Слабая связь может не иметь атрибутов.

Пример слабой связи (рис. 4.27): ключ (номер) строки в документе может не быть уникальным и должен быть дополнен ключом документа.

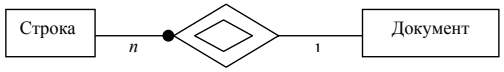


Рис. 4.27. Слабая связь

Связь «супертип-подтип» изображена на рис. 4.28. Общие характеристики (атрибуты) типа определяются в сущности-супертипе, сущность-подтип наследует все характеристики супертипа. Экземпляр подтипа существует только при условии существования определенного экземпляра супертипа. Подтип не может иметь ключа (он импортирует ключ из супертипа). Сущность, являющаяся супертипом в одной связи, может быть подтипом в другой связи. Связь супертипа не может иметь атрибутов.

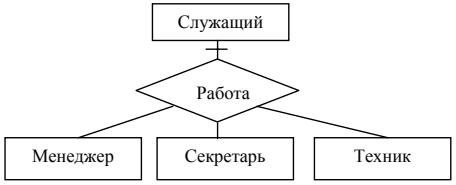


Рис. 4.28. Связь «супертип-подтип»

В ассоциативной связи каждый экземпляр связи (ассоциативный объект) может существовать только при условии существования определенных экземпляров каждой из взаимосвязанных сущностей. Ассоциативный объект — объект, являющийся одновременно сущностью и связью. Ассоциативная связь — это связь между несколькими независимыми сущностями и одной зависимой сущностью. Связь между независимыми сущностями имеет атрибуты, которые определяются в зависимой сущности. Таким образом, зависимая сущность определяется в терминах атрибутов связи между остальными сущностями [10].

В примере на рис. 4.29 самолет выполняет посадку на посадочную полосу в заданное время при определенной скорости и направлении ветра. Поскольку эти характеристики применимы только к конкретной посадке, они являются атрибутами посадки, а не самолета или посадочной полосы. Пилот, выполняющий посадку, связан гораздо сильнее с конкретной посадкой, чем с самолетом или посадочной полосой.

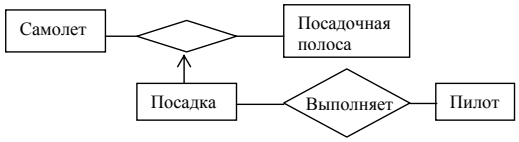


Рис. 4.29. Ассоциативная связь

Первичный ключ каждого типа сущности помечается звездочкой « * ».

ER-диаграмма должна подчиняться следующим правилам:

каждая сущность, каждый атрибут и каждая связь должны иметь имя (связь супертипа или ассоциативная связь может не иметь имени);

имя сущности должно быть уникально в рамках модели данных;

имя атрибута должно быть уникально в рамках сущности;

имя связи должно быть уникально, если для нее генерируется таблица БД;

каждый атрибут должен иметь определение типа данных;

сущность в необязательной связи должна иметь ключевой атрибут. То же самое относится к сильной сущности в слабой связи, супертипу в связи «супертип-подтип» и необязательной сущности в обязательной (полной) связи;

подтип в связи «супертип-подтип» не может иметь ключевой атрибут;

в ассоциативной или слабой связи может быть только одна ассоциативная (слабая) сущность;

связь не может быть одновременно обязательной, «супертип-подтип» или ассоциативной.

4.5. ПРИМЕР ИСПОЛЬЗОВАНИЯ СТРУКТУРНОГО ПОДХОДА

4.5.1. Описание предметной области

В данном примере используется методология Yourdon, реализованная в CASE-средстве Vantage Team Builder [10].

В качестве предметной области используется описание работы видеобиблиотеки, которая получает запросы на фильмы от клиентов и ленты, возвращаемые клиентами. Запросы рассматриваются администрацией видеобиблиотеки с использованием информации о клиентах, фильмах и лентах. При этом проверяется и обновляется список арендованных лент, а также проверяются записи о членстве в библиотеке. Администрация контролирует также возвраты лент, используя информацию о фильмах, лентах и список арендованных лент, который обновляется. Обработка запросов на фильмы и возвратов лент включает следующие действия: если клиент не является членом библиотеки, он не имеет права на аренду. Если требуемый фильм имеется в наличии, администрация информирует клиента об арендной плате. Однако, если клиент просрочил срок возврата имеющихся у него лент, ему не разрешается брать новые фильмы. Когда лента возвращается, администрация рассчитывает арендную плату плюс пени за несвоевременный возврат.

Видеобиблиотека получает новые ленты от своих поставщиков. Когда новые ленты поступают в библиотеку, необходимая информация о них фиксируется. Информация о членстве в библиотеке содержится отдельно от записей об аренде лент.

Администрация библиотеки регулярно готовит отчеты за определенный период времени о членах библиотеки, поставщиках лент, выдаче определенных лент и лентах, приобретенных библиотекой.

4.5.2. Организация проекта

Весь проект разделяется на 4 фазы: анализ, глобальное проектирование (проектирование архитектуры системы), детальное проектирование и реализация (программирование).

На фазе анализа строится модель среды (Environmental Model). Построение модели среды включает:

- анализ поведения системы (определение назначения ИС, построение начальной контекстной диаграммы потоков данных DFD и формирование матрицы списка событий ELM, построение контекстных диаграмм);
- анализ данных (определение состава потоков данных и построение диаграмм структур данных DSD, конструирование глобальной модели данных в виде ER-диаграммы).

Назначение ИС определяет соглашение между проектировщиками и заказчиками относительно назначения будущей ИС, общее описание ИС для самих проектировщиков и границы ИС. Назначение фиксируется как текстовый комментарий в «нулевом» процессе контекстной диаграммы.

Например, в данном случае назначение ИС формулируется следующим образом: ведение базы данных о членах библиотеки, фильмах, аренде и поставщиках. При этом руководство библиотеки должно иметь возможность получать различные виды отчетов для выполнения своих задач.

Перед построением контекстной DFD необходимо проанализировать внешние события (внешние объекты), оказывающие влияние на функционирование библиотеки. Эти объекты взаимодействуют с ИС путем информационного обмена с ней.

Из описания предметной области следует, что в процессе работы библиотеки участвуют следующие группы людей: клиенты, поставщики и руководство. Эти группы являются внешними объектами. Они не только взаимодействуют с системой, но также определяют ее границы и изображаются на начальной контекстной DFD как терминаторы (внешние сущности).

Начальная контекстная диаграмма изображена на рис. 4.30. В отличие от нотации Гейна-Сарсона внешние сущности обозначаются обычными прямоугольниками, а процессы — окружностями.

Список событий строится в виде матрицы ELM и описывает различные действия внешних сущностей и реакцию ИС на них. Эти действия представляют собой внешние события, воздействующие на библиотеку. Различают следующие типы событий:

Аббревиатура	Тип
NC	Нормальное управление
ND	Нормальные данные
NCD	Нормальное управление/данные
TC	Временное управление
TD	Временные данные
TCD	Временное управление/данные

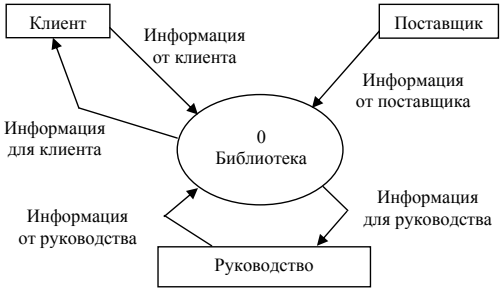


Рис. 4.30. Начальная контекстная диаграмма

Все действия помечаются как нормальные данные. Эти данные являются событиями, которые ИС воспринимает непосредственно, например изменение адреса клиента, которое должно быть сразу зарегистрировано. Они появляются в DFD в качестве содержимого потоков данных. Матрица списка событий имеет следующий вид:

№	Описание	Тип	Реакция
1	Клиент желает стать членом библиотеки	ND	Регистрация клиента в качестве члена библиотеки
2	Клиент сообщает об изменении адреса	ND	Регистрация измененного адреса клиента
3	Клиент запрашивает аренду фильма	ND	Рассмотрение запроса
4	Клиент возвращает фильм	ND	Регистрация возврата
5	Руководство предоставляет полномочия новому поставщику	ND	Регистрация поставщика
6	Поставщик сообщает об изменении адреса	ND	Регистрация измененного адреса поставщика
7	Поставщик направляет фильм в библиотеку	ND	Получение нового фильма
8	Руководство запрашивает новый отчет	ND	Формирование требуемого отчета для руководства

Для завершения анализа функционального аспекта поведения системы строится полная контекстная диаграмма, включающая диаграмму нулевого уровня. При этом процесс «библиотека» декомпозируется на 4 процесса, отражающие основные виды административной деятельности библиотеки. Существующие «абстрактные» потоки данных между терминаторами и процессами трансформируются в потоки, представляющие обмен данными на более конкретном уровне. Список событий показывает, какие потоки существуют на этом уровне: каждое событие из списка должно формировать некоторый поток (событие формирует входной поток, реакция — выходной поток). Один «абстрактный» поток может быть разделен на более чем один «конкретный» поток так, как это показано ниже:

Потоки на диаграмме верхнего уровня	Потоки на диаграмме нулевого уровня
Информация от клиента	Данные о клиенте, запрос об аренде
Информация для клиента	Членская карточка, ответ на запрос об аренде
Информация от руководства	Запрос отчета о новых членах, новый поставщик, запрос отчета о поставщиках, Запрос отчета об аренде, запрос отчета о фильмах
Информация для руководства	Отчет о новых членах, отчет о поставщиках, отчет об аренде, отчет о фильмах
Информация от поставщика	Данные о поставщике, новые фильмы

На приведенной ниже DFD (рис. 4.31) накопитель данных «библиотека» является глобальным или абстрактным представлением хранилища данных.

Анализ функционального аспекта поведения системы дает представление об обмене и преобразовании данных в системе. Взаимосвязь между «абстрактными» потоками данных и «конкретными» потоками данных на диаграмме нулевого уровня выражается в диаграммах структур данных (рис. 4.32).

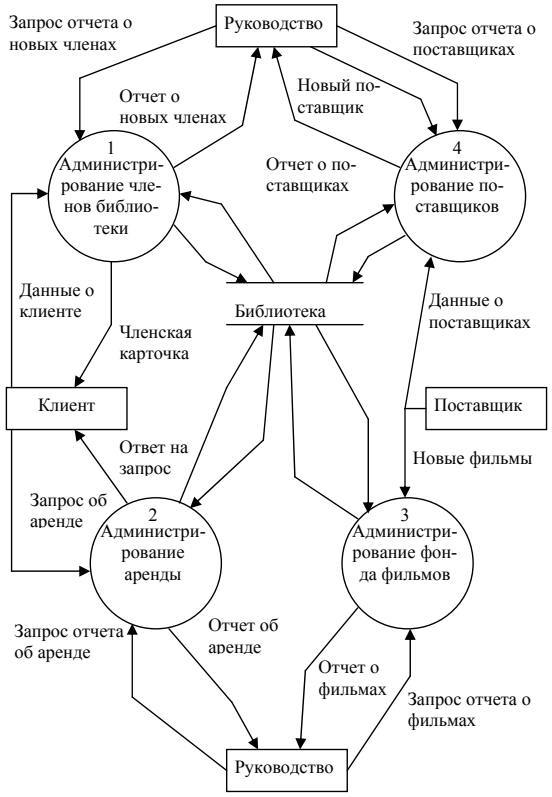


Рис. 4.31. Контекстная диаграмма

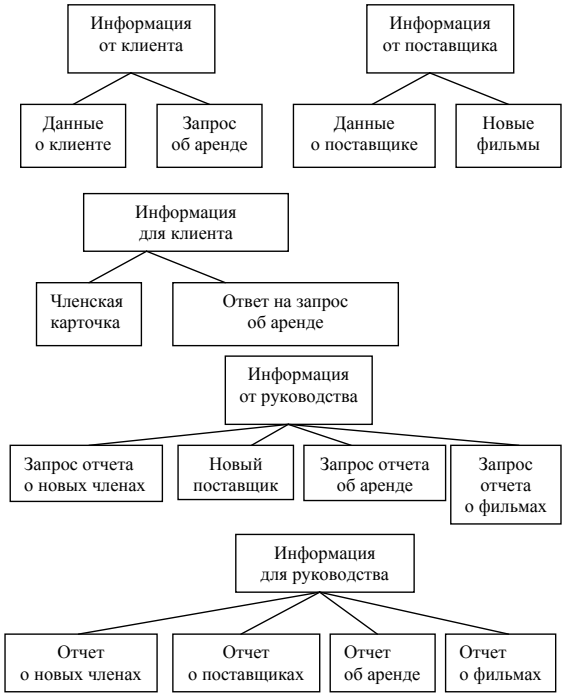


Рис. 4.32. Диаграмма структур данных

На фазе анализа строится глобальная модель данных, представляемая в виде диаграммы «сущность-связь» (рис. 4.33).

