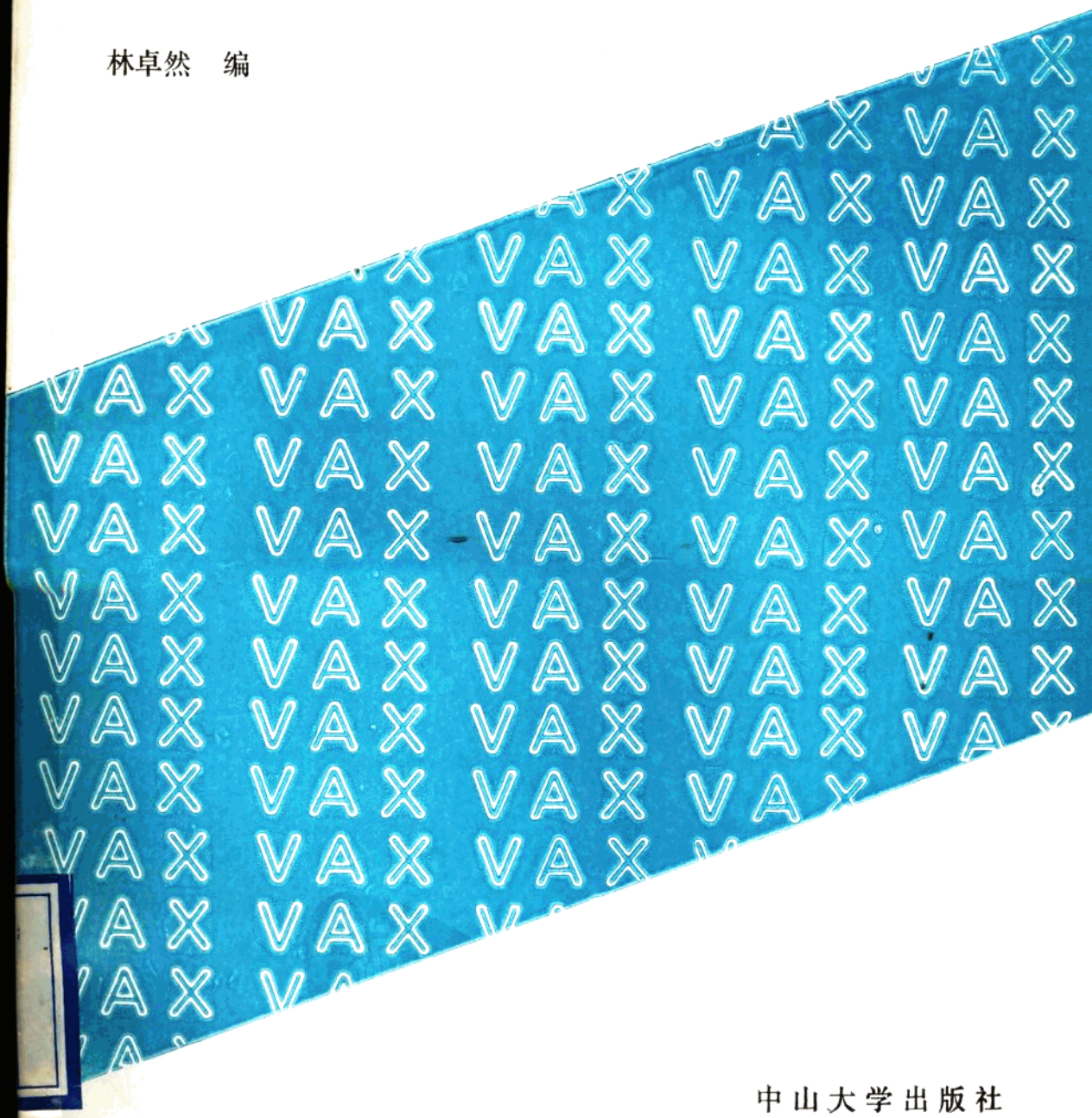
VAX Rdb/VMS

关系数据库使用入门

林卓然 编



中山大学出版社

内 容 简 介

本书介绍 VAX 系列机的关系数据库管理系统（Rdb／VMS），内容主要包括表达式、数据定义、数据操作、出错处理、数据库维护与管理、优化处理等。

本书在内容上既注意全面、系统，又力求通俗易懂。为方便读者掌握，本书给出了大量的设计实例。

本书可作为数据库使用培训班的教材，也可作为 VAX 系列机用户的参考书。

出版说明

为了适应广大 VAX 系列机用户学习的需要，中山大学出版社组织了有关人员编写 VAX 系列机入门丛书．丛书力求叙述简明易懂，突出应用和操作技巧，便于读者自学．

本丛书内容包括：

(1)《VAX　Rdb／VMS 关系数据库使用入门》；(2)《VAX　CFMS 中文表格管理系统使用入门》；(3)《VAX／VMS 操作系统入门》；(4)《VAX　BASIC 使用与技巧》．各本书在内容上相互独立，读者可根据需要来选用．

本丛书经肖金声教授审阅．中山大学计算中心李玉标主任对丛书的编写给予了大力支持和帮助．在此表示谢意．

由于时间仓促，错误、不足之处在所难免，敬请读者及同行指正．

1990.5.

目　　录

第一章　概述 …… (1)
§1.1　数据处理系统发展的三个阶段 …… (1)
§1.2　数据库系统的基本概念 …… (1)
§1.3　关系数据库模型 …… (2)
§1.4　使用 VAX Rdb/VMS 的初步认识 …… (4)
第二章　表达式 …… (12)
§2.1　数据类型 …… (12)
§2.2　值表达式 …… (18)
§2.3　条件表达式 …… (23)
§2.4　记录选择表达式 …… (26)
第三章　数据定义语句 …… (30)
§3.1　数据库定义 …… (30)
