

ISSN 0136-3549

3134-3823

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED

556

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

ТРИ
'83

ПОСТРОЕНИЕ ПРИКЛАДНЫХ СИСТЕМ
ОБРАБОТКИ ДАННЫХ -
СИСТЕМА ПАРЕС

Ep.6.7

556

**ТРИ
'83**

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

ПОСТРОЕНИЕ
ПРИКЛАДНЫХ
СИСТЕМ
ОБРАБОТКИ
ДАННЫХ -
СИСТЕМА
ПАРЕС

Электротехника и автоматика XXIII

Таллин 1983

С о д е р ж а н и е

1. Система ПАРЕС.....	3
2. В.А. Крахт. От концептуальной модели предметной области до структуры базы данных.....	5
3. В.А. Крахт. Логическое конструирование программ обработки данных реляционно-решетчатой структуры	29
4. Ю.Э. Эйвак. Алгоритмы реализации операторов поиска данных реляционно-решетчатой структуры.....	49
5. У.А. Пауклин. Алгоритмы реализации предикатов с кванторами, связанных с поиском данных реляционно-решетчатой структуры.....	61
6. М.Х. Вассиль. Реляционно-решетчатая структура базы данных.....	67
7. М.Х. Вассиль. Реорганизация и реструктуризация базы данных реляционно-решетчатой структуры.....	83
8. Т.В. Ордлик. Структуризация и описание коммуникационных данных в системе ПАРЕС.....	91
9. Э.П. Рооталу. Разработка систем обработки данных с базой данных реляционно-решетчатой структуры..	99

ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Труды ТПИ № 556

ПОСТРОЕНИЕ ПРИКЛАДНЫХ СИСТЕМ ОБРАБОТКИ
ДАННЫХ - СИСТЕМА ПАРЕС

Электротехника и автоматика XXIII

Редактор В. Кукк

Техн. ред. М. Тамме

Сборник утвержден коллегией Трудов ТПИ 10.05.83

Подписано к печати 11.10.83

Формат 60x90/16. Печ. л. 7,0 + 0,25 прил. Уч.-изд. л. 5,8

Тираж 350. МВ-08833

Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Зак. № 668

Цена 85 коп.

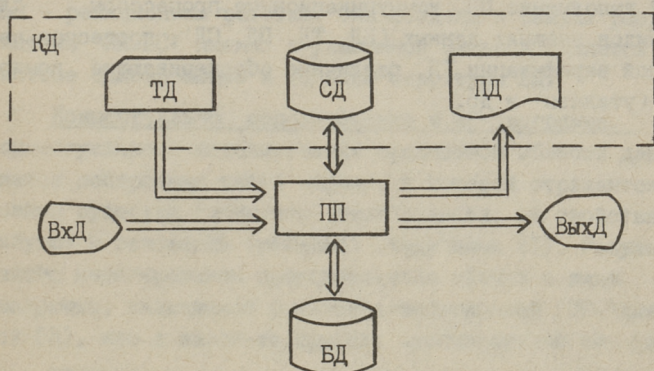


© Таллинский политехнический институт, 1983

СИСТЕМА ПАРЕС

Система ПАРЕС представляет собой систему программного обеспечения для ЕС ЭВМ (на базе ОС), сочетающую в себе основные функции системы программирования задач обработки данных и системы управления базами данных (СУБД). Она предназначена для создания прикладных систем обработки информации с базой данных реляционно-решетчатой структуры.

Прикладная система обработки информации, создаваемая при помощи системы ПАРЕС, включает в себя базу данных (БД) указанной структуры, множество прикладных программ (ПП), необходимых для актуализации БД, воспроизведения данных и решения проблемных задач, и множество связанных с ПП файлов коммуникационных данных (КД). Последние делятся на транзакционные данные (ТД), представляющие собой вводимые в систему исходные данные для актуализации БД, печатные данные (ПД) – выводимые из системы и оформленные в виде выходных документов результаты обработки, и совместимые данные (СД) – вводимые в систему или выводимые из нее данные, составляющие стандартные файлы данных. Кроме того, ПП мо-



гут быть связаны со стандартными входным и выходным файлами данных (ВхД, ВыхД).

Система ПАРЕС снабжена собственными входными языками высокого уровня. К ним относятся декларативные языки описания данных (ЯОД) и алгоритмический язык манипулирования данными (АЯМД).

К числу ЯОД системы относятся язык описания базовых данных (ЯОБД), состоящий из языка описания модели логической структуры БД и языка описания ареала (среды хранения) БД, и язык описания коммуникационных данных (ЯОКД), включающий в себя языки описания ТД, ПД и СД (ЯОТД, ЯОПД, ЯОСД). Эти языки обеспечивают автономное описание обрабатываемых данных и их независимость от программ обработки.

АЯМД системы является самостоятельным языком программирования, ориентированным на обработку данных реляционно-решетчатой структуры. Он объединяет основные функции базового языка и языка манипулирования данными общеизвестных СУБД с включающим языком. АЯМД является одновременно как языком запросов данных из БД, так и языком программирования задач обработки данных, хранимых в БД. На этом языке составляются все ПП, входящие в состав системы обработки информации, созданной при помощи системы ПАРЕС.

В состав системы ПАРЕС входят трансляторы со всех входных языков системы, управляющий процессор, обеспечивающий выполнение ПП, коммуникационные процессоры для всех видов внешних данных (БД, ТД, ПД, СД), процессор автономной верификации ТД, различные обеспечивающие компоненты (утилиты) и др.

ОТ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ ПРЕДМЕТНОЙ ОБЛАСТИ
ДО СТРУКТУРЫ БАЗЫ ДАННЫХ

Введение. Современный подход к проектированию баз данных (БД) основывается на концептуальном моделировании предметной области данных и определении структуры базы исходя из результатов такого моделирования. Возникновение и развитие этого подхода направлено на преодоление трудностей, связанных с обработкой хранимой в БД информации.

Необходимость построения БД на основе концептуальной модели предметной области впервые была высказана в предложениях ANSI/X3/SPARC [1]. С тех пор появилось много работ, посвященных концептуальному моделированию предметной области данных [2, 3, 4]. Но общее решение проблемы, на наш взгляд, до сих пор отсутствует.

В данной работе рассматривается новая метамодель представления концептуальной структуры предметной области данных – реляционная решетка. Она создана в рамках разработки системы "ПАРЕС" именно с указанной целью и является в этой системе одновременно и моделью структуры БД.

Концептуальное моделирование и ЭР-диаграмма. Целью концептуального моделирования предметной области данных является построение такой модели, в которой отражаются основные существа (энтитеты) этой области, их свойства (атрибуты) и отношения (реляции) между ними [1]. Результаты такого моделирования представляются обычно в виде особой диаграммы, называемой энтитетно-реляционной (ЭР-) диаграммой [3], или в какой-то другой, эквивалентной ей форме.

ЭР-диаграмма представляет собой граф особого вида. Особенностью ее является то, что в ней выделяется два основных вида элементов (вершин графа), представляющих суще-

ства и отношения соответственно. Часто в ЭР-диаграмме указывается еще третий вид элементов, представляющих атрибуты существ, а иногда и атрибуты отношений. Пример ЭР-диаграммы с выделением в ней указанных двух основных видов элементов приведен на фиг. 1.



Фиг. 1. Пример ЭР-диаграммы.

ЭР-диаграмма дает только самое общее представление о предметной области. Для ее полного описания требуется спецификация и формальное описание семантики всех элементов диаграммы.

Под существами понимаются абстрактные объекты (модельные образы), представляющие реальные или идеализированные объекты данной предметной области, рассмотренные отвлеченными от их свойств и связей с другими объектами.

Среди свойств каждого существа выделяется одно особое свойство - принадлежность к определенному типу (классу) объектов. Таким образом, каждое существо рассматривается как представитель объектов данного типа или как образ всего класса объектов данного типа. Тип объектов выражается именем существа.

Остальные свойства существа представляются в параметрической форме - приписыванием ему параметров, свойственных объектам данного типа. Эти параметры и принято называть атрибутами.

Отношения в качестве элементов модели выражают наличие некоторых взаимных связей между существами или, точнее, между объектами, представленными существами. Каждое отношение соединяет два, а иногда и большее число существ (фиг. 1). Эти соединения указываются дугами графа (ЭР-диаграммы). Число присоединенных к данному отношению существ называется его рангом. Ранг всех отношений, представленных на фиг. 1, равен двум. Такие отношения называются бинарными.

Отношения между существами определяют общую или глобальную структуру модели.

Атрибуты составляют информационную характеристику существ. Каждый атрибут данного существа понимается как переменная, которая при каждом объекте соответствующего типа принимает определенное значение из множества значений этого атрибута.

Помимо этого можно (и нужно) говорить и об атрибутах отношений. Такие атрибуты выражают "меру" или "силу" связи между связанными с данными отношениями объектами или т.п. Эти атрибуты должны быть присвоены именно соответствующим отношениям, а не связанным с ним существам, как это часто делается [2, 4].

Атрибуты существа (отношения) определяют структуру модели "в окрестности" данного существа (отношения). Эту структуру можно охарактеризовать как локальную структуру модели, связанную с данным существом (отношением).

Формальное описание элементов ЭР-диаграммы может быть представлено в теоретико-множественных терминах.

Например, если обозначить существа ОПЕР, ПРЕД и отношения ПРИЛ, ВКЛ (фиг. 1) буквами A, B, R, S соответственно, то соответствующий фрагмент описывается в теоретико-множественной интерпретации следующим образом:

$A \subseteq \Omega$ - задано множество операций A,

$B \subseteq \Omega$ - задано множество операций предметов B,

$R \subseteq A \times B$ - между множествами A и B задано отношение R,

$S \subseteq B \times B$ - в множестве B задано отношение S,

где Ω - множество всех объектов данной предметной области.

Формальное описание фрагмента модели по сути дела является описанием его глобальной структуры. Для полного описания модели требуется дополнение глобального описания описаниями локальной структуры.

Если существо или отношение E характеризуется атрибутами $a_1, \dots, a_k$ ($k \geq 1$), то это может быть представлено схемой $E(a_1, \dots, a_k)$, в конкретном случае, например, схемой ПРЕД(шифр, вес, цена). Такая схема и выражает локальную структуру модели, связанную с данным существом (отношением). Если нужно указать наличие у существа E атрибутов, но не важно, каких именно, то можно писать просто $E(\dots)$.

Описания глобальной и локальной структуры модели легко объединяются в одно, общее описание модели. Для рассмотренного фрагмента модели оно выглядит так:

$$\begin{aligned} A(\dots) &\in \Omega, & R(\dots) &\in A \times B, \\ B(\dots) &\in \Omega, & S(\dots) &\in B \times B. \end{aligned} \quad (I)$$

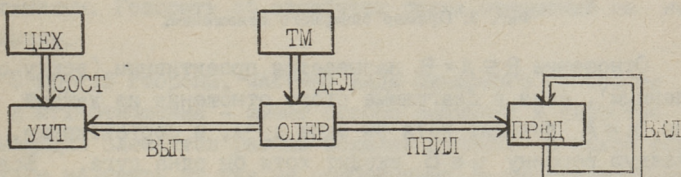
Модель концептуальной структуры предметной области данных, представленная в виде ЭР-диаграммы, характеризуется выделением в ней двух, качественно различных видов основных составных элементов — существ и отношений ранга $n \geq 2$. Это хорошо согласуется с интуитивным представлением о предметной области как о совокупности взаимосвязанных объектов. Таковую модель можно назвать биконцептуальной.

Однако с точки зрения формального описания (I) разница между существами и отношениями несущественна. Оба они представляются множествами и могут быть охарактеризованы атрибутами. С этой точки зрения отношения интерпретируются как существа. С другой стороны, опираясь на схему описания (I), легко заметить, что существа могут быть интерпретированы как отношения первого ранга — как унарные отношения. При этом понятие n -арного отношения естественным образом обобщается на случай $n \geq 1$, а разница между существами и отношениями становится уже не качественной, а количественной.

Проблема "интеграции" понятий существа и отношения и ее решение имеет в данной работе центральное значение. Но до того, как приступить к рассмотрению и решению этой проблемы, необходимо более подробно проанализировать некоторые свойства бинарных отношений.

ЭР-сеть и бинарные отношения. Важным частным видом концептуальных моделей являются модели, в которых встречаются только бинарные отношения. К их числу относятся, например, т.н. бинарные логические ассоциации [2].

Такие модели могут быть представлены в более простой форме, чем ЭР-диаграмма, а именно - в виде ориентированного графа, вершины которого представляют существа, а дуги - бинарные отношения между ними. Мы будем называть этот граф ЭР-сетью. Пример такой сети, соответствующей вышеприведенной ЭР-диаграмме (фиг. 1), дан на фиг. 2.



Фиг. 2. Пример ЭР-сети.

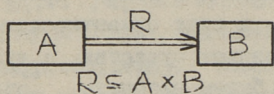
ЭР-сеть, вообще говоря, может иметь произвольную структуру. В частности, в ней могут существовать контуры (ориентированные циклы), в том числе и петли (фиг. 2). Но она может иметь и структуру более частного вида. Все это зависит от свойств отношений, представленных дугами графа. Поэтому изучение глобальной структуры ЭР-сети (и других концептуальных моделей) сводится к изучению бинарных отношений.

Если A и B - два множества и $R \subseteq A \times B$, то R называется бинарным отношением (соответствием) между множествами A и B , а последние - областями отправления и прибытия отношения R соответственно. Отношение R называется взаимным, если $A \cap B = \emptyset$, и собственным, если $A = B$. Отношение, обратное отношению R , обозначается через R^{-1} ($R^{-1} \subseteq B \times A$).

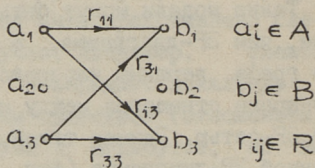
Отношения представляются двудольными графами. Мы различаем два вида таких графов - фрейм и диаграмму. Первый из них дает общее, второй - детальное представление об отношении (фиг. 3). Фрейм можно рассматривать как элемент ЭР-сети, диаграмму - как представление того, что скрывается за данным элементом сети.

Каждое отношение имеет определенную ориентацию, которая в общем случае является условной и служит только для

различения областей отправления и прибытия (фиг. 3). Она необязательно указывает наличие только односторонней связи между соответствующими множествами. Теоретически наличие отношения R между множествами A и B равносильно наличию отношения R^{-1} между B и A , и наоборот.



фрейм



диаграмма

Фиг. 3. Пример бинарного отношения.

Отношение $R \subseteq A \times B$ называется проективным (всюду определенным), если в диаграмме этого отношения из каждой вершины $x \in A$ выходит хотя бы одна дуга, и сюръективным, если в каждую вершину $y \in B$ входит хотя бы одна дуга. Если R проективно, то R^{-1} сюръективно, и наоборот.

В дальнейшем, если это особо не указывается, имеются в виду только проективные и сюръективные отношения.

В общем случае, оба отношения — R и его обратное отношение R^{-1} — являются многозначными отношениями. Такие отношения называются много-многозначными ($M:M^{-1}$) отношениями. Их можно назвать и ассоциативными отношениями. В частном случае отношение R может быть одно-многозначным ($I:M^{-1}$), много-однозначным ($M:I^{-1}$) или одно-однозначным ($I:I^{-1}$) отношением.

Отношение $R \subseteq A \times B$ называется $I:M^{-1}$ - или партитивным отношением, если в диаграмме отношения в каждую вершину $y \in B$ входит не более одной дуги и $M:I^{-1}$ - или функциональным отношением, если из каждой вершины $x \in A$ выходит не более одной дуги. Партитивное и функциональное отношение называется инъективным. Если R проективно, то R^{-1} функционально, и наоборот.

В концептуальных моделях (ЭР-сетях) встречаются все указанные виды бинарных отношений ($I:I^{-1}$ -отношение встречается очень редко). Например, в модели на фиг. 2 отношения ПРИЛ и ВКЛ, очевидно, являются ассоциативными, СОСТ и ДЕЛ — партитивными, а ВЫП — функциональным. Интуитивно только

ассоциативные отношения воспринимаются как "настоящие" бинарные отношения. В них больше всего высказывается специфика данной предметной области. Каждое такое отношение в качестве элемента концептуальной модели имеет специфическую, свойственную ему семантику. В частности, только в связи с этими отношениями имеет смысл говорить об атрибутах отношений.

В противоположность этому, частные виды бинарного отношения — $I:M$ - и $M:I$ -отношения — имеют везде примерно одинаковую семантику. Эту семантику можно выразить словами "состоит из" и "входит в состав" или тому подобными словосочетаниями. Говорить об атрибутах таких отношений не имеет смысла.

С другой стороны, частные виды бинарного отношения — $I:M$ - и $M:I$ -отношения — имеют особое значение в качестве связывающих элементов разных моделей данных. В такой роли мы будем называть их уже не отношениями, а соответствиями и обозначать буквами π и φ соответственно. При этом наличие $I:M$ -соответствия π между множествами A и B будем выражать схемой $\pi: A \dashv B$, а наличие $M:I$ -соответствия φ между теми же множествами — традиционной схемой: $\varphi: A \rightarrow B$ ($\varphi = \pi^{-1}$). Ориентация таких соответствий имеет особый смысл — в случае $I:M$ -соответствия она указывает, в каком направлении данное соответствие является многозначным, а в случае $M:I$ -соответствия — в каком направлении оно является однозначным.

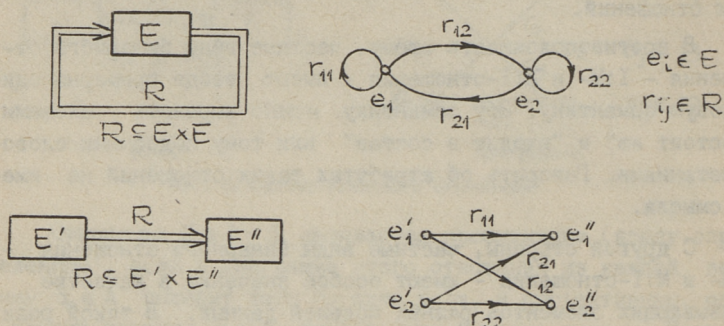
Наконец, $I:I$ -соответствия имеют в концептуальных моделях еще более узкий смысл. Они указывают обычно на то, что связанные с ними существа представляют одни и те же объекты в двух разных ролях. Семантику их можно выразить словами "представляет", "является ролью" или т.п.

Если $R \subseteq E \times E$, то говорят, что R является собственным отношением или отношением в множестве E . Последнее называется полем этого отношения. Если различать две роли поля — роль быть областью отправления (E') и роль быть областью прибытия (E'') данного отношения, то собственное отношение превращается во взаимное отношение $R \subseteq E' \times E''$.

Любое собственное отношение представимо в виде "обычного" ориентированного графа, причем здесь опять можно говорить о фрейме и диаграмме отношения (фиг. 4). С учетом

ролей поля собственного отношения "обычному" графу такого отношения всегда можно сопоставить эквивалентный ему двудольный граф, и, наоборот (фиг. 4).

Собственные отношения могут обладать теми же свойствами, что и взаимные отношения (проективностью, сюръективностью, партитивностью, функциональностью). Но помимо это-



Фиг. 4. Пример собственного бинарного отношения.

го, они могут обладать и рядом специфических свойств (рефлексивностью, симметричностью, транзитивностью и др.), о которых нельзя говорить в связи с взаимными отношениями. Из этих свойств в нашем анализе особое значение имеют только два - цикличность и ацикличность.

Собственное отношение $R \subseteq E \times E$ называется циклическим, если в его диаграмме содержится хотя бы один ориентированный цикл (контур) (без учета петель). В противном случае оно называется ациклическим.



Фиг. 5. Частные виды собственного бинарного отношения.

В зависимости от наличия у собственного отношения свойства цикличности или ацикличности, а также свойств паритивности и функциональности, можно выделить ряд важных частных видов собственного отношения. Примеры диаграмм их приведены на фиг. 5. Каждая из этих диаграмм имеет определенную, свойственную данному частному виду отношения структуру. Мы присваиваем этим структурам специфические имена - сеть, решетка, дерево и др. (фиг. 5).

Необходимо отметить, что термин "решетка" используется нами здесь и в дальнейшем в более широком смысле, чем это принято в математике (в теории решеток).

В концептуальных моделях (ЭР-сетях) встречаются все указанные виды собственных бинарных отношений, чаще всего такие, диаграмма которых представляет собой сеть (общий вид), решетку или дерево. В частности, собственное отношение ВКЛ, указанное в ЭР-сети на фиг. 2, очевидно, имеет диаграмму, представляющую собой решетку.

Регуляризация отношений. Бинарные отношения составляют лишь самый простой вид n -арных отношений. Помимо их, в состав концептуальной модели могут войти и отношения более высокого ранга. Например, если технологические операции (ОПЕР), прилагаемые к предметам обработки (ПРЕД), могут быть выполнены на разных технологических установках (УСТ), то это говорит о наличии некоторого отношения третьего порядка (тернарного отношения), скажем, ОПР (обработка) между этими сущностями: $ОПР \subseteq УСТ \times ОПЕР \times ПРЕД$.

В связи с этим возникает необходимость единой, более общей теории (конечных) отношений - теории регулярных отношений.

Регуляризация отношений имеет такую же цель, как и структуризация программ - унификацию компонентов общей структуры.

Регуляризация отношений связана с изменением форм представления отношений, обобщением их семантики и самого понятия отношения. В связи с этим необходимо различать отношения в "старом", традиционном смысле и отношения в "новом", определяемом ниже смысле. Мы будем называть первые условно декартовыми отношениями (Д-отношениями), вторые - регуляризованными или регулярными отношениями (Р-отношениями).

Для четкого различения понятий Д- и Р-отношения ниже приводятся формальные определения этих понятий.

Д-отношение: Пусть $R, E_1, \dots, E_n$ суть некоторые множества ($n \geq 2$). Множество R называется Д-отношением ранга n или n -арным Д-отношением между множествами $E_1, \dots, E_n$, а последние — его доменами, если и только если это множество R является подмножеством декартова произведения $E_1 \times \dots \times E_n$. В таком случае пишут $R \subseteq E_1 \times \dots \times E_n$.

Р-отношение: 1. Пусть R есть некоторое множество. Множество R называется Р-отношением ранга 0 или нольарным Р-отношением, если и только если это множество R содержит только один элемент. В таком случае пишут $R := \theta$.

