数据库系统原理

〔美〕J.D.厄尔曼 著

张作民 译

国防工业出版社

内 容 简 介

本书介绍数据库系统的概念、方法和某些主要理论结果。全书分为十章。前三章概述系统结构、基本物理数据组织方法和最主要的数据模型。第四至六章讨论关系数据库系统，包括各种有代表性的查询语言、数据库设计理论和查询优化方法。第七、八两章扼要介绍DBTG型系统和IMS系统。最后两章讨论数据完整性、安全性和并行操作下的数据相容性问题。各章末附有习题和参考文献注记。本书内容丰富，叙述简练，概念深入，又辅有例子。它可作为高等学校计算机科学、软件和其他与计算机数据处理有关的专业的大学生或研究生教材或教学参考书，也可供计算机信息系统和数据处理应用的分析员和程序员参考。

PRINCIPLES OF DATABASE SYSTEMS
Jeffrey D. Ullman
Computer Science Press 1980

*

数据库系统原理

〔美〕J.D.厄尔曼 著
张作民 译

*

国防工业出版社出版
新华书店北京发行所发行 各地新华书店经售
门头沟区印刷厂印装

*

850×1168 1/32 印张 13 1/4 346千字
1984年11月第一版 1988年5月第二次印刷 印数：14,601—21,600册
ISBN7—118—00299—2/TP34 定价：3.00元

译 者 序

J. D. 厄尔曼（Jeffery D. Ullman）教授是美国有名望的计算机科学家，他在计算机算法、自动机与语言、数据结构、编译系统和数据库系统等许多领域从事研究与教学工作，发表了大量论文，撰写了不少专著，《数据库系统原理》一书是他在一九八〇年出版的一部。

这本书是作者根据他在美国普林斯顿大学使用的讲义整理的，目前他在斯坦福大学计算机科学系为高年级学生和研究生讲授数据库课时仍以它为教科书。如同许多数据库教科书一样，书中包括了很多描述性材料，叙述数据库系统的一般原理，介绍现有的某些有代表性的系统在结构、数据模型、语言和物理实现方面的特点。然而，作者不是简单地罗列材料、拘泥于细节，而是从较高的观点，简练地叙述各种概念，并把它们和程序设计语言、算法和数据结构等联系起来。这本书的另一大特点是，作者花费了相当大的篇幅，总结了关系数据库设计、关系查询语言与优化实现方面的已有理论结果，同时还概括了在统计数据库的安全性、数据库并行操作的控制和数据库恢复方面的最新研究成果。这些材料在大部分数据库教科书里是找不到的。

全书共分十章。第一章概述数据库系统的总体结构和主要观念。第二章描述作为数据库存储结构基础的各种基本文件组织方法。第三章概括了目前现有数据库系统使用的三种主要数据模型。第四章讨论关系数据库的查询语言，证明了关系代数、元组关系演算和域关系演算的功能等价性，并介绍了各种有代表性的现有查询语言，证明了它们的关系完备性。第五章讨论关系数据库设计理论，仔细研究了函数依赖、多值依赖和有关的公理系统，也给出了各种关系范式的定义以及关系模式的分解理论。第六章

讨论关系查询优化的理论，给出了某些优化算法。第七和第八章分别介绍 DBTG 型系统和 IMS 系统的主要特点。第九章介绍一般数据库系统的完整性与安全性设备，并讨论了统计数据库所特有的安全性问题。最后在第十章讨论数据库的并行操作，介绍了几种模型和用来保证调度可串行化的协议，也讨论了在并行环境下的数据库恢复问题。每章后附有练习题和文献注记，进一步扩充了正文内容，介绍了课题的历史发展过程和某些文献的重要思想。

总之，本书用较短的篇幅叙述了丰富的内容，结构严谨，叙述简练，而且不要求读者有特别专门的预备知识。因此，它不失为一本较好的教科书和参考书，凡具有计算机程序设计基本知识的读者都可阅读本书，至多在理论证明部分遇到某些困难。

数据库的术语尚未统一，本译文尽量采用目前国内通用的术语，书末给出了主要术语的英汉对照。各专门术语的含义都可在正文中找到。原书使用了许多例子，习题中也涉及到一些实际例子。例中有些“名字”（如属性名）的含义未加说明，因为在西方人看来这些是不言自明的。为方便读者，译文中插进了某些必要的中文解释而未加译注，这些解释放在括弧里或原书也使用的语言注释符号/*…*/中。

