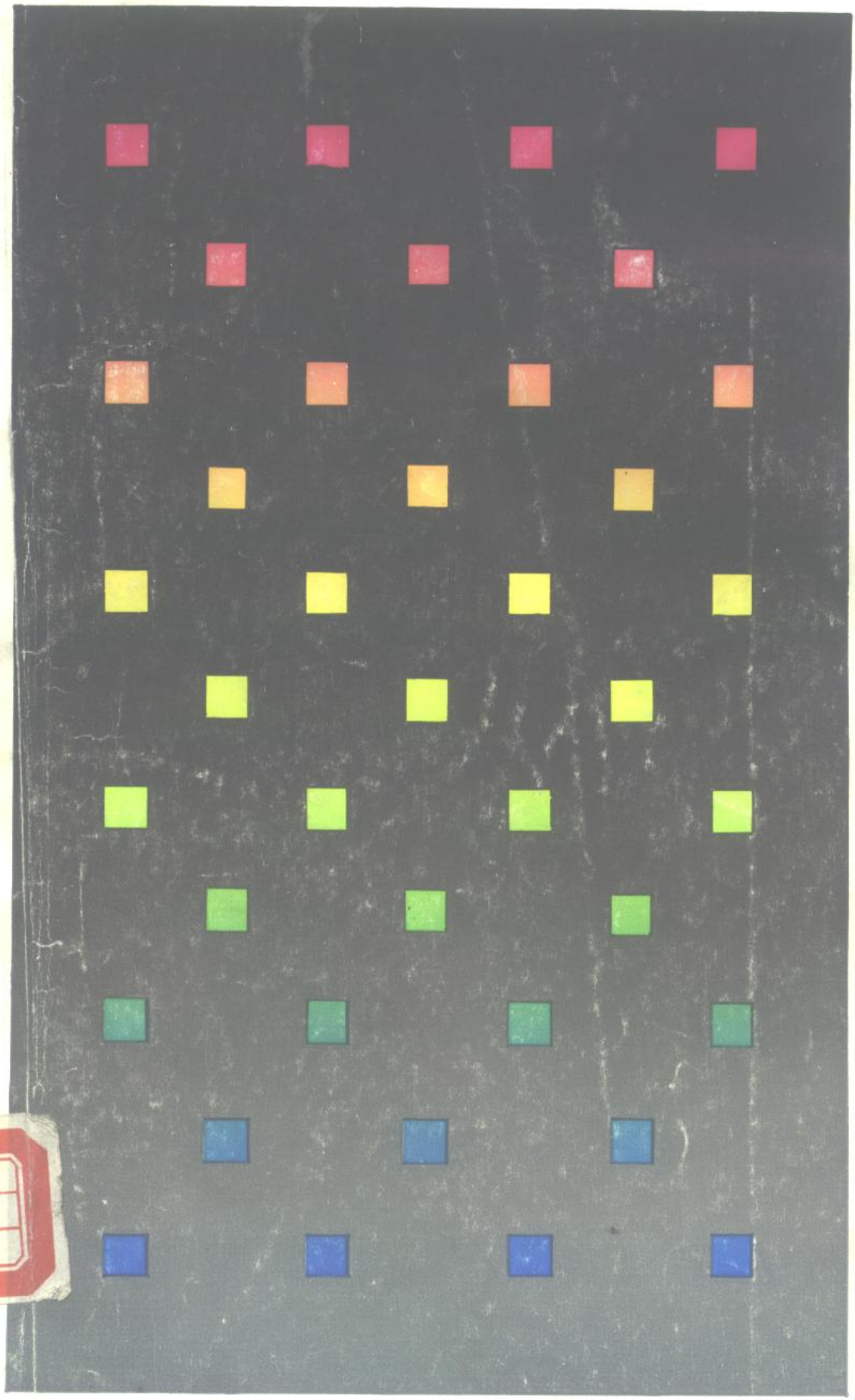
数据库管理系统实现技术

SHU JU KU GUAN LI XI TONG SHI XIAN JI SHU

周龙骧 编著



数据库管理系统实现技术

周龙骧　　编著

中国地质大学出版社

数据库管理系统实现技术
周龙骧　　编著
责任编辑　　张光前、李继英
责任校对　　杨　霖

中国地质大学(武汉市　喻家山)出版社出版
印刷厂印刷　湖北省新华书店发行

开本　787×1092　印张　15.875　字数　247千字
1990年12月第1版　1990年12月第1次印刷
印数　1－2000册　定价：4.20　元
ISBN 7－5625－0472－5/ TP. 10

