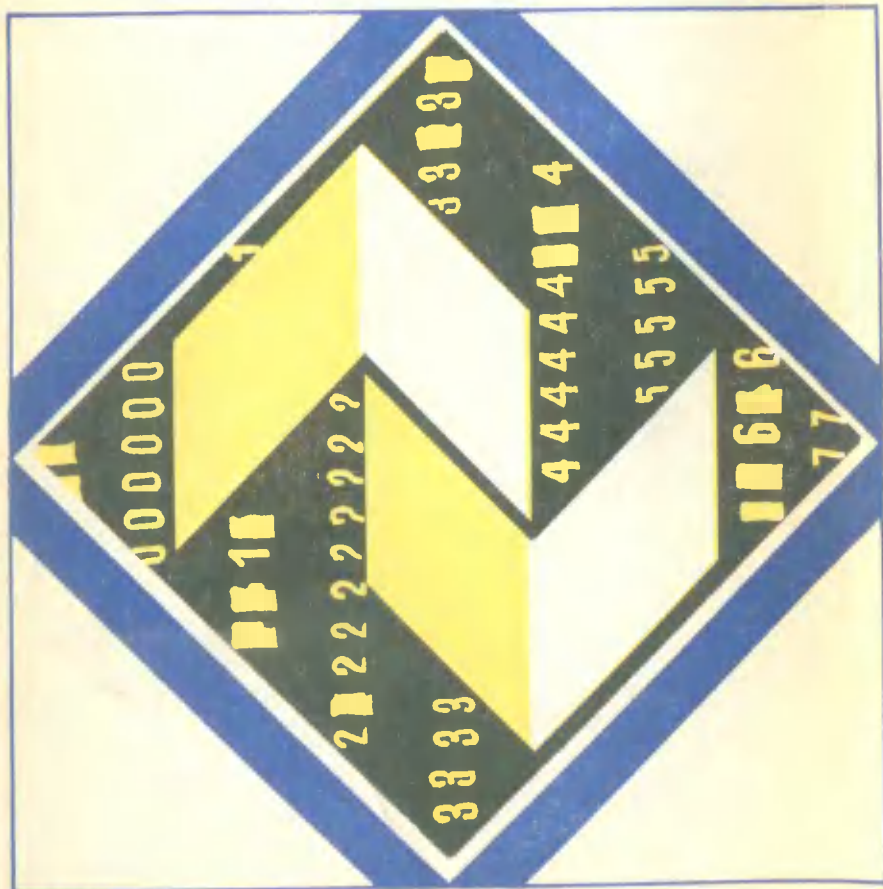


11 коп.

У23.96

Индекс 70067



ЗНАНИЕ

НОВОЕ
В ЖИЗНИ,
НАУКЕ,
ТЕХНИКЕ

СЕРИЯ
ТЕХНИКА

5'79

Н. И. Дагаев
Б. М. Рудзицкий
БАНКИ
ДАННЫХ



**НОВОЕ
В ЖИЗНИ,
НАУКЕ,
ТЕХНИКЕ**

**Н. И. Дагаев,
Б. М. Рудзицкий**

**Серия
«Техника»
№ 5, 1979 г.**

БАНКИ ДАННЫХ

**Издается
ежемесячно
в 1961 г.**

**Издательство
«Знание»
Москва
1979**

Николай Исаевич ДАГАЕВ,
Борис Михайлович РУДЗИЦКИЙ

БАНКИ ДАННЫХ

Гл. отраслевой редактор В. П. Демьянов. Редактор Г. И. Флиорент. Мл. редактор Н. А. Львова. Обложка художника А. А. Смирнова. Худож. редактор Т. С. Егорова. Техн. редактор А. М. Крассавина. Корректор В. В. Каночкина.

ИБ № 2274

Г00983. Индекс заказа 94405. Сдано в набор 2.03.79 г. Подписано к печати 1.03.79 г. Формат бумаги 84×108¹/₃₂. Бумага типографская № 3. Бум. л. 1. Печ. л. 2. Усл. печ. л. 3,38. Уч.-изд. л. 3,64. Тираж 56 770 экз. Издательство «Знание». 101835, Москва, Центр, проезд Серова, д. 4. Заказ 430. ГСП. Типография Всесоюзного общества «Знание», Москва, Центр. Новая пл., д. 3/4. Цена 11 коп.

Дагаев Н. И. и Рудзицкий Б. М.
Д20 Банки данных. М., «Знание», 1979.
64 с. (Новое в жизни, науке, технике. Серия «Техника», 5. Издается ежемесячно с 1961 г.).

В брошюре, рассчитанной на инженеров, техников, студентов технических вузов, рассматриваются проблемы создания банков данных. Основное внимание уделяется реляционному подходу к сбору, обработке и хранению информации. Приводятся конкретные примеры.

30102

73

© Издательство «Знание», 1979 г.

Рост объемов перерабатываемой информации в сфере использования электронно-вычислительной техники и изменение характера требований пользователей заставили пересмотреть такую традиционную область обработки информации, как управление данными. Затронуты были фундаментальные пласты: организация данных (включая их структуризацию и представление), определение данных (стало рассматриваться как ряд самостоятельных направлений), языковые и программные средства управления информацией, технические средства хранения и перезаписи данных, организационно-правовые положения и т. д.

Новый подход нашел наиболее яркое выражение в концепциях банков данных. Развитие теории и практики синтеза банков позволило принципиально по-новому подойти к вопросам управления информацией в автоматизированных системах. Появилась возможность строить интегрированные системы обработки данных любой степени сложности. Резко сократилось время реализации запросов пользователей, уменьшились расходы на проектирование и эксплуатацию автоматизированных систем.

Банки данных в настоящее время создаются для многих сфер и отраслей народного хозяйства: планирования, управления предприятиями, статистики, здравоохранения и др. Специфика каждой из них делает обязательным дифференцированный подход к созданию банков.

В предлагаемой читателям брошюре изложены основные проблемы организации банков данных; основной упор сделан на информационно-технологические аспекты.

Требования к АБД. Прежде всего сформулируем понятие «автоматизированный банк данных».

Оно — дискуссионно. Можно, однако, выделить признаки, отличающие автоматизированный банк данных (АБД) от традиционных систем организации данных.

1. Процессы обработки данных отделены от процессов их накопления, хранения и поиска.

2. В памяти ЭВМ содержатся динамические информационные модели объектов управления.

3. В существующих системах два уровня описания данных — логический уровень пользователя и физический уровень системы; в АБД имеется еще и логический уровень системы и др.

Можно рассматривать АБД как систему, самостоятельно реализующую весь процесс обработки информации, или как подсистему, реализующую лишь процедуры (операции) накопления, хранения и поиска данных. В первом случае АБД отождествляется с информационно-вычислительным комплексом (ИВК), во втором он — лишь подсистема ИПС (что и является предметом рассмотрения настоящей брошюры).

Перейдем непосредственно к определению требований к АБД. Первое требование — **неизбыточность** хранимой в нем информации. Проблема избыточности особенно остра для ИВК, перерабатывающих огромные объемы информации. Исключение идентичных данных, во-первых, упростит внесение изменений в информацию, во-вторых, резко сократит время обработки и увеличит объем памяти для размещения вновь вводимых данных.

Информация должна быть актуальна и сформирована в соответствии с установленными правилами и ограничениями, что обеспечивает «совместимость» объекта и информации. Пользователю необходима уверенность, что предоставляемая ему информация максимально приближенно отражает истинное положение вещей. Поступившая информация заменяет хранящуюся в АБД, но не освобождает банк от необходимости содержать «старую».

Далее. Поскольку в АБД поступают запросы, различающиеся по времени, необходимому для формирования ответа, объему привлекаемых данных и т. д., невозможно использовать лишь одну единицу хранения информации. Стало быть, структуры данных должны быть многократного применения, что удовлетворяет неизвестные при создании АБД запросы его будущих пользователей; кроме того, средства описания структур

должны быть независимы от прикладных программ пользователей. Это означает, иными словами, что прикладные программы независимы от изменений в структурах данных: так называемая **независимость по данным**. При функционировании АБД периодически меняются физические (на машинных носителях) и логические структуры, но программные и языковые средства не должны это «чувствовать». Иначе, будет необходимо корректировать программы, настраиваясь на новые условия.

Говорить о полной независимости применительно к АБД можно лишь тогда, когда независимы логическая структура и носители, программы и носители, программы и логическая структура.

Следующее требование — **целостность**. АБД — система с мультидоступом. Может возникнуть, например, ситуация: две (или несколько) программы одновременно обратятся к одним и тем же данным. Сработает блокировка и, если не предусмотреть специальных средств, возникнет аварийная обстановка.

Такие ситуации невозможны при выполнении требования целостности. Его суть, во-первых, в том, что данные и связи между ними находятся в соответствующем состоянии, во-вторых, программам присваиваются приоритеты.

С понятием «целостность» соприкасается понятие «защищенность». В АБД хранятся сведения, разноплановые описывающие различные сферы деятельности. Следовательно, тех или иных потребителей ограничивают в доступе к тем или иным данным. Защищенность более важна для справочно-информационного обслуживания, поскольку запрос может поступить и от «случайного» пользователя. Она предполагает защиту информации от востребования пользователями на всем протяжении ее передачи; точек связи, т. е. терминалов; информации от преобразования.

Автоматизированный банк данных должен быть **многорежимным**, т. е. работать в режимах пакетной обработки, разделения времени, сообщений, реального масштаба времени и др.

Необходимо распространить рамки подхода не только на «ближних», но и на «дальних» пользователей. Иначе говоря, организовать связь с АБД как через уст-

ройства ввода и вывода ЭВМ, так и через удаленные терминалы (режим телеобработки).

И наконец, последнее требование к АБД: **адаптируемость** к изменению условий функционирования, появлению дополнительных связей с другими системами, созданию новых программных и технических средств и др. Здесь перед разработчиками возникает задача создать устройства, которые могут использоваться как на данном уровне развития АБД, так и в перспективе. Например, в накопителе статистических данных (банк данных, эксплуатируемый в ГДР) предусматривалось применение программных средств в существующей и проектируемой системах обработки, а также вне накопителя.

Другая проблема — **перенос АБД** в иную техническую и программную среду. Это может потребоваться, поскольку в процессе совершенствования ИВК возникает объективная необходимость в улучшении его подсистемы накопления и поиска. Разрабатываются новые (улучшенные) технические, программные и другие средства и методы.

Организация данных. Внешний уровень организации данных связан с их смысловым значением — семантикой. Основным признак, относящий структуру данных к семантическому уровню, — это смысл данных, обусловленный природой объектов, их свойствами, поведением. Будем использовать понятие «**единица хранения**» — организованная совокупность показателей. **Показатель** — это наименьшая совокупность данных с определенным, например, экономическим смыслом.

Исторически сложилось, что основная единица хранения информации — это **документ**. Он предусматривает буквальные и формально преобразованные копии. Букральные сохраняют внешний вид и все реквизиты документа, имеют юридическую силу в течение заданного периода времени. Формальное преобразование документов имеет целью организацию автоматического обращения к ним со стороны пользователей. В зависимости от способа хранения реквизиты формально преобразованных копий идентифицируются ассоциативно или по описанию формы документа, которое содержится в служебной части базы данных. **База данных** — это поименованная совокупность данных, в которую входят собственно данные и служебная метainформация.

Документ как единица хранения имеет следующие недостатки: трудно совместно обрабатывать несколько документов; данные за прошлые отчетные периоды дублируются; логический, арифметический и внутридокументальный контроль сложен. Однако у документа есть и достоинства. Он объединяет специально подобранные данные регулярно решаемых задач; содержит редко запрашиваемые сведения; представляет собой неразделенную информационную совокупность.

Еще одна единица хранения информации — **регистр, группа записей**, объединенных по принципу списка однородных объектов, например регистр населения, регистр земельных участков, регистр промышленных предприятий и т. п. Запись включает перечень последовательно расположенных показателей. Признаки упорядочения показателей — шифры. В начале записи находится идентификатор объекта. Запись специально преобразована для хранения. И в этом ее отличие от документа.

Узловой проблемой проектирования регистра является определение его структуры. При этом необходимо учитывать:

1. Для ввода в регистр новых данных используются идентификаторы объектов. Поэтому структура регистра должна способствовать быстрому поиску.

2. Выбор данных должен осуществляться по всем предусмотренным классификационным признакам (например, отрасль, территория).

Чтобы выполнить эти требования, регистр разбивается на относительно независимые части. Их называют **сегментами**, или **блоками**. Выделяют блоки, например, по экономической общности (локальности) показателей. Показатели, одновременно вовлекаемые в обработку, имеют общие свойства. Так, информацию о месте работы человека и занимаемой им должности целесообразно объединять в общий блок, поскольку совместная обработка данных позволяет определить социальное положение.

Кроме регистров состояния, т. е. хранящих новейшие данные, например, за год, есть регистры движения. В них накапливаются показатели за несколько лет. Регистр движения фиксирует значения показателей за каждый отчетный (или другой) период, поэтому он сравним с несколькими регистрами состояния.

Последний тип регистра — целевой. В его задачи входит расширять программу регистров состояния и движения (например, наряду с регистром населения можно вести регистр воинского учета и т. д.). Применение этой единицы хранения обеспечивает:

- позэтапность создания базы данных;
- эффективное распределение сил и средств на разработку систем ведения регистров;

- простоту и удобство внесения изменений при обновлении, актуализации и выдаче данных;

- сравнительно несложную организацию справочного аппарата, и как следствие относительно скромные требования к пользователям при составлении запросов.

Но наряду с достоинствами есть и недостатки. Главный из них — необоснованное и неоправданное дублирование данных в регистрах различных функциональных подсистем. Дублирование возникает из-за «пересечения» показателей разнородных объектов. Помимо этого, сложно одновременно вовлекать в обработку данные нескольких подсистем и отвечать на запросы по произвольному сочетанию признаков, что особенно существенно при справочно-информационном обслуживании.

Наиболее гибкой с точки зрения пользователей единицей информации является фонд ассоциативной памяти (ФАП). Он строится из показателей, связанных родовидовыми отношениями. Массивы показателей обычно описываются последовательностью общих (родовых) понятий, каждому из которых соответствует упорядоченный список подчиненных ему более конкретных (видовых) понятий, составляющих объемы этих родовых понятий. К примеру, общее понятие «продукт», а подчиненное, раскрывающее и дополняющее его, — «единица измерения».

Фонд ассоциативной памяти позволяет пользователю получать данные, задавая свои вопросы непосредственно по семантике. Иными словами, обращение к ФАП строится ассоциативно, без ссылки на идентификаторы (адреса) регистров, блоков (сегментов) записей. В результате образуются массивы, удовлетворяющие нужды пользователей при справочно-информационном обслуживании, но не содержащие одинаковых показателей.

Еще одна единица хранения — рабочие массивы. Их

собирают из данных других единиц хранения. Делают это исходя из следующих соображений. Во-первых, ограниченный объем оперативной памяти не позволяет разместить в ней все обрабатываемые данные. Во-вторых, у данных, выбираемых из нескольких функциональных подсистем, имеется общая часть. В-третьих, одновременно преобразуется сравнительно малая совокупность данных от всей единицы хранения. В-четвертых, в отдельных случаях, например при справочно-информационном обслуживании, целесообразно реализовать такую организацию данных, которая бы уменьшала время ожидания пользователей.

Время хранения рабочих массивов — величина переменная. Например, для справочно-информационного обслуживания рабочие массивы существуют, как правило, до окончания обслуживания, затем уничтожаются.

Перейдем к рассмотрению вопросов организации данных на следующих двух уровнях: логическом уровне пользователя и логическом уровне системы.

В АБД имеются различные подходы к организации данных на логическом уровне системы. Например, выделяются следующие звенья структуры: элемент, группа, групповое отношение, статья, файл, база данных. Связаны они могут быть по-разному: в виде списков, деревьев, сетей и т. д. Рассмотрим один из вариантов организации данных на логическом уровне системы.

