目　　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第四章　层次方法

第一章曾就数据的层次模型作了一般的介绍。通常把用层次模型设计数据库的方法称为层次方法。本章将着重讨论这一方法。首先介绍一般概念，而后介绍一个有代表性的层次系统——*IMS*系统。

4.1　一般概念

4.1.1　树

层次模型是一种树结构。在此，仅把树的若干最重要的特征予以强调，记住这些特征，有助于掌握数据的层次模型。

（1）所有结点之间的连接都是从父结点到子结点。

（2）两个结点之间最多有一种连接。

（3）没有回路（*loop*）存在，即没有从自身出发又回到自身的连接。

（4）只有一个结点没有父结点，该结点称为树的根。

（5）除根外，任何结点都有而且只有唯一的父结点。

我们只讨论有序树。所谓有序树是对树中任一结点的所有子树都规定了先后次序的树。对层次模型来说，都按从左到右的顺序规定任一结点所有子树的先后次序。

树的一种情况是二叉树，它的每个结点最多有两个孩子，而一般树不限制结点的个数。如图4.1所示。

对有序树中每个结点访问一遍，称为树的遍历（*tree traversal*）。在树遍历中，有一个对结点扫描的先后次序问题，即树的遍历规则。在二叉树中，有前序、中序、后序三种遍历规则，若把二叉树的前序遍历规则推广到一般树，则可得如下推广前序遍历规则：

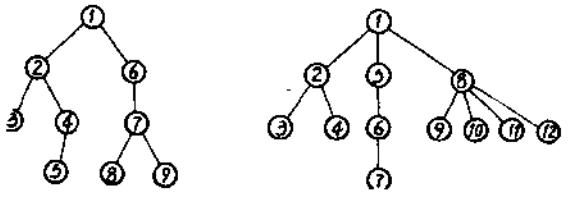


图4.1　二叉树与一般树

（1）首先访问树的根。

（2）从左到右遍历根的每棵子的。这也是一个递归的定义。

在层次模型中，把推广前序遍历规则确定的访问顺序称为层次顺序。层次顺序可用指针表示，这种指针为层次指针。利用层次指针可以把任何树变成链，图4.2（a）是图4.1右边的树变成的一条链。还有所谓子／孪指针。即每个结点的子指针指向它的最左子结点，而孪指针则指向它右边的第一个兄弟。利用子／孪指针可以把任何树变成二叉树（图4.2（b）中的二叉树就是图4.1中右边的树变来的）。

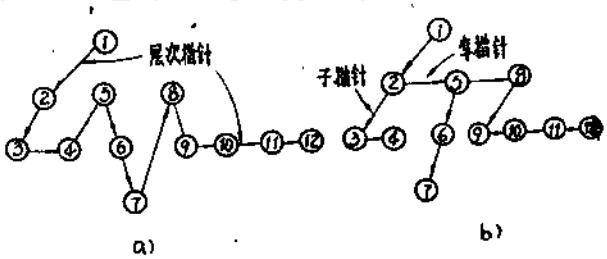


图4.2　层次指针与子／孪指针

为了扩充层次模型的应用范围，往往要把某些非树结构变成等效的树或森林。这里给出两种变换办法：

（1）冗余结点法。对一非树结构，先取一个无父结点的结点作为第一棵树的根，该结点的孩子作为该树根的孩子，按此方法又可以找到孩子的孩子作为树根的孙子。如果遇到该树中某处已出现过的结点，则填上它和它的子孙的付本，如此进行直至再没有孩子为止，第一棵树造完。如果原结构中还有在第一棵树中没有出现过的结点，则选取其中一个无父结点的结点作为第二棵树的根，再重复上述造树过程，直至原结构中所有结点都出现在已造的树中为止（图4.3）。

（2）虚拟结点法。虚拟结点法的造树过程基本上与冗余结点法相同，所不同的是：在遇到树中已出现过的结点时，不代之以该结点的付本和它的子孙的付本，而用该结点的一个虚拟结点放入树中。虚拟结点是一个指针，指向所代替的结点。例如用$V \cdot N$表示结点N的虚拟结点。

虚拟结点法的优点是：在进行物理组织时减少存储空间的浪费，避免产生潜在的不一致性；缺点是结点存储位置的移动可能引起虚拟结点中指针的修改。冗余结点法的优点是：结构清晰，允许结点存储位置的变动；缺点是需多占用存储空间，有潜在的不一致性。

