

**БИБЛИОТЕЧКА
ПРОГРАММИСТА**

Н. Е. ЕМЕЛЬЯНОВ

Введение в СУБД ИНЕС



БИБЛИОТЕЧКА ПРОГРАММИСТА

Н. Е. ЕМЕЛЬЯНОВ

ВВЕДЕНИЕ В СУБД ИНЕС



МОСКВА «НАУКА»

ГЛАВНАЯ РЕДАКЦИЯ

ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ

1988

ББК 22.18

Е60

УДК 519.6

Емельянов Н. Е. Введение в СУБД ИНЕС.— М.: Наука.
Гл. ред. физ.-мат. лит., 1988.— 256 с. (Библиотечка программиста.)
ISBN 5-02-013772-3

Описываются концепции построения информационной единой системы (ИНЕС) и основные этапы разработки баз данных средствами системы управления базами данных ИНЕС: описание данных, ввод данных в базу, средства высокого уровня доступа и обработки данных (система запросов), вывод данных. Подробно разбираются возможности применения ИНЕС на каждом этапе, приводится описание соответствующих языковых средств. Рассматриваются как практические вопросы технологии работы с базами данных, так и общие методические рекомендации по конструированию баз данных. Изложение сопровождается большим количеством примеров и иллюстраций.

Для пользователей ИНЕС, специалистов по разработке и эксплуатации различных автоматизированных систем, студентов, изучающих теорию и практику применения баз данных.

Табл. 21. Ил. 74. Библиогр. 65 назв.

Рецензент

член-корреспондент АН СССР О. И. Авен

Е 1702070000—106
053(02)-88 16-88

© Издательство «Наука».
Главная редакция
физико-математической
литературы, 1988

ISBN 5-02-013772-3

ОГЛАВЛЕНИЕ

Предисловие	5
Введение	7
В.1. База данных, СУБД, описание данных	7
В.2. Интерфейс с базой данных	10
В.3. Что такое ИНЕС?	13
В.4. Базовый динамический метод доступа	17
В.5. Макетный метод организации интерфейса с базами данных	20
В.6. ИНЕС как инструментальная система	26
Г л а в а 1. Описание данных	29
1.0. Введение	29
1.1. Общая характеристика модели данных ИНЕС	33
1.2. Простые данные	36
1.3. Составные (структурные) данные	37
1.4. Ссылочные данные	44
1.5. Пример описания данных	49
1.6. Синтаксис ЯОД	53
1.7. Трансляция и печать описания данных	57
1.8. Корректировка ДОД	59
Г л а в а 2. Проектирование баз данных в среде ИНЕС	62
2.0. Введение. Этапы проектирования баз данных	62
2.1. Одна или несколько баз данных	64
2.2. Выделение основных иерархических структур	66
2.3. Инверсные входы	70
2.4. Деревья-справочные (нормативно-справочная ин- формация)	74
2.5. Дерево разузлования и дерево классификации	77
2.6. Как реализовать сетевую ссылку	79
2.7. Определение типов вершин	80
Г л а в а 3. Ввод данных в базу	84
3.0. Введение	84
3.1. Основные понятия	90
3.2. Описание отображения входного документа	94
3.3. Язык описания компонент	109
3.4. Организация ввода	114

Г л а в а 4. Система запросов	123
4.0. Введение	123
4.1. Движение по базе данных	127
4.2. Рабочая область СЗ	146
4.3. Обращение к подпрограммам	155
4.4. Оформление запроса	158
4.5. Запрос с параметрами	161
4.6. Стандартные программы системы запросов	164
4.7. Синтаксис языка запросов	173
4.8. Работа с несколькими базами данных и словарями	178
4.9. Процедуры запросной системы	183
Г л а в а 5. Вывод данных	185
5.0. Введение	185
5.1. Отладочная печать базы данных и ее частей	201
5.2. Экспресс-вывод	203
5.3. Набор составного текста в рабочем поле	207
5.4. Макетный вывод	210
5.5. Управление потоками сообщений	226
Заключение	250
Список литературы	252

ПРЕДИСЛОВИЕ

В настоящей книге описываются основные возможности системы управления базами данных (СУБД) ИНЕС. СУБД ИНЕС позволяет построить концептуальную модель объекта (объекта управления, научной дисциплины или другой предметной области), составить соответствующее модели описание базы данных (БД), описать средствами макетного ввода любые формы входных документов и загрузить базу, получить средствами системы запросов и системы вывода любую справку из БД и оформить ее в виде выходного документа требуемого формата. ИНЕС включает также большой комплекс диалоговых и системных средств, которые в книге не рассматриваются. При изучении ИНЕС этим средствам посвящаются, как правило, отдельные курсы. Тем не менее в заключительном разделе главы, посвященной системе вывода, описываются средства переключения потоков сообщений, которые позволяют направить поток вывода на дисплей и тем самым перевести запрос из пакетного режима в диалоговый. При этом обеспечивается не только получение на экране дисплея результатов запроса, но и диалоговый режим ввода параметров и управления исполнением запроса.

Концепции, положенные в основу разработок подсистем ИНЕС, кратко излагаются во Введении, а также во вводных параграфах каждой главы. В отличие от основного текста книги, в этих частях некоторые распространенные понятия используются без определений. Для уточнения этих понятий даются ссылки на работы, в которых они рассматриваются подробно. Читатели, которые хотят научиться пользоваться системой, но не интересуются ее концепциями, могут опустить вводные параграфы. Напротив, для чтения всех остальных частей книги не требуется никаких дополнительных знаний.

Следует отметить, что использование СУБД ИНЕС доступно не только квалифицированным программистам, но и широкому кругу специалистов, знакомых только с основами программирования. Это связано с тем, что при построении АСУ, ИПС, АСОД средствами

СУБД принципиально изменяется стиль работы проектировщиков и программистов. Вместо прежних технических заданий на программы ввода и вывода конкретных форм и на алгоритмы обработки отдельных реквизитов входных форм для получения значений данных выходных форм, разрабатывается концепция объекта и технологический интерфейс пользователей с БД. Входные и выходные формы проектируются только из соображений удобства работы конечных пользователей — независимо друг от друга и от структуры БД. Очевидны преимущества такого подхода: интегрированность и стабильность БД, простота модернизации и развития интерфейса с базой.

Настоящая книга рассчитана на широкий круг читателей:

- прикладных программистов и проектировщиков ИПС, АСУ и АСОД, разрабатываемых на основе СУБД ИНЕС;

- системных программистов, занятых разработкой СУБД и программ окружения (средства ИНЕС могут использоваться и используются в сочетании с другими СУБД);

- администраторов БД, ведущих эксплуатацию баз данных, построенных средствами ИНЕС;

- студентов и аспирантов, изучающих СУБД и их применение в информатике;

- руководителей подразделений, в которых ведутся разработки на базе системы ИНЕС.

Главным источником трудностей в работе над книгой было постоянное развитие системы ИНЕС: приходилось перерабатывать уже написанные главы, вместо описания некоторых языковых конструкций в их текущем состоянии, хотелось доработать саму систему. Как правило, возможности, реализованные после 1985 г., оговариваются в тексте книги.

Семилетний опыт преподавания ИНЕС в МИСиС (Московский институт Стали и Сплавов) показывает, что каждая глава излагается в 1—4 лекциях и требует для полного усвоения 1—3 лабораторных занятий в дисплейном классе. В книге приводится много примеров, но ее ограниченный объем, к сожалению, не позволил дать упражнения к главам. С вариантами упражнений можно познакомиться в работе [31].

Автор выражает сердечную благодарность члену-корреспонденту АН СССР Авену О. И. и к.ф.-м.н. Марченко Н. В. за ряд полезных советов, к.т.н. Щелкачевой И. В., к.ф.-м.н. Титову В. К. и к.т.н. Жаринову А. Н. за помощь в работе над книгой.

ВВЕДЕНИЕ

В.1. База данных, СУБД, описание данных

Базы данных (БД) составляют в настоящее время основу компьютерного обеспечения информационных процессов, входящих практически во все сферы человеческой деятельности. В настоящее время без использования БД не может идти речи ни о каких АСУ [1]. Действительно, процессы обработки информации имеют общую природу и опираются на описание фрагментов реальности, выраженное в виде совокупности взаимосвязанных данных. БД являются эффективным средством представления структур данных и манипулирования ими. Распространение БД подкрепляется развитием их концепции и совершенствованием ЭВМ, составляющих техническую основу их реализации.

Концепция БД предполагает использование интегрированных средств хранения информации, позволяющих обеспечить централизованное управление данными и обслуживание ими многих пользователей. При этом БД должна поддерживаться в среде ЭВМ единым программным обеспечением, называемым *системой управления базами данных* (СУБД). СУБД вместе с прикладными программами решения определенных задач и данными называют *банком данных*.

Одно из основных назначений СУБД — поддержка программными средствами представления, адекватного реальности. Введем некоторые понятия, позволяющие уточнить эту роль и назначение баз данных.

Предметной областью называется фрагмент реальности, который описывается или моделируется с помощью БД и ее приложений. В предметной области выделяются *информационные объекты* — идентифицируемые объекты реального мира, процессы, системы, понятия и т. д., сведения о которых хранятся в БД.

Этапам реализации БД соответствуют уровни описания предметной области, изображенные на рис. 1: реальность в том виде, как она существует (а), концептуальное описание реальности (б),

представление описания в виде формального текста (в) и физическая реализация БД на машинных носителях (г).

Концептуальное описание предметной области предполагает выделение в ней информационных объектов, их характеристик (элементарных данных) и связей и в конечном счете представляется в виде структур поименованных данных. Для ввода в ЭВМ полученное описание должно быть представлено в терминах специального языка описания данных (ЯОД), который входит в комплекс средств СУБД.

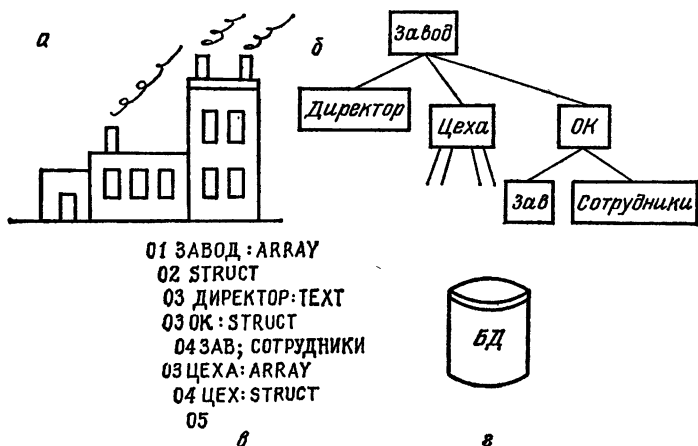


Рис. 1. а — Объект, каков он существует в реальном мире. б — Объект, каким его представляет проектировщик (концептуальное описание). в — Описание объекта в виде формального текста на ЯОД. г — Физическая модель

На основании текста ЯОД формируется собственно БД на внешних носителях. Описание объекта в базе составляется из совокупности характеристик — простых типов данных, которым приписываются конкретные значения. Простые данные объединяются в общее описание с помощью составных или структурных типов данных.

Простое (элементарное) данное — это наименьшая семантически значимая поименованная единица данных (например, ФИО, ДОЛЖНОСТЬ, АДРЕС и т. д.). Значение простого данного описывает представленную им характеристику объекта для каждого экземпляра объекта. Имена простых данных хранятся в описании БД, в то время как их значения запоминаются в самой БД.

Совокупность простых данных можно объединить в *составное данное* двумя способами. Во-первых, можно соединить несколько разнотипных данных. Например, данное АНКЕТА состоит из данных ТАБЕЛЬНЫЙ НОМЕР, ФИО, ГОД РОЖДЕНИЯ, ПОЛ, ДОЛЖНОСТЬ, ЗАРПЛАТА. По этому принципу образуется *структурное*

данное или *данное типа структура*. Описание структуры состоит из перечисления ее составных частей, значение — из значений составляющих ее данных. Во-вторых, составное данное может объединять совокупность однотипных данных (список сотрудников предприятия, послужной список сотрудника, список публикаций и т. п.).

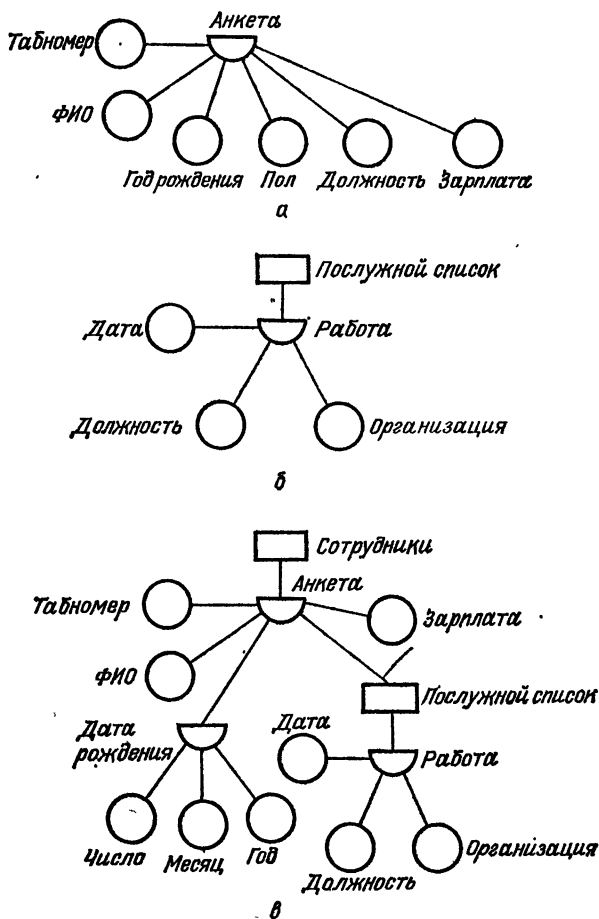


Рис. 2

Составное данное этого типа называется *массивом*. В описании массива достаточно указать описание одного элемента, значение массива представляется однородным списком значений его элементов.

В общем случае составные данные представляют собой объединенную под одним именем совокупность данных любых типов, в том

числе структур и массивов с произвольной глубиной вложенности составных данных. На рис. 2 представлено графическое изображение структуры (а), массива (б) и многоуровневого данного СОТРУДНИКИ (в) (в обозначениях гл. 1).

Элементы массива могут идентифицироваться ключом — данным, значения которого взаимно однозначно определяют экземпляры элементов.

Составные типы данных представляют иерархические связи значений данных в БД. Представление предметной области в виде структур данных с иерархическими связями носит название *иерархической модели данных*. В общем случае в базе могут быть определены также сетевые связи, позволяющие описать сеть — ориентированный граф произвольного вида. Представление предметной области в виде сетевых структур данных общего вида называется *сетевой моделью данных*. Сетевые связи реализуются путем отождествления отдельных данных БД.

Структура БД, заданная описанием данных, определяет важные общие характеристики базы. С одной стороны, данные базы должны представлять достаточно полное и точное описание предметной области, что позволяет обеспечить стабильность и эффективность развития БД. С другой стороны, структурные взаимосвязи данных определяют пути доступа к данным, и следовательно, описание данных решающим образом определяет эффективность функционирования приложений БД. Процесс построения концептуального описания с учетом всех необходимых факторов называется *процессом проектирования БД*.

В.2. Интерфейс с базой данных

Интерфейс определяет переход от представления данных в БД к представлению, принятому среди пользователей, и обратно. В общем случае пользователи представляют данные в виде документов различных видов, от произвольных текстов до справок и таблиц фиксированного формата. В основу концепции интерфейса ложится поэтому понятие документа, описывающее это разнообразие.

Под *документом* понимается произвольный структурированный текст, который может быть представлен на алфавитно-цифровых печатающих устройствах [28]. При этом под структурой текста понимается структура взаимосвязей данных, составляющих текст. Условие представимости текста на АЦПУ накладывает ограничения на оформление текста и позволяет отделить документы от машинной графики (рисунков произвольного вида).

В процессе проектирования, как правило, возникает необходимость точного учета структур данных документов. Для полного представления этих структур, естественно, могут использоваться

средства описания данных БД. Тем самым облегчается процесс сопоставления БД и документов при организации интерфейса.

Совместная реализация БД и интерфейса на единой концептуальной основе предполагает сопоставление соответствующих понятий концептуального описания (схемы) с понятиями пользователей

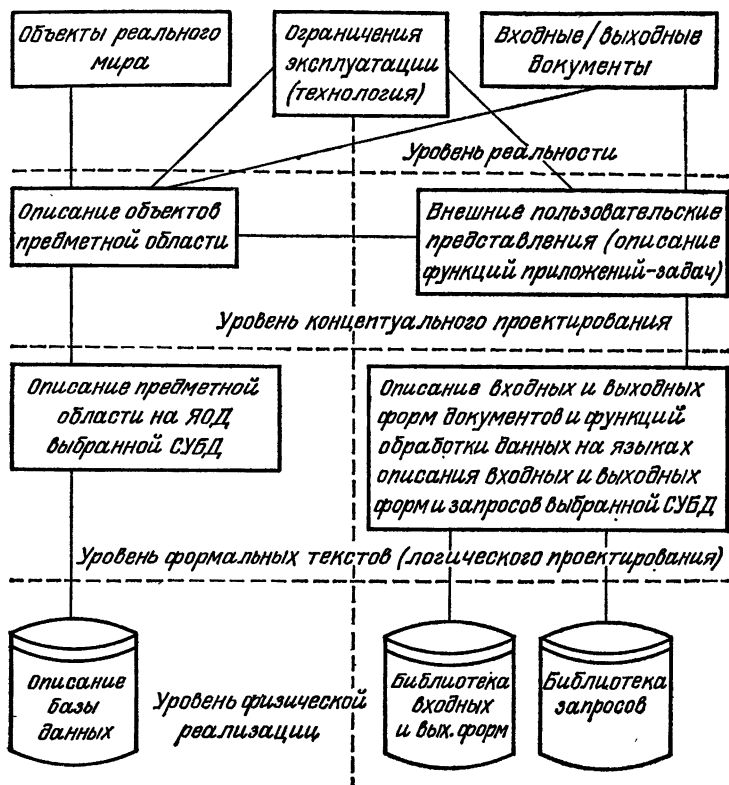


Рис. 3

(подсхемой). Это сопоставление можно провести с помощью рис. 3, представляющего расширение структуры уровней представления данных рис. 1.

На верхнем уровне рис. 3 представлена совокупность объектов предметной области и явно выделенная из них совокупность документов в том виде, как они существуют в реальной организации. Этот уровень представляет «исходный материал» для процессов проектирования и реализации базы.

На концептуальном уровне сосредоточен процесс проектирования БД — разработка структуры данных БД, представляющей

описание реальности и соответствующей требованиям пользователей. Конкретные функциональные требования пользователей и предполагаемое их обеспечение отображаются понятием *пользовательского представления данных*. В общем случае пользовательское представление включает так называемое *локальное внешнее представление* функций обработки данных, а также *определение форматов входных и выходных данных*. В процессе проектирования используются локальные представления функций, отображающие необходимые пользователю элементы данных и взаимосвязи. Локальные представления строятся на основании структур данных документов и используются в процессе проектирования одним из двух способов.

В первом случае процесс проектирования начинается с описания реальности (информационных объектов предметной области и связей между ними). Впоследствии полученная структура модифицируется таким образом, чтобы удовлетворялись функциональные требования приложений. Во втором случае процесс проектирования заключается в синтезе функциональных пользовательских представлений данных. При этом на определенном этапе в структуру данных включается описание предметной области с тем, чтобы убедиться в совместимости этого описания с функциональными требованиями. Выбор одного из двух подходов определяет общую ориентацию БД (см., например, в [54] сравнение объектных и функционально-ориентированных БД). Заметим, что третьим важным фактором проектирования являются вопросы технологии ведения БД, в первую очередь — их организационная сторона.

На концептуальном уровне представления данных специфицируются также задачи ввода и вывода информации на основании пользовательского представления данных. Задача ввода данных в общем случае включает извлечение данных из входных документов, их контроль и отображение из структуры документного представления в структуру БД. Задача вывода данных заключается в получении требуемых данных на основании информации, содержащейся в базе, и представлении их в виде документа заданной формы. Точное определение функциональных пользовательских требований на этапе концептуального проектирования и соотнесение их со структурой БД позволяет определить структуру и содержание запросов — функций извлечения и обработки данных. При этом выявляются элементы данных и взаимосвязи в БД, необходимые для обеспечения запроса, и определяются процедуры получения из них требуемых данных. Таким образом, на концептуальном уровне специфицируются все основные функции ведения и обработки данных в БД.

Результаты, полученные на концептуальном уровне представления данных, оформляются в виде текстов, написанных на языке СУБД. На основании этих текстов организуется физический уровень представления.

Интеграция БД со средствами интерфейса потребовала расширения функций программной поддержки базы. Первоначально в функции СУБД входила практически только поддержка модели данных, включающей характеристики структур данных, ограничения на значения данных и их соотношения, операции с данными *). Реализация базы данных средствами СУБД была первым шагом, за которым следовала серия проектов, нацеленных на генерацию нужного класса выходных документов и на обеспечение теледоступа.

Развитие СУБД привело к включению в ее состав всех основных средств, обеспечивающих реализацию и функционирование базы данных: описание данных, ввод и контроль данных, доступ к данным и манипулирование ими, вывод данных.

В настоящее время перспектива развития СУБД связывается с дальнейшей интеграцией средств СУБД и общесистемных программных средств. Реализация единого подхода к обработке информации предполагает создание единого комплекса матобеспечения, включающего СУБД и ее окружение, средства теледоступа и систему средств проектирования и программирования, поддерживаемые экспертными системами. Эта перспектива находит непосредственное отражение в развитии системы ИНЕС, общему описанию которой посвящается следующий раздел.

В.3. Что такое ИНЕС?

Информационная единая система (ИНЕС) является программным продуктом нового типа, направленным на повышение эффективности решения широкого круга задач, связанных с использованием ЭВМ [7].

В систему ИНЕС входят: система управления базами данных, поддерживающая модель данных с характерными свойствами семантических моделей; набор готовых диалоговых систем и средств их построения; системные средства (см. рис. 4). В частности, ИНЕС содержит целый спектр языковых средств для создания проблемных программ, ориентированных на различное взаимодействие с пользователем, — язык для описания сценариев диалога, язык описания входных и выходных форм, язык запросов и т.д.

ИНЕС не является механическим объединением своих частей, поскольку все они реализованы в единой системе программирования и пользуются едиными системными средствами, — например, системой обмена сообщениями, инструментарием блочного (структурного) программирования, общим базовым методом доступа. Этим обеспечивается языковое и структурное единство системы.

*) Это определение модели данных соответствует, например, [51, 64]. Часто встречается более узкое понимание модели как структуры данных.

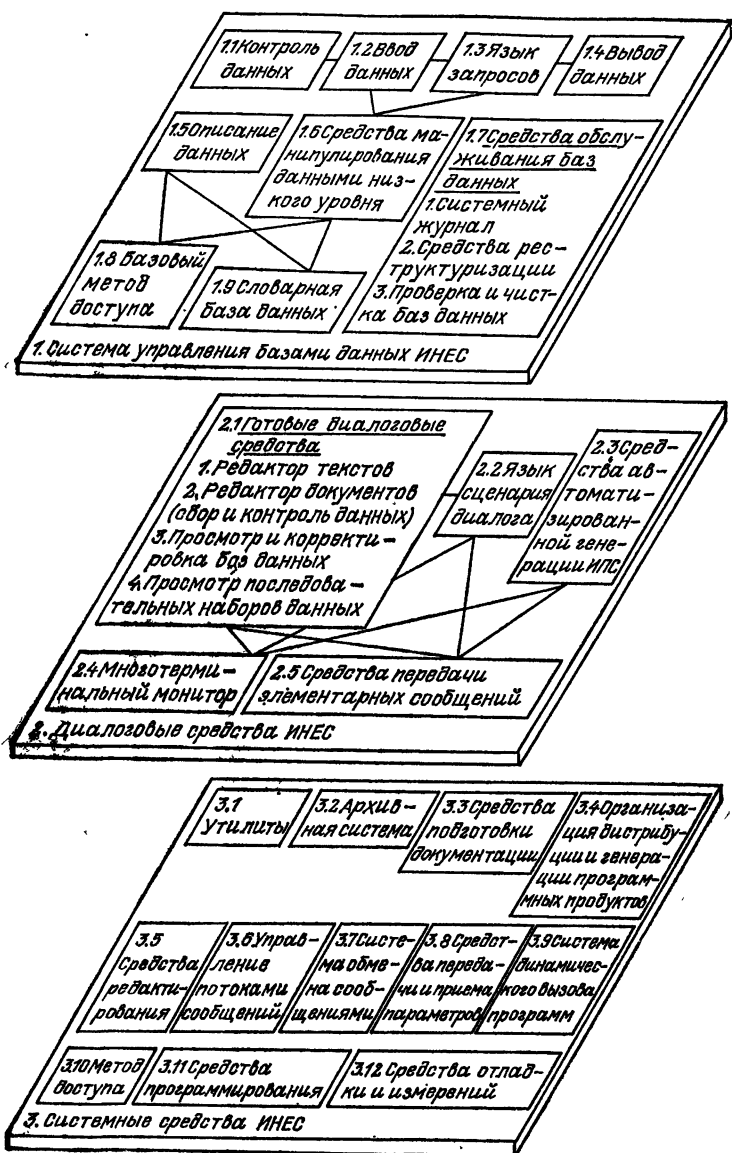


Рис. 4

Средствами ИНЕС решаются следующие основные задачи:

- создание информационных систем разного профиля;
- создание диалоговых систем;
- организация работы вычислительного центра;
- организация рабочих мест программистов;
- отладка и измерение характеристик функционирования программ;
- создание и распространение пакетов прикладных программ.

Важной особенностью ИНЕС является полнота обеспечения так называемого «жизненного цикла» задач, ведущихся в ее среде. Действительно:

1. Для создания баз данных в ИНЕС имеются средства описания БД, способные моделировать любую из известных структур данных; средства ввода данных в базу, контроля и редактирования документов; средства корректировки баз данных; средства запросов; управление печатью выходных форм и диалогом с пользователем ВД; обеспечение контроля за доступом; средства копирования, архивации и восстановления баз данных; средства контроля согласованности базы данных на различных уровнях — логическом и физическом.

2. Для организации диалоговых систем ИНЕС предоставляет набор готовых диалоговых средств: сбора и контроля данных, редактирования документов для ЕС ЭВМ, мини- и персональных ЭВМ; просмотра и корректировки баз данных; просмотра последовательных наборов данных. Кроме того, имеются средства для разработки диалоговых программ: язык высокого уровня для описания сценария диалога; средства автоматизированной генерации информационно-поисковых систем; функции работы с дисплеями из языков высокого уровня, не требующие перетрансляции программ при смене типа используемых дисплеев; многотерминальный монитор, обеспечивающий параллельную работу многих диалоговых программ в одном шаге задания без дополнительных затрат памяти; диалоговые возможности системы маршрутизации сообщений.

3. Для организации работы вычислительного центра в ИНЕС имеются: средства компактного хранения исходных текстов; процедуры трансляции, редактирования и выполнения программ на языках ИНЕС и всех общепринятых языках программирования; экранные редакторы различного назначения; архивная система; средства обеспечения работы с общими библиотеками; средства визуализации операционной обстановки; различные утилиты.

Перечисленные средства организации работы ВЦ позволяют обеспечить:

- доступность вычислительной системы для любого пользователя в любое время;
- экономное использование ресурсов вычислительной системы, в особенности — дисковой памяти;

— централизованное обслуживание информационного фонда вычислительного центра;

— повышение надежности хранения информации;

— минимальные требования к системной квалификации проблемных программистов;

— минимальные требования к квалификации операторов.

4. Для организации работы программиста ИНЕС предоставляет:

— Диалоговый экранный редактор, согласованный со средствами организации работы вычислительного центра. Редактор позволяет редактировать тексты любой логической структуры, запускать задания на исполнение, просматривать результаты, обслуживать диалоговые программы. Важной особенностью редактора ИНЕС является повышенная надежность сохранности информации и комфортность редактирующих функций.

— Средства программирования на ассемблере, обеспечивающие наглядную структуризацию программ (этим повышается уровень ассемблера как языка, при сохранении его возможностей в эффективной реализации). Средства программирования включают средства измерения и отладки программ на ассемблере и средства ввода и вывода, обычно трудоемко реализуемые на этом языке.

— Средства помощи программисту по реализации потребностей, являющихся типовыми или предполагающих углубленное знание операционной системы. Эти средства представлены набором простых для понимания макроопераций с естественными операндами.

5. В среде ИНЕС можно обратиться из любых языков программирования к измерительным и отладочным средствам, включающим: возможность выдачи информации о завершении задачи (имеет ряд преимуществ перед стандартным «дампом», позволяет при необходимости возобновлять работу программы в целом или отдельных ее частей); возможность слежения за динамикой рабочей памяти (позволяет определять целесообразность затрат на оперативную память и, в качестве побочного эффекта, получать распечатку требуемых областей памяти — целиком или только измененных частей); возможность получения статистики выполняемых команд в соответствии со структурой программы для выявления наиболее трудоемких частей программы; средства регистрации сообщений пользователя к системе и системы к пользователю, позволяющие восстановить код диалога с целью отладки.

6. Для разработки и распространения пакетов прикладных программ в ИНЕС имеются средства автоматической генерации заданий с регистрацией шагов этих заданий в журнале системы; система слежения за динамикой изменения наборов данных и отдельных разделов библиотек; система подготовки, редактирования и выдачи документации, позволяющая синхронизировать изменения в программах с изменениями в документации; система формирования

дистрибутивной ленты пакета, содержащей документацию к пакету; средства быстрой генерации пакета на вычислительном центре заказчика.

Важная особенность системы ИНЕС заключается в том, что она является оригинальной разработкой, не имеющей зарубежных аналогов. Масштабы и многообразие средств ИНЕС являются результатом непрерывного развития системы большим программистским коллективом в течение последних десяти лет. Преимуществом системы является возможность контакта с разработчиком, использование русского алфавита и экономное использование вычислительных ресурсов. Система не требует специального сопровождения системных программистов, регенерации ОС и т. д.

В настоящей книге рассматривается только СУБД ИНЕС, — верхний «слой» в схеме рис. 4, обеспечивающий разработку и использование БД в стандартной среде ОС ЕС ЭВМ. Три оригинальные идеи положены в основу программного обеспечения СУБД: базовый метод доступа, определяющий основные характеристики модели данных ИНЕС, макетный метод организации интерфейса с базой данных, обеспечивающий участие широкого круга пользователей в проектировании входных и выходных форм документов, и организация ИНЕС как инструментальной системы, позволяющая использовать подсистемы ИНЕС в различных сочетаниях для обслуживания индивидуального пользователя.

В.4. Базовый динамический метод доступа

Выбор метода доступа в СУБД определяет структуры физического хранения данных на внешней памяти и задает принципиальные выразительные возможности модели данных (см. [52], а также п. 2.0 о физическом, логическом и концептуальном уровнях организации БД). В общем случае содержательное описание предметной области, составленное в терминах структур данных (концептуальная модель), должно быть модифицировано в модель данных СУБД, поддерживаемую средствами заданного метода доступа. Например, хранение данных в виде записей фиксированного формата приводит к появлению трудоемкого этапа разбиения общей структуры данных на подструктуры — двумерные таблицы (приведение к нормальным формам). Применение таких записей характерно, например, для «классических» моделей данных (иерархическая, сетевая и реляционная модели).

Базовый динамический метод доступа, принятый в СУБД ИНЕС, базируется на известном принципе динамической организации массивов. Суть его поясняется ниже на примере разворачивающейся во времени ситуации.

При вводе данных в базу на внешней памяти ЭВМ открывается первый блок для накопления данных, и данные направляются в этот блок, где они выстраиваются в лексикографическом порядке. При этом в каждой вершине физической структуры данных (в терминах ИНЕС — дерева данных или ДД) хранится ссылка на первую подчиненную вершину и следующую за ней на том же уровне (из подчиненных общей исходной вершине). Например, дерево данных, изображенное на рис. 5а, хранится в виде рис. 5б. Схема первоначального заполнения блока приводится на рис. 6а. Когда емкость первого блока исчерпывается, на диске открывается второй

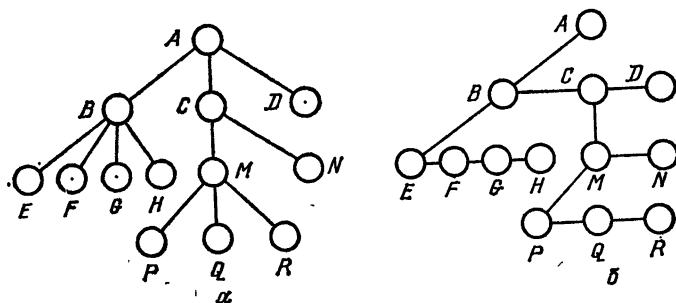


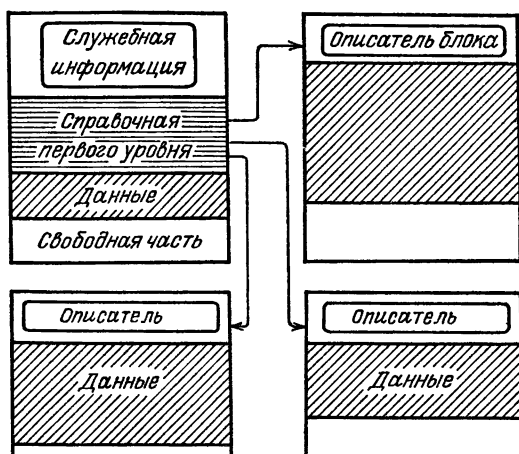
Рис. 5

блок, и данные распределяются приблизительно поровну между двумя блоками. Старшие данные (в смысле лексикографического порядка) остаются в первом блоке, младшие переносятся во второй блок. В оперативной памяти организуется справочная, куда выносятся номера блоков и старшие данные блоков. Следующее вводимое данное сравнивается в оперативной памяти со старшими данными блоков из справочной. В оперативную память вызывается блок, соответствующий лексикографическому значению нового данного, и оно занимает в нем свое место.

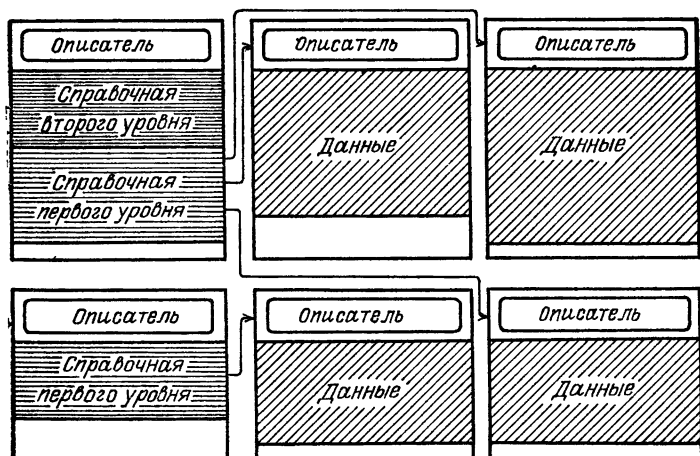
В общем случае динамическая организация памяти позволяет в любое место внешней памяти вставить произвольное количество новых данных: при переполнении одного из уже существующих блоков открывается следующий блок, куда пересылаются младшие данные: в справочную заносится старший ключ нового блока (см. схему состояния блоков на рис. 6б). Это свойство допускает динамическое изменение состава данных, а также снимает количественные ограничения на структуры данных, свойственные моделям, базирующимся на применении записи (глубина вложенности структур и массивов, размеры значений данных и т. д.). Концепция записи не является необходимым элементом модели данных ИНЕС (см. общую



a



б



в

Рис. 6

характеристику модели в пп. 1.1, 2.0). При переполнении справочной возникает справочная второго уровня, в строках которой хранятся ссылки на справочные первого уровня (см. схему на рис. 6а). В дальнейшем могут возникать справочные более высоких уровней.

В общем случае освободившиеся при удалении данных блоки поступают в резерв свободных, а освободившаяся память внутри блока поступает в резерв свободной памяти блока.

При доступе к данным, когда требуется по ключу найти соответствующую информацию, нужный блок ищется путем сравнения с ключами из справочной. Этот блок вызывается в оперативную память, после чего дальнейший поиск ведется внутри блока. Следует отметить, что при доступе к данным специальная подсистема ИНЕС имитирует для пользователя работу с базой, организованной в соответствии с ее описанием в виде иерархических и сетевых структур данных. Пользователь освобождается от знания фактической реализации базы.

Два важных утверждения характеризуют базовый динамический метод доступа ИНЕС:

- поиск конкретного данного по известной траектории, ведущей к нему от корня БД, осуществляется оптимальным образом, за минимальное число обменов с внешней памятью, — так как этот поиск ведется по многоуровневой логарифмической справочной;

- проход по всему ДД или любому его поддереву также оптимален, так как в каждом блоке хранится связный кусок ДД. После вызова блока в оперативную память совершается обход всего поддерева, хранящегося в этом блоке. Многочисленных перевызовов блоков не происходит.

Базовый динамический метод доступа является единственным методом доступа, принятым в ИНЕС (по аналогичному принципу организованы все использующиеся в ИНЕС основные наборы данных). Использование этого метода обеспечивает общность модели данных ИНЕС. Кроме того, пользователь ИНЕС избавлен от затруднений, связанных с выбором способа хранения данных.

В.5. Макетный метод организации интерфейса с базами данных

Модель данных, определяющая хранимые структуры данных и операции над ними, занимает, очевидно, центральное место в организации баз данных информационных систем. Вторым важнейшим фактором эффективного применения СУБД является организация интерфейса пользователя с БД.

Как упоминалось выше (см. п. 1.2 Введения), в общем виде задачей интерфейса является установление связи пользователя с БД, в частности — реализация универсальных средств отображения

структур данных БД в пользовательское представление данных (и обратно). Так как естественным представлением данных для пользователя является документ, проблема организации интерфейса с базой данных фактически выступает как проблема автоматизации ввода/вывода документов.

АНКЕТА		
ФИО <u>Гаврилов Юрий Петрович</u>		
ДАТА РОЖДЕНИЯ <u>25.02.1944</u>	ПОЛ <u>м</u>	
ДОЛЖНОСТЬ <u>старший инженер</u>	ОКЛАД <u>285</u>	
ТРУДОВАЯ ДЕЯТЕЛЬНОСТЬ		
ДОЛЖНОСТЬ	ДАТА ПОСТУПЛЕНИЯ	ОРГАНИЗАЦИЯ
<i>стажер</i>	01.08.1967	ВНИИЦАП
<i>инженер</i>	10.05.1968	ВНИИЦАП
<i>инженер</i>	01.10.1971	ПКБ АСУ
<i>ст. инженер</i>	12.12.1984	ПКБ АСУ
. . .		

Рис. 7

Сопоставление структуры данных БД с документами в рамках системы интерфейса предполагает выделение в документах содержания, также являющегося совокупностью взаимосвязанных данных. Содержание противопоставляется при этом оформлению документа (к оформлению документа относят, как правило, «пустографку», правила набора текстов, а иногда и вспомогательные переменные, такие, как номера страниц).

Два аспекта понятия документа (его содержание и форма) играют различную роль в подсистеме ввода/вывода. На рис. 7 представлен простейший документ — заполненная от руки данными пустографка (форма) анкеты. Для ввода в БД собственно данные должны быть извлечены из этого документа и представлены в машинно-ориентированном виде (см. на рис. 8 совокупность подготовленных для ввода данных из нескольких анкет; здесь знаком & обозначается конец повторяющейся группы данных), а знаком * — конец документа). При вводе эти данные распределяются по структурам БД в соответствии с заданным отображением (см. п. 3.0.1).

При выводе данные извлекаются из базы и обрабатываются средствами запроса, после чего они должны быть оформлены в виде документа. Для одних и тех же данных могут быть выбраны, оче-

видно, различные формы представления. На рис. 9 приводятся примеры оформления анкеты с помощью макетов-пустографок, изображенных на рис. 10. В этих макетах квадратными скобками обозначены окна — места размещения переменной информации, буквой Р обозначена часть макета, повторяющаяся в документе (в макетном выводе ИНЕС для обозначения окна используется строка из девяти ток с ведущим вопросительным знаком — см. п. 5.4).

Оформление полученных структур данных в виде документов произвольного формата является одной из центральных задач организации интерфейса (см. п. 5.0.1).

ГАВРИЛОВ ЮРИЙ ПЕТРОВИЧ/25.02.1944/М/СТАРШИЙ ИНЖЕНЕР/285/
СТАЖЕР/01.08.1967/ВНИИЦАП/
ИНЖЕНЕР/10.05.1968/ВНИИЦАП&
ИНЖЕНЕР/01.10.1971/ПКБ АСУ&
СТ. ИНЖЕНЕР/12.12.1984/ПКБ АСУ*

ДОРОХОВ ПАВЕЛ АНАТОЛЬЕВИЧ/12.08.1936/М/ИНЖЕНЕР/180/
РАБОЧИЙ/1954/АЗЛК&
СТУДЕНТ/1962/МВТУ&
ИНЖЕНЕР/1967/ПКБ АСУ*

КУЛАКОВА ГАЛИНА ИВАНОВНА/17.05.1940/Ж/ТЕХНИК/120/
ТЕХНИК/1960/ПКБ АСУ*

Рис. 8

Конечному пользователю необходимо предоставить возможность не только использовать конечные результаты счета, но и самому участвовать в процессе разработки приложений, т. е. в процессе программирования. Таким образом, в задачи интерфейса входит обеспечение пользователей-непрограммистов достаточно простыми средствами программирования.

Эта цель может быть достигнута путем передачи пользователю программирования отдельных компонент описания документа в системах ввода/вывода. Соответствующий метод организации интерфейса, основанный на выделении двух аспектов документа (формы и содержания), носит название макетного метода (см. [27, 28]).

Макетный метод организации интерфейса предполагает следующие 3 этапа подготовки ввода/вывода информации:

1. Разработка макета документа, содержащего разграфку, заголовки и другие неизменяемые части вводимого или выводимого документа, а также «окна», в которых может располагаться переменная информация.

Макет выполняется на обычных для ЭВМ или специальных бланках в виде, наглядно представляющем форму документа. Это позволяет разработчику непосредственно на бланке изменять формат,

вносить коррективы и исправлять ошибки. Макет на машинном бланке без кодирования пригоден к перфорации и вводу в ЭВМ для отладки и выдачи по нему документа, а также для записи в библио-

АНКЕТНЫЕ СВЕДЕНИЯ

ФИО: ГАВРИЛОВ ЮРИЙ ПЕТРОВИЧ

ПОЛ	ДОЛЖНОСТЬ	ОКЛАД	ДАТА РОЖДЕНИЯ
МУЖ	СТ. ИНЖЕНЕР	285	25.02.1944

СПИСОК ЗАНИМАЕМЫХ ДОЛЖНОСТЕЙ

- ДОЛЖНОСТЬ: СТАЖЕР
ДАТА ПОСТУПЛЕНИЯ: 1967
ОРГАНИЗАЦИЯ: ВНИИЦАП
- ДОЛЖНОСТЬ: ИНЖЕНЕР
ДАТА ПОСТУПЛЕНИЯ: 1968
ОРГАНИЗАЦИЯ: ВНИИЦАП
- ДОЛЖНОСТЬ: ИНЖЕНЕР
ДАТА ПОСТУПЛЕНИЯ: 1971
ОРГАНИЗАЦИЯ: ПКБ АСУ
- ДОЛЖНОСТЬ: СТ. ИНЖЕНЕР
ДАТА ПОСТУПЛЕНИЯ: 1984
ОРГАНИЗАЦИЯ: ПКБ АСУ

(а)

АНКЕТА

ФИО	ГАВРИЛОВ ЮРИЙ ПЕТРОВИЧ
ДАТА РОЖДЕНИЯ	25. 02. 1944
ПОЛ	МУЖ
ДОЛЖНОСТЬ	СТАРШИЙ ИНЖЕНЕР
ОКЛАД	285

ТРУДОВАЯ ДЕЯТЕЛЬНОСТЬ

ВНИИЦАП	СТАЖЕР, 1967
	ИНЖЕНЕР, 1968
ПКБ АСУ	ИНЖЕНЕР, 1971
	СТАРШИЙ ИНЖЕНЕР, 1984

(б)

Рис. 9

теку макетов (в последнее время разработка макетов все чаще ведется без бланков, с экрана дисплея).

2. Описание процесса обработки и получения нужных структур данных (содержания документа). Это описание определяет при вводе отображение переменных из окон документа в БД или в память,

АНКЕТНЫЕ СВЕДЕНИЯ			
ФИО: []			
ПОЛ	ДОЛЖНОСТЬ	ОКЛАД	ДАТА РОЖДЕНИЯ
[]	[]	[]	[]
СПИСОК ЗАНИМАЕМЫХ ДОЛЖНОСТЕЙ			
<div style="display: flex; justify-content: space-between;"> [] ДОЛЖНОСТЬ: [] } P </div> <div style="display: flex; justify-content: space-between;"> ДАТА ПОСТУПЛЕНИЯ: [] } </div> <div style="display: flex; justify-content: space-between;"> ОРГАНИЗАЦИЯ: [] } </div>			
(a)			
АНКЕТА			
ФИО ДАТА РОЖДЕНИЯ ПОЛ ДОЛЖНОСТЬ ОКЛАД	<div style="display: flex; justify-content: space-between;"> [] [] [] </div> <div style="display: flex; justify-content: space-between;"> [] [] [] </div>		
ТРУДОВАЯ ДЕЯТЕЛЬНОСТЬ			
<div style="display: flex; justify-content: space-between;"> [] [] </div> <div style="display: flex; justify-content: space-between;"> [] [] </div>	<div style="display: flex; justify-content: space-between;"> [] [] </div> <div style="display: flex; justify-content: space-between;"> [] [] </div>		
(б)			

Рис. 10

а при выводе — выбор по нужным критериям данных из БД в переменные документа (окна). По сути, является описанием подсхемы БД и сложных алгоритмов обработки и составляет функцию администратора БД.

3. Описание отображения данных документа в окна макета. Описание задает логическую структуру данных документа (состав и иерархию), определяет арифметические выражения для получе-

ния расчетных данных, устанавливает форматы данных, правила редактирования, способы подготовки данных (знаки-разделители), дополнительные имена и др. Это описание служит связующим звеном между макетом и описанием обработки данных. По трудности освоения соответствует формализму алгебраических формул (арифметических выражений).

Реализация перечисленных трех этапов подготовки документа требует различной программистской квалификации. Простейшие средства составления макетов могут быть переданы практически любому *конечному* пользователю *) (соответствующее теоретическое и практическое обучение занимает не более одного дня). Описания заполнения окон документа требуют большей квалификации, однако также могут быть переданы обученному заказчику (тем самым ему передается, например, возможность задавать арифметические выражения для обработки полученных из БД данных при отображении их в окна пустографики, средства представления даты и времени и т. д.).

Обработка данных требует, как минимум, освоения основных средств языка запросов высокого уровня, в сложных случаях используется и более широкий круг вспомогательных средств, а также программы, написанные на других языках программирования (ассемблер, ПЛ/1, фортран, кобол). Освоение языка запросов требует специальной подготовки.

Следует отметить, что наличие в системе обработки данных компонент (а) формы, (б) содержания и (в) отображения еще не означает, что в ней применяется макетный метод. Эти компоненты, вообще говоря, присутствуют в любой системе ввода-вывода и обеспечиваются в каждом языке программирования. Так, в фортране компонента формы обеспечивается оператором `FORMAT`, компонента содержания — оператором `PRINT`, компонента отображения — фортранной программой, включающей в себя эти операторы. В языке РПГ все три компонента обеспечиваются разными колонками бланка выходных данных и т. д. В то же время понятие макета документа отсутствует в универсальных языках программирования (фортран, ПЛ/1, кобол, РПГ).

В противоположность общепринятому неявному использованию компонент (а), (б) и (в), можно говорить о применении макетного метода, если явно выделены компоненты, введено понятие макета,

*) Конечные пользователи — это потребители результатов обработки данных, они или вовсе не программируют, или занимаются этим в силу необходимости решать свои задачи. Классификация конечных пользователей по искусству обращения с ЭВМ приводится, например, в работе: Rockart J. F., Flannery L. S. The Management of End User Computing. — Comm. of the ACM, 1983, V. 26, № 10. — P. 776—784.

который поддерживается системой как самостоятельная функциональная единица, и, кроме того, выполнение хотя бы одного этапа может быть предоставлено пользователю-непрограммисту. В типичном случае, как говорилось выше, составление макета документа предоставляется пользователю, после чего процесс обработки данных и заполнения окон документа обеспечивается программистом, возможно, с участием пользователя.

Применение идей макетного метода позволяет также ориентировать программные средства интерфейса на большее или меньшее участие программиста в генерации документа. В общем случае при разработке системы ввода-вывода можно сделать акцент на ту или иную компоненту описания документа, т. е. расширить языковые средства одной компоненты за счет другой или даже опустить какую-либо компоненту, предоставив системе ее генерацию. Такое перераспределение ролей характерно для различных средств системы ввода-вывода ИНЕС (см. гл. 3, 5). В целом применение идей макетного метода позволяет, с одной стороны, произвести разделение труда между разными категориями пользователей: конечными пользователями, прикладными программистами, администраторами задач и баз данных и, с другой стороны, — служит основой для разработки математического обеспечения интерфейса с БД.

В СУБД макет получил новое значение как внешнее пользовательское представление БД. Однако в настоящее время ни в одной промышленно-сопровождаемой СУБД для ЕС ЭВМ (кроме ИНЕС [59]) макетный принцип последовательно не применяется.

В.6. ИНЕС как инструментальная система

В общем случае можно противопоставить два подхода в выборе исходных принципов построения СУБД. В первом случае строится одноплановая, «монокенная» система. Комплекс идей, заложенный в основу проекта такой системы, последовательно проводится в жизнь и фактически не меняется за все время существования системы. При этом обеспечивается концептуальное единство системы со всеми его преимуществами. С другой стороны, с самого начала предопределяется та или иная неполнота охвата предметной области.

Другой подход предполагает постоянный анализ потребностей, возникающих при использовании СУБД, в процессе создания информационных систем. При этом каждый раз в идеале предполагается рассмотреть все потребности, понять их сущность и составить единую картину, на основе которой строится СУБД. Реально же разнопредставленные потребности приводятся к общему виду с помощью определенного комплекса идей, после чего формируется только более или менее полное представление о проблемной области. Процесс построения общей картины в области потребностей связан

с выделением частей, соответствующих определенным видам работы с данными. Набор средств СУБД, удовлетворяющих потребностям какой-либо части, образует подсистему СУБД.

Полученную совокупность потребностей, очевидно, нельзя представить себе ни вполне изученной, ни стабильной. Тем самым строящаяся СУБД должна допускать возможности развития, причем не только в смысле расширения, но и смещения акцентов и даже смены идей.

Второй подход к построению систем управления базами данных называется инструментальным (см. [5]). Именно он применен к построению системы ИНЕС.

При разработке системы ИНЕС был рассмотрен ряд вопросов, всегда возникающих при построении информационной системы. К ним относятся:

- 1) способ сбора и контроля исходных данных;
- 2) форма их представления;
- 3) способ ввода данных в ЭВМ;
- 4) виды представления данных в ЭВМ;
- 5) способы общения человека с информационной системой;
- 6) форма представления выходных данных и др.

Каждая из перечисленных сторон деятельности с данными требует математического обеспечения. Между тем проблемы развития языков описания данных и манипулирования данными, лежащие в центре внимания большинства работ по базам данных, заслонили тот факт, что для человека, разрабатывающего на основе СУБД конкретную информационную систему, этот круг вопросов отнюдь не является решающим. Простота описания входных и выходных форм, возможности терминального доступа и методы организации сценария диалога имеют не меньшее значение.

Любой из этих вопросов должен являться (и является) предметом специального исследования. Принципиальное значение приобретает в настоящее время факт включения (или невключения) этих вопросов в область действия СУБД, т. е. признание или непризнание принципа целостности математического обеспечения информационных систем.

В последние годы по отношению к СУБД было сформулировано требование функциональной полноты системы (см. [59]). В рамках этого требования целостность математического обеспечения информационных систем признается абсолютно необходимой с практической точки зрения. Напротив, стыковка самостоятельных, пусть даже совершенных подсистем управления данными, ввода данных, диалоговой подсистемы и др. признается очень сложной проблемой, часто неразрешимой в рамках конкретной практической задачи.

При разработке ИНЕС подсистемы, обеспечивающие перечисленные выше функции обработки данных, разрабатывались на осно-

ве принципа интерфейсной адаптации, обеспечивающего возможности сборки подсистем. На таком пути системы желаемых конфигураций получаются как композиции нужных модулей, сопряжение которых осуществляется программами-переходниками. Идея состоит в том, чтобы получать желаемую конфигурацию системы не посредством переделки уже имеющихся модулей, а посредством введения обладающих нужными свойствами переходников. Таким образом, каждая новая система — это, в основном старые модули плюс новые или модифицированные старые переходники. При достаточной накопившейся сложности переходников они могут объявляться модулями (неизменными программными образованиями) и, в свою очередь, сопрягаться со старыми модулями с помощью новых переходников. Такой принцип формирования желаемых конфигураций системы обеспечивает живучесть создаваемым программным средствам, а каждой выпускаемой системе — свойство быть открытой для новых модулей и одновременно быть специализированной системой.

Можно сказать, что построение ИНЕС как инструментальной системы обеспечивает индивидуальное обслуживание пользователей. Действительно, здесь ставится цель не обслуживания всех в образе некоторого «среднего» пользователя, а обслуживания каждого индивидуального пользователя *).

*) В издательстве «Наука» в 1988 г. выходит монография Иванова Ю. Н. «Теория информационных объектов и системы управления базами данных». Эта теория послужила в значительной мере теоретической основой СУБД ИНЕС, в то же время она совершенствовалась и развивалась в процессе разработки ИНЕС.

Г л а в а 1

ОПИСАНИЕ ДАННЫХ

1.0. Введение

1.0.1. Основные направления развития языков описания данных. Проблемы описания данных для представления их в памяти ЭВМ имеют в настоящее время принципиальное значение для нескольких направлений исследований. Разработчики СУБД в основном исследуют методы структуризации, размещения и защиты данных на внешней памяти (см., например, [24]). Авторы новых универсальных и специализированных языков программирования (ЯП) исследуют абстрактные типы данных, которые определяют не только форматы представления данных в оперативной памяти, как это было традиционно в ЯП, но и возможные операции над ними. Специалисты в области искусственного интеллекта исследуют взаимодействие данных и процедур их получения как единой совокупности средств представления знаний. И, наконец, разработчики семантических моделей исследуют свойства данных как таковых без учета способов их представления в ЭВМ (в первую очередь при этом имеются в виду задачи описания предметной области).

Так как все перечисленные области имеют общий предмет исследования, — описание данных, — то и сами исследования в значительной степени пересекаются и обогащают друг друга. Следует отметить, что характерные черты развития этих областей нашли отражение в языке описания данных ИНЕС [8]. Рассмотрим подробнее эти общие характеристики.

Проблематика моделирования данных связана с таким представлением данных, которое наиболее естественно отражает реальный мир и может поддерживаться компьютерными средствами (см. [64]). Как уже отмечалось, развитие СУБД первоначально было сосредоточено на создании программных средств, позволяющих поддерживать те или иные виды структур данных на внешней памяти ЭВМ. В дальнейшем в центре внимания оказались вопросы опи-

сания реального мира и представления семантики соответствующих данных. Семантические характеристики «классических» — иерархической, реляционной и сетевой — моделей данных, реализованных различными СУБД, подверглись критике (см. [63]). Возникло относительно самостоятельное направление по построению и исследованию свойств так называемых семантических моделей (как правило, не получивших еще компьютерной реализации). Средства, предлагаемые авторами семантических моделей, — это понятия и методика их применения, результат — схема, отражающая структуру и взаимосвязи данных. На практике семантические модели предназначаются также для помощи при анализе и формализации предметной области.

В ИНЕС вводится небольшое число типов данных общего вида, которые реализуют основные понятия семантических моделей и могут использоваться практически без ограничений при построении структур данных. Организация физического уровня хранения данных СУБД ИНЕС (см. п. 2 Введения) позволяет снять количественные ограничения на структуры модели данных, свойственные большинству реализованных в настоящее время моделей. Пользователю предоставляются средства, необходимые для описания разнообразных предметных областей. С точки зрения проектирования язык описания данных (ЯОД) ИНЕС может быть квалифицирован как язык концептуально-логического уровня описания данных (см. п. 2.0).

Проанализируем язык описания данных ИНЕС с точки зрения тенденций развития языков программирования (ЯП). Структуры данных в современных ЯП составляют их существенную часть, и не видно причин, по которым эти структуры должны принципиально отличаться от структур данных в СУБД. Есть два основных понятия, на которых формируется описание данных в алгоритмических языках: массив и структура. *Массив* — совокупность однотипных элементов, *структура* — совокупность элементов разного типа. Элементы массива идентифицируются индексом, элементы структуры идентифицируются именами. В СУБД также используются два аналогичных понятия, причем понятие структуры (кортежа, сегмента и т. п.) точно совпадает с соответствующими понятиями языков программирования, а массив (отношение) объединяет одноименные структуры, которые могут идентифицироваться ключом — частью структуры. Таким образом, происходит определенное сближение понятий массива в ЯП и СУБД: перечислимый индекс языка АДА представляется некоторым шагом в сторону ключевых массивов. Ясно, что языкам программирования понятие ключевого массива столь же необходимо, как и СУБД.

Рассмотрим структуру данных рис. 1.1 (на рисунке прямоугольником обозначаются массивы, полукругом — структуры, треуголь-

ником — ссылки, кругом — простые данные). В этой модели встречаются одинаковые по структуре данные, описывающие различные даты. Возможны два способа действия:

1) образовать массив элементов соответствующей структуры и записать в нее все даты, а в схеме БД подчинить элементы этого массива разным исходным вершинам;

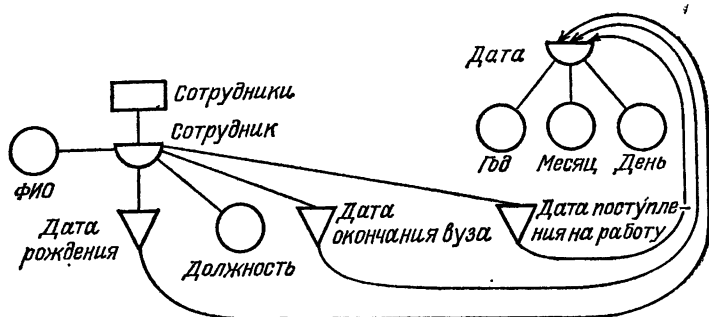


Рис. 1.1

2) описать отдельно структуру повторяющихся элементов, а в схеме использовать это описание в качестве «сложного шаблона».

Первый способ приводит к сетевой организации данных. При этом каждый элемент данных может, вообще говоря, быть подчинен многим элементам одного и того же или разных типов. Такой подход взят за основу и последовательно проведен в концепции КОДАСИЛ. Соответствующие конструкции в ЯП (паскаль, ада) и СУБД (ИНЕС) реализованы в виде ссылок (REFERENCE в ИНЕС, POINTER в ПЛ/1, ада).

Второй способ приводит нас к развиваемым современными ЯП концепциям описываемых (абстрактных) типов данных ([33]). *Типы данных* — это мощное средство образования сложных структур и операций над ними в ЯП. В настоящее время они используются также в рамках различных СУБД. Например, в одной из промышленных систем, построенных в среде ИНЕС, схема с применением шаблонов имеет 100 вершин, если же «развернуть» шаблоны, то — 13 000 вершин. Примером сложной структуры является также дерево рис. 1.2. Описание данных этого дерева не ограничивает фактическую длину ветвей иерархии в структуре самих данных.

В СУБД, как и в ЯП, *шаблоны* являются основой автоматизации. Они позволяют ввести конкретные операции над соответствующими типами данных, которые будут автоматически применяться в разных точках схемы и процедурах работы с БД (загрузка, контроль, образование внешних схем, печать, просмотр и корректировка и др.).

Язык описания данных ИНЕС находится в русле тенденции к сближению основных направлений развития описания данных. С одной стороны, он близок и понятен программистам, так как синтаксически близок к ПЛ/1 и расширяет средства описания данных

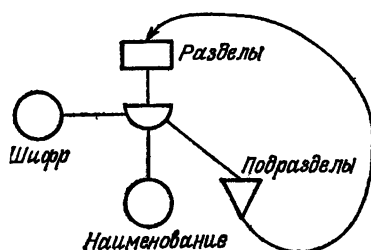


Рис. 1.2

языка ПЛ/1 (за счет таких конструкций, как ключевой массив, шаблон — аналог абстрактных типов данных и т. д.). С другой стороны, он может использоваться как средство концептуально-логического описания предметной области с непосредственной трансляцией последнего во внутреннее машинное представление логической модели данных БД. И,

наконец, основные функции языка аналогичны функциям языков описания данных других СУБД. В частности, данные, описанные средствами ЯОД, могут сохраняться в базе данных для использования во многих программах.

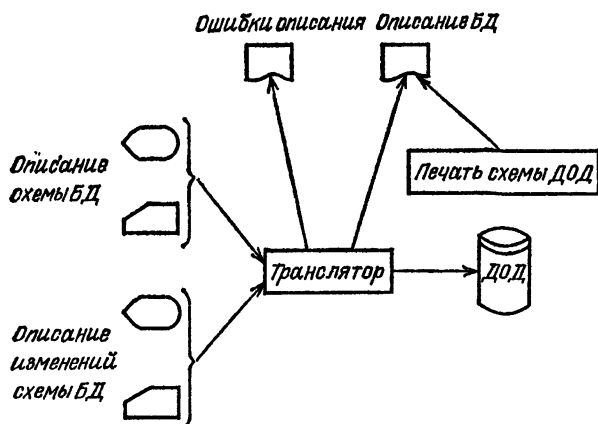


Рис. 1.3

Следует отметить, что перечисленные типы данных: структура, массив, ссылка, простые данные — являются основными, принятыми в ЯОД ИНЕС. Описания структур данных произвольного вида строятся как комбинации этих типов. В пользу выбора именно этих типов данных говорят тенденции развития описания данных в ЯП и других областях моделирования данных. Универсальность средств описания данных ИНЕС подтверждается также обширным

практическим опытом внедрения баз данных в самых разнообразных информационных системах (см. Заключение).

1.0.2. Средства описания данных ИНЕС. Ниже во Введении к каждой главе приводится схема функционирования соответствующей подсистемы. На схемах в виде блоков выделяются основные функции подсистем. Каждая функция реализуется процедурой ИНЕС, либо подпрограммой или функцией в программе пользователя. Для изображения машинных носителей, исходных текстов и наборов данных (перфокарты, диски, листинги и дисплеи) используются стандартные обозначения.

На рис. 1.3 представлена схема обработки описаний данных транслятором ИНЕС. На вход транслятора (см. п. 1.7) подается текст описания данных (см. пп. 1.2—1.6), представленный на перфокартах или в разделе библиотеки исходных текстов. После обработки диагностика и текст описания распечатываются. При отсутствии ошибок в результате работы транслятора на диске создается *дерево описания данных* (ДОД) в виде набора данных ОС. Этот набор данных постоянно используется в дальнейшем при загрузке данных в базу системой ввода и при доступе к ним средствами системы запросов. Для корректировки существующей базы текст изменений, а также набор данных ДОД подаются на вход транслятора (см. п. 1.8). Дальнейшая корректировка ДОД ведется по той же схеме: диагностика и текст нового описания распечатываются, новый ДОД записывается на диск. Использование модифицированного дерева описания данных не требует перезагрузки всей базы.

Блок печати позволяет получить на АЦПУ полный текст ДОД, соответствующий указанному в обращении набору данных.

1.1. Общая характеристика модели данных ИНЕС

Разработка описания данных является одним из наиболее ответственных этапов разработки БД и информационной системы в целом. С одной стороны, это описание должно отражать содержание предметной области, обеспечивая гибкость системы — адаптируемость ее к неизбежным изменениям функциональных требований. С другой стороны, описание определяет основные возможности доступа к данным БД и влияет на эффективность решения конкретных задач обработки данных. Выбор варианта описания связан также с технологией обработки данных, организационными факторами и т. д.

Описание данных задается в рамках модели данных СУВД, т. е. в рамках тех средств, которые обеспечивают правила структуризации данных и основные операции над ними (см. [35, 56]).

Описание данных задает класс реальных данных, имеющих соответствующую описанию структуру (так, описание анкеты задает

совокупность реальных анкет). Для наглядности описание данных может быть представлено в виде графа, вершинам которого сопоставляются данные, дугам — взаимосвязи данных (см. пример на рис. 1.4). Сами данные и их связи также могут быть представлены в виде графов.

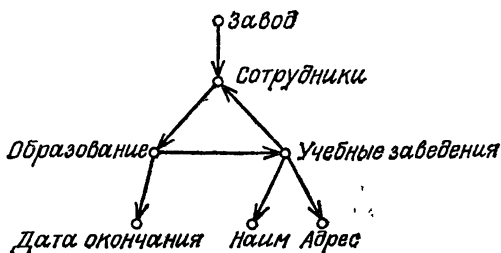


Рис. 1.4

В рамках ИНЕС поддерживается модель данных, фактически не вносящая ограничений в структуры данных. Это означает, что структура данных может быть сетью — ориентированным графом произвольного вида. Количество вершин и дуг сети при этом не ограничивается.

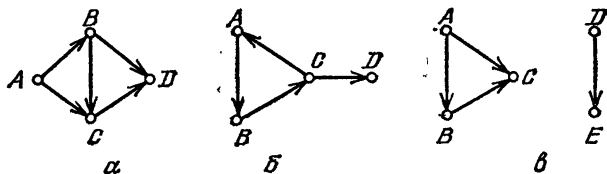


Рис. 1.5

Понятие сети строго вводится в теории графов. Сеть представляет собой совокупность вершин, соединенных попарно ориентированными дугами (стрелками). Примеры сетей приводятся на рис. 1.5. Можно сказать, что две вершины графа соединены *путем*, если из одной вершины можно попасть в другую, двигаясь последовательно по дугам в направлении стрелок (рис. 1.5а). Замкнутый путь, соединяющий вершину с самой собой, называется *циклом* (см. рис. 1.5б). Сеть называется *несвязной*, если она содержит *подграфы*, не соединяющиеся никаким путем (см. рис. 1.5в).

В общем случае модели данных ИНЕС могут содержать произвольное число циклов, а также могут быть представлены несвязными сетями.

Понятие пути играет важную роль в дальнейшем при использовании БД. Путь или траектория задается цепочкой вершин и позволяет однозначно идентифицировать конечное данное цепочки

в иерархических структурах (задается от корня дерева). Траекторией в графе данных задается также *последовательность доступа к данным* как на логическом уровне для пользователя, так и на уровне физического представления данных в базе. В системе ввода при помощи траекторий задаются *места расположения в базе вводимых данных* (см. гл. 3), в системе запросов траектория определяет доступ к данным (см. гл. 4).

Для представления сетевых структур данных в ИНЕС эти структуры должны быть преобразованы в *совокупность иерархий*, перевязанных *ссылками*. Иерархическая структура (дерево) — это

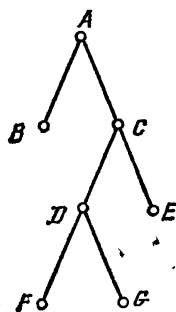


Рис. 1.6

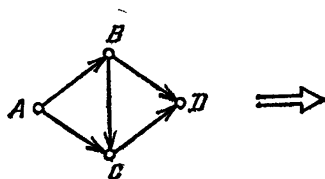


Рис. 1.7

сеть частного вида, в которой каждая вершина имеет только одну входящую дугу. При этом единственная вершина не имеет ни одной входящей дуги и называется *корнем дерева*. Вершины, не имеющие исходящих дуг, называются *терминальными*. (При изображении дерева корень принято рисовать наверху, а само дерево — «растущим» вниз, причем стрелки на дугах, идущие вниз, можно не рисовать — см. пример, дерева на рис. 1.6). Совокупность несвязанных деревьев называется *лесом*.