内 容 简 介

本书较为全面介绍了数据库管理系统中的基本理论、实现技术及这一领域的最新进展。主要包括数据库管理系统的体系结构、并发控制理论与技术、完整性理论与技术、存贮管理和存取路径管理技术、恢复管理技术、查询语言的编译与优化方法以及最近发展起来的外部接口技术。

本书内容全面，取材新颖，基本上覆盖了数据库管理系统实现方面的全部内容，反映了国际上 80 年代中后期的研究水平。本书是作者十多年来主持和参与分布式数据库管理系统及国产中型机和微型机上数据库管理系弘的实现项目的经验总结。本书适于计算机专业的大学高年级学生及研究生作为教材，也可供从事数据库实现与应用及数据处理等方面工作的科研人员使用。

前　言

本书是1986年为北京软件研究生院和中国科学院软件研究所的研究生讲授“数据库管理系统实现技术”课程而编写的讲义。这次出版又补充了这两年有关新进展的内容。

在数据库的大学课程中一般都开设了“数据库系统引论”的课程，但其实现技术则尚来不及讲授。因此在研究生课程中开设这一课程是需要的。它和“数据库系统引论”的关系相当于“编译技术”和“程序设计语言”的关系。

本讲义的编写参考了T. Haerder教授所著“数据库系统实现方法”一书。考虑到这几年在本领域的进展以及所面向的学生的情况，补充和新编写了第一、第七、第十二章，并重新编排和改写了各章节的结构和内容。

Haerder教授的书是原联邦德国计算机科学系(信息学系)高年级课程(相当于我国硕士课程)的范本，是笔者见到的这一领域中比较好的一本书。Haerder教授曾长期在IBM研究实验室工作，亲身参与System R等系统的研制，他的书反映了这一背景。

笔者近年主持研制分布式数据库管理系统C－POREL及微型机数据库管理系统CDB的开发经验是与本书的内容相符的，因而可以说本书介绍的技术是经过实际考验的。本书还曾在海南高级系统分析员讲习班，华东师范大学，上海科技大学多次讲授，证明是一本合适的教材。

中国科学院数学研究所

计算机科学研究室

周龙骧

1989.3.4.

目　录

第一章　数据库系统(DBS)的发展回顾及其体系结构综述

Ⅰ. DBS领域的发展回顾

数据库系统(DBS)领域的发展可以回溯到60年代初商用语言COBOL和文件系统发展的时候,它首先是从应用领域发展起来的。勿庸讳言,早期的发展大多从实际的应用出发,因而显得琐碎零乱,缺乏统一的、严格的理论基础。在DBS领域工作的人和当时计算机科学与技术的其它主流领域,如程序设计语言,编译系统,操作系统,计算理论等方面的人相比,并不那么有声有色,他们之间的交往也不是太密切的。但是到70年代情况有了很大不同。以E. F. Codd提出的关系理论为标志所引起的一系列研究工作,把DBS理论的研究推向纵深[2,27]。数据系统语言委员会(CODASYL)的数据库任务组提出的DBTG方案*使导航式DBS的体系结构和理论的研究以及实现技术的发展开始了一个新的阶段[22]。ANSI/SPARC三级体系结构[35]的提出则使DBS体系结构的构想更臻完善。IBM公司圣约瑟研究实验室关于System R的研制则从实现技术,从效率上为关系模型奠定了牢固的基础。

对DBS领域来说整个70年代是开拓和深化的时代,热火朝天的研究工作和大量的实践活动,吸引了研究领域和实际开发领域的许多优秀人才,例如J. D. Ullman, S. Ginsburg等人。在这段时间有关数据库的国际会议每年要举行好几次,有些会议如VLDB参加者逾千人。经过这段时期的工作,DBS的理论基础建立起来了,许多问题搞清楚了,各种高效的实现技术也都开发出来,并提供了有效的产品。可以说对集中式数据库系统来说,从理论到实际的系统,都已趋于成熟。1973年C. W. Bachman [22]和1981年E. F. Codd[23]分别获得了计算机科学界的大奖——图林奖——是对DBS领域卓有成效的工作的褒扬。

