PEARSON
Prentice Hall

空间数据库

Spatial Databases

A Tour

(美) Shashi Shekhar
Sanjay Chawla 著

谢昆青 马修军 杨冬青 等译

机械工业出版社
China Machine Press

"空间数据库是非常前沿的主题，其他著作只是介绍了其中的部分内容，而本书却详尽而全面地介绍了空间数据库的基本主题。"
——Alia I. Abelmoty，英国格拉摩根大学

"本书结构清晰，叙述严谨，条理性强，内容涵盖从表达到查询再到分析的整个领域，数据挖掘一章尤其精彩。"
——Michael F. Goodchild，NCGIA和加利福尼亚大学地理系

"我刚读完这本出色的空间数据库著作，这本书叙述简洁，内容广泛，是一本非常棒的书。"
——Jim Gray，微软研究院

"这是一部探讨GIS与数据库主题的独具特色的参考书，还没有一本书涉及到如此有价值和重要的领域，主题安排循序渐进，每章都为后面内容奠定坚实的基础。"
——Fred Petry，杜兰大学

"由于强调的是OGIS行业标准而非具体产品，所以本书对了解空间数据库技术发展水平极具价值。"
——Siva Ravada，Oracle公司空间数据产品部经理

这是一本讲述如何在GIS、CAD和多媒体系统应用中管理空间数据的综合性著作。它通过介绍概念模型、查询语言以及高效执行所采用的查询优化算法与空间存储索引方法，来帮助读者掌握空间数据库各个阶段的设计和实现。前沿研究与初级实践在本书中得到很好的结合，无论是学生还是专业人士，都能从本书获益。

本书的特点

- 全面而简洁地介绍空间数据库
- 自成体系，读者无需具备GIS或数据库的知识
- 广泛采用行业标准，如OGIS空间数据类型和操作
- 完全覆盖空间网络的建模、查询和存储方法
- 详细讨论有关空间数据挖掘的主题
- 兼顾前沿研究与商业化趋势
- 每章结尾都提供大量习题
- 包含大量通俗易懂的实例

作者介绍

Shashi Shekhar 明尼苏达大学计算机科学系教授，该校空间数据库研究组的负责人，在加州大学伯克利分校获得博士学位。他由于在空间数据库存储方法、数据挖掘以及地理信息系统等方面的贡献而成为IEEE特别会员。迄今为止发表了大量学术文章并担任多个组织在空间数据库专题方面的学术顾问。

Sanjay Chawla 马萨诸塞州Vignette公司的高级技术顾问，在田纳西大学获得博士学位。

www.PearsonEd.com

ISBN 7-111-13221-1

9 787111 132219

HZ BOOKS

华章图书

网上购书：www.china-pub.com

北京市西城区百万庄南街1号 100037

读者服务热线：(010)68995259，68995264

读者服务信箱：hzedu@hzbook.com

http://www.hzbook.com

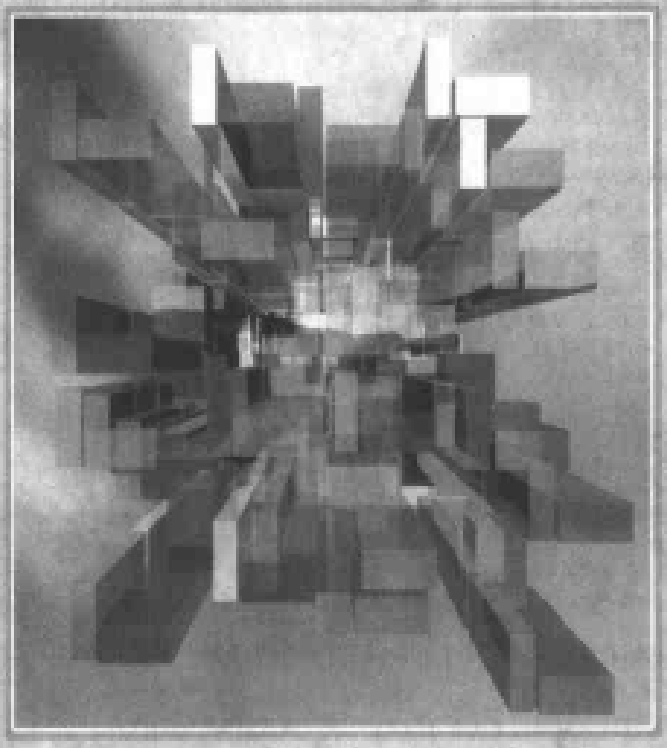
ISBN 7-111-13221-1/TP · 2969

定价：39.00 元

空间数据库

Spatial Databases

A Tour



(美) Shashi Shekhar
Sanjay Chawla 著

谢昆青 马修军 杨冬青 等译

机械工业出版社
China Machine Press

空间数据库是近年的热点研究领域，是一门前沿的交叉学科。本书全面介绍了空间数据库的概念、应用领域、查询语言、空间数据的索引和存储机制、空间查询处理和优化等内容，对空间数据挖掘和空间数据仓库也有精彩的论述。本书条理清晰，叙述严谨，实例丰富，曾得到业内权威人士的赞誉。本书的每章之后都附有习题，帮助读者检验学习效果。本书既适合作为计算机及相关专业的本科生、研究生的教材，也适合IT业的研究人员、技术人员阅读。对于想了解空间数据库的初学者来说，本书也是一本极有价值的参考书。

本书版权登记号：图字：01-2003-4986

图书在版编目（CIP）数据

空间数据库 /（美）沙克哈（Shekhar, S.）等著；谢昆青等译. -北京：机械工业出版社，2004.1

书名原文：Spatial Databases: A Tour

ISBN 7-111-13221-1

Ⅰ.空… Ⅱ.①沙… ②谢… Ⅲ.数据库系统 Ⅳ.TP311.13

中国版本图书馆CIP数据核字（2003）第100371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：朱 劼

北京昌平奔腾印刷厂 印刷 · 新华书店北京发行所发行

2004年1月第1版第1次印刷

787mm × 1092mm 1/16 · 20印张

印数：0 001-4 000册

定价：39.00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

译 者 序

空间信息是指与位置（特别是地理位置）有关的信息，它在信息中占有相当大的比例（有人统计可以达到80%）。然而，空间信息又有其特殊的一面，它具有诸如数据量巨大、结构复杂多样、操作是计算密集型的、具有自相关性等特性。随着IT技术的迅速发展，以GIS为代表的空间信息技术在各领域得到了应用，同时遥感等空间信息获取技术不断进步，现代社会对位置服务和分析决策的需要也日益迫切，因此深入研究和掌握空间信息技术的理论与方法的重要性也日益凸显出来。

本书深入浅出地介绍了空间数据库的相关内容和知识。作者以清晰、有条理和严谨的方式，系统地阐述空间数据库的表达、查询和分析等诸多方面．举例通俗恰当，各章都附有针对性练习。本书对空间数据挖掘和空间数据仓库等当前热点研究的精彩描述，得到了Benjamin Wah教授、Michael F. Goodchild教授以及微软的Jim Gray等业内权威人士的高度评价。空间数据库方面的专著可谓凤毛麟角，许多著作只是涉及其中的部分内容，而本书却系统全面地论述这个主题，并采用业界标准，兼顾前沿研究、商业化趋势以及初学者实践几个方面，是一本不可多得的教材。

本书的内容安排如下：

- 第1章介绍空间数据库学科的应用领域和背景知识。
- 第2章介绍空间数据模型。
- 第3章介绍传统查询语言的空间扩展，并对扩展SQL的空间特性进行讨论。
- 第4章论述空间数据存储和索引机制以及压缩方法，探讨提高查询处理性能的途径。
- 第5章介绍空间查询处理和优化的方法，包括过滤-精炼策略。
- 第6章是空间数据库技术在空间网络方面的应用，包括网络数据模型和查询语言。
- 第7章介绍空间数据挖掘，包括空间依赖概念及其建模方法。
- 第8章讨论栅格数据库、基于内容的检索方法以及空间数据仓库的发展趋势。

本书由谢昆青、马修军和杨冬青组织并参加翻译和审校。参加翻译工作的还有陈冠华、吴杰、张远志、张敏、雷小锋、马秀莉、张惠彬、应莺、张月祥、孙岩峰、徐丹、刘晨、韩亮、蔡颖琨和袁杰。

由于翻译水平有限，书中难免有疏漏和错误之处，欢迎读者批评指正。

译 者

序 1

早在三万五千年前，克鲁马努猎人就在法国拉斯科附近的洞穴岩壁上绘制行走轨迹图形和有意义的刻符来描绘迁徙路线。从那时起，人们就使用带有地理信息的图形了。今天，地理信息系统（GIS）得到广泛的应用：从跟踪北美驯鹿和北极熊的迁徙路线，确定石油开采对野生动物的影响，协助农民最低限度地施用杀虫剂，帮助公司供应部门的经理预测配送仓库的最佳设立位置，直到建立降水和航空照片信息与湿地在每年特定季节变干之间的联系。

从最严格的意义上讲，地理信息系统是一个计算机系统，用于汇集、存储、操纵和显示与位置相关的数据。然而，现代地理信息系统通常要从多个不同来源接收各种形式的数据，以便处理查询和帮助分析信息。从广义上讲，地理信息系统不仅能将地理信息转换和存储为数字形式进行分析，也必须能在空间数据库中进行收集、变换、聚集、索引、链接和挖掘。现代地理信息系统能够集成用其他方法很难关联起来的信息，同时可以结合地图化的变量来构建和分析新变量。

Shekhar和Chawla正是从这种高度完成了一项展示地理信息处理的原理和趋势的非凡工作。本书是讲述概念与方法的力作，循序渐进地讲解模型、语言和算法，层次清晰，结构合理。作者不仅解释概念，而且使用大量例子加以说明。本书强调了将空间数据集成到传统数据库中的很多重要主题，从深奥的空间建模本体论问题到重要的文件管理问题无不涉及。每一章后面都有许多引人思索的习题，帮助读者更好地理解书中的概念和算法。本书最后所展示的空间数据挖掘和空间数据库未来发展趋势尤为精彩，有助于读者了解新兴的研究领域。

本书适合作为地理信息系统方面交叉学科的课程教材，也可以作为该领域从业人员的参考书。即使没有受过正规数据库方面的培训，读者也可以轻松地理解和应用从书中学到的概念和算法。其他学科背景的人也能从本书所讲述的技术中获益，使这些技术得以广泛地应用于政府、商业和工业等部门。

本书是空间数据库领域的第一本著作，作为本书的读者，我确信，从这个令人兴奋和有重要意义的领域中所学到的一切将使你获益匪浅。

Benjamin Wah
伊利诺斯大学电子与计算机工程系教授
IEEE计算机协会主席

序　2

空间信息，也就是在某个空间框架（例如地球表面）中对象的位置信息，长期以来被视为特殊的计算问题。早在1972年，空间数据处理（spatial data handling）这一术语就开始使用，它是指一群研究人员的研究活动。这些研究人员承诺共同开发电子化的数据处理方法，用以提高诸如地图编撰、地图测量和空间数据分析等领域的生产率。在高层结构中经常会用到空间信息。尽管在20世纪60年代就开始出现的每一个经典数据库管理模型都对空间应用领域有所考虑，但不管是关系模型还是面向对象的建模都不能完全适合这一领域。关系模型能够较好地处理拓扑关系，但对表示横跨空间区域的复杂层次关系却无能为力；而面向对象模型能够处理拓扑和层次关系，但难以处理空间中重要的连续性现象。

阐述空间数据库这一内容纷繁的领域的书籍可谓凤毛麟角，因此本书一定会大受欢迎。它涵盖了整个空间数据库领域，以清晰的概念、有条理的结构和严谨的表述，介绍了从表现、查询到分析的各个方面。数据挖掘一章尤其受人欢迎，它不仅讲述传统的空间数据分析方法，而且介绍近几年来发展的许多新技术。这些技术利用当今的高速计算能力，能够在超大型空间数据库中自动搜索异常情况和模式。本书是为计算机系的学生编写的，但同样适用于那些具有其他学科背景又希望学到比一般地理信息系统教材更严谨和更基础的方法的学生。

空间数据库的重要性日益增加，其中一个原因在于其应用范围已超出传统GIS领域。位置和时间是鉴别和刻画信息的强有力方法，因为许多数据集都具有空间和时间的“印记”。地图和地球照片显然如此，而许多报告、书籍、照片以及其他类型的信息亦不例外。因此，位置信息成为在分布的信息源（如因特网）中搜索相关信息的强大基础。人们逐渐认识到，空间（和时间）提供了集成信息的重要方法，这些信息已经远远超出了传统的空间数据库和GIS领域。本书在最后一章探讨了其中一些问题，并对许多人深信空间数据库的重要地位在未来几年会快速提升的原因提出了自己的见解。

Michael F. Goodchild
美国国家地理信息及分析中心，加州大学地理系

前　言

近年来，许多计算机应用领域通过扩充数据库管理系统的功能来支持与空间相关的数据。空间数据库管理系统（spatial database management system, SDBMS）研究是找到有效处理空间数据的模型和算法的重要步骤。

经过20多年的发展，空间数据库已成为一个热点研究领域，其研究成果（如空间多维索引）开始应用于许多不同领域。正是已有应用的需求推动了空间数据库管理系统的研究，这些应用包括地理信息系统（geographical information system, GIS）和计算机辅助设计（computer-aided design, CAD），以及诸如多媒体信息系统、数据仓库、美国国家航空航天局（national aeronautics and space administration, NASA）的地球观测系统等潜在应用。这些空间应用拥有上百万的使用者。

商业数据库的主要厂商已推出专门处理空间数据的产品，其中包括ESRI开发的空间数据引擎（spatial data engine, SDE），以及Intergraph、Autodesk、Oracle、IBM和Informix等公司在对象-关系数据库服务器上开发的空间数据插件，研究的原型系统有Postgres、Geo2和Paradise。这些系统都提供一组空间数据类型（如点、线和多边形）和一组空间操作功能（求交（intersection）、闭合（enclosure）和距离（distance））。开放地理信息系统（Open Geographic Information System, OGIS）协会制定出一套空间数据类型和空间操作的现行标准，使得空间类型和操作可以像SQL3那样成为对象-关系查询语言中的一部分。为了增强性能，这些系统还为空间存取方法、空间范围查询以及空间连接提供了多维空间索引和算法。

把空间数据集成到传统数据库中意味着要在不同层次上解决许多重要问题，这些问题的范围广泛，从关于空间建模的深奥本体论问题（例如，“它应该是基于场的还是基于对象的”，很像物理学中的波粒二象性）到文件管理这类平凡但重要的问题。这些不同课题使空间数据库管理系统研究实际变为多学科的问题。

我们用一个国家数据集的例子来说明空间数据库的特别需求。一个国家至少有一个非空间的数据（国名）和一个空间数据（国界）。国名的存储或表示不会产生任何问题，但国界的存储或表示就不那么简单了。假定用一个直线段的集合表示国界，这时会要求数据库系统能支持空间数据类型“线”、“点”和“面”，以便对“国家”这个对象进行空间查询。操作和组合这些新的数据类型需要遵从某些固定的规则，于是空间代数便应运而生了。由于空间数据具有可视性和数据量庞大的性质，所以必须扩展数据库系统以提供可视化查询处理和特殊的空间索引。数据库的其他重要问题（如并发控制、批量加载、存储和安全机制等）也都必须重新加以考虑和调整，以便构建高效的空间数据库管理系统。

本书以明尼苏达大学的科学数据库（Csci 8705）这门研究生课程的讲义为蓝本。计算机科学系及其他系的研究人员与学生都认为该课程非常有用并能应用到他们的工作中。尽管这样的科目在学生中得到很好的反响并引起他们极高的兴趣，但市面上却找不到一本教材能满足听课

者对多学科的需要。最近，[Scholl等，2001]写的一本书主要介绍了与查询语言和访问方法有关的传统主题，并没有涉及像空间网络（比如，道路地图）和空间模式的数据挖掘这类流行的主题。Adam和Gangopadhyay在1997年编纂了一本GIS领域中与数据库相关的问题的专著，但是没有提到业界的最新技术。另一本关注GIS的书[Worboys著，1995]也只有两章讨论了数据库问题。这类书多数没有采用像OGIS这样的行业标准，也缺乏适当的辅导，例如在每一章节后面提供习题和讨论问题，帮助学生理解主要概念。毫无疑问，从事数据库、并行计算、多媒体信息、土木机械工程和林业方面的学术研究的人员迫切需要一本详尽叙述空间数据库的教材。一项调查表明，业界的专业人员，包括GIS和CAD/CAM软件的开发者，也希望了解有关空间数据库的知识。

在开始撰写本书之前，我们完成了一份“空间数据库：成就和研究需求”（Spatial Databases: Accomplishments and Research Needs）的调研报告，发表在 IEEE Transactions on Knowledge and Data Engineering (1999年1月)上。我们发现计算机科学方面的文献多侧重于研究某些特定问题（例如空间索引、空间连接算法），而很少涉及其他许多重要问题（例如空间数据的概念建模）。在本书有关这些问题的章节中，我们参考了来自GIS业界和非计算机科学专业的GIS研究人员的很多思想和观念。

从1998年的假期开始，我们着手编写本书。通过充实原有的课程讲义，完成了许多章的初稿。本书的初稿在明尼苏达大学的数据库课程中使用。随后，利用审校人、同事和学生的反馈，对本书进行了修订。

本书的特点如下：

- 本书旨在提供空间数据库管理系统的全面概述。它不仅涵盖传统主题（如查询语言、索引和查询处理），还包括许多目前流行的问题（如空间网络和空间数据挖掘）。
- 每章都提供一组习题，读者可以测试自己对核心概念的理解，将这些概念应用于新的领域，甚至举一反三地应用该章内容。本书的网站还安排附加的教学辅导（例如实验、课程讲义等）。
- 尽量将视野超越GIS。空间数据库管理系统技术在许多领域得到应用，包括多媒体信息、CAD/CAM、天文学、气象学、分子生物学和计算力学。
- 每章都尝试给出对象-关系数据库的框架，这是商业数据库应用的发展趋势。在适当的情况下，该框架允许空间数据库重用关系数据库的功能，同时根据需要来扩展关系数据库的功能。
- 采用符合标准（例如OGIS）的空间数据类型和操作来说明常见的空间数据库查询。这些遵从标准的类型和操作可以兼容像SQL-3这样的对象关系查询语言。
- 本书内容自成一体，读者无需具备GIS或数据库的知识。
- 完全覆盖空间网络的建模、查询和存储方法等各个方面。
- 详尽讨论有关空间数据挖掘的问题。
- 既有前沿研究的讨论，也有商业趋势的讨论。
- 包含很多易于理解的常识性应用领域的例子。

本书组织方式

本书共有8章，每一章介绍空间数据库的一个重要部分。第1章介绍空间数据库的基本概念。第2章重点介绍空间数据模型，引入场和对象的二分概念，以及它们在数据库设计中的应用。第3章探讨如何扩展传统查询语言来支持空间数据库，广泛讨论各种扩展SQL空间能力的提议。第4章描述空间数据存储和索引方案。由于空间数据库处理的是海量数据，所以提供合理的存储、压缩和索引方法来提高查询处理的性能，对于数据库管理系统来说至关重要。第5章从查询语言和索引入手，继续讨论查询处理和优化。此时读者会发现，要放弃或从根本上修改传统数据库的许多标准技术才能使之应用于空间问题。该章还引入空间查询处理中的过滤-精炼策略。第6章说明如何将空间数据库技术应用于空间网络，该章还将介绍网络数据模型和查询语言。第7章全面涵盖新兴的空间数据挖掘领域。在这一章我们为读者揭示出空间数据集合普遍存在的空间依赖性，以及如何对其建模并整合到数据挖掘过程中。第8章讨论空间数据库的发展趋势。

致谢

在本书的编写过程中，我们得到了许多人的帮助，在此向他们深表谢意。没有明尼苏达大学计算机科学系的Vipin Kumar教授和ESRI的主席Jack Dangermond博士的鼓励，本书是不会面世的。本书从ESRI的研究人员和产品中获益颇多。我们也要感谢Oracle公司的Siva Ravada博士和Illustra公司的Robert Uleman博士帮助我们理解他们的空间插件。感谢Alan Apt及其Prentice Hall出版社的优秀员工，他们在本书撰写过程中不断给予我们帮助和鼓励。一些未留姓名的评论者也对本书提出了宝贵的意见，这里一并表示感谢。

我们对空间数据库的研究还得到了很多组织的支持，这些组织包括联合国开发计划署（United Nations Development Programme, UNDP）、美国国家科学基金会（National Science Foundation, NSF）、美国国家航空航天局（National Aeronautics and Space Agency, NASA）、美国陆军研究实验室（Army Research Laboratories）、美国农业部（U.S. Department of Agriculture）、美国联邦公路管理局（Federal Highway Administration, FHA）、美国运输部（U.S. Department of Transportation）、明尼苏达州运输部（Minnesota Department of Transportation）、城市与区域事务中心（Center for Urban and Regional Affairs）以及Computing Devices International。其中许多研究项目总结为调研成果文献，并据此开发出用于空间数据库相关问题的技术。

特别要感谢明尼苏达大学计算机科学系空间数据库研究组的成员。他们从各个方面为本书的出版做出了贡献，包括查找文献、设计样例和图表、对各种方法提出见解，以及设计恰当的问题表述和有新意的解决方法。我们非常感谢Vatsavai Ranga Raju，他仔细地审阅并多次修改了本书的早期版本。还要感谢学习课程Csci 8701和Csci 8705的同学试用本书并为本书的修订提出有益的建议。

多年来与其他许多人士的探讨也使我们获益良多。他们是Marvin Bauer、Yvan Bedard、Paul Bolstad、Nick Bourbakis、Thomas Burk、John Carlis、Jai Chakrapani、Vladimir

Cherkassky、Douglas Chub、William Craig、Max Donath、Phil Emmerman、Max Egenhofer、Michael Goodchild、Ralf Hartmut Gueting、Oliver Gunther、John Gurney、Jia-Wei Han、Ravi Janardan、George Karypis、Hans-Peter Kriegel、Robert McMaster、Robert Pierre、Shamkant Navathe、Raymond Ng、Hanan Samet、Paul Schrater、Jaideep Srivastava、Benjamin Wah、Kyu-Young Whang和Micheal Worboys。

图 目 录

表 目 录

目　录

第1章
空间数据库简介

1.1 概述

我们正处在一个信息变革的时代。数据，作为推动这场巨大变革的原材料，不是在地表之下经过千百万年形成并被发现的，而是通过传感器和其他数据采集设备源源不断地收集起来的。例如，NASA的对地观测系统（earth observing system，EOS）每天都要产生1TB的数据。

卫星图像是空间数据的典型例子。为了从卫星图像中提取信息，数据处理必须在一个空间参考框架下进行，这个空间参考框架可以是地球表面。然而，卫星并不是空间数据的唯一来源，地球表面也不是可参照的唯一框架。一块芯片也常常作为一种参照框架。在医疗成像中，人体就可以视为空间参照框架。事实上，连一笔超市交易都可以作为空间数据的例子，只要这笔交易包含了诸如邮政编码这样的信息。对空间数据的查询或者命令称为空间查询。例如，“列出拥有超过万种图书的书店

名称”是一个非空间查询，而“列出明尼阿波利斯市方圆10英里以内的书店的名字”则是一个空间查询。本书将详细介绍存储、管理和检索空间数据的有效技术。

当前，数据通过数据库管理系统（database management system，DBMS）进行存储和管理。数据库及其管理软件是信息时代的成功案例，它们已渐渐渗透到我们日常生活的各个方面。如果没有它们，现代社会将会止步不前。然而，尽管它们取得了令人瞩目的成功，但业内专业人士普遍认为，大部分现有的DBMS无法管理空间数据，或者在管理空间数据的时候难以使用。为什么会这样呢？DBMS通常是作为存放商业和财会数据的简单而有效的仓库。关于雇员、供应商、顾客和产品的信息可以利用DBMS安全地进行存储并有效地进行检索。可能的查询集合是十分有限的，数据库则为了高效地回答这些查询而进行组织。因此，可以顺利地将DBMS从商务领域移植到政府机构和学术管理部门。

驻存在这些庞大数据库中的数据比较简单，通常包括数字、姓名、地址、产品描述等信息。这些DBMS能胜任那类为其量身设计的任务。比如，像“就销售额而言，列出1998年前十位顾客”这样的一个查询，DBMS可以很快回答出来，即使它需要扫描一个很大的顾客数据库也是如此。这样的命令通常称为“查询”，虽然它们不是某种“询问”。数据库无需扫描所有的顾客，它会利用索引来缩小搜索范围，就像我们查阅图书时所做的那样。而另一方面，“列出居住在离公司总部50英里以内的顾客”这样一个相对简单的查询则会难住数据库。要处理这个查询，数据库就必须把公司总部和顾客的地址变换到一个能够计算和比较距离的适当参照系统中，可能是经纬度坐标系。然后，数据库扫描整个顾客列表，计算顾客住所和公司之间的距离。如果距离小于50英里，保存该顾客的名字。这个过程无法利用索引来缩小搜索范围，因为传统的索引无法处理多维坐标数据的排序问题。这样，一条简单、合理的业务查询就会将DBMS置于无望的尴尬境地。因此，迫切需要能处理空间数据和空间查询的数据库。

1.2 空间数据管理的适用人群

各个行业的专业人员都有可能遇到空间数据的管理和分析问题。下面列出了可能用到空间数据管理的不同类型的专业人员，以及一个与他们工作相关的空间查询的例子。

移动电话用户 最近的加油站在哪儿？我回家的路上有无宠物食品店？

军队作战指挥官 敌军从昨晚起是否有明显的调动迹象？

保险公司风险经理 密西西比河沿岸的哪些房屋最有可能受下次洪水的影响？

医师 根据这名患者的核磁共振图像（MRI），我们是否医治过有类似病症的病人？

分子生物学家 基因组中氨基酸合成基因的这种拓扑结构能从数据库的其他序列特征图中找到吗？

天文学家 找出在类星体二弧分范围内的所有蓝星系。

气象学家 怎样才能测试和检验新研究出的全球变暖模型？

药剂研究者 哪些分子能与给定分子进行基于几何形状的对接？

运动员 棒球场中哪些座位是观看投手和击球手的最佳视角？电视摄像机应该安装在什么位置？

公司供货经理 根据未来顾客购物方式的发展趋势，建造物流仓库和零售店的最佳位置在哪里？

运输专家 应该如何扩充路网才能减少交通阻塞？

城市规划专家 新的城市用地开发是否会导致肥沃的农用土地的减少？

滑雪胜地拥有者 我的地产中，哪座山适合滑雪初学者使用？

农场主 怎样才能把喷洒在农场的杀虫剂用量降到最低？

高尔夫场开发商 考虑到天气因素、国家环保署有关杀虫剂规定、濒危物种保护法令、地价以及邻近地区人口分布等的限制，在哪里修建高尔夫球场可以得到最大利润？

应急服务 求助的人位于什么地方？抵达那里的最佳路线是什么？

1.3 GIS和SDBMS

地理信息系统（geographic information system，GIS）这一技术的出现，激发了人们开发空间数据库管理系统（spatial database management system，SDBMS）的兴趣。GIS提供了便于分析地理数据和将地理数据可视化的机制。地理数据就是以地球表面作为基本参照框架的空间数据。GIS提供了一套丰富的分析功能，它们可以对地理数据进行相应的变换。地理学家在GIS中集成了一系列丰富的技术，这也是GIS茁壮成长并在多学

科中应用的内在原因。表1-1列出了常用GIS操作的一小部分实例。

表1-1 常用GIS分析操作列表［Albrecht，1998］

搜索	专题搜索、按区域搜索、（再）分类
定位分析	缓冲区、廊道、叠加（分析）
地形分析	坡度/坡向、流域、排水网系
流分析	连接性、最短路径
分布	变化检测、接近、最近邻居
空间分析/统计	模式、中心、自相关、相似性检索、拓扑：孔描述
度量	距离、周长、形状、相邻、方向

利用GIS可以对某些对象和图层进行多种操作，而利用SDBMS则可以对更多的对象集和图层集进行更为简单的操作。比如，给出了国家的行政边界后，利用GIS可以列出该国家（如法国）的所有邻国。但是GIS在回答集合查询时反应相当迟缓，比如，列出那些邻国最多的国家，或者列出完全被另一国家包围的国家。SDBMS可以回答基于集合的查询，我们将在第3章中讲述。

SDBMS还可以用来处理存储在二级设备（如磁盘、光盘、光盘机等）上的海量空间数据，它使用专门的索引和查询处理技术完成任务。而且，SDBMS继承了传统DBMS所提供的并发控制机制，这一功能可以让多个用户同时访问共享的空间数据，并保持数据一致性。

GIS可以作为SDBMS的前端。在GIS对空间数据进行分析之前，先通过SDBMS访问这些数据。因此，利用一个高效的SDBMS可以大大提高GIS的效率和生产率。

1.4 空间数据库的三类用户

对空间数据进行合理的利用和管理可以提高生产率，这已成为科学家、管理人员以及商务专家的共识。同样重要的是，与其他形式的数据相比，空间的（spatial）或者地理的（geographic）或者地理空间的（geospatial）数据需要区别对待，这种概念已经逐渐渗透到所有主要的数据库厂商的考虑和计划之中。结果，在市场上出现了专门的空间产品，用来提高通用DBMS的空间处理能力。例如，Oracle、Informix和IBM就分别推出了空间附件，并冠以暗盒（cartridge）、数据刀片（datablade）这类隐喻性的名称，或者采用空间选项（spatial option）之类的温和叫法。

从主要数据库厂商的观点来看，管理空间数据需要专门的产品，但空间数据显然不是商务中使用的唯一数据类型。其实，空间数据并非仅有的特殊数据类型。比如，除了空间附件外，数据库厂商还发布了用于时序的（temporal）、可视的（visual）以及其他多媒体形式数据的附件。

另一方面，GIS厂商所定位的客户群体是那些只关注于空间数据分析的用户。这块特定的市场相对较小，其中包括科学界和政府部门的专家。与其他信息技术的用户相比，GIS用户更多是在封闭的环境中工作，使用特别为他们设计的专用数据库。为了管理数量不断增长的空间（和非空间）数据，并且链接到商业数据库中，GIS厂商推出了诸如ESRI的Spatial Data Engine这种中间件产品。在最近10年里，GIS业界的关注焦点发生了很大的变化，正如图1-1所示：GIS已经从最初作为一种用分层方式表示地理信息的软件系统，发展到关注地图代数和空间操作的地理信息（geographic information，GI）科学的阶段。随着个人计算的异军突起，GIS的焦点重新转移到在PC机上提供地理或空间服务。一个最好的例证是微软的Office 2000，它在传统的电子表格和数据库工具上增加了和空间相关的搜索引擎和地图软件。

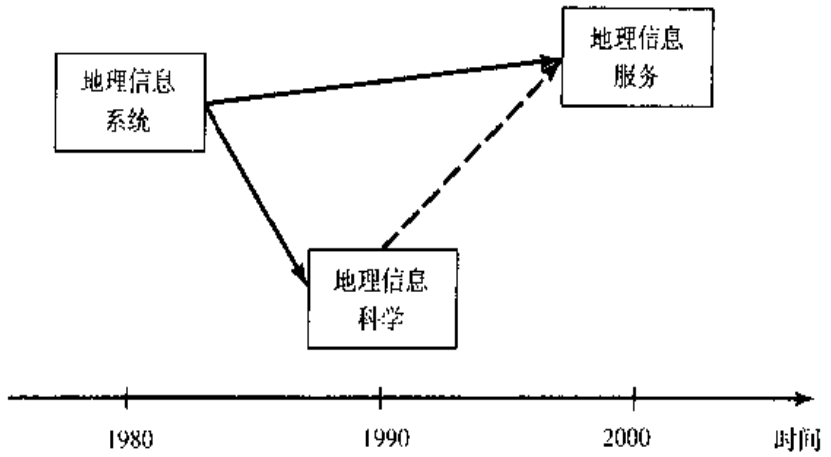


图1-1 GIS这一缩写在过去20年中的演化。在20世纪80年代，GIS表示地理信息系统，在20世纪90年代更流行的提法是地理信息科学（geographic information science），而现在GIS则朝着地理信息服务（geographic information service）的方向发展

随着Internet时代的到来，出现了另一批使用空间数据的用户群，但他们更喜欢在一个非常高级的、用户界面非常友好的层次上使用空间数据。比如，Internet上一

种很受欢迎的站点是为访问者提供导向地图。另一类站点提供了和空间数据相关的搜索引擎，这种引擎能够回答像“找出明尼阿波利斯市所有墨西哥餐馆”这样的查询。空间技术另一个有前景的用途是移动电话定位。美国联邦政府已于2001年10月颁布了规定：所有蜂窝电话的位置在67%的使用时间里必须是可追踪的，追踪精度为125米。这样，一方面人们总在评述着Internet革命“消灭”了地理的概念，与此同时，对于空间技术的需求却在不断增长。

近来，个人数字助理（personal digital assistant，PDA）、移动电话和双向寻呼机的普及给新的应用创造了许多机会。这样的应用有流动工作人员和基于位置服务。流动工作人员，顾名思义，他们工作在远程位置，如客户处、分公司或者野外现场。这些工作人员经常要为完成某项任务下载一段所需的数据，在远端使用这段数据，然后在每天工作结束的时候将改动更新（同步地）到主数据库上。这种场景的一个重要方面是：客户端保留有数据，并以离线方式在本地对数据进行操作。基于位置服务的使用是近年来出现的一个重要趋势，这类服务彻底改变了对用户地理位置的依赖。随着全球定位系统（GPS）的应用，可以很容易确定任何一个客户/使用者的精确位置，并根据用户的地理位置提出最佳解决方案。下面是几个基于位置服务的应用例子：定位一个顾客，寻找去加油站路上最近的比萨店，在一座陌生城市的道路图上突出显示旅游者所在位置，位置相关的事务和警报。基于位置服务的影响和重要性促使开放GIS协会（Open GIS Consortium，OGC）提出了开放位置服务（Open Location Service，OpenLS），希望能够将地理空间数据和地理操作的资源集成到位置服务和电信基础设施中去。

1.5 一个SDBMS的应用案例

前面我们用了一个简单的空间查询例子来说明传统数据库的不足之处。现在我们给出一个更具体的例子。图1-2是一幅由Landsat卫星提供的明尼苏达州Ramsey郡的TM（Thematic Mapper）图像。图像上还叠加了人口普查区界（粗黑线）和湿地位置（细黑线）。从这张图像中，可以很容易分辨出湖泊（北部深色的图块）和密西西比河（南部）。Ramsey郡面积大约有156平方英里，包括了圣保罗大部分的城区和郊区。图像由ArcView生成，ArcView是一个很常用的GIS软件。通常，一个空间数据库包括一些图像和矢量图层，矢量图层包括地块、交通、生态分区、土壤等。

图1-2里有四个图层：基本图像、人口普查区界图、湿地分布图，最后是县边界图（白色虚线）。

存储有关人口普查区信息（如它们的名字、地理区域、人口和边界等）的一种自然的方法是在数据库中创建下面的表：

```
create table census_blocks (
                              name          string ,
                              area          float,
                              population    number,
                              boundary      polyline );
```

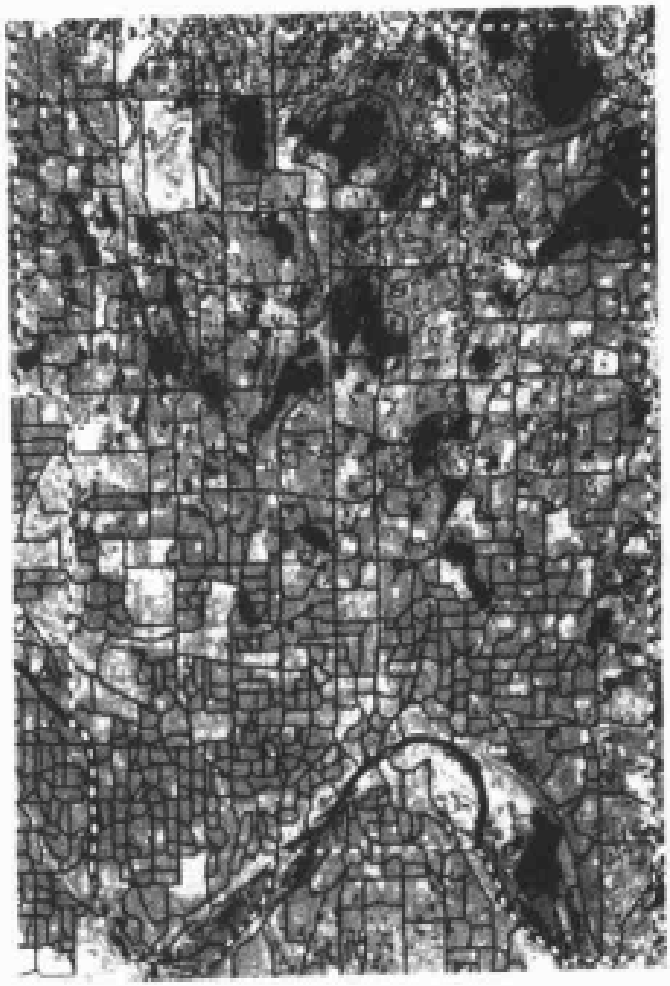


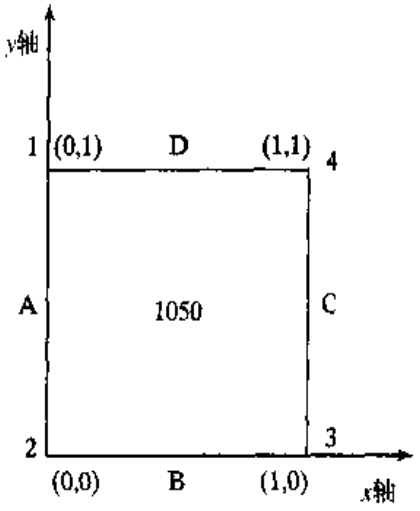
图1-2 由Landsat卫星提供的带空间信息叠加图层的明尼苏达州Ramsey郡的地图

在一个（关系）数据库中，所有有明确标识的对象、实体以及概念都表示为关系或者表。一个关系由一个名字和一组描述该关系特征的属性来定义。该实体的所有实例都作为元组存储在表中。在上面的代码段中，我们创建了一个名为

*census_block*的（关系）表，它有四个属性：name、area、population和boundary。创建表的时候，必须指定属性的类型，本例中属性类型分别是字符串（string）、浮点数（float）、整数（number）和多线（polyline）。多线是一种表示直线序列的数据类型。

图1-3是一个假定的人口普查区，并说明如何在表中存放该区的相关信息。但是，对于传统的关系数据库来说，这样的表是不自然的，因为多线并非内建的数据类型。避开这个难题的一种方法是，创建一个带有交叠（overlapping）属性的表的集合，如图1-4所示。另一种办法是使用存储过程。不过，对于一个新手来说，要实现这些表和过程太过于复杂。其中的关键在于，人口普查区的数据无法自然地映射到关系数据库中。为了缩小空间数据的用户视图与数据库实现之间的语义鸿沟，我们还需要更多的处理空间信息的构件。面向对象的软件方法提供了这方面的功能。

面向对象的软件方法基于用户定义数据类型的原理，它具有继承性和多态性。C++、Java和Visual Basic这些语言的广泛使用，表明软件业中已经牢固树立了面向对象的概念。我们的地块问题看起来似乎是面向对象设计的一个自然应用：声明一个polyline类和另一个land_parcel类。land_parcel类含有两个属性：字符串类型的*address*属性和*polyline*类型的*boundary*属性。我们甚至不需要*area*（面积）属性，因为可以在polyline类中定义一个*area*方法，在需要的时候计算任何地块的面积。这样就解决问题了吗？面向对象数据库（OODBMS）就是答案么？不，这还不够。



Census_blocks

Name	Area	Population	Boundary
1050	1	1839	Polyline((0,0),(0,1),(1,1),(1,0))

图1-3 边界ID为1050的人口普查区

Census_blocks

Name	Area	Population	boundary-ID
340	1	1839	1050

Polygon

boundary-ID	edge-name
1050	A
1050	B
1050	C
1050	D

Edge

edge-name	endpoint
A	1
A	2
B	2
B	3
C	3
C	4
D	4
D	1

Point

endpoint	*x*-coor	*y*-coor
1	0	1
2	0	0
3	1	0
4	1	1

图1-4 在关系数据库中，容纳多线数据类型所需的4个带重叠属性的表

数据库业界中关于关系与面向对象之间的争论就如同GIS业界中关于矢量与栅格之争论。毫无疑问，抽象数据类型（abstract data type，ADT）的引入增加了DBMS的灵活性，但是，在ADT能完全集成到DBMS之前，首先要解决两个对数据库的特殊制约：

- 尽管OODBMS产品已经面世多年，然而市场对此类产品的接受能力却很有限。这就减少了调整OODBMS产品性能所需的经济和工程耗费。其结果是，许多GIS用户将使用其他的系统而不是OODBMS来管理空间数据。
- SQL是数据库世界的“国际通用语言”，它与关系数据库模型紧密地联系在一起。SQL是一种声明性语言，即用户只需要描述所希望得到的结果，而不用关心产生结果的方法。例如，查询“找出所有与MY_HOUSE相邻的地块”用SQL可以表达为：

```
SELECT   M.address
FROM     land_parcel L, M
WHERE    Adjacent(L,M) AND
         L.address = 'MYHOUSE'
```

DBMS的任务就是实现查询语句中指定的操作。特别是，函数*Adjacent(L，M)*应该可以在SQL内调用。通常采用的SQL-92标准支持用户定义的函数，而下一个修订版本SQL-3/SQL 1999支持ADT和更多的数据结构，如列表、集合、数组和包。集成了ADT和其他面向对象设计原则的关系数据库称为对象关系数据库管理系统（OR-DBMS）。数据库技术的演化历史如图1-5所示。

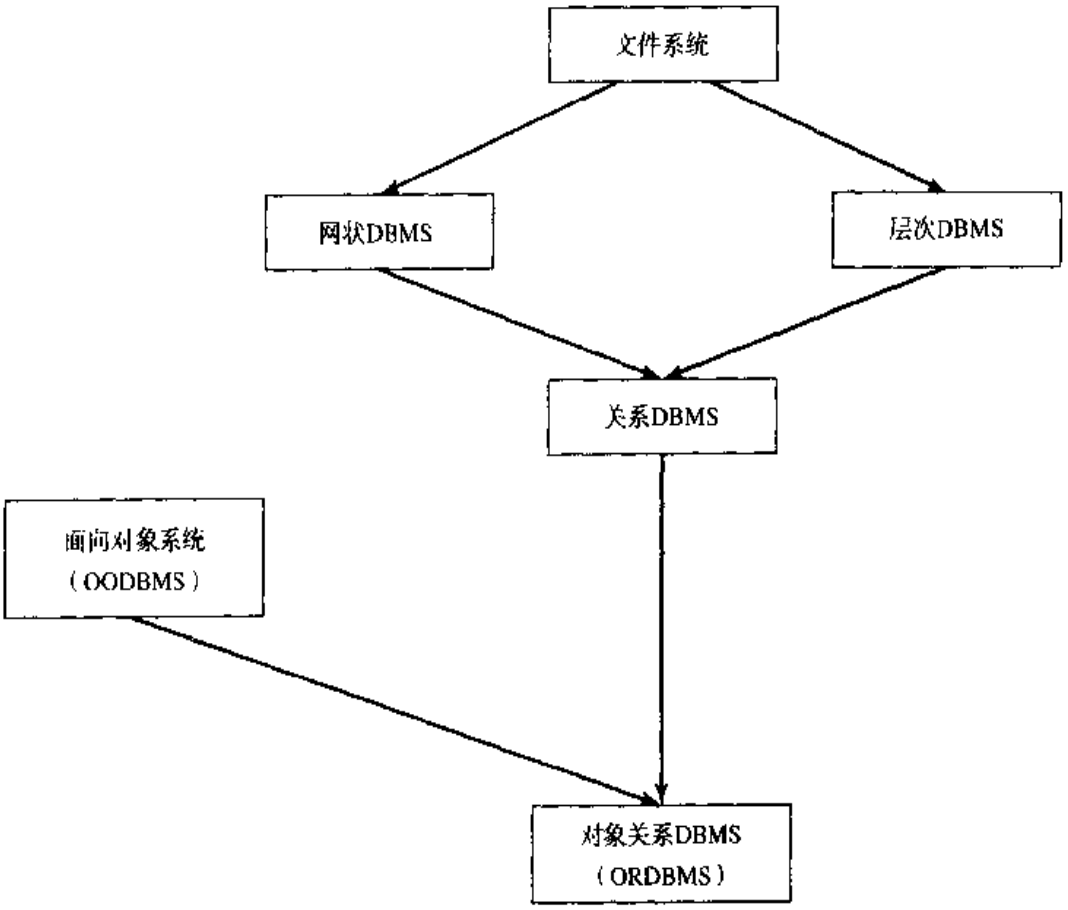


图1-5 数据库的演化［Khoshafian and Baker，1998］

当前这一代OR-DBMS提供了构建ADT的模块化方法。一个ADT可以嵌入到系统中，也可以从系统里删除，而不会影响系统的其他部分。虽然这种“插件”方法为DBMS带来了更强的功能，但用于操作优化的内建支持却少得可怜。我们重点研究满足空间数据需求的OR-DBMS。这样，就可以应用空间领域知识来改进系统的总体效率。现在我们可以给出一个SDBMS的定义，来设定本书讨论范围。

1）一个SDBMS是一个软件模块，它利用一个底层数据库管理系统（如OR-DBMS、OODBMS）。

2) SDBMS支持多种空间数据模型、相应的空间抽象数据类型（ADT）以及一种能够调用这些ADT的查询语言。

3) SDBMS支持空间索引、高效的空间操作算法以及用于查询优化的特定领域规则。

图1-6给出了在OR-DBMS上搭建的SDBMS的体系结构示意图。这是一个三层体系结构。（从左至右）顶层为空间应用，如GIS、MMIS（多媒体信息系统），或者CAD（计算机辅助设计）。该应用层并不直接与OR-DBMS打交道，而需要经过一个中间层与OR-DBMS交互，我们将这个中间层称为空间数据库（SDB）。中间层是封装大多数空间领域知识的地方，并被“插”入到OR-DBMS中。由此对于称为空间数据刀片（Spatial Data Blade、Illustra）、空间数据暗盒（Spatial Data Cartridge、Oracle）以及空间数据引擎（Spatial Data Engine、ESRI）的商业OR-DBMS产品也就不足为奇了。