Сеть приводится к совокупности иерархий путем «расщепления» вершин, имеющих более одной входящей дуги. На рис. 1.7 приводится пример такого преобразования для графа рис. 1.5а. В каждую отщепившуюся вершину помещается *указатель* на исходную вершину, что позволяет отождествлять их и тем самым полностью восстанавливать сеть. Указатели такого типа реализуются в ИНЕС ссылками REF (см. п. 1.4).

Итак, средства описания данных ИНЕС отражают как иерархический (древовидный), так и сетевой принцип организации данных. Результат трансляции *описания данных* в ИНЕС принято называть *деревом описания данных* (ДОД) (см. п. 1.0.2) в соответствии с первым принципом, хотя оно определяет и структуры общего вида. Собственно данные, составляющие базу и организованные в соответствии

с ДОД, называются *деревом данных* (ДД). Следует отметить, что выделение ДОД позволяет хранить имена и характеристики типов данных в единственном экземпляре. Полностью база данных определяется совокупностью ДОД и ДД.

В описании данных вершинам графа должны быть приписаны *типы данных*. Ниже описываются типы данных ИНЕС.

1.2. Простые данные

В графическом изображении ДОД терминальным вершинам соответствуют *простые данные* — элементарные неделимые единицы информации. Простое данное определяется *именем* и *типом* (совкупностью возможных значений) и представляет собой фиксированную характеристику объекта предметной области. Простые данные составляют собственно *содержание базы данных* и в то же время являются *минимальными элементами информации*, которые передаются от пользователя в базу и обратно. В описании данных фиксируются имя и тип простого данного. К основным типам относятся:

а) *Целый тип* (INT). Значения — целые числа со знаком. Количество цифр в числе не должно превосходить 9, так как под данное в базе отводится не более 4 байтов.

б) *Действительный тип* (REAL). Значения — действительные числа со знаком. Количество значащих цифр числа не должно превосходить 16, так как под двоичное представление числа в базе отводится до 8 байтов, включая представление знака, порядка и мантиссы.

в) *Десятичный тип* (DEC). Значения — действительные числа. Этот тип отличается от чисел типа REAL внутренним машинным представлением. Сложение, вычитание и умножение целых чисел типа DEC производятся без перевода в двоичное представление, с абсолютной точностью, что особенно важно при проведении бухгалтерских расчетов. Количество значащих цифр 31, так как в БД отводится до 16 байтов.

г) *Текстовый тип* (TEXT). Значения — тексты длиной не более 250 символов (о представлении в базе текстов большей длины см. п. 2.7);

д) *Тип русский текст* (RTEXT). Значения — тексты длиной не более 250 символов. Отличия этого типа от типа TEXT проявляются при использовании данного в качестве ключа массива, т. е. при автоматическом упорядочении данных в базе. Данные типа TEXT располагаются в базе в соответствии с латинским алфавитом, данные типа RTEXT — в соответствии с русским алфавитом.

е) *Логический тип* (LOG). Значения — любые последовательности двоичных символов длиной не более 250 байтов (битовая строка).

В описании данных, кроме имени и типа простого данного, может быть указана максимально допустимая длина его значения с помощью спецификации /SIZE = k/, а также формат стандартной печати данного с помощью спецификации /PFORM = п.м/ (см. п. 1.6). Реально в ДД простые данные хранятся в укороченном виде (урежаются пробелы в конце текстов, ведущие нули целых чисел и нули в конце логических данных). Следует отметить, что место под значение простого данного отводится в момент ввода этого данного в базу, отсутствующие данные местá в базе не занимают. Исключение здесь составляет использование простых данных в составе жестких структур (см. пп. 1.3, 2.7).

Простые данные описываются в ЯОД по схеме:

имя данного: тип [спецификации] ['значение']; причем тип задается соответствующим ключевым словом. Например, ФИО: RTEXT; ГОД РОЖДЕНИЯ: INT; ЗАРПЛАТА: REAL. «Значение» присваивается в ДОД константам.

Имена данных в ДОД составляются из букв, цифр и пробелов, они могут иметь длину до 29 символов. Первый символ — всегда буква, несколько пробелов подряд внутри имени интерпретируются как один пробел. В качестве имени ДОД может, таким образом, использоваться целая фраза, отражающая содержательный смысл данного. Следует, однако, заметить, что впоследствии длинные имена данных составляют дополнительную нагрузку в прикладном программировании.

В графе описания данных простые данные изображаются терминальными вершинами в виде точек или кружочков. Тип обозначается первой буквой соответствующего ключевого слова: I — целое, R — действительное, D — десятичное, T — текст, RT — русский текст.

1.3. Составные (структурные) данные

Нетерминальным вершинам ДОД соответствуют составные данные. *Составные (структурные)* данные представляют собой объединенную под общим именем *совокупность данных*, которые в свою очередь могут иметь различные типы, в том числе структурные (разрешается любая глубина вложенности структурных данных). Так, на рис. 1.6 данное А составлено из данных В и С, или из данных В, D и Е, или, что то же самое, из данных В, F, G и Е.

В дальнейшем для наглядности в текстовое представление структурных данных введено понятие уровня расположения вершин. *Уровнем вершины V* в иерархическом графе называется *число вершин* в пути, соединяющем ее с корнем дерева. Так, в графе рис. 1.6 вершина А имеет уровень 1, вершины В и С — уровень 2, вершины D и Е — уровень 3, вершины F и G — уровень 4. При изображении

нерархических структур вершины одного уровня, как правило, располагаются на одной горизонтали. Будем называть подчиненной ветвью вершины V поддевею, подчиненное этой вершине.

В дальнейшем для представления структурных данных ДОД будет использоваться уровневая запись: под каждой вершиной уровня n выписываются подчиненные ей ветви, начинающиеся с уровня $n+1$. Номер каждого уровня указывается явно, вершины одного уровня располагаются на одной вертикали. Очевидно, конец каждой ветви совпадает с повторным упоминанием уровня $n+1$, либо уровня с меньшим номером, либо с концом всего текста.

Заметим, что уровневая запись используется в дальнейшем для описания движений по дереву данных при записи данных в базу и при считывании их средствами запроса. В записи запроса одновременно учитывается иерархическая последовательность выполнения процедур обработки данных.

Структурные данные подразделяются в описании ДОД на типы: *структура, жесткая структура, массив*.

1) *Тип структура (STRUCT)*. Данное типа структура представляет собой совокупность различных данных, перечисленных в описании вместе с их именами и типами.

Структура описывается в ЯОД по схеме:

```
n имя: STRUCT
n+1 описание данного 1
n+1 описание данного 2
.....
```

Графически структура представляется полукруглой вершиной, которой подчинены изображения составляющих ее данных.

Пример структуры — данное АНКЕТА, составленное из данных ТАБНОМЕР (INT), ФИО (TEXT), ЗАРПЛАТА (REAL), ДАТА РОЖДЕНИЯ (STRUCT), причем в дату рождения входят данные ГОД (INT), МЕСЯЦ (INT), ЧИСЛО (INT). Графически это данное изображено на рис. 1.8.

Описание данного АНКЕТА на ЯОД имеет вид:

```
01 АНКЕТА: STRUCT
02 ТАБНОМЕР: INT
02 ФИО: TEXT/SIZE=40/
02 ЗАРПЛАТА: REAL
02 ДАТА РОЖДЕНИЯ: STRUCT
03 ГОД: INT; МЕСЯЦ: INT; ЧИСЛО: INT
```

Здесь задано ограничение на длину данного ФИО — 40 байтов.

В ИНЕС описание данных может быть составлено также в скобочном виде. В этом случае данные одного уровня разделяются зна-

ком «;» и заключаются в скобки. Описание структуры рис. 1.8 в скобочной записи имеет вид:

```
АНКЕТА: STRUCT(
ТАБНОМ: INT; ФИО: TEXT; ЗАРПЛАТА: REAL;
ДАТА РОЖДЕНИЯ: STRUCT(ГОД: INT; МЕСЯЦ: INT;
ЧИСЛО: INT))
```

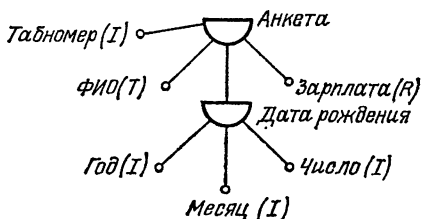


Рис. 1.8

Заметим, что описание подчиненной структуры может выноситься на первый уровень. Например:

```
01 АНКЕТА: STRUCT
02 ТАБНОМЕР: INT
02 ФИО: TEXT
02 ДАТА РОЖДЕНИЯ
01 ДАТА РОЖДЕНИЯ: STRUCT
02 ГОД: INT
02 МЕСЯЦ: INT
02 ЧИСЛО: INT
```

или:

```
АНКЕТА: STRUCT(ТАБНОМЕР: INT; ФИО: TEXT;
ЗАРПЛАТА: REAL; ДАТА РОЖДЕНИЯ);
ДАТА РОЖДЕНИЯ: STRUCT(
ГОД: INT; МЕСЯЦ: INT; ЧИСЛО: INT)
```

В дальнейшем данные описываются только в уровневом виде *).

2) Тип жесткая структура (FSTRUCT). Жесткая структура состоит из простых данных и жестких структур и хранится в базе как одно данное фиксированного формата. Для каждого простого данного, входящего в жесткую структуру, в ДОД указывается его длина. Возникновение значения любого элемента в жесткой струк-

*) Уровневая запись введена и преимущественно используется в ИНЕС с 1983 г. При этом списочная запись также полностью сохранена. В курсах обучения рекомендуется использовать только уровневую запись.

туре приводит к созданию в базе всей структуры целиком, т. е. отсутствующие данные занимают место. Все структуры, подчиненные жесткой, считаются жесткими (ее длина ≤ 250 байт).

Использование жестких структур позволяет увеличить скорость считывания группы данных: вся структура может быть считана из базы целиком, за одно обращение к системе доступа. В то же время сохраняется возможность индивидуальной работы с данными.

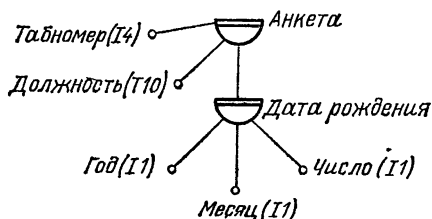


Рис. 1.9

Описание жесткой структуры на ЯОД аналогично описанию простой структуры, но с указанием типа FSTRUCT и обязательно с указанием длин простых данных.

Графически жесткая структура может изображаться аналогично простой, но с перечеркнутым полукругом или с указанием типа FSTRUCT рядом с наименованием. Описание структуры рис. 1.9 на ЯОД имеет вид:

```

01 АНКЕТА: FSTRUCT
02 ТАБНОМЕР: INT/SIZE=4/
02 ДОЛЖНОСТЬ: TEXT/SIZE=10/
02 ДАТА РОЖДЕНИЯ: STRUCT
03 ГОД: INT/SIZE=1/
03 МЕСЯЦ: INT/SIZE=1/
03 ЧИСЛО: INT/SIZE=1/
  
```

Здесь под ТАБНОМЕР отводится 4 байта — максимальный размер данного типа INT (соответствует 9-значному числу). Под данные ГОД, МЕСЯЦ, ЧИСЛО отводится по 1 байту. Заметим, что на рис. 1.9 рядом с типом указывается размер данного в байтах.

При использовании жестких структур как одного целого следует иметь в виду разницу в ограничениях на форматы пользователя по сравнению с остальными типами данных. В общем случае пользователь может задать в программе тип и формат отдельного данного, не совпадающий с указанными в ДОД. Система автоматически преобразует полученный от пользователя или передаваемый ему формат в формат ДОД. При работе с целыми жесткими структурами программа пользователя должна записывать или считывать всю

структуру в соответствии с порядком, типами и длинами простых данных, указанными в ДОД.

3) *Данное типа массив (ARRAY)* представляет собой набор однородных данных — элементов массива, причем число этих элементов в описании данных не задается и заранее не ограничивается. Элементы массива отличаются между собой только значениями, например, — отдельные анкеты в массиве анкет. Так как описания элементов одинаковы, описывается в ДОД и изображается графически только один элемент. Вершина типа массив изображается в графе

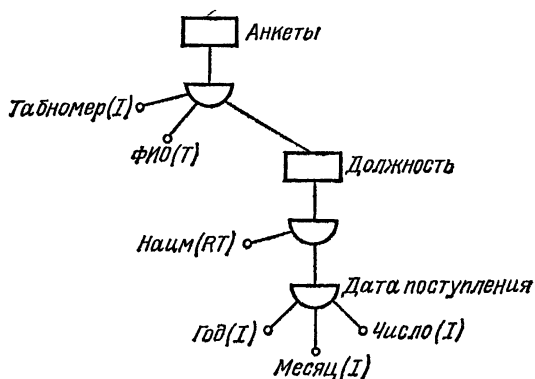


Рис. 1.10

описания данных прямоугольником, этой вершине всегда подчинен элемент массива — простое данное или структура. Допускаются вложенные массивы практически любого уровня вложенности. Пример двухуровневого массива приводится на рис. 1.10; элемент массива анкет характеризуется списком занимаемых ранее должностей.

В ИНЕС используются три типа массивов: *ключевой массив*, *нумерованный массив* и *простой массив*.

а) *Ключевой массив* является наиболее употребительным в конструкциях ИНЕС. Элементом массива является структура, в которой одно из простых данных объявлено ключом. Значение ключа задается при вводе данных и полностью идентифицирует элемент массива. По значению ключа система обеспечивает прямой доступ к элементу массива.

Для организации эффективного доступа к данным имеет значение порядок расположения элементов массива в базе. Этот порядок определяется отношением порядка на значениях ключей: обычный порядок «больше-меньше» для чисел и логических значений, лексикографический порядок для текстов и русских текстов (соответственно по латинскому и по русскому алфавиту).

Все типы массивов изображаются в виде прямоугольных вершин. В графическом изображении ключевого массива вершина подчеркивается горизонтальной чертой, наименование ключа или обозначение типа внутри вершины подчеркивается. Пример изображения ключевого массива приводится на рис. 1.11.

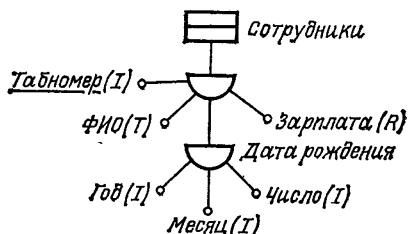


Рис. 1.11

Ключевой массив описывается в ЯОД по схеме:

п имя массива: ARRAY
 п+1 [имя элемента массива:] STRUCT/KEY=имя ключа/
 п+2 имя данного1: описание данного1
 п+2 имя данного2: описание данного2
 ...

Пример описания ключевого массива рис. 1.11 на ЯОД:

```
01 СОТРУДНИКИ: ARRAY
02 STRUCT/KEY=ТАБНОМЕР/
03 ТАБНОМЕР: INT
03 ФИО: TEXT
03 ЗАРПЛАТА: REAL
03 ДАТА РОЖДЕНИЯ: STRUCT
04 ГОД: INT; МЕСЯЦ: INT; ЧИСЛО: INT
```

Для описания ключевого массива вводится также специальный тип данного ARK:

п имя массива: ARK
 п+1 имя данного: простой тип/KEY/
 п+1 имя данного1: описание данного1
 п+1 имя данного2: описание данного2
 ...

Здесь, начиная с уровня п+1, описывается состав структуры элемента массива. Одно из простых данных должно иметь спецификацию /KEY/, определяющую ключ. Имя самого элемента не называется. Описание массива рис. 1.11 в этом варианте выглядит следующим образом:

01 СОТРУДНИКИ: ARK
 02 ТАБНОМЕР: INT/KEY/
 02 ФИО: TEXT
 02 ЗАРПЛАТА: REAL
 02 ДАТА РОЖДЕНИЯ: STRUCT
 03 ГОД: INT; МЕСЯЦ: INT; ЧИСЛО: INT

б) *Нумерованный массив* заводится в случае, если нет возможности задать ключ элемента массива, т. е. нет данного, по которому можно естественно упорядочить и однозначно проидентифицировать элементы массива. Элементы нумерованного массива при вводе в

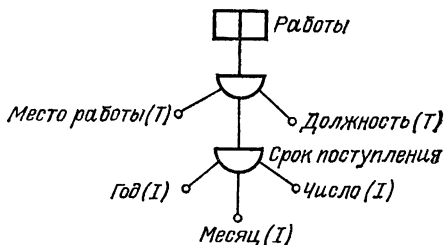


Рис. 1.12

базу автоматически нумеруются последовательными целыми числами, которые играют роль их идентификаторов. В том же порядке элементы располагаются в базе. Система обеспечивает быстрый поиск элемента по его номеру.

В графе описания данных нумерованный массив выделяется надписью NUM, либо вертикальной чертой внутри прямоугольника. Например, графическое изображение массива, представляющего список мест работы сотрудника и смену его должностей, имеет вид рис. 1.12.

В терминах ЯОД описание нумерованного массива имеет вид:

n имя массива: ARRAY/NUM=YES/
 n+1 описание элемента массива

Пример описания нумерованного массива рис. 1.12:

01 РАБОТЫ: ARRAY/NUM=YES/
 02 РАБОТА: STRUCT
 03 МЕСТО РАБОТЫ: TEXT
 03 ДОЛЖНОСТЬ: TEXT
 03 СРОК ПОСТУПЛЕНИЯ: STRUCT
 04 ГОД: INT; МЕСЯЦ: INT; ЧИСЛО: INT

При вводе данных можно задать шаг нумерации элементов нумерованного массива.

в) В ИНЕС существует также возможность организации *простого массива* — аналога последовательного файла. Элементы простого массива создаются в базе в порядке их ввода. Поиск элемента осуществляется путем последовательного перебора.

В терминах ЯОД простой массив описывается по схеме:

п имя массива: ARRAY

n+1 описание элемента массива

Отметим основные особенности присвоения имен в структурах ДОД. В общем случае для различных вершин ДОД могут использоваться одинаковые имена.

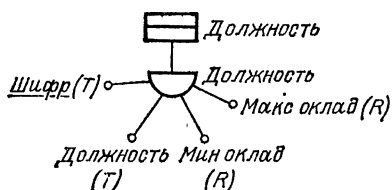


Рис. 1.13

Единственное ограничение здесь состоит в том, что одноименными не могут быть корневые вершины и вершины, подчиненные одной исходной. В то же время допускаются структуры типа рис. 1.13. Одинаковые имена в ДОД автоматически

возникают при использовании ссылок на шаблон (см. п. 1.4). Для однозначной идентификации вершин в ДОД используются составные имена — цепочки имен данных, образующих путь к вершине из корня. Так, терминальные вершины рис. 1.13 идентифицируются именами: ДОЛЖНОСТЬ.ДОЛЖНОСТЬ.ДОЛЖНОСТЬ.ДОЛЖНОСТЬ.ДОЛЖНОСТЬ.МИН ОКЛАД и т. д. Следует отметить, что, как правило, возникает необходимость проидентифицировать конкретное данное, т. е. вершину ДД, соответствующую той или иной вершине ДОД. В этом случае должен быть указан конкретный элемент массива, например: ДОЛЖНОСТЬ.##'A023'. ДОЛЖНОСТЬ. Различные способы представления траекторий ДД подробно описываются ниже, в гл. 3, 4.

Еще одна особенность описания данных состоит в том, что не все вершины ДОД должны иметь имена. Так, имя элемента ключевого массива (вершина типа STRUCT) не входит в траекторию ДД и практически нигде в прикладных программах не используется. В принципе и любая другая вершина может не иметь имени.

Всем вершинам система автоматически присваивает имена вида ##целое (например, ##15; эти имена распечатываются в тексте трансляции ДОД).

1.4. Ссылочные данные

Ссылочные данные или ссылки имеют различную природу и назначение. Общим у них является то, что значения таких данных представляют собой указатели на другие данные. В ИНЕС исполь-

вуются следующие типы ссылочных данных: *ссылка на значение* — для установления сетевых связей данных; *ссылка на шаблон* — для сокращения описания данных; *ссылка на словарное данное* — для сжатия базы.

В графе описания данных ссылочным данным соответствуют терминальные вершины в виде треугольников. Дуга со стрелкой соединяет эту вершину с вершиной-адресатом. Тип ссылки указывается в треугольнике первой буквой соответствующего ключевого слова.

1) *Ссылка на значение (REF)*. Данное этого типа в качестве своего значения содержит указатель на другое данное. При этом на уровне ДОД задается ссылка только на описание данного-адресата, например, — на описание элемента массива. Значение ссылки



Рис. 1.14

устанавливается при вводе данных в базу и хранится как аналог физического адреса данного-адресата. Так, если в ДОД указывалась ссылка на описание элемента массива, то при вводе данных устанавливаются связи с конкретными элементами массива. При доступе к данным наличие ссылки эквивалентно отождествлению данного-ссылки и данного-адресата.

В описании данных ссылка задается составным именем вершины-адресата. Например (см. рис. 1.14):

```

01 ШТАТНОЕ РАСПИСАНИЕ: ARRAY
02 ДОЛЖНОСТЬ: STRUCT/KBY=ШИФР/
03 ШИФР: TEXT
03 НАИМ: TEXT
03 МАКС ОКЛАД: REAL
03 МИН ОКЛАД: REAL
01 АНКЕТЫ: ARK
02 ТАБНОМЕР: INT/KBY/
02 ФИО: TEXT
02 ДОЛЖНОСТЬ: REF'ШТАТНОЕ РАСПИСАНИЕ.
    ДОЛЖНОСТЬ'
    
```

2) *Ссылка на шаблон (AS)*. Ссылка на шаблон означает, что данное со ссылкой описывается так же, как данное-адресат, хотя значе-

ния их различны. Ссылка на шаблон задается в описании данных составным именем вершины-адресата. Например (см. рис. 1.15):

```
01 АНКЕТЫ: ARK
02 ФИО: RTEXT/KEY/
02 ДАТА РОЖДЕНИЯ: STRUCT
03 ГОД: INT; МЕСЯЦ: INT; ЧИСЛО: INT
01 ОБРАЗОВАНИЕ: ARK
02 НАИМ: RTEXT /KEY/
02 СПЕЦИАЛЬНОСТЬ: TEXT
02 ДАТА ОКОНЧАНИЯ: AS'АНКЕТЫ..ДАТА
РОЖДЕНИЯ'
```

Две точки в описании ссылки AS соответствуют отсутствующему имени элемента массива.

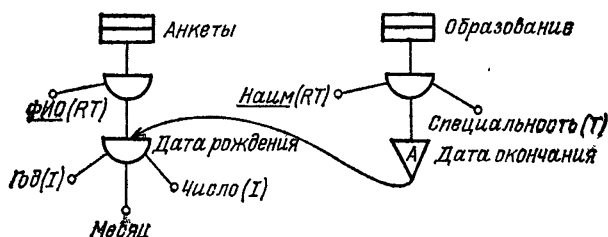


Рис. 1.15

Заметим, что если в описании данных ссылка на шаблон подчинена вершине-адресату, то это описание представляет потенциально бесконечную структуру (см. п. 2.5). Фактическая глубина таких структур в базе определяется глубиной иерархии вводимых данных.

3) Данные типа ссылки на словарь или словарные данные обеспечивают возможность в терминальной вершине дерева данных сослаться на значение, погруженное в словарь (словарь или словарная база данных хранится отдельно от основной БД). В основной базе словарные данные представлены сокращенными кодами своих значений [9].

Словарное данное может быть одного из трех типов: идентификатор, код, русский код. Для этих типов данных связь ВД со словарем осуществляется автоматически, так что для пользователя словарные данные не отличаются от обычных текстов. Набор данных, содержащий словарь, фиксируется при этом в тексте ДОД с помощью спецификаций: VN=имя словаря, DDN=DD-имя набора данных, DSN=DSN-имя набора данных (см. п. 1.6).

а) Применение словарных данных типа идентификатор (VOC) предусматривает автоматическое кодирование значений вводимых

данных. При вводе очередного значения в основную базу помещается его сокращенный шифр-идентификатор. Если это значение вводится впервые, то шифр генерируется системой и помещается также в словарь вместе с полным значением данного (таким образом, словарь заполняется по мере загрузки базы).

При считывании данного пользователь автоматически получает полное значение, присутствие словарной системы сказывается только в упорядоченности элементов ключевого массива: ключ типа VOC задает порядок элементов в соответствии со значениями шифров, но не данных. Словарные данные типа идентификатор описываются в ЯОД по схеме:

имя данного: VOC

б) При использовании словарных данных типа *код* (CODE) и *русский код* (RCODE) предполагается, что пользователем заранее подготовлен словарь, в котором содержатся возможные значения данного вместе с шифрами этих значений и, быть может, дополнениями-синонимами. При вводе слова, шифра или синонима в базу всегда будет помещен шифр, при чтении будет выдаваться шифр или основное слово, в зависимости от заданного режима. Отличие русского кода состоит в том, что в качестве типа ключа он обеспечивает русскую лексикографическую упорядоченность элементов массива.

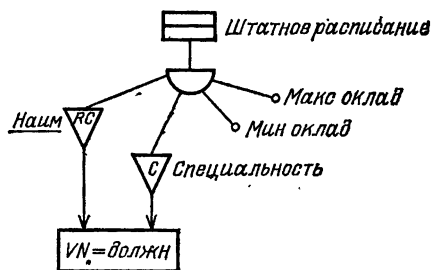


Рис. 1.16

Графически словарь может изображаться, например, в виде прямоугольника с именем словаря, на который указывают стрелки из соответствующих словарных вершин. Изображение словаря можно опустить.

Например, если в массиве ШТАТНОЕ РАСПИСАНИЕ ключ имеет тип RCODE (см. рис. 1.16), а соответствующий словарь задан в виде:

01/ДИРЕКТОР *

02/ЗАМЕСТИТЕЛЬ ДИРЕКТОРА/ЗАМ.ДИРЕКТОРА *





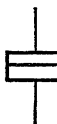
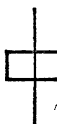
...





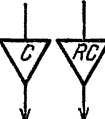
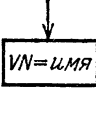

03/СТАРШИЙ ИНЖЕНЕР*

04/ИНЖЕНЕР*

то элементы массива ШТАТНОЕ РАСПИСАНИЕ в БД будут упорядочены в лексикографическом порядке кодов

Графическое изображение вершин в дереве описания данных

					
<i>Простое данные</i>	<i>Ключ массива</i>	<i>Структура</i>	<i>Жесткая структура</i>	<i>Ключевой массив</i>	<i>Нумерован- ный массив</i>

						
<i>Простой массив</i>	<i>Ссылка на шаблон</i>	<i>Ссылка на значение</i>	<i>Ссылка VDC</i>	<i>Ссылки CODE, RCODE</i>	<i>Словарь</i>	<i>Корневая вершина</i>

Обозначение типов данных в терминальных вершинах







<i>Обозначение</i>	<i>Тип данных</i>	<i>Количество байт</i>
	<i>Целое число</i>	<i>от 1 до 4</i>
	<i>Вещественное число</i>	<i>от 2 до 8</i>
	<i>Десятичное число</i>	<i>от 1 до 15</i>
	<i>Логическое данные</i>	<i>от 1 до 250</i>
	<i>Текст</i>	<i>от 1 до 250</i>
	<i>Русский текст</i>	<i>от 1 до 250</i>

Рис. 1.17

В терминах ЯОД вершина типа CODE (RCODE) описывается по схеме:

имя: CODE

При наличии в ДОД нескольких вершин типа CODE или RCODE соответствующие словари должны быть объединены (см. рис. 1.16). Неоднозначность в идентификации слов, возникающую из-за дублирования шифров, можно при этом предотвратить с помощью специального механизма префиксов. Префикс, состоящий из одной буквы, сопоставляется словарю через описание соответствующей вершины ДОД по схеме:

имя: CODE [/префикс/]

Например: СПЕЦИАЛЬНОСТЬ: CODE/C/

Префикс, указанный в ДОД, автоматически приписывается системой спереди к кодам словаря в словарной базе и должен указываться при обращении к данному основной базы. В базе данных код хранится без префикса.

Помимо автоматической связи со словарем, средства словарной системы обеспечивают непосредственный доступ к словарным записям. Этот доступ осуществляется из программ, написанных на различных языках программирования, в том числе из запросов ИНЕС (см. п. 4.8).

Список введенных выше графических обозначений типов данных приводится в таблице рис. 1.17. Эти обозначения могут использоваться с модификациями. Так, в приведенных выше рисунках терминальная вершина обозначалась точкой, корневая вершина не выделялась.

1.5. Пример описания данных

Приведем пример описания данных для дерева, представляющего анкетную информацию о сотрудниках некоторого завода (см. рис. 1.18). На примере этого дерева будут демонстрироваться возможности ИНЕС и в последующих главах. Заметим, что в реальности может быть представлен более гибкий и технологичный вариант этой структуры. Описание базы данных рис. 1.18 имеет вид:

```
01 &VOC/VN=НДС, DDN=NDC/
01 ЗАВОД: ARRAY
02 ЦЕХ: STRUCT/KEY=НАИМЕНОВАНИЕ/
03 НАИМЕНОВАНИЕ: RTEXT
03 ЧИСЛО МУЖЧИН: INT
03 ЧИСЛО ЖЕНЩИН: INT
03 ФОНД ЗАРАБОТНОЙ ПЛАТЫ: REAL
03 СОТРУДНИКИ: ARRAY
```

04 STRUCT/KEY=ФИО/
 05 ФИО: RTEXT
 05 ПОЛ: TEXT
 05 ДАТА РОЖДЕНИЯ: AS'ДАТА'
 05 ОБРАЗОВАНИЕ: STRUCT
 06 НЕВЫСШЕЕ: STRUCT
 07 ДАТА ОКОНЧАНИЯ: AS'ДАТА'

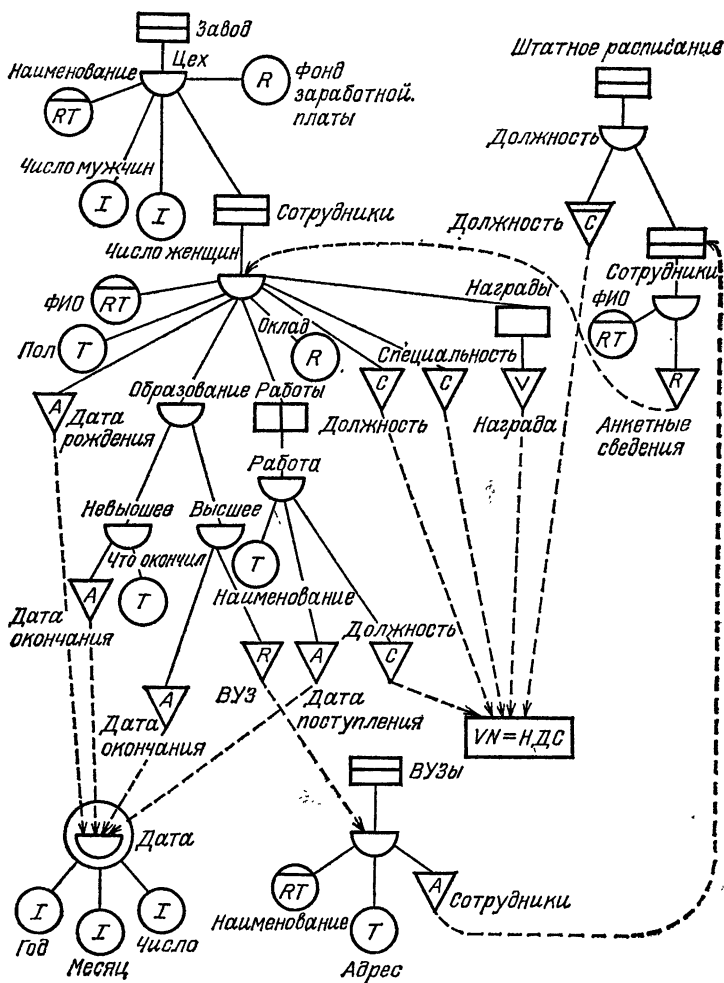


Рис. 1.18

07 ЧТО ОКОНЧИЛ: ТЕХТ
 06 ВЫСШЕЕ: STRUCT
 07 ДАТА ОКОНЧАНИЯ: AS'ДАТА'
 07 ВУЗ: REF'ВУЗЫ.'
 05 РАБОТЫ: ARRAY/NUM=YES/
 06 РАБОТА: STRUCT
 07 НАИМЕНОВАНИЕ: ТЕХТ
 07 ДАТА ПОСТУПЛЕНИЯ: AS'ДАТА'
 07 ДОЛЖНОСТЬ: CODE/Д/
 05 ОКЛАД: REAL
 05 ДОЛЖНОСТЬ: CODE/Д/
 05 СПЕЦИАЛЬНОСТЬ: CODE/С/
 05 НАГРАДЫ: ARRAY
 06 НАГРАДА: VOC
 01 ШТАТНОЕ РАСПИСАНИЕ: ARRAY
 02 ДОЛЖНОСТЬ: STRUCT/KEY=ДОЛЖНОСТЬ/
 03 ДОЛЖНОСТЬ: CODE/Д/
 03 СОТРУДНИКИ: ARRAY
 04 STRUCT/KEY=ФИО/
 05 ФИО: RTEXT
 05 АНКЕТНЫЕ СВЕДЕНИЯ: REF'ЗАВОД.ЦЕХ.
 СОТРУДНИКИ.'
 01 ВУЗЫ: ARRAY
 02 STRUCT/KEY=НАИМЕНОВАНИЕ/
 03 НАИМЕНОВАНИЕ: RTEXT
 03 АДРЕС: ТЕХТ
 03 СОТРУДНИКИ: AS'ШТАТНОЕ РАСПИСАНИЕ.
 ДОЛЖНОСТЬ.СОТРУДНИКИ'
 01 ДАТА: STRUCT
 02 ГОД: INT
 02 МЕСЯЦ: INT
 02 ЧИСЛО: INT

В схеме базы данных рис. 1.18 сотрудники распределены по цехам. Цех идентифицируется своим наименованием, сотрудник — простым данным ФИО. В анкете сотрудника представлены словарные данные типа CODE (СПЕЦИАЛЬНОСТЬ и ДОЛЖНОСТЬ) и типа VOC (НАГРАДА). Нумерованный массив РАБОТЫ характеризует список мест работы сотрудника и смену его должностей (при смене только должности заводится новый элемент массива с прежним наименованием работы). Все даты в дереве представлены по шаблону ДАТА.

Список должностей сотрудников завода, а также список законченных ими учебных заведений представлены в ДОД отдельными деревьями. Эти два дерева имеют одинаково устроенные списки

сотрудников, причем соответствующий массив СОТРУДНИКИ описывается только в дереве ШТАТНОЕ РАСПИСАНИЕ. Элементы этого массива содержат ссылки REF на анкеты сотрудников. В дереве ВУЗЫ список сотрудников завода описывается по шаблону. Заметим, что здесь значения ссылок REF отличаются от ссылок в дереве ШТАТНОЕ РАСПИСАНИЕ.СОТРУДНИКИ, играющего в данном случае роль шаблона.

Как упоминалось выше, база данных рис. 1.18 составлялась как учебный пример, позволяющий в простом и наглядном виде продемонстрировать основные возможности ИНЕС. Эта база не может считаться хорошо спроектированной. В главе 2 приводится ряд методических рекомендаций по проектированию баз данных, удовлетворяющих традиционным требованиям интеграции, целостности, производительности и т. д. Исходя из этих рекомендаций, к дереву описания анкет сотрудников рис. 1.18 можно сделать следующие замечания.

1. Неудачно выбран ключ ФИО в массиве СОТРУДНИКИ. Обычно в кадровых системах идентификатором сотрудника является табельный номер.

2. Массив СОТРУДНИКИ, представляющий описание одного из основных объектов проектирования, должен быть отделен от дерева ЗАВОД, описывающего административную структуру подразделений. При этом облегчается, например, корректировка данных: перевод сотрудника осуществляется не переносом поддерева, а только «перевешиванием» ссылки в дереве ЗАВОД.

3. В дереве ЗАВОД можно хранить многие сводные показатели, — например, число сотрудников дефицитных специальностей. По этим показателям организуется оперативное получение сводных и персональных справок по цехам в диалоговом режиме.

4. Дерево ЗАВОД может играть вспомогательную роль инверсного входа, осуществляющего сортировку сотрудников по цехам (другие виды сортировок представлены деревьями ШТАТНОЕ РАСПИСАНИЕ и ВУЗЫ). Как правило, инверсные входы нужны по многим реквизитам, поэтому хорошо завести общую инверсную справочную.

5. Ссылки типа REF на анкеты следует заменить использованием логических ссылок — программных переходов по ключам (см. п. 2.6). Такие переходы на первый уровень ДОД не отличаются по времени от переходов по ссылкам REF. Заметим, что интенсивное использование ссылок типа REF, как правило, оказывается недостатком базы с точки зрения технологии ее ведения.

В базе данных используется словарь, который содержит значения для данных ДОЛЖНОСТЬ, СПЕЦИАЛЬНОСТЬ и НАГРАДА. Первые два списка могут иметь большой объем. В этом случае следует разделить словари и работать с ними в неавтоматическом

режиме. Как правило, независимые словари большого объема не следует автоматически связывать с базой.

6. Нумерованный массив РАБОТЫ можно заменить ключевым, выбрав в качестве ключа дату поступления на работу. Для этого структура ДАТА ПОСТУПЛЕНИЯ должна быть заменена простым данным. Например, дата может представляться текстом вида «ГГ.ММ.ДД».

Очевидно, в процессе работы могут возникнуть и другие критические замечания.

1.6. Синтаксис ЯОД

В настоящем разделе приводится стандартная синтаксическая таблица языка описания данных ИНЕС. Напомним, что в таблице слова ЯОД пишутся заглавными буквами, а синтаксические единицы ЯОД, значения которых должны быть определены — строчными буквами. Символы $::=$, $\{$, $\}$, $[]$ и \dots (многоточие) используются в определенном смысле: символ « $::=$ » означает, что справа от него содержится определение для синтаксической единицы, находящейся слева; в фигурные скобки заключается вертикальный список возможных взаимоисключающих вариантов в определении синтаксической единицы; в квадратные скобки заключаются те части определения синтаксической единицы, которые могут быть опущены в зависимости от ее конкретного смысла (при заключении фигурных скобок в квадратные фигурные опускаются); многоточие идет после квадратных скобок и означает, что содержимое скобок может быть повторено любое число раз (в том числе — ни одного раза).

Синтаксическая таблица ЯОД выглядит следующим образом: описание данных $::=$ строка [строка] ...

$$\text{строка} ::= \left\{ \begin{array}{l} 01 \text{ умолчания} \\ 01 \text{ описание словаря} \\ \text{номер_уровня элемент} \end{array} \right\}$$

умолчания $::=$ DEFAULT ORDER = YES

описание словаря $::=$ &VOC/VN = имя словаря

$$\left\{ \begin{array}{l} \text{DDN} = \text{DD-имя словаря} \\ \text{DSN} = \text{DSN-имя словаря} \end{array} \right\}$$

номер_уровня $::=$ 0n[m]

n $::=$ цифра

m $::=$ цифра

$$\text{элемент} ::= \left\{ \begin{array}{l} [\text{имя}] \text{данное} \\ \text{имя} \end{array} \right\}$$

$$\text{данное} ::= \left\{ \begin{array}{l} \text{простое данное} \\ \text{структурное данное} \\ \text{массив} \\ \text{ссылочное данное} \\ \text{ссылка на словарь} \end{array} \right\}$$

$$\text{простое данное} ::= \left\{ \begin{array}{l} \text{INT} \\ \text{REAL} \\ \text{DEC} \\ \text{TEXT} \\ \text{RTEXT} \\ \text{LOG} \end{array} \right\} [\text{спецификации}][\text{'значение'}]$$

$$\text{структурное данное} ::= \left\{ \begin{array}{l} \text{STRUCT} \\ \text{FSTRUCT} \end{array} \right\} [\text{спецификации}]$$

$$\text{массив} ::= \text{ARRAY}[\text{спецификации}]$$

$$\text{ссылочное данное} ::= \left\{ \begin{array}{l} \text{REF} \\ \text{AS} \end{array} \right\} \text{'составное имя'}$$

$$\text{ссылка на словарь} ::= \left\{ \begin{array}{l} \text{VOC} \\ \text{CODE} \\ \text{RCODE} \end{array} \right\} [\text{спецификации}]$$

$$\text{спецификации} ::= / \text{спецификация} [\text{спецификация}] \dots /$$

$$\text{спецификация} ::= \left\{ \begin{array}{l} \text{NUM} = \text{YES} \\ \text{KEY} = \text{имя} \\ \text{SIZE} = \text{целое} \\ \text{PFORM} = \text{целое}[\text{.целое}] \\ \text{ORDER} = \text{YES} \\ \text{MM} = \text{целое} \\ \text{буква} \end{array} \right\}$$

$$\text{целое} ::= \text{цифра} \dots$$

$$\text{составное имя} ::= \text{имя} [\text{.имя}] \dots$$

$$\text{имя} ::= \text{буква} \left[\begin{array}{l} \text{буква} \\ \text{цифра} \\ \text{пробел} \end{array} \right] \dots$$

Ниже объясняется смысл синтаксических конструкций ЯОД. Описание данных предполагает описание всего ДД, которое в общем случае может состоять из нескольких компонентов (деревьев). Каждый компонент имеет отдельное описание, обозначенное в определении описания данных как элемент. На уровне описания данных могут быть также описаны элементы, соответствующие веткам дерева. Вершины, к которым эти ветки подвешены, представлены в тексте описания только именами. Каждой вершине ДОД со-

ответствует некоторый элемент описания данных. Элементы одного уровня, подчиненные общей вершине, могут описываться в одной строке. В этом случае они разделяются точкой с запятой.

Умолчания. Выражение `DEFAULT ORDER=YES` в начале описания данных означает, что спецификация `ORDER=YES` распространяется на все структуры, встречающиеся в описании.

Описание словаря опускается в случае отсутствия данных со ссылками на словарные данные. Если предполагается использовать словарь, автоматически связанный с базой (см. п. 1.4), то он должен быть описан в начале описания данных при помощи двух спецификаций. Первая задает имя словаря, вторая — набор данных, содержащий словарь (имя оператора `DD` или имя набора данных).

Номер уровня используется для представления текста ДОД в уровневом виде (см. п. 1.3). Описание словаря и умолчания задаются на уровне 01, номер уровня элемента определяется иерархическим порядком ДОД. Номер уровня представляет собой целое число с лидирующим нулем, содержащее не более двух значащих цифр. Он пишется с начала карты после одного или нескольких пробелов и отделяется от остального текста карты пробелом. Текст описания данных пишется на картах в позициях со 2-й по 71-ю включительно, символ * в 72-й позиции является признаком продолжения текста на следующую карту. Содержимое позиций 73—80 безразлично. Если карта начинается с символов «++» или «—» в позициях 1—2, то она считается комментарием и полностью игнорируется.

Элемент. Под элементом понимается описание данного любого уровня — от простого данного, соответствующего терминальной вершине, до сложной структуры, соответствующей части дерева или даже целому дереву (компоненту) описываемых данных. Каждой вершине ДОД в описании соответствует некоторый элемент.

Описываемое данное может иметь имя, которое предшествует собственно описанию данных и отделяется от него двоеточием. Если элемент состоит из одного имени, то соответствующее ему данное описывается в виде элемента на уровне 01.

Данное представляет собой описание данного. Данное с предшествующим ему именем составляет элемент. Данное в соответствии с его типом может быть простым данным, структурным данным, массивом, ссылочным данным и ссылкой на словарь.

Простое данное. Обязательной частью описания простого данного является указание его типа (`INT`, `REAL` и т. д.), за которым могут следовать спецификации и значение. Значение данного, если оно указано, будет постоянным на все время существования базы.

Структурное данное может быть типа `STRUCT` (структура) и `FSTRUCT` (жесткая структура). Структурное данное требует описания всех составляющих его данных. Каждое описание, представляющее собой поддереву ДОД, располагается на подчиненных

уровнях по правилам представления поддеревьев в уровневой записи (см. п. 1.3).

Массив в описании данных определяется ключевым словом **ARRAY**. Описание элемента массива располагается на подчиненном уровне.

Если массив нумерованный, то его описание должно содержать спецификацию **NUM=YES** за словом **ARRAY**. Если массив ключевой, то его элемент должен быть структурой, в описании которой за словом **STRUCT** следует спецификация **KEY=имя ключа**. Массив считается простым, если он не определен как ключевой или нумерованный.

Ссылочное данное. После ключевого слова **REF** (ссылка) или **AS** (шаблон) в описании ссылочного данного должно быть указано составное имя данного, на которое происходит ссылка. Составное имя заключается в апострофы. Описание ссылочного данного определяется составным именем.

Ссылка на словарь. В описании данного со ссылкой на словарь тип данного **VOC**, **CODE** или **RCODE** определяет тип обращения к словарю. В спецификациях в описании данного типа **CODE** или **RCODE** может быть указан префикс в виде буквы, который будет добавляться к коду при занесении значения данного в словарь или при поиске кода в словаре.

Спецификации задают дополнительную информацию, необходимую при описании данных. Например, при помощи спецификаций массив может быть объявлен нумерованным; для структуры, являющейся элементом ключевого массива, может быть определен ключевой элемент и т. п. Спецификации заключаются в наклонные черточки. Если при описании одного и того же данного необходимо задать несколько спецификаций, то они отделяются друг от друга вапатыми внутри наклонных черточек, при этом порядок, в котором они заданы, не имеет значения.

В описании данных предусмотрены следующие спецификации:

NUM=YES. Эта спецификация может быть указана только при описании массива. Соответствующий массив в этом случае объявляется нумерованным.

KEY=имя. Эта спецификация может быть задана только в структуре, являющейся элементом массива. Спецификация объявляет массив ключевым. Имя, указанное в спецификации, объявляет ключом соответствующий элемент описываемой структуры.

SIZE=целое. Эта спецификация может быть указана при описании как массивов, так и простых данных. Для массива спецификация определяет максимальное число элементов массива, для простого данного — максимальный размер в байтах. Если эта спецификация опущена при описании массива, то число элементов предполагается неограниченным. Спецификация **SIZE** используется для

описания данных жесткой структуры, а также для контроля данных, вводимых в БД. Реальные размеры массива в базе определяются введенным числом элементов, а размеры простых данных — их значениями.

PFORM=целое [целое]. Эта спецификация используется для определения формата вывода простых данных. Первое целое определяет общее число позиций, которое отводится в строке под значение данного, когда оно выводится на АЦПУ или на дисплей (если в процессе вывода не будут заданы другие размеры). Второе целое задается только для числовых данных, оно определяет число знаков, выводимых после десятичной точки (с округлением). Если размер текстового значения больше размера, указанного в спецификации PFORM, то значение этого данного при выводе переносится на следующие строки вывода в те же самые позиции с повторением формата строки.

ORDER=YES. Эта спецификация может быть задана только при описании структуры. При этом в ДОД сохраняется порядок элементов структуры, в котором они описаны в тексте на ЯОД. В противном случае порядок элементов определяется алфавитным (латинским) порядком имен данных, образующих структуру. Порядок элементов в структуре проявляется только при выводе на печать всех элементов структуры без указания имен, например, при распечатке БД процедурой ISPRBASE.

MM=целое. Используется только для данных типа DEC. Определяет число знаков после запятой (от 0 до 8). По умолчанию MM=0.

буква. Эта спецификация употребляется для обозначения префикса при описании ссылки на словарь типа CODE или RCODE.

Скобочный вариант записи ДОД легко может быть сопоставлен уровневому варианту. Для получения синтаксической таблицы в скобочном виде достаточно записи, относящиеся к описанию данных и строке, заменить записью:

описание данных:=[умолчания;][описание словаря;]элемент;
[элемент;]...

END;

Как упоминалось выше, в скобочном представлении совокупность элементов одного уровня, подчиненных общей вершине, заключается в скобки. Внутри скобок элементы разделяются между собой точками с запятой. Все описание заключается словом END;.

1.7. Трансляция и печать описания данных

Трансляция ДОД является заключительным этапом подготовки описания данных. Исходный текст описания представляется на перфокартах или в виде раздела библиотеки исходных текстов (см.

рис. 1.3). Результатом трансляции является набор данных (файл) на диске. Набор данных ДОД может быть временным (для отладки описания или для использования его в том же задании), либо постоянным (для использования в других заданиях). Набор данных ДОД используется при вводе данных и при работе с базой.

Трансляция ДОД осуществляется процедурой ISCREDO (в скобочной записи — процедурой ISCREDDOD). В предложении EXEC для этих процедур можно задавать следующие параметры:

R — объем оперативной памяти; по умолчанию R=180K;

DOD — задает имя результирующего набора данных в виде: DOD=имя набора. Если не указывать параметр DOD, то результирующий набор данных будет временным с именем &DOD;

V — том, на котором будет храниться набор данных;

S — количество блоков в результирующем наборе данных &DOD; по умолчанию S=9;

WS — количество блоков в рабочем файле (создается в процессе трансляции); по умолчанию WS=9;

D — указывает диспозицию файлов; по умолчанию D='PASS';

U — указывает устройство для набора данных DOD; по умолчанию U=SYSDA;

OUT — указывает DD-параметры выходного файла, предназначенного для распечатки текста описания данных и диагностических сообщений; по умолчанию OUT='SYSOUT=A';

BLK — указывает размеры блока (единицы аллокации файлов); по умолчанию BLK=1692.

Набор данных с текстом на ЯОД определяется DD-оператором с именем SYSIN.

Пример задания на трансляцию ДОД:

```
// EXEC ISCREDO,DOD=A2DOD1,V=USER1,  
// D='NEW,KEEP',S=15
```

текст на ЯОД

В результате трансляции ДОД будет размещен в наборе данных A2DOD1 на диске USER1.

П р и м е р:

```
// EXEC ISCREDO  
//SYSIN DD DSN=SOURCE(A2MKDOD), DISP=SHR
```

Текст на ЯОД берется из раздела A2MKDOD библиотеки SOURCE. ДОД размещается во временном наборе.

Оттранслированное дерево описания данных может быть распечатано с помощью процедуры ISPRDOD. Имя набора данных, подлежащего распечатке, задается параметром DOD оператора

EXEC. Параметры R, U, OUT — стандартны для процедур ИНЕС и означают следующее:

R — объем оперативной памяти, по умолчанию 100K;

U — тип устройства, на котором расположен набор данных DOD, по умолчанию U=SYSDA;

OUT — описание выходного файла для печати, по умолчанию OUT='SYSOUT=A'.

Параметр V или VDOD (любой) указывается для некаталогизированного набора данных. Пример задания на распечатку:

```
// EXEC ISPRDOD, DOD=A2DOD1, V=USER1
```

1.8. Корректировка ДОД

Процесс проектирования базы данных не ограничивается лишь ее разработкой, но растягивается на весь период ее существования. Проектирование начинается с создания исходной базы данных или ее части, затем работы по проектированию ведутся при развитии и модификации базы. Изменения вносятся в базу данных при создании на ее основе новых приложений с новыми требованиями к данным, либо при изменении требований к данным и их обработке в существующих приложениях. Перепроектирование базы данных может быть вызвано и требованиями повышения производительности прикладных программ *).

В процессе перепроектирования базы в дерево данных ИНЕС могут добавляться отдельные атрибуты или целые иерархические структуры и подструктуры данных. Для внесения этих добавлений достаточно изменить ДОД и ввести в базу новые данные — перезагрузки всей базы не требуется. Разрешаются следующие изменения ДОД:

- в любую структуру может быть добавлена вершина любого типа, в частности — вместе с поддеревом;

- любая вершина может быть уничтожена (вместе с ее поддеревом) или заменена новой вершиной с тем же именем и новым описанием.

Заметим, что новая вершина всегда должна быть описана заново, — в частности, в цепочку иерархически подчиненных вершин нельзя вставить новый уровень. Место вставки новой вершины (с поддеревом) указывается составным именем существующего ДОД. При этом цепочка имен не должна проходить через ссылки на шаблон. Синтаксически возможность внесения изменений в ДОД обеспечивается наличием двух типов указаний:

*) Наш опыт разработки АСУ показывает, что если система в течение года эксплуатации не модернизируется и не развивается как по составу данных, так и по формам входных и выходных документов, то она никому не нужна и эксплуатируется формально.

— DELETE — уничтожить вершину вместе с ее поддеревом;
 — OLD — вершина старая, ее описание надо сохранить (используется только при вставке новой вершины в ДОД, вершины которого упорядочены по спецификации ORDER=YES).

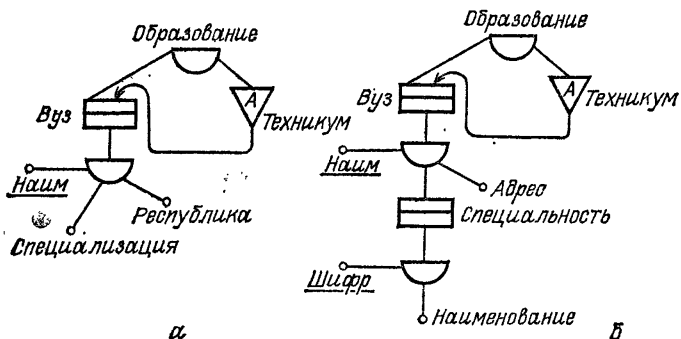


Рис. 1.19

Пример. Пусть старое дерево описания данных имеет вид (см. рис. 1.19а):

```
01 ОБРАЗОВАНИЕ: СТРУКТ (ВУЗ;ТЕХНИКУМ; АС'ОБ
РАЗОВАНИЕ.ВУЗ')
01 ВУЗ: АРРАУ
02 СТРУКТ/КЕУ=НАИМ/
03 НАИМ: РТЕХТ; РЕСПУБЛИКА: ТЕХТ
03 СПЕЦИАЛИЗАЦИЯ: ТЕХТ
```

Мы хотим вершину РЕСПУБЛИКА заменить на простое дан-
 ное АДРЕС, вершину СПЕЦИАЛИЗАЦИЯ уничтожить и на том
 же уровне завести новый массив СПЕЦИАЛЬНОСТЬ (см. рис.
 1.19б). Текст поправок на ЯОД будет иметь вид:

```
01 ОБРАЗОВАНИЕ.ВУЗ.
02 РЕСПУБЛИКА: DELETE
02 СПЕЦИАЛИЗАЦИЯ: DELETE
02 АДРЕС: ТЕХТ
02 СПЕЦИАЛЬНОСТЬ: АРРАУ
03 СТРУКТ/КЕУ=ШИФР/
04 ШИФР: ТЕХТ; НАИМЕНОВАНИЕ: ТЕХТ
```

При обработке поправок транслятор по возможности сохраняет
 доступ к данным, описание которых не изменилось. Если это не-
 возможно, транслятор печатает предупреждающее сообщение с ука-
 занием имен недоступных вершин.

В приведенном примере после внесения поправок из ДОД исчезнут вершины РЕСПУБЛИКА, СПЕЦИАЛИЗАЦИЯ и ОБЛАСТЬ, причем транслятор напечатает сообщение «стали недоступными данные из поддерева замененной вершины СПЕЦИАЛИЗАЦИЯ». Поддерево вершины ТЕХНИКУМ будет отражать измененную структуру шаблона (заметим, что корень шаблона заменять нельзя). Массивы ВУЗ и ТЕХНИКУМ можно обрабатывать по новому ДОД, в частности — заполнять данными массив СПЕЦИАЛЬНОСТЬ.

Внесение изменений в оттранслированный ранее ДОД совершается с помощью процедуры трансляции ISCREDO. При этом транслятору предъявляется текст изменений и набор данных со старым ДОД. Изменения могут быть сделаны прямо в этом же наборе данных. Кроме того, транслятор позволяет получить результат изменений в новом наборе данных, сохраняя старое описание.

Если требуется сохранить старое описание, а результат изменений поместить в новый набор данных, то нужно в процедуре указать PARM=M и старый ДОД описать DD-предложением с именем MASTER. Результат трансляции с изменениями будет помещен в набор данных по DD-предложению с именем DODTREE. Например:

```
// EXEC ISCREDO,PARM=M,DOD=A2DODNEW,  
// D=', KEEP', V=USER2  
//MASTER DD DSN=A2DOD1,DISP=OLD,UNIT=SYSDA,  
//          VOL=SER=USER1
```

текст изменений на ЯОД

По этому заданию транслятор создаст новый набор данных A2DODNEW на диске USER2 и занесет туда исправленный ДОД.

Если в процедуре не указан параметр M, то транслятор берет старый ДОД из DD-предложения с именем DODTREE и производит указанные изменения прямо в этом наборе данных. Например:

```
// EXEC ISCREDO,DOD=A2DOD1,D='MOD,KEEP',  
//          V=USER1
```

текст изменений на ЯОД

Изменения будут внесены в старый набор данных A2DOD1 на диске USER1.

Г л а в а 2

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ В СРЕДЕ ИНЕС

2.0. Введение. Этапы проектирования баз данных

Процесс проектирования баз данных средствами ИНЕС имеет существенные отличия от традиционного проектирования, включающего три этапа: концептуальное проектирование, логическое проектирование и физическое проектирование. Обычно на этапе концептуального проектирования разрабатывается независимое от СУБД описание базы данных — так называемая концептуальная модель или схема — удовлетворяющая требованиям всех приложений. На этапе логического проектирования — концептуальная схема преобразуется в структуры используемой СУБД. Физическое проектирование связано с фактической реализацией базы данных в определенной программно-аппаратной среде.

Важнейшим фактором проектирования на ИНЕС является отсутствие в нем этапа физического проектирования: физическая база данных реализуется автоматически средствами СУБД по заданному описанию логической модели (описанию ДОД). Другим важным фактором являются широкие возможности средств построения логической модели: система практически не накладывает ограничений на описание данных. Общность логического описания позволяет рассматривать его наравне с концептуальной моделью, а средства описания (язык описания данных) — наравне с языками описания концептуальных моделей.

Таким образом, в рамках ИНЕС содержание процесса проектирования сосредоточено на одном концептуально-логическом уровне проектирования и выражается в применении тех или иных формальных логических конструкций. Результатом проектирования является концептуальная схема БД, представленная графом со стандартными обозначениями. Естественно, что при этом методические приемы проектирования касаются в первую очередь типовых кон-

струкций логической модели и способов их применения. Ниже описываются типовые логические конструкции ДОД, представляющие собой набор универсальных средств проектирования логической модели: в их терминах выражается практически любая БД.

В общем случае к концептуальной модели БД предъявляется три вида требований: требования описания реальных объектов предметной области, функциональные требования решения задач и получения выходных данных и требования технологии ведения БД.

Объектные требования к БД заключаются в необходимости представить в концептуальной модели достаточно детализированные, точные и полные описания основных объектов предметной области. Иными словами, в модели должны быть описаны реальные объекты, процессы и понятия предметной области, представляющие самостоятельный интерес для пользователей системы. Основные объекты проектирования описываются на первых этапах процесса проектирования. При этом практически не учитываются способы использования данных в будущих прикладных программах, описания составляются в соответствии с реальной природой объектов и их взаимосвязей. Примеры основных объектов: в сфере контроля исполнения документов на предприятии — объекты ДОКУМЕНТ и ИСПОЛНИТЕЛЬ, в сфере учета успеваемости в вузе — объекты СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, ПРЕДМЕТ, в сфере управления поставками — объекты ЗАКАЗ, ИЗДЕЛИЕ, ЗАКАЗЧИК, ПОСТАВЩИК и т. д. Описание основных объектов используется в дальнейшем как основа для обеспечения информационных функций системы. Фактически основные описания формируют структуру модели, в рамках которой должны существовать функциональные требования к БД.

Функциональные требования к представлению данных в БД являются требованиями пользовательских приложений. Каждое приложение выполняет одну или несколько функций по обработке данных и получению результатов в заданном виде. Функции используют различные виды входных данных, а также дополнительные данные (справочная информация, результаты вычислений и т. д.). Реализация функциональных требований предполагает модификацию концептуальной схемы, в частности — введение вспомогательных структур данных. При этом проявляются основные противоречия процесса проектирования БД, требующие компромиссных решений, в первую очередь — противоречие между памятью БД и временем доступа к данным (как правило, скорость доступа можно увеличить за счет введения вспомогательных структур данных и увеличения объема памяти. О распределении данных знаний между декларативными и процедурными см. п. 4.0.1).

Требования технологии учитывают в первую очередь организационную сторону ведения БД. В самом общем виде эти вопросы

входят в область администрирования баз данных, объединяющую все связанные с базой данных административные и технические функции. В этом плане вопросы организационного обеспечения базы данных подробно рассматриваются, например, в [62]. В процессе проектирования учитываются, как правило, более частные факторы технологии.

Процесс проектирования концептуальной схемы БД средствами ИНЕС может быть представлен в виде трех этапов, см. [29]. Эти этапы являются итеративными: предполагается, что схема будет совершенствоваться путем ее перепроектирования. На первом этапе решается вопрос о выделении нескольких самостоятельных баз данных, либо об использовании одной общей. На втором этапе конструируется схема БД, причем для обеспечения функциональных требований предлагаются типовые конструкции инвертированных входов и справочных массивов, обсуждаются методы организации сетей путем введения динамических и постоянных ссылок. Кроме того, приводятся две типовые схемы БД, предназначенные для решения задач *разузлования* и *классификации*. На третьем этапе уточняются локальные конструкции схемы и типы отдельных данных.

2.1. Одна или несколько баз данных

На первом этапе процесса проектирования концептуально-логической модели БД решается вопрос о создании одной общей базы данных либо о разделении ее на несколько более мелких баз, а также о распределении данных и функций, выполняемых системой, по отдельным базам. Следует отметить, что база данных представляет собой самостоятельный набор данных в смысле ОС и поддерживается независимо. В то же время прикладные программы могут иметь доступ одновременно к двум, трем и более базам данных, т. е. с точки зрения функционирования отдельные базы данных не являются изолированными, между ними может быть установлена взаимосвязь в рамках пользовательской программы-запроса.

В решении вопроса о создании одной или нескольких баз данных учитываются основные типы требований к концептуальной модели:

1. Технология поддержания баз данных: количество коллективов, поддерживающих БД, организация поддержки; динамичность изменений БД; использование диалоговых средств; режим эксплуатации.

2. Функциональные характеристики проектируемой ИС: распределение отдельных задач (функций) по комплексам данных, основные характеристики этих задач (объем хранимых данных, допустимое время ответа на запрос и т. д.).

3. Состав и структура объекта описания (проектируемой системы), расчлененность или, наоборот, взаимоувязанность комплексов данных в этом описании.

Границы влияния каждого из названных факторов, как правило, не удается оценить точно. В целом можно сказать, что если комплексы данных поддерживаются независимо разными коллективами, служат для решения индивидуальных комплексов задач, имеют большие объемы (например, в пределах 100-мегабайтного диска) и отличаются по содержанию, то они должны быть организованы в виде отдельных баз данных. Напротив, комплексы данных, включенные в единый процесс поддержания, имеющие общие задачи и близкое содержание, целесообразно объединить в одну базу. При решении вопроса о делении баз необходимо учитывать также перспективы развития системы.

Вопросы поддержания баз данных имеют первостепенное значение в решении вопроса об их делении. Независимые процессы поддержания, рассчитанные на различную динамику обновления и использования отдельных частей базы, могут служить основанием для ее разделения. Базы могут быть разделены также при наличии нескольких коллективов, ответственных за поддержание отдельных частей базы. Например, база данных по управлению строительством содержит данные по стройматериалам, оборудованию, поставщикам, незавершенному производству и т. д. Если за ведение этих данных отвечают различные подразделения в организации, естественно поддерживать общий комплекс данных в виде совокупности взаимосвязанных БД. Связи между базами устанавливаются в рамках пользовательских программ. Если в другом случае вся база данных в целом поддерживается несколькими коллективами, происходит разделение ее на несколько однородных баз. Например, база данных по финансированию строительства в отрасли содержит показатели по строительным организациям, которые поддерживают сами организации в своих БД. Данные из этих баз периодически будут сливаться в одну интегрированную БД министерства, по которой можно получать сводные показатели по отрасли в целом.

Соотношение функциональных и объектных требований к базе данных сказывается в первую очередь в ее общей ориентации. С одной стороны, базы данных могут создаваться в расчете на конкретные приложения (так называемые прикладные базы данных), с другой стороны — целесообразна разработка предметных баз данных, соотнесенных с объектами предметной области, а не с вычислительными приложениями. В первом случае вложение данных в процессы обработки облегчает создание прикладной базы, но затрудняет ее адаптацию к новым приложениям. Во втором случае предметные базы данных строятся как независимое от приложений

представление основных объектов реальности, относящихся к проектируемой системе. Такое представление отличается стабильностью, так как при развитии системы меняются, в основном, функции использования данных, а не их типы. Кроме того, облегчаются проверка и поддержание целостности данных.

В качестве примера функционально-ориентированной базы данных можно привести БД со сводными показателями. Например, в кадровой системе база СОТРУДНИК содержит элементарные данные на каждого сотрудника предприятия, база ПОДРАЗДЕЛЕНИЕ — интегрированные данные на каждое структурное подразделение на всех уровнях управления. Из элементарных данных базы СОТРУДНИК путем перебора записей может быть получена любая выходная форма, но на ее обработку будет затрачено соответствующее время. Из интегрированных данных базы ПОДРАЗДЕЛЕНИЕ могут быть выданы регулярные выходные формы, причем обработка данных фактически сводится к выводу на АЦПУ уже готовой информации. Другой пример заключается в выделении в качестве отдельных БД функциональных логических конструкций, например, одного или нескольких инверсных входов.

Введение наряду с предметными БД прикладных функционально-ориентированных баз данных приводит к дублированию информации; иногда такие базы полностью строятся на основе данных, входящих в предметные БД. В последнем случае, как правило, режим эксплуатации предусматривает поддержку и обновление только предметных баз, по которым остальные БД периодически воспроизводятся. При этом данные могут переписываться из одной базы в другую средствами запросов, что обеспечивает практически все виды промежуточной обработки, либо специальными средствами перекачки данных (так называемые средства реструктуризации), которые имеют ограниченные возможности обработки данных, но позволяют сэкономить на прикладном программировании. Специальные средства ИНЕС предназначены для перекачки данных в инверсные входы.

В системе могут дополнительно создаваться временные вспомогательные базы данных, например, база данных для хранения и обработки поправок к основной БД, базы данных для выдачи регулярных выходных форм по требованию пользователя, базы для просмотра возможных вариантов плана и т. д.

2.2. Выделение основных иерархических структур

Как указывалось выше, основное содержание процесса проектирования логической структуры БД заключается в выборе и композиции типовых сетевых конструкций. Локальные характеристики структуры данных (выбор типов данных и отдельных иерархических

связей) рассматриваются на третьем этапе проектирования (см. п. 2.7).

Ниже приводится ряд типовых сетевых конструкций в составе БД и обсуждается их применение; на рисунках ссылки изображаются стрелками, соединяющими соответствующие вершины, причем исходная вершина изображается в виде треугольника. Ссылки, заданные на физическом уровне (ссылка REF), и ссылки, организованные программно (динамические ссылки, см. п. 2.6), изображаются одинаково. Кружком будет изображаться поддерево произвольной структуры, точкой — простое данное, а также структурная вершина элемента массива. Остальные обозначения соответствуют принятым в пп. 1.2—1.4.