进入80年代,DBS领域的发展方向已趋明确。如作为计算机特别是微机和通信网络的汇合点的分布式数据库系统的研究和发展已成为80年代计算机科学及其应用的主要方向之一[40]。到了80年代后期,经过十多年的研究与开发,DDBMS已有了若干原型系统(Prototype)研制成功,并已推出或正在推出一些市场产品,因此可以说分布式数据库系统的技术也已臻于成熟[108]。从80年代初开始(甚至可以上溯到70年代末),人们开始探讨数据库系统的新的研究方向。涉及于新的DBS样式,包括数据库机,数据库和人工智能的结合——演绎数据库,知识库,专家系统,数据库和计算机网络的结合;新的应用领域,包括管理信息系统(MIS),办公自动化系统(OAS),决策支持系统(DSS),计算机辅助设计(CAD),计算机辅助制造(CAM),计算机辅助软件工程(CASE),计算机辅助教育(CAE),计算机集成制造系统(CIMS);数据库作为大系统的组成部分的重要作用,包括第五代机系统及MIS, OAS, DSS,

*参见:"Data Base Task Group Report (Codasyl)",《电子计算机参考资料》1978年第12期。

CAD/CAM/CASE/CAE/CIMS；新的数据模型包括面向对象的模型；新的数据结构和表示，包括正文(text)、文档(document)、自然语言、图形、图象、语音等；新的用户接口和数据库理论等。这些新的方向使计算机科学和技术的各个领域有了新的结合和发展。由于研究工作的需要，一些新的刊物创办起来。可以期望，在这个十年将会有许多新的收获和进展，例如新一代DBMS的面貌将会更加明朗甚至会研制成功一些原型系统[107]。

II. 数据处理的发展和数据库系统

计算机的使用从40年代中期开始一直沿着一条固有的轨道发展，这就是力图使计算机的用户界面更为友善，更为方便，更得心应手。最早期的用户须得是"软硬兼施"的专家，算一道题目连硬件线路都得进行搭配。后来计算机较为普及以后，用户也得是专门的程序员，他们要自行编制I/O子程序，直接为数据的存贮分配操心。这是数据加工的手编程序阶段。

操作系统的出现是一个新的阶段，提供了通用的存取数据的方法，如SAM, ISAM等。将I/O, 存贮分配等标准化，从而用户不必为存贮分配操心，存贮分配结构参数的变化不再影响应用程序。

数据处理的第三个阶段是出现了文件管理系统，承担各种文件中的数据的排序，更新，准备和形成报告(如RPG, MARK Ⅳ)，用简单的描述型语言进行一定的数据处理。用户不再与数据的存贮格式和类型打交道，达到了一定的数据独立性。

从时间上来划分，人们通常把50年代到60年代初的基于顺序文件(即磁带文件)DMS称做第一代DMS。把60年代中期为止的基于直接存取文件，即基于分时，交互式终端，磁鼓，磁盘技术的DMS称做第二代DMS。这一时期的索引，杂凑(Hash)等技术已充分发展，可以说文件系统已经很成熟了。

第四个阶段是出现了通用的数据库管理系统，如IDS, IMS, CODASYL DBTG等系统，引入了DDL和DML，利用过程型语言来处理若干文件中的数据。这一代系统是努力使领航式(Navigation)的程序员仅仅面对逻辑的数据记录和逻辑的存取路径，而摆脱对这些逻辑结构的具体实现的依赖。应该指出的是许多名义上按DBTG方案实现的系统并未达到这一目标，程序员不得不在数据说明中开列一系列物理特征量。CODASYL1971年推荐文本是DBTG报告的发展，它标志以C. W. Bachman为首的第三代DMS达到了巅峰状态。Bachman在1973年获得了计算机科学界的最高奖——图林奖。他的图林演讲的题目是："作为导航员的程序员"[22]。