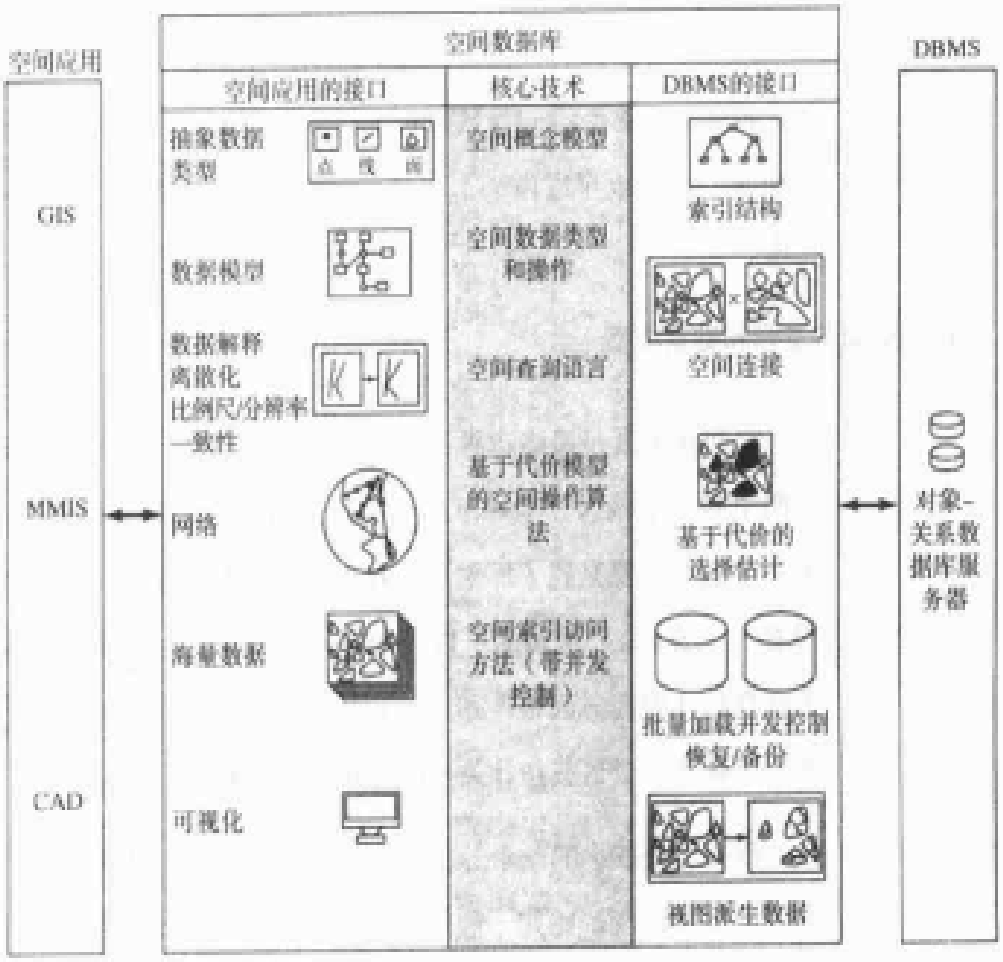


图1-6 三层体系结构

在本小节即将结束的时候，我们来回顾下面的两条特性，这是每个DBMS需要具备的核心特征，但对用户又是不可见的：

1) 持久性　处理临时和永久数据的能力。临时数据在程序结束后就消失了，而永久数据不仅在程序调用时可以使用，并且在系统和媒介崩溃后还能存在。这确保了DBMS在崩溃后可以顺利地恢复数据。在数据库管理系统中，永久对象的状态不断变化，有时会访问前面的数据状态。

2) 事务　事务将数据库从一个一致状态映射到另一个一致状态。这样的映射是原子性的（即要么完全执行要么完全放弃）。通常，许多事务都并发执行，DBMS实施串行化的执行顺序。数据库的一致性通过完整性约束来保证。数据库的所有状态都必须满足这些约束，以确保一致性。此外，为了维护数据库的安全，事务的作用域和使用者的访问权限有关。

1.6 空间数据库概览

在上一节的最后，我们介绍了在OR-DBMS上搭建SDBMS的三层体系结构。我们注意到，大多数空间领域知识都封装在中间层。本节将简要描述中间层的某些核心功能。本节也可以作为本书其余部分的主题的概述，或者是关于SDBMS的快速指南。

1.6.1 空间分类学和数据模型

空间其实就是最终边界，这并非仅就旅行而言，而且无法用一个简洁的描述来说明空间这个概念。考虑下面这句随处可闻的倒霉司机口头禅："我想不起杰克的家有多远了。要是到了那附近，我也许能想起他家在街道的哪一侧，但我肯定它紧挨着一个公园。"这句话可以简单说明我们大脑（思维）是如何建立地理空间的。大脑在估计距离方面很差劲，在记住方向和方位时稍强一点，但在（我们）回忆相邻、连接和包含这些拓扑关系的时候能力较强。

拓扑学是数学的一个分支，它专门研究关系，这种关系不会因基本空间的弹性形变而发生改变。比如，假定在一块橡胶垫板上绘制两个矩形，一个矩形包含另一个矩形，或者互相相邻，当拉伸、扭曲或者挤压橡胶板的时候，这两个矩形之间的

原来的关系不会改变！考查我们说话的语言也可以看出大脑组织空间的方式：对物体描述起决定性的是它的形状。这是否就是我们很难接受鲸是哺乳动物以及海马是鱼的原因呢？物体是用名词来描述的，有多少不同的形状，就用多少名词来描述。另一方面，物体之间的空间关系是用介词来描述的，很少涉及关于形状的描述。在“考夫曼协会位于文森特会堂的东南方” 这句话中，建筑物的形状对“东南”这个关系几乎没有任何影响，所以我们可以用简单矩形来表示建筑物而不会影响原有的关系。

空间分类学（space taxonomy）涉及了多种可用来组织空间的描述方法，其中包括拓扑的、网状的、方位的以及欧氏几何的。首先，要根据我们对空间建模感兴趣的原因，来选择合适的空间描述。表1-2给出了适用于每种空间模型的空间操作例子。但是要记住，没有能回答所有查询的通用的空间类型（模型）。

表1-2 不同的空间类型及示例操作

拓扑	相邻
网状	最短路径
方位	以北
欧几里得几何	距离

数据模型是一条或一组用于标识和表示空间参照对象的规则。明尼苏达州是“万湖之地”。该如何表示这些湖？凭直觉的一种直接方式是把每个湖表示成一个二维区域。同样，对一条河流，可以按比例将其表示为一条一维的曲线，一口井可以表示成零维的点。这就是对象（object）模型。对象模型很适合表示有固定形状的空间实体，如湖泊、道路网和城市。这种对象模型是概念化的，可以采用矢量（vector）数据结构将其映射到计算机中。矢量数据结构将区域映射成多边形，线条映射为多线，点映射为点。

场（field）模型通常用于表示连续的或无固定形状的概念，例如温度场或云区。一个场就是一种函数，它将基本参照框架映射到一个属性域上。对于温度来说，最常用的属性域是摄氏和华氏。在计算机中，场模型是用栅格（raster）数据结构来实现的。栅格数据结构把基本空间划分成均匀的网格。由于场值在空间上是自相关的（它们是连续的），所以每个栅格的值一般采用位于这个格子内所有场点的平均值表示。

场的其他常用数据结构还有不规则三角网（triangulated irregular network, TIN）、等高线和点网格（第2章和第8章会介绍更多相关内容）。

1.6.2 查询语言

到目前为止，根据我们的讨论可以很明显地看出，若要使关系查询语言（如SQL2）成为一种自然的空间查询语言，必须对其现有的功能加以扩充。尤其重要的是，SQL要具备内部指定空间ADT属性和方法的能力。业界正努力制定一套标准来扩展SQL，这套标准名为SQL-3，而目前普遍采用的标准是SQL-2。SQL-3支持ADT和其他数据结构。它规定了句法和语义，由厂商自行定制符合其要求的实现。

对于SQL的空间扩展，有一项普遍认可的标准。OGIS（Open GIS）协会（由重要的GIS和数据库厂商主持成立），提出了一套规范，把2D地理空间ADT整合到SQL中。所提出的ADT是基于对象模型的，并且包括了指定拓扑的操作和空间分析操作。我们将在第2章详细描述OGIS标准，并用一个例子来说明如何在SQL中执行空间查询。

1.6.3 查询处理

前面已经提到，数据库用户采用一种声明性查询语言（如SQL）与数据库交互。用户只需指定所希望的结果，而不用指出得到结果所用的算法。为了高效地执行这个查询，DBMS必须自动地实现一项计划。查询处理是指DBMS着手处理该查询的一系列步骤。

大体上可以把查询划分为两类：单遍扫描查询和多遍扫描查询。在单遍扫描查询中，被查询的表（关系）中的一条记录（元组）最多只被访问一次。因此，就时间而言，最差的情况是访问和处理表中所有记录，验证是否满足查询条件。我们在本章引入的第一个空间查询（“列出明尼阿波利斯市方圆10英里以内所有书店的名字”）就是一个单遍扫描查询。这个查询的结果是以市政府为中心，半径为十英里的圆内的所有的书店。这个查询同时也是一个空间范围（spatial range）查询，范围是指查询的区域。这里的查询区域是一个半径为十英里的圆。如果查询区域是一个矩形，这样的空间范围查询通常就称为窗口（window）查询。

连接查询是多遍扫描查询的原型。为了回答一个连接查询，DBMS必须检索及

合并数据库中的两个表。如果在处理这个查询中需要用到两个以上的表，这些表就可能会被两两处理。两个表基于一个共同的属性进行合并，也就是所谓的“连接”。由于一个表中的一条记录可能与第二个表中的多条记录关联，所以在连接过程中，可能会不止一次访问该条记录。在SDB语境中，当连接属性本质上是空间属性的时候，该查询就被看作空间连接（spatial-join）查询。现在，用一个例子来说明非空间连接和空间连接之间的区别。

图1-7说明了非空间连接和空间连接之间的区别。考虑图中所示的两个关系：SENATOR（参议员）和BUSINESS(公司)。SENATOR关系有四个属性：参议员的名字（NAME），参议员的社会保障号（SOC-SEC），参议员的性别（GENDER），以及参议员所代表的选区（DISTRICT）。选区是个空间属性，由一个多边形表示。另一个关系是BUSINESS，下面列出公司所有的信息：公司名字（B-NAME），它的拥有者（OWNER），拥有者的社会保障号（SOC-SEC），以及公司所在的位置（LOCATION）。位置是个空间属性，用一个位置点表示。考虑下面的查询：“找出拥有该公司的所有女性参议员的名字。”这个例子包括了一个非空间连接，其连接属性是参议员关系的社会保障号和公司所有者的社会保障号。在SQL中，这个查询可以写成下面形式：

```
SELECT   S.name
FROM     Senator S, Business B
WHERE    S.soc-sec = B.soc-sec AND
         S.gender = 'Female'
```

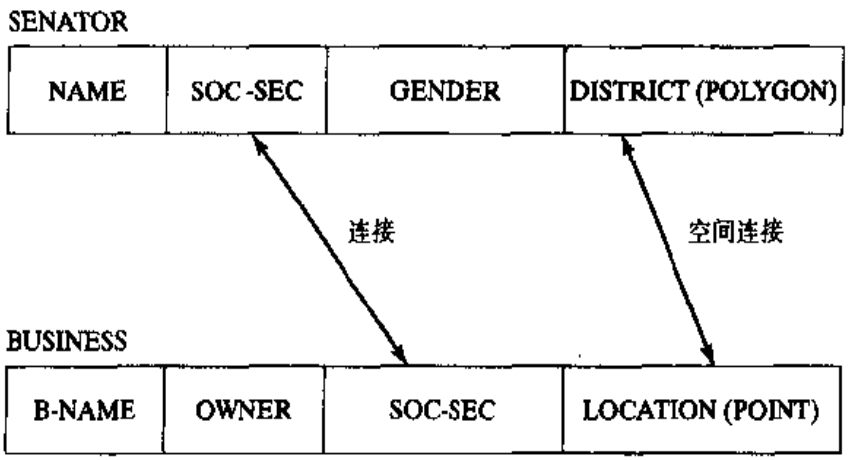


图1-7 用来说明连接与空间连接间的区别的两个关系

现在来考虑另一个查询，“找出其代表的选区面积大于300平方英里并在这个选区中拥有公司的所有参议员。”这个查询包括了一次空间连接操作，连接属性是位置和选区。这样，虽然在非空间连接中连接属性一定是相同的类型，但是在空间连接中连接属性可以是不同的类型。在上面例子中，连接属性是点和多边形。下面用SQL写出的查询：

```
SELECT   S.name
FROM     Senator S, Business B
WHERE    S.district.Area() > 300 AND
         Within(B.location, S.district)
```

SDBMS使用过滤-精炼策略来处理范围查询。这是一个两阶段的处理过程。第一步，将被查询的对象用它们的最小外包矩形（minimum bounding rectangle，MBR）来表示。这样做的理由是，一个查询区域与一个矩形之间的求交计算要比一个查询区域与一个任意形状的不规则空间对象之间求交计算要容易（计算代价小）。如果查询区域是个矩形，那么最多只需四次计算就可以确定两个矩形是否相交。这称之为过滤阶段，因为许多候选者在这一阶段就被剔除了。过滤阶段得到的结果包含了满足原始查询条件的候选者。第二阶段是对过滤的结果使用精确的几何条件进行处理。这是一个计算代价很大的过程，但在过滤阶段的帮助下，本阶段的输入集合只剩下很少的候选者。图1-8展示了一个过滤-精炼策略的例子。

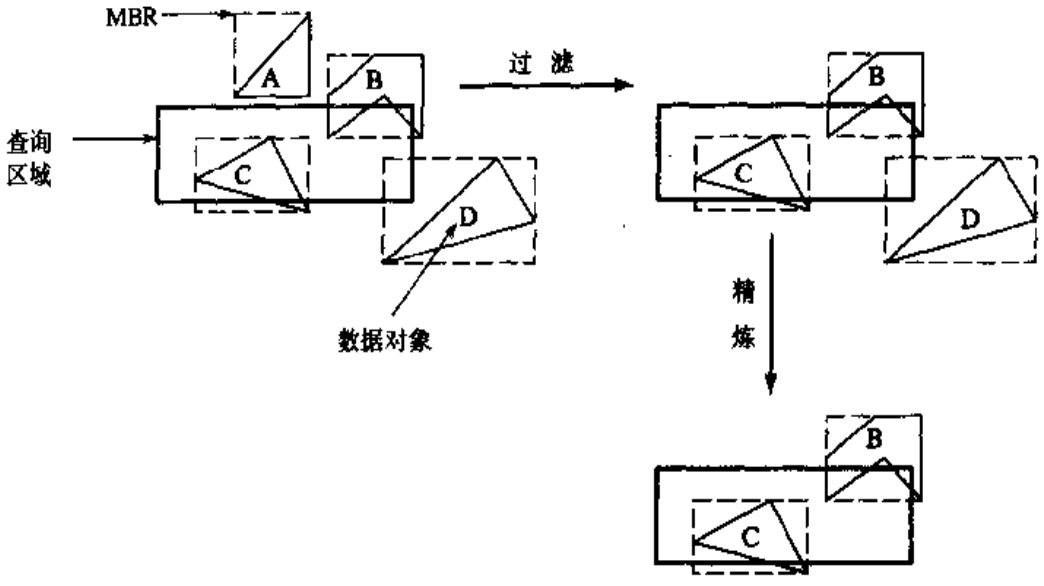


图1-8 用于减少计算时间的过滤-精炼策略

利用上面例子，现在来描述一个处理空间连接查询的过滤阶段的算法。这个算法基于平面扫描（plane sweep）技术，该技术用于计算几何对象之间的交，在计算几何中有广泛应用。

对许多空间连接查询来说，过滤阶段可以简化为确定全部矩形两两相交的问题。考虑两个矩形集合（图1-9b）$R = \{R_1, R_2, R_3, R_4\}$和 $S = \{S_1, S_2, S_3\}$，它们分别代表参与连接的两个表中空间属性的MBR。一个矩形T可以用其左下角（$T.xl$，$T.yl$）以及右上角（$T.xu$，$T.yu$）来确定，如图1-9a所示。我们将R和S中的所有的矩形按它们左下角的x值（即$T.xl$）来排序，经过排序的矩形如图1-9c所示。现在，把集合$R \cup S$中所有R和S的（排序后的）矩形放在一起，并做如下处理：

1) 从左至右移动一条扫描线（例如，垂直于x轴的线），停在$R \cup S$的第一个元素处。这就是具有最小$T.xl$值的矩形T。在我们的例子中（见图1-9a）是矩形$R4$。

2) 搜索S中已排序的矩形，直到抵达第一个矩形S^f，这里有$S^f.xl > T.xu$。显然，对于所有$1 \leqslant j < f$，关系 $[T.xl, T.xu] \cap [S^j.xl, S^j.xu]$ 存在（非空），在本例中，S^f就是S^1。注意上标以图1-9c的数组索引为序，即$S^1 = S_2$、$S^2 = S_1$、$S^3 = S_3$。这样S_2就是一个可能与R_4交叠的候选矩形。这一结论将在下一步确认。

3) 如果对任意$1 \leqslant j \leqslant f$，关系 $[T.yl, T.yu] \cap [S^j.yl, S^j.yu]$ 存在，则S^j与T相交。因此，这一步就确定了R_4与S_2的确是交叠的，并且$<R_4, S_2>$是连接结果的一部分。记录所有这样的信息，然后将矩形T从集合$R \cup S$中去掉。R_4从集合$R \cup S$中被剔除，因为它不再需要参与结果集中的其他相交对。

4) 继续移动扫描线来穿过集合$R \cup S$，直至碰到下一个矩形，在本例中是S_2。这时进行步骤2和3。

5) 当$R \cup S = \varnothing$时，处理结束。

经过空间连接算法的过滤阶段，得到了结果$<R_4, S_2>$、$<R_1, S_2>$、$<R_1, S_3>$、$<R_2, S_3>$和$<R_3, S_1>$，它们将作为精炼阶段的候选对，精炼阶段是基于对象的精确几何条件完成的。只要精确几何计算表明没有交叠，精炼阶段就可能会从最后的候选对中淘汰不符合要求的候选对。过滤阶段的主要目的是，尽可能多地淘汰不符合条件的对，从而减小精确几何计算的计算代价。第5章和第7章给出了处理空间查询

中基于磁盘算法的更多细节。

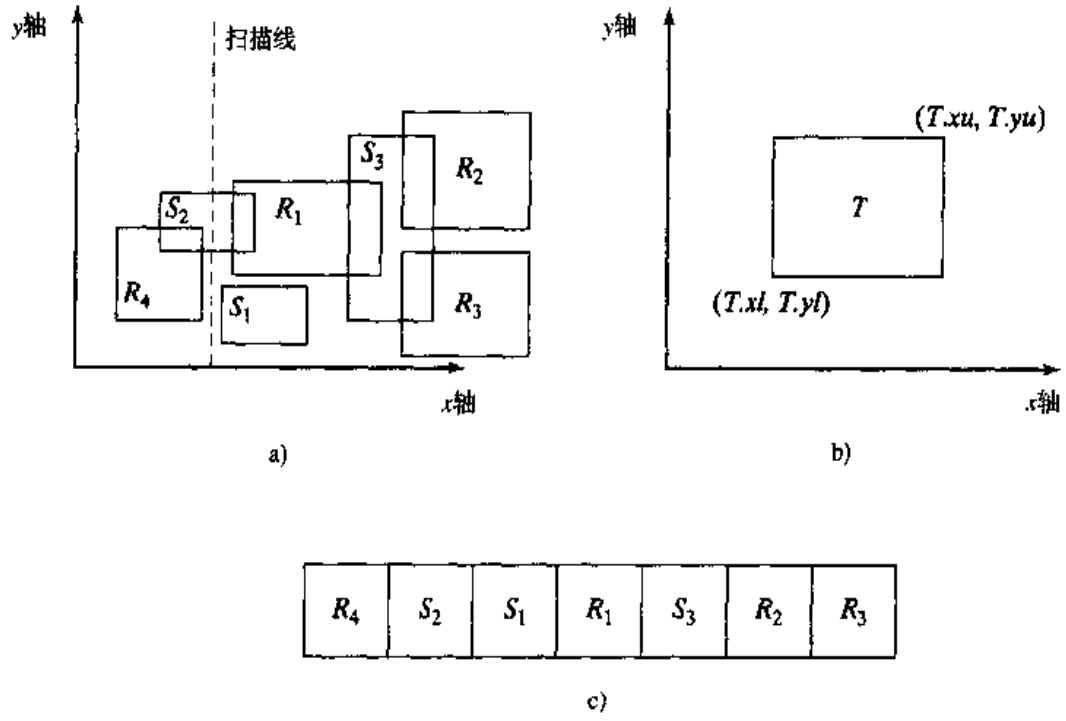


图1-9 确定相交矩形对。a) 两个矩形集合：*R*和*S*。b) 标出左下角和右上角坐标的矩形*T*。c) 排序后的矩形集合。注意，平面扫描算法的过滤特性在这个例子中，有12个可能的矩形对将被连接。过滤阶段将可能的数目降低到5。然后用准确几何测试来验证这5对对象是否满足查询谓词［Beckmann et al.，1990］

1.6.4 文件组织和索引

DBMS用来处理海量的数据。这就可以解释GIS数据分析和数据库环境下的算法的区别。前者主要关注减少算法的计算时间，它假定整个数据集都驻留在主存。而后者强调的是将计算时间和I/O（输入/输出）时间的总和减到最少。I/O时间是指数据从磁盘（硬盘）传输到主存所需的时间。这是因为，尽管价格不断下跌，但是主存仍不足以容纳许多大型应用的全部数据。两者的差异是概念上的，体现了人们对计算机基本设计的不同理解。对许多程序员来说，计算机主要包括两部分：CPU（中央处理单元）和无限量的主存（图1-10a）。另一方面，对于许多DBMS设计者来说，计算机包含三个部分：CPU、有限的主存，以及无限的硬盘空间（图1-10b）。虽然CPU可以直接访问主存中的数据，但在访问磁盘数据时还是要先将它们传输到主存中。花费在访问随机存储器（random access memory，RAM）数据和磁盘数据的时间是不可同日而

语的：前者大约是后者的十万分之一（2000年）。这个比率还在逐年降低，因为比起磁盘和二级存储设备，CPU和内存的速度变得越来越快。

这里（细节在第4章介绍）我们可以将二级存储设备（硬盘）比作一本书。从磁盘到主存之间的最小传输单位是页，表中的记录就像每页上有结构的文本行。用户提交的一条查询，其实就是搜索这本书某页中的几个选定的行。一些页面可以驻留在主存，但每次只能从中取出一页。为了加快搜索速度，数据库使用索引。为了搜索某一页中的一行，DBMS可以将一个表所涉及的所有页面取出来，并一行接一行地扫描，直到找到所需的记录为止。另一种做法是在索引中搜索所需的码（key）字，然后直接跳到索引所指定的页面。书中的索引条目按字母顺序排序。同样，如果这个索引构建在数字形式上，例如社会保障号，那么就可以按照数字顺序排序。

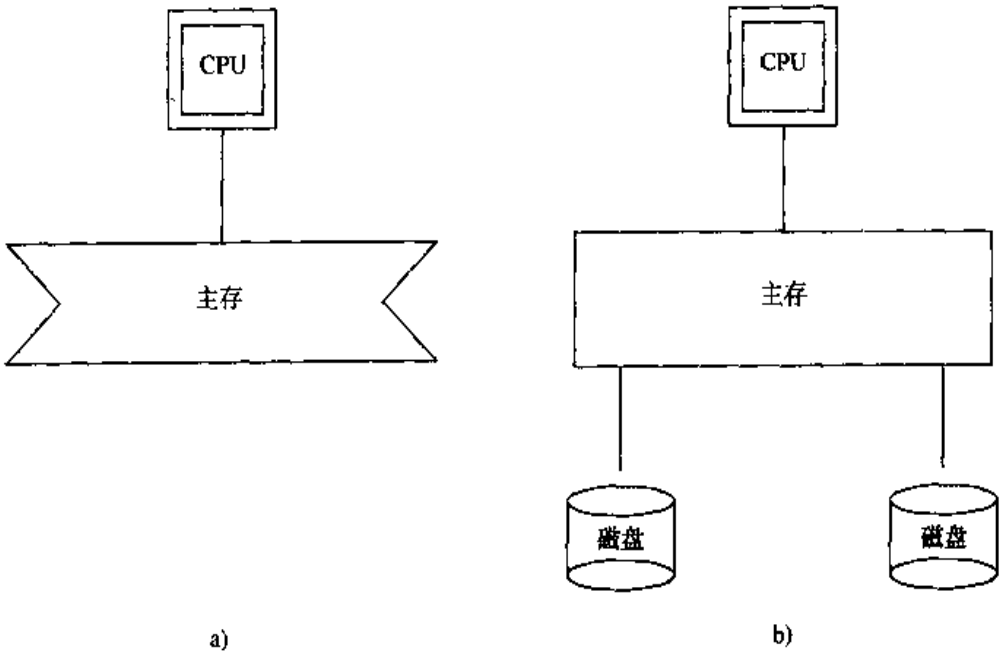


图1-10 a) 程序员的观点。b) DBMS设计者的观点

空间数据是可以分类和排序的，但这会丢失空间的相邻性（proximity）。这是什么意思呢？参照图1-11a，图中有一个按行排序的网格。现在来考虑4的相邻数字，在这个网格里4的相邻数据是3、7和8，但是如果按照排序的顺序来存储，它的相邻数字就成了3和5。因而，排序会破坏原有的相邻关系。为了找到更好的多维数据排序方法，人们进行了大量的研究工作。图1-11b展示了另一种方法。值得一提的是，没有哪一种全排序可以完全保持空间的相邻性。

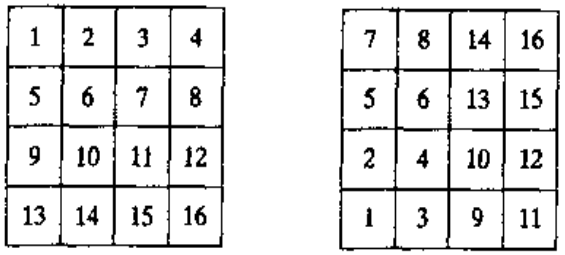


图1-11 对多维数据排序的两种方法。a) 行排序。b) Z排序。如果按照数字的升序来画线，Z模式就会很明显

B树（B表示Balanced）索引结构可以说是关系DBMS中用得最广泛的索引。有人认为，B树索引结构是关系数据库技术广为采用的主要原因。B树的实现主要依赖于索引域中排序的存在。图1-12反映了二叉树和B树的关键区别。B树的每个节点对应磁盘的一个页面。每个节点的条目数取决于索引域的特征和磁盘页面的大小。如果一个磁盘页面有m个键，那么B树的高度是$O(\log_m(n))$，这里n是总的记录数。对于一万亿（10^{12}）条记录来说，在$m = 100$的情况下，只需要6层的B树。这样，即使是面临如此大量的记录，对于指定一个键值，检索一条记录大约只需要读取6次磁盘。

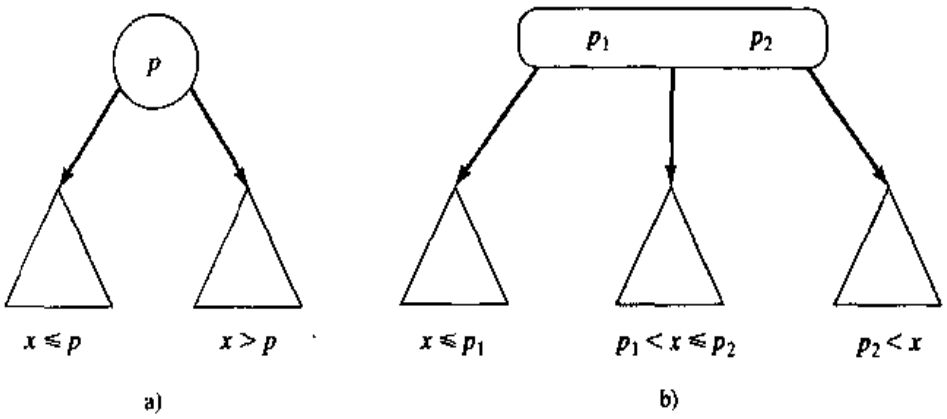


图1-12 a) 二叉树。b) B树

由于多维空间不存在自然排序，B树也就无法直接用于创建空间对象的索引。为了避免空间对象在自然排序方面的不足，通常把空间排序与B树结合起来。许多商业系统就采纳了这种方案。

R树数据结构是最早的专用于处理多维扩展对象的索引之一。它从根本上修改了B树的思想，以适应扩展的空间对象。图1-13给出一个R树组织扩展对象的例子。第4章将对此进行详细的介绍。

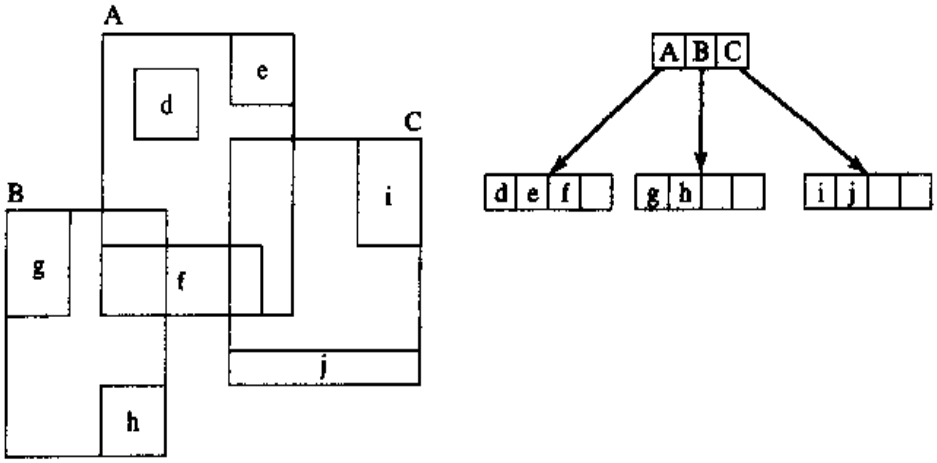


图1-13 为了处理空间对象将B树扩展为R树

1.6.5 查询优化

关系数据库技术的一项主要优势在于，它通过产生一个完善的查询评估计划来高效地执行查询。为了解释这一点，我们重新来看查询“找出拥有公司的所有女性参议员的名字。”这个查询实际上由两个子查询构成：一个选择查询和一个连接查询。“所有女性参议员的名字”是个选择查询，因为要从参议员列表中选出所有的女性参议员。而查询“找出拥有公司的所有参议员”属于连接查询，因为我们合并了两个表来处理这个查询。问题在于，以什么顺序处理这两个子查询：是先做选择后做连接，还是先做连接后做选择？别忘了连接是多遍扫描查询，而选择是单遍扫描查询，所以对较小的表来说连接操作比选择操作的代价更大。因此，显然应该先做选择再做连接。

再来考虑以前讨论过的空间查询：“找出其代表的选区面积大于300平方英里并在这个选区中拥有公司的所有参议员”。这个查询也是由两个子查询构成：一个范围查询和一个空间连接查询。范围子查询是“列出其代表的选区面积大于300平方英里的所有参议员名字。”空间连接查询是“找出在自己所代表的选区拥有公司的所有参议员。”同样，范围查询是单遍扫描查询，而连接是多遍扫描查询，不过此处有个很重要的区别。这里的单遍扫描查询要用到一个隐含的高代价函数：*S.district. Area()*，该函数用来计算每个选区的面积。当然，连接操作也涉及了一个计算代价昂贵的函数：*Within (B.location, S.district)*。因此，此时的关键在于，如何确定处理空间子查询的顺序。第5章和第7章将讲述这个重要的步骤。

1.6.6 数据挖掘

数据挖掘，即进行系统的搜索找出隐藏在电子数据中潜在的有用信息，它是目前在学术界内外非常热门的研究问题。大部分公司和政府机构都意识到，他们年复一年积累并且仍在不断收集的庞大数据可以用来进行系统的研究，为他们的运作提供新的启示。这样，本来被视为存储和管理的负担的数据，一下子就成了财富和利润的来源。下午购买尿布的顾客很可能也会购买啤酒，这个如今看来几乎带有神话色彩的故事，就是利用数据挖掘工具发现的。这极大地激发了人们设计和发明高效的数据挖掘工具的热情。到目前为止，对数据挖掘的研究工作大多都集中在算法设计和算法分析上面。数据库专家可以将他或她的专门知识用于两个前沿方向：

1) 设计适用于更大数据集的算法，或者设计可与数据挖掘算法一起使用的通用化工具。

2) 扩展SQL的“挖掘”功能，以便能够在SQL内调用挖掘工具。

有人估计，将近80%的数字形式的数据实际上是空间数据，这就驱使研究人员全力创造适用于空间数据本质的挖掘技术。例如，随机采样是一项常用的数据挖掘技术，用来减少所要分析的数据集的尺寸，同时又不会造成信息的明显丢失。传统的采样方法由于空间自相关性（spatial autocorrelation）面无法适用于空间数据。因此，需要把空间统计技术整合到挖掘算法中来处理空间数据。

对GIS来说，数据挖掘并不是新的技术。遥感图像的地图生成和分类是相对成熟的领域，它们显然与数据挖掘有关。历史上著名的空间数据挖掘例子包括：对造成1854年伦敦霍乱流行的受污染水源的辨认［Griffith，1999］，以及证明饮用水中的氟化物对牙齿健康有益。在第7章我们将详细讲述这一部分的内容。

1.7 小结

空间数据管理可用于许多学科，包括地理学、遥感、城市规划以及自然资源管理。空间数据库管理在解决一些有巨大挑战性的科学问题上发挥了重要作用（例如，全球气候变化和基因研究）。

有三类用户适合使用空间数据库管理系统。在策划营销活动、选择物流中心以及确定零售店位置时，公司用户希望用空间数据来扩充其他类型的信息。对于一个业务用户来说，空间数据是重要的辅助信息来源。

对于专门研究环境、自然资源以及地理学的学者来说，空间数据有着至高无上的重要性。这个用户群体已经在使用GIS产品来分析空间数据，但是随着数据量的快速增长，需要有专门的数据管理技术来处理具有特殊性质的空间数据。

第三类是希望用空间数据来使其经历和交互个性化的用户，特别是在万维网上。比如，搜索引擎得到的查询结果可以空间定位，这就更有意义。

1.8 参考书目

1.1 有关数据库技术的历史和DBMS的总体介绍，参见［Ramakrishnan，1998; Elmasri and Navathe，2000; Silberschatz et al.，1997］。对于RDBMS在处理空间数据方面不足的概述以及对象关系DBMS的总体介绍，参见［Stonebraker and Moore，1997］。

1.2 有关SDB的概述，参见［Scholl et al.，2001;Shekhar et al.，1999a］和超大型数据库（very large databases，VLDB）期刊的专题版，特别是［Guting，1994a］中的概述。

1.3 有关在现有的数据库系统中集成SDB以便用于CAD的应用的主题，参见［Kriegel et al.，2001］。SDBMS的例子参见［de La Beaujardiere et al.，2000］。

1.3.2 空间数据模型详见［Egenhofer，1991b;Worboys，1995］和［Laurini and Thompson，1992;Price et al.，2000］。

1.3.3 扩展SQL以便用于SDB的主题在［Egenhofer，1994］中有所讨论。OGIS为地理空间应用所做的对SQL扩展的规范在［OpenGIS，1998］有所描述。

1.3.4 在［Laurini and Thompson，1992;Worboys，1995］中广泛地涵盖了空间数据表示方面的内容。过滤-精炼模式在［Chrisman，1997］中有所讨论。

1.3.5 空间索引的标准参考见［Samet，1990］。最新的概述见［Gaede and Gunther，1998］。

1.3.8 SDB体系结构设计的有关问题在［Adam and Gangopadhyay，1997; Worboys，1995］中有所讨论。

1.9 习题

讨论

1. 讨论空间数据与非空间数据之间的区别。
2. 地理应用是一种常见的空间数据源。列出至少4种其他重要的空间数据源。
3. 相对于文件系统而言，将空间数据存储在DBMS中的好处是什么？
4. 如何利用对象关系数据库来实现一个SDBMS？
5. 列出空间数据库、CAD数据库和图像数据库之间的异同。
6. 矢量数据模型和栅格数据模型之间的相互影响经常可以与物理学中的波粒二象性作比较。讨论之。
7. 认知图的定义是“内存中存储的世界及其空间属性的内在表示。”人类的大脑能将空间对象表示为离散的实体吗？是否对栅格表示法存在内在偏见而更喜欢矢量表示法？
8. 排序是快速访问“传统数据”的普遍方法。为什么对空间数据排序却很困难？
9. 全球气候变化预测是科学界公认的最具挑战性的问题。SDBMS如何在该领域发挥作用？
10. 电子商务零售商采用某个设施通过Internet与全球客户联系。他们中的一些人宣称在Internet时代地理和位置无关紧要。你同意吗？说明你的观点。
11. 给出基于位置服务和位置无关服务的定义，并举例说明。
12. XML渐渐成为互联网应用中最普及的数据交换模型。讨论XML的研究对于空间数据库所产生的影响。
13. 比较和对比：

(i) GIS和SDBMS

(ii) OODBMS和ORDBMS

(iii) GI Systems和GI Services

(iv) 数据模型和查询语言

(v) 查询处理和文件组织及索引

(vi) 主存和磁盘

(vii) 查询和数据挖掘

14. 给出空间数据库的定义。除本章列出的空间数据库应用之外，再举出几个空间数据库应用的例子。

第2章
空间概念和数据模型

本章描述与空间数据库应用建模有关的技术。GIS是最流行的空间数据库应用，我们的讨论也着力反映这个事实。除了GIS之外，像CAD和天文学等含有明确的空间或几何成分的应用，也适用本章所描述的技术。

传统的数据库主要关注商务和管理应用这样的领域，它将重点放在高效且安全地处理大量相对简单的事务上。随着全球定位系统（global positioning system，GPS）这样的数据获取设备的价格的下降，人们能够更容易地通过Internet获取卫星和制图数据；同时，桌面计算机的计算能力也日益增长，因此必须重新定义数据库的功能。不能再将DBMS看成是一个完全封闭的数据储藏库，而是多系统计算环境中一个活跃的组成部分。事实上，将计算密集型的任务直接转移到DBMS中已是大势所趋。GIS就是该趋势的一个重要实例。

SDBMS将空间数据建模的特殊需求整合到系统中，其意义在于：

- 与传统商务数据相比，空间数据更为复杂，而旧式的数据库结构不足以处理

空间数据。

- 数据库设计和实现的问题通常是由计算机科学家来解决的，而空间数据的处理则落入地理学家、环境学家以及其他物理学家的研究范畴。长久以来，这些学科各自沿着不同的道路发展。

本章各部分的安排如下：2.1节描述几个不同的空间信息处理模型；2.2节给出数据库设计和建模的一般原则；2.3节主要研究实体-联系（entity-relationship，ER）模型在SDB上的扩展；2.4节讨论统一建模语言(unified modeling language，UML)，这是一个主要用于面向对象数据库设计的概念模型；最后对本章进行总结。

2.1 空间信息模型

我们引入一个州立公园（State-Park）的空间数据库例子，来说明空间数据建模中的各种概念。State-Park SDB由多个森林（forest）组成，这些森林又是不同树种的林分[Θ]（forest-stand）的集合。州立公园中有道路（road），并有一个管理员（manager）。州立公园中还有负责监控和扑灭火灾的消防站（fire-Station），以及星罗棋布的诸如野营地和办公室之类的设施（facility）。最后一点，州立公园中有河流（river）穿过并为各种设施供水。

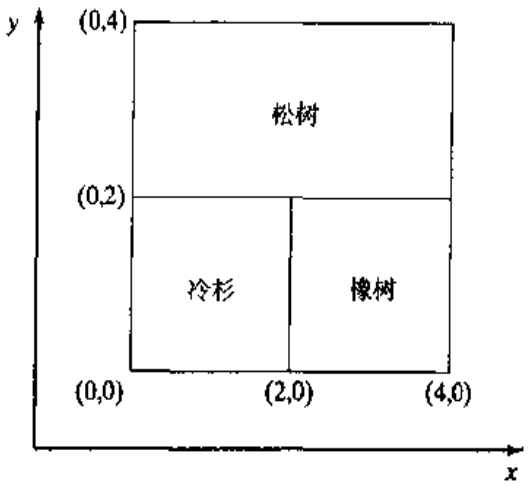
空间信息模型通常分为两大类：场（field）模型和对象（object）模型。我们借助图2-1的例子来解释这种二分法。考虑一种理想情况，将森林划分为同构区域，即每个区域只有一个（主要的）树种。本例中有3个树种：冷杉（Fir）、橡树（Oak）和松树（Pine）。

有两种互补的森林建模方法。从函数的角度看，森林可建模成一个函数。该函数的定义域是森林占据的地理空间，而值域是3个元素（树种的名称）的集合。设这个函数为f，它将森林所占据空间的每个点映射到值域的一个具体元素上。函数f是个分段函数，它在树种相同的地方取值恒定，而在树种发生变化处才改变取值。在GIS中，这个函数模型称为场模型。图2-1c的场模型是用分段函数来表示的。场模型的另一种表示方法是网格（grid），网格中的每个单元格（cell）

Θ 林分（forest-stand）是指一群树木或其他植物的集合体，占有一定面积，具有充分整齐的树种组织，龄级分配，且与邻近面积的森林或其他植物有明显区别。——译者注

标注着占主导地位的树种的名称，这种表示方法在各种林分边界极不规则的情况下更为适用。场的其他表示方法还有等值线，它显示了在某个物理参数（例如温度、气压）上具有固定值的轮廓线。

现在考虑函数f的值发生变化的地方。在明确规定树种之间界限的理想情况下，就可以得到多边形的边界，每个多边形都有一个唯一的标识符和一个非空间属性——树种的名称。这样，可以把森林建模成多边形（例如林分）的一个集合，每个多边形对应一个树种。这是对象模型的观点，如图2-1b所示。



a)

林分的对象观点

区域ID	主要树种	区域/边界
FS1	松树	[(0,2),(4,2),(4,4),(0,4)]
FS2	冷杉	[(0,0),(2,0),(2,2),(0,2)]
FS3	橡树	[(2,0),(4,0),(4,2),(2,2)]

b)

林分的场观点

$$f(x,y) = \begin{cases} \text{“松树,”} & 2 \leqslant x \leqslant 4;\ 2 < y \leqslant 4 \\ \text{“冷杉,”} & 0 \leqslant x \leqslant 2;\ 0 \leqslant y \leqslant 2 \\ \text{“橡树,”} & 2 < x \leqslant 4;\ 0 \leqslant y \leqslant 2 \end{cases}$$

c)

图2-1 对象-场的二分法。a) 一幅显示3种林分（松树、冷杉、橡树）的地图。b) 对象的观点：将地图表示为3个对象的集合，每个对象有唯一的标识符、主要的树种和一块区域。区域的边界（一个多边形）由坐标指定。c) 场的观点，这时区域中的每个点被映射为主要树种对应的值

对一个空间应用的建模来说，到底采用场模型还是对象模型，主要取决于应用

要求和习惯。对于形状不定的现象，例如火灾、洪水和危险物泄漏，当然采用边界不固定的场模型进行建模。其他一些空间现象也可用场来建模。场模型通常用于具有连续的空间变化趋势的情况的建模，如海拔、温度，以及土壤变化。其实，在遥感领域，主要利用卫星或飞机上的传感器收集地表数据，此时场模型是占主导地位的。而对象模型更多地用于运输网络（如道路）、地块的征税和合法所有权应用等方面的建模。据推测，对象模型可能是源自将土地作为财产来划分所有权的社会需求［Couclelis，1992］。

2.1.1 基于场的模型

对于空间应用来说，定义场模型要求我们确定3个组成部分：空间框架（spatial framework）、场函数（field function）和一组相关的场操作（field operation）［Worboys，1995］。

空间框架F是一个有限网格，这个网格加诸在基本空间上。所有度量都基于这个框架来完成。空间框架最常用的例子是地球表面的经度-纬度参照系。空间框架是一种有限的结构，由于离散化而导致的误差是在所难免的。一个包含n个可计算的函数或简单场$\{f_i,\ 1\leqslant i\leqslant n\}$的有限集

$$f_i\text{：空间框架} \rightarrow \text{属性域}(A_i)$$

将空间框架F映射到不同的属性域A_i中。对各种场函数和属性域的选择要取决于当时的空间应用。在森林的例子中我们用了一个单一场（single-field）函数，它是分段函数，其属性域为集合｛冷杉，橡树，松树｝。对于函数是单值而基本空间是欧氏平面的特殊情况，场自然就看成表面或等值线，它们是具有相同属性值的点的轨迹。基于场建模的第三个重要内容是场函数的操作规约。

不同场之间的联系和交互由场操作(field operation)来指定。场操作把场的一个子集映射到其他的场。场操作的例子有并(+)和复合(o)：

$$f+g: x \rightarrow f(x)+g(x)$$

$$fog: x \rightarrow f(g(x))$$

场操作可以分成三类［Worboys，1995］：局部的（local）、聚焦的（focal）

和区域的（zonal）。

1. 局部操作

对于一个局部操作，空间框架内一个给定位置的新场的取值只依赖于同一位置场的输入值。例如，考虑一个理想化的State-Park，它可完全划分成树、湖和草地。函数f和g定义为：

$$f(x)=\begin{cases}1 & \text{如果}x=\text{“树”}\\ 0 & \text{其他情况}\end{cases}$$

以及

$$g(x)=\begin{cases}1 & \text{如果}x=\text{“湖”}\\ 0 & \text{其他情况}\end{cases}$$

那么，$f+g$，即f与g的并就定义为：

$$(f+g)(x)=\begin{cases}1 & \text{如果}x=\text{“树”或者“湖”}\\ 0 & \text{其他情况}\end{cases}$$

这是一个局部操作的例子。

2. 聚焦操作

对于一个聚焦操作，在指定位置的结果场的值依赖于同一位置的一个假定小邻域上输入场的值。微积分中的极限（limit）运算就是一个聚焦操作。设$E(x, y)$是State-Park的高程场，即E给出空间框架F中位置(x, y)的高程值。因而，计算高程场的梯度$\nabla \cdot E(x, y)$就是一个聚焦操作，因为(x, y)的梯度值依赖于高程场在(x, y)的一个“小”邻域上取值。这里我们假定场是平滑变化的，其间没有像Forest例子中的分段函数那样的急剧跃变。

