Математика
и механика

УДК 65.011.56 : 681.3.06

В. И. ВОЮШ, В. И. ДРАВИЦА, А. И. ЗМИТРОВИЧ,
Ю. П. МЕЛЕШКО, Н. В. САБАНАЕВА

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ПРОЦЕДУР РАБОТЫ С БАЗАМИ ДАННЫХ

В связи с перспективным использованием систем управления базами данных (СУБД) в задачах обработки данных (АСУ, САПР и др.) одним из важных направлений в теории и практике баз данных (БД) является создание инструментальных средств для разработчиков прикладных программ и администратора БД. Появление новых СУБД и расширение функциональных возможностей существующих систем, существенное расширение круга решаемых задач на основе БД, необходимость совершенствования технологии работы с БД требуют создания теории проектирования БД, которая может определить набор инструментальных средств.

Процесс проектирования можно разделить на следующие основные этапы: инфологическое моделирование, логическое проектирование, физическое проектирование и оценка результатов в процессе функционирования БД или путем моделирования работы БД на контрольных задачах. Процесс проектирования БД в целом, как и его отдельные этапы, можно рассматривать с точки зрения степени их формализации и в связи с этим возможны различные направления практической реализации процесса проектирования, а именно:

а) ручное проектирование (низкая степень формализации), при котором практически все решения принимает разработчик БД;

б) автоматизированное проектирование (средняя степень формализации), предполагающее, помимо участия специалистов, наличие формальных алгоритмов процесса проектирования БД, на основе которых разработана и функционирует система автоматизации проектирования (САП) БД;

в) автоматическое проектирование (высокая степень формализации), когда САП БД автоматически проектирует БД на основе модели представления знаний о предметной области.

Важными процедурами проектирования БД являются разработки описаний для ее схемы, подсхем и написание операторов обращения к СУБД. Так, анализ структуры и состава программ АСУ одним приборостроительным заводом, построенных с использованием СУБД ОКА и предназначенных для решения 42 задач подсистем УТПП, ТЭП, ОУОП, УМТС, показывает, что описание несложной по структуре БД (таблицы DBD и PCB) включает около 1000 строк. В каждой из прикладных программ используется в среднем 4—5 обращений к СУБД ОКА, каждое из которых составляет около 20 строк.

Анализ структуры прикладных программ, построенных на основе БД, показывает, что их можно разделить на отдельные блоки (описание,

поиск, манипуляции над данными, вывод), каждый из которых поддается формальному описанию, что позволяет использовать принципы генерации для их автоматического построения.

Анализ описаний наборов данных СУБД СЕТОР (СЕТКА, СЕДАН) и обращений к системе дает следующие данные: процедура вставки 10 строк (145 знаков); процедура удаления 8 строк (137 знаков); процедура замены 11 строк (160 знаков); процедура поиск/чтение 7 строк (120 знаков); описание на языке описания данных для основного набора данных 5 строк (80 знаков) и для зависимого 5 строк (160 знаков).

Мы рассмотрим одну из важных частей инструментальных средств проектирования БД — автоматизацию проектирования описаний БД, описаний структуры БД для прикладных программ и описаний обращения к СУБД из прикладных программ, покажем возможность формализации процесса проектирования указанных описаний и создания на этой основе САП основных процедур работы с БД [1].

Принципы проектирования, как и сама САП, ориентированы на СУБД ОКА, хотя нет принципиальных трудностей для их применения к другим СУБД.

САП процедур работы с БД использует два программных средства: систему разделения времени и универсальный макрогенератор. Система разделения времени (СРВ) является наиболее эффективным программным средством проектирования в связи с тем, что она позволяет с дисплея (абонентского пункта) использовать команды СРВ для ввода, корректировки таких данных, как характеристики БД, сегменты с их характеристиками и совокупностью полей, поля данных с их характеристиками и т. д.

Универсальный макрогенератор дает возможность оформлять в виде макроопределений произвольный текст (этим текстом может быть и набор операторов любого языка программирования). Такое макроопределение может затем вызываться посредством непроцедурного макровызова, имеющего упрощенный синтаксис. При вызове текст макроопределения может модифицироваться в соответствии с заданными значениями операндов макровызова, что позволяет очень эффективно использовать универсальный макрогенератор для генерации процедур работы с БД.