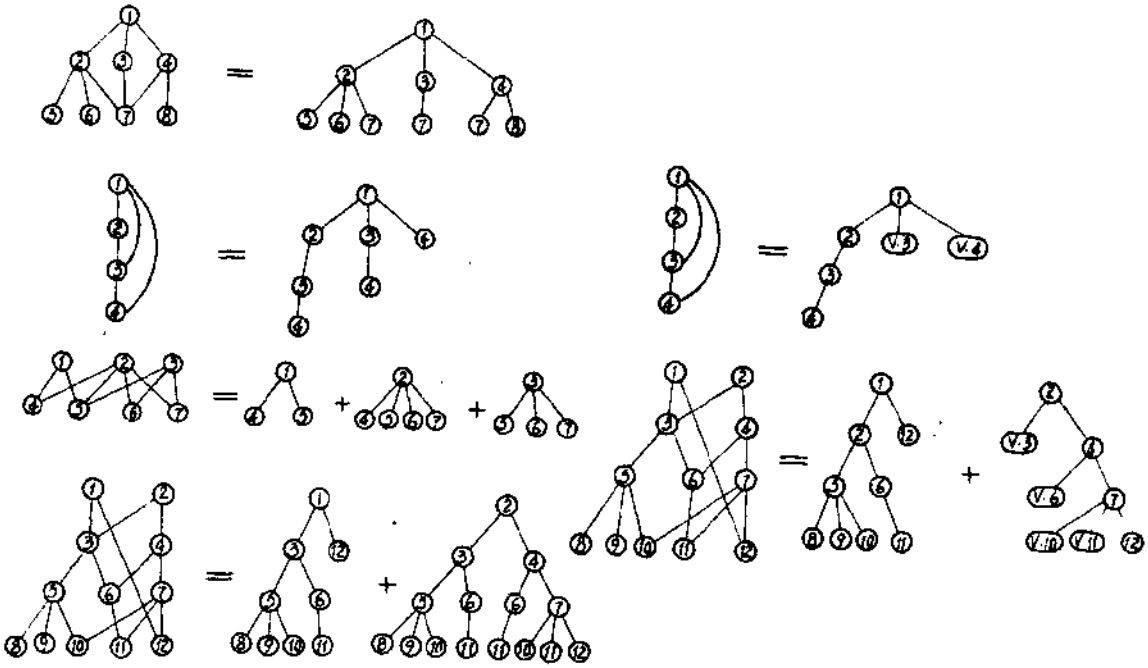


图 4—3　冗余结点法　　图 4—4　虚拟结点法

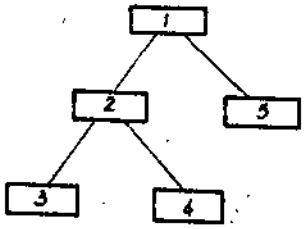
4.1.2　层次系统的数据模型

层次系统的数据模型是层次模型，它是以记录类型为结点的有序树或森林。每个结点都表示一个记录类型：

在层次模型中，每个记录类型可包含若干数据项，每一记录类型和它的数据项都必须命名，不同记录类型的名称应当不同，同一记录类型中不同数据项的名称也应不同。在两个记录类型之间最多只有父与子的联系。一个层次模型在理论上可以包含任意有限个记录类型和数据项。但任何实际的系统都会因为存储容量或其他原因而限制层次模型中包含的记录类型和数据项的个数。

图 4.6 给出具有五个记录类型的某校教学数据库的一个层次模型。现以该模型为例说明有关概念。

记录类型 *DEPT*（系）是树的根，它提供如下细节：*D#*（系号）、*TITLE*（系名）、*OTHER*（其他），并通过它的两个孩子提供 *COURSE*（课程）和 *TEACHERB*（教员）细节。



