

# 目 录

第一章 概述 .....	1-1
1.1 什么是数据库系统 .....	1-1
1.2 数据库系统的特点 .....	1-2
1.3 数据库系统的典型结构 .....	1-6
1.4 数据语言 .....	1-7
1.4.1 数据描述语言 .....	1-7
1.4.2 数据操作语言 .....	1-8
1.5 数据库管理系统 .....	1-9
1.5.1 数据字典 ( <i>Data dictionary</i> ) .....	1-10
1.5.2 数据库管理员 ( <i>DBA</i> ) .....	1-11
1.5.3 用户访问数据库的过程 .....	1-12
1.6 实体—联系方法 .....	1-14
1.7 数据模型 .....	1-17
第二章 存储结构 .....	2-1
2.1 引言 .....	2-1
2.1.1 文件的基本概念 .....	2-1
2.1.2 数据库操作速度的估计 .....	2-4
2.1.3 指示器 ( <i>Pointer</i> ) .....	2-5
2.1.4 关键字 ( <i>Keys</i> ) .....	2-5
2.1.5 钉定 ( <i>Pinned</i> ) 和未钉定的记录 .....	2-6
2.1.6 文件结构概述 .....	2-7
2.2 顺序文件 .....	2-10
2.2.1 如何确定关键字值的顺序 .....	2-10
2.2.2 顺序文件的存储组织 .....	2-10
2.2.3 顺序文件的查找 .....	2-11

2.3	随机结构之一——散列方法	2-12
2.3.1	散列方法的简要回顾	2-12
2.3.2	散列文件的设计	2-15
2.3.3	可扩充的散列	2-16
2.4	随机结构之二——索引结构	2-21
2.4.1	索引顺序文件	2-21
2.4.2	索引无序文件	2-21
2.4.3	索引的组织	2-22
2.4.4	索引文件的查找	2-25
2.5	$B$ -树	2-25
2.5.1	二叉树	2-25
2.5.2	$B$ -树	2-27
2.5.3	$B+$ 树	2-29
2.5.4	一个 $B+$ 树实例	2-31
2.6	变长记录文件	2-33
2.7	倒排文件	2-35
第三章	关系方法	3-1
3.1	关系及基本术语	3-1
3.2	关系运算	3-3
3.2.1	关系代数	3-3
3.2.2	元组关系演算	3-9
3.2.3	域关系演算	3-12
3.3	关于数据库的数据操作语言	3-14
3.3.1	基于关系代数的语言 <i>ISBL</i>	3-15
3.3.2	介于关系代数与演算之间的语言 <i>SEQUEL</i>	3-19
3.3.3	基于元组演算的语言 <i>QUEL</i>	3-27
3.3.4	基于域演算的语言 <i>QBE</i>	3-32
3.4	关系数据库的模式和子模式	3-37
3.4.1	源模式、目标模式及其物理映射	3-37
3.4.2	子模式、目标子模式及其映射	3-41

3.5	询问的优化	3 - 45
3.5.1	优化的一般策略	3 - 46
3.5.2	关系代数表达式的等价代换规则	3 - 47
3.5.3	关系代数表达式的优化算法	3 - 48
第四章	层次方法	4 - 1
4.1	一般概念	4 - 1
4.1.1	树	4 - 1
4.1.2	层次系统的数据模型	4 - 3
4.1.3	层次顺序与层次路径	4 - 5
4.1.4	层次系统的模式与子模式	4 - 7
4.2	IMS 系统的逻辑结构	4 - 8
4.2.1	IMS 的逻辑结构	4 - 8
4.2.2	IMS 的 DBD	4 - 9
4.2.3	IMS 的 PSB	4 - 12
4.3	IMS 的存储结构	4 - 14
4.3.1	HSAM	4 - 14
4.3.2	HISAM	4 - 15
4.3.3	HJDAM 和 HDAM	4 - 19
4.4	IMS 的数据子语言	4 - 25
4.4.1	子语言 DL/1	4 - 25
4.4.2	IMS 的应用程序	4 - 30
4.4.3	应用程序的运行	4 - 35
4.5	IMS 存储结构补充	4 - 36
4.5.1	辅数据集组	4 - 36
4.5.2	IMS 辅助索引	4 - 38
4.5.3	IMS 的逻辑数据库	4 - 40
第五章	DBTG 建议的网状模型的数据库系统	5 - 1
5.1	DBTG 系统的结构	5 - 1