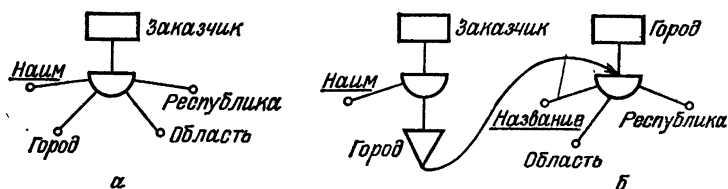


Рис. 2.1

Как указывалось выше, начало этапа логического проектирования состоит в описании основных объектов предметной области и существующих между ними взаимосвязей. Эти объекты представляются в базе в виде деревьев, которые в дальнейшем будут называться основными деревьями. (Основным деревьям противопоставляются вспомогательные: инверсные входы для быстрого доступа к данным, деревья, содержащие справочную информацию или результаты промежуточной обработки данных, и т. д.) Процесс описания основных объектов предметной области требует точного употребления конструкций ДОД. Например, на рис. 2.1 представлена в двух вариантах связь объекта ЗАКАЗЧИК с данными ГОРОД, ОБЛАСТЬ, РЕСПУБЛИКА. В первом случае эти данные считаются просто атрибутами объекта ЗАКАЗЧИК, во втором случае ГОРОД описывается как отдельный объект.

Другая типичная ситуация заключается в необходимости описать элементы, зависящие от нескольких ключей, например, данные АУДИТОРИЯ и ВРЕМЯ задаются в зависимости от ключей объектов ГРУППА и ПРЕДМЕТ (в этом случае говорят, что элементы данных АУДИТОРИЯ и ВРЕМЯ задаются составным ключом). Считая, что эти элементы описывают свойства связи ГРУППА-ПРЕДМЕТ, можно представить два варианта реализации этой ситуации (см. рис. 2.2). В случае рис. 2.2б ПРЕДМЕТ рассматривает-

ся как самостоятельный объект и вводится дополнительный объект ЛЕКЦИИ.

Некоторые объекты предметной области можно определить с помощью подклассов. При этом предполагается наличие общих

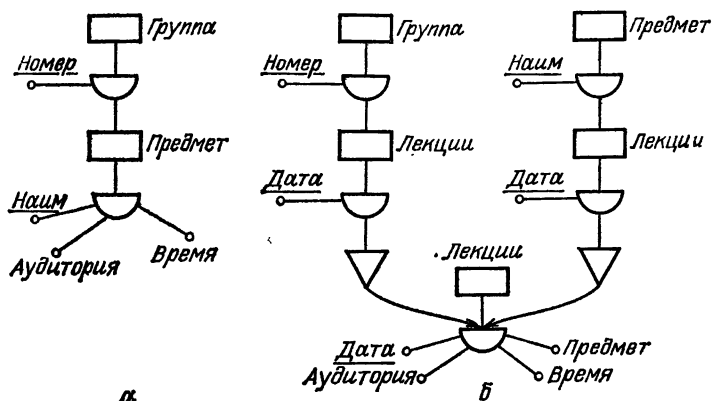


Рис. 2.2

атрибутов для определенного класса объектов и дополнительных атрибутов для каждого типа объекта. Например, в БД по снабжению описывается объект ЗАКАЗЧИК, причем для заказчиков из

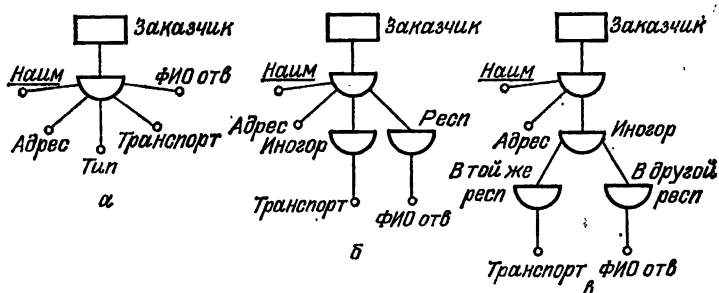


Рис. 2.3

других городов вводится данное ТРАНСПОРТ, а для других республик — фамилия ответственного за поставки ФИО ОТВ. На рис. 2.3 приводятся варианты реализации такого описания. В варианте рис. 2.3а классификация типов описана структурой ИНОГОР. Присутствие соответствующей структуре вершины в ДД означает, что ЗАКАЗЧИК имеет дополнительные атрибуты. В случае рис. 2.3а этот факт проверяется наличием вершин ТИП, ТРАНСПОРТ,

ФИО ОТВ, в случае рис. 2.36 — наличием вершины ИНОГОР или вершины РЕСП.

Наличие или отсутствие в базе вершины ДД, описанной в ДОД, может служить вспомогательным средством, позволяющим квалифи-

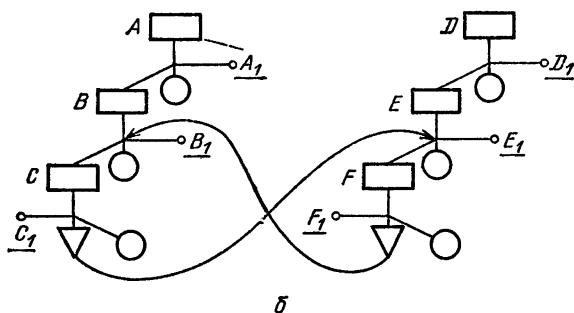
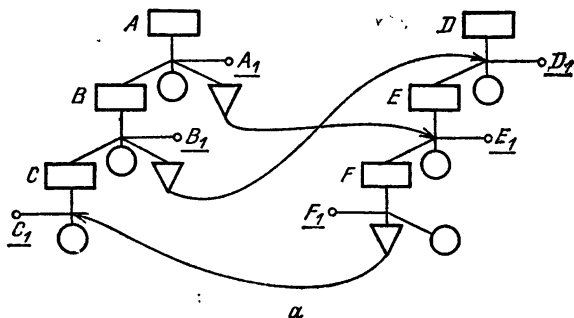


Рис. 2.4

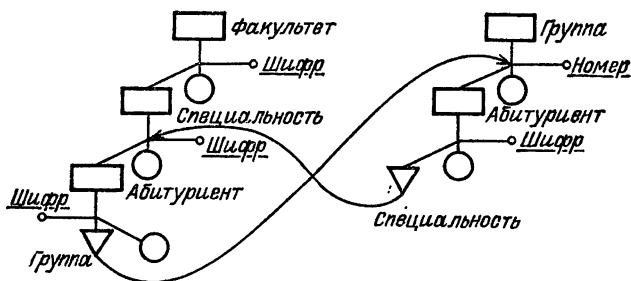


Рис. 2.5

цировать причины отсутствия информации в базе: отсутствие вершины означает в этом случае отсутствие соответствующего свойства или объекта в реальности, а наличие вершины без значения

означает, что сведения о значении отсутствуют (вершины создаются средствами ввода документов и средствами запросов).

Приведенные примеры подчеркивают важность точного использования средств описания данных. Как указывалось выше, точность и полнота описания предметной области определяют гибкость системы и возможность ее развития.

Одновременно с выбором основных деревьев могут быть установлены простейшие взаимосвязи между ними в виде ссылок. Вариант взаимных ссылок приводится схематически на рис. 2.4. Перекрестные ссылки, представленные на рис. 2.5, возникают, когда в двух цепочках ключей конечные ссылки делаются на середину соседней цепочки. При этом в базе данных возникают сети данных с замкнутыми циклами. Так, в ДОД цепочка «ФАКУЛЬТЕТ. СПЕЦИАЛЬНОСТЬ. АБИТУРИЕНТ.» заканчивается ссылкой на вершину «ГРУППА» в цепочке «ГРУППА. АБИТУРИЕНТ.». В свою очередь, вторая цепочка заканчивается ссылкой на вершину «СПЕЦИАЛЬНОСТЬ» в первой цепочке. Выбирая в качестве ключа к массивам АБИТУРИЕНТ в обеих цепочках одну и ту же фамилию, можно «пройти» по ним в базе и оказаться в исходной вершине. Аналогичные циклы содержит структура рис. 1.18.

Построение исходных основных деревьев и простейших связей отражает требования описания объекта и, как указывалось выше, формирует, тем самым, общую структуру, в рамках которой должны существовать функциональные требования. Последующие шаги проектирования логической структуры БД связаны с удовлетворением функциональных требований к данным. С этой целью вводятся вспомогательные деревья, устанавливаются новые ссылки, перестраивается уже существующая логическая структура (например, поддерево выделяется в качестве самостоятельного дерева, изменяется иерархия вершин и т. д.). Следует, однако, иметь в виду, что результатом проектирования должна быть структура, удовлетворяющая как функциональным требованиям, так и требованиям описания объекта.

Ниже приводятся конструкции, позволяющие удовлетворить функциональные требования к базам данных.

2.3. Инверсные входы

Одной из наиболее распространенных функций в задачах обработки данных является функция построения инвертированных списков. *Инвертированные списки* представляют собой списки объектов определенного типа, упорядоченные по значениям одного или нескольких атрибутов этих объектов. Например, список сотрудников, упорядоченный по лабораториям, список документов, упоря-

доченный по шифрам исполнителей и по срокам исполнения (для каждого исполнителя), список заказов на изделия, упорядоченный по поставщикам, по типам изделий и по цене в рамках одного типа и т. д. В более сложных случаях задаются условия на значения атрибутов, причем с помощью логических операций могут строиться составные условия в виде логических формул. Например, может быть составлено требование построить список изделий, относящихся к типам А1 или А2, ценой не выше 100 рублей и таких, которые поставляются из Москвы или Московской области.

Сортировка по инвертированным спискам объектов, представленных в базе общим массивом, является основой получения большинства видов выходных форм. Получение инвертированных списков задается в ИНЕС на уровне логической структуры либо обеспечивается средствами обработки последовательных наборов данных. В первом случае в ДОД задаются специальные вспомогательные деревья — так называемые инверсные входы, содержащие на нижнем уровне массивы-списки ссылок на объекты. Эти ссылки отсортированы последовательно по значениям заданных атрибутов. Во втором случае инвертированные списки задаются условиями сортировки последовательного набора данных, в который из базы заранее выдается полный список объектов.

Многоуровневый инверсный вход представляет собой цепочку вложенных массивов, каждому из которых сопоставлен в качестве ключа определенный атрибут. Цепочка заканчивается массивом ссылок на объекты отбора, представленные поддеревом базы. Вид отбора задается цепочкой атрибутов инверсного входа и фиксируется в ДОД. Способ отбора, определяется пользовательской программой: при спуске по цепочке на каждом уровне указывается значение очередного атрибута (ключ массива) либо задается перебор значений, возможно, с проверкой заданных в программе условий. Результат отбора представляет собой список либо совокупность списков ссылок на объекты, удовлетворяющие заданным условиям и значениям атрибутов. Инверсные входы позволяют быстро получать такие списки, так как фактически для каждой цепочки значений атрибутов в базе хранится готовый набор ссылок-идентификаторов объектов отбора (см. схему и пример на рис. 2.6).

Инверсный вход может содержать на разных уровнях вспомогательные терминальные вершины, которые позволяют в случае необходимости организовать дополнительный отбор элементов массивов (вершина ОТВ в инверсном входе — рис. 2.6б — служит основанием для отбора ответственных исполнителей). Эти вершины могут содержать также статистические данные, значения которых формируются в процессе ввода и обработки данных (можно ввести вершину, содержащую количество документов в подчиненном массиве; при этом подсчет элементов массива можно обеспечить сред-

ствами ввода данных). В общем случае многоуровневые деревья-входы могут содержать атрибуты, не входящие в основное дерево.

Инверсные входы, таким образом, предназначены для решения широкого класса задач на сортировку объектов. Объекты представлены в базе основными деревьями, условия отбора обеспечиваются

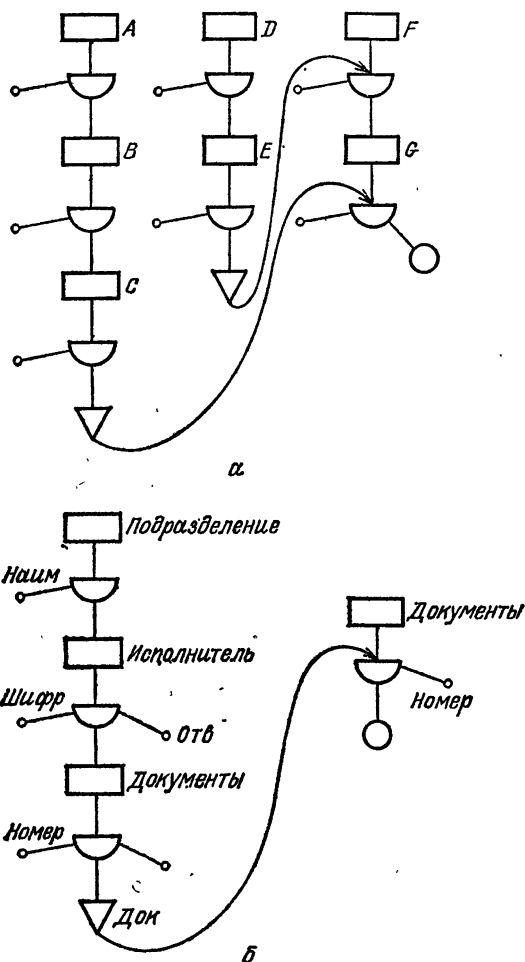


Рис. 2.6

пользовательской программой при проходе по базе. Преимущество схемы сортировки заключается в независимой и удобной корректировке как данных об объектах сортировки, так и инверсных вхо-

дов, представляющих различные классификации этих объектов (например, в базе на рис. 2.66 перевод документа в другое подразделение соответствует переносу ссылки, но не всего поддерева). Заметим, что инверсные входы могут быть отделены в самостоятельную базу.

В целом использование инверсных входов дает выигрыш во времени доступа к данным и одновременно приводит к дублированию информации (впрочем, относительное увеличение объема базы бывает, как правило, незначительным). Введение инверсных входов заметно увеличивает трудоемкость загрузки и корректировки данных, на которые приходится основная нагрузка по обработке данных (использование ссылки типа REF в инверсных входах может увеличить время загрузки базы в несколько раз).

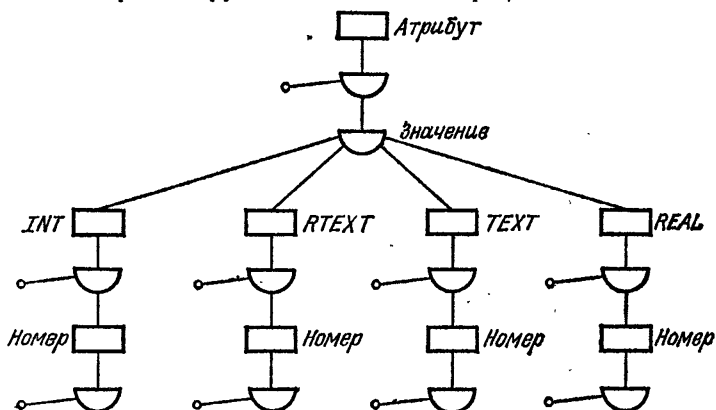


Рис. 2.7

Для обеспечения быстрого поиска объектов по значениям их атрибутов может быть использована так называемая поисковая документальная система ИНЕС, включающая атрибутивную справочную стандартного вида (ее ДОД представлен на рис. 2.7) и специальный язык поиска объектов по этой справочной. Атрибутивная справочная создается и поддерживается пользователем-программистом. Справочная включается в основной ДОД, либо составляет отдельную справочную базу, загружается по аналогии с инверсными входами. Язык поиска объектов позволяет кратко представлять логические формулы над условиями отбора, что дает экономию в прикладном программировании. Атрибутивная справочная является также основой для применения диалоговой справочной системы, позволяющей отбирать, суммировать и переупорядочивать сведения из БД, просматривать базу в заранее описанном формате, пользоваться подготовленной справочной информацией. Настройка системы

осуществляется программистом, после чего ею может пользоваться специалист в предметной области.

Второй способ получения инвертированных списков сводится к обработке последовательных наборов данных. Данные выводятся из базы в последовательные наборы в виде записей фиксированного формата и обрабатываются в соответствии с содержанием фиксированных в записи окон. Средства обработки записей позволяют сортировать их по любому окну (в порядке возрастания или убывания значений окна), задавать дополнительные условия сортировки и т. д. Результаты сортировки представляются в виде выходных документов нужной формы при помощи стандартных средств вывода системы ИНЕС. Заметим, что для вывода документа могут использоваться несколько последовательных наборов, подготовленных специальными проходами по базе.

Использование последовательных наборов данных как промежуточного этапа для обработки результатов запроса к БД имеет следующие преимущества:

1) За один просмотр базы данных можно рассчитать несколько выходных форм. Действительно, полный проход по базе обеспечивает получение в последовательном наборе общего списка объектов с нужными реквизитами. Этот список может быть отсортирован различными способами, после чего данные распечатываются в виде выходных документов стандартными средствами вывода. Каждая сортировка последовательного набора может по времени оказаться в несколько раз эффективнее специального прохода по базе.

2) При помощи нескольких последовательных наборов данных можно сгенерировать документ, иерархия которого не совпадает с иерархией хранения и обработки данных в базе (в частности, в этом документе итоги могут предшествовать реквизитам, по которым они рассчитаны).

В целом можно констатировать, что технологию выдачи документов через промежуточную обработку данных в последовательном наборе целесообразно использовать для регулярной выдачи группы выходных форм. Это дает значительный (иногда в несколько раз) выигрыш во времени и дополнительные возможности составления документа. Инверсные входы используются для обеспечения оперативного доступа к информации, в первую очередь при обращении к БД с терминала.

2.4. Деревья-справочные (нормативно-справочная информация)

Данные справочного характера в составе базы данных играют более или менее значительную роль в зависимости от типа информационной системы. В задачах АСУП важная часть сведений

относится к нормативно-справочной информации, причем используются десятки видов справочных данных. В базе данных в виде единой системы данных многоразового использования должны быть представлены многочисленные тома нормативов, справочников, ценников, картотек. Эффективность функционирования подсистем АСУП в значительной степени зависит от качества представления этих данных в базе. В то же время в других типах информационных систем справочная информация может практически отсутствовать.

Справочная информация отделяется в БД по признакам ее содержания, функционирования и технологии ведения. Как правило, она имеет относительно постоянный характер (по сравнению с основными комплексами данных) и используется как вспомогательная, хотя в развитых системах нормативно-справочной информации АСУП на справочных данных решается ряд самостоятельных задач. Важную роль в вопросах отделения справочной информации и ее организации в базе играет технология поддержания БД: справочная информация всегда поддерживается независимо от основных данных. Примеры нормативно-справочных данных: технологические, материальные, трудовые нормативы, цены и тарифы, классификаторы материалов, справочники наименований изделий и их составных частей и т. д.

В базах данных ИНЕС справочная информация может быть организована различными способами. Данные простой структуры (например, справочник наименований) можно поддерживать с помощью словарной системы в виде одной или нескольких словарных БД (об организации и использовании словарей см. пп. 1.4, 2.7). Использование словаря позволяет сочетать быстрый доступ с простой наглядной организацией данных, причем может быть организована двухключевая связь данных. В то же время возможности корректировки словарей ограничены.

Организация справочной информации в виде самостоятельных деревьев ДОО (см. схему и пример на рис. 2.8) позволяет поддерживать ее независимо от остальных структур данных. При этом не только облегчается процесс обновления информации, но и появляется возможность использовать стандартные средства ИНЕС для реорганизации и изменения ее структуры. Становится возможным расширение структуры информации без перезагрузки данных.

Справочная информация может быть отделена также в самостоятельную базу. Этим значительно облегчается ее отладка: при обнаружении ошибки в новом варианте базы можно вернуться в эксплуатируемой системе к старому ее варианту, продолжая отладку на реальной базе и задачах.

Как упоминалось выше, в развитых системах нормативно-справочной информации используются многочисленные виды спра-

вочных данных. При этом в одной задаче используются различные нормы, расценки, справочники, а одни и те же виды справочных данных применяются для решения многих задач (например, справочник наименований изделий и их составных частей, а также справочник применимости деталей в изделии используются практически во всех задачах управления материальными и трудовыми ресурсами). В результате для решения различных задач в системе создается единый комплекс нормативно-справочных данных сложной структуры.

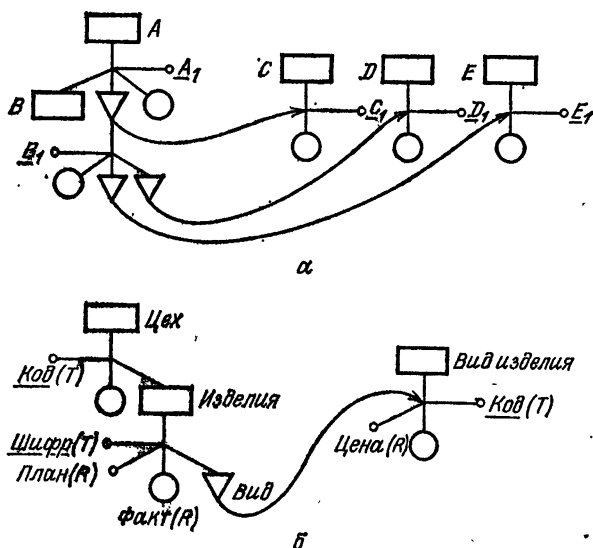


Рис. 2.8

В общем случае нормативно-справочная БД представляет собой совокупность перевязанных ссылками деревьев различного вида в том числе деревьев разузлования (см. ниже п. 2.5). Справочная БД строится и поддерживается как самостоятельная, с учетом интересов решения собственных задач. При этом в нее могут быть введены инверсные входы для ускорения доступа к данным. Другими словами, в нормативно-справочную базу могут, вообще говоря, входить все виды логических конструкций. Дальнейшее развитие такой ВД естественно приводит к образованию отдельной функциональной подсистемы справочного характера. Например, известны функциональные подсистемы НСИ в рамках АСУП, можно привести ряд примеров подсистем-классификаторов различных объектов («Классификатор предприятий отрасли») и т. д.

2.5. Дерево разузлования и дерево классификации

Дерево разузлования предназначено для решения так называемой задачи разузлования (комплектации) изделий. Главную роль в этой задаче играет получение полного перечня деталей, составляющих заданное изделие или список изделий. При этом предполагается, что изделие состоит из составных частей, которые также являются изделиями и могут состоять из частей, и т. д. Дерево комплектации представляет собой стандартное описание изделия, включающее список его составных частей. Очевидно, для описания состава

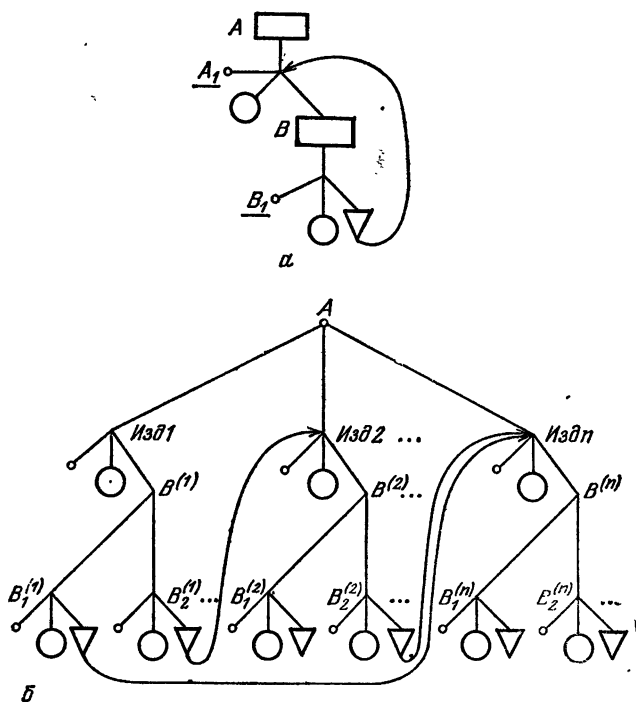


Рис. 2.9

изделия необходимо в составных частях его ссылаться на основное дерево (рис. 2.9а), т. е. делать ссылки на исходные корневые вершины. Цикличность полученного таким образом описания соответствует сложным сетевым структурам, образующимся на уровне хранения данных. Так как в базе данных на первом уровне содержатся данные по всем изделиям, участвующим в рассмотрении, то структура базы представляет собой совокупность деревьев, пере-

вязанных ссылками на их вершины (рис. 2.9б). Каждая такая сеть представляет полный состав исходного изделия, который может быть раскрыт в запросе простым движением по дереву данных.

Интересно сравнить граф разузлования с конструкцией аналогичного вида — графом классификации, в котором ссылка на вершину дерева данных заменена ссылкой на шаблон, т. е. на описание данных (рис. 2.10а). В данном случае шаблон содержит ссылку

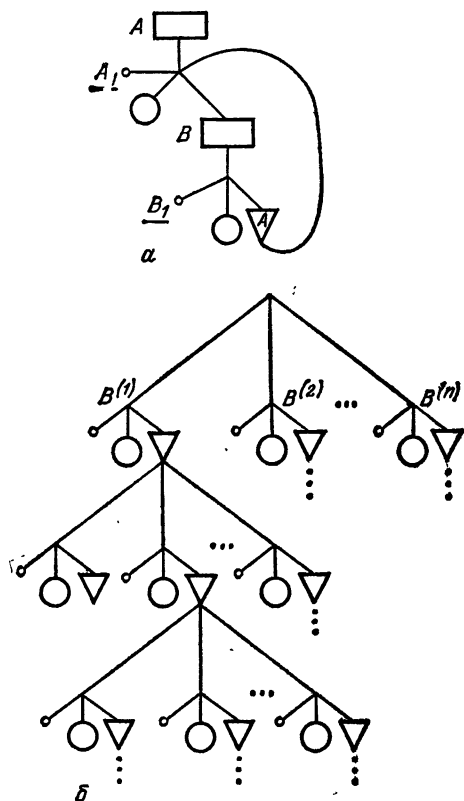


Рис. 2.10

на вышестоящую вершину, но этот цикл в описании означает только, что на уровне описания данных используются однотипные элементы. Структура данных в заполненной базе представляет собой чистую иерархию с произвольной, не ограниченной описанием длиной отдельных ветвей (фактическая длина ветвей соответствует вводимым данным, см. рис. 2.10б).

2.6. Как реализовать сетевую ссылку

Успешное применение перечисленных конструкций связано с выбором способа реализации ссылки, обеспечивающей неиерархические связи в базе данных. В ИНЕС эти связи можно реализовать двумя способами:

— на физическом уровне с помощью данного «ссылка типа REF». По этой ссылке система автоматически отождествляет исходную и конечную вершину в дереве данных: при обращении по месту хранения ссылки пользователь автоматически получает объект ссылки;

— на логическом уровне путем хранения информации об объекте ссылки в виде, определяемом разработчиком. В этом случае способ перехода к вершине-адресату зависит от способа хранения информации в месте ссылки. Переход осуществляется движением по базе, которое задается командами запроса. В типичном случае делается обращение к нужному дереву (к тому же самому, из которого сделана ссылка, к соседнему или даже к расположенному в другой базе данных) и в нем осуществляется спуск к вершине-«адресату» по ключам, хранящимся в исходной вершине ссылки. При организации ссылки на логическом уровне способ доступа не отражается в описании данных. Важно, что система позволяет осуществлять переход по логической ссылке без потери текущего состояния базы.

Заметим, что в некоторых случаях можно заменить обращение к данному по первому или второму способу простым дублированием чисел или текстов в нужном месте, например, если память, необходимая для организации ссылки REF (5 байтов), больше, чем длина самого данного. С другой стороны, даже в этом случае имеет смысл обращаться к данному из различных точек базы и избегать дублирования, если это данное играет важную роль, например, общего параметра.

Сравним основные преимущества физического и логического методов хранения ссылок.

а) С точки зрения затрат памяти на организацию ссылки и времени доступа к данным, расположенным на глубоких уровнях иерархии ДД, выигрыш дает ссылка типа REF. Действительно, для физической ссылки объем хранимой информации и время доступа не зависят от того, на каком уровне иерархии находится объект ссылки. Для перехода по логической ссылке необходимо использовать значения всех ключей, идентифицирующих объект в ведущей к нему цепочке массивов, поэтому память, как и время доступа, для логической ссылки линейно зависят от глубины расположения объекта в иерархии. Чем больше количество ключей, полностью идентифицирующих объект ссылки, тем менее эффективной становится логическая ссылка с точки зрения памяти и времени доступа.

Организация логической ссылки на первый уровень иерархии практически не отличается по памяти и по времени от перехода по ссылке REF.

б) С точки зрения технологии ведения БД значительное преимущество имеют логические ссылки. При частом обновлении базы и значительном увеличении ее объема внутрисистемное поддержание ссылки REF становится громоздким и малоэффективным, в частности, постепенно увеличиваются удельные затраты памяти и времени доступа по ссылке.

в) Важным преимуществом логических ссылок является возможность использовать их при установлении связей между различными БД. При этом обработка нескольких БД, как правило, бывает не сложнее работы с одной базой данных и осуществляется средствами языка запросов высокого уровня. Наиболее употребительным средством является здесь переключение баз в запросе, позволяющее одновременно поддерживать несколько активных точек в разных БД. Возможны и более сложные программы обработки, связанные, например, с вызовом запроса из программы ввода данных, или, наоборот, с обращением из запроса к программе ввода.

С другой стороны, с помощью физических ссылок удастся включать в структуру связи, определяемые во входном потоке или в процессе интерактивной работы по трудноформализуемым критериям. В случае, если объект ссылки может находиться на разных уровнях иерархии (конструкции «блуждающей ссылки», см. [29]), использование логических ссылок становится особенно неэффективным.

Кроме того, использование физических ссылок освобождает программиста от работы по программированию доступа к объекту ссылки, повышается наглядность программы.

В целом можно отметить, что возможность работы с несколькими БД, а также технологичность ведения логических ссылок во многих случаях дает им решающее преимущество перед ссылками типа REF.

2.7. Определение типов вершин

На заключительном, третьем этапе проектирования уточняются типы отдельных вершин базы, причем при необходимости перестраиваются локальные структурные связи.

2.7.1. Структуры и массивы.

1. При использовании структурных вершин, не имеющих подчиненных массивов, рассматривается возможность введения жесткой структуры. Как указывалось в п. 1.3, основная особенность жестких структур по сравнению с простыми заключается в том, что совокупность входящих в них элементарных данных хранится и

считывается как одно целое. Это приводит к существенной экономии времени доступа при одновременном считывании совокупности данных структуры. В то же время к формату данных предъявляются жесткие требования, а длина хранимого значения данного фиксируется заранее в ДОД. Это может привести, например, к потерям памяти при хранении текстов неопределенной длины. Потери памяти накапливаются также за счет отсутствия отдельных элементов структуры, так как при вводе хотя бы одного ее элемента место в базе отводится под всю структуру.

2. В общем случае при использовании структур, простых или жестких, может вставать вопрос о введении дополнительных структурных вершин, например, для объединения данных общего содержания. В этом случае дерево описания данных становится более обозримым, «удобочитаемым», дает возможность манипулировать образовавшимися логическими «кусками» при проектировании базы. Вводимые вершины позволяют также дополнительно структурировать программы доступа и обновления данных. Например, часто используемые или часто обновляемые данные могут быть отделены от редко используемых и редко обновляемых. Данные могут быть также отделены в особую структуру для обеспечения их защиты.

Недостатком дополнительных структурных вершин является введение лишних уровней описания и, соответственно, дополнительных движений по базе при записи и чтении данных.

3. В некоторых случаях данным типа структура (STRUCT) можно заменить массив. Каждому элементу массива сопоставляется в ДОД вершина структуры. Идентификатор (ключ) элемента становится именем вершины, а описание элемента массива под каждой вершиной дублируется. Очевидно, введение такой структуры возможно только для массива с фиксированным количеством элементов (список отделов предприятия, список месяцев года и т. д.). Преимуществом такой конструкции является жесткий контроль за вводимыми данными, а также некоторая экономия памяти за счет хранения наименований элементов в ДОД.

4. При использовании массивов встает вопрос о выборе типа массива. Как правило, предпочтение отдается ключевым массивам. Если невозможно выбрать идентификатор элемента массива, то организуется нумерованный массив.

5. Цепочку вложенных ключевых массивов можно сократить, ускорив доступ к данным. Два вложенных массива объединяются в один массив следующим образом: элементы массивов объединяются в одну общую структуру, а ключи — в один общий составной ключ. Недостаток такого преобразования состоит в значительном дублировании информации: все данные верхнего массива должны будут храниться с каждым данным нижнего.

2.7.2. Типы терминальных вершин.

1. При определении типов терминальных вершин решается вопрос о присвоении вершине словарного типа и об использовании словарной системы (см. п. 1.4).

Использование словарей позволяет, во-первых, сократить объем основной базы за счет хранения в ней сокращенных кодов данных и, во-вторых, поддерживать систему связей данных с их кодовыми эквивалентами (связи типа «наименование-код1-код2-...»). В последнем случае поддерживаемые связи могут быть многоключевыми, так как выборка осуществляется как по наименованию, так и по любому из кодов. При организации таких связей средствами основной БД возникают сложные структуры данных.

Для автоматического сжатия базы используются словарные данные типа VOC. Заполнение словаря значений этих данных осуществляется автоматически при их вводе в базу. Каждое обращение к данному типу VOC сопровождается автоматическим обращением к словарю, так что для пользователя это данное практически не отличается от простого текста.

В других случаях словарь составляется и поддерживается самим пользователем. Связь данных с их кодовыми эквивалентами реализуется в словаре в виде записи

шифр/наименование/дополнение1/дополнение2/...

Часть полей в записях словаря объявляется ключевыми, по любому из них можно искать запись средствами манипулирования данными словарной системы.

При использовании данных типа CODE и RCODE связь базы данных со словарем осуществляется автоматически. При загрузке в базу могут быть введены только данные из заданного словарем списка, что является хорошим средством контроля. При обращении к базе система, как и в первом варианте, автоматически выдает полное значение данного. Заметим, что база, содержащая данные типа VOC, CODE, RCODE, не может использоваться отдельно от словаря.

Для непосредственного обращения к словарю пользователю предоставляются специальные средства языка манипулирования данными словарной системы. За одно обращение можно, например, найти запись по одному из ключевых полей и считать значение любого другого поля этой записи. Коды или значения других ключевых полей словарных записей хранятся при этом в БД как простые данные. Связь может осуществляться одновременно с несколькими словарями. Серьезное преимущество неавтоматического режима работы со словарем заключается в возможности независимо отлаживать и использовать основную базу и словари.

2. На использование данных типа TEXT и RTEXT в ИНЕС накладывается ограничение: длина данного в базе не должна пре-

вышать 250 символов. Тексты большей длины могут задаваться и использоваться следующим образом.

Текстовое данное заменяется в ДОД нумерованным массивом с единственным элементом того же типа и вводится в ДД как значение этого элемента. Если значение данного превышает 250 байтов, то оно автоматически делится системой ввода на части по 250 байтов и каждая следующая часть присваивается очередному элементу массива. При считывании данное обрабатывается средствами языка запросов. Таким способом может быть введен и использован любой, сколь угодно длинный текст.

Проектирование логических схем баз данных с необходимостью предполагает оценку различных вариантов проекта как с точки зрения точности описания предметной области, так и с точки зрения ее функциональных характеристик, в первую очередь объема памяти базы и времени выполнения запросов. Опыт использования ИНЕС показывает, что применение перечисленных выше логических конструкций позволяет смоделировать широкий класс предметных областей. Оценка физических характеристик базы в традиционной схеме выполняется на этапе физического проектирования.

Для оценки физических характеристик баз данных ИНЕС могут применяться аналитические расчеты. Кроме того, для ИНЕС существуют методы оценки физических характеристик будущей базы данных на основе экспериментальных замеров в процессе ее создания (см. [65]). Как в первом, так и во втором случае результаты оценки помогают при сравнении и выборе вариантов логических схем, а иногда показывают необходимость перепроектирования.

На этом же этапе рассматриваются различные способы ускорения работы запросов (см. [29]).

Глава 3

ВВОД ДАННЫХ В БАЗУ

3.0. Введение

3.0.1. Ввод как задача отображения. Ввод данных в базу осуществляется в общем случае на основании входных документов, подготовленных пользователем. Эти документы, содержащие необходимую информацию, должны быть обработаны: данные должны быть извлечены из документов, проверены средствами контроля и введены в БД.

Задача ввода данных в рамках системы интерфейса является задачей отображения пользовательского представления данных (совокупности входных документов) в базу данных заданной структуры. При этом подсистемой ввода обеспечиваются универсальные средства отображения этих данных в структуры БД.

Необходимость в универсальных средствах ввода данных связана с большим разнообразием входных пользовательских документов. Форматы этих документов состоят из соображений удобства работы с документами конечных пользователей, независимо от структур хранения содержимого документов в базе данных. Кроме того, один документ может использоваться для загрузки в несколько баз данных (также и одна база может загружаться с помощью различных видов входных документов, число которых достигает иногда нескольких десятков и сотен). Важную роль играют постоянные изменения входных документов, возникающие при развитии и модернизации подсистем АСОД. Эти изменения происходят значительно чаще, чем изменения в концептуальной схеме БД. Нормой жизни эксплуатируемых («живых») приложений БД является постоянная модернизация внешних пользовательских представлений БД — входных и выходных документов — при сравнительно стабильной концептуальной схеме. Эти изменения должны эффективно поддерживаться средствами отображения входных данных в

структуры БД. Таким образом, отображение пользовательского представления данных, оформленного в виде входного документа произвольного формата, в структуру данных БД составляет главную задачу ввода.

Разнообразие форм входных документов отвечает разнообразию структур представленных в них данных. В противоположность распространенной точке зрения о представимости входных документов в виде реляционных таблиц, следует отметить, что документы, как правило, имеют более сложные структуры данных. В них встречаются такие недопустимые в реляционных таблицах элементы, как заголовки, промежуточные заголовки и итоги, повторяющиеся группы данных и т. д.

В общем случае средства интерфейса должны быть рассчитаны на структуры входных данных достаточно общего вида. На рис. 3.1а приводится пример простого входного документа, структура которого изображена на рис. 3.1б в соответствии с обозначениями, введенными в гл. 1 (названия предприятий условные).

Основная задача отображения структуры данных документа в структуру данных БД представляет собой крупным планом задачу реструктуризации. При этом возможны различные соотношения структур.

Отображение документа, не содержащего повторяющихся групп данных, в любую структуру данных БД не представляет труда: каждый элемент документа отображается в однозначно определенный элемент БД (см. пример на рис. 3.2а). Аналогично не вносит принципиальных трудностей наличие в документе повторяющихся групп данных, если их подчиненность другим данным документа сохраняется при отображении в БД (см. пример на рис. 3.2б). В этих случаях структура данных документа не противоречит иерархии БД, и задача ввода сводится к выделению подсхемы БД, соответствующей данным документа.

Собственно реструктуризация связана с изменением при вводе иерархии данных, когда подчиненность массивов данных в документе и в базе не совпадает (см. рис. 3.2в). Основная идея реализации такого отображения заключается в автоматической линеаризации документа в процессе ввода, т. е. превращении его в совокупность документов-строк. Впоследствии строки отображаются в БД по указанной пользователем подсхеме. На рис. 3.2в изображена соответствующая схема преобразования исходного документа.

В рамках ИНЕС описание поддерева БД, в которое производится ввод, линеаризация входного документа и отображение данных документа в БД задаются в терминах специального языка описания компонент (ЯОК). Ниже описывается применение этого языка и других компонент системы макетного ввода ИНЕС (СМВ). В целом эта система [23, 36] предназначена для массового ввода и корректи-

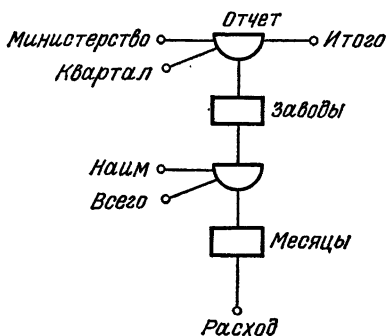
ровки данных в БД. Как показал опыт, практически любой используемый на практике документ может быть введен в БД средствами СМВ без дополнительного программирования.

*Квартальный отчет
Расход энергоресурсов отрасли по заводам
Министерство: МИНХИМПРОМ
Квартал: 1*

<i>Завод</i>	<i>Всего</i>	<i>Расход по месяцам</i>
<i>Луч</i>	<i>366</i>	<i>111</i>
		<i>237</i>
		<i>18</i>
<i>Красное знамя</i>	<i>395</i>	<i>107</i>
		<i>251</i>
		<i>37</i>
<i>Вымпел</i>	<i>325</i>	<i>93</i>
		<i>207</i>
		<i>25</i>

Итого: 1086

а



б

Рис. 3.1

3.0.2. Средства ввода ИНЕС. После того как база описана на языке ЯОД, текст описания оттранслирован и описание базы представлено в виде описания данных (ДОД), с помощью специальных средств ввода можно заполнять базу необходимыми данными. Наполнение базы приводит к образованию собственно базы данных (ДД). Средствами ввода можно также корректировать, дополнять или удалять информацию, хранящуюся в базе.

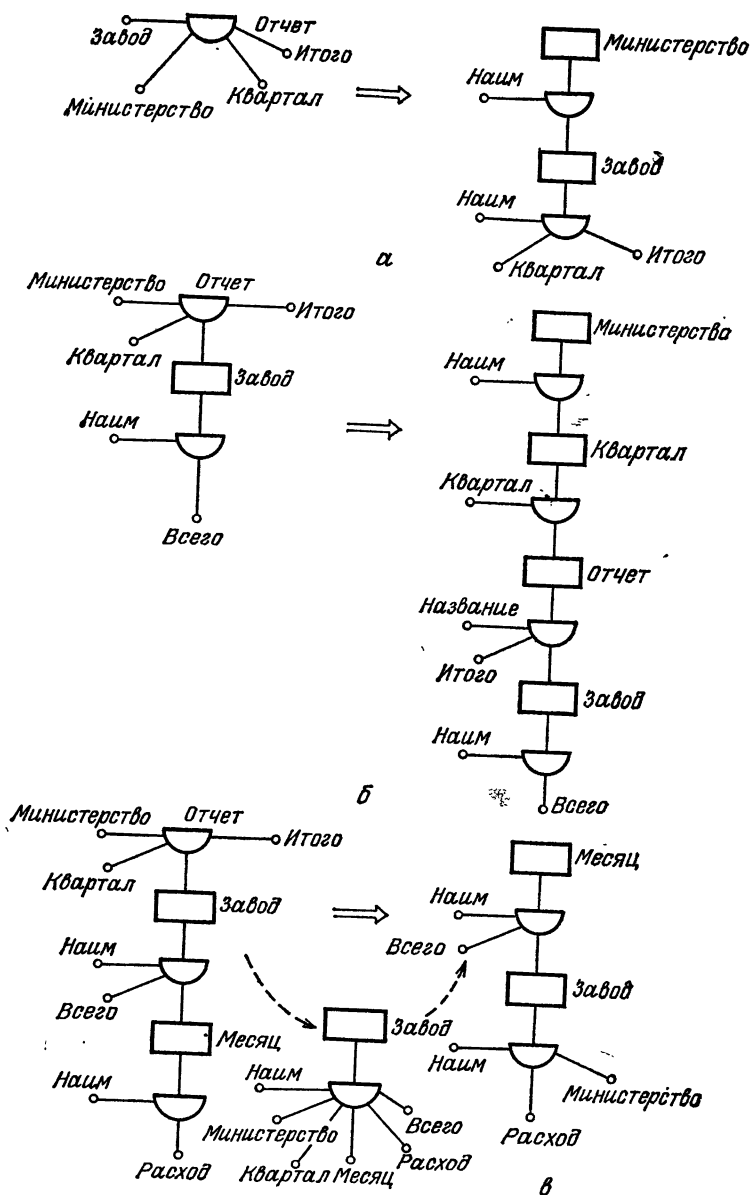


Рис. 3.2

Как упоминалось выше, для ввода в базу информация должна быть извлечена из документов пользователей и представлена в ваданном виде. В диалоговых дисплейных системах входная информация может быть представлена на экране в виде той или иной формы документа, ориентированной на пользовательское представление структуры данных [53]. В дисплейных системах ввода существенную роль играет макет (пустографка) документа. На его основе системой высвечивается форма документа, которая разворачивается на экране по мере ее заполнения [12]. В режиме пакетного ввода подготавливается и вводится фактически только изменяемая часть (содержание) документа. Входная информация может быть также подготовлена на внешних носителях какой-либо программой. Во всех случаях при вводе содержательная часть документа должна быть отделена от его оформления.

Ввод подготовленных машинных документов в СМВ ИНЕС производится в три этапа (см. рис. 3.3):

- ввод документа с внешнего носителя в оперативную память с помощью программ-анализаторов;
- контроль документа;
- запись информации из оперативной памяти в БД с помощью интерпретатора СМВ.

Вводимый средствами СМВ документ должен быть разбит на отдельные окна. Каждое окно документа имеет номер и содержит один или несколько реквизитов (в последнем случае каждый реквизит должен занимать определенные позиции в окне). Часть окон в документе может отсутствовать. В нем может быть также несколько окон, содержащих значения одних и тех же реквизитов в нескольких экземплярах (одни и те же характеристики разных объектов).

Окна из документов выделяются программами-анализаторами после ввода документов в оперативную память. В ИНЕС имеются три анализатора: анализатор CD — для ввода документов фиксированного формата [36], анализатор WC — для ввода документов с разделителями (см. п. 3.4.2) и ПЛ-анализатор — для ввода документов с описанием их формата в языке ПЛ/1 [20, 36]. При работе с анализатором CD окном является запись (в смысле ОС ЕС), поэтому каждый реквизит должен содержаться в определенных позициях записи. При вводе документов анализатором WC позиции, в которых находится окно, не играют существенной роли. Одно окно отделяется от другого специальным символом-разделителем. Для работы с ПЛ-анализатором формат документа описывается специальным оператором DOC.

Все анализаторы после ввода и анализа документов обращаются к средствам контроля (если это требуется) и затем передают управление интерпретатору СМВ для записи данных в БД.

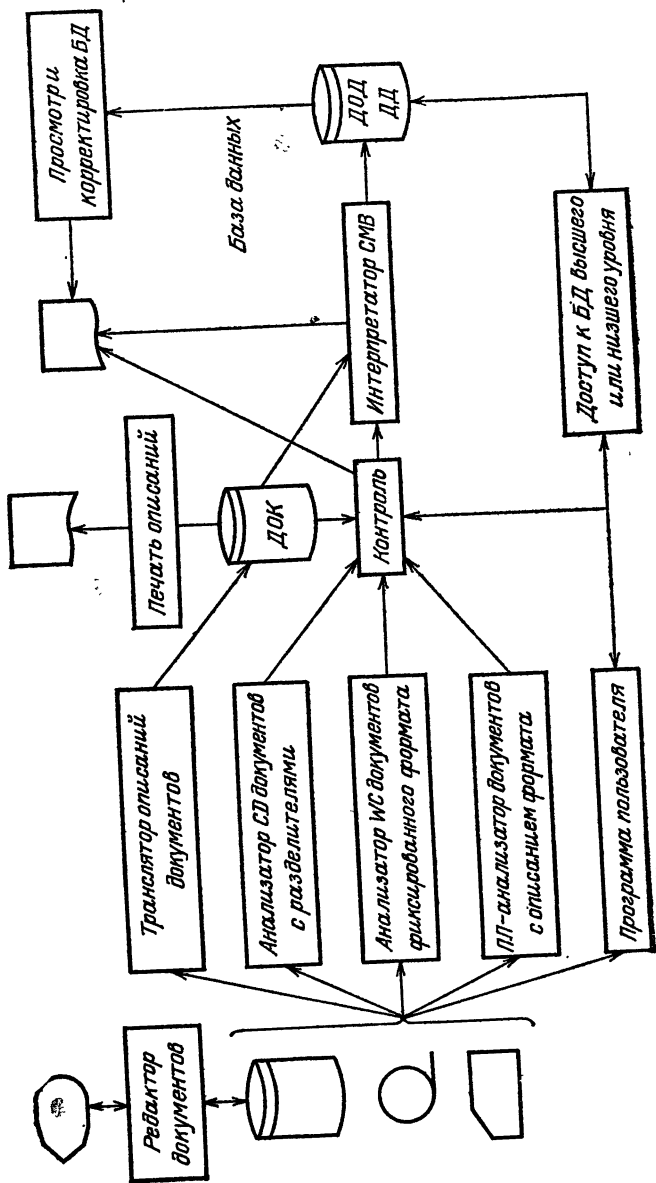


Рис. 3.3

Для контроля вводимых документов в составе ИНЕС имеется стандартный блок контроля A2EN. Он позволяет проверить содержимое окна на равенство или неравенство (больше, меньше) содержимому другого окна или константе, а также проверить контрольные суммы и установить, принадлежит ли указанное в окне данное списку допустимых значений. Программирование контроля, построенное по типу «заполни пустые места», легко усваивается и может быть передано конечному пользователю (см. описание контроля в [36, 60]). Стандартный блок контроля можно использовать совместно с программами контроля, созданными пользователями ИНЕС.

Интерпретатор СМВ предназначен для передачи данных из оперативной памяти в БД. Он используется анализаторами, а также может применяться для ввода в БД данных, полученных программным путем. Исходными данными для интерпретатора СМВ служат обработанный анализатором документ и описание отображения (дерево описания компонент), которое указывает, в какое место БД нужно ввести тот или иной реквизит из документа. Отображение данных описывается на языке описания компонент (см. пп. 3.2, 3.3) и является общим для всех анализаторов. Это описание необходимо оттранслировать, результат трансляции представляет собой *дерево описания компонент* (ДОК).

В тех случаях, когда пользователя не устраивает СМВ, он может создавать свои средства ввода.

Готовую базу данных можно проверить и откорректировать с экрана дисплея с помощью специальной диалоговой системы просмотра и корректировки БД ([19, 42]).

8.1. Основные понятия

8.1.1. Документы и окна документов. Исходные данные для заполнения базы подготавливаются пользователем в виде входных документов, имеющих, вообще говоря, произвольный формат. Для ввода в БД в документе должно быть выделено его содержание (совокупность данных) и задано отображение данных документа в структуры данных БД. Как указывалось выше (см. п. 2.3 Введения), здесь содержание документа противопоставляется его оформлению (макету).

Содержание идентифицируется в документах с помощью *окон* — фиксированных позиций макета, предназначенных для размещения переменной информации. Другими словами, окна — это те места в бланках (макетах) документов, которые заполняются при создании конкретного экземпляра документа. На рис. 3.4 показан пример макета входного документа для ввода в базу, описанную в гл. I (рис. 1.18). Окна в этом примере выделены точками.

Окна входного документа идентифицируются своими номерами. Номер окна не обязан быть порядковым номером в документе, он скорее определяет тип информации, для которой данное окно предназначено.

АНКЕТА		
1. ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО		
2. ДОЛЖНОСТЬ		
3. ДАТА РОЖДЕНИЯ 4. ПОЛ		
5. СПЕЦИАЛЬНОСТЬ		
СВЕДЕНИЯ ОБ ОБРАЗОВАНИИ		
ВЫСШЕЕ	НЕВЫСШЕЕ	
6. ВУЗ	9. ЧТО ОКОНЧИЛ (А)	
7. ДАТА ОКОНЧАНИЯ	10. ДАТА ОКОНЧАНИЯ	
8. АДРЕС		
.....		
ТРУДОВАЯ ДЕЯТЕЛЬНОСТЬ		
НАИМЕНОВАНИЕ ПРЕДПРИЯТИЯ	ДАТА ПОСТУПЛЕНИЯ	ДОЛЖНОСТЬ
11	12	13
.....
.....
.....
14. НАГРАДЫ		
.....		
.....		
.....		
15. ЦЕХ		
16. ОКЛАД		

Рис. 3.4

Для каждого макета должно быть подготовлено описание отображения данного документа в БД — разбора значений из окон макета. Различные виды таких отображений описываются в п. 3.2.

Перед вводом в ЭВМ совокупность окон документа представляется в определенном формате на перфокартах или других внешних

носителях (см. описание подготовки входных данных в п. 3.4.2). Может использоваться, например, раздел библиотеки исходных текстов или последовательный набор данных, созданный какой-либо программой. В соответствии с заданным описанием входного документа (описанием отображения документа в БД) содержимое окон при вводе автоматически распределяется по вершинам ДД.

Входной документ Дерево базы

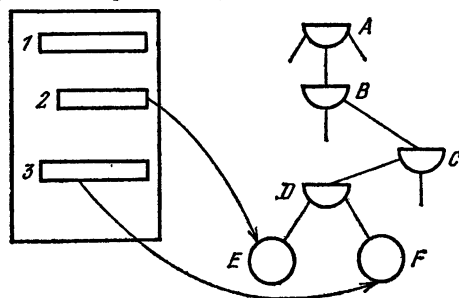


Рис. 3.5

3.1.2. Соответствие окон вершинам базы данных. Соответствие окон терминальным вершинам базы данных устанавливается так называемым описанием компонент ввода. Значения из этих окон будут присвоены в процессе ввода вершинам БД. Это описание составляется на *языке описания компонент* (ЯОК). Результатом трансляции является дерево описания компонент.

Кроме соответствия окон документа терминальным вершинам базы, описание компонент задает режимы ввода: создание новой вершины, изменение значения вершины, добавление к значению вершины, удаление вершины и другие.

Соответствие окон документа тем вершинам дерева, в которые будет перенесена содержащаяся в них информация, задается так:

траектория дерева = номер окна

Здесь слева от знака равенства указывается терминальная вершина дерева в виде траектории, справа — номер окна входного документа. Траектория — цепочка имен, разделенных точками, отличается от составного имени вершины в ДОД (см. п. 1.4) тем, что вместо имени элемента массива указывается идентификатор конкретного элемента массива (см. пп. 3.2.2—3.2.4).

Пример. Пусть соответствие окон входного документа вершинам дерева базы установлено так, как показано на рис. 3.5. Тогда в описании компонент это соответствие может быть таким:

A.B.C.D.E=2

A.B.C.D.F=3

3.1.3. Уровни в описании компонент ввода. Как видно из предыдущего примера, траектории, определяющие различные вершины ДД, могут иметь общие начальные части. Чтобы эти общие части не повторять для каждой вершины, в ЯОК введено понятие уровня.

На уровне 01 выписывается общая начальная часть всех траекторий данного описания компонент. Часть траектории, описанная на уровне n , является продолжением траектории непосредственно предшествующего уровня $n-1$.

Например, траектории предыдущего примера на языке ЯОК могут быть описаны в виде:

01 A.B.C.D.

02 E=2

02 F=3

Применение уровней приводит к наиболее короткому описанию совокупности траекторий. В принципе описание компонент ввода информации в дерево базы можно составить так, чтобы ни одно имя дерева не повторялось. Компактное описание удобно для пользователя и оптимально для ЭВМ, так как задает кратчайший путь движения по базе, что приводит к минимальному времени загрузки информации.

Если не все траектории ДОК имеют общую начальную часть, то уровень 01 должен быть пустым, т. е. строка, помеченная номером 01, не должна содержать ничего, кроме этого номера. Такая ситуация возникает, когда в базе, состоящей более чем из одного дерева, одни и те же входные документы заполняют разные деревья базы.

П р и м е р.

01

02 A.B.C=1

02 X.Y.Z=2

Уровень с номером n не обязательно указывает на разветвление траектории уровня $n-1$ (как говорилось, он указывает только на продолжение траектории), т. е. может быть только один уровень с номером n , непосредственно подчиненный некоторому уровню с номером $n-1$.

П р и м е р. Один из предыдущих примеров может быть представлен в виде:

01 A.B.

02 C.D.

03 E=2

03 F=3

Здесь уровень 03 указывает на разветвление траектории, а уровень 02 — нет.

3.2. Описание отображения входного документа

3.2.1. Заполнение веера терминальных вершин. Часто несколько терминальных вершин составляют одно структурное данное (*веер*). В этом случае можно для терминальных вершин не выделять отдельного уровня, а перечислять последние элементы траектории вместе с соответствующими окнами через запятую.

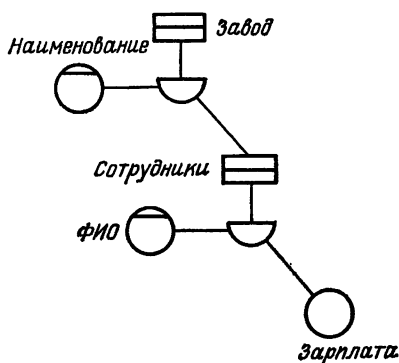


Рис. 3.6

Например, следующий фрагмент описания компонент

04 ИЗДЕЛИЕ
05 НАИМЕНОВАНИЕ=9
05 ВЕС=10
05 ЦЕНА=11

может быть записан короче:

04 ИЗДЕЛИЕ. НАИМЕНОВАНИЕ=9, ВЕС=10, ЦЕНА=11

3.2.2. Ввод в ключевой массив. В соответствии с определением траектории при проходе через элементы ключевого массива нужно указывать значение ключа (а не вершины ДОД). Значения всех необходимых ключей должны быть получены из входного документа или заданы априорно. Если значение ключа находится в окне с номером w , то соответствующая компонента траектории задается в виде $\#w$. Например, фрагмент базы данных на рис. 3.6 заполняется по макету, изображенному на рис. 3.4, в соответствии со следующим ДОК:

01 ЗАВОД. #15. СОТРУДНИКИ. #1. ЗАРПЛАТА=16

Возможна ситуация, когда значение ключа известно заранее. Тогда это значение в траектории записывается явно. Например:

01 ЗАВОД.АЗЛК.СОТРУДНИКИ.#1.ЗАРПЛАТА=16

Если в значении ключа при его явном указании в траектории содержатся символы, отличные от букв, пробела и цифр, то это значение надо заключить в апострофы. Например:

01 ЗАВОД.'А-17'.СОТРУДНИКИ.'ЦОЙ Ю. Н.'.
ЗАРПЛАТА=16 *)

3.2.3. Ввод в нумерованный массив. При вводе в нумерованный массив можно производить запись информации по заданному номеру элемента массива, или добавлять новые элементы в конец

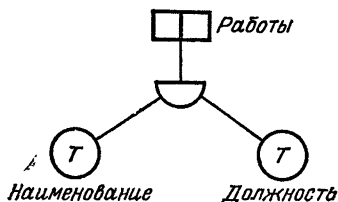


Рис. 3.7

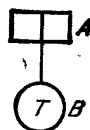


Рис. 3.8

массива с их автоматической нумерацией, или корректировать последний элемент.

В первом случае номер элемента может быть задан в траектории аналогично значению ключа при вводе в ключевой массив (см. п. 3.2.2). Например:

05 ПУБЛИКАЦИИ.#10.

06 НАИМЕНОВАНИЕ=11, ИЗДАТЕЛЬСТВО=12, ГОД=13

Здесь предполагается, что в десятом окне входного документа находится номер элемента нумерованного массива ПУБЛИКАЦИИ.

Во втором случае, при добавлении элементов в конец массива, компонента траектории, определяющая новый элемент, имеет следующий вид:

#0<d>/A/

где d обозначает шаг автоматической нумерации.

П р и м е р. Описание компонент для фрагмента базы данных, представленного на рис. 3.7, имеет вид:

07 РАБОТЫ.#0<1>/A/.НАИМЕНОВАНИЕ=11,
ДОЛЖНОСТЬ=13

*) Строки без номера уровня в гл. 3 являются продолжением предыдущих.

Указание /A/ означает, что вводимый элемент будет добавлен к массиву. Если это указание опустить, то запись произведется в последний элемент массива.

Если элемент нумерованного массива является терминальной вершиной, то имя этой вершины в траектории не указывается.

Например:

05 A.#0<1>/A/. = 10

Этот фрагмент описания компонент соответствует фрагменту базы, представленному на рис. 3.8.

3.2.4. Ввод в простой массив. При вводе в простой массив можно добавлять новые элементы в конец массива или исправлять последний элемент.

Компонента траектории, определяющая вводимый элемент массива, должна иметь вид:

#0/A/ или #0

Смысл указания /A/ тот же, что и для нумерованных массивов (см. п. 3.2.3).

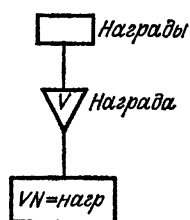


Рис. 3.9

Пример. Для заполнения фрагмента базы рис. 3.9 можно использовать следующий фрагмент ДОК:

06 НАГРАДЫ.#0/A/. = 14

Здесь, так же как и для нумерованного массива, имя терминальной вершины опущено.

3.2.5. Повторы окон. В системе ввода ИНЕС имеется возможность ввести сразу много элементов одного и того же массива из одного входного документа. В этом случае документ содержит повторы одних и тех же номеров окон. Например, в документе, представленном на рис. 3.4, имеются повторы окон с номерами 11—13 (описывающих работы) и окон с номером 14 (описывающих награды). Если в документе имеются повторы окон, то в компонентах траекторий ДОК нужно указать границы номеров группы повторяющихся окон в виде:

(р, q)

где р — минимальный, а q — максимальный номер окна в группе повторяющихся окон, описывающих элемент заполняемого массива БД.

Компоненты траектории ДОК с повторами описываются в следующем формате:

$\#w(p,q)$ — для ключевых или нумерованных массивов,
 $\#0 < d > (p,q) [/A/]$ — для нумерованных массивов,
 $\#0(p,q) [/A/]$ — для простых массивов.

Примеры.

1. Заполнение массивов РАБОТЫ и НАГРАДЫ базы данных рис. 1.18 по макету входного документа рис. 3.4 описывается следующим образом:

07 РАБОТЫ. $\#0 < 1 > (11,13) [/A/]$. НАИМЕНОВАНИЕ=11,
 ДОЛЖНОСТЬ=13
 07 НАГРАДЫ. $\#0(14,14) [/A/]$. =14

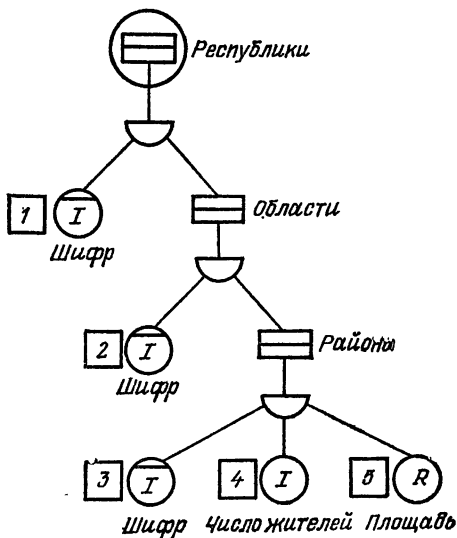


Рис. 3.10

2. Для ввода в базу данных рис. 3.10 информации из документа рис. 3.11 можно использовать следующее описание:

01 РЕСПУБЛИКИ. $\#1(1,5)$
 02 ОБЛАСТИ. $\#2(2,5)$
 03 РАЙОНЫ. $\#3(3,5)$
 04 ЧИСЛО ЖИТЕЛЕЙ=4
 04 ПЛОЩАДЬ=5

3.2.6. Части окон. Иногда бывает необходимо при занесении информации в базу не целиком брать значение окна документа, а выделить только часть окна. Часть окна выделяется указанием номера символа, с которого начинается часть, и числа символов,

СПИСОК РАЙОНОВ

Шифр республики	Шифр области	Шифр района	Число жителей	Площадь
1	2	3	4	5
01	50	101	1.2	12.3
		112	2.3	10.8
		117	1.7	15.2
	56	183	0.9	9.4
		191	5.6	13.5
03	78	215	3.1	17.3

Рис. 3.11

образующих эту часть. Это указание в описании компонент ставится непосредственно после номера окна в виде:

$w < p, r >$ или $< p >$

Здесь w обозначает номер окна, p — номер первого символа части окна, r — число символов части окна.

Если r отсутствует, то выбирается часть окна от символа с номером p до конца окна.

Примеры.

1. Пусть дата, состоящая из года, месяца и числа, содержится в одном окне с номером 3 в виде: ДД.ММ.ГГГГ, например 08.03.1962.

Такая дата заносится во фрагмент базы (рис. 3.12) при помощи следующего описания:

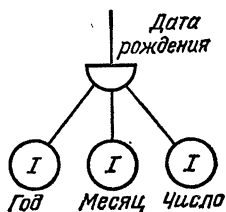


Рис. 3.12

05 ДАТА РОЖДЕНИЯ.ГОД=
3<7>, МЕСЯЦ=3<4,2>,
ЧИСЛО=3<1,2>

2. Пусть в окне номер 5 содержится шифр предприятия, состоящий из 2-значного номера главка и 3-значного номера предприятия в главке.

Этот шифр можно занести в БД при помощи следующего ДОК:

01 ГЛАВКИ.#5<1,2>(5,10)
02 ПРЕДПРИЯТИЯ.#5<3,3>

3.2.7. Накопление сумм. В терминальные вершины базы можно заносить числовую информацию из окон в режиме суммирования с уже присвоенными значениями терминальных вершин. В этих случаях в описании компонент вместо знака равенства используются

знаки «+» или «—». При первом занесении информации значение терминальной вершины считается равным нулю.

После знаков «+» или «—» вместо номера окна может стоять число, заключенное в кавычки. В этом случае в соответствующую терминальную вершину будет прибавляться (или вычитаться) указанное число.

Пр и м е р ы.

1. При заполнении фрагмента базы рис. 3.13 используется следующее описание:

01 ЗАВОД. #15. ФОНД ЗАРПЛАТЫ +16

Здесь при вводе информации об очередном сотруднике его зарплата, указанная в окне 16, прибавится к фонду зарплаты цеха.

2. Описание

05 A.B+ '3.14', C— '1'

означает, что к терминальной вершине В прибавится число 3.14, а из вершины С вычитается 1.

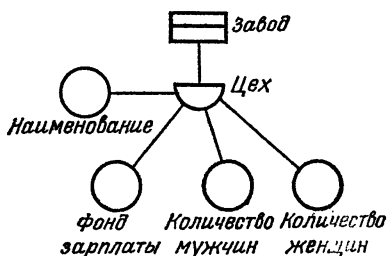


Рис. 3.13

3.2.8. Условия выполнения уровней. Заполнение базы может меняться в зависимости от значения тех или иных окон документа. Для этого в описании компонент используются условия выполнения уровней.

Условия записываются сразу после номера уровня в одном из четырех видов:

n /w/

n /w 7/

n /w=текст/

n /w 7=текст/

где n — номер уровня, w — номер окна или часть окна.

Условие 1-го вида считается выполненным, если во входном документе есть окно с номером w или часть окна w содержит символ, отличный от пробела.

Условие 2-го вида является отрицанием условия 1-го вида.

Условие 3-го вида считается выполненным, если значение в окне с номером w или в соответствующей части окна равно указанному в условии тексту.

Условие 4-го вида — отрицание условия 3-го вида.

Если текст содержит символы, отличные от букв и цифр, его нужно заключить в апострофы.

Если условие на заданном уровне ДОК не выполняется для документа, то этот уровень и все ему подчиненные для данного документа игнорируются.

П р и м е р ы.

1. При заполнении базы рис. 3.13 используется следующее описание:

01 ЗАВОД.#15.

02 ФОНД ЗАРПЛАТЫ+16

02 /5=ЖЕН/ КОЛИЧЕСТВО ЖЕНЩИН+'1'

02 /5=МУЖ/ КОЛИЧЕСТВО МУЖЧИН+'1'

Здесь предполагается, что в окне 5 указан пол сотрудника в виде «МУЖ» или «ЖЕН».

2. По описанию

03 /10=РСФСР/

04 /11=МИНВУЗ/ НИС=15

в вершину НИС будет занесено значение из окна 15 при условии, что значение окна 10 равно «РСФСР», а окна 11 — «МИНВУЗ».

3.

05 /20=X/ A=10

05 /20=Y/ A=10

Здесь в вершину А занесется значение из окна 10, если в окне 20 будет X или Y.

3.2.9. Организация сетевых связей (ссылок на значения). Как отмечалось в п. 1.4, при описании данных только устанавливается наличие сетевых связей (ссылок на значение), фактическая же организация ссылок производится в процессе ввода данных.