Генерация описания физической, логической базы данных и блоков связи прикладной программы. Блок DBD физической БД включает предложения SEGM, описывающие характеристики сегментов и предложения FIELD, определяющие характеристики полей, содержащихся в сегментах.

В процессе проектирования используются следующие наборы данных.

1. Набор данных, определяющий иерархическую структуру физической БД с записью следующей структуры:

<имя исходного сегмента> <имя подчиненного сегмента>

2. Набор данных имен сегментов с именами полей, входящих в сегменты, с записью следующего вида:

<имя сегмента> <имена полей сегмента>

3. Набор данных имен полей с их характеристиками, запись которого имеет следующую структуру:

<имя поля> <характеристики поля>

Набор данных (1), определяющий иерархическую структуру физической БД, может автоматически получаться программой проектирования оптимальной сетевой структуры БД или вводиться с терминала или с перфокарт.

На основании рассмотренных данных генератор проектирует блок DBD физической базы данных.

При проектировании описания логической БД необходимо создать блоки DBD от одной или нескольких физических БД, на основании которых создается логическая БД. В описании физических БД могут появляться предложения LCHILD, описывающие сегменты, которые связаны с сегментами той же или другой физической БД, и которые являют-

ся логически исходными. Блок DBD для логической БД включает также предложения SEGM.

Входной информацией для универсального макрогенератора при проектировании описаний логической БД являются те же наборы данных (1), (2) и (3), что и для проектирования описаний физической БД, а также набор данных о связях между сегментами с записями следующей структуры:

<имя БД-1> <имя сегмента> <имя БД-2> <имя сегмента>

Для генерации блоков PCB программист с терминала задает последовательность имен сегментов. Блок PCB генерируется, транслируется и загружается, а программисту на экран выдается имя PCB (если он его сам не указал).

Используются следующие наборы данных: 1) структура всех ФБД, ЛБД в форме, удобной для просмотра; 2) набор данных, содержащий имена уже созданных PCB.

Универсальный макрогенератор из операторов подъязыка данных BETA в СУБД ОКА порождает наборы операторов, реализующих манипуляции с БД. Подъязык данных BETA является надстройкой над одним из базовых языков программирования (ПЛ/1, КОБОЛ или АССЕМБЛЕР) и представляет собой набор условных операторов. Операторы считаются условными потому, что программист не использует их в явном виде, а сразу в прикладной программе записывает их в виде набора операторов базового языка программирования.