§3.2　字段定义 …… (32)
§3.3　关系定义 …… (36)
§3.4　窗口定义 …… (39)
§3.5　索引定义 …… (40)
§3.6　约束定义 …… (42)
§3.7　保护定义 …… (43)
第四章　数据操作语句 …… (49)
§4.1　调用数据库 …… (49)
§4.2　关系操作和形成记录流 …… (52)
§4.3　事务处理 …… (56)
§4.4　存贮记录 …… (59)
§4.5　修改记录 …… (59)
§4.6　消除记录 …… (60)
§4.7　检索记录 …… (61)
§4.8　处理分段串 …… (62)
第五章　出错处理 …… (66)
§5.1　错误的检测 …… (66)
§5.2　显示错误信息 …… (67)
§5.3　错误的陷井处理 …… (68)
§5.4　错误恢复 …… (73)
第六章　实例介绍——处理对照表数据库文件 …… (77)
§6.1　定义数据结构 …… (78)
§6.2　增、改、删数据库中的记录 …… (79)
§6.3　把代码换成中文名的外部函数 …… (85)

第七章　交互控制语句 …… (88)
第八章　数据库维护和管理语句 …… (91)
第九章　改善数据库的性能 …… (97)
§9.1　VAX Rdb/VMS 的内部特征 …… (97)
§9.2　使用 ANALYZE 语句来分析数据库性能 …… (98)
§9.3　使用 RDMS$DEBUG_FLAGS 来决定优化的策略 …… (99)
§9.4　设定有关系统参数以获得好的性能 …… (101)
§9.5　设定 Rdb/VMS 数据库的参数 …… (102)
§9.6　并行事务处理的控制 …… (103)
附录 …… (104)
1. VAX Rdb/VMS 保留字 …… (104)
2. Rdb/VMS 数据操作语句 …… (105)