2. Пусть $R, E_1, \dots, E_n$ суть некоторые множества ($n \geq 1$). Множество R называется Р-отношением ранга n или n -арным Р-отношением между множествами $E_1, \dots, E_n$, а последние — его доменами, если и только если для каждого E_i ($1 \leq i \leq n$) существует некоторое сюръективное (но не обязательно проективное) $I: M$ -соответствие π_i такое, что $\pi_i: E_i \rightarrow R$, или, что то же самое, что для каждого E_i существует некоторое проективное (но не обязательно сюръективное) $M: I$ -соответствие φ_i такое, что $\varphi_i: R \rightarrow E_i$ ($\varphi_i = \pi_i^{-1}$). В таком случае пишут $R \subseteq E_1$, если $n = 1$, и $R \subseteq E_1 \times \dots \times E_n$, если $n \geq 2$.

Основное отличие Р-отношения от Д-отношения заключается в том, что его ранг имеет более широкую область значений ($n \geq 0$), и в том, что оно имеет такую же семантическую категорию, как и его домены.

Однако под n -арным Р-отношением неявно понимается не только само множество R , но и связанные с ним (и упомянутые в определении) n $I: M$ -соответствий $\pi_i: E_i \rightarrow R$. В этой связи, допуская определенную неоднозначность обозначений, мы будем называть множество R собственным множеством данного Р-отношения R , а соответствия π_i — собственными связями этого Р-отношения. Эти связи считаются неявно заданными.

Необходимо еще обратить внимание на различие в применении знаков " $\subseteq$ " и " $\subset$ " в определениях Д- и Р-отношения. Оба эти знака выступают в данном контексте как символы некоторых метаотношений.

Все рассмотренные в предыдущих разделах данной работы отношения были представлены как Д-отношения. Каждому такому отношению может быть сопоставлено соответствующее ему Р-отношение. Обратное, вообще говоря, неверно, даже при $n \geq 2$. Однако, если понятием Д-отношения охватывать и мультиотношения, т.е. отношения, диаграмма которых представляет собой мультиграф, то и любому Р-отношению ранга $n \geq 2$ уже может быть сопоставлено соответствующее ему Д-отношение.

Ниже более подробно описываются формы представления и семантика Р-отношений в зависимости от ранга. При этом, если это не вызывает путаницы, Р-отношение называется просто отношением.

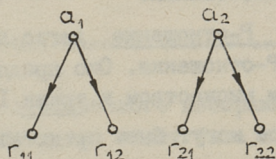
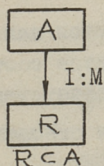
Нольарное Р-отношение имеет в данной теории особое место. Фрейм и диаграмма такого отношения являются тривиальными и выглядят так, как это изображено на фиг. 6. Запись $R := \emptyset$ означает только то, что R является нольарным отношением.



$$\begin{matrix} o \\ r \in R \end{matrix}$$

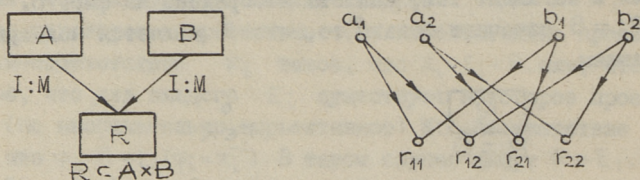
Фиг. 6. Пример нольарного Р-отношения.

Унарное Р-отношение, также как и нольарное Р-отношение, не имеет прямого аналога в виде Д-отношения. Унарное отношение $R \subset A$ выражает то, что объекты типа A определенным образом делятся на объекты типа R или что " A состоит из R " и т.д. Оно представляется соответствующим собственным множеством и одним I:M-соответствием (фиг. 7).



Фиг. 7. Пример унарного Р-отношения.

Бинарное Р-отношение (фиг. 8) выражает в другой форме то же самое, что и соответствующее ему (ассоциативное) бинарное Д-отношение (фиг. 3) – наличие некоторой взаимосвязи (R) между элементами двух множеств A и B. Но теперь эта связь представляется при помощи "объектов связи" в виде элементов собственного множества данного Р-отношения и двух I:M-соответствий (фиг. 8). Поэтому Р-отношение может быть рассмотрено как декомпонированное представление соответствующего Д-отношения. Такое декомпонированное представление используется, в частности, при отображении бинарных отношений в базах данных сетевой структуры CODASYL [5]. С теоретической точки зрения более интересным является то, что диаграмма бинарного Р-отношения как граф (фиг. 8) является т.н. тотальным графом графа, представляющего диаграмму соответствующего Д-отношения (фиг. 3).

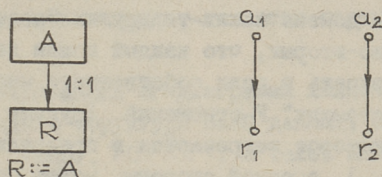


Фиг. 8. Пример бинарного Р-отношения.

Необходимо отметить, что если Д-отношение не является отношением общего вида (ассоциативным), а отношением частного вида (партитивным, функциональным или же инъективным), то в представлении соответствующего Р-отношения одна или даже обе собственные связи оказываются не I:M-соответствиями, а I:I-соответствиями.

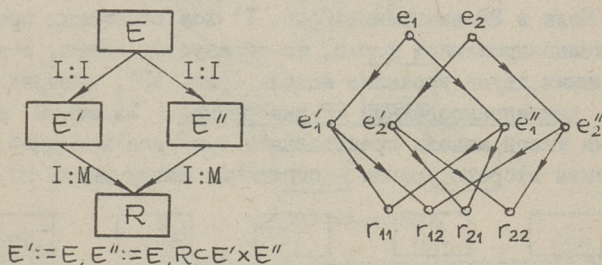
Тернарное Р-отношение легко понимается как обобщение бинарного Р-отношения. Оно представляется соответствующим собственным множеством и тремя I:M-соответствиями. Аналогичным образом могут быть представлены и отношения произвольного более высокого ранга.

Ролевое Р-отношение является частным видом унарного Р-отношения, собственная связь которого представляет собой 1:1-соответствие (фиг. 9). В таком случае вместо $R \subset A$ пишут $R := A$ (читается: R является ролевым отношением под A или, просто, R является ролью A). Необходимость в ролевых Р-отношениях возникает, например, при представлении собственных отношений.



Фиг. 9. Пример ролевого Р-отношения.

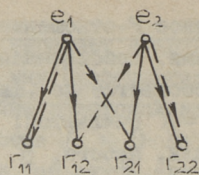
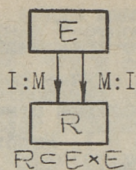
Собственное бинарное Р-отношение (фиг. 10) является регуляризованным образом соответствующего собственного Д-отношения (фиг. 4). Представление такого Р-отношения основывается на выделении двух ролей поля соответствующего Д-отношения (фиг. 10), хотя возможно его представление и на основе выделения только одной роли поля отношения.



Фиг. 10. Пример собственного бинарного Р-отношения.

В целях упрощения графического изображения собственного бинарного Р-отношения оно может быть изображено в условной, упрощенной форме так, как это показано на фиг. II.

Из принципов регуляризации отношений стало ясным, что собственные множества Р-отношений (R) и их домены ($A, B, \dots$) представляются одинаково - в виде множеств. Это значит, во-

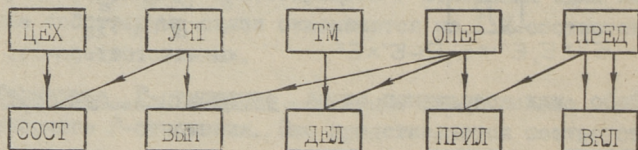


Фиг. 11. Условное представление собственного бинарного Р-отношения.

первых, что собственное множество любого Р-отношения может выступать в роли домена каких-то других "подчиненных" ему Р-отношений, и во-вторых, что каждый домен данного Р-отношения может выступать в роли собственного множества каких-то других "вышестоящих" Р-отношений. Другими словами, каждое множество, которое встречается в представлении Р-отношения $(R, A, B, \dots)$, с одной стороны, может быть интерпретировано как образ некоторого существа, а с другой стороны, как образ некоторого отношения между другими существами. В этом и заключается интеграция понятий существа и отношения.

ЭР-решетка и -дерево. Вернемся теперь к ЭР-диаграмме и -сети (фиг. 1, 2) с целью описания некоторых частных видов их. Но при этом мы будем рассматривать их пока в виде био-концептуальных моделей, имея в виду только возможность декомпонированного представления отношений ранга $n \geq 2$.

Если в ЭР-диаграмме (фиг. 1) все отношения представить в декомпонированной форме, то образуется новая, с явно выделенными двумя уровнями модель (фиг. 12), которая называется декомпонированной ЭР-диаграммой. Элементы первого уровня такой модели представляют существа исходной модели, элементы второго уровня - отношения между ними. Но в новой



Фиг. 12. Пример декомпонированной ЭР-диаграммы.

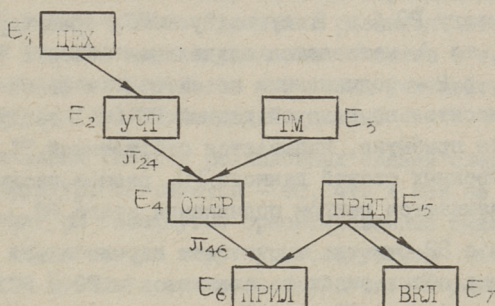
модели (фиг. 12) оба эти элемента рассматриваются как существа. Первые из них можно называть основными или базовыми существами, вторые - реляционными существами, образованными в результате декомпозиции отношений исходной

модели. В роли отношений декомпонированной модели выступают I:M-соответствия между основными и реляционными существами, определенные в результате декомпозиции отношений исходной модели. Некоторые из них могут оказаться и I:I-соответствиями.

Описанная декомпонированная модель (фиг. 12) является частным видом описываемой ниже модели - ЭР-решетки, которая сама, в свою очередь, является частным видом ЭР-сети.

Декомпонированная ЭР-диаграмма является концептуальным прообразом модели структуры базы данных, поддерживаемой многими СУБД (напр., TOTAL). В такой интерпретации элементам первого уровня модели соответствуют т.н. основные файлы базы, элементам второго уровня - т.н. файлы связи.

Декомпонированная ЭР-диаграмма в принципе описывается таким же образом, как и соответствующая исходная ЭР-диаграмма, т.е. по схеме (I).



Фиг. 13. Пример ЭР-решетки.

При переходе от обычной, недекомпонированной ЭР-диаграммы (фиг. I) к декомпонированной диаграмме (фиг. 12) мы не учитывали того, что некоторые отношения в исходной диаграмме являются отношениями частного вида I:M- или M:I-отношениями. Вполне логично оставить при таком переходе I:M-отношения неизменными, заменить M:I-отношения соответствующими обратными I:M-отношениями и представить в декомпонированной форме только M:M-отношения. В таком случае образуется новая модель (фиг. 13), которую мы называем ЭР-решеткой.

Под ЭР-решеткой, вообще говоря, мы понимаем ЭР-сеть, в которой все дуги представляют I:M-соответствия (некоторые из них, конечно, могут быть и I:I-соответствиями). Но вместе с наложением такого ограничения на семантику дуг ЭР-сети имеет место расширение роли семантики вершин сети. В новой модели - в ЭР-решетке (фиг. 13) - одна часть вершин (ЦЕХ, ТМ, ПРЕД) представляет исходные базисные существа, другая часть (ПРИЛ, ВКЛ) - реляционные существа (Р-отношения), а третья часть (УЧТ, ОПЕР) - и те и другие. В связи с этим целесообразно каким-то подходящим образом переименовать элементы модели.

Мы будем называть основные элементы (вершины) ЭР-решетки реляционными существами или объектами (PO), а элементы связи между ними (дуги графа) - реляционными связями (PC). Естественно предполагается, что за каждым PO скрывается целое множество абстрактных объектов соответствующего типа, а за каждой PC - некоторое I:M-соответствие между соответствующими множествами объектов.

Если между PO A и B существует PC, ориентированная от A к B, то A называется владельцем объекта B и связи между ними, а B - подчиненным объекта A и связи между ними. PC, относительно которой данный PO (B) выступает в роли области прибытия, называется собственной PC этого PO. Число собственных связей данного PO, равное числу его владельцев, называется рангом подчинения этого PO.

Говоря о ЭР-решетке мы в общем случае имеем в виду явное задание обоих видов ее элементов - PO и PC. Этому соответствует описание ЭР-решетки по следующей схеме:

$$R_i(\dots) \in \Omega, \quad \pi_{ij}: R_i \rightarrow R_j, \quad (2)$$

где через R_i обозначены PO, а через π_{ij} - PC между ними.

ЭР-решетка в таком представлении является концептуальным прообразом весьма важной модели структуры базы данных - сетевой модели CODASYL [5]. В такой трактовке PO-ам (вершинам ЭР-решетки) соответствуют типы записей базы, а PC-ям (дугам графа) - типы т.н. наборов (цепей записей). При этом первое соотношение в схеме (2) соответствует описанию записей типа R_i , а второе - описанию наборов типа π_{ij} .

Однако такое сопоставление ЭР-решетки и сетевой модели является в определенном смысле условным. Условность связана с тем, что в сетевой модели CODASYL встречаются различные структурные особенности (напр., возможность включения повторяющихся групп данных в состав записей, возможность построения наборов с разнотипными записями и др.), которые не учитываются в ЭР-решетке. Но на наш взгляд все они являются теоретически необоснованными усложнениями сетевой модели, которые мы рассматриваем как структурные аномалии этой модели.

ЭР-решетка обладает определенными общими структурными свойствами, которые определяются свойствами входящих в ее состав реляционных связей.

Основное свойство РС — это ее партитивность. Но помимо этого, если рассматривать только структурные свойства модели, то каждую РС нужно считать еще проективной и сюръективной. В этих двух свойствах отражается возможное "установившееся состояние" связи между соответствующими множествами объектов. С точки зрения "динамики развития" этой связи и всей модели (прибавление и удаление объектов и т.п.) РС уже не может быть проективной, но это можно учитывать при анализе структурных свойств модели в целом.

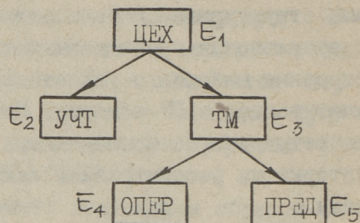
Из указанных свойств реляционных связей следует ацикличность структуры ЭР-решетки — то, что в ней не могут содержаться контуры (замкнутые ориентированные циклы). В свою очередь, из ацикличности структуры следует ее частичная упорядоченность. Поэтому можно сказать, что ЭР-решетка является частично упорядоченной моделью.

Частным видом ЭР-решетки является модель, в которой все РО являются объектами первого ранга (фиг. I4). Такую модель можно назвать ЭР-деревом. Но следует отметить, что модель такой структуры имеет более узкую область применения; то, что может быть представлено при помощи ЭР-решетки, в общем случае не может быть представлено при помощи ЭР-дерева (модель на фиг. I4 не эквивалентна модели на фиг. I3). Обратное, конечно, возможно.

ЭР-дерево является концептуальным прообразом т.н. иерархической модели базы данных, поддерживаемой, например, системой IMS. В такой интерпретации РО обозначают типы сег-

ментов базы, а РС - I:M-соответствия, определенные некоторой системой ссылок. Это соответствует неявному заданию РС.

Неявное задание реляционных связей возможно не только в случае ЭР-дерева, но и в случае ЭР-решетки. Такое представление РС осуществляется, например, при помощи соответствующих ключевых атрибутов. Такая ЭР-решетка является концептуальным преобразованием реляционной модели Кодда. В этой интерпретации реляционные объекты соответствуют отношениям (в смысле модели Кодда), а реляционные связи - т.н. интерреляционным связям.



Фиг. 14. Пример ЭР-дерева.

ЭР-решетка (фиг. 13) интерпретируемая как реляционная модель Кодда, описывается следующей схемой:

$$\begin{aligned}
 E_1(\dots) &\in \Omega & E_5(\dots) &\in \Omega \\
 E_2(\dots, k_{21}) &\in \Omega & E_6(\dots, k_{64}, k_{65}) &\in \Omega \\
 E_3(\dots) &\in \Omega & E_7(\dots, k'_{75}, k''_{75}) &\in \Omega, \\
 E_4(\dots, k_{42}, k_{43}) &\in \Omega
 \end{aligned} \quad (3)$$

где k_{ij} - ключевой атрибут РО-а E_j в качестве атрибута РО-а E_i . Отметим, что согласно принципам построения реляционной модели Кодда все интерреляционные связи в ней по существу обеспечивают только одностороннюю связь между отношениями (РО-ами) - снизу вверх.

Квазирегулярная реляционная решетка. Будем теперь рассматривать ЭР-решетку в свете приведенной выше теории регуляризации отношений. Ясно, что в данной ЭР-решетке (фиг. 13) РО УЧТ может быть рассмотрен как унарное Р-отношение под РО ЦЕХ, РО ОПЕР - как бинарное Р-отношение под объектами УЧТ и ТМ и т.д. Однако три РО - ЦЕХ, ТМ и ПРЕД - уже не так явно могут быть интерпретированы в виде Р-отношений. Они, очевидно, не являются нормальными Р-отношениями, так как не являются одноэлементными множествами. По сути дела каждый из них представляет определенный класс объектов данной предметной области. В этом смысле они вы-

ражают то же самое, что и унарные Р-отношения. Но в данном случае они не имеют явно указанного владельца (домена). Поэтому мы будем их называть квазиунарными Р-отношениями. Договоримся записывать то, что R является квазиунарным Р-отношением, при помощи записи $R := \Lambda$.

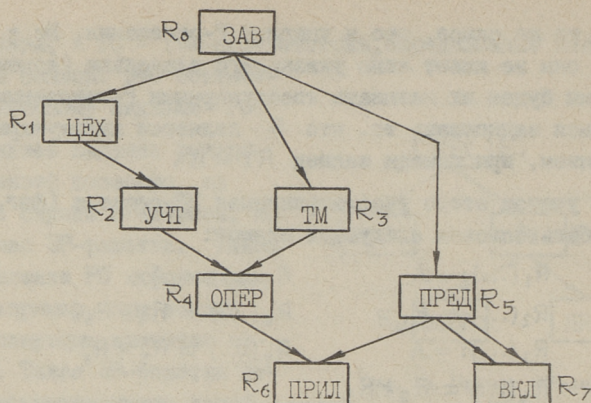
С учетом этого рассматриваемая ЭР-решетка (фиг. I3) может быть описана следующей схемой:

$$\begin{aligned} R_1(\dots) &:= \Lambda & R_5(\dots) &:= \Lambda \\ R_2(\dots) &= R_1 & R_6(\dots) &= R_4 \times R_5 \\ R_3(\dots) &:= \Lambda & R_7(\dots) &= R_5' \times R_5'' \\ R_4(\dots) &= R_2 \times R_3 \end{aligned} \quad (4)$$

ЭР-решетка в такой интерпретации называется квазирегулярной реляционной решеткой.

Из схемы (4) ясно, что все рассмотренные выше модели, в том числе и "традиционные" модели баз данных – сетевая, иерархическая и реляционная – представляют собой квазирегулярную реляционную решетку. В свете такой интерпретации ясно и то, что сетевая и реляционная модель имеют одинаковую "моделирующую мощность", в то время как "моделирующая мощность" иерархической модели является существенно более низкой. При помощи последней модели могут быть представлены только квазиунарные и унарные Р-отношения; в то время как при помощи сетевой и реляционной модели могут быть представлены Р-отношения произвольного ранга (сравнение этих моделей в других аспектах выходит за рамки данной работы).

Регулярная реляционная решетка. Квазирегулярная реляционная решетка (фиг. I4) станет логически завершенной, если в ее состав включить дополнительный РО R_0 так, чтобы он стал владельцем всех тех РО модели, которые до этого владельца не имели, т.е. тех РО, которые выше были объявлены квазиунарными Р-отношениями (фиг. I5). В результате этого последние станут "обычными" унарными Р-отношениями. Кроме того, предполагается, что этот, дополнительно введенный элемент модели R_0 является образом одноэлементного множества, единственный элемент которого представляет всю данную предметную область в целом. В данном случае ему присвоено имя ЗАВ (завод) (фиг. I5). Ясно, что этот РО может



Фиг. 15. Пример регулярной реляционной решетки.

быть рассмотрен как нольарное Р-отношение. Он является наибольшим элементом структуры модели и называется в этой роли также и корнем модели.

Дополненная указанным образом модель и называется нами регулярной реляционной решеткой или, просто, реляционной решеткой (Р-решеткой). Ее логическая завершенность относительно квазирегулярной модели явно высказывается в ее описании, которое в данном случае (фиг. 15) выражается схемой:

$$\begin{array}{ll}
 R_0(\dots) := \theta & R_4(\dots) \subset R_2 \times R_3 \\
 R_1(\dots) \subset R_0 & R_5(\dots) \subset R_0 \\
 R_2(\dots) \subset R_1 & R_6(\dots) \subset R_4 \times R_5 \\
 R_3(\dots) \subset R_0 & R_7(\dots) \subset R'_5 \times R''_5
 \end{array} \quad (5)$$

Регулярная (и квазирегулярная) решетка может быть рассмотрена одновременно как ЭР-решетка, состоящая из элементов двух видов - РО и РС- и как Р-решетка, состоящая из элементов одного вида - Р-отношений. В качестве ЭР-решетки она является биконцептуальной моделью, которая определяет некоторую упорядоченную базисную структуру модели. В качестве Р-решетки она является униконцептуальной моделью, определяющей некоторую интерпретационную суперструктуру над данной базисной структурой.

Базисная структура. Оба вида элементов базисной структуры рассматриваются как образы множеств элементов определенного вида. О представлении РО-ов в виде множеств мы уже говорили. Наряду с ними РС-и также могут быть представлены

в виде множеств. Но в качестве элементов РС-и (I:M-соответствия) целесообразно рассматривать не пары элементов владельца и подчиненного данной связи, как это обычно делается, а совокупности таких пар, связанных с одним элементом владельца, или, образно говоря, "пучки" дуг диаграммы РС-и, выходящие из одной и той же вершины области определения этой связи. Назовем такие элементы субъюнциями. Тем самым, каждая РС рассматривается как образ множества (типа) субъюнций, а каждая субъюнция в качестве элемента РС-и – как совокупность элементов, состоящая из одного элемента типа владельца данной связи, и ноля, одной или нескольких элементов типа подчиненного этой связи.

