HZ BOOKS
华章教育

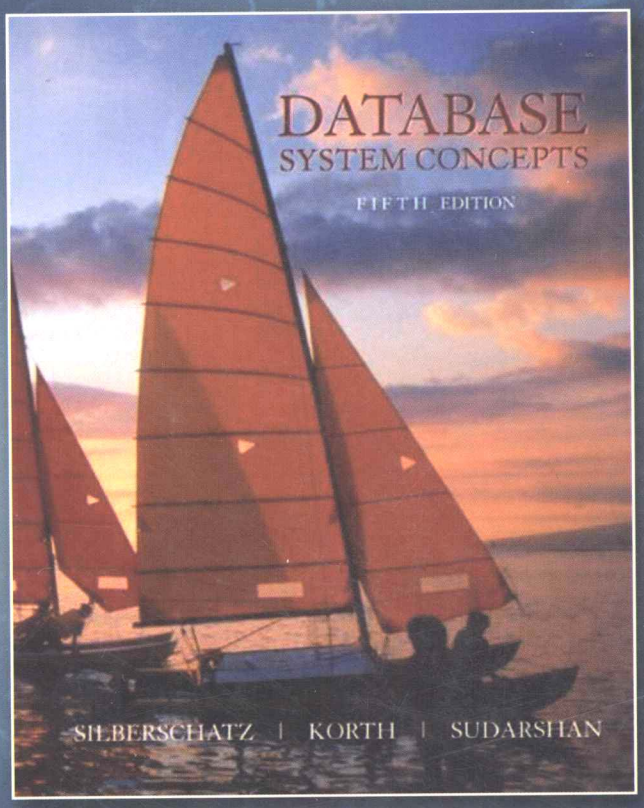
计 算 机 科 学 丛 书

原书第5版

本科教学版

数据库系统概念

Abraham Silberschatz 耶鲁大学
Henry F. Korth 里海大学
S. Sudarshan 印度理工学院 著
杨冬青 马秀莉 唐世渭 北京大学 等译
杨冬青 北京大学 改编



Database System Concepts

Fifth Edition

机械工业出版社
China Machine Press

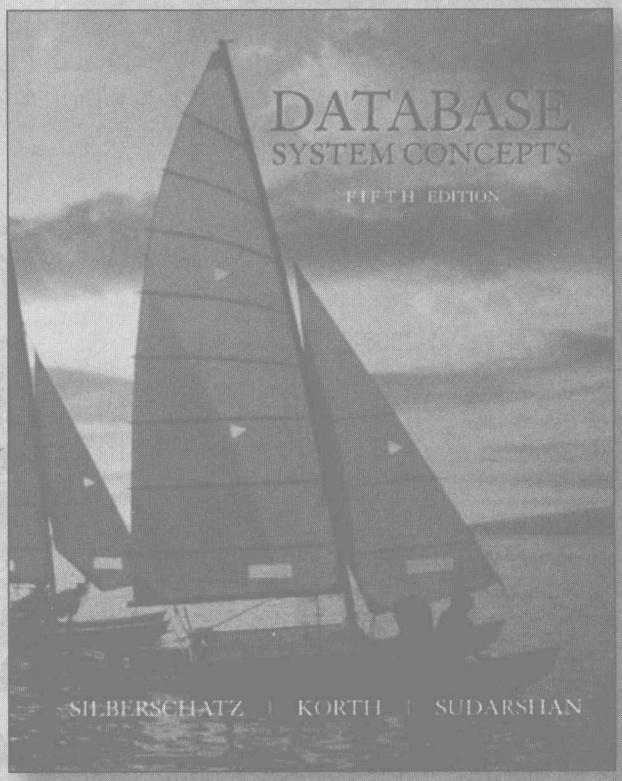
计 算 机 科 学 丛 书

原书第5版

本科教学版

数据库系统概念

Abraham Silberschatz 耶鲁大学 | Henry F. Korth 里海大学 | S. Sudarshan 印度理工学院 著
杨冬青 马秀莉 唐世渭 北京大学 等译
杨冬青 北京大学 改编