Информация базы интерпретируется как множество массивов. У всех массивов есть идентификаторы. Роль идентификатора выполняет терм, минимальный элемент логической структуры. Массив состоит из записей, а они, в свою очередь, — из сегментов. Сегмент же составляют термы. Таким образом, элементами структуры в порядке старшинства являются: массив, запись, сегмент, терм.

Любой элемент структуры доступен последовательно, т. е. с перебором предшествующих элементов, или непосредственно по идентификатору (ключу). Многоуровневая (иерархическая) структура обуславливает наличие мультиключа. Он формируется из ключей нескольких уровней. Элемент, входящий в реализацию структуры, может иметь числовые, текстовые или специальные значения. Каждая однородная реализация, например массив записей, имеет общие свойства, но значения записей могут быть различны (записи с информа-

цией об автомобилях индивидуального пользования или о маломерных судах).

Одной из наиболее интересных проблем проектирования АБД является конструирование его базы данных. Вначале по семантическим признакам выделяются термины. Далее термины упорядочиваются или изменяется порядок их следования. В общем случае упорядочивание затрагивает «горизонталь» и «вертикаль» структуры, т. е. возможна одноуровневая и иерархическая организация элементов данных. При этом термины объединяются в сегменты.

Для обоих типов организации элементов данных задаются правила поиска. В большинстве АБД доступ определяется на уровне записи. Различают доступ последовательный, по списку, непосредственными указаниями идентификаторов. Каждый элемент списка имеет справочный аппарат, позволяющий организовать двухстороннюю связь между двумя соседними элементами. Однако такое распределение не позволяет прямо адресоваться к любому элементу без просмотра предшествующих (последующих).

Сегменты упорядочиваются в пределах записей. При этом из всех сегментов базы выделяются устойчивые совокупности, присущие тому или иному объекту. Записи упорядочиваются в пределах массивов. Изменяется порядок их следования. Возможна строгая зависимость по возрастанию (убыванию) шифров или произвольная.

И наконец, в пределах базы упорядочиваются массивы.

При наличии логической зависимости между терминами различных сегментов составляется цепь адресов, позволяющая эффективно обрабатывать базу. Кроме того, если имеется зависимость между записями одного массива, записи объединяются (например, при обработке массива с «постоянно выделенными» признаками).

Поскольку АБД является системой с мультидоступом, т. е. к ней подключено большое количество пользователей, возникает необходимость увязать их требования к организации данных. Иными словами, из базы должны выделяться совокупности данных, организованные в виде, отличном от того, в каком они представлены в базе. Такая организация называется организацией данных на логическом уровне пользователя.

Вполне понятно, что каждому из пользователей требуется лишь часть информации, хранимой в базе. Тем не менее эта часть является подмножеством данных базы. Одно из важнейших преимуществ введения логического уровня пользователя — повышенная сохранность неиспользуемой части базы, сокращение времени обработки запросов и т. д.

Рассмотрим вопросы организации данных на последнем — физическом — уровне системы. Физическая организация данных связана с их размещением на носителях хранения и прямо не зависит как от смысла данных, так и от их логической структуры, хотя, например, информацию, одновременно вовлекаемую в обработку, целесообразно размещать в одном месте физической памяти (на одном носителе).

Программное обеспечение ЕС ЭВМ реализует пять видов физической организации данных: последовательный, индексно-последовательный, разделенный, прямой, телекоммуникационный. Характеристики носителей хранения являются ограничениями при выборе физической структуры данных. Для реализации последовательной и разделенной организации не обязательно иметь память прямого доступа.

Для пользователей очень удобно то, что в АБД имеется возможность расширить стандартные виды организации данных с помощью древовидных, сетевых и списковых структур. Дерево и сеть позволяют выражать сложные многоуровневые отношения между элементами структур данных. Сложность здесь в создании программных средств, обрабатывающих многосвязные сети. Для выражения иерархических отношений существуют комбинированные организации данных, например, основанные на сочетании прямой и индексно-последовательной организаций.

Определение данных — априорное задание ряда описаний данных для их машинной обработки. Средства определения данных удобнее всего рассмотреть на различных уровнях организации: семантический уровень — логический уровень системы — логический уровень пользователя — физический уровень системы. В настоящее время для описания данных на семантическом уровне имеются мощные средства — информационные языки дескрипторного типа. Следует, однако, отметить, что не во всех случаях они практи-

чески необходимы. В зависимости от условий обработки можно применять более скромные средства описания, например, с помощью кодировки строк и граф документа.

Простота справочного аппарата, к сожалению, как это ни парадоксально, усложняет функционирование системы. Нет отличий у одинаковых показателей, содержащихся в разных документах. При изменении структуры документа приходится корректировать (или вновь создавать) его описание. Кроме того, пользователь обязан знать и указывать идентификаторы строк и граф, содержащих запрашиваемые данные.

В целом метод кодирования строк и граф достаточно надежен. Однако данные, описанные таким методом, «не стыкуются» с данными, определенными по содержательным признакам.

Определение по содержательным признакам производится с помощью иерархических классификационных систем и дескрипторных систем. Первые состоят из двух частей — регистрационной и классификационной, указывающих соответственно идентификатор элемента и отношения вхождения в класс, группу, подгруппу и т. д. Как правило, классификатор строится на основе иерархической классификации. Пример иерархической — универсальная десятичная классификация (УДК). При обработке данных со структурой типа регистр ведутся классификаторы из записей с шифрами и наименованиями объектов наблюдения, упорядоченными по возрастанию шифров.

Работая с классификаторами при справочно-информационном обслуживании, сложно формулировать запросы. Пользователь обязан знать все применяемые идентификаторы (шифры) и их отношения вхождения. В результате возрастает трудоемкость обращения. Отдельные свойства объектов не всегда попадают в выбранную схему классификации, что заставляет ввести дополнительный справочный аппарат. Кроме того, ряд терминов не унифицирован или унификация выполнена не для всех случаев.

Недостатки иерархических классификационных систем частично ликвидированы в системах, основанных на дескрипторном принципе. Основное достоинство дескрипторных систем — возможность игнорировать любую, заранее заданную схему классификации. Для это-

го в предметной области специалистами отбираются ключевые слова, наиболее подробно описывающие все содержательные отношения между понятиями в данной области и, кроме того, позволяющие пользователям сформулировать запрос на получение необходимых данных.

В простейшем случае, когда такие системы не имеют правил построения смысловых отношений между дескрипторами, т. е. системы не имеют грамматики, сложность обработки запросов пользователей резко возрастает. Например, при функционировании дескрипторных систем в области документалистики по запросу пользователя ему могут быть даны не только запрашиваемые, но и иные похожие документы. И наоборот, пользователю могут быть не выданы все документы, дескрипторы которых он указал в запросе.

Более развитыми считаются системы с грамматикой. В них дескрипторы, относящиеся к хранимой информации, и дескрипторы, указанные в запросе пользователя, дополняются специальными символами (указателями) роли этих дескрипторов. В ряде случаев значение дескрипторов определяют исходя из их положения в «грамматическом шаблоне». По такому принципу построена система информационного поиска в ИК АН УССР.

Системы с грамматикой, основанной на принципах смыслового кодирования, используют специальные формализованные информационные языки. С их помощью записываются сведения, относящиеся к определенной предметной области, классу решаемых задач.

Синтезируются подобные языки на базе естественных разговорных языков, на которые накладываются ограничения, обусловленные особенностью машинной реализации. Примеры таких языков — информационный язык статистических показателей (ИЯП), семейство информационных языков плановых показателей (СИЯПП), язык RX-кодов и многие другие.

Информационные языки состоят, как правило, из следующих частей:

1. Алфавитный словарь дескрипторов. В нем в алфавитном порядке записываются все дескрипторы, выбранные для данной предметной области (класса задач). Следует иметь в виду, что этот словарь хранится в памяти машины, что способствует при реализации запро-

са быстрому поиску информации. В реальных системах в качестве дескрипторов могут выступать сочетания букв, цифр.

2. Алфавитный словарь ключевых слов. Основное отличие ключевого слова (понятия) от дескриптора в том, что оно не обязательно унифицировано. Кроме того, выбор ключевых слов не всегда строго однозначен, поскольку в любой предметной области, как правило, нет единых критериев выбора ключевых слов. В области экономики, например, ключевыми словами могут являться «валовая продукция», «зарботная плата», «реализованная продукция», «среднесписочная численность служащих» и т. д.

Между алфавитным словарем дескрипторов и алфавитным словарем ключевых слов существует взаимно-однозначное соответствие (каждому дескриптору соответствует определенное ключевое слово).

3. Специальный словарь отношений между дескрипторами. Такой словарь называется тезаурусом. В нем все ключевые слова, являющиеся синонимами, объединяются в группы эквивалентности. В каждой группе выделен дескриптор, что позволяет при обращении к информации, хранимой в ЭВМ, использовать любые синонимы из предусмотренных в группе эквивалентности.

В тезаурусе также указываются родо-видовые отношения между дескрипторами, например часть — целое. Предприятие является родом по отношению к входящему в него цеху, а само, в свою очередь, — видом для отрасли.

Все дескрипторы тезауруса разбиваются на предметные рубрики (поля). Каждое поле затем делится на подполя, а в его пределах дескрипторы упорядочиваются в алфавитном порядке.

Наряду с этим тезаурус имеет и другие способы организации, расширяющие возможность поиска информации. Так, организуется единый алфавитный вариант построения тезауруса, в котором все дескрипторы ранжируются в алфавитном порядке. Кроме того, создается номерной вариант тезауруса, в нем дескрипторы размещаются в порядке возрастания их кодовых значений.

4. Грамматика языка, система правил построения его выражений. В зависимости от типа языка грамматика может иметь разный уровень развития, но в любом случае должна обеспечивать строгое, без каких-

либо исключений построение фраз на информационном языке. Для этих целей в грамматику включаются специальные указатели роли дескрипторов и грамматический шаблон. Грамматический шаблон представляет собой конструкцию (структуру), каждое из полей которой предназначено для фиксации определенного (одного) дескриптора (ключевого слова). Например, поле для фиксации единицы измерения (кг, м, с), поле для фиксации процесса (производство, распределение) и т. д.

5. Каталог множественных показателей. Его задача — зафиксировать отношения между дескрипторами, но те, которые постоянно (достаточно постоянно) существуют в определенной предметной области. Наличие каталога множественных показателей упрощает процесс составления запроса. Примерами постоянно существующих отношений между понятиями могут служить заработная плата — начисленная, валовая продукция — отгруженная и др.

Рассмотрим вопросы определения данных на логическом уровне системы и логическом уровне пользователя.

Подключение к АБД большого числа пользователей ставит задачу согласовать требования пользователей в части определения данных. Поскольку каждый пользователь может применять свои средства определения информации, а она хранится централизованно в базе, необходимо увязать «взгляды» пользователей на локальные данные, которые хранятся в одной базе.

Решение проблемы заключается в создании общего (единого) описания базы данных. Такое определение нередко называется «схема». Она выполняется на языке описания данных (ЯОД) для схемы. Языки ЯОД могут быть различных типов: повествовательные, ключевых слов, разделителей и фиксированных позиций.

Язык описания данных формирует различные логические отношения между элементами структуры. Кроме того, он реализует дополнительные функции: присвоение блокировок секретности, описание стратегий поиска; описание процедур контроля; спецификация отличий в представлении данных в базе и программе пользователя и т. д.

В начальный момент создания базы данных ее описание переводится в машинное представление и посто-

янно находится в памяти ЭВМ. Допустимо хранить схему в виде одного блока или блока, разделенного на части. Каждая строка схемы может иметь идентификатор, что позволяет вводить строки в ЭВМ в произвольной последовательности. При отсутствии идентификатора порядок строк строго определен.

Общий принцип организации схемы — ее независимость от прикладных программ, благодаря их отдельной трансляции.

Важным свойством АБД следует считать спецификацию отличий в представлении информации в программе пользователя (логический уровень пользователя) и базе данных (логический уровень системы). Определение данных, доступных программе (часто называется «подсхема»), создается до начала ее выполнения и является подмножеством схемы. Это обусловлено тем, что отдельным пользователям требуется лишь часть информации, содержащейся в базе, со структурой, не совпадающей со схемным представлением, т. е. программа манипулирует с частью описания — подсхемой.

Определение данных на логическом уровне пользователя реализует язык описания данных для подсхемы.

Таким образом, разделение определений информации на логическом уровне пользователя и системы согласовывает процессы хранения информации внутри и вне базы. Средством, определяющим данные на физическом уровне системы, является язык управления внешними устройствами (ЯУВ).

Логические структуры базы данных, определенные на ЯОД, должны размещаться в физической памяти ЭВМ. Для этого ЯУВ задает необходимые спецификации: размеры буферов ввода (вывода); организацию файлов; форматы записей, режимы обработки при буферизации и др. Важнейшая задача ЯУВ — обеспечить независимость прикладных программ как от изменений физической структуры данных, так и от перемещения данных по носителям хранения. С этой целью выполняется следующее:

1. В логической структуре выделяются блоки (области), и они отображаются на носители хранения независимо от прикладных программ.

2. Отдельным элементам структуры базы, например сегментам, присваиваются неизменяемые значения, называемые иногда ключами обращения. Ключи обра-

щения отображаются в физические адреса в соответствии с правилами, указываемыми разработчиками. Возможно лексикографическое расположение постоянных значений в памяти или произвольное. Формат постоянных значений определяется программистом.

Для конкретного носителя хранения формируется описание его физической структуры — модуль обмена. Модули обмена хранятся в библиотеке модулей и вызываются при запросе носителя определенного типа.

Функции управления данными в АБД выполняет под контролем администратора АБД система управления базами данных (СУБД) (администратор АБД — термин, оформившийся после появления банков данных. Под «администратором» понимают коллектив высококвалифицированных системных программистов, ответственных за создание и эксплуатацию АБД). СУБД — это пакет прикладных программ; он входит в состав специального математического обеспечения ЭВМ. Известны различные подходы к определению состава СУБД и реализуемых ею задач. Поэтому рассмотрим обобщенную схему управления данными.

Основной инструмент взаимодействия пользователя с базой данных — язык команд (ЯК). Он дополняет язык программирования функциями обращения к базе данных. С помощью его операторов запрашивается и передается информация в базу, а также обрабатываются системные индикаторы. В программе обработки имеются средства, обеспечивающие обращение к СУБД и передающие параметры, необходимые для выполнения оператора ЯК. Другой вариант — явное включение оператора ЯК в программу. По принципу использования ЯК различаются СУБД с базовым языком, с замкнутой организацией. В первом случае алгоритмический язык высокого уровня, например ПЛ/1, дополняется средствами обработки базы данных — операторами ЯК. Включение понимается в смысле добавления операторов ЯК в программу на алгоритмическом языке. Базовый язык определяет конструкцию и количество операторов ЯК. Например, в одном из вариантов КОБОЛ расширяется двенадцатью операторами ЯК; изменяется логический порядок записей, добавляются новые записи и др. Кроме того, имеются три оператора для обслуживания системных индикаторов.

Замкнутые системы ориентированы на создание

максимальных удобств в формулировании специализированных запросов. Необходимые операторы выбираются исходя из требований всех пользователей. Как правило, это многократно применяемые операторы. При обработке встроенный в систему алгоритм реализует каждый оператор.

В отличие от СУБД с базовым языком в системах с замкнутой организацией пользователю нет необходимости скрупулезно управлять последовательностью обработки. Поэтому замкнутые системы считаются непроцедурными. Кроме того, пользователь не заботится о перемещении данных по иерархии памяти, например, с магнитных носителей в оперативную память ЭВМ.

На организацию ЯК влияет ряд факторов. Построение массивов в базе, например, обуславливает тип используемых операторов. Требования пользователей и возможности языков определения данных накладывают ограничения на конструкции ЯК. И наконец, конструкции ЯК зависят также от способа связи СУБД с прикладной программой.