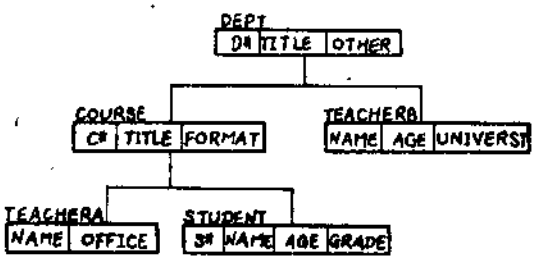


图 4.5 层次模型图示法　　图 4.6 教学数据库层次模型

记录类型 *COURSE* 提供如下细节：*C**（课程号）、*TITLE*（课程名）、*FORMAT*（规格），并通过它的两个孩子提供 *TEACHERA*（开课教员）和 *STUDENT*（学生）细节。

TEACHERA、*STUDENT*、*TEACHERB* 三个记录类型均无孩子，其细节分别在各自的数据项中。

若按推广前序遍历规则依次对模型中的各记录类型编码，则称之为类型码，可作为系统识别记录类型的一种内码。对图 4.6 中的层次模型可建立一张如同表 4.1 一样的登记表，该表可作为层次模型目标模式的表格形式。

层 次 模 型 登 记 表　　表 4.1

类型码	记录类型名	所包含的数据项名称	它的父类型	它的子类型
1	*DEPT*	*D**, *TITLE*, *OTHER*	无	*COURSE*, *TEACHERB*
2	*COURSE*	*C**, *TITLE*, *FORMAT*	*DEPT*	*TEACHERA*, *STUDENT*
3	*TEACHERA*	*NAME*, *OFFICE*	*COURSE*	无
4	*STUDENT*	*S**, *NAME*, *AGE*, *GRADE*	*COURSE*	无
5	*TEACHERB*	*NAME*, *AGE*, *UNIVERST*	*DEPT*	无

在层次模型中，每个记录类型均可包括任意有限个记录。就任何记录类型中的一个给定记录而言，它可以包含它的任何子记录类型中的任意有限个（可能一个也没有）记录作为自己的孩子。现将教学数据库层次模型中各记录类型所包含的记录、以及记录之间的父子联系用图 4.7 表示（为了图示清楚，模型中记录的个数取得较少）。其中根类型有三个记录，它们各自与自己的子孙记录类型中的有关记录一起构成三棵树。每一棵这样的树称为层次

模型的一个值（或实例）。一个层次模型的所有当前值，包含了该模型的所有当前数据。

图 4.6 和图 4.7 将作为以后 *IMS* 系统使用的样本模型和样本数据，读者者必须熟悉其结构。当然这个模型并不太好。因为不同课程的学生太多数是相同的，同一学生的姓名（*NAME*）和年龄（*AGE*）多次重复出现，造成数据冗余和潜在的不一致性。保留这一缺点不影响后面的叙述，反而能使读 者看到问题所在。

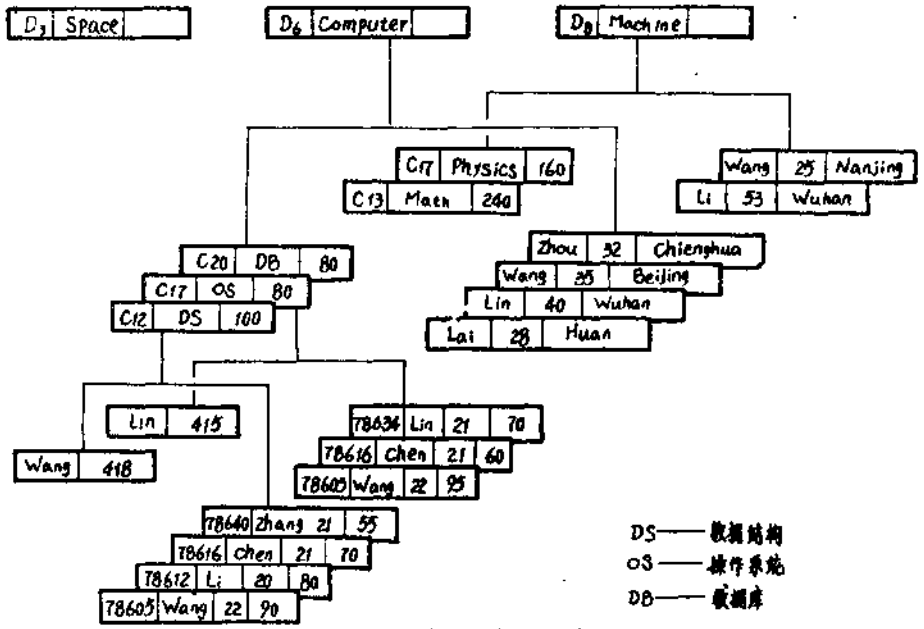


图 4.7　教学数据库层次模型的值

4.1.3 层次顺序与层次路径

在层次模型中，为了给一个记录类型中的记录排列顺序，必须指定一个数据项作它的关键字。对图 4.6 的模型指定各记录类型的关键字如下.