Описание ссылки, соответствующей вершине типа REF, в описании компонент может иметь один из двух видов:

имя=(траектория) или имя=(s)

Здесь «имя» означает имя вершины типа REF, «траектория» — это траектория, ведущая от корня базы данных к той вершине, на которую организуется ссылка. s — это метка в траектории ДОК, уже пройденная в процессе ввода данного документа.

П р и м е р. Для организации в базе рис. 1.18 ссылки из вершины АНКЕТНЫЕ СВЕДЕНИЯ на вершину СОТРУДНИК существуют два способа:

1)

01

02 ЗАВОД.#15.СОТРУДНИКИ.#1

....

02 ШТАТНОЕ РАСПИСАНИЕ.#2

03 СОТРУДНИКИ. #1

04 АНКЕТНЫЕ СВЕДЕНИЯ=(ЗАВОД. #15.
СОТРУДНИКИ. #1)

2)

01

02 ЗАВОД. #15. СОТРУДНИКИ. #1. (1)

02 ШТАТНОЕ РАСПИСАНИЕ. #2

03 СОТРУДНИКИ. #1

04 АНКЕТНЫЕ СВЕДЕНИЯ=(1)

Отметим, что второй способ эффективнее, так как движение в базе данных по траектории ЗАВОД—СОТРУДНИК совершается в этом случае один раз (в первом случае — два).

3.2.10. Шаблоны. Подобно тому, как в ЯОД ссылка на шаблон дает возможность сокращать описание данных и описывать потенциально бесконечные деревья (см. пп. 1.4, 2.5), в ЯОК аналогичное средство — шаблон — позволяет избавляться от повторений в описании компонент и дает возможность загружать потенциально бесконечные деревья.

Шаблон представляет собой выделенную часть описания компонент, которая подставляется транслятором в те места основного описания компонент, где указаны обращения к шаблону.

В шаблоне некоторые числовые параметры (например, номер окна, границы повторности и т. п.) могут быть указаны как формальные параметры в виде:

@целое

Фактическое значение таких параметров получается сложением числа, указанного при обращении к шаблону в виде

□имя шаблона(число)

с числом, стоящим после знака @.

Обращение к шаблону может иметь также вид:

□имя шаблона (начальное значение, шаг, конечное значение)

В этом случае указанный шаблон подставляется многократно с изменением формальных параметров от начального значения с указанным шагом до конечного значения.

Описание шаблона ставится в начале описания компонент и начинается с уровня 01, помеченного одним или двумя символами.

Продemonстрируем применение шаблонов на примерах.

Пример 1. Пусть поддерево БД устроено так, как показано на рис. 3.14. Заполнение поддерева осуществляется, например, при помощи следующего ДОК:

n КОЛИЧЕСТВО

n+1 /5=1/ КВАРТАЛ1.СОРТ1=10,СОРТ2=11,СОРТ3=12

n+1 /5=2/ КВАРТАЛ2.СОРТ1=10,СОРТ2=11,СОРТ3=12

n+1 /5=3/ КВАРТАЛ3.СОРТ1=10,СОРТ2=11,СОРТ3=12

n+1 /5=4/ КВАРТАЛ4.СОРТ1=10,СОРТ2=11,СОРТ3=12

Здесь в зависимости от значения 5-го окна информация из окон 10, 11 и 12 заносится в веер терминальных вершин, соответствующих одному из четырех кварталов. Сократить это описание

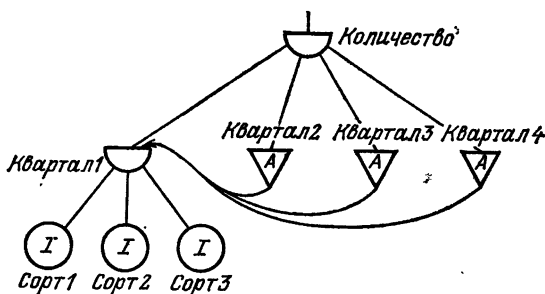


Рис. 3.14

можно, выделив повторяющуюся часть в шаблон:

ШК 01 СОРТ1=10,СОРТ2=11,СОРТ3=12

01

....

n КОЛИЧЕСТВО

n+1 /5=1/ КВАРТАЛ1. ☐ ШК

n+1 /5=2/ КВАРТАЛ2. ☐ ШК

n+1 /5=3/ КВАРТАЛ3. ☐ ШК

n+1 /5=4/ КВАРТАЛ4. ☐ ШК

Значок ☐ вместе со следующими за ним буквами означает, что на это место транслятор подставит описание шаблона, помеченное указанными буквами, увеличивая при этом номера всех уровней шаблона на величину номера уровня, в который производится подстановка.

Таким образом, описание компонент преобразуется транслятором в следующее описание (эквивалентное первоначальному):

01

....

n КОЛИЧЕСТВО

n+1 /5=1/ КВАРТАЛ1.

n+2 СОРТ1=10,СОРТ2=11,СОРТ3=12

n+1 /5=2/ КВАРТАЛ2.

n+2 СОРТ1=10,СОРТ2=11,СОРТ3=12
.... и т. д.

Пример 2. Предположим, что для той же части базы, которая показана на рис. 3.14, мы хотим заносить информацию не из одних и тех же окон документа, как в предыдущем примере, а для каждого квартала из своих окон, в соответствии со следующим описанием:

n КОЛИЧЕСТВО

n+1 КВАРТАЛ1.СОРТ1=10,СОРТ2=11,СОРТ3=12
n+1 КВАРТАЛ2.СОРТ1=13,СОРТ2=14,СОРТ3=15
n+1 КВАРТАЛ3.СОРТ1=16,СОРТ2=17,СОРТ3=18
n+1 КВАРТАЛ4.СОРТ1=19,СОРТ2=20,СОРТ3=21

Это описание можно представить по-другому, используя формальные параметры в шаблоне.

ШК 01 СОРТ1=@0,СОРТ2=@1,СОРТ3=@2
01

.....

n КОЛИЧЕСТВО

n+1 КВАРТАЛ1. ☐ ШК(10)
n+1 КВАРТАЛ2. ☐ ШК(13)
n+1 КВАРТАЛ3. ☐ ШК(16)
n+1 КВАРТАЛ4. ☐ ШК(19)

Здесь значок @ означает, что следующее за ним число будет увеличено транслятором на число, указанное в скобках в обращении к шаблону. Так, например, вместо @1 будет подставлено 11 при замене первого обращения к шаблону, 14 — при замене второго обращения и т. д. После всех подстановок получится описание, эквивалентное исходному (как и в предыдущем примере, появится дополнительный уровень).

Пример 3. Пусть документы, которые вводятся в базу, устроены так, что последовательность окон каждого документа содержит информацию следующего вида:

Заказчик, адрес, заказчик, адрес и т. д.

То есть в окнах с нечетными номерами содержится информация о том, кто является заказчиком, а в окнах с четными номерами — адреса заказчиков.

Если в каждом документе подготовлена информация не более чем о 10 заказчиках, то вводить такие документы в соответствующую базу можно, например, при помощи следующего описания компонент:

01 ЗАКАЗЧИКИ.
02 #1.АДРЕС=2

02 #3.АДРЕС=4

...

02 #19.АДРЕС=20

Это описание громоздко. Оно может быть представлено значительно короче при помощи шаблона, обращение к которому предусматривает изменение формальных параметров с заданным шагом:

ШЗ 01#@0.АДРЕС=@1

01 ЗАКАЗЧИКИ. □ШЗ(1,2,19)

Здесь в обращении к шаблону в скобках стоят три числа. Это значит, что шаблон будет вставлен на место обращения к шаблону в нескольких экземплярах. В каждом экземпляре будут изменяться числа, помеченные знаком @. В первом экземпляре подстановка вместо чисел со знаком @ произойдет так, как при обращении к шаблону □ ШЗ(1), во втором — как при обращении □ ШЗ(3), а в последнем экземпляре — как при обращении □ ШЗ(19). Другими

словами, первое из трех чисел, указанных в скобках в обращении к шаблону, означает добавку, на которую изменятся числа со знаком @ при подстановке 1-го экземпляра шаблона, второе число означает шаг изменения этой добавки при подстановке следующих экземпляров, а третье число — максимальное значение этой добавки при подстановке последнего экземпляра шаблона.

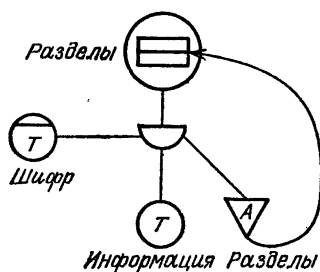


Рис. 3.15

В одном описании компонент ввода документов может быть несколько шаблонов, причем в самом описании шаблона может быть обращение к другому шаблону. Более того, описание шаблона может содержать обращение к самому себе, как это показано в следующем примере.

Пример 4. В главе 1 говорилось о том, что дерево описания данных может иметь неограниченную длину, быть потенциально бесконечным (см. пп. 1.4, 2.5). Структура такого вида изображена на рис. 3.15.

Пусть документ для загрузки такой БД представляет информацию в следующем виде:

Шифр раздела, информация раздела, шифр подраздела, информация подраздела, ... и т. д.

В окнах с нечетными номерами заданы шифры подчиненных друг другу разделов, в окнах с четными номерами — информация о разделах.

Описание компонент ввода таких документов в указанную базу имеет следующий вид:

ШР 01 РАЗДЕЛЫ.#@0.ИНФОРМАЦИЯ=@1, □ ШР(@2)
01 □ ШР(1)

Здесь используется рекуррентное обращение шаблона к самому себе, позволяющее загружать потенциально бесконечные деревья.

3.2.11. Режимы. При каждой компоненте в описании компонент ввода документов может быть указан режим прохождения соответствующей вершины базы данных при ее загрузке.

Имеются следующие режимы:

- U — обновление,
- R — чтение,
- W — запись,
- D — удаление,
- E — удаление с индикацией ошибки,
- A — добавление,
- X — замена,
- S — перебор.

Режим указывается одной из перечисленных букв, заключенной в наклонные черточки.

Режим обновления (U) означает, что если соответствующая вершина в дереве данных существует, то происходит движение в эту вершину (для терминальной вершины с обновлением в ней информации). Если соответствующей вершины нет, то эта вершина создается и в нее осуществляется движение.

В *режиме чтения (R)* происходит движение в вершину, если она существует. Если вершины нет, то выдается сообщение об ошибке.

В *режиме вписи (W)* создается соответствующая вершина и в нее производится движение. Если соответствующая вершина уже существует, то выдается сообщение об ошибке.

При движении в вершину в режимах D или E происходит уничтожение этой вершины со всеми ей подчиненными. В режиме E отсутствие соответствующей вершины приводит к сообщению об ошибке.

Режим A описывается в пп. 3.2.3, 3.2.4.

Если режим не указан, то по умолчанию предполагается режим U, кроме тех траекторий, которые организуют ссылки (см. п. 3.2.9). В этих траекториях по умолчанию принимается режим R.

Если после буквы, обозначающей режим, стоит символ «!», то в случае ошибки, вызванной данным режимом, все следующие компоненты для вводимого документа не выполняются.

Печать сообщения об ошибках можно подавить, указав после буквы режима символ «*».

Пример 1. Пусть документ увольнения сотрудника содержит в 1-м окне ФИО, во 2-м окне название цеха и в 3-м — зарплату увольняемого сотрудника. Тогда описание компонент для ввода такого документа в БД рис. 1.18 имеет вид:

01 ЗАВОД.#2
02 СОТРУДНИКИ.#1/E/
02 ФОНД ЗАРПЛАТЫ—3

Если бы в этом примере после буквы Е не был указан символ «!», то при отсутствии в базе сотрудника с указанной в 1-м окне фамилией было бы выдано сообщение об ошибке, но тем не менее из фонда зарплаты вычлась бы зарплата, указанная в 3-м окне.

Пример 2. Пусть входной документ в первых трех окнах содержит: ФИО, название цеха и название вуза, который окончил сотрудник. Такой документ может быть введен в базу рис. 1.18 при помощи следующего описания компонент:

01 ВУЗЫ.#3.СОТРУДНИКИ.#1.
02 АНКЕТНЫЕ СВЕДЕНИЯ=(ЗАВОД/U/.#2.
СОТРУДНИКИ.#1)

Здесь данные вводятся в дерево ВУЗЫ, причем организуется ссылка на анкетные сведения заданного сотрудника в дереве ЗАВОД. Если в дереве ЗАВОД отсутствует указанный в документе сотрудник, то его фамилия, имя и отчество будут введены в это дерево, так как в траектории ссылки указан режим U. Если бы использовался режим по умолчанию R, то при отсутствии в дереве ЗАВОД соответствующего сотрудника ссылка бы не организовывалась.

Режим замены (X) означает, что для терминальных вершин режим X работает как режим U, а для нетерминальных вершин — сначала из ДД удаляется вершина со всеми ее подчиненными, а затем производится ввод данных (в соответствии с ДОД).

Режим перебора (S) применяется для организации перебора повторностей документа без движения по базе данных. Используется в тех случаях, когда нужно сначала произвести линеаризацию документа, а затем осуществлять ввод в базу данных (см. п. 3.2.12).

3.2.12. Линеаризация входного документа. Линеаризация документа необходима в тех случаях, когда иерархия данных входного документа не соответствует иерархии БД. Если в документе нет повторяющихся окон, то он соответствует любой БД, так как можно выбирать из такого документа значения окон в порядке иерархии БД. Если же документ имеет повторности окон и иерархия документа отличается от иерархии БД, то нельзя осуществить ввод рассмотренными ранее средствами. Действительно, компоненты $k(p,q)$, $0(p,q)/A/$ и $0 < s > (p,q)/A/$ выполняют сразу две функции:

1) движение по БД (по ключу или в следующий элемент массива, см. п. 3.2.5);

2) организация перебора повторностей окон входного документа.

Если иерархия входного документа не соответствует иерархии БД, то нельзя совмещать эти функции, необходимо выполнять их

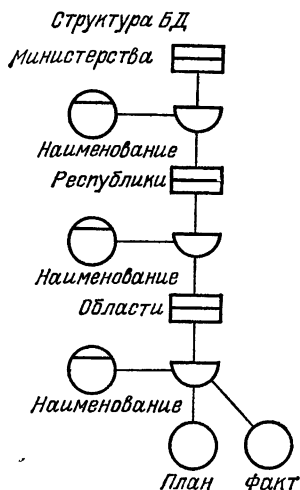


Рис. 3.16

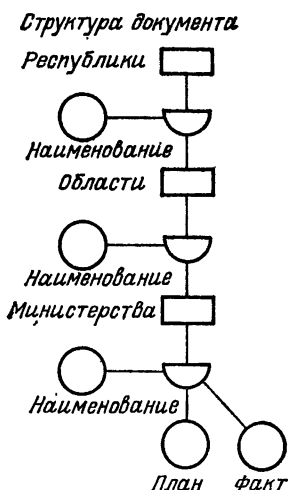


Рис. 3.17

отдельно. Движение по БД задается при этом компонентами k , $0/A/$ и $0 < s > /A/$ (см. пп. 3.2.2—3.2.5). Перебор повторностей без движения по БД (цикл по повторностям) выполняется при помощи компонент $k(p,q)/S/$, $0(p,q)/S/$. Отличаются эти компоненты тем, что в первом случае окно k обязательно должно присутствовать в каждой повторности ($p < k < q$); во втором случае любое окно в любой повторности может отсутствовать.

Если на каком-то уровне указана компонента, организующая цикл по повторностям документа, то для каждой повторности выполняется тело цикла — все подчиненные компоненты, т. е. продолжение данного уровня и все подчиненные уровни. Если внутри тела цикла вновь указаны компоненты перебора, то организуется внутренний цикл. Число таких внутренних циклов не ограничивается. Если границы подчиненной компоненты перебора вложены в границы внешней, то перебор внутренних повторностей ведется внутри элементов внешних повторностей; если не вложены, то с каждой внешней повторностью перебираются все варианты подчиненной повторности.

Пример 1. Пусть нужно осуществить ввод в базу данных, изображенную на рис. 3.16, документов (рис. 3.17), имеющих заданную форму и нумерацию окон (рис. 3.18).

Республика	Область	Министерство	План	Факт
1	2	3	4	5
РСФСР	Московская	МИНЧЕРМЕТ	40	45
	Ленинградская	МИНЦВЕТМЕТ	30	31

Рис. 3.18

Очевидно, что иерархия документа (рис. 3.17) не соответствует иерархии базы данных. Поэтому следует сначала накопить значения из окон документа, используя режим S, а затем осуществить запись в базу данных:

01 #1(1,5)/S/.#2(2,5)/S/.#3(3,5)/S/

02 МИНИСТЕРСТВА.#3.РЕСПУБЛИКИ.#1.ОБЛАСТИ.#2.

03 ПЛАН=4,ФАКТ=5

Пример 2. Предположим, что необходимо ввести в базу данных (рис. 3.19) марки автомобилей и списки унифицированных

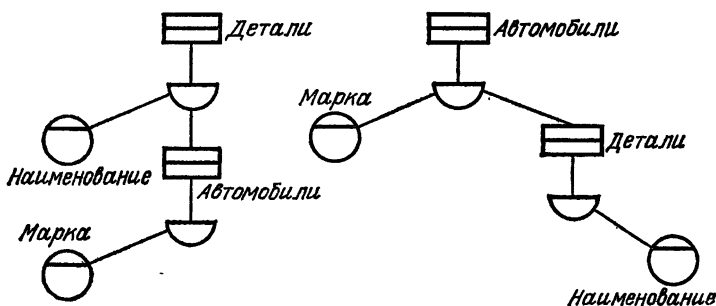


Рис. 3.19. Структура БД

деталей для каждой марки автомобиля по формам рис. 3.20. Структура этого документа представлена на рис. 3.21. Задача состоит в том, чтобы к каждой марке автомобиля приписать один и тот же список деталей (все детали относятся ко всем маркам автомобилей), и наоборот, к каждой детали — все автомобили.

Марка автомо- биля	Детали
1	2
ВАЗ 2101	Карбюратор
ВАЗ 2102	Масляный фильтр
ВАЗ 2103	Воздухоочиститель
...	...

Рис. 3.20

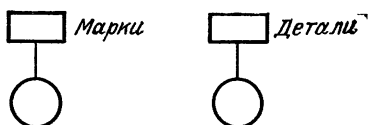


Рис. 3.21. Структура доку-
мента, изображенного на рис.
3.20

Это можно сделать, линейаризовав документ перебором всех пар марок и деталей с организацией соответствующего ввода в базу данных:

```
01 #1(1,1)/S/.#2(2,2)/S/
02 АВТОМОБИЛИ.#1.ДЕТАЛИ.#2
02 ДЕТАЛИ.#2.АВТОМОБИЛИ.#1
```

3.3. Язык описания компонент

3.3.1. Синтаксис ЯОК. В настоящем разделе приводится синтаксическая таблица языка описания компонент (ЯОК). Правила описания синтаксиса те же, что и при описании ЯОД (см. п. 1.6).

Синтаксическая таблица ЯОК

строка ::= $\left\{ \begin{array}{l} 00 \text{ имя формы} \\ [fg] 01[/\text{условие}/] [\text{описание компонент}] \\ \dots \\ mn[/\text{условие}/] [\text{описание компонент}] \end{array} \right\}$

описание компонент ::= компонента { , } компонента ...

компонента ::= $\left\{ \begin{array}{l} \text{имя}[/\text{режим}/] \\ \text{'значение'}/[\text{режим}/] \\ \# \text{окно}[(\text{от}, \text{до})][/\text{режим}/] \\ \# 0[\langle \text{шаг} \rangle][(\text{от}, \text{до})][/\text{A}/] \\ \left\{ \begin{array}{l} \text{имя} \left\{ \begin{array}{l} = \\ - \\ + \end{array} \right\} \left\{ \begin{array}{l} \text{окно} \\ \text{'значение'} \end{array} \right\} \end{array} \right\} \\ (\text{номер ссылки}) \\ \left\{ \begin{array}{l} \text{имя} = \left\{ \begin{array}{l} (\text{описание компонент}) \\ (\text{номер ссылки}) \end{array} \right\} \\ \bigcirc fh[(\text{от}, \text{шаг}, \text{до})] \end{array} \right\} \end{array} \right\}$

условие::=окно[\neg]= $\left\{ \begin{array}{l} \text{значение} \\ \text{'значение'} \end{array} \right\}$

окно::=номер окна[<начало[,длина]>]

режим::= $\left\{ \begin{array}{l} U \\ R \\ W \\ X \\ S \\ D \\ E \end{array} \right\} \left[\left\{ \begin{array}{l} * \\ I \end{array} \right\} \right]$.

номер окна::=[@]целое без знака

начало::=[@]целое без знака

от::=[@]целое без знака

до::=[@]целое без знака

шаг::=[@]целое без знака

номер ссылки::=[@]целое без знака

значение::= $\left\{ \begin{array}{l} \text{целое число} \\ \text{десятичное число} \\ \text{текст} \end{array} \right\}$

f::=буква

g::= $\left\{ \begin{array}{l} \text{буква} \\ \text{цифра} \\ \text{пробел} \end{array} \right\}$

h::= $\left\{ \begin{array}{l} \text{буква} \\ \text{цифра} \\ \text{пусто} \end{array} \right\}$

m::=цифра

n::=цифра

Описание компонент ввода документов состоит из строк, которые пишутся на бланках до 71-й позиции. Каждая строка имеет номер уровня — двузначное число от 00 до 99. Номер уровня пишется в любых позициях, начиная с 4-й. Уровни 00 и 01 пишутся всегда в позициях 4 и 5.

Если строка не умещается до 71-й позиции, то в 72-й позиции любой символ, отличный от пробела, означает перенос на следующую строку.

Начинается описание компонент со строки с уровнем 00. Эта строка служит для задания имени описания компонент ввода документа. Имя пишется с позиции 7 и имеет не более 8-ми символов. Это имя будет использовано при вводе соответствующих документов (см. пп. 3.3.3, 3.4.2, 3.4.3).

Строка уровня 01, имеющая метку в позициях 1—2, и все строки подчиненных уровней составляют описание шаблона, поименованного указанной меткой. Описание компонент может содержать один или несколько шаблонов.

В описании компонент ввода документа должна быть ровно одна строка с уровнем 01 без метки, с которой начинается собственно описание компонент. Описания шаблонов должны предшествовать основному описанию компонент.

Каждая строка ДОК описывает отрезок траектории, по которой происходит движение при вводе данных. Разветвления траектории описываются подчиненными уровнями (см. п. 3.1.3).

Компоненты траектории отделяются точками, компоненты заполняемого веера терминальных вершин отделяются запятыми (см. п. 3.2.1).

После номера уровня может быть указано (в наклонных черточках) условие выполнения этого уровня и всех ему подчиненных (см. п. 3.2.8).

В синтаксической таблице:

«имя» — это имя вершины дерева;

«значение» — числовая или текстовая константа, например, значение ключа (см. п. 3.2.2) или текст, заносимый в терминальную вершину, или число, добавляемое к значению в терминальной вершине (см. п. 3.2.7);

«окно» — это либо номер окна, из которого берется значение, либо номер окна с указанием в угловых скобках, с какого байта берется значение («начало»), и длина этого значения в байтах («длина») (см. п. 3.2.6);

«#окно» — значение ключа, которое берется из указанного окна (см. п. 3.2.2);

«(от,до)» — границы повторности (см. п. 3.2.5);

«##0» — признак прохождения через элемент простого или нумерованного массива (см. пп. 3.2.3, 3.2.4);

< шаг > — шаг нумерации элементов нумерованного массива (см. п. 3.2.3);

«(номер ссылки)» — метка места в траектории, на которое делается ссылка (см. п. 3.2.9);

«(описание компонент)» — траектория, идущая от корня дерева к месту, на которое организуется ссылка (см. п. 3.2.9);

«Q fh(от, шаг, до)» — обращение к шаблону, где fh — метка шаблона, «от» — начальное значение параметра, «до» — конечное значение, «шаг» — шаг изменения параметра (см. п. 3.2.10);

«@целое» — употребляется только в описании шаблона и является признаком формального параметра (см. п. 3.2.10);

«режим» — см. п. 3.2.11.

3.3.2. Примеры описания компонент. Примеры описания компонент ввода, приведенные ниже, демонстрируют возможные способы заполнения или изменения базы данных, графическое изображение которой дано на рис. 1.18.

Пример 1. База данных заполняется при помощи входных документов, изображенных на рис. 3.4. Соответствующее описание ДОК имеет вид:

00 АНКЕТА

ШД 01 ЧИСЛО=@0<1,2>,МЕСЯЦ=@0<4,2> ,

ГОД=@0<7>

01

02 ЗАВОД.#15.

03 ФОНД ЗАРАБОТНОЙ ПЛАТЫ+16

03 /4=МУЖ/ ЧИСЛО МУЖЧИН+'1'

03 /4=ЖЕН/ ЧИСЛО ЖЕНЩИН+'1'

03 СОТРУДНИКИ.#1.(1).

04 ПОЛ=4,ДОЛЖНОСТЬ=2,

04 ОКЛАД=16,СПЕЦИАЛЬНОСТЬ=5

04 ДАТА РОЖДЕНИЯ. ☐ ШД(3)

04 ОБРАЗОВАНИЕ.

05 /6/ ВЫСШЕЕ.ВУЗ=(ВУЗЫ/У/.#6)

06 ДАТА ОКОНЧАНИЯ. ☐ ШД(7)

05 /9/ НЕВЫСШЕЕ.ЧТО ОКОНЧИЛ=9

06 ДАТА ОКОНЧАНИЯ. ☐ ШД(10)

04 РАБОТЫ.#0<1>(11,13)/А/.

05 НАИМЕНОВАНИЕ=11,ДОЛЖНОСТЬ=13

05 ДАТА ПОСТУПЛЕНИЯ. ☐ ШД(12)

04 НАГРАДЫ.#0(14,14)/А/. =14

02 ШТАТНОЕ РАСПИСАНИЕ.#2.

03 СОТРУДНИКИ.#1.АНКЕТНЫЕ
СВЕДЕНИЯ=(1)

02 ВУЗЫ.#6.

03 АДРЕС=8

03 СОТРУДНИКИ.#1.АНКЕТНЫЕ
СВЕДЕНИЯ=(1)

Здесь на уровне 00 задано имя описания компонент АНКЕТА. Это имя должно быть указано при вводе в потоке документов на карте %%ФОРМА или %%FORMA или в процедуре ISINPUT подпараметром Ф или F параметра Р (см. пп. 3.4.2, 3.4.3).

Во второй строке описывается шаблон заполнения даты с именем ШД. Этот шаблон будет подставлен при трансляции в три разных места описания компонент (в вершины ДАТА РОЖДЕНИЯ, ДАТА ПОСТУПЛЕНИЯ, ДАТА ОКОНЧАНИЯ). При этом нули в шаблоне будут заменены на фактические значения параметров

(соответственно 3,7 или 10,12), указывающие номера окон документов, из которых будут выбираться эти даты. ЧИСЛО будет выбираться из первых двух позиций этих окон, МЕСЯЦ — из позиций 4-5, а ГОД — начиная с 7-й позиции до конца окна.

Третья строка содержит только номер уровня 01, так как описание компонент в этом примере предполагает заполнение двух деревьев базы данных с разными корнями ЗАВОД и ШТАТНОЕ РАСПИСАНИЕ (уровни 02). Эта конструкция уже рассматривалась в п. 3.1.3. Все конструкции остальных строк этого примера также достаточно подробно рассматривались в пп. 3.2.1—3.2.11.

Пример 2. Описание компонент документа для увольнения. На рис. 3.22 представлен бланк сведений о сотруднике, подлежащем увольнению. Все сведения этого бланка, кроме ФИО и ЦЕХ,

БЛАНК УВОЛЬНЕНИЯ

1. ФИО
2. Цех
3. Зарплата
4. Должность
5. Пол
6. Вуз

Рис. 3.22

можно было бы извлечь из самой базы, но для этого необходимо обратиться к средствам доступа к базе, например, к языку запросов. При использовании только средств ввода все указанные данные берутся из входного документа для корректировки сводных сведений о цехах и деревьях ШТАТНОЕ РАСПИСАНИЕ и ВУЗЫ. Описание компонент имеет вид:

00 УВОЛН

01

02 ЗАВОД.#2/R/

03 СОТРУДНИКИ.#1/E/

03 ФОНД ЗАРАБОТНОЙ ПЛАТЫ —3

03 /5=МУЖ/ ЧИСЛО МУЖЧИН —'1'

03 /5=ЖЕН/ ЧИСЛО ЖЕНЩИН —'1'

02 ШТАТНОЕ РАСПИСАНИЕ.#4/R/

03 СОТРУДНИКИ.#1/E/

02 ВУЗЫ.#6/R/.СОТРУДНИКИ.#1/E/

Пример 3. Описание компонент для документа со сведениями о вузах (рис. 3.23) имеет вид:

00 ВУЗЫ

01 ВУЗЫ.#1(1,2).АДРЕС=2

Наименование вуза	Адрес
1	2

Рис. 3.23

3.4. Организация ввода

3.4.1. Процедура трансляции описания компонент. В результате трансляции описания компонент образуется набор данных ДОК, который используется при вводе документов в базу по соответствующему макету.

Для трансляции описания компонент ввода документов служит процедура ISDOC. Процедура имеет следующие параметры:

DOC — задает имя набора данных ДОК; если этот параметр не указан, то запись происходит во временный набор данных с именем &DOC);

V или VDOC — любой из этих параметров определяет том с набором данных ДОК;

D — определяет диспозицию набора данных ДОК, по умолчанию D='MOD,PASS';

S — число блоков в наборе данных ДОК, по умолчанию S=9. Заметим, что S можно оценить по формуле $S=0.1K+1$, где K — число компонент описания;

R — объем оперативной памяти; по умолчанию R=170K.

Примеры. Задание на трансляцию без сохранения результата (для отладки) имеет вид:

```
// EXEC ISDOC
Текст на ЯОК
```

Если результат трансляции нужно сохранить в наборе A2DOC1 на диске с именем USER1, нужно подготовить задание:

```
// EXEC ISDOC,DOC=A2DOC1,V=USER1,D=',KEEP'
```

Текст на ЯОК

Результирующий набор данных может задаваться также DD-предложением с именем DOCTREE.

3.4.2. Подготовка ввода документов. При подготовке ввода информации в базу данных из входных документов на перфокарты переносятся только содержательная информация — значения окон документов.

Значения окон отделяются друг от друга специальными знаками-разделителями. Эти знаки могут быть определены самим пользователем в специальной перфокарте %%ЗНАКИ (см. п. 3.4.3). По умолчанию используются следующие разделители:

- < — начало номера окна,
- > — конец номера окна,
- / — переход к следующему окну,
- * — конец документа.

Таким образом, угловые скобки (знаки < и >) выделяют номер окна, который ставится непосредственно перед самим окном.

Если документ начинается с окна с номером 1, то его номер можно не указывать, а начинать документ непосредственно с содержимого 1-го окна.

Если номер окна на единицу больше предыдущего, то его номер можно не указывать, а поставить перед его значением разделитель перехода к следующему окну (например, знак «/»).

Пример 1. Входной документ вида:

<1>ИВАНОВ И.И.<2>ИНЖЕНЕР<3>10.03.1951
<4>МУЖ<5>ПРОГРАММИСТ*

может быть представлен следующим образом:

ИВАНОВ И.И./ИНЖЕНЕР/10.03.1951/МУЖ/
ПРОГРАММИСТ*

Пример 2. Документ для загрузки БД рис. 1.18 по макету рис. 3.4 имеет вид:

ИВАНОВ И.И./ИНЖЕНЕР/20.03.51/МУЖ/ПРОГРАММИСТ
<6>МИСИС/01.07.1975
<11>ЗИЛ/30.01.1969/ТЕХНИК
<11>ЗИЛ/15.03.1972/СТАРШИЙ ТЕХНИК
<11>МИСИС/01.09.1975/ИНЖЕНЕР
<14>МЕДАЛЬ ВДНХ
<15>ЛТР/160*

В этом документе имеются повторы окон с номерами 11—13 (см. п. 3.2.5). Указаны номера только первых окон отдельных частей документа.

Обратим внимание на то, что разделители не должны встречаться в содержательных частях документа. Если необходимо некоторые из указанных символов (<, >, /, *) использовать внутри самих окон, то соответствующие им разделители надо заменить на

другие символы, не встречающиеся в окнах. Разделители задаются специальной картой %%ЗНАКИ.

Какому разделителю соответствует каждый из указанных в карте %%ЗНАКИ символов, определяется позицией расположения символа после знака «:» (двоеточие). Позиции определяют значения разделителей в следующем порядке:

- 1) конец документа,
- 2) начало номера окна,
- 3) конец номера окна,
- 4) переход к следующему окну,
- 5) переход к следующему пункту,
- 6) повтор текущего пункта.

Например, определение разделителей по умолчанию равносильно присутствию следующей карты:

%%ЗНАКИ: * < > /

Если в карте %%ЗНАКИ стоят пробелы в позициях, то это означает, что соответствующие разделители во входных документах не используются.

П р и м е р 3.

%%ЗНАКИ: *);

- 1) ИВАНОВ И.И.;
- 2) ИНЖЕНЕР;
- 7) МИСИС;
- 8) 01.07.1975*

Разделители «переход к следующему пункту» и «повтор текущего пункта» используются, когда документ специальным образом разбивается на части. Номера первых окон выделенных частей называются пунктами. Разбиение документа на части задается специальной картой:

%%ПУНКТЫ: N1, N2, ... , Nk

где N_i — пункты, заданные в порядке возрастания и $N_1 = 1$. Если текущий номер окна равен j и удовлетворяет неравенствам $N_i < j < N_{i+1}$, то разделитель «переход к следующему окну» приводит к окну с номером N_{i+1} , а разделитель «повтор предыдущего пункта» — к окну с номером N_i .

П р и м е р 4. Документ примера 2 может быть представлен при помощи карты %%ЗНАКИ и %%ПУНКТЫ следующим образом:

%%ЗНАКИ: * < > / # &

%%ПУНКТЫ: 1, 6, 11, 14, 15

ИВАНОВ И.И./ИНЖЕНЕР/20.03.1951/МУЖ/
ПРОГРАММИСТ#

МИСИС/01.07.1975#
ЗИЛ/30.01.1969/ТЕХНИК&
ЗИЛ/15.03.1972/СТАРШИЙ ТЕХНИК&
МИСИС/01.09.1975/ИНЖЕНЕР#
МЕДАЛЬ ВДНХ#
ЛТР/160*

В одном наборе входных документов могут содержаться документы, вводимые по разным формам (описаниям компонент). Группе документов, вводимых по одному и тому же описанию ДОК, должна предшествовать специальная карта:

%%ФОРМА: имя ДОК

или

%%FORMA: имя ДОК

Здесь указывается «имя ДОК», которое значится на уровне 00 в соответствующем описании.

Пример 5.

%%ФОРМА: АНКЕТА

%%ЗНАКИ: * < > /

документ примера 2

%%ФОРМА: ВУЗЫ

МГУ/ЛЕНИНСКИЕ ГОРЫ*

МИСИС/ЛЕНИНСКИЙ ПРОСПЕКТ,4*

...

Карта %%ФОРМА может отсутствовать, если все документы вводятся по одной форме. В этом случае имя ДОК задается подпараметром Ф процедуры ISINPUT (см. п. 3.4.3).

3.4.3. Процедура ввода. Для ввода документов в базу данных по заданным описаниям компонент служит процедура ISINPUT. Эта процедура предполагает наличие дерева описания данных (ДОД) и дерева описания компонент (ДОК). Процедура заносит данные из вводимых документов (набор SYSIN) в дерево данных (ДД).

Процедура ISINPUT имеет следующие параметры:

DOD, DOC, DD — задают имена наборов данных ДОК, ДОД и ДД соответственно (по умолчанию DOD=&DOD, DOC=&DOC, DD=&DD);

VDOD, VDOC, VDD — задают имена томов с наборами данных соответственно ДОД, ДОК и ДД;

V — задает имя общего тома для наборов ДОД, ДОК и ДД;

D — определяет диспозицию набора ДД; по умолчанию D='MOD,PASS';

S,BLK,U,OUT — задаются по аналогии с процедурой ISCREDO (см. п. 1.7);

R — объем оперативной памяти; по умолчанию $R=190K$;
P — этот параметр записывается в следующем виде:

$P = \text{'позиционный подпараметр, ключевые подпараметры'}$.

Позиционный подпараметр — четырехзначное число, которое определяет следующие режимы при работе программы ввода данных:

0200 — игнорировать последние 8 байтов 80-байтной записи;
1000 — обращение к блоку контроля, если он задан;
2000 — печать не только ошибочных документов;
4000 — печать краткой информации о вводе документов;
8000 — удаление ведущих пробелов во входных данных;
0001 — печать трассы прохождения по дереву данных при занесении информации из документов в базу;

0100 — печать какой-либо информации о входном документе.

Любая совокупность этих режимов печати получается простым сложением соответствующих четырехзначных чисел. Например, $P=2101$ означает, что будут распечатываться документы в исходном виде и трассы их прохождения по базе. При отсутствии ключевых подпараметров кавычки могут быть опущены. По умолчанию $P=9100$.

Ключевые подпараметры при задании параметра P перечисляются через запятую. Они имеют следующие форматы:

$\Phi = \text{имя ДОК}$ (или $F = \text{имя ДОК}$) — указывает имя описания компонент, по которому будут вводиться документы входного потока (после появления карты %% ФОРМА ввод пойдет по другому ДОК, — см. п. 3.4.2);

$P = (N_1, N_2, \dots, N_k)$ или $P = (N_1, N_2, \dots, N_k)$ — этот подпараметр заменяет карту %% ПУНКТЫ, если ее нет во входном наборе (см. п. 3.4.2);

$Z = \text{hзнакиh}$ — этот подпараметр заменяет карту %% ЗНАКИ, если ее нет во входном потоке. Здесь «знаки» обозначают набор символов, определяющих разделители так же, как и в карте %% ЗНАКИ после двоеточия (см. п. 3.4.2), h обозначает символ, ограничивающий с двух сторон набор символов «знаки» (он должен быть отличен от символов этого набора);

$B = \text{объем памяти}$ (или $V = \text{объем памяти}$) — этот подпараметр задает объем оперативной памяти в байтах, необходимой для размещения самого большого вводимого документа, по умолчанию $B=3000$.

3.4.4. Примеры заданий на описание и заполнение базы данных.
В заключение приведем пример задания на заполнение базы данных по учету кадров завода, которая уже рассматривалась ранее (см. рис. 1.18). Задание содержит описание базы данных, описание компонент входных документов и сами входные документы.

```
//DOD   EXEC ISCREDO,DOD=A2DOD1,D='KEEP',
//   V=USER1
      текст описания базы на ЯОД (см. п. 1.5)
//DOC1  EXEC ISDOC,DOC=A2DOC1,D='KEEP',
//   V=USER1
      текст ДОК АНКЕТА (см. п. 3.3.2)
//DOC2  EXEC ISDOC,DOC=A2DOC1,V=USER1,D=OLD
      текст ДОК ВУЗЫ (см. пример 3 из п. 3.3.2)
//DOC3  EXEC ISDOC,DOC=A2DOC1,V=USER1,D=OLD
      текст ДОК УВОЛН (см. пример 2 из 3.3.2)
//INPUT  EXEC ISINPUT,DOD=A2DOD1,DOC=A2DOC1,
//        DD=A2DD1,V=USER1,D='KEEP'
%%ФОРМА: АНКЕТА
%%ЗНАКИ: *<>/#&
%%ПУНКТЫ: 1,6,11,14,15
ИВАНОВ И.И./ИНЖЕНЕР/20.03.1951/МУЖ/
ПРОГРАММИСТ#
МИСИС/01.07.1975#
АЗЛК/30.01.1969/ТЕХНИК&
ЗИЛ/15.03.1972/СТАРШИЙ ТЕХНИК&
ЗИЛ/01.09.1975/ИНЖЕНЕР#
МЕДАЛЬ ВДНХ#
ЛТР/160*

...
%%ФОРМА: ВУЗЫ
МИСИС/ЛЕНИНСКИЙ ПРОСПЕКТ,4*
МИСИ/ШЛЮЗОВАЯ НАБ.,8*
МГУ/ЛЕНИНСКИЕ ГОРЫ*

...
%%ФОРМА: УВОЛН
ТРОФИМОВ А.Н./ЗИЛ/150/ИНЖЕНЕР/МУЖ/МИСИ*
ПЛАТОНОВА Н.Т./ЗИЛ/120/ТЕХНИК/ЖЕН*

...
```

3.4.5. Процедура ввода в словарную базу данных. Если в БД имеются ссылки на словарь (вершины типа VOC, CODE, RCODE), то в шаге загрузки (//EXEC ISINPUT) должен быть указан DD-оператор с DD-именем словаря. Это имя задается в тексте описания данных оператором:

&VOC/VN=имя словаря,DDN=dd-имя словаря/

Здесь VN=имя определяет конкретный словарь в наборе данных, содержащем, вообще говоря, несколько словарей. Это имя указывается при автоматической работе со словарем только в операторе &VOC.

При записи данного в вершину типа VOC (компонента ЯОК вида имя=окно) данное автоматически кодируется словарной системой. Код данного записывается в вершину БД, а само данное (в единственном экземпляре) записывается в словарную БД. Таким образом, специальной предварительной загрузки словаря для вершин типа VOC не требуется.

Для загрузки данных в вершины типа CODE и RCODE требуется предварительная загрузка словарной БД связками слов-синонимов. Процедура ISVOCINP позволяет загружать такие словари. В дальнейшем словарь, созданный процедурой, может использоваться либо в режиме автоматической связи с БД (вершины CODE или RCODE), либо независимо от БД. В последнем случае обращение к нему осуществляется при помощи функции VOC из языка вапросов (см. п. 4.8.1) или из языков программирования. Словарь, созданный процедурой ISVOCINP, можно использовать также для автоматической кодировки и хранения данных типа VOC.

Процедура ISVOCINP имеет следующие параметры:

VOC — dsn-имя словаря (по умолчанию &VOC);

D — диспозиция словаря; по умолчанию D='PASS';

S — число блоков; по умолчанию S=9;

R — объем оперативной памяти; по умолчанию R=128K;

P — параметр, определяющий режим работы, записывается в виде P=четырёхзначное число. Первые две цифры этого числа соответствуют первым цифрам позиционного подпараметра P процедуры ISINPUT (см. п. 3.4.3). Третья цифра означает:

0 — не урезать пробелы в конце входных данных;

1 — урезать пробелы в конце входных данных.

Четвертая цифра означает:

5 — запись словаря со связями, все связи ключевые;

8 — запись словаря со связями, в которых не все связи ключевые.

По умолчанию P=9115, т. е. урезать пробелы, все слова в связи ключевые.

При подготовке данных приняты такие же разделители, как в процедуре ISINPUT, задавать их можно при помощи карты %%ЗНАКИ (см. п. 3.4.4). По умолчанию приняты разделители %%ЗНАКИ: * < > /

В окнах с 1 по 50 задаются элементы связки слов. В окне номер 100 задается имя словаря в наборе данных VOC. В окнах с 101 до 150 задаются префиксы, префикс i-го элемента связки задается в окне 100+i. Окна 201—250 имеют смысл только для режима 8. Если в окне 200+i указано слово KEY, то i-е слово в связке ключевое, в противном случае i-е слово неключевое. Действие окон с номерами 100—150 и 200—250 распространяется на все последующие документы до замены этих окон.

Ключевым называется такое слово, по которому однозначно определяется связка слов-синонимов. Если в связке слов существует неключевое слово, то по этому слову нельзя найти связку, но можно прочесть из словаря неключевое слово-синоним при помощи функции %VOC (см. п. 4.8.1).

Пример 1.

```
// EXEC ISVOCINP,P=9118,VOC=VOCORG,V=USER2,
```

```
// D=',KEEP'
```

```
<100> КЛОРГ
```

```
<201> KEY/KEY//KEY/
```

```
<104> А*
```

```
135742/НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ  
СИСТЕМНЫХ ИССЛЕДОВАНИЙ АН СССР/НИИСИ/3521*  
821758/ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР АН СССР/ВЦ АН  
СССР/1325*
```

```
589352/НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ  
СОЦИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ/НИИСИ/0521*
```

....

```
<104> П*
```

```
287138/ГВЦ ГОСПЛАНА СССР/ГВЦ ГПСССР/127*
```

В этом примере загружается словарь-классификатор организаций. Имя словаря КЛОРГ (окно 100), 1-е, 2-е и 4-е слова в связке слов являются ключевыми, 3-е слово — неключевое. Если загружать такой словарь в режиме 5, то при записи третьей связки будет выдано диагностическое сообщение о том, что слово НИИСИ уже имеется в словаре. В нашем примере загрузка идет в режиме P=9118 и окно 203 пустое, следовательно, слово НИИСИ будет неключевым и связка введется в словарь. К значениям четвертых слов в связке, в которых задается отраслевой шифр организации, приписывается префикс, который для организаций АН СССР равен «А», а для организаций Госплана СССР равен «П».

Пример 2.

```
// EXEC ISVOCINP,P=9115,VOC=HCD,V=USER1,
```

```
// D=',KEEP'
```

```
%% ЗНАКИ:*&()
```

```
(100)НДС
```

```
(101)Д*
```

```
01&ДИРЕКТОР*
```

```
02&ЗАМЕСТИТЕЛЬ ДИРЕКТОРА&ЗАМ.ДИРЕКТОРА*
```

....

```
(101)С*
```

```
01&СЛЕСАРЬ*
```

```
02&МОНТАЖНИК*
```

....

В этом примере загружается словарь НДС учебного примера. Этот словарь состоит из нескольких словарей: должность, специальность, награды. Так как шифры совпадают, то для загрузки их в общий словарь они снабжаются префиксами Д,С. В словаре будут храниться следующие связки слов: Д1—ДИРЕКТОР, Д2—ЗАМЕСТИТЕЛЬ ДИРЕКТОРА, . . . , С1—СЛЕСАРЬ, С2—МОНТАЖНИК, . . . и т. д. Награды загружаются в словарь автоматически при заполнении базы. При этом соответствующие шифры генерируются системой.

Существуют два способа приобретения знания: посредством доказательства и посредством опыта.

Бэкон

Г л а в а 4

СИСТЕМА ЗАПРОСОВ

4.0. Введение

4.0.1. Влияние двойственной природы знаний на банки данных. Любое знание имеет двойственную природу — это знание фактов и знание процедур получения новых фактов из уже имеющихся. Такая двойственность имеет следующее важное свойство: одно и то же знание можно представить как факт в одной системе и как процедуру — в другой. Например, факт «Волга впадает в Каспийское море» в одной из геофизических баз данных получается как результат выполнения процедуры пересечения хранящегося в базе данных множества точек, определяющего течение реки Волги, с множествами точек, определяющими контуры морей.

Чем более изучен предмет знаний, тем проще переход от одного вида знаний к другому. Есть знания, которые в силу сложности или плохой изученности предмета не поддаются формализации и представляются пока либо исключительно процедурными, либо только «фактографическими». Например — умение человека узнавать знакомых по их изображениям или наличие многочисленных историй болезней с неизвестными способами диагностики.

При работе с банками данных важно учитывать эту двойственную природу знаний. В частности, свойство взаимозаменяемости фактов и процедур приходится учитывать на каждом этапе работы с базой: при проектировании баз данных, в описаниях схемы, под-схем и внешних пользовательских представлений баз данных, при манипулировании данными.

Одним из основных способов повышения эффективности работы базы данных является ввод в базу избыточной информации при проектировании, т. е. замена процедур готовыми фактами. Примерами здесь могут служить: инверсные входы для обеспечения быстрого доступа по известным значениям неключевых данных

(см. п. 2.3); классификаторы и тезаурусы, при помощи которых вводятся новые сетевые связи (см. п. 2.5); сводные и расчетные показатели, которые хранятся в базе данных наряду с элементарными данными.

Что лучше: хранить в базе данных минимальное количество данных, а все производные от них получать в результате выполнения процедур, либо хранить избыточные данные для обеспечения оперативных ответов на соответствующие типовые запросы? Этот вопрос — один из главных в проектировании баз данных. Как правило, для повышения оперативности ответов на запросы нужно большее число данных (ответов или заготовок ответов) хранить в базе, превращая тем самым часть процедурных знаний в «фактографические». Сокращение времени ответа достигается при этом за счет большего расхода дисковой памяти, а также времени в процессе ввода и предварительной обработки данных.

При проектировании баз данных одним из центральных является вопрос об организации предварительной обработки тех или иных данных различными способами, а также о способах хранения результатов обработки. Обработка данных в ИНЕС реализуется средствами системы запросов и средствами ввода и вывода. Так, в системах ввода и вывода имеется возможность накопления сумм (операции «+» и «—» при вводе см. п. 3.2.7). В системе вывода имеются возможности записи результатов поиска и расчетов в последовательный набор данных для быстрой дополнительной обработки и распечатки их по заданным формам, а также средства записи в последовательный набор отредактированных к печати выходных документов для оперативной выдачи их на экран в диалоговом режиме (см. [13.20]).

Роль предварительной обработки данных играет также реструктуризация базы, которая осуществляется специальными средствами макетной реструктуризации ИНЕС (см. [40]) либо средствами запроса (см. п. 4.6). Организованная с помощью этих средств перезапись базы позволяет обеспечить любую ее реструктуризацию: смену и уничтожение уровней иерархии, организацию новых сетевых связей и др. Типичный случай реструктуризации — «накачка» инверсных входов и классификаторов.

В языке запросов влияние двойственности знаний наиболее ощутимо, выполнение запроса логически разбивается на части, относящиеся к обработке процедурами различных групп данных. При этом последовательность доступа к данным (по той или иной траектории ДД) определяет последовательность выполнения процедур обработки, а наличие данных — сам факт выполнения процедур (фактически доступ к данным управляет выполнением процедур). Таким образом, запрос представляет собой сложную смесь считываний данных из базы и выполнения процедур, соответственно

сам запрос имеет двойственную природу: с одной стороны, он имеет черты описания подсистемы, так как определяет подмножество данных (фактов), извлекаемых из базы данных, с другой — черты, свойственные обычным программам, так как описывает алгоритмы (процедуры обработки и получения данных). Интересно, что в представлении текста запроса используется общая внешняя форма уровневой записи как для данных, так и для процедур их обработки. Уровневая запись процедур оказывается удобной, так как дает возможность наглядно представить подчиненность блоков запроса и процедур обработки, условных конструкций, тел циклов и т. д.

Система запросов предоставляет наиболее богатые средства обработки данных. В то же время все средства ввода, контроля, запроса, вывода в ИНЕС имеют возможность подключения подпрограмм пользователя, т. е. дополнительных процедур обработки. Подпрограммы могут быть параметризованными запросами, написанными на языке запросов, и/или подпрограммами, написанными на языках программирования.

Таким образом, двойственность представления знаний обеспечивается в ИНЕС широкими возможностями хранения и работы с совокупностью данных и процедур.

4.0.2. Средства системы запросов ИНЕС. Для работы с данными в *системе запросов* (СЗ) ИНЕС применяется *язык запросов* (ЯЗ) (см. пп. 4.1—4.7 и [25, 37]). Соответственно основными компонентами системы запросов являются транслятор, интерпретатор и библиотека стандартных функций СЗ (см. рис. 4.1).

Система запросов выполнена в соответствии со стандартами ОС ЕС. Оттранслированные запросы могут быть исполнены сразу интерпретатором, либо обработаны редактором связей ОС ЕС и записаны в архив запросов, представляющий собой библиотеку загрузочных модулей.

Запросы из архива могут исполняться как обычные загрузочные модули либо собираться с помощью редактора связей с другими модулями, написанными на ЯЗ и/или других языках программирования, и затем передаваться на выполнение.

Результатом работы транслятора ЯЗ является массив (программа), который интерпретируется интерпретатором СЗ.

Существует динамическая связь ЯЗ с другими языками программирования ОС ЕС (фортран, ПЛ/1, кобол, ассемблер, язык сценария диалога ИНЕС), т. е. из программ, написанных на ЯЗ, можно обратиться к программам, написанным на других языках, и наоборот. Из запроса можно обратиться на выполнение других или того же самого запроса.

Запрос можно выполнить как отдельное задание, включающее его трансляцию и выполнение, либо только его трансляцию, либо

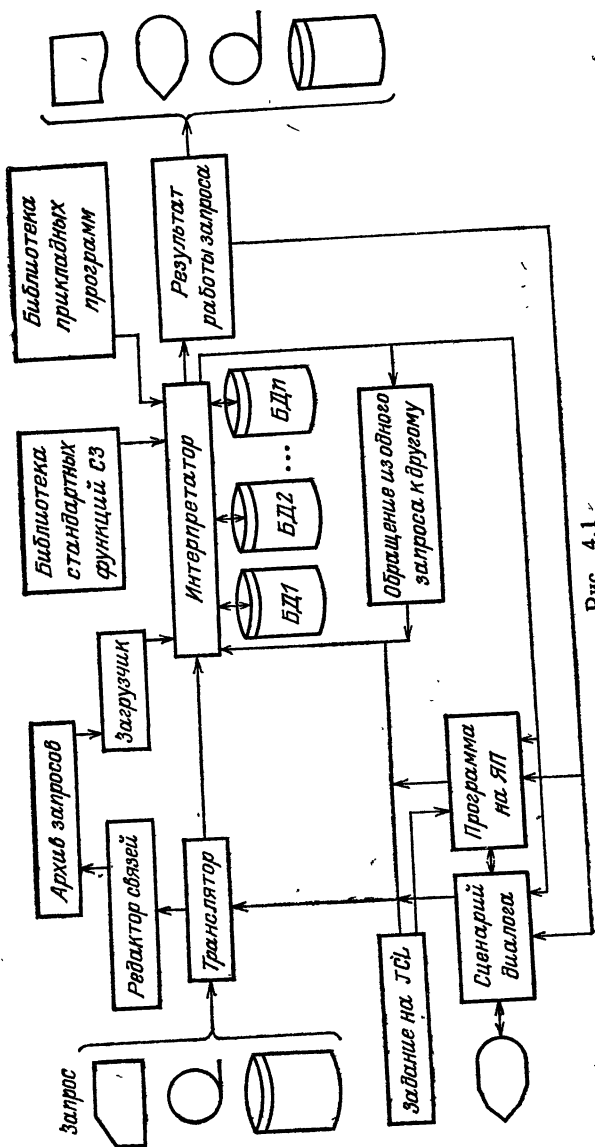


Рис. 4.1

только выполнение ранее оттранслированного запроса (см. п. 4.9). Текст запроса можно набрать на экране дисплея или сформировать программным путем в памяти ЭВМ для передачи транслятору и интерпретатору. Типовые параметризованные запросы могут быть заранее оттранслированы. Запуск их на исполнение может осуществляться как отдельное задание с параметрами, как программа, запускаемая из сценария диалога с принятыми с экрана параметрами, а также из программы на языках программирования ОС ЕС с параметрами, которые могут быть введены с внешних носителей, рассчитаны или извлечены из БД. Запуск ранее оттранслированного запроса осуществляется в последнем случае обычным оператором CALL.

4.1. Движение по базе данных

4.1.1. Движение по траектории. Как упоминалось выше, для организации доступа к данным средствами СЗ должно быть задано движение по дереву данных. Спуск по уровням дерева от корня или текущей точки базы задается в запросе траекторией — цепочкой идентификаторов БД, разделенных точками (см. п. 3.1.2; в литературе по базам данных употребляется также понятие конкатенированного или составного ключа). Пример траектории (рис. 4.2):

ЗАВОД.А17.СОТРУДНИКИ.ИВАНОВ.

В запросе эта траектория задает спуск на уровень вершин ЗАРПЛАТА и ДОЛЖНОСТЬ. В общем случае при выполнении запроса знак точки в траектории означает движение вниз по дереву данных, имя — движение на вершину текущего уровня с этим именем.

В записи траектории при прохождении ключевого массива текстовый ключ записывается аналогично именам ДОД. Если ключ содержит символы, отличные от букв и цифр или не начинается с буквы, то его следует заключить в кавычки и поставить перед ним знак #. Например (рис. 4.2):

ЗАВОД.#'А17/03'.СОТРУДНИКИ.#'ПЕТРОВ П.В.'

Если ключ — целое число, то кавычки можно не ставить, например:

ЛАБОРАТОРИЯ.#32.

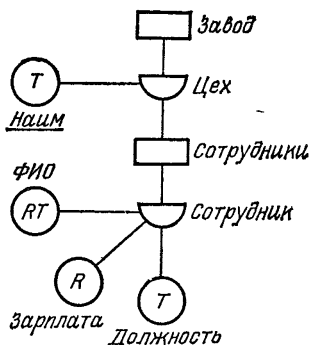


Рис. 4.2

Значение имени элемента траектории может храниться в рабочем поле раздела WSECT (см. п. 4.2). В этом случае имя в траектории задается в виде #&имя поля. Например, (рис. 4.2):

ЗАВОД.#&ЦЕХ.СОТРУДНИКИ.#&ФИО.

Для доступа к данным в запросе траектории должны выписываться строго в соответствии с деревом данных.

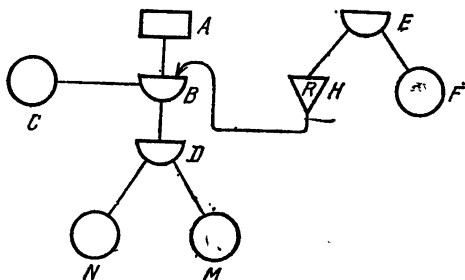


Рис. 4.3

Траектория может проходить через вершину типа REF (ссылка на значение), в этом случае в траектории указывается имя вершины, содержащей ссылку, и не указывается имя вершины, на которую

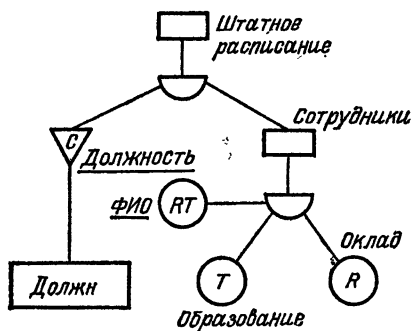


Рис. 4.4

направлена ссылка. Траектория строится таким образом в связи с тем, что при движении в вершину типа REF автоматически происходит переход по ссылке. Например, для деревьев сети, изображенной на рис. 4.3, допустимы траектории:

Е.Н.С или Е.Н.Д.М, а также А.В.С или А.В.Д.М.

Если ключом массива является вершина типа CODE без префикса, то можно указывать в качестве ключа любой элемент соот-

ветствующей связки синонимов. Например, если в ключевом массиве ШТАТНОЕ РАСПИСАНИЕ (см. рис. 4.4) ключом является ДОЛЖНОСТЬ типа CODE со ссылкой на словарь ДОЛЖН, а в словаре ДОЛЖН были записаны связки слов:

01/ДИРЕКТОР*
02/ЗАМЕСТИТЕЛЬ ДИРЕКТОРА/ЗАМ.ДИРЕКТОРА*
03/ГЛАВНЫЙ ИНЖЕНЕР*
04/ВЕДУЩИЙ ИНЖЕНЕР*
.....

то допустимыми будут траектории:

ШТАТНОЕ РАСПИСАНИЕ.ВЕДУЩИЙ ИНЖЕНЕР.
СОТРУДНИКИ.

или

ШТАТНОЕ РАСПИСАНИЕ.##'ЗАМ.ДИРЕКТОРА'.
СОТРУДНИКИ.

или

ШТАТНОЕ РАСПИСАНИЕ.##2.СОТРУДНИКИ.

и т. д. Если ключом является вершина типа CODE с префиксом, то при использовании в траектории первого элемента связки его нужно указывать с префиксом.

Траектория может рассматриваться как законченный запрос. Например, по записи:

ЗАВОД.А17.СОТРУДНИКИ.ИВАНОВ.%%PRINT('1',
ЗАРПЛАТА)

(рис. 4.2) будет напечатано:

ЗАРПЛАТА=180;

если зарплата сотрудника Иванова из цеха А17 равна 180 рублям.

В этом примере выдача результата на печать производится при помощи функции PRINT (см. п. 5.2), которая будет использоваться и в дальнейшем. Простейший вариант обращения к оператору имеет вид:

%%PRINT(формат, имя1, имя2, . . .)

Формат '0' задает вывод перечисленных данных в виде таблицы, формат '1' — в виде списка.

Другой пример запроса (рис. 4.5):

ЗАВОД.##'А17/03'.СОТРУДНИКИ.##'ПЕТРОВ П.В.'.
ДАТА РОЖДЕНИЯ. %%PRINT('0', ЧИСЛО, МЕСЯЦ, ГОД)

Результат его выполнения:

Число	Месяц	Год
11	02	1939

Для выполнения запросов достаточно воспользоваться процедурой ISREQCLG (см. п. 4.9), т. е. оформить шаг задания:

```
// EXEC ISREQCLG
```

```
ЗАВОД.А17.СОТРУДНИКИ.ИВАНОВ.
```

```
%%PRINT('1',ЗАРПЛАТА)
```

для первого примера, или

```
// EXEC ISREQCLG
```

```
ЗАВОД.##'А17/03'.СОТРУДНИКИ.##'ПЕТРОВ П.В.'.
```

```
ДАТА РОЖДЕНИЯ. %%PRINT('1',ЧИСЛО,МЕСЯЦ,ГОД)
```

для второго примера.

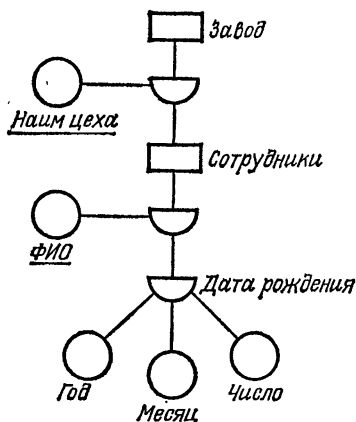


Рис. 4.5

Очевидно, для получения любого данного из базы достаточно вводить соответствующую траекторию.

Примеры (см. рис. 1.18).

1. Выдать число мужчин и женщин цеха В13.

Запрос:

```
ЗАВОД.В13.%%PRINT('0',ЧИСЛО МУЖЧИН,ЧИСЛО  
ЖЕНЩИН)
```

Результат:

Число мужчин	Число женщин
173	120

2. Выдать адрес Московского института стали и сплавов (МИСиС).

Запрос:

ВУЗЫ.МИСИС.%%PRINT('1',АДРЕС)

Результат:

АДРЕС=МОСКВА, ЛЕНИНСКИЙ ПРОСПЕКТ,4;

3. Найти адрес вуза, который окончил Иванов И. И. из цеха А17.

Запрос:

ЗАВОД.А17.СОТРУДНИКИ.#'ИВАНОВ И.И.'.
ОБРАЗОВАНИЕ.ВЫСШЕЕ.ВУЗ.
%%PRINT('1',НАИМЕНОВАНИЕ,АДРЕС)

Результат:

НАИМЕНОВАНИЕ=МГУ; АДРЕС=МОСКВА, ЛЕНИНСКИЕ ГОРЫ;

4. Найти зарплату заместителя директора Петрова П. В.

Запрос:

ШТАТНОЕ РАСПИСАНИЕ.ЗАМЕСТИТЕЛЬ ДИРЕКТОРА.
СОТРУДНИКИ.#'ПЕТРОВ П.В.'.АНКЕТНЫЕ СВЕДЕНИЯ.
%%PRINT('1',ОКЛАД)

Результат:

ОКЛАД=400;

Иерархию движения по базе данных можно отразить в тексте запроса, разбив запись движения на уровни. Так, траектория движения по базе рис. 4.2 может быть представлена в виде:

01 ЗАВОД.#'А17/03'.
02 СОТРУДНИКИ.
03 #'ИВАНОВ И.И.'

Как и в описании компонент ДОК (см. п. 3.1.3), подчинение уровней в такой записи соответствует иерархии вершин при движе-

нии по базе. Отдельному уровню может соответствовать любой отрезок траектории. Движение, описанное на уровне n , является продолжением движения, описанного на ближайшем предшествующем уровне $n-1$ (в общем случае допускаются пропуски в нумерации уровней, тогда движение уровня n продолжает ближайший предшествующий уровень с меньшим номером). Например, сведения о вузе, который окончил Иванов И. И. из цеха А17, и о занимаемой им должности можно получить при помощи запроса:

00 ТЕХТ

01 ЗАВОД.А17.СОТРУДНИКИ.'ИВАНОВ И.И.'

02 %%PRINT('1',ФИО)

02 ОБРАЗОВАНИЕ.ВЫСШЕЕ.

03 ДАТА ОКОНЧАНИЯ.

04 %%PRINT('1',ГОД)

03 ВУЗ.

04 %%PRINT('1',НАИМЕНОВАНИЕ,АДРЕС)

02 %%PRINT('1',ДОЛЖНОСТЬ)

Здесь фрагмент запроса (совокупность движений и операций), обеспечивающий распечатку сведений об образовании, подчинен вершине ОБРАЗОВАНИЕ (что записано на уровне 02) и ограничен ближайшим уровнем. 02.

Таким образом, при помощи уровневой записи может быть организована совокупность «ветвящихся» траекторий доступа к различным данным.

Следует иметь в виду, что при записи запроса по уровням — «уровневой записи», — в противоположность простой «скобочной» записи, должны исполняться простые правила: запрос начинается с предложения 00 ТЕХТ, либо просто с уровня 00 с пробелами (в дальнейших примерах настоящей главы эта строка с нулевым уровнем, как правило, опускается); номер уровня начинается в позициях со 2-й по K -ю, где K задается параметром LCARD в процедуре запроса ISREQCLG (см. п. 4.9); запись, не начинающаяся с номера уровня, считается продолжением предыдущей. Более подробно правила уровневой записи изложены в п. 4.4.2.

4.1.2. Перечисление. Движение по базе данных на одном уровне может быть задано конструкцией перечисления. Эта конструкция в запросе имеет вид:

(имя1, имя2, . . . , имя K)

Применение конструкции перечисления приводит к повторению для каждого элемента списка совокупности подчиненных движений и операций, т. е. подчиненного фрагмента запроса. Например, по запросу:

ЗАВОД.А17.СОТРУДНИКИ.(ПЕТРОВ,СИДОРОВ).

%%PRINT('0',ФИО,ЗАРПЛАТА)

ФИО и зарплата Петрова и Сидорова будут напечатаны в виде таблицы по формату печати '0':

ФИО	Зарплата
ПЕТРОВ	180
СИДОРОВ	200

Перечисление может быть задано на нескольких уровнях.

Пример.

5. Перебрать по два министерства, Минвуз и Минпрос, в каждой из трех заданных республик РСФСР, УССР, БССР и распечатать фамилии министров (рис. 4.6).

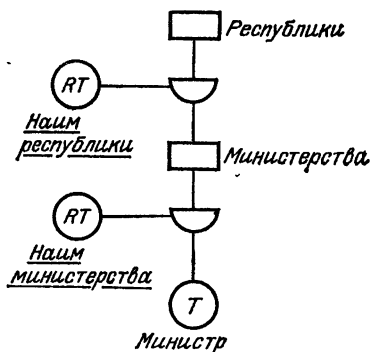


Рис. 4.6

Запрос:

01 РЕСПУБЛИКИ.(РСФСР,УССР,БССР).

02 МИНИСТЕРСТВА.(МИНВУЗ,МИНПРОС).

03 %%PRINT('0',НАИМ МИНИСТЕРСТВА,МИНИСТР)

Результат:

Наим министерства	Министр
МИНВУЗ	СОКОЛОВ А. П.
МИНПРОС	ПЕТРОВ Н. К.
МИНВУЗ	МИХЕЕВ К. Т.
МИНПРОС	ВАСИЛЬЕВ П. Т.
МИНВУЗ	СОЛОВЬЕВ Т. А.
МИНПРОС	ДЕМИДОВ Т. П.