3. 区域操作

区域操作很自然地与聚集运算符或微积分中的积分运算（$\int dxdy$）有关联。在森林的例子中，分段函数把森林映射到属性集｛冷杉，橡树，松树｝中，同时还将基本空间划分成三个多边形，或者区域（zone）。计算每个树种的平均高度就是一个区域操作。

2.1.2 基于对象的模型

在基于对象的建模中，其关键是把空间信息抽象成明确的、可识别的和相关的事物或实体，称之为对象。例如，我们可以通过公园中的林分、河流、湖和道路来刻画一个公园，所有的这些实体显然是可区分和可识别的，它们是否相关依赖于我们所要建模的应用。每个对象都有一套刻画它的属性集。与传统数据库建模中普遍采用的对象/实体相比，空间对象的最主要的特点在于它的属性可以分为截然不同的两类：空间属性和非空间属性。对象通过其空间属性与包含它的基本空间进行交互。在森林例子中，表示冷杉的林分是个空间对象，表示空间范围的多边形是空间属性，而名称“冷杉”是非空间属性，这是一个字母数字型的属性。一个空间对象可以有多个空间属性，例如表示多层次的概括。例如，地图上的一条路，就可以根据地图的比例尺表示成一条线或一个多边形。

2.1.3 空间数据类型

对基于对象的空间信息模型来说，其关键问题是选择一组基本空间数据类型，来满足对地图常用形状建模的需求。这些年陆续提出了许多建议，其中OGIS规范［OGIS，1999］正逐渐得到大家的认同。图2-2给出了用UML符号表示的二维空间几何体的基本构件及其相互关系。我们将在2.4节中讨论UML符号。

最为通用的形状是由“空间表示体系”所描述的“几何体”来表示的，“空间表示体系”是一个坐标系统，类似于经度/纬度或其他公认框架。“几何体”分为4类：点（point）、线（curve）、面（surface）和几何体集合（geometry collection）。点描述一个零维对象的形状，例如世界地图上的城市中心。线描述一维对象的形状，例如世界地图上的河流。线对象通常用线串（linestring）来近似表示，它由两个或更多的点表示。最简单的线串是一条连接两个或更多点的直线段。面则描述了二维对象的形状，例如世界地图上的国家。面通常用多边形建模。几何体集合表示复杂的形状，诸如油井的集合，一群岛屿等。几何体集合有三种类型，即多点（multipoint）、多线（multicurve）和多面（multisurface）。几何体集合空间数据类型保证了OGIS空间数据类型在几何操作上的闭合性（closure），这些操作包括几何并、几何差或

几何交操作。举例来说，如果对加拿大和魁北克边界作几何差运算，尽管它们都是“面”空间数据类型，但运算结果是“多面”。闭合性对于支持多步查询和数据处理很有用。

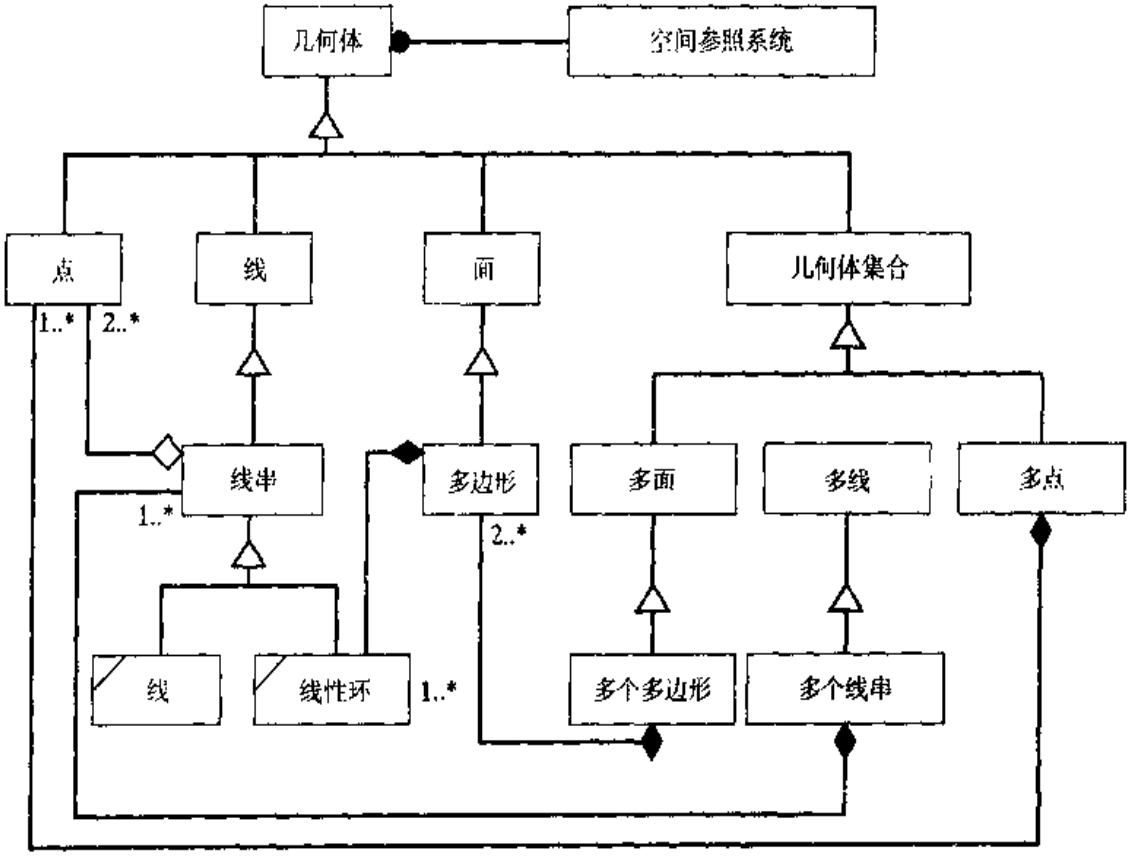


图2-2 OGIS提出的关于空间几何体的基本构件（采用UML概念表示）

2.1.4 空间对象的操作

空间对象之间如何交互？在基于场的建模中，场函数主要决定操作的类型。毕竟，不可能将梯度运算用于非平滑的场！在基于对象的建模中，基本内嵌空间决定对象间所具有的关系。现在，我们来描述内嵌空间和关联关系的类型。

1. 面向集合的

在所有内嵌空间中，最简单且最通用的类型是面向集合的内嵌空间。这种集合可以利用一些常见的关系，即在基于集合的关系中常见的并、交、包含和属于关系。层次关系（如森林包含林分，州立公园包含森林，州包含州立公园）就适于用集合理论来建模。

2. 拓扑的

为了直观地理解拓扑空间，设想在一块橡胶片上画两个相接的多边形，然后进行拉伸或扭曲（但不能切割或折叠）来改变橡胶片的形状，这时两个多边形的邻接性维持不变。相接（meet）是拓扑属性的一个例子，研究保持拓扑属性不变的变换（形变）的领域称为拓扑学。考虑带有国家行政边界的世界地图，无论地图是画在球面还是在平面空间上，相邻国家总是彼此相接。而多边形的面积则显然不是拓扑属性。事实上，不同国家的相对面积通常在不同地图上会有所变化。在很多平面地图上，靠近赤道的国家相对于靠近两极的国家而言面积减少了。表2-1列出了常见的一些拓扑属性和非拓扑属性［Worboys，1995］。

从空间/地理数据库的角度看，相接（meet）、包含（within）和交叠（overlap）这类拓扑关系最有可能被空间数据库管理系统的用户查询。一个给定的地块与危险废弃物场相邻吗？河流洪水泛滥区与提出的高速公路网交叠吗？这些都是拓扑关系的例子。空间数据库中两种常见拓扑查询是：

- 找出所有与给定对象存在拓扑关系R的对象。
- 对象A和B之间存在什么样的拓扑关系？

在一个平面$\Re^2$上，两个对象A和B之间的二元拓扑关系要基于以下的相交情况：即A的内部（A°）、边界（∂A）和外部（A^-）与B的内部（B°）、边界（∂B）和外部（B^-）之间的交［Egenhofer，1994］。对象的这六个部分构成九交（nine-intersection）矩阵，它定义了一个拓扑关系，可以用下面的矩阵来表达这个关系：

$$\Gamma_9(A,B)=\begin{pmatrix} A^\circ\cap B^\circ & A^\circ\cap\partial B & A^\circ\cap B^- \\ \partial A\cap B^\circ & \partial A\cap\partial B & \partial A\cap B^- \\ A^-\cap B^\circ & A^-\cap\partial B & A^-\cap B^- \end{pmatrix}$$

考虑取值有空（0）和非空（1），可以确定有$2^9=512$种二元拓扑关系。对于嵌在$\Re^2$中的二维区域，有八个关系是可实现的，并且它们彼此互斥且完全覆盖。这些关系为：相离（disjoint）、相接（meet）、交叠（overlap）、相等（equal）、包含（contain）、在内部（inside）、覆盖（cover）和被覆盖（covered by）。

图2-3显示了如何使用九交矩阵来表示拓扑关系。例如，在九交模型中，相离关系可以用图2-3左上角的布尔矩阵表示。0值说明*interior* (*A*)与*interior* (*B*)或

boundary (*B*)没有公共点。类似地，*interior*(*B*)与*boundary* (*A*)没有公共点，*boundary* (*A*)与*boundary* (*B*)也没有公共点。

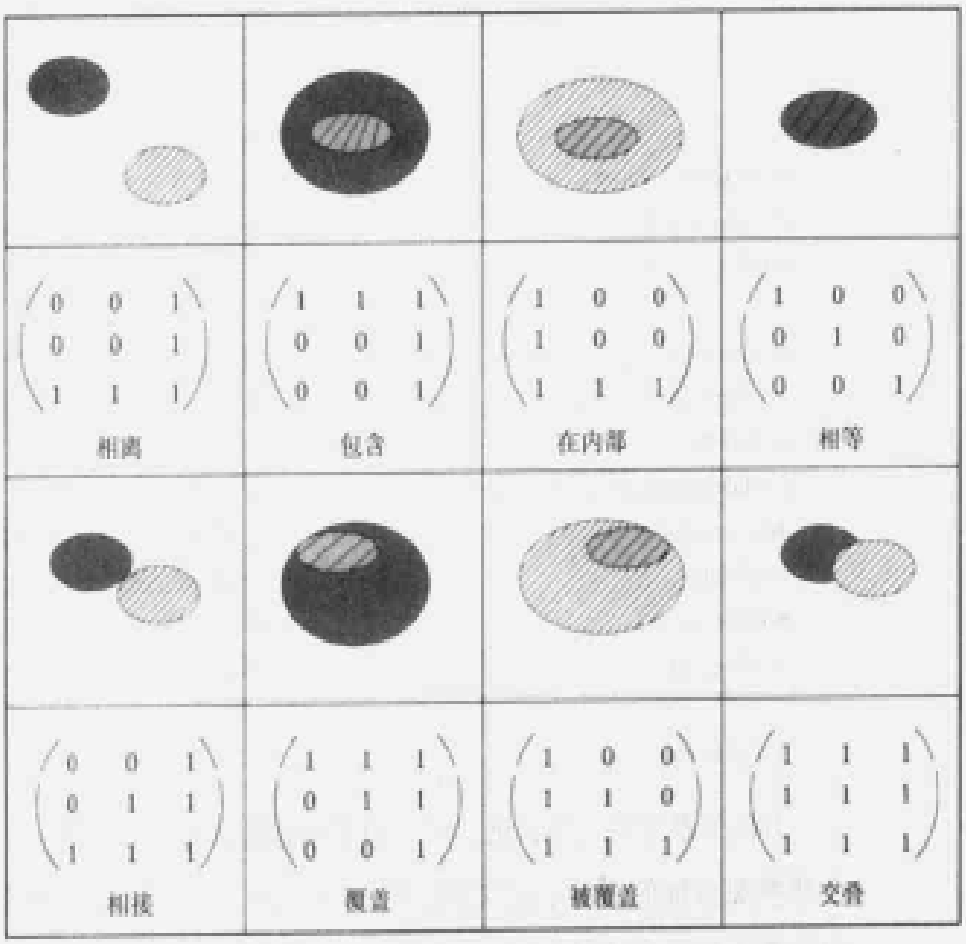


图2-3 九交模型［Egenhofer et al.，1989］

对于其他空间数据类型对，例如（*point*，*surface*）、（*point*，*curve*），其拓扑关系可以用类似方式定义。如表2-1所示，一个点可以在一个面的内部、外部或者边界上；一条线可以穿过（cross）面的内部，或者与一个面相接，或者与一个面相离；一个点可以是一条线的端点、内点或不在线上。线之间的关系要复杂一些，是一个仍在研究的领域［Clementini et al.，1993］。

表2-1 拓扑和非拓扑操作举例

拓扑的	
endpoint(point，arc)	点是弧的端点
simple-nonself-intersection(arc)	非自交的弧
on-boundary(point，region)	温哥华在加拿大和美国的边界上
inside(point，region)	明尼阿波利斯市在明尼苏达州内

（续）

outside(point，region)	麦迪逊市在明尼苏达州之外
open(region)	加拿大的内部是个开域(不包括其边界）
close(region)	Carleton郡是个闭域（包括其边界）
connected(region)	瑞士是个连通域（对于区域上的任两点，都有完全内含在该区域上的路径将这两点连接起来），而日本不是连通域
inside(point，loop)	点在环中
crosses(arc，region)	路（弧）穿过森林（区域）
touches(region，region)	明尼苏达州（区域）是威斯康星州（区域）的邻州
touches(arc，region)	州际90号高速公路（弧）经过密歇根湖（区域）
overlap(region，region)	土地覆盖（区域）和土地利用（区域）相重叠
非拓扑的	
Euclidean-distance point, point	两点间的距离
direction(point，point)	麦迪逊市在明尼阿波利斯市的东面
length(arc)	单位向量的长度是1个单位
perimeter(area)	单位正方形的周长是4个单位
area(region)	单位正方形的面积是1个平方单位

3. 方位的

方位关系可为三类：绝对的、相对目标的和基于观察者的。绝对方位关系是在全球参照系统的背景下定义的，例如东、西、南、北、东北等等。相对目标的方位关系根据与所给目标的方向来定义，例如左、右、前、后、上、下等等。基于观察者的方位关系是按照专门指定的，称为观察者的参照对象来定义的。

4. 度量空间

用数学的术语讲，集合X在满足以下条件时就称为一个度量空间：如果对X中的任意一对点x和y，都存在一个与之关联的实数$d(x, y)$，称为x到y的距离（也称为一种度量），且对于X中任意的x，y，z都满足如下性质：

1) $d(x, y) \geq 0$且$d(x, x) = 0$

2) $d(x, y) = d(y, x)$

3) $d(x, y) \leq d(x, z) + d(z, y)$

任何满足上述性质的函数称为X上的一个度量。

在度量空间中，对距离也进行了很好的定义。距离函数可以导出对应空间上的

一个拓扑结构，因此每个度量空间也是一个拓扑空间。在网络或图环境中，度量空间扮演着重要的角色。优化距离和最短行程时间的查询在度量空间的环境中得到了很好的解决。

5. 欧氏空间

设R是实数域，R上的向量空间（vector space）V是向量v的非空集合，它们之间有两种运算：

1) 加法：对所有的$u\ v \in V$，$u+v \in V$

2) 积：对所有的$\alpha \in R$，$v \in V$，$\alpha u \in V$

在该向量空间中，除了存在0向量之外、加法和积这两个运算还必须满足其他一些公理。对向量空间的完整讨论可参见［Blyth and Robertson，1999］。

如果存在一个（极小）有限向量集$\{e_1, e_2, \ldots, e_n\}$使得任何$v \in V$都可表示成这些$e_i$的线性组合，即存在$\alpha_1,\ldots\alpha_n \in R$，使得：

$$v = \alpha_1 e_1 + \ldots + \alpha_n e_n$$

则称该向量空间是有限维的。在三维空间中，e_i对应于人们熟悉的x、y、z坐标轴。如果我们在向量空间中添加内积（用一对尖括号表示）的概念，将得到一个欧氏空间。在欧氏空间的环境下，可以定义所有的空间关系（包括集合的、拓扑的、度量的和方位的（北/南））。

2.1.5 动态空间操作

迄今为止，我们定义的大多数操作都是静态的（static），从某种意义上说，操作数不受操作的应用影响。例如，计算一条线的长度对线本身没有任何影响。动态（dynamic）操作则会改变它所作用的对象。三种基本的动态操作是创建（create）、销毁（destroy）和更新（update）。所有的动态操作都是基于这三个操作衍生而来的［Worboys，1995］。表2-2列出了一些创建和更新的例子。

融合（merge）操作的一个例子是“地图再分类”，许多GIS软件产品都支持这样的操作。考虑将国家地图按照各国家主要的宗教信仰（例如，无宗教信仰、伊斯

兰教、基督教、佛教和印度教）重新划分。如果两个相邻的国家具有相同的主流宗教信仰，那么融合操作将它们之间的共同边界去掉，这样就重新划分了该地图。其他类似的动态空间操作例子可以在GIS的制图投影和地图编辑的功能里找到。

表2-2 动态空间操作的典型示例［Worboys，1995］

创建	reproduce(X)	产生一个精确的复制X的复制品
	generate(X)	生成一个依赖于X但不复制X的对象
	split(X)	生成聚集是X的各个对象
	merge(X，Y，Z)	归并X，Y，Z以创建一个对象
更新	translate	在平面上移动对象的位置
	rotate	改变朝向但不改变形状
	scale	改变对象的大小，但形状不变
	reflect	对象关于一条直线的镜面反射
	shear	按照预定义的方式使对象发生形变

还有其他一些类型的空间对象操作。例如，有的操作可以定义在扩展对象的形状上，以便回答这样一些查询：哪个对象近似方形？哪两个对象能像拼图一样吻合起来？空间对象操作在制药方面也很有用，通过发现那些可与其他分子中某个区域对接的分子可以设计新药。另一类空间操作是基于可见性关系的，这类操作可以解决像“列出高尔夫球场上看到9号洞的最佳视角位置”这样的查询。希望在不久的将来，能出现人们一致认可的用于处理此类操作的数学框架。

2.1.6 将空间对象映射到Java

在定义了空间层次之后（见图2-2及相应的一系列空间关系），现在我们可以很容易地使用如Java这样的面向对象程序设计语言为这些概念进行编码。

我们给出了一个如何用Java编写特定查询的例子，这个查询例子是：“找出Maple Campground周围方圆10英里之内的旅游公司”。

采用Java的`Facility`类和`FacilitySet`类来对旅游公司和营地进行建模。独立的`Query`类用于对距离计算和查询本身进行建模。`Facility`类有三个属性，分别为*name*、*type*和*location*。属性*type*用来区分旅游公司和营地；设施位置的空间数据类型是*point*。该程序假定空间数据类型*point*可以作为空间库（如ESRI的

SDE）的一部分使用。这个库同时提供了在空间数据类型*point*之上的*distance*函数。FacilityClass类有三个方法。方法*facility*是用来初始化新对象的构造函数。方法*getName*获取*name*属性和方法，而*withinDistance*用于检测另一设施是否在一个给定距离之内。

FacilitySet类用于设施集合的建模。例如，它可建模一组旅游公司。它有两个属性：*maxSize*和*FacilityTable*。*maxSize*记录了设施集合的最大大小。*facilityTable*用来存储每个设施的信息。FacilitySet只有一个方法，它从一个文件中读取各个设施的信息并初始化FacilitySet实例的属性。

Query类实现了查询“找出Maple营地方圆10英里之内的旅游公司”。该类仅有一个名为*main*的方法，它使用循环来遍历旅游公司列表，检查各旅游公司到Maple营地的距离。

讨论Java程序的主要目的是要说明空间数据类型和操作可以应用在有别于SQL（将会在本书后面的部分用到）的宿主语言中。其次，是为了比较空间查询所需的编程工作量。在编写简单的空间查询时，利用SQL将大大减少所需的代码量，同时也减少了性能调整和数据结构选取的负担。建议读者在后面的几章中通过比较SQL和Java来回顾这些观点。

```
import java.lang.*;
import java.io.*;
import java.util.*;


/* assume class Point is given, which contains two attributes: x and y with
double type, and also some member functions, including distance (Point) */

/* Assume the original data is stored in file ''facilityFile''.
Each line in the file represents a facility; use @@ as its delimiter, e.g.
   Maple @@ campground @@ 2.0 @@ 3.0
   Office @@ Tourist-Office @@ 6.0 @@ 8.9 */

public class Facility {
   protected String name;
   protected String type;
   protected Point  location;

   public Facility (String name, String type, Point location) {
```

```
    this.name = name;
    this.type = type;
    this.location = location;
  }

  public String getName() {
    return name;
  }

  public boolean withinDistance(Facility f, double d) {
    if (this.location.distance(f.location) < d)
      return true;
    else
      return false;
  }
}

public class FacilitySet {
  const maxSize = 50;
  protected Facility[maxSize] facilityTable;

  /* read from file filename and initialize the facility table */
  public FacilitySet(String filename) {
    BufferedReader in = new BufferedReader (new FileReader(filename));
    String inline;
    StringTokenizer strLine;
    int i=0 ;
    String token;

    while ((inline = in.readLine())!= null) {
      strLine = new StringTokenizer(inline, "@@");

      /* read name */
     token = strLine.nextToken();
     FacilityTable[i].name = token;

     /* read type */
     token = strLine.nextToken();
     FacilityTable[i].type = token;

     /* read x coordinate */
      token = strLine.nextToken();
      FacilityTable[i].location.x = Double.valueOf(token).doubleValue();

      /* read x coordinate */
      String type token = strLine.nextToken();
      FacilityTable[i++].location.y = Double.valueOf(token).doubleValue();

   }
  }
}
```

```
public class FacilityDemo {

   public static void main(String[] args) {

     Facility f = new Facility("Maple", "Campground", Point(2.0,4.0));
     Facility[] fTable = new FacilitySet("facilityFile");
     String[] resultTable = new string[fTable.length];

     int j=0;
     for (int i=0; i < fTable.length; i++) {
       if (f.withinDistance(fTable[i], 2.0)
          && fTable[i].type = "Tourist-Office")
          resultTable[j++] = fTable[i].name;
   }
  }
}
```

2.2 数据库设计的三个步骤

至此，我们在本章中讨论了两种空间信息模型：对象模型和场模型。采用这些模型的做法是从数据建模所涉及的空间领域本质的概念的角度来说明的。现在，我们从数据库设计的角度来介绍传统的数据建模方法。我们的最终目标是“调和”这两个概念体系。

数据库应用通过三个设计步骤来进行建模［Elmasri and Navathe, 2000］。首先，采用高层次的概念数据模型（conceptual data model）来组织所有与应用相关的可用信息。在概念层上，重点关注应用的数据类型及其联系和约束。设计过程的这个阶段不考虑具体实现细节。概念模型通常用浅显的文字，结合简单一致的图形符号来表示。实体-联系（entity relationship, ER）模型是所有概念设计工具中最为流行的一种。

第二步，也称为逻辑建模阶段，与概念数据模型在商用DBMS上的具体实现有关。商用DBMS中的数据由实现模型来组织。实现模型的例子有：层次模型、网状模型和关系模型。其中，关系模型是目前商用数据库所实现的最为广泛的模型之一。在关系模型中，数据类型、联系和约束都被建模为关系（relation）。与关系模型联系紧密的是形式化查询语言的关系代数（relational algebra, RA）。RA由一些简单的操作组成，这些操作能够查询用关系方式组织的数据。关于RA的详细介绍，请见第3章。

关系模型并不能满足空间数据建模的要求，［Herring，1991］对此的解释如下：

> 关系代数刻画了关系数据库的查询能力。关系数据库能够回答任何以关系代数表示的查询，从而成为普遍接受的关系数据库传统应用的模型。
>
> 相反，还没有广为接受的地理信息数学模型，这给空间数据查询语言和空间数据库的设计造成了困难。此外，GIS与关系数据库之间有着相当大的语义鸿沟，也导致了复杂性和不便之处。

第7章用一个具体例子来说明如果没有附加设定，RA对于表述传递闭包（transitive closure）是无能为力的，而传递闭包是重要的图形操作，它与2.1.5节中提到的再分类操作［Delis et al.，1994］紧密关联。

最后，数据库设计的第三个步骤是物理设计的建模，它解决数据库应用在计算机中具体实现时方方面面细节。有关存储、索引和内存管理等问题都在这一阶段考虑和解决，我们将在后面的几章中讨论这些问题。现在，先介绍一下ER模型。

2.2.1 ER模型

数据库设计的第一步是提出“微型世界”的概念模型。构建概念模型的目的是以一种避开计算机隐喻的方式来表达这个微型世界，从而把应用中的概念与实现细节分离开来。对于概念数据建模来说，有许多可用的设计工具，ER模型是其中最为流行的工具之一。ER模型同关系模型无缝地整合在一起，而关系数据模型又是三个数据库设计阶段的第二步中最流行的逻辑模型之一。除了用ER模型方法和用面向对象设计方法设计的模型外，UML是另一个流行的概念建模工具。我们将在下一节详细讨论UML，我们在这里使用ER模型对*State-Park*例子进行建模。

1. 实体和属性

在ER模型中，微型世界被划分成一个个实体（entity），由属性（attribute）来描述实体性质，并通过联系互相关联。实体是物理上或者概念上独立存在的事物或对象。在*State-Park*例子中，FOREST、RIVER、FOREST-STAND、ROAD以及FIRE-STATION都是实体。

实体由属性来刻画其性质。例如，*name*是实体FOREST的属性。唯一标识实体

实例的属性（或属性集）称为码（key）。在我们的例子中，假定任意两条道路均不能同名的话，实体ROAD的*name*属性就是一个码。本例中数据库的所有ROAD实例都有唯一的名称。尽管这不是概念设计的问题，但DBMS中必须有一个机制来保证这种约束。

属性可以是单值或多值的。*species*（树种）是FOREST-STAND的单值属性。我们利用本例的情况来解释多值属性。FACILITY实体有一个*Pointid*属性，它是该实体实例的空间位置的唯一标识。我们假定，由于地图比例尺的缘故，所有FACILITY实例都要用点来表示。一个给定的设施可能会跨越两个点对应的位置，这时，*Pointid*属性就是多值的。其他实体也会有类似情况。

假设要存储有关FOREST的*elevation*（高程）信息，由于*elevation*的值在FOREST实体内部会变化，我们将该属性作为多值属性，因为不支持场数据类型。

2. 联系

除了实体和属性外，构成ER模型的第三个要素是联系（relationship）。实体之间通过联系相互作用和关联。在前面的小节中，我们已经提到了空间联系，这里来关注联系的一般概念。虽然多个实体可以同时参与一个给定的联系，但我们只讨论二元（binary）联系，即两个实体间的联系。有三种基于基数约束的联系：一对一、多对一和多对多。

(1) 一对一(1∶1)

在一对一的联系中，一个实体中每个实例只能与其他参与实体的一个实例相联系。例如，实体MANAGER和FOREST之间的联系*manages*就是一个一对一的联系，即一个FOREST只能有一个MANAGER，而一个MANAGER只能管理一个FOREST。

(2) 多对一(*M*∶1)

多对一联系可将一个实体的多个实例与另一个参与该联系的实体的一个实例相连接。*belongs_to*是实体FACILITY与FOREST之间的一个多对一联系，这里假定每个设施仅仅属于一个森林，但每个森林可以有多个设施。

(3) 多对多(*M*∶*N*)

有时候一个实体的多个实例会与另一个参与该联系的实体的多个实例相联系。实体RIVER和FACILITY之间的联系*supplies_water_to*正是这样的一个联系。有时候，联系也可以拥有属性。*supplies_water_to*有一个*Volume*属性，用来跟踪一条河流向一个设施供水的水量。

3. ER图

与ER模型相关的是ER图，ER图为概念模型提供了图形化的表示方法。在ER图中，实体用矩形表示；属性表示为椭圆，并用直线与表示实体的矩形相连；联系则表示为菱形。联系的基数（cardinality）（包括1：1、*M*：1或*M*：*N*）标注在菱形的旁边。码的属性加下划线，而多值属性用双椭圆表示。State-Park例子的ER图如图2-4所示，其中有7个实体，即FOREST-STAND、RIVER、ROAD、FACILITY、FOREST、FIRE-STATION和MANAGER。实体FOREST的属性有*name*、*elevation*和*polygonid*。*name*是唯一的标识，即每片森林有唯一的名称。图中还给出了8个联系。实体FOREST参与了6个联系，而实体FIRE-STATION只参与了一个名为*monitors*的联系。基数约束表明每个消防站只监控一片森林，但一片森林可被许多消防站监控。有些联系是空间上固有的，包括*cross*（穿过）、*within*（在内部）和

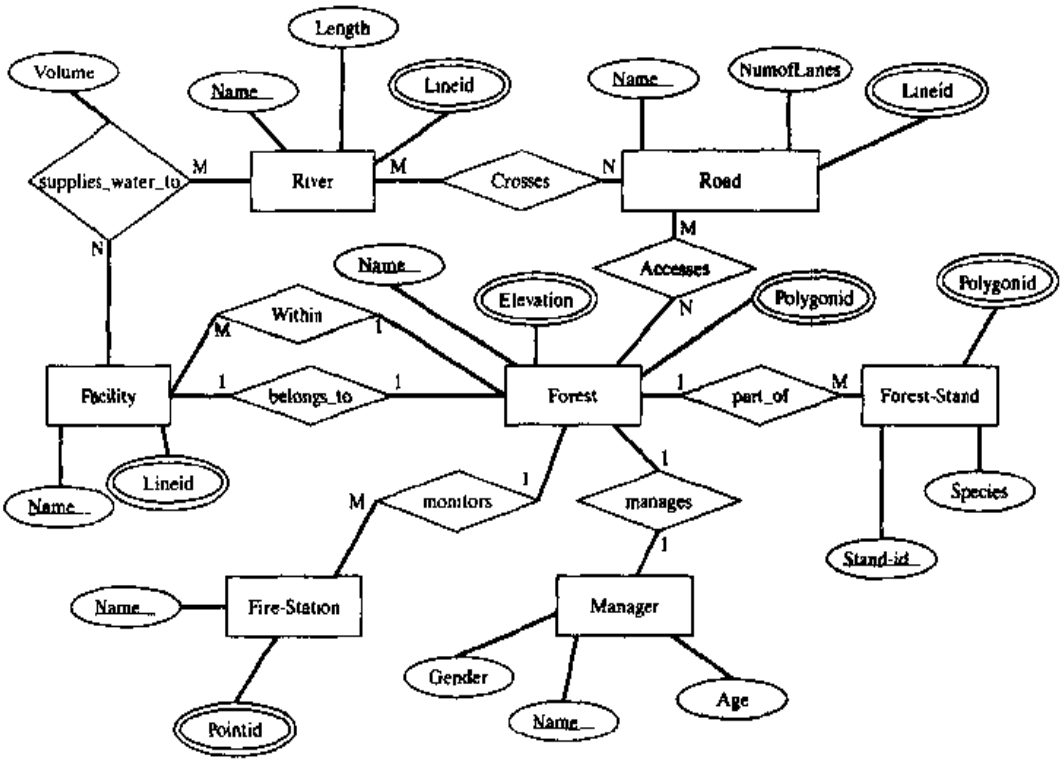


图2-4 州立公园例子的ER图

part-of（部分），而图中许多其他空间联系是隐含的。例如，一条河流穿过一条道路在图中是标明的，而一条河流穿过一片森林则是隐含的。

2.2.2 关系模型

Codd在1970年提出用关系模型描述数据。从那时起，关系模型逐渐成为最流行的逻辑数据模型之一。关系模型的广为流行和强大能力要归功于其简洁的结构。我们利用*State-Park*例子来解释关系模型的术语。假设我们需要组织州立公园所有森林的可用数据，这时可以用一个名为Forest表的形式来组织这些信息，把一系列可用的信息在列入表的列（column）中。对于表Forest，相关联的数据由三部分组成：森林的*name*（名称），它的*elevation*（高程）以及*spatial geometry*（空间的几何形状）。

该表称为一个关系(relation)，其列称为属性(attribute)。Forest的每个不同的实例用表中一个行区分。每一行被称为一个元组（tuple），表中行或列出现的顺序并不重要。因此，关系是一个无序的元组集合。表名与列名合在一起构成了关系模式（relation schema），行（或元组）的集合称为关系实例(relational instance)。列的数目称为关系的度（degree）。Forest是一个三度的关系。类似地，关于不同的林分和河流的数据也可以组织在不同的表中。

属性可以取哪些值？在传统的数据库应用中，称为域（domain）的属性的数据类型是有限制的，其中包括整型、浮点型、字符串、日期型以及其他的域。此外，传统数据库不支持用户自定义的数据类型。在Forest表中，属性*name*能很好地满足这种有限集合的要求，但属性*elevation*和*geometry*却不能满足。这就是说传统的关系数据库技术难以满足SDB的原因之一。尽管如此，我们将展示如何将空间信息映射到关系数据模型中。在对象-关系数据模型中，有理由假设添加新的基本域或者如OGIS规范所指定的数据类型，这正是我们要在下一节讨论的内容。

为了确保数据的逻辑一致性，必须维持关系模式上的某些约束。这些约束包括：码约束，实体完整性（entity integrity）约束和参照完整性(referential integrity)约

束。码约束规定每个关系必须有一个主码（primary key）。码是关系属性的一个子集、码值在整个关系的元组中是唯一的。一个关系中可能有很多码，用来标识关系中元组的码称为主码。实体完整性约束规定了主码不能取空值。设置该约束的理由是显而易见的：如果主码可以为空值的话，将无法用来唯一地识别元组。不同关系之间逻辑上的一致性联系可通过实施参照完整性来维持。为了解释参照完整性，我们首先介绍外码（foreign key）的概念。外码是一个关系的属性集，这个关系被复制到另外一个关系中。参照完整性约束规定：外码的属性值要么是另一关系的主码值，要么为空值。如果一个关系包含外码，则称该关系参照（refer）另一关系。

2.2.3 将ER模型映射到关系模型

许多软件包（也称为CASE工具）可以将ER模型转化为关系模式。这样的软件包有ERwin、Oracle Designer 2000和Rational Rose等等。这种转化工具让数据库设计者们在设计概念数据模型时可以把重点放到应用领域的需求上。要不是因为有空间属性，ER模型就可以无缝地、直观地映射成关系模型。整个过程有6个基本步骤：

1) 将每个实体映射成一个单独的关系。实体的属性映射成关系的属性。类似地，实体的码映射成关系的主码。图2-4的ER图所对应的关系模式如图2-5所示。

2) 对于基数为1：1的联系，我们将任一实体的码属性作为其他关系的一个外码。例如，关系Manager有一个外码属性，它对应于Forest的主码*name*。

3) 如果联系的基数是*M*：1，就把“1”侧的关系的主码作为“*M*”侧关系的外码。例如，关系Fire-Station用关系Forest的主码作为其外码。

4) 对基数为*M*：*N*的联系，则处理方式完全不同。每个*M*：*N*联系被映射成一个新的关系。关系的名称就是联系的名称，而关系的主码由参与实体的主码对组成。如果联系有属性的话，那它就成为新关系的属性。例如，Supplies_Water_To是Facility实体和River实体之间的一个*M*：*N*联系，河流和设备的名字组成主码，而属性*volume*成为新表的一个属性。注意，*M*：*N*的空间联系Road-Crosses-River变成了一个新表Road-Crosses-River。

Forest-Stand

Stand-id	Species	Forest-name
(Integer)	(varchar)	(varchar)

Fstand-Geom

Stand-id	Polygonid
(Integer)	(Integer)

River

Name	Length
(varchar)	(Real)

River-Geom

Name	Lineid
(Integer)	(Integer)

Road

Name	NumofLanes
(varchar)	(Integer)

Road-Geom

Rname	Lineid
(varchar)	(Integer)

Facility

Name	Forest-name	Forest-name-2
(varchar)	(varchar)	(varchar)

Facility-Geom

Name	Pointid
(varchar)	(Integer)

Forest

Name
(varchar)

Forest-Geom

Name	Polygonid
(varchar)	(Integer)

Fire-Station

Name	ForName
(varchar)	(varchar)

Fstation-Geom

Name	Pointid
(varchar)	(Integer)

Supplies_Water_To

FacName	RivName	Volume
(varchar)	(varchar)	(Real)

Road-Access-Forest

RoadName	ForName
(varchar)	(varchar)

Manager

Name	Age	Gender	ForName
(varchar)	(Integer)	(varchar)	(varchar)

图2-5 州立公园例子的关系模式

5）对于多值属性，创建一个具有两个列的新的关系：一列对应该多值属性，

另一列对应拥有该多值属性的实体的码。多值属性和对应实体的码合起来构成新关系的主码。例如，Forest-Stand实体有一个多值属性*polygonid*，它是一个标识号（整数），用于标识所在城市的几何位置。*polygonid*之所以是一个多值属性，是因为Forest-Stand可能会跨越两个分离的区域（例如，一条公路可能穿过一个林分）。因而，我们得到一个关系Fstand-Geom。类似地，还有关系Forest-Geom、River-Geom、Road-Geom、Facility-Geom和Fstation-Geom。

6) *elevation*这个属性需要用一种截然不同的方式进行处理。首先，我们已经指出，*elevation*是一个多值属性，因而显然需要一个新的名为Elevation的关系，新关系的属性是*Forest_Name*、*Elevation*和*Pointid*，如图2-6所示。*elevation*的属性记录了森林在*Pointid*位置的高度。在这个表中，所有三个属性共同构成了主码。

Polygon

Polygonid	Seq-no	Pointid
(Integer)	(Integer)	(Integer)

Line

Lineid	Seq-no	Pointid
(Integer)	(Integer)	(Integer)

Point

Pointid	Latitude	Longitude
(Integer)	(Real)	(Real)

Elevation

Forest-name	Pointid (F.K.)	Elevation
(varchar)	(Integer)	(Real)

图2-6 point、line、polygon和elevation的关系模式

空间表

在关系模型中，ER图的空间属性和空间变化属性必须用特殊方式进行处理。新的域（例如空间对象）被表示为新的关系。原有关系的主码成为新关系的外码，新关系所表示的实体带有采用新域的属性。如前所述，*pointid*、*lineid*和*polygonid*是一些新的域，可以作为独立的关系进行建模。对应这些属性，分别有关系：Point、Line以及Polygon（见图2-6）：

1) Point表有三个属性：*pointid*、*latitude*和*longitude*。尽管还有许多其他参照系，但经度-纬度参照系是人们最熟悉的，而且所有其他参照系都可由它导出。

2) 两点确定一条直线段，因此，Line表的*pointid*属性是对应Point表的外码。

*seq-no*属性表示点的序号，这些点组成一条由*lineid*标识的线。

3) Polygon表与Line表相似，但它多了一条约束，即首尾点的序号指的是同一个*pointid*。

2.3 趋势：扩展ER模型表达空间概念

前面叙述了将ER图的空间属性转化为空间表，但这种转化并未充分利用2.1.3节所描述的空间数据类型，它只是把空间属性作为一般的非空间属性对待。这里，我们介绍一个新的趋势：用象形符号扩展ER图，以便专门处理空间数据类型。这将减少ER图以及所产生的关系模式的复杂度，同时改进空间建模的质量。空间联系（例如Road-Crosses-River）就可以从ER图中省略，用隐式的方式表示。关系模式中的表达多值空间属性的关系和M：N空间联系也就不需要了。

如前面所讨论的，至少在直观上，ER模型不能表达空间建模中的特定语义。具体来说，ER模型的不足之处在于：

1) ER模型的最初设计隐含了基于对象模型的假设。因此，场模型无法用ER模型进行自然的映射。

2) 在传统的ER模型中，实体之间的关系由所要开发的应用来导出，而在空间建模中，空间对象之间总会有内在的联系。例如，前面讨论的所有拓扑关系都是两个空间实体之间联系的有效实例。如何将这些联系整合到ER模型中，而又不使ER图变得复杂呢？

3) 建模空间对象所使用的实体类型和“地图”的比例尺有关。一个城市是用点还是用多边形表示和地图的分辨率有关。在概念模型中，如何表达同一个对象的多种表现形式？

用象形图扩展ER模型

为了使空间应用的概念建模更加简单和直观，提出了许多对ER模型进行扩展的方法。其主要思想是增加某种结构来接受和表达空间推理的语义，同时保持图形表示的简洁性。最近，提出了用象形图（pictogram）来注释和扩展ER图的方法。

空间联系（包括拓扑的、方位的和度量的联系）隐含在任何两个具有空间成分的实体之间。例如，在实体Forest和River之间很自然会考虑拓扑关系——*cross*。在ER图中包含这种*cross*联系并不能转达更多有关该应用建模的结构信息。

我们将说明如何用象形图来表达空间数据类型、比例尺以及空间实体的隐含关系的。我们将以BNF范式（Bachus-Naur form）的语法符号来表示象形图的扩展，当然，这里并不要求一定要熟悉BNF的符号来理解随后的内容。有关这类语法符号的信息可以在任何介绍编译器的标准计算机科学的教材中或许多参考书中找到。

1. 实体象形图

(1) 象形图

象形图是一种将对象插在方框内的微缩图表示，这些微缩图用来扩展ER图，并插到实体矩形框中的适当位置。一个象形图可以是基本的形状，也可以是用户自定义的形状。

<象形图> → <形状>
→ *
→ !