译文订正了原书中的某些笔误和印刷错误。译者虽然力求翻译准确，但受水平限制，误译之处在所难免，望读者指正。

本书的翻译得到了中国人民大学萨师煊教授和他的研究生的支持与帮助，也得到了北京市计算中心周小刚等同志的帮助，译者在此表示衷心的感谢。

1982.8

前　言

很明显，数据库系统课程目前在计算机科学的大学生和研究生教学计划中占有重要的地位。然而，它不象那些较为传统、已被较好开发的系统领域，如编译系统和操作系统，在这些领域，原理部分和实践方面的材料多年前就较好地融合在一起了，而数据库系统这门课程的内容却一向是以描述性材料为主的。

本书是根据我在普林斯顿大学为四年级大学生和一年级研究生开设的一门课程里使用的讲义整理而成的。在这门课程里，我试图把数据库系统纳入到计算机科学的主流中去，并把数据库的概念与程序设计语言、算法和数据结构等其他领域中的概念联系起来。课程里包括了大量的描述性材料，因为考虑到学生已习惯于通常的程序设计语言，他们可能会感到数据库查询语言颇不寻常。还有，适合于数据库的数据结构与通常程序设计语言所用的数据结构也不尽相同，这是因为，数据库的规模之大已使得许多原来只有理论兴趣的结构变成现实可行的了。

可是，在讲述各部分内容时，我加进了当前已有的有关理论结果。那些很重要的、已被发现有用的是关于关系和并行控制的概念。我在本书里花了大量的篇幅描述关系、关系代数和关系演算，以及利用这些概念设计出来的查询语言。我也阐述了如何利用关系数据库的理论来设计好的系统，并讨论了关系数据查询语言的查询优化。本书还用一章讨论了在多进程并行操作环境下保证数据库相容性的问题，介绍了最新提出来的一些协议。

书中每章末都附有练习题，用来检查对基本概念的理解；但也有一些练习扩充了正文中的思想。最难的题目标上了双星号，稍难的标有单星号。

作者谨向A. 阿霍 (Al Aho)、B. 贝克 (Brenda Baker)、

P. 德约恩 (Peter deJong)、R. 费根 (Ron Fagin)、V. 海斯拉科斯 (Vassos Hadzilacos)、Z. 凯德姆 (Zvi Kedem)、H. 考斯 (Hank Korth) 和J. 斯品特恩 (Joseph Spinden) 表示谢意，他们给本书提出了若干意见和建议。也感谢为本书手稿打字的G. 皮特 (Gerree Pecht) 以及普林斯顿大学给予的支持。

J. D. 厄尔曼

目　录

第一章　数据库系统概述

在这一章里，我们考虑典型数据库管理系统中的各级不同抽象，并且讨论这样一个系统的主要功能。然后讨论一个能借以衡量数据库系统表示和处置真实数据能力的“真实世界”模型。这个模型称为“实体—联系”模型，将在 1.4 节里描述。

1.1　数据库系统

让我们来考虑一个企业，如某航空公司，它有大量数据要长时间周期性地保存在计算机里。航空公司的数据可能包括有关旅客、航班、飞机和公司职工等信息。可能要表示的典型联系有订票（哪些旅客订了哪些航班的座票？）、机组人员（谁在哪些航班担任驾驶员、副驾驶员、……？）和检修记录（每一架飞机由谁在何时最后一次检修的？）。

象上面这样的在一段时期内保存在计算机里的数据，称之为数据库。容许人们使用和修改这些数据的软件称为数据库管理系统(DBMS)。DBMS 的主要作用是容许用户以抽象的字眼同数据打交道，而不是象计算机硬件存储数据那样。从这个意义上说，DBMS 很象一个如 APL 这样的（特）高级语言的解释程序，它容许用户指定需要做什么，而无须留心或只须很少留心详细的算法或系统采用的数据表示方法。不仅如此，在 DBMS 的情形，用户所看到的数据和计算机中实际存储的数据之间的关系，甚至可能比 APL 数组和数组在存储器中的表示之间的关系还少。

DBMS 还能够而且应该完成许多其他功能，其中包括：

1. 安全性　并非每个用户都应该存取全部数据。例如，如果人事信息被存储，只是那些有权且需要了解工资情况的人才能存取这些数据。我们将在第九章讨论 DBMS 的这一方面。