Названия соответствующих республик можно выдать на печать, вставив дополнительное обращение к PRINT. Запрос примет вид:

01 РЕСПУБЛИКИ.(РСФСР,УССР,БССР).

02 %%PRINT('1',НАИМ РЕСПУБЛИКИ)

03 МИНИСТЕРСТВА.(МИНВУЗ,МИНПРОС).

04 %%PRINT('0',НАИМ МИНИСТЕРСТВА,МИНИСТР)

Результат:

НАИМ РЕСПУБЛИКИ=РСФСР;

Наим министерства	Министр
МИНВУЗ МИНПРОС	СОКОЛОВ А. П. ПЕТРОВ Н. К.

НАИМ РЕСПУБЛИКИ=УССР;

Наим министерства	Министр
МИНВУЗ МИНПРОС	МИХЕЕВ К. Т. ВАСИЛЬЕВ П. Т.

НАИМ РЕСПУБЛИКИ=БССР;

Наим министерства	Министр
МИНВУЗ МИНПРОС	СОЛОВЬЕВ Т. А. ДЕМИДОВ Т. П.

В уровневой записи конструкция перечисления
(имя1,имя2, ... ,имяK)f

может быть оформлена еще и следующим образом:

п. . .

· n+1 имя1

n+1 имя2

. . .

$n+1$ имяК
 $*n$ f

На уровне n в начале конструкции записывается движение, предшествующее конструкции перечисления, а на уровне $*n$ — последующее. Здесь f — подчиненный фрагмент запроса, а в качестве элементов перечисления могут использоваться не только имена, но и произвольные фрагменты запроса.

Запрос на распечатку ФИО и зарплаты Петрова и Сидорова (см. предыдущий пример) в такой записи имеет вид:

```
01 ЗАВОД.А17.СОТРУДНИКИ.
02 ПЕТРОВ
02 СИДОРОВ
*01 %%PRINT('0',ФИО,ЗАРПЛАТА)
```

4.1.3. Перебор всех элементов уровня. Перебор всех элементов текущего уровня, подчиненных общей вершине, задается в запросе

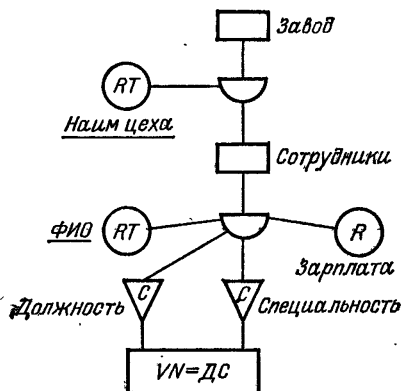


Рис. 4.7

оператором ALL. При этом для каждого элемента уровня выполняется подчиненный фрагмент запроса.

Примеры.

6. Распечатать список всех сотрудников цеха А17 (рис. 4.7).
 Запрос:

```
ЗАВОД.А17.СОТРУДНИКИ.ALL.
%%PRINT('0',ФИО,ДОЛЖНОСТЬ,СПЕЦИАЛЬНОСТЬ,
ЗАРПЛАТА)
```


Результат:

ФИО	Должность	Специальность	Зарплата
ИВАНОВ И.И.	ИНЖЕНЕР	ИНЖЕНЕР-ФИЗИК	180
ПЕТРОВ П.П.	ТЕХНИК-		170
...	НАЛАДЧИК		

7. Распечатать список сотрудников всего завода (рис. 4.7).
Запрос:

```
ЗАВОД.ALL.%%PRINT('1',НАИМ ЦЕХА)  
СОТРУДНИКИ.ALL.%%PRINT('0',ФИО,ДОЛЖНОСТЬ,  
ЗАРПЛАТА)
```

Результат:

НАИМ ЦЕХА=A1;

ФИО	Должность	Зарплата
БЕЛОВ В. Н.	ИНЖЕНЕР	180
БУРОВ И. С.	СЛЕСАРЬ	200
...		

НАИМ ЦЕХА=A2;

...

В уровневой записи оператору перебора ALL, заданному на уровне n, подчиняется фрагмент запроса, расположенный на продолжении этого уровня и ниже, до ближайшего следующего уровня с номером n или до конца запроса. Например, запрос на распечатку списка сотрудников всего завода имеет вид:

01 ЗАВОД.ALL.

02 %%PRINT('1',НАИМ ЦЕХА)

02 СОТРУДНИКИ.ALL.

03 %%PRINT('0',ФИО,ДОЛЖНОСТЬ,ЗАРПЛАТА)

См. также пример 29 в п. 4.2.2.

4.1.4. Перебор элементов уровня, удовлетворяющих заданному условию. Перебор элементов уровня с проверкой некоторого условия задается в запросе конструкцией вида:

ALL|(условие) или: ALL COND(условие)

Если условие, заданное в конструкции, выполнено, то текущей точкой становится соответствующий элемент, и выполняются следующие за условием операции запроса. (Перед проверкой условия производится спуск на первый уровень поддерева, подчиненного текущему элементу, после проверки — подъем.)

Условие может быть задано одним из следующих способов:

1. Траектория движения по базе данных. Условие считается выполненным, если движение возможно.

2. Сравнение двух арифметических выражений по схеме:

$$\begin{array}{c} \text{арифметическое} \\ \text{выражение 1} \end{array} \left\{ \begin{array}{c} = \\ \neq \\ > \\ >= \\ < \\ <= \end{array} \right\} \begin{array}{c} \text{арифметическое} \\ \text{выражение 2} \end{array}$$

где арифметическое выражение представляет собой формулу из знаков арифметических операций $+$, $-$, $*$, $/$, скобок, констант, имен рабочих полей и вершин базы (текстовые константы заключаются в кавычки). Условие считается выполненным, если сравнение удовлетворяется.

3. Кванторы существования и общности, заданные по схеме:

имя массива `EXIST COND(условие 1)`

имя массива `EVERY COND(условие 1)`

В первом случае условие в целом выполнено, если существует хотя бы один элемент массива, удовлетворяющий условию 1. Во втором случае — если все элементы массива удовлетворяют условию 1.

4. Составное условие, образованное при помощи логических функций `AND`, `OR`, `NOT`, скобок и условий 1, 2, 3, 4. Составное условие выполнено, если полученная логическая функция истинна. В составное условие может в начале или в конце входить фрагмент запроса.

Примеры.

8. Выдать список сотрудников завода, имеющих награды (рис. 1.18).

Запрос:

`ЗАВОД.ALL.СОТРУДНИКИ.ALL COND(НАГРАДЫ),`
`%%PRINT('0',ФИО,ДОЛЖНОСТЬ)`

Результат распечатается в виде таблицы.

9. Выдать по цехам список сотрудников, получающих больше 300 рублей (рис. 1.18).

Запрос:

`01 ЗАВОД.ALL.%%PRINT('1',НАИМЕНОВАНИЕ)`

02 СОТРУДНИКИ.ALL COND(ЗАРПЛАТА>300).

03 %% PRINT('0',ФИО,ЗАРПЛАТА)

10. Выдать список сотрудников завода, окончивших МВТУ (рис. 1.18).

Запрос:

01 ЗАВОД.ALL.

02 СОТРУДНИКИ.ALL COND(ОБРАЗОВАНИЕ.ВЫСШЕЕ
ВУЗ.НАИМЕНОВАНИЕ='МВТУ').

03 %% PRINT('0',ФИО,
ОБРАЗОВАНИЕ,ВЫСШЕЕ.ДАТА ОКОНЧАНИЯ.ГОД)

Результат:

ФИО	Год
ИВАНОВ И. К.	1953
ПЕТРОВ А. П.	1968
...	...

В запросе примера 10 при выходе по траектории на вершину ВУЗ типа REF текущая точка запроса автоматически переключается на вершину-адресат (элемент массива ВУЗЫ), а после проверки условия возвращается на проверяемый элемент массива СОТРУДНИКИ. В операции PRINT в качестве имени второго данного указана траектория, ведущая к нему из текущей точки.

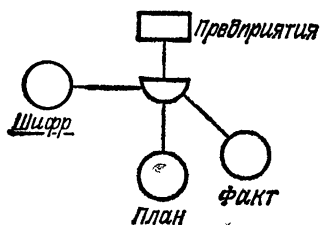


Рис. 4.8

11. Выдать список предприятий, выполнивших план менее чем на 98% (рис. 4.8).

Запрос:

ПРЕДПРИЯТИЯ.ALL COND(ФАКТ/ПЛАН<0.98).

%% PRINT('0',ШИФР,ПЛАН,ФАКТ,ФАКТ/ПЛАН*100)

Результат:

Шифр	План	Факт	Факт/план*100
А-1223	20 000	18 360	91.800
...

В операции PRINT последнее данное задано арифметическим выражением.

12. Выдать список сотрудников, когда-либо работавших на заводе ЗИЛ (рис. 1.18).

Запрос:

```
ЗАВОД.ALL.СОТРУДНИКИ.ALL COND  
(РАБОТЫ.EXIST COND(НАИМЕНОВАНИЕ='ЗИЛ')),  
%%PRINT('1',ФИО)  
РАБОТЫ.ALL.  
%%PRINT('0',НАИМЕНОВАНИЕ,ДОЛЖНОСТЬ,  
        ДАТА ПОСТУПЛЕНИЯ.ГОД)
```

Результат:

ФИО=ИВАНОВ А.К.;

Наименование	Должность	Год
АЗЛК	СТАЖЕР	1957
АЗЛК	ТЕХНИК	1958
ЗИЛ	ТЕХНИК	1962
...

13. Выдать список сотрудников, работавших только на заводе ЗИЛ (рис. 1.18).

Запрос:

```
ЗАВОД.ALL.СОТРУДНИКИ.ALL COND  
(РАБОТЫ.EVERY COND(НАИМЕНОВАНИЕ='ЗИЛ')),  
%%PRINT('1',ФИО)  
РАБОТЫ.ALL. %%PRINT('0',НАИМЕНОВАНИЕ,  
        ДОЛЖНОСТЬ)
```

Результат распечатается в виде таблицы аналогично примеру

12. В левой колонке таблицы будет стоять наименование «ЗИЛ».

14. Выдать список по цехам на 1983 год сотрудников-мужчин старше 60 лет, получающих больше 300 рублей, и женщин старше 55 лет, получающих больше 250 рублей (рис. 1.18).

Запрос:

```
ЗАВОД.ALL. %%PRINT('1',НАИМЕНОВАНИЕ)  
СОТРУДНИКИ.ALL COND  
((ПОЛ='МУЖ' AND ДАТА РОЖДЕНИЯ.ГОД<1923  
  AND ЗАРПЛАТА>300) OR  
(ПОЛ='ЖЕН' AND ДАТА РОЖДЕНИЯ.ГОД<1928  
  AND ЗАРПЛАТА>250)).  
%%PRINT('0',ФИО,ДОЛЖНОСТЬ,ЗАРПЛАТА)
```

Запись условия в этом запросе можно видоизменить, вставив перед ним фрагмент запроса:

```
ЗАВОД.ALL.%%PRINT('1',НАИМЕНОВАНИЕ)
СОТРУДНИКИ.ALL COND((&ПОЛ:=ПОЛ)
(&ДАТА:=ДАТА РОЖДЕНИЯ.ГОД)(&ЗАРПЛ:=
ЗАРПЛАТА)
(&ПОЛ='МУЖ' AND &ДАТА<1923 AND &ЗАРПЛ>300)|
(&ПОЛ='ЖЕН' AND &ДАТА>1928 AND &ЗАРПЛ>250)).
%%PRINT('0',ФИО,ДОЛЖНОСТЬ,ЗАРПЛАТА)
```

Здесь перед проверкой условий значения нужных данных запоминаются в рабочих полях запроса с именами ПОЛ, ДАТА, ЗАРПЛ (эти поля должны быть описаны в специальном разделе запроса WSECT — см. п. 4.2). В таком варианте запроса собственно проверка условия не связана с движениями по базе данных, эти движения выполняются в операциях присвоения.

В уровневой записи запроса запись условия, начиная с COND, можно размещать на одном или нескольких уровнях, подчиненных оператору перебора ALL. Во втором случае собственно подчиненный фрагмент запроса начинается с символа* и того же номера уровня, что и в записи оператора _COND.

Запрос примера 12 в такой записи имеет вид:

```
01 ЗАВОД.ALL.
02 СОТРУДНИКИ.ALL
03 _COND РАБОТЫ.
04 EXIST COND(НАИМЕНОВАНИЕ='ЗИЛ')
*03 .%%PRINT('1',ФИО)
04 РАБОТЫ.ALL.
05 %%PRINT('0',НАИМЕНОВАНИЕ,ДОЛЖНОСТЬ,
ДАТА ПОСТУПЛЕНИЯ.ГОД)
```

Перечисленных средств запроса достаточно, чтобы получать данные по любым разрезам базы данных с нужными условиями отбора.

4.1.5. Дополнительные средства перебора элементов данного уровня. В общем случае перебор элементов, удовлетворяющих некоторому условию, организуется по схеме:

$$\text{описание перебора} ::= \left\{ \begin{array}{c} \text{COND} \\ | \\ \text{WHILE} \end{array} \right\} (\text{условие})$$

В предыдущем п. 4.1.4 разбирались варианты использования этой конструкции с оператором перебора ALL. Аналогичным образом организуется перебор с другими описателями:

ALL_NEXT — перебор всех элементов, следующих за текущим;

ANY — перебор всех элементов до первого, удовлетворяющего условию;

(идентификатор1, идентификатор2, ...) — перечисление перебираемых элементов.

Примеры.

15. Распечатать список сотрудников цеха A17, начиная с Гришина А. М. (рис. 1.18).

Запрос:

```
ЗАВОД.A17.%%PRINT('1',НАИМЕНОВАНИЕ)
СОТРУДНИКИ.(#'ГРИШИН А.М.',ALL_NEXT).
%%PRINT('0',ФИО,ДОЛЖНОСТЬ,ЗАРПЛАТА)
```

Результат:

НАИМЕНОВАНИЕ=A17;

ФИО	Должность	Зарплата
ГРИШИН А. М.	ТЕХНИК	150
ИВАНОВ А. К.	СЛЕСАРЬ	170
...	...	

Если заранее неизвестно, работает ли Гришин А. М. в цехе A17, то нужно дать указание системе встать на ФИО «ГРИШИН А.М.», либо на ближайший следующий элемент. Это делается с помощью функции SPECREQ (см. п. 4.6.9).

16. Найти любого сотрудника, закончившего МАИ и занимающего должность начальника цеха (рис. 1.18).

Запрос:

01 ЗАВОД.ALL.

```
02 СОТРУДНИКИ.ANY COND(ДОЛЖНОСТЬ=
'НАЧАЛЬНИК ЦЕХА' AND ОБРАЗОВАНИЕ.
ВЫСШЕЕ.ВУЗ.НАИМЕНОВАНИЕ='МАИ').
```

```
03 %%PRINT('1',ФИО,ОКЛАД)
```

17. Выбрать из списка пяти цехов те, в которых относительно количество женщин меньше 40% (рис. 1.18).

Запрос:

```
ЗАВОД.(A3,A8,A15,B2,B23) COND
```

```
(ЧИСЛО ЖЕНЩИН/ЧИСЛО МУЖЧИН<0.4).
```

```
%%PRINT('0',НАИМЕНОВАНИЕ,ЧИСЛО МУЖЧИН
ЧИСЛО ЖЕНЩИН)
```

По конструкции:

описатель перебора WHILE(условие)

организуется перебор элементов до тех пор, пока условие выполнено.

18. Выдать список 50 сотрудников цеха A17, начиная с первого (рис. 1.18).

Запрос:

```
01 ЗАВОД.A17.СОТРУДНИКИ.(&K:=1) ALL WHILE  
(&K<50).
```

```
02 %%PRINT('O',ФИО,ДАТА РОЖДЕНИЯ.ГОД,ОКЛАД)
```

```
02 (&K:=&K+1)
```

Здесь количество сотрудников подсчитывается в рабочем поле &K.

19. Выдать список первых 20 сотрудников, награжденных орденом Красной Звезды (рис. 1.18).

Запрос:

```
01 (&R:=0) ЗАВОД.ALL.
```

```
02 СОТРУДНИКИ.ALL WHILE((&R<20) AND (НАГРАДЫ.  
EXIST COND(TVAL='ОРДЕН КРАСНОЙ ЗВЕЗДЫ'))).
```

```
03 (&R:=&R+1) %%PRINT('I',ФИО,ПОЛ)
```

Здесь используется переменная TVAL (см. п. 4.2.2).

4.1.6. Неидентифицированные движения на данном уровне.
До сих пор движение на определенный элемент уровня задавалось идентификатором этого элемента вида: имя, #имя, #число, #&имя-поля, либо описателями перебора (см. пп. 4.1.4, 4.1.5). Неидентифицированное движение в элемент уровня задается конструкциями:

FIRST — движение на первый элемент уровня;

NEXT — движение на следующий (по отношению к текущей точке) элемент уровня;

PREVIOUS — движение на предыдущий элемент уровня;

LAST — движение на последний элемент уровня.

При использовании этих конструкций следует иметь в виду, что порядок элементов ключевого массива в базе данных — лексикографический или возрастающий по значениям ключа.

Примеры.

20. Выдать для каждого цеха фамилии первых трех сотрудников по списку (рис. 1.18).

Запрос:

```
ЗАВОД.ALL.%%PRINT('I',НАИМЕНОВАНИЕ)
```

```
СОТРУДНИКИ.(FIRST,NEXT,NEXT),
```

```
%%PRINT('O',ФИО)
```

21. Для каждого сотрудника, закончившего МВТУ, указать первое место работы и должность (рис. 1.18).

Запрос:

```
ЗАВОД.ALL.СОТРУДНИКИ.ALL COND(ОБРАЗОВАНИЕ,  
ВЫСШЕЕ.ВУЗ.НАИМЕНОВАНИЕ='МВТУ').  
%% PRINT('1',ФИО)  
РАБОТЫ.FIRST.%% PRINT('1',НАИМЕНОВАНИЕ,  
ДОЛЖНОСТЬ)
```

Результат:

```
ФИО=ИВАНОВ А.А.;  
НАИМЕНОВАНИЕ=АЗЛК; ДОЛЖНОСТЬ=ИНЖЕНЕР;  
ФИО=КОЛОСОВ О.И.;  
НАИМЕНОВАНИЕ=ЗИЛ; ДОЛЖНОСТЬ=ИНЖЕНЕР;  
....
```

22. Для каждого начальника цеха указать предыдущую должность (рис. 1.18).

Запрос:

```
ШТАТНОЕ РАСПИСАНИЕ.НАЧАЛЬНИК ЦЕХА.  
СОТРУДНИКИ.ALL.АНКЕТНЫЕ СВЕДЕНИЯ.  
%% PRINT('1',ФИО, ДОЛЖНОСТЬ)  
РАБОТЫ.(ANY COND(ДОЛЖНОСТЬ='НАЧАЛЬНИК  
ЦЕХА'),PREVIOUS).%% PRINT('1',ДОЛЖНОСТЬ)
```

Результат:

```
ФИО=АГАПОВ В.К.;  
ДОЛЖНОСТЬ=НАЧАЛЬНИК ЦЕХА;  
ДОЛЖНОСТЬ=ВЕДУЩИЙ ИНЖЕНЕР;  
ФИО=ВОРОНОВ М.В.;  
ДОЛЖНОСТЬ=НАЧАЛЬНИК ЦЕХА;  
ДОЛЖНОСТЬ=НАЧАЛЬНИК УЧАСТКА;  
....
```

В уровневой записи использование неидентифицированных движений подчиняется общим правилам записи траектории.

4.1.7. Движение в корень базы данных. Движение в корень какого-либо дерева базы данных из текущей точки осуществляется оператором DOWNROOT. Такое движение применяется, например, в базах данных, содержащих нормативно-справочную информацию в виде отдельных деревьев (см. п. 2.4), если связь между деревьями поддерживается по согласованным ключам (ссылка REF не используется). В уровневой записи оператор DOWNROOT располагается на подчиненном уровне.

Пример:

23. Для каждой женщины, работающей в цехе A17, указать ее зарплату, а также средний оклад и общее количество сотрудников, работающих в той же должности (рис. 4.9).

Запрос:

01 ЗАВОД.A17.СОТРУДНИКИ.ALL COND (ПОЛ='ЖЕН').

02 (&ФИО:=ФИО)&ДОЛЖН:=ДОЛЖНОСТЬ)
(&ЗАРПЛ:=ЗАРПЛАТА)

03 DOWNROOT.

04 ШТАТНОЕ РАСПИСАНИЕ.#&ДОЛЖН.

05 %%PRINT('I', &ФИО, &ДОЛЖН, &ЗАРПЛ,
СРЕДНИЙ ОКЛАД,КОЛ)

Результат:

ФИО=ИВАНОВА Н.И.; ДОЛЖН=СЛЕСАРЬ; ЗАРПЛ=200;

СРЕДНИЙ ОКЛАД=195; КОЛ=80;

ФИО=НИКОЛАЕВА А.В.; ДОЛЖН=НАЛАДЧИЦА;

ЗАРПЛ=175;

СРЕДНИЙ ОКЛАД=180; КОЛ=55;

...

В этом запросе рабочие поля &ФИО, &ДОЛЖН, &ЗАРПЛ используются для хранения соответствующих данных до момента

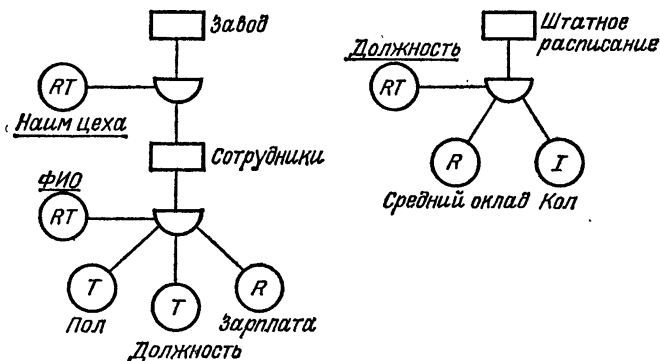


Рис. 4.9

их выдачи. Как уже указывалось, для выполнения запроса эти поля должны быть описаны. В функции PRINT рабочие поля используются для печати первых трех данных.

Следует отметить, что оператор DOWNROOT может входить в траекторию данной функции PRINT. Например (рис. 4.9):

01 ЗАВОД.A17.СОТРУДНИКИ.ALL COND(ПОЛ='ЖЕН'),

02 (&ДОЛЖН:=ДОЛЖНОСТЬ)

```
02 %%PRINT('1',ФИО,&ДОЛЖН,ЗАРПЛАТА,  
DOWNROOT.ШТАТНОЕ РАСПИСАНИЕ.#&ДОЛЖН.  
СРЕДНИЙ ОКЛАД)
```

4.1.8. Условное движение. Условное движение задается в виде:

IF условие THEN фрагмент запроса1 [ELSE фрагмент запроса2];
Условие задается одним из способов, описанных в п. 4.1.4: траекторией, сравнением арифметических выражений, выражением с кванторами, составным условием. При выполнении запроса после проверки условия текущая точка возвращается в исходное положение

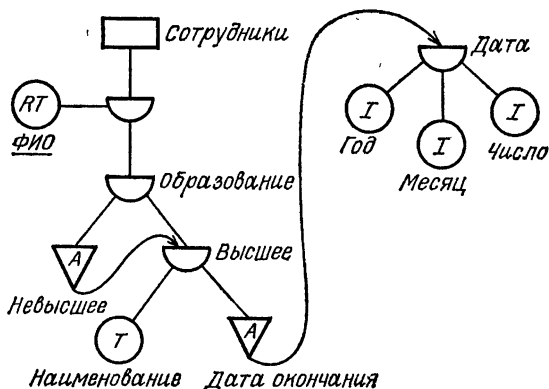


Рис. 4.10

и выполняется фрагмент 1, если условие выполнено, или фрагмент 2, если условие не выполнено *).

Пример.

24. Распечатать сведения об образовании сотрудников (рис. 4.10).

Запрос:

```
СОТРУДНИКИ.ALL.%%PRINT('1',ФИО) ОБРАЗОВАНИЕ.  
IF ВЫСШЕЕ THEN ВЫСШЕЕ. ELSE НЕВЫСШЕЕ.;  
%%PRINT('1',ДАТА ОКОНЧАНИЯ.ГОД)
```

Альтернатива ELSE фрагмент 2 может, вообще говоря, отсутствовать в условной конструкции. Например (рис. 1.18):

```
ЗАВОД.ALL.СОТРУДНИКИ.ALL.%%PRINT('1',ФИО)  
IF НАГРАДЫ THEN
```

*) В версиях ИНЕС начиная с мая 1987 г. имеется новая форма оператора: IF (условие); если условие, указанное в операторе, выполнено, то выполняется подчиненный фрагмент запроса, в противном случае — нет.

```

%%PRINT('1',ОБРАЗОВАНИЕ.ВЫСШЕЕ.ВУЗ.
      НАИМЕНОВАНИЕ);
%%PRINT('0',РАБОТЫ.ALL.НАИМЕНОВАНИЕ)

```

Здесь после фамилии сотрудника всегда будет напечатан список мест его работы, а для получивших награды — еще и сведения о высшем образовании перед списком.

Условная конструкция в уровневой записи может полностью располагаться на одном уровне. В этом случае ее ограничением является точка с запятой. Например (рис. 1.18):

```

01 ЗАВОД.ALL.СОТРУДНИКИ.ALL.
02 %%PRINT('1',ФИО)
02 IF НАГРАДЫ THEN %%PRINT('1',ОБРАЗОВАНИЕ.
      ВЫСШЕЕ.ВУЗ.НАИМЕНОВАНИЕ);
02 %%PRINT('0',РАБОТЫ.ALL.НАИМЕНОВАНИЕ)

```

В уровневой записи сложная условная конструкция записывается в виде:

```

n_IF условие
n_THEN фрагмент запроса 1
n_ELSE фрагмент запроса 2

```

Для каждого уровня в этой конструкции совокупность подчиненных уровней содержит условие после IF, либо фрагмент запроса по выполнению или невыполнению условия после THEN и ELSE. Ограничением конструкции является конец подчиненной совокупности уровней. Например, запрос примера 24 представится в виде:

```

01 СОТРУДНИКИ.ALL.
02 %%PRINT('1',ФИО)
02 ОБРАЗОВАНИЕ.
03_IF ВЫСШЕЕ
03_THEN ВЫСШЕЕ.ВУЗ.
04 %%PRINT('1',ДАТА ОКОНЧАНИЯ. ГОД)
03_ELSE НЕВЫСШЕЕ.
04 %%PRINT('1',ДАТА ОКОНЧАНИЯ.ГОД)

```

В уровневой записи повышается наглядность представления вложенных условных конструкций (см. п. 5.5.6).

4.2. Рабочая область СЗ

Рабочая область системы запросов предназначена для хранения значений данных. Значения записываются в поля рабочей области в процессе выполнения запроса. Запись значений осуществляется при помощи оператора присваивания.

4.2.1. Описание рабочей области запроса. Рабочая область описывается специальным разделом запроса, в скобочной записи этот раздел имеет вид:

****WSECT:(поле1,поле2, . . .)**

Рабочее поле может быть простым и составным. Описание простого поля задается конструкцией:

[кратность] имя [формат]

Кратность — целое число от 1 до 32 767 — определяет количество элементов данного формата, составляющих рабочее поле. Кратность, равную единице, можно не указывать. Если кратность больше единицы, то поле называется массивом (при обращении к элементу массива следует указывать его порядковый номер — индекс, см. ниже).

Имя — идентификатор рабочего поля, состоящий из букв, цифр и знаков подчеркивания, причем первый знак — буква. Длина имени должна не превышать 8 символов.

Формат определяет тип и длину элементарного рабочего поля. Он может принимать следующие значения:

[F] — целое 4 байта — этот формат можно не указывать;

[H] — целое 2 байта;

[E] — плавающее 4 байта;

[D] — плавающее 8 байтов;

[<целое>] — текст длины <целое>, от 1 до 256 байтов;

Описание составного (структурного) рабочего поля задается конструкцией:

[кратность] имя (поле1,поле2, . . . ,полеN)

где поле *i* — описание *i*-го подчиненного рабочего поля.

Пример описания рабочей области:

****WSECT:(ЦЕХ[20],ФОНД[E],20ВУЗ[30],
БАНКЕТА(ФИО[30],10РАБОТА(НАИМ[20],ГОД[H]))))**

Здесь описаны два простых рабочих поля, массив кратности 20 и массив составных полей кратности 5. В такой рабочей области можно одновременно разместить сведения о шифре цеха, фонде заработной платы, о 20 вузах и об анкетах пяти сотрудников: их фамилии, имена, отчества и до десяти мест работы каждого. Место работы, в свою очередь, описывается наименованием (текст длины 20 байтов) и годом поступления (целое 2 байта). Общий объем этой рабочей области 1874 байта.

В уровневой записи раздел WSECT описывается на нулевом уровне. Разбивка по уровням может использоваться для описания структурных рабочих полей запроса. Например, приведенное выше

описание рабочей области в уровневой записи выглядит следующим образом:

```
00 WSECT
01 ЦЕХ[20],ФОНД[E],20ВУЗ[30]
01 5АНКЕТА
02 ФИО[30]
02 10РАБОТА
03 НАИМ[20],ГОД[Н]
```

Раздел WSECT должен содержать описания всех рабочих полей, упоминающихся в запросе, кроме, быть может, некоторых элементарных рабочих полей типа F. Описание этих последних формируется автоматически при трансляции запроса и размещается в конце раздела WSECT. При распределении памяти выравнивание рабочих полей на границу полуслова, слова и двойного слова не делается (начало всей памяти рабочих полей располагается на границе двойного слова). Перед выполнением запроса все поля обнуляются.

Обращение к простой переменной рабочего поля в тексте запроса имеет вид:

&имя поля

Например, &ЦЕХ. Обращение к простому элементу массива имеет вид:

&имя поля[индекс]

где индекс может быть задан числом или простым рабочим полем типа F. Например: &ВУЗ[3], &ВУЗ[&К].

При обращении к простым полям составных рабочих полей его элементы перечисляются через двоеточие, например: &АНКЕТА[1]:ФИО, &АНКЕТА[3]:РАБОТА[&К]. Можно опустить любой начальный фрагмент составного имени, если это не приведет к недоразумениям. Например, при следующем описании рабочих полей:

```
**WSECT: (АНКЕТА(ФИО[30],ОБРАЗОВАНИЕ(НАИМ[20],
ДАТА[8])))
```

В запросе эквивалентны обозначения &АНКЕТА:ОБРАЗОВАНИЕ:ДАТА и &ДАТА.) Если в обращении опустить индекс последнего массива, то идентифицируется весь массив. Например, &АНКЕТА[3]:РАБОТА идентифицирует весь массив работ в третьей анкете. Если опустить в конце указание на элемент составного поля, то идентифицируется все это поле, например: &АНКЕТА[3]:РАБОТА[2] идентифицирует наименование и год второй работы в третьей анкете. Таким образом, в запросе можно использовать как элементарное рабочее поле, так и массив или составное поле целиком,

4.2.2. Загрузка рабочих полей. В запросе загрузка рабочих полей осуществляется при помощи операции присваивания. Операция присваивания имеет вид

(куда:=что присвоить)

Здесь «куда» может быть именем переменной рабочего поля либо траекторией, ведущей в терминальную вершину дерева данных (если эта вершина создана; вершина дерева данных создается в процессе ввода данных либо при помощи специальной функции языка запросов п. 4.6.7).

В правой части операции присваивания может стоять имя (траектория) терминальной вершины базы данных или рабочего поля; арифметическое выражение над рабочими полями и вершинами БД; системная переменная TVAL, содержащая значение текущей терминальной вершины; системная переменная NKI, содержащая значение ключа или номера текущего элемента массива. В общем случае правая часть может быть фрагментом запроса,

Например:

$(\&A[\&I]:=(\text{ПЛАН}-\text{ФАКТ})/\text{ПЛАН}*100)$

Здесь ПЛАН и ФАКТ — имена вершин ДОД.

$(\&M:=\text{IF } A=B \text{ THEN } C \text{ ELSE } D;)$

Здесь в рабочее поле M занесется значение из вершины C или D.

В рамках арифметических выражений знак арифметического действия отменяет все предыдущие движения по базе, произведенные

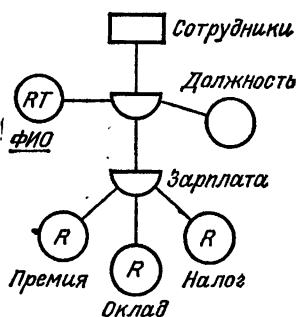


Рис. 4.11

от знака присваивания или от ближайшей открывающей скобки. Вся конструкция присваивания заключается в скобки и движение по базе внутри этих скобок также отменяется после выполнения оператора.

П р и м е р ы.

25. Указать должность и зарплату сотрудников (рис. 4.11).

Запрос:

```
**WSECT(ФИО[20],ЗАРПЛ[E])
СОТРУДНИКИ.ALL(&ФИО:=НКІ).
(&ЗАРПЛ:=ЗАРПЛАТА.(ОКЛАД+ПРЕМИЯ—НАЛОГ))
%% PRINT('0',&ФИО,ДОЛЖНОСТЬ,&ЗАРПЛ)
```

26. Составить список сотрудников, поступивших на ЗИЛ в 1980 году, указать их возраст в момент поступления (рис. 1.18),

Запрос:

```
**WSECT:(ВОЗРАСТ,ЦЕХ[5],ФИО[20])
ЗАВОД.ALL (&ЦЕХ:=НКІ).
СОТРУДНИКИ.ALL (&ФИО:=НКІ) COND(РАБОТЫ.
.EXIST COND
(НАИМЕНОВАНИЕ='ЗИЛ') AND
(ДАТА ПОСТУПЛЕНИЯ.ГОД=1980)).
(&ВОЗРАСТ:=1980—ДАТА РОЖДЕНИЯ.ГОД)
%% PRINT('0',&ФИО,&ЦЕХ,&ВОЗРАСТ)
```

27. Для сотрудников цеха А17 указать образование и возраст при поступлении на работу (рис. 1.18),

Запрос:

```
00 WSECT
01 ВОЗР,ФИО[20],ОБР[30]
00 ТЕХТ
01 ЗАВОД.А17.СОТРУДНИКИ.АЛЛ (&ФИО:=НКІ).
02 ОБРАЗОВАНИЕ.
03 IF ВЫСШЕЕ THEN
    (&ОБР:=ВЫСШЕЕ.ВУЗ.НАИМЕНОВАНИЕ)
    ELSE (&ОБР:=НЕВЫСШЕЕ.ЧТО ОКОНЧИЛ);
02 (&ВОЗР:=РАБОТЫ.FIRST.
    ДАТА ПОСТУПЛЕНИЯ.ГОД—ДАТА
    РОЖДЕНИЯ.ГОД)
02 %% PRINT('0',&ФИО,&ОБР,&ВОЗР)
```

28. Напечатать список сотрудников, имеющих более трех наград.

Запрос:

```
00 WSECT
01 ФИО[20]
00 ТЕХТ
01 ЗАВОД.ALL.СОТРУДНИКИ.ALL(&ФИО:=НКІ)
02_COND
03 (&К:=0)
03 НАГРАДЫ.ALL(&К:=&К+1)
03 (&К>3)
```

*02 (&N:=&N+1).

03 %% PRINT('0',&N,&ФИО,ДАТА РОЖДЕНИЯ.ГОД)

Результат:

N	ФИО	Год
1.	БЕЛОВ А. К.	1929
2.	ЗАЙЦЕВ Т. Л.	1932
3.	КОЛОСОВ А. Т.	1919

В запросе 28 переменные &K и &N резервируются автоматически как целые. Переменная &K используется как счетчик наград, а переменная &N суммирует количество награжденных.

29. Найти сотрудника, получающего максимальную зарплату (рис. 1.18).

Запрос:

```
**WSECT(ФИО[20],ДОЛЖН[20],ЗАРПЛ[E])
ЗАВОД.ALL.СОТРУДНИКИ.ALL COND (ЗАРПЛАТА>
>&ЗАРПЛ).
(&ФИО:=ФИО)(&ДОЛЖН:=ДОЛЖНОСТЬ)(&ЗАРПЛ:=
:=ЗАРПЛАТА),
%% PRINT('1',&ФИО,&ДОЛЖН,&ЗАРПЛ)
```

Здесь запятая перед оператором PRINT действует соответственно перечислению (см. п. 4.1.2): она отменяет предыдущие движения до открывающей скобки, в данном случае — до начала запроса. Оператор PRINT будет действовать после того, как закончится выполнение цикла.

Такой запрос естественно выражается в уровневой записи:

```
00 WSECT
01 ФИО[20],ДОЛЖН[20],ЗАРПЛ[E]
00 TEXT
01 ЗАВОД. ALL.
02 СОТРУДНИКИ.ALL COND(ЗАРПЛАТА>&ЗАРПЛ),
03 (&ФИО:=ФИО)
03 (&ДОЛЖН:=ДОЛЖНОСТЬ)
03 (&ЗАРПЛ:=ЗАРПЛАТА)
01 %% PRINT('1',&ФИО,&ДОЛЖН,&ЗАРПЛ)
```

С помощью арифметических действий над рабочими полями можно получить любые расчетные и сводные статистические сведения по БД.

4.2.3. Оператор цикла. Для обработки массивов рабочих полей предназначен оператор цикла DO. Цикл задается по схеме:

DO имя=описатель цикла, описатель цикла, . . . ;тело цикла
или по схеме:

DO WHILE условие;тело цикла

Здесь имя указывает целочисленное рабочее поле, содержащее индекс;

описатель цикла::=значение [BY шаг] [TO конец]

значение, шаг и конец — целые числа (константы или рабочие поля);

тело цикла — фрагмент запроса. В тело цикла входят операторы данного и всех подчиненных ему уровней. В скобочной записи ограничителем тела цикла является запятая (запятые внутри фрагмента должны быть заключены в скобки).

Тело цикла выполняется многократно до тех пор, пока не будут исчерпаны значения индекса, указанные в описателях. По второй схеме тело цикла выполняется до тех пор, пока не нарушится заданное условие.

Пр и м е р ы.

30. Распечатать сведения о первых десяти сотрудниках, кроме второго. Сведения хранятся в рабочем массиве АНКЕТА.

Запрос:

****WSECT:(10АНКЕТА(ФИО[20],ГОДР[F],ЗАРПЛАТА[E]))**

....

DO &I=1,3 TO 10;

%%PRINT('O', АНКЕТА[&I])

Результат:

ФИО	Год	Зарплата
АНТОНОВ А. К.	1936	175
БАРАНОВ Н. С.	1944	210
БЕЛОВ А. И.	1940	190
...		

31. Распечатать список сотрудников с табельными номерами от 10000 до 19999 (рис. 4.12).

Запрос:

СОТРУДНИКИ.(#10000,DO WHILE (NKI<=19999); NEXT).

%%PRINT('O',ТАБНОМ,ФИО,ОКЛАД)

П р и м е ч а н и е. Этот запрос выполняется, если есть сотрудник с ТАБНОМ=10000. При возможном отсутствии такого сотрудника запрос нужно модифицировать (см. п. 4.6.9.).

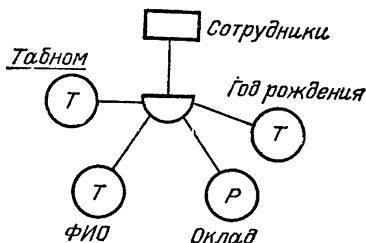


Рис. 4.12

32. Распечатать гистограмму по годам рождения сотрудников (рис. 1.18).

Запрос:

```
00 WSECT
01 90КОЛ
00 TEXT
01 СОТРУДНИКИ.ALL(&ВСЕГО:=&ВСЕГО+1)
02 (&K:=ДАТА РОЖДЕНИЯ.ГОД-1900)
02 IF (&K<1) THEN (&K:=1);
02 (&КОЛ[&K]:=&КОЛ[&K]+1)
01 DO &I=1 TO 90;
02 (&ГОД:=1900+&I)(&ЧИСЛО:=&КОЛ[&I])
02 %%PRINT('0', &ГОД,&ЧИСЛО)
01 %%PRINT('1',&ВСЕГО)
```

Результат:

Год	Число
1901	0
...	...
1930	25
1931	26
1932	31
1933	22
...	...

ВСЕГО=2382;

Полученная таблица представляет зависимость количества сотрудников от года рождения. Эту зависимость можно вывести в виде графика (см. п. 5.3).

В уровневой записи оператор цикла и тело цикла можно оформить в виде отдельных уровней, что целесообразно, если они достаточно сложны. Такое оформление имеет вид:

```
p_DO [описатели цикла]
  p+1 описатели цикла
  ...
  *п тело цикла
  p+1 [продолжение тела цикла]
```

Тело цикла ограничено ближайшим следующим уровнем с номером п. Точка с запятой в конце оператора цикла в этом случае не ставится.

4.2.4. Очистка и печать рабочих полей запроса. Очистка рабочих полей запроса производится по оператору

```
%CLRWS(&A1,&A2, ...)
```

где A1, A2, ... — имена рабочих полей. В результате выполнения команды текстовые рабочие поля, перечисленные в скобках, заполняются пробелами, числовые — нулями. Если параметры в операторе отсутствуют, то чистятся все рабочие поля запроса. Если у элемента составного имени, являющегося массивом, отсутствует указание номера, то чистится весь массив. Например,

```
**WSECT:(A[F],2B(5P(K[E],N)))
%CLRWS(&A,&B[1]:P)
```

Почистятся элементы:

```
&A
&B[1]:P[1]:K
&B[1]:P[1]:N
&B[1]:P[2]:K
&B[1]:P[2]:N
...
&B[1]:P[5]:N
```

Чистка рабочих полей производится, например, перед присвоением им значений и выводом на печать. Если данные распечатываются в цикле, и часть из них может отсутствовать, то соответствующие рабочие поля надо чистить на каждом шаге цикла.

Печать рабочих полей запроса производится по оператору

```
%OUTWS(&A1,&A2,...)
```

где A1, A2, ... — имена рабочих полей. Значения этих полей распечатываются в виде:

```
A1=значение A1; A2=значение A2; ...
```

Если параметры отсутствуют, то распечатываются все рабочие поля запроса. Если у элемента составного имени не указан номер

(для массива) или не указаны подчиненные элементы, то соответствующий массив или структура распечатываются полностью (следует иметь в виду, что на печать выводятся только первые 50 символов значений рабочих полей). Например:

```
**WSECT:(A[F],4P(N[3],L[E]))
%OUTWS(P[3])
```

Результат:

P[3]: N=значение N(3)

P[3]: L=значение L(3)

Печать рабочих полей используется для отладки запросов.

4.3. Обращение к подпрограммам

В запросе могут использоваться:

1. Подпрограммы на языке запросов, включенные в текст основного запроса, — так называемые блоки запроса.

2. Подпрограммы на любом языке программирования, в том числе на ЯЗ, если они заранее оттранслированы, отредактированы и записаны в библиотеку загрузочных модулей.

4.3.1. Обращение к блоку языка запросов. Блоки в ЯЗ представляют собой поименованные фрагменты запроса, тексты которых приводятся в основном запросе в специальном разделе. Общий вид этого раздела в скобочной записи:

```
**BLOCKS:(имя блока1=(тело блока1),
            имя блока2=(тело блока2),...)
```

Имя блока состоит из букв, цифр и знаков подчеркивания, причем начинается с буквы. Тело блока представляет собой фрагмент запроса.

Обращение к блоку из запроса имеет вид:

Имя блока

- Это обращение означает требование исполнить тело блока. Обращение к блоку может быть рекурсивным.

33. Задача разузлования. Распечатать состав изделий (см. рис. 4.13) в виде таблицы таким образом, чтобы список частей изделия А располагался в соседней колонке ниже и правее изделия А. Аналогично должен быть распечатан состав каждой части изделия А, состав всех частей каждой части и т. д. Предполагается, что уров-

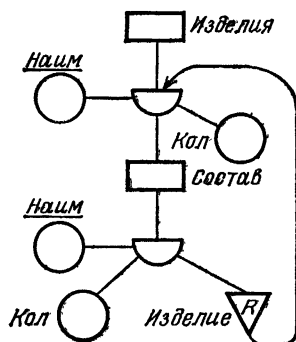


Рис. 4.13

ней вложенности не более четырех,— следовательно, состав изделия помещается в таблице из четырех колонок.

Запрос (используется фортранный формат (см. 5.2.)):

00 WSECT

01 N

01 4H

02 НАИМ [20]

02 КОЛ[4]

00 BLOCK B

01 &N:=&N+1

01 СОСТАВ.ALL.

02 %CLRWS (&H)

02 (&H[&N] : НАИМ:=НАИМ)

02 (&H[&N] : КОЛ : =КОЛ)

02 %%PRINT('F(5X,'|',4(A12,'|'))',
DO &I=1 TO 4; &H[&I]:НАИМ)

02 %%PRINT('F(5X,'|',4(A12,'|'))',
DO &I=1 TO 4; &H[&I] : КОЛ)

02 ИЗДЕЛИЕ.␣B

01 &N:=&N-1

00 TEXT

01 (%CLRWS)

01 ИЗДЕЛИЯ.ALL.

02 %%PRINT ('F (15X,A20/,5X,53"—')',НАИМ)

02 ␣B

02 %%PRINT('F(5X,53"—')')

Результат выполнения запроса:

ИЗДЕЛИЕ 1

ИЗД 1.1 КОЛ 1.1	ИЗД 1.1.1 КОЛ 1.1.1	ИЗД 1.1.1.1 КОЛ 1.1.1.1 ИЗД 1.1.1.2 КОЛ 1.1.1.2 ...	
ИЗД 1.2 КОЛ 1.2	ИЗД 1.2.1 КОЛ 1.2.1	...	
...	

ИЗДЕЛИЕ 2

ИЗД 2.1 КОЛ 2.1 ...			
---------------------------	--	--	--

...

Блоки используются в тех случаях, когда один и тот же фрагмент запроса повторяется несколько раз в разных местах. Введение блока позволяет сократить запись, сделать запрос более наглядным, а также структурировать сложные запросы. Пример использования блоков в сложном запросе приводится в п. 5.5.6.

В уровневой записи каждый блок оформляется в тексте запроса отдельно и имеет вид:

00 BLOCK имя блока=(тело блока)

где тело блока записано в скобочном виде либо в виде:

00 BLOCK имя блока

01 тело блока

02 ...

Заметим, что при описании блока в скобочной записи после его исполнения первоначальный уровень (текущая точка) в ДД не восстанавливается. Например (рис. 1.18):

```
**BLOCKS:(ИВАНОВ=(СОТРУДНИКИ.ИВАНОВ))
```

```
ЗАВОД.А17. ОИВАНОВ.
```

```
%%PRINT('1',ОКЛАД,ДОЛЖНОСТЬ)
```

Для того чтобы уровень восстановился, необходимо описать тело блока как локальную конструкцию (см. [37]), т. е. заключить его в квадратные скобки:

```
**BLOCKS: (имя блока=([тело блока])
```

Напротив, при использовании уровневой записи уровень траектории автоматически восстанавливается после того, как закончено выполнение блока. Чтобы этого не происходило, в описание блока надо ввести специальное указание:

```
00 BLOCK/NOSVBSLEV,NOTVAL/ имя блока
```

```
01 тело блока .
```

```
02 ...
```

4.3.2. Обращение к внешней подпрограмме. В запросе могут использоваться стандартные подпрограммы СЗ (см. п. 4.6), а также подпрограммы пользователя, написанные на любом языке програм-

мирования и представленные в библиотеке загрузочных модулей. Обращение к такой подпрограмме имеет вид:

% имя модуля (параметр1, параметр2, ...)

Параметрами могут служить константы и рабочие поля. Если параметры опущены, то обращение к подпрограмме следует взять в скобки. Например:

%CLRWS(&A), но (%OUTWS)

Для сокращения времени последующих обращений к подпрограмме можно в запросе заранее загрузить ее с помощью стандартной программы REQLD (см. п. 4.6.1).

В системе запросов ИНЕС обеспечивается также возможность рекурсивного обращения к запросам. При этом запрос, к которому делается обращение, может быть представлен в виде исходного текста во входном потоке шага задания либо в виде загрузочного модуля (см. описания программ REQTR, REQLDREQ, REQEX и др. в [37]).

4.4. Оформление запроса

4.4.1. Разделы запроса. Разделы запроса входят в текст запроса как составные части, имеющие свое наименование и назначение. К основным разделам относятся:

TEXT — текст запроса (основная часть);

WSECT — описание рабочих полей (см. п. 4.2.1);

BLOCKS — описание блоков (см. п. 4.3.1);

LET — описание замен;

OUTFORMS — описание заполнения окон выходных форм (см. п. 5.4.3).

Запрос в целом представляет собой совокупность разделов. Разделы описываются в произвольном порядке (при этом раздел LET действует только на нижеследующую часть запроса). Описание раздела в скобочной записи имеет вид:

**<имя раздела>:(<тело раздела>)

Тело раздела TEXT представляет собой основной текст запроса, его можно не оформлять. Правила описания разделов WSECT и BLOCKS приводились выше.

Раздел LET предназначен для описания синтаксических замен в тексте запроса. Эти замены позволяют ввести для конструкций запроса произвольные обозначения. При трансляции и исполнении запроса, написанного в терминах введенных обозначений, эти обозначения автоматически заменяются на старые конструкции.

Тело раздела LET состоит из последовательности взятых в скобки описаний замены. Описание замены имеет вид:

((<новый1>)[(<новый2>)]...(<старый>))

Здесь <новый1>, <новый2>, ... — синтаксические фрагменты, указывающие, ЧТО надо заменять при трансляции запроса, <старый> — указывает, НА ЧТО заменять. Например, запрос:

```
**LET:(((WF)(ФАМИЛИЯ))((WD)(ДОЛЖНОСТЬ))
((COTR)(СОТРУДНИКИ)))
```

```
COTR.ALL COND(WD='СТ.ИНЖ.').%%PRINT('1',WF)
```

эквивалентен запросу:

```
СОТРУДНИКИ.ALL COND(ДОЛЖНОСТЬ='СТ.ИНЖ.').
%%PRINT('1',ФАМИЛИЯ)
```

В запросе разрешается замена произвольных синтаксических конструкций (имена ДОД, имена рабочих полей и т. д.). Следует, однако, иметь в виду, что при трансляции поиск в тексте запроса левых частей замен происходит без какой бы то ни было проверки смысла и синтаксиса. В частности, левая часть может быть опознана в середине слова.

Описание замен оставляет действительными старые выделенные слова. Для того чтобы отменить старое выделенное слово, нужно отметить его звездочкой:

((<новый1>)(*<новый2>*)) ... (*<старый>))

Например, при замене:

((И)(*AND))((ИЛИ)(*OR))

«И» и «ИЛИ» будут считаться разделителями, а «AND» и «OR» — не будут.

Раздел **OUTFORMS** предназначен для описания макетного вывода документа. Правила оформления этого раздела приводятся ниже (см. п. 5.4.3).

4.4.2. Уровневая запись запроса. Для наглядного представления запроса предназначена уровневая запись запроса. Подчинение уровней в уровневой записи соответствует иерархии вершин ДД при движении по базе, а также порядку выполнения функциональных частей запроса (выделение условных конструкций, тела цикла, разбиение запроса на блоки). Выше приводились правила представления в уровневом виде иерархии движений по базе (см. пп. 4.1.1—4.1.7) и основных функциональных конструкций (см. пп. 4.1.8, 4.2.3, 4.3.1), а также правила составления разделов **WSECT** (п. 4.2.1) и **BLOCKS** (п. 4.3.1).

При оформлении разделов в уровневой записи наименование раздела помещается на нулевом уровне, а тело раздела — на уровнях, подчиненных наименованию:

00 <наименование раздела>

01 <тело раздела>

02 ...

Заголовок раздела TEXT можно опустить:

00

01 <тело раздела>

Тело раздела LET всегда записывается в обычном «скобочном» синтаксисе, уровни внутри раздела не распознаются. Например:

00 LET

((%SR)(%SPECREQ))((ИМЯ)(НАИМ)(НАИМЕНОВАНИЕ))

Представление раздела OUTFORMS в уровневом виде описывается ниже (п. 5.4.3).

Следует отметить, что запрос может быть записан частично в скобочном виде (начало запроса), частично — в уровневом (конец запроса). После того как в тексте впервые встретился нулевой уровень, уровневая запись предполагается до конца запроса.

Структура отдельной записи (перфокарты) в уровневом запросе является более сложной, чем при отсутствии уровней. В скобочном представлении каждая следующая запись считается продолжением предыдущей, длина отдельной записи задается параметром LCARD процедуры запроса (по умолчанию LCARD=72, см. п. 4.8.1).

В уровневом запросе структура отдельной записи имеет вид рис. 4.14. Номер уровня располагается в начале записи и представляет собой любое целое число от 0 до 99999 (как правило

	Поле уровня		Операторы запроса		Комментарий	
1	2	K	K+2	M-2	M	80

Рис. 4.14

используются двузначные числа). Первая цифра номера уровня должна помещаться в поле уровней, т. е. в позициях со 2-й по K-ю включительно. Если в записи есть номер уровня, то операторы запроса отделяются от него пробелом и располагаются за ним до позиции M-2 включительно. Следующая карта является продолжением предыдущей, если в позиции M-1 стоит отличный от пробела символ или если следующая карта не имеет номера уровня. В этом случае операторы запроса располагаются в позициях с 1-й по M-2.

Позиция М—1, а также текст в позициях с М по 80-ю не являются значимыми и могут использоваться для записи комментариев. Если в первых двух позициях записи стоит «++» или «—», то вся запись рассматривается как комментарий.

Позиции записи М и К задаются параметром LCARD=(М, К) в процедурах ISREQCL и ISREQCLG (см. п. 4.9).

4.5. Запрос с параметрами

Для регулярного решения типовых задач в информационных системах возникает необходимость в использовании регламентированных запросов с параметрами. В системе ИНЕС параметры запросов могут задаваться:

1. В последовательном наборе данных;
2. В поле PARM оператора EXEC языка управления заданиями (параметр PU процедур запросов);
3. В параметрах оператора CALL языков программирования (при вызове запроса из соответствующей программы);
4. На экране дисплея при работе в диалоговом режиме (см. п. 5.5).

Для считывания значений параметров в рабочие поля запроса используются специальные средства СЗ.

4.5.1. Чтение параметров из последовательного набора. В СЗ имеются средства для бесформатного и форматного чтения параметров из последовательного набора.

1. Бесформатное чтение последовательного набора выполняется с помощью обращения:

```
%REQGETAL(&P,DDN,L,K)
```

где Р — имя составного рабочего поля; DDN — имя DD-предложения, определяющего набор данных; L, К — целые числа.

Обязателен только первый операнд. В результате обращения будет считано К первых записей (или все, если К не задано). Из каждой записи возьмется не более L (если L задано) и не более 256 первых байтов. Эти части будут перенесены в оперативную память по адресам: A(P), A(P)+L, A(P)+2L, . . ., если L задано, или вплотную друг к другу (A(P) — адрес начала поля Р). При этом не обращается внимания ни на структуру Р, ни на его тип, ни на его длину. Например:

```
**WSECT:(АНКЕТА(ДАТА[8],ФИО[22]),  
ДОЛЖН(ДОЛЖНОСТЬ[15],СПЕЦИАЛЬНОСТЬ[15]))  
...  
%REQGETAL(&АНКЕТА, 'VVOD',30,2)
```

Первая запись из входного набора с DD-именем VVOD будет считана в рабочее поле АНКЕТА, вторая — в поле ДОЛЖН.

2. Форматное чтение последовательного набора выполняется программой INPWS. Обращение к программе имеет вид:

%INPWS(x,y,&z,&a1, . . . ,&aK)

где x — восьмисимвольное имя DD-предложения последовательного набора;

y — формат фортрана (допустимы спецификации A,X,/);

&z — рабочее поле формата F, после выполнения программы содержит число прочтенных записей:

a1, . . . ,aK — поля WSECT.

За одно обращение вводится очередная группа данных из последовательного набора данных. После перебора всех окон макета следующие данные вводятся по формату последней группы, заключенной в круглые скобки. Когда все данные введены, ввод останавливается, в рабочее слово &z помещается число записей, введенных по данному обращению. Если перед обращением последовательный набор данных был уже исчерпан, в &z помещается нуль.

П р и м е р ы.

```
**WSECT:(I,D(A[E],B[4]))
```

```
...
```

```
%INPWS('INPUT ','(3X,A3/2A4)',&N,&I,&D)
```

```
...
```

```
//INPUT DD *
```

```
220
```

```
5.3 ZZZZ
```

```
/*
```

Программа INPWS запишет 220 в рабочее поле &I и заполнит структурное поле &D: в поле &A запишется 5.3, в поле &B — значение 'ZZZZ'.

```
**WSECT: (N,ФИО[20],ЗАРПЛАТА[E])
```

```
00 TEXT
```

```
01 (&F:=0) DO &I=1 TO 1000 WHILE &F=0;
```

```
    %INPWS('DDN ','(A20,A6)',&N,&ФИО,&ЗАРПЛАТА)
```

```
02 _IF&N=0
```

```
02 _THEN фрагмент запроса, обрабатывающий  
    очередную пару параметров ФИО и ЗАРПЛАТА
```

```
02 _ELSE (&F:=1)
```

```
....
```

```
//DDN DD *
```

```
ИВАНОВ И. И. 195.70
```

```
ПЕТРОВ А. Б. 203.50
```

```
....
```

4.5.2. Чтение параметра из процедуры запроса. Чтение параметра шага задания процедуры запросной системы выполняется только для запросов, запущенных с помощью процедур ISREQGO или ISREQCLG (см. п. 4.9). Обращение к программе чтения параметра в запросе имеет вид:

%REQPARMO(&P)

где P — имя составного рабочего поля. В результате обращения в память WSECT, начиная с P, будет перенесен текст параметра PU процедуры запроса. При этом не учитываются ни структура, ни тип, ни длина поля P.

4.5.3. Передача параметров предварительно оттранслированному запросу. К предварительно оттранслированному модулю запроса можно обращаться из языков программирования и языков ИНЕС, передавая и принимая от него параметры. Это обращение имеет вид:

CALL имя модуля(параметр1,параметр2,.. .)

Для приема и передачи параметров в системе запросов имеется три стандартных программы: REQPARMA, REQPRIN, REQPROUT. Программа REQPARMA по обращению

%REQPARMA(&I)

где &I — ячейка связи, настраивается на прием параметров и обратную передачу результатов работы запроса.

%REQPRIN (&I,ном1,перем1,длина1,ном2,перем2,длина2,...) переписывает передаваемые запросу параметры, которые определяются номером, переменной-приемником WSECT и длиной.

Программа REQPROUT по аналогичному обращению **%REQPROUT (&I,ном1,перем1,длина1,ном2,перем2,длина2,...)** передает исходной программе результаты работы запроса.

П р и м е р. Пусть запрос с именем ZAPI имеет вид:

****WSECT:(A,10B[12],C,D[200])**

%REQPARMA(&A)

%REQPRIN(&A,1,&B,120,2,&D,200)

....

%REQPROUT(&A,3,&C,4)

К этому запросу сделано обращение:

CALL ZAPI(ARG1,ARG2,OTB)

По этому обращению поле ARG1 длиной 120 байтов пересылается в массив &B, поле ARG2 длиной 200 байтов пересылается в переменную &D, а после выполнения содержательной части запроса переменная &C длины 4 байта переписывается в поле OTB.

4.6. Стандартные программы системы запросов

4.6.1. Предварительная загрузка программ, используемых в запросе. Загрузка программ в память, позволяющая сократить время последующих обращений к ним в запросе, осуществляется программой REQLD. Формат обращения:

```
%REQLD('имя1','имя2',...)
```

Здесь имя1, имя2, ... — имена загружаемых программ (до 64 имен). Имя программы — текстовая константа длиной до 8 символов. Обращение к REQLD без параметров или без первого параметра означает загрузку программ рекурсивного обращения к СЗ (см. [37]). Например,

```
%REQLD('EDIT','CLRWS')
```

В память загружаются программы EDIT и CLRWS.

```
%REQLD('OUTWS')
```

В память загружаются программа OUTWS и программы рекурсивного обращения к СЗ.

4.6.2. Программа получения текущей даты и времени. Обращение к программе имеет вид:

```
%DATIME(&A)
```

где А — структурное рабочее поле. В результате выполнения программы в поле А независимо от ее внутреннего строения заносится содержимое буфера DATIME. Размер буфера 31 байт. В нем подряд размещены семь значений типа Н (часы, минуты, секунды, день месяца, номер месяца, год, день года), затем текст, содержащий те же значения (кроме дня года) в виде:

```
чч.мм.сс.дд.мм.гг
```

В поле А попадает весь буфер, если его длина больше или равняется 31, в противном случае занесется начало буфера по длине поля, например:

```
**WSECT:(ДАТ(ВРЕМЯ(ЧАС[Н],МИН[Н],СЕК[Н]),  
ДАТА(ДЕНЬ[Н],МЕС[Н],ГОД[Н]),ДГОДА[Н]),ДАТ[17])  
%DATIME(&ВРЕМЯ)
```

Если необходимы не все данные, то в описании рабочего поля можно выделить только нужные части. Например, если нужна только дата в текстовом представлении, можно задать:

```
**WSECT:(Д(X[23],ДАТА[8]))  
%DATIME(&Д)
```

4.6.3. Программа групповой пересылки. Программа групповой пересылки A9MOVE осуществляет считывание или запись значений группы заданных вершин поддерева ДД с оптимальным обходом поддерева. Алгоритм оптимального обхода формируется при первом обращении к программе и используется при последующих обращениях уже без использования ДОД. Соответственно выигрыш во времени (примерно в 2,5 раза) накапливается с увеличением количества обращений.

При обращении к A9MOVE текущей в БД должна быть структурная вершина. В обращении заказывается чтение и запись значений любых терминальных вершин подчиненного ей поддерева. После выполнения A9MOVE текущее положение в базе не изменится. Формат обращения:

% A9MOVE(&яс, <куда_1> <что_1>, ...)

Здесь &яс — ячейка связи, выделенная в запросе для обращения на A9MOVE с обходом заданного дерева. При обращении &яс должна содержать 0 или 1 (при &яс=1 программой A9-MOVE будет включена трасса, которая покажет поиск в ДОД заказанных вершин; при повторных обращениях трасса не работает, так как нет движения по ДОД).

Пара параметров <куда> <что> заказывает одну операцию чтения/записи. Порядок таких пар в обращении безразличен.

Для чтения из базы параметр <куда> должен быть идентификатором рабочего поля, параметр <что> может быть текстовой константой, задающей имя терминальной вершины БД, подчиненной текущей вершине (имя дерева используется только при первом обращении, в дальнейшем значение параметра <что> безразлично), или рабочим полем, содержащим имя вершины БД.

Для записи в базу параметр <куда> должен быть текстовой константой, задающей имя терминальной вершины, параметр <что> может быть текстовой константой или рабочим полем. При записи в базу заказанные вершины создаются, если их нет в ДД, и в них записываются указанные константы или значения рабочих полей. Например (см. рис. 4.15):

```
**WSECT:(ЯС,CZ[2],X[10],Y(Y1[5],Y2),V,W[2])
(&ЯС:=0) (&CZ:='D') (&W:='C1')
A.ALL
% A9MOVE(&ЯС,&X,'B11',&Y:Y1,
'B22',&Z,&CZ,&W,&W,'C2',&V, 'D','НАЗВАНИЕ')
```

Здесь при обходе дерева A1 последовательно осуществляются действия:

— значения из вершин B11 и B22 присваиваются рабочим полям &X и &Y:Y1;

— значение из вершины, имя которой задано в поле &CZ, присваивается полю &Z;

— полю &W присваивается значение из вершины, имя которой задано в этом же поле, т. е. из C1;

— в вершину C2 записывается содержимое поля &V, а в вершину D — слово НАЗВАНИЕ.

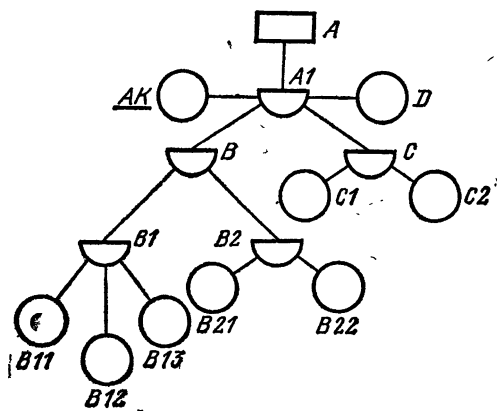


Рис. 4.15

Отметим, что после оператора ALL, задающего перебор элементов массива A, точка не ставится, так как при обращении к A9MOVE текущей должна быть вершина структуры.

Параметр, задающий имя дерева, может быть опущен, если имя дерева совпадает с идентификатором рабочего поля.

4.6.4. Специальные функции системы запросов. Специальные функции системы запросов по своей роли аналогичны стандартным программам. Обращение к специальным функциям имеет общую форму и выглядит следующим образом:

%SPECREQ(операция, другие параметры)

Здесь операция — имя конкретной операции в кавычках или ее порядковый номер в таблице операций; другие параметры — операнды данной операции. Их наличие и формат зависят от первого параметра.

Ниже приводится таблица операций специальных функций СЗ. Некоторые из этих функций разбираются в настоящем разделе более подробно. Функция DOWNROOT описывалась выше (см. п. 4.1.7), к ней можно обращаться просто по названию. Функция SWBASE описывается в п. 4.8.2.

В дальнейшем в качестве первого параметра будет использоваться наименование функции, при этом подразумевается равноправное использование ее номера.

Таблица операций

№	Имя	Параметры	Комментарий
1	MVSTRUC	&X1—куда заносить, &X2—откуда брать	MOVE STRUCTURE— перенос значения из рабо- чего поля &X2 в поле &X1
2	SUBSTR	N1—с какого байта брать, N2—сколько байтов брать, оба пара- метра числа или цело- численные рабочие поля	SUBSTRING—высечение подстроки из последнего элемента стека
3	SHOW	режим, [параметр1, па- раметр2,...]	диагностическая печать СЗ; установка флагов СЗ
4	DELV		DELETE VERTEX— уничтожение текущей вершины в ДД
5	CRTV	направление, имя вер- шины, режим созда- ния, режим работы со ссылками	CREATE VERTEX— создание подчиненной или соседней вершины в ДД
6	DCONC	составной ключ	DOWN CONCATENATI- ON KEY—движение вниз по ДД по указан- ной траектории
7	GETREF	&X—поле для ссылки	GET REFERENCE— взять ссылку в текущей вершине
8	PUTREF	&X—поле для ссылки	PUT REFERENCE—за- писать ссылку в текущей вершине
9	BRANCH	&X1 (типа F)—адрес таблицы базы, откуда переписывать, &X2 (ти- па F)—адрес таблицы базы, куда переписы- вать	переписывание поддерва ДД из одной базы в дру- гую
10	FIND	значение ключа	поиск ключа, большего или равного данному
11	DOWNROOT		движение по базе «в ко- рень»
12	SWBASE	&X (типа F)—адрес таблицы базы	SWITCH BASE—пере- ключение СЗ на другую базу или на другую те- кущую точку той же базы
13	FSTRUCT	&F (составное поле)	чтение жесткой структу- ры из базы в составное рабочее поле

4.6.5. Перенос данных из одного рабочего поля в другое. Перенос данных из одного рабочего поля в другое без преобразования форматов осуществляется с помощью операции **MVSTRUC**. При этом перенос данного позволяет разбить его на подполя нового структурного поля или, наоборот, объединить уже существовавшие подполя данного. Обращение к операции имеет вид:

%SPECREQ('MVSTRUC', &X1,&X2)

Операция заносит в рабочее поле **&X1** значение из рабочего поля **&X2**. Длина переносимого поля равна минимуму из длин **&X1** и **&X2**, причем этот минимум не должен превышать 256 байтов. Перенос происходит побайтно, типы и форматы рабочих полей не рассматриваются.