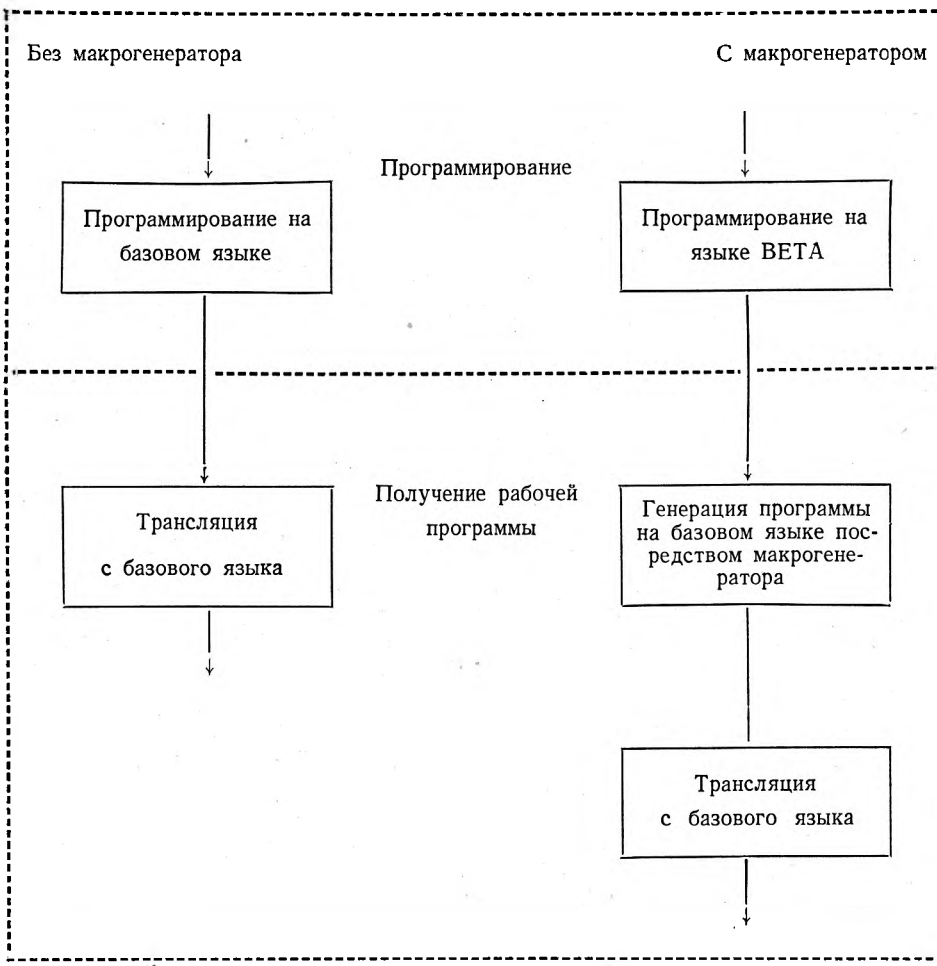

Рис. 1. Схема процесса разработки прикладной программы, работающей с базами данных

Обращение к СУБД для выполнения требуемой функции происходит с помощью операторов базового языка. В связи с тем, что обращение к СУБД на базовом языке имеет типовую структуру и стандартный состав операторов, имеется возможность использовать универсальный макрогенератор для получения такого обращения (рис. 1).

Язык BETA составляют операторы, реализующие операции над базами данных, и определяющие блок связи с программой (PCB). При программировании на базовом языке для реализации каждой операции над базами данных программист должен написать несколько операторов присваивания, устанавливающих аргументы поиска массива, и вызов программы, реализующей операцию над данными. При использовании универсального макрогенератора для этого достаточно написать один макровызов, а макрогенератор построит для него нужный набор операторов на базовом языке. Операторы, реализующие операции над базами данных, записываются в стандартном формате:

<оператор> ::= <код> <код языка> <пробелы> <операнд I> [<, операнд N>...]

<код> ::= GUOKA|GNOKA|GNPOKA|GHUOKA|GHNOKA|
GHNPOKA|ISRTOKA|DELOKA|REPOKA

<пробелы> ::= <пробел> [<пробел>...]

<операнд 1> ::= название PCB

<операнд N> ::= (<название поля>, <'значение поля'>)

<код языка>::=P|K (P — для ПЛ/1, K — для КОБОЛА)

Код определяет операцию над базами данных ОКА. В принципе синтаксис операторов может быть выбран таким, чтобы определяемым языком программирования не только удобно было пользоваться программисту, но и чтобы генерация на базовый язык программирования шла максимально эффективно. В данном случае операнды оператора задаются в виде списков, каждый из которых соответствует сегменту базы данных, первый элемент задает сегмент базы данных, второй — поле сегмента, третий — аргумент поиска сегмента (АПС — SSA). Третий или второй и третий элементы списка могут опускаться.

Кроме операций над базами, нужно определить аргумент поиска сегмента. Оператор для определения SSA может иметь следующий синтаксис:

<оператор> ::= <DCLOKA> <код> <пробелы> <имя PCB> <, операнд 1> [<, операнд N>...]

<операнд 1> ::= (<название SSA>, <длина области, выделяемой для SSA>)

<операнд N> ::= (<имя поля> , <тип поля> , <длина поля> , <'значение поля'>)

<имя поля> ::= <имя переменной, которая используется для определения аргумента SSA>

<тип поля> ::= тип переменной

<длина поля> ::= длина области, выделяемой для аргумента SSA.

Определение АПС требует обычно задания нескольких структур, является громоздким и поэтому оператор DCLOKA упрощает его задание. В пределах одной программной системы во многих программах используются одинаковые АПС. Такие АПС выгодно оформить в виде макроопределений и вызывать их посредством универсального макрогенератора, что значительно сокращает объем прикладной программы.