Между различными типами диаграмм существуют следующие взаимосвязи:

ELM — DFD: события — входные потоки, реакции — выходные потоки;

DFD — DSD: потоки данных — структуры данных верхнего уровня;

DFD — ERD: накопители данных — ER-диаграммы;

DSD — ERD: структуры данных нижнего уровня — атрибуты сущностей.

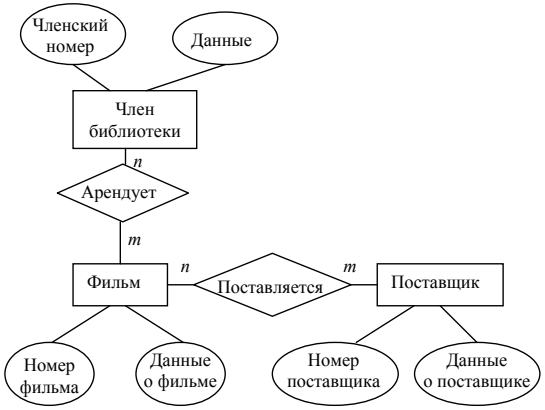


Рис. 4.33. Диаграмма «сущность-связь»

На фазе проектирования архитектуры строится предметная модель. Процесс построения предметной модели включает:

- детальное описание функционирования системы;
- дальнейший анализ используемых данных и построение логической модели данных для последующего проектирования базы данных;
- определение структуры пользовательского интерфейса, спецификации форм и порядка их появления;
- уточнение диаграмм потоков данных и списка событий, выделение среди процессов нижнего уровня интерактивных и неинтерактивных, определение для них мини-спецификаций.

Результатами проектирования архитектуры являются:

- модель процессов (диаграммы архитектуры системы SAD и мини-спецификации на структурированном языке);
- модель данных (ERD и подсхемы ERD);
- модель пользовательского интерфейса (классификация процессов на интерактивные и неинтерактивные функции, диаграмма последовательности форм FSD (Form Sequence Diagram), показывающая, какие формы появляются в приложении и в каком порядке. На FSD фиксируется набор и структура вызовов экранных форм. Диаграммы FSD образуют иерархию, на вершине которой находится главная форма приложения, реализующего подсистему. На втором уровне находятся формы, реализующие процессы нижнего уровня функциональной структуры, зафиксированной на диаграммах SAD.

На фазе детального проектирования строится модульная модель. Под модульной моделью понимается реальная модель проектируемой прикладной системы. Процесс ее построения включает:

- уточнение модели базы данных для последующей генерации SQL-предложений;
- уточнение структуры пользовательского интерфейса;
- построение структурных схем, отражающих логику работы пользовательского интерфейса, модель бизнес-логики SCD (Structure Charts Diagram) и привязка их к формам.

Результатами детального проектирования являются:

- модель процессов (структурные схемы интерактивных и неинтерактивных функций);
- модель данных (определение в ERD всех необходимых параметров для приложений);
- модель пользовательского интерфейса (диаграмма последовательности форм FSD, показывающая, какие формы появляются в приложении и в каком порядке, взаимосвязь между каждой формой и определенной структурной схемой, взаимосвязь между каждой формой и одной или более сущностями в ERD).

На фазе реализации строится реализационная модель. Процесс ее построения включает:

- генерацию SQL-предложений, определяющих структуру целевой БД (таблицы, индексы, ограничения целостности);
- уточнение структурных схем SCD и диаграмм последовательности форм FSD с последующей генерацией кода приложений.

На основе анализа потоков данных и взаимодействия процессов с хранилищами данных осуществляется окончательное выделение подсистем (предварительное должно быть сделано и зафиксировано на этапе формулировки требований в техническом задании). При выделении подсистем необходимо руководствоваться принципом функциональной связанности и принципом минимизации информационной зависимости. Необходимо учитывать, что на основании таких элементов подсистемы, как процессы и данные, на этапе разработки должно быть создано приложение, способное функционировать самостоятельно. С другой стороны, при группировке процессов и данных в подсистемы необходимо учитывать требования к конфигурированию продукта, если они были сформулированы на этапе анализа.

4.6. ОБЩАЯ ХАРАКТЕРИСТИКА И КЛАССИФИКАЦИЯ CASE-СРЕДСТВ

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО.

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различ-

ных CASE-средств, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее следующими основными характерными особенностями:

- мощными графическими средствами для описания и документирования ИС, обеспечивающими удобный интерфейс с разработчиком и развивающими его творческие возможности;
- возможностью интеграции отдельных компонентов CASE-средств, обеспечивающей управляемость процессом разработки ИС;
- возможностью использования специальным образом организованного хранилища проектных метаданных (репозитория).

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный жизненный цикл ПО) содержит следующие компоненты;

- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;
- средства разработки приложений, включая языки 4GL и генераторы кодов;
- средства конфигурационного управления;
- средства документирования;
- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы жизненного цикла. Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает: отдельные локальные средства, решающие небольшие автономные задачи (tools); набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС (toolkit); полностью интегрированные средства, поддерживающие весь жизненный цикл ИС и связанные общим репозиторием. Помимо этого CASE-средства можно классифицировать по следующим признакам:

- применяемым методологиям и моделям систем и БД;
- степени интегрированности с СУБД;
- доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств и включает следующие основные типы:

- средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области (Design/IDEF (Meta Software), BPwin (Logic Works));
- средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Te-am Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (Макро-Проджект)). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;
- средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;
- средства разработки приложений. К ним относятся средства 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) и др.) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично — в Silverrun;
- средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin и S-Designor. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Вспомогательные типы CASE-средств включают:

- средства планирования и управления проектом (SE Companion, Microsoft Project и др.);
- средства конфигурационного управления (PVCS (Intersolv));
- средства тестирования (Quality Works (Segue Software));
- средства документирования (SoDA (Rational Software)).

На сегодняшний день российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin+BPwin;
- S-Designor;
- CASE.Аналитик.

Описание перечисленных CASE-средств приведено в разделе 5. Кроме того, на рынке постоянно появляются как новые для отечественных пользователей системы (например, CASE /4/0, PRO-IV, System Architect, Visible Analyst Workbench, EasyCASE), так и новые версии и модификации перечисленных систем.

4.7. ХАРАКТЕРИСТИКИ СОВРЕМЕННЫХ CASE-СРЕДСТВ

4.7.1. Silverrun

CASE-средство Silverrun американской фирмы Computer Systems Advisers, Inc. (CSA) используется для анализа и проектирования ИС бизнес-класса [10] и ориентировано в большей степени на спиральную модель жизненного цикла. Оно применимо для поддержки любой методологии, основанной на раздельном построении функциональной и информационной моделей (диаграмм потоков данных и диаграмм «сущность-связь»).

Настройка на конкретную методологию обеспечивается выбором требуемой графической нотации моделей и набора правил проверки проектных спецификаций. В системе имеются готовые настройки для наиболее распространенных методологий: DATARUN (основная методология, поддерживаемая Silverrun), Gane/Sarson, Yourdon/DeMarco, Merise, Ward/Mellor, Information Engineering. Для каждого понятия, введенного в проекте, имеется возможность добавления собственных описателей. Архитектура Silverrun позволяет наращивать среду разработки по мере необходимости.

Silverrun имеет модульную структуру и состоит из четырех модулей, каждый из которых является самостоятельным продуктом и может приобретаться и использоваться без связи с остальными модулями.

Модуль построения моделей бизнес-процессов в форме диаграмм потоков данных BPM (Business Process Modeler) позволяет моделировать функционирование обследуемой организации или создаваемой ИС. В модуле BPM обеспечена возможность работы с моделями большой сложности: автоматическая перенумерация, работа с деревом процессов (включая визуальное перетаскивание ветвей), отсоединение и присоединение частей модели для коллективной разработки. Диаграммы могут изображаться в нескольких предопределенных нотациях, включая Yourdon/DeMarco и Gane/Sarson. Имеется также возможность создавать собственные нотации, в том числе добавлять в число изображаемых на схеме дескрипторов определенные пользователем поля.

Модуль концептуального моделирования данных ERX (Entity-Relationship eXpert) обеспечивает построение моделей данных «сущность-связь», не привязанных к конкретной реализации. Этот модуль имеет встроенную экспертную систему, позволяющую создать корректную нормализованную модель данных посредством ответов на содержательные вопросы о взаимосвязи данных. Возможно автоматическое построение модели данных из описаний структур данных. Анализ функциональных зависимостей атрибутов дает возможность проверить соответствие модели требованиям третьей нормальной формы и обеспечить их выполнение. Проверенная модель передается в модуль RDM.

Модуль реляционного моделирования RDM (Relational Data Modeler) позволяет создавать детализированные модели «сущность-связь», предназначенные для реализации в реляционной базе данных. В этом модуле документируются все конструкции, связанные с постро-ением базы данных: индексы, триггеры, хранимые процедуры и т. д. Гибкая изменяемая нотация и расширяемость репозитория позволяют работать по любой методологии. Возможность создавать подсхемы соответствует подходу ANSI SPARC к представлению схемы базы данных. На языке подсхем моделируются как узлы распределенной обработки, так и пользовательские представления. Этот модуль обеспечивает проектирование и полное документирование реляционных баз данных.

Менеджер репозитория рабочей группы WRM (Workgroup Repo-sitory Manager) применяется как словарь данных для хранения общей для всех моделей информации, а также обеспечивает интеграцию модулей Silverrun в единую среду проектирования.

Платой за высокую гибкость и разнообразие изобразительных средств построения моделей является такой недостаток Silverrun, как отсутствие жесткого взаимного контроля между компонентами различных моделей (например, возможности автоматического распространения изменений между DFD различных уровней декомпозиции). Следует, однако, отметить, что этот недостаток может иметь существенное значение только в случае использования каскадной модели жизненного цикла ПО.