Пр и м е р.

****WSECT:(ДАТА[6],Д(ГОД[2],МЕС[2],ЧИСЛО[2]))**

...

%SPECREQ('MVSTRUC',&Д,&ДАТА)

Если **&ДАТА='830215'**, то после выполнения операции **&ГОД='83'**, **&МЕС='02'**, **&ЧИСЛО='15'**.

4.6.6. Уничтожение текущей вершины ДД. Текущую вершину дерева данных можно удалить при помощи операции **DELV**. При этом автоматически уничтожаются все подчиненные вершины, а текущей становится псевдовершина данного уровня. Обращение к операции имеет вид:

%SPECREQ('DELV')

Пр и м е р.

34. В базе данных заданы массив анкетных данных абитуриентов и списки их по группам с указанием полученных оценок (рис. 4.16). Удалить анкеты абитуриентов, получивших двойки.

Запрос:

****WSECT:(ГР,ШИФР[10])**

ГРУППЫ.ALL(&ГР:=NK1) %%PRINT('1',&ГР).

АБИТУРИЕНТЫ.ALL (&ШИФР:=NK1) COND

(ЭКЗАМЕНЫ.EXIST1(ОЦЕНКА=2)).

DOWNROOT.

АНКЕТЫ.##&ШИФР %SPECREQ('DELV')

%%PRINT('1',&ШИФР)

Распечатается список шифров выбывших абитуриентов по группам.

4.6.7. Создание вершин ДД. Вершина ДД, непосредственно подчиненная текущей или соседняя с ней, может быть создана при помощи операции **CRTV**. После исполнения операции созданная

вершина становится текущей (если произошла ошибка и вершина не создалась, то текущей становится псевдовершина указанного уровня). Обращение к операции имеет вид:

%SPECREQ('CRTV', 'направление', имя, 'режим', ['S'])

Здесь направление указывает положение создаваемой вершины: DOWN — движение вниз, создается вершина, подчиненная текущей; LEVEL — движение на уровне, создается вершина, соседняя с текущей.

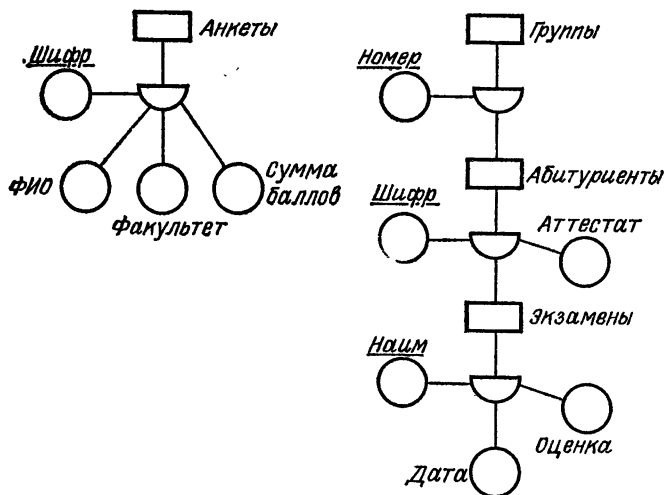


Рис. 4.16

Имя задает имя создаваемой вершины в ДОД или значение ключа создаваемого элемента массива. Имя может задаваться константой или переменной любого допустимого типа и формата.

Режим задает режим создания вершины: COND — переход в уже существующую вершину, иначе создание новой; NEW — создание новой вершины, ошибка, если вершина существует; ADD — создание вершины независимо от состояния базы; Y — режим COND для структуры или элемента ключевого массива, режим ADD для элемента простого или нумерованного массива. По умолчанию — Y.

Параметр 'S' отключает автоматический переход по ссылке (REF=STOP), что позволяет работать со ссылочной вершиной как с терминальной. Параметр задается только при создании вершины типа REF.

Пример.

35. В базе данных (рис. 4.16) требуется для сдавших все экзамены подсчитать сумму набранных баллов, включая оценку аттестата, и записать ее в анкету (вершина СУММА БАЛЛОВ).

```

00 WSECT
01 L,K,S,ШИФР[10]
00 TEXT
01 ГРУППЫ.ALL.АБИТУРИЕНТЫ.ALL(&ШИФР:=NK1),
02 %CLRWS(&S,&L) ЭКЗАМЕНЫ.ALL.
03 IF ОЦЕНКА>=3 THEN (&S:=ОЦЕНКА+&S)
                     ELSE (&L:=1);
02_IF (&L=0)
02_THEN (&K:=&K+1) (&S:=АТТЕСТАТ+&S)
03 DOWNROOT.АНКЕТЫ.#&ШИФР
04 %SPECREQ('CRTV','DOWN','СУММА БАЛЛОВ')
05 (СУММА БАЛЛОВ:=&S)
05 %%PRINT('I',&K,&ШИФР,&S)

```

Распечатается список сдавших и суммы набранных баллов.

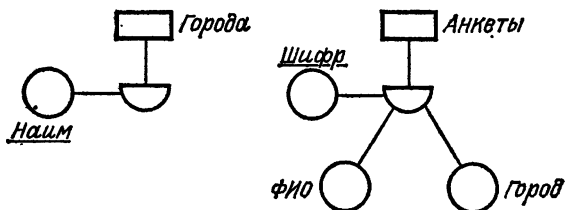


Рис. 4.17

36. Заполнить список городов, из которых приехали абитуриенты. Список задан в описании базы отдельным поддеревом (рис. 4.17).

```

00 WSECT
01 N,ГОР[30 ]
00 TEXT
01 АНКЕТЫ.ALL.%CLRWS(&ГОР)(&ГОР:=ГОРОД)
02 DOWNROOT. ГОРОДА.
03_IF NOT #&ГОР
03_THEN %SPECREQ(5,'LEVEL',&ГОР,'NEW')
04 (&N:=&N+1)

```

Здесь ГОРОДА — ключевой массив, &ГОР — содержит значение ключа. Если элемент с ключом &ГОР уже существует, то действие (&N:=&N+1) не выполняется. В противном случае создается новый элемент массива и значение &N увеличивается на единицу (в поле &N подсчитывается общее количество городов).

4.6.8. Создание сетевых связей. Для организации сетевой связи средствами запроса необходимо взять ссылку (адресный указатель

на вершину-адресат) и поместить ее в соответствующую исходную вершину типа REF.

Ссылка на текущую вершину загружается в рабочее поле при помощи операции GETREF, Обращение к операции имеет вид:

```
%SPECREQ('GETREF',&X)
```

Здесь &X — символьное рабочее поле длиной 5 байтов, в которое заносится значение ссылки. Например:

```
**WSECT:(ШИФР[10],R[5])
```

...

```
АНКЕТЫ.#&ШИФР %SPECREQ('GETREF',&R)
```

Ссылка на элемент массива АНКЕТЫ заносится в &R.

Запись значения ссылки в текущую вершину типа REF осуществляется операцией PUTREF, если движение в эту вершину произошло по операции CRTV с параметром 'S' (см. п. 4.6.7). Обращение к операции PUTREF имеет вид:

```
%SPECREQ('PUTREF',&X)
```

Здесь &X — символьное рабочее поле, содержащее значение ссылки.

Пример.

37. По результатам экзаменов заполнить список абитуриентов, набравших проходной балл 20 (список1) и не набравших его

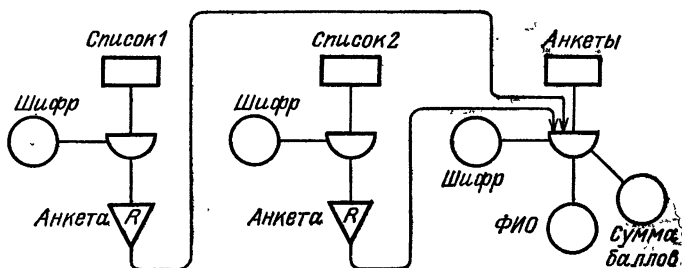


Рис. 4.18

(список2). Списки заданы отдельными деревьями в описании данных (рис. 4.18).

Запрос:

```
**WSECT: (ШИФР[10],R[5], ...)
```

```
**LET: ((%S)(%SPECREQ))
```

```
**BLOCKS: (B=(  
%S('CRTV','DOWN',&ШИФР)
```

```
%S('CRTV','DOWN','АНКЕТА','NEW','S')
%S('PUTREF',&R)))
**TEXT:
%S('CRTV','LEVEL','СПИСОК1')
%S('CRTV','LEVEL','СПИСОК2')
АНКЕТЫ.ALL(&ШИФР:=NKI) %S('GETREF',&R).
IF (СУММА БАЛЛОВ>=20) THEN
DOWNROOT. СПИСОК1QB
ELSE DOWNROOT. СПИСОК2QB;
```

Ссылка на очередной элемент массива АНКЕТЫ запоминается в поле &R и заносится в элемент массива СПИСОК1 или СПИСОК2, в зависимости от значения вершины СУММА БАЛЛОВ.

4.6.9. Поиск по идентификатору ближайшей вершины уровня. Поиск вершины, заданной идентификатором, осуществляется при помощи операции FIND. Если такая вершина отсутствует на данном уровне, то текущей становится ближайшая за ней вершина уровня. Если поиск не удался, текущей становится псевдовершина. Обращение к операции имеет вид:

```
%SPECREQ('FIND',идентификатор)
```

Здесь идентификатор — константа в кавычках или переменная любого допустимого формата. В массиве поиск ведется по значениям ключа. Например,

```
СОТРУДНИКИ.%SPECREQ('FIND','И')
```

После выполнения операции FIND текущим будет первый из элементов массива СОТРУДНИКИ (см. рис. 4.11), ключ которого — фамилия — начинается с буквы И или ближайшей буквы русского алфавита.

4.6.10. Чтение значений элементов жесткой структуры. Элементы жесткой структуры можно читать из БД порознь, так же как и элементы простой структуры. Жесткая структура в целом считывается при помощи операции FSTRUCT. Обращение к ней имеет вид:

```
%SPECREQ('FSTRUCT',&мя)
```

где &мя — рабочее поле типа структура.

Операция берет совокупность значений элементов жесткой структуры в текущей точке базы и заносит ее в рабочее поле. При этом проверяется, что рабочее поле — структура, а текущая точка находится в жесткой структуре. Операция не выполняется, если это не так.

Чтение структуры происходит без учета типов и длин ее подчиненных элементов, поэтому формат структуры в разделе WSECT должен совпадать с форматом жесткой структуры в ДОД. При

выполнении этого условия в результате операции FSTRUCT элементы структуры рабочего поля WSECT будут содержать значения из терминальных вершин базы.

Пр и м е р.

```
**WSECT: (B(C[3],D(E1[E],E2[F],F[H])))
```

...

```
A.B %SPECREQ('FSTRUCT',&B)
```

Вершина В в описании ДОД принадлежит жесткой структуре А и сама является жесткой структурой:

```
01 A:FSTRUCT
```

```
02 A1:TEXT/SIZE=5/
```

```
02 B:FSTRUCT
```

```
03 C:TEXT/SIZE=3/
```

```
03 D: FSTRUCT
```

```
04 E1:REAL/SIZE=4/; E2:INT/SIZE=4/
```

```
03 F:INT/SIZE=2/
```

После выполнения операции в рабочем поле подряд будут лежать значения из вершин С, Е1, Е2, F, т. е. текст длиной 3 байта, число длиной 4 байта, целое число 4 байта и целое длиной 2 байта.

4.7. Синтаксис языка запросов

Синтаксис языка запросов описывается в приведенных выше стандартных обозначениях (см. пп. 1.6, 3.3.1). Алфавит ЯЗ содержит символы [,] (квадратные скобки), которые используются как метасимволы при описании синтаксиса. Для отличия символов ЯЗ от метасимволов в таблице подчеркнуты символы ЯЗ.

Уровневая запись:

запрос::=строка [строка] ...

строка::=

{	00 WSECT	}
	[mn] элемент[,элемент]...	
	00 BLOCK имя блока	
	[mn] фрагмент запроса	
	00 LET:(замена[, замена] ...)	
	00 OUTFORM имя макета	
	01 имя части	
	02 заполнитель	
	00 [TEXT]	
	[mn] фрагмент запроса	
	mn_фрагмент оператора	
	*mn_фрагмент запроса	

фрагмент оператора ::= $\left\{ \begin{array}{l} \text{IF условие} \\ \text{THEN фрагмент запроса} \\ \text{ELSE фрагмент запроса} \\ \text{COND условие} \\ \text{DO WHILE условие} \\ \text{DO \&имя раб.поля=описатель цикла} \\ \text{[,описатель цикла]...} \end{array} \right\}$

m ::= цифра

n ::= цифра

Скобочная запись:

запрос: ! = [раздел] ... [****TEXT:**] [(фрагмент запроса)]

раздел: ! = $\left\{ \begin{array}{l} \text{описание рабочих полей} \\ \text{описание блоков} \\ \text{описание замен} \\ \text{описание заполнения макетов} \end{array} \right\}$

описание рабочих полей: ! = ****WSECT:** (элемент[, элемент]...)

описание блоков: ! = ****BLOCKS:** (имя блока=(фрагмент запроса),
[имя блока=(фрагмент запроса)]...)

описание замен: ! = ****LET:** (замена[, замена]...)

описание заполнения макетов: ! = ****OUTFORMS:**

(имя макета=(описание заполнения частей)

[,имя макета=(описание заполнения частей)]...)

описание заполнения частей: ! = имя части=(заполнитель

[,заполнитель]...), [имя части=(заполнитель[, заполнитель]...)]...

Общая часть синтаксической таблицы для уровневой и скобочной записи:

элемент: ! = $\left\{ \begin{array}{l} \text{простой элемент} \\ \text{составной элемент} \end{array} \right\}$

простой элемент: ! = [кратность] имя рабочего поля [[формат]]

кратность: ! = целое

формат: ! = $\left\{ \begin{array}{l} \text{F} \\ \text{E} \\ \text{D} \\ \text{H} \\ \text{целое} \end{array} \right\}$

составной элемент: ! = [кратность] имя рабочего поля [(элемент

[,элемент] ...)]

замена: ! = ((текст—обозначение)...(действительный текст))

заполнитель: ! = $\left\{ \begin{array}{l} \text{арифметическое выражение} \\ \text{фрагмент запроса} \\ \text{системная переменная} \end{array} \right\}$

$$\text{арифметическое выражение} ::= \left\{ \begin{array}{l} \text{идентификатор рабочего поля} \\ \text{идентификатор БД} \\ \text{константа} \\ \text{NKI} \\ \text{TVAL} \\ \text{формула из арифметических выражений, знаков операций } +, -, *, / \text{ и скобок } (,) \end{array} \right\}$$

$$\text{константа} ::= \left\{ \begin{array}{l} \text{'текст'} \\ \text{число} \\ \text{число.[цифра] ...} \end{array} \right\}$$

$$\text{число} ::= \left[\begin{array}{c} + \\ - \end{array} \right] \text{цифра[цифра] ...}$$

$$\text{фрагмент запроса} ::= \left\{ \begin{array}{l} \text{оператор [оператор] ...} \\ \text{(фрагмент запроса)} \\ \text{фрагмент запроса1, фрагмент запроса2} \end{array} \right\}$$

$$\text{системная переменная} ::= \left\{ \begin{array}{l} \text{'E###DATE'} \\ \text{'E###NPAGE'} \\ \text{'E###NPD'} \end{array} \right\}$$

$$\text{оператор} ::= \left\{ \begin{array}{l} \text{оператор движения} \\ \text{оператор присваивания} \\ \text{обращение к подпрограмме} \\ \text{оператор цикла} \\ \text{условный оператор} \end{array} \right\}$$

$$\text{оператор движения} ::= \left\{ \begin{array}{l} \text{идентификатор базы данных} \\ \text{неидентифицированное движение} \\ \text{перебор} \end{array} \right\}$$

$$\text{идентификатор базы данных} ::= \left\{ \begin{array}{l} \text{ИМЯ} \\ \text{\# 'ИМЯ'} \\ \text{\# число} \\ \text{\# \&ИМЯ} \end{array} \right\}$$

$$\text{неидентифицированное движение} ::= \left\{ \begin{array}{l} \text{FIRST} \\ \text{LAST} \\ \text{NEXT} \\ \text{PREVIOUS} \\ \text{DOWNROOT} \end{array} \right\}$$

$$\text{перебор} ::= \left\{ \begin{array}{l} \left\{ \begin{array}{l} \text{ALL} \\ \text{ALL_NEXT} \\ \text{(идентификатор БД,} \\ \text{идентификатор БД, ...)} \end{array} \right\} \left[\begin{array}{l} \text{WHILE} \\ \{ \text{COND} \} \end{array} \right] \text{(условие)} \\ \text{ANY} \left\{ \begin{array}{l} \text{COND} \\ | \end{array} \right\} \text{(условие)} \end{array} \right\}$$

$$\text{условие} ::= \left\{ \begin{array}{l} \text{траектория} \\ \text{арифметическое выражение} \left\{ \begin{array}{l} = \\ > \\ < \\ >= \\ <= \\ \neg = \end{array} \right\} \text{арифметическое выражение} \\ \text{имя массива} \cdot \left\{ \begin{array}{l} \text{EXIST} \\ \text{EVERY} \end{array} \right\} \left\{ \begin{array}{l} \text{COND} \\ | \end{array} \right\} (\text{условие}) \\ \text{NOT } (\text{условие}) \\ (\text{условие}) \text{ AND } (\text{условие}) \\ (\text{условие}) \text{ OR } (\text{условие}) \end{array} \right.$$

оператор присваивания ::= (куда:=что)

$$\text{куда} ::= \begin{cases} \text{идентификатор рабочего поля} \\ \text{идентификатор БД} \end{cases}$$
$$\text{что} ::= \begin{cases} \text{арифметическое выражение} \\ \text{фрагмент запроса} \end{cases}$$

идентификатор рабочего поля ::= &имя рабочего поля[индекс]
[:имя рабочего поля[индекс]]...

$$\text{индекс} ::= \left\{ \begin{array}{l} \text{целое число} \\ \text{идентификатор целого рабочего поля} \end{array} \right\}$$
$$\text{обращение к подпрограмме}::=\left\{\begin{array}{l}\% \text{имя подпрограммы(параметр} \\ \quad \quad \quad [,\text{параметр}] \dots) \\ \cup \text{ имя блока}\end{array}\right\}$$
$$\text{параметр} ::= \begin{cases} \text{идентификатор рабочего поля} \\ \text{константа} \end{cases}$$
$$\text{оператор цикла} ::= \left\{ \begin{array}{l} \text{DO WHILE условие;} \\ \text{DO } \&\text{имя рабочего поля} = \text{описатель цикла} \\ \quad [\text{.описатель цикла}] \dots; \end{array} \right\}$$

описатель цикла ::= значение [BY шаг] [TO конец] [WHILE условие]

$$\left\{ \begin{array}{l} \text{значение} \\ \text{шаг} \\ \text{конец} \end{array} \right\} ::= \left\{ \begin{array}{l} \text{целое число} \\ \&\text{имя рабочего поля} \end{array} \right\}$$

условный оператор ::= IF условие THEN фрагмент запроса
[ELSE фрагмент запроса];

Как упоминалось выше, в настоящее время приняты два способа оформления запроса: в уровневой и скобочной записи. Уровеньная запись наглядно отражает структуризацию запроса. Скобочная запись в ранних версиях ИНЕС была единственной.

В уровневой записи запрос представляет собой совокупность строк, распределенных по уровням. Номер уровня ставится в начале строки. Строка, не имеющая номера уровня, считается продолжением предыдущей (см. п. 4.4.2).

На нулевом уровне описываются основной и вспомогательные разделы. Имя основного раздела указывать не обязательно: 00 [TEXT].

Под разделом понимается один из вспомогательных разделов: WSECT, BLOCK, LET, OUTFORM (см. п. 4.4.1).

Для условного оператора (п. 4.1.8), оператора перебора (п. 4.1.4) и оператора цикла (п. 4.2.3) вводится специальная многоуровневая запись сложного условия, описателя цикла и подчиненных фрагментов запроса. В этой записи после номера уровня ставится знак подчеркивания перед фрагментами операторов (фрагменты операторов начинаются со слов IF, THEN, ELSE, COND, DO).

Подчиненный фрагмент запроса в конструкции перечисления и тело цикла могут начинаться со строки, номер уровня которой помечен звездочкой (см. пп. 4.1.2, 4.2.3).

В скобочной записи запрос состоит из тех же разделов: WSECT, BLOCKS, LET, OUTFORMS, TEXT. Разделы оформляются по схеме:

**имя раздела:(тело раздела)

Раздел TEXT оформлять не обязательно. В отличие от уровневой записи разделы BLOCKS и OUTFORMS включают описания соответственно всех блоков и заполнения всех макетов (см. пп. 4.4.1, 5.4.3).

Описание рабочих полей состоит из описаний элементов через запятую или на разных уровнях (см. п. 4.2.1).

Под элементом понимается описание простого или составного рабочего поля. Простой элемент без кратности описывает элементарное поле указанного формата. Если кратность указана и имеет значение больше 1, то это значит, что описывается массив однородных полей указанного формата. Если опущен формат, то подразумевается, что описываемое поле или массив полей имеет формат F.

Имя рабочего поля начинается с буквы и состоит из набора букв, цифр и знаков подчеркивания любой длины, но идентифицируется по первым восьми символам.

Кратность — целое число от 1 до 32767, определяет количество элементов массива.

Формат определяет тип и длину элементарного рабочего поля и принимает следующие значения: F — целое в 4 байта, H — целое в 2 байта, E — плавающее в 4 байта, D — плавающее в 8 байтов. Если формат имеет значение «целое число», то это значит, что

рабочее поле имеет текстовый тип, а само целое число указывает число символов в текстовом поле (от 1 до 256 байтов).

Составной элемент описывает рабочее поле, объединяющее несколько рабочих полей (в общем случае разных форматов), каждое из которых в свою очередь может быть составным элементом. В уровней записи подчиненные элементы располагаются на подчиненных уровнях.

Описание блоков выделяет часть запроса в один или несколько блоков (см. п. 4.3.1). Имя блока начинается с буквы и состоит из набора букв, цифр и знаков подчеркивания.

Описание замен предоставляет возможность вводить сокращенные обозначения частей текста запроса или заменять одни имена или обозначения на другие (см. п. 4.4.1). В левых скобках замены задается обозначение того текста запроса, который указан в правых скобках.

Описание основных конструкций запроса приводилось выше: арифметическое выражение (п. 4.1.4), системная переменная (п. 5.4.3), оператор движения (пп. 4.1.1—4.1.7), оператор присваивания (п. 4.2.2), обращение к подпрограмме (п. 4.3), оператор цикла (п. 4.2.3), условный оператор (п. 4.1.8).

4.8. Работа с несколькими базами данных и словарями

4.8.1. Работа со словарем. Словарные базы данных (словари) используются для сжатия базы и кодирования данных. Связки выражений, составляющие словарь, образуются из выражений-синонимов: сокращенного кода, наименования, краткого наименования и т. д. Обращение к словарю содержит код из первой позиции связки либо, при неавтоматической связи со словарем, значение из определенной позиции связки, объявленной ключевой. Результат обращения содержит второй либо произвольный указанный элемент связки.

Как упоминалось выше (см. пп. 1.4, 2.7.2), со словарем может быть организовано два вида связи — автоматическая и неавтоматическая. Автоматическая связь задается при описании данных: нужным вершинам ДОД присваивается тип «ссылки на словарь» (VOC, CODE, или RCODE), словарь в целом задается в ДОД спецификацией:

VOC/DDN=DD-имя словаря, VN=имя словаря/

Стандартный режим работы со словарем предполагает чтение второго элемента связки для вершин типа CODE, RCODE и чтение текста для вершин типа VOC. В этом случае не требуется обраче-

ния к специальным функциям: открытие словаря, работа с ним и закрытие происходят автоматически.

При наличии автоматической связи со словарем чтение первого элемента связки в вершинах типа CODE и RCODE производится с помощью специальной функции AIRQCODE. Обращение к функции имеет вид:

`%AIRQCODE(&X)`

где &X — рабочее поле. После выполнения функции в это поле заносится значение первого элемента связки (кода) текущей вершины (обращения к словарю при этом не происходит, так как первый элемент связки хранится непосредственно в базе данных). Например

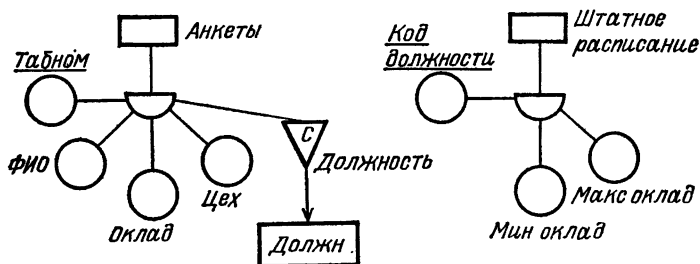


Рис. 4.19

пусть для базы данных (рис. 4.19) требуется распечатать список со-
рудников с указанием кода и полного наименования их должности,
Запрос имеет вид:

```
00 WSECT
01 Д[10]
00 TEXT
01 АНКЕТЫ.ALL.
02 ДОЛЖНОСТЬ %AIRQCODE(&Д)
02 %%PRINT('0',ТАБНОМ,ФИО,&Д,ДОЛЖНОСТЬ)
```

Работа с элементами словарной связки, кроме первого и второ-
го, а также работа со словарями, не связанными автоматически
с базой данных, ведется при помощи специальной функции VOC.
В этом случае открытие словаря, закрытие и работа с ним произво-
дятся с помощью обращений к этой функции. Команда открытия
словаря имеет вид:

`%VOC(&яс,'OPEN(I)',имя1,имя2)`

Здесь &яс — ячейка связи (рабочее поле типа F), идентифицирую-
щая конкретный словарь. При открытии в нее заносится адрес таб-
лицы, описывающей словарь;

'OPEN(I)' — режим работы со словарем, рассчитанный на чтение (существуют также режимы, рассчитанные на создание и обновление словаря, см. [40]);

имя1 — восьмисимвольное имя словаря, заданное рабочим полем или константой в кавычках;

имя2 — DD-имя словаря, задается аналогично.

В режиме чтения можно по любому ключевому элементу связи прочесть в словаре любой элемент, ключевой или неключевой. Соответствующее обращение к словарю имеет вид:

%VOC(&яс, 'WFPW', имя1, длина1, номер, имя2, длина2)

Здесь &яс — ячейка связи;

'WFPW' — режим работы со словарем;

имя1 — ключевой элемент связи, с которым делается обращение к словарю, задается константой или рабочим полем;

длина1 — длина ключевого элемента;

номер — номер нужного элемента связи, задается числом или рабочим полем типа F;

имя2 — рабочее поле, в результате выполнения функции в него заносится нужный элемент связи;

длина2 — длина значения нужного элемента (параметр можно опустить).

Закрытие словаря целесообразно производить для освобождения памяти.

Обращение на закрытие имеет вид:

%VOC(&яс, 'CLOSE')

В запросе можно открыть и одновременно использовать несколько словарей.

В обращении к словарю в режиме чтения необходимо указывать длину слова, с которым делается обращение. Так как эта длина не всегда известна, введена специальная функция вычисления фактической длины значения текстового рабочего поля. Обращение к функции имеет вид:

%REQLC(имя, длина)

Здесь имя — текстовое рабочее поле;

длина — рабочее поле типа F, в которое заносится фактическая длина значения первого параметра.

Пр и м е р. Пусть для базы данных рис. 4.19 требуется распечатать список должностей с указанием их кода и полного наименования. Запрос имеет вид:

00 WSECT

01 I[F], L[F], Д[9], ДОЛЖН[40]

00 TEXT

01 %VOC(&I, 'OPEN(I)', 'VOCNAME', 'DDVOC')

```

01 ШТАТНОЕ РАСПИСАНИЕ.ALL(&Д:=НКI),
02 %REQLC(&Д,&L)
02 %CLRWS(&ДОЛЖН)
02 %VOC(&I,'WFPW',&Д,&L,2,&ДОЛЖН)
02 %%PRINT('O',&Д,&ДОЛЖН)
01 %VOC(&I,'CLOSE')

```

4.8.2. Работа с несколькими базами данных. Для работы с несколькими базами данных в рамках одного запроса используется специальная функция СЗ, позволяющая «переключать» текущую точку запроса на другую БД. При этом каждая текущая точка идентифицируется адресом таблицы базы. Адрес запоминается в рабочей ячейке при помощи специальной программы СЗ. Таким образом, для работы средствами запроса с одной или несколькими текущими точками в одной или нескольких БД необходимы две программы: программа REQBASE, позволяющая запомнить адрес таблицы базы в рабочей ячейке, и специальная функция SWBASE, позволяющая менять текущую точку в соответствии с указанным адресом.

Программа REQBASE позволяет запомнить адрес таблицы базы, с которой в настоящий момент ведется работа, открыть новую базу, а также закрыть базу, если работа с ней закончена. Обращение к программе REQBASE имеет вид:

```
%REQBASE(&X, режим, DD-имя1, DD-имя2)
```

Здесь &X — рабочая ячейка; она содержит адрес таблицы базы после обращения к программе (если задан режим открытия базы) или до обращения (если задан режим закрытия);

режим — символьная константа в кавычках или рабочее поле; может иметь значения: опущен — запомнить в &X адрес таблицы базы, с которой в настоящий момент работает запрос, 'OPEN' — открыть базу данных по указанным DD-предложениям для ДОД и ДД, запомнить адрес таблицы базы в &X, 'CLOSE' — закрыть базу данных, адрес которой содержится в &X, 'REOPEN' — за одно обращение закрыть базу, адрес которой содержится в &X, и открыть новую базу, по указанным DD-предложениям ДОД и ДД, запомнить в &X новый адрес таблицы. Режимы 'OPEN(I)' и 'REOPEN(I)' задают открытие базы только на чтение данных;

имя1 — символьная константа или рабочее поле длины 8 байтов, по умолчанию 'DDTREE'. Задаёт DD-имя ДОД в режиме открытия базы.

имя2 — символьная константа или рабочее поле длины 8 байтов, по умолчанию 'DDTREE'. Задаёт DD-имя ДД в режимах открытия базы.

П р и м е р ы.

```
%REQBASE(&B)
```

После обращения на программу REQBASE в рабочем поле &B будет содержаться адрес таблицы базы, с которой запрос работает в данный момент.

```
%REQBASE(&B,'REOPEN(I)', 'DDTREE1')
```

Перед обращением в поле &B должен находиться адрес таблицы некоторой заранее открытой базы. После обращения эта база будет закрыта, в ячейке &B будет адрес таблицы базы, открытой по DD-именам DODTREE и DDTREE1. В эту базу будет запрещена запись.

```
%REQBASE(&B,'CLOSE')
```

Перед обращением в поле &B находился адрес таблицы базы. После обращения эта база данных закрыта.

Специальная функция SWBASE переключает выполнение запроса на новую текущую точку, заданную адресом таблицы базы в рабочем поле. Обращение к программе имеет вид:

```
%SPECREQ('SWBASE',&B) или %SPECREQ(12,&B)
```

Здесь &B — рабочее поле типа F, в котором хранится адрес таблицы базы. Новая текущая точка может быть расположена в той же самой или в другой БД. Для возвращения в исходную точку нужно снова воспользоваться функцией SWBASE.

Рекомендуется для каждой текущей точки зафиксировать рабочую ячейку, содержащую адрес соответствующей таблицы, а также выделять в блоки части запроса, относящиеся к фиксированной базе или к фиксированной текущей точке. Каждый такой блок начинать с переключения на данную базу и заканчивать обратным переключением, см. [87].

Пр и м е р. Пусть в базе данных (рис. 4.19) требуется распечатать фамилии работников, получающих больше максимального оклада. Запрос имеет вид:

```
00 WSECT
01 B1[F],B2[F]
01 ИД[5],ТАБНОМ[9],ФИО[40]
00 BLOCK B1
01 %SPECREQ(12,&B1)
00 BLOCK B2
01 %SPECREQ(12,&B2)
00 BLOCK НАЧАЛО
01 %REQBASE(&B1)
01 %REQBASE(&B2,'OPEN(I)', 'DOD1', 'DD1')
01 □B2.ШТАТНОЕ РАСПИСАНИЕ.□B1
00 BLOCK МАКСЗАР
01 □B2□ОБР□B1
```

```

00 BLOCK ОБР
01 #&ШД.(&МАКСОКЛ:=МАКС ОКЛАД)
00 TEXT
01 ○НАЧАЛО
01 АНКЕТЫ.ALL(&ТАБНОМ:=NKI).
02 (&ФИО:=ФИО)
02 ДОЛЖНОСТЬ %AIRQCODE(&ШД)
03 ○МАКСЗАР
02_IF &МАКСОКЛ <ОКЛАД
02_THEN %%PRINT('I',&ТАБНОМ,&ФИО)

```

4.9. Процедуры запросной системы

Основная схема работы с запросной системой характерна для реализации в ОС языков программирования высокого уровня. По этой схеме работа с запросом организуется в несколько этапов:

- трансляция запроса, результат которой — объектный текст — заносится в библиотеку ОС;

- сборка одного или нескольких объектных текстов запросов и, быть может, объектных текстов программ, написанных на других языках программирования. В результате образуется загрузочный модуль ОС;

- исполнение заранее оттранслированного запроса в качестве программы или подпрограммы.

Ниже будет рассматриваться упрощенный вариант схемы, по которому исходный текст запроса преобразуется непосредственно в загрузочный модуль при помощи специальной процедуры СЗ.

Трансляция запроса в загрузочный модуль осуществляется процедурой ISREQCL. В предложении EXEC можно задавать для нее следующие параметры:

R — объем памяти, по умолчанию 140K;

RASM — объем памяти для шага ASM, по умолчанию 128K;

LCARD — количество символов, которое читается из каждой записи, по умолчанию LCARD=72 (см. п. 4.4.2);

SR — имя библиотеки исходных модулей, в которой может храниться текст запроса, по умолчанию SR='SOURCE';

NAME — имя модуля в библиотеке исходных модулей SR;

LIB — имя библиотеки, в которую заносится загрузочный модуль, по умолчанию, LIB='LDPROG'.

Текст запроса берется из модуля NAME библиотеки SR, транслируется, редактируется и записывается в библиотеку LIB как загрузочный модуль под тем же именем NAME. Заметим, что наложенные значения имен библиотек SOURCE и LDPROG предполагают работу с библиотеками, созданными при установке системы ИНЕС. Рекомендуется для хранения пользовательских исходных

Текстов, объектных и загрузочных модулей создавать самостоятельные библиотеки.

Исходный текст запроса может вводиться с перфокарт (в этом случае библиотека &SR должна иметь модуль I#RQDUM, состоящий из единственной пустой записи). При вводе с перфокарт параметр NAME не задается.

Для исполнения оттранслированного запроса используется процедура ISREQGO. Процедура имеет параметры: NAME, SR, P, PU, R, DOD, DD. Процедура выполняет модуль NAME из библиотеки SR. Специальный параметр P задает режим печати исполнения запроса: P=1 — трасса движений по БД и доступа к данным, P=2 — аварийная печать интерпретатора СЗ, P=4 — трасса интерпретатора СЗ, P=<сумма> — заказ нескольких печатей. Параметр PU=текст считывается из запроса программой REQPARMO (см. п. 4.5.2). Объем памяти задается параметром R (по умолчанию R=128K). ДОД и ДД задаются, как обычно, параметрами DOD и DD, либо — DD-предложениями с именами DODTREE и DDTREE.

Для отладки запросов используется процедура ISREQCLG. Процедура берет исходный текст запроса из входного потока SYSIN, по LCARD символов из каждой записи. Запрос транслируется, редактируется во временную библиотеку под именем &GO и выполняется. В остальном процедура ISREQCLG устроена как «сумма» процедур ISREQCL и ISREQGO. Память для трансляции задается параметром RCP.

Содержание не бесформенно, а форма одновременно и содержится в самом содержании и представляет собою нечто внешнее ему.

Гегель

Г л а в а 5

ВЫВОД ДАННЫХ

5.0. Введение

5.0.1. Классификация документов по формам их представления. Результаты обработки данных, извлеченных из БД, оформляются в виде совокупности документов разнообразного вида и назначения. В развитых приложениях возникает необходимость обеспечить множество (до нескольких сотен) пользовательских представлений данных, выраженных в формах выходных документов. Фактически отображение БД в выходные документы составляет основное содержание ее приложений.

СУБД, обладающие развитыми языками запросов, средствами этих языков обеспечивают получение произвольных структур данных, соответствующих пользовательским представлениям. Задачей подсистемы вывода является, главным образом, оформление полученных результатов, т. е. представление их в виде документов подводящей формы.

Таким образом, подсистема вывода должна обеспечить представление структур данных в любых видах реально существующих (машинно-представимых) документов. В общем случае под документом здесь может пониматься произвольный структурированный текст, представимый на алфавитно-цифровых печатающих устройствах (АЦПУ). Структура текста — это структура взаимосвязанных данных, составляющих содержание документа. Условие представимости на АЦПУ накладывает ограничения на оформление текста (форму документа) и позволяет отделить документы от машинной графики.

Содержание документа может быть представлено весьма сложными структурами данных (см. п. 3.0.1). При этом документ с одним и тем же содержанием может быть оформлен многими разными способами. Как мы уже отмечали в п. 3.0.1, на практике выбирается не-

которая форма документа, наиболее адекватная его содержанию и использованию. В тех приложениях, где документ выводится и на АЦПУ, и на дисплей, бывает целесообразно выделить особые формы для того и другого представлений, специальные формы могут существовать также для передачи по каналам связи и хранения исходных

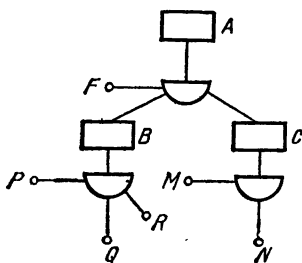


Рис. 5.1

данных и т. д. Несмотря на бесконечное разнообразие конкретных форм представления, их можно разбить на четыре класса: *последовательности, таблицы, иерархии, смешанное представление*. В настоящем разделе рассматриваются особенности этих классов на формальных и реальных примерах. В этой главе описываются методы вывода документов разных классов средствами ИНЕС.

Рассмотрим документ, содержание которого имеет структуру, представленную на рис. 5.1 (при изображении структуры содержания выходного документа мы будем пользоваться обозначениями

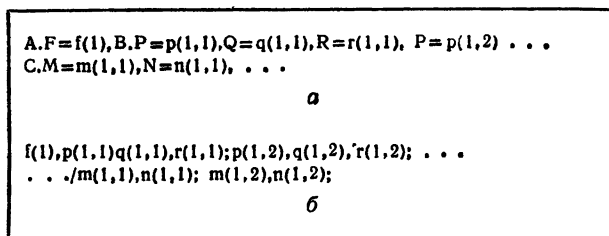


Рис. 5.2. Представление данных рис. 5.1 в виде последовательностей: все данные выводятся а) с именами, б) без имен

глав 1, 2, 3). Представление документа в виде линейной *последовательности* (составного текста) состоит в дополнении последовательности данных поясняющими словами, наименованиями и разделителями, которые позволяют однозначно понять смысл текста. Такое представление отличается, как правило, наибольшей компактностью, особенно в тех случаях, когда БД слабо заполнена. Именно благодаря этому свойству оно эффективно для передачи по каналам связи, регулярно применяется при ручной подготовке (набивке) документов, широко используется в музейных каталогах. Возможный вариант представления данных рис. 5.1 в виде последовательности изображен на рис. 5.2. На этом рисунке и в дальнейшем большими буквами будут обозначаться имена данных (A, F, B, P, Q

- * [25]. ИНВ.НОМЕР 37164
* КОРОБОЧКА. СЕРЕБРО. ЗАПАДНАЯ ЕВРОПА. 19-Й ВЕК. ОТДЕЛ
* ДРАГОЦЕННЫХ МЕТАЛЛОВ. ПОСТУПЛЕНИЕ 1899 ГОДА, ДАР
* ОТ КРЕСТЬЯНИНА П. С. КУЗНЕЦОВА. ВЕС: 51.8 Г, ВЫСОТА:
* 3.6 СМ, ДЛИНА: 5.0 СМ, ШИРИНА: 5.0 СМ. ДРАГ. МЕТАЛЛ;
* СЕРЕБРО — 50.8 Г. ДРАГ.КАМЕНЬ: АЛЬМАНДИН, ГОРНЫЙ
* ХРУСТАЛЬ, МАТЕРИАЛ УКРАШЕНИЯ: ЗОЛОТО, ГОРНЫЙ
* ХРУСТАЛЬ, АЛЬМАНДИНЫ, КАМНИ. ТЕХНИКА
* ИЗГОТОВЛЕНИЯ: КОВКА, ЧЕКАНКА, РЕЗЬБА; УКРАШЕНИЯ;
* ЗОЛОЧЕНИЕ; СОЕДИНЕНИЕ ДЕТАЛЕЙ ШАРНИРНОЕ.
*
* ВНЕШНИЙ ВИД: КОРОБОЧКА С КРЫШКОЙ,
* ВОСЬМИЛОПАСТНАЯ, С ШИРОКИМ НИЗОМ, СУЖИВАЮЩАЯСЯ
* КВЕРХУ, КРЫШКА НА ШАРНИРЕ.
*
* ИЗОБРАЖЕНИЯ: НА ДНЕ, В ШИТКЕ — БУКЕТ ЦВЕТОВ;
* ТУЛОВО — ЛИСТЬЯ; ОРНАМЕНТЫ; ТУЛОВО — ЗАВИТОК;
* ТУЛОВО — ПАЛЬМЕТКА;
*
* *****
* [26]. ИНВ. НОМЕР 37940
* ЧАРКА. СЕРЕБРО. МОСКВА. 1693. ОТДЕЛ ДРАГОЦЕННЫХ
* МЕТАЛЛОВ. ПОСТУПЛЕНИЕ 1900 ГОДА, КУПЛЕНА ОТ
* ГРИГОРИЯ ФРОЛОВА. ВЕС: 82.1 Г, ВЫСОТА: 3.2 СМ, ДЛИНА:
* 8.8 СМ, ШИРИНА: 8.8 СМ. ДРАГ. МЕТАЛЛ: СЕРЕБРО — 82.1 Г.
* МАТЕРИАЛ УКРАШЕНИЯ: ЗОЛОТО. ТЕХНИКА ИЗГОТОВЛЕНИЯ:
* ЛИТЬЕ, РЕЗЬБА, ЧЕКАНКА.
*
* ВНЕШНИЙ ВИД: ЧАРКА НА ПОДДОНЕ С ПРОРЕЗНОЙ
* РУЧКОЙ — ПОЛКОЙ.
*
* ИЗОБРАЖЕНИЯ: ТУЛОВО, ДНО — ПТИЦА; ТУЛОВО, ДНО —
* РЫБЫ; ТУЛОВО, ДНО — ИЗОБРАЖЕНИЕ КИТА,
* ПОГЛОЩАЮЩЕГО ИОНУ; РУЧКА — ЛЕВ; РУЧКА —
* ИЗОБРАЖЕНИЕ ЛЬВА И ЕДИНОРОГА, ДЕРЖАЩИХ ШИТ;
* ОРНАМЕНТЫ; РУЧКА — ЗАВИТОК; КЛЕЙМА; ГОРОД МОСКВА,
* 1693—1694.
* ВЫСТАВКИ, 1. ЭКСПОЗИЦИИ: ДОМ БОЯРИНА, 1936 ГОД, ГИМ.
*
* *****
* [27]. ИНВ.НОМЕР: 41873
* СТОПА. СЕРЕБРО. МОСКВА. 17-Й ВЕК, КОНЕЦ. ОТДЕЛ
* ДРАГОЦЕННЫХ МЕТАЛЛОВ. ПОСТУПЛЕНИЕ 1903 ГОДА,
* КУПЛЕНА ОТ ЯКОВА ЧЕРНОМОРДИКА. ВЕС: 434.5 Г,
* ВЫСОТА: 22.8 СМ. ДРАГ.МЕТАЛЛ: СЕРЕБРО — 434.5 Г.
* МАТЕРИАЛ УКРАШЕНИЯ: ЗОЛОТО. ТЕХНИКА ИЗГОТОВЛЕНИЯ;
* КОВКА, РЕЗЬБА; УКРАШЕНИЯ; ЗОЛОЧЕНИЕ.
*
* ВНЕШНИЙ ВИД: СТОПА ВОСЬМИГРАННАЯ, С РАСТРУБОМ
* КВЕРХУ. НАДПИСИ: ВЕНЕЦ СТОПЫ — НАДПИСЬ: СТАКАН
* ВЕЛИКИХ ГОСУДАРЕЙ КРЕСТОВОГО ДИАКА И РИЗНИЧАГО
* АНТОНА АЛЕКСЕЕВА СЫНА МУРОМЦОВА ПИТИ ИЗ НЕГО
* НА ЗДРАВЬЕ И НА ВЕСЕЛИЕ (КИРИЛЛИЦА, ВЯЗЬ). ГРАНЬ
* СТОПЫ — НАДПИСЬ: ИМЯ СИВИЛЛЫ И ЕЕ ИЗРЕЧЕНИЕ.
* ИЗОБРАЖЕНИЕ И НАДПИСЬ НА КАЖДОЙ ГРАНИ
* (КИРИЛЛИЦА, ВЯЗЬ). ИЗОБРАЖЕНИЯ: ПО ГРАНЯМ СТОПЫ —
* ИЗОБРАЖЕНЫ ВОСЕМЬ СИВИЛЛ; ОРНАМЕНТЫ: ПО ГРАНЯМ
* СТОПЫ — ЗАВИТКИ ОБРАЗУЮТ ОВАЛЬНЫЕ КАРТУШИ;

Рис. 5.3. Страница музейного каталога. Пример печати фрагмента БД в виде последовательности

и т. д.), малыми буквами — значения данных ($f(i)$, $p(i,j)$, $q(i,j)$, $v(i,j)$ и т. д.).

В последовательности рис. 5.2а все данные снабжены именами, в последовательности рис. 5.2б представлены только данные, разделенные специально выделенными знаками: «,» — переход к следующему данному, «;» — конец структуры, «/» — конец массива структур.

Реальные документы с данными, представленными в виде последовательностей, приводятся на рис. 5.3 (музейный каталог), в примерах 1—6 п. 3.4.2, в работе ([58]). Таким же образом выдаются служебные характеристики сотрудников на основе сведений кадровой БД (см. Конев В. С. Аттестация специалистов на ЭВМ// Банки данных в АСУ: Труды школы ВДНХ.— Калинин: НПО ЦПС, 1986.— С. 154—158).

Часто применяется *табличное* (двумерное) представление данных, отличающееся наглядностью. Замечательным его свойством является возможность производить инверсию двух индексов (ключей): действительно, мы фиксируем либо один, либо другой индекс, двигаясь по столбцу или строке таблицы.

Можно выделить следующие виды таблиц:

а) Таблица фиксированного формата с постоянными, не зависящими от конкретного содержания размерами и расположением разграфки — см. пример на рис. 5.4.

б) Таблица с промежуточными заголовками. Под каждый заголовок отводится полоса в одну или несколько строк с одним или несколькими окнами. Иерархия заголовков убывает сверху вниз. Данные рис. 5.1 представлены таким способом на рис. 5.5 (см. также пример документа на рис. 5.6).

в) Таблица с заголовками горизонталей. Заголовки распечатываются в документе в виде столбцов. Иерархия заголовков убывает, как правило, слева направо, подчиненная информация располагается в правой части документа. Данные рис. 5.1 представлены в таком виде на рис. 5.7 (см. также документ рис. 5.8).

г) Таблица с заголовками вертикалей. В документах этого вида несколько значений одного заголовка располагаются в ряд, так что подчиненная информация занимает столбец, как правило, под заголовком. При этом заголовку может подчиняться несколько подзаголовков, расположенных в ряд и разбивающих текст на подзаголовки. Данные рис. 5.1 представлены в виде такой таблицы на рис. 5.9. Пример такого же оформления данных приводится в документе рис. 5.10.

Оформление данных в виде многоуровневой *иерархии* позволяет выявить сложную структуру данных. Суть такого представления заключается в том, что каждое данное выводится в виде отдельной записи, снабженной явно указанным уровнем или местом записи

ГОРОД: ЛЕНИНГРАД
 ВЫ РАБОТАЕТЕ С ИНЕС С 1979 Г.
 НАЗВАНИЕ ОРГАНИЗАЦИИ:
 БАЛТИЙСКОЕ МОРСКОЕ ПАРОХОДСТВО
 ОТВЕТСТВЕННЫЙ ЗА ВНЕДРЕНИЕ: КОНЕВ В С.
 ТЕЛЕФОН: 259-82-82

ВАШ ШИФР: ** БМП **
 ВЗЯТО ВЕРСИЙ;

НАЗВАНИЕ ПОДСИСТЕМЫ: КАДРЫ, ТРУД, ЗАРПЛАТА
 НАЗВАНИЕ АСУ: АСУ МОРФЛОТ

1. ТИП ПОДСИСТЕМЫ

2. МАСШТАБЫ ПРИМЕНЕНИЯ

3. СТАДИЯ РАЗРАБОТКИ

4. КОЛИЧЕСТВЕННЫЕ ХАР-КИ БД:

КОЛИЧЕСТВО ДЕРЕВЬЕВ В БАЗЕ

КОЛИЧЕСТВО ВСЕХ ВЕРШИН ДОД

КОЛИЧЕСТВО ССЫЛОК 'REF'

ОБЪЕМ ПАМЯТИ БАЗЫ (В МГБ)

ВХОДНЫЕ ДОКУМЕНТЫ:

А) КОЛ-ВО ФОРМ

Б) ПОТОК ВВОДА ДОКУМЕНТОВ

ВЫХОДНЫЕ ДОКУМЕНТЫ:

А) КОЛ-ВО ФОРМ

Б) ПОТОК ВЫВОДА ДОКУМЕНТОВ

КОЛИЧЕСТВО СЛОВАРЕЙ

ОБЪЕМ ПАМЯТИ СЛОВАРЕЙ (В МГБ)

КАК ЧАСТО РЕШАЕТСЯ ЗАДАЧА

ИНТЕНСИВНОСТЬ ОБНОВЛЕНИЯ БД

ИНТЕНСИВНОСТЬ ИСПОЛЬЗОВАНИЯ

ДИАЛОГА

АДМИНИСТРАТИВНЫЙ УРОВЕНЬ

ИСПОЛЬЗОВАНИЯ ДИАЛОГА

5. ТРУДОЕМКОСТЬ РАЗРАБОТКИ:

ВСЕГО ЛЮДЕЙ

В ТОМ ЧИСЛЕ ПРОГРАММИСТОВ

В ТОМ ЧИСЛЕ СИСТЕМНЫХ ПР-В

ВСЕГО ЗАТРАЧЕНО ВРЕМЕНИ

В ТОМ ЧИСЛЕ НА ПРОГР-НИЕ

6. ЭКОНОМИЧЕСКИЙ ЭФФЕКТ (РУБ)

7. СРОК СДАЧИ В ЭКСПЛУАТАЦИЮ

8. КАКИЕ ЧАСТИ ИНЕС

ИСПОЛЬЗУЮТСЯ

9. ДРУГИЕ СУБД

КОМПЛЕКСНАЯ АСУ
 НА ЕДИНОЙ БД
 ОРГАНИЗАЦИЯ
 ПРОИЗВОДСТВЕННАЯ
 ЭКСПЛУАТАЦИЯ

10

1300

10

75

30

15000/МЕС

120

11000/МЕС

1

4

ЕЖЕДНЕВНО

3%/ДЕНЬ

—

—

22

11

—

42 МЕС

42 МЕС

300 ТЫС

1982 ГОД

ВСЕ ЧАСТИ ИНЕС

НЕТ

Рис. 5.4. Учетная карточка подсистемы, разработанной средствами СУБД ИНЕС. Пример таблицы фиксированного формата

в структуре. Это представление применяется в языках программирования (кобол, ПЛ/1) для описания данных, оно широко используется в ИНЕС. Например, этот способ представления данных

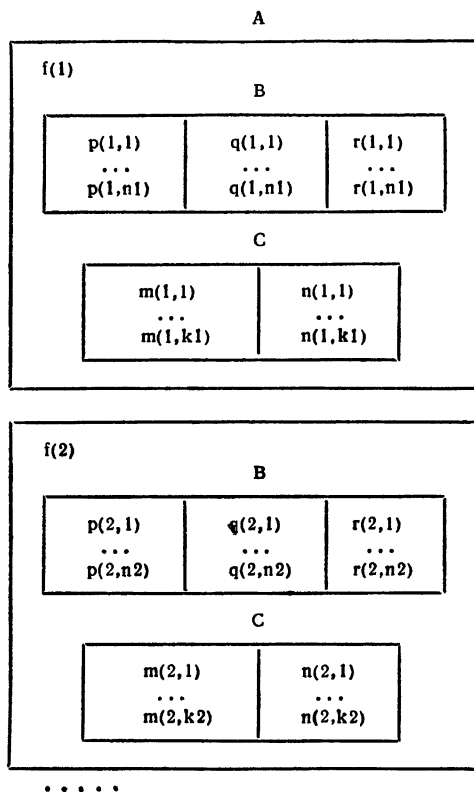


Рис. 5.5

используется в процедуре ISPRBASE для распечатки значений БД (см. рис. 5.11).

В любом тексте, в котором заголовки разделов снабжаются уникальными номерами (например, в описании, составленном по стандартам единой системы подготовки документации), применяется по существу иерархический способ оформления.

Данные рис. 5.1, представленные таким способом, изображены на рис. 5.12. На рис. 5.12а каждое данное имеет номер уровня (расстояние от корня), на рис. 5.12б каждое данное снабжено уникальным десятичным номером, которое однозначно определяет место этого данного в иерархии.

РЕСПУБЛИКА: РСФСР
ГОРОД: МОСКВА
ТИП ПОДСИСТЕМЫ: КАДРЫ

ИНДЕКС	НАИМЕНОВАНИЕ	МАСШТАБ
ВЗМИ	КАДРОВЫЙ СОСТАВ МИНВУЗА РСФСР	ОТРАСЛЬ
ВНИИЭТО	КАДРОВЫЙ УЧЕТ	ОРГАНИЗАЦИЯ
ВЭИ	КАДРЫ	ОРГАНИЗАЦИЯ
ГВЦ ЦСУ	ПЕРСОНАЛЬНЫЙ УЧЕТ ЛИЦ, ИМЕЮЩИХ УЧЕНУЮ СТЕПЕНЬ ДОКТОРА НАУК	СТРАНА
ГИПИЛКП	ТРУД И ЗАРАБОТНАЯ ПЛАТА	ОТРАСЛЬ
МИНГ	КАДРЫ И СТАТИСТИКА	ОРГАНИЗАЦИЯ
МИФИ	КАДРЫ	ОРГАНИЗАЦИЯ
МИСИ	КАДРЫ	ОРГАНИЗАЦИЯ
МИСИО	КАДРЫ	ОРГАНИЗАЦИЯ

ТИП ПОДСИСТЕМЫ: БУХГАЛТЕРСКИЕ РАСЧЕТЫ

ИНДЕКС	НАИМЕНОВАНИЕ	МАСШТАБ
ВНИИЭТО	РАСЧЕТ ЗАРАБОТНОЙ ПЛАТЫ	ОРГАНИЗАЦИЯ
ГВЦММФ	УЧЕТ ТРУДА И ЗАРАБОТНОЙ ПЛАТЫ	ОТРАСЛЬ
ГИМ	УЧЕТ ТРУДА И ЗАРАБОТНОЙ ПЛАТЫ	ОРГАНИЗАЦИЯ
ИПУ	РАСЧЕТ ЗАРАБОТНОЙ ПЛАТЫ	ОРГАНИЗАЦИЯ
МОЩНИИС	УЧЕТ КАССОВЫХ ОПЕРАЦИЙ	ОРГАНИЗАЦИЯ
ЦГЭ	ФОНД ЗАРАБОТНОЙ ПЛАТЫ	ОРГАНИЗАЦИЯ

.....

ГОРОД: НОВОСИБИРСК
ТИП ПОДСИСТЕМЫ: ЗАДАЧИ САПР

ИНДЕКС	НАИМЕНОВАНИЕ	МАСШТАБ
ВЦСОАН	СИСТЕМА ПОИСКОВОГО КОНСТРУИРОВАНИЯ	ОРГАНИЗАЦИЯ
НГУ	КОМПЛЕКС ЗАДАЧ САПР	ОРГАНИЗАЦИЯ

.....

Рис. 5.6. Пример таблицы с промежуточными заголовками

На практике, как правило, применяется четвертый класс представления данных — *смешанное* представление данных — сочетание первых трех классов представления в одном документе. Часть заголовков указывается в виде иерархии отдельных уровней, другие по способам б), в), г) представления таблиц. В окна таблиц могут помещаться как единичные данные, так и сложные составные тексты, набранные из совокупности простых данных с поясняющими словами и разделителями.

А

f(1)	p(1, 1)	Q	q(1, 1)
	p(1, 2)	R	r(1, 1)
		Q	q(1, 2)
		R	r(1, 2)
	p(1, n1)	Q	q(1, n1)
		R	r(1, n1)
	m(1, 1)		n(1, 1)
	m(1, k1)		n(1, k1)
f(2)	p(2, 1)	Q	q(2, 1)
	p(2, 2)	R	r(2, 1)
...

Рис. 5.7

Данные рис. 5.1 представлены таким смешанным способом на рис. 5.13. Здесь F и f(i) представлены в иерархическом виде; p(i,j), m(i,j) — заголовками слева; B,C,P,M,N,Q,R — заголовками сверху; q(i,j), r(i,j) — последовательностями (см. также документы, изображенные на с. 246 и 217).

5.0.2. Применение макетного метода в системе вывода. Как мы уже отмечали, все документы рис. 5.2, 5.5, 5.7, 5.9, 5.12, 5.13 имеют одно и то же содержание (рис. 5.1). Все эти документы относятся к разным классам представления одних и тех же данных. Для того чтобы выбрать класс представления, нужно решить, какие данные будут представлены последовательностями, какие в виде уровней, как будут представлены заголовки в таблицах. В то же время выбор класса документа еще не определяет однозначно его форму. В рамках одного класса можно выбрать, например, разную ширину колонок таблицы, ввести те или иные поясняющие слова и разделители.

В ИНЕС оформление документа определяется макетом документа (см. п. 5.4.1) и описанием правил набора сложных текстов (по-

РЕСП	ГОРОД	ТИП ПОДСИСТЕМЫ	ВСЕГО
УССР	ДНЕПРОПЕТРОВСК	КАДРЫ	1
		НОРМАТИВНО-СПРАВОЧНАЯ	1
		ИНФОРМАЦИЯ	1
		НОМЕНКЛАТУРНЫЕ ЗАДАНИЯ АСУ ВУЗ	2
	ДОНЕЦК	БУХГАЛТЕРСКИЕ РАСЧЕТЫ	2
		ИСС	2
	ЗАПОРОЖЬЕ	КОНТРОЛЬ ИСПОЛНЕНИЯ	1
		ИНЖЕНЕРНЫЕ И НАУЧНЫЕ РАСЧЕТЫ	2
	КИЕВ	КАДРЫ	2
		КОНТРОЛЬ ИСПОЛНЕНИЯ	4
		ПЛАНОВЫЕ РАСЧЕТЫ	2
		НОРМАТИВНО-СПРАВОЧНАЯ	3
		ИНФОРМАЦИЯ	
		НОМЕНКЛАТУРНЫЕ ЗАДАНИЯ	2
		ОПЕРАТИВНОЕ УПРАВЛЕНИЕ	3
		ТЭПП	1
		СТАТИСТИЧЕСКАЯ ОТЧЕТНОСТЬ	2
		КОМПЛЕКСНЫЕ АСУ	2
ИСС		2	
ИПС		1	
ХАРЬКОВ	МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ	1	
	СНАБЖЕНИЕ		
	КОНТРОЛЬ ИСПОЛНЕНИЯ	1	
	ИСС	1	
	ЗАДАЧИ САПР	1	
	КОМПЛЕКСНЫЕ АСУ	1	

Рис. 5.8. Пример таблицы с заголовками горизонталей

А

f(1)								f(2)
p(1, 1)		...	p(1, n1)		m(1, 1)	...	m(1, k1)	...
Q	R	...	Q	R	N	...	N	...
q(1, 1)	r(1, 1)	...	q(1, n1)	r(1, n1)	n(1, 1)	...	n(1, k1)	

Рис. 5.9

МИНИ- СТЕР- СТВО	АН СССР								МИНВУЗ ...			
НАИ- МЕНО- ВАНИЕ ТИПА П/С	ИПС		П/С АСУП		ИСС		САПР		П/С АСУП		ИСС ...	
ВНЕД- РЕНИЕ ИНЕС	ДЕЛ АЮТ	ВНЕ ДР	ДЕЛ АЮТ	ВНЕ ДР	ДЕЛ АЮТ	ВНЕ ДР	ДЕЛ АЮТ	ВНЕ ДР	ДЕЛ АЮТ	ВНЕ ДР	ДЕЛ АЮТ	ВНЕ ДР ...
КОЛИ- ЧЕСТ- ВО ОР- ГАНИ- ЗАЦИЙ	2	1	1	0	1	0	1	0	13	11	2	2 ...

Рис. 5.10. Пример таблицы с заголовками вертикалей

1	ГОРОД	КЛЮЧВ. МСВ	
2	Г	СТРУКТУРА	
3	НАЗВАНИЕ	КЛЮЧ РУССК. ТЕКСТ	МОСКВА
3	ПОЛЬЗОВАТЕЛИ	КЛЮЧВ. МСВ	
4	ПЛ	СТРУКТУРА	
5	ШИФР	КЛЮЧ РУССК. ТЕКСТ	ВНИИНО
5	АКТЫ ВНЕДРЕНИЯ	КЛЮЧВ. МСВ	
6	#0	СТРУКТУРА	
7	ДАТА	КЛЮЧ ТЕКСТ	82.12.30
7	ВНЕДРЕНИИ	ТЕКСТ	П
6	#0	СТРУКТУРА	
7	ДАТА	КЛЮЧ ТЕКСТ	83.04.18
7	ВНЕДРЕНИИ	ТЕКСТ	П
7	ЭФФЕКТ	ЧИСЛО	70000
4	ПЛ	СТРУКТУРА	
5	ШИФР	КЛЮЧ РУССК. ТЕКСТ	ВЦ МЕТРО
5	АКТЫ ВНЕДРЕНИЯ	КЛЮЧВ. МСВ	
6	#0	СТРУКТУРА	
7	ДАТА	КЛЮЧ ТЕКСТ	83.09.28
7	ВНЕДРЕНИИ	ТЕКСТ	О
7	ЭФФЕКТ	ЧИСЛО	115752
..	..		

Рис. 5.11. Пример печати фрагмента БД в уровневом виде

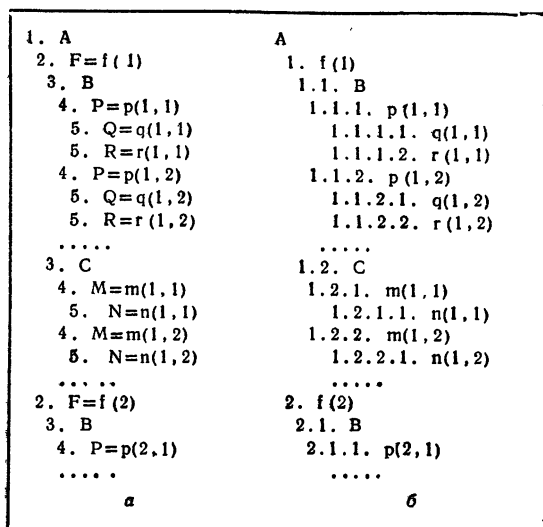


Рис. 5.12. Представление данных рис. 5.1 в виде иерархии. а—каждое данное снабжено номером уровня (глубиной от корня); б—данные имеют уникальные десятичные номера

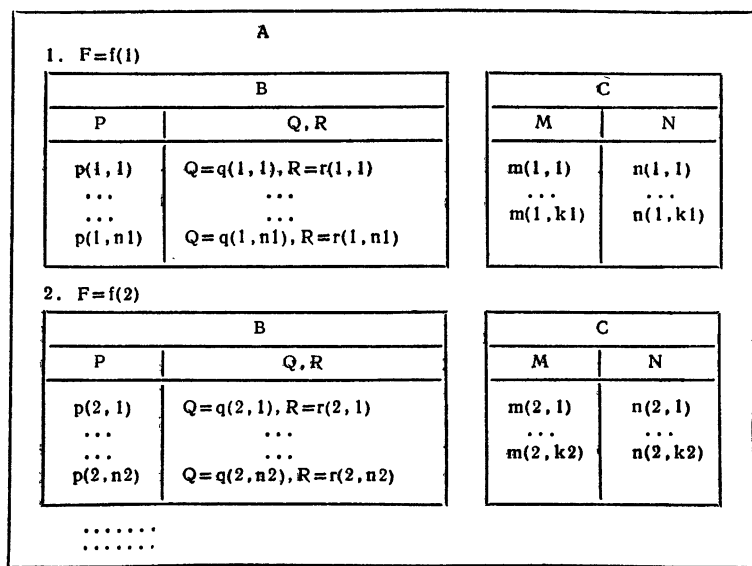


Рис. 5.13

следовательностей — см. д. 5.3). Содержание документа определяется а) программой генерации документа, которая кроме своей главной функции — выбора данных из БД, необходимых для заполнения окон макета, — определяет порядок вывода частей документа (см. п. 5.4.3), и б) описанием заполнения окон макета (см. п. 5.4.2).