数据处理的第五阶段的特征是使用户勿须导航，在更高级的层次上面对逻辑的数据结构，例如关系，使用基于集合论的非过程型的描述型语言接口，例如关系代数，这样来达到高度的数据独立性。同时使程序员完全摆脱数据的管理，诸如存贮、检索、修改等的实现方法和效率问题完全由系统自动完成，从而使用户可以完全专心于他的应用问题，这就大大提高了生产率。关系模型的奠基人E. F. Cood在1981年接受图林奖时的演讲题目就是："提高生产率的现实基础"[23]。美国国家标准协会(ANSI)1986年10月批准的美国国家标准SQL是以Codd为首的第四代DMS达到顶峰的标志。

以上从纵的方向介绍了数据处理的五个发展阶段或四代数据管理系统(DMS)。从横的方

向看，这就是从以程序为中心向以数据为中心的转变。传统的程序设计语言如 FORTRAN, ALGOL60,BASIC, PL/1, ALGOL68, PASCAL 以及最新的 Ada, 都是典型的以程序为中心的，数据或文件只是程序加工的对象。但是数据库系统则不同，它是以数据为中心，围绕着数据进行查、添、删、改。

下面看一下以程序为中心的程序设计所用的传统的文件管理系统(FMS)和以数据为中心的数据库系统(DBS)的不同点：

以程序为中心的 FMS	以数据为中心的DBS
• 以独立解决问题为特征，以程序为中心。各个用户，各个应用程序都有各自独立的数据文件。	• 以数据为中心，数据库中的数据为各个用户所公用。
• 冗余度大。同一企业，同一单位各个应用程序各自独立地管理自己的文件，各文件存贮了重复的数据。	• 冗余变小。
• 应用程序自己负责数据的完整性，安全性检查。由于各应用程序分散独立负责检查和修改，难于保证数据的一致性，例如关联完整性(Referential Integrity)。多个副本修改困难，不能保证及时修改，导致不一致。	• 数据的完整性，安全性，一致性由系统统一负责检查和维护。一般的 DBS 其完整性约束规则约占系统概念模式描述程序量的 80%，检查和维护的工作量很大。
• 数据独立性差。每个应用程序有自己的专用数据，数据的结构面向应用，应用程序密切依赖数据结构的细节。应用程序的存取逻辑依赖于数据的存贮结构。当硬件软件变更或更新时引起存贮结构变化，应用程序要跟着变化，物理数据独立性差。当逻辑数据结构变化时如增加新的场(field)，新的实体(Entity) 时，或重构某些文件时(如一个文件分解为几个文件时)，应用程序要随之改变，亦即逻辑数据独立性差。	• 采用分层体系结构，数据独立性强，各层之间是透明的，对实现细节加以隐蔽，接口清晰。当一级变化时不影响其它级。
• 不能自动恢复。当硬件故障或系统崩溃(Crash) 时，无法恢复。	• 系统自动恢复。硬件故障或系统崩溃时，DBMS 的恢复子系统可保存已处理数据及现场，系统恢复后再从断点自动继续运行。

以上纵横两个方向的发展，导致数据库系统的建立。有了 DBS，使用户从既要致力于他的应用，又要操心数据的管理、检查转变为只专心考虑他想要解决的应用问题。他不必被要求去知道怎样存取数据，通过哪些存取路径，以及用了哪些实现技术。这就是“知必所需”的原则，既提高了数据独立性，又提高了生产效率。

在论述 DBS 体系结构之前介绍一个例子，以便让读者回忆一下数据独立性是有益的。

设有一个雇员关系：

EMPLOYEE

NAME	OFFICE	JOB	SALARY
SMITH	PARIS	SALES	15000
JONES	BONN	SALES	18000
CLARK	BOISE	SALES	12000
JONES	BOSTON	SERVICE	17000
KENT	PARIS	SERVICE	15000
DAVIS	LONDON	SERVICE	13000
JACOB	RIO	SALES	12000