5. 2	DBTG的数据模型 .....	5 - 2
5.2.1	记录类型 .....	5 - 2
5.2.2	络类型 (Set type) .....	5 - 3
5.2.3	络事件 (Set occurrence) .....	5 - 5
5.2.4	事物联系的DBTG表示法 .....	5 - 7
5. 3	记录类型描述及其存储映射 .....	5 - 10
5.3.1	DBTG句法使用的符号 .....	5 - 10
5.3.2	记录类型的描述 .....	5 - 11
5.3.3	记录类型的存储映射 .....	5 - 13
5.3.4	记录类型举例 .....	5 - 16
5. 4	络类型描述及其存储映射 .....	5 - 17
5.4.1	络类型 (Set mode) .....	5 - 17
5.4.2	络次序 (Set order) .....	5 - 18
5.4.3	从记录类型性质的描述 .....	5 - 23
5.4.4	络选择 (Set selection) .....	5 - 24
5.4.5	络类型举例 .....	5 - 25
5. 5	模式数据描述 .....	5 - 29
5. 6	子模式数据描述 .....	5 - 29
5.6.1	子模式与模式的区别 .....	5 - 30
5.6.2	子模式举例 .....	5 - 30
5. 7	数据操作语言 (DML) .....	5 - 32
5.7.1	程序的运行 .....	5 - 32
5.7.2	DML语句概述 .....	5 - 33
5.7.3	应用程序举例 .....	5 - 38

## 第六章 关系数据库的规范化理论 .....

6. 1	关系模式的规范化概述 .....	6 - 1
6. 2	关系数据库的设计理论 .....	6 - 5
6.2.1	函数依赖 (Functional Dependency) .....	6 - 6
6.2.2	计算闭包 .....	6 - 10

6.2.3	依赖集的复盖 .....	6 - 12
6.3	关系模式的分解 .....	6 - 13
6.3.1	分解的定义 .....	6 - 14
6.3.2	联接的不丢失性( <i>Lossless join</i> ) .....	6 - 14
6.3.3	联接不丢失性的检验 .....	6 - 14
6.3.4	分解对依赖的保持 .....	6 - 16
6.4	关系模式的规范化 .....	6 - 17
6.5	结果为 <i>BCNF</i> 的联接不丢失性分解 .....	6 - 19
6.6	多值依赖及第四范式 .....	6 - 22

## 第五章 DBTG 建议的网状模型的数据库系统

DBTG(Data Base Task Group—数据库任务组)是美国 CODASYL(Conference on Data System Language—数据系统语言协商会)下属的一个组织。由 CODASYL 于 1971 年四月通过的 DBTG 报告中所确定的方法,作为网状系统的代表。因为网状数据库比关系类型数据库查询路径快,效率高,故大多数公共机器均采用网状的。

### 5.1 DBTG 系统的结构

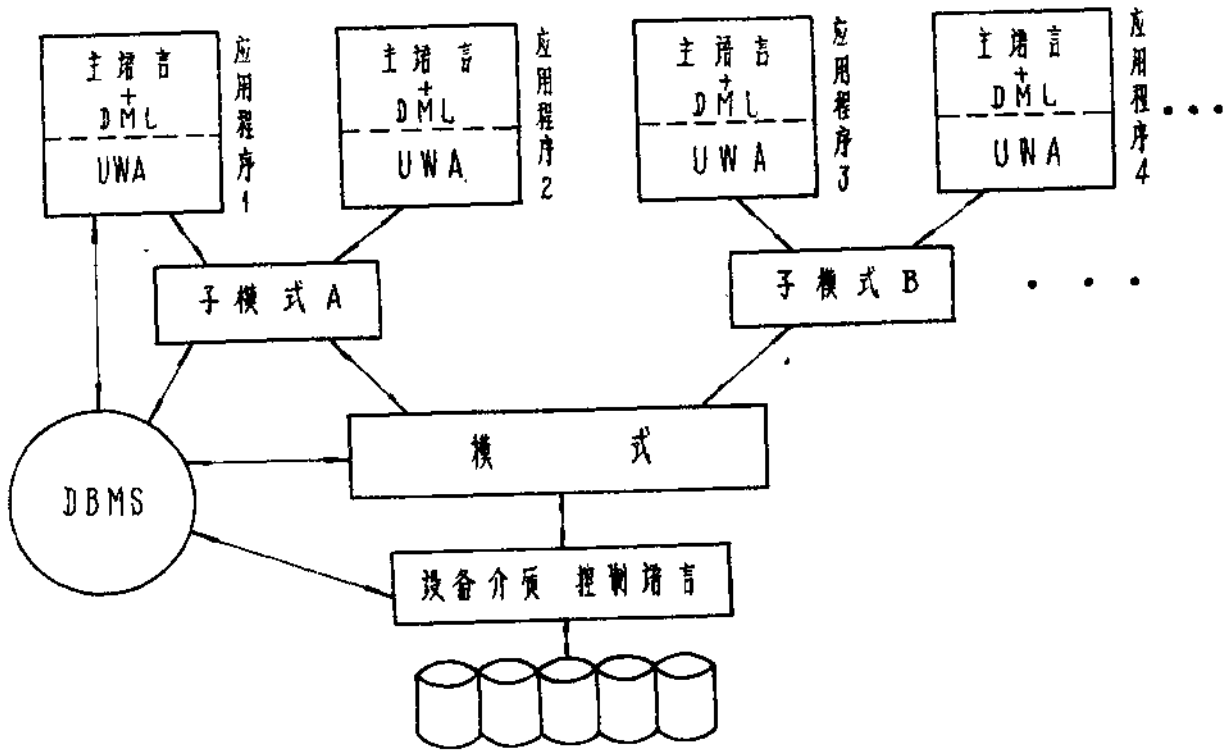


图 5.1 DBTG 的系统逻辑结构

DBTG 系统的结构如图 5.1 所示,包括如下五个主要部分:

(1) 模式数据描述语言 (Schema DDL)。它主要是描述数据模型的结构,但也涉及非结构方面的一些内容。DBTG 报告把模式定义为一个数据

库的描述。一个数据库只有一个模式，但一个系统中可以有許多数据库，因而一个系统也可以有多个模式。

(2) 子模式数据描述语言 (*Sub-Schema DDL*)。DBTG 定义子模式为模式的一部分，子模式是提供一种识别数据库某一特定部分的方法，是用户与数据库的接口。在系统中，一个子模式只能属于一个模式，而不能跨越两个或两个以上的模式，但一个模式中 can 包含多个子模式。一个应用程序必须而且只能使用一个子模式，才能处理数据库的数据。

(3) 数据操作语言 (*DML-Data Manipulation Language*)  
它是对数据库中的数据进行某些操作的语言，用户用主语言加 DML 编写应用程序。

(4) 设备介质控制语言 (*DMCL-Device Media Control Language*)

它用来描述一个具体的系统所使用的设备，常把 DMCL 的语句交织在模式 DDL 中。

(5) 数据库管理系统 (*DBMS-Data Base Management System*)

它将源模式变成目标模式，子模式的源形式变成目标形式及对 DML 处理。每执行一次 DML 语句，进程工作区和系统缓冲区之间交换一个记录。而状态码空间用于存放每个 DML 语句执行的有关信息和若干当前指针 (每个域、络类型、记录类型位置)。

## 5.2 DBTG 的数据模型

DBTG 是以记录类型为结点的网络 (*Network*) 数据模型。网络也称丛 (*plex*)，分解成若干个二级树来表示它。网络与树不同处：(1) 有的结点可能有多个父结点；(2) 两结点之间可能有多种有向连接；(3) 可能有回路存在。

### 5.2.1 记录类型

记录类型是指为具有同样结构的一类记录的描述。它是记录的框架，是数据项的分级结构 (图 5.2)。

数据项又分为初等项和组项。初等项是可命名的最小数据单位，组项是由两个或两个以上相邻接的数据项组成的一个命名数据单位，组项可由

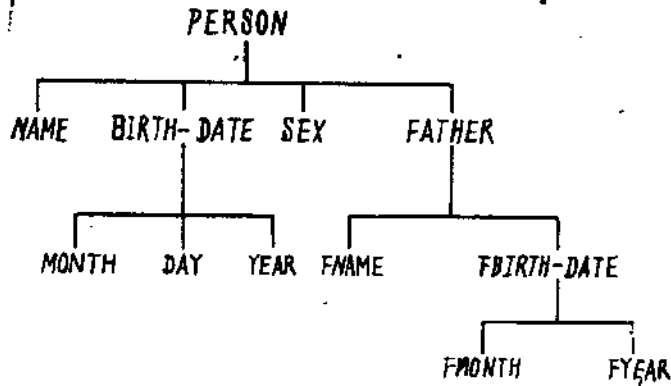


图5. 2

初等项或其他的组项组成。

图 5. 2表示一个 *PERSON* 记录类型，用 *DBTG* 数据描述语言可将它描述如下：

```

RECORD NAME IS PERSON ;
  02 NAME, TYPE IS CHARACTER 12.
  02 BIRTH-DATE ;
    03 MONTH ; PICTURE IS 99.
    03 DAY ; PICTURE IS 99.
    03 YEAR ; PICTURE IS 9(4).
  02 SEX ; PICTURE IS A.
  02 FATHER ;
    03 FNAME ; PICTURE IS A(12).
    03 FBIRTH-DATE ;
      04 FMONTH ; PICTURE IS 99.
      04 FYEAR ; PICTURE IS 9999.
  