类型码	记　录　类　型	关　键　字
1	*DEPT*	$D^{\bullet}$
2	*COURSE*	$C^{\bullet}$
3	*TEACHERA*	*NAME*
4	*STUDENT*	$S^{\bullet}$
5	*SEACHERB*	*NAME*

可使用层次顺序来安排一个记录在整个模型中的顺序：一个记录的层次顺序是这样确定的：

（1）对每个记录而言，用类型码作前缀，加上它的关键字值作为该记录的顺序值。

（2）对根记录而言，层次顺序值就是它的顺序值。

（3）对非根记录而言，其层次顺序值由父记录的层次顺序值作前缀，再加上自身的顺序值组成。

（4）把各记录的层次顺序值左边对齐比较大小，其值上升的顺序就是层次模型中所有记录的层次顺序。这与推广前序遍历规则确定的层次顺序一致。

例如：记录*D6*，*Computer*，…）的层次顺序值为1*D6*，记录（*C17*，*OS*，80）的层次顺序值为1*D62C17*，记录（78640，*Zhang*，21，55）的层次顺序值为1*D62C12478640*。由于1*D62C12478640*＜1*D62C17*，所以按层次顺序，记录（78640，*Zhang*，21，55）应排在记录（*C17*，*OS*，80）的前面。

按层次顺序存储和查找记录是层次系统的访问方法之一。但这种办法费时，查找效果与顺序扫描一样。为了加快查找速度，必须指出从根记录开始到目标记录的一条查找路径（要相应存储有关指针），即从根开始经过目标记录所有直系祖先的路径，称之为层次路径。如目标记录是（78612，*Li*，20，80），那么到达它的层次路径是（*D6*，*Computer*，…）→（*C12*，*DS*，100）→（78612，*Li*，20，80）。查找时可用*DL/1*子语言写出如下的查找操作：

```
GU  DEPT      (D# ='D6')
    COURSE    (C# ='C12')
    STUDENT   (S# ='78612')
```

在层次路径上，从根记录到目标记录的各个关键字值依次联成一串，称为目标记录的“全联关键字”（*fully concatenated key*）。一个记录的全联关键字等于它的层次顺序值去掉类型码。如记录（78612，*Li*，20，80）的层次顺序值是1*D62C12478612*，而它的全联关键字是*D6C1278612*。

此为试读，需要完整PDF请访问：www.ertongbook.com

下面用图4.8画出图4.7中所有可能的层次路径。其中圆圈表示一个记录。圆圈内的字符表示记录的关键字。

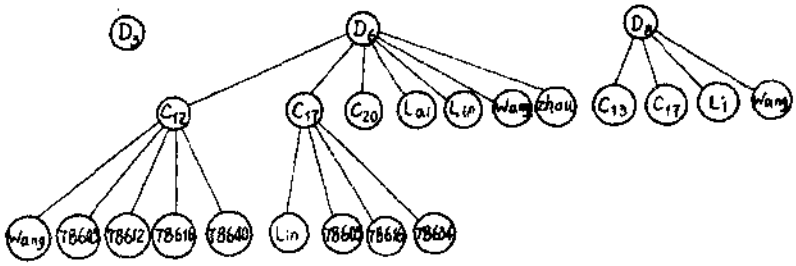


图4.8　层次路径

4.1.4　层次系统的模式与子模式

层次数据库系统模式的主要内容是数据库中所含数据的层次模型。如图4.6就是用图示法表示的一个模式。模式中根记录类型排在层次结构的第一级（如*DEPT*），它的孩子排在第二级（如*COURSE*，*TEACHERB*），孩子的孩子排在第三级（如*TEACHERA*，*STUDENT*），……。直到再没有孩子为止。任何一个记录类型（或记录）的子孙，都称为该记录类型（或记录）的从属。除根记录类型（记录）外，不允许任何一个无父记录类型（记录）的记录类型（记录）存在。即删去一个记录类型（记录）也就删去了它的所有从属。