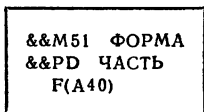


Рис. 5.14

К оформлению документа относится также описание размещения его на совокупности конечных страниц. Разбивка документа, превосходящего размеры страницы, в общем случае предполагает дополнительное оформление каждой страницы — распечатку ее заголовка и номера, оформление конца страницы, а также — при автоматической выдаче документа — соблюдение определенных правил

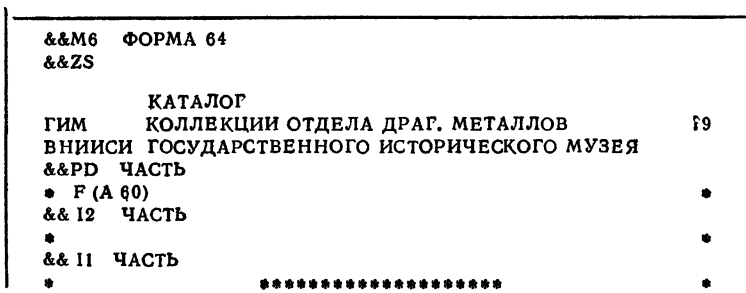


Рис. 5.15. Макет выходного документа рис. 5.3

расположения частей документа на листах (например, заголовок не может завершать страницу, определенные разделы документа должны печататься с новой страницы и т. п., см. п. 5.4.6 и [32]). Для документов рис. 5.2 и 5.3 главную роль играет описание правил набора текста (последовательности), макеты этих документов, представленные на рис. 5.14 и 5.15, выполняют роль указателей ширины для полосы вывода. Макет рис. 5.15 определяет также оформление страниц. Для документа рис. 5.4 макет, состоящий из одной части (см. рис. 5.16), полностью определяет его оформление. Макеты документов рис. 5.5 и 5.13 изображены на рис. 5.17 и 5.18. В приведенных примерах макетов окна задаются в виде ?9. . .9 или F(A целое). В первом случае символ «?» определяет начало окна, девятки —

ГОРОД: F(A30)
 ВЫ РАБОТАЕТЕ С ИНЕС С ?9999999
 НАЗВАНИЕ ОРГАНИЗАЦИИ:
 F(A60)
 ОТВЕТСТВЕННЫЙ ЗА ВНЕДРЕНИЕ: F(A33)
 ТЕЛЕФОН: ?99999999

ВАШ ШИФР:** F(A10)**
 ВЗЯТО ВЕРСИЙ: ?9

НАЗВАНИЕ ПОДСИСТЕМЫ: F(A40)
 НАЗВАНИЕ АСУ: F(A45)
 1. ТИП ПОДСИСТЕМЫ
 2. МАСШТАБЫ ПРИМЕНЕНИЯ
 3. СТАДИЯ РАЗРАБОТКИ
 4. КОЛИЧЕСТВЕННЫЕ ХАР-КИ БД:
 КОЛИЧЕСТВО ДЕРЕВЬЕВ В БАЗЕ
 КОЛИЧЕСТВО ВСЕХ ВЕРШИН ДОД
 КОЛИЧЕСТВО ССЫЛОК 'REF'
 ОБЪЕМ ПАМЯТИ БАЗЫ (В МГБ)
 ВХОДНЫЕ ДОКУМЕНТЫ:
 А) КОЛ-ВО ФОРМ
 Б) ПОТОК ВВОДА ДОКУМЕНТОВ
 ВЫХОДНЫЕ ДОКУМЕНТЫ:
 А) КОЛ-ВО ФОРМ
 Б) ПОТОК ВЫВОДА ДОКУМЕНТОВ
 КОЛИЧЕСТВО СЛОВАРЕЙ
 ОБЪЕМ ПАМЯТИ СЛОВАРЕЙ (В МГБ)
 КАК ЧАСТО РЕШАЕТСЯ ЗАДАЧА
 ИНТЕНСИВНОСТЬ ОБНОВЛЕНИЯ ВД
 ИНТЕНСИВНОСТЬ ИСПОЛЬЗОВАНИЯ
 ДИАЛОГА
 АДМИНИСТРАТИВНЫЙ УРОВЕНЬ
 ИСПОЛЬЗОВАНИЯ ДИАЛОГА
 5. ТРУДОЕМКОСТЬ РАЗРАБОТКИ:
 ВСЕГО ЛЮДЕЙ
 В ТОМ ЧИСЛЕ ПРОГРАММИСТОВ
 В ТОМ ЧИСЛЕ СИСТЕМНЫХ ПР-В
 ВСЕГО ЗАТРАЧЕНО ВРЕМЕНИ
 В ТОМ ЧИСЛЕ НА
 ПРОГРАММИРОВАНИЕ
 6. ЭКОНОМИЧЕСКИЙ ЭФФЕКТ (РУБ)
 7. СРОК СДАЧИ В ЭКСПЛУАТАЦИЮ
 8. КАКИЕ ЧАСТИ ИНЕС
 ИСПОЛЬЗУЮТСЯ
 9. ДРУГИЕ СУБД

F(A45)
 F(A45)
 F(A45)
 ?99
 ?9999
 ?9
 ?999
 ?99999
 ?99999/?999999
 ?99999
 ?99999/?999999
 ?9
 ?9.999
 F(A15)
 ?99% /?999999
 ?9.9ЧАС/?999999
 F(A40)
 ?99
 ?99
 ?9
 ?99 МЕГ
 ?99 МЕГ
 ?99999 ТЫС
 19?9 ГОД
 F(A45)
 F(A45)

Рис. 5.16. Макет документа рис. 5.4

размер окна. Во втором случае начало окна определяется символом F, длина — целым числом (см. п. 5.4.2).

Средства вывода ИНЕС в ряде случаев производят автоматическую генерацию макета. Это целесообразно, когда нет жестких требований к форме представления — задается только вид выводимого

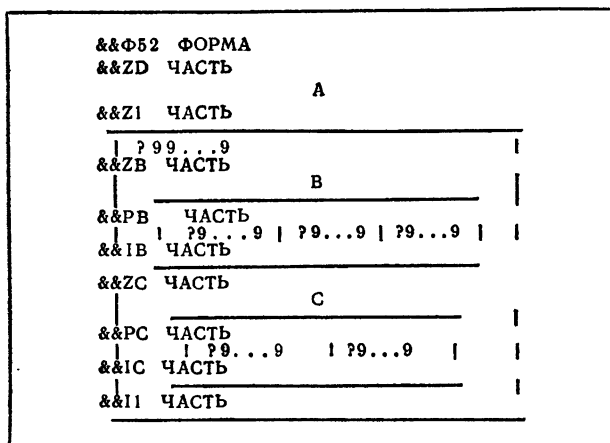


Рис. 5.17

документа. Так организован вывод последовательностей (см. п. 5.2, функция `%%PRINT('I', ...)`); таблиц с промежуточными заголовками (сочетание `%%PRINT('I', ...)` и `%%PRINT('O', ...)` — см. п. 5.2 и примеры из гл. 4); таблиц с заголовками строк и одноуровневыми заголовками колонок `%%PRINT('O', ...)`; уровневого представления БД, ДОД, ДОК (процедуры `ISPRBASE`, `ISPRDOD`, `ISPRDOC`), а также графического представления ДОД (см. рис. 5.19, процедура `ISGRDOD` в версии 3.5 и выше [35]).

5.0.3. Средства вывода СУБД ИНЕС. Система вывода ИНЕС включает в себя следующие основные компоненты (см. рис. 5.20):

Печать на АЦПУ всей БД или заданных ее частей в виде, соответствующем структуре хранения данных, с указанием имен, типов и значений переменных. Печать используется в основном для отладки описания данных и наполнения БД (процедура `ISPRBASE`, см. п. 5.1).

Экспресс-вывод значений переменных с их наименованиями в виде списков или таблиц; формат вывода генерируется автоматически. Экспресс-вывод применяется для получения справок из БД (программа `PRINT` с форматами 'O' и 'I', см. п. 5.2).

Программа набора сложных текстов предназначена для формирования составных текстов из совокупности данных и постоянных разделителей (программа EDIT, см. п. 5.3).

Комплекс средств макетного вывода предназначен для вывода документов сложной структуры. Эти средства используются из системы запросов (функция %%PRINT,— см. п. 5.4.4). В общем случае

```
&& ФОРМА
&&ZD ЧАСТЬ
```

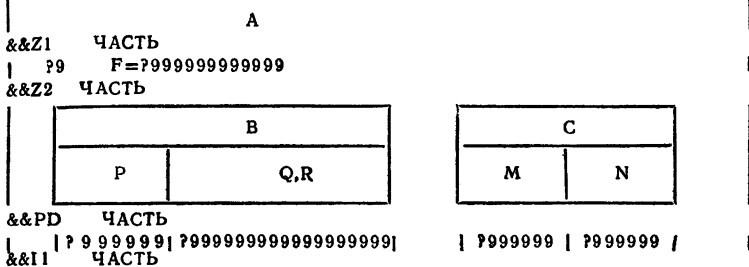


Рис. 5.18

они могут использоваться также из языков фортран, ПЛ/1, кобол (программа OUTPART, см. [39]).

Макетный вывод последовательных наборов данных осуществляется процедурами ISOUTFIL, ISOUTSRT (см. [39]). Для отладки макетов служит процедура ISMAKET (см. п. 5.4.9).

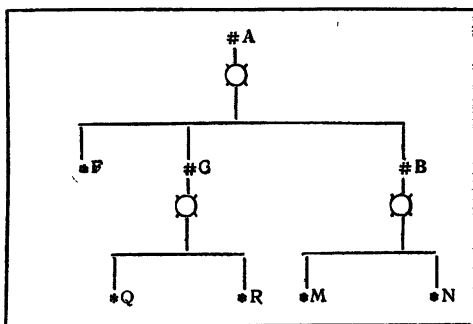
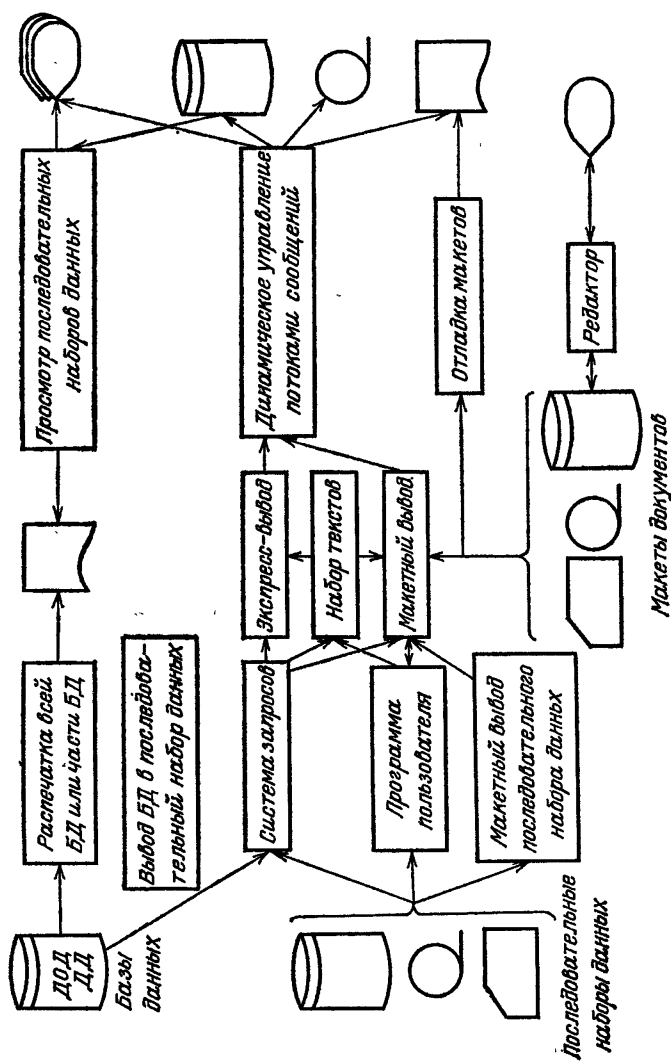


Рис. 5.19

Вывод значений переменных рабочей области запроса в последовательный набор данных, используется для получения из БД данных в требуемых форматах, например, для расчетных задач или



для последующих сортировок и распечаток (программа A2RQPUTS, см. п. 5.4.8).

С помощью системы динамического управления потоками сообщений (система ULINE — см. п. 5.5) любая система, разработанная для эксплуатации в пакетном режиме, может быть выполнена в режиме диалога.

Подсистема просмотра с дисплея последовательных наборов данных входит в состав диалоговых средств ИНЕС (см. [21, 42]).

5.1. Отладочная печать базы данных и ее частей

В процессе создания и эксплуатации БД возникает необходимость в контрольных распечатках отдельных частей базы. Простым средством получения таких распечаток является процедура ISPRBASE. Эта процедура печатает в одну строку информацию о каждой вершине ДД в следующем порядке:

- номер уровня вершины в ДД;
- имя вершины;
- номер элемента нумерованного массива или слово «ключ» для вершин, являющихся ключами;
- тип вершины в ДОД;
- значение терминальной вершины в ДД.

Если терминальная вершина создана в базе данных, но ей не присвоено значение, то на последнем месте печатаются два минуса. Если вершина не создана в БД, то о ней нет никакой печати.

Информация о вершинах распечатывается в порядке их обхода в ДД сверху вниз и слева направо, см. рис. 5.11. Следует отметить, что при отсутствии в ДОД спецификации ORDER=YES порядок вершин в структурах соответствует лексикографическому порядку их имен (по латинскому алфавиту), а не расположению в описании ДОД. Распечатка БД, полученная с помощью процедуры ISPRBASE, аналогична распечатке ДОД, полученной в результате трансляции или с помощью процедуры ISPRDOD.

Если требуется распечатать только часть БД или несколько частей, то следует определить эти части во входном наборе данных с именем SYSIN. Каждая подлежащая распечатке часть БД определяется в этом наборе с помощью траектории спуска, описанной между картами

./ ВНИЗ

и

./ ПУСК

Траектория спуска описывается с помощью траектории, причем каждое имя ДОД, номер или ключ пишутся на отдельной перфокарте (строке), начиная с первой позиции. Например, для рас-

печатки сведений о местах работы сотрудника Иванова (цех А17) из базы данных рис. 1.18 следует в набор SYSIN поместить следующее описание:

```
./ ВНИЗ  
ЗАВОД  
А17  
СОТРУДНИКИ  
ИВАНОВ  
РАБОТЫ  
./ ПУСК
```

Для ограничения распечатки уровней БД сверху и снизу служит карта вида:

```
./ УРОВЕНЬ n,m
```

где n — номер уровня, с которого следует начать распечатку БД, а m — номер уровня, до которого следует распечатывать базу данных.

Для распечатки штатного расписания из той же БД следует поместить в SYSIN следующую группу карт:

```
./ ВНИЗ  
ШТАТНОЕ РАСПИСАНИЕ  
./ УРОВЕНЬ 2,4  
./ ПУСК
```

По такому указанию распечатываются названия должностей с перечислением списков фамилий людей, их занимающих.

Процедура ISPRBASE имеет следующие параметры:

1. DOD=имя набора. Указывается DSN-имя набора данных, содержащего оттранслированный ДОД. По умолчанию DOD=&DOD.

2. DD=имя набора. Указывается DSN-имя набора данных, содержащего ДД. По умолчанию DD=&DD.

3. V=имя тома. С помощью этого параметра можно указать имя тома, на котором находятся некаталогизированные наборы с ДОД и ДД. Если наборы каталогизированы, то параметр можно не указывать.

4. VDOD=имя тома, VDD=имя тома. Эти параметры указываются в том случае, если наборы с ДОД и ДД размещены на разных томах.

5. P=REFSTOP. Указание этого параметра отключает автоматический переход по ссылкам типа REF. Если ДД имеет цикл, то это указание необходимо, так как иначе программа печати заикнется,

6. R=объем памяти. Указывается объем памяти в Кбайтах, необходимый для выполнения шага задания. По умолчанию R=130K.

Примеры обращений:

а) // EXEC ISPRBASE

Процедурой ISPRBASE осуществится полная распечатка БД из наборов данных с именами &DD и &DOD.

б) // EXEC ISPRBASE,DOD=DODBAS,DD=DDBAS,
// V=WORK

Задается печать БД, расположенной на томе WORK.

в) // EXEC ISPRBASE,DOD=DODBAS,DD=DDBAS,
// VDOD=WORK1,VDD=WORK2

Печать базы данных, для которой ДОД расположен на томе WORK1, ДД — на томе WORK2.

Задание на распечатку фрагментов базы, описанных в приведенных выше примерах, будет выглядеть следующим образом:

// EXEC ISPRBASE,DOD=DODBAS,DD=DDBAS

./ ВНИЗ

ЗАВОД

A17

СОТРУДНИКИ

ИВАНОВ

РАБОТЫ

./ ПУСК

./ ВНИЗ

ШТАТНОЕ РАСПИСАНИЕ

./ УРОВЕНЬ 2,4

./ ПУСК

//

Здесь задается печать БД о каталогизированных наборах данных ДОД и ДД.

5.2. Экспресс-вывод

Экспресс-вывод реализован в виде стандартной функции СЗ. Он позволяет выводить необходимые данные из БД и рабочих полей СЗ, а также результаты вычислений по арифметическим формулам. Экспресс-выводом мы уже пользовались в гл. 4 (см. п. 4.1.1 и др.), почти все запросы в примерах этой главы написаны с его использованием. В заказе на выдачу документа средствами экспресс-вывода указывается содержание документа: список через запятую данных и арифметических выражений. Кроме того, могут быть даны указания о формате вывода: вывести последовательностью или таблицей. При этом формат последовательности или таблицы генери-

руется автоматически. Формат вывода можно также задать по аналогии с фортранным оператором FORMAT.

Обращение к экспресс-выводу имеет вид:

%% PRINT(формат, список переменных)

где

формат::= $\left\{ \begin{array}{l} '0' \text{ — печать в виде таблицы} \\ '1' \text{ — печать в виде последовательности} \\ 'F(\text{фортранный формат})' \text{ — печать по указанному} \\ \text{формату} \end{array} \right\}^*$

список переменных::= фрагмент запроса[,фрагмент запроса]...

Обращение к такому оператору вызывает выполнение указанных фрагментов, причем соответствующее движение по ДД начинается от текущей точки на момент обращения. При выполнении этих фрагментов выводятся результаты вычисления указанных в них арифметических выражений, а также значения из рабочих полей СЗ и простых вершин БД, упомянутых во фрагментах запросов как терминальные (нужно отличать терминальную вершину дерева запроса от терминальной вершины БД. Терминальная вершина фрагмента запроса есть элемент запроса, у которого пуста подчиненная ветвь; ею может быть идентификатор рабочего поля или уход на подпрограмму — см. [37]).

Если в качестве фрагмента указывается имя составного рабочего поля, то выводятся все его простые подчиненные поля.

Примеры.

%% PRINT('1', НАИМЕНОВАНИЕ, АДРЕС)

Значения из вершин БД выводятся в виде списка (см. пример 3 п. 4.1).

%% PRINT('0', ФИО, ДОЛЖНОСТЬ, СПЕЦИАЛЬНОСТЬ,
ЗАРПЛАТА)

Значения из вершин БД выводятся в виде таблицы (см. пример 6 п. 4.1).

%% PRINT('1', ФИО, &ДОЛЖН, ЗАРПЛАТА,
DOWNROOT.ШТАТНОЕ РАСПИСАНИЕ. #&ДОЛЖН.
СРЕДНИЙ ОКЛАД)

Значения из простых вершин БД ФИО и ЗАРПЛАТА, рабочего поля ДОЛЖН и вершины СРЕДНИЙ ОКЛАД выводятся в виде списка (см. пример 23 п. 4.1.7).

%% PRINT('0', A, B, A+B, (A+B)/B*100)

*) Начиная с версии 3.5, формат можно не заключать в апострофы и не удваивать при этом апострофы внутри фортранного формата.

В виде таблицы будут выведены четыре числа: А и В (значения простых числовых вершин БД), а также $A+B$ и $(A+B)/100B$.

%%PRINT('1',&M)

Если М является структурным рабочим полем, то будут распечатаны все простые поля этой структуры. Если М является рабочим полем типа массив, то распечатываются все элементы массива.

При указании формата '0' данные выводятся в следующем виде:

имя 1	имя 2	...	имя п
данное 1	данное 2	...	данное п

Имя здесь означает последнее имя составного имени терминальной вершины (или составного имени рабочего поля), данное означает соответствующее значение вершины. Например, данное с составным именем А.В.С.Д и со значением 137.2 будет распечатано в виде:

D
137.2

Если данное в операции PRINT указывается как результат арифметических действий над терминальными вершинами и константами, то в качестве имени в заголовке таблицы распечатается арифметическое выражение над последними именами составных имен и константами. Например, значение выражения $(A.B.C + A.B.D) * 5$ при $A.B.C = 139$ и $A.B.D = 10$ будет выведено в виде:

$(C+D)*5$
745

Размеры колонок и вид редактирования значений данных при таком выводе можно задать значением спецификации PFORM (формата печати) в описании данных. Если спецификация PFORM для терминальной вершины ДД отсутствует, размер соответствующей колонки выбирается автоматически в зависимости от размеров имени и данного.

При многократном обращении к функции PRINT для выдачи одной и той же группы данных с изменяющимися значениями вывод производится следующим образом:

имя 1	имя 2	...	имя n
данные 1.1 данные 1.2 ...	данные 2.1 данные 2.2	данные n.1 данные n.2

Имена данных в этом случае будут печататься только в начале каждой страницы или при изменении набора данных.

При указании формата '1' данные выводятся в виде последовательности:

имя1=данные1; имя2=данные2; ... имяN=данныеN;

При выборе данных из массивов может возникнуть ситуация, при которой разные элементы массива, имеющие одно и то же последнее имя составного имени, выводятся одновременно. При выводе этих данных программой PRINT без указания формата или с форматом '0' происходит повторение одного и того же имени с разными значениями.

При указании формата '1' данные с одинаковыми именами выводятся в виде:

имя1=данные1.1,данные1.2,...,данные1.N1; имя2=данные2.1,данные2.2,...,данные2.N2; ..., имяM=данныеM.1,данныеM.2,...,данныеM.N;

При обращении %%PRINT('F(. . .)', список данных) данные из списка выводятся в соответствии с указанным форматом Фортрана.

Допустимы следующие спецификации:

'текст' — печать указанного текста;

[n]X — пропуск n позиций в строке, по умолчанию n=1;

Am — указание окна длиной m символов;

Fr.q — указание окна длиной r символов с q знаками после запятой;

/ — переход к следующей строке;

— пробел в начале строки вызывает переход к следующей строке;

1 — в начале строки вызывает переход к следующей странице;

n(фрагмент формата) — n-кратное повторение фрагмента формата.

При распечатке в окна формата заносятся только значения данных из списка, имена данных не учитываются. Если список переменных в обращении к PRINT пуст, то формат распечатается в пустыми окнами.

Примеры.

```
% % PRINT('F(1,///,21X,'СПРАВКА')')
```

По такому обращению будет подведено начало следующей страницы (так как первый байт в строке равен 1), пропущено три строки (по спецификациям «/») и в четвертой строке, начиная с 21-й колонки (спецификация 21X), будет напечатано слово СПРАВКА.

```
% % PRINT('F(3X,'ФАМИЛИЯ:',A30,/,3X,'ИМЯ:',  
A10,', ОТЧЕСТВО:', A10)',Ф,И,О)
```

Если в момент выполнения этой функции в переменных Ф,И,О находится текст ИВАНОВ ИВАН ИВАНОВИЧ, то будет напечатано:

```
ФАМИЛИЯ:ИВАНОВ  
ИМЯ:ИВАН      , ОТЧЕСТВО:ИВАНОВИЧ
```

Длина формата фортрана в обращении не должна превышать 255 символов. Формат большего размера можно занести в массив рабочих полей запроса и при обращении указать косвенно.

Пример.

```
**WSECT:(4F[250], . . . )  
(&F[1]:='F(формат')(&F[2]:='продолжение формата')  
(&F[3]:='конец формата')  
.  
.  
.  
% % PRINT(&F,список переменных)
```

Если возникает необходимость освободить память от комплекса PRINT, включая программы редактирования EDIT (см. ниже), то используется обращение вида % % PRINT('*'), по которому буфера уничтожаются и программы комплекса PRINT из памяти удаляются. При этом, если предыдущее обращение было % % PRINT('0'), то выдаваемая таблица завершается подчеркиванием. Подчеркивание таблицы делается также при переходе с печати по формату '0' на печать по формату '1'.

5.3. Набор составного текста в рабочем поле

Набор составного текста в рабочем поле-буфере производится с помощью функции EDIT. Для набора составного текста необходимо очистить поле-буфер и затем занести в него каждую составную

часть. Набранный текст можно выводить на печать при помощи системы макетного вывода либо экспресс-выводом.

Формат обращения для очистки буфера:

% EDIT('C', &буфер[, символ переноса])

Здесь 'C' — параметр, задающий в команде режим чистки; буфер — рабочее поле-буфер, предназначенное для накопления заносимых значений. Буфер описывается в разделе WSECT как составное поле, причем последние 4 байта используются системой как служебные (два байта отводятся под текущий адрес загрузки в буфере и один байт — под символ переноса текста).

Например, если описаны два буфера:

BUF(A[250], C[4]) и B(25A[180], S[H], D[2])

то команды очистки могут быть следующими:

% EDIT('C', &B)

или

% EDIT('C', &BUF, '/')

Формат обращения для записи в буфер имеет вид:

% EDIT('W', буфер[, разделитель до][, операнд заноса]
[, шаблон][, разделитель после])

Здесь 'W' — параметр, задающий режим заноса;

буфер — буфер заноса;

операнд заноса определяет заносимое значение. Задается рабочим полем или явно — в виде текста, заключенного в кавычки. Значение, задаваемое операндом заноса, заносится в буфер с текущей позиции. Последовательное применение команды в режиме 'W' с различными операндами заноса позволяет записать подряд значения этих операндов в буфер;

шаблон — указывает длину заносимого значения, а для чисел также количество заносимых знаков после запятой. Шаблон задается явно в апострофах либо как рабочее поле; имеет вид: M.N либо M, где M — длина заносимого значения, N — число знаков после запятой. Все заносимые рабочие поля выравниваются и заполняются пробелами по заданному шаблону: тексты — справа, числа — слева. Если шаблон не указан, дополнение значений пробелами не производится, у текстов пробелы в конце опускаются;

разделители — тексты, которые записываются в буфер до и после операнда заноса. Разделитель может задаваться символьным рабочим полем либо в явном виде (пробелы в начале и в конце разделителей не опускаются).

Примеры.

1. Распечатать в строчку сведения о сотруднике (рис. 1.18).
Запрос:

```
**WSECT:(ГОДР,ГОДОК,ЦЕХ[5],ФИО[20],НАИМ[30],
A(АНКЕТА[80],C[4]))
(&ЦЕХ:='A17')(&ФИО:='ПАВЛОВ Н.К.')
ЗАВОД.##&ЦЕХ.СОТРУДНИКИ.##&ФИО.
(&ГОДР:=ДАТА РОЖДЕНИЯ.ГОД)
ОБРАЗОВАНИЕ.ВЫСШЕЕ.
(&НАИМ:=ВУЗ.НАИМЕНОВАНИЕ)
(&ГОДОК:=ДАТА ОКОНЧАНИЯ.ГОД)
%EDIT('C',&A)
%EDIT('W',&A,' ЦЕХ: ',&ЦЕХ,',';)
%EDIT('W',&A,' ФИО: ',&ФИО,',';)
%EDIT('W',&A,' ДАТА РОЖДЕНИЯ: ',&ГОДР,','ГОД.')
%EDIT('W',&A,' ЗАКОНЧИЛ ',&НАИМ)
%EDIT('W',&A,' В ',&ГОДОК,',' ГОДУ.')
%%PRINT('I',&АНКЕТА)
```

Результат:

АНКЕТА= ЦЕХ: A17; ФИО: ПАВЛОВ Н.К.; ДАТА
РОЖДЕНИЯ: 1942 ГОД.ЗАКОНЧИЛ МИФИ В 1969 ГОДУ.

2. Распечатка графиков. Распечатать гистограмму численности сотрудников завода по годам их рождения (см. пример 32 п. 4.2.3). Задан массив рабочих полей 90КОЛ[F], в котором значение К-го элемента соответствует количеству сотрудников К-го года рождения, год задан двумя последними цифрами,

Запрос:

```
00 WSECT
01 90КОЛ[F]
01 BUF(N[100],C[4])
...
01 DO &K=1 TO 90;
02 %EDIT('C',&BUF)
02 %EDIT('W',&BUF,','&K,',' I')
02 DO &K1=1 TO &КОЛ[&K];
03 %EDIT('W',&BUF,','*)
02 %%PRINT('F(X,A75)',&N))
```

Результат:

```
...
31 I*****
32 I*****
33 I*****
```

34 I*****
35 I*****
36 I*****
37 I*****
38 I*****
39 I*****

...

В этом запросе значение переменной &N распечатывается оператором PRINT по фортранному формату.

5.4. Макетный вывод

Экспресс-вывод располагает ограниченными возможностями формирования выходных документов. Если требуемая форма документа не обеспечивается средствами экспресс-вывода, то следует обратиться к макетному выводу.

Подготовка вывода документа по заданному макету состоит из трех этапов:

- подготовка макета документа;
- подготовка заполнения окон макета;
- составление запроса, управляющего формированием и выводом документа.

Как правило, макет документа и запрос составляются независимо. При этом макет готовит сам заказчик, а запрос составляет администратор базы данных или программист, знающий содержание БД. Второй этап является промежуточным звеном между первым и третьим этапами и может быть выполнен достаточно подготовленным заказчиком. Такое деление дает возможность одновременно вести работу разных этапов, а также выводить одни и те же данные по разным формам, и наоборот, по одной и той же форме — разные данные.

Вывод документа осуществляется в процессе выполнения запроса.

Подготовка макета документа начинается с составления макета на бланках в том виде, в каком документ должен быть получен (см. язык описания макета в п. 5.4.2). Непосредственно с этих бланков производится набивка макета. При помощи процедуры ISMAKET выполняется контроль и распечатка макета (см. п. 5.4.9).

На втором этапе составляется описание заполнения окон макета, в котором указываются переменные или арифметические выражения, значения которых переносятся в макет. Это описание включается в запрос в виде специального раздела OUTFORM (см. п. 5.4.3).

Запрос осуществляет извлечение и обработку данных с последовательным выводом фрагментов (частей) документа при помощи функции PRINT (см. п. 5.4.4).

5.4.1. Макет документа. Для разработки макета документа необходимо разбить документ на части. Часть документа — это такая последовательность строк (горизонтальная полоса), которая в документе появляется как жесткое целое. Примерами частей являются: заголовок (шапка) документа, заголовки и окончания страниц, заголовки и итоги разделов документа и т. п.

Макет документа есть совокупность макетов его частей.

Макет документа содержит только описание формы документа, он отличается от документа следующим:

- вместо конкретных значений данных, извлеченных из ВД, в нем отмечены места (окна), в которые данные будут заноситься в процессе вывода;

- если какая-либо часть документа входит в документ многократно с различным наполнением окон, то в макете она описывается один раз;

- если какое-либо текстовое данное заносится в окно макета с переносами, то соответствующая строка макета выделяется в отдельную часть (см. печать каталога рис. 5.3 с помощью макета рис. 5.15).

- макет документа содержит вспомогательные записи, идентифицирующие форму документа и ее части.

Примеры документов и их макетов приводятся на рис. 5.3—5.5, 5.9, 5.15—5.18.

5.4.2. Подготовка макета выходного документа. В зависимости от длины строки документа строку макета можно изображать на одном, двух или трех сложенных рядом 80-колонных бланках для перфорации. После перфорации каждая строка макета собирается соответственно из одной, двух или трех перфокарт. При подготовке макета в диалоговом режиме используется (если позволяет редактор) одна длинная строка.

В текущей версии (версия 3.5) длина строки документа должна не превышать 160 байтов.

Первая строка макета имеет вид:

&&имя-формы ФОРМА [n],[m],[k]

Здесь имя-формы — любой набор до восьми символов.

Параметры n, m, k задают число символов, используемых соответственно на первой, второй и третьей перфокарте (строке дисплея) для изображения строки макета. Отсутствие этих трех параметров означает, что каждая строка макета изображается на двух перфокартах (строках), причем первые 70 символов располагаются на первой перфокарте, остальные (до 70) — на второй.

Макет каждой части документа начинается с идентифицирующей строки вида:

&&имя-части ЧАСТЬ [CH][,+k][,CK]

Здесь имя-части — любой набор из двух символов, используемый для идентификации описываемой части документа. Параметры CH и (или) CK обеспечивают смену страницы соответственно перед и (или) после размещения данной части документа. Если есть параметр +k, а параметр CH отсутствует, то производится протяжка бумаги на k строк перед выводом данной части документа. При наличии операторов CH и +k производится перемещение на k строк вперед на новой странице. При наличии параметра CK производится повод конца страницы после печати части.

Функционально выделены следующие части: заголовок документа — ZD, заголовок страницы — ZS, заголовки промежуточные — Z1—Z9, итоги промежуточные — I1—I9, период документа — часть, имя которой начинается с буквы P (например, PD или P1), конец страницы — KS, наполнитель страницы — FP, конец документа — KD.

Макет части документа изображается точно так же, как он выглядит в документе. Разграфка и постоянная информация (линии, подчеркивания, надписи и т. д.) каждой строки документа переносятся в соответствующую строку макета. Окна макета размечаются специальным образом.

Окно в макете начинается с символа «?» (вопросительный знак). Для размещения целой n-разрядной переменной после знака «?» помещается p—1 девяток (знак «—» занимает разряд, а знак «+» не печатается, если нет специального указания). Например, для записи в окно макета целого четырехзначного положительного числа окно должно иметь вид ?999. Если p=1, то окно задается одним знаком вопроса.

Для размещения действительной переменной с p целыми разрядами и k разрядами после десятичной точки в окне после символа «?» ставится p—1 девяток, затем десятичная точка и еще k девяток. Например, для записи действительного положительного числа с двумя целыми и двумя десятичными разрядами окно должно иметь вид ?9.99.

В описании окна для текстовой переменной в p символов после символа «?» ставится p—1 девяток. Например, для размещения в макете переменной из 6-ти позиций окно должно иметь вид ?99999.

В макетах для задания переменной и постоянной информации можно применять и форматы языка фортран. В этом случае в макете задается буква F, за которой в скобках помещается необходимый формат (см. п. 5.2). Например, строку макета:

НАИМЕНОВАНИЕ ГРУЗА : ?99999999999999

можно изобразить с использованием для описания переменной формата языка фортран:

НАИМЕНОВАНИЕ ГРУЗА : F(A16)

Горизонтальную черту в макете можно либо нарисовать, указав, например, 120 раз символ «—», либо закодировать: F(120'—').

5.4.3. Описание заполнения окон макета. Описание переменных, значения которых заносятся в окна макета, задается при обращении к функции PRINT (см. п. 5.4.4) или в специальном разделе запроса с именем OUTFORM. Формат раздела OUTFORM имеет вид:

00 OUTFORM имя макета

01 имя части 1

02 заполнитель

....

01 имя части 2

02 заполнитель

....

где заполнитель — это арифметическое выражение, фрагмент запроса или системная переменная.

Каждая строка уровня 02 (заполнитель) описывает заполнение очередного окна (совокупности окон) части макета. Окна заполняются слева направо и сверху вниз.

В простейшем случае в качестве заполнителя могут использоваться: рабочее поле, идентификатор БД, константа, системная переменная.

В описании заполнения используются следующие системные переменные:

E##NPAGE — номер текущей страницы документа;

E##DATE — текущая дата в виде ДД.ММ.ГГ;

E##NPD — номер текущей периодической части (с именем, начинающимся на букву P).

Пр и м е р.

00 OUTFORM M1

01 ZD

02 'E##DATE'

01 PD

02 ФИО

02 ДОЛЖНОСТЬ

02 &ЗАРПЛАТА

01 ZS

02 'E##NPAGE'

В этом примере макет M1 заполняется следующим образом: в единственное окно части ZD заносится текущая дата, в три окна

части PD заносятся данные из вершин ФИО и ДОЛЖНОСТЬ базы данных и рабочего поля ЗАРПЛАТА, в окно части ZS заносится номер текущей страницы.

В общем случае в качестве заполнителя могут использоваться арифметические выражения и сложные фрагменты запроса. На пример:

```
00 WSECT
01 S
02 B[10]
02 C[5]
02 D[E]
01 100A[H]
00 OUTFORM M137
01 Z1
02 &S
01 PD
02 DO &I=1 BY 10 TO 91; &A[&I]
01 I1
02 IF &A[1]>&A[2] THEN &B ELSE &C;
02 (&A[1]-&A[2])*&D
```

В этом примере три окна части Z1 заполняются элементами составного рабочего поля S. Заполнение части PD связано с выполнением цикла: в 10 окон этой части заносятся элементы массива рабочих полей A[1], A[11], ..., A[91]. Заполнение первого окна части I1 зависит от значений рабочих полей A[1] и A[2], значение второго окна вычисляется с помощью арифметического выражения.

Фрагменты запроса могут быть и более сложными, включающими движение по базе данных. Как и в экспресс-выводе, в окна макета заносятся значения из простых вершин БД, являющихся терминальными вершинами фрагмента запроса (см. п. 5.2).

В скобочной записи формат раздела OUTFORM имеет вид:

```
**OUTFORMS: (имя макета=(описание заполнения частей)
[,имя макета=(описание заполнения частей)]...)
```

где описание заполнения частей имеет формат:

```
имя части=(заполнитель[,заполнитель]...),имя части=
=(заполнитель[,заполнитель]. . .),...
```

5.4.4. Управление формированием и выводом документа. Управление формированием и выводом выходного документа в запросе осуществляется с помощью функции PRINT. Обращение к ней имеет

вид:

%%PRINT([имя макета имя части[,заполнитель]. . .)

Здесь имя макета (до 8 символов) должно быть указано при первом обращении к данному макету. В задании на исполнение запроса это имя присваивается DD-оператору, содержащему описание набора данных с текстом макета. При последующих обращениях имя макета можно не указывать.

Имя части — это двухсимвольное имя части макета документа.

Понятие заполнителя описано в п. 5.4.3. Список заполнителей, указанный в обращении к функции PRINT, позволяет опустить соответствующее описание заполнения части в разделе OUTFORM. Следует отметить, что описание заполнения частей ZS и KS, содержащих окна, обязательно помещается в разделе OUTFORM, так как вывод этих частей производится автоматически (см. п. 5.4.7).

Как упоминалось выше, каждый заполнитель может быть сложным фрагментом запроса. Выполнение функции PRINT предполагает, что простые заполнители — рабочие поля заранее подготовлены в запросе, а сложные будут выполняться от текущей точки при обращении. После выполнения каждого фрагмента-заполнителя текущая точка восстанавливается.

Начальная настройка системных переменных (E##NPAGE, E##DATE, E##NPD) производится при обращении на вывод части ZD (заголовков документа). Если эта часть представлена в макете, одновременно производится вывод заголовка документа. Выводом части с именем KD завершается вывод документа (подводится конец страницы).

При выводе части документа размеры заданных переменных могут превосходить размеры соответствующих окон. Для однострочных частей предусмотрен режим переноса текстовых переменных путем повторной печати части. При этом продолжения текстовых переменных заносятся в те же окна. Печать части повторяется до полного вывода всех текстовых переменных.

Если числовое данное (целая часть данного) не помещается в отведенное ему окно, то перенос не производится, в окно заносится знак «?». Знаки после запятой округляются до позиций, заданных в окне макета.

Пример.

Пусть в окна макета части:

&&P1 ЧАСТЬ

| F(A14)

| F(A37)

| ?99.9 |

заносятся данные: в 1-е окно — Смирнов Олег Игоревич;
 во 2-е окно — Ордена: Красной Звезды; «Знак Почета»; Медали:
 «За оборону Москвы»; «За победу над Германией»;
 в 3-е окно — число 258.78.
 Часть P1 распечатается в виде: *

СМИРНОВ ОЛЕГ	ОРДЕНА: КРАСНОЙ ЗВЕЗДЫ;	258.8
ИГОРЕВИЧ	ЗНАК ПОЧЕТА; МЕДАЛИ:	
	ЗА ОБОРОНУ МОСКВЫ;	
	ЗА ПОБЕДУ НАД	
	ГЕРМАНИЕЙ;	

При занесении в те же окна данных: в 1-е окно — Смирнов-
 Сокольский Иннокентий Константинович;
 во 2-е окно — Медаль ВДНХ;
 в 3-е окно — 1017.2, часть P1 распечатается в виде:

СМИРНОВ-	МЕДАЛЬ ВДНХ	?
СОКОЛЬСКИЙ		
ИННОКЕНТИЙ		
КОНСТАНТИНОВИЧ		

Заметим, что если в макет части P1 примера добавить лишнюю строку (подчеркивание), то переносы текста производиться не будут, в окнах напечатаются начальные части соответствующих переменных. Для вывода полных текстов с подчеркиванием периодов в макете вводится дополнительная часть, например:

```
&&P1
| F(A14)          | F(A37)          | 999.9|
&&A1
F (63' _')
```

Использование запроса и функции PRINT для формирования и вывода документа предполагает движение по БД с последователь-

ным выводом частей документа. Тем самым предполагается соответствие структуры БД структуре выходного документа. Если такого соответствия нет, используется прием линеаризации данных (см. п. 5.4.6).

5.4.5. Пример вывода документа. В качестве примера рассмотрим ведомость штатного состава работников завода:

10.12.86

СТР. 1

ВЕДОМОСТЬ ШТАТНОГО СОСТАВА ЗАВОДА

№№ П/П	ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО	ДОЛЖНОСТЬ	ЗАРАБОТНАЯ ПЛАТА
1	2	3	4
ЦЕХ ПРОКАТНЫЙ			
1	ИВАНОВ И. И.	СЛЕСАРЬ	180.00
2	ПЕТРОВ П. П.	МАСТЕР	150.00
...			
ЦЕХ ЛИТЕЙНЫЙ			
19	СИДОРОВ С. С.	ИНЖЕНЕР	120.00
...			

СТР. 2

ВЕДОМОСТЬ ШТАТНОГО СОСТАВА ЗАВОДА

1	2	3	4
52	СКВОРЦОВ М. К.	ЛИТЕЙЩИК	280.00
2358	ЯКОВЛЕВ А. Н.	СТ. ИНЖЕНЕР	250.00
ВСЕГО: 2358 ЧЕЛОВЕК			

В этом примере таблица построена из регулярно повторяющихся частей. В макете эти части описываются следующим образом:

&&WED1 ФОРМА

&&ZD ЧАСТЬ

????????

СТР. 99

ВЕДОМОСТЬ ШТАТНОГО СОСТАВА ЗАВОДА

№№ П/П	ФАМИЛИЯ, ИМЯ, ОТЧЕСТВО	ДОЛЖНОСТЬ	ЗАРАБОТНАЯ ПЛАТА
1	2	3	4
&&Z1 ЧАСТЬ			
	ЦЕХ	????????????????????????????????	
&&PD ЧАСТЬ			
????	????????????????	????????????	????.99
&&ZS ЧАСТЬ			

СТР.99

ВЕДОМОСТЬ ШТАТНОГО СОСТАВА ЗАВОДА

1	2	3	4
&&I1 ЧАСТЬ			
&&KS ЧАСТЬ			
&&KD ЧАСТЬ			
	ВСЕГО: 9999 ЧЕЛОВЕК		

Здесь часть с именем I1 выводится после окончания данных по текущему цеху. В этом случае печатается подчеркивающая черта. Затем печатаются сведения о следующем цехе. Часть KS печатается автоматически в конце страницы.

Для вывода документа на основании сведений базы данных рис. 1.18 можно составить следующий текст запроса:

```
00 OUTFORM WED1
01 ZS
02 'E###NPAGE'
00 TEXT
01 %%PRINT('WED1.ZD','E###DATE','E###NPAGE')
01 ЗАВОД.ALL.
02 %%PRINT('Z1',НАИМЕНОВАНИЕ)
02 СОТРУДНИКИ.ALL.
```

Части	Имена частей	Комментарии		
Сведения по СССР	ZD			
РСФСР	Z1			
Москва	Z2			
Завод 1	PD	элементарные части	раздел 1.1 второго уровня	
Завод 2	PD			
.....				
Завод n	PD			
Итого по Москве	I2			
Ленинград	Z2			
Завод 1	PD		раздел 1.2 второго уровня	
.....				
Завод m	PD			
Итого по Ленинграду	I2			
Итого по РСФСР	I1			
УССР	Z1			
.....				
Итого по УССР	I1			
.....				
Итого по СССР	KD			

Рис. 5.21

```

03 %%PRINT('PD','E###NPD',ФИО,ДОЛЖНОСТЬ,
ОКЛАД)
02 %%PRINT('II')
01 %%PRINT('KD','E###NPD')

```

5.4.6. Алгоритм разбиения документа на страницы. Как упоминалось выше, документ имеет логическую структуру, элементами которой являются части документа. Весь документ можно предста-

вить как текст, обрамленный частями ZD (заголовок документа) и KD (конец документа — см. рис. 5.21). Каждый раздел документа может обрамляться частями Zi и Ii — промежуточными заголовками и итогами. Внутри раздела могут быть периодически повторяющиеся части.

При выводе документа встает задача разрезать его по горизонталям на полосы, не превышающие размер листа устройства вывода (АЦПУ, дисплей). Разрезка документа осуществляется при помощи частей ZS — заголовок страницы и KS — конец страницы, которые автоматически вставляются в документ и разбивают его на страницы. При этом первая страница начинается частью ZD (заголовок документа) и заканчивается частью KS, все страницы со второй до предпоследней начинаются частями ZS и заканчиваются KS, а последняя страница начинается частью ZS и заканчивается частью KD (конец документа — см. рис. 5.22).

Размер листа определяется четырьмя системными переменными E###LZ, E###LP, E###LI, E###LPAGE. Эти переменные содержат целые числа, определяющие соответственно, до какой строки могут появляться на листе заголовочные части (первый символ име-

ZD
Zi
.....
PD
KS

1-я страница

ZS
PD
.....
KS

2-я страница

.....
.....

ZS
PD
.....
II
KD

Последняя страница

Рис. 5.22

ни — Z), периодические элементарные части (первый символ имени — P), итоговые и все остальные части (см. рис. 5.23).

Задание размера листа не одним, а четырьмя размерами позволяет, как правило, обеспечить печать на одном листе заголовков вместе с первой периодической частью и итогов вместе с последней

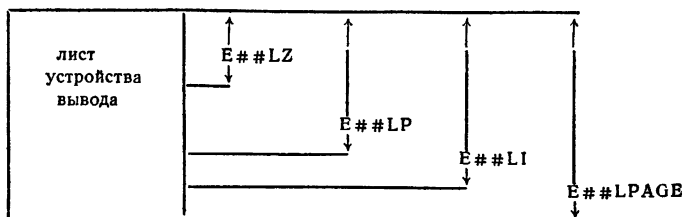


Рис. 5.23

периодической частью, т. е. не отрывать заголовок от первого периода и итог от последнего. Действительно, для этого достаточно размеры листа подобрать так, чтобы в строки между $E##LZ$ и $E##LP$ поместилась любая периодическая часть, в строки между $E##LP$ и $E##LI$ поместился любой итог, в строки от $E##LI$ до $E##LPAGE$ — часть KS.

При обращении к системе макетного вывода — оператор `%%PRINT(имя части, ...)` — производится проверка, помещается ли указанная часть на свободной части текущего листа. Проверка производится по одному из трех размеров $E##LZ$, $E##LP$, $E##LI$ в соответствии с именем выводимой части. Если часть помещается, то она выводится на текущем листе. Если не помещается, то на текущем листе выводится автоматически часть KS, затем дается подвод бумаги к линии изгиба (управляющий байт печати — I) и выводится часть ZS, и только после этого выводится уже на новой странице часть, указанная в операторе PRINT.

Особый случай возникает при выводе части, которая имеет в макете спецификацию СН (страница в начале) или СК (страница в конце). В первом случае часть выводится на новой странице без оформления конца предыдущей (KS) и начала новой (ZS). При спецификации СК после вывода части производится подвод к началу нового листа также без вывода частей KS и ZS. Использование этих спецификаций удобно в тех случаях, когда промежуточные заголовки и итоги содержат в себе оформление страниц, отличное от KS и ZS.

В макете документа может быть указана часть с именем FP (заполнитель страницы). В этом случае перед выводом части KS автоматически выводится часть FP до тех пор, пока лист не заполнится до линии $E##LI$. Если указать однострочную часть FP,

то это позволяет получать листы строго определенного размера. Например, можно печатать листы по форме единой системы подготовки конструкторской документации (см. рис. 5.24).

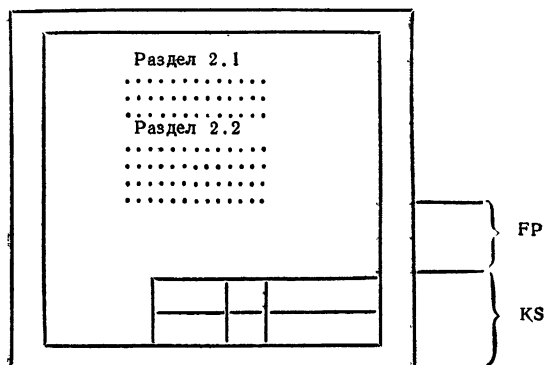


Рис. 5.24

5.4.7. Настройка системы вывода. Перед обращением к системе вывода (`%%PRINT`) можно настроить ее при помощи функции

`%A4LLBO80('PUT', , 'исп1', п1, 'исп2', п2, . . .),`

где `исп1` — имена системных переменных, а `п1` — константы или переменные системы запросов. Функция заносит в системные переменные `исп1` значения `п1`. Тип `п1` должен совпадать с типом `исп1`, так как преобразования формата функция не производит.

При помощи настройки можно управлять форматом страницы вывода (алгоритм разбивки документа на страницы описан в п. 5.4.6.) и параметрами редактирующих программ.

Системные переменные, определяющие формат страницы: `E##LZ`, `E##LP`, `E##LI`, `E##PAGE`. Значения их по умолчанию — 58, 60, 62, 62. Чтобы изменить их, достаточно обратиться к функции `A4LLBO80`.

П р и м е р. Нужно выводить документы на страницы со следующими размерами: `E##LZ=55`, `E##LP=62`, значения для `E##LI` и `E##PAGE` лежат соответственно в переменных `LI` и `P`, которые имеют в `WSECT` тип `F`. Для этого достаточно выполнить перед печатью следующую настройку:

`%A4LLBO80('PUT', , 'E##LZ', 55, 'E##LP', 62,
'E##LI', &LI, 'E##PAGE', &P)`

Настройка позволяет также управлять редактированием выводимых данных. Системные переменные, управляющие редактированием:

E##ZERO — определяет, как выводить числа, равные нулю, тип переменной — один символ. Если этот символ равен нулю (X'00'), что соответствует умолчанию, то нули выводятся в виде 0.0...0 (число знаков после десятичной точки определяется форматом окна, в которое заносится нуль). Если символ **E##ZERO** не нуль, то в окно заносится вместо нуля этот символ.

E##FILER — символ для заполнения старших позиций окна при занесении чисел (по умолчанию — пробел).

EPOINT — символ, который выводится в качестве десятичной точки чисел (по умолчанию — точка).

5.4.8. Вывод документов, иерархия которых не соответствует БД. Если иерархия выходного документа не соответствует иерархии БД, то при помощи запроса, программ **A2RQPUTS** и **OUTFM** можно провести линейризацию БД, отсортировать полученный последовательный набор и затем выдать нужный документ.

Программа **A2RQPUTS** служит для переписи данных средствами языка запросов из базы данных в последовательный набор данных (ПНД) и для чтения записей из ПНД в поля **WSECT** запросной системы. Заметим, что ПНД с записями фиксированного формата, сформированный программой **A2RQPUTS**, может использоваться как набор исходных данных для стандартных и пользовательских прикладных программ.

Открытие последовательного набора данных:

%A2RQPUTS(режим,размер,DD-имя)

Первый операнд определяет режим открытия: **OPEN** — открыть ПНД на вывод полей из **WSECT**; **OPEN(I)** — открыть ПНД на чтение записей в поля **WSECT**.

Второй операнд задает размер записи в создаваемом ПНД (для режима **OPEN**).

DD-имя здесь и ниже — имя **DD-оператора**, описывающего ПНД.

Вывод поля **WSECT** в очередную запись ПНД осуществляется по обращению:

%A2RQPUTS('PUT',поле, ,DD-имя)

Ввод очередной записи ПНД в поле **WSECT** осуществляется по обращению:

%A2RQPUTS('GET',поле, ,DD-имя,флаг)

Флаг служит для сообщения о конце ПНД: флаг=1, если массив ПНД окончен, флаг=0 в противном случае.

Закрытие ПНД производится по обращению:

%A2RQPUTS('CLOSE',DD-имя)

Программа **OUTFM** служит для сортировки последовательных наборов. Обращение к программе:

%OUTFM('икп1=з1[,икп2=з2]. . .', '0')

где $икп_i$ — имя i -го ключевого параметра, z_i — значение i -го параметра.

Имеются следующие параметры:

$F=DUMMY$ — обязательный параметр. (Если указано $F=$ имя макета, то, кроме сортировки, будет осуществлена и печать документа по указанному макету, программа заполнения в этом случае должна быть описана при помощи макроопераций $LIST$ и $LCODE$, см. [39].)

$D=$ имя — задает имя DD-оператора, описывающего сортируемый набор данных, по умолчанию $D=FPS$;

$O=$ имя — определяет имя DD-оператора, описывающего набор данных, в который выводится отсортированный массив. Если указана только буква O среди параметров, то отсортированный массив будет записан на место исходного;

$S=(с, д, ф, п[с, д, ф, п]. . .)$ — определяет поля, по которым производится сортировка, каждое поле описывается четырьмя подпараметрами:

$с$ — смещение в байтах поля в записи,

$д$ — длина поля,

$ф$ — формат представления данного в поле (CH — символьный латинский, RT — символьный русский, FI — числа с фиксированной запятой, для полей $WSECT F$ и H , FL — числа с плавающей запятой для полей $WSECT E$ и D),

$п$ — порядок сортировки: A — по возрастанию, D — по убыванию;

$SIZE=$ число — необязательный параметр, определяющий точное число записей в сортируемом наборе;

$SIZE=E$ число — необязательный параметр, определяющий приблизительное число записей в наборе;

$E15=$ имя — необязательный параметр, определяющий имя блока пользователя, которому передается управление после прочтения записи из входного набора. В первом регистре при обращении к блоку содержится адрес введенной записи. Ответ блока должен содержаться в 15 регистре: 0 — продолжить обработку, 4 — исключить запись, 8 — прекратить дальнейшую обработку массива;

$R=$ число — определяет в килобайтах размер оперативной памяти, доступной программе сортировки, по умолчанию $R=60$.

Для работы программы необходимы следующие наборы данных:

— сортируемый набор данных, определяется параметром $D=$ имя;

— $SORTLIB$ — библиотека, содержащая программу сортировки, например:

```
//SORTLIB DD DSN=SYS1.SORTLIB,DISP=SHR
```

— SYSOUT — набор данных для выходных сообщений, например:

```
//SYSOUT DD SYSOUT=A
```

или

```
//SYSOUT DD DUMMY
```

— SORTWK01, SORTWK02, SORTWK03 — рабочие наборы данных для сортировки, например:

```
//SORTWK01 DD DSN=&W1,UNIT=SYSDA,  
// SPACE=(CYL,(2), ,CONTIG)
```

5.4.9. Процедуры вывода. Процедура ISMAKET предназначена для контроля макета документа. Параметр процедуры P задает вид распечатки макета и принимает значения:

P=1 — построчная и общая распечатка макета;

P=2 — общая распечатка макета без деления его на части,

Общая распечатка макета дает возможность получить общий вид документа с выделенными в нем окнами. DD-предложение с именем SYSIN задает последовательный набор данных с текстом описания макета. Если в описании макета неверно заданы управляющие карты, то при его распечатке выдаются соответствующие диагностические сообщения.

Трансляция, редактирование и исполнение запроса, использующего макетный вывод, могут осуществляться с помощью стандартных процедур запросной системы (см. п. 4.9). Если запрос должен сформировать выходной документ по заданному макету, то в шаге GO (исполнение запроса) должно быть указано DD-предложение, имя которого совпадает с именем макета в разделе OUTFORMS и в обращении к PRINT. В DD-предложении описывается набор данных или раздел библиотеки исходных текстов, содержащий описание макета. Например:

```
// EXEC ISREQCLG,DOD=DODBAS,DD=DDBAS,  
// V=WORK1  
//GO.WED1 DD DSN=SOURCE(TWED),DISP=SHR
```

Здесь описание макета содержится в разделе TWED библиотеки SOURCE. Заметим, что текст описания должен храниться в распакованном виде.

Иногда макет документа помещается во входном потоке задания после карты:

```
//имя-макета DD*
```

или

```
//GO.имя-макета DD*
```

5.5. Управление потоками сообщений

5.5.1. Назначение и возможности. В данном параграфе рассматривается средство, при помощи которого любая система, разработанная для эксплуатации в пакетном режиме, может быть запущена в диалоговом режиме [16, 17]. Для организации диалогового режима работы в ИНЕС имеется большой комплекс средств (см. рис. 4). Достоинство разбираемого в этом параграфе средства состоит в том, что для организации работы с дисплеем не требуется никаких новых знаний, достаточно переключить поток сообщений на дисплей. Другое достоинство этого средства заключается в том, что все сообщения по мере готовности сразу появляются на экране дисплея без накопления их в промежуточном наборе данных, это во многих случаях заметно повышает реактивность систем.

Выводимые программные сообщения, независимо от их сложности, формируются в конечном счете как последовательность строк. Нижним уровнем программного обеспечения вывода сообщений являются средства вывода подготовленной к печати (отредактированной) строки. Управляя этими элементарными средствами вывода, мы тем самым управляем выводом сообщений в целом. Для пользователя интерес представляют не отдельные строки, а составленные из них документы, т. е. последовательности строк. Так как программа может формировать не одну, а несколько последовательностей строк, выводя строку, нужно указывать, к какой последовательности строк она относится. Для каждой последовательности строк нужно указать, куда ее следует выводить (на печать, на диск, на экран дисплея и т. д.). Всюду ниже вместо слов «последовательность строк» мы будем использовать термин «поток сообщений».

Желательно иметь систему вывода сообщений, которая позволяет программисту легко и унифицированно использовать разнообразные устройства, включая дисплеи, и при этом может взять на себя технически громоздкие детали программирования.

Программные средства вывода СУБД ИНЕС рассчитаны на вывод сообщений различной сложности. На верхнем уровне иерархии находится «система макетного вывода документа сложной структуры», далее следуют «система заполнения и вывода части документа» (близкая по возможностям к фортранному оператору PRINT) и «система-редактор элементарных и групповых полей». На нижнем уровне иерархии находится средство ULINE, обеспечивающее вывод предварительно сформированной (отредактированной) строки. Именно эта система ULINE отвечает за инвариантность программы по отношению к используемым устройствам.

В простейшем случае ULINE выводит сообщения в набор данных, описанный DD-оператором с именем PRINT1, причем от поль-

зователя не требуется ни операций открытия и закрытия, ни создания каких-либо управляющих таблиц.

Организация потока производится операцией открытия. Если пользователь не произвел открытия потока, то оно будет произведено автоматически при выводе первого сообщения в этот поток. Наиболее употребительными в системе ИНЕС являются поток PRINT, используемый по умолчанию, и поток DBSH. В основном используется поток PRINT, поток DBSH предназначается для диагностических сообщений.

ULINE позволяет организовать выдачу сообщений также и при наличии нескольких подзадач. Для каждой подзадачи создаются свои потоки сообщений, даже если имена потоков совпадают. Например, если в программе пользователя нет операций открытия OPEN, то сообщения, выдаваемые в потоки PRINT из разных подзадач, будут выводиться в наборы данных, описанные операторами DD с именами PRINT1, PRINT2, PRINT3 и т. д.

Имеются следующие режимы работы системы ULINE:

- открытие потока (OPEN);
- закрытие потока (CLOSE);
- получение сообщения от оператора дисплея (GET);
- вывод вопроса на дисплей и получение ответа на него (PUTGET).

Последние два режима используются, если операцией открытия была произведена настройка потока на вывод сообщений на дисплей.

Обращение к системе ULINE возможно из языков ИНЕС (ЯЗ, ЯСД), а также из ассемблера, фортрана, кобола, ПЛ/1, оптимизирующего ПЛ/1 с помощью модуля A2ULAN.

5.5.2. Организация потоков сообщений. Поток сообщений организуется операцией открытия, которая производится либо по явному указанию пользователя, либо автоматически при выводе первой строки сообщения.

При открытии указывается, каким образом нужно обрабатывать выводимые в поток сообщения. Возможны следующие режимы:

- вывод в набор данных, заданный именем DD оператора или адресом таблицы DCB;
- вывод на терминал (на дисплей или на устройство печати);
- вывод на несколько других потоков (через A2ULAN для копирования сообщений);
- подавление вывода.

При работе с дисплеем можно не только выводить, но и принимать сообщения от оператора дисплея. Запрос на открытие потока не обязан исходить из программы, которая будет осуществлять

вывод. Нужно лишь, чтобы он был сделан в рамках той же подзадачи, перед обращением на программу вывода.

Автоматическое открытие производится при выводе первой строки в неоткрытый ранее поток вывода. Если имя потока PRINT (это имя используется по умолчанию), то производится настройка потока на вывод в набор данных, описанный оператором DD с именем PRINT1 (PRINT2, PRINT3 и т. д. при многозадачной работе). Если имя потока отлично от PRINT, то имя оператора DD формируется из имени потока приписыванием к нему справа цифры 1 (цифры 2, 3 и т. д. в случае многозадачной работы). Если такое DD есть в шаге задания, то поток настраивается на вывод в набор данных, описанный этим DD. В противном случае поток объявляется эквивалентным потоку PRINT.

Если при открытии объявить поток «фиктивным», то сообщения, выводимые в этот поток, будут теряться. Можно открыть поток через A2ULAN, указав до пяти имен потоков, в которые будут фактически направляться сообщения, направляемые в этот поток. Этот режим предназначен для копирования сообщений на устройства разных типов, например, на терминал и на печать. Если первый из потоков для копировки открыт на дисплей, то с дисплея можно управлять передачей сообщений в остальные потоки. При закрытии потока производятся следующие действия:

- закрывается набор данных, если поток был настроен на работу с набором данных;

- выводится на экран содержимое буфера и строка «*****». Затем высвечивается сообщение «???».

5.5.3. Использование терминалов. Работа с терминалами производится через систему терминального ввода-вывода (CTBV) ИНЕС ([42]). Имеется возможность не только выводить сообщения на экран дисплея, но и получать сообщения от оператора дисплея. Выводимые строки высвечиваются на экране дисплея сверху вниз, начиная со второй строки экрана. При исчерпывании экрана в его правом верхнем углу высвечивается сообщение «???». После ответа оператора экран очищается (находившиеся на нем сообщения теряются), и на экран выводятся следующие строки, каждое сообщение занимает одну или две строки на экране дисплея (если вторая строка состоит из пробелов, то она не выводится).

Возможны два режима вывода сообщений на дисплей: построчный и поэкранный. В первом случае строка сразу же выводится на экран дисплея. Во втором случае строки сначала попадают в буфер, находящийся в оперативной памяти, размер которого равен размеру экрана. Вывод информации на экран дисплея производится лишь после того, как буфер заполнится. В любом случае при необходимости что-либо вывести на экран дисплея программа ULINE только запускает вывод и возвращает управление программе пользователя.

Ожидание окончания вывода производится при выполнении следующей макрокоманды ULINE. Благодаря этому вывод на экран дисплея идет одновременно с другими действиями программы пользователя (с вычислениями, с обменов с диском и т. п.), что часто позволяет обеспечить работу терминала с технической скоростью.

В тех случаях, когда на терминал выводится документ, содержащий различного рода заголовки, часто бывает нужно, чтобы заголовки, относящиеся к выведенным на экран частям документа, были всегда видны на экране. Для этой цели пользователь должен указать номер уровня каждой части документа; номер уровня указывается первым символом (символ управления протяжкой бумаги не считается) первой строки части документа. В остальных случаях этот байт должен содержать пробел. Самый старший заголовок имеет нулевой номер уровня, непосредственно входящие в него подзаголовки, — первый уровень и т. д. Части документа, не являющиеся заголовками, должны иметь самый большой номер уровня. Вместе с каждой частью документа на экране будут присутствовать части всех уровней, меньших, чем ее уровень, причем из частей, имеющих одинаковый уровень, будет выбрана та, которая является ближайшей предшествующей к рассматриваемой части.

Оператор дисплея может послать сообщение в ЭВМ либо в ответ на заданный ему вопрос, либо по своей инициативе. Для задания вопроса оператору и получения на него ответа служит режим PUTGET. Передать сообщение по своей инициативе оператор может, когда в правом верхнем углу экрана находится надпись «???». Вызвать появление этой надписи можно простым нажатием клавиши «ввод». В процессе выполнения ULINE в режиме PUT или GET это сообщение будет принято программой ULINE и запомнено в буфере. Программа пользователя сможет получить это сообщение в режиме GET.

В ответ на сообщение «???» оператор дисплея может либо отказаться от ответа, нажав клавишу «ввод», либо передать сообщение одним из следующих четырех способов:

1. Набрать сообщение на верхней строке и нажать клавишу «ввод».

2. Нажать функциональную клавишу. Сообщение будет состоять из пяти символов, первые три символа — «*PF», следующие два символа — код клавиши в STBV.

3. Переместить курсор на любую строку, кроме первой, и нажать клавишу «ввод». Если курсор находится внутри поля, ограниченного символами <и>, то сообщением будет все это поле. Если курсор находится вне такого поля, то сообщением будет содержимое ближайшего справа такого поля. Если и правее курсора такого поля нет, то сообщением будет считаться находящийся в строке текст, расположенный между положением курсора (включая и ту

позицию, на которую он указывает) и концом строки. Этой возможностью можно пользоваться при работе с терминалами ЕС7066, ЕС7920 и МЕРА7910, при работе в позкранном режиме и в построчном режиме с удержанием заголовков.

4. В выводимом на экран тексте можно предусмотреть поля, предназначенные для внесения информации оператором дисплея («окна»). Начало и конец окна указываются символами [и]. При работе с дисплеем ЕС7066 эти символы будут видны на экране. При работе с другими дисплеями эти символы заменяются на символы управления форматом экрана. Если на экране в момент ввода будут высвечены такие окна, то их содержимое (измененное или не измененное оператором) будет передано программе вслед за сообщением оператора, переданным одним из предыдущих способов. От этого сообщения и друг от друга поля будут отделены символом Х'05' («конец поля»).

Таким образом, первый и третий способы ответа различаются по положению курсора на экране в момент ввода. Третий способ является надежным с точки зрения защиты от ошибок оператора при наборе сообщения. Он позволяет при хорошем уровне наглядности сэкономить время ответа оператора, так как текст выводится программно. В каждом конкретном случае ввода оператор сам выбирает между первым и третьим способами ответа. С каждым из них может сочетаться четвертый способ, если его возможность была затребована при открытии и его допускает текст, находящийся в этот момент на экране.

Программа ULINE полученные от оператора дисплея сообщения никак не обрабатывает, а лишь передает их программам пользователя. Исключениями являются команда «*OFF» и команда «*C». По команде «*OFF» программа ULINE перестает выводить сообщения на экран дисплея (они будут просто теряться) до тех пор, пока в программе пользователя не выполнится команда ULINE в режиме GET или PUTGET. В этом случае программа пользователя получит сообщение «*OFF» и сможет продолжать вывод сообщений. Кроме того, можно отменить команду «*OFF» простым нажатием на «ввод». Если команда «*OFF» дана в режиме с удержанием заголовков, то в буфере ULINE поддерживается правильная картинка; после отмены команды экран будет сформирован с сохранением всех необходимых заголовков. После команды «*C» по исчерпанию экрана не будет высвечиваться сообщение «???» и не будет ожидания нажатия клавиши «ввод» — сообщения будут выводиться непрерывно. Для отмены этой команды достаточно нажать клавишу «ввод».

Работа с устройством печати, подключенным к терминальной станции, аналогична выводу на АЦПУ. Если нужна возможность получать сообщения от оператора, то необходимо при открытии потока указать дисплей, с которого будут приниматься сообщения.

Операция ULINE в режиме GET позволяет проверить, было ли передано оператором дисплея сообщение в ЭВМ или нет. Если было получено сообщение, то код возврата равен нулю. Если оператор заполнил находящиеся на экране окна, то первый символ сообщения будет X'05'. Содержимое одного окна отделяется от содержимого следующего также символом X'05'.

При выполнении операции ULINE в режиме PUTGET в правом верхнем углу экрана будет высвечен вопрос оператору. До тех пор, пока оператор не ответит, программа будет находиться в состоянии ожидания. Ответ передается, как и при операции GET.

5.5.4. Обращение к системе ULINE. Модуль A2ULAN предназначен для обращения к ULINE из программ, написанных на языках ИНЕС (в частности, на языке запросов), на фортране, коболе, ПЛ/1, оптимизирующем ПЛ/1, ассемблере. Обращение из ПЛ/1 отличается от других отсутствием описателей длины, строчных переменных и массивов строчных переменных. Обращение из других языков, кроме языка запросов, производится оператором CALL вместо знака %. Формат обращения:

%A2ULAN(имя операции, IERR, параметры операции)

Имя операции может принимать следующие значения:

OPEN(TV) — открыть поток на терминал;

OPEN(COPY) — открытие для копировки сообщений в несколько потоков;

OPEN(DUM) — режим DUMMY — сообщения будут подаваться;

CLOSE — закрытие с уничтожением управляющей таблицы;

GET — прием сообщения с терминала;

PUTGET — вывод вопроса и прием ответа;

IERR — имя переменной, в которую будет загружен код возврата после операции.

Во всех операциях параметр NAME задает имя потока, если он не задан, то используется NAME='PRINT'.

Параметры операции задаются в зависимости от операции. Можно опустить несколько последних или все параметры операции, значения опущенных параметров принимаются по умолчанию.