第一章　概 述

§1.1　数据处理系统发展的三个阶段

目前，在整个电子计算机应用领域中，数据处理是最大的一个应用方面．据统计，在计算机的全部工作量中数据处理约占80%，而且这个范围还在不断扩大．计算机数据处理几乎渗透到人类生活和生产的各个领域，例如，人事档案管理，财务会计管理，设备库存管理，情报资料管理，医院病案管理等等．

数据处理系统主要经历了三个阶段．

第一阶段是人工数据管理系统阶段．这一时期主要是50年代到60年代初，当时计算机还主要用于科学计算，实际上没有专门的软件系统对数据进行管理．计算机本身只作为一种计算工具，一个程序面向一种数据．进行程序设计时也要对数据结构、存贮方式、输入输出方式进行设计．

第二阶段是文件系统阶段．这一时期计算机不仅用于科学计算，也用于管理工作中．尤其是在磁盘一类的大容量外存贮器使用以后，产生了专门管理数据文件的软件系统．这时期的数据能够以文件形式长期保存在计算机的外存贮器中，可以随时对数据进行查询、修改、增删等处理．程序和数据有了一定的独立性，不必再全部由编程人员考虑数据的物理结构，从而实现了以文件为单位的数据共享．但是，所有这些进展都是局限在一定范围内的，有些根本性问题仍没有解决．存在的主要问题是，由于数据和应用程序间仍相互依赖，数据并不完全独立，数据结构一改变，应用程序也必须相应改变．正因为这样，各用户建立自己的文件，文件内部数据并不能完全共享，造成大量重复和存贮空间的浪费．此外，对文件没有统一管理机构，其安全性和完整性等得不到很好的保证．这些问题都有待于软件的进一步发展去解决．

到60年代末出现了数据库管理系统，初期的数据库管理系统正是为了克服文件系统的缺点而设计的．数据库管理系统阶段是数据处理系统发展的第三个阶段．

§1.2　数据库系统的基本概念

数据库可以定义为：以一定的组织方式存贮在一起的相互有关的数据的集合．它能以最佳的方式，最少的重复为多个用户或应用程序服务．而数据的存贮与使用数据的应用程序无关，数据和程序都有较高的独立性；数据的增加、更新和检索等使用通用的控制方式．

数据库系统是在文件管理系统的基础上发展起来的，两者区别主要在于：文件中的数据只能为某些特定应用程序所独享，而数据库中的数据却可以用不同的组合形式同时为多个应用程序或用户共享．

一般地把数据库仅仅理解为上述的数据集合，而数据库管理系统则指的是对其进行管理的软件系统．它包括数据描述语言及其翻译程序，数据操作语言及其编译程序，数据管理例行程序等．

§1.3 关系数据库模型

数据库的设计方法很多，较为流行的是层次方法、网络方法和关系方法．VAX Rdb／VMS是属于后一种的数据库管理系统．

关系方法是发展较晚的一种数据库方法．它的数据模型称为关系模型．在这种模型中，数据均以二维表的形式出现．每个二维表称为一个关系．例如，表1-1便是一个关系．

表1-1 课程关系

编 号	课程名称	学分	总学时	周学时
011011	文学概论	4	68	3
011021	中国现代文学史	2	34	2
011030	语言学概论	4	51	3
011041	现代汉语	4	51	3
011501	写作	3	51	3
⋮	⋮	⋮	⋮	⋮

表1-2表示这种二维表的一般形式．

表1-2 关系的一般形式

A_1	A_2	……	A_n
V_{11}	V_{12}		V_{1n}
V_{21}	V_{22}		V_{2n}
⋮	⋮		⋮
V_{m1}	V_{m2}		V_{mn}

每个关系均有一个名称，称为关系名，例如表1-1，我们可以命名它为课程关系．一个二维表由行和列组成，表中的一个列相当于一个数据项，代表一个属性，在数据库中称它为一个字段（FIELD）．每个字段必须具有相同的数据类型．横向的一行为一个元组，相当于一个记录．