Database System Concepts

Fifth Edition

机械工业出版社
China Machine Press

《数据库系统概念》是数据库系统方面的经典教材之一，其内容由浅入深，既包含数据库系统基本概念，又反映数据库技术新进展。国际上许多著名大学包括斯坦福大学、耶鲁大学、康奈尔大学、印度理工学院等都采用本书作为教科书。我国也有多所大学采用本书作为本科生和研究生的数据库课程的教材和主要教学参考书，收到了良好的效果。

本书基于该书第5版进行改编，内容更加精练和实用，体系更加符合国内教学情况，适合作为国内高校计算机及相关专业本科生教材，也可供数据库领域的技术人员参考。

本书版权登记号：图字：01-2008-0337

图书在版编目（CIP）数据

数据库系统概念（原书第5版·本科教学版）/（美）希尔伯沙茨（Silberschatz, A.）等著；杨冬青等译. —北京：机械工业出版社，2008.4

（计算机科学丛书）

书名原文：Database System Concepts, Fifth Edition

ISBN 978-7-111-23422-7

Ⅰ.数…　Ⅱ.①希…　②杨…　Ⅲ.数据库系统—高等学校—教材　Ⅳ.TP311.13

中国版本图书馆 CIP 数据核字（2008）第 016886 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：姚　蕾

北京市慧美印刷有限公司印刷 · 新华书店北京发行所发行

2008 年 4 月第 3 版第 1 次印刷

184mm×260mm · 26 印张

标准书号：ISBN 978-7-111-23422-7

定价：45.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

改编者序

数据库系统是对数据进行存储、管理、处理和维护的软件系统，是现代计算环境中的一个核心成分。随着计算机硬件、软件技术的飞速发展和计算机系统在各行各业的广泛应用，数据库技术的发展尤其迅速，引人注目。有关数据库系统的理论和技术是计算机科学技术教育中必不可少的部分。《数据库系统概念》是一本经典的、备受赞扬的数据库系统教科书，其内容由浅入深，既包含数据库系统基本概念，又反映数据库技术新进展。它已被国际上许多著名大学所采用，并多次再版。

我们先后将《数据库系统概念》一书的第3版、第4版和第5版译成中文，由机械工业出版社分别于2000年春季、2003年春季和2006年秋季出版发行。国内许多大学采用《数据库系统概念》作为本科生和研究生数据库课程的教材或主要教学参考书，收到了良好的效果。

本书基于《数据库系统概念》第5版进行改编，保留其中的基本内容，压缩或删除了一些高级内容，其目的是使它更适合于本科生的数据库课程教学需要。

本书的前10章是最基本的内容，讲述数据库系统的基本概念，包括对数据库系统的性质和目标的综述，对关系数据模型和关系语言的介绍，对数据库设计过程、关系数据库理论以及数据库应用设计和开发（包括Web应用和安全问题等）的详细讨论。这一部分中还介绍了新型的数据库系统——对象－关系数据库，以及XML数据库设计和查询的相关内容（包括对XML Schema和XQuery的更深入的描述和对SQL/XML标准的介绍）。本书可作为大学本科数据库概论课程的教材或主要参考资料，教师可以重点讲授前10章，并介绍第11至15章中的部分内容。

限于改编者水平，改编中疏漏和错误难免，欢迎批评指正。

杨冬青

2007年于北京大学

前 言

数据库管理已经从一种专门的计算机应用发展为现代计算环境中的一个重要部分，因此，有关数据库系统的知识已成为计算机科学教育中的一个核心的部分。

本书改编自《数据库系统概念》第5版，适合作为本科生三年级或四年级数据库入门课程的教科书。在本书中，我们讲述数据库管理的基本概念，这些概念包括数据库设计、数据库语言、数据库系统实现等多个方面。除了这些作为入门课程的基本内容外，本书还包括一些可作为课程补充性材料或作为高级课程介绍性材料的高级内容。