Наряду с рассмотренными ранее средствами СУБД: языком описания данных, языком управления внешними устройствами и языком команд для ее нормальной работы необходимо следующее программное обеспечение: управляющая система, программы сбора статистики, программы сервиса. Управляющая система координирует работу всех компонентов СУБД. Помимо этого, она организует взаимодействие СУБД с прикладной программой и операционной системой. Программы сбора статистики предназначены для сбора и накопления сведений об ошибках во время функционирования СУБД, о степени использования носителей информации, о количестве обращений к различным массивам и т. д. Помимо программ сбора статистики, в этом участвуют и стандартные средства операционной системы.

Исследование собранной статистики завершается принятием решения о необходимости улучшить организацию АБД. С этой целью сервисные программы под контролем администратора АБД выполняют действия, направленные на совершенствование качества управления данными. Они кодируют (декодируют) информацию, перемещают данные по носителям хранения, сжимают (расширяют) данные, собирают «мусор» и т. д. Суть последнего действия — удаление ненужных дан-

ных из базы. Оно может быть двух видов: логическое и физическое. Первое не приводит к исключению данных из базы, т. е. они продолжают оставаться в ней. Например, группа пользователей потребовала удалить определенного вида данные и ввести новые. После реализации требования для нее они перестают существовать, но физически находятся в базе. Через определенный интервал времени администратор АБД анализирует заявки всех групп пользователей на исключение и ввод информации и, если в них не встречаются противоречия, данные физически выводятся из базы. Для работы программы «сбор мусора» требуется специальная информация: адрес в памяти каждого исключаемого элемента, его связи с другими элементами, длина этих элементов.

В СУБД предусмотрены и другие сервисные программы: ведущие ежедневный протокол (журнал) работы СУБД; выдающие сведения об ошибках, обнаруженных во время выполнения операторов ЯК, и т. д.

Перейдем непосредственно к процессам управления данными в АБД. Управление данными — широкое понятие и применительно к СУБД предполагает следующее:

1. Управление созданием базы данных.
2. Управление вводом и размещением данных в базе.
3. Управление поиском и выводом данных из базы.
4. Управление реструктуризацией базы.

Реструктуризация базы способствует улучшению условий ее использования (сокращение времени ожидания, стоимостных затрат на обслуживание и т. д.). Реструктуризацию можно представить как функцию создания базы, поскольку она включает в основном те же действия, например преобразование имеющейся базы в новую, в соответствии с заданными условиями.

Управление созданием базы происходит под контролем администратора АБД. При этом логические структуры загружаются на носители хранения, определяют стратегии поиска, присваиваются ключи защиты, информация задается имена и т. д. При создании базы данных ведется отчет о производимых операциях, который затем анализируется пользователями.

Запрос на обслуживание, например, на ввод новых данных или выборку имеющихся в базе, оформляется

на одном из языков, доступных пользователю. В плане управления данными все действия рассматриваются на семантическом уровне.

Так, с АБД могут взаимодействовать следующие категории пользователей: внешние запросчики (например, дирекция предприятия), системные аналитики, прикладные программисты, программисты-системники, администратор АБД, администратор системы и др. Применение СУБД с замкнутой организацией сокращает количество пользователей, например, совмещаются категории программистов.

Итак, поступил запрос от внешнего запросчика. С повышением уровня формализации запроса, с одной стороны, облегчается труд программиста, с другой — предъявляются более жесткие требования к информированности пользователя о необходимых ему данных. Целесообразно поэтому разбить процесс формализации запроса на несколько этапов.

Сначала запрос переводится в вид, доступный ЭВМ, и управление данными переходит на логический уровень пользователя. Для выполнения программы должна быть составлена ее подсхема. С этой целью транслируется определение данных, оформленное на ЯОД для подсхемы, или в рабочую область программы вводится часть схемы. Рабочая область — это часть оперативной памяти ЭВМ, которая выделяется для каждой выполняемой программы. В рабочей области происходит, в частности, преобразование логической организации данных из схемного в подсхемное (и наоборот) представление. По завершении описанных действий программа начинает исполняться.

В момент появления оператора ЯК, который рассматривается как обращение к базе данных (переход к управлению данными на логическом уровне системы), параметры обращения передаются в специальные поля оперативной памяти, называемые системными индикаторами. В них может храниться информация, например, о характере ошибок. Обращение иногда дополняется сведениями из схемы или подсхемы.

Далее программа анализа опознает встретившийся оператор ЯК. Опознавание завершается его выполнением. До этого, однако, проверяется правомочность доступа и прочитываются системные индикаторы.

Управляющая система формирует запрос к опера-

ционной системе ЭВМ, одновременно передавая ей результаты работы программы анализа. Управление данными переходит на физический уровень системы. Модули операционной системы, обслуживающие конкретную физическую организацию, осуществляют обмен информацией с внешними устройствами и направляют ее в буферную область оперативной памяти ЭВМ. Обработка запроса в одном направлении завершилась.

Операционная система возвращает данные и передает функции управления СУБД. Если нет никаких нарушений, сбоев и т. д., СУБД формирует ответ для прикладной программы, передавая данные из буферной области оперативной памяти в рабочую область программы. При этом меняется организация данных на основе различий их описания в схеме и подсхеме.

В заключение процесса обработки запроса к базе управление возвращается следующей команде прикладной программы, т. е. информация поступает на логический уровень пользователя.

Реляционные банки данных. Реляционный подход к разработке общецелевых АБД предполагает использование принципов и методов управления, которые организованы в форме информационных таблиц (ИТ). Банки данных, в которых реализован этот подход, называют реляционными банками данных (РБД). Данные, хранимые в РБД, определяются в терминах математической теории отношений и исчисления предикатов первого порядка.

Определение структур данных, принятых в реляционных банках данных (РБД), основывается на изучении информации, описывающей конкретную предметную область (сферу приложения РБД). Предметной областью могут быть отрасль народного хозяйства, предприятие и другие объекты реальной действительности. В связи с автоматизацией процессов обработки данных необходимо решить ряд задач по анализу информации предметных областей. Анализ информации понимается, с одной стороны, как сам процесс исследования предметной области, а с другой — как результат процесса.

Результат анализа информации некоторой предметной области — это, по существу, ее описание. Чтобы достичь общности описания, будем использовать понятия (категорий) «объект», «характеристики объекта».

Информацию, описывающую различные предметные области в принятых категориях, будем называть объектно-характеристической информацией. Изучение свойств объектно-характеристической информации позволяет выбрать способы ее обработки и представления в базе данных, не зависящие от специфики приложений.

Объектно-характеристические таблицы. Описания предметных областей могут различаться формой выражения. Эта форма определяет способ записи на уровне предметной области. Отметим, что при выборе формы выражения информации следует исходить из категорий объектов и их характеристик. Способ записи информации в этих категориях происходит от одного из наиболее естественных и традиционных способов отображения, а именно табличного. Табличный способ записи — («представления») информации широко используется в практической деятельности, например при составлении счетов фактур, приходных ордеров, расчетных листов, учетных карточек, платежных требований и т. д.

Если информация представлена в виде строк, имеющих одинаковый формат <объект, характеристики объекта>, такую форму выражения называют объектно-характеристической таблицей (ОХТ). Представление результатов анализа информации предметной области в форме ОХТ осуществляет пользователь, имеющий дело с этой предметной областью. Это позволяет нам сформулировать основное назначение ОХТ — хранить информацию о предметной области в компактном, привычном, легко обозримом виде.

Произвольная ОХТ имеет двухмерную структуру (форму) и может быть представлена в виде матрицы с именованными столбцами. По строкам перечисляются объекты, представленные именами (кодowymi обозначениями), по столбцам — характеристики (точнее, классы характеристик). Совокупность объектов, определяемых одной ОХТ, составляет класс однородных объектов. Количество объектов, равное числу строк ОХТ, задает ее размерность по вертикали, количество характеристик соответствует числу столбцов и определяет размерность по горизонтали.

Для информационного отображения предметной области в целом необходимо сформировать несколько ОХТ.

Способ описания предметных областей поясним на примере фрагмента «Поставщики—получатели». Этот фрагмент составляют однородные объекты: предприятия, поставляющие некоторые детали (или просто поставщики); предприятия («назовем их получателями»), которым поставляются эти детали, и, наконец, сами детали. Объектам фрагмента предметной области «Поставщики—получатели» сопоставляются четыре именованных класса: 1) наименование поставщика; 2) наименование получателя; 3) номер комплекта; 4) номер детали.

Названия (имена) классов раскрывают содержательный смысл их элементов. Например, элементы класса 2 — суть наименования (условные) получателей. Вообще однородные объекты предметной области могут быть представлены в ее описании либо своими именами — отдельными словами, группами слов, отрезками связного текста, либо идентификаторами — буквенно-цифровыми кодами. Однако вопрос о форме записи не является принципиальным.

Для большей наглядности поставщики и получатели представлены своими наименованиями, а детали — номерами. Кроме того, предполагается полнота классов: НАИМЕНОВАНИЕ ПОСТАВЩИКА, НАИМЕНОВАНИЕ ПОЛУЧАТЕЛЯ, НОМЕР КОМПЛЕКТА, НОМЕР ДЕТАЛИ. Считается, что названные классы представляют все без исключения объекты фрагмента «Поставщики—получатели».

Совокупность имен (кодowych обозначений) объектов — одна из составных частей объектно-характеристической информации данного фрагмента. Как правило, имена объектов располагаются слева и составляют первый столбец ОХТ. В традиционной терминологии совокупность единиц информации (ЕИ) имен (кодowych обозначений объектов) называется «подлежащим». Столбцам, подлежащим ОХТ, присваиваются уникальные имена, раскрывающие содержательный смысл ЕИ. НАИМЕНОВАНИЕ ПОСТАВЩИКА (сокращенное название «НАИМ-ПОСТ») — пример полного наименования столбца. Однако удобнее оперировать сокращенными названиями: НАИМ-ПОЛУЧ (полное — НАИМЕНОВАНИЕ ПОЛУЧАТЕЛЯ), НОМ-ДЕТ (полное — НОМЕР ДЕТАЛИ) и т. п.

Если первый столбец ОХТ, где перечисляются име-

на объектов, считать подлежащим, то сказуемое — это все остальные столбцы. Столбцы-сказуемые представляют собой характеристики объектов. Пусть объекты-поставщики характеризуются с точки зрения того, в каком городе они расположены и каковы их коды. Указанным характеристикам поставим в соответствие классы: КОД ПОСТАВЩИКА, ГОРОД. Информационные единицы классов — соответственно кодовые обозначения поставщиков и названия городов, где они расположены. Оформим указанные ЕИ в виде двух столбцов и присвоим им уникальные имена: КОД-ПОСТ (сокращение полного наименования КОД ПОСТАВЩИКА), ГОРОД. Единицы информации, формирующие в своей совокупности эти столбцы, — еще одна (наряду с именами объектов-поставщиков) составляющая объектно-характеристической информации о фрагменте «Поставщики — получатели».

Итак, имеются две информационные совокупности: имена поставщиков, а также конкретные характеристики — кодовые обозначения и названия городов. Оформим имеющиеся ЕИ в виде ОХТ (назовем ее «Поставщики») из трех поименованных столбцов: НАИМ-ПОСТ, КОД-ПОСТ, ГОРОД.

Аналогичным образом оформляются ЕИ, описывающие два других класса однородных объектов — получателей деталей и т. д. Формируются три поименованных ОХТ (см. пример 1). Получатели (НАИМ-ПОЛУЧ, КОД-ПОЛУЧ); Поставки (НОМ-ДЕТ, НАИМ-ПОЛУЧ, ГОРОД-ПОЛУЧ, КОД-ПОСТ); Комплекты-детали (НОМ-КОМПЛ, НОМ-ДЕТ, КОД-ДЕТ). Здесь введена сокращенная запись объектно-характеристической информации. После имени самой таблицы в круглых скобках перечислены имена ее столбцов.

ОХТ «Комплекты-детали»

ОХТ «Получатели»

НОМ-КОМПЛ	НОМ-ДЕТ	КОД-ДЕТ	НАИМ-ПОЛУЧ	КОД-ПОЛУЧ
Комплект-1	Деталь-2	10	Искра	2
Комплект-1	Деталь-4	5	Костер	6
Комплект-1	Деталь-5	15	Молния	3
Комплект-2	Деталь-1	3	Сигнал	5
Комплект-2	Деталь-2	2	Буревестник	1
Комплект-3	Деталь-3	1	Рассвет	4
Комплект-3	Деталь-5			

ОХТ «Поставщики»

НОМ-ДЕТ	НАИМ-ПОЛУЧ	ГОРОД-ПОЛУЧ	КОД-ПОСТ
Деталь-1	Сигнал	Киев	32
Деталь-2	Искра	Москва	23
Деталь-2	Буревестник	Ленинград	92
Деталь-5	Молния	Москва	47
Деталь-5	Буревестник	Ленинград	47
Деталь-3	Рассвет	Киев	32
Деталь-4	Костер	Москва	32

ОХТ «Поставщики»

НАИМ-ПОСТ	КОД-ПОСТ	ГОРОД
Салют	92	Москва
Луч	23	Ленинград
Заря	32	Москва
Пламя	47	Киев

Пример 1: Объективно-характеристические таблицы фрагмента предметной области «Поставщики—получатели».

Подведем итог. Информационное отображение предметной области (в нашем случае фрагмента «Поставщики—получатели») имеет своим результатом:

- 1) выделение однородных объектов (поставщиков, получателей, деталей) и объединение их в классы;
- 2) выделение классов характеристик объектов каждого класса, например характеристика ГОРОД;
- 3) идентификацию объектов путем присвоения им уникальных имен;
- 4) конкретизацию характеристик применительно к каждому выделенному объекту, например выяснение названий городов.

Формализация сведений об отображаемых объектах приводит к понятию «данные».

Табличные представления данных. Представление данных в реляционной базе называют табличным. Тем самым подчеркивается связь представленных в реляционной базе данных с объектно-характеристическими таблицами. Представление данных в таб-

личной форме имеет специфику, обусловленную особенностями ее компонентов.

Каждая ЕИ, стоящая на пересечении строки и столбца, является единичной (не составной) для данной ОХТ. Правда, допускается возможность «нулевых» ЕИ, значения которых в данный момент времени не известны.

С точки зрения обработки данных работа с ОХТ сводится к выполнению некоторых действий над ними. Иногда возникает потребность в совмещении таблиц. Пользователю могут понадобиться части столбцов ОХТ или только отдельные строки. Манипулирование (оперирование) ОХТ как раз и заключается в выполнении такого рода действий со столбцами и строками ЕИ. Таким образом, таблицы, описывающие предметную область, оказываются операндами, над которыми выполняются соответствующие действия (процедуры).

Объектно-характеристическая информация, рассматриваемая с учетом требований машинной обработки, истолковывается при реляционном подходе в качестве данных. Соответственно табличные представления данных — это ОХТ, рассматриваемые с позиций машинной обработки.

Различают два уровня рассмотрения объектно-характеристических таблиц: логический (процедурный), семантический (непроцедурный). В зависимости от уровня рассмотрения различают логические и семантические представления таблиц. Учитывая это, будем говорить о логическом и семантическом аспектах понятия данные.

Логические табличные представления данных (ЛТП-данных). В силу структурных особенностей ОХТ реляционная база данных оказывается гораздо проще, чем в существующих АБД. С точки зрения логического уровня база данных — это набор двумерных (плоских) файлов, не связанных друг с другом. Совместная обработка (связывание) файлов использует содержательную общность их элементов данных. Смысловые отношения в реляционной базе реализуются программно. Так, возможна совместная обработка файлов, содержащих сведения о поставках и поставщиках. Это объясняется наличием в соответствующих ОХТ одного общего столбца — КОД-ПОСТ. Именно этот столбец связывает названные ОХТ и, следовательно, файлы, логически представляющие их.

Поясним термин «двухмерный (плоский) файл». Им обозначается информационная совокупность, в которой отсутствуют иерархические связи, повторяющиеся группы элементов данных, всякого рода ссылки на них.

В логическом плане реляционная база — частный случай тех структур данных, которые приняты в традиционных банках данных, иерархических и сетевых. Поэтому любая ОХТ представима в современных АБД и, следовательно, может быть описана в терминах соответствующего языка определения данных. В дальнейшем, когда идет речь о табличных представлениях данных логического уровня, предполагается возможность их определения на основе некоторого базового языка.