和一个办公室关系：

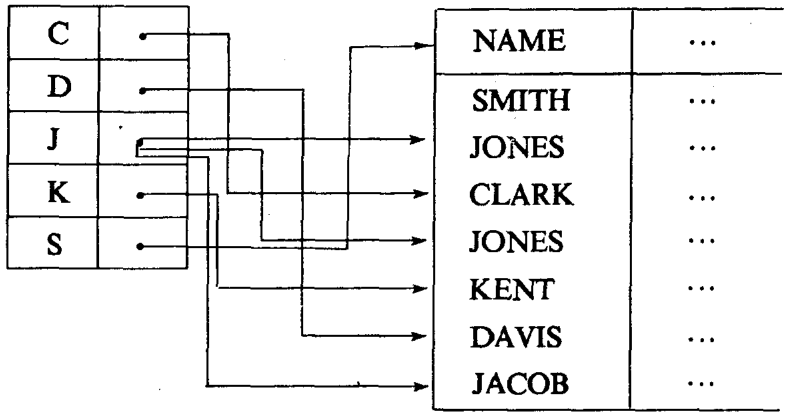
OFFICE

LOCATION	MANAGER	PHONE
SAN JOSE	BLASGEN	7152
PARIS	PORTAL	9123
LONDON	PORTAL	3278
BONN	ROWER	1287

在磁盘中雇员数据的存贮安排可能是下面的样子:

SMITH PARIS SALES 15000 JONES …

按字母顺序建立的人名索引可能是:



如果有一个查询是找出在PORTAL的办公室中雇员的职务为SERVICE的雇员名。用SQL语言来写这个询问就是:

```
SELECT    NAME
FROM      EMPLOYEE    OFFICE
WHERE     EMPLOYEE .  OFFICE = OFFICE . LOCATION
          AND  OFFICE . MANAGER = "PORTAL"
          AND  EMPLOYEE . JOB = "SERVICE";
```

以上询问仅仅指出了所需的是什么(What),但未指出如何去得到它(How)。

要完成这个查询至少可以有三种选择:

1. 先查关系EMPLOYEE,找出所有JOB为SERVICE的元组。再对每一找出的元组按其OFFICE属性值在关系OFFICE中查该雇员是否属于PORTAL。

2. 先查关系OFFICE找出PORTAL的办公地点,然后按此地点查关系EMPLOYEE,找出JOB为SERVICE的雇员。

3. 先将一个关系或两个关系排序,然后查找。

但是用户不必关心选择哪一种办法,不必考虑是否要通过索引,是否要排序等等。这就减轻了用户的负担,提高了他的生产效率。这些有关存贮,存取路径的"细节"用户不仅不必知道,

而且禁止他利用这些"细节"去写更高效的程序。于是当存贮结构,存取路径结构变化时用户的应用程序保持不变,这就是数据独立性。

设有两个定期执行的事务:

T_1:列出给定办公室的所有雇员。

T_2:列出从事某一职业的所有雇员。

如果事先知道事务 T_1 要花 99% 的时间运行而事务 T_2 只花 1% 的时间运行。则正确的做法应该是对 EMPLOYEE 关系的 OFFICE 属性建立索引,在盘中相同办公室雇员的记录应放在一起存贮(簇聚(Cluster)存贮)。

若情况发生变化,T_1, T_2 的运行时间倒了过来,T_1 花费 1%,而 T_2 花费 99%,则应撤消对于 OFFICE 属性建立的索引,建立对 JOB 属性的索引。存贮也要重整,将相同 JOB 的雇员记录放在一起存贮。

但是事务 T_1, T_2 一点也不需要改变!这就是存取路径独立性和物理存贮独立性。

现在再看一个例子。设 INVENTORY 数据库关系的模式最初是针对一种部件只能在一个仓库存放而设计的:

INVENTORY

WAREHOUSE #	PART #	QUANTITY ON HAND

用于列出货源不足的部件的程序可写为:

```
SELECT PART #, QUANTITY ON HAND
FROM INVENTORY
WHERE QUANTITY ON HAND THRESHOLD;
```

如果情况改变为一种部件可在多个仓库存放,则上述查询应改为:

```
SELECT PART , SUM (QUANTITY ON HAND)
FROM INVENTORY
GROUP BY PART #
HAVING SUM (QUANTITY ON HAND) THRESHOLD;
```