我们仅要求读者熟悉基本的数据结构、计算机组织结构和一种高级程序设计语言（例如Java、C或Pascal）。书中的概念都以直观的方式加以描述，其中的许多概念都基于银行运营的例子加以阐释。本书中包括重要的理论结果，但省略了形式化证明，取而代之的是用图表和例子来说明为什么结论是正确的。对于形式化描述和研究结果的证明，读者可以参考文献注解中列出的研究论文和高级教材。

本书中所包括的基本概念和算法通常都基于当今的商品化或试验性的数据库系统中采用的概念和算法。我们的目标是在一个通常环境下描述这些概念和算法，没有与某个特定的数据库系统绑定。

在这本《数据库系统概念》第5版的改编版本中，我们保留了原书第5版的基本内容，压缩或删除了一些高级内容，其目的是使得改编版本更适合于本科生的数据库入门课程使用。下面我们简单描述本书内容的组织。

本书的组织

本书组织成五个主要部分：

- **综述**（第1章）。第1章对数据库系统的性质和目标进行了一般性综述。解释了数据库系统的概念是如何发展的，各数据库系统的共同特性是什么，数据库系统能为用户做什么，以及数据库系统如何与操作系统交互。这一章还引入了一个数据库应用的例子：由多个分行组成的一个银行企业。这个例子用作贯穿全书的运行实例。
- **第一部分：关系数据库**（第2章至第5章）。第2章介绍了数据的关系模型，包括基本概念以及关系代数。该章还给出了对完整性约束的简要介绍。第3章和第4章重点介绍最具影响力的面向用户的关系语言——SQL。第3章给出了对SQL的基本介绍；而第4章则描述了SQL的一些更高级的特性，包括编程语言和支持SQL的数据库之间如何交互。第5章介绍其他的一些关系语言，包括关系演算、QBE和Datalog。

  这部分中的章节描述了数据操纵，包括查询、修改、插入和删除（假设已有一个模式设计）。关于模式设计的问题延迟到第二部分讲述。
- **第二部分：数据库设计**（第6章至第8章）。第6章给出了数据库设计过程的概要介绍，主要侧重于用实体－联系数据模型进行数据库设计。实体－联系模型为数据库设计问题以及在数据模型的约束下捕获现实应用的语义时所遇到的问题提供了一个高层视图。

  第7章介绍了关系数据库设计理论。这一章讲述了函数依赖和规范化，重点强调了

提出各种范式的原因，以及它们的直观含义。这一章以关系设计的概览开始，依赖于对函数依赖的逻辑蕴涵的直观理解。在完整描述函数依赖理论之前先介绍了规范化概念，函数依赖理论在第 7 章稍后部分讨论。教师可以只选用 7.1 节至 7.3 节这些较前面的章节，而不会丢失连贯性。不过，完整地讲授这一章将有利于学生较好地理解规范化概念，从而进一步理解函数依赖理论中一些较艰深的概念。

第 8 章讲述了应用设计和开发。这一章侧重于用基于 Web 的界面构建数据库应用。另外，这一章还讲述了应用安全性。

- **第三部分：基于对象的数据库和 XML**（第 9 章和第 10 章）。第 9 章介绍基于对象的数据库。该章讲述了对象 - 关系数据模型，该模型扩展了关系数据模型以支持复杂数据类型、类型继承和对象标识。该章还描述了用面向对象的编程语言来访问数据库。

  第 10 章介绍数据表示的 XML 标准，它正日益广泛地应用于复杂数据交换和存储。这一章还描述了 XML 的查询语言。
- **第四部分：数据存储、查询和事务管理**（第 11 章至第 13 章）。第 11 章简单介绍了物理存储介质，描述了记录是如何映射到文件，然后又如何映射到磁盘中的比特的；并描述了数据库系统使用的几种索引类型。第 12 章描述了如何处理查询，给出了用于实现单独操作的算法，并描述了查询优化过程。第 13 章详细阐述了事务的概念，包括事务的原子性、持久性、隔离性和其他特性；介绍了几种实现隔离性的并发控制技术，并描述了数据库恢复管理部件，它实现了数据库的原子性与持久性。
- **第五部分：高级话题**（第 14 章和第 15 章）。第 14 章介绍了数据仓库的概念并解释了什么是数据挖掘和联机分析处理（OLAP），包括 SQL：1999 中对数据挖掘和数据仓库的支持。第 15 章讨论了性能评测标准、性能调整、标准化和遗产系统应用的转移。