层次数据库系统的子模式，就是从模式的层次结构中删去（或不删去）某些非根记录类型而剩下的层次结构。图4.9中上面两个是图4.6的子模式，下面两个则不是。

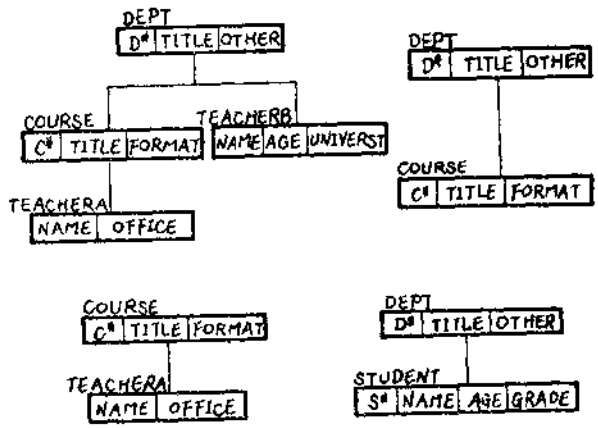


图4.9　层次系统的子模式与非子模式

4.2 *IMS* 系统的逻辑结构

目前流行的大型数据库系统中有许多是基于层次的。其中最著名的是 *IBM* 公司研制的 *IMS*(*Information Management System*)。*IMS* 系统已经在世界范围内广为使用（约1000个以上地区），成为一个知名的数据库系统。所以许多文献都把它作为层次数据库的典型代表：

它有自己独特的术语，为避免概念混淆，将其术语与一般术语列表对照于下：

术 语 对 照 表 表 4.2

一般术语	*IMS* 术 语
记录类型(*record type*)	段类型(*segment type*)
记录(*record*)	(段)事件(*occurrence*)
数据项(*data item*)	域(*field*)
关键字(*key*)	顺序域(*sequence field*)
模式(*schema*)	物理数据库记录类型(*PDBRT*)
子模式(*subschema*)	逻辑数据库记录类型(*LDBRT*)

系统常用缩写字的全名为：*PDBRT*——物理数据库记录类型(*Physical Database Record Type*)；*LDBRT*——逻辑数据库记录类型(*Logical Database Record Type*)；*DBD*——数据库描述(*Database Description*)；*PCB*——程序通讯块(*Program Communication Block*)；*PSB*——程序说明块(*Program Specification Block*)。

4.2.1. *IMS* 的逻辑结构

IMS 系统是一个三级结构数据库系统。它的模式是物理数据库记录类型（这是 *IMS* 术语，没有物理存储的含义）的集合。每个 *PDBRT* 是图4.6形式的一棵树，并用一个 *DBD* 定义，模式到存储的映射也写入 *DBD* 中。一个 *PDBRT* 所有事件的有序集合构成一个物理数据库，即实际存储的数据库。一个企业的数据可能存放在几个物理数据库中。*IMS* 系统的所有物理数据

库都存放在外存设备上，构成存储数据库。

用户按照外模型（即子模式）操作数据。一个用户的外模型是逻辑数据库记录类型的集合，每一个 *LDBRT* 用一个 *PCB* 定义，并且 *LDBRT* 到 *PDBRT* 的映射（即子模式到模式的映射）也写入 *PCB* 中。一个用户的所有 *PCB* 组成的集合构成 *PSB*，它定义该用户的外模型。

用户用主语言（*PL/1*，*COBOL* 或 *Assember Language*）写应用程序，并在需要访问数据库的地方插入 *DL/1* 操作，该操作通过调用系统的一个唯一的过程实现。一个 *DL/1* 操作所能传送的最小数据单位称为访问单位，它是一个段事件。用户应在应用程序中给每个段类型准备一个 *I/O* 区。作为他的程序和数据库之间传输数据的工作区间。*I/O* 区的地址作为调用数据操作过程的参数。