2. 完整性 某些类型的相容性约束（即数据应当具有的性质）能够由DBMS来检验，如果它被通知这样做的话。最容易检验的是值的性质，例如预订同一航班的旅客数不应超过飞机的定员数。稍难检验的是那些涉及值的相等或不相等而与值本身无关的要求（例如，两架飞机不可以被派给同一航班）。第五章讨论结构完整性的某些方面，第九章对完整性作一般的讨论。

3. 同步控制 常常有许多用户在同时运行存取同一数据库的程序。DBMS应当提供保护，以防止由两个几乎同时发生在同一数据项上的操作而导致的不相容。比如说，假设在大约相同的时刻，两个订票员发出申请，各要预订999次航班的一个座位。每一申请都引起一个程序被执行，这个程序可能先查出还有多少空座位（比如说1），然后减去1，再把剩下的座位数存放到数据库里。假如DBMS没有适当地安排这两个处置（即订票程序的两次执行）的次序，两个旅客有可能订上同一座位。我们将在第十章研究保证正确同步控制的措施。

4. 故障防护与恢复 应当提供制作数据库的后备拷贝和在硬件或软件出错之后恢复数据库的设备。这个课题也在第十章讨论。

1.2 DBMS中的各级抽象

计算机处理的是二进制字位，而最终用户处理的是象航班或给飞机指派机组人员这样的抽象。因此，显然在这两者之间将有许多级的抽象。图1.1给出了关于分级抽象的一种较为标准的观点。在那里，我们从三级不同的抽象看同一数据库（它可能是利用同一DBMS软件的许多数据库中的一个）。应当强调，只有物理数据库是实际存在的。对于物理数据库，我们不想在二进制字位级而宁愿在更高一点级别上去看它，即认为物理数据库是文件和简单数据结构的集合。

概念数据库是物理数据库的抽象表示（或者也可以说物理数据库是概念数据库的实现），而每一个视图则是概念数据库的某一

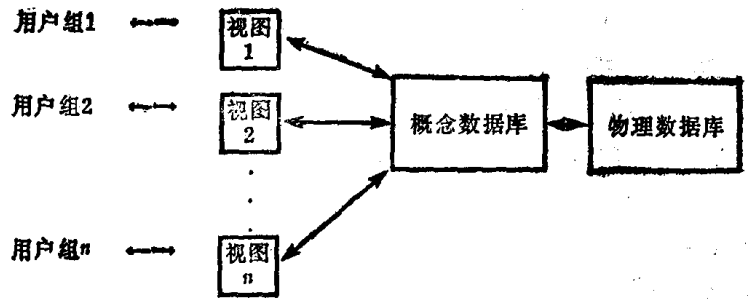

图1.1 数据库系统中的诸级抽象

部分的抽象。视图与概念数据库在抽象级上的差别一般是不大的，它们处理的都是象“旅客”这样的抽象事物和象“订票”一类的抽象联系。

模式和实例

除了图 1.1 那样的分级抽象外，还需要从另一“垂直的”方面来理解数据库。当设计数据库时，所关心的是数据库的规划。当使用数据库时，所关心的乃是数据库中出现的实际数据。我们把一个数据库的当前内容称为这个数据库的一个实例。注意，数据库中的数据经常在变，而数据库的规划则在一段长时间内保持不变（虽然未必永远不变）。

规划的内容包括列举数据库所涉及到的各个实体类型、这些类型实体之间的联系以及在一个抽象级上的实体和联系用更低（更具体）一级抽象表示的方式。对于规划，我们使用术语模式。于是，概念模式就是对概念数据库的规划，而物理数据库规划则叫做物理模式。视图的规划常常简称为子模式。

例1.1 为说明模式与实例之间的区别，假设有一个关于花草的数据库。概念模式可能包括实体型FLOWER（花）和HABITAT（产地）及其他；还可能包括FLOWER与HABITAT之间的联系GROWS-IN（产于）及其他联系。概念数据库中可能包括类型为FLOWER的实体rose（玫瑰花）和tulip（郁金

香）以及诸如 tulip GROWS-IN Holland（郁金香产于荷兰）、rose GROWS-IN Texas(玫瑰产于得克萨斯)和rose GROWS-IN Tralee（玫瑰产于特拉利）等信息。这里，Holland、Texas和 Tralee 是实体型 HIBITAT 的三个实例。

在下一级抽象即物理数据库模式里可能规定FLOWER 型和HIBITAT 型的实体各是长度为10的字符串，规定 GROWS-IN 用链表表示，每一种花有一个产地链表，其表头通过对花名应用某个特定的散列函数确定。□