Для автоматической генерации схем баз данных у Silverrun существуют мосты к наиболее распространенным СУБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server, SQLBase, Sybase. Для передачи данных в средства разработки приложений имеются мосты к языкам 4GL: JAM, PowerBuilder, SQL Windows, Uniface, NewEra, Delphi. Все мосты позволяют загрузить в Silverrun RDM информацию из каталогов соответствующих СУБД или языков 4GL. Это позволяет документировать, перепроектировать или переносить на новые платформы уже на-

ходящиеся в эксплуатации базы данных и прикладные системы. При использовании моста Silverrun расширяет свой внутренний репозиторий специфичными для целевой системы атрибутами. После определения значений этих атрибутов генератор приложений переносит их во внутренний каталог среды разработки или использует при генерации кода на языке SQL. Таким образом можно полностью определить ядро базы данных с использованием всех возможностей конкретной СУБД: тригге-ров, хранимых процедур, ограничений ссылочной целостности. При со-здании приложения на языке 4GL данные, перенесенные из репозитория Silverrun, используются либо для автоматической генерации интерфейсных объектов, либо для быстрого их создания вручную.

Для обмена данными с другими средствами автоматизации проектирования, создания специализированных процедур анализа и проверки проектных спецификаций, составления специализированных от-четов в соответствии с различными стандартами в системе Silverrun имеется 3 способа выдачи проектной информации во внешние файлы:

1) система отчетов. Можно, определив содержимое отчета по репозиторию, выдать отчет в текстовый файл. Этот файл можно затем загрузить в текстовый редактор или включить в другой отчет;

2) система экспорта/импорта. Для более полного контроля над структурой файлов в системе экспорта/импорта имеется возможность определять не только содержимое экспортного файла, но и разделители записей, полей в записях, маркеры начала и конца текстовых полей. Файлы с указанной структурой можно не только формировать, но и загружать в репозиторий. Это дает возможность обмениваться данными с различными системами: другими CASE-средствами, СУБД, текстовыми редакторами и электронными таблицами;

3) хранение репозитория во внешних файлах через ODBC-драй-веры. Для доступа к данным репозитория из наиболее распространенных систем управления базами данных обеспечена возможность хранить всю проектную информацию непосредственно в формате этих СУБД.

Групповая работа поддерживается в системе Silverrun двумя способами:

1) в стандартной однопользовательской версии имеется механизм контролируемого разделения и слияния моделей. Разделив модель на части, можно раздать их нескольким разработчикам. После детальной доработки модели объединяются в единые спецификации;

2) сетевая версия Silverrun позволяет осуществлять одновременную групповую работу с моделями, хранящимися в сетевом репозитории на базе СУБД Oracle, Sybase или Informix. При этом несколько разработчиков могут работать с одной и той же моделью, так как блокировка объектов происходит на уровне отдельных элементов модели.

Имеются реализации Silverrun трех платформ — MS Windows, Macintosh и OS/2 Presentation Manager — с возможностью обмена проектными данными между ними.

4.7.2. Designer/2000 + Developer/2000

CASE-средство Designer/2000 2.0 фирмы ORACLE является интегрированным CASE-средством, обеспечивающим в совокупности со средствами разработки приложений Developer/2000 поддержку полного жизненного цикла ПО для систем, использующих СУБД ORACLE.

Designer/2000 представляет собой семейство методологий и поддерживающих их программных продуктов. Базовая методология Designer/2000 (CASE*Method) — структурная методология проектирования систем, полностью охватывающая все этапы жизненного цикла ИС. В соответствии с этой методологией на этапе планирования определяются цели создания системы, приоритеты и ограничения, разрабатывается системная архитектура и план разработки ИС. В процессе анализа строятся модель информационных потребностей (диаграмма «сущность-связь»), диаграмма функциональной иерархии (на основе функциональной декомпозиции ИС), матрица перекрестных ссылок и диаграмма потоков данных.

На этапе проектирования разрабатывается подробная архитектура ИС, проектируется схема реляционной БД и программные модули, устанавливаются перекрестные ссылки между компонентами ИС для анализа их взаимного влияния и контроля за изменениями.

На этапе реализации создается БД, строятся прикладные системы, производится их тестирование, проверка качества и соответствия требованиям пользователей, создаются системная документация, матери-алы для обучения и руководства пользователей. На этапах эксплуатации и сопровождения анализируются производительность и целостность системы, выполняется поддержка и при необходимости модификация ИС.

Designer/2000 обеспечивает графический интерфейс при разработке различных моделей (диаграмм) предметной области. В процессе построения моделей информация о них заносится в репозиторий. В состав Designer/2000 входят следующие компоненты:

Repository Administrator — средства управления репозиторием (создание и удаление приложений, управление доступом к данным со стороны различных пользователей, экспорт и импорт данных);

Repository Object Navigator — средства доступа к репозиторию, обеспечивающие многооконный объектно-ориентированный интерфейс доступа ко всем элементам репозитория;

Process Modeller — средство анализа и моделирования деловой деятельности, основывающееся на концепциях реинжиниринга бизнес-процессов BPR (Business Process Reengineering) и глобальной системы управления качеством TQM (Total Quality Management);

Systems Modeller — набор средств построения функциональных и информационных моделей проектируемой ИС, включающий средства для построения диаграмм «сущность-связь» (Entity-Relationship Diag-rammer), диаграмм функциональных иерархий (Function Hierarchy Diagrammer), диаграмм потоков данных (Data Flow Diagrammer) и средство анализа и модификации связей объектов репозитория различных типов (Matrix Diagrammer);

Systems Designer — набор средств проектирования ИС, включающий средство построения структуры реляционной базы данных (Data Diagrammer), а также средства построения диаграмм, отображающих взаимодействие с данными, иерархию, структуру и логику приложений, реализуемую хранимыми процедурами на языке PL/SQL (Module Data Diagrammer, Module Structure Diagrammer и Module Logic Navigator);

Server Generator — генератор описаний объектов БД ORACLE (таблиц, индексов, ключей, последовательностей и т. д.). Помимо продуктов ORACLE, генерация и реинжиниринг БД может выполняться для СУБД Informix, DB/2, Microsoft SQL Server, Sybase, а также для стандарта ANSI SQL DDL и баз данных, доступ к которым реализуется посредством ODBC;

Forms Generator — генератор приложений для ORACLE Forms. Генерируемые приложения включают в себя различные экранные формы, средства контроля данных, проверки ограничений целостности и автоматические подсказки. Дальнейшая работа с приложением выполняется в среде Developer/2000;

Repository Reports — генератор стандартных отчетов, интегрированный с ORACLE Reports и позволяющий русифицировать отчеты, а также изменять структурное представление информации.

Репозиторий Designer/2000 представляет собой хранилище всех проектных данных и может работать в многопользовательском режиме, обеспечивая параллельное обновление информации несколькими разработчиками. В процессе проектирования автоматически поддерживаются перекрестные ссылки между объектами словаря и могут генерироваться более 70 стандартных отчетов о моделируемой предметной области. Физическая среда хранения репозитория — база данных ORACLE.

Генерация приложений, помимо продуктов ORACLE, выполняется также для Visual Basic.

Designer/2000 можно интегрировать с другими средствами, используя открытый интерфейс приложений API (Application Program-ming Interface). Кроме того, можно использовать средство ORACLE CASE Exchange для экспорта/импорта объектов репозитория с целью обмена информацией с другими CASE-средствами.

Developer/2000 обеспечивает разработку переносимых приложений, работающих в графической среде Windows, Macintosh или Motif. В среде Windows интеграция приложений Developer/2000 с другими средствами реализуется через механизм OLE и управляющие элементы VBX. Взаимодействие приложений с другими СУБД (DB/2, DB2/400, Rdb) реализуется с помощью средств ORACLE Client Adapter для ODBC, ORACLE Open Gateway и API.

Среда функционирования Designer/2000 и Developer/2000 — Windows 3.x, Windows 95, Windows NT.

4.7.3. Локальные средства (ERwin, BPwin, CASE.Аналитик)

ERwin — средство концептуального моделирования БД, использующее методологию IDEF1X. ERwin реализует проектирование схемы БД, генерацию ее описания на языке целевой СУБД (ORACLE, Informix, Ingres, Sybase, DB/2, Microsoft SQL Server, Progress и др.) и реинжиниринг существующей БД. ERwin выпускается в нескольких различных конфигурациях, ориентированных на наиболее распространенные средства разработки приложений 4GL. Версия ERwin/OPEN полностью совместима со средствами разработки приложений PowerBuilder и SQLWindows и позволяет экспортировать описание спроектированной БД непосредственно в репозитории данных средств.

Для ряда средств разработки приложений (PowerBuilder, SQLWindows, Delphi, Visual Basic) выполняется генерация форм и прототипов приложений.

Сетевая версия ERwin ModelMart обеспечивает согласованное проектирование БД и приложений в рамках рабочей группы.

BPwin — средство функционального моделирования, реализующее методологию IDEF0.

CASE.Аналитик 1.1 является практически единственным в настоящее время конкурентоспособным отечественным CASE-средством функционального моделирования и реализует построение диаграмм потоков данных в соответствии с методологией, описанной в подразделе 2.3. Его основные функции:

построение и редактирование DFD;

анализ диаграмм и проектных спецификаций на полноту и непротиворечивость;

получение разнообразных отчетов по проекту;

генерация макетов документов в соответствии с требованиями ГОСТ 19.XXX и 34.XXX.

Среда функционирования: MS Windows 3.x или Windows 95.

База данных проекта реализована в формате СУБД Paradox и является открытой для доступа.

С помощью отдельного программного продукта (Catherine) выполняется обмен данными с CASE-средством ERwin. При этом из проекта, выполненного в CASE.Аналитик, экспортируется описание структур данных и накопителей данных, которое по определенным правилам формирует описание сущностей и их атрибутов.

4.7.4. Объектно-ориентированные CASE-средства (Rational Rose)

Rational Rose — CASE-средство фирмы Rational Software Corpo-ration (США) — предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации [10]. Rational Rose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: Буча, Рамбо и Джекобсона. Разработанная ими универсальная нотация для моделирования объектов UML (Unified Modeling Language) претендует на роль стандарта в области объектно-ориенти-рованного анализа и проектирования. Конкретный вариант Rational Rose определяется языком, на котором генерируются коды программ (C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro). Основной вариант Rational Rose/C++ позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на языке C++. Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонентов в новых проектах.

В основе работы Rational Rose лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты. В их число входят диаграммы классов, состояний, сценариев, модулей, процессов.

В составе Rational Rose можно выделить 6 основных структурных компонент: репозиторий, графический интерфейс пользователя, средства просмотра проекта (browser), средства контроля проекта, средства сбора статистики и генератор документов. К ним добавляются генератор кодов (индивидуальный для каждого языка) и анализатор для C++, обеспечивающий реинжиниринг — восстановление модели проекта по исходным текстам программ.

Репозиторий представляет собой объектно-ориентированную базу данных. Средства просмотра обеспечивают «навигацию» по проекту, в том числе перемещение по иерархиям классов и подсистем, переключение от одного вида диаграмм к другому и т. д. Средства контроля и сбора статистики дают возможность находить и устранять ошибки по мере развития проекта, а не после завершения его описания. Генератор отчетов формирует тексты выходных документов на основе содержащейся в репозитории информации.

Средства автоматической генерации кодов программ на языке C++, используя информацию, содержащуюся в логической и физической моделях проекта, формируют файлы заголовков и файлы описаний классов и объектов. Создаваемый таким образом скелет программы может быть уточнен путем прямого программирования на языке C++. Анализатор кодов C++ реализован в виде отдельного программного модуля. Его назначение состоит в том, чтобы создавать модули проектов в форме Rational Rose на основе информации, содержащейся в определяемых пользователем исходных текстах на C++. В процессе работы анализатор осуществляет контроль правильности исходных текстов и диагностику ошибок. Модель, полученная в результате его работы, может целиком или фрагментарно использоваться в различных проектах. Анализатор обладает широкими возможностями настройки по входу и выходу. Например, можно определить типы исходных файлов, базовый компилятор, задать информацию, которая должна быть включена в формируемую модель, и элементы выходной модели, которые следует выводить на экран. Таким образом, Rational Rose/C++ обеспечивает возможность повторного использования программных компонентов.