在练习方面，我们保持《数据库系统概念》第 5 版的做法，把习题划分成两部分：实践习题（practice exercise）和习题（exercise）。实践习题的答案在《数据库系统概念》第 5 版的网页（http://www.db-book.com/）可以得到。我们鼓励学生独立解决这些实践习题，然后用网页上的答案来检查自己的答案。其他习题的答案只有讲课教师能得到（参看下面的“Web 主页和教学补充材料”，以获取如何得到答案的信息）。

讲课教师注意事项

本书包括基本内容和一些高级内容，在一个学期内也许不能讲授所有这些内容。我们以符号“**”把某些节标记为高级内容，教师可根据各自学校的实际情况，省略这些节，而仍能保持内容的连续性。较难的习题（可以忽略）同样用符号“**”标记了出来。

本书的前 10 章是最基本的内容，对于入门性课程来说，教师可以重点讲授前 10 章，并介绍第 11 至 15 章中的部分内容。

Web 主页和教学补充材料

《数据库系统概念》的 WWW 主页的网址是：

http://www.db-book.com/

该主页包括以下内容：

- 所有各章的幻灯片
- 实践习题的答案

- 三个附录
- 最新勘误表

下列附加材料仅提供给采用本书作为教材的教师:[⊖]

- 书中所有习题答案的教师手册
- 实验材料
- 模型课程的教学提纲

⊖ 想要得到这些附加教辅资料的教师，请联络本书原版的出版方——麦格劳·希尔教育出版公司（电话：010－62790299转108，800－810－1936；邮件 instructorchina @ mcgraw-hill. com）索要相关资料。——编辑注

目　录

第二部分 数据库设计

第三部分 基于对象的数据库和 XML

第四部分 数据存储、查询和事务管理

第五部分 高级话题

第1章 引 言

数据库管理系统(database-management system，DBMS)由一个互相关联的数据的集合和一组用以访问这些数据的程序组成。这个数据集合通常称作**数据库**(database)，其中包含了关于某个企业的信息。DBMS的主要目标是要提供一个可以方便、高效地存取数据库信息的环境。

设计数据库系统的目的是为了管理大量信息。对数据的管理既涉及信息存储结构的定义，又涉及信息操作机制的提供。此外，数据库系统还必须提供所存储信息的安全性保证，即使在系统崩溃或有人企图越权访问时也应保障信息的安全性。如果数据将被多用户共享，那么系统还必须设法避免可能产生的异常结果。

在大多数组织中信息是非常重要的，因而计算机科学家发展了大量的用于有效管理数据的概念、技术。这些概念和技术正是本书所关注的。在这一章里，我们将简要介绍数据库系统的基本原理。

1.1 数据库系统的应用

数据库的应用非常广泛，以下是一些具有代表性的应用：

- 银行业：用于存储客户的信息、账户、贷款以及银行的交易记录。
- 航空业：用于存储订票和航班的信息。航空业是最先以地理上分布的方式使用数据库的行业之一。
- 大学：用于存储学生的信息以及课程注册和成绩的信息。
- 信用卡交易：用于记录信用卡消费的情况和产生每月清单。
- 电信业：用于存储通话记录，产生每月账单，维护预付电话卡的余额和存储通信网络的信息。
- 金融业：用于存储股票、债券等金融票据的持有、出售和买入的信息；也可用于存储实时的市场数据，以便客户能够进行联机交易，公司能够进行自动交易。
- 销售业：用于存储客户、产品及购买信息。
- 联机的零售商：用于存储以上所述的销售数据，以及实时的订单跟踪，推荐品清单的生成，还有实时的产品评估的维护。
- 制造业：用于管理供应链，跟踪工厂中产品的生产情况、仓库和商店中产品的详细清单以及产品的订单。
- 人力资源：用于存储雇员 工资 所得税和津贴的信息，以及产生工资单。