```

### 5. 2. 2 络类型 (Set type)

在数据库中不仅要考虑记录，还要考虑记录之间的联系。记录之间的联系在 *DBTG* 中叫做络。一种联系为一个络，另一种联系为另一个络。此种联系中，有一个处于主导地位的记录类型称为主记录 (*owner*) 类型；处于从属地位的记录类型则称为从记录 (*member*) 类型。对络类型必须命名。



络类型本质上是一棵二级树，主记录类型是树根，从记录类型是树叶。在 DBTG 数据模型中会遇到下面三种络类型。

(1) 单从络类型。只包含一个从记录类型，如图 5.3 所示。

(2) 多从络类型。该络类型中含有二个或二个以上从记录类型 (图 5.4)。

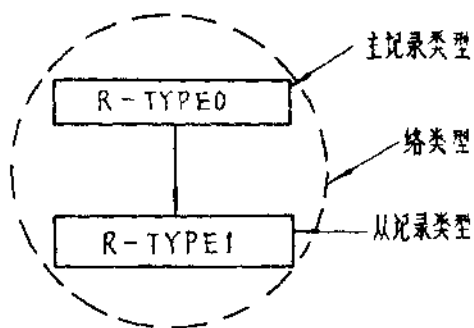


图 5.3 单从络类型

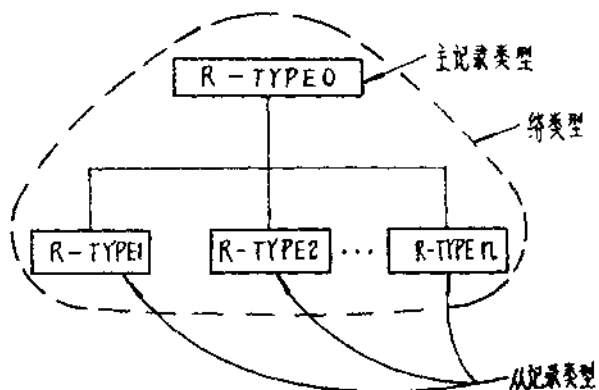


图 5.4 多从络类型

(3) 奇异络类型或系统络类型。这种络类型没有主记录类型，因而谓之奇异的 (Singular); 或者说它的主记录类型就是系统本身，因而又称为系统络类型 (System Owned Set type)，见图 5.5。