Параметры DDNAME, NAME, TYPE задаются символьной строкой в апострофах или символьной переменной длины не более восьми символов. Для фортрана строка короче восьми символов должна кончаться пробелами (пробелы внутри строки не допускаются).

Открытие потока на терминал. Формат обращения:

%A2ULAN('OPEN(TV)', IERR, TYPE, NAME)

TYPE — задает режим работы с терминалом — последовательность управляющих символов в любом порядке. Допустимо использование следующих символов:

S — поэкранный режим;
L — построчный режим;
C — удерживать заголовки на экране;
W — работать с защищенными окнами, отмеченными квадратными скобками.

Настройка (открытие) на копировку в несколько потоков.
Формат обращения:

%A2ULAN('OPEN(COPY)', IERR, NAME, N1, N2, N3, N4, N5)

N1, N2... — имена потоков, в которые будут направляться сообщения, направленные в поток NAME, задаются так же, как NAME.

Поток N1 считается главным — ему переадресуются GET и PUTGET потока NAME. Если оператор передал (по GET или PUTGET потока NAME) команду «*NOCOPY» или «*НЕ КОПИ» или «*НЕКОПИ», то сообщения в потоки N2, N3, ... будут теряться до отмены этой команды командой «*COPY» или «*КОПИ».

Подавление сообщений. Формат обращения:

%A2ULAN('OPEN(DUM)', IERR, NAME)

Сообщения, направленные в поток, будут теряться.

Закрытие потока сообщений. Формат обращения:

%A2ULAN('CLOSE', IERR, NAME)

Прием сообщений от оператора дисплея. Формат обращения:

%A2ULAN('GET', IERR, STR, LSTR, NAME)

STR — символьная переменная, для ПЛ/1 обязательно переменной длины; LSTR — описатель длины STR.

Задание вопроса оператору и получение ответа. Формат обращения:

%A2ULAN('PTGET', IERR, ST1, LST1, ST2, LST2, NAME)

Примеры и пояснения к ним на с. 240—241.

5.5.5. Разбор сообщений. Здесь рассматриваются средства разбора сообщения, переданного в виде списка, по соответствующим полям WSECT. Для этого имеются две программы REQSTRAN и RQBYNAME.

Программа REQSTRAN предназначена для чтения из символьной строки в стек данных — внутреннюю рабочую память интерпретатора запросной системы. Результатом ее работы является упорядоченный набор данных, некоторые из которых могут иметь имена. При анализе строки программа делит ее по символам-разделителям «конец имени» и «конец данного». Фрагмент строки, заканчивающийся разделителем первого типа, считается именем, второго — данным. Допускается нулевая длина как имени, так и данного.

Программа анализа строки не производит никакого контроля имен данных. Длина имени и данного не может превышать 250 байтов. Если среди последовательных 250 байтов строки не встретилось ни одного разделителя, то после них считается автоматически вставленным разделитель «конец данного».

Два списка символов-разделителей могут быть заданы при обращении. Это позволяет избежать трудностей ввода данных со специальными символами. В частности, программист может предусмотреть возможность задания этих символов самим оператором терминала, что может повысить наглядность и облегчить его работу. По умолчанию список разделителей «конец данного» — X'05', «,» и «;», «конец имени» — «=» и «:».

Обращение к программе может осуществляться только из программы на языке запросов. Формат обращения:

%REQSTRAN(буфер,длина,список1,список2)

Параметры обращения:

1. Рабочее поле произвольной структуры или параметр опущен. В случае, если параметр опущен, будет произведен разбор последнего элемента стека данных. Длина поля будет вычислена по описанию WSECT, но типы всех подчиненных полей проигнорированы: все поле вместе будет рассматриваться как одна длинная строка символов.

2. Числовая константа или рабочее поле, задающее длину текстовой строки, заданной первым операндом. Может быть задана явная константа, рабочее поле формата F или N, или структура или массив из двух полей формата N.

3. Текстовая константа или рабочее поле задает список разделителей «конец данного». Если параметр опущен, то считается, что задан список разделителей: X'05', «,», «;».

Задавая параметр, можно потребовать оставить действие символа из списка разделителей по умолчанию или ликвидировать его действие, задав в соответствующей позиции «+» или «—» соответственно.

4. Текстовая константа или рабочее поле задает список разделителей «конец имени». Если параметр опущен, то считается, что задан список разделителей «=» и «:».

Можно сохранить действие некоторых разделителей из списка по умолчанию аналогично третьему параметру.

Программа RQBYNAME производит занесение данных с именами из стека данных интерпретатора запросной системы в рабочие поля программы на языке запросов.

При обращении к программе ввода данных с именами может быть передано имя структуры рабочих полей запроса. В таком случае все поиски имен (и, естественно, присваивания) будут вы-

полняться только среди подчиненных этой структуре. Это позволяет защититься от доступа по имени к непараметризуемой переменной. Если такое имя опущено, будет просматриваться вся секция рабочих полей.

Программа анализирует стек данных от начала к концу, т. е. в порядке поступления в него данных (слева направо по исходной символической строке). В стеке при этом могут встречаться элементы разного типа — *данное с именем, данное без имени и имя без данного*. Поясним, как программа пытается трактовать элементы разного типа.

Элемент типа «данное без имени» всегда воспринимается как требование записать его в *следующее* элементарное рабочее поле, т. е. в следующий элемент в массиве терминальных полей или в следующее в порядке левого обхода по описанию терминальное рабочее поле в структуре. «Текущей точкой» в дереве рабочих полей становится это поле. Следующим терминальным за структурным именем считается первое терминальное из подчиненных ему.

При встрече элемента стека с именем программа пытается¹ сопоставить ему какое-либо имя в описании рабочих полей (алгоритм поиска описан ниже). Дальнейшие действия зависят от того, является ли найденное поле в описании структурным или терминальным. Если это структурное поле, то оно становится текущей точкой, а данное, если оно есть, записывается опять-таки в первое подчиненное терминальное рабочее поле. Точно так же происходит обработка терминального имени с данным. Если же имя терминальное, а данного нет, то программа трактует это как требование продублировать первое встретившееся данное с таким именем, т. е. занести в данную терминальную вершину первое данное, которое встретится за ним в стеке.

Пусть, например, в секции рабочих полей запроса описана структура «РАЗМЕРЫ», состоящая из элементарных полей «ВЫСОТА», «ДЛИНА» и «ШИРИНА». Тогда при записи результата разбора строки:

РАЗМЕРЫ:ДЛИНА=10,ВЫСОТА=ШИРИНА=5;

структурное имя «РАЗМЕРЫ» будет трактоваться как уточняющее последующие имена «ВЫСОТА», «ДЛИНА» и «ШИРИНА». Имя элементарного поля «ВЫСОТА» будет воспринято как требование записать в поле «ВЫСОТА» то же данное 5, что и в поле «ШИРИНА». Того же результата можно было бы достигнуть следующим образом:

РАЗМЕРЫ=5,10,5;

В этом примере имя «РАЗМЕРЫ» по-прежнему воспринимается как уточняющее, после чего происходит запись в три подчиненных

рабочих поля. Заметим, что теперь их значения надо указывать в порядке описания. Так же оформляется запись в массив терминальных полей.

Изменить только второй и третий элемент в структуре «РАЗМЕРЫ» можно таким образом:

РАЗМЕРЫ:ДЛИНА=10,5;

Данное 5 будет записано в следующее элементарное поле после поля «ДЛИНА», т. е. в поле «ШИРИНА». Здесь уже имя «ДЛИНА» выступает как уточняющее позицию в структуре «РАЗМЕРЫ».

Опишем теперь алгоритм поиска имен. Его знание существенно, во-первых, при наличии одноименных полей в структуре рабочих полей (для их идентификации необходимо использовать уточняющие имена) и, во-вторых, при наличии в описании массивов структурных полей (в частности, многомерных массивов). Алгоритм перехода к следующему элементу массива тесно связан с алгоритмом поиска имен.

При поступлении нового имени из стека данных ищется минимальная объемлющая структура текущей точки в дереве рабочих полей, и в этой структуре от текущей точки до ее конца в порядке левого обхода производится поиск этого имени. Если он закончился успешно, то текущая точка сдвигается на вновь найденную, и работа заканчивается. В противном случае происходит поиск имени от начала структуры до текущей точки. Если имя найдено, то помимо установки текущей точки в структуре рабочих полей запроса программа пытается выполнить «шаг вперед» в дереве массивов, подчиненных структуре и объемлющих как старую, так и новую текущие точки. Сначала делается попытка нарастить индекс в массиве нижнего уровня. При превышении размерности этот индекс отбрасывается в 1, и производится такая же попытка на следующем уровне иерархии массивов. Наглядно можно себе представить алгоритм перехода так: повторим описание структуры имен, подчиненной каждому массиву столько раз, какова его размерность. Тогда в получившемся дереве (соответствующем теперь уже взаимно однозначно рабочим полям программы) поиск имени всегда происходит просто в порядке левого обхода от текущей точки.

Если в минимальной объемлющей структуре поиск не увенчался успехом, то текущая точка переставляется в эту структуру и производятся те же действия. При этом в поддереве текущей точки поиск уже не производится. Если имя не было найдено во всей структуре рабочих полей, то фиксируется ошибка.

Пусть, например, в секции рабочих полей запроса описан массив А размерности 10, каждый элемент которого состоит из терминальных рабочих полей А, В и С. При вводе в этот массив следующих данных: А:С=1, А=2, В=2, В=3, В=4, С=4, С=5, В=6 значе-

ние 6 будет записано в элементарное поле В из 6-го элемента массива А. Заметьте, что второе упоминание имени А трактуется как имя элементарной переменной А, а не массива А.

Если желательно обеспечить возможность указывать имена нижнего уровня, относящиеся к одному элементу массива, в произвольном порядке, то можно ввести в описание рабочих полей дополнительный фиктивный уровень иерархии: объявить элемент массива А структурой Х, которая состоит из полей А, В и С. Тогда переход к новому элементу массива произойдет только при упоминании имени Х, а имена А, В и С можно указывать в произвольном порядке: $A:C=1, A=1, B=1$; $X:A=2, C=2$; $X:B=3, A=3$.

Заметим, что теперь повторное упоминание имени нижнего уровня приведет к записи поверх предыдущего значения в том же элементе массива А, например, при вводе из предыдущего примера все записи будут произведены в первый элемент массива А, А примет значение 2, В — 6, С — 5.

Программа может производить поиск имен на полное совпадение и на совпадение с началом имени переменной из секции рабочих полей.

Для обработки встретившихся ошибок (неудача при поиске имени, переполнение массива или всей обрабатываемой структуры) может быть указан блок обработки ошибок пользователя. Ему будет среди прочего передан участок стека данных от места ошибки до следующего данного с именем. Если блок обработки ошибок не был указан при обращении, произойдет распечатка этого участка стека стандартной программой ввода с именами.

Программа ориентирована на использование в комплексе с программой REQSTRAN для параметризации исполнения запросов с заданием параметров в ключевом формате. Параметризация может происходить через поле PARM, через входной набор данных и в коде исполнения запроса в интерактивном режиме — с приемом имен и значений параметров с терминала.

Обращение к программе происходит в два приема. Сначала нужно обратиться к программе инициализации ввода RQBYNAME, предоставив ей переменную связи — элементарную рабочую ячейку. Если при обращении не было ошибок (это можно проверить по коду условия после возврата из программы RQBYNAME), то в этой переменной будет сформирован адрес блока на языке запросов, внутри которого будет осуществляться собственно ввод, на который и нужно обратиться.

Программа RQBYNAME имеет три параметра: «куда», «переменная связи», «адрес блока обработки ошибок».

1. Рабочее поле из WSECT запроса, в подполя которого будет производиться запись. Если параметр опущен, то разрешена запись в любое поле из WSECT,

2. Переменная связи — рабочая ячейка. Начальное значение 0 соответствует требованию точного совпадения имен, 1 — разрешению сокращения имен до первых символов. Значение переменной не должно портиться программой пользователя до окончания ввода по именам (в частности, блоком обработки ошибок).

3. Адрес блока обработки ошибок пользователя. Может быть передан в любом рабочем поле формата F. Если параметр опущен, то при ошибке программой OUTST будет распечатан ошибочный участок стека.

Схема использования:

```
**WSECT:(SYSTEM(LINK,BUF[80],2LBUF[H],...),USER(...))
**BLOCKS: (ERROR=(печать сообщения об ошибке))
%A2ULAN('OPEN(TV)')
(&LBUF:=длина) (&AERR:=%A(□ERROR))
.....
IF
%A2ULAN('GET',&ERR,&BUF,&LBUF)
THEN
%REQSTRAN(&BUF,&LBUF)
IF %RQBYNAME(&USER,&LINK,&AERR)
THEN □&LINK;;
.....
```

Пример приведен на с. 241,244(строки 100—104).

5.5.6. Пример запроса с использованием системы ULINE. Пример построен на основе базы данных об архитектурных памятниках. На рис. 5.25 показан фрагмент базы данных, использованной в запросе. Основная часть базы с вершиной «памятники» содержит сведения об архитектурных памятниках, расположенных в разрезе по республикам, областям и функциональному назначению. Допустимые шифры перечисленных параметров хранятся вместе с наименованиями в дереве с вершиной «классификатор». Не используемые в запросе части базы на рисунке не показаны.

Предлагаемый пример запроса позволяет вывести на дисплей описание памятников, удовлетворяющих заданным параметрам. Пользователь, запрашивающий сведения о памятниках, должен ввести шифры интересующих его параметров. Если шифры ему неизвестны, то пользователь запрашивает список имеющихся шифров и программа высветит его на дисплей вместе с соответствующими шифрам наименованиями. Отметив нужную строку, пользователь передает выбранный шифр программе. По ходу выполнения запроса пользователь может включить или выключить печать на АЦПУ копии выводимого на дисплей текста, отменить или вновь включить показ каких-либо реквизитов описания, оставить данный запрос и пустить новый с другими параметрами или окончить задачу.

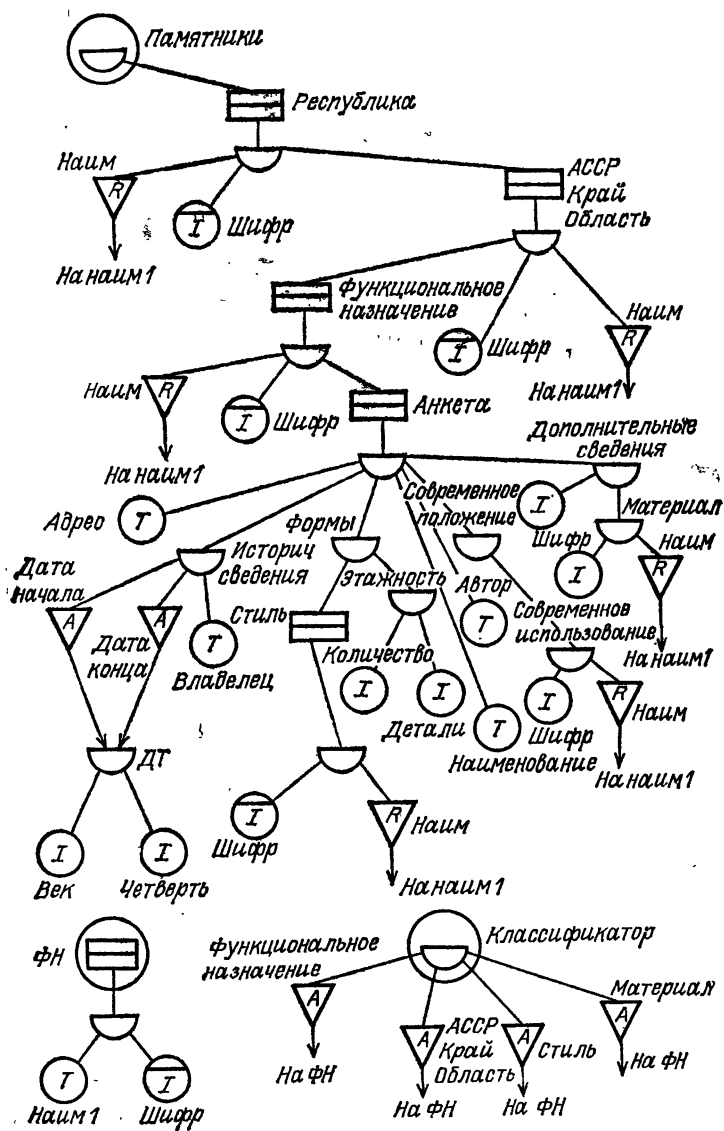


Рис. 5.25

Опишем более подробно диалог лица, сидящего за терминалом, и программы. В качестве параметров здесь выбраны: область, функциональное назначение, стиль и материал. Вначале перед пользователем высвечивается пустографка с четырьмя окнами для шифров:

ОБЛАСТЬ	<[]>
ФУНКЦ. НАЗН.	<[]>
СТИЛЬ	<[]>
МАТЕРИАЛ	<[]>

Одновременно в верхнем правом углу высвечивается текст, приглашающий пользователя задать шифры параметров. В ответ пользователь ставит в окна нужные ему шифры или, если он не знает шифр интересующего его параметра, ставит в соответствующее окно знак вопроса. После того как окна заполнены и пользователь нажал клавишу «ввод», шифры из окон принимаются программой в соответствующие переменные рабочего поля.

Если вместо шифра пользователь поставил знак вопроса, то программа высвечивает список всех шифров данного реквизита вместе с их наименованиями. В правом верхнем углу высвечивается сообщение с предложением выбрать нужный шифр. Теперь, чтобы передать шифр программе, нужно подвести курсор к строке с выбранным шифром и нажать клавишу «ввод». Так будут обработаны все знаки вопроса, поставленные в окна пустографки. Если пользователь не отметит никакой шифр и на месте шифра останется знак вопроса, то повторится показ пустографки с приглашением задать шифры. Пробёл в качестве значения шифра означает отмену ограничения на данный реквизит.

После того как шифры заданы, высвечивается заполненная пустографка, где рядом с шифрами стоят соответствующие этим шифрам наименования, а в правом верхнем углу — текст: «подтвердите выбор или введите новые шифры». Нажав клавишу «ввод», пользователь подтверждает выбранные шифры и пускает программу на выполнение. Он может также заменить шифр или опять поставить знак вопроса, если обнаружил ошибку.

Как только в запросе найден очередной памятник, на дисплей выводится его описание. Пунктами описания являются наименование памятника, адрес, век, стиль, материал и современное использование. После заполнения очередного экрана происходит пауза. Нажав «ввод», пользователь может пустить показ дальше, либо он может набрать сообщение для программы. С помощью этого сообщения пользователь имеет возможности:

- а) отменить печать протокола на АЦПУ командой *NOCOPY;
- б) отменить вывод каких-либо пунктов описания, набрав сообщение «имя1=НЕТ, имя2=НЕТ, . . .», например: «ВЕК=НЕТ, СТИЛЬ=НЕТ»;

в) включить вывод ранее отмененных пунктов, например «ВЕК=ДА»;

г) прервать выполнение запроса, набрав «СТОП»;

д) окончить задачу, набрав «КОНЕЦ» или «*END».

Перечисленные возможности реализуются с помощью средств передачи элементарных сообщений — системы ULINE. Ниже приводится более подробный разбор работы программы.

Программа имеет блочную структуру и состоит из блока задания шифров реквизитов «ПАРАМ», блока показа возможных значений шифров «ОБРВОП», вызываемого в случае ввода пользователем знака «?» и блока «ЗАПРОС», в котором происходит поиск памятников, удовлетворяющих заданным реквизитам, засылка сведений об этих памятниках в переменные рабочего поля, формирование описания с помощью функции EDIT и вывод этого описания на дисплей.

Остановимся подробнее на использовании системы ULINE. Вслед за высветкой пустографки с четырьмя окнами для шифров команда на картах 23,24 (см. пример) высвечивает текст: «заполните окна: шифром, знаком «?» или пробелом». Так как обращение к ULINE произведено в режиме PUTGET, то система переходит в состояние ожидания ответа. После того как окна заполнены и пользователь нажал клавишу «ввод», содержимое окон принимается в переменные WSECT, начиная с разделителя P0, указанного в команде (по способу 4, см. п. 5.5.3). В WSECT следом за P0 описаны четыре шифра ШО,ШФ,ШС,ШМ с односимвольными разделителями P1,P2,P3, нужными для приема промежутков между окнами. Полуслова D1, D2 и D3, D4 — две пары ограничителей. В первый ограничитель пары заносится максимальная длина сообщения, во второй — фактическая. В режиме PUT фактическая длина D4 задается в программе, в режиме GET — автоматически заносится системой после приема сообщения.

В том случае, когда пользователь затребовал значения имеющихся шифров, поставив знак вопроса, будет вызван блок «ОБРВОП», который производит высветку шифров и прием выбранного шифра. Шифры высвечиваются в окошках из угловых скобок, по одному в строке, рядом с шифром высвечивается соответствующее ему наименование. Теперь нужно подвести курсор к строке с выбранным шифром и нажать «ввод». В этом случае команда

A2ULAN('GET',&K,&B,&D3)

(на карте 47) произведет прием шифра в переменную B (по способу 3, см. п. 5.5.4). Если же шифров немного и весь список поместился в объем экрана, то паузы для засылки нужного шифра у пользователя не было. В этом случае команда прием не произвела, значение кода возврата K равно четырем и команда на карте 48 высветит

пользователю текст: «выберите нужное», а затем переводит систему в состояние ожидания ответа.

Аналогично используется режим PUTGET в блоке «ПАРАМ» (карта 37). После высветки заполненной пустографки нужна пауза, чтобы пользователь мог ответить программе.

После подтверждения шифров вызывается блок «ЗАПРОС», где происходит поиск и вывод на дисплей описания памятников, удовлетворяющих заданным реквизитам. Следом за командами вывода на карте 100 стоит команда приема возможного сообщения пользователя

```
IF %A2ULAN('GET',&K,...) THEN ...;
```

Форма записи этой команды эквивалентна двум операторам

```
%A2ULAN('GET',&K,...)
```

```
IF &K=0 THEN ...;
```

Код возврата K становится нулем, если прием сообщения произошел. Набрать сообщение пользователь может во время паузы после заполнения очередного экрана. С помощью этого сообщения можно реализовать описанные выше возможности управления. Это выполнено с помощью обращения к встроенной функции языка запросов RQBYNAME, которая раскладывает значения переменных из сообщения по именам этих переменных в структуру УПР. По заданному параметру I LINK=1 можно набирать имя не полностью, например: «АВТ=НЕТ» вместо «АВТОР=НЕТ». Значения «СТОП», «КОНЕЦ» или «*END», набранные без имени, принимаются в переменную ВЫКЛ, стоящую на первом месте структуры. Эта переменная управляет прерыванием выполнения запроса и окончанием задачи.

Вывод описания найденных памятников происходит в режиме с удержанием заголовков. Номер уровня заголовка задается в описании печатаемых частей макета в разделе OUTFORM. В частях макета управляющий байт содержит окно под этот номер. Макет, по которому идет вывод, приложен к тексту программы. Запрос запускается процедурой

```
//NAME JOB (A2,303,...,314),'фамилия',CLASS=B  
// EXEC ISREQCL,NAME=A2RF,R=220K,LCARD='72,15',  
//      TIME.REQ=5  
// EXEC ISREQGO,NAME=A2RF,R=280K  
//DODTREE DD DSN=MKDOD,UNIT=D,DISP=OLD,  
//      VOL=SER=BUSER1  
//DDTREE DD DSN=MKDD,UNIT=D,DISP=OLD,  
//      VOL=SER=BUSER1  
//A2PAM DD DSN=SOURCE(A2NPAM),DISP=SHR  
//DISPLAY1 DD UNIT=068
```

```
//DBSH1 DD SYSOUT=A
//PRINT2 DD SYSOUT=A
```

Параметр LCARD процедуры ISREQCL объявляет уровневую запись запроса. После выхода задачи на счет открывается терминал (карты 11,12). Если задача работает под монитором, то это выделяемый задаче дисплей, в противном случае это дисплей, описанный в DD-предложении с именем DISPLAY1. Здесь открыт прием с одновременным копированием выходного потока на АЦПУ. Эта печать протокола может быть отключена с дисплея командой *NOCOPY. Если не нужен режим копии, то карту 11 следует убрать, а в карте 12 вместо TV поставить PRINT.

Пример. Ниже приводится текст программы запроса. Нумерация карт в правом столбце используется только для ссылок из текста (п. 5.5.6).

```
00 WSECT 0
01 PO[1],ШО[2],P1[1],ШФ[2],P2[1],ШС[2],P3[1],ШМ[2], 1
   D1[Н],D2[Н],D3[Н],D4[Н], 2
   K[Н],I[Н],A[58],B[2], 3
   НОБЛ[58],НФН[59],СТНОБЛ[58],СТНФН[59] 4
01 BUF 5
02 ВВ[250],С[Н],D[2]
01 BUF1[80]
01 ОП 6
02 НАИМП[69],АДРП[250],ВЕКП[F],НСТ[69],НМАТ[58],АВТП[30],
   СВРИС[69],ВЛАД[250],ЭТАЖ[F] 7
01 УПР 8
02 ВЫКЛ[5],ВЕК[3],СТИЛЬ[3],АВТОР[3],ВЛАДЕЛЕЦ[3],
   МАТЕРИАЛ[3],ЭТАЖНОСТЬ[3],ИСПОЛЬЗ[3] 9

00 TEXT 10
01 %A2ULAN('OPEN(COPY)',&K,'PRINT ','TV ','DBSH ') 11
01 %A2ULAN('OPEN(TV)',&K,'SCW ','TV ') 12
01 DO WHILE &ВЫКЛ= 'КОНЕЦ' AND &ВЫКЛ= '*END'; 13
02 ОПАРАМ 14
02 ОЗАПРОС 15
01 %A2TV1('CLOSE') 16

+ + БЛОК ЗАДАНИЯ ПАРАМЕТРОВ ЗАПРОСА 17
00 BLOCK ПАРАМ 18
01 (%CLRWS) (&D1:=80) 19
02 DO WHILE &I=0; 20
03 %%PRINT('A2PAM.ZD') 21
03 (&D3:=12) (&D2:=44) 22
03 %A2ULAN('PTGET',&K, 23
   'ЗАПОЛНИТЕ ОКНА: ШИФРОМ, ЗНАКОМ'?' или
   ПРОБЕЛОМ',&D1,&PO,&D3) 24
03 IF &ШО=? THEN (&A='АССР КРАЙ ОБЛАСТЬ') 25
   ООБРВОП (&ШО:=&B); 26
03 IF &ШФ=? THEN (&A='ФУНКЦИОНАЛЬНОЕ 27
   НАЗНАЧЕНИЕ')
```

ОБРВОП (&ШФ;=&В);	28
03 IF &ШС='?' THEN (&А:='СТИЛЬ') ОБРВОП (&ШС;=&В);	29
03 IF &ШМ='?' THEN (&А:='МАТЕРИАЛ') ОБРВОП (&ШМ;=&В);	30
03 (&I:=1) КЛАССИФИКАТОР.	31
04 IF &ШО=' ' THEN АССР КРАЙ ОБЛАСТЬ. #&ШО. (&НОБЛ:=НАИМ);	32
04 IF &ШФ=' ' THEN ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ. #&ШФ. (&НФН:=НАИМ);	34
04 IF &ШС=' ' THEN СТИЛЬ.#&ШС.(&НСТ:=НАИМ);	34
04 IF &ШМ=' ' THEN МАТЕРИАЛ.#&ШМ.(&НМАТ:=НАИМ);	35
03 %%PRINT('А2РАМ.ЗД') (&D3:=12) (&D2:=41)	36
03 %A2ULAN('PTGET',&К,'ПОДТВЕРДИТЕ ВЫБОР ИЛИ ВВЕДИТЕ НОВЫЕ ШИФРЫ',&D1,&P0,&D3)	37
03 IF &ШО='?' OR &ШФ='?' OR &ШМ='?' OR &ШС='?' THEN (&I:=0);	39
++ БЛОК ПОКАЗА ВОЗМОЖНЫХ ЗНАЧЕНИЙ ПАРАМЕТРА	
00 BLOCK ОБРВОП	41
01 (&К:=1) (&В:='?') %%PRINT('F("11",2X,A76)',&А)	42
02 КЛАССИФИКАТОР.#&А.	43
03 ALL WHILE(&К=0).	44
04 %%PRINT('F(" 2","<","A2,"">","A66)',ШИФР,НАИМ)	45
04 (&D3:=2)	46
04 %A2ULAN('GET',&К,&В,&D3) (&D2:=15)	47
02 IF &К=4 THEN %A2ULAN('PTGET',&К,'ВЫБЕРИТЕ НУЖНОЕ',&D1,&В,&D3);	48
00 BLOCK ПЕРЫВ	49
01 &ВЫКЛ= 'СТОП' AND &ВЫКЛ= 'КОНЕЦ' AND &ВЫКЛ= '*END'	50
++ ВЫПОЛНЕНИЕ ЗАПРОСА	
00 BLOCK ЗАПРОС	51
01 %%PRINT('Z1') (&СЧ:=0)	52
01 ПАМЯТНИКИ. РЕСПУБЛИКА. ALL WHILE ОПЕРЫВ.	54
02 АССР КРАЙ ОБЛАСТЬ.	55
03 IF &ШО=' ' THEN ALL WHILE ОПЕРЫВ ELSE #&ШО; (&НОБЛ:=НАИМ) ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ.	56
IF &ШФ=' ' THEN ALL WHILE ОПЕРЫВ ELSE #&ШФ; (&НФН:=НАИМ) АНКЕТА. ALL WHILE ОПЕРЫВ.	58
04 (&L:=0)	59
04 IF &ШС=' ' OR (&ШС=' ' AND ФОРМЫ. СТИЛЬ. #&ШС) THEN (&L:=1);	60
04 IF NOT (&ШМ=' ' OR (&ШМ=' ' AND ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ. МАТЕРИАЛ. ШИФР=&ШМ)) THEN (&L:=0);	61
05 IF &L=1	62
05 THEN (&СЧ:=&СЧ+1) %CLRWS (&ОП)	63
++ ЗАСЫЛКА ВЫВОДИМЫХ СВЕДЕНИЙ	
06 (&АДРП:=АДРЕС) (&НАИМП:=НАИМЕНОВАНИЕ) (&АВТП:=АВТОР)	64
[ИСТОРИЧЕСКИЕ СВЕДЕНИЯ.(&ВЕКП:=ДАТА НАЧАЛА.ВЕК) IF &ВЕКП='0' THEN(&ВЕКП:=ДАТА	65

КОНЦА.ВЕК);	69
(&ВЛАД:=ВЛАДЕЛЕЦ)] (&НМАТ:=ДОПОЛНИТЕЛЬНЫЕ	
СВЕДЕНИЯ.МАТЕРИАЛ.НАИМ)	71
(&СВРИС:=СОВРЕМЕННОЕ ПОЛОЖЕНИЕ.	72
СОВРЕМЕННОЕ ИСПОЛЬЗОВАНИЕ.НАИМ)	73
(&ЭТАЖ:=ФОРМЫ.ЭТАЖНОСТЬ.КОЛИЧЕСТВО)	74
++ НАБОР ВЫВОДИМЫХ СВЕДЕНИЙ В ПЕРЕМЕННУЮ BUF	75
06 %EDIT('C,&BUF, '/')	76
IF &ВЕКП= 'O' AND &ВЕК= 'НЕТ'	77
THEN %EDIT('W,&BUF, 'ВЕК: ', &ВЕКП, ', ');	78
IF &АВТП= ' ' AND &АВТОР= 'НЕТ'	79
THEN %EDIT('W,&BUF, 'АВТОР: ', &АВТП, ', ');	80
IF &ВЛАД= ' ' AND &ВЛАДЕЛЕЦ= 'НЕТ'	81
THEN %EDIT('W,&BUF, 'ВЛАДЕЛЕЦ: ', &ВЛАД, ', ');	82
07_ IF &СТИЛЬ= 'НЕТ'	83
07_ THEN [ФОРМЫ.СТИЛЬ. %EDIT('W,&BUF, 'СТИЛЬ: ')	
ALL. (&НСТ:=НАИМ) %EDIT('W,&BUF, &НСТ, ', ')]	85
(&С:=&С-1) %EDIT('W,&BUF, ', ')]	86
06 IF &НМАТ= ' ' AND &МАТЕРИАЛ= 'НЕТ'	87
THEN %EDIT('W,&BUF, 'МАТЕРИАЛ: ', &НМАТ, ', ');	88
IF &ЭТАЖ= 'O' AND &ЭТАЖНОСТЬ= 'НЕТ'	89
THEN %EDIT('W,&BUF, 'ЧИСЛО ЭТАЖЕЙ: ', &ЭТАЖ, ', ');	90
IF &СВРИС= ' ' AND &ИСПОЛЬЗ= 'НЕТ'	91
THEN %EDIT('W,&BUF,	92
'СОВРЕМЕННОЕ ИСПОЛЬЗ-Е: ', &СВРИС, ', ');	93
IF &СТНОБЛ= &НОБЛ THEN %%PRINT('Z2')	94
(&СТНОБЛ:= &НОБЛ);	
IF &СТНФН= &НФН THEN %%PRINT('Z3')	
(&СТНФН:= &НФН);	
++ ПЕЧАТЬ СВЕДЕНИЙ ОБ ОЧЕРЕДНОМ НАЙДЕННОМ	
ПАМЯТНИКЕ	96
%%PRINT('PD')	97
%%PRINT('Z5') (&D3:=80)	98
++ ВОЗМОЖНЫЙ ПРИЕМ КОМАНДЫ С ДИСПЛЕЯ	99
06_ IF %A2ULAN('GET', &K, &BUF1, &D3)	100
06_ THEN %REQSTRAN(&BUF1, &D4)	101
07_ IF (&ILINK:=1) %RQB YNAME(&УПР, &ILINK)	102
07_ THEN <input type="checkbox"/> &ILINK	103
01 IF <input type="checkbox"/> ПЕРЕРЫВ THEN %%PRINT('Z4');	104
00 OUTFORM A2РАМ	105
01 ZD= &ШО, &НОБЛ, &ШФ, &НФН, &ШС, &НСТ, &ШМ, &НМАТ	106
01 Z2=2, &НОБЛ	107
01 Z3=3, &НФН	108
01 PD=4, &СЧ, &НАИМП, &АДРП, &ВВ	109
01 Z4= &СЧ	110

Далее следует текст макета, по которому идет вывод. Он записан в библиотеку исходных текстов под именем A2РАМ.

&АФОРМАРАМ ФОРМА 80

&ZD ЧАСТЬ СН

ОБЛАСТЬ <[?]>F(A58)
ФУНК.НАЗН<[?]>F(A58)
СТИЛЬ <[?]>F(A58)
МАТЕРИАЛ <[?]>F(A58)

&&Z1 ЧАСТЬ СН

1 СПИСОК ПАМЯТНИКОВ УДОВЛЕТВОРЯЮЩИХ ПАРАМЕТРАМ
ЗАПРОСА

F(69'_)

№	НАИМЕНОВАНИЕ	АДРЕС	ОПИСАНИЕ
---	--------------	-------	----------

F(69'_)

&&Z2 ЧАСТЬ

PI F(A58)

&&Z3 ЧАСТЬ

PI F(A58)

&&PD ЧАСТЬ

PI?99I F(A15) I F(A16) I F(A26)

ПАРАМЕТРАМ ЗАПРОСА УДОВЛЕТВОРЯЮТ ?999 ПАМЯТНИКОВ

F(69'_)

Ниже приведен протокол работы запроса. Тримя знаками***
отмечен искусственно добавленный текст, комментирующий действия
оператора.

ВЕРСИЯ ИНЕС 3.4 ОТ 09.06.83

ОБЛАСТЬ <[]>
ФУНК.НАЗН<[]>
СТИЛЬ <[]>
МАТЕРИАЛ V[]>

*** В первое и четвертое окно введен знак вопроса

АССР КРАЙ ОБЛАСТЬ

<11>АРХАНГЕЛЬСКАЯ

<17>ВЛАДИМИРСКАЯ

<19>ВОЛОГОДСКАЯ

<22>ГОРЬКОВСКАЯ

<24>ИВАНОВСКАЯ

<27>КАЛИНИНГРАДСКАЯ

<28>КАЛИНИНСКАЯ

<29>КАЛУЖСКАЯ

<34> КОСТРОМСКАЯ

<38>КУРСКАЯ

<46>МОСКОВСКАЯ *** Отмечена эта строка

<49>НОВГОРОДСКАЯ

<58>ПСКОВСКАЯ

<86>КАРЕЛЬСКАЯ АССР

<88>МАРИЙСКАЯ АССР

<89>МОРДОВСКАЯ АССР

МАТЕРИАЛ

< 1>КИРПИЧ

< 2>КАМЕНЬ ***Отмечена эта строка

< 3>ДЕРЕВО

< 4>КИРПИЧ И КАМЕНЬ

< 5>КАМЕНЬ И ДЕРЕВО

< 6>КИРПИЧ И ДЕРЕВО
 < 7>МЕТАЛЛ
 < 8>ЖЕЛЕЗОБЕТОН
 < 9>ЗЕМЛЕБИТНОЕ СООРУЖЕНИИ

ОБЛАСТЬ <[46]>МОСКОВСКАЯ
 ФУНК.НАЗН<[]>
 СТИЛЬ <[]>
 МАТЕРИАЛ <[2]>КАМЕНЬ

*** Программа просит подтвердить выбор
 *** В ответ нажата клавиша «ВВОД»

СПИСОК ПАМЯТНИКОВ, УДОВЛЕТВОРЯЮЩИХ ПАРАМЕТРАМ ЗАПРОСА

IN	НАИМЕНОВАНИЕ	АДРЕС	ОПИСАНИЕ
1	МОСКОВСКАЯ		
1	КУЛЬТОВЫЕ СООРУЖЕНИЯ		
1 1	Ц. НИКОЛЬСКАЯ	1 СТУПИНСКИЙ	1 ВЕК; 16;
1 1		1 Р-Н	1
1 1		1 С. ЕГАНОВО	1 СТИЛЬ; ДОПЕТРОВСКОЕ
1 1		1	1 ЗОДЧЕСТВО;
1 1		1	1 МАТЕРИАЛ; КАМЕНЬ;
1 1		1	1 ЧИСЛО ЭТАЖЕЙ; 1;
1 1		1	1 СОВРЕМЕННОЕ
1 1		1	1 ИСПОЛЬЗ-Е;
1 1		1	1 ВЕСХОЗНОСТЬ
1 1		1	1
1 2	Ц. ЗНАМЕНИЯ	1 КОЛОМЕНСКИЙ	1 ВЕК; 19;
1 1		1 Р-Н,	1 ВЛАДЕЛЕЦ; НОВИКОВ Я.В.
1 1		1 С. НЕПЕЦИНО	1
1 1		1	1 СТИЛЬ; КЛАССИЦИЗМ;
1 1		1	1 МАТЕРИАЛ; КАМЕНЬ;
1 1		1	1 ЧИСЛО ЭТАЖЕЙ; 1;
1 1		1	1 СОВРЕМЕННОЕ
1 1		1	1 ИСПОЛЬЗ-Е;
1 1		1	1 ПРОМЫШЛЕННО-
1 1		1	1 ХОЗЯЙСТВ. ХАРАКТЕР;
1 1		1	1
1 1	ПРОМЫШЛЕННО-ХОЗЯЙСТВЕННЫЕ И ТРАНСПОРТНЫЕ		
1 1	СООРУЖЕНИЯ		
1 3	КАМЕННЫЙ	1 Г. СЕРПУХОВ,	1 ВЕК; 18;
1 1	ПОГРЕБ С	1 УЛ. 2-Я	1 ВЛАДЕЛЕЦ;
1 1	НАПОГРЕБНИЦЕЙ	1 МОСКОВ-	1 КИШКИН Д. И.;
1 1		1 СКАЯ, 8—19	1 СТИЛЬ;
1 1		1	1 ОБЩЕЕВРОПЕЙСКОЕ
1 1		1	1 БАРОККО, ПОЗИЦИИ, НЕ
1 1		1	1 ПРЕДУСМОТРЕННЫЕ
1 1		1	1 ПЕРЕЧЕНЕМ РАЗДЕЛА;
1 1		1	1 МАТЕРИАЛ; КАМЕНЬ;
1 1		1	1 ЧИСЛО ЭТАЖЕЙ; 1
1 1		1	1 СОВРЕМЕННОЕ
1 1		1	1 ИСПОЛЬЗ-Е;
1 1		1	1 ПРОМЫШЛЕННО-
1 1		1	1 ХОЗЯЙСТВ. ХАРАКТЕР;
1 1		1	1

1	1	ОГРАДЫ И СТОРОЖКИ	1
1	41	КРЕПОСТНЫЕ	1 Г. СЕРПУХОВ, 1 ВЕК; 16;
1	1	СТЕНЫ	1 УЛ. 1 СТИЛЬ; ДОПЕТРОВСКОЕ
1	1	МОНАСТЫРЯ	1 ОКТЯБРЬСКАЯ, 1 ЗОДЧЕСТВО;
1	1		1 40 1 МАТЕРИАЛ; КАМЕНЬ;
1	1		1 1 СОВРЕМЕННОЕ
1	1		1 1 ИСПОЛЬЗ-Е;
1	1		1 1 БЕСХОЗНОСТЬ;
1	1		1 1

ПАРАМЕТРАМ ЗАПРОСА УДОВЛЕТВОРЯЮТ 4 ПАМЯТНИКОВ

*** Приглашение на запуск следующего запроса

ОБЛАСТЬ <[1]>
 ФУНК. НАЗН <[1]>
 СТИЛЬ <[1]>
 МАТЕРИАЛ <[1]>

*** Введен знак вопроса во второе окно

ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

< 1>ПАМЯТНИК ГРАДОСТРОИТЕЛЬНОГО ИСКУССТВА
 < 2>ЖИЛЫЕ ЗДАНИЯ
 < 3>АДМИНИСТРАТИВНО-ОБЩЕСТВЕННЫЕ ЗДАНИЯ
 < 4>КУЛЬТОВЫЕ СООРУЖЕНИЯ
 < 5>ВОЕННО-ОБОРОНИТЕЛЬНЫЕ СООРУЖЕНИЯ
 < 6>ПРОМЫШЛЕННО-ХОЗЯЙСТВЕННЫЕ И ТРАНСПОРТНЫЕ
 СООРУЖЕНИЯ
 < 7>САДОВО-ПАРКОВЫЕ СООРУЖЕНИЯ
 < 8>АРХИТЕКТУРНЫЕ МОНУМЕНТЫ
 < 9>ИНФОРМИРУЮЩИЕ СООРУЖЕНИЯ
 <10>ОГРАДЫ И СТОРОЖКИ
 <11>САДЫ И ПАРКИ
 <12>ПРУДЫ И КАНАЛЫ
 <13>ЖИЛЫЕ КОМПЛЕКСЫ

*** Отмечена пятая строка

ОБЛАСТЬ <[1]>
 ФУНК. НАЗН <[5]>
 СТИЛЬ <[1]>
 МАТЕРИАЛ <[1]>

СПИСОК ПАМЯТНИКОВ, УДОВЛЕТВОРЯЮЩИХ ПАРАМЕТРАМ ЗАПРОСА

1	N	1	НАИМЕНОВАНИЕ	1	АДРЕС	1	ОПИСАНИЕ	1
1			КАЛИНИНСКАЯ					1
1			ВОЕННО-ОБОРОНИТЕЛЬНЫЕ СООРУЖЕНИЯ					1
1	11		КАВАЛЕРИЙСКИЕ	1	Г. КАЛИНИН,	1	1 ВЕК; 18;	1
1	1		КАЗАРМЫ	1	ПЕРВОМАЙСКАЯ	1	1 СТИЛЬ:	1
1	1			1	НАБ, 36	1	1 КЛАССИЦИЗМ;	1
1	1			1		1	1 МАТЕРИАЛ: КИРПИЧ;	1
1	1			1		1	1 ЧИСЛО ЭТАЖЕЙ; 2;	1
1	1			1		1	1 СОВРЕМЕННОЕ	1
1	1			1		1	1 ИСПОЛЬЗ-Е;	1
1	1			1		1	1 СМЕШАННЫЙ	1
1	1			1		1	1 ХАРАКТЕР;	1

МОСКОВСКАЯ				I
I	21 СЕВЕРО-	I	СТУПИНСКИЙ Р-Н	I ВЕК; 17;
I	I ВОСТОЧНАЯ	I	ПОС.	I СТИЛЬ;
I	I КРУГЛАЯ БАШНЯ	I	БЕЛОПЕСОЦКАЯ	I ДОПЕТРОВСКОЕ
I	I	I	СЛОБОДА	I ЗОДЧЕСТВО;
I	I	I		I МАТЕРИАЛ; КИРПИЧ;
I	I	I		I ЧИСЛО ЭТАЖЕЙ; 2;
I	I	I		I СОВРЕМЕННОВ
I	I	I		I ИСПОЛЬЗ-Е;
I	I	I		I БЕСХОЗНОСТЬ;
I	I	I		I
I	31 СЕВЕРО-	I	СТУПИНСКИЙ Р-Н	I ВЕК; 17;
I	I ЗАПАДНАЯ	I	ПОС.	I СТИЛЬ;
I	I КРУГЛАЯ БАШНЯ	I	БЕЛОПЕСОЦКАЯ	I ДОПЕТРОВСКОЕ
I	I	I	СЛОБОДА	I ЗОДЧЕСТВО;
I	I	I		I ЭКЛЕКТИКА;
I	I	I		I МАТЕРИАЛ; КИРПИЧ;
I	I	I		I ЧИСЛО ЭТАЖЕЙ; 2;
I	I	I		I СОВРЕМЕННОВ
I	I	I		I ИСПОЛЬЗ-Е;
I	I	I		I БЕСХОЗНОСТЬ;
I	I	I		I
I	I	ПАРАМЕТРАМ ЗАПРОСА УДОВЛЕТВОРЯЮТ		3 ПАМЯТНИКОВ

ОБЛАСТЬ <[]>

ФУНК. НАЗН <[]>

СТИЛЬ <[]>

МАТЕРИАЛ <[]>

*** Введен знак вопроса в первое окно

*** и цифра 2 во второе

АССР КРАЙ ОБЛАСТЬ

<11>АРХАНГЕЛЬСКАЯ

<17>ВЛАДИМИРСКАЯ

<19>ВОЛОГОДСКАЯ

<22>ГОРЬКОВСКАЯ

<24>ИВАНОВСКАЯ

<27>КАЛИНИНГРАДСКАЯ

<28>КАЛИНИНСКАЯ

<29>КАЛУЖСКАЯ

<34>КОСТРОМСКАЯ

<38>КУРСКАЯ

<46>МОСКОВСКАЯ

<49>НОВГОРОДСКАЯ

<58>ПСКОВСКАЯ

<86>КАРЕЛЬСКАЯ АССР

<88>МАРИЙСКАЯ АССР

<89>МОРДОВСКАЯ АССР

ОБЛАСТЬ <[11]>АРХАНГЕЛЬСКАЯ

ФУНК. НАЗН <[2]>ЖИЛЫЕ ЗДАНИЯ

СТИЛЬ <[]>

МАТЕРИАЛ <[]>

СПИСОК ПАМЯТНИКОВ, УДОВЛЕТВОРЯЮЩИХ ПАРАМЕТРАМ ЗАПРОСА

IN	НАИМЕНОВАНИЕ	АДРЕС	ОПИСАНИЕ	I
1	АРХАНГЕЛЬСКАЯ			1
1	ЖИЛЫЕ ЗДАНИЯ			1
1	11 ДОМ	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 19;	1
1	1 ФИЛИППОВОЙ	1 Д. КЛИМОВСКАЯ	1 СТИЛЬ: ПОЗИЦИИ, НЕ	1
1	1	1	1 ПРЕДУСМОТРЕННЫЕ	1
1	1	1	1 ПЕРЕЧНЕМ РАЗДЕЛА;	1
1	1	1	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 1;	1
1	1	1	1 СОВРЕМЕННОЕ	1
1	1	1	1 ИСПОЛЬЗ-Е:	1
1	1	1	1 ПО ПЕРВОНАЧАЛЬНОМУ	1
1	1	1	1 НАЗНАЧЕНИЮ;	1
1	1
1	1	1	1	1
1	61 ДОМ	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 19;	1
1	1 МАРТЫНОВА	1 Д. Б. ЗАВОЛЖЬЕ	1 СТИЛЬ: КЛАССИЦИЗМ,	1
1	1	1	1 СТИЛИЗАТОРСТВО	1
1	1	1	1 ВТОРОЙ И ПОСЛЕДНЕЙ	1
1	1	1	1 ТРЕТИ 19 ВЕКА;	1
1	1	1	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 2;	1
1	1	1	1 СОВРЕМЕННОЕ	1
1	1	1	1 ИСПОЛЬЗ-Е:	1
1	1	1	1 БЕСХОЗНОСТЬ;	1
1	1	1	1	1

*** Командой «ВЛА=НЕТ, СТИ=НЕТ, ИСП=НЕТ» отключены

*** соответствующие пункты описания

1	61 ДОМ ТОРЕВА	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 20;	1
1	1	1 Д. Б. ЗАВОЛЖЬЕ	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 2;	1
1	1	1	1	1
1	71 ДОМ ЖУКОВА	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 19;	1
1	1	1 Д. ЗАНИВА	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 1;	1
1	1	1	1	1
1	81 ДОМ	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 19;	1
1	1 МАРТЫНОВА Н.	1 Д. Б. ЗАВОЛЖЬЕ	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 1;	1
1	1	1	1	1
1	91 ДОМ	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 19;	1
1	1 ПРОКУШИНОЙ	1 Д. ПАРХАЧЕВСКАЯ	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 2;	1
1	1	1	1	1
1	101 ДОМ БАТУНОВА	1 КОНОШСКИЙ Р-Н,	1 ВЕК: 19;	1
1	1	1 С. КРЕМЛЕВО	1 МАТЕРИАЛ: ДЕРЕВО;	1
1	1	1	1 ЧИСЛО ЭТАЖЕЙ: 2;	1
1	1	1	1	1

*** Задача окончена командой «КОНЕЦ»

ЗАКЛЮЧЕНИЕ

Первые информационные системы на базе СУБД ИНЕС были разработаны и сданы в эксплуатацию в 1978 г., промышленное распространение ИНЕС началось с декабря 1978 г. В начале 1983 г. число ее пользователей превысило 1000, а к 1988 г. их зарегистрировано более 1500 (в 180 городах). Следует, однако, отметить, что организации, получившие версии ИНЕС, не давали обязательств по их использованию.

Одно из основных преимуществ ИНЕС как программного продукта состоит в ее простоте и доступности для самостоятельного освоения и внедрения пользователями. На практике для абсолютного большинства пользователей, внедривших в эксплуатацию задачи на основе ИНЕС, единственной формой сопровождения системы были и остаются консультации по конкретным вопросам программирования средствами ИНЕС. В этих условиях сведения о внедрении задач на основе ИНЕС в опытную и промышленную эксплуатацию были получены в 1981 г. от 100 организаций, в 1983 г. — примерно от 200 организаций. Таким образом, как в 1981 г., так и в 1983 г. ИНЕС эксплуатировалась не менее чем в каждой пятой организации из числа получивших систему.

Установка ИНЕС, ее поддержание и использование не требуют больших затрат труда системных программистов. В 1981 г. 40%, а в 1983 г. 52% организаций, внедривших системы на базе ИНЕС, специально для работы с ИНЕС системных программистов не выделяли. Остальные организации выделяли, как правило, не более двух системных программистов.

Количественные характеристики баз данных получены в результате обследования внедрений ИНЕС, проведенного в мае — июне 1983 г. (предыдущее обследование было проведено осенью 1981 г. [4,11]) и приуроченного к началу распространения новой версии ИНЕС. Анкетный опрос пользователей проводился одновременно с оповещением их о выпуске этой версии. Банк «Пользователи», который поддерживается во ВНИИСИ с первых месяцев рас-

пространения ИНЕС, дал возможность наладить автоматизированную распечатку и рассылку листов с анкетой и оповещением, адресованных конкретным пользователям. Заполненные листы, а также акты о внедрении ИНЕС в опытную и промышленную эксплуатацию, присланные во ВНИИСИ (всего около 230 анкет и актов), стали основным материалом для анализа [11].

Набор средств программирования, предоставляемых СУБД, и интенсивность использования этих средств в различных разработках характеризуют степень универсальности СУБД. Данные обследования позволяют оценить интенсивность использования отдельных подсистем ИНЕС, что отражено в следующей таблице (следует отметить, что в таблице процент применения различных средств ИНЕС в 230 обследованных организациях подсчитывался без учета их оборудования, т. е. наличия в них терминалов).

Подсистема	Частота применения, %	
	в организациях, работающих с ИНЕС	в организациях, внедривших ИНЕС*)
Средства ввода	92,3	93,4
Средства манипулирования данными высокого уровня	91,7	92,8
Средства манипулирования данными низкого уровня	45,5	45,6
Средства вывода	85,2	86,0
Средства ведения словарей	67,7	64,0
Диалоговые средства просмотра и корректировки БД	53,2	53,7
Диалоговые средства просмотра результатов счета	35,3	35,3
Средства организации диалога	37,2	38,2
Все составляющие ИНЕС	15,9	29,4

*) Обследование, 1987 г. дало следующие данные: 96, 89, 58, 92, 75, 70, 47, 63, 39.

В заключение можно отметить, что около 60% пользователей ИНЕС не применяют других СУБД. В организациях, работающих одновременно с несколькими СУБД, в среднем используется 1—2 СУБД, максимум — 6 СУБД, не считая ИНЕС.

Проведенное обследование показало, что в настоящее время ИНЕС успешно применяется для решения разнообразных типов задач, причем сфера ее применения непрерывно расширяется.

СПИСОК ЛИТЕРАТУРЫ

1. Авен О. И. Что такое АСУ?.— 2-е изд.— М.: Наука, 1985.— 180 с.
2. Арлазаров В. Л., Емельянов Н. Е., Дюкалов А. Н. и др. Информационная система ИНЕС//Автоматика и телемеханика, 1979.— № 6.— С. 109—121.
3. Арлазаров В. Л., Емельянов Н. Е. Общее описание системы ИНЕС//Проблемы МСНТИ/МЦНТИ, 1982.— № 1.— С. 46—59.
4. Арлазаров В. Л., Емельянов Н. Е., Астрина И. В. Организация создания и распространения программного продукта (на опыте системы ИНЕС)//Автоматика и телемеханика, 1984.— № 5.— С. 47—52.
5. Арлазаров В. Л. СУБД как инструментальная система.— Препринт/ВНИИСИ АН СССР.— М., 1985.— 32 с.
6. Арлазаров В. Л., Астахов А. Д., Брудно А. А. и др. Организация обменов с внешней памятью в СУБД ИНЕС//Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 4—9.
7. Арлазаров В. Л., Донской М. В., Емельянов Н. Е. Структура и возможности ИНЕС//Банки данных в автоматизированных системах: Труды школы ВДНХ.— Калинин.: НПО ЦПС.— 1986. С. 7—13.
8. Арлазаров В. Л., Емельянов Н. Е. Теоретический анализ распространенных концепций баз данных//Банки данных: Труды 3-й Всесоюзной конференции.— Таллин, 1985.— С. 191—202.
9. Астахов А. Д. Организация эффективного доступа на основе хеширования//Программирование, 1983.— № 2.— С. 37—43.
10. Артре Ш. Структурный подход к организации баз данных/Пер. с англ.— М.: Финансы и статистика, 1983.— 316 с.
11. Астрина И. В., Емельянов Н. Е. Количественные характеристики баз данных (обзор по материалам обследования применений СУБД ИНЕС)//Вопросы радиоэлектроники: Сер. АСУПР, 1984.— № 4.— С. 68—86.
12. Богданов Д. С., Емельянов Н. Е., Славянов Н. Н. Макетный редактор документов//Вопросы ведения баз данных средствами СУБД ИНЕС.— М.: ВНИИСИ, 1985.— С. 35—42.

13. Борисова Л. П., Емельянов Н. Е., Иванова Н. А., Солдатов В. А. Вывод данных в ИНЕС//Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 65—75.
14. Брудно А. А., Шерман А. А. Системный журнал СУБД ИНЕС//Вопросы разработки системы управления базами данных с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 99—106.
15. Брудно А. А., Шерман А. А. Средства защиты данных в СУБД ИНЕС//Вопросы разработки и ведения баз данных средствами СУБД ИНЕС.— М.: ВНИИСИ, 1985.— С. 90—98.
16. Годунов А. Н., Егоров Г. В., Емельянов Н. Е. и др. Средства вывода элементарных сообщений и управление передачей сообщений в системе ИНЕС.— Препринт/ВНИИСИ АН СССР.— М., 1984.— 40 с.
17. Годунов А. Н., Емельянов Н. Е., Романов А. П., Свердлов С. С. Управление выводом сообщений в системе ИНЕС//Программирование.— 1984.— № 6.— С. 52—57.
18. Годунов А. Н., Емельянов Н. Е., Свердлов С. С. Многотерминальный монитор системы управления базами данных ИНЕС//Программирование.— 1982.— № 2.— С. 64—69.
19. Годунов А. Н., Емельянов Н. Е., Свердлов С. С. и др. Диалоговые средства информационной системы ИНЕС//Автоматика и телемеханика.— 1982.— № 10.— С. 115—118.
20. Годунов А. Н., Емельянов Н. Е., Хабарова С. А., Чернышева И. Б. Средства системы ИНЕС для ввода документов сложной структуры в языке ПЛ-1//Программирование.— 1983.— № 2.— С. 37—43.
21. Годунов А. Н., Емельянов Н. Е., Талалай А. Б. Просмотр текстов//Программирование.— 1984.— № 1.— С. 69—75.
22. Годунов А. Н., Донской М. В. Средства системы ИНЕС для создания и ведения наборов данных//Программирование.— 1986.— № 3.— С. 76—83.
23. Голоусикова Н. И., Деза В. Н., Леман А. А., Усков А. В. Ввод данных в ИНЕС//Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 41—45.
24. Дейт К. Введение в системы баз данных/Пер. с англ.— М.: Наука, 1980.— 464 с.
25. Диниц Е. А., Егоров Г. В., Иванова Н. А. и др. Язык высокого уровня для манипулирования данными (язык запросов) и его реализация в ИНЕС//Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 48—53.
26. Егоров Г. В. Организация работы запросной системы с несколькими базами данных//Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 53—58.
27. Емельянов Н. Е. Макетный метод организации интерфейса с базами данных//Вопросы радиоэлектроники: Сер. АСУПР, 1985.— № 1.— С. 84—88.
28. Емельянов Н. Е. Проблемы автоматизации документирования.— Препринт/ВНИИСИ АН СССР.— М., 1986.— 62 с.

29. Емельянов Н. Е., Щелкачева И. В. Проектирование баз данных в среде СУБД ИНЕС.— Препринт/ВНИИСИ АН СССР.— М.: 1986.— 60 с.
30. Емельянов Н. Е., Чернышева И. Б. Программирование диалоговых систем для ЕС ЭВМ//Прикладная информатика. Вып. 1.— М.: Финансы и статистика, 1986.— С. 91—107.
31. Емельянов Н. Е., Жаринов А. Н., Солдатов В. А. Лабораторный практикум по курсу «Банки данных»: Учеб. пособие для вузов.— М.: МИСиС, 1988.— 120 с.
32. Емельянов Н. Е., Солдатов В. А. Форматирование предпечатных наборов данных//Вопросы информационной технологии.— М.: ВНИИСИ, 1986.— С. 53—57.
33. Замулин А. В., Скопин И. Н. Типы данных в языках программирования//Прикладная информатика. Вып. 2.— М.: Финансы и статистика, 1984.— С. 68—92.
34. Информационная система для ЕС ЭВМ (ИНЕС): в 17 томах/ В. Л. Арлазаров, А. Д. Астахов, Е. Д. Богданова, Л. П. Борисова, А. А. Брудно, Н. Е. Бузикашвили, Е. В. Водкин, Е. М. Глушанков, А. Н. Годунов, Е. В. Голенко, Н. И. Голоусикова, Я. Ю. Гольфанд, В. Н. Деза, Е. А. Диниц, Е. Ю. Долгопятова, З. Ю. Долгопятова, М. В. Донской, Г. В. Егоров, Н. Е. Емельянов, Н. А. Иванова, Ю. Н. Иванов, А. В. Карванов, А. Ф. Кирсанов, О. А. Кузнецова, А. А. Леман, Т. Г. Лепешова, А. Б. Мерков, Е. Л. Плискин, Ф. В. Поцелуева, Г. С. Разина, М. З. Розенфельд, А. П. Романов, Л. Н. Сауков, С. С. Свердлов, Н. Н. Славянов, В. А. Солдатов, А. Б. Талалай, В. В. Токарев, В. В. Усков, И. А. Фарраджев, В. В. Фарсобиная, Н. А. Филиппев, М. Б. Фролов, Е. М. Фокина, М. Е. Фурман, С. А. Хабарова, И. Б. Чернышева, В. В. Чистяков, А. А. Шерман, И. В. Щелкачева,— Калинин: НПО «Центрпрограммсистем», 1984.— 998 с.
35. Информационная система для ЕС ЭВМ (ИНЕС). Описание данных. 4718983.00001—01 33 01.— Калинин: НПО «Центрпрограммсистем», 1984.— 24 с.
36. Информационная система для ЕС ЭВМ (ИНЕС). Ввод данных. 4718983.00001—02 33 02.— Калинин: НПО «Центрпрограммсистем», 1984.— 108 с.
37. Информационная система для ЕС ЭВМ (ИНЕС). Система записей. 4718983.00001—03 33 03.— Калинин: НПО «Центрпрограммсистем», 1984.— 168 с.
38. Информационная система для ЕС ЭВМ (ИНЕС). Доступ к базе данных. 4718983.00001—04 33 04.— Калинин: НПО «Центрпрограммсистем», 1984.— 95 с.
39. Информационная система для ЕС ЭВМ (ИНЕС). Вывод данных. 4718983.00001—05 33 05.— Калинин: НПО «Центрпрограммсистем», 1984.— 46 с.
40. Информационная система для ЕС ЭВМ (ИНЕС). Словарная система. 4718983.00001—06 33 06.— Калинин: НПО «Центрпрограммсистем», 1984.— 47 с.
41. Информационная система для ЕС ЭВМ (ИНЕС). Обслуживание баз данных. 4718983.00001—07 33 07.— Калинин: НПО «Центрпрограммсистем»,— 1984.— 62 с.
42. Информационная система для ЕС ЭВМ (ИНЕС). Терминальные средства. 4718983.00001—08 33 08.— Калинин: НПО «Центрпрограммсистем», 1984.— 123 с.
43. Информационная система для ЕС ЭВМ (ИНЕС). Информацион-

- но-справочные системы. 4718983.00001—01 33 09. — Калинин: НПО «Центрпрограммсистем», 1984.— 49 с.
44. Информационная система для ЕС ЭВМ (ИНЕС). Системные средства. 4718983.00001—01 32 01. — Калинин: НПО «Центрпрограммсистем», 1984.— 54 с.
 45. Информационная система для ЕС ЭВМ (ИНЕС). Средства программирования. 4718983.00001—01 32 02.— Калинин: НПО «Центрпрограммсистем», 1984.— 101 с.
 46. Информационная система для ЕС ЭВМ (ИНЕС). Утилиты общего назначения. 4718983.00001—01 32 03.— Калинин: НПО «Центрпрограммсистем», 1984.— 58 с.
 47. Информационная система для ЕС ЭВМ (ИНЕС). Установка ИНЕС. 4718983.00001—01 34 01.— Калинин: НПО «Центрпрограммсистем», 1984.— 14 с.
 48. Информационная система для ЕС ЭВМ (ИНЕС). Описание контрольного примера. 4718983.00001—01 91 01.— Калинин: НПО «Центрпрограммсистем», 1984.— 22 с.
 49. Информационная система для ЕС ЭВМ (ИНЕС). Описание применения. 4718983.00001—01 31 01.— Калинин: НПО «Центрпрограммсистем», 1984.— 27 с.
 50. Иофинова М. Е., Кирсанов А. Ф., Кузнецова О. А. и др. Диалоговая система доступа RED//Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 95—99.
 51. Калинин Л. А. Методы и средства интеграции неоднородных баз данных.— М.: Наука, 1983.— 423 с.
 52. Карзанов А. К., Талалай А. Б. Физическая организация иерархических и сетевых структур данных в ИНЕС// Вопросы разработки системы управления базами данных с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 9—23.
 53. Леман А. А. Система ведения картотек СУБД ИНЕС// Вопросы разработки и ведения баз данных средствами СУБД ИНЕС.— М.: ВНИИСИ, 1985.— С. 42—45.
 54. Мартин Дж. Планирование развития автоматизированных систем.— М.: Финансы и статистика, 1984.— 196 с.
 55. Опыт использования баз данных ИНЕС: Материалы семинара/ МДНТП; под ред. Н. Е. Емельянова, М. В. Донского — М.: МДНТП, 1983.— 112 с.
 56. Поцелуева Ф. В., Розенфельд М. З. Описание данных ИНЕС// Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ.— 1983.— С. 32—38.
 57. Поцелуева Ф. В., Розенфельд М. З. Система элементарного доступа к базам данных в ИНЕС// Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 38—41.
 58. Славянов Н. Н. Использование базы данных для подготовки музейных каталогов// Вопросы ведения баз данных средствами СУБД ИНЕС.— М.: ВНИИСИ, 1985.— С. 81—95.
 59. Системы управления базами данных для ЕС ЭВМ: Справочник./ Под ред. В. М. Савинкова.— М.: Финансы и статистика, 1984.— 224 с.
 60. Солдатов В. А. Стандартные средства контроля входной информации// Вопросы разработки СУБД с проблемным окружением.— М.: ВНИИСИ, 1983.— С. 47—48.
 61. Ульман Дж. Основы систем баз данных / Пер. с англ.— М.: Финансы и статистика, 1983.— 334 с.

62. Уэлдон Дж.-Л. Администрирование баз данных.— М.: Финансы и статистика, 1984.— 207 с.
63. Цаленко М. Ш. Семантические и математические модели баз данных // Итоги науки и техники: Сер. Информатика / ВИНТИ. Т. 9.— М.: ВИНТИ, 1985.— 207 с.
64. Цикритзис Д., Лоховски Ф. Модели данных.— М.: Финансы и статистика, 1985.— 344 с.
65. Щелкачев И. В. Оценка памяти и времени работы баз данных, созданных средствами СУБД ИНЕС // Вопросы разработки и ведения баз данных средствами СУБД ИНЕС.— М.: ВНИИСИ, 1985.— С. 67—76.

Емельянов Николай Евгеньевич

ВВЕДЕНИЕ В СУБД ИНЕС

Серия «Библиотечка программиста», выпуск 54

Редактор *О. И. Сухова*

Художественный редактор *Г. М. Коровина*

Технический редактор *В. Н. Кондакова*

Корректоры *О. А. Бутусова, И. Я. Кришталъ*

ИБ № 12833

Сдано в набор 05.05.87. Подписано к печати 03.05.88. Т-09585.
 Формат 84×108/32. Бумага тип. № 2. Усл. печ. л. 13,44.
 Усл. кр.-от. 13,65. Уч.-изд. л. 15,59. Тираж 24 500 экз.
 Заказ № 881. Цена 1 р.

Ордена Трудового Красного Знамени издательство «Наука»
 Главная редакция физико-математической литературы
 117071 Москва В-71, Ленинский проспект, 15

Ордена Октябрьской Революции и ордена Трудового Красного
 Знамени МПО «Первая Образцовая типография» имени
 А. А. Жданова Союзполиграфпрома при Государственном комитете
 СССР по делам издательств, полиграфии и книжной торговли
 113054 Москва, Валовая, 28

Отпечатано во 2-й типографии издательства «Наука».
 121099 Москва Г-99, Шубинский пер., 6. Заказ 1715

Цена 1 р.

5B

E 601