亦即应用程序(前一个查询)在关系模式改变时不是不变的,这就是说它不具有逻辑数据独立性。

有了这些准备,现在来看 DBS 的体系结构。

Ⅲ. DBS 体系结构综述

上一节引出的 DBS 是一个贮存管理和检查大量数据,为用户提供查、添、删、改数据对象的各种服务的大型系统,它实际上由两部分构成:其一是数据库 DB,指集中存贮的、互相联系的数据总和;其二是数据库管理系统 DBMS,指管理上述数据的系统。DBS 就介于用户和贮存于计算机存贮器中的数据之间。

从各种各样的特定用户到存于外存的数据之间的鸿沟是如何由 DBS 来填平的呢?亦即

DBS是如何将各种各样的数据请求转换到实在的数据存取呢？描述这一点的总框架就是DBS的体系结构。

对DBS的体系结构曾进行过深入而广泛地研究，产生过许多著名的模型。其中一项著名的工作是M. E. Senko等的数据独立存取模型DIAM[14,15]（Data Independent Architecture Model Access）。DIAM将DBS划分为四级互相衔接而又相互独立的层次，它们是：

- 逻辑数据结构(Entity Set Model 实体集模型)
- 逻辑存取路径结构(String Model 串模型)
- 存贮结构(Encoding Model 编码模型)
- 存贮器分配结构(Physical Device Model 物理设备模型)

这四级的每一级均由自己的描述语言说明，各级互相衔接又相互独立，每级都尽量少对邻接的下一级作出假定。从而各级留有充分的余地，各级的修改不致反作用于上一级。

由底向上，存贮器分配结构级负责存贮介质上数据的存贮分配，涉及存贮介质的参数和性质。例如盘类型、柱面号、道号、道长、块长、溢出区位置等。

存贮结构级负责描述数据记录的物理存贮，实现存取路径。它已完全脱离具体的物理设备，认为存贮器是一个整个的主存，进行线性编址。通过一个专门的系统成份来执行从存贮结构向存贮器分配结构的映射(mapping)。这样当存贮器分配结构级更新物理设备的类型和参数时，就同上一级隔离开来，在存贮结构上运行的程序勿须改变，而只须改变映射过程，从而达到了设备独立性。

逻辑存取路径结构级说明数据记录间的相互联系，存取和选择记录的功能以及对记录进行排序的功能。它说明为系统提供了哪些存取手段，但对实现方法则加以掩蔽。这样当对某存取路径引入新的实现技术时，不会影响到仅仅使用了此存取路径名字和功能的系统成份，从而获得存贮结构的独立性。

逻辑数据结构级把用户和一切存取隔离开。它所提供的逻辑数据结构对一切应用均为中性，例如规范化的关系。它所表示的是DBS所刻划的现实世界的一个剖面或称微观世界，它完整地，无冗余地表达了微观世界的信息。在这里存取路径对用户是透明的，与其上运行的程序无关，从而达到存取路径独立性。

至此我们还可以给DIAM加上一层第五级，即面向特定用户的一级。这一级提供面向特定用户的视图，这些视图可模拟其它数据模型，除关系型外还可是网状型和层次型，甚至支持各种专用语言以至接近自然语言的语言。G. M. Nijssen断言[34]下一代DBS体系结构应能支持主要的数据模型共存于一DBS之中并具有多级模型的全部特点。第五级将特定用户的逻辑视图同系统的逻辑数据结构的可能变化隔离开来，达到了逻辑数据独立性。逻辑数据结构的变更只影响视图如何定义和映射，不影响视图本身。

另一著名的影响深远的关于DBS体系结构的模型是美国国家标准研究所的标准计划和要求委员会(The Standards Planning and Requirements Committee of the American National Standards Institute)提出的ANSI/X3/SPARC框架。它是一种三级模式，即概念模式，外模式和内模式。其核心是概念模式。概念模式描述所刻划的现实世界，对微观世界的逻辑数据结构提供一种完整的，无冗余的，中性的表示。以概念模式为核心，一方面映射到描述各种物理结构的内模式；另一方面映射到一系列导出的外模式，亦即用户数据模型。

ANSI/SPARC DBS体系结构的示意见图 1.1 。图1.1 中的映射和转换的细节可以精细化为图1.2 。

按照 ANSI/SPARC 框架,数据库系统体系结构的另一种形象表示则如图 1.3 所示[2]。

另一个著名的同时也是 DBS 标准化的第一个方案就是 CODASYL DBTG方案。DBTG 方案如图 1.4(a)所示由模式 / 子模式两级组成。模式从系统的角度描述数据库,包含有数据结构(Record),逻辑存取路径(LOCATION MODE , SET OCCURRENCE SELECTION),存贮结构(CHAIN , POINTER ARRAY)等的说明。在模式中不包含存贮介质和存贮器分配结构的说明。