Каждая ОХТ представляет собой множество записей фиксированной длины. Запись состоит из элементов данных, представляющих объекты и характеристики ОХТ. Все записи включают одинаковое число элементов. Одинаковые записи отсутствуют.

Семантические представления данных. Сразу же уточним, что употребляемый нами термин «семантика» синонимичен словосочетаниям «понятийное содержание», «содержательный смысл», «смысловое значение».

Под семантическим представлением (СЕМПом) произвольной ОХТ понимается совокупность высказываний, раскрывающих содержательный смысл элементов данных. Эти высказывания утверждают, что элемент представляет собой определенный объект или конкретную характеристику (значение характеристики). Другими словами, формируются суждения о наличии у элемента данных (данного) свойств «быть именем объекта» и «быть значением характеристики». Возможность и целесообразность построения такого рода суждений обусловлены следующим.

Вне наименований столбцов ОХТ их ЕИ зачастую оказываются неопределенными. Неопределенной величиной, к примеру, является ЕИ «130 руб.». Смысл этой ЕИ раскрывают высказывания вида: «130 руб. есть средний заработок» и т. п. До тех пор пока ЕИ «130 руб.» не будет поставлено в соответствие одно из такого рода суждений, нельзя ничего определенного сказать о ее содержательном смысле.

Другой пример: ЕИ «Москва», принадлежащая и ОХТ «Поставщики» и ОХТ «Поставки». Поэтому не

совсем ясно, какой смысл имеет эта ЕИ. Но если мы располагаем высказываниями «Москва — город, в котором расположен поставщик»; «Москва — город, в котором расположен получатель», одно из них можно поставить в соответствие ЕИ «Москва». Тогда ее смысл выяснится окончательно.

В обоих случаях, таким образом, отдельная ЕИ сопоставлялась с высказыванием, которое выражало (представляло) ее смысл (семантику). Вообще выяснение содержательного смысла единиц объектно-характеристической информации предполагает построение высказываний двух типов:

- 1) «...есть имя объекта класса...»;
- 2) «...есть значение характеристики класса...».

Вместо первого многоточия помещаются: в суждении 1 — полные наименования классов однородных объектов, в суждении 2 — полные наименования классов характеристик.

В качестве примера приведем фрагмент семантического описания ОХТ «Поставки». Рассмотрим столбец НАИМ-ПОЛУЧ и сформулируем высказывания, выражающие смысл его элементов. «Искра есть имя объекта класса НАИМЕНОВАНИЕ ПОЛУЧАТЕЛЯ» — пример такого высказывания. По определению совокупность такого рода суждений (высказываний) есть семантическое представление (СЕМП) столбца НАИМ-ПОЛУЧ.

Понятие СЕМПа одного столбца естественно обобщается на случай группы столбцов, а затем — и на произвольную ОХТ в целом.

Формальный аппарат и модели данных. Стремление к строгому и ясному представлению отображающей информации привело к созданию формальной теории реляционных баз (банков) данных. Теоретическую базу реляционного подхода составляют современная алгебра и математическая логика. Поскольку данная брошюра выходит в серии «Техника», ограничимся содержательной стороной вопроса и не будем касаться математических аспектов.

Формализация ОХТ предполагает переход от оперирования ЕИ к манипуляциям с символами. Для этого нужно построить специальный символический (формальный) язык. Специфика формализации как особой тех-

ники (совокупности приемов) работы с табличными представлениями данных предполагает полное отвлечение от их смыслового содержания. На формальном уровне обработка данных сводится, по существу, к чисто механическим действиям с символами формального языка. Процесс же решения задач пользователя рассматривается как дедуктивный вывод одних знаков из последовательностей из других.

Эволюция взглядов на формальный аппарат реляционного подхода имеет два этапа. На первом этапе доминировал процедурный подход к описанию и реализации процессов обработки данных. Теоретической базой процедурного подхода была современная алгебра, в особенности один из ее разделов — теория отношений. Ориентация знакового (формального) аппарата (т. е. совокупности знаков и правил оперирования ими) теории отношений на табличные представления данных (ТПД) привела к созданию так называемой реляционной алгебры (R-алгебра).

Реляционная алгебра — это прикладная алгебраическая система, которая позволяет «выражать» ТПД и выполнять необходимые операции над ними. Особо подчеркнем прикладную направленность R-алгебры. На ее основе разработан и реализован язык манипулирования данными очень высокого уровня, называемый в литературе алгебраическим подязыком данных.

Второй этап в развитии формального аппарата реляционных банков данных отличается непроецедурный подход к решению задач пользователей РБД.

Встав на путь формализации процессов обработки данных на семантическом уровне, Е. Кодд — основоположник реляционного подхода — предложил формальный аппарат, близкий к исчислению предикатов первого порядка. Ориентированный на работу с СЕМПами данных, он получил название реляционного исчисления (R-исчисления); R-исчисление — это прикладная логическая система, предназначенная для записи запросов к реляционной базе данных. Вывод ответа рассматривается как процедура доказательства истинности запроса — теоремы, исходя из аксиом — СЕМПов данных, хранимых в реляционной базе.

Каждому уровню представления ОХТ отвечает свой формальный аппарат. Объекты R-алгебры — математические описания (аналоги) ОХТ. Эти описания являют-

ся результатом формализации ЛТП-данных; R-исчисление имеет дело с таблицами, определяемыми в терминах математической логики. Определения ОХТ в системе понятий R-исчисления строятся путем формализации СЕМНов данных.

Таким образом, формализация ОХТ преследует две цели: построение математической модели (аналога) произвольной ОХТ, причем предполагается ее использование в качестве операндов операций R-алгебры; создание средств определения (задания) произвольных ОХТ с учетом ограничений, накладываемых (в терминах R-исчисления) на их состав и структуру.

Математическое описание логических структур. Изучение логических структур обычно начинают со следующей абстракции: элементы ЛТП-данных рассматриваются как слова в некотором алфавите; алфавит объединяет все (нетождественные) символы (знаки), употребляемые для записи ЕИ, — русские и латинские буквы, десятичные цифры, разделители «точка», «дефис», «запятая» и т. д. Под элементом данных будем понимать некоторое слово, т. е. всякую линейную последовательность символов алфавита, имеющую содержательный смысл.

О таких словах (т. е. некоторых знаковых последовательностях) говорят, что они обладают семантикой. Запись ЛТП-данных — это линейная последовательность слов, называемая комплектом элементов данных. Соответственно файл, т. е. набор записей, трактуется как множество комплектов элементов данных. Исходные понятия: алфавита, слова (элемента данных), комплекты элементов, множества комплектов составляют основу математического описания ОХТ. Определение (описание) произвольной ОХТ выполняется в нескольких шагов. С каждым шагом связано некоторое базисное понятие R-алгебры.

Вскроем понятие домена — одной из основных конструкций R-алгебры. Доменом называется именованное множество всевозможных слов фиксированной длины, обладающих семантикой. Наличие семантики устанавливается относительно заданной предметной области. Так, множество названий городов (Москва, Киев, Ленинград и т. д.) — пример домена предметной области ОХТ «Поставщики — получатели». Этот домен (присвоим ему имя «ГОРОД») включает названия городов не-

зависимо от того, соответствуют ли они предприятиям; поставщикам или получателям.

Домены именуются так, что само наименование ограничивает класс возможных слов — элементов данных. Когда говорят, скажем, о домене ГОРОД, подразумевают именно множество названий городов, а не множество имен поставщиков или получателей. Домены определяют исходя из содержательных соображений. При формализации ЛТП-данных домены явным образом вводятся в математическое описание произвольной ОХТ. Выбор доменов осуществляется с учетом семантики ЕИ. Приведем пример.

Пусть требуется определить ОХТ «Комплекты — детали». Ее ЕИ принадлежат разным классам: номера комплектов представляют объекты, номера и количество деталей — характеристики. Семантика элементов ОХТ «Комплекты — детали» определяется с учетом их принадлежности к указанным классам. Например, слова «Комплект-1» и «Деталь-1» различаются по смыслу, поскольку первое представляет объект класса НОМЕР — КОМПЛЕКТА, второе — характеристику класса НОМЕР — ДЕТАЛИ. Используя различия в семантике слов — элементов ОХТ, зафиксируем три именованных домена.

Домены НОМЕР — КОМПЛЕКТА и НОМЕР — ДЕТАЛИ включают слова, обозначающие номера всех тех комплектов и деталей, которые рассматриваются в пределах фрагмента «Поставщики — получатели». Третий домен — это конечное множество положительных натуральных чисел, характеризующих комплектность.

Первый шаг в определении ОХТ, таким образом, связан с определением доменов — множеств слов фиксированной длины, представляющих объект и характеристики предметной области. Для ОХТ «Комплекты — детали» исходным пунктом ее определения в терминах R-алгебры служит выбор трех доменов: НОМЕР — КОМПЛЕКТА, НОМЕР — ДЕТАЛИ, КОЛИЧЕСТВО — ДЕТАЛЕЙ.

Следующий шаг — построение так называемого декартового произведения выбранных доменов. Декартовым произведением доменов ОХТ «Комплекты — детали» (обозначается через НОМЕР — КОМПЛЕКТА \times НОМЕР — ДЕТАЛИ \times КОЛИЧЕСТВО — ДЕТАЛЕЙ) называется множество всевозможных упорядо-

ненных троек, где первый элемент взят из домена НОМЕР—КОМПЛЕКТА, второй — из домена НОМЕР—ДЕТАЛИ, третий — из домена КОЛИЧЕСТВО — ДЕТАЛЕЙ. Декартово произведение доменов есть множество всевозможных упорядоченных комплектов (кортежей) и слов (элементов данных), в которых первый элемент взят из первого домена, второй — из второго домена и т. п.

Для иллюстрации зададим декартово произведение доменов ОХТ «Получатели»: НАИМ-ПОЛУЧ — Искра, Костер, Молния, Сигнал, Буревестник, Рассвет {КОД-ПОЛУЧ = {1, 2, 3, 4, 5, 6}}.

Декартово произведение НАИМ-ПОЛУЧ \times КОД-ПОЛУЧ определяет упорядоченные пары элементов:

<Искра, 1>, <Искра, 2>, <Искра, 3>

<Искра, 4>, <Искра, 5>, <Искра, 6>

<Костер, 1>, <Костер, 2>, <Костер, 3> и т. п.

Упорядоченность пар обусловлена порядком следования доменов:

КОД-ПОЛУЧ \times НАИМ-ПОЛУЧ = <1, Искра>,

<1, Костер>, <1, Молния>, <1, Сигнал>,

<1, Буревестник>, <1, Рассвет> и т. п.

Через декартово произведение перечисляются все логически мыслимые комбинации элементов доменов. Выбор комбинаций, отвечающих строкам заданной ОХТ, осуществляется на очередном — третьем — шаге процесса описания ОХТ. На этом шаге вводится в рассмотрение базисное понятие «отношение». Степень (арность) отношения R определяется числом доменов, участвующих в декартовых произведениях. Например, отношение имеет степень 2, если в его образовании принимают участие два домена.

Отношением R на множестве элементов ЛТП данных называется подмножество декартова произведения соответствующих доменов. При этом предполагается, что выбор подмножества определяет, какие комплекты (кортежи) элементов находятся в отношении R .

Содержательный смысл здесь состоит в выборе таких комплектов (кортежей), которые соответствуют строкам заданной ОХТ. Например, совокупность кортежей <Искра, 2>, <Костер, 6>, <Молния, 3>, <Сигнал, 5>, <Буревестник, 1>, <Рассвет, 4> задает ОХТ «Получатели».

В свою очередь, каждый из перечисленных кортежей

принадлежит декартовому произведению НАИМ-ПОЛУЧ \times КОД-ПОЛУЧ. Нетрудно убедиться, что совокупность строк ОХТ «Получатели» является подмножеством произведения доменов НАИМ-ПОЛУЧ и КОД-ПОЛУЧ. Следовательно, таблица «Получатели», а также ее логическое представление в виде двухмерного файла формально определяются как некоторое отношение R между элементами доменов НАИМ-ПОЛУЧ и КОД-ПОЛУЧ.

Подобным образом можно определить остальные ОХТ фрагмента «Поставщики—получатели». Так, отношение R , выполненное для отдельных кортежей декартового произведения НАИМ-ПОСТ \times КОД-ПОСТ \times ГОРОД, описывает (определяет) ОХТ «Поставщики».

И наконец, завершающий шаг процесса определения ОХТ, введение понятия «связь». Связь, как и отношение, — суть алгебраическая конструкция, являющаяся формальным аналогом ЛТП — данных. Раскроем сущность понятия связь.

Исходя из вышеизложенного ОХТ «Получатели» может быть определена отношением: ПОЛУЧАТЕЛИ (НАИМ-ПОЛУЧ, КОД-ПОЛУЧ). Теперь изменим упорядоченность списка атрибутов. Получим новое отношение ПОЛУЧАТЕЛИ (КОД-ПОЛУЧ, НАИМ-ПОЛУЧ). Указанные отношения определяют две ОХТ, отличающиеся друг от друга лишь порядком расположения столбцов. Однако упорядоченность столбцов в произвольной ОХТ не существенна, поскольку каждый ее столбец может быть идентифицирован по имени. Если пользователь знает имя требуемого столбца, доступ к нему организуется независимо от места, занимаемого им в ОХТ. Знание имен столбцов освобождает пользователя от необходимости запоминать порядок следования столбцов. Значит любые две ОХТ, отличающиеся друг от друга лишь упорядоченностью столбцов, не различаются с точки зрения пользователя. Это свойство ОХТ должно, разумеется, найти свое отражение в их математическом описании.

Эквивалентными будем считать такие отношения, которые различаются только порядком следования атрибутов. Так, отношения ПОЛУЧАТЕЛИ (НАИМ-ПОЛУЧ, КОД-ПОЛУЧ) и ПОЛУЧАТЕЛИ (КОД-ПОЛУЧ, НАИМ-ПОЛУЧ) должны быть признаны эквивалентными. Когда известно одно из отношений, эквивалент-

ные ему образуются перестановкой атрибутов этого отношения. В частности, на атрибутах отношения ПОСТАВЩИКИ можно получить $3! = 6$ перестановок. Другими словами: класс эквивалентности по объявленному эталону ПОСТАВЩИКИ (НАИМ-ПОСТ, КОД-ПОСТ, ГОРОД) образует вместе с ним еще 5 отношений:

1) ПОСТАВЩИКИ (КОД-ПОСТ, НАИМ-ПОСТ, ГОРОД);

2) ПОСТАВЩИКИ (КОД-ПОСТ, ГОРОД, НАИМ-ПОСТ);

3) ПОСТАВЩИКИ (ГОРОД, КОД-ПОСТ, НАИМ-ПОСТ) и т. п.

Класс эквивалентных (в указанном смысле) отношений принято называть «связью».

Работа с данными основывается на том, что связь — это отношение, рассматриваемое с точностью до порядка доменов. В этом случае допускается использовать одинаковые обозначения как для записи связи, так и для записи отношения. Например, запись ПОЛУЧАТЕЛИ (НАИМ-ПОЛУЧ, КОД-ПОЛУЧ) выражает отношение, но ее можно рассматривать как связь — класс, объединяющий два эквивалентных отношения: ПОЛУЧАТЕЛИ (КОД-ПОЛУЧ, НАИМ-ПОЛУЧ), ПОЛУЧАТЕЛИ (НАИМ-ПОЛУЧ, КОД-ПОЛУЧ).

Из сказанного вытекает, что определить (описать) в терминах R -алгебры произвольную ОХТ — это значит:

1) выбрать домены, на элементах которых задается отношение «образовывать строку ОХТ»;

2) образовать декартово произведение доменов;

3) выделить подмножество комплектов (кортежей) из элементов доменов, таких, что удовлетворяют отношению «образовывать строку ОХТ»;

4) ввести атрибуты, значениями которых являются элементы кортежей, удовлетворяющие заданному отношению;

5) сформировать класс эквивалентности, объединяющий отношения, которые отличаются друг от друга порядком доменов.