В результате разработки проекта с помощью CASE-средства Rational Rose формируются следующие документы:

диаграммы классов;
диаграммы состояний;
диаграммы сценариев;
диаграммы модулей;
диаграммы процессов;
спецификации классов, объектов, атрибутов и операций;
заготовки текстов программ;
модель разрабатываемой программной системы.

Последний из перечисленных документов является текстовым файлом, содержащим всю необходимую информацию о проекте (в том числе необходимую для получения всех диаграмм и спецификаций).

Тексты программ являются заготовками для последующей работы программистов. Они формируются в рабочем каталоге в виде файлов типов «.h» (заголовки, содержащие описания классов) и «.cpp» (заготовки программ для методов). Система включает в программные файлы собственные комментарии, которые начинаются с последовательности символов «//##». Состав информации, включаемой в программные файлы, определяется либо по умолчанию, либо по усмотрению пользователя. В дальнейшем эти исходные тексты развиваются программистами в полноценные программы.

Rational Rose интегрируется со средством PVCS для организации групповой работы и управления проектом и со средством SoDA для документирования проектов. Интеграция Rational Rose и SoDA обеспечивается средствами SoDA.

Для организации групповой работы в Rational Rose возможно разбиение модели на управляемые подмодели. Каждая из них независимо сохраняется на диске или загружается в модель. В качестве подмодели может выступать категория классов или подсистема.

Для управляемой подмодели предусмотрены операции:
загрузка подмодели в память;
выгрузка подмодели из памяти;
сохранение подмодели на диске в виде отдельного файла;
установка защиты от модификации;
замена подмодели в памяти на новую.

Наиболее эффективно групповая работа организуется при интеграции Rational Rose со специальными средствами управления конфигурацией и контроля версий (PVCS). В этом случае защита от модификации устанавливается на все управляемые подмодели, кроме тех, которые выделены конкретному разработчику, признак защиты от записи устанавливается для файлов, которые содержат подмодели, поэтому при считывании «чужих» подмоделей защита их от модификации сохраняется и случайные воздействия окажутся невозможными.

Rational Rose функционирует на различных платформах: IBM PC (в среде Windows), Sun SPARC stations (UNIX, Solaris, SunOS), Hewlett-Packard (HP UX), IBM RS/6000 (AIX).

Контрольные вопросы

1. Какие основные задачи призваны решать CASE-технологии?
2. В чем сущность структурного подхода к проектированию ИС?
3. Назовите основные элементы методологии SADT.
4. Из чего состоит функциональная модель в методологии SADT?
5. Приведите основные положения методологии Гейна-Сарсона моделирования потоков данных.
6. Из каких шагов состоит процесс построения иерархии диаграмм потоков данных?
7. Какие методологии моделирования данных Вам известны?
8. В чем отличие CASE -метода Баркера и методологии IDEF1?
9. Какие инструментальные CASE-средства Вам известны? Дайте им краткую характеристику.

5. ГЕОИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

5.1. ИСТОКИ ВОЗНИКНОВЕНИЯ

Первые геоинформационные системы (ГИС) были созданы в США и Канаде в конце 60-х годов XX в. К концу 70-х годов в США насчитывалось уже до 60-ти эксплуатируемых ГИС различного назначения. Постепенно ГИС становятся настолько важным направлением научных и практических работ, что во многих странах их планирование и разработка осуществляются на государственном уровне. В 80-е годы практически во всех промышленно развитых странах возникло понимание необходимости крупных капиталовложений в эту стратегически важную и быстро развивающуюся новую область информатики. Работы по ГИС ведутся, например, в Великобритании, Японии, Германии, Швеции, Норвегии и даже в Китае.

Первые эффективные и мощные ГИС были созданы в недрах военно-промышленного комплекса США. Практический опыт, накопленный при эксплуатации ГИС в военной области, сейчас успешно используется и вне военной сферы для управления ресурсами, экономикой и для планирования. Общепризнанными мировыми лидерами в области ГИС являются фирма INTERGRAPH и институт ESRI (Envi-ronmental Systems Research Institute Inc.) в США, разработки которых в настоящее время стали, фактически, эталонным подходом к проблематике ГИС. Фирма INTERGRAPH существует около 25 лет. За это время она выполнила, в частности, заказы следующих организаций:

- картографического управления МО США,
- ВВС США,
- армии США.

Что касается гражданских работ, то здесь фирма INTERGRAPH специализируется в следующих направлениях:

- картировании для государственных органов,
- землепользовании,
- муниципальных системах,
- энергоснабжении,
- разведке природных месторождений,
- телефонии.

Годовые объемы продаж программного обеспечения и технических средств, производимых фирмой, оцениваются в сотни миллионов долларов.

По аналогичным направлениям работает институт ESRI. ВВС США имеют контракты с ESRI на поставку программных средств, обе-спечивающих привязку шахт пусковых установок межконтинентальных баллистических ракет стратегического назначения, а также программных средств для навигации самолетов. В отличие от фирмы INTERGRAPH институт ESRI не занимается производством аппаратного обеспечения, и его программные продукты функционируют на стандартных технических средствах. Объемы продаж программных средств фирмой INTERGRAPH (без аппаратного обеспечения) и институтом ESRI сравнимы с объемами продаж для графических станций. Если же говорить о PC-совместимых ПЭВМ, то здесь несомненным мировым лидером является институт ESRI, а фирма INTER-GRAPH занимает только 4-е место.

Особенно бурными темпами технология ГИС стала развиваться с 1987 года. Это можно объяснить следующими причинами:

накоплением огромного объема геоинформации;

достижениями в технической области, например переходом на гигабайтовые и терабайтовые накопители информации;

научными достижениями в области ГИС;

наличием положительного мирового опыта, свидетельствующего об эффективности инвестиций в ГИС.

По западным оценкам количество действующих ГИС после 2000 года может достигнуть 15-ти миллионов. Россия также не осталась в стороне от этого процесса. Сейчас мы наблюдаем стремительное продвижение геоинформационных технологий, осознание того факта, что многие проблемы управления территорией (городом, областью и т. п.) связаны с географической информацией и что естественным путем их решения является применение концепции ГИС. Этому же процессу способствует и постепенный переход на рыночную экономику, и возникающие в связи с этим новые вопросы собственности (в частности, на землю), конверсия, проводимая в ВПК, более свободный обмен научными идеями и техническими достижениями с развитыми странами Запада. Конверсия способствовала быстрому росту в гражданском секторе отечественных организаций, предлагающих свои услуги для внедрения геоинформационных технологий различного назначения.

5.2. ОПРЕДЕЛЕНИЕ ГИС

Попытаемся разобраться с термином «геоинформационная система» и связанными с этим понятием вопросами. Можно дать следующее определение: геоинформационная система (географическая информационная система) — это интегрированная сумма компьютерных технологий для управления ресурсами территорий путем ввода, обработки, хранения, передачи, анализа и представления взаимозависимых географических и алфавитно-цифровых данных.

ГИС — это инструмент для принятия решений в области управления и экономики, это средство планирования и развития. ГИС предоставляет единую пространственную систему отсчета для накопленных и вновь вводимых данных, обеспечивая при этом интеграцию графических и алфавитно-цифровых данных.

Попытаемся осветить некоторые программные и технические аспекты проблематики ГИС. Специализированные программные средства (поставляемые как продукт), используемые для разработки и функционирования ГИС, будем называть ГИС-продуктами. В настоящее время их количество в мире достигает 300-т. Кроме того, сложившейся практикой считается применение в области ГИС различных элементов современных информационных технологий:

операционных систем;
концепций открытой системы;
коммуникационных протоколов;
графического пользовательского интерфейса;
реляционных СУБД с языком запросов SQL;
графических БД;
стандартных растровых и векторных форматов графических данных.

Как правило, ГИС-продукты имеют модульную структуру и ясный интерфейс между модулями, что позволяет постепенно наращивать программные мощности эксплуатируемой системы.

Вторым важным свойством ГИС-продуктов является их многоплатформенность, т. е. реализация на различных видах техники и для различных операционных систем. Например, в качестве файл-сервера может быть выбрана ЭВМ типа VAX, рабочая станция — под UNIX или PC NetWare, а в качестве рабочих мест — графическая рабочая станция под UNIX или PC с DOS или Windows. Как правило, реализуется возможность и централизованной, и децентрализованной обработки и хранения данных. Это свойство позволяет по мере необходимости переходить к более производительным конфигурациям вычислительной техники.

Третье свойство ГИС-продуктов — это их представление как программной среды для разработки собственных программных приложений ГИС, т. е. тех функций, которые не могут быть реализованы или реализуются неэффективно непосредственно ГИС-продуктом. Здесь же можно отметить и мощные средства для конвертирования графических и неграфических данных.

Отсюда можно сделать вывод, что ГИС как программно-техни-ческий комплекс — это гибкая развивающаяся среда, пользуясь которой Вы можете быть уверены в том, что вместе с качественным ростом Ваших потребностей ГИС естественным образом предоставит Вам возможность использовать более мощную и эффективную технологическую среду без потери уже накопленных данных и алгоритмов.

С другой стороны, как это ни печально, многие современные ГИС рассчитаны на профессионалов в области ГИС. Поэтому правильным на сегодняшний день выходом является создание геоинформационного центра для эксплуатации ГИС, который в принципе может взять на себя решение такой сложной и трудоемкой задачи, как ведение графической информации. Правда, уже сейчас намечаются пути для привлечения в ГИС и непрофессионалов. Этому могут способствовать такие новейшие продукты, как ARC/VIEW института ESRI или SICAD/VIEW фирмы SIEMENS NIXDORF. Специалисты отмечают, что ГИС-продукты ведущих фирм становятся все более и более схожими, а их функциональные отличия продолжают уменьшаться. Это означает, что в конечном итоге и геоинформационные технологии в каком-то смысле могут быть стандартизованы и точно описаны.

В настоящее время многие регионы России пришли к выводу о необходимости автоматизации управления территорией на основе концепции о многоцелевой многопользовательской ГИС муниципального уровня. Понятно, что проектирование такой системы еще более усложнено и требует большой организационной подготовки. Не секрет, что многие организации, обладающие некоей частью картографического материала, пытаются искать самостоятельные решения в области ГИС, что ведет к неоправданному параллелизму в работах, отсутствию концептуального подхода, к неясным перспективам развития, невозможности занесения информации отраслевых уровней послойно. Если удастся объединить усилия различных местных организаций с часто несовпадающими интересами в области ГИС, то это явится первым шагом к успешному проектированию муниципальной ГИС. В ходе проектирования такой ГИС необходимо выполнить большой объем организационных работ:

определить владельцев исходной информации, оценить ее качество;
провести инвентаризацию, определить методы получения недостающей информации;
определить состав первоочередных и потенциальных пользователей;
определить требуемые уровни точности информации и масштабы представления географических данных;
определить требования к конечной продукции.

Следует также учесть, что многие задачи требуют привлечения значительных научных сил, так как имеют чисто научную постановку, например задача комплексной оценки земли.

5.3. ОСНОВНЫЕ ПОНЯТИЯ

Для дальнейшего рассмотрения геоинформационной технологии необходимо ввести несколько общих понятий.