正如以上所列举的，数据库已经成为当今几乎所有企业不可缺少的组成部分了。

在20世纪最后的40年中，数据库的使用在所有的企业中都有所增长。在早期，很少有人直接和数据库系统打交道，尽管没有意识到这一点，他们还是与数据库间接地打着交道，比如，通过打印的报表(如信用卡的对账单)或者通过代理(如银行的出纳员和机票预订代理等)间接与数据库发生关系。自动取款机的出现，使用户可以直接和数据库进行交互。计算机的电话界面(交互式语音应答系统)也使得用户可以直接和数据库进行交互，访问者可以通过拨号和按电话键来输入信息或选择可选项，来找出如航班的起降时间，或注册学校的课程等。

20世纪90年代末的互联网革命急剧地增加了用户对数据库的直接访问。很多组织将他们访问数据库的电话界面改为Web界面，并提供了大量的在线服务和信息。比如，当你访问一家在

线书店，浏览一本书或一个音乐集时，其实你正在访问存储在某个数据库中的数据。当你确认了一个网上订购，你的订单也就保存在了某个数据库中。当你访问一个银行网站，检索你的账户余额和交易信息时，这些信息也是从银行的数据库系统中取出来的。当你访问一个网站时，关于你的一些信息可能会从某个数据库中取出，并且选择出那些适合显示给你的广告。此外，关于你访问网络的数据也可能会存储在一个数据库中。

因此，尽管用户界面隐藏了访问数据库的细节，大多数人甚至没有意识到他们正在和一个数据库打交道，然而访问数据库已经成为当今几乎每个人生活中不可缺少的组成部分。

也可以从另一个角度来评判数据库系统的重要性。如今，像 Oracle 这样的数据库系统厂商是世界上最大的软件公司之一，并且在微软和 IBM 等这些有多样化产品的公司中，数据库系统也是其产品线的一个重要组成部分。

1.2 数据库系统的目标

数据库系统是作为商业数据计算机化管理的早期方法而产生的。作为 20 世纪 60 年代这类方法的典型实例之一，比如一个银行的某个部门需要保存所有客户及储蓄账户的信息。在计算机上保存这些信息的一种方法是将它们存放在操作系统文件中。为了使用户可以对信息进行操作，系统中应有一些对文件进行操作的应用程序，包括：

- 处理某账户的存款、贷款的程序。
- 创建新账户的程序。
- 查询账户余额的程序。
- 产生每月对账单的程序。

这些应用程序是由系统程序员根据银行的需求编写的。

随着需求的增长，新的应用程序被加入到系统中。例如，一个储蓄银行决定开设支票账户，那么这个银行就要建立新的永久性文件来保存该银行提供的所有支票账户的信息，进而就有可能需要编写新的应用程序来处理在储蓄账户里不曾遇到的问题，例如透支。因此，随着时间的推移，越来越多的文件和应用程序就会加入到系统中。

以上所描述的典型的**文件处理系统**(file-processing system)是传统的操作系统所支持的。永久记录被存储在多个不同的文件中，人们编写不同的应用程序来将记录从有关文件中取出或加入到适当的文件中。在数据库管理系统(DBMS)出现以前，各个组织通常都采用这样的系统来存储信息。