Такое биконцептуальное представление реляционной решетки является основой для ее интерпретации в качестве модели базы данных реляционно-решетчатой структуры [6]. При представлении реляционной решетки в виде униканцептуальной модели ее элементами являются Р-отношения, определенные реляционными объектами и их собственными связями (субъюнциями). В такой интерпретации Р-решетка выступает как логически завершенная и семантически полная метамодель концептуального моделирования предметной области данных. Логическая завершенность этой модели заключается в ее униканцептуальности и в возможности представления Р-отношений всех рангов $n \geq 0$, а также в рекурсивном принципе построения Р-решетки, выражающемся в возможности представления отношений между отношениями. Ее семантическая полнота связана с широким спектром возможностей интерпретации элементов модели – корень (нольарное Р-отношение) представляет предметную область в целом, унарные Р-отношения, непосредственно подчинены корню, – основные классы объектов данной предметной области, а все остальные подчиненные им Р-отношения – отношения (связи) между этими классами или другими отношениями (фиг. 15). При этом среди "остальных" могут быть и унарные Р-отношения, которые в таком случае представляют различные подмножества классов объектов или отношений. Ввиду явно выраженного реляционного характера элементов Р-решетки, концептуальное моделирование, основанное на применении этой метамодели, может быть охарактеризовано как реляционное моделирование.

В заключение отметим, что развитие теории Р-решеток направлено не только на усовершенствование методов концептуального моделирования предметной области данных и проектирования баз данных [7, 8], но и на усовершенствование программирования задач обработки данных на базе использования более агрегированного представления обрабатываемых данных [9, 10].

Л и т е р а т у р а

1. ANSI/X3/SPARC DBMS Study Interim Report, 75-02-08.
2. Bracchi G., Paolini P., Pelagatti G. Binary logical associations in data modelling. Modelling in data base management systems. - Proceedings of the IFIP Working Conference on Modelling in DBMS. North-Holland Publishing Company, 1976, p. 125-141.
3. Chen P.P.S. The entity-relationship model - Toward a unified view of data. - ACM Transactions on Database Systems, 1, 1, March 1976, p. 9-36.
4. Davenport R.A. Data analysis for database design. - The Australian Computer Journal, 1978, vol. 10, N 4, p. 122-137.
5. Дейт К. Введение в системы баз данных. М., Наука, 1980, 463 с.
6. Вассиль М.Х. Реляционно-решетчатая структура базы данных. См. наст. сб., с. 67.
7. Роталу Э.П. Реляционно-решетчатая концептуальная модель учебного процесса вуза. - Сб. Кибернетика и вуз. Томский политехнический ин-т, 1980, вып. 15, с. 144-152.
8. Крахт В.А., Роталу Э.Н. Проектирование баз данных на основе реляционно-решетчатой концептуальной модели предметной области. - Управляющие системы и машины. 1981, 4, с. 22-28.
9. Крахт В.А. Логическое конструирование программ обработки данных реляционно-решетчатой структуры. См. наст. сб. с. 29.
10. Роталу Э.П. Разработка систем обработки данных с базой данных реляционно-решетчатой структуры. См. наст. сб., с. 99.

W. Kracht

From the Conceptual Model of
Problem Area to Database Structure

Summary

Representation of the conceptual model of problem area by means of entity-relationship diagrams and formalization problems of the model are discussed. Relations of various kinds as the elements of the conceptual model are distinguished. The concepts of regular relation and relational latticework are introduced and explained. The latter is represented simultaneously as a tool for conceptual modelling and as a database structure model. Possibilities for interpretation of all well-known database structure models in terms of the relational latticework model are shown.

ЛОГИЧЕСКОЕ КОНСТРУИРОВАНИЕ ПРОГРАММ ОБРАБОТКИ
ДАННЫХ РЕЛЯЦИОННО-РЕШЕТЧАТОЙ СТРУКТУРЫ

Введение. Разработка прикладных систем обработки информации со сложной интегрированной базой данных (БД) является трудоемкой и сложной задачей. Основным объемом этой работы связан с разработкой прикладных программ обработки данных, хранимых в базе, основные трудности – с исключением, выявлением и исправлением различных логических (алгоритмических) ошибок в программах. Чем сложнее структура обрабатываемых данных, тем труднее решаются и эти проблемы. Поэтому весьма актуальным является усовершенствование логических основ программирования, логических средств систем программирования и методов логического конструирования программ обработки данных.

В данной работе приводятся результаты теоретического исследования, направленного на разработку строгих формальных основ логического конструирования программ обработки данных реляционно-решетчатой структуры [1]. Эти результаты получены на основе анализа и обобщения возможностей языка DAMAL по обработке данных указанной структуры [2]. Они представляются в виде теории Л-программ, под которыми понимаются программы, составленные на Л-языке, являющемся развитием и обобщением языка DAMAL.

Теория Л-программ или, точнее, некоторые ее основные положения, представляются в виде формализованной теории, при построении которой соблюдаются основные концепции представления формализованных теорий – концепции, которые в наиболее чистой форме высказываются, например, при построении основ математики [3].

Основное внимание в работе уделяется представлению соответствующей формальной системы и описывающей ее метатео-

рии. Содержательная теория Л-программ представляется только в той мере, в какой это необходимо для понимания существа дела. При этом фрагменты работы, касающиеся только содержательной теории Л-программ, отделяются от основного текста скобками //...//.

// Основные положения. Всякая Л-программа представляется как средство обработки данных, прилагаемое к обрабатываемым данным в качестве объекта обработки. Такое средство представляется в виде совокупности вложенных друг в друга или следующих друг за другом операторов Л-языка, операндами которых являются определенные структурные единицы обрабатываемых данных, называемые квантами данных. Эти кванты определяются описаниями данных и представляются в программе присвоенными им именами, называемыми терминами. Последние выступают в роли переменных Л-программ.

Каждое вхождение любого оператора Л-языка определяет некоторый процесс или подпроцесс обработки данных, называемый операцией (обработкой). В общем случае каждая такая операция состоит из одной или нескольких составляющих операций, которые сами также могут иметь составляющие. В ряде случаев в составе их указываются и логические условия, которые представляют условия приложения операторов, определяющих эти операции, к своим операндам. В дальнейшем они называются предикатами.

Понятия оператора, операции, предиката и кванта (терма) данных являются основными понятиями содержательной теории Л-программ. Обозначающие их термины - "оператор", "операция" и др. - используются и в метатеории для обозначения соответствующих формальных объектов. То же самое касается и других, вводимых в дальнейшем терминов содержательной теории Л-программ.

Обрабатываемые данные. Все обрабатываемые Л-программами данные делятся на внешние и внутренние. К внешним относятся данные, которые существуют вне самих обрабатываемых программ, а к внутренним данные, которые определяются в самих программах обработки.

В данной работе имеется в виду только один вид внешних данных - данные, составляющие БД рассматриваемой системы обработки информации. Они называются базовыми (Б)-дан-

ными. Предполагается, что они составляют БД реляционно-решетчатой структуры [1, 4]. При этом квантами обрабатываемых Б-данных считаются сегменты БД [4].

Под внутренними данными или программными (П-) данными понимаются только отдельные элементы данных или данные, составляющие двухуровневые иерархические структуры (векторы) данных, называемые коллекциями. Каждый отдельно определенный элемент или коллекция П-данных рассматривается как самостоятельный квант П-данных. //

Теория Л-программ предполагает описание всех обрабатываемых данных на Л-языке, или, точнее, на расширенном соответствующим образом Л-языке. При этом предусматривается, что описание Б-данных представляется в виде автономного описания образованной ими БД данной системы обработки информации, называемого схемой БД (Л-схемой), а описание П-данных - в виде декларации данных в самих Л-программах.

Описание данных. Описание элемента (П- или Б-) данных (с именем) x представляется, например, схемами:

x : integer, initially q ,

x : real, initially q ,

x : string (w), initially q ,

где q - константа, соответствующая типу значения элемента, w - целое. Описание коллекции $x(a_1, \dots, a_m)$ с элементами ($1 \leq i \leq n$) в качестве ее компонентов представляется схемой:

$$x: \text{collection of } a_1: \dots, \dots, a_m: \dots \text{ eod } x$$

(eod - end of data).

Описание Б-данных представляется как описание соответствующей реляционной решетки, состоящее из описаний отдельных (Р-)отношений [1], которым в БД соответствуют сегменты базы. Схемы их описания зависят от ранга отношения. Ноль-арное, унарное и n -арное ($n > 1$) отношения, которые в теории реляционных решеток [1] представляются соотношениями:

$$X(a_1, \dots, a_m) := \Theta, X(a_1, \dots, a_m) \subset Y, X(a_1, \dots, a_m) \subset Y_1 x \dots Y_n,$$

описываются (в виде сегментов БД) на Л-языке соответственно со следующими схемами (описания элементов, представляющих атрибуты отношений, опущены):

$\bar{x}$: collection of ... eod, root, eor $\bar{x}$,
 $\bar{x}$: collection of ... eod, related to $\bar{y}$, eor $\bar{x}$,
 $\bar{x}$: collection of ... eod, related to $\bar{y}_1$,

 and to $\bar{y}_n$, eor $\bar{x}$,

где $\bar{x}$ — имя типа сегментов, отображающих в БД отношение X (eor — end of relation).

В этих схемах имя $\bar{x}$ понимается как описатель типа представления сегментов БД.* Множество (существующих в БД в данный момент) сегментов типа $\bar{x}$ обозначается через $(\bar{x})$.

Вместе с этим вводятся три общие предиката, схемы представления и семантика которых выражаются следующим образом:

- (1) $(\bar{x}) = \theta$ — сегмент типа $\bar{x}$ является корневым сегментом БД,
- (2) $(\bar{x}) \subset (\bar{y})$ — сегменты типа $\bar{x}$ являются подчиненными сегментами сегментов типа $\bar{y}$ (независимо от ранга сегментов $\bar{x}$),
- (3) $\rightarrow (\bar{x})$ — сегменты типа $\bar{x}$ образуют точку входа в БД (являются сегментами непосредственного доступа).

Описание БД в целом (Л-схема) выглядит так:

database schema S: set of $\bar{x}_1: \dots, \dots, \bar{x}_k: \dots$, eos S,

где S — имя описываемой Л-схемы, а $\bar{x}_1: \dots$ — описания сегментов (eos — end of schema).

Л-схема с именем S, содержащая описание сегмента (отношения) $\bar{x}$, обозначается в дальнейшем через $\bar{S}(\bar{x})$.

Формальные объекты. В теории Л-программ выделяются следующие основные виды формальных объектов (символов) — операторы, встроенные и определяемые функции, номены и термы (переменные).

Операторы представляются ключевыми словами Л-языка или словосочетаниями таких слов (включая и определенные знаки препинания). Все эти слова заимствованы из английской лексики.

Встроенные функции также представляются ключевыми словами Л-языка. Но определяемые функции или, точнее, их имена представляются буквой F.

Номены представляются прописными латинскими буквами. Ими обозначаются имена Л-программ и Л-схем, а также имена составляющих компонентов Л-программы - операций.

Термы обозначаются строчными латинскими буквами. Но вхождение их в Л-программу связывается с определенными формами представления.

Метапеременные обозначаются греческими буквами, а также прописными латинскими буквами. Они явно объявляются в каждом конкретном случае.

Декларация термов. Термы Л-программы определяются их декларациями в программе.

Декларация термина П-данных совмещается с описанием соответствующего кванта данных и представляется схемой:

`declare x: . . . ,`

где $x: \dots$ - описание элемента или коллекции П-данных с именем x . Это имя объявляется П-термом, представляющим соответствующий квант П-данных. Если этот квант является коллекцией данных, то его компоненты (a_i) называются субтермами данного П-терма (предполагается, что имена компонентов разных квантов П-данных данной Л-программы отличаются друг от друга). Декларация термина x обозначается сокращенно через $D(x)$.

Декларация термина Б-данных, называемая его экспозицией, представляется схемой:

`expose x: twin of  $\bar{x}$ ,`

где x - определяемый терм, а $\bar{x}$ - претерм.

В этой схеме x понимается как образ нового, вторичного типа кванта Б-данных, сопоставляемого определенному Л-схемой первичному типу $\bar{x}$ (прообразу). Но в других операторах Л-языка он выступает уже в роли переменной, область определения которой является множество ($\bar{x}$). Экспозиция термина x обозначается сокращенно через $E(x)$.

Денотат и иннотат термина. С каждым термом в качестве переменной Л-программы связывается два значения - его дено-

тат и иннотат. Денотатное значение термина α выражается (в соответствующих формах представления) самим этим термом α , а его иннотатное значение – составным именем $\alpha.\alpha$. С каждым субтермом α данного термина α связывается только одно значение – его иннотат, которое в общем случае выражается составным именем $\alpha.\alpha$.

// Под денотатом Б-терма α понимается находящийся в "поле зрения" программы некоторый квант (сегмент) Б-данных соответствующего типа $\bar{\alpha}$. Он называется текущим сегментом типа $\bar{\alpha}$, представленным термом α . Предполагается, что в ходе выполнения программы в разные моменты времени в роли текущего сегмента могут выступать разные сегменты базы соответствующего типа, в чем и заключается изменение денотатного значения данного термина. Областью возможных значений термина α является множество всех, находящихся в данный момент в БД сегментов типа $\bar{\alpha}$ ($\bar{\alpha} \in (\bar{\alpha})$). В такой роли сегменты БД рассматриваются отвлеченными от их "информационного содержания".

Под иннотатом Б-терма (и его субтерма) понимается "информационное содержание" сегмента (его компонента), являющегося денотатом данного термина, т.е. вектор значений компонентов текущего сегмента данного типа (значение компонента).

Аналогичным образом определяются также и денотат и иннотат П-терма (и его субтерма). Но множество возможных значений $(\bar{\alpha})$ П-терма α является одноэлементным множеством, единственный элемент которого определяется декларацией этого термина. Поэтому каждый П-терм представляется как переменная с фиксированным в данной программе денотатным значением. //

Формы представления значения термов. Для различения денотатного и иннотатного значения термов предусматриваются различные формы представления их в Л-программе. В качестве таких форм выделяются четыре формы представления значения термов, называемые д-образом, и-формой, ф-формой и д-формой. Первая и последняя из них связаны с представлением денотатного значения Б-термов, вторая и третья – с представлением иннотатного значения П- и Б-термов.

Д-образ (денотатный образ) является в определенном смысле базисной формой представления Б-термов. Он встреча-

ется в предикатах и операциях, определяемых в дальнейшем. Для Π -термов д-образ не определяется.

Д-образ имеет четыре основных модификации, которые на $\mathcal{L}$ -языке выражаются следующими схемами (x, y - термы, причем в случае последних двух схем предполагается, что $(x) \subset (y)$):

- (1) $\text{root } x$, (2) $\text{any } x$, (3) $x \text{ in } y$, (4) $y \text{ of } x$

Каждая из этих схем отражает определенное утверждение о термах, встречающихся в этих схемах. Эти утверждения и их семантика выражаются соответственно следующим образом:

- (1) $x \in (\bar{x}), (x) = \theta$ - x представляет корневой сегмент (типа $\bar{x}$),
 (2) $x \in (\bar{x})$ - x представляет некоторый сегмент типа $\bar{x}$,
 (3) $x \in y(\bar{x})$ - x представляет сегмент типа $\bar{x}$, находящийся в подчинении текущего сегмента y (типа $\bar{y}$),
 (4) $y(\bar{x}) \ni x$ - y представляет сегмент (типа $\bar{y}$), в подчинении которого находится текущий сегмент x типа $\bar{x}$,
 где $y(\bar{x})$ - подмножество сегментов типа $\bar{x}$, подчиненных текущему сегменту y (типа $\bar{y}$).

И-форма (иннотатная форма) является схемой, по которой определяется представление иннотата Π - и B -термов и их субтермов в Λ -программе. Она определяется следующим образом.

- 1-2. Если t - B -терм, а s - его субтерм, то $t.t$ и $t.s$ - и-формы.
 3-6. Если t - Π -терм, а s - его субтерм, то $t.t$, t , $t.s$, s - и-формы.

Иннотатное значение любого терма и его субтерма зависит от денотата этого терма. Поэтому о и-форме нужно говорить как об и-форме, зависящей от соответствующего терма.

Ф-форма (функциональная форма) является схемой, на основе которой генерируются новые иннотатные значения термов и их субтермов. Она определяется следующим образом.

1. Всякая константа есть ф-форма.
 2. Всякая и-форма есть ф-форма.
 3. Всякая n -местная функция есть ф-форма.
 4. Если F - n -местная функция и α - ф-форма, то $F(\alpha)$ - ф-форма.
 5. Если F - n -местная функция и $\alpha_1, \dots, \alpha_n$ ($n > 1$) - ф-формы, то $F(\alpha_1, \dots, \alpha_n)$ - ф-форма.

Из определения ϕ -формы следует, что за исключением двух тривиальных случаев, соответствующих пунктам I и 3 определения, значение каждой конкретной ϕ -формы зависит — непосредственно или через функцию — от иннотатного значения по меньшей мере одного термина. Следовательно, о ϕ -форме также нужно говорить как о ϕ -форме, зависящей от какого-то термина или каких-то термов.

Функции, используемые в ϕ -формах, должны быть описаны средствами $\mathcal{L}$ -языка.

Такое описание можно, например, (в случае одноместной функции) представить так:

define F: integer function (integer).

Д-форма (денотатная форма) является схемой, по которой представляются определенные, предусмотренные на $\mathcal{L}$ -языке функции от денотатов термов. Они определяются следующим образом.

I. Если x Б-терм, то $\text{cur}(x)$ — д-форма.

// Эта форма представляет текущий сегмент типа x . //

2-5. Если x, y — Б-термы такие, что $(\bar{x}) \subset (\bar{y})$, то

- | | |
|------------------------------------|-------------------------------------|
| (2) $\text{cur}(x \text{ in } y)$ | (4) $\text{prior}(x \text{ in } y)$ |
| (3) $\text{next}(x \text{ in } y)$ | (5) $\text{head}(y \text{ of } x)$ |

— д-формы (**cur** — **current**).

// Схема (2) представляет текущий сегмент подмножества сегментов вторичного типа x , подчиненных текущему сегменту y , схемы (3) и (4) — соответственно последующий и предыдущий сегменты в этом подмножестве относительно его текущего, а схема (5) — сегмент типа y , в подчинении которого находится этот текущий сегмент. //

Д-формы явно зависят от термов, указанных в них.

Предикат. Предикаты делятся на простые (элементарные) и составные. В состав последних входят другие, составляющие предикаты. Они называются подпредикатами данного составного предиката, и, в свою очередь, также могут быть составными. Каждый предикат является подпредикатом самого себя.

Предикаты определяются следующим образом.

I. Логические константы **true** и **false** суть предикаты.

2. Всякий логический субтерм П- и Б-терма есть предикат.

3-5. Если α - и-форма и β - ф-форма, то $\alpha = \beta$, $\alpha < \beta$, $\alpha > \beta$ (и т.п.) - предикаты.

// Эти предикаты представляют арифметическое сравнение значения компонента данных, представленного и-формой, со значением указанной ф-формы. //

В приведенных схемах сравнения участвуют и- и ф-формы, из которых по крайней мере значение и-формы зависит от денотата некоторого термина. Поэтому истинностное значение определяемого предиката тоже зависит от денотата этого термина. Для выражения того, что значение предиката P зависит от (денотата) термина x пишут $P(x)$.

6. Если γ и δ - д-формы, то $\gamma = \delta$ - предикат.

// Представляется сравнение значений двух денотатных функций. //

Поскольку значение всякой д-формы зависит от денотата термов, представленных в этой форме, то истинностное значение предиката по данной схеме также зависит от (денотата) этих термов. Это выражается так, как это уже было сказано, т.е. в виде $P(x)$.

Все предикаты, определяемые по вышеуказанным пунктам 1-6, являются простыми предикатами.

7-9. Если P, Q - предикаты, то $\text{not } P$, $P \text{ and } Q$ и $P \text{ or } Q$ - предикаты.

// Эти предикаты представляют логические условия, образованные из условий P и Q при помощи логических союзов "не", "и" и "или". //

Предикаты, определяемые по этим схемам, являются уже составными предикатами. Предикаты P и Q являются их подпредикатами. Если какой-либо из них зависит от некоторого термина, то и определяемый предикат зависит от этого термина.

I0-I2. Если x, y - Б-термы такие, что $(\bar{x}) \subset (\bar{y})$, а $P(x, y)$ и $Q(x, y)$ - предикаты, то

(II) $\text{all}(x \text{ in } y)(P(x, y))$

(I2) $\text{ex}(x \text{ in } y)(P(x, y))$

(I3) $\text{ex}(y \text{ of } x)(Q(y, x))$

- предикаты.

// Предикат по схеме (I0) представляет логическое условие, которое считается истинным, если заданное условие P удовлетворяется при всех сегментах типа x , подчиненных текущему сегменту типа y . Предикат по схеме (II) представляет логическое условие, которое считается истинным, если заданное условие P удовлетворяется хотя бы при одном сегменте типа x , подчиненном текущему сегменту типа y .

Предикат по схеме (I2) выражает логическое условие, которое считается истинным, если условие Q удовлетворяется для сегмента типа y , являющегося владельцем текущего сегмента типа x . //

По схемам (I0)-(I2) определяются составные предикаты с кванторами, выраженными ключевыми словами Л-языка "all" (квантор всеобщности) и "ex" (квантор существования).

Указанный в схемах (I0)-(I2) квантор прилагается к терму, указанному первым в следующем за квантором д-образе, связывает этот терм (в общеизвестном смысле этого слова) и фиксирует его область действия. Под областью действия понимается участок Л-программы (по ее тексту), внутри которого денотат данного терма считается определенным при помощи данного квантора, а вне ее - неопределенным. В случае схем (I0) и (II) кванторы связывают терм x и фиксируют его область действия; в эту область входит только подпредикат P . В случае схемы (I2) квантор связывает терм y и фиксирует его область действия; в эту область входит только подпредикат Q . Терм, который связывается квантором, называется связанным (в данном, определяемом предикате), а терм, который квантором не связывается, свободным.

Операторы и операции. Операторы Л-языка делятся на процедурные или обрабатывающие и декларативные операторы. Первые являются основными операторами языка, вторые - вспомогательными. К последним относятся указанные выше операторы декларации П-терма, экспозиции Б-терма и определения функции. Процедурные операторы делятся на общие и специальные.

Общие операторы не имеют непосредственного отношения к обрабатываемым данным и формам их представления. Они только расширяют функциональные возможности языка по алгоритмизации процессов обработки.

Специальные операторы, наоборот, имеют непосредственное отношение к обрабатываемым данным, их структуре и формам представления. Они являются главными операторами основного подмножества Л-языка.

Специальные операторы делятся на и- и д-операторы, предназначенные для обработки иннотатного и денотатного значений термов программы соответственно. Первичными среди них являются д-операторы.