关系方法的主要特点是表现在它的数据描述的统一性．就是说，描述的对象与对象之间的联系等都只能用关系来表示．而关系本身必须是规范化的．它的每个列必须是基本数据项．如表 1-3，书类不是基本数据项，而是由文、理、工、农、医五个数据项组成的组合项．因此，这个表不符合关系的二维条件．

表 1-3

书 名	书　　类				
	文	理	工	农	医

关系方法具有严格的数学基础，它对数据的各种处理主要以集合代数为根据．

下面我们举出一个简单的例子．表 1-4 和表 1-5 列出了学生的学籍关系(STATUS)和成绩关系(RESULT)，这两个关系可简单地构成学生资料数据库(定名为 STLIB)．

表 1-4　学籍关系 (STATUS)

学号 (NO__X)	姓名 (NAME__C)	性别 (SEX__C)	出生年月 (BORN__X)	班号 (CNO__X)
70101	林业章	男	6905	701
70102	何　青	男	7011	701
70103	罗星星	女	7007	701
70205	陈大克	女	7001	702
70207	黄木材	男	6912	702
70210	陈广华	男	7009	702

注：　学号＝入学年份 ＋ 班序号 ＋ 学生序号

表 1-5　成绩关系 (RESULT)

学号 (NO__X)	语文 (CHIN__L)	数学 (MATH__L)	英语 (ENG__L)
70101	93	100	95
70102	87	75	56
70103	58	69	79
70205	68	87	93
70207	73	57	80
70210	89	99	84

当然，作为数据库管理系统，不仅要提供建立数据库的手段，还要提供许多有关的其他操作，如处理数据库（如增、删、改数据库中的记录），供用户查询、输出查询信息和报表等。

§1.4 使用 VAX Rdb／VAX 的初步认识

VAX Rdb／VMS 有下列四类语句：

数据定义语句（见第三章）

数据操作语句（见第四章或附录 2）

交互控制语句（见第七章）

数据库维护及管理语句（见第八章）

使用这些语句来进行数据库操作和管理，可以通过以下三种途径：

(1) 使用 CRDO（Chinese Relation Database Operator，中文关系数据库操作）交互实用程序。CRDO 提供了用户与 Rdb／VMS 的会话式环境，在这一环境下，可以使用几乎所有的 Rdb／VMS 语句。

(2) 使用高级语言预编译程序（precomp）。在这种情况下，可以把 Rdb／VMS 数据操作语句放在高级语言的程序中。

(3) 使用可调用的 CRDO 程序接口。如果 Rdb／VMS 没有为使用的语言提供预编译程序，那么就必须使用可调用的 CRDO 程序接口。在这种情况下，应用程序将包含一系列被命名为 RDB$INTERPRET 的过程调用。

在本书中，主要介绍 CRDO 会话操作和预编译程序设计。

1. CRDO 会话操作

在 DCL 命令状态下，打入

```
$CRDO
```

（注：CRDO 是系统定义的全局符号，其含义是调用 CRDO 交互实用程序）

便可进入 CRDO 会话状态，其提示符如下：

```
CRDO>
```

在此状态下，当用户打入 Rdb／VMS 语句时，CRDO 就会立即解释执行。[说明：在CRDO环境下，也可以进行 RDO（英文关系数据库操作）的所有操作]