В геоинформационных системах при переводе карт в цифровую векторную форму пространственная и содержательная определенность объектов фиксируется по-разному. Собственно в цифровой карте фиксируются пространственные объекты, связи и отношения между ними, а также пользовательские идентификаторы пространственных объектов, обеспечивающие связь с их содержательными характеристиками. Содержательные характеристики объектов фиксируются в виде таблиц, каждая запись в которых соотносится с определенным пространственным объектом цифровой карты через пользовательский идентификатор, указанный и в записи, и в цифровой карте. Кроме того, на более высоком уровне содержательная определенность объектов фиксируется в принятой схеме выделения на исходной карте конкретных слоев (например, слой месторождений нефти, слой месторождений газа, слой административного деления и др.), а пространственная определенность — в выделении слоев цифровых карт по типу пространственных объектов (например, месторождения нефти, выражающиеся в масштабе карты как полигоны и как точки, разносятся в соответствующие слои полигональных и точечных объектов соответственно для удобства редактирования и идентификации).

При оцифровке карт, например с помощью дигитайзера, они представляются в векторном формате, где элементарными объектами являются точки, считываемые дигитайзером, а структура связей между этими точками, формирующая на их основе более сложные объекты (отрезки, дуги, полигоны) задается обслуживающими ввод и редактирование программными средствами.

При оцифровке карт выделяются три типа объектов, к которым можно отнести любой из имеющихся на карте [12]:

1) ***точечный объект***. Объект, локализованный в пункте, чьи размеры слишком малы, чтобы можно было отразить его форму (границы, площадь) в масштабе карты. Может также представлять некий условный объект, не имеющий размеров, например отметку высот;

2) ***линейный объект*** — объект, локализованный в виде линии, чья ширина не выражается в масштабе карты-источника (река, дорога и т. д.) Может также представлять некий условный объект, например границу;

3) ***полигональный объект*** — объект, имеющий площадь, выражающуюся в масштабе карты-источника. Определяется замкнутым контуром и его внутренней областью, например лес, озеро.

При переводе карты в цифровую форму объекты абстрагируются от своей географической (или геологической, например) сущности, и работа с ними как с пространственными объектами в среде цифровой карты проводится, опираясь на следующие определения:

точка — пара координат *X*, *Y*;

отрезок — линия, соединяющая две точки;

вершина (вертекс) — начальная или конечная точка отрезка;

дуга (линия) — упорядоченный набор связных отрезков (или вершин);

узел — начальная или конечная вершина дуги;

висячий узел — узел, принадлежащий только одной дуге, у которой начальная и конечная вершины не совпадают;

псевдоузел — узел, принадлежащий только двум дугам либо одной дуге (замкнутой дуге), у которой начальная и конечная вершины совпадают. Исключением является узел, принадлежащий двум дугам, одна из которых самозамкнута в этом узле, а другая примыкает к ней в этом узле (такой узел является нормальным);

нормальный узел — узел, принадлежащий трем и более дугам. Исключением является узел, принадлежащий двум дугам, одна из которых самозамкнута в этом узле, а другая примыкает к ней в этом узле (такой узел тоже является нормальным);

висячая дуга — дуга, имеющая висячий узел;

замкнутая дуга — дуга, у которой совпадают начальная и конечная вершины (у такой дуги имеется только один узел);

полигон — единичная область, ограниченная (находящаяся внутри) замкнутой дугой или упорядоченным набором связных дуг, которые образуют замкнутый контур;

покрытие — набор файлов, фиксирующий в виде цифровых записей пространственные объекты (точки, дуги, полигоны) и структуру отношений между ними. Пустое покрытие — покрытие, в котором отсутствуют пространственные объекты;

слой — покрытие, рассматриваемое в контексте его содержательной определенности (растительность, рельеф, административное деление и т. п.) или его статуса в среде редактора (активный слой, пассивный слой);

цифровая карта — цифровая модель геосистемы, представленная в виде слоя или композиции нескольких слоев;

внутренний идентификатор пространственного объекта — целое число, являющееся служебным идентификатором системы (уникальное для каждого объекта данного покрытия и назначаемое автоматически в процессе работы редактора), может изменяться системой в процессе работы;

пользовательский идентификатор пространственного объекта — целое число, предназначенное для связи объектов цифровой карты с базой (таблицами) тематических данных, назначаемое пользователем. Пользовательские идентификаторы назначаются или изменяются только пользователем.

Каждому слою карты можно поставить в соответствие несколько таблиц БД, содержащих атрибутивную информацию об объектах данного слоя.

Каждый объект слоя цифровой карты имеет пользовательский идентификатор — целое число, уникально определяющее объект. Таблицы, подсоединяемые к слою, должны содержать первым полем пользовательский идентификатор объекта. Таким образом устанавливается однозначное соответствие между записью в таблице, содержащей характеристики объекта, и самим объектом на карте.

Для описания пространственно распределенных данных используются две основные модели: ***векторная*** и ***растровая*** [12].

Векторные системы хранят описания пространственных объектов в виде последовательности координат точек. Эти точки представляют либо отдельные точечные объекты; либо линейные объекты, аппроксимированные ломаными линиями; либо площадные объекты, ограниченные полигонами (многоугольниками). Каждый объект имеет уникальный идентификатор. Неграфическая информация об объектах хранится в традиционных базах данных, причем ключом доступа является идентификатор объекта.

Растровые (или сеточные, ячеечные) системы ориентированы не на представление отдельных объектов, а на описание какого-либо аспекта (параметра) области пространства. Описываемая область разбивается на ячейки фиксированного размера (как правило прямоугольные). Пространство описывается как матрица значений, каждое из которых представляет одну ячейку. Значение может быть идентификатором объекта, кодом качественного параметра или количественной характеристикой. Рассмотрим для примера некий растровый образ. Пусть некоторая ячейка имеет значение 10. Если образ представляет собой карту территориальных округов, то это означает, что ячейка принадлежит округу № 10 (идентификатор объекта). Если образ — это почвенная карта, данная ячейка находится на участке, имеющем почву типа 10 (код качественной характеристики). Если образ — это рельеф местности, ячейка представляет собой площадку, расположенную на высоте 10 метров (количественная характеристика).

Каждый класс систем имеет определенные преимущества и недостатки.

Векторные системы более удобны для реализации функций баз данных. Они, как правило, обеспечивают более эффективное хранение данных, так как хранится информация только о границах, а не о «внутренности» объектов. В системах этого класса легче реализуются пространственные запросы (например определить площадь полигона, на который указывает мышь; длину ломаной линии, указанной с помощью мыши и т. п.). Легко можно строить простые тематические карты, представляющие результаты запросов типа «Канализационные линии диаметром более одного метра, проложенные до 1950 года» или «Участки, которые будут урезаны при расширении данной дороги на 10 метров» и т. п. Кроме того, векторные системы обеспечивают более высокую точность определения метрических характеристик при меньших накладных расходах и позволяют легко выводить карты хорошего качества на перьевые плоттеры. Недостаток чисто векторных систем — большие трудозатраты при вводе картографического материала и трудность решения аналитических задач, использующих «сеточные» методы. Типичные представители векторных систем — ARC/INFO (ESRI), MapInfo (Mapinfo).

Растровые системы более ориентированы на анализ пространственно распределенных данных. Поскольку данные хранятся на однородной сетке, достаточно эффективно реализуются различные модельные расчеты (перенос загрязнений, анализ выпадения осадков, эрозия почв, стабильность растительности). Проще строить числовые модели местности (трехмерное представление). Поскольку спутниковые снимки хранятся в виде растра, растровые системы естественным образом используются при интерпретации данных дистанционного зондирования. Непосредственный ввод изображений путем сканирования менее трудоемкий, чем при оцифровке дигитайзером (хотя стоимость сканера выше). С другой стороны, это достигается за счет хранения избыточного количества данных. Кроме того, растровые системы обеспечивают, как правило, меньшую точность, чем векторные. Для повышения эффективности таких систем используют технологии сжатия данных. Для реализации пространственных запросов нужно использовать специальные структуры хранения растра. Типичные представители — ERDAS, ER/Mapper.

По мере развития ГИС стало понятно, что желательно использовать оба подхода. В результате современные ГИС обычно поддерживают обе модели данных.

5.4. ОПИСАНИЕ ГИС-СИСТЕМ

5.4.1. Системы института ESRI

В данном разделе приводится информация о ГИС, разработанных американской фирмой ESRI (Environment Systems Research Institute, Inc. — Институт исследования систем окружающей среды), на примере которых мы и покажем основной комплекс возможностей ГИС-технологии.

Эта фирма более 25-ти лет участвует в различных разработках, связанных с решениями научных проблем, задач планирования и уп-равления. Одной из первых фирма начала выпуск коммерческих программных продуктов для создания геоинформационных систем (ГИС, GIS). Ее система ARC/INFO была первой ГИС, ориентированной на выполнение функций базы данных.

Фирма участвовала как в разработках, заказанных Министерством обороны США (привязка шахт межконтинентальных баллистических ракет, перенацеливание их полетов, поддержание инженерных сооружений баз ВВС, навигация боевых самолетов), так и в многочисленных гражданских разработках (поддержание функционирования местных органов власти, фирм, занимающихся эксплуатацией инженерных сооружений типа дорог, линий электропередач, коммуникационных сетей, сетей водоснабжения и канализации, транспортных систем; оценка воздействия на окружающую среду, сохранение природных ресурсов, составление систем карт по различной тематике и различным регионам и т. д.)

Фирма ориентируется на универсальные аппратные средства. Ее продукты работают на персональных компьютерах класса IBM PC, рабочих станциях Sun, Digital, HP, Prime, Silicon Graphics, NEC, Intergraph, IBM, Data General и больших ЭВМ (mainframe) IBM.

Фирма придерживается довольно жесткой стратегии в отношении инструментальных разработок других фирм. Она обязательно приобретает все заслуживающие внимания разработки в собственность (иногда фирма-разработчик становится филиалом ESRI). Это позволяет обеспечить одновременный перенос инструментальных средств на новые платформы. Политика фирмы Intergraph в этой области отличается — дополнительные инструментальные средства остаются собственностью их разработчиков, в результате чего не всегда возможен их перенос. Так, по сведениям из разных источников, из 40 программных систем, разработанных для специализированной аппаратуры Inter-graph, на универсальные платформы типа Sun перенесено около 20).

Большое внимание уделяется обеспечению интеграции в продукты фирмы данных из других систем. Практически обеспечивается непосредственная поддержка или конвертирование форматов большинства популярных систем. Кроме того, разработаны специальные обменные форматы.

Эти форматы, в отличие от внутренних форматов фирмы, открыты, и разработчики других систем могут (и на самом деле делают это) обеспечивать экспорт данных для систем ESRI или импорт из них. Есть примеры интеграции в среду ARC/INFO (основного продукта ESRI) специализированных приложений (система управления электроснабжением сделана на базе ARC/INFO и включает специализированное приложение для анализа электросетей, выполненное фирмой, специализирующейся в этой области).

Основные ГИС ESRI включают ARC/INFO, PC ARC/INFO, ArcView и ArcCAD. ARC в названии всех продуктов — это английское слово «дуга», подчеркивающее, что в основе продуктов лежит векторная модель данных; INFO — это название специальной СУБД, используемой в продуктах для рабочих станций и в старых версиях для PC. Кроме того, имеется ряд дополнительных продуктов — GRID, TIN, COGO, NETWORK, DBI-DB2, DBI-AS/400, PC TIN, PC SEM, Application Starter Kits.

Базовым продуктом является ARC/INFO. Это наиболее полная система, реализующая все функции ГИС и содержащая средства для разработки своих приложений. Работает на рабочих станциях под управлением ОС Unix с оболочкой X Windows.