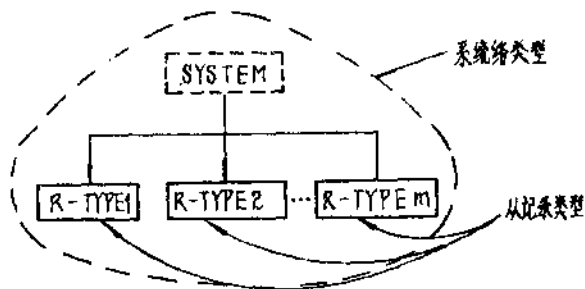


图 5.5 系统络类型

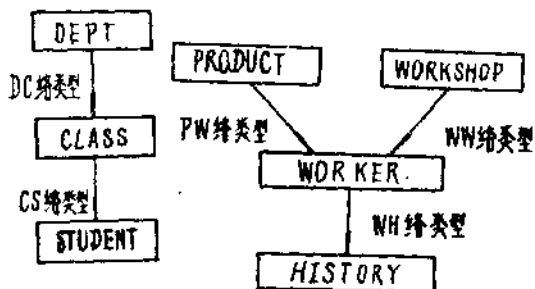


图 5.6 层次结构

图 5.7 Y-结构

络类型是记录类型之间联系的一种表示，它是构造网络模型的建筑块。而 DBTG 数据模型应遵循如下规则：

(1) 一个记录类型可以是一个络类型的从记录类型，同时又是另一个络类型的主记录类型如图 5. 6 的记录类型 *CLASS*。

(2) 一个记录类型可以同时是若干个络类型的从记录类型，如图 5. 7 的记录类型 *WORKER*。

(3) 在任何两个记录之间可以定义有限个络类型，见图 5. 8。

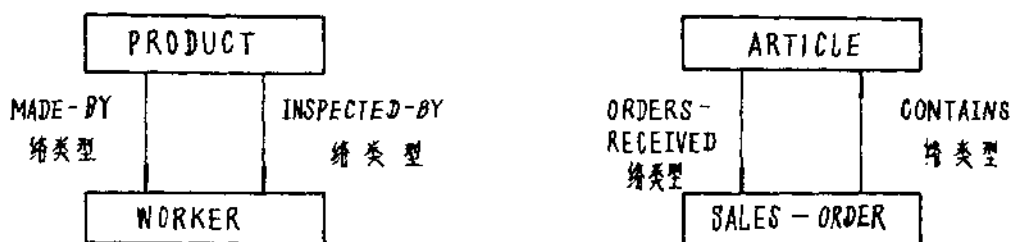


图5.8 两个记录类型之间的多个络类型

(4) 允许回路存在(图 5. 9)，但不允许自回路存在，即不允许在同一络类型中一个记录类型即是它的主记录类型，又是它的从记录类型。见图 5. 9。

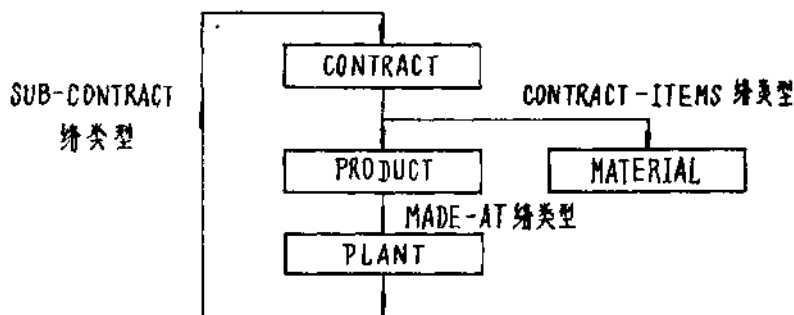


图5.9 包含回路和多从络类型的结构

(5) 一个网络模型可以包含有限个记录类型和络类型。表示上述规则的图 5. 3~图5. 9，称为 *Bachman* 图，作为说明数据库模型的一种图介技术。

### 5. 2. 3 络事件 (*Set occurrence*)

络类型的每个实例称为络事件，简称络。它是以记录为结点的一棵二级树，其根是所在络类型中主记录类型的某个记录，称为主记录；其叶是所在络类型中从记录类型的记录，称为从记录。一个络类型的主记录类型包含多少个记录，它的络类型也就包含多少个络。在系统络类型中，因为把主记录类型理解成系统，而系统是唯一的，所以系统络类型只含有唯一的一个络。

一个络可以非空也可以空。空的涵义是说络中有主记录而无从记录（如图 5.10(-)中络 PW3）。非空的络总有一个或多个从记录与主记录相联系（图 5.10中 (-) 的络 PW1 和 WG1）。

必须强调的是：一个记录仅可连到一个给定络类型的一个络上，而不能同时连接到它的几个络上；但是，一个记录可以连接到不同络类型的几个络上，只要这些络不处于同一络类型中。

在图 5.10中 (-) 用 Bachman 图表示络。PW 络类型有三个络：PW1、PW2 和 PW3。PW1 的主记录为 P1，从记录为 W1、W5；PW2 的主记录为 P2，从记录为 W4；PW3 是空的，主记录为 P3，无从记录。WG 络类型有三个络：WG1、WG5、WG4。WG1 的主记录为 W1，从记录为 G5、G7；WG5 的主记录为 W5，从记录为 G6；WG4 的主记录为 W4，从记录为 G2、G3、G8、G10。

在图 5.10中 (二) EMPLOYEE 是两个络类型的主记录类型。ES 络类型有两个络：ES1、ES2。EC 络类型也有两个络 EC1、EC2。

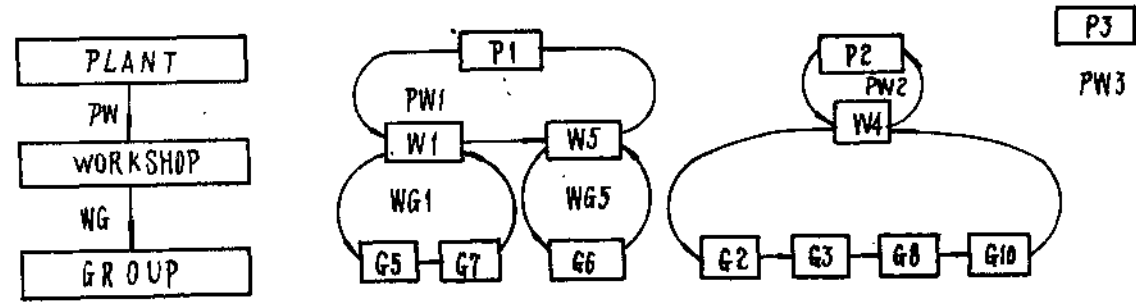


图 5.10 络事件图解 (一)

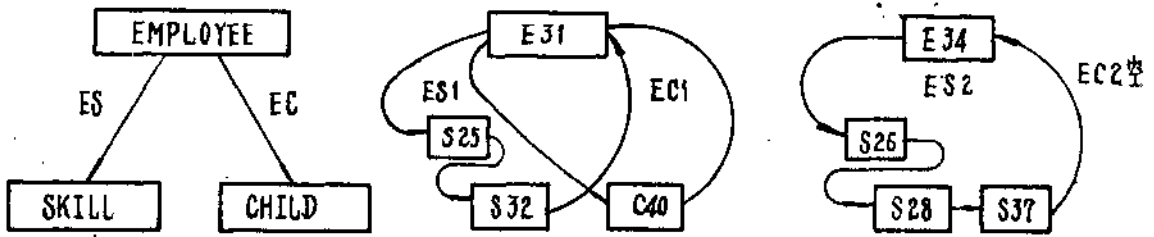


图 5.10 络事件图解 (二)

#### 5.2.4 事物联系的 DBTG 表示法

DBTG 方法是用记录类型、络类型构造数据模型，用以表示现实世界事物的各种联系。我们知道，每一类事物都可用一个记录类型表示，问题是如何表示记录类型之间的联系。

(1) 一对一联系。有两种方法表示  $A$ 、 $B$  两个记录类型之间的一对一联系，其一是把两者合并为一个记录类型；其二是两者之间定义一个络类型 (图 5.11)。