Операторы, приложенные (прямо или косвенно) к термам Л-программы в качестве их операндов, определяют составляющие единицы Л-программы - операции. В зависимости от используемых операторов различают и- и д-операции, а также операции общего вида, представляющие собой обобщение двух предыдущих.

Операторы описываются на основе схем операций, определенных ими.

И-операция. И-операции являются простыми (элементарными) операциями. Они определяются следующим образом.

1. Если α - и-форма и β - ф-форма, то $\text{set } \alpha = \beta$ - и-операция.

//Оператор set является оператором присваивания. //

2-3. Если $\alpha_1, \dots, \alpha_n$ ($n \geq 1$) - и-формы, то $\text{get } (\alpha_1, \dots, \alpha_n)$ и $\text{put } (\alpha_1, \dots, \alpha_n)$ - и-операции.

//Операторы get и put являются операторами ввода и вывода данных соответственно. //

В приведенных схемах и-операций в явном виде не указана их зависимость от термов. Но поскольку все и- и ф-формы зависят, как это было указано, от некоторых термов, то и-операции, в которых эти формы выступают в качестве операндов, тоже должны быть объявлены зависящими от этих термов. Для выражения того, что и-операция Q зависит от термина x , пишут $Q(x)$.

Д-операция. Д-операции определяются операторами, каждый из которых имеет только один операнд. Но они определяются этими операторами как составные операции (кроме одного, тривиального случая), в состав которых помимо оператора и его операнда входят другие, составляющие операции, а иногда

и предикаты. Эти составляющие операции называются подоперациями данной, определяемой операции. Они могут быть и-операциями (простыми) или д-операциями (составными).

Каждая д-операция считается подоперацией самой себя. Сам д-оператор, определяющий данную д-операцию, представляется в схеме операции уже не одним, а несколькими ключевыми словами Л-языка.

Операнды д-операторов представляются в большинстве случаев в виде вышеупомянутых д-образов, например, в виде x in y , в состав которых в общем случае входит не один, а два (или даже несколько) термов. Но действительным (собственным) операндом, к которому прилагается данный д-оператор, является из них только первый. При этом д-оператор определяет область действия этого термина и вместе с этим, как правило, связывает этот терм. В этом отношении д-операторы выступают подобно кванторам в предикатах. По существу они и представляют определенные операционные кванторы, приложенные к денотатному значению термов.

В связи с этим в д-операциях, так же как и в предикатах, различаются свободные и связанные термы.

Свободным является терм, от денотатного значения которого зависит данная и- или д-операция. Для выражения зависимости операции Q от термина x используется запись $Q(x)$.

Связанным является терм, к которому прилагается д-оператор, выступающий в роли операционного квантора (не все д-операторы являются такими). Определенная таким оператором операция уже не может зависеть от такого связанного термина. Это значит, что число ее свободных термов оказывается на единицу меньшим, чем число (возможных) свободных термов ее подоперации.

Связывание оператора с каким-то термом фиксирует и область действия этого термина. Под ней по-прежнему понимается участок программы (по ее тексту), внутри которого денотат данного термина считается определенным данным д-оператором, а вне ее — неопределенным. Но в случае д-операции область действия связанного термина не определяется так просто, как это делается в случае предикатов. Ниже она указывается в каждом конкретном случае отдельно.

Д-операции определяются следующим образом.

1. Пустой оператор `skip` есть д-операция.

2. Если M – номен, $D(x)$ – декларация П-терма x , $Q(x)$ – и-или д-операция, то

M : `open data`

$D(x) Q(x)$

M : `close data`

– д-операция.

// Эта операция объединяет описание П-данных и процесс обработки их в единое целое.//

Оператор, на основе которого определяется данная операция, связывает П-терм x и фиксирует его область действия. В эту область входит сама декларация $D(x)$, а также под-операция Q определяемой д-операции. Вместо одной декларации П-терма может стоять последовательность деклараций разных П-термов.

В последующих пунктах определения д-операции фраза "М-номен" опускается, но подразумевается.

3. Если $\bar{S}(x)$ – Л-схема, содержащая описание претерма $\bar{x}$ Б-терма x , $E(x)$ – экспозиция Б-терма x , Q – операция, то

M : `open database S`

$E(x) Q$

M : `close database S`

– д-операция.

// Эта операция связывает описание БД с программой.//

Оператор, определяющий данную операцию, не связывает и не фиксирует области действия ни одного Б-терма, несмотря на то, что в состав определяемой операции входит экспозиция Б-терма. Он просто фиксирует "участок" программы, внутри которого возможно связывание и фиксация области действия Б-терма x другими операторами. Этот участок можно назвать областью действия имени схемы S . В эту область входит под-операция Q определяемой д-операции. Вместо одной экспозиции Б-терма может стоять последовательность экспозиций разных Б-термов.

В дальнейшем Б-терм называется просто термом.

4. Если x - терм такой, что $x \in (\bar{x})$, и $(x) := \theta$, $P(x)$ - предикат, $Q(x)$, R , - и- или д-операции, то

```
M: if there exists root  $x$ 
    such that  $P(x)$ 
    then  $Q(x)$ 
    else  $R$ 
    end M
```

- д-операция.

// Эта операция представляет извлечение из БД корневого сегмента. Если такой сегмент существует в БД и удовлетворяет условию P , то он превращается в текущий сегмент типа x , после чего выполняется операция Q . В противном случае выполняется операция R . //

Оператор, определяющий данную операцию, связывает терм x и фиксирует его область действия. В эту область входят предикат P и операция Q , но не операция R .

5. Если x - терм такой, что $\rightarrow(x)$, key - и-форма, $P(x)$ - предикат, $Q(x)$, R - и- или д-операция, то

```
M: if there exists any  $x$  (key)
    such that  $P(x)$ 
    then  $Q(x)$ 
    else  $R$ 
    end M
```

- д-операция.

// Эта операция представляет извлечение из БД сегментов, для которых определен непосредственный доступ по ключу. //

Оператор, определяющий данную операцию, связывает терм x и фиксирует его область действия в точности таким же образом, как и оператор, определяющий предыдущую д-операцию. Предполагается, что определяемая д-операция входит в область действия некоторого П- или Б-терма, от которого зависит и-форма key .

6. Если x, y - термы такие, что $(\bar{x}) \subset (\bar{y})$, $P(x, y)$ - предикат, $Q(x, y)$, $R(y)$ - и- или д-операции, то

```
M: if there exists  $x$  in  $y$ 
    such that  $P(x, y)$ 
    then  $Q(x, y)$ 
```



```

else R(y)
end M

```

- д-операция.

// Эта операция представляет извлечение из БД одного подчиненного сегмента заданной субъюнкции по заданному условию. //

Оператор, определяющий данную операцию, связывает терм x (но не терм y) и фиксирует его область действия. Во всем остальном этот оператор не отличается от операторов предыдущих двух д-операций.

7. Если x, y - термы такие, что $(\bar{x}) \subset (\bar{y})$, $P(y, x)$ - предикат, $Q(y, x), R(x)$ и- или д-операции, то

```

M: if there exists y of x
    such that P(y, x)
    then Q(y, x)
    else R(x)
end M

```

- д-операция.

// Эта операция представляет извлечение из БД сегмента, являющегося владельцем заданной субъюнкции и удовлетворяющего заданному условию. //

Оператор, определяющий данную операцию, связывает терм y (но не терм x) и фиксирует его область действия. Во всем остальном этот оператор не отличается от операторов трех предыдущих д-операций.

8. Если x, y - термы такие, что $(\bar{x}) \subset (\bar{y})$, $P(x, y)$ - предикат, $Q(x, y)$ - и- или д-операция, то

```

M: for each x in y
    such that P(x, y)
    do Q(x, y)
end M

```

- д-операция.

// Эта операция представляет последовательное извлечение из БД всех подчиненных сегментов заданной субъюнкции, удовлетворяющих заданному условию. Если в БД в подчинении текущего сегмента типа y существуют сегменты типа x , удо-

влетворяющие условию P , то все они превращаются по очереди в текущий сегмент типа x , причем при каждом из них выполняется операция Q . //

Оператор, определяющий данную операцию, связывает терм x (но не терм y) и фиксирует его область действия. В эту область входят предикат P и подоперация Q .

9. Если $x, y_1, \dots, y_n$ - термы такие, что $(\bar{x}) \subset (\bar{y}_1), \dots, (\bar{x}) \subset (\bar{y}_n)$, $Q(x, y_1, \dots, y_n)$ - и- или д-операция, то

$M: \text{create } x \text{ in } y_1, \dots, y_n$

$Q(x, y_1, \dots, y_n)$

$M: \text{insert } x \text{ in } y_1, \dots, y_n$

- д-операция.

// Эта операция представляет процесс создания и прибавления в БД нового сегмента типа x , владельцами которого по описанию базы данных являются сегменты типа $y_1, \dots, y_n$. Процесс начинается созданием "пустого" сегмента типа x , продолжается определением его "информационного содержания" операцией Q и заканчивается прибавлением созданного сегмента в БД в подчинение указанных владельцев. //

Оператор, реализующий данную операцию, связывает терм x (но не термы $y_1, \dots, y_n$) и фиксирует область действия этого терма. В эту область входит подоперация Q . Но эта подоперация должна быть в данном случае такой, в которой никаким образом не используется денотатное значение терма x . В этом отношении данный оператор является особым.

10. Если $x, y_1, \dots, y_n$ - термы такие, что $(\bar{x}) \subset (\bar{y}_1), \dots, (\bar{x}) \subset (\bar{y}_n)$, $Q(x, y_1, \dots, y_n)$ - и- или д-операция, то

$M: \text{hold } x \text{ in } y_1, \dots, y_n$

$Q(x, y_1, \dots, y_n)$

$M: \text{replace } x \text{ in } y_1, \dots, y_n$

- д-операция ($M=M(x, y_1, \dots, y_n)$).

// Эта операция представляет процесс корректировки содержания существующего в БД сегмента типа x . Процесс начинается с фиксации корректируемого сегмента под его владельцами, продолжается созданием нового, скорректированного варианта сегмента операцией Q и заканчивается заменой в БД старого (фиксированного) сегмента его скорректированным вариантом. //

Оператор, реализующий данную операцию, не связывает и не фиксирует области действия ни одного термина.

II. Если $x, y_1, \dots, y_n$ - термины такие, что $(\bar{x}) \subset (\bar{y})$, ..., $(\bar{x}) \subset (\bar{y})$, а $Q(x, y_1, \dots, y_n)$ - и- или д-операция, то

M: hold x in $y_1, \dots, y_n$

$Q(x, y_1, \dots, y_n)$

M: delete x in $y_1, \dots, y_n$

- д-операция.

// Эта операция представляет процесс удаления из БД существующего в ней сегмента типа x . Процесс начинается фиксацией удаляемого сегмента под его владельцами, продолжается произвольным процессом Q и заканчивается самим удалением сегмента из БД. //

Оператор, реализующий данную операцию, не связывает формально ни одного термина. Но он исключает из области действия термина x участок программы, который остается "за" определяемой операцией. В этом смысле он также является особым оператором.

I2. Если x, y_1, y_2 - термины такие, что $(\bar{x}) \subset (\bar{y}_1)$, $(\bar{x}) \subset (\bar{y}_2)$ и $(\bar{y}_1) = (\bar{y}_2)$, $Q(x, y_1, y_2)$ - и- или д-операция, то

M: disjoint x in y

$Q(x, y_1, y_2)$

M: rejoin x in y_2

- д-операция.

// Эта операция представляет процесс изменения экземпляра владельца подчиненного сегмента заданной субъюнкции. Процесс начинается с фиксации переключаемого сегмента типа x в подчинении его старого владельца, продолжается поиском нового владельца операцией Q и заканчивается подключением фиксированного сегмента x к новому владельцу. //

Оператор, реализующий данную операцию, не связывает и не фиксирует области действия ни одного термина.

Операция как обобщение д-операции определяется так:

1. Всякая д-операция есть операция.

2. Если P - предикат, Q, R - операции, то

M: begin Q end M

M: if there holds P then Q else R end M

M: while there holds P do Q end M

- операции.

Определенная таким образом операция является формальным объектом, правила построения которого (с учетом всего вышеизложенного) представляют в определенном смысле базисные правила построения Л-программ. Их можно характеризовать как правила образования операций в качестве компонентов Л-программы.

Но в этих правилах неопределенным остался один вопрос - конкретизация составляющих предикатов и операций данной операции. Предполагалось, что вместо них в каждом конкретном случае подставляются какие-то конкретные предикаты и операции, которые должны быть построены по тем же самым правилам образования. Ясно, что при этом должны быть соблюдены определенные правила подстановки, которые по существу и определяют, какие Л-программы могут быть построены на основе приведенных схем операций. Эти правила можно охарактеризовать как правила преобразования (вывода) операций.

Подстановка. Ввиду того, что в состав операций в общем случае входят составляющие двух видов - предикаты и операции, нужно говорить и о подстановках двух видов - о подстановке одного предиката вместо другого в данной составной операции и о подстановке одной операции вместо другой в данной составной операции. Правилами подстановки должна быть обеспечена определенная логическая корректность (Л-корректность) конструируемых программ, касающаяся вхождения термов программы, связывания их составляющими предикатами и операциями программы, а также построения самих этих предикатов и операций. Требования, которые при этом должны быть соблюдены, аналогичны соответствующим правилам построения формул исчисления предикатов в логике. Относительно предикатов Д-программы эти требования совпадают с правилами построения формул исчисления предикатов. Относительно операций Л-программы они являются соответствующим обобщением правил построения формул исчисления предикатов.

Исходя из этого правила подстановки предикатов и операций определяются следующим образом.

Пусть M - операция, Q - ее подоперация (или предикат), а T - некоторая другая операция (соответственно предикат). Подстановка операции (предиката) T вместо операции (предиката) Q в операции M L -корректна, если выполняются следующие условия (правила L -подстановки):

1. Каждый связанный терм операции (предиката) T должен отличаться от тех термов операции M , в область действия которых входит операция (предикат) Q .
2. Каждый свободный терм операции (предиката) T должен совпадать с одним из тех термов операции M , в область действия которых входит операция (предикат) Q .

Помимо этого при подстановке операции в операцию действует еще дополнительное правило, расширяющее возможности подстановки операции в операцию. Оно позволяет подставлять вместо подоперации данной операции целую последовательность операций и формулируется следующим образом:

Пусть M -операция, Q - подоперация операции M , а $T = T_1, \dots, T_n$ ($n \geq 1$) - некоторая последовательность операций. Подстановка последовательности операций T вместо операции Q в операции M L -корректна, если при каждой операции T ($1 \leq i \leq n$) этой последовательности удовлетворяются вышеизложенные условия подстановки операции в операцию.

На основе введенных правил подстановки определяются L -корректные предикаты и операции.

L -корректный предикат определяется так.

1. Логическая константа есть L -корректный предикат.
2. Если L - L -корректный предикат, P - его подпредикат, а S - некоторый предикат, то результат L -корректной подстановки S вместо P в L - L -корректный предикат.

L -корректная операция определяется следующим образом:

1. Пустой оператор есть L -корректная операция.
2. Если M - L -корректная операция, P и Q - соответственно ее L -корректный подпредикат и подоперация, а T - некоторая операция, то результат L -корректной подстановки S вместо P в L - L -корректная операция.

В этих определениях в сжатой форме и высказываются правила логического конструирования L программ. Сама L -программа определяется так:

Если R, S - номены, Q - L -корректная операция, то
routine $R(S)$; execution of Q end R
- L -программа.

Номены R и S обозначают соответственно имя определяемой L -программы и имя связываемой с ней L -схемы. Последнее выступает в роли параметра программы.

Л и т е р а т у р а

1. К р а х т В.А. От концептуальной модели предметной области до структуры базы данных. См. наст. сб., с. 5.

2. К р а х т В.А., Э й в а к Ю.Э. Алгоритмический язык манипулирования данными ДАМАЛ. Таллин, ТПИ, 1982. 118 с.

3. К л и н и С.К. Введение в метаматематику. М., Изд-во иностранной литературы, 1957, 526 с.

4. В а с с и л ь М.Х. Реляционно-решетчатая структура базы данных. См. наст. сб., с. 67.

W. Kracht

Logical Construction of Latticework Structured Data Processing Programs

Summary

Possibilities of formal construction of logically correct programs for relational latticework structured data processing on the basis of program representation in the form of formal system objects are discussed. For this purpose the theory of L -programs involving the informal program theory, the formal system and the metatheory is developed and presented. Operations, predicates and data terms are distinguished as objects of the theory. L -correctability and substitution rules for program components are defined.

АЛГОРИТМЫ РЕАЛИЗАЦИИ ОПЕРАТОРОВ ПОИСКА ДАННЫХ
РЕЛЯЦИОННО-РЕШЕТЧАТОЙ СТРУКТУРЫ

Введение. В работе [1] показано, что реляционно-решетчатая структура (PPC) базы данных (БД), поддерживаемая системой ПАРЕС, отображается в памяти ЭВМ таким же образом, как и сетевая структура БД по предложениям КОДАСИЛ. Но интерпретируется и описывается она совершенно по-другому [1, 2]. Значительное отличие PPC от сетевой структуры БД высказывается и при манипулировании данными. Язык манипулирования данными (ЯМД) системы ПАРЕС объединяет основные функции базового языка и ЯМД известных систем с включающим языком, опираясь при этом на совершенно другой "образ мышления" [3]. Этот "образ мышления" является более близким (по принципам манипулирования) к реляционной, нежели к сетевой системе и во многом похож на результаты последних работ Дейта [4].

Манипулирование данными, особенно на основе БД сетевой структуры, тесно связано с ведением т.н. текущих объектов манипулирования — текущих записей данного типа, цепей (наборов) данного типа и т.д. В этом плане хорошо известна общая стратегия ведения текущих в сетевых системах по предложениям КОДАСИЛ. В принципе такая стратегия могла бы быть принята за основу манипулирования данными, хранящимися в БД PPC. Отраженная в [3] стратегия ведения текущих по существу является весьма близкой к соответствующей стратегии по предложениям КОДАСИЛ [5]. Однако все это оказывается непригодным с точки зрения предложенных в [6] логических принципов построения программ обработки данных.

Исходя из основных положений теории Л-программ [6], в данной работе приводятся упрощенные алгоритмы реализации

операторов поиска усовершенствованного языка DAMAL. Вместе с этим предлагается и новая стратегия ведения текущих, обеспечивающая выполнение этих операторов согласно теории Л-программ. Эта стратегия радикально отличается от стратегии ведения текущих систем типа КОДАСИЛ - по ней соответствующие текущие обновляются или, точнее, устанавливаются лишь при непосредственной необходимости. Отметим, что подобная тенденция развития наблюдается в ряде систем, разработанных в последние годы [7].

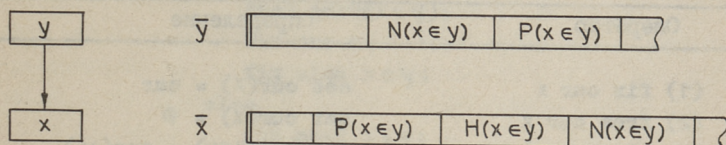
Текущие сегменты. Рассмотрим некоторый фрагмент БД, находящийся в "поле зрения" программы обработки и состоящий из двух реляционных объектов (типов сегментов) y и x , между которыми существует реляционная связь (РС), ориентированная от y к x (фиг. 1). Обозначаем через $\bar{y}$ и $\bar{x}$ текущие экземпляры сегментов соответствующего типа, находящиеся в буфере программы. Среди системных полей сегмента $\bar{y}$ существуют два, а среди системных полей сегмента $\bar{x}$ - три поля, связанные с представлением РС между сегментами типа y и x (фиг. 1) [1]. Обозначим их (имея в виду их семантику) следующим образом:

- $\bar{y}.N(x \in y)$ - ссылка на следующий сегмент субъюнкции ($x \in y$), представленная в текущем сегменте типа y ;
- $\bar{y}.P(x \in y)$ - ссылка на предыдущий сегмент субъюнкции ($x \in y$), представленная в текущем сегменте типа y ;
- $\bar{x}.N(x \in y)$ - ссылка на следующий сегмент субъюнкции ($x \in y$), представленная в текущем сегменте типа x ;
- $\bar{x}.H(x \in y)$ - ссылка на сегмент владельца субъюнкции ($x \in y$), представленная в текущем сегменте типа x ;
- $\bar{x}.P(x \in y)$ - ссылка на предыдущий сегмент субъюнкции ($x \in y$), представленная в текущем сегменте типа x .

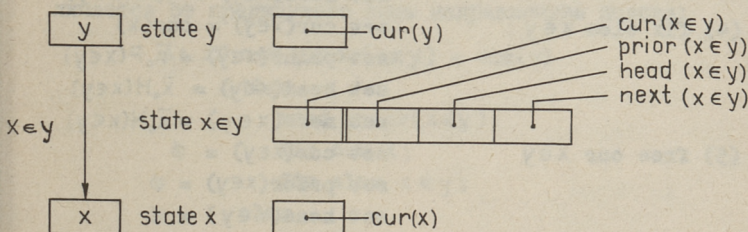
Все эти поля имеют одинаковый формат. Содержанием их является адрес некоторого сегмента, представленный в виде значения соответствующего т.н. ключа БД.

Система индикации текущих отражает ход процесса манипулирования базовыми данными (в программе обработки) и используется для организации работы системы. В ней ведется учет для всех типов сегментов (PO) и субъюнкций (PC), используемых в программе. Каждый тип сегмента представлен в системе одноэлементным вектором (state x и state y , см. фиг. 2), который содержит ссылку на текущий сегмент данно-

го типа. Состояние каждой субъюнкции характеризуется вектором ($state\ x \in y$, фиг. 2), состоящим из четырех элементов - из ссылки на текущий сегмент субъюнкции, из ссылок на соседей и владельца этого же текущего сегмента субъюнкции[6].



Фиг. 1. Текущие сегменты БД в буфере программы.



Фиг. 2. Векторы текущего состояния типов сегментов и субъюнкции в программе обработки.

Все перечисленные ссылки также представляются в виде значения соответствующего ключа БД. Помимо сказанного, используется еще одна системная переменная (cur) для обозначения текущего сегмента системы.

По теории Л-программ при запуске программы обработки все векторы текущих находятся в начальном состоянии, т.е. они заполнены некоторым начальным значением, обозначаемым через " $\emptyset$ ". В ходе выполнения программы содержание векторов текущего состояния изменяется согласно логике программы и теории Л-программ. При завершении программы обработки все векторы состояния должны опять находиться в начальном состоянии.