在文件处理系统中存储组织信息的主要弊端包括：

- **数据的冗余和不一致**(data redundancy and inconsistency)。由于文件和程序是在很长的一段时间内由不同的程序员创建的，不同文件可能有不同的结构，不同程序可能采用不同的程序设计语言写成。此外，相同的信息可能在几个地方(文件)重复存储。例如，某个客户的地址和电话号码可能既在由储蓄账户记录组成的文件中出现，又在由支票账户记录组成的文件中出现。这种冗余除了导致存储和访问开销增大外，还可能导致**数据不一致性**(data inconsistency)，即同一数据的不同副本不一致。例如，某个客户地址的更改可能在储蓄账户记录中得到反映而在系统的其他地方却没有。
- **数据访问困难**(difficulty in accessing data)。假设银行的某个高级职员需要找出所有居住在某个特定邮编地区的客户的姓名，于是他要求数据处理部门生成这样的一个列表。由于原始系统的设计者并未预料到会有这样的需求，因此没有现成的应用程序去满足这个需求。但是，系统中却有一个产生所有客户列表的应用程序。这时该职员有两种选择：一种是取得所有客户的列表并从中手工提取所需信息，另一种是要求数据处理部门让某个

系统程序员书写相应的应用程序。这两种方案显然都不太令人满意。假设编写了相应的程序，几天以后这个职员可能又需要将该列表减少到只列出账户余额不少于 10 000 美元的那些客户。可以预见，产生这样一个列表的程序又不存在，这个职员就再次面临前面那两种都不尽如人意的选择。

这里需要指出的是，传统的文件处理环境不支持以一种方便而高效的方式去获取所需数据。我们需要开发通用的、能对变化的需求做出更快反应的数据检索系统。

- **数据孤立**(data isolation)。由于数据分散在不同文件中，这些文件又可能具有不同的格式，编写新应用程序来检索适当数据是很困难的。
- **完整性问题**(integrity problem)。数据库中所存储数据的值必须满足某些特定的**一致性约束**(consistency constraint)。例如，某类银行账户的余额永远不能低于某个预定的值(如 25 美元)。开发者通过在各种不同应用程序中加入适当的代码来增强系统中的这些约束。然而，当新的约束加入时，很难通过修改程序来体现这些新的约束。尤其是当约束涉及不同文件中的多个数据项时，问题就变得更加复杂了。
- **原子性问题**(atomicity problem)。如同别的机械或电子设备一样，计算机系统也会发生故障。一旦故障发生，数据就应被恢复到故障发生以前一致的状态，对很多应用来说，这样的保证是至关重要的。让我们看看把 A 账户的 50 美元转入 B 账户的这样一个程序。假设在程序的执行过程中发生了系统故障，很可能 A 账户上减去的 50 美元还没来得及存入 B 账户，这就造成了数据库状态的不一致。显然，为了保证数据库的一致性，这里的借和贷两个操作必须是要么都发生，要么都不发生。也就是说，转账这个操作必须是*原子性的*——它要么全部发生要么根本不发生。在传统的文件处理系统中，保持原子性是很难做到的。
- **并发访问异常**(concurrent-access anomaly)。为了提高系统的总体性能以及加快响应速度，许多系统允许多个用户同时更新数据。实际上，如今最大的网上零售商每天就可能有来自购买者对其数据的数百万次访问。在这样的环境中，并发的更新操作相互影响，可能导致数据的不一致。设 A 账户中有 500 美元，假如两个客户几乎同时从 A 账户中取款，分别取出 50 美元和 100 美元，这样的并发执行就可能使账户处于一种错误的或者说不一致的状态。假设每个取款操作对应执行的程序是读取原始账户余额，在其上减去取款的金额，然后将结果写回。如果两次取款的程序并行执行，可能它们读到的余额都是 500 美元，并将分别写回 450 美元和 400 美元。账户中到底剩下 450 美元还是 400 美元要视哪个程序后写回结果而定，而实际上这两种结果都是错的，正确的值应该是 350 美元。为了消除这种情况发生的可能性，系统必须进行某种形式的管理。但是，由于数据可能被多个不同的应用程序访问，这些程序相互间事先又没有协调，管理就很难进行。
- **安全性问题**(security problem)。并非数据库系统的所有用户都可以访问所有数据。例如在银行系统中，工资发放人员只需要看到数据库中关于银行员工信息的那部分。他们不需要访问关于客户账户的信息。但是，由于应用程序总是即兴加入到文件处理系统中来，因此这样的安全性约束难以实现。