Манипулирование логическими структурами связано с некоторой моделью (назовем ее реляционной моделью представлений данных или короче — моделью данных),

специфическим кругом задач, которые можно решать, используя операции над связями. Задание на выполнение манипуляций в базе данных принимает форму алгебраического выражения.

В простейшем виде алгебраическое выражение эквивалентно процедуре, которая изменяет состав и структуру одной или двух ОХТ. Задание на обработку ОХТ может быть усложнено до набора подобных процедур. Сложное алгебраическое выражение предписывает преобразование совокупности ОХТ.

В основе формальных преобразований ОХТ лежит хорошо известная алгебра бинарных отношений. Некоторая специфика здесь в том, что можно выполнять операции над связями разных степеней.

Поскольку произвольную ОХТ можно представить только в виде связи (т. е. класса отношений, эквивалентных с точностью до порядка элементов кортежей), обработка ЛТД-данных сводится к манипуляциям над связями реляционной базы. Понятие связи формальным образом описывает логическую структуру реляционной базы. Формализованные логические представления данных, рассматриваемые вместе с процедурами их обработки, становятся алгебраическими объектами — составляющими реляционной модели (R -модели). Стало быть, понятие R -модели, естественно, возникает из самого алгебраического описания данных.

Действительно, математическое понятие «отношение» адекватно табличным представлениям данных любой степени сложности. С добавлением столбцов в какую-либо ОХТ увеличивается лишь степень соответствующей ее связи. Но независимо от этого данная связь считается множеством кортежей, каждый из которых представляет конкретные объекты и характеристики.

Итак, табличные представления данных логического уровня, естественно, обобщаются в единой системе понятий — домен, декартово произведение, отношение, связь, кортеж.

Математическое описание СЕМПов данных. Один из наиболее трудных моментов реляционного подхода — необходимость обобщить системы высказываний, раскрывающих смысл ЕИ. Имеют место два способа отображения ОХТ на СЕМПы данных. Первый рассматривался ранее, где разбирались высказывания, утверждающие наличие свойств «быть име-

нем объекта» и «быть значением характеристики». Сущность второго способа заключается в следующем.

Математическое описание ОХТ основывается на понятии отношения. Строятся высказывания, фиксирующие отношения между ЕИ. Совокупность таких высказываний, сформулированных к произвольной ОХТ, дает ее семантическое представление.

Пусть требуется получить СЕМПы данных ОХТ «Поставщики». Если бы мы построили систему высказываний, фиксирующих отношения между ЕИ, то одно из них выглядело бы так: «САЛЮТ есть наименование поставщика, имеющего кодовое обозначение 92 и расположенного в городе Москва». Высказывание раскрывает семантику ЕИ первой строки ОХТ «Поставщики». По этому образцу формулируются высказывания к остальным строкам ОХТ.

Эти высказывания утверждают, что ЕИ принадлежат какой-либо строке, т. е. фиксируют отношение «образовывать строку ОХТ». Его участники — имена объектов и конкретные характеристики. От числа ЕИ в строке ОХТ, или, что то же, количества столбцов ОХТ зависит структура подобных высказываний.

Аналогом отношения «образовывать строку» на множестве ЕИ служит хорошо известное из математики отношение степени 3 «образовывать сумму»: $X = Y + Z$. Сами по себе элементы X , Y , Z и даже пары элементов (XY , YZ , XZ) не образуют сумму $X = Y + Z$. В образовании суммы принимают участие только комплекты из трех ЕИ.

Сформулированное выше высказывание можно упростить, если записать его в виде «суммы»: «САЛЮТ» = «92» + «МОСКВА».

Знак «=» здесь символизирует фразу «есть имя поставщика», имеющего кодовое обозначение «92»; знак «+» — фразу «и расположенного в городе». Сделаем два замечания. Во-первых, представление ОХТ в виде высказываний, фиксирующих отношение между ЕИ, эквивалентно описанию в системе алгебраических понятий декартового произведения, отношения и связи. Во-вторых, система высказываний, раскрывающих семантику строк произвольной ОХТ, характеризует ее структуру.

Построение высказываний, фиксирующих отношения между ЕИ одной строки, — первый путь решения

проблемы формализации СЕМПов данных. Однако окончательного решения проблемы этот путь не дает.

Следующие примеры показывают, насколько разнообразны высказывания относительно ЕИ фрагмента «Поставщики—получатели»:

«Салют есть наименование поставщика, имеющего кодовое обозначение 92 и расположенного в городе Москва»;

«Искра есть наименование получателя, имеющего кодовое обозначение 2»;

«Комплект-1» есть номер комплекта, формируемого из деталей с номером «Деталь-2» в количестве 10 штук»;

«Деталь-1 есть номер детали, поставляемой поставщиком с кодовым обозначением 92 получателю, который имеет имя «Сигнал» и расположен в Киеве.

Итак, построение высказывательных форм (предикатов), утверждающих наличие отношения между ЕИ предметной области, не позволяет выработать общее определение ОХТ. Остается второй путь — попытаться по-иному сформулировать саму высказывательную форму, сделать ее инвариантной к составу и структуре ОХТ.

Самое естественное — пытаться игнорировать структурные особенности ОХТ. Поэтому определим произвольную таблицу как множество строк. Конечно, этого недостаточно, так как строки таблиц обычно различаются не только по составу, но и по содержанию. Однако различия проявляются лишь при описании ОХТ в терминах предикатов—отношений, участниками которых являются отдельные ЕИ. Укрупненное описание ОХТ на уровне строк, когда игнорируются их состав и содержание, позволяет абстрагироваться от структурных особенностей таблиц. При таком описании каждая строка рассматривается первоначально в единстве своих ЕИ, т. е. как «отдельное данное». В свою очередь, произвольная ОХТ определяется как множество строк. Подобное определение, безусловно, не зависит от структуры строк, а значит и таблицы в целом.

Попробуем выразить эту мысль несколько строже. Рассмотрим два комплекта ЕИ: <САЛЮТ, 92, МОСКВА>; <САЛЮТ, 92, ЛЕНИНГРАД>. До сих пор формулировались высказывания, которые раскрывали семантику каждой ЕИ комплекта: «Салют есть наиме-

нование поставщика, имеющее кодовое обозначение 92», и т. д. Возможно, однако, несколько иной подход, когда всякий комплект ЕИ рассматривается как нечто нерасчлененное, не состоящее из совокупности элементов данных. Единственный вопрос, который мог бы нас заинтересовать в этой связи, заключается в том, является ли тот или иной комплект строкой (отдельным данным) ОХТ «Поставщики». Ответом на вопрос служат значения высказываний типа «некоторый комплект ЕИ есть строка ОХТ «Поставщики». Например, высказывание «Комплект <САЛЮТ, 92, ЛЕНИНГРАД> есть строка ОХТ «Поставщики» оказывается ложным, в силу чего этот комплект не принадлежит множеству строк заданной ОХТ. Другими словами, <САЛЮТ, 92, ЛЕНИНГРАД> не обладает свойством принадлежности к заданной ОХТ.

Подобным же образом анализируется высказывание «Комплект <САЛЮТ, 92, МОСКВА> есть строка ОХТ «Поставщики». Оно, очевидно, истинно, поскольку комплект, определяемый этим высказыванием, входит в множество строк ОХТ «Поставщики». Можно также сказать, что комплекту <САЛЮТ, 92, МОСКВА> присуще свойство принадлежности к заданной ОХТ.

Таким образом, оба высказывания утверждают, что определяемые ими комплекты обладают вполне конкретным свойством. Содержательный смысл свойства, приписываемого этим комплектам, легко устанавливается из анализа самих высказываний. Это свойство формулируется по-разному и может быть выражено неполными предложениями типа:

«Комплект ЕИ ...есть строка ОХТ»,

«Комплект ЕИ ...принадлежит ОХТ»,

«Комплект ЕИ ...входит в множество строк ОХТ».

Учитывая это, примем следующее терминологическое соглашение: свойство комплектов ЕИ, фиксируемое (определяемое) посредством неполного предложения «Комплект ...входит в множество строк ОХТ», будем именовать свойством «быть строкой». Само неполное предложение назовем предикатом вхождения (P).

Предикат вхождения — это логическая функция с одной переменной, связывающая комплекты ЕИ реляционной базы с хранимыми в ней ОХТ. Свойство «быть строкой» определяется на множестве комплектов ЕИ

данных. Ему удовлетворяют комплекты, принадлежащие множеству строк заданной ОХТ. Фраза «Есть строка ОХТ «Поставщики» — типичный пример словесной формулировки предиката вхождения.

Символ S обозначает переменную величину. Она «пробегает» множество строк базы данных, каждая из которых служит ее значением. Переменную S принято называть строковой. Комплекты <ЛУЧ, 23, ЛЕНИНГРАД>, <ИСКРА, 2>, <КОМПЛЕКТ-1, ДЕТАЛЬ-2, 10> — возможные значения строковой переменной. Если строковую переменную заменить каким-нибудь ее значением, предикат вхождения преобразуется в высказывание.

Подстановка в предикат множества значений строковой переменной порождает целый ряд высказываний, которые всегда будут или истинными, или ложными. «Истинность» означает, что значением строковой переменной служит строка, принадлежащая заданной ОХТ. В противном случае, т. е. когда значение строковой переменной не входит в множество строк заданной ОХТ, предикат вхождения принимает значение «ложно». Так, например, для предиката вхождения $P(S)$, определяющего ОХТ «Поставщики», имеем: P (ЛУЧ, 23, ЛЕНИНГРАД) — истинно; P (ИСКРА, 2) — ложно; P (КОМПЛЕКТ-1, ДЕТАЛЬ-2, 10) — ложно.

Как видим, истинность и ложность предиката $P(S)$ зависит от того, является ли значение S строкой ОХТ «Поставщики». Тем самым предикат однозначно определяет множество строк реляционной базы, содержащих сведения о поставщиках.

Распределение значений предиката на истинные и ложные для всех комплектов ЕИ базы данных подразделяет ее на два класса строк: принадлежащих ОХТ «Поставщики» (1), не принадлежащих ей (2). Каждая строка базы данных относится к одному и притом только к одному из этих классов. Подставляя в предикат вхождения комплекты ЕИ, принадлежащие классу (1), получим истинные высказывания, а комплекты класса (2) — ложные. Совокупность истинных высказываний будем считать определением ОХТ «Поставщики».

В каждом конкретном случае имя предиката подбирается с таким расчетом, чтобы оно указывало, на то,

какая именно ОХТ определяется этим предикатом. Так, имя «ВХОЖДЕНИЕ В ОХТ «ПОСТАВЩИКИ» — одно из приемлемых обозначений предиката вхождения, определяющего ОХТ «Поставщики». Правда, оно громоздко. Избегать громоздких обозначений можно с помощью другого предоставления. Оно лишено строгости, но зато наглядно, поскольку предикатам вхождения присваиваются имена определяемых ими таблиц. Например, определением ОХТ «Поставщики» служит предикат вхождения ПОСТ (S): «Комплект входит в множество строк базы данных».

Изложенный подход к описанию объектно-характеристической информации в терминах R -исчисления — упрощенный и не отражает структуру ОХТ. С помощью предиката вхождения, по существу, описывается множество комплектов ЕИ, а не их совокупность, структуризованная в форме таблицы. По имени предиката организуется доступ лишь к ОХТ в целом. Но сам по себе предикат вхождения еще не обеспечивает обращения к отдельным столбцам и строкам (группам столбцов и строк).

Чтобы восполнить этот пробел, введем базисное понятие «срез». Под срезом понимается переменная величина, «пробегающая» множество компонентов строк базы данных. Область изменения среза в отличие от строковой переменной составляют не строки в целом, а их составные части. Компонент любой строки — это некоторый комплект (назовем его частичным) данных, состоящий из ЕИ этой строки.

Будучи значением среза, каждый частичный комплект рассматривается как отдельное данное, т. е. в нерасчлененном виде. Следовательно, понятие среза является более общим, чем понятие строковой переменной. Последняя есть частный случай среза, значения которого суть полные комплекты данных реляционной базы. Соответственно можно обобщить одноместный предикат вхождения «Комплект S есть строка базы данных». В общем случае, очевидно, следует говорить не о строке как таковой, а частичных комплектах ЕИ этой строки. Однако не все частичные комплекты данных произвольной ОХТ являются составными частями (компонентами) ее строк. Так, частичный комплект <ЛУЧ, МОСКВА> не является компонентом строк ОХТ «Поставщики», хотя и содержит их ЕИ. В свою

очередь, частичный комплект <ЛУЧ, ЛЕНИНГРАД> уже оказывается компонентом строки <ЛУЧ, 23, ЛЕНИНГРАД>.

Сказанное относится ко всем частичным комплектам из ЕИ таблицы «Поставщики». Это позволяет говорить о наличии свойства «быть компонентом строки базы данных». Этим свойством обладает комплект <ЛУЧ, ЛЕНИНГРАД>, но <ЛУЧ, МОСКВА> ему уже не соответствует (см. пример 1). Фразы «Частичный комплект q есть компонент строки базы данных», «Частичный комплект q входит в множество компонентов строк базы данных» фиксируют указанное свойство. Эти фразы — высказывательные формы, утверждающие наличие или отсутствие у комплектов данных свойства «быть компонентом строки реляционной базы».

В целом можно утверждать, что формализация СЕМПов данных, опирающаяся на категорию «свойства», обеспечивает адекватное описание ОХТ в терминах R -исчисления. В R -исчислении изучаются правила построения сложных высказывательных форм из элементарных, а также правила, согласно которым из истинности или ложности элементарных предикатов следует истинность или ложность сложных. Подстановка в предикат (высказывательную форму) конкретных значений порождает систему высказываний, принимающих значение «истинно» или «ложно». Система высказываний, порождаемых предикатом вхождения, представляет собой реляционную модель СЕМПов данных. Следовательно, моделирование данных и процессов их обработки на семантическом уровне осуществляется в системе понятий предиката вхождения, строковой переменной и среза.

Средства языкового общения. Языковые средства, применяемые в РБД, рассчитаны на различные категории пользователей — от наименее сведущих в программировании до наиболее искусственных. Различают два класса языковых средств: подязыки данных и языки запросов. Языковые средства первого класса имеют процедурную направленность и ориентированы на программиста-профессионала. Ко второму классу отнесены непроцедурные языки, рассчитанные на пользователя, не имеющего профессиональной подготовки в области программирования.

Учитывая уровень имеющихся процедурных подъязыков данных, подразделим их на два типа.

Подъязыки типа «элемент за элементом». Простейшие процедурные языки. Ими описывают (определяют) множества строк реляционной базы и таблиц элементов данных. Однако языки типа «элемент за элементом» оперируют лишь с отдельными строками таблиц.

Последовательная обработка данных, «элемент за элементом», характерна для ранних реляционных систем. По этой причине интерфейс, использующий подъязык этого типа, называют примитивным. В то же время отдельные операторы используются в манипуляциях над данными в развитых РБД и включаются в состав их языковых средств.

Работа с данными здесь сводится к примитивным манипуляциям со связями посредством команд создания и уничтожения, строк, их включения, модификации, передвижения и исключения. Это позволяет оптимизировать обработку данных, хранимых в реляционной базе.

Алгебраические подъязыки данных. У них разные операционные возможности. С учетом различий в способах обработки данных языки этого типа делятся на ограниченные и полные. Первые имеют дело лишь с совместимыми связями. Это значит, что, во-первых, операнды должны быть однородны в семантическом плане, во-вторых, иметь одинаковую так называемую *арность* (степень). Помимо этого, состав процедур обработки связей базы данных ограничен только классическими теоретико-множественными операциями — объединение, пересечение и вычитание множеств данных.

В отличие от ограниченных полные языки позволяют обрабатывать разнотипные данные. Кроме того, наряду с теоретико-множественными операциями полный алгебраический подъязык данных предоставляет возможность производить ряд специальных операций над связями — проекция, ограничение, произведение, соединение, деление связей и т. п.