Поскольку ARC/INFO и остальные пакеты ориентированы на реализацию самых общих функций баз данных, расширенных для работы с пространственно распределенной информацией, и реализуют универсальные подходы при работе с картографической информацией, область их применения крайне широка. Информацию о применениях можно найти, например, в рубриках User Applications (приложения, разработанные пользователями), ESRI Applications (приложения, разработанные ESRI), International News (Международные новости), Conservation News (Новости в области охраны окружающей среды), ежеквартальной газеты ARC News. Можно привести следующие типичные применения:

приложения, обеспечивающие функционирование местных органов власти, — учет и налогообложение недвижимости, земельных участков; оценка и управление собственностью; выдача разрешений на строительство; планирование развития дорожного строительства;

общественная безопасность — анализ криминогенной обстановки;

распределение ресурсов и диспетчеризация полицейских служб, служб пожарной охраны, аварийных служб;

управление инженерными сетями — водоснабжение, канализация, транспорт, электросети, кабельное телевидение;

справочные системы для туризма;

демографический анализ;

картирование и описание заповедных зон;

оценка воздействия разливов нефти на окружающую среду.

ARC/INFO

Первая версия пакета ARC/INFO была выпущена в 1982 г. (до этого фирма ESRI работала в области GIS более 10-ти лет). Эта версия работала на мини-компьютере Prime. В начальный период выпускалось по две версии пакета в год. Сейчас новые версии появляются примерно раз в год. Пакет работает на рабочих станциях Apollo, Data General AViiON, DEC (DECstation и VAXstation), HP (300, 400, 700, 800), IBM (RISC System/6000), NEC, Intergraph, Silicon Graphics, Sun (Sun-3, Sun-4 и SPARCstation), на мини-компьютерах Prime, Data General, DEC VAX и на больших ЭВМ IBM (под VM/CMS и MVS/XA). На рабочих станциях ARC/INFO работает под управлением различных версий UNIX с оболочкой X Windows. Есть реализации для ОС VAX/VMS фирмы Digital (ранее Digital Equipment, DEC).

Ввод данных. Пакет предлагает использование различных технологий ввода данных. Ранние версии позволяли только оцифровывать карты вручную с помощью дигитайзеров различных форматов. Затем были добавлены возможности ввода данных из других систем ГИС или САПР (как векторных, так и растровых). В последних версиях в пакет включен модуль Image Integrator, позволяющий непосредственно использовать отсканированное изображение. Его можно включать в выводимые карты или оцифровывать с помощью мыши в увеличенном виде. В общем виде технология ввода следующая:

- листы карты оцифровываются дигитайзером (под управлением ARC/INFO), или сканируются, или обрабатываются иным способом в других системах;
- данные из других систем преобразуются в формат ARC/INFO модулем DATA CONVERSION;
- в результате предыдущих шагов получаются «покрытия» (тематические или иные слои карты). Покрытие — это единица хранения графической информации в ARC/INFO. Оно состоит из точек, линий, соединяющих их, и меток полигонов, образованных замкнутыми линиями;
- выполняется операция «очистки» покрытия, которая определяет и по возможности устраняет ошибки оцифровки (линии, концы которых «подвисли», т. е. не лежат на других линиях или на рамке покрытия, полигоны без меток). Для автоматического устранения ошибок используются различные допуски;
- если остались неустраненные ошибки, покрытие корректируется вручную модулем ARCEDIT. Этот модуль позволяет редактировать покрытие на фоне других покрытий (например, можно исправлять карту землепользования, «подложив» под нее карты рельефа, гидрологической сети, дорожной сети). Допускается до шести фоновых покрытий. Можно объявить одно из фоновых покрытий опорным. Тогда координаты передвигаемых или добавляемых точек будут автоматически «привязываться» к координатам ближайших точек опорного покрытия. В качестве «подложки» может также использоваться растровый образ. Именно это позволяет выполнять ручную оцифровку отсканированных карт. Используется многооконный режим. Одно и то же покрытие можно рассматривать в разных аспектах, используя до 12-ти окон. ARCEDIT позволяет также вводить в покрытие аннотации (надписи на карте), которые могут организовываться достаточно сложным образом (например, выводиться вдоль кривой линии, как это принято делать с названиями рек, и т. п.);
- этапы очистки и редактирования покрытий повторяются итеративно, пока не исчезнут неустраненные ошибки. Для сверки покрытия с исходным материалом выдается твердая копия покрытия в единицах оцифровки (масштаб 1:1), желательно на прозрачную пленку. Полученная копия накладывается на оригинал и сверяется визуально;
- выполняется операция «построения топологии». При этом для покрытий, содержащих полигоны, строятся отношения соседства полигонов, для всех помеченных полигонов вычисляются площади и периметры (в единицах оцифровки), а для линий — длины дуг;
- если нужно, выполняется стыковка листов. Для этого все необходимые покрытия преобразуются в общую систему координат (используя мощный аппарат преобразований карт и проекций, имеющийся в ARC/INFO), затем выполняется операция попарной стыковки. При этом ARC/INFO пытается сама найти продолжения линий с одного листа на другой, а затем предлагает пользователю вручную исправить возможные ошибки. Для полученного таким образом объединенного покрытия строится топология; при оцифровке и редактировании каждый географический объект получает уникальный идентификатор. После окончания редактирования покрытия можно заполнять табличную часть базы данных, связывая данные с объектами покрытий с помощью этих идентификаторов. ARC/INFO имеет встроенную СУБД реляционного типа. Возможно преобразование данных из других популярных СУБД. Кроме того, в состав пакета входит модуль DATABASE INTEGRATOR, который обеспечивает одновременное взаимодействие с несколькими различными СУБД (ORACLE, INGRES, SYBASE, In-formix, DB2, Rdb, SQL/400 и т. п.) без явного преобразования данных.

Различные покрытия, относящиеся к одному проекту, можно объединять в библиотеки. Модуль LIBRARIAN (библиотекарь) ARC/INFO обеспечивает несколько уровней управления и защиты как для библиотек в целом, так и для отдельных слоев. Кроме того, он позволяет работать с отдельными покрытиями, изображающими разные листы одной и той же карты, не строя общее покрытие. Библиотекарь сам «понимает», что объекты в разных покрытиях, имеющие одинаковый идентификатор, на самом деле — части одного объекта.

Вывод информации. Для получения результирующих карт используется модуль ARCPLOT. Он содержит мощный набор команд для организации «страницы», которая может содержать несколько карт, полученных из различных покрытий и растровых образов или даже из обменных файлов AutoCAD-DXF (врезки, легенды, рамки, внерамочные надписи и символы, текстовые фрагменты). Можно отбирать отображаемые объекты, пользуясь многоступенчатыми операторами пространственного и атрибутного выбора (т. е. можно отобрать группу объектов, затем уточнить выбор по дополнительным критериям и т. д.) Полученные карты могут отображаться на дисплее или выводиться на различные графические устройства или в файлы. Поддерживается большое количество перьевых и электростатических плоттеров, принтеров (как матричных, так и лазерных), формат PostScript и ряд других.

Манипулирование данными. ARC/INFO имеет развитый аппарат манипулирования данными. Для работы с табличными (атрибутными) данными используется стандартная идеология реляционных СУБД. Можно выпол-

нять всевозможные корректировки данных, операции соединения, выбор. Более сложные операторы позволяют получать суммарные статистики по колонкам таблиц (суммы, дисперсии, средние).

Реляционные операторы типа Select (Select — выбор, Reselect — довыбор, ASelect — выбор с добавлением) расширены для работы с пространственными данными. Можно указывать конкретный объект, выбирать группы объектов, попадающих внутрь заданного прямоугольника, окружности или полигона.

Предусмотрены сложные манипуляции с покрытиями. Базовой операцией является наложение покрытий. Есть несколько модификаций этой операции. Можно, например, «обрезать» покрытие, используя другое как «кулинарную форму» или «матрицу пресса». Можно «стереть» часть покрытия, пересекающуюся с другим покрытием, можно построить «врезку» объектов одного покрытия в другое, построить объединенное покрытие, «пересечение» покрытий. Дополнительные операции — исключение «осколочных» полигонов (т. е. многочисленных полигонов малой площади, возникающих, например, при наложении покрытий; при независимой оцифровке общей границы координаты вершин, естественно, будут слегка отличаться и при наложении таких покрытий образуются фиктивные полигоны), исключение лишних границ (например, возникших при стыковке листов).

Еще одна важная операция — построение буферных зон. Она позволяет построить полигоны, границы которых удалены на заданные расстояния (фиксированные или зависящие от внешних условий) от объектов исходного покрытия. Данная операция широко используется для пространственного анализа: например, чтобы построить карту площадок, удаленных не более 100 метров от шоссейной дороги и находящихся не ближе 300 метров от русла реки, надо построить 100-метровую буферную зону для дороги, 300-метровую — для реки; найти пересечение покрытия, содержащего все площадки, с буферной зоны дороги, а затем стереть область полученного покрытия, используя в качестве маски буферную зону реки.

Разработка приложений. ARC/INFO имеет довольно мощный макроязык AML, предназначенный для разработки приложений. AML имеет средства как для работы с объектами покрытий и их атрибутами, так и для работы с объектами графической оболочки X Windows — окнами, меню, бланками диалогов. Можно вести диалог с пользователем, ме-нять значения переменных AML, анализировать их, выполнять действия с покрытиями, вызывая различные модули ARC/INFO и выполняя их команды. Кроме того, начиная с версии 6.0, в ARC/INFO включена возможность вызова программ, написанных на языке C. Для этого разработан пакет ArcSdl (Arc Software Development Library) содержащий библиотеку процедур, дающих программам на C доступ к объектам ARC/INFO. В «программе» на AML можно проанализировать результат завершения модуля ARC/INFO и при ошибке выдать понятную диагностику или предпринять соответствующие действия. Это позволяет полностью избавить пользователя от работы на уровне командной строки ARC/INFO. Хорошим примером приложения, разработанного на AML, являются демонстрационные примеры ARC/INFO.

PC ARC/INFO

Пакет PC ARC/INFO является упрощенной версией ARC/INFO для рабочих станций. Он предназначен для работы на PC AT. Может использоваться для разгрузки рабочих станций с полной версией ARC/INFO, для организации отдельных рабочих мест в рамках общей структуры ГИС и для самостоятельной работы.

Упрощения PC ARC/INFO по сравнению с ARC/INFO носят как количественный, так и качественный характер. Количественные ограничения заключаются в том, что из-за малых ресурсов ПК пакет поддерживает меньшее число объектов в покрытиях, меньшее число фоновых покрытий, работает значительно медленнее. Качественные ограничения заключаются в том, что PC ARC/INFO выполняет ограниченный набор функций полного ARC/INFO либо некоторые функции выполняются в упрощенном варианте.

Из всего множества инструментов PC ARC/INFO остановимся на ***PC NETWORK***, что позволит нам увидеть типовые методы решения сетевых задач, присущие геоинформационным системам. Данный модуль используется для решения оптимальных задач на сетях и для реализации адресного геокодирования. Решаются два типа оптимальных задач — задача назначения (распределения) и задача маршрутизации.