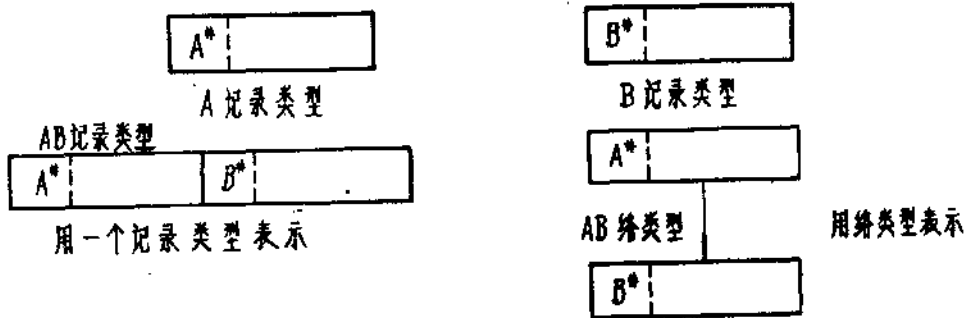


图 5.11 表示一对一联系的方法

(2) 一对多联系。当  $A$ 、 $B$  两记录类型是一对多联系时，定义一个络类型 ( $A$  为主记录类型， $B$  为从记录类型) 表示这种联系。

(3) 多对多联系。当两记录类型之间是多对多联系时，创建一个新记录类型，其关键字由原来两个记录类型的关键字连接而成，并分别在新记录类型 (作从记录类型) 与两个老记录类型 (作主记录类型) 之间各定义一个

络类型。例如图 5-12 中，记录类型 *COURSE* 和 *STUDENT* 之间是多对多的联系，这时建立一个新记录类型 *CS*，它由 *COURSE* 的关键字 *C\** 和 *STUDENT* 的关键字 *S\** 以及他们的相交数据 (*GRADE*) 组成。同时建立两个络类型 *C-CS* 和 *S-CS* (主记录类型分别为 *COURSE* 与 *STUDENT*，从记录类型为 *CS*)。

络类型 *C-CS* 包含络 *C-CS1*、*C-CS2*、*C-CS3*、*C-CS4* 络类型 *S-CS* 包含络 *S-CS1*、*S-CS2*、*S-CS3*、*S-CS4*、*S-CS5*。其中络 *C-CS3* 和 *S-CS5* 为空络。

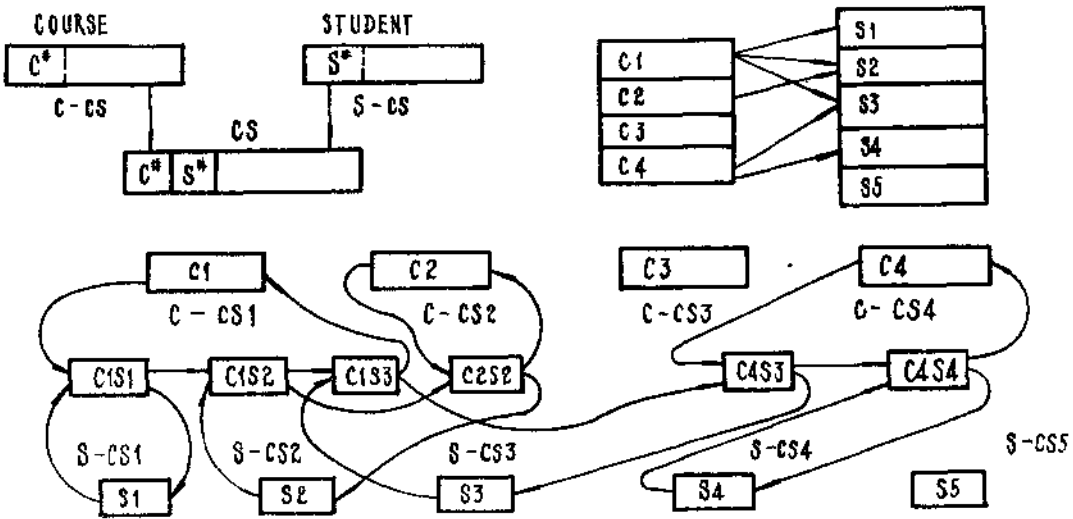


图 5-12 表示多对多联系的方法

(4) 冗余法。DBTG 的络规则指出，一个记录只能属于一个络类型中的某一个络。但实际情况并非都能满足这一规则，如不同教员讲授的课程大多不同，但两个以上教员讲授同一课程的也有。这样，如果在教员和课程之间定义一个络类型，就会产生某一课程记录（如 *C13*）属于两个以上络的情况（图 5-13）。