Подъязыки алгебраического типа реализуют групповую (множественную) обработку данных. Операндами алгебраических процедур служат множества строк, не обязательно однородных по составу и содержанию. Выполнение процедур рассматривается как формирование

новой связи. Подъязыки алгебраического типа относятся к классу языков очень высокого уровня. Программы, написанные в терминах языков этого типа, компактны и наглядны, не содержат циклов и операторов условного перехода.

Алгебраические подъязыки данных можно рассматривать как развитую форму стандартных языков программирования, например ПЛ/1. Операторы алгебраических подъязыков могут быть оформлены как команды обращения к подпрограммам стандартного языка, интерпретируемые в реляционной системе.

Непроцедурные языки. К настоящему моменту уже созданы или находятся в стадии разработки языки запросов трех типов. Языки первого типа близки к естественному и ориентированы на неподготовленного (случайного) пользователя, например язык вопросно-ответной системы RENDEZVOUS. Пользователь формулирует запрос на естественном языке. Система задает пользователю уточняющие вопросы. В процессе уточнения запроса в систему поступает информация, необходимая для его однозначного понимания. Затем система переводит запрос пользователя на свой внутренний язык.

Языки второго типа называют графическими. Графические языки используются при работе с дисплеями. Представляет интерес разработка фирмы ИБМ — язык Query by Example. Он — составная часть развитой реляционной системы SBA, которая предназначена для автоматизации организационного управления.

И наконец, языки третьего типа — языки, не предназначенные для работы с данными в условиях применения дисплеев и не ориентированные на случайных пользователей. Языки эти рассчитаны, как принято говорить, на пользователей-полупрофессионалов. В зависимости от специфики применяемых средств формализации таблиц различают языки, опирающиеся на *R*-исчисление, и языки, основанные на отображении.

Языки, основанные на *R*-исчислении. Напомним, что *R*-исчисление — это формальный аппарат, по правилам которого порождаются сложные высказывательные формы, определяющие требуемые множества элементов данных. Языки, опирающиеся на *R*-исчисление, слишком формальны, что затрудняет их усваивание случайными пользователями,

Языки, основанные на отображении (SLICK, SQUARE, SEQUEL и т. д.). Их относят к разряду «нематематизированных» языков запросов. Под термином «отображение» здесь понимается многозначная функция. Отображение задается на множестве элементов связи базы данных. В отображении участвуют атрибуты одной или двух связей. В процессе отображения значения одного атрибута (группы) сопоставляются со значениями другого атрибута (группы). Сущность понятия отображения проанализируем на примере языка SEQUEL.

Единицей запроса в этом языке принята директива: ВЫБРАТЬ..., ИЗ..., ГДЕ.... Вместо многоточий проставляются: имена столбцов, откуда выбираются элементы данных; имя хранимой в реляционной базе таблицы, к которой формулируется запрос; условие на выборку элементов других столбцов. Директива эквивалентна поисковой процедуре. Поясним это на примере запроса к ОХТ «Поставщики» «Перечислить наименования поставщиков из Москвы»:

ВЫБРАТЬ — НАИМ-ПОСТ; ИЗ — ПОСТАВЩИКИ;

ГДЕ — ГОРОД = МОСКВА.

Директива реализует выборку наименований поставщиков из всех строк таблицы, где представлен элемент МОСКВА.

Встроенные функции. Класс задач, решаемых средствами R -алгебры и R -исчисления, характеризуется понятием «неарифметическая обработка данных». Характерной чертой этого класса задач является представление данных в виде множеств строк, а также в сложности и большом разнообразии способов их преобразования. Так, в R -алгебре есть операции, которые позволяют получать сложные структуры из более простых, и, наоборот, из более сложных — простые. Наряду с этим предусмотрены средства, позволяющие довольно точно определять поисковые предписания на выборку данных.

Специфика средств работы с данными в реляционных системах обусловлена тем, что в основу этих средств положены хорошо развитый язык и аппарат теории отношений и исчисления предикатов. Это способствует достаточной согласованности языковых

средств, посредством которых формулируются задания на выполнение преобразований в реляционной базе (запросы на выдачу ответа), с неарифметической природой процессов обработки данных в РБД.

Несколько иначе обстоит дело с вычислительными процессами, т. е. такими, реализация которых основывается на применении преимущественно арифметических операций над числами. Дело в том, что ни R -алгебра, ни R -исчисление не позволяют реализовать в наиболее простой и естественной форме вычислительные процедуры, типичные, например, для обработки экономической информации.

Подсчет количества объектов, представленных в ОХТ, суммирование числовых величин с получением итога по одному столбцу, то же по группе столбцов и т. д. — вот далеко не полный перечень элементарных вычислительных операций, реализация которых желательна в развитых РБД. Отсутствие средств реализации такого рода процедур заметно снижает эффективность реляционных систем, ориентированных на обработку экономической или иной числовой информации.

Отдельные, часто применяемые операторы, предписывающие выполнение вычислительных процедур, обычно встраиваются в реляционную систему. Стандартизация вычислительных процедур расширяет круг задач, решаемых средствами R -алгебры и R -исчисления. Встроенные функции организуются в виде библиотеки программ. К ним обращаются из программ пользователя, написанных либо на языке R -алгебры, либо в терминах R -исчисления. В обоих случаях указываются имена функций.

Назначение аппарата встроенных функций — расширение возможностей языковых средств РБД. В практике обработки данных чаще всего применяются три типа функций: арифметические, логические; функции, основанные на отображении.

Арифметические функции. Встроенные функции этого типа предназначены для реализации чисто учетных операций (процедур) над элементами данных. Простейшая из них — процедура, которая определяет количество строк таблицы, выводимой в ответ на запрос. Стандартизация этой процедуры приводит к встроенной функции ПОДСЧЕТ, т. е. к поименованному опера-

тору, предписывающему подсчет числа элементов искомого множества. Функция применяется к произвольным связям базы данных и позволяет получить новую информацию, которую нельзя получить, используя средства R -алгебры и R -исчисления. Покажем это на следующем условном примере.

Предположим, что пользователя интересуют сведения о том, сколько поставщиков расположено в Москве. Задача решается в два этапа. На первом формируется новая ОХТ, представленная классом эквивалентности строк из ОХТ «Поставщики». Этот этап как явствует из предыдущего изложения, реализуется средствами R -исчисления. Подсчет же строк с их помощью осуществить невозможно. Необходима функция ПОДСЧЕТ, применение которой к классу строк, эквивалентных константе «Москва», дает окончательное решение задачи.

Стандартным оператором ПОДСЧЕТ не исчерпывается перечень арифметических встроенных функций. К этому типу относится также функция СУММА, реализующая преимущественно сложение количественно-суммовых значений характеристик ОХТ. Иногда при формулировании запросов к базе данных требуется определить минимальные и максимальные значения элементов ОХТ; тогда применяют встроенные функции МИН и МАКС. Операндами функций служат произвольные связи реляционной базы.

При обращении к функциям СУММА, МИН, МАКС указываются: имя связи, которая представляет заданную ОХТ; срезы, идентифицирующие столбцы ОХТ, элементы которых подвергаются арифметической обработке.

Логические функции. Встроенные функции этого типа принимают только два значения: «истина» и «ложь» (соответственно «да» и «нет», 1 и 0 и т. п.). Основное назначение логических функций — обеспечить выборку информации из базы данных, исходя из заданных логических условий. Опишем механизм действий логических функций НАИБОЛЬШИЙ и НАИМЕНЬШИЙ. Операнды функций — произвольные ОХТ (точнее, связи), где зафиксирован хотя бы один столбец. Предполагается, что среди значений зафиксированного столбца имеется по крайней мере одно наибольшее (наименьшее). Функции играют роль логического условия,

причем действием, выполняемым при значении «истина» («да», «1» и т. п.), всегда является выборка строки из заданной ОХТ.

Рассмотрим особенности определения условий выборки. Прежде всего отметим, что функции не предполагают жесткого деления элементов зафиксированного столбца ОХТ данных на «наибольшие» («наименьшие») и не являющиеся «наибольшими» («наименьшими»). Все элементы упорядочиваются (ранжируются). Истина и ложность логических функций устанавливаются с учетом ранга. Поясним это на примере.

Пусть нас интересует информация о поставщиках. Зафиксируем столбец ГОРОД и введем лексикографический порядок на множестве МОСКВА, ЛЕНИНГРАД, КИЕВ его элементов. Тогда наибольшим окажется элемент КИЕВ, ближайшим к нему — значение ЛЕНИНГРАД, следующий элемент — МОСКВА. Каждому значению столбца ГОРОД сопоставим числовую величину — ранг. Учитывая порядок, значению КИЕВ присвоим ранг 1, ЛЕНИНГРАД — 2, МОСКВА — 3.

Условие выборки строк из ОХТ «Поставщики» определим с помощью логической функции НАИБОЛЬШИЙ. Ранг выступает в качестве проверяемого параметра, который задается при написании запроса (задания на обработку) к базе данных. От значения параметра (в нашем случае — ранга города, где расположен поставщик) зависит, будет или не будет выполнена выборка строк из ОХТ «Поставщики». Логическое условие считается выполненным для тех строк, где ранг значений характеристики ГОРОД равен величине, зафиксированной в запросе.

Понятие «ранг значений» помогает пользователю однозначно выражать свои информационные потребности. Иначе говоря, явное задание ранга при всех обращениях к реляционной базе способствует исключению нечетких формулировок. Например, отыскать наибольшее (наименьшее) значение характеристики ГОРОД. Среди всех элементов этого столбца отыскивается не просто наибольшее (наименьшее), а значение определенного ранга.

Специфика механизма действия логических функций раскрывает их назначение. Функции эти не предназначены для реализации операций, связанных с подсчетом строк ОХТ или суммированием значений характери-

стик этих таблиц. Функции **НАИБОЛЬШИЙ** и **НАИМЕНЬШИЙ** реализуют соответственно отношения **БОЛЬШЕ**, **МЕНЬШЕ**, определяемые на множестве элементов любого столбца произвольной ОХТ. Назначение логических функций — определение условий выборки строк по рангу, который специфицирует наибольшие и наименьшие элементы данных. Этим они отличаются от арифметических функций **МАКС** и **МИН**, которые предназначены для определения минимального и максимального значений.

Различие между указанными функциями продемонстрируем на примере ОХТ «Поставки». Пусть в ней представлены различные значения характеристики **КОЛ-ДЕТ**. Эти значения информируют о количестве деталей, поставляемых одними предприятиями, поставщиками, другим, получателям.

Теперь предположим, что пользователя интересуют сведения о минимальных (максимальных) поставках. Возможны два варианта. Если каждая поставка характеризуется в ОХТ показателем «количество поставленных деталей», естественно возникает вопрос о минимальном (максимальном) значении этого показателя. Ответом на вопрос служит числовая величина — основание показателя «количество поставленных деталей». Средством получения ответа здесь служит функция **МИН** (**МАКС**). Следовательно, применение функций **МИН** (**МАКС**) к ОХТ поставки дает ответ на запрос: каково минимальное (максимальное) число поставляемых деталей. Реализация этого запроса предполагает выборку из множества элементов столбца **КОЛ-ДЕТ** ОХТ «Поставки» одного, являющегося или наибольшим, или наименьшим.

Второй вариант имеет место в том случае, когда состав сведений о минимальных (максимальных) поставках не исчерпывается показателем количества поставляемых деталей. В частности, может возникнуть потребность в информации о поставщиках, поставляющих минимальное (максимальное) количество деталей. В этом случае нет необходимости в минимальных (максимальных) значениях характеристики **КОЛ-ДЕТ**, зато требуется выборка строк, содержащих эти значения. Иначе говоря, необходима выборка строк из ОХТ «Поставки» по наименьшему (наибольшему) значению характеристики **КОЛ-ДЕТ**. Именно для реализации та-

кой выборки и предназначена функция **НАИМЕНЬШИЙ** (**НАИБОЛЬШИЙ**). Поэтому логические функции отвечают на вопрос, в какой строке представлено наибольшее или наименьшее значение зафиксированной характеристики.

Логические функции **НАИБОЛЬШИЙ** и **НАИМЕНЬШИЙ** следует использовать при выборке строк, элементы которых обладают свойством **быть наибольшим** или **быть наименьшим**. Нетрудно убедиться, что эти же свойствами обладают строки ОХТ «Поставщики», «Получатели», «Поставки». Отсюда вытекает возможность применения функций **НАИБОЛЬШИЙ** и **НАИМЕНЬШИЙ** к указанным таблицам. Ниже в качестве примера приведем перечень запросов, при формировании которых целесообразно использовать логические функции.

1. В каком городе расположено больше всего поставщиков?
2. Какие поставщики расположены только в одном городе?
3. Какие поставщики поставляют наибольшее количество изделий?
4. Какие изделия поставляются наименьшим числом поставщиков? и т. д.

Функции, основанные на отображении. Хорошей иллюстрацией функций этого типа служат стандартные операторы **РАЗРЕЗ** и **ИТОГ**. Охарактеризуем процесс их исполнения.

Роль операнда встроенной функции **РАЗРЕЗ** выполняет произвольная ОХТ реляционной базы, состоящая, как минимум из двух столбцов. Пусть имеется ОХТ «Комплекты—детали», где в качестве объектов выступают номера комплектов, а значениями характеристик являются номера и количество деталей:

Номер комплекта (НОМ-КОМПЛ)	Номер детали (НОМ-ДЕТ)	Количество (КОЛ-ДЕТ)
Комплект-2	Деталь-1	3
Комплект-1	Деталь-4	5
Комплект-1	Деталь-5	15
Комплект-3	Деталь-3	1
Комплект-1	Деталь-2	10
Комплект-2	Деталь-2	2
Комплект-3	Деталь-5	4

Один из столбцов ОХТ-операнда фиксируется и играет роль параметра запроса (задания на обработку). Два первых столбца ОХТ «Комплект—деталь» равноправны в том смысле, что каждый из них можно принять в качестве параметра. Сначала зафиксируем столбец НОМ-ДЕТ и выполним по нему разрез ОХТ, а затем построим новую таблицу — разрез ОХТ «Комплект—детали» по столбцу НОМ-КОМПЛ.

Выполнение разреза ОХТ по столбцу НОМ-ДЕТ заключается в следующем. Задается отображение столбца НОМ-ДЕТ на столбец НОМ-КОМПЛ, т. е. с каждым номером детали сопоставляется номер изделия. Это отображение не является однозначным, так как значению «Деталь-2», к примеру, соответствует несколько значений: «Комплект-1», «Комплект-2». Нас интересует число элементов столбца НОМ-КОМПЛ, которые связаны (т. е. находятся в одной строке) с каждым значением характеристики НОМ-ДЕТ. Например, с элементом «Деталь-5» связаны два значения «Комплект-1» и «Комплект-3», а с элементом «Деталь-1» — единственное значение «Комплект-2». Рассмотрим запись <Деталь-5, 2>, информирующую о количестве комплектов, в которые входит деталь под номером 5. Назовем эту запись разрезом ОХТ «Комплекты—детали» по значению «Деталь-5». По аналогии определим разрез таблицы по элементу «Деталь-1» в виде пары <Деталь-1, 1>.

Совокупность такого рода записей, полученных в результате разреза по всем элементам столбца НОМ-ДЕТ, дает разрез ОХТ «Комплекты—детали» по этому столбцу. По определению этот разрез есть новая ОХТ из двух столбцов — НОМ-ДЕТ, КОЛИЧЕСТВО:

НОМ-ДЕТ	КОЛИЧЕСТВО
Деталь-1	1
Деталь-2	2
Деталь-3	1
Деталь-4	1
Деталь-5	2

Разрез по столбцу НОМ-КОМПЛ выполняется аналогично:

НОМ-КОМПЛ	КОЛИЧЕСТВО
Комплект-1	3
Комплект-2	2
Комплект-3	2

Как видим, применение функции РАЗРЕЗ к произвольной ОХТ изменяет структуру последней. Из таблицы-операнда удаляются все столбцы, за исключением одного, по которому выполняется разрез. К нему добавляется второй — КОЛИЧЕСТВО. Каждое значение характеристики КОЛИЧЕСТВО указывает на число строк ОХТ-операнда с одинаковыми элементами столбца таблицы, по которому выполняется ее разрез.