Задача назначения заключается в распределении оптимальным образом клиентов по центрам обслуживания (например, школьников по школам так, чтобы они тратили минимум времени на дорогу; домов по почтовым отделениям, чтобы минимизировать время или потребление топлива на доставку почты и т. п.) Задача маршрутизации состоит в выборе оптимального в некотором смысле маршрута (например, минимум времени для школьного автобуса, чтобы собрать всех школьников и довезти их до школы; минимум времени, требующегося патрульной полицейской машине для проверки неблагополучных точек, и т. п.) Сеть описывается как набор центров (узлов, имеющих определенную емкость, из которой ресурсы добавляются в сеть, как, например, почтовые отделения, обслуживающие определенное количество абонентов, или в которую они изымаются из сети, как в случае школ); связей (линий перемещения ресурсов); остановок (узлов, где ресурсы пополняются, например школьники садятся в автобус, или убывают, например дом, где выгружается часть почты); поворотов (описания возможных поворотов в узлах, где пересекаются связи) и барьеров (узлов, преграждающих путь потоку ресурсов, например временно перекрытая улица). Элементы сети имеют специальные атрибуты. Связи и повороты имеют импеданс — атрибут, показывающий, какое сопротивление потоку ресурса оказывает данный элемент (например, длина улицы играет роль импеданса — чем длиннее улица, тем больше времени нужно, чтобы ее проехать; поворот из-под знака «Стоп» имеет больший импеданс, чем поворот при отсутствии знака, и т. п.). Импеданс может быть различным по разным направлениям. Можно указать отрицательный импеданс, что обозначает полную «непроходимость» связи или поворота. Связи и остановки могут иметь атрибут «потребление ресурса» (количество школьников, живущих на улице, количество газет, которые надо выгрузить у данного дома). Центры имеют атрибут «емкость». Для решения задачи назначения конфигурация сети считывается из покрытия ARC/INFO (импедансы связей и поворотов должны быть заданы в таблице атрибутов), к ней добавляются центры и барьеры (из файлов базы данных или интерактивно), после чего запускается процедура решения. В результате дуги покрытия связываются с центрами (может задаваться до 50-ти центров). Можно получить графическое представление назначений. Для этого каждому центру присваивается определенный тип и цвет линии, и все дуги, назначенные данному центру, рисуются с использованием таких линий.

При решении задачи маршрутизации пользователь задает узлы сети, через которые должен проходить маршрут, и узлы, содержащие остановки, а система строит оптимальный маршрут. Пользователь может сохранять несколько маршрутов, интерактивно изменять их и т. п.

Часто координаты узлов и дуг в сетевых покрытиях естественно задаются в виде адресов. Переход от адресов к координатам называется геокодированием. Типичная задача геокодирования — адресный поиск — заключается в том, что, получив файл со списком адресов, система должна выдать точечное или линейное покрытие с объектами, находящимися по указанным адресам. Эта задача достаточно сложна, если учитывать, что понятие адреса не является однозначно определенным. Адрес одного и того же объекта может вводиться разнообразному, но система должна быть в состоянии все равно находить объект. В модуле PC NETWORK для проведения геокодирования сначала создается адресное покрытие. При этом вводится набор адресов (включая альтернативные). Адреса разбиваются на части (улица, номер дома, правая/левая сторона, четный/нечетный адрес, почтовая зона и т.п.) Для приложения, требующего выполнения адресного поиска, строится специальная таблица «штрафов». Каждый адрес, заданный в файле адресов, анализируется, и в нем выделяется название улицы (или улиц, если адрес задается пересечением улиц). После этого в адресном покрытии находятся «кандидаты» на совпадение. Каждый кандидат оценивается по штрафной таблице — за каждый тип несовпадения назначается определенный штраф. Сумма штрафов вычитается из 100, и полученный результат сравнивается с минимально допустимым, также указываемым в таблице штрафов. Если оценка кандидата больше минимальной, он проходит. Из всех кандидатов отбирается объект с максимальной оценкой. Если для какого-то адреса объект не найден, запускается процедура обработки отклоненных адресов, которая позволяет исправить адрес в интерактивном режиме или включить его как альтернативный в адресное покрытие.

Следует иметь в виду, что при анализе адресов используется формат адреса, принятый в США (типы улиц, почтовые зоны).

ArcView

Пакет ArcView разработан ESRI для того, чтобы дать доступ к данным, хранящимся в форматах ARC/INFO, значительно более широкому кругу пользователей. Если ARC/INFO и PC ARC/INFO являются средствами профессионалов в области ГИС, то ArcView разрабатывался как инструмент для слабо подготовленных пользователей. Он стоит на порядок меньше, чем ARC/INFO (PC ARC/INFO), и обеспечивает интерактивный доступ ко всем типам данных, поддерживаемых ARC/INFO: векторным покрытиям, табличным данным, моделям местности, сеточным данным, растровым изображениям. Фирма ESRI рассматривает ArcView как стратегический продукт. Используется современный графический интерфейс, практически одинаковый на всех платформах. ArcView является сетевым продуктом. Он может обеспечить доступ с любого рабочего места к данным, хранящимся на сервере локальной сети. ArcView «понимает» разницу в форматах данных PC и рабочих станций и позволяет работать как с теми, так и с другими, не выполняя явных преобразований.

Основа пакета — многооконная среда. Данные организуются в «виды» (View). Каждый вид состоит из набора тем. Тема может содержать объекты одного или нескольких покрытий и набор легенд. Объекты покрытия могут входить в несколько тем. Пользователь ArcView может выбрать набор тем, которые он хочет отобразить. Графические данные (карты, растровые изображения) выводятся в одном окне, а табличные — в другом, в стиле, напоминающем электронные таблицы. Одна и та же тема может отображаться в нескольких окнах (с различным увеличением, например). В графическом окне можно выполнять панорамирование (прокрутку) и изменение масштаба. Пользователь может легко менять легенды, настраивать палитры и штриховки. Можно выбирать объекты графически, указывая их курсором или рамкой, которая может иметь форму прямоугольника, окружности или произвольного полигона, или по атрибутам в табличном окне. Можно выполнять манипуляции над табличными данными и отображать результаты в графическом окне. Можно получать суммарные статистики для табличных данных. Предусмотрены средства вывода вида на графические устройства или в файлы, что позволяет легко включать картографические изображения в документы, подготавливаемые в издательских системах или мощных текстовых процессорах (типа Word For Windows).

Следует иметь в виду, что, говоря о неподготовленных пользователях, персонал ESRI имеет в виду уровень «неподготовленности» американских менеджеров или технического персонала, т. е. людей, не изучавших ГИС, но использующих компьютеры (в том числе с графическими средами типа Windows) в своей повседневной деятельности. Для нашего персонала необходимо предусматривать довольно длительный процесс обучения.

ArcCAD

Пакет ArcCAD разработан ESRI для пользователей, которым нужно одновременно использовать ГИС и САПР. ArcCAD является расширением известного пакета AutoCAD (v11 и старше). Модель данных AutoCAD расширена для создания топологических и пространственных отношений. При этом ArcCAD использует объекты AutoCAD для рисования и редактирования графики. В результате ArcCAD может непосредственно исполь-

зовать данные ARC/INFO и PC ARC/INFO, файлы рисунков AutoCAD, данные, публикуемые в рамках программы ArcDATA. Он имеет также доступ к данным в формате dBASE, DXF, IGES. С другой стороны, данные, созданные в ArcCAD, могут непосредственно использоваться в ARC/INFO, PC ARC/INFO и ArcView. При этом не теряются возможности самого AutoCAD: в меню AutoCAD просто добавлен пункт «ArcCAD». При переходе в режим ArcCAD в меню появляется пункт «AutoCAD» для возврата в AutoCAD.

Язык AutoLISP пакета AutoCAD расширен средствами обработки географической информации. Функции ArcCAD доступны из любой библиотеки AutoCAD. ArcCAD может выполняться вместе с любым приложением в AutoCAD. Это очень удобно для таких пользователей, как архитекторы, занимающиеся городской планировкой, дорожные службы и т. п. Ограничения ArcCAD: не включены функции модулей NETWORK и DATA CONVERSION.

GRID

Этот дополнительный модуль включает в ARC/INFO растровую модель данных. Растровая модель данных позволяет эффективно решать задачи, связанные с представлением непрерывных характеристик пространства, — находить уклоны и экспозиции по высотам и т. п. Эффективность ее связана с тем, что координаты не являются атрибутами объектов, а определяются положением ячейки в образе пространства, т. е. могут легко вычисляться. Растровая модель также эффективнее для представления динамических процессов, которые распространяются по принципу не «от объекта к объекту», а «от точки к точке». Примеры: модели распространения лесных пожаров, миграции стад оленей и т. п. GRID позволяет выполнять многочисленные логические и алгебраические операции с образами. Имеется специальный язык для описания требуемых операций. GRID естественным образом используется для обработки спутниковых снимков (систем SPOT и LANDSAT, например). Имеются операторы фильтрации, позволяющие улучшать качество образов.

TIN

Модуль TIN используется для построения трехмерных моделей местности методом триангуляции. Он состоит из набора макропрограмм на AML для основной части системы ARC/INFO, и для ARCPLOT.

При анализе данных, связанных с земной поверхностью, часто бывает нужно решать задачи типа построения водоразделов и тальвегов, определения видимости различных участков поверхности из разных точек, определения уклонов и экспозиций склонов, вычисления объемов под сложными поверхностями, построения различных сечений и профилей. Подобные задачи относятся к числовому моделированию местности. При наличии данных на регулярной сетке используют растровые методы анализа. При наличии нерегулярных данных (например, данных топосъемки) более эффективно использование триангуляции, при этом поверхность аппроксимируется набором треугольников, вершины которых располагаются в точках с известными значениями данных.

COGO

Этот модуль используется для обработки данных геодезической съемки. В нем решаются различные специфические задачи, главным образом задачи аналитической геометрии. К ним относятся нахождение пересечений отрезков прямых, дуг кривых, восстановление перпендикуляров и нормалей, построение касательных к нескольким кривым, вписывание и описывание окружностей и т. п. Эти проблемы возникают при расчетах координат различных объектов по данным полевой съемки, а также при инженерном проектировании. Модуль используется также при построении навигационных систем с привязкой к глобальным системам позиционирования (спутниковым).

5.4.2. Системы ЦГИ ИГ РАН

Несомненным лидером отечественной ГИС-индустрии является Центр геоинформационных исследований института географии Российской академии наук (ЦГИ ИГ РАН), представляющий на рынок свои, пользующиеся заслуженной популярностью, продукты — Geo-Draw, GeoGraph и GeoConstructor.

GeoDraw

GeoDraw для Windows — векторный топологический редактор для создания цифровых карт — является одним из программных средств геоинформационных систем, разрабатываемых Центром геоинформационных исследований ИГ РАН. Идеология, лежащая в основе Geo-Draw, включает следующие положения:

- GeoDraw является инструментом для создания высококачественных цифровых карт, учитывающих требования ведущих мировых ГИС;
- создаваемая и редактируемая в GeoDraw структура пространственных данных цифровой карты (включая отношения связности, смежности, соседства, вложенности объектов и др.) гарантирует при соблюдении технологии корректную фиксацию и изменение отношений между пространственными объектами, их связи с базой атрибутивных данных, позволяет преобразовывать созданные в GeoDraw цифровые карты в другие ГИС (как топологические, например ARC/INFO, так и нетопологические — MapInfo и др.) без дополнительных накладных расходов на редактирование;
- мощные средства трансформации создаваемых цифровых карт (преобразования более 40 типов картографических проекций, широкий набор преобразований плоскости и др.) позволяет решать задачи их интеграции (осуществлять «склейку» листов, «посадку» одних карт на другие с образованием многослойной структуры и др.);
- GeoDraw является легким в освоении программным продуктом, отражающим многолетний опыт работы коллектива ЦГИ ИГ РАН с ведущими мировыми ГИС, сотнями пользователей GeoDraw, тысячами карт и планов разнообразной тематики и масштабов — от 1:500 до 1:50000000.