В качестве примера рассмотрим БД с физической структурой, изображенной на рис. 2.

На рис. 3 приведен пример программы, выбирающей сегменты ДЕТ, которые содержат в поле КД значения 'М' при условии, что деталь относится к группе изделия 'NEW'.

Предполагая, что PCB для данного примера оформлено в виде макроопределения и вызывается посредством макровызова PCBIOKA, этот фрагмент программы при использовании универсального макрогенератора будет выглядеть следующим образом:

```
PCBIOKA
. . .
GUOKA_PCBN1, (GI,' NEW'__'),(KD,'M')
```

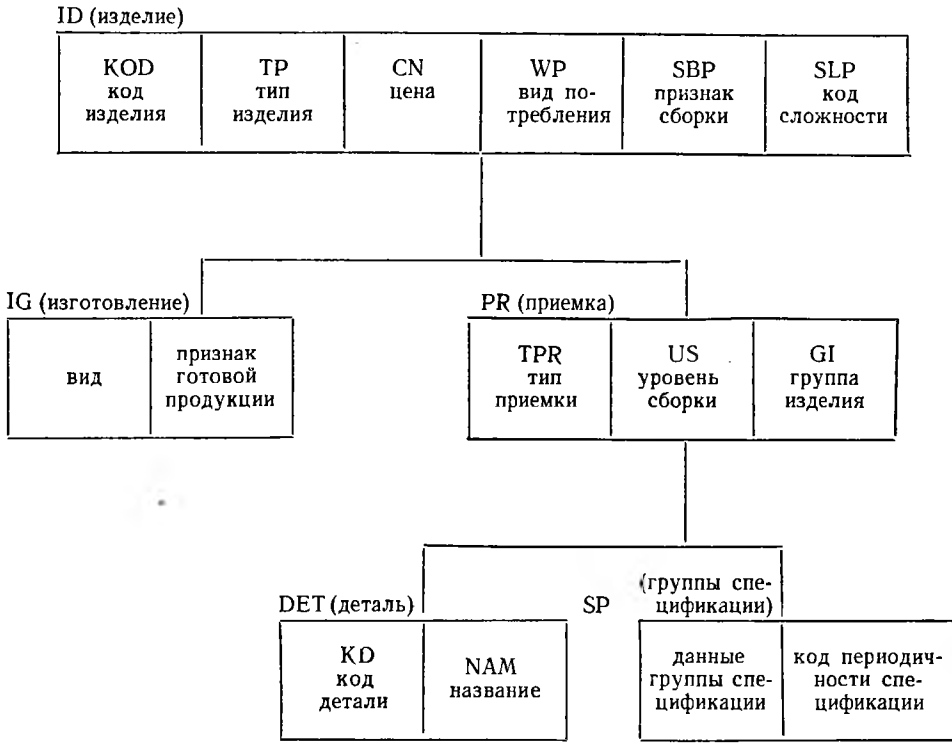

Рис. 2. Структура базы данных

```
BETTPLI: PROC(BASE1) APTIONS(MAIN);
    DCL 1 PCBN1 BASED(BASE1),
          2 DCBDN CHAR(8),
          2 SEGLEV CHAR(2),
          2 STATUS CHAR(2),
          2 PRCOPOT CHAR(4),
          2 SEGNAME CHAR(8),
          2 LENGKEY FIXED BIN(31, 0),
          2 NUMBSEG FIXED BIN(31, 0);
    DCL BASE1 POINTER;
    DCL KOLPAR FIXED BIN(31, 0) INIT(5) STATIC;
    DCL OPERATION CHAR(4) INIT('GU___');
    DCL REZULT CHAR(23);
    DCL SEG1 CHAR(23);
        1 S1 DEF SEG1 UNAL,
          2 KOD1 FIXED(14),
          2 TP1 CHAR(1),
          2 CN1 FIXED(4),
          2 WP1 CHAR(1),
          2 SBP1 FIXED(2),
          2 SLP1 CHAR(1);
    DCL SEG2 CHAR(8),
        1 S2 DEF SEG2 UNAL,
          2 TPR2 CHAR(1),
          2 US2 FIXED(2),
          2 GI2 CHAR(5);
    DCL SEG3 CHAR(7),
        1 S3 DEF SEG3 UNAL,
          2 KD3 CHAR(1),
          2 NAM3 CHAR(6);
          . . .
```

```
    GI2='NEW___';
    KD3='M';
CALL PLITBETT(KOLPAR, OPER, PCBN1, SEG1, SEG2, SEG3);

END1: END BETTPLI;
```