Коротко о ситуациях, требующих выполнения разреза таблиц базы данных. По существу встроенная функция РАЗРЕЗ оценивает повторяемость значений характеристик произвольной ОХТ. Следовательно, функция отвечает на вопрос, сколько раз встречается в таблице тот или иной элемент ее столбца. Для выяснения содержательного смысла функции РАЗРЕЗ рассмотрим ОХТ «Поставки» и «Поставщики». Зададимся вопросом: сколько поставщиков поставляют детали с одинаковым номером. Ответ дает разрез ОХТ «Поставки» по столбцу НОМ-ДЕТ. Далее разрез ОХТ «Поставщики» по значению «Москва» дает ответ на вопрос, сколько имеется поставщиков в Москве.

Поскольку функция РАЗРЕЗ основана на отображении элементов столбцов, применять ее целесообразно лишь при неоднозначном соответствии между ними. Если, скажем, ОХТ состоит только из двух столбцов, представляющих лишь наименования и взаимно однозначно соответствующие им коды объектов, применение функции РАЗРЕЗ не имеет смысла, так как все значения характеристики КОЛИЧЕСТВО новой таблицы-разреза будут равны 1.

Другой широко применяемой встроенной функцией является ИТОГ. Функция реализует сложение количественно-суммовых значений ОХТ-операнда и подсчет итогов по строкам таблицы, содержащим одинаковые элементы других столбцов. Возможность применения функции обусловлена наличием в данной ОХТ характеристики КОЛ-ДЕТ. Значения этой характеристики можно подсчитывать с учетом элементов любого из двух столбцов: НОМ-КОМПЛ и НОМ-ДЕТ. Предположим, что пользователя интересует вопрос, какое количество деталей, сгруппированных по номерам, представлено в ОХТ «Комплекты—детали».

Решение задачи сводится к суммированию количе-

ственных значений из строк ОХТ с одинаковыми элементами столбца НОМ-КОМПЛ. По этому столбцу, безусловно, и следует подсчитывать итог.

В процессе исполнения функции ИТОГ меняется структура таблицы-операнда. В частности, новая таблица — результат применения функции ИТОГ — состоит всего из двух столбцов: в первом перечислены номера всех комплектов, представленных в ОХТ «Комплекты—детали», во втором — количество деталей в комплекте:

НОМ-КОМПЛ	КОЛИЧЕСТВО
Комплект-1	30
Комплект-2	5
Комплект-3	5

Теперь обратимся к столбцу НОМ-ДЕТ. Применение функции ИТОГ в этом случае позволяет получить сведения о применяемости деталей, представленных в ОХТ «Комплекты—детали». Результатом здесь служит итоговая таблица, информирующая о том, из скольких деталей состоят комплекты.

НОМ-ДЕТ	КОЛИЧЕСТВО
Деталь-1	3
Деталь-2	12
Деталь-3	1
Деталь-4	5
Деталь-5	4

Таким образом, функция ИТОГ реализуется в два этапа. На первом группируются строки с одинаковыми элементами: либо столбцы НОМ-КОМПЛ, либо НОМ-ДЕТ. Второй этап — две процедуры: подсчет итоговых значений характеристики КОЛИЧЕСТВО и построение итоговой таблицы — результата использования функции ИТОГ.

Эволюция реляционных систем. Концепция реляционных баз данных в настоящее время находится на стадии становления. Дело в том, что круг реляционных систем, отвечающих современным представлениям о структуре и функциях АБД, крайне ограничен. До сих пор нет систем, на примере реализации которых можно было бы продемонстрировать наглядно основные преимущества реляционного подхода. Рано говорить о единой трактовке понятия «реляционный

банк данных». Имеющиеся в основном экспериментальные, реляционные системы существенно отличаются друг от друга. Более того, точки зрения немногочисленных разработчиков конкретных систем не согласованы. Поэтому представляется целесообразным в начале проследить эволюцию реляционных систем, а затем перейти к обсуждению вопросов организации реляционных банков данных.

Системы I поколения. Реляционные системы I поколения — LEAP и TRAMP — появились во второй половине 60-х годов. Они оперировали только с унарными и бинарными отношениями. Сфера применения систем накладывает ограничения на модели данных в LEAP и TRAMP. Указанные системы не были предназначены для управления большими базами данных, которые сейчас создаются, например, для нужд планирования развития народного хозяйства. Они ориентировались на информационное обслуживание пользователей, осуществляющих последовательный поиск небольших объемов информации.

Модель данных в системах I поколения отображает ассоциативные (семантические) связи между объектами конкретной сферы применения (обычно их два, и они одного типа). Работа с данными осуществлялась при помощи примитивного алгебраического языка. В каждый момент времени он оперирует только с одной строкой таблицы. Параллельно могут обрабатываться лишь унарные отношения, т. е. множества элементарных единиц информации.

Системы I поколения отличаются крайне низким уровнем языковых средств. Так, язык LEAP есть, по существу, усовершенствованный традиционными теоретико-множественными операциями АЛГОЛ-60. Язык TRAMP более выразителен и позволяет обращаться с разнотипными таблицами с помощью специальных операций, не имеющих аналогов в теории множеств. В результате упрощается взаимодействие с базой данных, поскольку запросы пользователя на выдачу новой, не содержащейся в базе информации интерпретируются в терминах реально хранимых таблиц.

Язык системы TRAMP, как и любой другой реляционной системы I поколения, не обеспечивает множественной обработки данных. Иначе говоря, система манипулирует либо с отдельными строками, либо с их ком-

понентами. Множество строк, в своей совокупности образующих таблицу, не является предметом операций языка системы TRAMP.

Системы II поколения. В реляционных системах II поколения уже используется алгебраический подязык данных. У первых моделей, таких, как RDF и STDS, алгебраический подязык данных ограничен, в нем отсутствуют такие операторы над таблицами, как «соединение», «деление», «произведение» и «ограничение». Выразительная сила языков RDF и STDS находится на уровне теоретико-множественных операций над таблицами. Поэтому операционные возможности систем II поколения не выходят за пределы обычной обработки множеств элементов данных.

Несмотря на процедурную ограниченность систем I поколения, они обеспечивают множественную обработку данных. Именно это и отличает их от систем I поколения. Так, в качестве информации, подлежащей обработке в языках RDF и STDS, выступают таблицы с одинаковым числом столбцов. Помимо этого, допускается использование логических операторов, заданных на множестве переменных (кванторов), что создает определенные удобства пользователям при работе с множествами взаимосвязанных таблиц.

На степень связей — математических представлений таблиц — в системах II поколения ограничения не накладываются. Исключением является система RDF. Ее реляционная модель бинарна; в то время как в других системах (STDS, MACAIMS, IS, RDMS) она n -арна.

Системы III поколения. Создание систем III поколения знаменует качественно новый этап в развитии концепции реляционных баз данных. Этот этап определяют два важнейших фактора. Во-первых, качественно изменились средства работы с данными. Язык запросов окончательно отделился от алгебраического подязыка данных и стал самостоятельной компонентой системы. Пользователь получил возможность общения с реляционной базой данных в терминах реляционного исчисления. Это обеспечило непроедурную запись запросов, интерпретация которых полностью возлагается на систему.

Во-вторых, намечился переход от системной модели данных к подмодели (части модели), выражающей точ-

ку зрения пользователя на реальные объекты, свойства, отношения.

Созданы предпосылки для разграничения двух уровней логической независимости данных. На первом уровне различные физические структуры используются для хранения одних и тех же таблиц. Это было реализовано уже в системах II поколения. В настоящее время достигнут второй уровень, который обеспечивает программную реализацию представлений пользователя о данных. Тем самым гарантируется высокая степень независимости данных, а пользователь отделяется (насколько это возможно) от структур хранения.

Из РБД III поколения интерес представляет система *R*, предназначенная для экспериментальных исследований в области архитектуры баз данных. Разработка системы *R* преследовала цель реализации полного круга задач, вытекающих из концепции реляционных баз данных. Заметим, что другие применения реляционного подхода были связаны с решением частных задач. В частности, известны реляционные системы, рассчитанные только на одного пользователя, что упрощает контроль и восстановление данных.

Система *R* состоит из двух частей:

подсистемы управления памятью, которая распоряжается внешними устройствами, распределением памяти, открывает доступ к отдельным строкам, выполняет операции обновления, удаления, обеспечивает восстановление;

подсистемы управления данными, которая обеспечивает целостность базы данных, строит каталог внешних имен (поскольку подсистема управления памятью оперирует только с внутрисистемными именами), оптимизирует исполнение запросов.

Значительное внимание уделено в системе *R* контролю данных. В этой системе, во-первых, имеется способ контроля за доступом, разрешающий или запрещающий манипуляции с таблицами. Каждый пользователь обладает определенными полномочиями на работу с данными, например, может быть уполномочен на создание новых таблиц и использование старых, хранящихся в реляционной базе.

Во-вторых, обеспечивается контроль целостности баз данных. Предполагается, что информация, хранящаяся в базе, удовлетворяет определенным правилам,

принятым в системе. В системе *R* утверждения относительно целостности реляционной базы выражаются предикатами о данных. В процессе функционирования системы гарантируется истинность этих предикатов.

Кроме того, в реляционную базу данных вносятся все необходимые изменения. Например, таблицу со сведениями о кадровом составе постоянно корректируют, учитывая прием, увольнение или перемещение сотрудников.

В системе организован непроцедурный интерфейс с пользователем. Эффективность использования АБД резко возрастает при применении непроцедурных языков запросов. Кроме того, эти языки — действенное средство оптимизации исполнения запросов.

Если база данных ориентирована на какое-либо одно приложение, ее структура легко оптимизируется с точки зрения именно этого приложения; при этом запросы (задания на обработку) исполняются исходя из принятой оптимальной структуры. Однако для многоцелевых баз данных такая локальная оптимизация неэффективна. Система должна сама оптимизировать процесс поиска данных в реляционной базе и их выборку. В системе *R* возможность такой оптимизации обеспечивается использованием непроцедурного языка запросов *SEQUEL*, в котором есть полный набор манипуляций над таблицами. Помимо *SEQUEL*, в состав языковых средств системы *R* входят операторы последовательной обработки («исключить», «вставить», «заменить» и др.).

Независимость данных определена в системе как независимость прикладных программ от изменений структуры хранения и стратегии доступа. Достигается она двумя способами. Первый основан на отделении логического представления данных от методов их организации в системе, т. е. можно использовать различные физические структуры при запоминании одной и той же таблицы.

Второй способ обеспечивает программную реализацию различных представлений пользователей данных.

Организация банков данных. Развитые реляционные АБД (II и III поколений) имеют многоуровневую организацию. В функциональном плане реляционные системы рассматриваются как надстройка над операционной системой (ОС) ЭВМ. Для большинства реляционных систем характерно наличие трех уровней организации.

Назначение первого уровня — обеспечение интерфейса с опе-

рационной системой ЭВМ. Поэтому его организация в значительной мере обусловлена особенностями конкретных ЭВМ и ОС. Категория пользователей реляционной базы и семантика хранимой информации здесь не существенны. Для иллюстрации отметим систему *RDMS*. К первому уровню ее организации относятся функции определения, обновления и каталогизации таблиц, открытия, закрытия файлов описаний.

Функции первого уровня реализуются специальными языковыми средствами. Так, в системе *IS* наряду с теоретико-множественными и, собственно, реляционными процедурами в самостоятельную группу выделены системные операции — запоминание, поиск, именованное соотношение.

С точки зрения первого уровня организации представляют интерес характеристики объемов информации, хранимой в реляционной базе. Так, в проектируемой системе *RAP* предполагается размещать (во внешней памяти объемом 10^8 бит) таблицы с числом столбцов, не превышающим 100. Другой пример: база данных *MACAIMS* отображается в двухмерный файл из 10^8 сегментов. При этом достигается прямая адресация 2^{18} 36-битовых слов. В системе *RDMS* обеспечивается доступ к базам данных объемом до миддона байт.

Второй уровень организации реализует операции над таблицами представлениями базы данных. Здесь организуется интерфейс с пользователем-программистом. Пользователь вводит в стандартный язык программирования набор операций над таблицами данных. Совокупность этих операций образует алгебраический подязык данных. Взаимодействие пользователей с реляционной базой — результат использования этого подязыка, в терминах которого составляются задания на обработку данных.

Ко второму уровню организации реляционных систем относятся также вопросы целостности базы данных и защиты ее от несанкционированного доступа. Наиболее полно они были решены в системе *R*.

Третий уровень — общение с пользователем, не имеющим профессиональной подготовки в области программирования. Средства общения служат непроцедурные языки запросов. Понятия «язык запросов» и «подязык данных» стали различать с момента появления *R*-исчисления.

Несмотря на большие операционные возможности языков запросов, основанных на *R*-исчислении, практика их применения показала, что они сложны и требуют от пользователей математической подготовки. Появление языков, использующих понятие «отображение», открыло новое направление в организации непроцедурного интерфейса с пользователями баз данных. Простота правил написания запросов, основанных на отображении, дала возможность в короткий срок разработать соответствующие непроцедурные языки, легко усваиваемые пользователями-непрограммистами.

Основная проблема, связанная с использованием языков запросов, — эффективная интерпретация запросов к реляционной базе и оптимизация вывода ответов на них. От решения этой проблемы во многом зависит производительность АБД. В системе *ZETA*, например, разработан оптимизатор-генератор, воспринимающий запросы, относящиеся одновременно к нескольким таблицам. Аналогичную функцию в системе *OMEGA* выполняет язык связей и селекции, осуществляющий отбор (селекцию) таблиц с учетом

их связей с другими данными, хранимыми в базе. И наконец, имеются системы с весьма развитым интерфейсом, к примеру, та же система ZETA. В нее входит достаточно простое устройство генерации, с помощью которого пользователь может создать свой собственный язык запросов, и устройство для распознавания фаз естественного языка.

Примеры реализаций. Рассмотрим наиболее интересные проектные решения, реализованные в современных РБД. Это поможет читателю более глубоко разобраться в сущности реляционного подхода и перспективах его развития.

Разграничение физических и логических структур. Имеются системы, где совпадают логические и физические формы представления данных. Так, в RDMS логической независимости данных нет. Каждая ОХТ в этой системе организуется в форме неупорядоченного файла, помещаемого в непрерывную зону памяти объемом до полумиллиарда байт. В системе MACAIMS впервые структуры хранения (физические структуры) были отделены от структур представления данных, относящихся к логическому уровню. Таблицы здесь рассматриваются как составные единицы информации и хранятся отдельно от элементов данных — целых строк, а также отдельных значений объектов и характеристик.

Система MACAIMS реализует различные способы отображения произвольных таблиц в памяти машины. Иначе говоря, предусматриваются средства представления объектно-характеристических таблиц в конкретные физические структуры данных. Это позволяет хранить каждую таблицу в наиболее подходящей для нее форме. При выборе необходимой физической структуры учитывается, во-первых, специфика решаемых задач, во-вторых, эффективность функционирования системы в целом. Проблема эффективного отображения табличных представлений данных решается с помощью программных средств, называемых модулями стратегии. Для каждой допустимой в системе структуры хранения создается определенный модуль. Он предназначен для манипуляции с таблицами, интерпретированными в терминах одной из допустимых физических структур. Например, в качестве основной структуры хранения в MACAIMS принят список.

Принятый в MACAIMS подход, использующий понятие «модули стратегии», был развит при проектировании системы MORIS. В ней также предусмотрены различные способы отображения таблиц и возможность оперирования с ними с учетом представления в допустимых структурах хранения. Помимо этого, в системе MORIS по-новому решен вопрос о представлении данных на логическом уровне, т. е. пользователь имеет возможность получить необходимые сведения (манипулировать данными в привычном для него виде). При этом в представлении пользователя о данных включены иерархические связи между объектами предметной области, их свойствами, отношениями. Эти иерархические связи формируются исходя из ОХТ, хранимых в реляционной базе. Тем самым в MORIS впервые были разграничены логические уровни пользователя и системы: один уровень отражает точку зрения пользователя на приложение (класс задач), с которым он работает; второй — взгляд на данные в целом с позиций реляционной модели и базы данных.