GeoDraw для Windows позволяет:

- осуществлять перевод карт и планов в цифровую форму посредством векторизации по растровой подложке, при помощи дигитайзера, ввода значений координат объектов по имеющимся данным или по результатам измерений на местности;
- вводить и редактировать пространственные объекты типа точки, дуги, полигона при помощи дигитайзера, «мыши», клавиатуры путем ввода координат или импорта из открытых текстовых форматов;
- использовать широкий спектр функций отображения пространственных объектов на экране: изменение масштаба отображения, сдвиг изображения в процессе цифрования текущей дуги, отображение только определенных типов узлов и слоев и т. д.;
- подгружать столько слоев, сколько позволит конфигурация компьютера; оперативно менять их статус и атрибуты отображения;
- осуществлять топологическое согласование объектов и создавать корректную многослойную структуру при помощи широкого набора операций над топологической структурой — создание линейно-узловой структуры, цифрование общих границ полигонов один раз и сборка полигонов из дуг, захват произвольных частей объектов из одного слоя в другой и др.;
- выделять группы объектов в карте или в связанной с ней таблице, удалять, копировать, генерализовать, идентифицировать только выделенные группы;
- осуществлять преобразования цифровых карт из различных картографических проекций в географические координаты и обратно;
- осуществлять аффинные, локально-аффинные, проективные, квадратичные и полиномиальные (5-й степени) преобразования, поворот оси;
- использовать набор функций по идентификации пространственных объектов цифровых карт для связи с базами атрибутивных данных, включая присвоение объектам пользовательских идентификаторов, нахождение объектов, не имеющих таких идентификаторов, или объектов с определенными идентификаторами, генерирование отчета об имеющихся пользовательских идентификаторах и др.;
- подгружать в среду редактора таблицы атрибутивных данных, осуществлять проверку идентификации объектов по табличным данным, при необходимости вводить и редактировать записи таблицы для конкретных объектов карты, показывать текущий объект таблицы на карте или объект, выделенный на карте — в таблице, осуществлять проверку соответствия карты с таблицей;
- экспортировать и импортировать данные в широко используемые форматы (GEN PC ARC/INFO, MIF/MID MapInfo, VEC IDRISI, DXF AutoCAD).

GeoGraph

GeoGraph для Windows — ГИС уровня конечного пользователя, позволяющая осуществлять некоторый универсальный общий набор функций ГИС, удовлетворяющий большинство пользователей в различных предметных областях, — создание композиций слоев цифровых карт, связанных с базами данных; тематическое картографирование; запросы от карты к таблице и от таблицы к карте; оформление карты; вывод карт в твердой копии на различные устройства и др.

GeoGraph для Windows дает возможность:

- создавать электронные карты или атласы как композиции картографических слоев, выбираемых пользователем (включая векторные и растровые), и связанных с ними таблиц атрибутивных (тематических) данных;
- управлять таблицами атрибутивных данных (создавать таблицы, связывать их с цифровыми картами, редактировать, менять структуру таблиц и др.);
- управлять масштабированием изображения;
- осуществлять поиск или выбор объектов на карте с отображением результатов в таблице атрибутивных данных;
- выбирать объекты вручную или на основе задания «запросов по образцу» к атрибутивным таблицам с отображением результатов на карте;
- проводить электронное тематическое картографирование;
- осуществлять измерения по карте;
- находить области, удовлетворяющие задаваемым условиям для произвольного набора цифровых карт электронного атласа (динамический оверлей слоев);
- выводить твердые копии карт на любые печатающие устройства, доступные для Windows.

Идеология GeoGraph для Windows достаточно проста. Создаются карты или атласы, в каждый из которых Вы можете поместить некоторый набор слоев цифровых карт. Такие слои могут быть созданы в других программных средах (например, посредством GeoDraw), или непосредственно в среде GeoGraph (слои типа Grid или Cosmetic). Также в качестве слоев могут быть загружены растровые изображения. С каждым векторным слоем может быть связан набор таблиц с атрибутивными (тематическими) данными. Для формата GeoDraw все

файлы, относящиеся к конкретному слою, должны находиться в отдельной директории с именем слоя и все файлы внутри этой директории должны иметь то же имя, что и имя директории (но различные расширения).

Вы можете создать новую карту или новый атлас под заданным Вами именем и поместить в них набор слоев цифровых карт, а к слоям привязать набор таблиц тематических данных. Результаты такой компоновки для каждой карты (атласа) могут быть сохранены в файле определенного образца с расширением .mp, который в дальнейшем может быть снова загружен (открыт) в системе GeoGraph.

Вы можете открыть любую карту (атлас) и затем работать внутри нее, сохранить в любой момент текущие результаты работы с картой, загрузить другую карту.

Если Вы хотите создать новую карту, то необходимо выбрать соответствующий пункт главного меню, и будет создана новая («пустая») карта, слои и таблицы к которой Вы можете привязать самостоятельно.

Если Вы открыли карту или атлас, все слои, связанные с картой, будут представлены в окне Управления слоями. Слои могут быть включены или выключены для отображения, слоям может быть присвоен диапазон масштабов, при которых они должны быть видимыми, и т. п. В карту (или атлас) Вы можете добавить новый слой путем выбора соответствующих файлов с диска.

Выбор карт и другие основные операции выполняются в системе через главное меню, расположенное в верхней части окна системы GeoGraph. Также после того, как принадлежащие карте (атласу) слои будут загружены, Вы можете использовать дополнительные возможности управления, предоставляемые в двух окнах, — окне Управления слоями (Легенда) и окне Инструментов (Инструментарий). Эти окна также могут быть включены или выключены для отображения через главное меню (пункт Окна). В окне Управления слоями Вы можете включить/выключить любой слой для отображения, выбрать слой для дальнейших запросов, удалить слой из списка слоев, отнесенных к карте, передвинуть слой в списке вверх или вниз, изменить тематическую классификацию для слоя. Окно Инструментов дает Вам возможность управлять отображением карты, изменяя масштаб, сдвигать и центрировать изображение, осуществлять выборки объектов по карте на экране, измерять расстояния и др.

Вы можете связывать с каждым слоем имеющиеся таблицы атрибутивных данных; импортировать/экспортировать таблицы; записывать в таблицы результаты измерений по карте; создавать новые таблицы, связываемые с активным слоем; загружать любую таблицу и производить с ней ряд операций (изменение структуры, сортировку, редактирование, выборки вручную, «запросы по образцу» с отображением результатов выборок на карте), пользуясь пунктами главного меню или «горячими клавишами».

Ключевое положение в GeoGraph занимает Администратор данных, в котором с каждым векторным слоем можно связывать множество следующих объектов:

- формы, создаваемые пользователем для вывода справочной информации об объектах;
- запросы к таблицам, связанные с каждым слоем;
- макросы, т.е. внешние исполняемые программы или внутренние функции GeoGraph, которые могут быть заданы пользователем как для отдельных слоев, так и для конкретных объектов;
- темы — варианты тематического картографирования;
- селекции — нахождение пространственных пересечений или попаданий выбранных объектов одних слоев в объекты других слоев;
- графики — представление результатов тематических и других классификаций в виде различных графиков;
- SQL-запросы, используемые наравне с QBE-запросами;
- надписи, отображаемые либо нет со слоем.

Вы можете непосредственно в среде GeoGraph создавать новые слои пространственных объектов, так называемые косметические слои. Косметический слой — нетопологический векторный слой (т. е. в нем не фиксируется структура отношений пространственных объектов), создаваемый в системе GeoGraph. В один слой можно помещать объекты разных типов — точки, дуги, полигоны. Большинство операций с косметическими слоями аналогично операциям с другими слоями, за рядом указанных далее исключений.

В процессе или в конце сеанса работы Вы можете сохранить уже достигнутые результаты (список слоев, положение карты и окон на экране, классификацию объектов и назначенные графические переменные и др.).

Карта может быть подготовлена к печати в виде макета печати. При печати карты будет выводиться текущий отображаемый на экране фрагмент карты, название карты и состав легенды. В макет печати могут быть включены также тексты, графики, растровые изображения и др.

Отображение пространственных объектов векторной цифровой карты (точечных, линейных или площадных) в среде геоинформационных систем осуществляется следующим образом. Всем объектам слоя пользователь присваивает выбранную графическую переменную (для полигонов — цвет заливки, тип штриховки; для линейных объектов — тип, цвет, толщину линии; для точечных объектов — символ определенной формы, размера и т. п.). При тематической классификации объекты слоя классифицируются по значениям одного из полей любой таблицы атрибутивных данных, связанной со слоем, и графическая переменная присваивается в этом случае для каждого класса. При отображении пространственных объектов в среде ГИС на устройствах вывода (на дисплее, принтере и т. п.) объекты будут отображаться присвоенной им графической переменной (характеризующей их принадлежность к определенному слою или классу).

GeoGraph предоставляет большой выбор отображения и представления атрибутивных данных. Компоненты для представления данных будем в дальнейшем называть объектами. В системе GeoGraph существуют следующие объекты: таблица (table), форма (form), запрос (query).

Система поддерживает работу со следующими таблицами в форматах: Paradox (*.db), dBase (*.dbf), MS Access 2.0, а также с СУБД Oracle, MS SQL Server, Sybase в режиме «клиент-сервер».

GeoGraph предоставляет возможность экспортировать и импортировать данные в формат текстового файла типа ASCII (расширение файлов *.csv).

Работа с атрибутивными данными в GeoGraph осуществляется с помощью инструментального средства Borland Database Engine (BDE), которое обеспечивает непосредственный доступ к файлам БД (Paradox, dBase) и дает возможность организации доступа к удаленным серверам СУБД (Oracle, Informix и т. д.). В основе BDE лежит технология Integrated Database API (IDAPI), которая включает IDAPI-инфраструк-туру и обработчик запросов.

Каждому слою карты можно поставить в соответствие несколько таблиц БД, содержащих атрибутивную информацию об объектах данного слоя.

GeoConstructor

GeoConstructor — инструментальное средство для создания ГИС-приложений, выполненное в стандарте VBX и позволяющее создавать ГИС-приложения в различных средах визуального программирования (Microsoft Visual Basic, Microsoft Visual C++, Borland Delphi, Borland C++, dBase for Windows).

Все приложения базируются на ГИС-библиотеке, разработанной и постоянно развиваемой в Центре геоинформационных исследований. Все программное обеспечение написано на языке Си++.

Контрольные вопросы

1. Какие типы объектов присутствуют на электронной карте?
2. Чем отличается полигональный объект от линейного?
3. В чем суть задачи геокодирования?
4. Что такое «буферная зона»?
5. Охарактеризуйте две основные модели, используемые для описания пространственно распределенных данных.

СПИСОК ЛИТЕРАТУРЫ

1. Баузр Ф.Л., Гооз Г. Информатика. — М.: Мир, 1990.
2. Дуров И. Современное состояние языков и средств разметки документов. http://www.jetinfo.ru/2000/1/1/article1.1.2000.html.
3. Питтс Н. XML за рекордное время / Пер. с англ. — М.: Мир, 2000.
4. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова. — СПб.: Символ-Плюс, 1999.
5. XML. Шаг за шагом: Практическое пособие / Пер. с англ. – М.: Изд-во ЭКОМ, 2000.
6. Белоногов Г.Г., Богатырев В.И. Автоматизированные информа-ционные системы. — М.: Советское Радио, 1973.
7. Саймон А. Обработка транзакций // СУБД. — 1997. — № 2. — С. 70–82.
8. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. — М.: Нолидж, 2000.
9. Спирли Э. Корпоративные хранилища данных. Планирование, разработка, реализация. Т. 1. — М.: Вильямс, 2001.
10. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. — М.: Финансы и статистика, 1998.
11. Калянов Г.Н. Case-технологии. Консалтинг при автоматизации бизнес-процессов. — М.: Горячая линия —Телеком, 2000.
12. Цветков В.Я. Геоинформационные системы и технологии. — М.: Финансы и статистика, 1998.