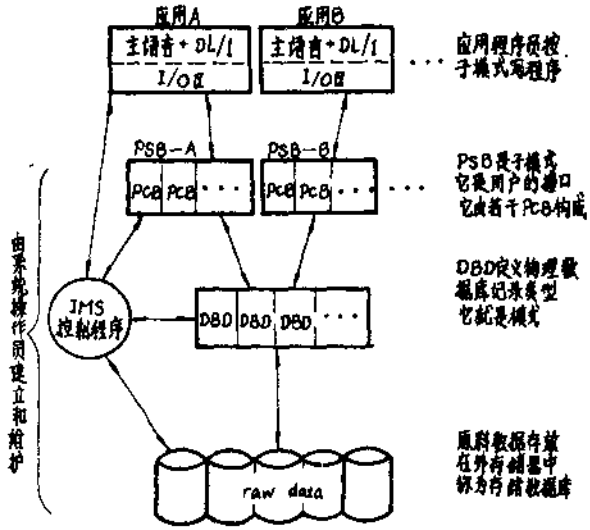


图 4—10　*IMS* 系统的结构

4.2.2　*IMS* 的 *DBD*

DBD 描述一个 *PDBRT* 和它到存储的映射。它由 *DBA* 建立和维护。在源形式下，它是 *IBM360* 系统汇编语言宏语句的集合。例如，图 4.6 的教学数据库用 *DBD* 描述的形式如下：

```
 1 DBD      NAME=EDUCPDBD, ACCESS=HSAM
 2 DATASET  DD1=EDUCHSAM, DEVICE=3330, …
 3 SEGM     NAME=DEPT, BYTES=52, …
 4 FIELD    NAME=(D*, SEQ, U), BYTES=2, START=1, …
 5 FIELD    NAME=TITLE, BYTES=20, START=3, …
 6 FIELD    NAME=OTHER, BYTES=30, START=23, …
 7 SEGM     NAME=COURSE, PARENT=DEPT, BYTES=31, …
 8 FIELD    NAME=(C*, SEQ, U), BYTES=3, START=1, …
 9 FIELD    NAME=TITLE, BYTES=25, START=4, …
10 FIELD    NAME=FORMAT, BYTES=3, START=29, …
11 SEGM     NAME=TEACHERA, PARENT=COURSE, BYTES=15, …
12 FIELD    NAME=(NAME, SEQ, U), BYTES=12, START=1, …
13 FIELD    NAME=OFFICE, BYTES=3, START=13, …
14 SEGM     NAME=STUDENT, PARENT=COURSE, BYTES=22, …
15 FIELD    NAME=(S*, SEQ, U), BYTES=5, START=1, …
16 FIELD    NAME=NAME, BYTES=12, START=6, …
17 FIELD    NAME=AGE, BYTES=2, START=18, …
18 FIELD    NAME=GRADE, BYTES=3, START=20, …
19 SEGM     NAME=TEACHERB, PARENT=DEPT, BYTES=44, …
20 FIELD    NAME=(NAME, SEQ, M), BYTES=12, START=13, …
21 FIELD    NAME=AGE, BYTES=2, START=13, …
22 FIELD    NAME=UNIVERST, BYTES=30, START=15, …
23 DBDGEN
24 FINISH
25 END
```

以上教学数据库的DBD描述包含五种语句，语句左边的数为语句标号，各语句的含义如下所述：

（1）DBD——这是数据库描述开始的第一个宏语句，有如下参数：NAME项给出该DBD的名称（在IMS中所有命名最长不能超过8个字符）；ACCESS项给出对物理数据库的访问方法。是 4.3 中将要讨论的四种存储结构（HSAM、HISAM、HDAM 和HIDAM）之一。由于有索引和逻辑数据库（见 4.5），所以ACCESS还有 INDEX 和 LOGICAL 两种

方式。

（2）*DATASET*——该语句指出物理文件的细节。本例指出物理文件名为*EDUCHSAM*，它存储在3330型磁盘上。其他项目被略去。

（3）*SEGM*——该语句定义一个段类型，其参数*NAME*给出段名；*PARENT*指出该段类型的父段类型（仅在根段类型才能省略它）；*BYTES*指出该段所有域的字节数，这里也省略了一些参数。