子模式是模式的简单子集,它面向特定的应用,使应用程序和描述数据库的模式隔离开来。如图 1.4 (b)所示,当模式发生变化时,基于子模式之上的应用程序勿须改变,而只须改变从模式到子模式的映射即可。也如该图所示,这种映射的重新建立牵涉到每一个子模式。加上 DBTG 方案的导航式的特点,这个方案只保证了设备独立性,而未完全保证数据独立性。

ANSI/SPARC 三级模式的影响很深远,很多其它方案都按它的思想进行了改进或澄清。例如 1978 年的CODASYL DBTG 的修正方案,采纳了三级模式的概念[37,38],如图 1.4 (c),(d)所示。Senko 的 DIAM 模型也按 ANSI/SPARC 三级模式进行了解释[39],如图 1.5 所示。

以上所述几种 DBS 体系结构的方案,汇总起来列成一对比的表格,如图1.6 所示[36]。

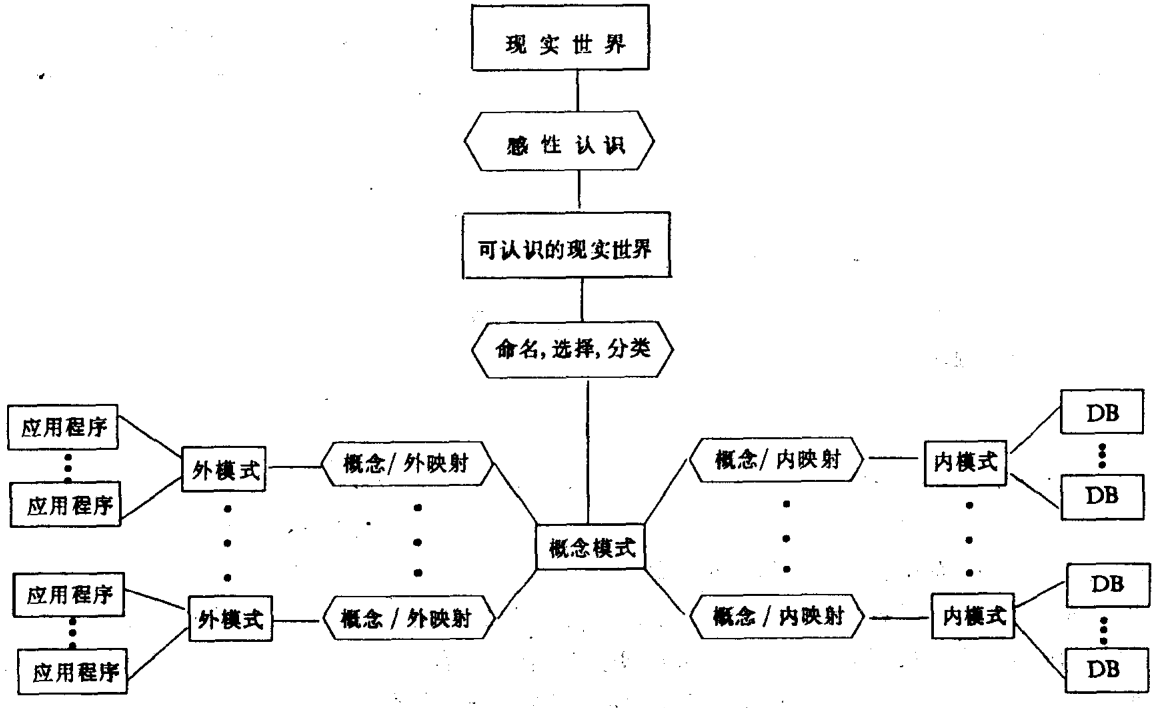


图 1.1 ANSI/SPARC DBS的体系结构

Ⅳ. 数据模型

在DBS 体系结构中一个重要的核心问题是数据独立性。与这个问题密切相关的是数据模型的概念。