要退出 CRDO 状态，有两种方法，第一种方法是使用以下命令

```
CRDO>EXIT
```

第二种方法是按下 CTRL／Z 功能键。退出 CRDO 状态后，系统将返回到 DCL 命令状态。

例 1 定义学生资料数据库（STLIB，见§1.3），操作如下：

```
CRDO>DEFINE  DATABASE  'STLIB'.
```

执行后将在缺省目录里定义一个名为"STLIB"的数据库（缺省类型为.RDB）。因未定义该数据库的数据结构，故还不能立即使用它。

注意，本语句后随的句点"."是需要打入的。CRDO 规定，所有的数据定义语句都必

须以句点结束。

例 2 定义数据库 STLIB 的字段，操作如下：

```
CRDO>DEFINE FIELD NO_X DATATYPE IS TEXT SIZE IS 5.
CRDO>DEFINE FIELD NAME_C DATATYPE IS TEXT SIZE IS 8.
CRDO>DEFINE FIELD SEX_C DATATYPE IS TEXT SIZE IS 2.
CRDO>DEFINE FIELD BORN_X DATAYPE IS TEXT SIZE IS 4.
CRDO>DEFINE FIELD CNO_X DATATYPE IS TEXT SIZE IS 3.
CRDO>DEFINE FIELD CHIN_L DATATYPE IS SIGNED LONGWORD.
CRDO>DEFINE FIELD MATH_L DATATYPE IS SIGNED LONGWORD.
CRDO>DEFINE FIELD ENG_L DATATYPE IS SIGNED LONGWORD.
CRDO>COMMIT
```

上述操作采用 DEFINE FIELD 语句定义了八个字段，指定各字段的名称和数据类型，其中有 5 个字符串（TEXT）字段和 3 个带符号的长字整数（SIGNED LONGWORD）字段。对于字符串字段，还规定其长度，如 SIZE IS 2 表示长度为 2。

最后一个语句 COMMIT 是提交处理，它将前面语句所进行的操作结果，写入到数据库中。

例 3 定义数据库 STLIB 的学籍关系 STATUS，操作如下：

```
CRDO>DEFINE RELATION STATUS.
Cont>NO_X.
Cont>NAME_C.
Cont>SEX_C.
Cont>BORN_X.
Cont>CNO_X.
Cont>END STATUS RELATON.
CRDO>COMMIT
```

说明：上述操作采用 DEFINE RELATION 语句定义了关系 STATUS。当打入第一行后，因语句未完，系统会自动显示出续行提示符“Cont>”，直至接收 END STATUS RELATION 为止。

例 4 定义数据库“STLIB”的成绩关系 RESULT，操作如下：

```
CRDO>DEFINE RELATION RESULT.
Cont>NO_X.
Cont>CHIN_L.
Cont>MATH_L.
Cont>ENG_L.
Cont>END RESULT RELATION.
CRDO>COMMIT
```

关于 CRDO 会话操作，以下还要说明几点：

(1) 在 CRDO 会话状态下，用户也可以使用 DCL 命令，只要以字符“$”开头就行了，如

```
CRDO> $DIR  *.RDB
```

(2) 上述例 1～例 4 都是在 CRDO 会话状态下直接打入语句来执行功能的．但这种方法不易修改输错的字符，也不易保存所需的语句系列以供重复使用，因此常用的是另一种方法，即首先编辑一个 CRDO 命令文件（后缀为.RDO），如

```
CRDO> $HEDT  CRELIB.RDO
DEFINE  DATABASE  "STLIB.RDB".
DEFINE  FIELD  NO_X  DATATYPE  IS  TEXT  SIZE  IS  4.
```

：　　} 见例 2～例 4 所用语句

```
END  RESULT  RELATION.
COMMIT
CTRL/Z
*EXIT
CRDO>
```

然后再执行命令文件，操作如下：

```
CRDO>@CRELIB
```

这样也能达到例 1～例 4 的操作效果．

(3) 常用的两个 CRDO 命令如下：

① HELP：求助，如

```
CRDO>HELP  DEFINE
```

可以了解 DEFINE 语句使用方法．

② SHOW：显示，如

```
CRDO>SH  RELATION
```

可以显示出最近启用的数据库中的关系设置情况．又如

```
CRDO>SH  FIELDS
```

可以显示出最近启用的数据库中的字段设置情况．

建立了数据库的数据结构后，就可以对它进行数据存取和管理．

例 5　为学籍关系“STATUS”输入数据，使用的语句如下：

```
INVOKE DATABASE FILENAME 'STLIB'
START_TRANSACTION READ_WRITE
STORE S IN STATUS USING
S.NO_X = "70 101";
S.NAME_C = "林业章";
S.SEX_C = "男";
S.BORN_X = "6905";
S.CNO_X = "701"
END_STORE
COMMIT
FINISH
```