Рис. 3. Фрагмент программы работы с БД на ПЛ/1

Макроопределения, соответствующие приведенным операторам языка BETA, должны программироваться на языке универсального макрогенератора с использованием базового языка программирования. Без больших сложностей программировать макроопределения можно только, используя эффективные макрогенерации, которыми обладает универсальный макрогенератор [2]: средства обработки переменного количества операндов макровызова, использование глобальных индексированных переменных разных типов, возможность удобной обработки элементов списка, записанного в качестве операндов макровызова.

Средства базовых языков высокого уровня (ПЛ/1, КОБОЛ) не имеют подобных возможностей и поэтому не могут выполнять препроцессорную обработку макроопределений, соответствующих операторам BETA. Языки программирования КОБОЛ и ПЛ/1 позволяют только вызывать определенные наборы операторов языка, но возможности преобразования этих наборов сильно ограничены. Поэтому средства этих языков создают определенные удобства при работе с базами данных при вызове заготовленных часто используемых типовых процедур работы с БД, но практически бесполезны в прикладных программах при разнообразных логических базах данных. Для автоматического проектирования таких процедур эффективным средством является универсальный макрогенератор.

ЛИТЕРАТУРА

1. Дравица В. И., Змитрович А. И., Лепешинский Н. А. Об одной системе автоматизированного проектирования баз данных: Первая Всесоюзная конференция «Банки данных».— Тбилиси, 1980.
2. Воюш В. И.— Программирование, 1979, № 4.

Поступила в редакцию 03.02.81.

Кафедра математического обеспечения АСУ

УДК 517.966

А. А. ЛЕВАКОВ

НЕКОТОРЫЕ СВОЙСТВА МНОГОЗНАЧНЫХ ОТОБРАЖЕНИЙ И ОДНА ТЕОРЕМА СУЩЕСТВОВАНИЯ РЕШЕНИЙ ДИФФЕРЕНЦИАЛЬНЫХ ВКЛЮЧЕНИЙ

Пусть E — сепарабельное банахово пространство; $T=[t_0,\ t_1]$ — отрезок в R; μ — мера Лебега на T, $\varkappa$ — множество всех непустых замкнутых подмножеств E; $\alpha(A,\ B)=\max\{\sup\limits_{x\in A}\rho(x,\ B),\ \sup\limits_{y\in B}\rho(A,\ y)\}$ — отклонение по Хаусдорфу множеств $A,\ B\in\varkappa$. Последовательность (A_n), $A_n\in\varkappa$ называют сходящейся, если существует $A\in\varkappa$ такое, что $\alpha(A_n,\ A)\to 0$. Как известно [1], если $\alpha(A_n,\ A_m)_{n,\ m\to\infty}\to 0$, то последовательность (A_n) является сходящейся. Последовательность многозначных отображений $(A_n(\cdot))$, $A_n(\cdot): J\to\varkappa$, $J\subset T$, назовем сходящейся почти при всех $t\in J$ к $A(\cdot)$ (равномерно на J к $A(\cdot)$), если $\alpha(A_n(t),\ A(t))\to 0$ почти при всех $t\in J$ (равномерно на J). Функцию $a(\cdot): T\to E$ называют сечением многозначного отображения $A(\cdot): T\to\varkappa$, если $a(t)\in A(t)$ почти при всех $t\in T$. Многозначное отображение $A(\cdot): T\to\varkappa$ назовем интегрируемым, если существует счетное семейство $\{x_n(\cdot)\}$, $n=1,\ 2,\ldots$, интегрируемых по Бохнеру сечений отображения $A(\cdot)$ таких, что $\{x\in E\mid x=x_n(t),\ n=1,\ 2,\ldots\}$ плотны в $A(t)$ почти при всех $t\in T$. Интегралом $\int\limits_{t_0}^{t_1}A(t)\,d\mu$ называем мно-