Содержание векторов индикации текущих обновляется специальными операторами фиксации состояния. Список этих операторов вместе с соответствующими определениями их семантики приведен в таблице I, причем определения семантики

этих операторов представлены в виде эквивалентных последовательностей операции присваивания.

Т а б л и ц а I

Операторы фиксации значения векторов индикации
текущего состояния

Оператор	Определение
(1) <code>fix cur x</code>	<code>set cur(x) = cur</code>
(2) <code>free cur x</code>	<code>set cur(x) = ϕ</code>
(3) <code>fix head x $\in$ y</code>	<code>set cur(x $\in$ y) = cur(y)</code> <code>set prior(x $\in$ y) = $\bar{y}_0.P(x \in y)$</code> <code>set head(x $\in$ y) = cur(y)</code> <code>set next(x $\in$ y) = $\bar{y}_0.N(x \in y)$</code>
(4) <code>fix elem x $\in$ y</code>	<code>set cur(x $\in$ y) = cur(x)</code> <code>set prior(x $\in$ y) = $\bar{x}_0.P(x \in y)$</code> <code>set head(x $\in$ y) = $\bar{x}_0.H(x \in y)$</code> <code>set next(x $\in$ y) = $\bar{x}_0.N(x \in y)$</code>
(5) <code>free cur x $\in$ y</code>	<code>set cur(x $\in$ y) = ϕ</code> <code>set prior(x $\in$ y) = ϕ</code> <code>set head(x $\in$ y) = ϕ</code> <code>set next(x $\in$ y) = ϕ</code>

Операции поиска. В этой системе операции поиска делятся на простые (элементарные) и составные. Среди них центральное место занимают составные операции поиска, а элементарные операторы служат, в основном (в качестве базиса), для реализации этих составных операций.

Элементарные операторы поиска являются средствами, при помощи которых строится большинство сложных операций поиска данных из БД РРС, определенных в [6]. К ним относятся три оператора:

- `find next x  $\in$  y,`
- `find prior x  $\in$  y,`
- `find head x  $\in$  y.`

Алгоритмы выполнения этих операторов определяются, в основном, на базе операторов фиксации состояния векторов текущих. Помимо сказанного, используется еще один оператор (`read`) для обращения к структурам хранения БД, при помощи которого осуществляется чтение (по ключу БД) очередного сегмента из БД.

O find next $x \in y$ - оператор поиска очередного (последующего) сегмента в экземпляре субъюнкции. Семантика его определяется следующим образом:

```
A: if not next( $x \in y$ ) = cur( $y$ )
    then
        read next( $x \in y$ );
        fix cur  $x$ ;
        fix elem  $x \in y$ ;
    else
        fix head  $x \in y$ ;
end A;
```

O find prior $x \in y$ - оператор поиска очередного (предыдущего) сегмента в экземпляре субъюнкции. Этот оператор отличается от предыдущего лишь направлением поиска:

```
A: if not prior( $x \in y$ ) = cur( $y$ )
    then
        read prior( $x \in y$ );
        fix cur  $x$ ;
        fix elem  $x \in y$ ;
    else
        fix head  $x \in y$ ;
end A;
```

O find head $x \in y$ - оператор поиска сегмента-владельца экземпляра субъюнкции. Этот оператор является эквивалентным следующей последовательности операторов:

```
read head  $x \in y$ ;
fix cur  $y$ ;
fix head  $x \in y$ ;
```

Поскольку в составных операциях циклического поиска сегментов из БД (см. ниже) опущен директор направления поиска, то далее из трех рассмотренных операторов потребуются лишь два - **find next** и **find head**.

Составные операции поиска. В теории Л-программ предлагаются следующие составные операции поиска [6]:

- выборка сегмента корня БД;
- выборка сегмента по ключу идентификации;
- выборка подмножества подчиненных субъюнкции;
- выборка одноэлементного подмножества подчиненных субъюнкции;

- выборка владельца субънкции.

Приведем упрощенные алгоритмы их реализации.

Выборка корня БД PPC представляется в Л-программе в виде следующей операции:

```
A: if there exists root x
    such that P(x)
    then Q1(x)
    else Q2
end A;
```

Эта операция связывает терм x - следовательно, до и после выполнения данной операции, x является неопределенным. Это означает также, что переменная $cur(x)$ должна до и после выполнения операции иметь значение " ϕ ". Терм x считается неопределенным и для подоперации Q_2 . Однако, он является свободной переменной для предиката $P(x)$ и подоперации $Q_1(x)$. Сказанное обеспечивается следующей схемой реализации:

```
A: begin
    read r;
    B: if not-found = 'false'
        then
            fix cur x;
            C: if P(x)
                then
                    Q1(x)
                    free cur x;
                else
                    free cur x;
                    Q2
                end C;
            else
                Q2
            end B;
        end A;
```

где r - адрес корневого сегмента БД, зафиксированный системой;

not-found - имя системной переменной, предназначенной для передачи программе сообщений об отсутствии в БД сегмента по заданному значению ключа БД.

0 Выборка сегмента по ключу идентификации определяется в Л-программе так:

```
A: if there exists any x (key)
    such that P(x)
    then Q1(x)
    else Q2
end A;
```

Она является по отношению к связыванию термина x совершенно аналогичной предыдущей операции и отличается от нее только по способу определения адреса отыскиваемого сегмента:

```
A: begin
    readr key, hash(key);
    B: if not-found = 'false'
        then
            fix cur x;
            C: if P(x)
                then
                    Q1(x)
                    free cur x;
                else
                    free cur x;
                    Q2
                end C;
            else
                Q2
            end B;
        end A;
```

где **readr** - модификация оператора **read** для чтения сегментов из БД, размещение которых организовано способом хэширования;

hash - функция (процедура) при помощи которой значению ключа идентификации (**key**) сопоставляется соответствующее значение ключа БД.

0 Выборка подмножества подчиненных субъекции представляется по схеме:

```
A: for each  $x \in y$ 
    such that P(x,y)
    do Q(x,y)
end A;
```


Здесь операция также связывает терм x - его значение до и после выполнения операции является неопределенным. Но внутри операции (для $Q(x,y)$) он является свободной переменной, значение которой (точнее значение переменных $cur(x)$, $cur(x \in y)$ и т.д.) изменяется в ходе циклического выполнения этой операции. Терм y является свободной (фиксированной) переменной как вне, так и внутри этой операции. Схема реализации его выглядит следующим образом:

```

A: begin
    fix head  $x \in y$ ;
    find next  $x \in y$ ;
    B: while not  $cur(x \in y) = head(x \in y)$ 
        do
            C: if  $P(x)$ 
                then
                     $Q(x,y)$ 
                end C;
            find next  $x \in y$ ;
        end B;
    free cur  $x \in y$ ;
    free cur  $x$ 
end A;
```

О Выборка одноэлементного подмножества подчиненных сублинкции:

```

A: if there exists  $x \in y$ 
    such that  $P(x,y)$ 
    then  $Q_1(x,y)$ 
    else  $Q_2(y)$ 
end A;
```

С точки зрения связанности термов x и y эта операция в большей мере аналогична предыдущей. Разница их заключается в том, что в данном случае для подоперации $Q_2(y)$ терм x является неопределенным. Схема его реализации представляется теперь так:


```

A: begin
    set L = 'true';
    fix head  $x \in y$ ;
    find next  $x \in y$ ;
    B: while not  $\text{cur}(x \in y) = \text{head}(x \in y)$ 
        do
            C: if  $P(x)$ 
                then
                     $Q_1(x, y)$ 
                    fix head  $x \in y$ ;
                    set L = 'false';
                else
                    find next  $x \in y$ ;
            end C;
        end B;
    free cur  $x \in y$ ;
    free cur  $x$ ;
    D: if L = 'true'
        then
             $Q_2(y)$ 
        end D;
    end A;

```

О Выборка владельца субъюнкции выполняется в Л-программе следующей операцией:

```

A: if there exists  $y \exists x$ 
    such that  $P(y, x)$ 
    then  $Q_1(y, x)$ 
    else  $Q_2(x)$ 
end A;

```

В отличие от вышеприведенных операций поиска сегментов субъюнкции в данном случае связывается переменная y (которая обозначает владельца некоторого экземпляра субъюнкции), а переменная x остается свободной. Но ее значение является фиксированным в пределах всей операции. Переменная y является свободной для предиката $P(y, x)$ и подоперации $Q_1(y, x)$, но не определена для подоперации $Q_2(x)$. Схема реализации этой операции выглядит так:


```

A: begin
    fix elem  $x \in y$ ;
    find head  $x \in y$ ;
    B: if  $P(y, x)$ 
        then
             $Q_1(y, x)$ 
            free cur  $y$ ;
            free cur  $x \in y$ ;
        else
            free cur  $y$ ;
            free cur  $x \in y$ ;
             $Q_2(x)$ ;
        end B;
    end A;

```

Заклучение. Нами рассмотрены общие принципы политики ведения текущих при реализации операции поиска ЯМД для сетевых-подобных систем. Эти принципы являются результатом как теоретического анализа, так и практического опыта, полученного в ходе разработки, реализации и применения системы ПАРЕС [3, 8]. Приведенные алгоритмы являются основой проверки правильности программ обработки (в смысле [6]) во время их выполнения. Рассмотрение вопросов оптимизации этих алгоритмов, а также рассмотрение алгоритмов операторов обновления выходят за рамки данной работы.

Л и т е р а т у р а

1. В а с с и л ь М.Х. Реляционно-решетчатая структура базы данных. См. наст. сб. с. 83.
2. К р а х т В.А. От концептуальной модели предметной области до структуры базы данных. См. наст. сб. с. 5.
3. К р а х т В.А., Э й в а к Ю.Э. Алгоритмический язык манипулирования данными DAMAL. Таллин, ТПИ, 1982, 118 с.
4. Date C.I. An introduction to the unified database language (UDL). - 6 th Int. Conf. Very Large Data Bases, 1980, New York, 1980, p. 15-32.

5. О л л е Т.В. Предложения КОДАСИИ по управлению базами данных. М., Финансы и статистика, 1981. 287 с.

6. К р а х т В.А. Логическое конструирование программ обработки данных для реляционно-решетчатой структуры. См. наст. сб., с. 29.

7. Naveen Prakash, P. Sadanantan, S.M. Desai, K. Rama Rao and P.N. Gejji. Samhita - A data base management system. - The Computer Journal, 1981, vol. 24, N 1, p. 48-51.

8. К р а х т В.А., Э й в а к Ю.Э. Система ведения баз данных и манипулирования данными "ПАРЕС". Общее описание. Таллин, ТПИ, 1979, 37 с.

J. Eivak

Algorithms for Implementation of Latticework

Data Retrieval Operators

Summary

High level retrieval operations for relational latticework database are considered. Concepts of database currency indicators management and basic retrieval operators needed for implementation of these high level retrieval operations are discussed.

АЛГОРИТМЫ РЕАЛИЗАЦИИ ПРЕДИКАТОВ С КВАНТОРАМИ,
СВЯЗАННЫХ С ПОИСКОМ ДАННЫХ РЕЛЯЦИОННО-
РЕШЕТЧАТОЙ СТРУКТУРЫ

В работе [1], посвященной логическому конструированию программ обработки данных, изложены основные принципы построения Л-программ и указаны основные составные их конструкции. Среди других конструкций указаны и конструкции составных операторов поиска данных в базе реляционно-решетчатой структуры, а также языковые конструкции для представления предикатов поиска.

В работе [2] рассматривается реализация этих операторов поиска данных на расширенном Л-языке. Приведены алгоритмы реализации составных операторов поиска данных на основе элементарных операторов поиска и операторов фиксации состояния БД.

Данная работа является логическим продолжением обоих указанных работ. В ней приводятся алгоритмы реализации составных логических условий с кванторами на расширенном Л-языке, который можно рассматривать как усовершенствованный вариант языка манипулирования данными DAMAL системы ПАРЕС.

Логические условия, определенные в Л-языке, встречаются только в составных операторах поиска данных этого языка. Они называются в теории Л-программ [1] предикатами и делятся на простые и составные предикаты. Частным видом последних являются предикаты с кванторами. Простые предикаты и предикаты, построенные из простых при помощи только логических союзов (и, или, не), мало отличаются от аналогичных конструкций в общеизвестных языках программирования. Поэтому в данной работе рассматриваются только предикаты с кванторами. Алгоритмы реализации их представляются на основе

базисных операторов поиска данных и операторов фиксации векторов состояния БД, описанных в [2].

Предикаты с кванторами имеют три формы представления [1].

(1) $\text{all } (x \in y) (P(x,y))$,

(2) $\text{ex } (x \in x) (P(x,y))$,

(3) $\text{ex } (y \ni x) (P(y,x))$.

Первая форма предиката с квантором предназначена для проверки истинности условия, выраженного составляющим предикатом $P(x,y)$, при всех подчиненных элементах субъюнкции $(x \in y)$. Логическое условие, представленное по схеме (I), считается истинным, если предикат $P(x,y)$ удовлетворяется при всех подчиненных элементах указанной субъюнкции. Алгоритм проверки истинности предиката, построенного по схеме (I), представляется следующим образом:

```
A: begin
    set L = true
    fix head  $(x \in y)$ 
    find next  $(x \in y)$ 
    B: while not cur  $(x \in y)$  = head  $(x \in y)$ 
        do
            C: if  $P(x,y)$ 
                then find next  $(x \in y)$ 
                else set L = false
                    fix head  $(x \in y)$ 
            end C;
        end B;
    free cur  $(x \in y)$ 
    free cur x
end A;
```

где L - логическая переменная, представляющая истинностное значение проверяемого условия.

Вторая форма предиката с квантором предназначена для проверки истинности условия, выраженного составляющим предикатом $P(x,y)$ при некотором элементе субъюнкции $(x \in y)$. Логическое условие, представленное по схеме (2), считается истинным, если среди подчиненных элементов существует элемент, при котором удовлетворяется условие $P(x,y)$.

Алгоритмы проверки предиката по схеме (2) представляются так:

```
A: begin
  set L = false
  fix head ( $x \in y$ )
  find next ( $x \in y$ )
  B: while not cur ( $x \in y$ ) = head. ( $x \in y$ )
    do
      C: if P( $x, y$ )
        then      set L = true
                  fix head ( $x \in y$ )
        else      find next ( $x \in y$ )
      end C;
    end B;
  free cur ( $x \in y$ )
  free cur ( $x$ )
end A;
```

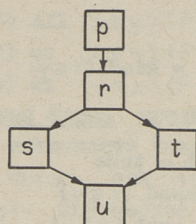
Третья форма предиката с квантором отличается от предыдущей тем, что она предназначена для проверки истинности условия $P(x, y)$ не у подчиненных субъекции, а у ее владельца. Логическое условие, построенное по схеме (3), считается истинным, если условие $P(x, y)$ выполняется при владельце субъекции.

Алгоритм реализации предиката по схеме (3) выглядит следующим образом:

```
A: begin
  set L = false
  fix elem ( $x \in y$ )
  find head ( $x \in y$ )
  B: if P( $y, x$ )
    then set L = true
    end B;
  free cur ( $x \in y$ )
  free cur ( $y$ )
end A;
```

Во всех трех формах предикатов с квантором составляющий предикат $P(x, y)$ сам может быть предикатом с квантором. Но кванторы в таких составляющих предикатах прилагаются уже к

другим подчиненным субъюнкциям базы. Проверка истинности таких условий приводит к рекурсивному применению приведенных алгоритмов. Построение проверки таких условий иллюстрируется примером.



$p(\dots) \subset \dots$

$r(\dots) \subset r$

$s(\dots) \subset r$

$t(\dots) \subset r$

$u(\dots) \subset s \times t$

Фиг. 1. Фрагмент схемы БД.


```

A1: begin
  set L = true
  fix head (ser)
  find next (ser)
  B1: while not cur (ser) = head (ser)
    do
      C1: if
        A2: begin
          set L = false
          fix head (ues)
          fix next (ues)
          B2: while not cur (ues) = head (ues)
            do
              C2: if
                A3: begin
                  set L = false
                  fix elem (uet)
                  find head (uet)
                  B3:
                    if P(t)
                      then set L = true
                    end B3;
                  free cur (uet)
                  free cur (t)
                  end A3;
                then set L = true
                  fix head (ues)
                else find next (ues)
                end C2;
              end B2;
            free cur (ues)
            free cur (u)
          end A2;
        then find next (ser)
        else set L = false
          fix head (ser)
        end C1;
      end B1;
    free cur (ser)
    free cur (s)
  end A1;

```

Л и т е р а т у р а

1. Крахт В.А. Логическое конструирование программ обработки данных для реляционно-решетчатой структуры. См. наст. сб., с. 29.

2. Эйвак Ю.Э. Алгоритмы реализации операторов поиска данных реляционно-решетчатой структуры. См. наст. сб., с. 49.

U. Pauklin

Algorithms for Implementation of Logical Conditions for Relational Latticework Database

Summary

Concepts of logical conditions of data manipulation language for relational latticework database are discussed. Algorithms for implementation of conditions with quantifiers are presented.

РЕЛЯЦИОННО-РЕШЕТЧАТАЯ СТРУКТУРА БАЗЫ ДАННЫХ

Введение. В настоящее время основой каждой системы обработки информации является база данных (БД). Возможности и эффективность информационной системы, а также удобство пользования системой во многом зависят от выбранной организации данных в БД - т.е. от структуры базы данных. В [1] изложены принципы моделирования предметной области информационной системы. Результатом концептуального моделирования по предложенной методике является модель предметной области в форме реляционной решетки. Эта же модель положена в основу представления структуры базы данных в системе ПАРЕС. В данной работе рассматриваются вопросы отображения реляционно-решетчатой модели в памяти ЭВМ и способы описания реляционно-решетчатой структуры БД.

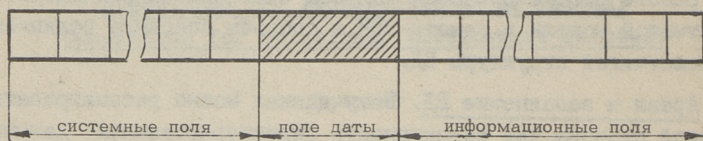
Ареал и заполнение БД. Базу данных можно рассматривать, с одной стороны, как совокупность структурированных данных о предметной области, а с другой стороны, как некоторое пространство внешней памяти ЭВМ, предназначенное для размещения данных. Это пространство памяти мы будем в дальнейшем называть ареалом БД, оно является средой хранения для вышеупомянутой совокупности данных. Ареал БД заполняется данными, отражающими концептуальную структуру предметной области. Обычно под базой данных подразумевается именно заполнение (содержание) ареала БД.

Ареал БД в системе ПАРЕС может состоять из одного или нескольких регионов. Регионы базы данных представляют собой наборы данных в смысле операционной системы. Регионы создаются по принципам страничной организации памяти и характеризуются числом страниц и размером страницы.

Основы отображения. Заполнение ареала БД в системе ПАРЕС является отображением реляционно-решетчатой модели (RPM)

предметной области. Согласно [1], в РРМ выделяется два вида составных элементов – реляционные объекты (РО) и реляционные связи (РС). Каждый РО рассматривается как образ некоторого множества объектов, а каждая РС как образ некоторого множества субъюнкций. В базе данных каждый РО отображается множеством однотипных сегментов, а каждая РС множеством соответствующих совокупностей логически связанных между собой сегментов, которые также можно назвать субъюнкциями. В каждую субъюнкцию входит точно один сегмент типа владельца реляционной связи и ноль, один или несколько сегментов типа подчиненного РС. При этом каждый сегмент БД может войти в состав только одной субъюнкции данного типа. Каждому атрибуту РО в БД соответствует элемент или составной компонент данных в сегментах соответствующего типа.

Структура сегментов. Каждый сегмент в базе данных системы ПАРЕС состоит из полей двух видов – системных и информационных. Кроме того, в каждом сегменте предусмотрено еще одно специальное поле – поле даты обновления сегмента. Структура сегмента приведена на фиг. 1.



Фиг. 1. Структура сегмента БД в системе ПАРЕС.

Системные поля служат для идентификации сегментов системой и для связывания сегментов между собой – т.е. для представления субъюнкций. Системные поля определяются системой на основе описания структуры БД, они используются и обновляются системой и системными утилитами, но являются недоступными для прикладных программ (ПП) пользователей, работающих с базой данных. Число системных полей сегмента зависит от способа размещения сегмента и от того, в состав каких субъюнкций входит данный сегмент.

Поле даты служит для запоминания даты последнего обновления сегмента, оно определяется и обновляется системой, од-

нако, его значение доступно прикладным программам и им можно пользоваться при обработке.

Информационные поля служат для представления информации о предметной области, а также для представления ключей размещения и упорядочения сегментов. Информационные поля, их типы и размеры определяются пользователем (проектировщиком) БД в описании модели БД, они обновляются и используются прикладными программами. Исключение составляют вышеупомянутые ключи, значения которых используются системой для ведения БД. Иногда в некоторых типах сегментов базы данных информационные поля могут отсутствовать.

Типы элементов данных. Для описания информационных полей сегментов в системе ПАРЕС предусматриваются следующие типы элементарных данных:

1. Числовые типы данных:

а) типы числовых данных во внешнем (форматном) представлении:

- в форме с фиксированной точкой - тип F,
- в экспоненциальной форме - тип E,

б) типы числовых данных во внутреннем (арифметическом) представлении:

- в форме упакованного десятичного числа - тип DECIMAL,
- в форме двоичного целого числа - тип INTEGER,
- в форме двоичного вещественного числа - тип REAL.

2. Символьный (литерный) тип данных - тип CHARACTER.

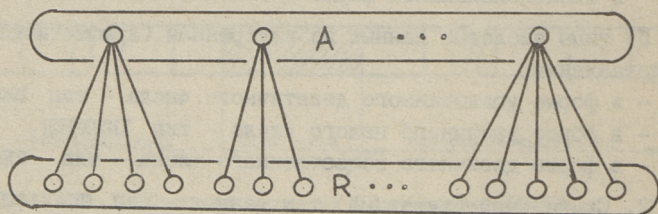
3. Неопределенный тип данных - тип SPACE.