说明：(1) 上述语句执行后，将在关系"STATUS"上存入一个记录。要存入多个记录，可以多次执行上述语句所组成的命令文件，只要在执行之前把数据改动一下就行了。关于存贮数据的更好方法，以后再介绍。

(2) 首行的 INVOKE DATABASE 语句的作用是调用数据库。当不再使用数据库时可以采用 FINISH 语句来关闭数据库（见末行）。

(3) START_TRANSACTION 语句用来启动一个事务处理，同时指明当前进行的是读/写（READ_WRITE）操作。在 Rdb/VMS 中，提供只读（READ_ONLY）和读/写（READ_WRITE）两种操作，没有单独写（WRITE）操作，故此例选择 READ_WRITE。

(4) 一个事务处理采用 COMMIT 语句（见本例）或 ROLLBACK 语句来结束。COMMIT 能结束一个事务处理，同时如果在事务期间（从 START_TRANSACTION 开始）已作过改动数据库的操作，则将写入到数据库中去。ROLLBACK 也能结束一个事务处理，但它将取消从 START_TRANSACTION 开始以来对数据库所做过的所有改动。

(5) STORE 语句把记录存入关系 STATUS 中，存贮数据到 END_STORE 为止。S 为上下文变量，代表关系 STATUS。因为在一个事务处理中可能用到多个关系中的字段，为区别起见，必须为每个字段加上相应的上下文变量，以此表示某字段是属于哪个关系的。上下文变量名可以任选，习惯上采用单个字母来表示，如 S、R、A、B 等。

模仿例 5 的做法，我们也可以为成绩关系 RESULT 输入数据。

例 6 读取学籍关系 STATUS 中的所有记录和显示出来，采用的语句如下：

```
INVOKE DATABASE FILENAME 'STLIB'
START_TRANSACTION READ_ONLY
FOR D IN STATUS
PRINT D.*
END_FOR
COMMIT
FINISH
```

说明：(1) 第 2 行的 START_TRANSACTION 语句用来启动一次只读事务处理．对于本例此语句也可以省略．当没有发出 START_TRANSACTION 语句就执行数据操作语句（如 FOR 语句）时，Rdb／VMS 会自动启动一个只读事务处理．在这种情况下，允许简单的数据检索，但不能进行写操作．

(2) 第 3 行的 FOR 语句形成了包括关系 STATUS 中所有记录的记录流．PRINT 为输出语句，它用来输出记录流中每一个记录的所有字段值．D.* 表示记录中所有字段．

例 7 显示出 702 班学生的学号及姓名，采用的语句如下：

```
INVOKE DATADASE FILENAME "STLIB"
FOR S IN STATUS WITH S.CNO_X = "702"
PRINT S.NO_X, S.NAME_C
END_FOR
COMMIT
FINISH
```

说明：上述 FOR 语句形成了关系 STATUS 中班号为"702"的记录流．WITH 子句指明了形成记录流的条件．PRINT 语句输出该记录流中每个记录的学号及姓名．

2. 程序设计

例 8 假设已通过 CRDO 会话方式定义了数据库"STLIB"及其数据结构（见例 1～例 4），现要为该数据库的成绩关系（RESULT）输入数据．编写的 BASIC 预编译源程序如下：

```
10 ! 调用数据库
   &RDB& INVOKE DATABASE FILENAME "STLIB"
   ! 输入数据
   INPUT_LOOP:
     WHILE -1%
       INPUT "学号="; ST.NO.X$
       EXIT INPUT_LOOP IF ST.NO.X$ = "0"
       INPUT "语文="; CHIN.L%
       INPUT "数学="; MATH.L%
       INPUT "英语="; ENG.L%
       ! 开始一个事务处理
       &RDB& START_TRANSACTION READ_WRITE
       ! 存入记录
       &RDB& STORE S IN RESULT USING      &
         S.NO_X=ST.NO.X$;                 &
         S.CHIN_L=CHIN.L%;                &
         S.MATH_L=MATH.L%;                &
         S.ENG_L=ENG.L%
       &RDB& END_STORE
```

```
        ! 结束事务处理及提交处理
        &RDB&  COMMIT
      NEXT
    ! 关闭数据库
    &RDB&  FINISH
200 END
```