物理数据库

我们处理的最低一级抽象是物理数据库。物理数据库总是驻在磁盘或磁带这样的辅助存储设备上。可以在几个抽象级上看物理数据库：从如 PL/I 一类程序设计语言中的记录和文件一级，可能经过由 DBMS 所基于的操作系统支持的逻辑记录一级，直到存储设备上的二进制字位和物理地址这一级。第二章讨论实现物理数据库所用的主要数据结构，第四、七、八章将指出某些重要的现有数据库系统有代表性的实现特点。

概念模式及其数据模型

正如已说过的，概念模式是同某个企业有关的真实世界的抽象。就已经讨论的航空公司而言，它大致是在旅客、航班等这一级上。DBMS 提供一个数据定义语言，用来定义概念模式，而且很可能还包括概念模式用物理模式实现的某些细节。数据定义语言是一种高级语言，它使人们能用一个“数据模型”来描述概念模式。一个适当的数据模型例子是有向图（常称网状模型），其中结点代表类似实体（如全部旅客或所有的航班）的集合，弧表示联系（如给航班指派飞机）。

选择一个数据模型是困难的，因为这个模型必须有足够丰富的结构，以便能够描述真实世界中各种有意义的方面。不仅如此，它还应当使自动确定物理模式实现概念模式的有效方法成为可

能。应该强调的是，尽管 DBMS 可以用来建立小型数据库，但不要忘记，许多数据库可能包含上百万字节，因此，低效的实现可能是灾难性的。很明显，如果对于要找的特定信息在物理数据库中的位置没有任何线索，则有可能要花上几小时查遍整个数据库才能找到它。

已经在数据库系统中使用的数据模型至少有三种是主要的：

1．层次模型 该数据模型是树，其中结点代表实体集，如航班，一结点的儿子同这个结点之间有某一特定的联系。例如，预订同一航班的所有旅客可以是该航班结点的儿子。

2．网状模型 这就是前面已经提到的有向图模型。

3．关系模型 这种模型以关系的集合论概念为基础。一个关系乃是 k-元组（k 是某固定常数）的集合。例如，预订机票可以用3-元组

FLIGHT-NO DATE PASSENGER

/*航班号 日期 旅客*/

的集合 BOOKINGS表示，即BOOKINGS＝{(n, d, p)| 旅客 p 预订了日期 d 的第 n 次航班}

这三种数据模型将在第三章介绍，与每一模型有关的理论和技术随后分别讨论。第四至六章讨论关系模型。第七章讨论DBTG建议，它是网状模型中影响最大的一个。第八章讨论 IMS，一个主要的层次模型系统。在 1.4 节，我们将讨论一种更一般些的模型，叫做实体—联系模型；在某种意义上说，这种模型概括了上面所说的三种模型。

视　图

视图或子模式是概念数据库或概念模式的某一部分的抽象模型。例如，一个航空公司可以提供一项计算机化了的订票服务业务，它包括与航班和旅客有关的数据和一组程序。这些程序和使用它们的人并不需要了解公司职工文件和机组人员的分配。调度人员可能需要了解航班、飞机和职工文件的某些方面（如哪些飞

行员有资格驾驶波音747）情况，但他们不需要了解职工的工资及哪些旅客预订了哪一航班的机票。

从某种意义上说，视图不过是一个小的概念数据库，它和概念数据库有同一抽象级。但是在另外的意义上，视图可以是比概念数据库“更加抽象”的东西，因为视图所涉及的数据可能是由概念数据库构造出来的，而并非数据库里实际存在的。

我们举一个典型例子。人事部门可能有一个包括每个职工年龄的视图。可是，年龄不大可能在概念数据库里找到，因为每天都会有一些职工的年龄要变。因此，更为可能的是，概念数据库包含每个职工的出生日期。假设有一用户程序，它以为它在同一个含有年龄信息的视图打交道。但实际上，当它向数据库申请某一职工的年龄值时，DBMS却把这一申请翻译成“当前日期减去出生日期”（这对概念数据库来说是有意义的），并且这一计算会根据从物理数据库中取出相应的数据来完成。