对于这不符 DBTG 规则的络，可用冗余法改造它为符合 DBTG 的络。如图 5-13 中的记录 *C13* 增加一个副本，就得到符合 DBTG 规则的两个络（图 5-14）。

冗余法不一定都行得通。其原因：① 冗余很多时浪费很多存贮空间。② 两个记录类型可能有相交数据，如在 *COURSE* 和 *STUDENT* 间定义络类

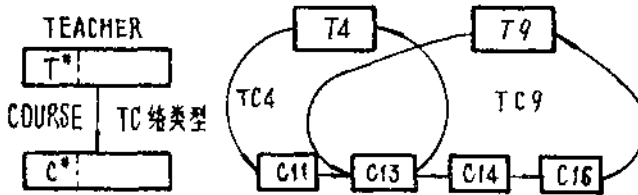


图 5-13 不符合 DBTG 规则的络

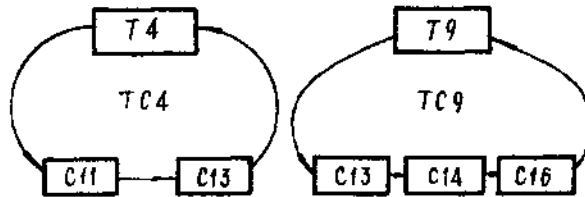


图 5-14 冗余法

型 (主记录类型为 COURSE), 用冗余法满足络规则, 这时容易查找学习某一课程的所有学生, 但难以查找某一学生学习的所有课程。见图 5-15。

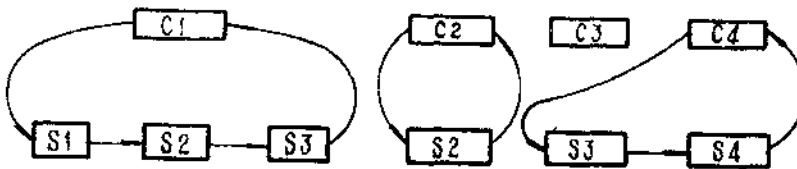


图 5-15 用冗余法表示多对多的联系

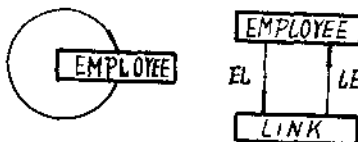


图 5-16 单个记录类型的环

### (5) 单个记录类型的自回路

以行政机构为例，在一个部中，从部长到科员都是职工，即从记录类型都是职工。从领导关系讲就比较复杂，部长领导局长，局长领导处长，处长领导科长，科长领导科员。从记录类型讲，则领导关系是环类型的(图 5-16)。

在DBTG中，不允许一个记录即是作为一个络的首记录，又作为一个络的从记录。因此，引进一个新的记录类型LINK作为EMPLOYEE的替身，并定义两个络类型EL(其主记录类型是EMPLOYEE)和LE(其主记录类型是LINK)。LINK记录类型可以只包含一个数据项，如EMPLOYEE的关键字。在EL络类型中，每个络只有一个主记录和一个从记录；而在LE络类型中，每个络有一个主记录和若干个(也可没有)从记录EMPLOYEE。部门中人员管理关系正是通过两个络类型EL和LE表示的(如图 5-17)。

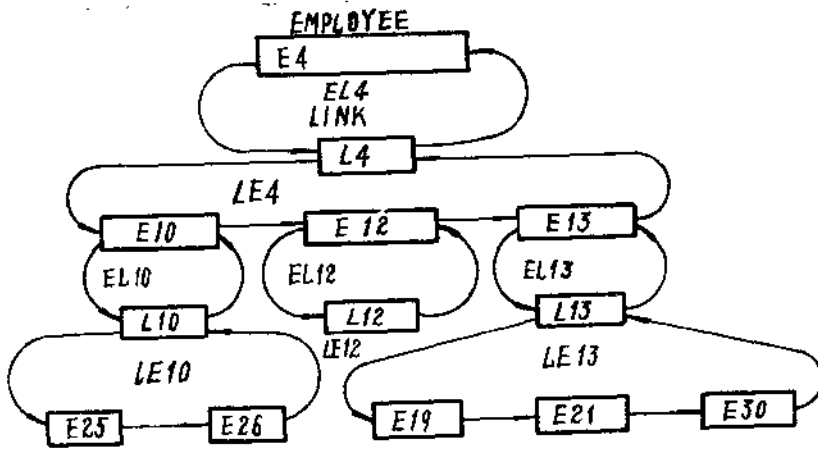


图 5-17 EMPLOYEE 自回路路表示

### 5.3 记录类型描述及其存储映射

DBTG 描述语言比较复杂，首先讨论各部分，然后再给出完整句法。

#### 5.3.1 DBTG 句法使用的符号

DBTG 在数据描述语言的表示方法中，使用了如下符号：

$\left[ \begin{matrix} A \\ B \\ C \end{matrix} \right]$ 
 $\left\{ \begin{matrix} A \\ B \\ C \end{matrix} \right\}$ 
 $\left\| \begin{matrix} A \\ B \\ C \end{matrix} \right\|$ 
 $(, A) \dots$