关于上述程序，说明如下：

(1) 要进行 Rdb／VMS 数据操作，必须使用一种主程序语言（简称主语言），主语言可以是 BASIC、COBOL、FORTRAN 或 PASCAL．上述源程序是采用 BASIC 作为主语言的．

(2) 这种源程序由 Rdb／VMS 数据操作语句及主语言语句组成．主语言语句按语言规定编写，而 Rdb／VMS 语句则必须加上&RDB&语句标志（PASCAL 不需要）．如

```
&RDB&  COMMIT
```

此外规定语句标志&RDB&应是程序行中第一个非空字符．在 BASIC 中，不允许在语句标志前加行号，如

```
10  &RDB&  ……
```

是错误的．在同一个程序行中，Rdb／VMS 语句和主语言语句不能混用．

(3) 在某些场合下，可以使用语言续行字符．对于 BASIC 语言，续行字符为“&”，例如

```
&RDB&  STORE  SH  IN  SALARY_HISTORY  USING
&RDB&  SH.EMPLOYEE_ID=EMP.ID
&RDB&  END_STORE
```

可以写成

```
&RDB&  STORE  SH  IN  SALARY_HISTORY  USING      &
SH.EMPLOYEE_ID=EMP.ID                            &
END_STORE
```

对于 COBOL，续行符号为短下横线“_”，如

```
&RDB&  STORE  SH  IN  SALARY_HISTORY  USING
-              SH.EMPLOYEE_ID=EMP.ID
-              END_STORE
```

(4) 上述源程序必须由一种称为预编译程序（由系统提供）来完成编译，故也称为预编译源程序．预编译程序能调用主语言的编译程序来编译源程序中的主语言语句，也能处理 Rdb／VMS 语句．它把 Rdb／VMS 语句转换成一系列子程序调用，并把这些子程序调用放入编译的语言代码中．

预编译源程序的文件类型为．RBA（BASIC）、．RCO（COBOL）、．RFO（FORTRAN）和．RPA（PASCAL）．设上述源程序名为 PROG1，使用的有关操作命令如下（在 DOL 命令状态下打入）：

①编辑源程序的命令如下:

```
$HEDT  PROG1.RBA
```

②预编译命令如下:

```
$PREBAS  PROG1
```

PREBAS 是系统定义的全局符号，其含义是调用 BASIC 预编译程序。

编译后新产生三种输出文件。一是类型为.BAS 的源文件，它通常与原预编译源程序相关联，但不能编辑它和重新编译，它只是做为预编译过程中的一种中间产物而已。二是目标文件，类型为.OBJ，以后采用 LINK 命令连接的就是这个文件。三是错误文件，当预编译期间出错时，预编译程序会在终端上显示出这些错误，并且把错误写到一个名字为 RDBERR.LOG 的错误文件。

③连接命令如下:

```
$LINK  PROG1,〈中文系统模块〉
```

这跟往常的连接方法一样。

④运行命令如下:

```
$RUN  PROG1
```

这也跟往常做法相同。

COBOL 和 FORTRAN 的预编译程序在处理方法上与上述类似，唯独 PASCAL 不同。PASCAL 预编译程序只产生一个输出文件，类型是.PAS。它不能自动调用 PASCAL 编译程序，因此，在使用连接命令之前，用户必须采用 PASCAL 编译程序来编译输出文件（由预编译程序产生的）。