象形图的语法

(2) 形状

形状是象形图中的基本图形元素，它代表着空间数据模型中的元素。一个模型元素可以是基本形状、复合形状、导出形状或备选形状。许多对象具有简单的基本形状。

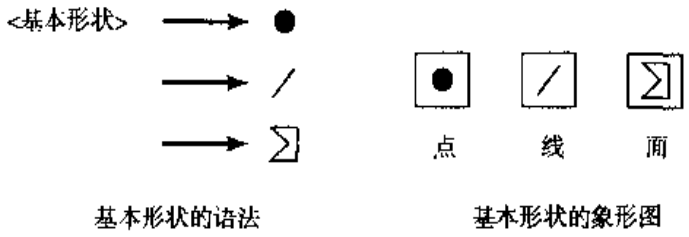
<形状> → <基本形状>
→ <复合形状>
→ <导出形状>
→ <备选形状>

形状的语法

(3) 基本形状

在一个矢量模型中，基本元素有点、线和多边形。在一般的应用中，大多数空

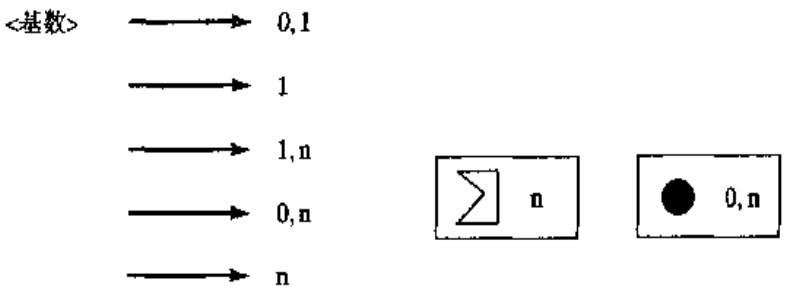
间实体是用简单形状来表示。在森林的例子中，我们把设施表示成点（0维），把河流或道路网表示成线（1维），把森林区域表示成多边形（2维）。



基本形状的语法　　基本形状的象形图

(4) 复合形状

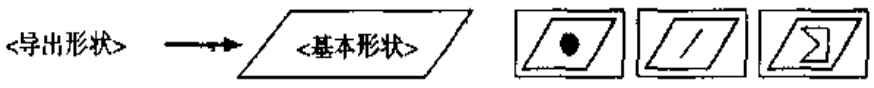
为了处理那些不能用某个基本形状表示的对象，我们定义了一组聚合的形状，并用基数来量化这些复合形状。例如，河流网可以用线的象形图的连接表示且其基数为n。类似地，对于一些无法在某个给定比例尺下描绘的要素，我们用0作为其基数。



基数的语法　　使用了基数的多重形状的象形图

(5) 导出形状

如果一个对象的形状是由其他对象的形状导出的，那么就用斜体形式来表示这个象形图。例如，我们可以从美国的州界形状导出美国的形状。



导出形状的语法　　导出形状的象形图

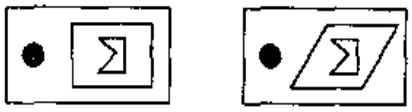
(6) 备选形状

备选形状可以用于表示某种条件下的同一个对象。例如，根据比例尺，一条河流可以表示成一个多边形或一条线。

<备选形状> ⟶ <基本形状> <导出形状>

⟶ <基本形状> <基本形状>

备选形状的语法



备选形状的象形图

(7) 任意形状

对于形状的组合，我们用通配符（*）表示，它表示各种形状，例如，一个灌溉网是由泵站（点）、水渠（线）以及水库（多边形）所组成的。

任意可能的形状

(8) 用户自定义形状

除了点、线和多边形这些基本形状外，用户还可以定义自己的形状。例如，为了表达更多的信息，用户可能更愿意使用感叹号之类的象形图来表示灌溉网。

用户自定义形状

2. 联系象形图

联系象形图用来构建实体间联系的模型。例如，*part-of*用于构建道路与路网之间联系的模型，或是用于把森林划分成林分的建模。

Part_of（网络） Part_of（分区）

联系的象形图

使用象形图扩展的ER图见图2-7。其中，Facility和Fire-Station实体用点的象形图表示，River和Road表示成线的象形图，而Forest和Forest-stand用多边形的象形图表示。Forest与Forest-stand之间的*part_of*联系在图中表示出来。这张图清楚地反映出象形图增强了ER图对空间语义的表达能力。

part_of（分区）象形图暗含有3个空间完整性约束：

1) forest_stand在空间上彼此“分离”，即空间中任意一点至多属于一个

forest_stand。

2) forest_stand在空间上位于森林“内部”，是森林的一部分（part_of）。

3) 所有forest_stand的几何并集在空间上“覆盖”它们所属的森林。

这些空间完整性约束描述了空间的集合分区（set-partition）语义。

比较图2-4与图2-7，可以看出象形图增强ER模型的优势。值得注意的是，图2-7并不显得杂乱，因为这里只有很少的显式联系和属性。空间联系和属性是隐含的。其次，图2-7显示了在空间联系上的更多的信息。例如，尽管图2-7没有显式列出“河流穿过森林”和“消防站在森林之中”这些联系，但是从图中可以看出这些隐

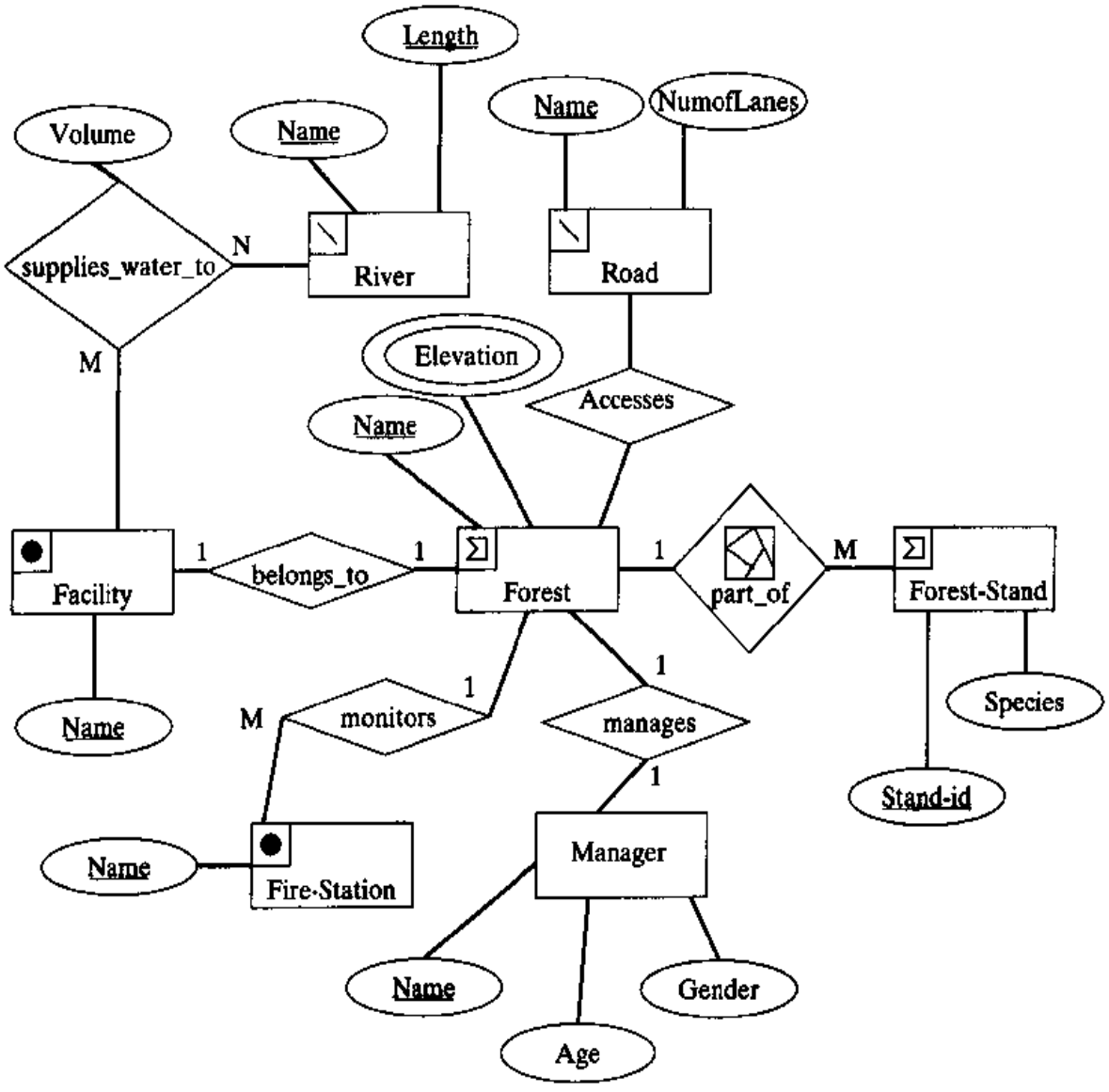


图2-7 州立公园例子的带象形符号的ER图

含的联系。part_of（分区）象形图所暗含的空间完整性约束也是原来没有的。最后，图2-7的关系模式要比图2-4的关系模式更为简单，由M：N的空间联系生成的关系和空间数据类型都被省略。

2.4 趋势：用UML构建面向对象数据模型

C++和Java这类面向对象程序设计语言的流行促进了OODBMS的发展。OODBMS日益流行的原因是，将概念数据库模式直接映射到面向对象的语言中可以减低阻抗失配。阻抗失配是指一个层次上的模型转化到另一层次上的模型（例如，从ER模型映射为关系数据模型）时所遇到的困难程度。

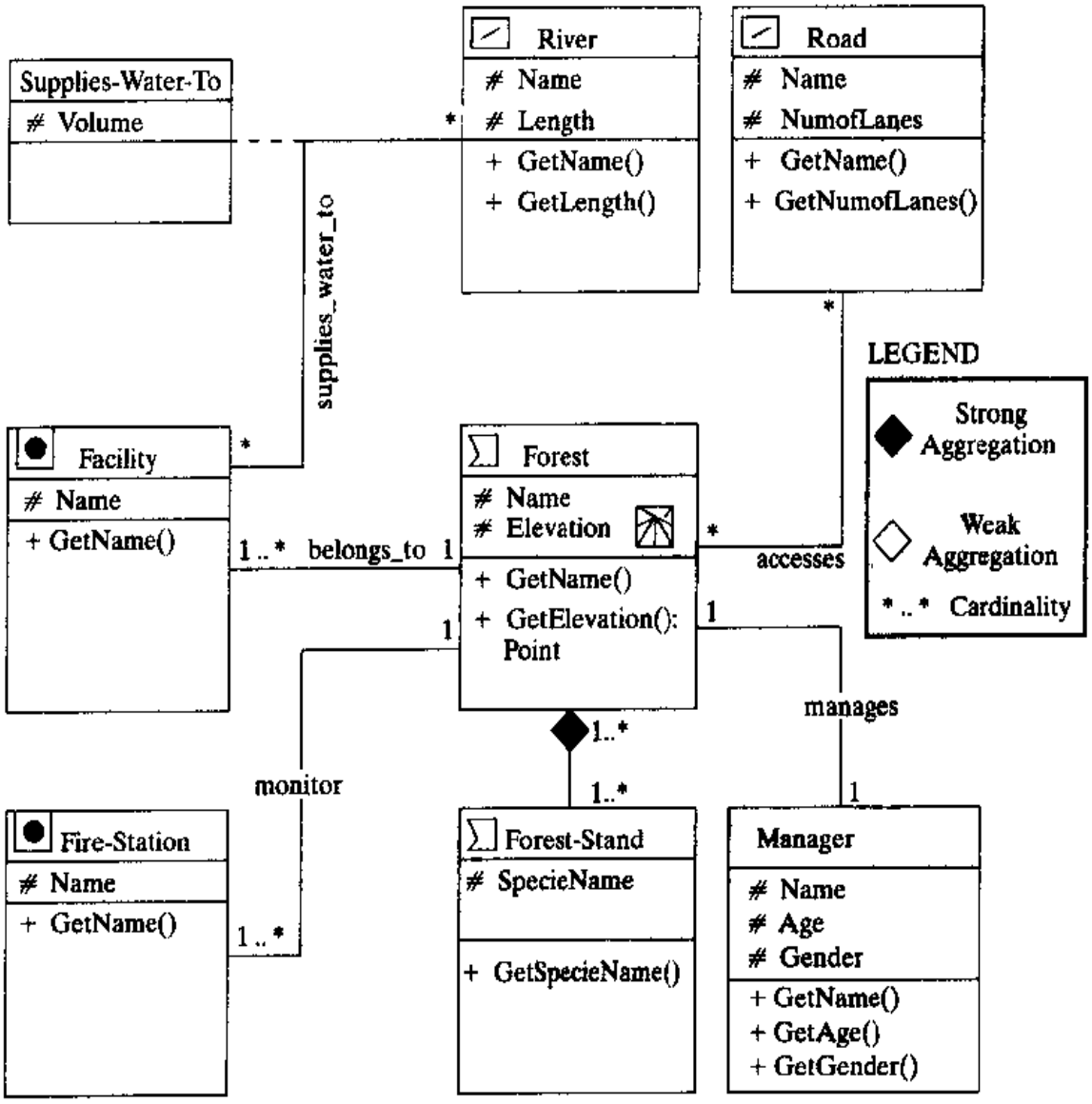


图2-8 State-Park例子的UML类图

UML［Booch et al.，1999］是用于面向对象软件设计的概念层建模的新兴标准之一。它是一种综合型语言，用于在概念层对结构化模式和动态行为进行建模。就数据库设计而言，我们只对系统静态结构的建模感兴趣。现在，我们来使用类图而不是ER图来建模。图2-8是图2-4所示*State-Park*例子的一个等价UML类图(UML class diagram，UMLCD)表示。

下面，我们结合State-Park的例子来扼要描述UMLCD的符号：

- 类（class）是所有在应用中具有相同性质的对象的封装，它等价于ER模型中的实体。例如，设施就是一个类，至少它抽象地包含了州立公园中的功能性单元，例如休息室、营地和旅游公司。正如对ER图所做的那样，我们也能对类图进行象形图扩展。这样就可以清楚看出Facility类的实例是空间对象。更确切地说，它们的空间几何形状用点来表示。这样一来，有关点对象的所有空间属性、联系和操作都适用于Facility类。
- 属性（attribute）描述类的对象。与ER符号不同，UMLCD没有符号表示码属性，这是因为在面向对象（OO）系统中，所有对象都有一个系统生成的唯一标识。属性还有一个与之关联的作用域或可见性。作用域规定是从类的内部还是外部访问属性，这是控制数据库设计中模块化程度的关键。作用域有三个级别，每一个级别都有特定的符号：

1) +（公有的）：属性可以被任意类访问和操纵。

2) -（私有的）：只有属性所在的类才可以访问这个属性。

3) #（受保护的）：从父类派生的类（“子类型化”）可以访问该属性。State-Park例子中的属性全部选用受保护的属性。

- 方法（method）是一些函数，它们是类定义的一部分，用来修改类的行为或状态。类的状态由属性的当前值来体现。在面向对象的设计中，属性只能通过方法来访问。在Facility类中，通过方法*GetName()*来访问属性*name*。
- 关系（relationship）将一个类与另一个类或者它自己相联系。它类似于ER模型中的联系。对于数据库建模，UMLCD有三种重要的关系：聚合、泛化和关联。

1) 聚合（aggregation）是UMLCD中特有的概念，它描述了部分-整体关系。例如，类Forest-Stand与类Forest是部分-整体关系。一个类有时可以作为多个其他类的一部分。为了区分这两种情况，我们把前者称为强聚合，而把后者称为弱聚合。

2) 泛化（generalization）在图2-2所示的空间层次语境中得到最好说明。例如，Geometry类是其子类Point、Line和Polygon的泛化。

3) 关联（association）反映了不同类的对象之间是如何联系的。和ER模型的联系一样，如果一个关联涉及两个类，那它就是二元的；如果涉及三个类，那么它是三元的（ternary）。关系*supplies_water_to*就是Facility类与River类之间的一个关联。注意，图2-8中有一个Supplies_Water_To类，它通过虚线与关联*supplies_water_to*相连。这是因为该关联本身就有一个*Volume*属性。

比较ER与UML

ER和UMLCD的主要概念在表2-3中作了归纳。前面的讨论表明，在这两个建模范型之间有很多相似之处，但也有一些差异。

表2-3 ER与UMLCD中的概念

ER中的概念	UMLCD中的概念
实体	类
属性	属性
码属性	
	方法
继承	继承
聚合	聚合
弱实体	

ER图中的“实体”概念与UML类图中的“类”概念相近。实体和类都有属性，并且都参与到诸如继承和聚合这样的联系中。但类除属性之外还包括方法。方法是封装了逻辑和计算代码的过程或函数，而ER模型的实体一般不会包含方法。类图可以对类的属性及方法进行建模，这些类来自于面向对象的程序设计语言，例如Java。

UMLCD中有方法的概念，它可用来修改类的状态和行为。

ER模型的有些概念在类图中是不可用的。其中之一是弱实体的概念。弱实体的唯一标识必须依赖另一个实体。我们来看由用户名和域名组成的电子邮件地址。例如，对于root@cs.umn.edu来说，“root”是用户名，而“cs.umn.edu”是域名。显然，用户名root不是唯一的，它要依赖域名来组成唯一标识。由于类图和面向对象的范型都假定所有的对象都有唯一的标识，因而不支持弱实体。

在ER中，每个实体的实例由一个用户定义的显式标识来描述，而在UMLCD中，则假定系统会为每个类的实例生成唯一的标识。

2.5 小结

空间信息建模可以分为两大类：场模型和对象模型。场模型对形状不定的空间结构的建模非常有用，这种结构的空间变化是平滑的，例如瀑布、云层、烟和悬浮微粒等。对象模型适用于对离散的、可识别的结构进行建模，例如对国家进行建模。

与每一种模型相关联的是一组刻画模型的操作。与场模型相关的操作可以分为三类：局部的、聚焦的和区域的。与对象模型相关的操作可以按其数学性质划分为：面向集合的、欧氏的、拓扑的和度量的。

ER模型广泛用于数据的概念层建模。虽然ER模型的设计并不是专用于某种形式的数据，但它主要适用于简单的数据类型和结构。ER模型可以通过一系列次序清晰的步骤映射成数据库中的关系模型。

ER模型和关系模型必须进行扩展才能整合空间数据的空间特性。ER模型可通过引入象形图来进行扩展，象形图用符号表示各种空间数据类型和关系。类似地，关系模型也可通过增加新的数据类型及其相关的操作来进行扩展。例如，空间应用中有三个众所周知的数据类型：点、线和多边形。作为OODBMS广泛使用的概念建模语言，也可通过象形图来扩展UML，以便对空间数据建模。

2.6 参考书目

2.1 对象-场的二分法是GIS和空间建模领域中颇有争议的话题。大多数采集的数据适用于场类型的分析，而数据库建模则偏爱对象的观点。有关对象-场的争论历史和心理背景，可参见［Couclelis，1992］。

2.1.1 本章是依据［Worboys，1995］来讨论基于场的建模的。

2.1.2 基于对象的空间数据模型的拓扑观点是由［Egenhofer，1991a］提出的，并由［Clementini et al.，1993］进行扩展。

2.1.4 关于在方位上的对象建模的进一步讨论可参见［Shekhar et al.，1999a］。

2.2 数据库设计及其相关的问题可以在任何介绍数据库的教材中找到。例如，可参阅［Elmasri and Navathe，2000; Ramakrishnan，1998］。

2.3.1 我们关于ER模型的象形图扩展的讨论是依据［Shekhar et al.，1999c］进行的。这篇论文通过用空间象形图对ER模型进行扩展（PEER），对空间数据库进行概念数据建模，并且提供了一种将PEER转化为逻辑数据模型的映射方法。

2.4 空间数据的ER建模扩展是由［Hadzilacos and Tryfona，1997；Shekhar et al.，1997］提出的，但尚未获得人们的一致认可。采用基于UML表达方式进行地理空间应用数据库建模的进一步讨论见［Brodeur et al.，2000］。

2.7 习题

1. 讨论“场模型符合科学的严密性和理性，而对象模型则更符合人的思维”。
2. 在天气预报中，气压、温度和风通常采用场来建模。但公众更愿收到离散的实体方面的信息。例如，“气流的前锋将减弱”或“高气压将减弱”［Goodchild，1986］。你能够举出另一个场-对象二分法的例子吗？
3. 湖泊有时可建模成对象。你能够给出一个用场模型对湖泊进行建模的例子吗？湖泊的边界是良定的吗？
4. 匹配下面的列：

标称的	摄氏温度
有序的	绝对温度

区间的　　社会安全号

比率　　色谱

5. 设计一个ER图来表示World的地理和政策要素。World包含三个实体：国家、城市和河流。根据这三个实体回答下列问题：
   - 如果两国之间有外交关系，则通过联系*diplomatic_ties*将它们关联起来。请将该联系在ER图上画出并标出其基数。
   - 水是21世纪经常讨论的热门话题。如果两国共有一条河，但这并不代表两国都可以使用该河的河水。试表示实体Country和River之间的联系*river_owned_by*。
   - 在众多实例中，一国的首都并非该国的商业中心。例如：纽约是美国的商业中心，上海是中国的商业中心，悉尼是澳大利亚的商业中心。试表示实体City和Country之间的联系*business_capital*。
6. 考虑用OO语言（如Java）来描述OGIS层次结构。你该如何对继承进行建模？例如，类MultiPoint继承了类Point和类GeometryCollection的性质。你又将如何构建关联和基数约束的模型？在哪里会用到抽象类？
7. 研究利用UML［Booch et al.，1999］进行数据建模。哪些UML图是与数据建模相关的？它们在哪些地方优于ER？你将如何在UML中表达主码、外码、实体、联系、基数约束、参与约束和象形图这些概念？
8. 使用UML对State-Park的例子进行建模。
9. 将下列的操作分别归类到局部操作、焦点操作或区域操作中：a) 坡度，b) 被雪覆盖的公园，c) 新设施的地点选择，d) 平均人口以及e) 人口总数。
10. 许多OGIS中的空间数据类型参照的是平面世界，这包括线串和多边形。但是，当对地球表面上较长河流或面积较大国家的形状进行建模时会有较大的误差。请找出一些球面上的空间数据类型来减小球面上大对象的误差。哪些空间关系（如拓扑的、度量的）会受平面近似的影响？
11. 回顾关于用Java程序实现的查询："找出在Maple营地方圆10英里以内所有的旅游公司"。FacilityDemo类中的*main()*方法随着旅游公司数量的增加，其执行速度线性地变慢。借助第1章中的平面扫描算法的思想，设计

新的算法来加快搜索速度。

12. 开发一个转化规则，将概念数据模型（如ER图）中的象形图转化为OGIS的空间数据类型，并能够将其嵌入逻辑数据模型（比如Java、SQL）或物理模型。

13. 考虑一些与图2-3中那8个矩阵不同的3×3的布尔矩阵，为什么将这些矩阵定义为$\Re^2$上的2维表面中的拓扑关系是没有意义的？举出一个3维表面上的拓扑关系的例子，且它不能用图2-3中的8个操作来建模。

14. 空间数据有时候可以看作多维数据的特例，多维数据是包含在多维空间（如温度、压力、音量和时间）中的数据。将欧氏空间中的空间数据（如经度、纬度）和其他多维数据进行比较。

15. 将OGIS类型层次（2.2）中的简单空间数据类型与2.1.6节定义的象形图进行比较。给出下列OGIS数据类型所对应的象形图：point、line string、polygon、multipoint、multiline string和multipolygon。

16. OGIS和象形图都包括6种空间数据类型：point、line string、polygon、multipoint、multiline string和multipolygon。

    (a) 简单说明这6个基本数据类型组成的集合在基本的几何操作（例如几何并、几何交、几何差等）下是封闭的。如果数据类型的一个集合S中的元素在操作O的作用后所得的结果数据类型仍然在S中，则称S在O下是封闭的。

    (b) 对查询语言来说，为什么闭包是很有意义的性质？

17. 考虑6个空间数据类型组成的基本集合。给世界/国家地图中，选择最“自然”的数据类型来表示下列实体。假设实体有：国家、河流、湖泊、公路和城市。当地图的比例尺发生变化时，空间数据类型的选择会如何变化？你如何在象形图中表示这种对比例尺的依赖性？

18. 研究图2-4中的ER图。对于如下的问题它作了哪些规定：

    (i) 一个管理员可以管理几片森林？

    (ii) 一个管理员可以管理多少片林分？

    (iii) 多少消防站可以监控一片森林？

(iv) 多少设施可以属于一片森林?

(v) 河流和森林之间的空间关系是什么?

(vi) 同一片森林中的各林分间的关系是什么?

19. 研究图2-7中的ER图，重新回答第18题中列出的问题，答案会有所不同吗？写出图2-7所对应的关系模式。

20. 研究图2-5中的关系模式，并且找出M：N联系的所有缺失的表。同时找出图2-4中1：1和1：N联系所缺少的外码。图2-6中的空间表是否还应包含额外的表来表示点集合、线串集合或多边形集合?

21. 用象形图修订图2-7中的ER图，使之满足如下需求：

(i) 公路可以采用线串（表示道路中心线）或多边形（表示占用的土地块）来表示。

(ii) 设施可以采用点集合或多边形集合来表示。

(iii) 公路是道路网的一部分。

(iv) 管理员有一个邮政地址和一个地图位置。

22. 研究公众领域中数字化的空间数据集（如人口普查，TIGER文件）。为这些数据集开发一个概念数据模型（如ER模型和ER图）。

23. 假设某人想修改State-Park例子中的数据模型，以便一个消防站可以监控多片森林。对ER图（图2-4和图2-7）以及关系模式（图2-5）进行相应的修改。

24. 在用镜子成像时，对象的左右是相反的，但上下则不会。使用绝对的或相对的简图加以解释。

第3章
空间查询语言

作为与数据库交互的主要手段，查询语言是数据库管理系统的一个核心要素。SQL是用于关系数据库管理系统（RDBMS）的一种常见的商业查询语言。它在一定程度上基于形式化查询语言——关系代数(RA)，并且易于使用，既直观又通用。由于SDBMS是一种扩展的DBMS，所以它既可以处理空间数据，也可以处理非空间数据，所以，很自然地希望能扩展SQL来支持空间数据。

正如上一章所指出的，关系模型在高效处理空间数据方面有一定的局限性。空间数据是“复杂的”，它由多边形、线、点混合而成，而关系模型比较适合于处理简单的数据类型，如整型、字符串、日期类型等。

面向对象程序设计中的一些概念，如用户自定义的类型、属性以及方法的继承等，非常适合于处理复杂数据的建模。随着关系模型和SQL的广泛应用，人们把简

单数据类型和面向对象模型的功能结合起来，产生了一种新的“混合”（hybrid）范型的数据库管理系统——对象关系数据库管理系统（OR-DBMS）。

利用OR-DBMS带来的一个必然结果就是要求对SQL进行扩展，使其支持对象的功能。因而，产生了SQL在OR-DBMS上的标准：SQL3。由于我们所要处理的是空间数据，所以这里只考察SQL3在空间方面的扩展和函数库。

空间数据的一个独有特性是，它与用户交互的“自然”媒介是可视化的，而非文本形式的。因此，任何空间查询语言都应该支持复杂的图形化可视组件。不过我们已经说过，本书只关注SQL的非图形化空间扩展。以下是本章各节的简要介绍。3.1节介绍World（世界）数据库，本章中的所有查询都是基于这个数据库的。3.2节和3.3节分别对关系代数(RA)和SQL进行简要介绍。3.4节主要讨论SQL对于空间扩展的需求，还介绍OGIS标准对地理空间数据进行的SQL扩展。3.5节介绍如何用OGIS扩展SQL来完成一些常见的空间查询。3.6节介绍了SQL3以及Oracle8对SQL3子集的实现。

3.1 标准数据库查询语言

用户利用查询语言与存储在数据库管理系统中的数据进行交互。与传统程序设计语言不同的是，数据库查询语言比较容易掌握和使用。在这一节我们介绍两种查询语言。首先介绍关系代数(RA)，这是一种形式化的语言，并未在商业数据库上实现。第二种是结构化查询语言（SQL），这是在数据库中使用最为广泛的语言。关系代数的重要性在于它构成了SQL的核心。

World数据库

为了更好地介绍关系代数和SQL，我们引入了一个示例数据库——World数据库，本章的例子和练习中都会用到这个数据库。它由三个实体组成：Country、City和River。图3-1和表3-1分别是这个数据库的带象形图的ER图和示例数据表。数据库的模式如下所示。需要指出的是，ER图中带下划线的属性是主码，例如，Name就是Country表、City表和River表的主码。

Country(*Name*: varchar(35), *Cont*: varchar(35), *Pop*: integer,
GDP:Integer, *Life-Exp*: integer, *Shape*:char(13))
City(*Name*: varchar(35), *Country*: varchar(35), *Pop*: integer,
Capital:char(1), *Shape*:char(9))
River(*Name*: varchar(35), *Origin*: varchar(35), *Length*: integer,
Shape:char(13))

实体Country有六个属性。国家的名字(*Name*)和该国家所属的大洲 (*Cont*)是字符串类型，最大长度为35。人口(*Pop*)和国民生产总值(*GDP*)是整型，GDP是一个国家在一个财政年度中创造的物品和服务的价值的总和。*Life-Exp*属性表示一个国家居民的预期寿命，以年计。对*Shape*属性需要作些解释，*Shape*属性对应表3-1中的*Shape*列，它代表国家的几何形状。在关系数据库（其数据类型受到限制）中，*Shape*属性就是一个指向*Shape*表的外码。在对象关系数据库或者面向对象数据库中，*Shape*属性的类型则是多边形这种抽象数据类型(abstract datatype，ADT)。因为目前我们主要介绍基本的RA和SQL，所以在3.4节之前都不会对*Shape*属性进行查询。

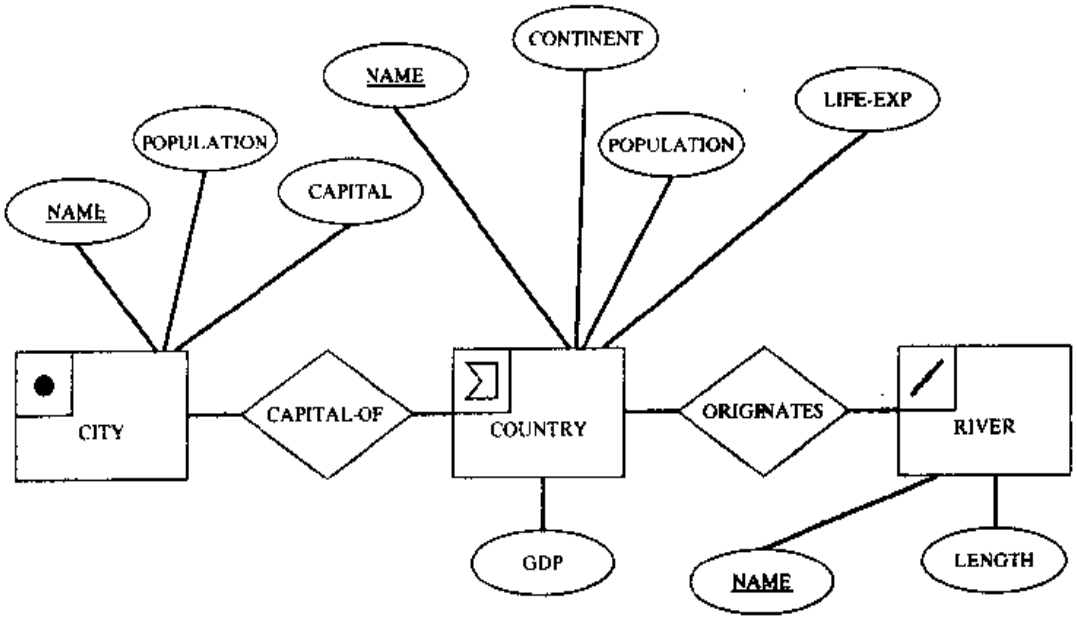


图3-1 World数据库的ER图

表3-1 World 数据库中的表

COUNTRY	Name	Cont	Pop (millions)	GDP (billions)	Life-Exp	Shape
	Canada	NAM	30.1	658.0	77.08	Polygonid-1
	Mexico	NAM	107.5	694.3	69.36	Polygonid-2
	Brazil	SAM	183.3	1004.0	65.60	Polygonid-3
	Cuba	NAM	11.7	16.9	75.95	Polygonid-4
	USA	NAM	270.0	8003.0	75.75	Polygonid-5
	Argentina	SAM	36.3	348.2	70.75	Polygonid-6

a) Country表

（续）

CITY	Name	Country	Pop (millions)	Capital	Shape
	Havana	Cuba	2.1	Y	Pointid-1
	Washington, D.C.	USA	3.2	Y	Pointid-2
	Monterrey	Mexico	2.0	N	Pointid-3
	Toronto	Canada	3.4	N	Pointid-4
	Brasilia	Brazil	1.5	Y	Pointid-5
	Rosario	Argentina	1.1	N	Pointid-6
	Ottawa	Canada	0.8	Y	Pointid-7
	Mexico City	Mexico	14.1	Y	Pointid-8
	Buenos Aires	Argentina	10.75	Y	Pointid-9

b) City表

RIVER	Name	Origin	Length (kilometers)	Shape
	Rio Parana	Brazil	2600	LineStringid-1
	St. Lawrence	USA	1200	LineStringid-2
	Rio Grande	USA	3000	LineStringid-3
	Mississippi	USA	6000	LineStringid-4

c) River表

关系City有5个属性：*Name*（名字）、*Country*（国家）、*Pop*（人口）、*Capital*（首都）和*Shape*（形状）。*Country*属性是一个指向Country表的外码。*Capital*是一个定长的字符串，其长度为1，表示这个城市是否为国家的首都。*Shape*属性是指向一个点形状表的外码，同前面提到Country关系时一样，在掌握用于SQL3的OGIS数据类型之前，我们不会查询*Shape*所在列。

关系River有4个属性：*Name*(名字)、*Origin*(源头)、*Length*(长度)和*Shape*（形状）。*Origin*属性是指向Country关系的外码，指明河流发源于哪个国家。*Shape*属性是指向线串形状表的外码，*Shape*属性中的几何信息不足以确定河流发源于哪个国家。表间属性重名的问题可以通过在属性名前加表名和点来解决，即表名.属性名。例如，Country.Name、City.Name、River.Name唯一地标识了不同表中的Name属性。我们还需要关于河流流向的信息，它在第7章的空间网络查询中是很重要的。

3.2 关系代数

关系代数(RA)是与关系模型相关联的形式化查询语言。代数（algebra）是一个数学结构，它由两个不同的元素集合(Ω_a, Ω_o)组成。Ω_a是运算对象（operand）的集合，Ω_o是运算（operation）的集合。代数必须满足许多公理，其中最重要的就是对运算对象进行运算后得到的结果必须仍属于Ω_a。代数的一个简单例子是整数集合：运算对象是整数，运算包括加法和乘法。在第8章中，我们将讨论关于栅格和影像

对象的代数。

关系代数只有一种类型的运算对象和六种基本运算，这个运算对象是关系（表），六种运算包括：选择（select）、投影（project）、并（union）、笛卡儿积（cross-product）、差（difference）以及交（intersection）。下面将详细介绍这些基本运算。

3.2.1 选择和投影运算

关系代数提供了两个运算：选择和投影，用来操纵单个关系中的数据。选择运算检索关系表中行的子集，而投影运算抽取列的子集。例如，用下面的关系代数表达式可以列出Country表中所有位于北美洲（NAM）的国家：

$$\sigma_{\text{cont=NAM}}(Country)$$

这个运算的结果如表3-2a所示。通过比较选择运算符，选择运算σ把指定的行检索出来，本例中的选择运算符是*cont = 'North-America'*。输入关系的模式并不会被选择运算符所修改。选择运算的形式化语法如下所示：

$$\sigma_{<\text{选择运算符}>}(\text{关系})$$

投影运算π可以把关系中所有行的某些列提取出来，例如，用以下表达式可以把Country表中所有国家的名字检索出来：

$$\pi_{\text{Name}}(Country)$$

表3-2 关系代数中两个基本运算（选择和投影）的运算结果

Name	Cont	Pop (millions)	GDP (billions)	Life-Exp	Shape
Canada	NAM	30.1	658.0	77.08	Polygonid-1
Mexico	NAM	107.5	694.3	69.36	Polygonid-2
Cuba	NAM	11.7	16.9	75.95	Polygonid-4
USA	NAM	270.0	8003.0	75.75	Polygonid-5

a) 选择

Name
Canada
Mexico
Brazil
Cuba
USA
Argentina

b) 投影

Name
Canada
Mexico
Cuba
USA

c) 选择和投影

投影运算的形式化语法如下所示：

$$\pi_{<\text{属性列表}>}(\text{关系})$$

我们可以把选择运算和投影运算结合起来，利用下面表达式可以列出位于北美洲（NAM）的国家的名字，结果如表3-2c所示。

$$\pi_{\text{Name}}(\sigma_{\text{Cont=NAM}}(\textit{Country}))$$

3.2.2 集合运算

从最基础层面上说，关系是一个集合，因此所有集合运算在关系代数中都是有效的。集合运算可以应用在并运算相容（union-compatible）的关系之上。如果两个关系具有相同的列数，共享同样的域，同时列的顺序从左到右一致，我们就说这两个关系是并运算相容的。

- **集合并**(Union)：如果R和S是两个关系，则$R \cup S$返回R或者S中所有的元组。例如，我们可以通过集合并运算列出所有符合下列条件的国家：它们要么在北美洲，要么有一条河流发源于北美洲：

1) $R = \pi_{\text{Name}}(\sigma_{\text{Cont=NAM}}(\text{Country}))$

2) $S = \pi_{\text{Origin}}(\text{River})$

3) $R \cup S$

得到的结果关系如表3-4a所示。需要注意的是，属性*R.Name*和*S.Origin*具有同样的域，因为*S.Origin*是指Country.Name，这一点保证了R和S是并运算相容的。

- **集合差**(Difference)：$R-S$返回所有包含在R中面不包含在S中的元组。例如，可以用集合差运算来列出所有位于北美洲但不是河流（列在River表中）发源地的国家，表3-4b是应用集合差运算得到的结果。

1) $R = \pi_{\text{Name}}(\sigma_{\text{Cont=NAM}}(\text{Country}))$

2) $S = \pi_{\text{Origin}}(\text{River})$

3) $R-S$

- **集合交**(Intersection)：对于两个并运算相容的关系R和S来说，集合交运算$R \cap S$

返回所有既在R中又在S中的元组。注意，虽然集合交运算使用起来十分便利，但实际上它并不是必须的，因为它能够通过集合差运算得到，即$R \cap S = R - (R - S)$。用集合交运算可以很方便地得到既在南美洲又是河流发源地的国家，结果如表3-4c所示。

1) $R = \pi_{\text{Namc}}(\sigma_{\text{Cont=SAM}}(\text{Country}))$

2) $R = \pi_{\text{Origin}}(\text{River})$

3) $R \cap S$

- **笛卡儿积**(Cross-Product)：这个运算可以应用于任何一对关系，而不是只能应用于前面所提到的那些具有并运算相容的两个关系上。$R \times S$返回一个关系，其模式由R的全部属性加上S的全部属性组成，表3-3给出了一个抽象的例子来说明笛卡儿积。注意，我们使用了级联点注（cascading dot notation），即在属性前加了关系名和点来进行修饰，以此来区分来自两个关系的属性。

表3-3 关系R和S笛卡儿积运算

R	R.A	R.B
	A_1	B_1
	A_2	B_2

a) 关系R

S	S.C	S.D
	C_1	D_1
	C_2	D_2

b) 关系S

$R \times S$	R.A	R.B	S.C	S.D
	A_1	B_1	C_1	D_1
	A_1	B_1	C_2	D_2
	A_2	B_2	C_1	D_1
	A_2	B_2	C_2	D_2

c) $R \times S$

表3-4 集合运算的结果

NAME
Canada
Mexico
Brazil
Cuba
USA

a) 集合并

NAME
Canada
Mexico
Cuba

b) 集合差

NAME
Brazil

c) 集合交

3.2.3 连接运算

选择和投影运算用于从单个关系中提取信息，当需要在多个关系表之间进行查询时就要使用连接（join）运算。连接运算可以理解成在笛卡儿积的基础上进行选择运算。常见的连接运算是条件（conditional）连接，而条件连接中一个重要和特殊的情况是自然（natural）连接。

1. 条件连接

在两个关系R和S之间，通用的条件连接$\bowtie_c$由下式表示：

$$R \bowtie_c S = \sigma_c (R \times S)$$

条件c通常是指R和S的属性。例如，可以使用连接运算来查询人口超过墨西哥的国家(见表3-5)：

1) $R = \pi_{Name,\ Pop}(Country)$

2) $S = R$(S是R的一个副本)

3) 得到笛卡儿积$R \times S$。$R \times S$关系的模式（schema）为：

$R \times S$	R.Name	R.Pop	S.Name	S.Pop

4) 应用条件，即关系S中人口数超过墨西哥的国家：

$$U = R \bowtie S = \sigma_{(R.Name \neq \text{'Mexico'}) \wedge (R.Pop > S.Pop)}(R \times S)$$

表3-5 条件连接运算的步骤

$R \times S$	R.Name	R.Pop	S.Name	S.Pop
	⋮	⋮	⋮	⋮
	Mexico	107.5	Canada	30.1
	Mexico	107.5	Mexico	107.5
	Mexico	107.5	Brazil	183.3
	Mexico	107.5	Cuba	11.7
	Mexico	107.5	USA	270.0
	Mexico	107.5	Argentina	36.3
	⋮	⋮	⋮	⋮

a) $R \times S$的一部分

R.Name	R.Pop	S.Name	S.Pop
Mexico	107.5	Canada	30.1
Mexico	107.5	Cuba	11.7
Mexico	107.5	Argentina	36.3

b) $R \times S$上的选择操作

2. 自然连接

条件连接的一个重要特例是自然连接（natural join）。在自然连接中，两个关系的公共属性只用到相等性（equality）选择条件，而结果中只有一列来表示公共的等值连接属性。例如，可以用自然连接运算来查找河流发源地国家的人口数。下面是具体步骤：

1) 重新命名Country关系为C，River关系为R。

2) 构造笛卡儿积$C \times R$。

3) 在属性$C.Name$和$R.Origin$上对这两个关系作连接，这两个属性是属于同一个域的，即

$$C \bowtie_{C.Name = R.Origin} R$$

4) 在自然连接中，选择条件是十分明确的，因而无需在连接公式中显式地写成下标。

5) 在属性$Name$和Pop上作投影运算，得到最终结果：

$$\pi_{Name,\ Pop}(C \bowtie R)$$

3.3 SQL基础

SQL最初是由IBM开发出来的一种商业查询语言。从那时起，它就成为关系数据库管理系统(RDBMS)的标准查询语言。SQL是一种声明性语言，即用户只需描述所要的结果即可，而不必描述获得结果的过程。

SQL语言至少由两部分组成：数据定义语言（data definition language，DDL）和数据操纵语言（data modification language，DML）。DDL用于创建、删除和修改数据库中表的定义，而DML则用于查询、插入、删除、修改DDL中定义好的表中的数据。SQL还包括用于数据控制语言的其他语句。下面将给出SQL的简要介绍，我们的目标是让读者充分了解这种语言，以便他们能够理解3.4节所讨论的空间扩展。有关SQL更为详细和全面的介绍可以在任何一本数据库标准教材中找到［Elmasri and Navathe，2000;Ullman and Widom，1999］。

3.3.1 DDL

关系模式的创建以及表的添加和删除都在SQL的DDL中完成。例如，下面的SQL语句定义了3.2节所介绍的City关系模式，Country表和River表由表3-6定义。

```
CREATE  TABLE CITY (
        Name   VARCHAR(35),
        Country   VARCHAR(35),
        Pop   INT,
        Capital   CHAR(1),
```

```
Shape   CHAR(13),
PRIMARY KEY   Name )
```

CREATE TABLE语句用于定义一个关系模式中的关系，CITY是表的表名，该表有四列，必须指定每列的名字和相应的数据类型。*Name*和*Country*两个属性必须是长度不超过35个字符的ASCII字符串；*Population*是整数类型；*Capital*属性的值是字符*Y*或*N*。在SQL92中，所有的数据类型是固定的，用户不能自己定义数据类型。这里，我们未列出所有数据类型，它们可以在标准数据库教材中查到。最后，*Name*属性是关系的主码，这样，表中每一行必须有一个唯一值来对应*Name*属性。可以用DROP TABLE命令把不再会用到的表从数据库中删除。DDL的另一个重要命令是ALTER TABLE，该命令用于修改关系的模式。

表3-6 SQL语言定义的Country表和River表的模式

```
CREATE    TABLE Country (
          Name  VARCHAR(35),
          Cont VARCHAR(35),
          Pop INT,
          GDP INT,
          Life-Exp FLOAT(2)⊖
          Shape  CHAR(15),
          PRIMARY  KEY  (Name) )
```

a) Country表模式

```
CREATE    TABLE River (
          Name VARCHAR(35),
          Origin VARCHAR(35),
          Length INT,
          Shape CHAR(15),
          PRIMARY DEY  (Name) )
```

b) River表模式

3.3.2 DML

按照DDL语句定义完成了表的创建之后，就可以开始存入数据了，这项工作也称作“填充表”，由SQL的DML来完成。例如，下面语句添加一行到River表中：

```
INSERT INTO River(Name, Origin, Length)
  VALUES('Mississippi','USA', 6000)
```

如果关系中的所有属性值未被指定，则会自动用默认值代替，最常用的默认值是“NULL”。如果试图向River表中插入另一行Name = 'Mississippi'，那么DBMS将会拒绝执行，因为在DDL中已经把Name属性定义为主码。

从表中删除行的基本格式如下：

⊖ 原书中缺少该属性，但后文中会用到，故补充之。——译者注

```
DELETE FROM TABLE WHERE <条件>
```

例如，下面语句从River表中删除刚才插入的行：

```
DELETE  FROM   River
        WHERE  Name = 'Mississippi'
```

3.3.3 SQL查询的基本格式

当用DDL定义了数据库模式，并且各个表都录入了数据，就可以用SQL的查询语句从数据库中抽取数据的相关子集。SQL查询的基本语法非常简单：

```
SELECT   column-names
FROM     relations
WHERE    tuple-constraint
```

这种格式等价于关系代数中由π、σ、⋈组成的表达式。SQL的SELECT语句还有其他一些子句：关于聚集的（例如，GROUP BY、HAVING）、关于结果排序的（例如，ORDER BY），等等。SQL还支持嵌套的查询方式。下面用一组例子来详细说明。

3.3.4 SQL查询示例

这里举一些例子来说明如何用SQL表述不同类型的查询，目的是体会SELECT语句的多种风格及其强大功能。所有查询的表都出自3.1.1节所举的WORLD示例数据库，各查询的结果参见表3-7和3-8。

1) **查询**：列出CITY表中所有城市及其所属的国家。

```
SELECT  Ci.Name, Ci.Country
FROM    CITY Ci
```

注释 这条SQL语句等价于RA的投影运算。由于查询中并没有对应的RA选择运算，所以该SQL表达式少了WHERE子句。还要注意其中可选的级联点注，因为CITY表已更名为Ci，所以它的属性引用为Ci.Name和Ci.Country。

2) **查询**：列出CITY表中是首都的城市的属性。

```
SELECT  *
```

```
FROM    CITY
WHERE   CAPITAL='Y'
```

注释 这条SQL语句等价于RA的选择操作。在SQL语句中，RA的选择运算是由WHERE子句而非SELECT子句指定！SELECT中的“*”表示必须列出CITY表的所有属性。

表3-7 选择、投影、选择并投影操作

Name	Country	Pop(millions)	Capital	Shape
Havana	Cuba	2.1	Y	Point
Washington, D.C.	USA	3.2	Y	Point
Brasilia	Brazil	1.5	Y	Point
Ottawa	Canada	0.8	Y	Point
Mexico City	Mexico	14.1	Y	Point
Buenos Aires	Argentina	10.75	Y	Point

a) 查询2：选择

Name	Country
Havana	Cuba
Washington, D.C.	USA
Monterrey	Mexico
Toronto	Canada
Brasilia	Brazil
Rosario	Argentina
Ottawa	Canada
Mexico City	Mexico
Buenos Aires	Argentina

b) 查询1：投影

Name	Life-exp
Mexico	69.36
Brazil	65.60

c) 查询3：选择并投影

表3-8 查询例子的结果

Cl.Name	Co.Pop
Brasilia	183.3
Washington, D.C.	270.0

a) 查询4

Cl.Name	Cl.Pop
Washington, D.C.	3.2

b) 查询5

Average-Pop
2.2

c) 查询6

Cont	Continent-Pop
NAM	2343.05
SAM	676.1

d) 查询7

Origin	Min-length
USA	1200

e) 查询8

Co.Name
Mexico
Brazil
USA

f) 查询9

3) **查询**：列出Country表中人均寿命低于70岁的国家的属性。

```
SELECT  Co.Name, Co.Life-Exp
FROM    Country Co
WHERE   Co.Life-Exp < 70
```

注释 这条SQL语句等价于RA的πoσ操作。该例中所要投影的属性Co.Name和Co.Life-Exp由SELECT子句指定，而选择条件由WHERE子句指定。

4) **查询**：列出GDP超过1万亿美元的国家的首都和人口数。

```
SELECT  Ci.Name, Co.Pop
FROM    City Ci, Country Co
WHERE   Ci.Country = Co.Name AND
        Co.GDP > 1000.0 AND
        Ci.Capital= 'Y'
```

注释 这是一种表示连接运算的隐式方法，SQL2和SQL3也支持显式的连接运算。在本例中，City和Country两个表在它们的公共属性Ci.country和Co.name上进行匹配，并且，分别在City和Country表上指定两个选择条件。注意，由于两个关系中有同名属性，造成结果集中的属性有可能不确定，这个问题在本例中通过使用级联点注来解决。

5) **查询**：圣劳伦斯河发源地国家的首都的名字是什么，该城市的人口是多少。

```
SELECT  Ci.Name, Ci.Pop
FROM    City Ci, Country Co, River R
WHERE   R.Origin = Co.Name AND
        Co.Name = Ci.Country AND
        R.Name = 'St. Lawrence' AND
        Ci.Capital= 'Y'
```

注释 这条SQL语句涉及三个表之间的连接：River表和Country表在*Origin*和*Name*属性上进行连接；Country表和City表在*Name*和*Country*属性上进行连接。在River表和City表上各有两个选择条件。

6) **查询**：City表中列出的非首都城市的平均人口是多少?

```
SELECT  AVG(Ci.Pop)
FROM    City Ci
WHERE   Ci.Capital= 'N'
```

注释 AVG（平均）是聚集操作的一个例子，RA中没有聚集操作。除了AVG之外，其他的聚集操作还有：COUNT（计数）、MAX（最大值）、MIN（最小值）以及SUM（求和）。聚集操作增强了SQL的功能，因为这些操作可以在获取的数据上进行计算。

7) **查询**：求出各个大洲的平均GDP。

```
SELECT    Co.Cont, Avg(Co.GDP) AS Continent-GDP
FROM      Country Co
GROUP BY  Co.Cont
```

注释 与基本SQL查询相比，该SQL语句有了较大变化、因为出现了GROUP BY子句。GROUP BY子句根据句中所列的属性对表进行分组，即把所有属性值相同的元组放在同一分组内。在本例中，*Co.cont*的取值只有两种可能：NAM和SAM，所以Country表就被分成了两组，对每一组计算平均*GDP*，并把结果存到SELECT了句指定的*Continent-GDP*属性中。