方括号 花括号 双竖线 省略号

(1) 方括号。称为选择符，表示只能使用其中一项（如 *A* 或 *B* 或 *C*）或一项不用。

(2) 花括号。称择一符，表示必须而且只能使用其中一项。

(3) 双竖线。称为多选符，表示至少使用其中一项，当然可以使用其中若干项。

(4) 省略号。称为重复符，表示圆括号中的项可重复使用，而圆括号可为上述符号之一。

在 *DBTG* 句法中，语言部分用大写字母书写，而小写字母表示由用户提供的名称。大写单词中其下划有横线者为必写词；未划者可省略。在 *DBTG* 句法中，常见有两个缩写字：

$$\left\{ \begin{array}{l} dbd-name = database \ data \ name \ (\text{数据库数据名}) \\ db-id = database \ identifier \ (\text{数据库标识符}) \end{array} \right\}$$

均指模式中已说明的某个初等项或组项，但 *dbd-name* 不一定有唯一性，因为不同记录类型中的数据项可能重名；而 *db-id* 必定是唯一的。把一个 *dbd-name* 变成 *db-id* 时，可能要写下标或加限定，说明它是那个记录类型的数据项。于是两者的关系：

$$db-id = dbd-name \left( (integer-1 [, integer-2] \dots) \right) \\ \left[ \underline{QN} \ record-name \right]$$

### 5.3.2 记录类型的描述

记录类型是数据项的一个分级结构，也是数据库与主语言的主要联系点。

(1) 初等项。记录类型的每个初等项后面置有 *TYPE* 条款说明其数据特征，即：

$$\begin{array}{l} \text{TYPE IS } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \underline{BINARY} \\ \underline{DECIMAL} \end{array} \right\} \\ \left\{ \begin{array}{l} \underline{FIXED} \\ \underline{FLOAT} \end{array} \right\} \\ \left\{ \begin{array}{l} \underline{REAL} \\ \underline{COMPLEX} \end{array} \right\} \\ \left\{ \begin{array}{l} \underline{BIT} \\ \underline{CHARACTER} \end{array} \right\} \\ \underline{DATA-BASE-KEY} \end{array} \right\} \\ \left. \begin{array}{l} \left[ integer-1 [ integer-2 ] \right] \\ [ integer-3 ] \\ implementor-name \end{array} \right\} \end{array}$$



其中各个词的含义：*BINARY*—二进的，*DECIMAL*—十进制的，*FIXED*—定点，*FLOAT*—浮点，*REAL*—实的，*COMPLEX*—复的，*BIT*—位串，*CHARACTER*—字符串，其后的整数表示该项的长度。实数有如下几种组合：*REAL BINARY FIXED REAL BINARY FLOAT REAL DECIMAL FIXED REAL DECIMAL FLOAT*；复数亦有类似组合。*DATA-BASE-KEY*说明该数据项作为该记录的数据库关键字*DBK*。*implementor-name*是留给系统设计者规定的一种类型。*integer-1*是有效位数，*integer-2*是因子。

此处*TYPE*条款是纯*PL/1*的，它反映*PL/1*处理复数、位串和各类实数的能力。也可以用*PICTURE*条款来代替它，*PICTURE*的作用主要是说明非数值计算的数据项，如日期、号码、地址等。例如：

```
BIRTH-DATE PICTURE IS 99X 99X 9(4)
NAME PICTURE IS A(15)
SALARY PICTURE IS 9(4) V99
```

其中，符号9表示0到9中任何十进制数字，X表示符号集中任一字符，A表示任一字母或空白，括号中的数字表示前一符号的重复次数，V表示小数点。如9(4)V99表示六位十进制数字，小数点前面四位，后面两位。

(2) 组项。几个初等项或组项组成组项。因为数据项在一个记录类型中有处在第几级的问题，故在描述记录时，每一个数据项要给一个级数（可取1到99的数），同级数据项的级数应相同，低级数据项的级数应大于高级数据项。如记录的级数取01，下面数据项的级数可取02，再下级可取03，依此类推。句法形式为：

```
[Level-number] dbd-name-1 [ ; TYPE IS --- ] [ ; PICTURE IS --- ]
```

(3) 重复组。如果一个记录事件的某一数据项取一组值，则称该数据项有重复组。又分为：①固定重复组，即所有记录的某数据项重复取的次数相同。②可变重复组，即数据项在各记录中重复取值的次数不同。一个记录类型最多允许三个可变重复组，并要放在记录类型最后。指出重复取值次数的条款为：

```
OCURS { integer-4 } TIMES
      db-id-1
```

其中*integer-4*表示大于0的整数，说明固定重复次数，*db-id-1*