При выполнении прикладных программ системой обеспечивается автоматическое преобразование числовых данных из одного типа представления в другой. При присвоении значения поля данных одного типа полям данных других типов пользователь должен учесть диапазоны представимых значений в полях данных типов. Все значения полей числовых типов сравнимы между собой, но не сравнимы, например, со значением поля символьного типа. Полю данных неопределенного типа может быть присвоено значение поля любого типа и наоборот. Форма представления и характер данных для полей типа SPACE определяется конкретной прикладной программой. Ответственность

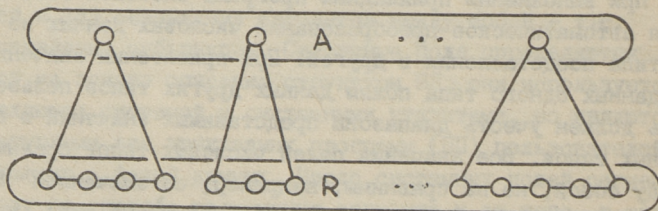
за правильность возможных преобразований в таком случае несет пользователь (прикладной программист).

Представление субъонкций. Понятие субъонкции как отображения реляционной связи близко к понятию набора в сетевой модели КОДАСИЛ [2, 3]. В описании сетевой модели наборы описываются явно в отдельных статьях описания [4], но при определении РРМ реляционные связи явно не описываются. Они определяются косвенно собственными связями реляционных объектов [1].

Существует ряд возможностей реализации субъонкций (наборов), в [5] дан обзор таких возможностей. По существу, реализация субъонкции заключается в создании иерархических связей между сегментом типа владельца РС и соответствующими сегментами типа подчиненного РС. Наиболее естественным образом с концептуальным представлением субъонкции согласуются два принципиальных способа организации таких связей, которые можно называть параллельной и последовательной реализацией субъонкций.



Фиг. 2. Представление реляционной связи диаграммой соответствия.

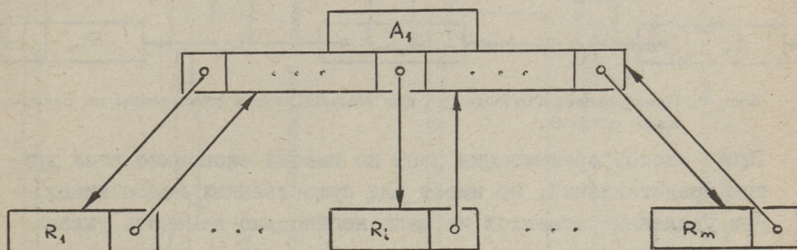


Фиг. 3. Представление реляционной связи цепной диаграммой.

На фиг. 2 и 3 представлен концептуальный прообраз субъонкции в РРМ – реляционная связь. На фиг. 2 РС между множе-

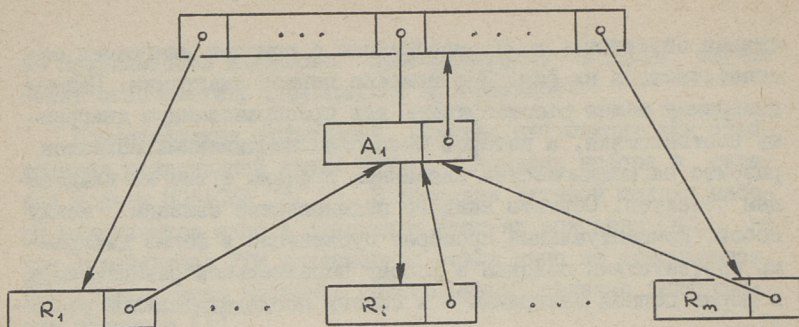
ствами объектов A и R изображена с помощью диаграммы соответствия, а на фиг. 3 с помощью цепной диаграммы. Цепную диаграмму можно рассматривать как преобразованную диаграмму соответствия, в которой множество подчиненных объектов разбито на подмножества связанных с одним и тем же владельцем объектов. Объекты каждого подмножества связаны между собой. Концептуальный прообраз субъюнкции в форме диаграммы соответствия положен в основу параллельной реализации, а в форме цепной диаграммы – в основу последовательной реализации.

На фиг. 4 приведен способ параллельной реализации субъюнкции. В каждом сегменте типа владельца РС существует список или массив указателей, каждый из которых указывает на один сегмент типа подчиненного РС. В подчиненных сегментах субъюнкции имеются указатели на сегмент владельца. Достоинством такой реализации субъюнкции является быстрый поиск сегментов в БД, недостатком – переменное число указателей в сегменте владельца. Это означает, что в БД сегменты одного и того же типа имеют различную длину, что существенно усложняет ведение БД.



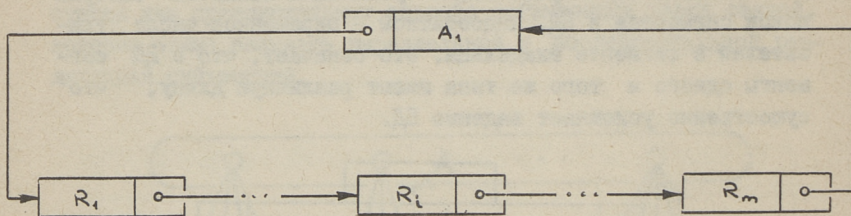
Фиг. 4. Представление субъюнкции при помощи массива указателей.

Чтобы избежать этого недостатка, можно использовать способ, изображенный на фиг. 5. При таком способе вместо списка указателей переменной длины в сегменте типа владельца РС имеется указатель на подобный список, расположенный отдельно от сегмента и называемый индексом. Этим достигается одинаковая длина однотипных сегментов, но для ведения индексов субъюнкции требуются дополнительные средства [4].



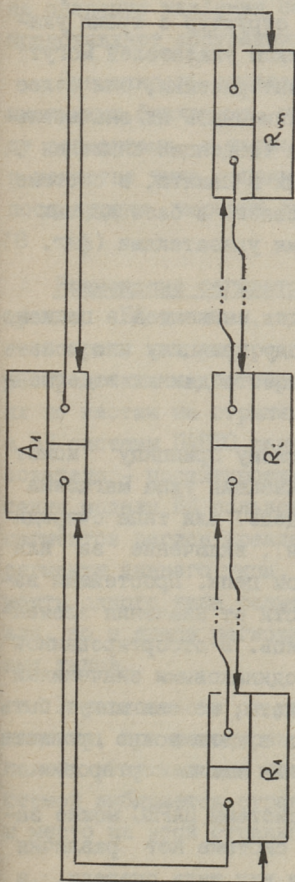
Фиг. 5. Представление субъюнкции при помощи индекса указателей.

Основной последовательной реализацией субъюнкций является применение кольцевых структур или т.н. цепей [3, 5]. Простейшей формой кольцевых структур является цепь с одним указателем в каждом сегменте, указывающим на следующий сегмент в субъюнкции. Такая цепь изображена на фиг. 6.

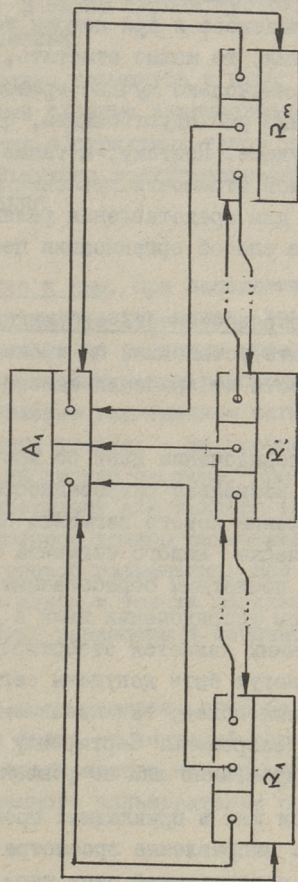


Фиг. 6. Представление субъюнкции при помощи цепи с указателем на следующий сегмент.

Этот способ организации цепи по памяти экономнее всех других представлений, но имеет ряд существенных недостатков. При удалении сегментов из цепи необходимо изменить указатель в предыдущем сегменте, но найти этот сегмент можно лишь после просмотра всей цепи. Поэтому, а также исходя из рассуждений о возможности восстановления цепи после аппаратного сбоя, целесообразнее использовать цепь с указателями на следующий и предыдущий сегмент, приведенную на фиг. 7. Но и такой способ организации цепи имеет свой недостаток. При обработке цепи часто возникает необходимость прекратить просмотр подчиненных цепи и перейти к владельцу. Но при организации цепи согласно фиг. 6 и 7 для перехода к владельцу необходимо пройти по цепи до ее конца, т.е. до владельца. Для предотвращения потерь машинного времени при таких переходах прибегают к использованию еще одного указателя -



Фиг. 7. Представление субъонкции при помощи цепи с указателями на следующий и предыдущий сегменты.



Фиг. 8. Представление субъонкции при помощи цепи с указателями на следующий и предыдущий сегменты и на входе.

указателя на владельца субъюнкции. Цепь с тремя указателями в каждом сегменте типа подчиненного является наиболее эффективной из кольцевых структур. Такая цепь изображена на рис. 8.

Если сравнивать способы представления субъюнктий при помощи индексов и при помощи кольцевых структур с тремя указателями, то можно отметить, что индексы указателей могут иметь несколько лучшие временные характеристики, они более экономно расходуют память, однако, реализовать их значительно труднее. Поэтому, а также с учетом тенденции снижения удельной стоимости хранения информации в памяти, в системе ПАРЕС для представления реляционных связей в базе данных выбран способ организации цепи с тремя указателями (фиг. 8).

Упорядоченность субъюнкции. Порядок сегментов в цепи может быть установлен по хронологическому принципу или в зависимости от значения какого-либо элемента данных в сегменте.

Упорядочение цепи по хронологическому принципу может иметь несколько разновидностей: организация типа магазина (включение нового сегмента первым в цепь) или типа очереди (включение нового сегмента последним), включение за или перед последним обработанным сегментом цепи. Простейшим вариантом упорядочения цепи в зависимости от значения элемента данных является отсортированная цепь. В отсортированной цепи могут быть допущены сегменты с одинаковыми значениями ключевых полей, так называемые дубликаты, но они могут быть также запрещены. Сортировку сегментов в цепи можно провести по возрастанию или по убыванию значения ключа сортировки.

Так как в прикладных программах системы ПАРЕС можно задавать направление просмотра цепи, в системе нет различия между организацией цепи типа магазина или типа очереди, а также между убывающей или возрастающей сортировкой. Различаются три вида упорядочения:

- отсортированная цепь без дубликатов (SORT),
- отсортированная цепь с дубликатами (SORTWD),
- хронологическая организация, новый сегмент включается в цепь последним.

Способы включения новых сегментов после или перед обработанным последним сегментом предоставляют возможность определения порядка сегментов в цепи на основе более сложных взаимосвязей между сегментами.

Влияние организации субъюнкции можно распространить и на операцию удаления сегментов. В таком случае субъюнкции представляют абстрактные типы данных.

Возможность определения порядка сегментов в цепи специальными функциями, учитывающими сложные взаимосвязи между сегментами, а также применение абстрактных типов данных в качестве типа организации субъюнкции предусматривается в последующих версиях системы ПАРЕС.

Размещение сегментов и доступ к ним. При заполнении ареала БД данными, т.е. сегментами разных типов, возникает вопрос, как распределить сегменты по регионам ареала, как распределить их по страницам в регионах и как распределить их по местам на страницах. Управление размещением сегментов в БД системы ПАРЕС ведется на двух уровнях - на уровне пользователя - проектировщика БД и на уровне системы. При описании модели БД пользователь определяет для каждого типа сегментов регион ареала БД, в котором должны размещаться сегменты данного типа, а также способ размещения. Все сегменты одного типа размещаются в одном и том же регионе ареала, но в одном регионе могут быть размещены и сегменты разных типов.

В зависимости от указанного пользователем способа размещения, при некоторых способах также от значений ключевых полей, и от наличия свободного пространства на страницах, системой выбирается страница указанного пользователем региона и место на этой странице.

В системе ПАРЕС предусмотрены четыре способа размещения сегментов. Первый способ предусматривает размещение сегмента по заданному адресу. Этот способ применяется всегда только для корневого сегмента. Корневой сегмент размещается всегда в начале первой страницы первого региона ареала БД. Второй способ предусматривает рандомизированное

размещение сегментов в заданном регионе, причем номер страницы для размещения сегмента вычисляется функцией расстановки на основе значения ключа размещения. Предусматривается также равномерное распределение сегментов данного типа с заданным шагом в регионе владельца одной собственной связи (третий способ), и размещение около владельца одной собственной связи (четвертый способ).

В зависимости от способа размещения сегментов к ним можно из прикладных программ обратиться по-разному - т.е. существует несколько способов доступа. Самый общий способ доступа - это последовательный доступ через какую-то цепь. Для этого сначала надо найти владельца интересующей нас цепи, а потом путем просмотра цепи найти нужный сегмент. Однако для того, чтобы начинать поиск по цепям, необходим еще хотя бы один способ так называемого прямого доступа к сегментам, чтобы зафиксировать сегмент, с которого следует начинать поиск. В системе ПАРЕС таких способов прямого доступа два и они соответствуют способу размещения корневого сегмента и способу размещения путем рандомизации. Доступ к корневому сегменту производится по заданному зафиксированному адресу, а к сегментам, размещенным в зависимости от значений функции расстановки, путем задания соответствующих значений ключевых элементов. Помимо этого, корневой сегмент и рандомизированные сегменты можно найти и последовательным доступом, но первое обращение из III к БД должно быть выполнено с использованием способа прямого доступа.

Операции обновления БД. В системе ПАРЕС для обновления базы данных предусмотрены четыре операции обновления БД - это операция включения нового сегмента в БД, операция удаления сегмента из БД, операция модификации (изменения значений информационных полей) сегмента и операция переключения сегмента из одного экземпляра цепи в другой, т.е. в подчинение другого экземпляра владельца.

Операция включения сегмента (INSERT) означает запоминание информационных полей сегмента в БД, присваивание значений системным полям и полю даты включаемого сегмента и изменение системных полей других сегментов, связанных с

новым сегментом. Новый сегмент включается в экземпляры цепей, владельцы которых заранее зафиксированы прикладной программой.

Операция удаления сегмента (DELETE) вызывает изменение системных полей соседних с удаляемым сегментом сегментов в цепях, к которым принадлежал удаляемый сегмент, а также освобождение участка памяти на странице, занимаемого ранее сегментом. Если удаляемый сегмент является владельцем какой-нибудь цепи, то удаляются и все его подчиненные сегменты и т.д.

Операция модификации сегмента (REPLACE) вызывает изменение системных полей в этом и в других сегментах лишь в том случае, если изменились значения ключевых полей, т.е. если изменилось размещение сегмента или изменился порядок цепи. В любом случае изменяется значение поля даты.

Операция переключения (REJOIN) вызывает изменения в системных полях сегментов, которые были связаны с переключаемым сегментом до переключения, в системных полях сегментов, которые будут связаны с сегментом после переключения, и в системных полях самого переключаемого сегмента. При этом сохраняются и не изменяются все подчиненные переключаемому сегменту сегменты.

Состав и объем БД. Какое пространство памяти выделить под базу данных, каким будет объем создаваемой БД – на эти вопросы необходимо дать ответ уже до загрузки БД, на этапе проектирования. Объем БД зависит от структуры БД, от структуры сегментов и от ожидаемого числа сегментов каждого типа. При выделении памяти под БД следует учесть также способы размещения сегментов, так как для эффективной рандомизации сегментов потребуется некоторый резерв памяти.

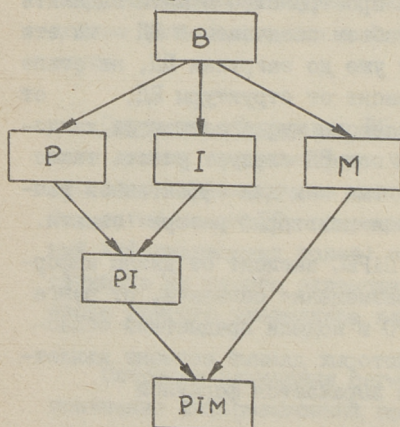
Длина сегмента в системе ПАРЕС зависит от длины информационных полей, от способа размещения сегмента, от ранга подчинения соответствующего РО в модели предметной области и от числа сублинкций, в которых данный сегмент является владельцем. Длина сегмента выражается формулой

$$L = 4(2 + 3n + 2m + r) + 8 + \sum l_i,$$

где n - ранг подчинения P_0 , m - число субъюнкций, в которых сегмент является владельцем, $r=1$, если сегмент с рандомизированным размещением, $r=0$ при остальных способах размещения, $\sum l_i$ - общая длина информационных полей сегмента.

Оценка ожидаемого числа сегментов каждого типа является трудной задачей. Легче количественно охарактеризовать субъюнкции, задавая т.н. коэффициент переноса субъюнкции - среднее число подчиненных в данном типе субъюнкции. В системе ПАРЕС пользователь частично освобожден от оценки объема БД. Если при описании модели БД задавать коэффициенты переноса для одной собственной связи каждого P_0 , то система сама сделает необходимые вычисления и выдаст соответствующие рекомендации по выбору ареала БД с учетом всех вышеупомянутых моментов.

Описание модели и ареала БД. Для описания структуры БД в системе ПАРЕС имеется специальный язык описания модели базы данных (ЯОМБД), на котором пользователь описывает реляционно-решетчатую модель. При создании языка особое внимание уделялось тому, чтобы, с одной стороны, предоставить пользователю удобные средства для описания PPM, а с другой стороны, лишить его возможности описывать противоречивую структуру БД.



$B(\text{NAIM}, \dots) := \theta$
 $P(\text{NAIM}, \text{ADRES}, \text{NOMER}, \dots) \subset B$
 $I(\text{NAZV}, \text{NOMER}, \text{CENA}, \dots) \subset B$
 $M(\text{NAZV}, \text{NOMER}, \dots) \subset B$
 $PI(\text{PLAN}, \text{FAKT}, \dots) \subset P \times I$
 $PIM(\text{PLAN}, \text{FAKT}, \dots) \subset PI \times M$

B - производственное объединение
 P - предприятие
 I - изделие
 M - календарный месяц
 PI - выпуск изделия предприятием
 PIM - распределение выпуска изделия предприятием по месяцам

Фиг. 9. Реляционно-решетчатая модель предметной области для АСУ производственным объединением.

DECLARE MODEL OASU: SET OF

B: COLLECTION OF
 NAIM: CHAR(20);
 EOD,
 ROOT OF MODEL,
 LOCATED IN OBL1, EOR B;

P: COLLECTION OF
 NAIM: CHAR(30);
 ADRES: COL OF
 GOROD: CHAR(15);
 ULICA: CHAR(30);
 DOM: CHAR(3);
 EOD ADRES;
 NOMER: CHAR(3); EOD,
 RELATED TO B BY SORT(NOMER) WITH COEFF 20,
 LOCATED BY HASH(NOMER) IN OBL2, EOR P;

I: COLLECTION OF
 NAZV: CHAR(10);
 NOMER: CHAR(4);
 CENA: DEC(6,2); EOD,
 RELATED TO B BY SORT(NOMER) WITH COEFF 200,
 LOCATED BY HASH(NOMER) IN OBL1, EOR I;

M: COLLECTION OF
 NAZV: CHAR(10);
 NOMER: CHAR(2); EOD,
 RELATED TO B BY SORT(NOMER) WITH COEFF 12,
 LOCATED BY HEAD, EOR M;

PI: COLLECTION OF
 PLAN: LONG INTEGER;
 FAKT: LONG INTEGER; EOD,
 RELATED TO P WITH COEFF 40
 AND TO I,
 LOCATED BY HEAD, EOR PI;

PIM: COLLECTION OF
 PLAN: INTEGER;
 FAKT: INTEGER; EOD,
 RELATED TO PI WITH COEFF 12
 AND TO M,
 LOCATED BY HEAD, EOR PIM;

EOS OASU;

Фиг. 10. Описание модели БД на ЯОМБД системы ПАРЕС.

DECLARE AREA OBLAST: MEDIUM FOR OASU,

SET OF

OBL1: REGION OF 20 PAGES BY 3520 BYTES;
OBL2: REGION OF 100 PAGES BY 7200 BYTES;
EOS OBLAST;

Фиг. 11. Описание ареала БД на ЯОАБД системы ПАРЕС.

Описание модели БД состоит из описаний отдельных реляционных объектов. Для каждого РО определяются его имя, структура и типы информационных полей соответствующего сегмента, собственные реляционные связи объекта и способы упорядочения и размещения сегмента. При этом способы равномерного распределения и размещения около владельца в качестве владельца рассматривают РО, указанный в описании первой собственной связи объекта.

На фиг. 9 приведен пример PPM предметной области и на фиг. 10 описание этой модели на ЯОМБД системы ПАРЕС.

Для описания ареала БД в системе ПАРЕС разработан язык описания ареала базы данных (ЯОАБД). Описание ареала на этом языке состоит из описаний отдельных регионов. Для каждого региона указывается его имя, число страниц и размер страницы. Описание ареала на основе рекомендаций, выданных системой при трансляции описания модели, приведено на фиг. 11.

Закключение. По внутреннему построению БД реляционно-решетчатой структуры мало отличается от сетевой модели КОДАСИЛ. Однако при описании и интерпретации этой модели реляционные связи остаются скрытыми (униконцептуальность модели) [11]. Это придает данной структуре новые качества, которые выявляются при описании концептуальной модели предметной области и, особенно, при обработке БД [6]. В данной работе эти аспекты модели не рассматривались.

Л и т е р а т у р а

1. Крахт В.А. От концептуальной модели предметной области до структуры базы данных. См. наст. сб., с. 5.
2. Дейт К. Введение в системы баз данных. М., Наука, 1980. 464 с.
3. Олле Т.В. Предложения КОДАСИЛ по управлению базами данных. М., Финансы и статистика, 1981. 286 с.
4. Язык описания данных КОДАСИЛ. М., Статистика, 1981, 183 с.

5. М а р т и н Дж. Организация базы данных в вычислительных системах. М., Мир, 1980. 662 с.

6. К р а х т В.А. Логическое конструирование программ обработки данных реляционно-решетчатой структуры. См. наст. сб., с. 29.

M. Vassil

Relational Latticework Database Structure

Summary

Mapping of the relational latticework database model into storage structure and the internal structure of the database of such kind are discussed. Constituents of the database are described. An example of database descriptions on the data definition language of the software system PARES is given.

РЕОРГАНИЗАЦИЯ И РЕСТРУКТУРИЗАЦИЯ БАЗЫ ДАННЫХ
РЕЛЯЦИОННО-РЕШЕТЧАТОЙ СТРУКТУРЫ

Введение. Реляционно-решетчатая структура базы данных (БД) отражает концептуальную структуру предметной области [1, 2]. Но предметная область не является статической, в ней постоянно происходят различного рода изменения. Многие из них могут быть предусмотрены уже при проектировании БД.