8) **查询**：对每一个至少是两条河流发源地的国家，找到发源于它的最短的河流。

```
SELECT    R.Origin, MIN(R.length) AS Min-length
FROM      River R
GROUP BY  R.Origin
HAVING    COUNT(*) > 1
```

注释 这条SQL语句和上一条语句有点类似。所不同的是，HAVING子句允许对利用GROUP BY子句得到的各个组再进行条件选择。本例中，只考虑有多于一个成员的组。

9) **查询**：列出GDP超过加拿大的国家。

```
SELECT  Co.Name
FROM    Country Co
WHERE   Co.GDP      > ANY  ( SELECT  Co1.GDP
                             FROM    Country Co1
                             WHERE   Co1.Name = 'Canada' )
```

注释 这是一个嵌套查询的例子，即在查询中还包括另一个子查询。当查询需要在一个并不存在的中间表上进行判定时，嵌套查询就必不可少了。嵌套的子查询一般出现在WHERE子句中，实际上它们也可以在SELECT子句和FROM子句中出现，只不过这种情况很少见。ANY是一个集合比较运算符。有关嵌套查询的完整介绍可查阅数据库的标准教材。

3.3.5 RA和SQL小结

RA是一种形式化的数据库查询语言，尽管通常不在商业DBMS上实现它，但它却构成了SQL的重要核心。SQL是目前使用最为广泛的数据库语言，它由两部分组成：DDL和DML。数据库表模式是由DDL指定和填充的，而实际查询则通过DML

表述。前面我们给出了有关SQL的简要介绍，更详细的信息可以通过查阅数据库教材得到。

3.4 扩展 SQL以处理空间数据

虽然RA和SQL都是功能强大的查询处理语言，但它们还是有不足之处，其中最主要的一点是，它们通常只提供简单的数据类型：整型、日期型、字符串型等。空间数据库(SDB)的应用必须能处理像点、线和多边形这样的复杂的数据类型，为此，数据库厂商采取了两个对策：一种是采用*blob*来存储空间信息；另一个方法是建立一种混合系统，即通过GIS软件把空间属性存储在操作系统的文件中。SQL不能处理以blob形式存储的数据，而把处理blob形式数据的任务交给了应用程序［Stonebraker and Moore，1997］，这种解决方案既低效又缺乏美感，因为数据必须依赖于宿主语言的应用程序代码。在混合系统中，空间属性存储在一个单独的操作系统文件中，这样就无法利用传统数据库服务，如查询语言、并发控制以及索引支持。

面向对象系统对扩展DBMS的功能来支持空间（复杂）对象有重大的影响。这种程序扩展了关系数据库的面向对象特性，最终产生对象关系数据库管理系统(OR-DBMS)的通用框架。OR-DBMS的关键特性是它支持SQL的最新版本：SQL3/SQL99，这一版本支持用户自定义类型（就像Java或者C++）。我们的目标是研究SQL3/SQL99，以便用它来操纵和获取空间数据。

空间SQL的基本要求是，采用更贴近人们对空间理解的概念，为空间数据提供更高层次的抽象［Egenhofer，1994］。这可以通过引入面向对象中用户自定义ADT的思想来实现。ADT由用户自定义类型和相关的函数组成。例如，若要把地块以多边形方式存储在数据库中，则相应的ADT可以是*polygon*类型和某个关联函数（方法），比如`adjacent`组成。`adjacent`函数用于多个地块，判断它们之间是否有共同的边界。而用抽象（abstract）一词是因为最终用户无需知道实现这些关联函数的细节，他们只需关心接口，即了解哪些函数可用以及这些函数的输入参数和输出结果的数据类型。

3.4.1 OGIS标准的SQL扩展

OGIS是由一些主要软件供应商组成的联盟，它负责制定与GIS互操作相关的行

业标准。OGIS的空间数据模型可以嵌入到各种编程语言中，例如C、Java、SQL等等，本节主要介绍SQL的嵌入。

OGIS是基于如图2-2所示的几何数据模型之上的。我们来回顾一下，该数据模型包括一个基类GEOMETRY，这个基类是非实例化的（即，不能把对象定义为GEOMETRY的一个实例），但它规定了一个适用于其子类的空间参照系统。有四个主要的子类是从GEOMETRY这个超类派生出来的，这四个子类是Point、Curve 、Surface和GeometryCollection，每个类还关联了一组操作，这组操作作用于类的实例，表3-9中列出了一些重要操作及其定义。

在OGIS标准中，所指定的操作可分成三类：

1) 用于所有几何类型的基本操作。例如，SpatialReference返回所定义对象几何体采用的基础坐标系统。常见的参照系统的例子包括：人们熟悉的经纬度(latitude and longitude)系统和用得最多的统一横轴墨卡托（Universal Traversal Mercator，UTM）。

2) 用于空间对象间拓扑关系的操作测试。例如，overlap判断两个对象内部是否有一个非空的交集（详见第2章）。

3) 用于空间分析的一般操作。例如，distance返回两个空间对象之间的最短距离。

表3-9 SQL的OGIS标准定义的一些操作[OGIS, 1999]

基本函数	SpatialReference()	返回几何体的基本坐标系统
	Envelope()	返回包含几何体的最小外接矩形
	Export()	返回以其他形式表示的几何体
	IsEmpty()	如果几何体是空集则返回真
	IsSimple()	如果几何体是简单的（即不自交）则返回真
	Boundary()	返回几何体的边界
拓扑/集合运算符	Equal	如果两个几何体的内部和边界在空间上相等，则返回真
	Disjoint	如果内部和边界都不相交，则返回真
	Intersect	如果几何体不相交，则返回真
	Touch	如果两个面仅仅是边界相交但是内部不相交，则返回真
	Cross	如果一条线和面的内部相交，则返回真

（续）

空间分析	Within	如果给定的几何体的内部不和另一个几何体的外部相交，则返回真
	Contains	判断给定的几何体是否包含另一个给定的几何体
	Overlap	如果两个几何体的内部有非空交集，则返回真
	Distance	返回两个几何体之间的最短距离
	Buffer	返回到给定几何体的距离小于或等于指定值的几何体的点的集合
	ConvexHull	返回几何体的最小闭包
	Intersection	返回由两个几何体的交集构成的几何体
	Union	返回由两个几何体的并集构成的几何体
	Difference	返回几何体与给定几何体不相交的部分
	SymmDiff	返回两个几何体与对方互不相交的部分

3.4.2 OGIS标准的局限性

OGIS规范仅仅局限用于空间的对象模型，从前面的章节可以看出，空间信息有时可以很自然地映射到基于场的模型。OGIS正在开发针对场数据类型和操作的统一模型。第8章将会介绍与基于场模型有关的操作，这种场模型或许会整合到OGIS未来的标准中。

即使在对象模型中，对于简单的选择-投影-连接查询来说，OGIS的操作也有局限性。用GROUP BY 和HAVING子句来支持空间聚集查询确实会出问题（参见习题4）。最后，OGIS标准过于关注基本拓扑的和空间度量的关系，而忽略了对整个度量操作的类的支持，也就是说，它不支持那些基于方位（例如，北、南、左、前等）谓词的操作。OGIS标准还不支持2.1.5节中讨论到的动态的、基于形状以及基于可见性的操作。

3.5 强调空间的查询示例

下面采用OGIS的数据类型和操作对World数据库进行SQL查询，这些查询强调Country、City和River之间的空间关系。首先，我们来重新定义关系模式，假设在SQL中可以使用OGIS的数据类型和操作，修订后的关系模式如表3-10所示。

1) **查询**：列出Country表中所有与美国（USA）相邻的国家名字。

```
SELECT   C1.Name AS "Neighbors of USA"
FROM     Country C1, Country C2
WHERE    Touch(C1.Shape, C2.Shape) = 1 AND
         C2.Name = 'USA'
```

注释 谓词Touch检测两个几何对象是否彼此相邻而又不交叠。这是一个用于判断相邻几何对象的很有用的操作。Touch操作是OGIS标准所定义的八个拓扑谓词中的一个。拓扑运算的良好性质之一是在许多几何变换下具有不变性。特别地，为World数据库选择的坐标系不会影响拓扑运算的结果。

拓扑运算可以应用于几何类型的多个不同组合。因此，在理想情况下，这些运算能以“重载”的形式来定义。但是，许多对象关系DBMS并不支持面向对象的类继承和函数重载这样的概念，所以，出于实用的目的，应该针对所适用的几何类型的每一个组合分别定义这些运算。

表3-10 基本表

```
CREATE   TABLE   Country(
         Name    varchar(30),
         Cont    varchar(30),
         Pop     Integer,
         GDP     Number,
         Life-Exp  FLOAT(2)
         Shape   Polygon);
```

a)

```
CREATE   TABLE   River(
         Name    varchar(30),
         Origin  varchar(30),
         Length  Number,
         Shape   LineString);
```

b)

```
CREATE   TABLE    City (
         Name     varchar(30),
         Country  varchar(30),
         Pop      integer,
         Shape    Point );
```

c)

2) **查询**：找出River表中所列出的河流流经的国家。

```
SELECT   R.Name C.Name
FROM     River R, Country C
WHERE    Cross(R.Shape, C.Shape) = 1
```

注释 Cross也是一个拓扑谓词，它常常用于判断LineString与Polygon之间（如本例情况）或一对LineString之间是否相交。

3) **查询**：对于River表中列出的河流，在City表中找到距离其最近的城市。

```
SELECT C1.Name, R1.Name
FROM   City C1, River R1
WHERE  Distance (C1.Shape, R1.Shape)  <
               ALL (SELECT Distance(C2.Shape, R1.Shape)
               FROM        City C2
               WHERE       C1.Name <> C2.Name
                           )
```

注释 Distance是一个返回实数值的二元运算，它可作用于任何几何对象的组合上。在本例中有两处使用到这个运算：WHERE子句和子查询中的SELECT子句。

4) **查询**：圣劳伦斯河能为方圆300公里以内的城市供水，列出能从该河获得供水的城市。

```
SELECT  Ci.Name
FROM    City Ci, River R
WHERE   Overlap(Ci.Shape, Buffer(R.Shape,300)) = 1 AND
        R.Name = 'St. Lawrence'
```

注释 一个几何对象的Buffer是指以该对象为中心并由Buffer运算的参数作为尺寸的几何区域，本例中，查询指定了Buffer区域的大小。很多GIS应用都采用Buffer运算，包括洪水泛滥区管理以及城市与乡村划分法则。图3-2给出了Buffer运算的图形表示，从该图可以看出，如果河流发洪水，城市A和B很可能会受影响，而城市C不会受影响。

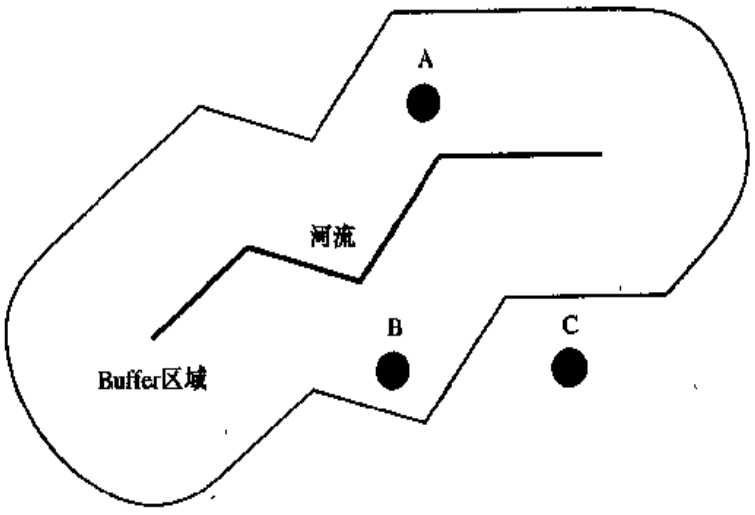


图3-2 河流的Buffer区域以及在该区域内外的点

5) **查询**：列出Country表中每个国家的名字、人口和国土面积。

```
SELECT  C.Name, C.Pop, Area(C.Shape) AS "Area"
FROM    Country C
```

注释 这条查询语句说明了Area函数的用途，该函数只适用于Polygon（多边形）和MultiPolygon（多个多边形）这两种几何类型。面积的计算显然取决于World数据库所采用的坐标系。例如，如果Country元组的形状是以经纬度形式给出，那么在计算面积之前要作一个中间坐标系变换。对于Distance和Length两个函数也要注意类似问题。

6) **查询**：求出河流在流经的各国境内的长度。

```
SELECT  R.Name, C.Name, Length(Intersection(R.Shape, C.Shape))
        AS "Length"
FROM    River R, Country C
WHERE   Cross(R.Shape, C.Shape) = 1
```

注释 二元运算Intersection的返回值是一个几何类型。Intersection运算与Intersects函数不同，后者是一个拓扑谓词，用于判断两个几何体是否相交。LineString和Polygon的Intersection可以是Point类型或LineString类型。在本例中，如果一条河流的确流经一个国家，则结果将是一条LineString。这样，Length函数将返回该河流流经每个国家时的非零长度。

7) **查询**：列出每个国家的GDP以及其首都到赤道的距离。

```
SELECT  Co.GDP, Distance(Point(0,Ci.Shape.y),Ci.Shape)
        AS "Distance"
FROM    Country Co, City Ci
WHERE   Co.Name = Ci.Country AND
        Ci.Capital = 'Y'
```

注释 查找数据库中各数据集之间的隐含关系已经超出了标准数据库的功能范围。现在的DBMS都适合进行联机事务处理(on-line transaction processing，OLTP)，而本例中的查询则属于联机分析处理（on-line analytical processing，OLAP）范畴。OLAP归属于数据挖掘领域，我们将在第8章对这方面的问题进行探讨。现在能做的是列出所有的首都并分别求出它们到赤道的距离。

Point(0，Ci.Shape.y)是赤道上的一个与Ci.Name的首都实例有相同经度的点，查询结果如表3-11所示。

表3-11 查询7的结果

Co. Name	Co. GDP	Dist-to-Eq (in Km)
Havana	16.9	2562
Washington, D.C.	8003	4324
Brasilia	1004	1756
Ottawa	658	5005
Mexico City	694.3	2161
Buenos Aires	348.2	3854

8) **查询**：按其邻国数目的多少列出所有国家。

```
SELECT    Co.Name, Count(Co1.Name)
FROM      Country Co, Country Co1
WHERE     Touch(Co.Shape, Co1.Shape)
GROUP BY  Co.Name
ORDER BY  Count(Co1.Name)
```

注释 在这个查询中，所有至少有一个邻国的国家将根据其邻国的个数进行排序。

9) **查询**：列出只有一个邻国的国家。如果一个国家与另一个国家在陆地上有一条共同的国界，那么这个国家就是另一个国家的邻国。根据这个定义，某些岛国（例如，冰岛）就没有邻国。

```
SELECT    Co.Name
FROM      Country Co, Country Co1
WHERE     Touch(Co.Shape, Co1.Shape))
GROUP BY  Co.Name
HAVING    Count(Co1.Name) = 1
SELECT  Co.Name
FROM    Country Co
WHERE   Co.Name IN
        (SELECT   Co.Name
        FROM     Country Co, Country Co1
        WHERE    Touch(Co.Shape, Co1.Shape)
        GROUP BY  Co.Name
        HAVING    Count(*) = 1)
```

注释 这里，我们在FROM子句中用到了一个嵌套查询，FROM子句的查询结果是一个表，它由互为邻国的国家对组成。GROUP BY子句根据国家的名字对该结果表进行分组。最后，HAVING子句选择出只有一个邻国的国家，它的作用类似于WHERE子句，所不同的是，在HAVING子句中必须包含像

count、sum、max和min这样的聚集函数。

10) **查询**：哪一个国家的邻国最多。

```
CREATE VIEW Neighbor AS
SELECT                  Co.Name, Count(Co1.Name) AS num_neighbors
FROM                    Country Co, Country Co1
WHERE                   Touch(Co.Shape, Co1.Shape)
GROUP BY                Co.Name

SELECT   Co.Name, num_neighbors
FROM     Neighbor
WHERE    num_neighbor = (SELECT Max(num_neighbors)
         FROM Neighbor)
```

注释 这条查询说明视图在简化复杂查询时的作用。第一个查询（视图）计算每一个国家的邻国数，第二个查询从Neighbor视图中选出邻国数最大的国家。这个视图创建了一个虚拟表，该虚拟表可以像普通表一样用于后续的查询。

3.6 趋势：对象-关系SQL

OGIS标准指定了数据类型及其相关的运算，它们对GIS这样的空间应用来说是必不可少的。例如，Point数据类型的一个重要运算是Distance，它负责求出两个点之间的距离。而length运算用在Point数据类型时从语义上看就不正确了。这和concatenation（串联）运算更适用于Character（字符）型数据而非Integer（整型）的争论很相似。

在关系数据库中，所有数据类型都是固定的，而对于对象关系数据库或面向对象数据库来说，就不存在这种限制，因为这些数据库支持用户自定义数据类型。这种特性显然是一个优点，尤其是处理GIS这类非传统数据库应用时更加有用。不过从另一方面看，数据库应用开发人员在构建符合语法和语义的数据类型方面的负担则更重了。为了减轻开发人员的负担，商业数据库厂商推出了特殊应用的“软件包”，它给数据库用户提供无缝接口。例如，Oracle就有专为GIS领域开发的软件包——Spatial Data Cartridge。

针对OR-DBMS提出的SQL标准SQL3/SQL99可以使用户在关系数据库的框架内

定义自己的数据类型，下面来介绍SQL3标准中对用户自定义空间数据类型有帮助的两个特点。

3.6.1 SQL3概览

相对于目前SQL方案所采用的SQL2/SQL92，SQL3/SQL99主要在以下两方面进行扩展：

1) **ADT**：可以使用CREATE TYPE语句来定义ADT。和面向对象技术中的类一样，ADT由一组属性和访问这些属性值的成员函数组成，成员函数可以隐含地修改数据类型中的属性值，因面也就能改变数据库的状态。ADT可以作为关系模式中某一列的类型。为了访问封装在ADT中的数据值，必须在CREATE TYPE中定义一个成员函数。例如，下面的脚本创建了一个Point类型，并定义了一个成员函数Distance：

```
CREATE TYPE  Point  (
          x  NUMBER,
          y  NUMBER,

   FUNCTION  Distance(:u Point,:v Point)
             RETURNS NUMBER
                    );
```

u和v前的冒号表示这两个变量是局部变量。

2) **行类型**：行类型是用于定义关系的类型，它指定了关系的模式。例如，下面的语句创建了一个行类型Point：

```
CREATE ROW TYPE  Point  (
               x  NUMBER,
               y  NUMBER );
```

这样就可以创建一个表作为这个行类型的实例：

```
CREATE TABLE Pointtable of TYPE Point;
```

这里我们把重点放在使用ADT上面不是在行类型上，因为OR-DBMS是关系数据库的扩展，用ADT作为列类型能很自然地与OR-DBMS的定义保持一致。

3.6.2 对象关系模式

Oracle8是由Oracle公司开发的对象关系数据库管理系统(OR-DBMS)，其他数据

库公司也开发类似产品，例如IBM。OR-DBMS实现了SQL3标准的一部分，ADT在这种系统中称作“对象类型”。

下面将介绍在Oracle8中如何构造Point、LineString和Polygon这三种基本空间数据类型。

```
CREATE  TYPE Point AS OBJECT (
        x   NUMBER,
        y   NUMBER,
        MEMBER FUNCTION Distance(P2   IN Point) RETURN NUMBER,
        PRAGMA RESTRICT_REFERENCES(Distance, WNDS));
```

Point类型有两个属性x和y，以及一个成员函数Distance。PRAGMA指出了下列事实：函数Distance不会修改数据库的状态，因为使用了WNDS（Write No Database State）。在OGIS标准中定义了许多与Point类型相关的函数，这里只简单地给出一个。创建了Point类型之后，它就可以作为一个属性类型用于关系中。例如，City的关系模式可以定义如下：

```
CREATE  TABLE    City (
        Name     varchar(30),
        Country  varchar(35),
        Pop      int,
        Capital  char(1),
        Shape    Point   );
```

关系模式一旦定义，这个表就能够以通常方式录入数据了。例如，下面的语句把巴西的首都Brasilia（巴西利亚）加到数据库中：

```
INSERT INTO CITY( Brasilia', 'Brazil', 1.5, 'Y',
                  Point(-55.4,-23.2));
```

LineString类型的构造相对Point类型来说要复杂一些。首先创建一个中间类型LineType：

```
CREATE TYPE LineType AS VARRAY(500) OF Point;
```

LineType类型是一个以Point类型为成员的变长数组，最大长度为500。如果一个类型被定义成Varray，那就不能为其定义成员函数。因此这里创建另外一个类型LineString：

```
CREATE  TYPE LineString AS OBJECT (
        Num_of_Points INT,
        Geometry LineType,
        MEMBER FUNCTION Length(SELF IN) RETURN NUMBER,
        PRAGMA RESTRICT_REFERENCES(Length, WNDS));
```

属性Num_of_Points保存了每个LineString类型的实例的大小（即有多少个点）。现在可以定义River表的模式了：

```
CREATE  TABLE   River(
        Name    varchar(30),
        Origin  varchar(30),
        Length  number,
        Shape   LineString  );
```

当向River表插入数据时，必须清楚表所涉及的不同数据类型。

```
INSERT INTO RIVER('Mississippi', 'USA', 6000,
                  LineString(3, LineType(Point(1,1),Point(1,2),
                  Point(2,3)))
```

Polygon类型和LineString类型类似，其类型和表的创建以及数据的插入等语句如表3-12所示。

表3-12 Country表的创建语句

```
CREATE TYPE PolyType AS VARRAY(500) OF Point
```

a)

```
CREATE  TYPE Polygon AS OBJECT (
        Num_of_Points INT,
        Geometry PolyType ,
        MEMBER FUNCTION Area(SELF IN) RETURN NUMBER,
        PRAGMA RESTRICT_REFERENCES(Length, WNDS));
```

b)

```
CREATE  TABLE     Country(
        Name      varchar(30),
        Cont      varchar(30),
        Pop       int,
        GDP       number,
        Life-Exp  number,
        Shape     LineString  );
```

c)

```
INSERT INTO  Country('Mexico', 'NAM', 107.5, 694.3, 69.36,
             Polygon(23, Polytype(Point(1,1), ..., Point(1,1)))
```

d)

3.6.3 查询示例

1) **查询**：求出City表列出的所有城市两两之间的距离。

```
SELECT  C1.Name, C1.Distance(C2.Shape) AS ''Distance''
FROM    City C1, City C2
WHERE   C1.Name <> C2.Name
```

注释 注意，SELECT子句中Distance函数使用了面向对象的表示法，把它与3.5节中判断距离的Distance(C1.Shape, C2.Shape)表示法作比较。WHERE子句中的谓词是为了确保不会对一个城市求自己到自己的距离。

2) **查询**：用Shape属性中包含的几何信息来验证River表中每条河流的长度。

```
SELECT  R.Name, R.Length, R.Length() AS ''Derived Length''
FROM    River R
```

注释 这条查询语句用于验证数据，River表的Length属性已经保存了河流长度信息，使用Length()函数可以检查表中数据的完整性。

3) **查询**：列出与美国接壤的国家的名字、人口和国土面积。

```
SELECT  C2.Name, C2.Pop, C2.Area() AS ''Area''
FROM    Country C1, Country C2
WHERE   C1.Name = 'USA' AND
        C1.Touch(C2.Shape) = 1
```

注释 Area()函数应该是支持Polygon ADT的一个自然函数，除了Area()外，该查询还使用了一个拓扑谓词Touch。

3.7 小结

本章讨论了数据库查询语言，本章包含如下主题。

*RA*是与关系模型相关的一种形式化查询语言，它（如果不是从未，也是）极少在商业系统中进行实现，但它却构成了SQL的核心。

*SQL*是应用最为广泛的查询语言，它是一种声明性语言，即用户只需指明所要查询的结果即可，而不需要指定得到结果的过程。SQL增加了许多重要的函数来对RA进行扩展，其中包括用于对查询的数据进行分析处理的聚集函数。

*OGIS*标准建议使用一组空间数据类型和函数，这对空间数据查询来说至关重要。

*SQL3/SQL1999*是用于对SQL进行对象-关系上扩展的一个标准平台。它并不是专门针对GIS或者空间数据库的，而是涵盖一般的对象关系数据库。最自然的情况是，OGIS推荐标准将实现SQL3的一个子集。

3.8 参考书目

3.1、3.2、3.3 对关系代数和SQL的完整介绍可以在任何数据库的入门教材中找到，也可以参考以下著作：［Elmasri and Navathe，2000; Ramakrishnan，1998;Ullman and Widom，1999］。

3.4、3.5 SQL在空间应用方面的扩展在［Egenhofer，1994］中进行了探讨。OGIS的文档［OpenGIS，1998］是对SQL的不同空间扩展进行协调的一种尝试。有关支持空间数据分析的查询语言例子可以参阅［Lin and Huang，2001］。

3.6 SQL1999/SQL3是目前已经采用的对SQL进行对象-关系扩展后的标准，在一些商业产品（包括Oracle的Oracle8和IBM的DB2）上已经实现了这个标准的一个子集。

3.9 习题

习题1和2中的查询将用到表3-1。

1. 请用关系代数表示以下查询：

   a) 找出GDP大于5000亿而小于100万亿的国家。

   b) 列出有河流发源的各国家的居民平均寿命。

   c) 找到位于南美洲或者人口少于200万的城市。

   d) 列出不位于南美洲的城市。

2. 用SQL语句写出习题1中的查询。

3. 用SQL语句写出下面的查询：

   a) 计算人口少于1亿的国家的个数。

   b) 找出北美洲GDP最低的国家，不要使用MIN函数。（提示：使用嵌套查询。）

c) 列出北美洲所有的国家以及首都人口少于500万的国家。

d) 找出GDP排名第二的国家。

4. Reclassify（再分类，见2.1.5节）是一个基于非空间属性来对空间几何体进行组合的聚集函数。该函数可以利用已经存在的对象来创建新的对象，一般是通过去掉在选择的属性上具有相同值的两个相邻多边形的公共边界来实现的。能否通过OGIS操作和带有空间数据类型的SQL92来实现再分类操作？解释之。

5. 请讨论图2-2中的几何数据模型。假设在“世界”尺度下，城市可以用点数据类型来表示，那么国家该用什么数据类型来表示？注意：新加坡、梵蒂冈和摩纳哥都是国家。OGIS标准所推荐的空间函数对该模型的实现有什么潜在的作用？

6. [Egenhofer，1994] 对SQL的空间应用扩展提出了一系列的要求，这些要求如下表所示。这些建议中哪些已经被OGIS SQL标准所采用？对于那些还没有被采用的，试讨论其未被采用的可能的原因。

空间ADT (Spatial ADT)	包含相关操作的空间抽象数据类型体系结构
图形表示(Graphical presentation)	与空间数据交互的自然媒介
结果合并(Result combination)	合并查询序列的结果
上下文(Context)	通过包含没有明确请求的信息来把结果放入上下文
内容检查(Content examination)	提供机制来引导画图进展
指出所选部分(Selection by pointing)	在地图上指出所选部分
显示操纵(Display manipulations)	改变空间对象及其各部分的图形表示
图例(Legend)	描述性的图例
标注(Label)	用于方便理解图画的标注
地图比例尺的选择(Selection of map scale)	制作地图应允许用户继续把他们的技术应用于解释被画对象的实际大小和选择特定的绘制比例尺
关心的区域(Area of interest)	用于限定一个特定地貌的关心区域的工具

7. OGIS标准包含了一组拓扑的空间谓词，请问该如何对这个标准进行扩展以支持方位性谓词（directional predicate），例如East、North、North-East等等。请注意这些方位谓词有可能是模糊不清的，例如“North-East到哪里结束？而East又从哪里开始？”

8. 这道习题将涉及到维度扩展的九交模型DE-9IM，它是对第2章中介绍的Egenhofer的九交模型的扩展。DE-9IM的矩阵如下所示：

$$\Gamma_9(A,B)=\begin{pmatrix} dim(A^\circ\cap B^\circ) & dim(A^\circ\partial B^\circ) & dim(A^\circ\cap B^-) \\ dim(\partial A\cap B^\circ) & dim(\partial A\cap\partial B) & dim(\partial A\cap B^-) \\ dim(A^-\cap B^\circ) & dim(A^-\cap\partial B) & dim(A^-\cap B^-) \end{pmatrix}$$

9IM与DE-9IM的关键区别是，在DE-9IM中只要求几何对象的维度，而不必检测矩阵的每一个项是否为空。平面二维对象的维度可以取四个值：-1（空集）、0（点）、1（直线）、2（面积不为零的对象）。在许多实例中，矩阵各个项的具体值是无关紧要的。下面是矩阵中各项的取值范围。

T：*X*和*Y*必定相交，即$dim\ (X\cap Y)$ = 0、1、2。其中*X*和*Y*分别是*A*和*B*的内部、外部或者边界。

F：$dim\ (X\cap Y)$ = -1，则*X*和*Y*肯定是不相交的。

*：是否有交集是无关紧要的，即$dim\ (X\cap Y)$ = {-1，0，1，2}

0：$dim\ (X\cap Y)$ = 0

1：$dim\ (X\cap Y)$ = 1

2：$dim\ (X\cap Y)$ = 2

下面是两个相等对象的符号矩阵：

$$\begin{pmatrix} T & * & F \\ * & * & F \\ * & * & * \end{pmatrix}$$

a) touch和cross两个拓扑运算的符号矩阵是什么？注意，这个符号矩阵和数据类型的组合有关，例如点/点组合的矩阵和多多边形/多多边形组合的矩阵是不一样的。

b) 下面的符号矩阵表示的操作及其数据类型是什么？

$$\begin{pmatrix} 1 & * & T \\ * & * & F \\ T & * & * \end{pmatrix}$$

c) 考虑图3-3中的示例，对应的9IM和DE-9IM中的符号矩阵分别是什么，请讨论DE-9IM是否优于9IM。

9. 用SQL语句写出下面的查询，请使用OGIS扩展的数据类型和函数。

a) 列出City表中距离华盛顿特区5000英里以内的城市。

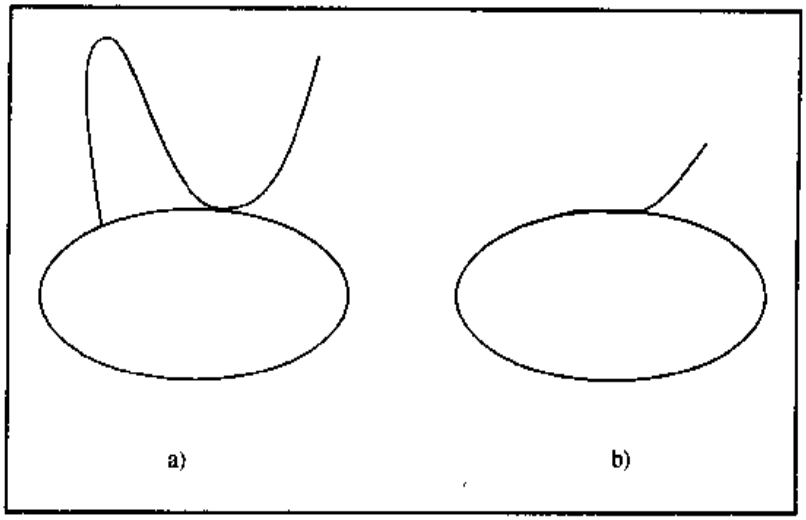


图3-3 示例对象［Clementini and Felice，1995］

b) 位于阿根廷和巴西的Rio Paranas河的长度是多少?

c) 阿根廷和巴西接壤吗?

d) 列出完全在赤道以南的国家。

10. 假设有如下的关系模式：

RIVER(NAME:char, FLOOD-PLAIN:polygon, GEOMETRY:linstring)
ROAD(ID:char, NAME:char, TYPE:char, GEOMETRY:linstring)
FOREST(NAME:char, GEOMETRY:polygon)
LAND-PARCELS(ID:integer, GEOMETRY:polygon, county:char)

把下面的问题转换成SQL语句并使用OGIS扩展的数据类型和函数：

a) 列出穿过Itasca State Forest的所有河流的名字。

b) 列出所有经过Francis Forest的公路的名字。

c) 位于Montana河的泛滥区域内的公路都有可能受洪灾的影响，请确定出这些道路。

d) 距离红河 2英里以内或者距离Big Tree State Park 5英里以内不允许进行域市开发，请确定出位于这些区域内的地块和郡。

e) 列出一条做为国界的河流。

11. 学习使用编译器工具（如YACC，Yet Another Compiler Compiler）。开发一个语法模式，用来根据带象形符号的实体联系图（entity relationship diagram，ERD）产生SQL3的数据定义语句。

12. 如何使用九交模型或者OGIS拓扑操作来对下面的空间关系进行建模?

a) 河流（LineString）发源于某个国家（Polygon）。

b) 一个国家被另一个国家完全包围。例如，意大利完全包围梵蒂冈。

c) 一条河流入另一条河。例如，密苏里河流入密西西比河。

d) 森林按林分分区。

13. 为本章附录中的每个关系代数查询写出相应的SQL语句。

14. 用象形图重画图3-4中的ER图，其中的Fishing-Opener和Distance两个属性在新图中应该如何表示？然后再把新图翻译成相应的SQL3/OGIS语句用于生成表。

15. 考虑图1-3和1-4中的表，请对人口普查区使用各种表达形式的SQL查询来计算它的空间属性（例如，面积、周长），并比较用哪一种表现形式得到的SQL查询更简单。

16. 请回顾2.1.6节中的Java程序，为3.6.3中的空间查询设计相应的Java程序，并把它与用SQL3/OGIS写的查询进行比较。

17. 用SQL3的用户自定义数据类型定义OGIS中的几何体的聚集数据类型。

18. 请回顾2.2.3节中关于州立公园的例子，并用OGIS空间数据类型给出创建相关表的SQL语句。

19. 考虑基于形状的查询，例如，列出形状与女式长靴类似的国家，或者列出有近似正方形的人口普查区。请问为了支持这样的查询，应如何对SQL3/OGIS进行扩展。

20. 考虑基于可视性的查询，例如，列出从某个观察点以及观察方向可以看到（未被遮住）的对象，试列出一组数据类型和操作以扩展SQL3/OGIS来支持这样的查询。

3.10 附录：州立公园数据库

State Park数据库由两个实体（Park和Lake）组成，这两个实体的属性和它们之间的关系如图3-4所示，该ER图所对应的关系模式如下所示，表3-13将这些实体以及它们之间联系进行了实体化。

StatePark(Sid: integer, Sname: string, Area: float, Distance: float)
Lake(Lid: integer, Lname: string, Depth: float, Main-Catch: string)
ParkLake(Lid: integer, Sid: integer, Fishing-Opener: date)

上述模式说明了三个实体：StatePark、Lake和ParkLake。StatePark表示明尼苏达州所有的州立国家公园，其属性包括一个唯一的国家标识符Sid、公园的名字Sname、以平方公里为单位的面积Area以及公园与明尼阿波利斯之间的距离Distance。实体Lake也有一个唯一的标识符Lid、名称Lname、湖的平均深度Depth以及湖内主要鱼种Main-catch。实体ParkLake用于在StatePark和Lake两个实体之间进行一体化查询，它对每个位于州立公园的湖泊进行了标记，它的属性有Lid、Sid以及所对应湖泊开始捕鱼的季节Fishing-Opener。这里假设不同湖泊有不同的Fishing-Opener。

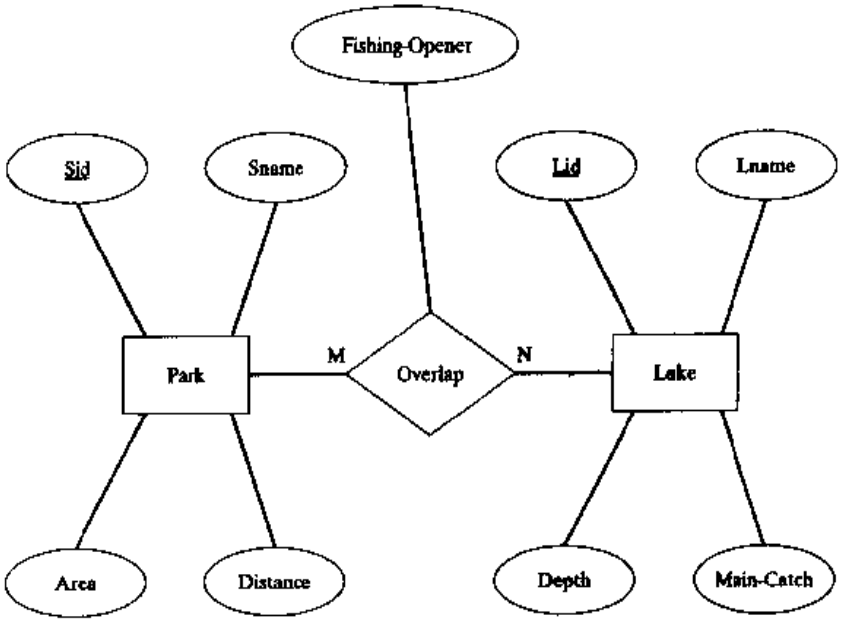


图3-4 州立公园数据库的ER图

表3-13 州立公园数据库的表

Park	Sid	Sname	Area	Distance
	S1	Itasca	150.0	52
	S2	Woodbury	255.0	75
	S3	Brighton	175.0	300

a) Park

Lake	Lid	Lname	Depth	Main-Catch
	100	Lino	20.0	Walleye
	200	Chaska	30.0	Trout
	300	Sussex	45.0	Walleye
	400	Todd	28.0	Bass

b) Lake

ParkLake	Lid	Sid	Fishing-Opener
	100	S1	05/15
	200	S1	05/15
	300	S3	06/01

c) ParkLake

RA查询示例

下面将通过一些例子来说明如何利用前面所定义的关系运算符来获取和操作数据库中的数据，我们采用这样的格式：先描述查询，然后给出等价的关系代数表达式，最后再给出代数表达式的相关注释，包括代数表达式的另一种形式。

查询：找出包含Lid为100 的Lake的State Park的名称。

$$\pi_{Spname}(StatePark \bowtie \sigma_{Lid=100}(ParkLake))$$

注释 首先选出ParkLake中Lid为100的元组，然后将结果集与StatePark关系按Sid码进行自然连接，最后把连接结果在StatePark的名字Spname上进行投影。可以用更名运算符ρ把这个查询分成几个部分。更名运算符用于对一个复杂查询中出现的中间关系进行命名，它也可以用于对关系的属性进行重命名。例如，下式

$$\rho(Newname(1 \rightarrow Att1), Oldname)$$

把关系Oldname重命名为Newname，同时也把从左往右数起的第一个属性重命名为Att1。

利用这种命名规范，可以把上面的查询分成以下三部分：

$$\rho(Temp1, \sigma_{Lid=100}(ParkLake))$$

$$\rho(Temp2, Temp1 \bowtie StatePark)$$

$$\pi_{Spname}(Temp2)$$

该查询的另一个表示方法为：

$$\pi_{spname}(\sigma_{Lid=100}((StatePark \bowtie ParkLake))$$

从实现的角度来看，这个表达式比前一个查询更难实现，因为它要在一个较大的集合上进行连接运算，而连接运算又是关系代数的五种运算中代价最高的一种。

1) **查询**：找出包含以Trout（鲑鱼）为MainCatch（主要鱼种）的Lake的StatePark的名字。

$$\pi_{Spname}(StatePark \bowtie (ParkLake \bowtie \sigma_{Main\text{-}Catch='Trout'}(Lake)))$$

注释 在这个查询中连续使用了两个连接运算。不过首先要缩减集合的大

小，这里先选出主要鱼种为鲑鱼的湖泊。然后，把结果集与ParkLake在Lid码上进行连接，紧接着在Sid上作与StatePark的连接，最后把结果投影到StatePark的名字上。

2) **查询**：找出Itasca州立公园里湖泊的主要鱼种。

$$\pi_{\text{Main-Catch}}(\text{Lake} \bowtie (\text{ParkLake} \bowtie \sigma_{\text{Spname='Itasca'}}(\text{StatePark})))$$

注释 这条查询与前一个查询非常类似。

3) **查询**：找出至少有一个湖泊的公园的名字。

$$\pi_{\text{Spname}}(\text{StatePark} \bowtie \text{ParkLake})$$

注释 在Sid上进行的连接生成了一个中间关系，其中StatePark关系的元组被附加到ParkLake的元组上，最后，把结果投影到Spname。

4) **查询**：列出主要鱼种为鲈鱼（bass）或者突眼河鲈（walleye）的湖泊所在公园名字。

$$\rho(\text{TempLake}, \sigma_{\text{Main-Catch = Bass'}}(Lake) \cup \sigma_{\text{Main-Catch = 'Walleye'}}(Lake)$$
$$\pi_{\text{spname}}(\text{TempLake} \bowtie \text{ParkLake} \bowtie StatePark))$$

注释 这里我们第一次用到了并运算。首先选择主要鱼种是鲈鱼或者突眼河鲈的湖泊，然后在Lid上与StatePark连接，并再在Sid上与ParkLake连接，最后在Spname上投影得到所要的结果。

5) **查询**：列出主要鱼种为鲈鱼和突眼河鲈的湖泊所在公园名字。

$$\rho(TempBass, \pi_{Spname}(\sigma_{Main-Catch='Bass'} \bowtie ParkLake \bowtie StatePark))$$
$$\rho(TempWall, \pi_{Spname}(\sigma_{Main-Catch='Walleye'} \bowtie ParkLake \bowtie StatePark))$$
$$TempBass \cap TempWall$$

注释 这是一个正确但却没有结果的查询（因为设计的数据库模式中每个湖泊都只有一种主要鱼种）。

6) **查询**：找出至少包含两个湖泊的StatePark名字。

$$c\rho(Temp, \pi_{Sid,Spname,Lid}(StatePark \bowtie ParkLake))$$
$$\rho(Temppair, Temp \times Temp)$$
$$\pi_{Spname}\sigma_{(Sid1=Sid2)\wedge(Lid1\neq Lid2)}Temppair$$

7) **查询**：写出距离明尼阿波利斯至少50英里并且所包含的湖泊不是以鲑鱼为主要鱼种的州立公园的标识号Sid。

$$\pi_{sid}(\sigma_{distance>50}StatePark) - \pi_{sid}((\sigma_{main-catch='Trout'}Lake \bowtie ParkLake) \bowtie StatePark)$$

第4章
空间存储和索引

在本章中，我们描述从空间数据库中获得数据的有效方法，这经常是通过使用索引（index）完成的。和一本书最后的索引一样，数据库的索引可以用来快速访问一条特定查询所请求的数据，而无需遍历整个数据库。我们将描述特别适用于空间数据库的索引。迄今为止，我们对SDB的讨论大都局限于与空间数据的建模和查询相关的较高层次的问题。在本章中，我们将研究物理数据库的设计，从某种意义上说，这是深入到“黑匣子”的内部。

物理数据库设计对确保各种查询的合理性能至关重要，这些查询是用优雅简洁的高级逻辑语言书写的，例如SQL。这些高级语言屏蔽了实现算法和数据结构。若使用不成熟的物理数据库设计，即便是回答一条最简单的SQL查询，也会出现令终端用户难以忍受的响应时间。历史上，物理数据库设计技术（如B+树索引）由于为多种SQL查询提供了合理的响应时间，因而十分信赖地被大规模应用到关系数据库技术之中。例如，从一百亿条记录的集合中查找一条与给定码相匹配的记录，即使整个数据集位于一个磁盘中，也可在几分之一秒钟的时间内通过十次磁盘访问完成。查询处理技术的发展（在第5章中讨论）以超越硬件性能改进的速度在不断提高系

统性能。

DBMS系统是为处理海量数据而设计的。为了理解数据是如何存储的，我们需要简单地探究一下典型二级存储（secondary storage）设备的设计和几何构造。熟悉二级存储的特征有助于我们为存储数据设计相应的策略，以便能用一种高效的方式检索数据。对物理设计层面的基本看法是，数据集经常过大，无法在计算机的主存中存储，而访问二级存储的时间则要比访问主存慢几个数量级。数据在主存和二级存储之间的来回传送是产生性能瓶颈的原因。因而，一个好的物理数据库设计目标就是让这种数据传输量保持为一个绝对最小值。

空间存储结构的目标是方便空间选取和连接查询。这就是说，在响应一条查询时，空间存取方法将只需查询该空间中所有对象的一个关联子集，并返回查询结果集。

空间索引的基本思想，实际上也是所有的空间查询过程的基本思想，就是对近似(approximation)的使用。这种方法可以让索引结构按照一个或多个空间码来管理对象，这些空间码是比对象本身更简单的几何对象。一个最基本的例子是外包框（围住对象的与坐标轴平行的最小矩形）。对于网格的近似来说，是用规则的网格将空间划分成单元格，对象由一系列与之相交的单元格表示。使用近似可以产生一种用于过滤和精炼查询过程的策略：首先在近似的基础上，执行一个过滤步骤，它返回一个候选集，作为完全满足某个谓词的所有对象的超集；其次，在精炼步骤中，对于每一个候选对象（或者空间连接中的每一对候选对象）用精确的几何信息进行检查。由于使用了外包框，所以大多数空间数据结构都被设计成可以存储一组点（点值）或一组矩形（线或者区域值）。

这样的结构提供了插入(insert)、删除(delete)和成员查询(member)（查找一个已存储的点或矩形）等操作，用于管理这种类型的集合。除此之外，这种结构还支持一个或多个查询操作。对于所存储的矩形和点，还有如下一些重要操作：

- 点查询(point query)：找出所有包含给定点的矩形。
- 范围查询(range query)：找出所有位于给定矩形中的点。
- 最近邻居(nearest neighbor)：找出距查询点最近的点。
- 距离扫描(distance scan)：按与给定点距离的增序列出所有点。

- 相交查询(intersection query)：找出所有与给定矩形相交的矩形。
- 包含查询(containment query)：找出所有完全包含在给定矩形中的矩形。
- 空间连接查询(spatial join query)：找出所有相互交叠的矩形对。

4.1节将详细描述空间数据的物理存储要求，包括存储设备的几何结构、缓冲区的管理策略、文件结构的概念、聚类以及空间填充曲线。4.2节将描述空间数据的索引策略。4.3节重点介绍有关并发控制和空间连接索引的问题。