这里还要说明的是，本书中大多数程序都是由 BASIC 语言写成的，但所介绍的基本方法，对任何具有 Rdb / VMS 预编译接口的程序语言来说，都是适用的。

例 9 统计并输出语文科的最高分和平均分。程序如下:

```
10 !   调用数据库
  &RDB&  INVOKE  DATABASE  FILENAME  "STLIB"
20 ! 启动一个事务处理
  &RDB&  START_TRANSACTION  READ_ONLY
30 ! 统计语文科的最高分
  &RDB&  GET          &
           MAX.L%=MAX  E.CHIN_L  OF  E  IN  RESULT  &
         END_GET
40 ! 计算语文科的平均分
  &RDB&  GET      &
           AVER.S=AVERAGE  F.CHIN_OF  F  IN  RESULT  &
         END_GET
50 ! 结束事务处理
  &RDB&  COMMIT
```

```
60 ! 输出最高分和平均分
  PRINT  MAX.L%, AVER.S
70 ! 关闭数据库
  &RDB&  FINISH
  END
```

说明：例中 30 行的 MAX 和 40 行的 AVERAGE 都是 Rdb／VMS 统计表达式的关键字，其含意分别是计算关系 RESULT 中字段 CHIN_L 的最大值和平均分．GET 语句表示取值，30 行的 GET 取最大值存入程序变量 MAX.L%中，40 行的 GET 取平均分存入程序变量 AVER.S 中．这里的 MAX.L%和 AVER.S 也称为语言变量，或称主变量．

预编译程序设计几乎可以按照 CRDO 的相同语句格式来使用 Rdb／VMS 数据操作语句，但要注意，GET 语句只能在程序中用，PRINT 语句只能用在 CRDO 中，此外，当向数据库送值和从数据库取值时，程序中语句要用到主变量，见此例 30 行和 40 行．若采用 CRDO 会话方式，则实现 30 行语句功能的操作如下：

```
CRDO>PRINT  MAX  E.CHIN_L  OF  E  IN  RESULT
```

第二章　表达式

§ 2.1 数据类型

1. Rdb / VMS 数据类型

表 2-1　列出了 Rdb / VMS 支持的数据类型.

表 2-1

Rdb / VMS 数据类型	对应的 VAX 数据类型	大小	范围 / 精度	其他参数
SIGNED WORD	带符号的单字整数	16 位	−32768～32767	比例因子(SCALE) n 为带符号整数
SIGNED LONGWORD	带符号的长字整数	32 位	-2^{31}～$2^{31}-1$	比例因子(SCALE) n 为带符号整数
SIGNED QUADWORD	带符号的四字整数	64 位	-2^{63}～$2^{63}-1$	比例因子(SCALE) n 为带符号整数
F-FLOATING	F-单精度浮点数	32 位	近似 7 位十进制数	无
G-FLOATING	G-扩展精度浮点数	64 位	近似 15 位十进制	无
DATE	绝对日期和时间	64 位	不能用	无
TEXT	ASCII 文本	n 个字节	0～16383 个字符	n = 无符号整数. 字符数
VERYING TEXT 或 STRING	ASCII 文本	可变	0～16383 个字符	n = 无符号整数. 字符数
SEGMENTED STRING(分段串)	无	可变	0～64K 个字符	无

2. 数据类型的转换

(1) 预编译程序的数据类型转换

预编译程序给定了一系列变量，使之在主变量和数据之间起着过渡作用. 赋给这些变量的数据类型依赖于使用的数据库字段的数据类型，以及采用的是什么预编译程序. 表 2-2 列出了有关 BASIC、FORTRAN 及 PASCAL 预编译程序的数据类型转换. 表中没列出的 Rdb / VMS 数据类型，或列为"无"的都不加以转换. COBOL 预编译的程序不存在转换问题，因此，选择的程序数据类型要与编译程序赋给的数据类型相一致.