另外，*IMS*系统规定，在一个*PDBRT*中包含的段类型最多不得超过255个，同时在任何一条层次路径上包含的段数最多不能超过15个，即树的深度不能大于15。

（4）*FIELD*——该语句定义一个域。它属于紧靠它前面的*SEGM*语句所指其段类型。其参数*NAME*指出该域的名称；*BYTES*指出该域的字节数，*START*指出该域从该段第几个字节开始。*SEGM*语句后的第一个*FIELD*语句说明该域是该段的顺序域。例如标号为8的*FIELD*语句中有：*NAME*=(*C**, *SEQ*, *U*)，这里*C**是该域名称；*SEQ*是顺序域标记；*U*的含义是唯一，即对根段而言没有两个段事件在顺序域取相同值，对非根段而言，该段类型中对应于同一父段事件的所有事件在顺序域中取唯一值，也就是说同一父段事件的所有孩子不会有两个在顺序域中取相同值的，而不同父段事件的孩子在顺序域中是否取相同值，不影响这里给顺序域赋予*U*的含义。例如图4.7中，尽管*COURSE*段类型的事件(*C*17，*OS*，80)和(*C*17，*Physics*，160)在顺序域中有相同值*C*17，但前者的父事件是(*D*6，*Computer*，…)，后者的父事件是(*D*8，*Machine*，…)，由于两者的父事件不同，故仍可把*C**说明为*U*。若在*U*处置换为*M*，如标号为20的*FIELD*语句中*NAME*=(*NAME*，*SEQ*，*M*)的情况，这时*M*指出同一父段事件的孩子中可以有在顺序域中取相同值者。在本例中，它说明一个系的教员中可能有重名者。书写时*U*可以省略，但*M*不能省略。在采用*HISAM*和*HIDAM*存储结构时，要求根段类型的顺序域取值必须是唯一的，因为那时要对根段事件建立索引。

在*FIELD*语句中，还有说明数据类型的参数*TYPE*：*TYPE*=*X*，说明该域为十六进制数；*TYPE*=*P*，说明为组装十进数(*packed decimal*)，即一个字节放两个十进制数字；*TYPE*=*C*，说明为字母数字形式。

（5）*DBDGEN*、*FINISH*、*END*——按此顺序书写这三个语句，表示结束数据库*DBD*的定义。

在书写*DBD*时，应注意上述语句的顺序。定义段类型的*SEGM*语句必须按所定义的段类型类型码上升的顺序出现，第一个*SEGM*语句必须定义根段类型。由于*IMS*系统按照*SEGM*语句顺序和*PARENT*参数识别一个段类型在层次体系中的位置，故改变*SEGM*语句的顺序意味着模式结构的改变。

在数据库中装入多个*DBD*时，须按图4.11的形式写好，并将其送入系统。而后启动汇编器，把源形式的*DBD*翻译成目标形式，并存储到系统的库中。当*IMS*控制程序要使用模式时，就可以对它进行访问。

```
DBD ...
...
DBDGEN
FINISH
END
DBD ...
...
DBDGEN
FINISH
END
...
DBD ...
...
DBDGEN
FINISH
END
```

第一个*DBD*

第二个*DBD*

第*n*个*DBD*

图4.11 多个*DBD*的书写形式

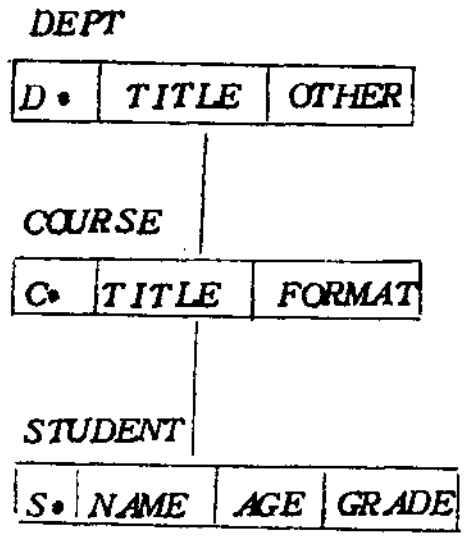


图4.12 教学数据库的一个*LDBRT*

4.2.3 *IMS*的*PSB*

前曾述及，用户使用的外模型是逻辑数据库记录类型(*LDBRT*)的集合，而每个*LDBRT*的一个*PCB*定义。一个*LDBRT*是从某个*PDBRT*中删去(或不删去)某些非根段类型和所属孩子而得到的。删除图4.6*PDBRT*中段类型*TEACHERA*和*TEACHERB*后，得到图4.12的一个*LDBRT*，并把该*LDBRT*的*PCB*写于其后。

此为试读，需要完整PDF请访问：www.ertongbook.com