4.1 存储：磁盘和文件

传统数据库事务和程序设计语言应用程序的需求从某种意义上说是正交的(orthogonal)。例如，用于计算矩阵逆的一个C语言程序是一种CPU密集型的操作，它涉及复杂的公式和计算。在程序的开始，将实际的矩阵数据放入主存。函数操作的数据集都足够小，能够一次装入主存。CPU访问主存的速度是非常快的(纳秒级)，因而程序的效率与CPU速度直接相关。传统数据库应用涉及到大量数据，尽管内存的价格在持续下降，但将全部数据存储到主存中的做法仍然是不可行的。DBMS需要安排部分数据使之易于在主存与一个二级存储设备（通常是一个磁盘驱动器）之间来回传输。由于对二级存储的访问速度相对来说很慢（微秒级），这成为一个严重的瓶颈。如果数据在主存中，操作这些数据的函数都很简单，包括如(<、=、>、*MIN*、*MAX*、*AVG*) 操作。因此，对于一个传统数据库应用来说，数据从二级存储到主存之间的传输代价（即I/O代价）就是衡量效率的主要标准。

对于空间数据库来说，情形就更加复杂。首先，空间应用涉及的函数其计算复杂性与程序设计语言的应用程序不相上下。其次，空间数据库的存储需求一般来说要比传统数据库更高。例如，一个低分辨率的美国卫星图像就会消耗30兆磁盘空间！表4-1是传统DBMS、应用程序和SDBMS在CPU和I/O相对代价方面的特征。

表4-1 传统DBMS、应用程序和SDBMS在CPU和I/O相对代价方面的特征

	CPU代价	I/O代价
DBMS	低	高
C程序	高	低
SDBMS	高	高

事实上，空间数据库的特点就是太大，它甚至都不能用一个中等规模的二级存储来装载，数据会溢出到三级（tertiary）存储中，例如，磁带驱动器。磁带是顺序存储设备，即要存取磁盘最后一个字节的数据，需要转完整盘磁带，并且磁带经常是脱机的。在衡量一个DBMS性能时，并未计入从磁带读取数据的时间代价。另一方面，二级存储（当然也包括主存）是随机存取设备，与CD-ROM十分类似。

4.1.1 磁盘的几何结构和含义

要理解磁盘驱动器的几何结构，可以用CD-ROM的播放器做类比。金属的磁盘或圆形磁盘片放置在一根主轴上并旋转。磁道（作为潜在的数据驻留地）是圆形磁盘片上向边缘延伸的许多同心圆环。由于有多个磁盘片，所有同直径的磁道就组成一个柱面(cylinder)。

每个磁道都划分成扇区，扇区的大小由驱动器的厂商规定。一个磁盘块有整数倍个扇区，这个倍数可在磁盘初始化时设定。磁盘块（或简称为页面）是磁盘与主存之间的最小传输单元。由于这些块都分布在磁道上，因而一个磁盘从逻辑上说是一维设备。数据从磁盘向主存传输时使用由每个磁盘片的磁头组成的阵列。当DBMS要求从磁盘获取一个块时，磁头一致、机械地移动，停在该块所在的磁道上。所有磁头都位于它们所在磁盘片的同一磁道上。当然磁头绝不会碰到实际的磁盘片，那会毁坏磁盘，造成数据丢失。一旦磁头移动到正确的磁道，目标块在磁头下方旋转，块上磁化的信息就通过磁头拷贝到主存中。因此，整个过程可以分为三步，每一步都有一个特征（平均）时间。首先是寻道时间（t_s），它衡量磁头到达特定磁道所用的时间。其次是延迟时间（t_l），它是块旋转到磁头下方所用的时间。最后一个是传输时间（t_t），即置于正确位置后磁头读或写块中数据的实际时间。因此全部存取时间（t_a）可用下式计算：

$$t_a = t_s + t_l + t_t$$

而且通常满足下面不等式：

$$t_s > t_l > t_t$$

虽然传输时间t_t在磁盘初始化的时候已经固定，但将数据按照一定策略放置在磁盘上仍然可以在很大程度上减少寻道时间t_s和延迟时间t_l。表4-2是Western Digital

Caviar AC36400磁盘驱动器的物理和性能参数。

表4-2 Western Digital Caviar AC36400磁盘驱动器的物理和性能参数

格式化容量	6448MB
柱面数	13 328
每磁道的扇区数	63
每个扇区字节数	512
盘片数	3个双面的
寻道时间	9.5ms
延迟时间	5.5ms

4.1.2 缓冲区管理器

缓冲区管理器是DBMS中一个软件模块，专门负责管理主存与二级存储之间的数据传输。由于一个典型数据库事务所需交互的数据量远远大于主存的容量，因此，缓冲区管理器需要为高效的事务处理实施一个协议。具体地说，必须确保事务不会因为一部分数据不在主存中而停顿下来。管理器所实施的协议被称为置换策略，因为它们处理页面在主存与二级存储之间的置换。对于关系数据库的缓冲区管理，只利用传统的虚拟内存页面置换算法，如最近最少使用（LRU）算法，可能是不够的。

在关系数据库管理系统中，缓冲区管理主要基于关系查询行为。一组被频繁访问的页面称为频繁访问集（hot set）。其行为在循环（例如，嵌套循环）中比较常见。在这个方案中，为每个查询分配一个与其频繁访问集大小相当的本地缓冲池，新的查询只有在其频繁访问集放入内存后才能进入系统进行处理。尽管频繁访问集为关系数据库提供了一个页面访问（引用）模式的精确模型，但它是基于LRU置换方案的。DBMIN算法［Chou and DeWitt，1985］是以查询本地集模型（QLSM）为基础的。QLSM的页面引用模型不依赖任何特定的页面置换方案（如频繁访问集模型使用的LRU算法）。QLSM将数据库操作的引用模式特征化为顺序引用、随机引用和分层引用。缓冲区是以一个文件实例（表）为单位进行分配和管理的。和一个文件实例关联的缓冲页面集合被看作是它的本地集（local set）。本地集的大小根据查询计划和数据库的统计信息来决定。簿记（bookkeeping）通过维护一个全局页表和全局空闲链表来实现。如果在本地集和全局页表中找到了

所请求的页面，就直接返回这一页，同时更新该页的使用情况统计信息。如果没有找到这一页，则把该页读入本地集合（一个空页）中。如果没有可用的空页（例如，本地集的大小超过了最大阈值），就要根据本地集所指定的页面置换规则，替换一个已经存在的页面。一项详细的模拟研究发现，使用DBMIN比使用频繁集访问算法的吞吐量高出7到13个百分点。

4.1.3 域、记录和文件

页的概念是一个很好的抽象，有利于理解数据在不同内存设备之间的移动。而在高层次的交互中，将数据看作用文件、记录和域这种层次结构组织起来的集合能更有效地进行交互。文件是记录的集合，一个文件（可能）跨越多个页面。一个页面是槽 (slot) 的集合，每个槽包含一条记录，每条记录都是相同或不同类型的域的集合。

在COUNTRY表中的记录大小是size (name) + size (cont) + size (pop) + size (GDP) + size (life-exp) + size (shape) ≤30 + 30 + 4 + 4 + 4 + 8 = 80字节。其中，假定给一个整型数据分配4字节的存储空间，给指针分配8字节存储空间，指针指向shape空间数据类型的实际存储位置。为了简化起见，该计算假设给varchar（N）数据类型分配N个字节，尽管实际上它可能用更少的空间。COUNTRY表中有6条记录，需要480字节的存储空间。如图4-1所示，它们可以存储到磁盘的同一个扇区中。类似的，CITY表的每条记录大小是73字节，存储9条记录需要657字节，假设每个扇区512字节，则可以存储到磁盘的两个扇区中。每个扇区还包含格式信息，用于识别空闲空间、每条记录的开头和可能的域、下一个扇区的地址等。可以指定某些扇区作为查找各个表第一个扇区的目录。图4-1忽略了实际存储的很多细节，但仍有一些基本信息可以说明在二级存储中如何进行表的物理存储。

使用这些不同的参数，就出现了许多针对特定应用来专门组织域、记录和文件的方法。例如，一条记录的域可以是定长或变长的；文件中的记录可以是有序或无序的；文件可以组织成链表或页面目录。每种组织方法都经过了彻底的检验，各有优劣，它们的详细介绍可以在任何一本介绍数据库的书中找到［Ramakrishnan，1998］。

二进制大对象（BLOB）域类型在空间数据库的发展中起了重要作用。传统的数据库不能显式地处理点、线和多边形这些复杂的数据类型，但它们却支持将复杂对象转换成二进制表示并存储到BLOB域中。通过这个方法RDBMS就可以对复杂数据类型进行管理并提供事务支持。例如，Oracle的RDBMS提供了一个LONG RAW域来存储超过256字节的字节串。尽管如此，BLOB域在技术上还不能算作一种数据类型，因为RDBMS将一个BLOB视作没有任何结构的无格式数据。特别是，BLOB域上没有可用的查询操作。

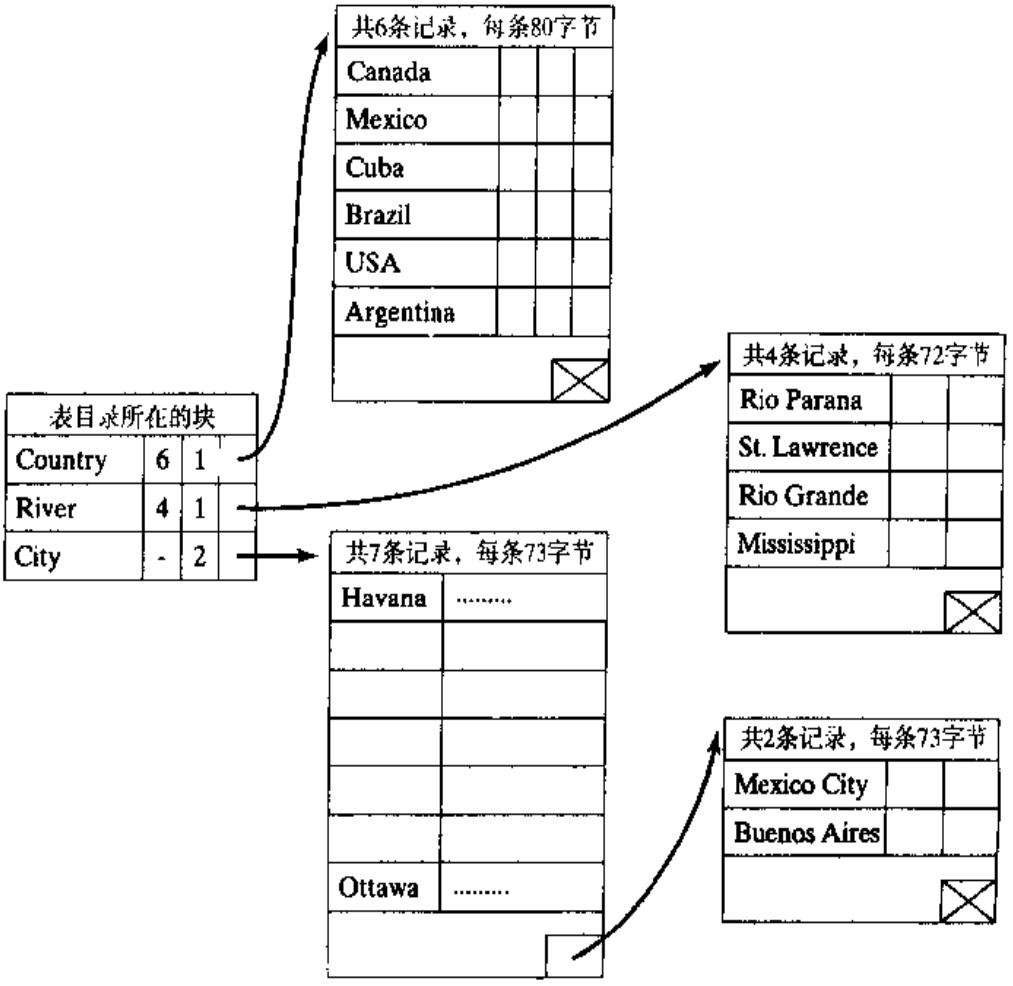


图4-1 将记录从Country、City和River表映射到磁盘页

4.1.4 文件结构

文件结构指文件中记录的组织形式。最简单的组织形式是无序文件（unordered file），也叫堆(heap)，其中记录并没有特定的顺序。图4-1中存储RIVER表的文件就

是一个无序文件。根据给定的关键码（如name）查找一条记录需要扫描文件中的记录。在最坏情况下，文件的所有记录都要被检查，所有存储该文件数据的磁盘页面都要被访问。平均来说，需要检索一半的磁盘页面。无序文件的主要优点是在进行插入操作时可以很容易地在文件末尾插入一条新记录。

更成熟的文件组织形式包括散列文件（hashed file）和有序文件（ordered file）。散列文件组织使用散列函数把记录分到一系列散列单元中。散列函数将事先选择一个主码域（如*city.name*）的值映射到一个散列单元中，它采用非常简单的计算完成这项工作。图4-2给出了一个散列文件，它有四个散列单元，每个单元存储在一个独立的磁盘页面中。对于小于等于6个字符的名字散列函数返回1；对于7至8个字符的名字散列函数返回2；对于9至10个字符散列函数返回3；对于11个以上字符的名字散列函数返回4。城市的名字通过散列函数分别映射到散列单元1到4中。散列函数的可取之处在于它能够把数量大致相同的记录放入每个散列单元中。散列文件这种组织方式对于点的查询、插入和删除操作都非常有效，如果能选择适当的散列函数来组织文件，可以在一个常数时间（例如两次磁盘访问）内完成查询，而与文件中记录的个数无关。散列文件组织方式并不适合范围查询，例如，查找所有名字以

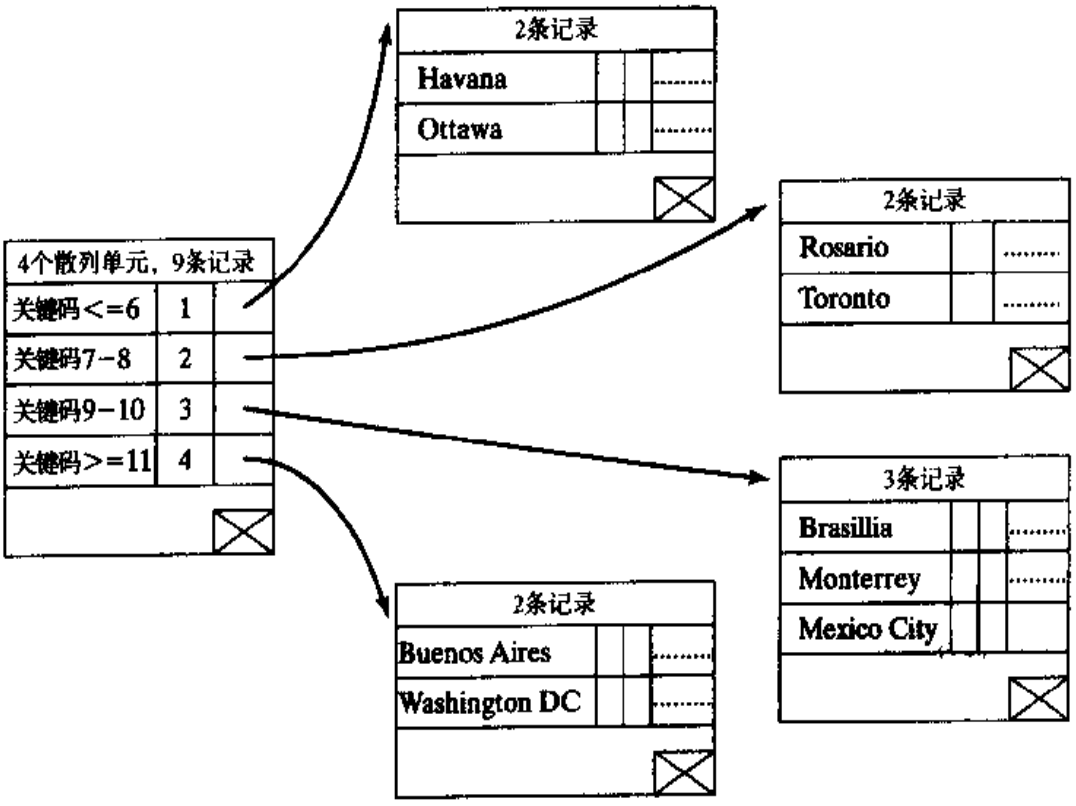


图4-2 City表的散列文件组织方式

字母“B”开头的城市的详细信息，因为所有散列单元中都可以包含符合这个条件的记录。

有序文件根据给定的主码域对记录进行组织。图4-3给出了以*city.name*为主码域的City表的记录的有序文件组织方式。可以使用折半查找算法根据给定主码属性值查找记录。范围查询也可以用折半法查到第一个符合条件的记录，然后扫描后续记录。折半查找算法的代价随文件中记录数的增加呈对数增长。在一个有100万条记录的文件中，要查找包含唯一城市名字的记录，则需要访问$\log_2(10^6) = 20$次磁盘。实际的代价可能会低一些，因为每个磁盘页面包含多条记录。

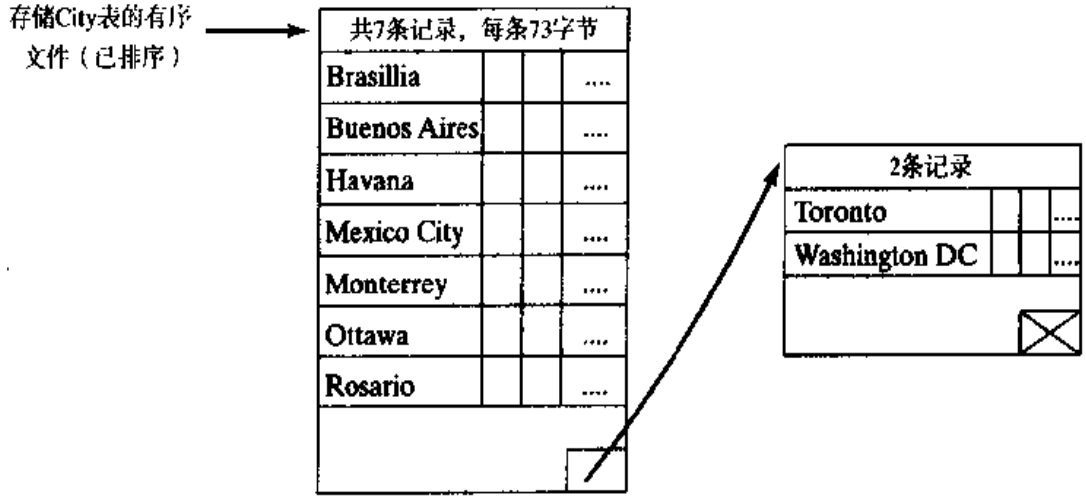


图4-3 City表的有序文件组织方式

我们注意到，有序文件组织方式不能直接应用在空间领域，例如，除非对多维空间中的点定义一个全序，否则无法对城市的位置排序。有序文件组织方式还可以根据对空间数据集的文件组织方式而概括成空间聚类。

4.1.5 聚类

按其最抽象的形式来说，聚类的目的就是降低响应常见的大查询的寻道时间（t_s）和等待时间（t_l）。对于空间数据库来说，这意味着在二级存储中，空间上相邻的和查询上有关联性的对象在物理上应当存储在一起。SDBMS可以支持三种聚类[Brinkhoff and Kriegel，1994]，用于提供有效的查询处理：

1）内部聚类（internal clustering）：为了加快对单个对象的访问，一个对象的全

部表示都存放在同一个磁盘页面中，这里假设它小于页面的空闲空间；否则，这个对象就要存储在多个物理上连续的页面中。在这种情况下，对象占用的页面数至多比存储该对象所需的最小页面数大一。

2) 本地聚类（local clustering）：为了加快对多个对象访问的速度，一组空间对象（或者近似）被分组到同一页面。这种分组可以依照数据空间中对象的位置（或近似）来施行。

3) 全局聚类（global clustering）：与本地聚类相反，一组空间邻接的对象并不存储在一个而是多个物理上邻接的页面中，这些页面可以由一条单独的读命令访问。

空间聚类技术的设计比传统聚类技术要复杂得多，因为在空间数据所处的多维空间中根本没有天然的顺序。事实上，存储磁盘从逻辑上说是一维的设备，这使问题变得复杂。因此需要一个从高维空间向一维空间的映射，该映射是距离不变（distance-preserving）的，这样空间上邻近的元素映射为直线上接近的点，而且一一对应，即空间上不会有两个点映射到直线的同一个点上。为达到这一目标，提出了许多映射方法（它们都不能完全理想地满足这一目标）。最突出的方法包括*Z*序（Z-order）、格雷码（Gray code）和*Hilbert*曲线。为了简化起见，我们先不讨论更为复杂的空间聚类方法，例如，最小割边图划分（6.5.2节）和几何学方法（4.2.2节），它们计算代价更大，但可以在批量装载中提供更好的聚类。

现在来考察用置换规则如何统一生成Z曲线和Hilbert曲线［Asano et al., 1997］。图4-4和4-5显示了构造过程。首先将$N \times N$网格的区域作为一个单元。进行迭代，在第i次迭代中，$i = 0, \ldots, n-1$（对于$N = 2^n$），我们把$N \times N$网格的区域分割成$2^i \cdot 2^i$块，每块大小是$2^{n-i} \times 2^{n-i}$。

这样，在每次迭代中，所有单元都被替换成四个大小为$2^{n-i-1} \times 2^{n-i-1}$的块。此外，还可以对块进行旋转。两个点表示单元中曲线的入口和出口点。在单元中并没有进一步精炼，两个点都重合在该单元的中心；在每次精炼中，点都位于精炼后分区相应的角单元上；最后，结果集右侧的点之间的线依次表示后续块。

1. Z曲线和Hilbert曲线算法

空间填充（space-filling）曲线是利用一个线性顺序来填充空间，可以获得以下从一端到另一端的曲线，如图4-4所示。

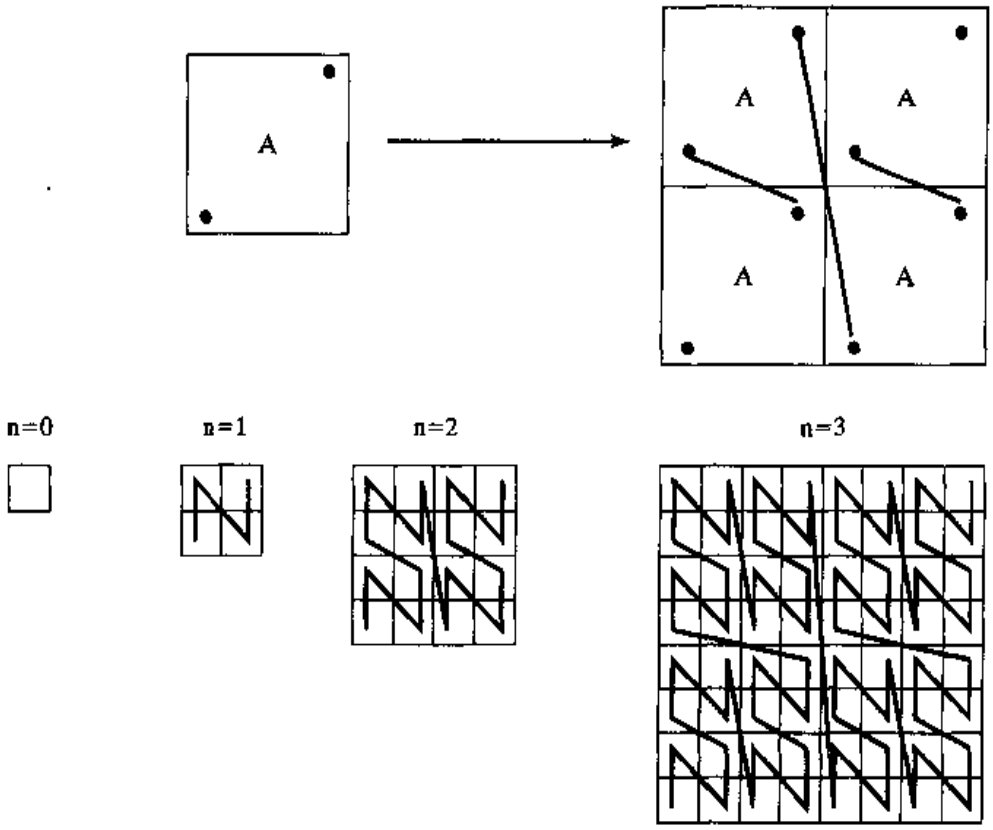


图4-4 生成一条Z曲线［Asano et al.，1997］

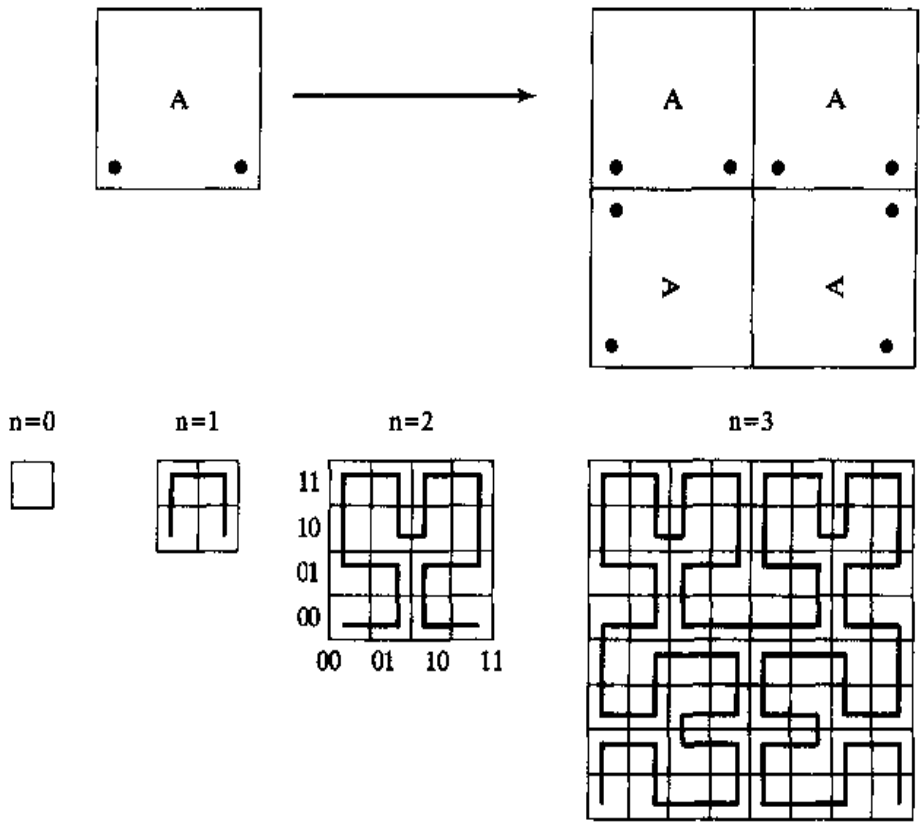


图4-5 生成一条Hilbert曲线［Asano et al.，1997］

下面我们介绍生成Z曲线和Hilbert曲线的算法［Faloutsos and Roseman, 1989］。

2. Z曲线

1) 读入x和y坐标的二进制表示。

2) 隔行扫描二进制数字的比特到一个字符串，参见图4-6。

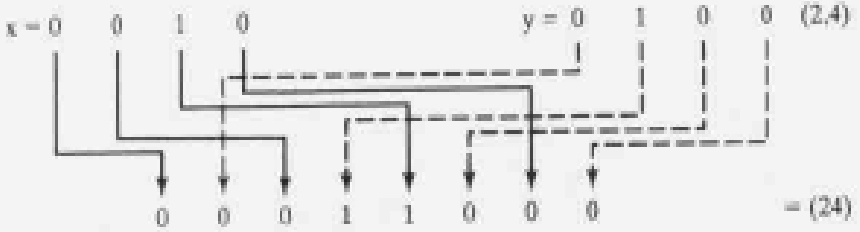
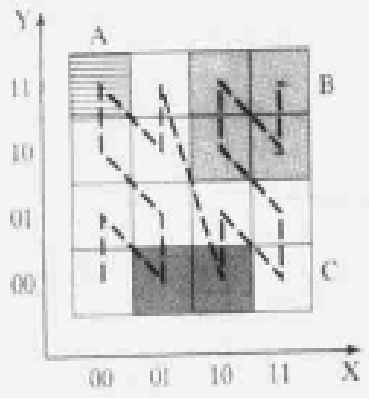


图4-6 计算z值的例子

3) 计算出结果二进制串的十进制值。

图4-7是另一个给出了x、y坐标，查找z值的例子。



对象	点	x	y	隔行扫描	z值
A	1	00	11	0101	5
B	1				
	2				
	3				
	4				
C	1	01	00	0010	2
	2	10	00	1000	8

图4-7 查找z值的记录

3. Hilbert曲线

1) 读入x和y坐标的n比特二进制表示。

2) 隔行扫描二进制比特到一个字符串。

3) 将字符串自左至右分成2比特长的串s_i，其中$i = 1, \ldots, n$。

4) 规定每个2比特长的串的十进制值d_i，例如“00”等于0，“01”等于1；“10”等于3；“11”等于2。

5) 对于数组中每个数字j，如果

- $j=0$ 把后面数组中出现的所有1变成3，并把所有出现的3变成1。
- $j=3$ 把后面数组中出现的所有0变成2，并把所有出现的2变成0。

6) 将数组中每个值按步骤5转换成二进制表示（2比特长的串），自左至右连接所有的串，并计算其十进制值。

图4-8是使用以上算法进行转换的一个例子。

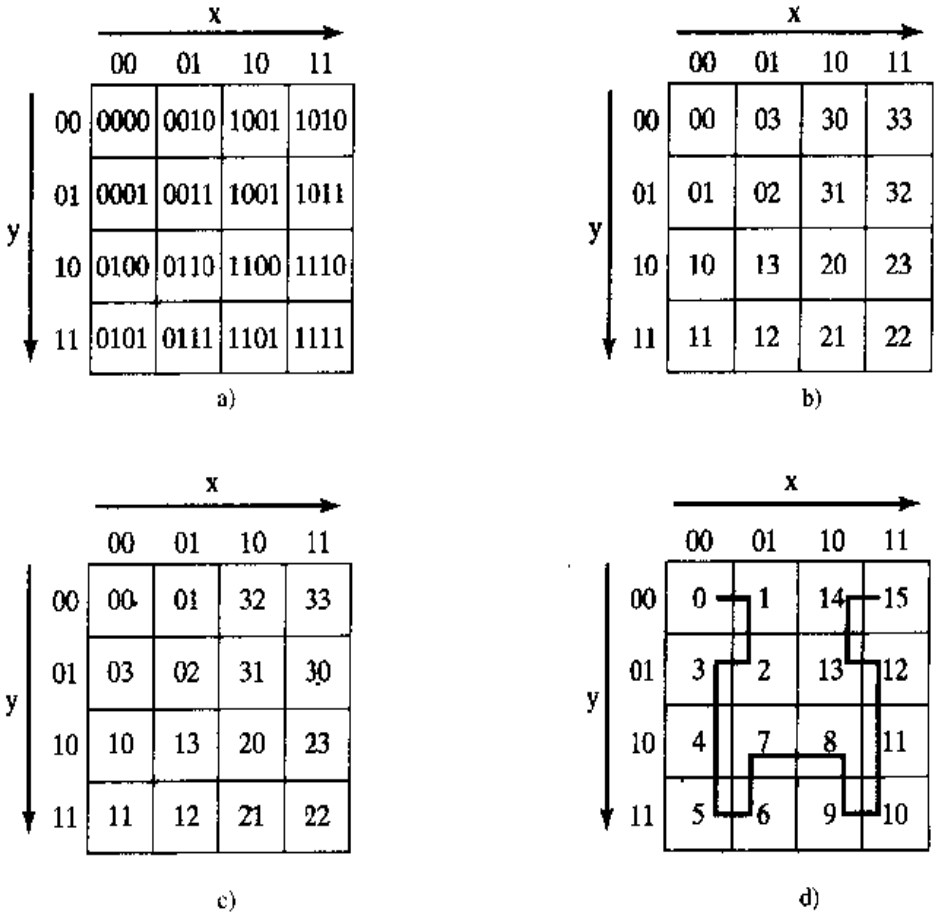


图4-8 Hilbert曲线转换的例子

4. 磁盘访问的度量

为了分析Z曲线和Hilbert曲线的聚类性质，我们假设一个有限粒度（单元个数）的多维空间，其中每个点都对应一个网格单元。曲线为每个单元指定一个整数值。在理想的情况下，这种映射会带来更少的磁盘访问。然而，磁盘访问的次数依赖于很多因素，例如磁盘页面容量、分割算法、插入顺序等。对一个给定查询，这里将采用聚类或连续行程（continuous run）或在给定查询代表的子空间中每个网格点的

散列单元平均数，来作为度量空间填充曲线聚类性能的标准［Moon et al.，1996］。如果每个网格点都映射到一个磁盘块，那么这种度量刚好对应于涉及附加寻道时间的不连续的磁盘访问数量。例如，如果需要访问10个散列单元来回答一个查询，这10个散列单元分别是：2、4~8、10~14、16。在这种情况下，有四个聚类。另一方面，如果被访问的散列单元是9~13、15~19，那就只对应两个聚类。从聚类的角度说，第二种安排从性能上要比第一种高50%，如图4-9所示。

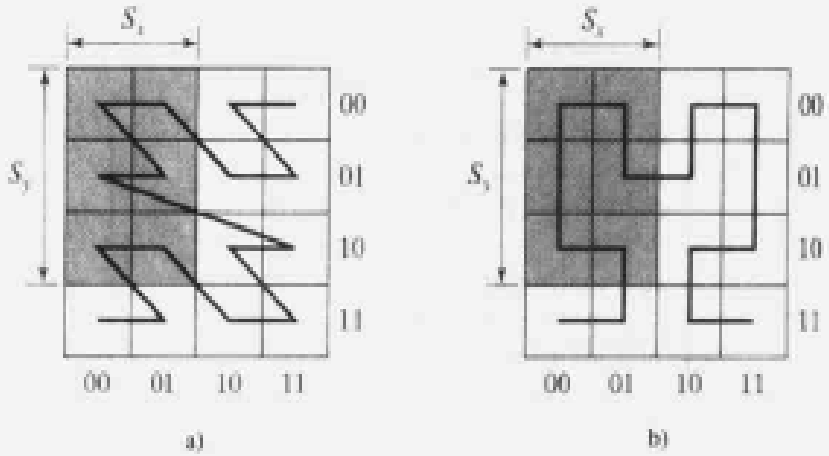


图4-9 聚类的图示：a) Z曲线的两个聚类。b) Hilbert曲线的两个聚类

请注意，Hilbert曲线的方法要比Z曲线好一些，因为它没有斜线。不过，Hilbert算法和精确入口点及出口点的计算量都要比Z曲线复杂。

5. 区域处理

要把一个N维空间中的区域转换成点，可以对扩展对象的最小正交外包矩形使用维数的2倍（2N）次Z曲线。一个区域通常划分成一个或多个片（块），每个块可由一个z值描述。块（block）是原始图像由四叉树分割的一个或多个部分的正方形区域。四叉树分割是递归地把空间分成四个相等的部分。像素显然是1×1块。例如，图4-7中记为C的区域可划分成C_1和C_2两个像素，其z值为：

$$z_{C_1} = 0010 = (2)_{10}$$

$$z_{C_2} = 1000 = (8)_{10}$$

图4-7中记为B的区域包括四个像素，它们的z值有公共前缀11；在这种情况下，B的z值刚好就是这个公共前缀：

$$z_B = 11$$

或者使用“*”代表“任意字符”：

$$z_B = 11**$$

一个大的区域可以分成许多片；从效率的角度考虑，通常用一个粗分辨率区域近似表示这样的区域。

总之，每个对象（和范围查询）都可以用该块的z值唯一地表示（或者用它的块的最大和最小z值表示）。每个这样的z值都以（z值、对象id、其他属性……）的形式作为记录的主码，并且可以插入到一个主码文件结构中，例如B+树。可以用相同方法处理在同一地址空间上的其他对象；它们的z值将插入到同一棵B+树。这些块可以用于高效处理范围查询。

6. 访问方法的算法

使用Z序方法可以处理我们早先列出的所有查询：

- 点查询：使用二分法在排序文件中查找给出的z值，或在基于z值的B树索引上使用B树搜索。
- 范围查询：查询形状可以翻译成一组z值，就像它是一个数据区域一样。通常，我们选择它的近似表示，尽量平衡z值的数量和近似中出现额外区域的数量。然后搜索数据区域的z值以匹配z值。这种匹配的有效性由以下几点可以看出：用z_1和z_2分别代表两个z值，其中z_1是较短的一个，并未失去一般性；对于相应的区域（比如块）r_1和r_2，只有两种可能：1）如果z_1是z_2的前缀（例如，$z_1 = 1***$，$z_2 = 11**$或$z_1 = *1**$，$z_2 = 11**$），则r_1完全包含r_2；2）两个区域不相交（例如，$z_1 = *0**$，$z_2 = 11**$）。
- 最近邻居查询：Z序空间中的距离并不能很好地对应原始的X-Y坐标空间中的距离。可以采用Z序的B树来处理最近邻居的查询。首先，计算查询点p_i的z值，并从B树中找到数据点p_j和最接近的z值。然后计算p_i与p_j之间的距离r，以p_i为中心r为半径进行范围查询。我们检验所有得到的点并返回与查询点距离最短的那个点。
- 空间连接：空间连接算法是范围查询算法的一般化。设S是空间对象集（例如lakes），R是另一个对象集（例如railway-line segment）。空间连接“找出所有

穿过湖的铁路”将按如下方法进行处理：将集合S的元素转化成z值并排序；将集合R的元素也转化成排序后的z值列表，合并两个z值表。其中，在确定交叠时要小心处理代表“任意”字符的“*”。

采用Z序的B树会有一些缺陷。一个问题和连接操作有关：如果网格的结构不一致，索引的分区就不能直接连接，为了易于处理空间连接就不得不重新计算索引。作为空间填充曲线，Z序的另一个缺点是它有很长的对角线跨越，连接这些跨越的连续z值在X-Y坐标系中距离很大。Z序的空间聚类可以通过使用Hilbert曲线来改进。

4.2 空间索引

索引文件是用来提高数据文件查询效率的辅助文件。索引文件中的记录只有两个域，即码值和数据文件中的页面地址。索引文件中的记录通常是有序的（可能使用一个空间填充曲线），还可以用专门搜索数据结构来进一步组织，如使用B树、R树、Grid文件等。图4-10是一个city表的二级索引文件，以name域作为主码域。由于索引文件中的记录只有38字节，假设磁盘页面地址是8字节，所以这9条记录可以放到一个磁盘页面中。索引记录被排序，并且包括含有适当记录的数据文件的磁盘页面地址。数据文件本身可以是不按关键码排序的。

如果数据文件的记录是按主码排序的，那么索引就只需要保存数据文件的每个磁盘页面第一个主码域值，如图4-11所示。这种索引被称为主索引。

由于索引文件中的记录是有序的，那么即使数据文件无序，也可以使用折半查找来搜索索引文件。此外，索引文件日益缩小（根据磁盘页面数量），因此也加快了搜索速度。一旦找到一条适当的索引记录，就可以通过索引记录所指向的数据文件磁盘页面，获得数据记录。

空间索引结构用一组桶（bucket）（通常对应二级存储的页面）来组织对象。每个桶有一个关联的桶区域，即包含了存储在桶中全部对象的一部分空间。桶区域通常是矩形的。对于点数据结构来说，这些区域通常是不相交的，它们将空间分区使每个点正好属于一个桶。对于一些矩形数据结构来说，桶区域可能是交叠的。

提供空间索引主要有两种方法：

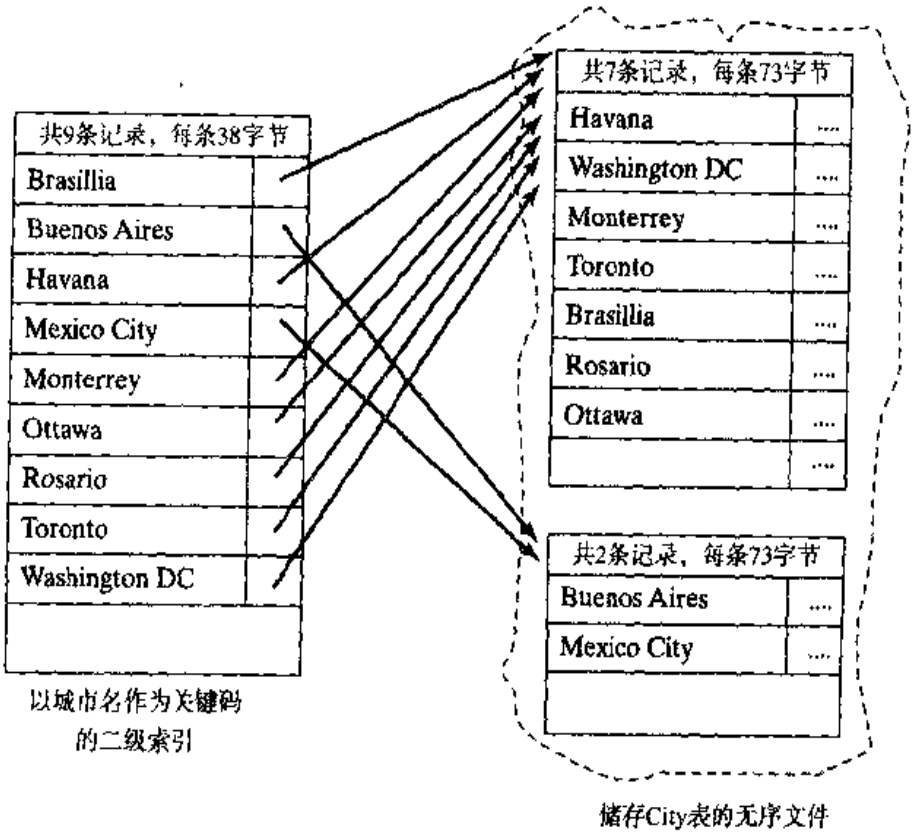


图4-10 City表中的二级索引

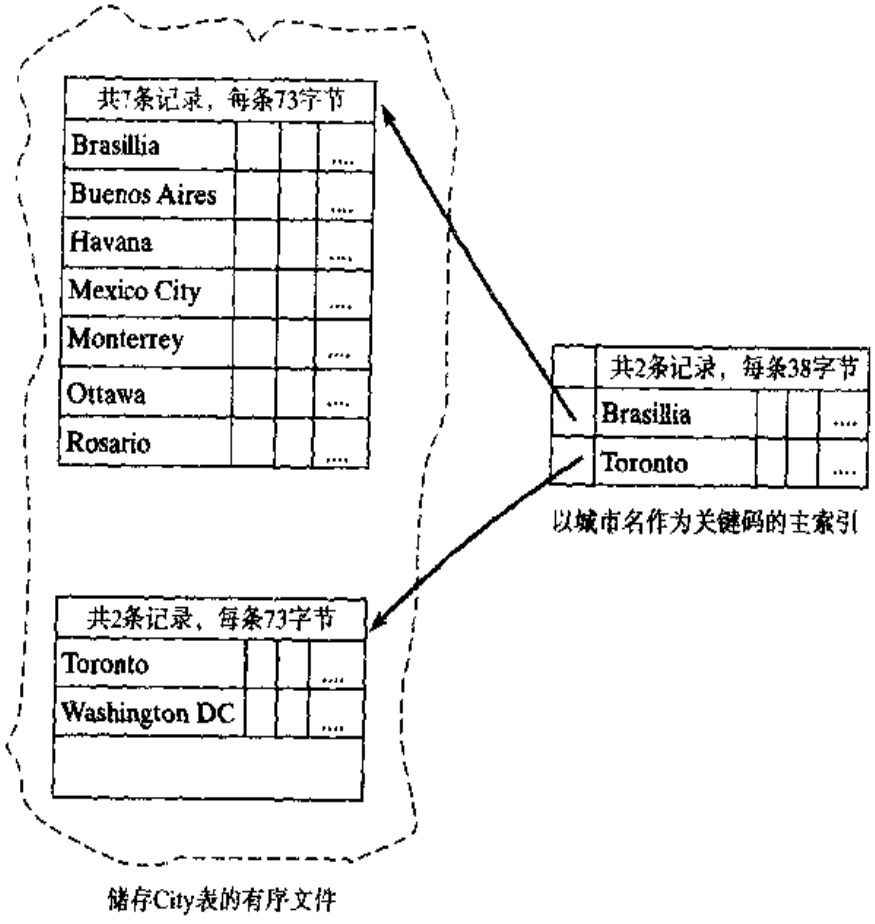


图4-11 City表中的主索引

1) 在系统中加入专门的外部空间数据结构，为空间属性提供如同B树之于线性属性的功能。

2) 使用空间填充曲线（如Z序、Hilbert曲线）将空间对象映射到一维空间，以便空间对象存储在标准的一维索引（例如B树）中。

除了空间选择外，空间索引还支持其他的操作，例如空间连接、查找最符合待查询值的对象等。

4.2.1 网格文件

最简单的存取多维点方法之一就是利用固定网格（fixed grid）存取结构（例如经度-纬度网格）。如图4-12所示，固定网格方法将嵌入的n维空间划分成同等大小的桶。实现固定网格的数据结构是一个n维数组。单元中的点可以存储在一个动态结构中（例如链表）来处理溢出。这种结构对于处理静态一致分布式数据（例如卫星影像）很有用。然而，固定网格结构是固定的，它的目录稀疏而巨大。