Однако при проектировании БД невозможно предвидеть все аспекты развития предметной области. Вследствие таких непредвиденных изменений БД может перестать соответствовать своей предметной области и потребуются ее перестройка — реорганизация.

В данной работе рассматриваются проблемы и возможности реорганизации базы данных реляционно-решетчатой структуры.

Реорганизация и реструктуризация БД. База данных реляционно-решетчатой структуры характеризуется двумя схемами или описаниями данных — схемой ареала и схемой модели содержимого БД [2]. Первая из них определяет объем и структуру среды хранения БД, вторая — внутреннюю логическую структуру БД, что обычно и понимается под структурой БД. Опираясь на понятия ареала и модели БД можно различить два вида перестройки БД реляционно-решетчатой структуры — реорганизацию и реструктуризацию БД.

Под реорганизацией БД в системе ПАРЕС понимается перезагрузка содержимого БД при изменении ее ареала. Модель БД при реорганизации не изменяется.

Под реструктуризацией БД понимается перезагрузка содержимого БД при изменении модели БД. При этом может также измениться структура среды хранения.

Для реорганизации, а также для реструктуризации БД необходимо составить и выполнить специальную программу. Составление такой программы является сложной и трудоемкой задачей, особенно при обширных моделях БД. Поэтому, большой научный и практический интерес представляет автоматизация реорганизации и реструктуризации.

В системе ПАРЕС предусматриваются два способа автоматизации реорганизации и реструктуризации БД. Первый способ заключается в использовании специальных системных утилит для реализации наиболее часто встречающихся изменений ареала. Вторым и основным способом является автоматическая генерация программ реорганизации и реструктуризации.

Реорганизация с помощью системных утилит. Как указывалось в [2], ареал БД системы ПАРЕС состоит из одного или нескольких регионов, каждый из которых характеризуется числом страниц и размером страниц. При изменении ареала эти параметры для каждого региона могут быть либо увеличены, либо уменьшены. Использование утилит реорганизации предусмотрено в системе ПАРЕС только для тех случаев, когда число страниц и размер страницы каждого региона либо увеличивается, либо остается неизменным. В таких случаях утилиты реорганизации переписывают содержание БД из старого ареала в новый по страницам, изменяя при этом необходимую системную информацию. В остальных случаях изменения ареала необходимо генерировать программу реорганизации по таким же принципам, как и при реструктуризации.

Реструктуризация с помощью сгенерированных программ. Содержание процесса реструктуризации определяется структурой существующей, исходной БД и структурой новой, целевой БД. Так как структура БД описывается ее моделью, можно говорить об исходной и целевой моделях БД. Автоматическая генерация программы реструктуризации возможна, если отображение целевой модели в исходную является однозначным.

Если язык описания модели БД является хорошо формализованным, то система может определить это отображение на основе описаний исходной и целевой модели БД. Такой моделью с хорошо формализованным описанием является, например, реляционная модель Кодда [3]. В то же время язык описания данных КОДАСИЛ [4] является неформальным, и в системах, основывающихся на использовании этой модели, от-

ображение исходной модели в целевую необходимо задавать специальным описанием изменений [5].

Описание реляционно-решетчатой модели (PPM) базы данных, по нашему мнению, хорошо формализовано, и поэтому в системе ПАРЕС отображение исходной модели в целевую определяется описаниями соответствующих моделей. Но автоматическая генерация программы реструктуризации выполняема не при всех произвольных изменениях в модели БД. Некоторые изменения модели являются недопустимыми, так как для них невозможно дать однозначную интерпретацию в БД. Для их интерпретации потребуются дополнительная информация. Анализ PPM позволяет выявить класс допустимых изменений в модели БД.

Допустимые изменения в PPM при реструктуризации БД. Об изменениях в PPM можно говорить на двух уровнях – на уровне глобальной и на уровне локальной структуры. Глобальная структура PPM представляется ориентированным графом, вершины которого соответствуют реляционным объектам (PO), а дуги – реляционным связям (PC) [1]. Локальная структура PPM определяется внутренней структурой каждого PO и соответствующего сегмента БД.

На уровне глобальной структуры при реструктуризации БД допускаются следующие изменения, при которых выполняема автоматическая генерация программ реструктуризации:

- добавление новых реляционных объектов таким образом, чтобы ни один из новых PO не являлся бы владельцем какого-то PO исходной модели;

- добавление новых реляционных связей между PO исходной модели и добавленными PO, а также между реляционными объектами исходной модели, если в ней существует ориентированный путь от владельца новой PC к подчиненному PO новой PC;

- удаление реляционных объектов, если при этом не нарушается связность графа, представляющего PPM;

- удаление реляционных связей, если при этом не нарушается связность графа;

- расширение модели таким образом, чтобы корневой PO исходной модели являлся бы унарным PO под корневым объектом целевой модели.

На уровне локальной структуры РРМ допустимы следующие изменения:

- добавление и удаление информационных полей в сегментах;
- изменение типов представления данных в информационных полях;
- перегруппировка и переупорядочение информационных полей, создание новых агрегатов данных в сегментах и исключение старых агрегатов данных.

Кроме того, для каждого реляционного объекта может быть допущено изменение способа размещения и способа упорядочения подчиненных сегментов сублинций [2]. Однако при изменении способов размещения сегментов и упорядочения их может случиться, что при выполнении сгенерированной программы происходит потеря информации. Это может быть вызвано недопустимостью дубликатов в целевой модели.

Выделенный класс допустимых изменений в РРМ позволяет осуществить генерацию программ реструктуризации даже при значительных преобразованиях структуры предметной области. Недопустимые изменения (например, слияние или расщепление РО) обычно можно реализовать путем декомпозиции и описывать как некоторую последовательность допустимых изменений. Реструктуризация в таких случаях выполняется в несколько этапов. Для этого описывается промежуточная модель БД, в некоторых случаях даже несколько промежуточных моделей, так, что целевая модель первого этапа являлась бы исходной моделью второго этапа и т.д. Изменения структуры БД на каждом этапе должны быть допустимыми для автоматической генерации программ реструктуризации. Между выполнением отдельных этапов в промежуточную БД путем обновления БД вводится дополнительная информация, необходимая для выполнения следующего этапа реструктуризации. Использование промежуточных моделей и поэтапное выполнение реструктуризации еще значительно расширяют возможности применения автоматической генерации программ реструктуризации в системе ПАРЕС.

Независимость прикладных программ от реструктуризации.
Целью реструктуризации является, с одной стороны, сохранение имеющейся информации о предметной области при ее изме-

нении и, с другой стороны, сохранение доступа к данным для созданных прикладных программ. Независимость прикладных программ от изменений структуры БД не может быть абсолютной, так как для программ, составленных по старой модели БД, могут оказаться недоступными объекты, связи и поля данных, которые удалены из модели БД.

Добавление новых РО и РС, а также новых полей или агрегатов данных не потребует никакой переделки существующих программ. Это же можно сказать об изменении способа упорядочения субъюнкций. Отмена прямого доступа к какому-то объекту потребует корректировки программ, использующих прямой доступ к этому объекту.

После удаления РО и РС, а также полей агрегатов данных из модели БД, больше нельзя пользоваться программами, использующими эти объекты, связи или поля. Эти программы уже неработоспособны, их придется изменить или написать новые программы, соответствующие новой модели БД.

При реорганизации БД обеспечивается полная независимость прикладных программ от изменения ареала. Все прикладные программы останутся после реорганизации работоспособными.

Генерация и выполнение программ реструктуризации.

Генерация программы реструктуризации в системе ПАРЕС проводится автоматически на основе описаний исходной и целевой моделей. При этом проверяется совместимость моделей. Если необходимые преобразования оказываются допустимыми, то генерируется текст программы реструктуризации на языке ДАМАЛ [6]. Алгоритм перенесения сегментов из одной базы в другую реализуется средствами комплексирования программ - корутинами. Программа транслируется и выполняется обычным образом, как и другие прикладные программы. В качестве подпрограмм программы реструктуризации могут использовать системные утилиты реорганизации.

Выполнение программ реструктуризации начинается с переноса корневого сегмента в новую БД, далее по типам переносятся сегменты, соответствующие унарным РО под корнем и т.д. При записи сегмента в новую базу в сегментах старой базы запоминается адрес сегмента в новой базе. Это позволяет в ходе выполнения программы реструктуризации обеспе-

чить прямой доступ ко всем сегментам новой базы независимо от способа размещения сегмента. Такой прямой доступ необходим для фиксации владельцев субъединиц при перенесении подчиненных сегментов.

Закключение. Средства реорганизации и реструктуризации базы данных позволяют значительно повысить "живучесть" создаваемых при помощи системы ПАРЕС банков данных и информационных систем. Автоматическая генерация программ реструктуризации освобождает пользователей БД от трудоемкой и рутинной задачи составления программ перезагрузки БД и позволяет им сосредоточить свое внимание на решении более творческих задач.

Л и т е р а т у р а

1. Крахт В.А. От концептуальной модели предметной области до структуры базы данных. См. наст. сб., с. 5.
2. Василь М.Х. Реляционно-решетчатая структура базы данных. См. наст. сб., с. 67.
3. Дейт К. Введение в системы баз данных. - М., Наука, 1980. 464 с.
4. Язык описания данных КОДАСИЛ. М., Статистика, 1981, 183 с.
5. Щербakov Н.В. Система автоматизации программирования процедур реорганизации баз данных. - Л., ЛПИ, 1982. 16 с.
6. Крахт В.А., Эйвак Ю.Э. Алгоритмический язык манипулирования данными DAMAL. Таллин, ТПИ, 1982. 118 с.

M. Vassil

Relational Latticework Structured Database

Reorganisation and Restructuring

Summary

Reorganisation and restructuring concepts for relational latticework database are defined. Facilities for automation of restructuring processes are described. Assumptions of such restructuring and the class of admissible modifications of relational latticework model are discussed.

СТРУКТУРИЗАЦИЯ И ОПИСАНИЕ КОММУНИКАЦИОННЫХ
ДАННЫХ В СИСТЕМЕ ПАРЕС

Введение. Создание и использование любой информационной системы с базой данных (БД) связано с процессами коммуникации (обмена) информации между БД и другими внешними – коммуникационными – наборами данных. Коммуникацию можно рассматривать как процесс переноса данных из одного набора в другой – из какого-то набора (файла) коммуникационных данных (КД) в БД или наоборот. Но, так как структуры этих наборов разные, то значит, что перенос данных всегда связан с их переструктурированием. Процесс коммуникации реализуется соответствующими программами.

При использовании систем управления базами данных (СУБД) с включающим языком программы коммуникации составляются на базовом языке системы (напр., ПЛ/I, КОБОЛ и др.). В состав программ включаются операторы языка манипулирования данными (ЯМД). В этих же программах описываются и вводимые в базу или выводимые из нее КД. Но структура самой БД описывается на другом, предназначенном для этого языке описания данных (ЯОД), используемой СУБД.

Достоинством такого подхода является возможность составления описания данных (КД) и программ на общеизвестных языках. Но при этом теряется независимость между описываемыми КД и программой их обработки.

В отличие от таких систем (СУБД) в системе ПАРЕС описания КД и составление программ их обработки (коммуникации) разделены друг от друга. КД описывается на отдельном, предусмотренном для этого входном языке системы – на языке описания КД (ЯОКД). Программы обработки составляются на алгоритмическом языке манипулирования данными ДАМАЛ. Описание КД связывается с программами обработки во

время выполнения этих программ. Тем самым обеспечивается независимость программ обработки от обрабатываемых КД. Описание их может быть изменено в определенных пределах без необходимости изменения (и перетрансляции) программ и наоборот.

Целью данной работы является представление общих принципов структурирования и описания КД. Составление самих программ коммуникации в этой работе не рассматривается. Эта сторона вопроса освещается, например, в [1].

Виды коммуникационных данных. В системе ПАРЕС можно выделить три вида КД – транзакционные данные (ТД), печатные данные (ПД) и совместные данные (СД).

ТД – это данные, предназначенные для загрузки и обновления БД. Они выступают относительно прикладной программы (ПП) в роли вводимых данных. Предполагается, что они образуются на основе входных документов.

ПД – это результат обработки данных, извлеченных из базы и подготовленных для вывода на АЦПУ. Они выступают в роли выводимых данных и представляются в виде выходных документов.

СД – это данные, представленные в виде стандартного файла данных на внешнем накопителе ЭВМ. Они могут выступать в роли вводимых и выводимых данных и являются связывающим звеном между данной системой и другими системами обработки информации.

Общая структура и основы описания. Все КД представляются в системе ПАРЕС как файлы данных, составленные из одиновых логических записей. Запись ТД рассматривается как образ входного документа, запись ПД – как образ выходного документа, а запись СД – как запись "обычного" файла данных на внешних накопителях ЭВМ.

Структура таких записей является объектом описания. Описание этой структуры понимается как описание КД соответствующего вида. Записи ТД называются в системе условно листами, а записи ПД – страницами.

Описаниями данных определяются их структура, типы элементов структуры, имена всей структуры и ее компонентов, а иногда и некоторые другие атрибуты.

КД всех видов представляются как древовидные структуры данных. Описание их в системе ПАРЕС основывается на принципах представления древовидных структур данных, предложенных М. Джексоном [2]. В соответствии с этим выделяются три вида составных компонентов структуры – последовательность, выбор, повторение – и один простой вид компонента – атом. В системе ПАРЕС указанные составные компоненты именуются коллекцией (COL), селекцией (SEL) и секвенцией (SEQ) соответственно. Они представляются на входных языках системы следующим образом:

S : COLLECTION OF	S : SELECTION OF
A : ;	A : ;
B : ;	B : ;
EOD S;	EOD S;
S : SEQUENCE (k) OF	
A : ;	
EOD S;	

где S – описываемый составной компонент, A, B – его составляющие компоненты, которые сами могут быть составными или простыми, а k – число (коэффициент) повторений секвенции. (EOD – end of data).

Рассмотренные выше схемы описания распространяются на все виды КД. Но для каждого из них предусматривается возможность отражения специфики данного вида КД. Более подробно рассматриваются принципы структурирования и описания ТД и ПД как наиболее специфических видов КД в системе ПАРЕС. Принципы структурирования СД мало отличаются от соответствующих принципов в других системах (ПЛ/I, КОБОЛ).

Компоненты и описание ТД. Основой описания структуры ТД является структура информации на входном документе, с которого предусматривается ввод данных в БД. В соответствии с этим выбираются и средства описания ТД на языке описания КД. Для описания ТД требуются только два вида составных компонентов структуры – коллекция и секвенция. Селекция в данном случае не имеет практического смысла.

В качестве атома структуры ТД выделяются два основных типа проблемных элементов – символьный для представления текстовой информации и цифровой – для представления числовых данных во внешней (символьной) форме. Они представляют собой описатели типа C и F соответственно, причем различаются несколькими форматами:

- $C(w)$ - строка символов длиной в точности w знаков,
 $C(<w)$ - то же длиной меньше чем w знаков,
 $F(w)$ - целое десятичное число длиной в точности w знаков,
 $F(<w)$ - то же длиной меньше чем w знаков,
 $F(w,d)$ - дробное десятичное число длиной в точности w знаков с дробной частью длиной d знаков,
 $F(<w,d)$ - то же длиной меньше чем w знаков.

Кроме проблемных элементов при описании ТД используется еще один вспомогательный элемент (SEP). При помощи этого элемента указываются специальные символы, выбранные пользователем системы для разделения отдельных структурных единиц ТД в общем потоке символов, представляющих эти данные.

Для иллюстрации возможностей ЯОКД при описании ТД рассмотрим примеры.

Пусть описываемые ТД представляют собой последовательность (файл) записей, каждая из которых может быть рассмотрена как вектор данных, состоящий из трех компонентов

номер	название	кол-во
-------	----------	--------

Такая структура ТД описывается на ЯОКД следующим образом:

```

DECLARE LEAF SCHEMA SI1 : SET OF
  PRIM1 : COL OF
    NOMER : F(6) ;
    NAZV : C(<21); SEP ' , ' ;
    KOLVO : F(<4); SEP ' ; ' ;
  EOD PRIM1; SEP ' + ' ;
EOS SI1;
  
```

С учетом указанных разделителей одна запись ТД представляется в виде

246725ИСКРА, 35, +

В качестве другого примера рассмотрим случай, когда запись входного файла имеет структуру с повторяющейся группой данных:

поставщик	деталь 1	кол-во 1
	деталь 2	кол-во 2
	деталь 3	кол-во 3
	деталь 4	кол-во 4

Соответствующая структура описывается так:

```
DECLARE LEAF SCHEMA SI2 : SET OF
  PRIM2 : COL OF
    POST : C(< 21); SEP '/';
    TAB : SEQ (< 5) OF
      IZD : COL OF
        DET : C(< 4); SEP ',';
        KOL : F(< 4); SEP ',';
      EOD IZD;
    EOD TAB; SEP ' ';
  EOD PRIM2; SEP ' + ';
EOS SI2;
```

Эта схема предполагает представление одной записи ТД, например, в виде следующего потока символов

КВАНТ/БОЛТ, 30, ГАЙКА, 25, ; +

Текст описания ТД, составленный на ЯОКД, обрабатывается соответствующим компонентом системы - транслятором описания ТД. В результате этого образуется внутрисистемное описание ТД. Вместе с этим выдается и макет подготовки данных.

Внутрисистемное представление описания ТД является основой для автономной верификации (проверки) правильности подготовки ТД. Для этого в системе ПАРЕС предусматривается специальный компонент - верификатор ТД. В ходе такой верификации выявляются все структурные ошибки, допущенные при подготовке данных - отсутствие элементов, несоответствие их объявленному типу, несоблюдение формата.

Схемы описания ТД используются программами загрузки и обновления БД для определения доступа к отдельным элементам вводимых данных, определения типа элементов с целью автоматического преобразования формы представления их и т.д. В этом заключается основное значение схем описания ТД. Применение такой схемы в этих целях иллюстрируется в работе [2].

Компоненты и описание ПД. Основой описания ПД является требуемая структура информации на выходном документе, выделяемом в результате обработки данных, хранимых в БД. Для

описания ПД в общем случае требуются все три вида составных компонентов структуры – коллекция, селекция и секвенция. Но ввиду того, что запись (логическая страница) ПД представляет собой двухмерную структуру данных – выходной документ, оформленный на бумаге АЦПУ или представляемый на экране дисплея, – требуется усовершенствование средств структуризации данных таким образом, чтобы оказалось возможным представление таких двухмерных структур. С этой целью выделяют два вида коллекции данных – столбец (COLUMN, SIM) и строка (LINE). Под столбцом понимается коллекция, компоненты которой размещаются на странице ПД один под другим, а под строкой – коллекция, компоненты которой размещаются на странице ПД один рядом с другим. С учетом этого секвенциями могут быть представлены повторения строк или повторения столбцов, а селекциями – альтернативные варианты представления строк и столбцов.

В качестве атома структуры ПД выделяются такие же два основных типа проблемных элементов, как и при описании ТД. Но здесь имеют место только три формата их представления: $S(w)$, $F(w)$, $F(w, d)$.

Особенностью ПД является то, что при описании их большое значение имеет возможность дополнения выводимых проблемных данных различными редакционными элементами – прибавление текстовых констант в виде заголовий таблиц, введение горизонтальных и вертикальных линий и т.д.

Для этого при описании ПД на ЯОКД предусматривается ряд специальных редакционных элементов.

Для иллюстрации возможностей ЯОКД при описании ПД рассмотрим пример. Допустим, что требуется описание выходного документа, имеющего следующий формат:

СПИСОК ДЕТАЛЕЙ

название	количество	цена	сумма
----------	------------	------	-------

данные из БД

Данная структура описывается в виде:

```
DECLARE PAGE SCHEMA SP : SET OF
  VDOK : CIM OF
    AA : LINE OF
      STEP 13;
      'СПИСОК ДЕТАЛЕЙ'; EOD AA;
    SKIP 3;
    LINE ' = ';
    BB : LINE OF
      'НАЗВАНИЕ';
      STEP 7;
      'КОЛ-ВО';
      STEP 4;
      'ЦЕНА';
      STEP 6;
      'СУММА'; EOD BB;
    LINE ' - ';
    TAB : SEQ (30) OF
      STR : LINE OF
        NAZV : C (12);
        STEP 4;
        KOL : F (4);
        STEP 4;
        CENA : F (6,2);
        STEP 3;
        SUM : F (8,2); EOD STR;
      EOD TAB;
    EOD VDOK;
  EOS SP;
```

Текст описания ПД, составленный на ЯОКД, обрабатывается соответствующим компонентом системы – транслятором описания ПД, в результате чего образуется его внутрисистемный эквивалент. Вместе с этим выдается и макет выходного документа, на основе чего можно легко исправить допущенные при описании документа ошибки или ввести коррективы.

Внутреннее описание ПД используется прикладными программами для оформления выводимых данных согласно описанию ПД. Благодаря этому существенно упрощается составление про-

грамм обработки данных, в которых предусматривается вывод результатов на АЦПУ. Пример использования описания ПД для оформления результатов обработки приведен в [2].

Л и т е р а т у р а

1. **Д а к с о н М.А.** Principles of Program Design. London - New York - San Francisco, Academic Press, 1975.

2. **Р о о т а л у Э.П.** Разработка систем обработки данных с базой данных реляционно-решетчатой структуры. См. наст. сб., с. 99.

T. Ordlik

Communication Data Structuring and Description in the Software System PARES

Summary

General concepts of information exchange between database and input-output data files of information processing system implemented by means of the software system PARES are discussed. Input-output data structure and description in data definition language of the system are presented.

РАЗРАБОТКА СИСТЕМ ОБРАБОТКИ ДАННЫХ С БАЗОЙ ДАННЫХ
РЕЛЯЦИОННО-РЕШЕТЧАТОЙ СТРУКТУРЫ

Введение. Проектирование прикладных систем обработки данных с интегрированной базой данных (БД) является сложной проблемой. Оно охватывает проектирование БД и разработку прикладных программ (ПП) обработки, предназначенных для актуализации базы, воспроизведения данных из базы и решения различных проблемных задач. При этом нужно учитывать возможности применяемых средств – систем управления базами данных (СУБД) и систем программирования.