Аналогичным образом строится взаимодействие с данными в системе IS, где так же, как и в MORIS, разграничиваются модель

базы данных и представления пользователей. Все интересующие пользователя таблицы оформляются в терминах алгебраического подязыка данных. Эта процедура называется определением (описанием) пользовательских представлений данных. Определение данных в системе IS следует рассматривать как оформление всех необходимых логических отношений между элементами структур данных на уровне пользователя.

В качестве средства записи (или, как принято говорить, системы нотации) здесь выступает алгебраический подязык данных. Такие определения предполагают выполнение операций R-алгебры (объединение, пересечение, вычитание множеств, а также проекция, произведение связей и т. п.) на массиве таблиц базы данных; цель — формирование «пользовательского» представления о данных.

Поскольку определения записываются в языке R-алгебры, их надо транслировать. При первом обращении пользователя к таблице она формируется на основе хранимого в системе определения. Сформированная таблица может далее вновь быть размещена в базе. Следовательно, создаются предпосылки для постоянного увеличения объема обрабатываемых табличных представлений независимо от источника их формирования. Наличие в базе таблиц, сформированных при первом обращении, позволяет впоследствии обращаться уже непосредственно к ним.

При определении новых соотношений в качестве исходных выступают как основные (хранимые в базе данных) таблицы, так и табличные представления данных, ранее синтезированные системой. Это значительно расширяет возможности системы в части отображения реальных связей объектов, свойств и отношений сферы приложения. Тем самым снимается ограничение на источник формирования представления пользователей, что выгодно отличает IS от других реляционных систем, где таблицы определяются исходя из фиксированного (с момента загрузки до очередной реорганизации) состава базы данных.

Свобода выбора источника определения таблиц в IS позволяет упростить представление пользователя путем комбинирования основных (хранимых в реляционной базе) и дополнительно вводимых данных.

Организация информации. Интересным примером в этой области может служить система MACAIMS. Единичные информации рассматриваются в ней в двух аспектах: в качестве элементов данных, объединяемых в множества; как, собственно, таблицы, имеющие некоторое фиксированное число столбцов. При первом появлении в системе каждая ЕИ получает числовой идентификатор фиксированного формата — инвентарный (регистрационный) номер. Далее все обращения в ссылки на ЕИ осуществляются указанием присвоенного ей инвентарного номера. Инвентарные номера присваиваются так, чтобы их можно было использовать в качестве операндов. Например, инвентарный номер ЕИ А будет больше инвентарного номера ЕИ В, если А и В принадлежат одному множеству данных и А больше В.

Преимущество принятого в MACAIMS способа идентификации элементов данных в следующем. Во-первых, существенно экономится память системы — благодаря замене в базе данных имен отдельных ЕИ их числовыми идентификаторами. Во-вторых, представляется возможность сравнивать не сами ЕИ (из одного множества данных), а их инвентарные номера. В-третьих, ускоряется обработ-

ка множества данных, поскольку элементы данных различаются по своему формату, а у всех инвентарных номеров они единичны. Кроме того, время тратится на «упрощенном» выполнении часто используемой операции сравнения.

Безусловно показателна организация данных в RAPID — одной из новейших реляционных систем. Разработчики отошли от традиционных представлений данных в виде множества строк и набора проименованных доменов: в реляционной базе данных RAPID таблицы хранятся в виде столбцов. В целях экономии памяти столбцы закодированы. Такой способ реализации реляционной модели данных предпочтительнее традиционного: база данных становится более гибкой и приобретает такое важное качество современных систем, как способность адаптации к динамически меняющимся запросам пользователей.

Подъязыки данных и языки запросов. Основные направления здесь уже определены; выявлены основные типы языков и их назначение. Сведения о языковых средствах манипулирования с реляционной базой данных были систематизированы в предыдущих разделах брошюры. Поэтому кратко остановимся на интересном конкретном примере реализации алгебраического подъязыка данных и языка запросов.

В системе используется достаточно полный алгебраический язык данных, выполняющий не только теоретико-множественные, но и реляционные операции. Имеется возможность обращения к библиотеке встроенных функций типа СУММА, МАКС и т. д. В распоряжение пользователя система предоставляет около 100 процедур. Каждая из них реализована в виде самостоятельного модуля и поэтому доступна для любой программы, написанной на выполняющем языке PL/I. Модульный принцип программирования процедур представляет возможность их простого расширения. RDBMS относится к классу систем с ограниченным форматом языка запросов. Его конструкции подобны операторам КОБОЛА и имеют общий формат. Кроме того, есть специальный язык управления дисплеями, команды которого доступны пользователю-непрограммисту. В системе обеспечивается быстрый ответ на запросы с выводением больших объемов информации на несколько экранов одновременно.

Расширяемость. Наиболее полно и последовательно идея расширяемости воплощена в системе IS. Однако в первой версии системы полной расширяемости достичь не удалось (не был разработан механизм расширения языка). Применительно к ней речь идет лишь о функциональной расширяемости и расширяемости данных. Под функциональной расширяемостью понимают способность системы к наращиванию (увеличению) числа выполняемых функций. Это означает, что в любой момент пользователь может включить (встроить) новую функцию в системную библиотеку. В результате система приспосабливается к изменившимся потребностям пользователей.

Расширяемость данных определяется как возможность расширения (задания) в системе новых таблиц. Механизм расширения данных основан на использовании определений.

Управление памятью. Техника логического разбиения памяти, принятая в IS, иллюстрирует один из возможных подходов к решению проблемы эффективного управления памятью. Для хранения данных в IS выделяется специальная зона памяти. С течением

времени первоначальный объем данных можно уменьшить. В случае переполнения зоны хранения данных система автоматически выполняет защитные меры: под каждую таблицу, переполняющую объем памяти, система автоматически освобождает участок в ней. До переполнения в нем размещалась некоторая таблица, которая стирается (уничтожается). Таблицы, стираемые при переполнении, идентифицируются по определенным признакам (критериям), например по числу обращений к элементам данных базы. Все таблицы, удаляемые из базы данных при ее переполнении, сохраняются в системе. Несмотря на свое физическое уничтожение, таблицы существуют логически, т. е. доступны пользователю.

Помимо переполнения, таблицы автоматически уничтожаются еще в двух случаях. Во-первых, из зоны хранения удаляется любая ранее сформированная таблица, если изменено хотя бы одно множество элементов данных, участвовавших в ее формировании. Как и при переполнении, описание удаленной таблицы сохраняется, что обеспечивает возможность ее повторного формирования. Формирование осуществляется с учетом всех проведенных изменений. Во-вторых, система стирает сформированные таблицы, которые не оказывают заметного влияния (по крайней мере с некоторого момента времени) на ее функционирование.

Рассмотрим некоторые банки, созданные для различных отраслей народного хозяйства СССР, например банк данных универсальной структуры «Банк». Его проектирует Научно-исследовательский институт управляющих машин и систем (НИИУМС). Он предназначен для генерации и эксплуатации базы данных, размещенной на магнитных дисках. В базе данных каждая единица информации запоминается однократно, время поиска единиц данных незначительно, имеется возможность произвольного группирования данных по принципу их логической взаимосвязи.

Применение системы «Банк» позволяет сократить сроки программирования задач АСУ, улучшить использование запоминающих устройств, облегчить создание и корректировку базы данных. Он будет применяться в двух основных областях: АСУ и информационно-справочные системы. В обоих случаях он рассчитан на стандартную конфигурацию технических средств ЭВМ ЕС 1020 и операционную систему ДОС/ЕС. Исходя из размеров базы данных, можно подключать дополнительные дисковые устройства. Для размещения программного хозяйства требуется минимально 14 К оперативной памяти.

База данных «Банка» имеет списковую организацию. Элементы списков — записи. Записи объединяются в массивы. В пределах массива у них одинаковая структура, но различные значения. В списках каждый элемент хранит адрес последующего элемента. Состав записей и адреса связей определяются в соответствии с требованиями пользователей и спецификой решаемых задач. Иными словами, организация базы данных зависит от конкретной предметной области.

Программное хозяйство системы «Банк» состоит из наборов универсальных процедурных модулей, обслуживающих программы и т. д. Универсальные программные модули предназначены для операций с логическими структурами данных. В их функции входят включение, поиск, замена, удаление записей из базы. Модули универсальны, т. е. их можно настраивать на структуру базы данных, указанную пользователем. Общение пользователя с базой осу-

существляется на логическом уровне (пользователям нет необходимости задавать определения физических структур данных).

Набор макроопределений используется, с одной стороны, для обращения к процедурам доступа, с другой — для описания структуры базы данных. Обращение выполняется с помощью программы пользователя, составленной на включающем языке (ассемблер).

Основное назначение обслуживающих программ — обеспечение надежности хранения информации и выполнение отдельных вспомогательных действий. Например, обслуживающие программы занимаются восстановлением базы данных после сбоев оборудования, а также реорганизацией ее структуры — по истечении определенного периода времени. Для этого имеется специальная обслуживающая программа, в задачи которой входит сбор и анализ статистических данных по эксплуатации банка.

Другой отечественный банк данных — система ОКА, совокупность языковых и программных средств для использования как средних, так и больших по объему баз данных при решении самых различных задач. Основное назначение системы: организация данных для создания, взаимодействия, введения, расширения больших баз данных и их использования, обеспечение этих баз средствами обработки и обслуживания и организации доступа к ним на значительном расстоянии, организация эффективной подсистемы теле связи с большим объемом обрабатываемой информации и малым временем ожидания.

Рассматриваемый банк данных существует в двух модификациях: система баз данных ОКА(1) и система баз данных со средствами передачи данных ОКА(2). Первая вырабатывает входные рабочие потоки в пакетном режиме, вторая является проблемно-ориентированной и планирует работу на основе входных сообщений. Обе модели работают под управлением ОС ЕС. Минимальный набор средств для работы: устройства ввода-вывода, которые требуются для ОС ЕС; прикладные программы, не входящие в СУБД ОКА; база данных; накопитель на магнитной ленте, необходимый для обслуживания и распределения ресурсов системы; 500 цилиндров на дисках ЕС-5050 для хранения программ или же эквивалентная память на дисках ЕС-5061. Дополнительные устройства ввода-вывода для работы в режиме телеобработки — по крайней мере один абонентский пункт АП-2, один АП-1 и два накопителя на магнитной ленте ЕС-5010. Дополнительно могут быть использованы АП-63, АП-70, АП-11. Система ОКА в минимальной конфигурации требует оперативной памяти емкостью 128 К для пакетного режима и 256 К для режима телеобработки.

Представляет интерес система КАМА, которую нельзя считать СУБД в «чистом» виде, но она предназначена для обслуживания баз данных, работающих в режиме телеобработки. Операционные возможности КАМА характерны для систем, использующих терминалы: переключение сообщений, их обработка, сбор информации, ввод директив, диалоговый ввод данных. Система обеспечивает: управление смешанным телекоммуникационным оборудованием; единую базу данных для всех прикладных программ; одновременное выполнение различных программ; управляемый доступ к базам данных; управление ресурсами для обеспечения непрерывной работы системы; управление приоритетами обработки.

Работает КАМА под управлением ОС ЕС и может быть использована в моделях ЕС ЭВМ в АСУ различного уровня. Мини-

мальный набор средств для ее функционирования — процессор с емкостью оперативной памяти 128 К, четыре накопителя на магнитных дисках ЕС-5052, один накопитель на магнитной ленте ЕС-5017, устройство ввода с перфокарт ЕС-6012, комплекс ЕС-7066 (4 алфавитно-цифровых дисплея и пишущая машинка с общим устройством управления) и ОС ЕС-4.0 с графическим доступом.

Предложение построить АБД со списковой организацией нацело воплощение в проектировании СУБД НАБОБ. Она рассчитана на функционирование в среде ДОС ЕС. Система гарантирует независимость данных от использующих их прикладных программ; защиту данных от сбоев оборудования и от искажения программой; разграничение доступа к данным; избыточное их представление; непротиворечивость; интегрированное хранение и дифференцированное использование; возможность работы с разнообразными структурами данных.

Основные языковые средства НАБОБ — язык определения данных (ЯОД), язык манипулирования данными (ЯМД). Первый предназначен для описания структуры данных на логическом уровне системы. Операторы, ЯМД осуществляют взаимодействие прикладной программы с базой данных, предоставляя средства для их выборки, добавления новой информации, изменения имеющихся отношений между данными.

Интересны реляционные системы, которые проектируются в СССР. Для построения региональных АСУ необходимо иметь возможность объединять различные локальные базы. Это вызвано тем, что базы в различных АСУ могут иметь несовместимые структуры данных (различные модели данных). Решить проблему может создание единого описания всех баз, данные которых вовлекаются в обработку. Причем, описание не должно зависеть ни от типа ЭВМ, ни от применяемых СУБД.

Созданное описание, удовлетворяющее названным требованиям, называется виртуальным, а банки — виртуальными. По такому принципу строится СИЗИФ — система интегрированного запоминания информации. В ней каждой локальной базе специальными языковыми средствами (виртуальными языками) ставится в соответствие совокупность информационных таблиц (ИТ). Полный набор ИТ есть виртуальная реляционная база. Для работы с ней используется язык манипулирования виртуальными данными.

Другой пример. Реляционная система РЕГЕНД, которая наиболее наглядно демонстрирует преимущества реляционного подхода. Это совокупность языковых и программных средств, выполняющих обработку запросов пользователей к базе. Наряду с этим в нее встроены средства администратора АБД. РЕГЕНД генерирует программы обработки данных в АСУ в тех случаях, когда имеется относительно небольшой объем вычислений. Допускается использование языка Кобол, как включающего, т. е. дополняемого операторами языка манипулирования данными (ЯМД).

Отличительная черта РЕГЕНД — применение системы тождеств реляционной алгебры, что обеспечивает эффективное исполнение запросов — приведение их к виду, наиболее приспособленному для реализации. Отметим, что ЯМД РЕГЕНД основан на R-алгебре, поскольку именно это гарантирует эффективное построение процедур оптимизации запросов; объем памяти 86 К. Наборы данных системы размещаются на магнитных дисках типа ЕС-5050, ЕС-5061 и магнитных лентах.

Рассмотрение всех аспектов организации и функционирования банков данных, безусловно, выходит за пределы брошюры. Авторы старались сформулировать лишь основное представление о построении и эксплуатации банков данных в автоматизированных системах. Из поля зрения авторов «выпали» вопросы определения функций администратора банка данных, применения средств операционной обстановки, исполнения функций СУБД в пакетном и диалоговом режимах и некоторые другие. Часть из этих вопросов все же была затронута в брошюре, но не «самостоятельно», а в качестве пояснения к излагаемому материалу.

Следует еще раз подчеркнуть, что понятие «банк данных» многоаспектно. И разработка банков не завершена. Потребуются усилия различных специалистов, чтобы банк данных стал отвечать тем требованиям, которые предъявляются к нему на современном уровне развития средств обработки массовой информации.

Не случайно многие министерства и ведомства в СССР ведут проектирование банков данных для самых разнообразных сфер приложения. Можно отметить работы, проводимые в ИК АН УССР, НИЦЭВТ ИНЭУМ, ЦНИИТУ, ЦЭМИ и ИПМ АН СССР, НИИ ЦСУ СССР и т. д. Результатами этих работ явились системы СИНБАД, ИНФОР, БАЗИС, УПД-6, ВЕГА, СИОД 1, СИОД 2, СИОД-АСБТ, СЕКУНДА, АДОНИС, ИСХОД, АСПИД и др.

Необходимо еще раз упомянуть, что уже есть реляционные банки данных, как, очевидно, ясно читателю. Они значительно упрощают работу пользователей, не имеющих профессиональной подготовки в области программирования. Почти все разработанные в нашей стране системы управления данными ориентированы на ЕС ЭВМ. Это позволяет надеяться, что они найдут широкое применение при построении АСУ.