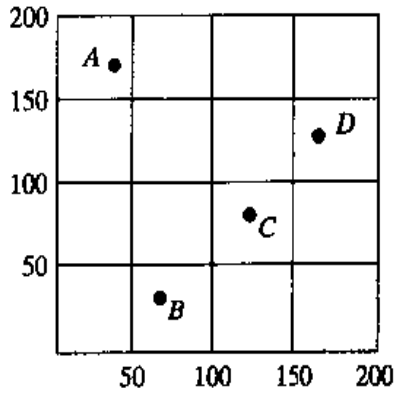


图4-12 点(A、B、C、D)的固定网格结构

为了获得更好的灵活性和性能，［Nievergelt et al., 1984］引入了网格文件（grid file）。网格文件对于精确匹配和部分匹配检索都有相对较好的I/O性能。网格文件采用多属性索引技术来分割嵌入的n维空间。从局部上说，它用一个网格方法，即在分割方向上划分所有区域。使用网格文件的目标就是为了实现二次磁盘访问原则：一次访问获得目录项，另一次访问得到实际桶以获取实际记录。更具体地说，网格文件有两部分。一部分包括一个n维网格目录（grid directory），目录中每一项指向一个数据桶。图4-13是一个$n = 2$的n维网格目录。第二部分是由称为线性比例（linear

scale）的一维数组组成的结构。这个数组用来标识网格目录的索引，这个索引引用了包含对象（记录）的块或桶。

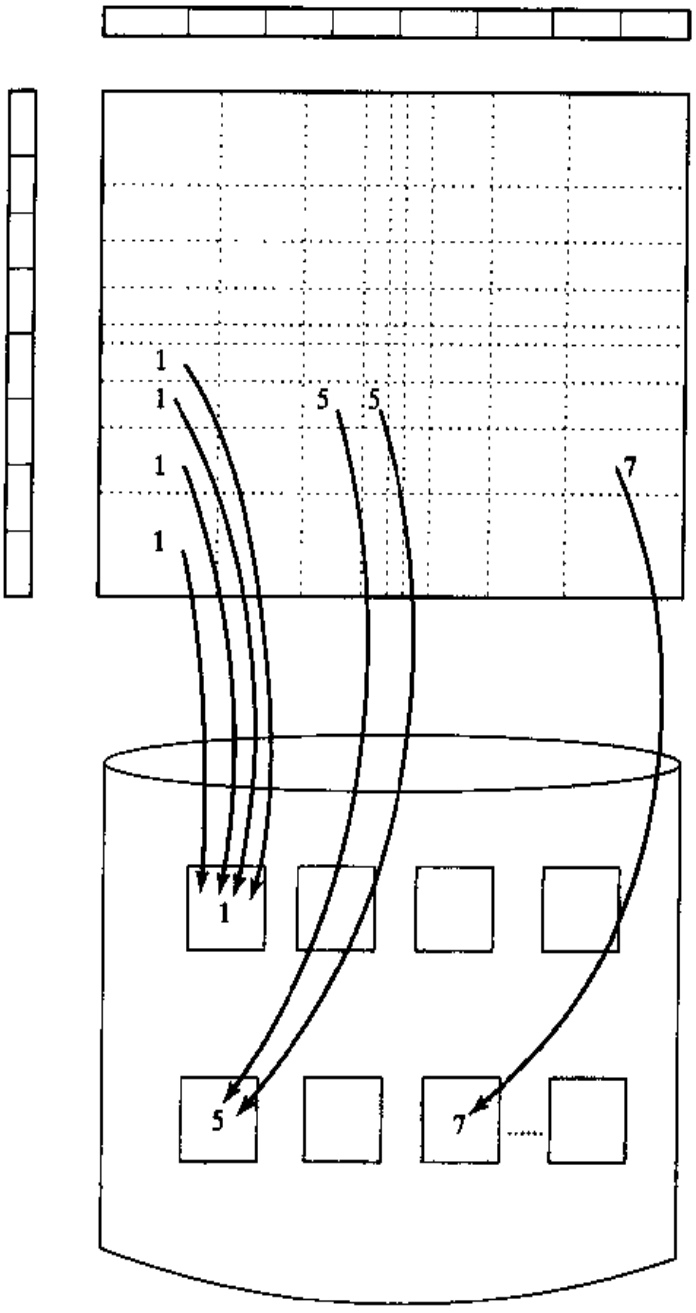


图4-13 一个二维网格目录和数据页面

为了说明使用网格文件进行搜索的过程，考虑图4-14所示的结构。图中的桶有*A*、*B*、*C*、*D*、*E*、*F*、*G*、*H*、*I*、*J*、*K*、*L*、*M*。注意，其中网格目录项(1，1)和(1，2)“共享”同一个桶。这就意味着属于该区域的点或记录存储在同一桶中。对于目录项(2，3)和(3，3)，以及(1，4)和(2，4)来说也是如此。现在假设给出了一个点(60，

80)，并希望定位到存储$X = 60$、$Y = 80$的记录/对象的桶。对应于x轴的线性比例生成索引3 ($X = 60$)，对应于y轴的线性比例有索引4 ($Y = 80$)。网格目录项(3，4)是桶C，因此，$X = 60$、$Y = 80$的对象定位到桶C。

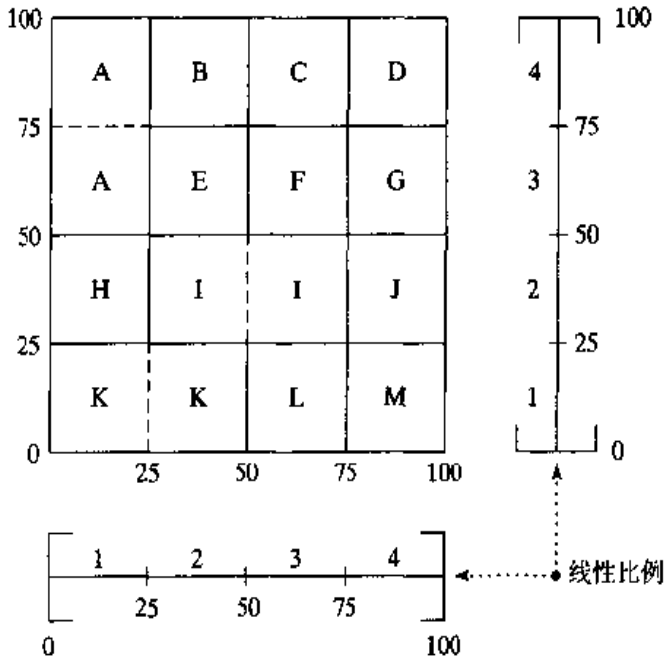


图4-14 线性比例的网格文件

根据维数和局部性，不仅可以用一个超平面分区（或沿着某一维），还可以在分裂方向上扩展分裂到所有的区域。网格文件的搜索时间是很短的。对于精确匹配查询来说，只作一次对目录的磁盘访问和一次数据访问。然而，网格文件本质上会造成目录非常松散，因而浪费主存缓冲区和二级存储。换言之，超立方体可能只有非常少的（或没有）记录，邻近目录项可能指向同一个数据块。这样就意味着，对于局部匹配和范围查询来说，需要扫描很多只有少量数据块的目录项。

4.2.2 R树

最早支持扩展对象存取方法之一的是Guttman提出的R树［Guttman，1984］。R树是一个高度平衡树，它是B树在k维上的自然扩展。R树中用对象的最小外包矩形（MBR）来表示对象。R树有以下几条特性：

1) 每个叶结点包含m至M条索引记录（其中$m \leqslant M/2$），除非它是根结点。

2) 一个叶结点上的每条索引记录了（I，元组标识符），I是最小外包矩形，在空间上包含了所指元组表达的k维数据对象。

3) 每个非叶结点都有m至M个子结点，除非它是根结点。

4) 对于非叶结点中的每个条目（I，子结点指针），I是在空间上包含其子结点中矩形的最小外包矩形。

5) 根结点至少有两个子结点，除非它是叶结点。

6) 所有叶结点出现在同一层。

7) 所有MBR的边与一个全局坐标系的轴平行，如图4-15所示。

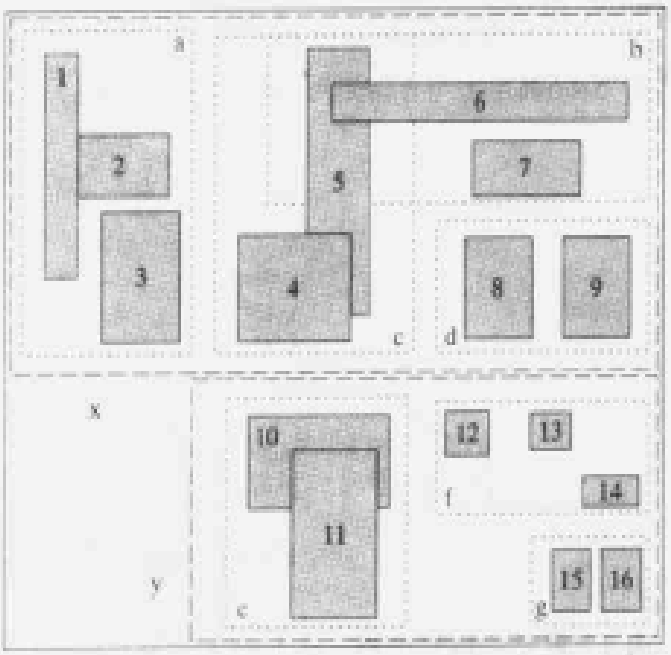


图4-15 一个空间对象集合

树的每个结点对应一个磁盘页面。一个叶结点包括一组项，其格式为（I，元组标识符），其中I为MBR，元组标识符是数据库中存储对应于MBR的对象的元组唯一标识符。用$I = (I_0, \cdots, I_{k-1})$表示$I$，其中$I_i$在沿方向$i$的一个闭合区间$[a, b]$上。也考虑不定区间，用$a$、$b$或者两个都是无穷大来表示。

非叶结点由多个格式为（I，子结点指针）的项组成，其中I是子结点指针（child-pointer）指向的更低层结点项中所有矩形的MBR。树中每个结点最多有M个条目，最少有m个（其中$m \leqslant M/2$），除非它是根。根结点至少有两个子结点，除非它是叶。

图4-15是在二维空间中的一组空间对象（MBR）。图4-16是对应图4-15中一组MBR的R树。树的每个结点最多有3个项。树中叶结点上的对象用阴影标出。

点查询和范围查询在R树中可以用自顶向下递归的方法进行处理。查询点（或区域）首先同根结点中每个项（I，子结点指针）进行比较。如果查询点在I中（或查询区域与其交叠），则查找算法就递归地应用在子结点指针指向的R树结点上。该过程直至R树的叶结点为止。使用叶结点中选出的项来检索与选中空间主码关联的记录。

例如，考虑搜索图4-16中与矩形5交叠的对象。根结点中的项x与矩形5交叠，搜索沿着R树的这一分支继续。在x中，项b和c与矩形5交叠，因而继续搜索。最后矩形4、5和6确定为与查询矩形5交叠的叶结点项。

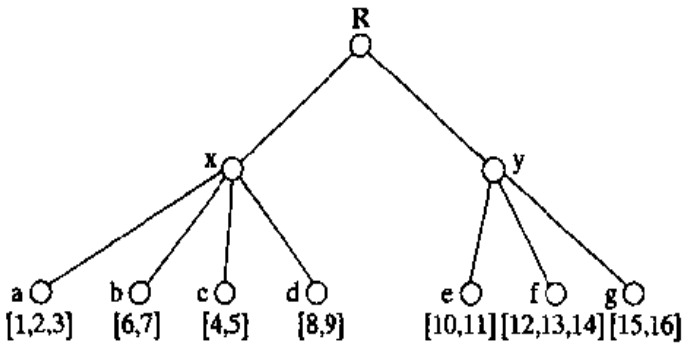


图4-16 R树的层次结构

由于R树是一棵平衡树，在与插入对象对应的叶结点已满的情况下，插入操作可能会导致结点向根部分裂。叶面分裂算法是非常复杂的，不过，R树已经在许多商用数据库系统中实现了，它们支持通用的存取方法和合理大小（1024字节）的磁盘页面。

R树的最大层数是 $\lfloor \log_m N \rfloor - 1$，其中$N$是树中项的总数。在最坏情况下，除根之外，结点空间的利用率是m/M。如果m大于3或4，树将水平扩展，这样，几乎所有的空间都用于存储包含索引记录的叶结点。m值增大通常就意味着R树的深度相对变浅。例如，一个索引1亿个矩形的R树，当$m = 100$时深度是5。R树是一个动态的结构，还允许存储异构的对象，例如存储图4-15所示的点、线和非零区域等对象。

搜索的性能取决于两个参数：覆盖（coverage）和交叠（overlap）。树的某一层

的覆盖是指这一层所有结点的MBR所覆盖的全部区域。这样，覆盖是对静态空间的间接衡量，或者是树覆盖的空白区。树中某一层的交叠是指在该层上被多个与结点关联的矩形所覆盖的全部空间区域。交叠使得查找一个对象时必须访问树的多个结点。R树的这个问题意味着，即使尽量减少交叠，但搜索操作的最差性能仍然是无法估量的。

从上面的讨论可以看出，若要得到一个高效的R树，覆盖和交叠都应该最小，而且交叠的最小化比覆盖的最小化更加关键。为解决这个问题，产生其他基于R树的变种和备选结构。例如packed R树、R*树和R+树。

packed R树方法假定数据是相对静态的，并且数据对象在构造树之前就已知。数据库第一次建立时，可以为将覆盖和交叠最小化而进行有效的组织。此后的插入和删除都遵循Guttman的原始R树结构。R*树是R树的一个变种，它和对树的中间结点中每个最小外包矩形的区域、边缘和交叠的复合优化有关。

在R+树中，空间对象的MBR可能被树中非叶结点的矩形分割。R+树有如下特点：

1) 对于中间结点的每个项（*I*，*child-pointer*），当且仅当*R*被*I*覆盖时，以*child-pointer*指向的结点为根的子树包括一个矩形*R*。唯一的例外是当*I*是一个叶结点的矩形。在这种情况下，*R*与*I*只交叠。

2) 对中间结点的任何两个项（I_1，$child\text{-}pointer_1$）和（I_2，$child\text{-}pointer_2$），I_1与I_2之间的交叠是零。

3) 根至少有两个子结点，除非它是叶结点。

4) 所有的叶结点都在同一层。

中间结点的所有矩形都是不相交的，因而中间结点项之间产生零交叠。如果一个对象的MBR被两个或多个R+树高层结点中的矩形分割，那么与这些非叶结点中矩形相联系的每个项都有指向这个对象的一个后继的叶结点。这样，树的高度增加（虽然只是轻微的），但搜索操作的性能会有很大提高。

图4-17是为前面图4-15中的一组空间对象建立的R+树。图4-18显示了新的MBR。假设每个结点最多有3项。指向两个对象的指针是复制的，因为对应的每个对象都与两个叶结点的矩形交叠。需要注意的是，围住一个公共对象的那些外包矩形应该

彼此相接，例如，对象5是MBR b和c的公共对象，但是为了清晰起见，它们在图4-18中没有相交。

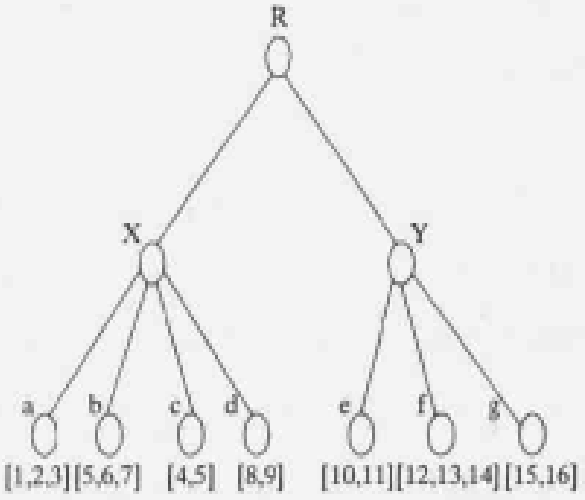


图4-17 R+树的层次结构

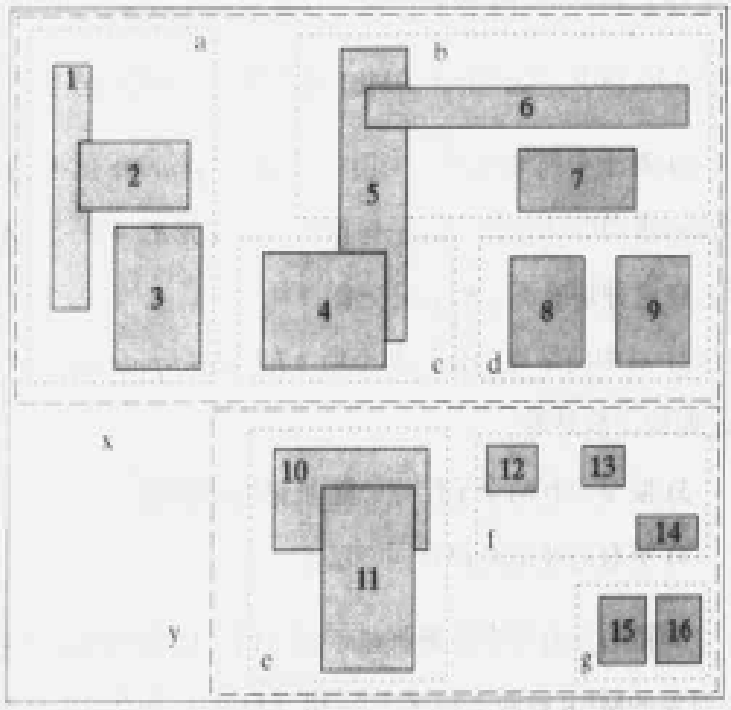


图4-18 R+树内部结点的矩形

对于高度动态的数据来说，R+树比packed R树更适用，因为它们确保操作能持续高效。Packing算法也可以应用于R+树，以便对树中的数据进行初始的有效安排。同R树相反，由于R+树的第一个特点，所以它可能会向下分裂。因此，分裂结点的选择是很重要的。

当一个结点溢出时，各自的矩形就被一个超平面分割。这个超平面与k（在k维空间中）中一个方向平行，而且还会有不同的位置。超平面的选择可以基于几个标准，例如减少树的覆盖或高度。对于后者，必须选择能使矩形分割的数量最小的超平面。

4.2.3 代价模型

R树是空间数据库中最流行的索引结构之一，我们可以在文献中找到R树的若干变种，这就出现了一个问题：哪一种变种更好、更为有效？为了比较这些变种，我们首先需要有准确的代价模型。R树和R+树的选择查询性能可以在［Faloutsos and Sellis，1987］中找到。该模型假设数据是一致分布的，并且树的每个结点都被填满（packed树）。以平均结点数和查询窗口大小的函数作为预测查询性能的分析代价模型，可以在［Kamel and Faloutsos, 1993; Pagel et al., 1993b］中找到。这些模型获取在区域、周长和对象个数之间的定量关系，并作为衡量R树性能的一个函数。其中大部分模型将性能当作单独访问结点数的函数，而忽略了缓冲区的使用。在［Leutenegger and Lopez，2000; Theodoridis et al.，2000a］中分别介绍了基于缓冲区的代价预测模型。［Leutenegger and Lopez，2000］扩展了［Kamel and Faloutsos, 1993; Pagel et al., 1993b］中描述的模型，考虑了缓冲区的情况。他们提出了用于统一存取访问和数据驱动访问的代价模型。评估实验表明，少量的缓冲区可以超线性地提高对于结构良好的R树的点查询性能，对于结构不好的R树也能带来线性增长。［Theodoridis et al.，2000a］为选择和连接查询提供的代价模型，作为以连接查询的路径缓冲区为基础的数据性质函数。其评估实验表明，对于一维数据，连接查询可以获得10%~30%的性能增益，而对于二维数据则可以提高50%。

4.3 趋势

4.3.1 用于对象分解的TR*树

对象的近似用来降低对多边形作精确几何检测的数目，对象分解可以提高这种检测的速度。考虑在多边形中点的检测。对于有几千个顶点的多边形来说，这个测

试相当耗费时间。另一方面，只有对象的很少一部分与判断对象是否包含点相关。这样就产生了对象分解的思想，对象分解的基本思想是将对象分解成许多满足一些定量约束的简单局部组件（例如三角形、不规则多边形、凸多边形等）。

使用对象分解，几何检测应只用于对象组件，这要比检测整个多边形更有效率。为了判断哪一个组件与特定检测相关，通常把一个对象的组件在特殊的空间结构中进行组织。

分解策略如下所示：

1) 在预处理步骤中，使用平面扫描算法，将多边形对象分解成最小数量的互不相交的不规则多边形组（如图4-19）。

2) 由于不能在分解过程所产生的不规则多边形集合上定义一个完整的空间顺序，因而无法对这些不规则多边形应用折半查找。所以，可以在空间搜索中应用R树。

3) 为了提高几何检测的速度，［Schneider and Kriegel，1991］提出了TR*树，它是R树的一个变种，主要用于将主存操作最小化并存储被分解对象的不规则多边形。TR*树结构的主要特点是最小化了每个结点项的数量。要把被访问对象的TR*树表示完全装入主存中才能进行空间查询处理。

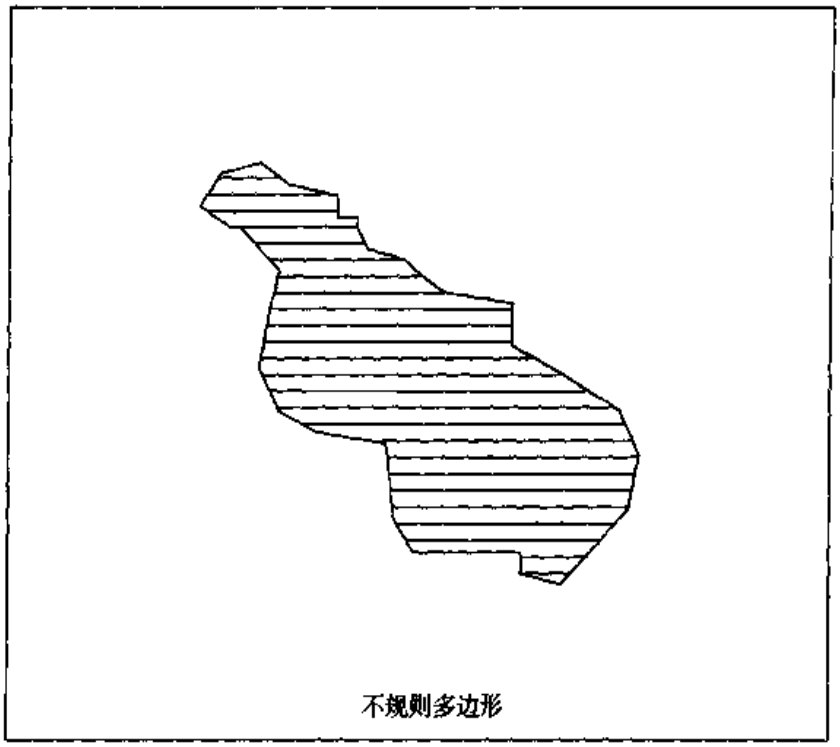


图4-19 分解对象的不规则多边形

4.3.2 并发控制

DBMS要支持多个用户程序同时的或并发的（concurrent）访问一个共享数据库。为了理解DBMS如何管理并发操作，可以将一个数据库看成一个对象的集合，而一个用户程序的执行，或者是事务（transaction），看成一系列对数据库对象的读和写为了提高效率，一个事务的动作可以与其他事务交叉执行，但是要确保包含的事务之间是逻辑偏序或可串行化的。

DBMS中的事务和操作系统执行的C程序之间重要的不同，可以用一个缩写词*ACID*来概括。原子性（atomicity）保证一个事务的所有动作要么都执行，要么都不执行。一致性（consistency）意味着一个事务不会违反数据库中预定义的数据一致性约束。隔离性（isolation）是指一个事务的动作可以在不知道并发执行的其他事务的情况下被理解。持久性（durability）确保一个事务的影响能够在系统崩溃的情况下持续有效。

DBMS通常使用并发控制技术来支持对数据库的并发访问，例如封锁协议（locking protocol）就是一种并发控制技术。如下两个原则定义了被广泛的使用的严格两段锁协议（Strict 2PL）：

1) 对于对象上的读的动作，事务需要在该对象上的一个共享锁。互斥锁支持写的动作。

2) 所有和一个事务相关的锁都在事务完成时释放。

对于分层次索引结构，特别是B+树来说，需要对前面描述的封锁协议进行修改以利用树结构的优点。对于搜索和插入操作，以下叙述说明了修改的基本原则：

- 非叶结点的功能是将搜索引导至存储实际数据的叶结点。因此，如果B树索引每层的结点都已经成为链式(threaded)，搜索操作就不需要任何的锁。
- 如果每一层的索引结点都已经成为链式，插入操作在任何时候都只需要在一个结点上的互斥锁。

对于R树来说，搜索和更新操作和B树的这些操作不一样，原因如下：1) R树的关键码是多维的MBR，其上没有定义线性的顺序，2) 关键码可能交叠。对于搜索操作来说，它意味着查询矩形不得不下降到所有相交或完全包含规定范围的子树中。

此外，它不能确保子结点包含任意感兴趣的关键码，即使它的外包矩形和搜索范围相交也是如此。对于R树中的更新操作，向上传播不仅在结点分裂时出现，在外包矩形更改时也会出现。

一个提高B树并发度的策略是确保在给定时间内只有一个结点被锁定，这是由[Lehman and Yao，1981]提出的。这个策略随后被[Kornacker and Banks，1995]修改以用于R树。R树的另一个并发控制技术由[Chakrabarti and Mehrotra，1999]提出。

B链接树加入了从一个结点指向同一层中右兄弟结点的右链接指针，使它们依据主码排序。即使一个结点分裂，B树搜索过程仍然可以遍历它。搜索过程可以识别出已经分裂的结点，因为分裂会导致结点中最高的关键码小于查询关键码。搜索过程可以根据右链接指针查看由于分裂而新加入的页面。当搜索过程到达一个最高关键码大于查询关键码的结点时，沿着右链接的遍历就结束了。类似的，一个插入过程可以在最初搜索期间避免耦合锁定以到达正确的叶结点。如果页面分裂，在把一个右兄弟结点插入到这一层的链接列表之后，插入过程就可以释放结点上的锁。由于有并发分裂，所以可能需要遵循右链接。

B链接策略依赖关键码的线性顺序来决定子结点何时分裂以及搜索过程需要向右遍历多远。R树中的关键码是MBR，它并没有显示自然线性顺序。

R链接树为每个结点分配一个唯一的逻辑顺序码（logical sequence number，LSN）。LSN随时间单调增长。搜索和插入操作使用LSN识别分裂结点和右链接链遍历的终点。R链接树中每个结点项包括一个码MBR、一个指向子结点的指针和预期的子结点的LSN。当一个结点分裂时，旧的LSN被赋值给新的（右兄弟）结点，并为旧的结点产生一个新的LSN。一个沿树向下的过程可以通过比较结点的LSN和在其父结点所预期的LSN而探测到分裂。在分裂的情况下，沿着右链接指针进行搜索，直到找到一个LSN匹配的结点为止。

R链接树是平衡树。它的索引结点包括一组项和一个右链接r。树中每层的结点通过右链接形成一个单向链表。内部结点项包括一个码矩形k、一个指针p和一个预期的LSN l。对于预期的LSN有两种情况：

1) 一般情况：子结点层l的结构和它的父层$l+1$一致。在一般情况下，指针p指向子结点N。子结点N的右链接指向Null或者另一个结点R，其中R也被父层$l+1$的某个项所指向。在图4-20中，项x指向结点child-1；x的LSN和child-1的LSN相匹配，而且child-1的右链接指向结点child-2，parent-2的项w也指向child-2。

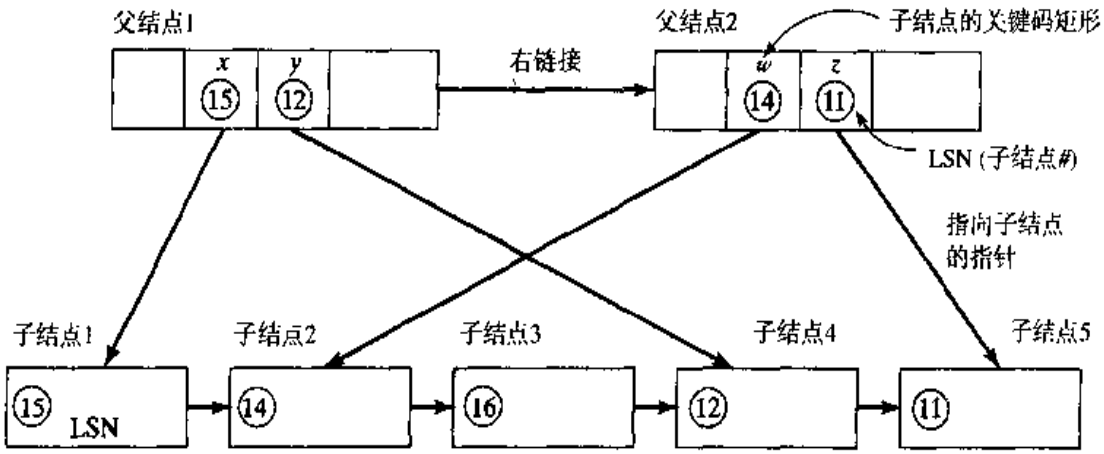


图4-20 一个R链接树的一部分［Kornacker and Banks，1995］

2) 其他情况：子结点层l的结构比它的父层$l+1$新。在这种情况下，指针p指向子结点N，其中LSN (N) >l。子结点层l有一个结点N'，N' 的LSN是l。子结点N'可以通过层l的链接列表中N的右链接到达。考虑图4-20中parent-2的项w。LSN(w)小于LSN(child-2)且与LSN(child-3)相等。child-3可以通过child-2的右链接到达。child-3在父层中还没有项。这种情况是由于child-2结点的分裂产生的，它将通过完成这次分裂反映到父层上。

4.3.3 空间连接索引

空间连接索引是用于优化空间连接查询过程的数据结构。连接索引提高了对更新率低的表进行递归连接查询的性能。连接索引对于一些新的应用是很有意义的，例如拥有海量数据并且限制更新的数据仓库。

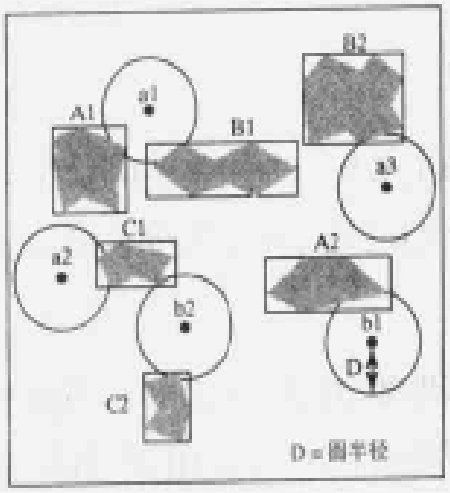
连接索引通常表示为在关系的页面或者其代理之间的一个双向图。当缓冲区的页面数量固定时，连接计算的问题就转换为确定一个页面访问的顺序的问题，按照这个顺序能够以数量最少的冗余页面访问来计算连接。这是一个NP-hard问题，所以不大可能存在一个多项式时间的解决方法。一些参考文献中使用了聚类的方

法，将和连接相关的一个或两个表进行分组以降低总的页面的访问次数。简单的启发式方法就是通过排序将一个表的页面分组或者使用增量聚类方法。

1. 连接索引：基本概念

考虑一个有两个关系的数据库，这两个关系是facility和forest stand。facility有一个代表其位置的点属性，forest stand有一个矩形属性，用外包框表示它的范围，其中表示其范围的矩形可以单独存储。点用地图上的x和y坐标表示。矩形用代表左下角和右上角的点表示。

如图4-21a所示，点$a1$、$a2$、$a3$、$b1$、$b2$代表facility的位置，多边形$A1$、$A2$、$B1$、$B2$、$C1$、$C2$是表示forest stand范围的边界框。每个位置周围的圆形表示一个facility周围方圆D以内的区域。每个forest边界周围的矩形代表forest stand的最小正交外接矩形（MOBR）。图4-21b是这个数据集的两个关系R和S。关系R通过一个唯一的标识符$R.ID$、位置（x、y坐标）和其他非空间属性表示facility。关系S通过唯一的标识符$S.ID$、MOBR和非空间属性表示forest stand。MOBR（X_{LL}，Y_{LL}，X_{UR}，Y_{UR}）用其左下角点(X_{LL}，Y_{LL})和右上角点（X_{UR}，Y_{UR}）的坐标表示。现在考虑如下的查询Q："查找所有距离每个facility方圆D之内的forest stand。"这个查询要求在facility和forest stand关系上，以它们的空间属性为基础进行连接。空间连接比等值连接复杂得多，并且是Θ连接的特殊情况，其中Θ是一个空间谓词，例如：相接、交叠和交叉。查询Q是一个空间连接的例子。



a) R与S的空间属性

关系R
（facility位置）

ID	位置（X，Y）	非空间数据
a1	(7.9, 16.7)	(...)
a2	(3.4, 11.4)	(...)
a3	(19.5, 13.1)	(...)
b1	(18.7, 6.4)	(...)
b2	(9.5, 7.1)	(...)

关系S
（林分边界）

ID	MOBR (X_{LL}, Y_{LL}, X_{UR}, Y_{UR})	Non-Spatial Data
A1	(3, 12.2, 6.6, 16)	(...)
A2	(13.4, 7.7, 19.5, 10)	(...)
B1	(7.6, 12.7, 15, 5.2)	(...)
B2	(15.5, 15, 20.4, 19)	(...)
C1	(5.1, 8.9, 9.1, 10.9)	(...)
C2	(7.5, 2.9, 7, 15.1)	(...)

连接索引

R.ID	S.ID
a1	A1
a1	B1
a2	C1
a3	B2
b1	A2
b2	C1
b2	C2

b) 关系R、关系S的表以及连接索引

图4-21 在两个关系上建立一个连接索引

空间连接算法用来查找满足查询Q的facility和forest stand对。换句话说，如果空间数据很少更新，那么可以利用连接索引将结果子集实体化以便提高未来的查询Q的处理速度。图4-21b所示是一个有两个列的连接索引。连接索引的每个元组表示表$JOIN\ (R,\ S,\ distance\ (R.Location,\ S.MOBR) < D)$中的一个元组。通常，连接索引中的元组可能还包括指向R和S的页面的指针，这些页面上保存有R和S的相关元组。本章为了简化图表，我们忽略了指针的信息。

一个连接索引描述了两个关系的对象之间的关联。假设一个关系的每个元组都有代理（系统为元组、页面等定义的标识符）用于唯一地标识元组。连接索引是一系列成对的代理，其中每对代理标识连接的一个结果元组。参与连接结果的元组由它们的代理给出。在形式上，假设R和S是两个关系。考虑R和S在R的属性A和S的属性B上的连接。连接索引是这种关系的连接的一个抽象。如果F定义了连接谓词，那么连接索引就可以由一个集合$JI = \{(r_i,\ s_j) | F\ (r_i.A,\ s_j.B)\ is\ true\ for\ r_i \in R\ and\ s_j \in S\}$给出，其中$r_i$和$s_j$分别是$R$中第$i$个元组和$S$中第$j$个元组的代理。例如，考虑图4-21中的facility和forest stand关系表。facility 关系和forest stand关系在每个关系的空间属性上进行连接。这个连接的连接索引包括和空间连接谓词相匹配的元组ID。

一个连接索引可以用双向图$G = (V_1,\ V_2,\ E)$来描述，其中V_1包括关系R的元组ID，V_2包括关系S的元组ID。如果连接索引中有一个和$(v_i,\ v_j)$对应的元组，那么边(edge) 的集合E包括一条边$(v_i,\ v_j)$其中$v_i \in R$，$v_j \in S$。双向图用图中相连的顶点建模所有的关联元组。在图中，和结点连接的边叫做那个结点的入射边，一个结点的入射边的数量叫做这个结点的度。

我们用图4-22说明使用线性的码进行等值连接的元组级邻接矩阵和空间连接之间的主要区别。图4-22a是等值连接的邻接矩阵。水平坐标是一个关系中各不相同元组id值；垂直坐标是另一个关系中各不相同的元组id值。阴影区域是点的集合，这些点表示在两个关系的笛卡儿积的所有元组对中，满足等值连接谓词的元组对。白色区域表示不满足等值连接谓词的元组对。图4-22b表示的信息和图4-22a表示的信息相同，其中每个关系中的元组已经根据连接属性排序。注意，对于线性有序的连接属性来说，阴影区域逼近对角线。连接处理算法（例如，归并排序）可以利用这个性质。图4-22c是一个用于Θ连接（如空间连接）的邻接矩阵。注意，连接属性

（例如，空间位置）通常不是线性的，并且没有天然的排列顺序。然而，对邻接矩阵行和列重新排序会产生尽可能多的逼近对角线的点（满足空间连接谓词的对象id对）。经过重新排序后通常会产生一个类似图4-22d所示的邻接矩阵，其中有一定数量的阴影区域仍然远离对角线。空间连接的连接处理算法还要解决如何处理对角线之外的阴影部分。

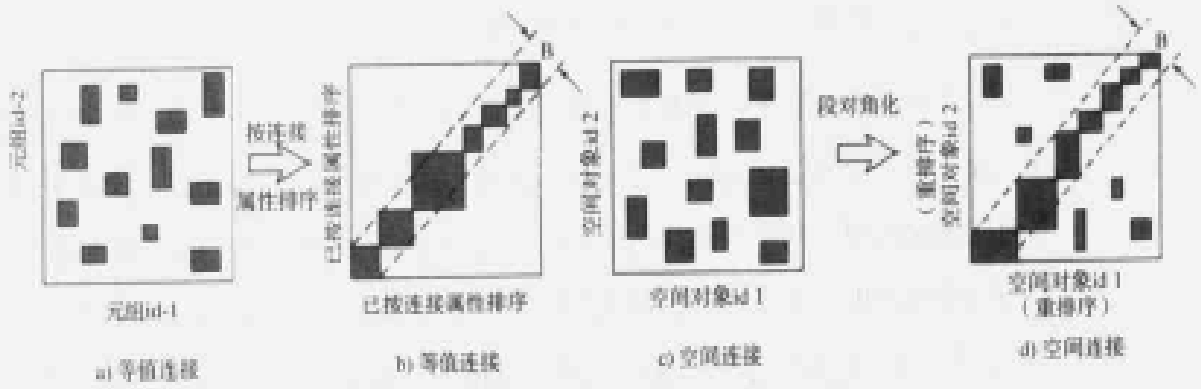


图4-22 等值连接的元组级邻接矩阵和空间连接的比较

2. 页面连通图和页面访问顺序

当两个关系间的连接关系在页面层描述的时候，我们就可以得到一个页面连通图（PCG）。PCG［Merrett et al.，1981］$B_0 = (V_1, V_2, E)$是一个双向图，其中顶点集V_1表示来自第一个关系的页面，顶点集V_2表示来自第二个关系的页面。边的集合的构造方法如下：一条边加入到V_1的页面（结点）v_1^i和V_2的页面（结点）v_2^j之间，当且仅当连接索引中至少有一对对象(r_i, s_j)，其中，$r_i \in v_1^i$，$s_j \in v_2^j$。图4-23是一个从图4-21b中的连接索引得到的页面连通图。结点（a，b）表示关系R的页面，结点（A，B，C）表示关系S的页面。图$B_0 = (V_1, V_2, E)$的一个min-cut结点分区［Hagen and Kahng，1991;Kernighan and Lin，1970］将V中的结点划分成不相交的子集，使得入射结点在两个不同分区的边的数量最少。min-cut分区的cut-set是一组关连接点处于两个不同分区的边的集合。近几年陆续出现了解决这个问题的快速而有效的启发式算法［Karypis et al.，1998;Karypis and Kumar，1998］。它们可以用于在一个PCG中聚类页面。

连接索引有助于提高连接处理的效率，因为它记录了所有满足连接谓词的元组对。给定一个连接索引JI，可以用推导出的PCG安排一个高效的页面访问顺序用于

获取数据页面。其CPU代价是固定的，因为连接每对元组的代价是固定的，并且被连接的元组个数也是固定的。另一方面，I/O代价和页面访问顺序有关。当内存中的缓冲区空间有限时，一些页面不得不多次从磁盘读入。页面访问顺序（其他依次是连接索引聚类和基本关系的聚类）决定I/O代价。

例子 举例说明连接的I/O代价和数据页面访问顺序之间的依赖关系，可以采用图4-23所示的页面连通图的一个实例来辅助安排访问顺序。假设缓冲区空间有限，在高速缓存了整个页面连通图之后，最多允许将关系的两个页面放入内存。考虑两页访问顺序：1) (a，A，b，B，a，C，b)，2) (a，A，b，C，a，B)。每个顺序可以使用有限的两个页面来连接计算的结果。然而，在第一种情况下，共有7次页面访问，在第二种情况下共有6次页面访问。注意，页面访问次数的下限是5，因为在PCG中有5个不同的页面。然而，对于两个缓冲区空间来说，没有能达到5次页面访问的页面访问顺序。因为环(a，A，b，C，a)要求至少将三个页面放入内存中才能避免冗余的页面访问。对于三个页面的缓冲区空间，(a，B，A，C，b)的页面访问顺序可以只需5次页面访问。

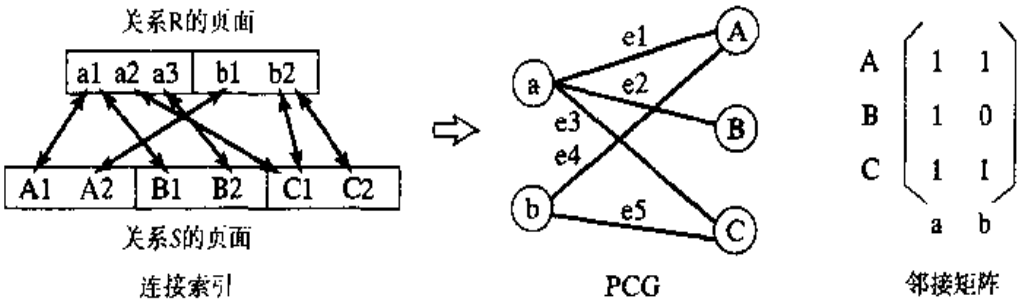


图4-23 从一个连接索引建立一个PCG

4.4 小结

经典的存储和访问方法在管理空间数据集时面临着许多挑战。经典的方法是为有全序的关键码面设计的，自然不能扩展到多维空间中使用的关键码。许多空间对象（例如多边形）往往都是经典方法所不能支持的扩展对象。最后，空间选择谓词（例如，交叠、穿过、最近邻居）都要比经典方法所支持的谓词更丰富。

空间填充曲线（例如Z曲线、Hilbert曲线）为多维空间中的点强加了一个全序。

使用这个全序，空间点对象可以使用经典的索引结构（比如B+树）进行检索。空间查询可以转化成在使用了空间填充曲线的基本B+树上的范围查询。

网格文件是用于空间点的空间索引结构。各个维可以独立地分区。分区点存储在维向量中，它在空间上强加了一个网格结构。网格目录存储指针，这个指针指向存储相关记录的物理页面。

R树是采用矩形搜索码的平衡树。叶结点包括标识符和空间对象的最小外包矩形。中间结点包括子结点中数据的地址和最小外包矩形。R+树和R*树对R树的性能进行了改进。

R链接树为对R树索引结构的并发访问提供了有效的封锁协议。连接索引用来加速空间连接的速度。TR*树用于检索空间扩展对象（例如，多边形）内部的边和结点。

4.5 参考书目

4.1.5 对Hilbert空间填充曲线在聚类上的性质的进一步讨论，见［Leutenegger and Lopez，2000］。对于空间聚类的详细讨论，见［Brinkhoff and Kriegel，1994; Faloutsos and Roseman，1989］。

4.2 在［Samet，1990］中有对空间索引的详细探讨。［Gaede and Gunther，1998］中也为空间索引提供了概述。［den Bercken et al.，1997］讨论了批量载入操作，它为可能的大数据集建立了一个初始的多维索引结构。在商业产品中实现可以支持中等维度数据的索引类型的详细介绍，见［Kanth et al.，1999］。

4.2.2 Z序和R树索引方法已经普及到商业产品中。R树在［Guttman，1984］中有相关介绍。Z序被［Orenstein and Manola，1988］在数据库社团中进行介绍。

4.2.3 R树的代价分析模型的介绍可以在［Faloutsos and Sellis，1987;Kamel and Faloutsos，1993; Pagel et al.，1993b］中找到。缓冲区对于R树性能的影响在［Leutenegger and Lopez，2000; Theodoridis et al.，2000a］中进行介绍。

4.3.2 空间数据库环境下的并发控制仍然是一个尚待研究的领域。我们的讨论参

考［Kornacker and Banks，1995］。［Chakrabarti and Mehrotra，1999］讨论了多维访问方法中的并发控制问题。

4.3.3 连接索引在数据仓库这一迅速扩展的领域中有着广泛的应用。我们的讨论参考［Shekhar et al.，1999b］的著作。

4.6 习题

1. 对于图4-24中的一组叶结点和中间结点，画出相应的R树结构，并列出用虚线表示的查询矩形所要搜索的结点。第一层结点是1和2。第二层结点是A、B、C、D和E。叶结点包括a、b、...、k。
2. 找出图4-9中像素（01，10）和（11，11）的z值和Hilbert曲线。
3. 大空间对象（例如，大多边形或LineString）是许多空间索引的挑战。可以在一个索引的多个结点之间复制指向对象的指针，或者将大对象分解成小碎片再存储到不同索引结点。前者导致在索引中沿多条路径进行搜索并且使后处理的排除计算量倍增，而后一种方法则要在处理后将碎片合并重新构造对象。
   (a) R树和网格文件如何处理大对象?
   (b) 一种最新的方法是将扩展的空间对象按大小进行分组，并且单独检索每组索引。这种方案的优点和缺点分别是什么?
4. 比较和对比以下术语：
   (a) 聚类和索引
   (b) R树和网格文件
   (c) Hilbert曲线和Z序
   (d) 主索引和二级索引
5. 以下性质分别属于哪种结构：R树、网格文件和采用Z序的B树。
   (a) 平衡（例如：所有的数据页面指针的距离（从根/目录到数据页面指针）相同）。
   (b) 固定深度（例如：预先定义一个常数作为距离（从根/目录到数据页面指针））。
   (c) 无交叠：索引中给定层的不同结点之间是互相不交的。

(d) 每结点50%利用率：索引中没有一半是空的结点。

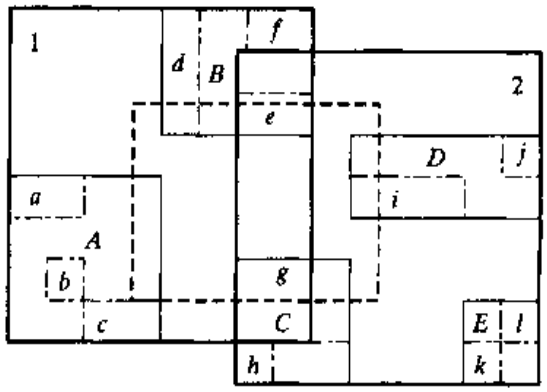


图4-24 练习题1的图示

6. (a)利用按照下列顺序插入的点，画出最终的网格文件结构。假定每个磁盘页面最多储存3条记录。

(0，0)，(0，1)，(1，1)，(1，0)，(0，2)，(1，2)，(2，2)，(2，1)，(2，0)