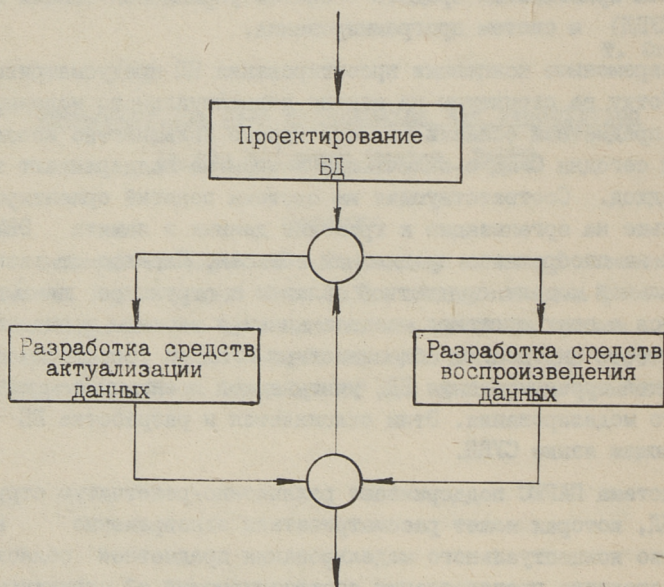
Современные концепции проектирования БД предусматривают разработку ее структуры на основе концептуального моделирования предметной области [1, 2]. Однако большинство используемых сегодня СУБД с включающим языком не поддерживают такой подход. Соответствующая им система понятий ориентирована больше на организацию и хранение данных в памяти ЭВМ, нежели на отображение предметной области. Переход от концептуальной модели предметной области к структуре данных является в этих системах неоднозначным и сложным процессом. Практически ни одна из общеизвестных СУБД не сопровождается методикой проектирования БД, учитывающей принципы концептуального моделирования. Этим осложняется и разработка ПП на включающем языке СУБД.

Система ПАРЕС поддерживает реляционно-решетчатую структуру БД, которая может рассматриваться одновременно как средство концептуального моделирования предметной области и как средство представления соответствующей ей структуры базы данных [3]. На основе этого возможно развитие единой методики проектирования БД, охватывающей основные этапы проектирования, начиная с построения концептуальной модели предметной области вплоть до разработки структуры БД. Осно-

вы такой методики изложены в [4]. Дополнительно к этому в [5] изложены принципы логического конструирования программ обработки данных указанной структуры.

В свете этого в данной работе рассматриваются общие вопросы и ход проектирования системы обработки данных с БД реляционно-решетчатой структуры на основе использования системы ПАРЕС в качестве средства реализации.

Основные этапы. При создании системы обработки данных (СОД) с интегрированной БД можно выделить две стадии - анализ (исследование) предметной области и проектирование компонентов СОД-БД и ПП. Целью анализа предметной области является получение достаточно полного и глубокого представления о данной области, определение целей и задач обработки информации, выявление пользователей системы и необходимой им информации и т.д. Целью проектирования СОД является разработка проекта и прикладного программного обеспечения системы.



Фиг. 1. Ход разработки СОД.

Разработка системы является итеративным процессом, в котором можно выделить три основных этапа (см. фиг. I):

- проектирование базы данных;
- разработка средств актуализации данных;
- разработка средств воспроизведения и проблемной обработки данных.

Каждый этап такого процесса связан с анализом предметной области, с проектированием программных средств, с их реализацией и отладкой. Процесс заканчивается внедрением программных средств.

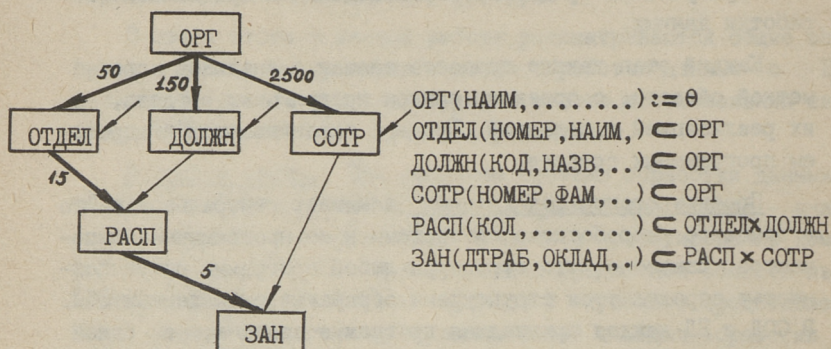
Проектирование базы данных занимает центральное место в процессе разработки всей системы и ее программных компонентов. Известно, что структура любой программы почти полностью определяется структурами обрабатываемых данных [6]. В СОД с БД каждая прикладная программа связана, с одной стороны, с БД, а с другой стороны, с коммуникационными данными (КД), вводимыми в базу или выводимыми из нее. Если СОД действительно строится на основе интегрированной БД, то, очевидно, структура базы оказывает влияние на структуру всех ПП данной СОД. Отсюда вытекает решающее значение структуры БД с точки зрения сложности и трудоемкости разработки всей СОД и проблема выбора оптимальной в этом смысле структуры базы. На наш взгляд, такую структуру можно определить только на основе концептуального моделирования предметной области.

Отправной точкой концептуального моделирования является то, что основные задачи проблемной обработки данных основываются на определенной системе понятий данной области, которая мало изменяется при переходе от одной задачи к другой и отражает основное содержание этой области. Оно заключается в выявлении этой системы понятий и в ее формализации, в результате чего и образуется концептуальная модель предметной области. На основе этой модели определяется информационная структура БД.

Внешнее проектирование БД, в основном, и заключается в составлении указанной концептуальной модели. Построение и представление такой модели в виде реляционной решетки можно проиллюстрировать конкретным примером.

Допустим, что перед нами стоит задача построения БД системы учета кадров в некоторой организации. Введем в нашу модель в качестве ее корня [3] объект ОРГ (организация).

который представляет данную предметную область в целом (см. фиг. 2).



Фиг. 2. Концептуальная модель системы учета кадров.

С точки зрения системы учета кадров в организации в качестве ее основных объектов, очевидно, нужно выделить структурные подразделения – отделы (ОТДЕЛ), должности (ДОЛЖН) и сотрудников (СОТР). Они станут унарными Р-отношениями модели под корневым объектом. Между объектом ОТДЕЛ и ДОЛЖН существует некоторое бинарное Р-отношение, которое можно назвать РАСП (штатное расписание) и которое представляет распределение должностных мест по отделам. Если учесть то, что разные сотрудники могут в течение времени занимать различные должностные места в подразделениях данной организации, и необходимость отражения такого передвижения сотрудников в БД, то, очевидно, что между объектами РАСП и СОТР существует бинарное отношение ЗАН (занятость). Оно представляет занятость должностных мест по штатному расписанию сотрудниками. Атрибуты этого Р-отношения характеризуют условия приема сотрудников на работу. В результате такого рассуждения и образуется реляционно-решетчатая модель, приведенная на фиг. 2.

Данная модель рассматривается одновременно и как модель структуры БД. В этой интерпретации основные элементы концептуальной модели (реляционные объекты) понимаются как сегменты базы соответствующего типа, а связывающие элементы в концептуальной модели (реляционные связи) – как субъюнкции (связанные цепи) сегментов базы [3].

Внутреннее проектирование БД. Целью внутреннего проектирования является спецификация глобальной структуры и ее основных элементов — сегментов и субънкций.

Спецификация глобальной структуры заключается в выборе точек входа (точек непосредственного доступа) и главных собственных связей (субънкций), а также коэффициентов подчинения для всех главных связей. Выбор точек входа связан с определением путей доступа к сегментам базы. Эти вопросы решаются на основе содержательного анализа предметной области. В нашей модели (см. фиг. 2) точки непосредственного доступа отмечены боковыми стрелками, а главные собственные связи — жирными линиями. Для каждого из них указан коэффициент подчинения.

Спецификация сегмента заключается в определении элементов данных, представляющих атрибуты отображаемого Р-отношения, информационных характеристик этих элементов (тип, длина поля) и в выборе ключей идентификации. Последние используются в качестве ключа размещения, если данный сегмент является сегментом с непосредственным доступом, и ключа сортировки, если данный сегмент является подчиненным упорядоченных субънкций.

Спецификация субънкции заключается в выборе способа организации субънкции (упорядоченная или неупорядоченная) и порядка следования подчиненных сегментов субънкции, если по организации она объявлена упорядоченной.

Внутреннее проектирование базы заканчивается описанием структуры БД на соответствующем входном языке системы ПАРЕС. Описание состоит из двух частей — описания ареала и модели БД [7].

Описание ареала могло бы выглядеть в данном случае так:

```
DECLARE AREA KADR: MEDIUM FOR PERSON,  
SET OF  
KADR1: REGION OF 350 PAGES BY 3250 BYTES;  
EOS KADR;
```


а описание модели следующим образом:

```
DECLARE MODEL PERSON: SET OF
  ORG: COLLECTION OF
    NAIM: CHAR(32); EOD,
    ROOT OF MODEL,
    LOCATED IN KADR1, EOR ORG;
  OTDEL: COLLECTION OF
    NOMER: CHAR(4);
    NAIM: CHAR(24); EOD,
    RELATED TO ORG BY SORT(NOMER) WITH COEFF 50,
    LOCATED BY HASH(NOMER) IN KADR1, EOR OTDEL;
  -----
  RASP: COLLECTION OF
    NDOL: CHAR(3);
    NOTD: CHAR(4);
    KOL: INTEGER; EOD,
    RELATED TO OTDEL BY SORT(NDOL) WITH COEFF 15
    AND TO DOLZN BY SORT(NOTD),
    LOCATED BY HEAD, EOR RASP;
  ZAN: COLLECTION OF
    DTRAB: CHAR(8);
    OKLAD: DEC(5,2);
    ----- EOD,
    RELATED TO RASP WITH COEFF 5,
    AND TO SOTR BY SORT(DTRAB),
    LOCATED BY HEAD, EOR ZAN;
  EOS PERSON;
```

Разработка средств актуализации БД. Целью данного этапа разработки системы является разработка ПП для загрузки, корректировки и обновления базы данных на алгоритмическом языке манипулирования данными DAMAL системы ПАРЕС [8].

Во всех БД всегда можно выделить относительно постоянную часть и активно изменяющуюся часть. В сегментах постоянной части БД обычно хранится нормативная информация о предметной области. Содержание этой части базы определяется при начальной загрузке и обновляется очень редко. В нашей модели, очевидно, такими сегментами являются ОРГ, ОТДЕЛ, ДОЛЖН и РАСП. Основные изменения состояния предметной области отражаются в изменениях сегментов активной части БД, которые поэтому подвергаются постоянному обновлению. В ходе такого обновления базы прибавляются новые сегменты, корректируется содержание компонентов существующих сегментов базы, удаляются ненужные сегменты, изменяются владельцы подчиненных субъонкций. В нашей модели к активной части базы относятся объекты СОТР и ЗАН. Во вносимых в эти сегменты изменениях отражается динамика кадровой деятельности органи-

зации (прием и увольнение сотрудников, переводы на другие должности или в другие подразделения и т.д.).

Ввод данных в базу можно осуществлять со стандартного входного файла (с экрана дисплея или с перфокарт) или с файла транзакционных данных (ТД). Выбор того или иного способа ввода зависит в основном от объема и сложности структуры вводимых в базу данных. При небольших объемах и несложной структуре исходной информации как для начальной загрузки, так и для обновления сегментов БД используется первый способ ввода данных.

Допустим, например, что требуется изменение фамилий некоторых сотрудников. Возможная программа для решения этой задачи выглядит на языке DAMAL следующим образом:

```
PROGRAM ZAG13: BEGIN
  DECLARE STR: COLLECTION OF
    FAM: CHAR(24); EOD STR;
  OPEN DTBS KADR(PERSON) REGS(KADR1) FOR UPDATE;
  GET(SOTR.NOMER, STR.FAM);
  P1: WHILE SOTR.NOMER = '*'
    DO P2: IF EX ANY SOTR(NOMER)
      THEN HOLD SOTR;
        SET SOTR.FAM=STR.FAM;
        REPLACE SOTR;
      END P2;
    GET(SOTR.NOMER, STR.FAM);
  END P1;
  CLOSE DTBS KADR;
END ZAG13;
```

Данная программа предполагает, что исходные данные (табельный номер сотрудника и новая фамилия) вводятся при помощи оператора GET с экрана дисплея или с перфокарт. В ходе выполнения программы по заданному ключу прямого доступа (номер сотрудника) осуществляется поиск сегмента SOTR и замена значений информационных полей в соответствующем экземпляре сегмента. Процесс обновления (P2) выполняется повторно до поступления признака конца исходных данных (знак *).

Актуализация БД на основе ТД состоит из ряда шагов - описания исходных данных, образования набора ТД и составления программы актуализации базы (ввода данных в базу). Проиллюстрируем это на примере.

Допустим, что требуется обновление БД данными о переводах сотрудников из одного отдела (подразделения) в другой. Для решения этой задачи требуются следующие данные о переводимых сотрудниках - номер сотрудника (НОМЕР), код должности (КДОЛ), зарплата (ЗАРПЛ), код отдела (КОТД) и дата перевода (ДАТА). Они описываются в качестве записи файла ТД следующим образом:

```
DECLARE LEAF SCHEMA KAD31; SET OF
```

```
  ZAPIS: COLLECTION OF
```

```
    NОMER: C(5), SEP ',';
```

```
    КДОЛ: C(3), SEP ',';
```

```
    ЗАРПЛ: F(6,2), INTERVAL('70'-'500'), SEP ',';
```

```
    КОТД: C(4), SEP ',';
```

```
    ДАТА: C(8), SEP ',';
```

```
  EOD ZAPIS; SEP '+';
```

```
EOS KAD31;
```

Образование набора ТД заключается в подготовке соответствующих исходных данных, в их верификации, (проверке) на основе составленного описания [9] и в формировании файла проверенных данных - файла ТД, готовых к вводу в БД при помощи составляемой программы актуализации базы.

В программе ввода данных описывается алгоритм процесса переноса их из файла ТД в БД:


```

-----
OPEN TRFL AAA(KAD31);
P1: FOR ALL KAD31 DO
    SET OTDEL.NOMER=KAD31.KOTD;
    P2: IF EX ANY OTDEL(NOMER)
        THEN
            SET DOLZN.NOMER=KAD31.KDOL;
            P3: IF EX ANY DOLZN(NOMER)
                THEN
                    P4: IF EX(RASP IN OTDEL)
                        (HEAD(RASP IN DOLZN)=CUR(DOLZN))
                            THEN
                                SET SOTR.NOMER=KAD31.NOMER;
                                P5: IF EX ANY SOTR(NOMER)
                                    THEN
                                        P6: IF EX(ZAN IN RASP)
                                            (HEAD(ZAN IN SOTR)=CUR(SOTR))
                                                THEN
                                                    PUT ('ЗАПИСЬ СЪЕДИТЕЛ'ET')
                                                    ELSE
                                                        CREATE ZAN;
                                                        SET ZAN.DTRAB=KAD31.DATA;
                                                        SET ZAN.OKLAD=KAD31.ZARPL;
                                                        INSERT ZAN;
                                                    END P6;
                                                END P5;
                                            END P4;
                                        END P3;
                                    END P2;
                                END P1;
                            END P1;
                        END P1;
                    END P1;
                END P1;
            END P1;
        END P1;
    END P1;
END P1;
-----
CLOSE TRFL AAA;
-----

```


Программа предполагает, что проверенные исходные данные представлены в виде файла ТД под именем ААА, а их описание - схемой данных КАД31. Для всех записей файла ААА выполняется процесс Р2. На каждом шагу этого процесса на основе входной информации осуществляется поиск сегментов ОТДЕЛ, ДОЛЖН и СОТР и извлечение из базы сегментов РАСП и ЗАН. Если в БД уже существует сегмент ЗАН, удовлетворяющий условию извлечения, то выдается соответствующее сообщение (ЗАПИСЬ СУЩЕСТВУЕТ). В противном случае создается новый сегмент.

Разработка средств воспроизведения данных. Целью данного этапа является разработка ПП для получения справок, генерации отчетов и решения различных проблемных задач (анализа, прогнозирования, планирования и т.д.) на ЯМД DAMAL [8]. Круг решаемых задач воспроизведения информации почти полностью определяется структурой и информационным содержанием БД. Опыт разработки СОД с интегрированной БД показывает, что, если структура БД действительно отражает концептуальную структуру предметной области данной базы, практически все первоначально предусмотренные и заранее непредвиденные задачи воспроизведения данных решаются на основе такой базы всегда сравнительно просто.

При использовании системы ПАРЕС результаты обработки данных могут быть выведены либо в виде стандартного выходного файла (на экран дисплея или на АЦЦУ), либо в виде файла выходных документов, отредактированных средствами системы требуемым образом, т.е. в виде т.н. файла ПД [9]. Выбор того или иного способа представления выходной информации зависит, в основном, от объема и структуры выводимой информации и требований к ее оформлению.

Воспроизведение данных с выводом результатов в стандартный выходной файл используется для получения ответов на простые запросы формирования различных справок и т.д. В таком случае задача сводится к составлению запроса данных на языке DAMAL. Например, для получения списка лиц, принятых на работу, начиная с 1983 года, составляется следующая простая программа (запрос):


```

-----
P1: FOR ALL (SOTR IN ORG)
    SUCH EX (ZAN IN SOTR)(ZAN.DTRAB > = '19830101')
    DO
        PUT (SOTR.FAM,ZAN.DTRAB);
    END P1;
-----

```

В результате выполнения этой программы на экран дисплея (или на АЦПУ) выдается список лиц, удовлетворяющих условию поиска, с указанием фамилии сотрудника и даты приема его на работу.

Подготовка средств воспроизведения данных с формированием файла отредактированных выходных документов состоит из двух шагов – описания выходного документа и составления самой программы воспроизведения. Описание выходного документа определяет структуру выводимых данных, форматы выводимых элементов данных, а также реквизиты оформления документа (заголовки и т.д.).

Допустим, например, что требуется составление отчета о заполненности штатного расписания и пусть результаты необходимо представить в следующем виде:

ЗАПОЛНЕННОСТЬ ШТАТНОГО РАСПИСАНИЯ

ПОДРАЗДЕЛЕНИЕ: ПО СОСТОЯНИЮ

НАИМЕНОВАНИЕ ДОЛЖНОСТИ	ПО ПЛАНУ	ЗАПОЛНЕНО
.....		
.....		

и т.д.

Описание такого документа на языке ЯОКД [9] системы ПАРЕС выглядит следующим образом:

```
DECLARE HALF PAGE SCHEMA VYD1: SET OF
      DOKUM: CIM OF
      R1: LINE OF
            STEP 15;
            'ЗАПОЛНЕННОСТЬ ШТАТНОГО РАСПИСАНИЯ'; EOD R1;
      R2: LINE OF
            'ПОДРАЗДЕЛЕНИЕ'; OTDEL: C(24); STEP 2;
            'ПО СОСТОЯНИЮ'; DATA: C(8); EOD R2;
      SKIP 2;
      LINE '-';
      R3: LINE OF
            'НАИМЕНОВАНИЕ ДОЛЖНОСТИ'; STEP 5;
            'ПО ПЛАНУ ЗАПОЛНЕНО'; EOD R3;

      LINE '-';
      TABL: SEQ (40) OF
            STROKA: LINE OF
                  NAIM: C(32); MEST: F(6);
                  STEP 5; FAKT: F(6); EOD STROKA;
      EOD TABL;

      EOD DOKUM;
      EOS VYD1;
```

Прикладная программа на языке DAMAL для решения данной задачи выглядит так:


```

PROGRAM DOK1: BEGIN
  DECLARE STR: COLLECTION OF
    ZAN: INTEGER;
    DATA: CHAR(8); EOD STR;

  - - - - -
  GET(STR.DATA);
  OPEN PRFL BBB(VYD1);
  P1: FOR ALL (OTDEL IN ORG) DO
    CREATE VYD1;
    SET VYD1.OTDEL=OTDEL.NAIM;
    SET VYD1.DATA=STR.DATA;
    P2: FOR ALL (RASP IN OTDEL) DO
      FIND HEAD (RASP IN DOLZN);
      CREATE VYD1.STROKA;
      P3: FOR ALL (ZAN IN RASP)
        SUCH ZAN.UHOD > STR.DATA
        DO
          SET STR.ZAN=SUM(STR.ZAN,1);
        END P3;
      SET VYD1.NAIM=DOLZN.NAIM;
      SET VYD1.MEST=RASF.KOL;
      SET VYD1.FAKT=STR.ZAN;
      INSERT VYD1.STROKA;
    END P2;
    INSERT VYD1;
  END P1;
  CLOSE PRFL BBB;

  - - - - -
  END DOK1;

```

Результатные данные выводятся в файл печатных данных (ПД) под именем BBB, описанных схемой VYD1. Отчет оформляется по отдельным отделам организации. Оператором GET вводится дата учета. В программе использована арифметическая функция SUM для подсчета заполненных должностных мест в подразделении.

Подготовка программ ввода/вывода данных в режиме диалога при использовании системы ПАРЕС мало отличается от подготовки их для пакетного режима работы. Для этого в общем случае требуется включение в программы только допол-

нительных операторов, обеспечивающих ввод/вывод сообщений на экран дисплея для ведения диалога с пользователем.

Л и т е р а т у р а

1. ANSI/X3/SPARC. Study group on data base management systems: Interim Report FDT. Bulletin of ACM SIGMOD - 1975, vol. 7, N 2, p. 1-140.

2. D a v e n p o r t R.A. Data analysis for database design. - The Australian Computer Journal, 1978, vol. 10, N 4, p. 122-137.

3. К р а х т В.А. От концептуальной модели предметной области до структуры базы данных. См. наст. сб., с. 5.

4. К р а х т В.А., Р о о т а л у Э.П. Проектирование баз данных на основе реляционно-решетчатой концептуальной модели предметной области. Управляющие системы и машины, 4, 1981, с. 22-28.

5. К р а х т В.А. Логическое конструирование программ обработки данных реляционно-решетчатой структуры. См. наст. сб., с. 29.

6. J a c k s o n M.A. Principles of program design. London - New York - San Francisco, Academic Press, 1975, p. 298.

7. В а с с и л ь М.Х. Реляционно-решетчатая структура базы данных. См. наст. сб., с. 83.

8. К р а х т В.А., Э й в а к Ю.Э. Алгоритмический язык манипулирования данными DAMAL. Таллин, 1982. 118 с.

9. О р д л и к Т.В. Структуризация и описание коммуникационных данных в системе ПАРЕС. См. наст. сб., с. 91.

E. Rootalu

Design of Information Processing Systems with Relational Latticework Database

Summary

General problems and design techniques of information processing system with relational latticework database are considered. Software system PARES as a tool for system components implementation is used.



EESTI AKADEEMILINE RAAMATUKOGU



1 0200 00089778 9

Цена 85 коп.