说视图是概念数据库的抽象的另一个、而且更重要的方面是：在子模式和它所基于的概念模式里，联系可以是不同的。例如，在概念数据库里，航班可以是旅客的集合，而在订票的视图里，一个旅客可以是他所预订的航班的集合。这一区别可能是重要的，也可能不重要，取决于定义概念数据库和定义视图所用的数据模型。实际上，视图不一定要用与概念数据库相同的数据模型来定义，即使大多数既支持概念模式又支持视图的数据库系统对两者使用了相同的数据模型。

1.3 数据库的其他概念

设计、实现和使用一个大型数据库需要许多人的努力。仍以航空公司数据库为例。这个数据库的最终用户一般不是程序人员，譬如是一个订票员。订票员可能坐在终端前，打入一个简单命令，例如BOOK。这个命令启动一个程序，它开始同订票员对话，依一定次序请订票员提供信息，如请他“打入旅客的姓名”和“打入要预订的航班号”。程序在收到所需要的信息后开始询问

数据库，看是否还有空座位。如果有，则适当地修改数据库，以反映又一座位已被预订。如果没有，则发一消息通知订票员。

这个操作看起来是简单的；但怎样编写这个程序，使它能同数据库通讯呢？图 1.1 意味着用户只能同视图通讯，可视图离物理数据库还有两级。这一问题的解决在于：有若干专门的语言被用来在各抽象级上定义数据库及操作数据库。

设计概念模式

我们已经说过，概念模式是用数据定义语言(它是 DBMS 的一部分）定义的。这个语言不是一种描述过程的语言，而是依特定的数据模型描述实体类型和实体类型之间联系的一些符号。数据定义语言在数据库设计时用，在设计被修改时也用，但是它不用于检索和修改数据本身。数据定义语言也几乎总是包括一些这样的语句，它们以某种抽象的字眼描述数据库的物理布局应该怎样，而详细的物理数据库设计是由处理数据定义语言语句的 DBMS 子程序来作的。

在实现一个大型数据库时，其概念模式的设计将由一个一般称为数据库管理员的人担任，他通常需要和一班程序员协作并且与那些主管建立数据库的人，如公司的管理人员商量。数据库管理员和他的工作班子利用数据定义语言规定概念模式及其到物理数据库的实现。在数据库使用过程中，所有那些从总体上影响数据库的操作都由数据库管理员负责。这包括：

1．建立视图的子模式。

2．向用户授予对数据库或它的某部分的使用权限。

3．当原设计已被证明有缺陷或企业要求改变时，修改概念模式。

4．如果数据库使用情况表明，采用另外的物理数据库组织会更有效时，修改概念模式到物理模式的实现。

5．制作数据库的后备拷贝，修复已给数据库造成的损害。

子模式的描述及其同概念模式的对应要求有一个子模式数据

定义语言。它常常与数据定义语言十分类似，但在某些情形，子模式语言可以使用与数据定义语言不同的数据模型。其实，可以有几种不同的子模式语言，每一种使用一个不同的数据模型。数据库管理员的职能(2)，即数据库存取权限的控制，需要有一专门的语言描述要授予的权力和授权的对象。这个语言可以是数据定义语言的一部分，也可以是数据处置语言的一部分。职能(3)和(4)是利用数据定义语言完成的，而为保证系统可靠性的职能(5)将通过 DBMS 提供的专门手段来实行。

应用程序

假设某航空公司数据库的概念模式已经设计，并决定建立一个自动订票系统。订票系统用的子模式及其用概念模式表示的含义由使用它的应用程序员班子设计，数据库管理员给予他们存取相应于该视图那部分概念数据库的权力。这些抽象在图 1.2 中说明。

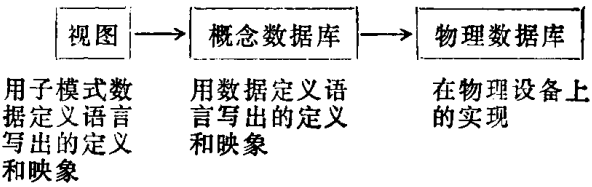

图1.2　抽象链

现在，应用程序员已经可以写询问和操作视图并通过它询问和操作概念与物理数据库的程序了，例如写前面提到的BOOK程序。操作数据库要求一个专门的语言，叫做数据处置语言或查询语言。这种语言可以用来表达如下这样的命令：

1．从数据库中检索出六月二十日第 999 次航班可供预订的座位数。

2．置八月三十一日第 123 次航班可供预订座位数为27。

3．找出八月二十四日由 ORD（芝加哥奥黑尔机场）飞往 JFK（纽约肯尼迪机场）的某个航班。