以上问题以及一些其他问题，促进了 DBMS 的发展。接下来我们将看一看数据库系统为了解决上述在文件处理系统中存在的问题而提出的概念和算法。在本书的大部分篇幅中，我们在讨论企业中典型的数据处理应用时总以银行企业作为实例。

1.3 数据视图

数据库系统是一些互相关联的数据以及一组使得用户可以访问和修改这些数据的程序的集

合。数据库系统的一个主要目的是给用户提供数据的抽象视图，也就是说，系统隐藏关于数据存储和维护的某些细节。

1.3.1 数据抽象

一个可用的系统必须能高效地检索数据。这种高效性的需求促使设计者在数据库中使用复杂的数据结构来表示数据。由于许多数据库系统的用户并未受过计算机专业训练，系统开发人员通过如下几个层次上的抽象来对用户屏蔽复杂性，以简化用户与系统的交互：

- **物理层**(physical level)。最低层次的抽象，描述数据实际上是怎样存储的。物理层详细描述复杂的低层数据结构。
- **逻辑层**(logical level)。比物理层层次稍高的抽象，描述数据库中存储什么数据及这些数据间存在什么关系。这样逻辑层就通过少量相对简单的结构描述了整个数据库。虽然逻辑层的简单结构的实现可能涉及复杂的物理层结构，但逻辑层的用户不必知道这样的复杂性。逻辑层抽象是被数据库管理员所使用的，管理员必须确定数据库中应该保存哪些信息。
- **视图层**(view level)。最高层次的抽象，只描述整个数据库的某个部分。尽管在逻辑层使用了比较简单的结构，但由于一个大型数据库中所存信息的多样性，仍存在一定程度的复杂性。数据库系统的很多用户并不需要关心所有的信息，而只需要访问数据库的一部分。视图层抽象的定义正是为了使这样的用户与系统的交互更简单。系统可以为同一数据库提供多个视图。

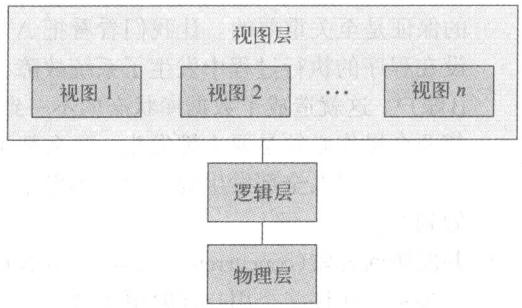


图 1-1　数据抽象的三个层次

这三层抽象的相互关系如图 1-1 所示。

通过与程序设计语言中数据类型的概念进行类比，我们可以弄清各层抽象间的区别。大多数高级程序设计语言支持结构化类型的概念。例如，用一种类似 Pascal 的语言，我们可以定义如下记录：

```
type customer = record
                  customer_id : string;
                  customer_name : string;
                  customer_street : string;
                  customer_city : string;
                end;
```

以上代码定义了具有四个字段的新记录 *customer*。每个字段有一个字段名和所属类型。对一个银行来说，可能包括几个这样的记录类型：

- *account*，包含字段 *account_number* 和 *balance*。
- *employee*，包含字段 *employee_name* 和 *salary*。

在物理层，*customer*、*account* 或 *employee* 记录可能被描述为连续存储单元(如字或字节)组成的存储块。编译器为程序设计人员屏蔽了这一层的细节。与此类似，数据库系统为数据库程序设计人员屏蔽了许多最低层的存储细节。而数据库管理员可能需要了解数据物理组织的某些细节。

在逻辑层，每个这样的记录通过类型定义进行描述，正如前面的代码段所示。在逻辑层上同时还要定义这些记录类型的相互关系。程序设计人员正是在这个抽象层次上使用某种程序设计语言进行工作。与此类似，数据库管理员常常在这个抽象层次上工作。