明确指出维向量、网格目录和数据页面。

(b) 将(a)中的结构改为R树结构。假定一个内部索引结点最多能储存3个指向其他索引结点的指针，一个叶结点能储存指向3个数据记录的指针。

7. 利用按照以下顺序插入的点码，画出最终的网格文件结构：(0，0)，(1，1)，(2，2)，(3，3)，(4，4)，(5，5)，(5，1)，(5，2)，(5，3)，(5，4)，(5，0)。假定每个磁盘页面最多储存2条记录。明确指出维向量、网格目录和数据页面。本题有多个解决方案，希望你给出一个验证的解决方案。

8. 将上题改为Z序的B树结构。假定一个索引结点最多能储存3个指向其他索引结点的指针。一个叶结点最多能储存指向2个数据记录的指针。

9. 将上题改为R树结构。假定一个索引结点最多能储存3个指向其他索引结点的指针，一个叶结点最多能储存指向2个数据记录的指针。

10. 考虑一个矩形集合，其中矩形的左下角位于上题的点集合中。假定每个矩形都是3个单位长，2个单位宽。画出2个可能的R树。假定一个索引结点最多能储存3个指向其他索引结点的指针，一个叶结点最多能储存指向2个数据记录的指针。

11. 对图4-9中的4×4网格，将起点改至左下角，计算其Hilbert函数。

12. 什么是R链接树？它在空间数据库中有何作用？

13. 为图4-15中的数据集画出一个R链接树。

14. 什么是连接索引？列举几个适合利用连接索引的应用。

15. 什么是TR*树？把它与R树进行比较。

16. 对于图4-14中的网格文件，以下查询分别需要访问哪些目录项？

    (a) (30，30)上的点查询。

    (b) x(10，30)和y(10，30)上的范围查询。

    (c) 检索点(30，30)的最近邻居的查询。

17. 对于图4-16中的R树，以下查询分别需要访问哪些结点？

    (a) 对矩形5中的一个点的点查询。

    (b) 包含矩形3和矩形12的最小正交矩形的范围查询。

18. 对于图4-17中的R+树，上题中的查询分别需要访问哪些结点？

19. 对于图4-7，为对象B所包含的像素计算Z序。使用Z序存储时，比较图4-7中对对象B和对象C进行范围查询的计算代价。

20. 对于图4-7，为对象A、B、C所包含的像素计算Hilbert值。使用Hilbert曲线顺序存储时，比较图4-7中对对象B和对象C进行范围查询的计算代价。

第5章
查询处理与优化

对数据库的查询是通过像SQL这样的高级声明性语言来表达的。将查询映射为一系列由空间索引和存储结构支持的操作是数据库软件的责任，其主要的目标是在尽可能少的时间内正确地处理查询。在本章中，我们将主要关注在空间数据库环境中用于处理和优化查询的技术。在关系数据库中，绝大多数查询是由一组固定的基本操作构成的，这些基本的关系操作是构成复杂查询的基本构件。因此查询处理和优化可以分为两个步骤：1）为每个基本的关系运算符设计并调整算法，2）利用第一步的信息把高层查询映射为这些基本关系运算符的组合并进行优化。这一策略同样可以应用到空间数据库环境中，但有一些重要的区别。首先，空间数据库的应用领域多种多样，还没有一套被一致认可的能够满足所有的情况基本构件（例如空间操作）。我们仅仅在空间操作的类别上达成了一致。其次，空间操作的算法既是CPU密集型的又是I/O密集型的，这就使得设计过程比设计传统数据库更为复杂，因为在传统数据库中通常假定I/O代价大大高于CPU代价，并且假定磁盘访问少的算法

就是好的算法。

本章的主要内容如下。5.1节描述了重要的空间运算符的算法设计和调整。5.2节考虑了查询优化的一般问题，并展示了如何优化由空间和非空间谓词组成的查询。5.3节探讨了一些空间索引结构的定量分析技术。5.4和5.5节分别介绍了分布式空间数据库系统和并行空间数据库系统。

5.1 空间操作计算

5.1.1 概述

从查询处理的角度来看，空间数据库与关系数据库之间至少有三个主要区别[Brinkhoff et al., 1993]：

1) 与关系数据库不同，空间数据库没有固定的运算符集合可以充当查询计算的基本构件。

2)空间数据库要处理非常大量的复杂对象。这些对象具有空间范围，而且不能自然地排序成一维数组。

3) 检测空间谓词需要用到计算量极大的算法，所以不能再假定I/O代价在CPU的处理代价中占主导地位。

5.1.2 空间操作

空间操作可以分为四组[Gaede and Gunther, 1998]：

1) 更新操作：标准数据库操作，例如修改、创建等等。

2) 选择操作：可以分为两种：

(a) 点查询（point query, PQ）：给定一个查询点P，找出所有包含它的空间对象O：

$$PQ(p) = \{O | p \in O.G \neq \emptyset\}$$

其中$O.G$为对象O的几何信息。

例如，考虑如下查询："找出所有包含SHRINE的河流冲积平原。"SHRINE是一个点类型的常量。

(b) 范围或区域查询（range or regional query, RQ）：给定一个查询多边形P，找

出所有与之相交的空间对象O。当查询多边形是一个矩形时，称这个查询为窗口查询。这类查询有时也称作范围查询。

$$RQ(P) = \{O | O.G \cap P.G \neq \emptyset\}$$

举个例子，“检索所有与Nile冲积平原相交叠的林分。”

3) 空间连接：和关系数据库中的连接运算符类似，空间连接是非常重要的运算符之一。当两个表R和S基于一个空间谓词θ进行连接时，则该连接称为空间连接。空间连接的一个变形是地图覆盖（map overlay），它也是GIS中的一个重要运算符。这个操作将两个空间对象集合合并以形成一个新的集合。这些新对象的集合的“边界”由覆盖操作所指定的非空间属性来决定。例如，如果覆盖操作给两个相邻对象的一个非空间属性赋相同的值，那么这些对象就“合并”了。

$$R \bowtie_\theta S = \{(o, o') | o \in R, o' \in S, \theta(o.G, o'.G)\}$$

谓词θ的一些例子如下：

- intersect（相交）
- contains（包含）
- is_enclosed_by（被包围）
- distance（距离）
- northwest（西北）
- adjacent（邻接）
- meets（接触）
- overlap（交叠）

空间连接的一个例子是“找出所有交叠的林分和河流冲积平原。”用SQL来表达这个查询的方法如下所示：

```
SELECT   FS.name, FP.name
FROM     Forest_Stand FS, Floodi_Plain FP
WHERE    overlap(FS.G, FP.G)
```

4) 空间聚集：空间聚集的一个例子是“找出距离露营地最近的河流。”空间聚集通常都是最近邻居搜索问题的变体，即给定一个对象o'，找出所有距离o'最近的

对象o。

$$NNQ(o') = \{o | \forall o'' : dist(o'.G, o.G) \leq dist(o'.G, o''.G)\}$$

5.1.3 对象操作的两步查询处理

空间查询处理会涉及复杂的数据类型，例如，一个湖泊的边界可能需要数千个矢量来精确表示。空间操作通常采用图5-1所示的两步算法来高效地处理这些大对象：

1) 过滤步骤：在这一步中，空间对象表示为相对简单的近似，比如MBR。举个例子，考虑如下点查询："找出所有符合下列条件的河流：它们的冲积平原与SHRINE交叠。"用SQL形式表示如下：

```
SELECT   River.Name
FROM     River
WHERE    overlap(River.Flood-plain, :SHRINE)
```

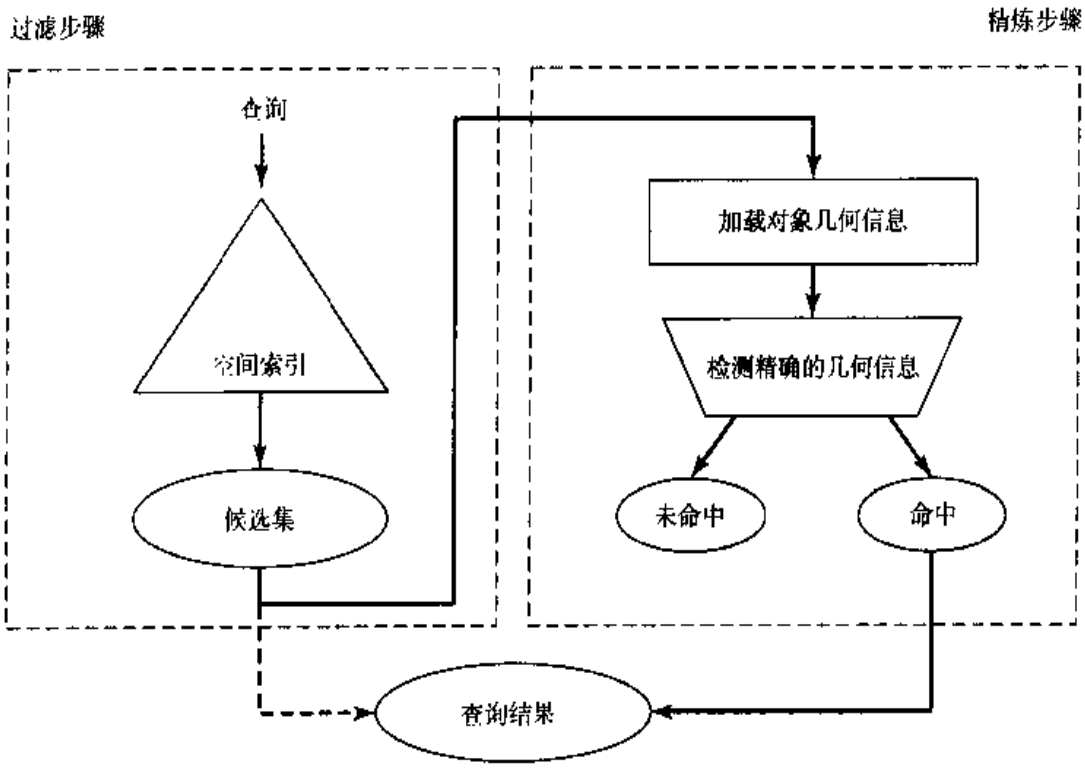


图5-1 多步处理 [Brinkhoff et al., 1994]

我们通过在参数前面加":"来表示用户定义参数，例如：SHRINE。现在如果用MBR来近似表示所有河流的冲积平原，那么判断一个点是否在MBR内比检查这个点

是否在一个表示冲积平原精确形状的不规则多边形内的代价要小得多。这个近似检查的结果是真实结果集的超集。超集有时被称为候选集。空间谓词有时也可以被替换成一个近似以简化查询优化器。例如，在这个步骤中，touch (River, Flood-plain, :SHRINE) 可以替换成overlap(MBR(River, Flood-plain, :SHRINE), MBR(:SHRINE))。有些空间运算符，例如inside（在内部）、north-of（在北部）、buffer（缓冲区分析）可以近似成相应MBR之间的交叠关系。这样的转换能够保证使用精确几何体的最终结果中的元组不会在过滤步骤中被排除掉。

2) 精炼步骤：在这里进行检查的是候选集中每个元素的精确几何信息和精确的空间谓词。这通常需要使用CPU密集型的算法。这一步骤有时可以在空间数据库以外的某个应用程序（比如GIS中）进行，这个应用程序会用到空间数据库在过滤步骤产生的候选集。

5.1.4 空间选择技术

现在开始介绍空间选择中使用的算法。如前所示，有两类主要的空间选择：点查询和范围查询。考虑以下选择查询：

```
SELECT   F.Name
FROM     Forest-Stand F
WHERE    intersects(F.Geometry, :WINDOW)
```

在这个查询中，我们想要标识所有与给定窗口相交的林分的名称。这是范围（区域）查询的一个例子。

下面是点查询的一个例子：

```
SELECT   F.Name
FROM     Forest-Stand F
WHERE    within(:POINT, F.Geometry)
```

这个查询搜索包含:POINT的林分。处理这些查询还有哪些其他的方法呢？答案和与包含待查询的关系的文件的组织方式有关。下面描述另外三个方法。

1. 数据未排序且没有索引

在这种情况下，唯一的选择就是采用穷举法扫描整个文件并判定每一条记录是否满足谓词。根据关系的大小，利用两步处理（过滤加精炼）可能对判定谓词

*intersect*非常有用。如果林分用它的MBR来近似表示，那么范围查询的近似步骤就对应着检验矩形与矩形的交。这个处理与没有过滤就处理整个对象相比，其代价是非常低的。扫描的代价为$O(n)$，其中n是关系Forest-Stand中包含的页数。对于点查询，过滤步骤包括检验一个点在矩形中的操作。如果至多有一片林分包含:POINT，那么平均只需要访问一半的页面。

2. 建立空间索引

我们可以使用空间索引方法来访问几何数据。通常使用R树族来索引关系的空间属性的MBR，这样就可以避免扫描整个文件从而减少系统负载（见4.2.2节）。采用搜索树，点查询通常可以在$O(\log n)$时间内完成处理。使用索引树时查询的代价依赖于许多因素，包括数据矩形的分布、查询窗口的大小、树的高度以及用于构造索引树结点的Packing算法等等。采用R树的缺点是不同分支的MBR可以交叠，这就可能导致沿着索引树的不同分支进行搜索。R树的一个变体R+树避免了内部结点的交叠，R+树的主要问题是数据矩形可能在多个内部结点上存在复本，这会导致搜索时间增加和结点频繁溢出。

3. 空间填充曲线散列

尽管多维空间中没有自然的排序，但是存在一对一的连续映射，可以将多维空间的点映射到一维空间。这样就可以在更高维的空间上间接地进行排序。常见的空间填充曲线的例子就是行序Peano曲线和Hilbert曲线，但它们都不具备如下性质，即所有在多维空间中“位置邻近”的记录在映射后的范围空间中仍保持邻近。一旦数据按空间填充曲线“排序”，B树［Orenstein, 1986］索引就可以用于排序后的项以加快搜索速度（见4.1节）。这一情况在点数据的情况下最为适用，点查询的代价为$O(\log_B(n))$，其中B为分块因子。对于范围查询，代价大约为$O(\log_B(n)) + \frac{\text{查询结果集的大小}}{\text{记录在页面上的聚集程度}}$。尽管Z序不像其他空间访问方式（例如R树）那样有效，但它是为数不多的已经嵌入商用数据库系统的空间数据结构之一。

5.1.5 一般的空间选择

在绝大多数情况下，选择条件可能是许多“原语”选择条件的组合。在传统数据库中，一般条件首先表示为连接范式（conjunctive normal form, CNF），然后使用

散列和索引树的组合来加快查询处理的速度。对于一般的空间选择来说，按什么顺序处理CNF中的各个条件是很重要的，因为不同空间条件的处理代价会有很大差别。这与传统数据库中的情况很不一样，因为传统数据库中假定所有非空间条件的处理代价是大致相同的。Hellerstein [Hellerstein and Stonebraker, 1993]建议采用一个新的度量标准等级（rank），该度量标准考虑空间函数的选择性和代价以决定谓词的顺序：

$$\text{等级} = \frac{\text{选择性}-1}{\text{特征代价}}$$

其中：

$$\text{选择性}(p) = \frac{\text{输出}(p)\text{的集合大小}}{\text{输入}(p)\text{的集合大小}}，p\text{表示一个谓词}$$

特征代价（differential cost）是谓词对单个元组的代价。尽管特征代价的概念看起来并不适合度量谓词的代价，因为它以关系作为输入，但它更适合度量用户自定义方法的代价。此外，特征代价的关键性质在于它在函数的生命期间保持不变，可以与选择性一起保存在系统目录中。主要的结论是谓词应该按等级的升序依次处理。

5.1.6 空间连接操作算法

连接操作是组合两个关系的基本方式。从概念上说，连接的定义为笛卡儿积后跟一个选择条件。这种方式在实践中的代价可能非常昂贵，因为它要求在应用选择条件之前先计算笛卡儿积。对于空间数据库尤其如此，所以在执行笛卡儿积之前先采用一些特别的算法。我们只关注过滤-精炼两步范型中的过滤步骤。这样，空间交叠操作简化为求矩形与矩形的交，与从二级存储检索页面的I/O代价相比而言，这个代价是很小的。考虑下面用SQL表达的连接查询：

```
SELECT   *
FROM     Forest-Stand F, River R
WHERE    overlap(F.Geometry, R.Flood-plain)
```

假定关系Forest-Stand（简写为F）占用M个页面，每页p_F个元组，关系River（简写为R）占用N个页面，每页p_R个元组。此外，假定为一对对象执行overlap函数的平均时间为T。

1. 嵌套循环

在这个算法中，枚举F和R所有可能的元组对，并用overlap函数相互检查。

```
forall tuple f ∈ F
        forall tuple r ∈ R
        if overlap(F.Geom, R.Flood-Plain)
        then add < f,r > to result
```

这里扫描外层关系F，对F的每个元组扫描整个关系R。假定有3个内存缓冲区，忽略回写结果的代价，这样I/O代价为$M + M = N$[⊖]。这个代价说明该算法是不可取的，因此前面提到的嵌套循环算法很少被使用。一种改进的办法是有效利用可用的缓冲区页。如果有B个可用的缓冲区页，那么首先传输外层关系F的$B-2$页，然后用剩下两个缓冲区中的一页存放内层关系的扫描。用最后一个缓冲区页回写<f, r>元组，其中f属于$\{F_1, \ldots, F_{B-2}\}$，r∈R页面。使用缓冲区之后的I/O代价显著地下降为$M+N*\left\lceil\frac{M}{B-2}\right\rceil$。

有索引的嵌套循环

如果其中一个关系建有索引，那么可以在内层循环中利用有索引的关系以利用其优点，这样就不需要在每一次迭代时完全扫描整个内层关系。使用内层关系上的索引来检索能与存储在主存中的外层关系的元组相匹配的候选元组，可以取代范围查询。

2. 树匹配 [Brinkhoff et al., 1993]

如果两个关系都有空间索引（例如R树），那么就可以使用树匹配策略。回顾一下R树索引并用它来阐明这个策略。树结点包含形如 (*ref*, *rect*) 的项，其中*ref*为指向子结点的指针。*rect*项为子结点的项的MBR或空间对象的MBR，这取决于该结点是否为叶结点。二级存储中对应非叶结点的页称为目录页，对应实际数据的页称为数据页。

现在考虑两个矩形的集合R_1与R_2，这两个集合分别表示两个其空间对象用R树检索的空间关系。我们要使用R树结构来设计空间连接操作的算法。空间谓词为两

⊖ 此处原书误写为M+M*N，译者加以修正。——译者注

个矩形的交：若（r_1, r_2）满足下列条件，则属于连接关系：

$$rect(r_1) \cap rect(r_2) \neq \emptyset$$

如果I_{R_1}和I_{R_2}分别为关系R和S的索引树的结点占用的页面数，那么I/O代价的下限为$I_{R_1} + I_{R_2}$。

该算法基于这样一条规律：非叶结点的矩形能够容纳所有子结点中的矩形的MBR。因此，如果两个目录项$Er1$和$Er2$不相交，那么其后子树中的所有数据矩形必然也不相交。如果目录项相交，那么子树的某个数据矩形则有可能会相交。

基本算法如下：

```
SJ(R1, R2:R_NODE);
01     BEGIN
02           FOR (all Er2 in R2) DO
03               FOR (all Er1 in R1) DO
04                   IF (overlap(Er1.rect, Er2.rect)) THEN
05                      IF (R1 and R2 are leaf pages) THEN
06                         output(Er1.oid, Er2.oid)
07                      ELSE IF (R1 is a leaf page) THEN
08                         ReadPage(Er2.ptr);
09                         SJ(Er1.ptr, Er2.ptr)
10                      ELSE IF (R2 is a leaf page) THEN
11                         ReadPage(Er1.ptr);
12                         SJ(Er1.ptr, Er2.ptr)
13                      ELSE
14                         ReadPage(Er1.ptr), ReadPage(Er2.ptr);
15                         SJ(Er1.ptr,Er2.ptr)
16                      END-IF
17                   END-IF
18               END-FOR
19           END-FOR;
20     END.
```

如果我们把一个关系作为数据矩形的源，另一个关系则作为查询窗口的源，那么连接算法的代价就由空间范围查询的代价决定。该思想已经被[Aref and Samet, 1994]用来导出空间连接查询的分析公式。在下一节中将概述范围查询和连接查询的代价分析。

3. 基于分块的空间归并连接

[Patel and Dewitt, 1996]提出了一个新的方法，称为基于分块的空间归并连接。

下面描述该算法的过滤步骤：

• 过滤步骤：给定两个关系F和R，这一步骤可以描述如下：

1) 对F和R中的每个元组构造*key-pointer*元组，*key-pointer*由元组的唯一OID和空间属性的MBR组成。新关系称为F^{kp}和R^{kp}。

2) 如果关系F^{kp}和R^{kp}都可以装入主存，那么连接关系可以用平面扫描算法处理。

3) 如果关系太大而不能全部装入主存，则将两个关系都分成P块，即F_1^{kp}，…，F_p^{kp}和R_1^{kp}，…，R_p^{kp}。

• 分块：分块必须满足以下两个约束：

1) 对每一个F_i^{kp}，R^{kp}中与之交叠的元素位于R_i^{kp}。

2) F_i^{kp}和R_i^{kp}都位于主存。

还有一些其他的空间连接策略，例如外部平面扫描或基于连接索引的方法。实际上，设计与比较空间连接策略仍是一个活跃的研究领域。我们介绍了一些有代表性的策略，有兴趣的读者可以进一步参阅相关的技术文献。

5.1.7 空间聚集操作策略：最近邻居

最近邻居查询在许多应用中都很常见。例如，电子商务网站接收到书籍订单后应该把订单发送至最近的配送中心。4.1节末尾讨论了解决这个问题的两遍算法。第一遍检索包含查询对象QO的数据页D，以确定D中任意对象到QO的最小距离d。第二遍通过一个范围查询检索与QO的距离在d之内的对象以确定最近邻居。这个方法重用了用于空间选择（例如点查询和范围查询）的算法。一遍处理最近邻居查询需要用到与空间选择完全不同的算法。我们介绍一个有代表性的分而治之算法，它最早由[Roussopoulos et al., 1995]提出，这个方法使用了一对距离度量，即搜索修剪条件和搜索算法。

距离度量包括min-distance(Point P, Rectangle R) 和min-max-distance(P, R)。如果P在R之内或在R的边界上，则min-distance(P, R)为0。如果P在R之外，则min-distance (P, R)为P到R的任意边界的欧几里德距离。欧几里德距离表示R中任意对象到P的距离的下限。min-max-distance (P, R)为P到R中任一包含顶点V的表面上最远点之间的距离，其中V为R距离点P最近的顶点。R树的构造保证R树中存在矩形R内

的对象O、满足distance (O, P) ≤min-max-distance(P, R)。min-distance(P, R)提供了在最近邻居查找中子树的“乐观”排序，而min-max-distance(P, R)提供了所谓的“悲观”排序。

搜索修剪策略也可以基于以下度量。例如，对于MBR M，如果存在另一MBR M' 满足min-distance (P, M) >min-max-distance (P, M')，则可以排除MBR M。同样，如果存在对象O满足distance (P, O)<min-distance (P, M)，则也可以排除MBR M。最后，如果存在一个MBR M满足distance (P, O)>min-max-distance (P, M)，则可以排除对象O。

最近邻居的搜索算法从R树的根结点开始遍历整个树。例如，R树的广度优先遍历将访问当前结点中所有内部结点的子结点的MBR，并利用上面提到的规则进行修剪。剩下的子结点将在下一轮迭代中被展开。最后一次迭代将从按树的层次顺序进行修剪所幸存的MBR中获得一组叶结点（数据库对象级）。该算法要计算每个叶结点到查询点P的距离以决定最近邻居。也可以选择深度优先遍历或其他的R树遍历方式。这个算法可以扩展为查找K个最近邻居，只要稍稍修改修剪规则保留K个最好的候选者即可。最后，修剪标准同样适用于基于矩形的索引方法，比如网格文件、四叉树等等。

5.2 查询优化

第3章中提到过，查询通常使用像SQL这样的高级声明性语言来表达。这意味着用户只需指明结果集，而获取结果的策略则交由数据库负责。用于度量策略或者计算计划的标准，就是执行查询需要的时间。在传统数据库中，该度量很大程度上取决于I/O代价，因为可用的数据类型和对这些类型进行操作的函数相对来说都是易于计算的。而空间数据库的情况就不同了，因为它包含了复杂数据类型和CPU密集型的函数。这样，在空间数据库中选择一个优化策略的任务比在传统数据库中更为复杂。

查询优化器（query optimizer）是数据库软件中的一个模块，它用于产生不同的计算计划并确定适当的执行策略。查询优化器从系统目录中获得信息，并结合一些启发式规则和动态规划技术以制定合适的策略（见图5-2）。即使可能，查询优化器

也很少执行最好的计划，这是因为优化计算十分复杂[Selinger et al., 1979]。一般的思想是避免最差的计划而选择一个较好的计划。查询优化器所承担的任务可以分成两部分：逻辑转换和动态规划[Adam and Gangopadhyay, 1997]。

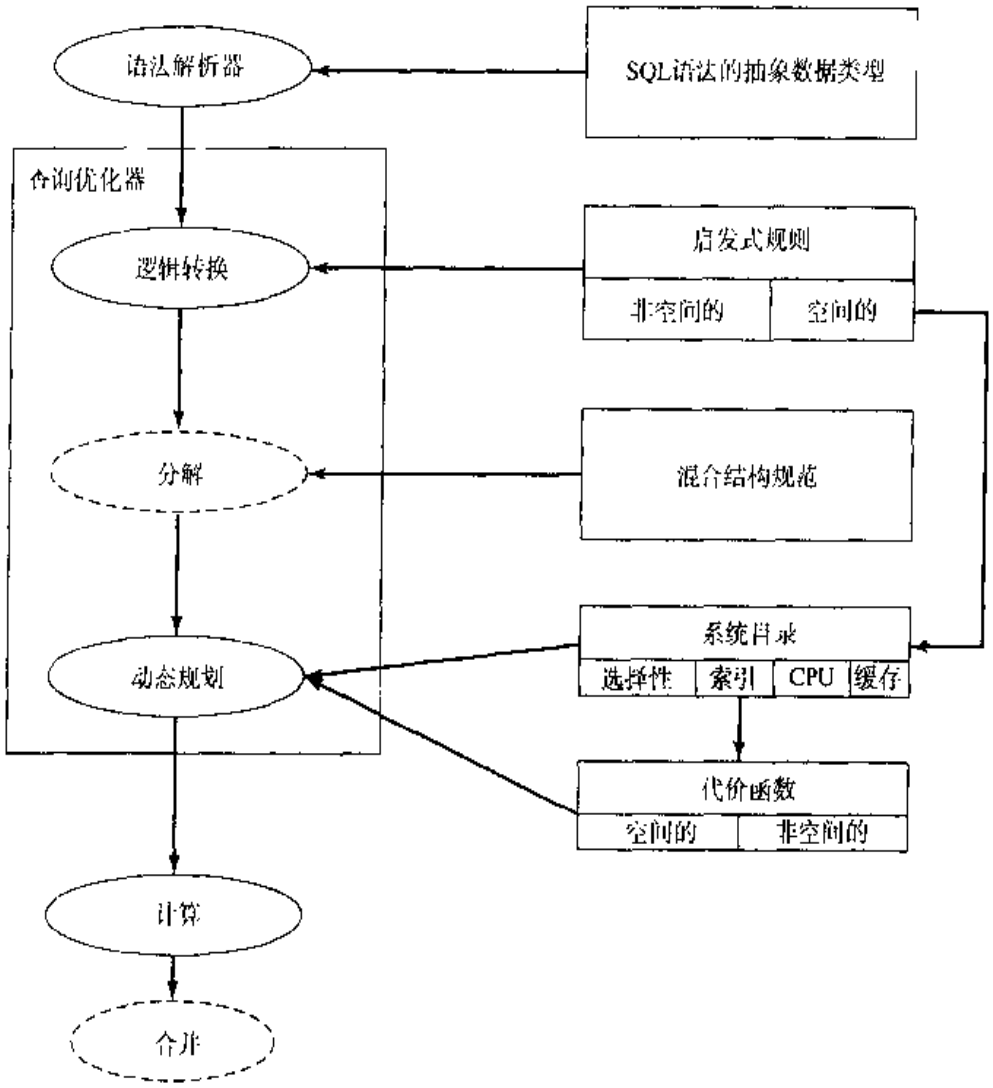


图5-2 查询优化器的模式

5.2.1 逻辑转换

1. 语法分析

在查询优化器对查询进行处理之前，必须由语法分析器来扫描高级声明性语句。语法分析器检查语法并将语句转换成一个查询树。在传统数据库中，数据类型和函数都是固定的，语法分析器相对比较简单。空间数据库是可扩展的数据库系统，可以支持用户自定义的类型和方法。因此与传统数据库相比，空间数据库的语法分析

器相应地也更加复杂，因为它要识别并管理用户定义的数据类型并将其转换成语法正确的查询树。在查询树中，叶结点对应着所涉及的关系，而内部结点对应着组成查询的基本操作。回忆一下，基本操作包括选择（SELECT）、投影（PROJECT）、连接（JOIN）和其他集合操作。查询处理从叶结点开始自底向上处理，直到根结点上的操作执行完成。考虑第3章中出现的查询：

"找出所有面积大于20平方公里并且距离营地小于50公里的湖泊。"

```
SELECT   L.Name
FROM     Lake L, Facilities Fa
WHERE    Area(L.Geometry) > 20 AND
         Fa.Name = 'campground' AND
         Distance(Fa.Geometry, L.Geometry) < 50
```

这里，我们假定面积没有预先计算并保存在关系当中。Area是一个空间函数，每次调用该函数来计算面积。由上面查询生成的查询树如图5-3所示。

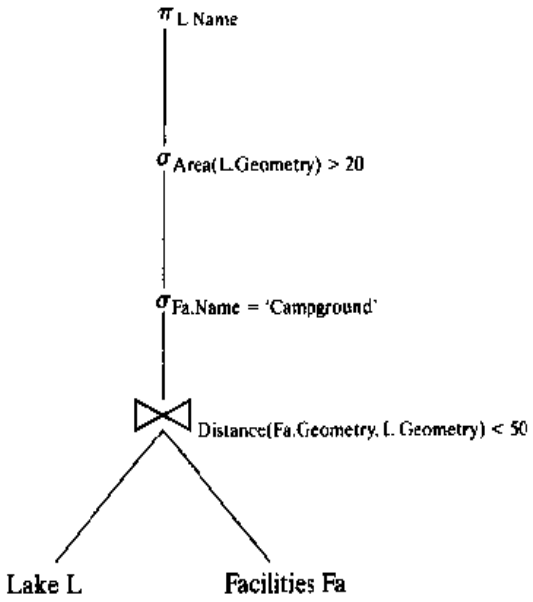


图5-3 查询树

2. 逻辑转换

如果仅仅用图5-3中的查询树作为制定执行计划的基础，那这个策略可能是很拙劣的。连接操作的代价很大，其代价和所涉及关系大小的乘积有关，因此我们希望减少连接操作所涉及关系的大小。一种方法是将非空间的选择操作下推（见图5-4）：

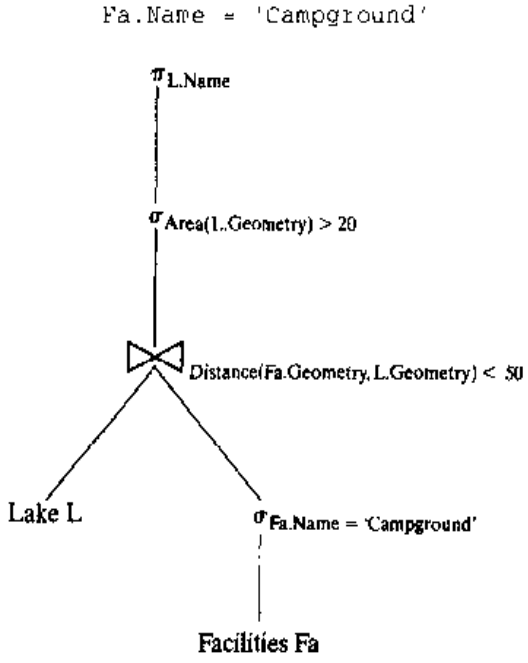


图5-4 下推：选择操作

在逻辑转换步骤中，语法分析器生成的查询树被转换成等价的查询树。这种等价是从关系代数继承的一组形式化规则的结果。在枚举出等价树之后，可以应用一些启发式规则将那些显然不是最终执行策略的候选者过滤掉。将非空间选择操作朝叶结点的方向下推只是启发式规则的一个例子。逻辑转换类似于查询处理中的过滤步骤，这是删除那些不是最佳的查询树的一条捷径。虽然有一些确定的启发式规则适用于传统数据库，但这些规则在空间数据库中可能会产生二义性。例如，考虑如果将下列所示的空间选择操作下推会出现什么情况（见图5-5）：

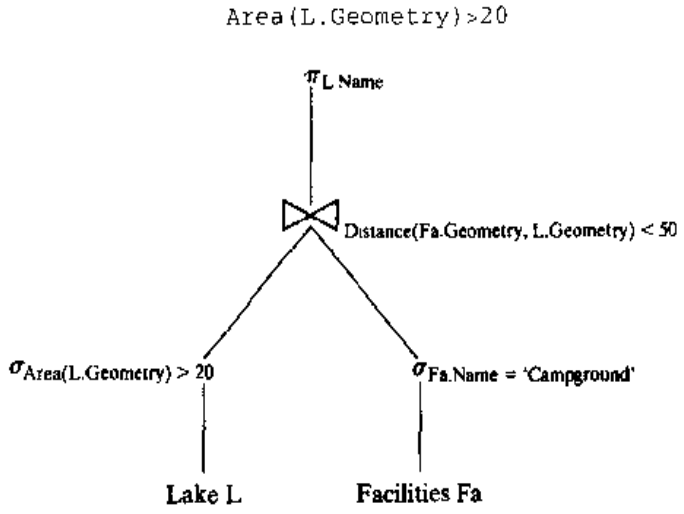


图5-5 选择下推并不总是有效

看起来一个空间连接应该在一个空间谓词之后执行。但现在有两个用户定义的函数，即Area（求面积）和Distance（求距离），它们的执行代价必须在比较后分级。如果Area函数的计算密集度比Distance函数高出几个数量级，那么建议不要在连接之前执行空间选择。这样，关系数据库中主要的启发式规则“在连接和二元操作之前执行选择和投影”就不是在任何条件下都适用了。我们能做的就是非空间选择和投影运算符应该朝查询树的叶子结点方向下推。有没有其他启发式规则能适用于代价高昂的空间函数呢？[Hellerstein and Stonebraker, 1993]提出谓词 p的等级概念来给不同的谓词分级：

$$\text{等级} = \frac{\text{选择性}-1}{\text{特征代价}}$$

其中：

$$\text{选择性}(p) = \frac{\text{输出}(p)\text{的集合大小}}{\text{输入}(p)\text{的集合大小}}，p\text{表示一个谓词}$$

特征代价是谓词对单个元组的代价。尽管有观点认为特征代价因为以关系作为输入而不适合于度量谓词的代价，但它们更适合度量用户定义方法的代价。此外，差异代价的关键性质是它在函数的整个生命周期内保持不变，可以与选择性一起保存在系统目录中。主要结论是谓词应该按等级的升序处理。

另一种计划空间由关系代数中一些良定的规则来生成。这些规则定义源于代数中的概念：交换律（commutativity）、结合律（associativity）和分配律（distributivity）。下面分别为选择、投影和连接操作描述规则[Ramakrishnan, 1998]。这些运算符组合的等价规则随后列出。

3. 选择

Selection运算符有两个重要的等价关系：

1) $\sigma_{c_1 \wedge c_2 \wedge \cdots \wedge c_n}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(\ldots(\sigma_{c_n}(R))\ldots))$

这个规则使我们可以将多个选择组合为一个选择。对于空间查询处理来说，其中选择条件可以是空间和非空间谓词的组合，因此将所有非空间条件右移可能是很值得的（比如，在空间选择之前执行所有的非空间选择）。

2) $\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$

这条规则说明选择条件可以按任意顺序进行判定。因为代价的差别很大，所以最好在判定空间条件之前先判定非空间条件。

4. 投影

• 如果a_i为一组属性且对于所有$i = 1...n-1$有$a_i \subset a_{i+1}$，那么

$$\pi_{a_1}(R) \equiv \pi_{a_1}(\pi_{a_2}(\ldots(\pi_{a_n}(R))\ldots))$$

5. 笛卡儿积和连接

• 连接操作是可交换的。

$$R \bowtie S \equiv S \bowtie R$$

• 连接操作是可结合的。

$$R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$$

• 由上两条规则还可以推导出：

$$(R \bowtie T) \bowtie S \equiv (T \bowtie R) \bowtie S$$

6. 选择、投影和连接

对于两个或多个运算符，有如下一些等价规则：

• 如果投影运算符保留的属性和选择条件有关，则

$$\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$$

• 如果选择条件c涉及一个属性且该属性只在R中出现而不在S中出现，则

$$\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie R$$

• 投影可以与连接一起计算：

$$\pi_a(R \bowtie S) \equiv \pi_{a_1}(R) \bowtie \pi_{a_2}(S)$$

其中a_1为出现在R中的a的子集，而a_2为出现在S中的a的子集。

5.2.2 基于代价的优化：动态规划

动态规划是从一组执行计划中确定最优执行策略的技术。最优解决方案是根据代价函数推导而来的。每个计划使用其代价函数来评估，代价最小的计划就是最优计划。在动态规划步骤中，我们关注查询树的每个结点，枚举处理该结点的所有可能的执行策略。在组建整个查询树的过程中，每个结点的不同处理策略组成了计划

空间。计划空间的基数通常很大，性能也可能相差几个数量级。而且，基本的假定认为代价函数所需的各种参数都是简化的，有时甚至是不正确的。此外，“优化时间”（optimization time），即选择一个执行计划所需的时间必须保持最小。所有这些都意味着（正如前面所说的）我们的目标只是选择一个较好的计划而不是最好的计划。

动态规划的核心在于利用代价函数评估每个执行策略。一个较好的代价函数必须考虑如下因素：

1) **访问代价**：从二级存储搜索和传输数据的代价。

2) **存储代价**：存储查询的执行策略所产生的临时关系的代价。

3) **计算代价**：执行主存内运算的CPU代价。

4) **通信代价**：在客户端与服务器之间传递信息的代价。

一般来说，假定I/O代价在所有查询处理的代价中占主导地位。但在空间条件下就不是这样了，必须考虑访问代价和计算代价。此外，随着万维网和网络编程语言（例如Java）的出现，空间数据库可能分布在多个站点。在这种情况下，通信代价也必须作为代价函数的考虑因素。

1. 系统目录

系统目录维护着代价函数所需的信息，以便设计最优执行策略。这些信息包括每个文件的大小，一个文件的记录数，记录占用的块数，还可能包括关于索引和索引属性的信息、谓词的选择性和特征代价也包括在内。在某些情况下，将代价高昂的用户定义函数物化并索引其函数值以便进行快速检索是非常有用的。例如，在上面例子中，`Area(Geometry)`函数可以预先计算并索引其函数值。这就明显地加快了查询计算的速度，系统目录可以为这些“急需的”函数存储信息。

2. 代价函数

关系数据库使用的代价函数[Stonebraker and Moore, 1997]是下面这个式子的变形：

$$\text{cost} = \text{Exp (records-examined)} + K^{*}\text{Exp (pages-read)}$$

其中，Exp (records-examined) 为预计读取的记录数，可以作为CPU时间的度量；Exp(pages-read)为预计从外存读取的页数，可以作为I/O时间的度量；因子K度

量CPU资源相对于I/O资源的重要程度。

5.3 空间索引结构分析

本节对SDBMS中一个不可或缺的部分——空间索引结构进行量化分析，主要描述执行一个空间查询需要的磁盘访问量。尽管要重点讨论范围查询，但也对空间连接（如交叠）操作进行了介绍。

空间索引结构将空间对象映射到二级存储的页面，这些页面称为存储桶（bucket）。映射是基于邻近性的，也就是说，一般认为将位置上靠近的对象映射到同一页上。当然这并不总是可行，因为存储桶的大小是固定的（例如，8KB）。对象可以是点类型，也可以具有一定的空间范围，通常用MBR表示一个非点的对象。对应于一个存储桶的物理空间称为桶区域（bucket region）。点数据的桶区域通常会划分数据空间，而非点数据的桶区域会交叠而不覆盖整个空间。

首先假定空间维度d = 2并且地址空间规范化为一个单位正方形。令$B = \{B_1, ..., B_m\}$为桶的集合，$R(B) = \{R(B_1), ..., R(B_m)\}$为对应相离的桶区域的集合。此外，对于一个给定的查询对象Q，满足以下条件：

- $P_i(Q)$为Q与$R(B_i)$相交的可能性。
- $P(Q, j)$为Q与某个确定的桶区域j相交的可能性。

那么预计与Q相交的桶数$E(Q)$为

$$E(Q) = \sum_{j=0}^{m} j \cdot P(Q, j)$$

不难看出[Pagel et al., 1993a]：

$$E(Q) = \sum_{j=0}^{m} j \cdot P(Q, j) = \sum_{i=1}^{m} P_i(Q)$$

1. 点查询

如果点查询Q等可能地出现在单位面积的地址空间的任一点，那么$P_i(Q)$等于$R(B_i)$的面积。如果$R(B_i)$是一个矩形$(n_{i,1}, n_{i,2})$，那么：

$$E(Q) = \sum_{i=1}^{m} n_{i,1} * n_{i,2}$$

2. 范围查询

先考虑一维的情况，即$d = 1$。也就是说，假定查询对象Q为区间(q_l, q_r)，桶区域$R(B_i)$为区间$(d_{i,l}, d_{i,r})$。那么有：

引理1 假定$(q_r - q_l) + (d_{i,r} - d_{i,l}) < 1$，那么

$$P_i(Q) = (q_r - q_l) + (d_{i,r} - d_{i,l})$$

证明：

$$\begin{aligned} P_i(Q) &= \text{Prob}\{q_l \in (d_{i,l}, d_{i,r})\} + \text{Prob}\{(q_l \notin (d_{i,l}, d_{i,r})) \wedge (q_r \in (d_{i,l}, d_{i,r}))\} \\ &= (d_{i,r} - d_{i,l}) + \text{Prob}\{q_l \in (d_{i,l} - (q_r - q_l), d_{i,l}))\} \\ &= (d_{i,r} - d_{i,l}) + d_{i,l} - (d_{i,l} - (q_r - q_l)) \\ &= (q_r - q_l) + (d_{i,r} - d_{i,l}) \end{aligned}$$

这样，一维范围查询的磁盘访问量的预计数目为：

$$E(Q) = \sum_{i=1}^{m} (d_{i,r} - d_{i,l}) + m * (q_r - q_l)$$

由此很容易推广到一般的情况。对于桶区域$n_{i,1} \times n_{i,2}$上的一个范围Q $(q_1 \times q_2)$，磁盘访问数为

$$E(Q) = \sum_{i=1}^{m} \prod_{j=1}^{2} (q_j + n_{i,j})$$

上面的公式对所有空间数据结构都有效，并突出了最小化MBR周长和面积的需求，可以作为代价估算和查询优化的有用工具。唯一没有考虑的因素是关于MBR大小的信息，即上面公式中的$n_{i,j}$。下面描述对于索引结构的R树族，如何根据给定的很少的数据集参数来获取这个信息[Theodoridis and Sellis, 1996]。

定义 若有一组桶区域m，平均大小为$n = (n_1, n_2)$，则其密度D为包含单位正方形上一个给定的点的桶区域的平均数目。$D = D(m, n) = m \cdot n_1 * n_2$

[Theodoridis and Sellis, 1996]已经展示了对于一个范围查询$Q = (q_1 \times q_2)$，以下公式成立：

1)

$$E(Q)=\sum_{j=1}^{h-1}\left\{m_j\cdot\prod_{k=1}^{2}(n_{j,k}+q_k)\right\}$$

其中h为树的高度（根位于$j=h$层而叶子位于$j=1$层），m_j为j层的预期结点数，$n_{j,k}$为在k维的平均结点范围。

2) R树的高度为：

$$h=1+\left\lceil\log_{ac\cdot mc}\frac{m}{ac\cdot mc}\right\rceil$$

其中，ac为平均结点容量（通常为70%），mc为R结点中的最大项数。

3) j层的结点数为：

$$m_j=\frac{m}{(ac\cdot mc)^j}$$

4) j层的平均结点范围为：

$$n_{j,k}=\left(\frac{D_j}{m_j}\right)^{0.5}$$

5) D_j为j层的密度：

$$D_j=\left\{1+\frac{D_{j-1}^{0.5}-1}{(ac\cdot mc)^{0.5}}\right\}^2,\ \text{其中}\ D_0=D$$

3. 空间连接的代价分析

下面为上一节介绍的树匹配连接算法提供了相应的代价函数 [Theodoridis et al., 1998]。

假设有两个空间数据的集合，基数分别为m_{R_1}和m_{R_2}，项存储在R树索引R_1和R_2中，我们的目标是计算执行一个连接查询预计需要的结点访问数（NA）。在连接算法中，缓冲区管理起着至关重要的作用，所以结点访问和磁盘访问之间有很大区别，不等式$DA\leqslant NA$总是成立的。不考虑细节，树匹配连接查询的公式为：

$$NA\_total(R_1,R_2)=\sum_{i=1}^{h-1}\{NA(R_1,j)+NA(R_2,j)\}$$

其中，$NA(R_i, j)$, $i = 1, 2$, 为R_i在 j 级的结点访问量。这个可以由上面R树的公式计算出来。

5.3.1 枚举可选的计划

在用公式表示了查询之后，优化器枚举一组计划并执行其中计算代价最小的计划。在枚举不同计划时会用到关系代数的等价关系以及每个空间与非空间算子可用的实现技术。一个查询计划由一个查询树和每个结点上的附加符号组成，这些附加的符号用于指定结点运算符的访问方式。例如，考虑如下查询：

"找出与一个给定窗口（WINDOW）相交且与河流冲积平原交叠的林分的名称"：

```
SELECT   F.Name
FROM     Forest-Stand F, River R
WHERE    Intersect(F.Geometry, :WINDOW) AND
         Overlap(F.Geometry, R.Flood-Plain)
```

对以上查询进行评估的一个查询计算计划如图5-6所示。它说明空间连接操作是采用树匹配方式后跟一个利用R树索引方法的范围查询操作来实现的。投影操作的应用一带而过，也就是说，没有物化前面操作的中间结果。还可以选择其他的查询计算计划。例如，在连接操作中可以用块嵌套循环来代替树匹配算法，也可以选择一个不同的但是代数等价的查询树作为查询计算计划的基础。事实上，当在连接操作之前计算选择操作时，为选择谓词选用R树索引会更有意义。

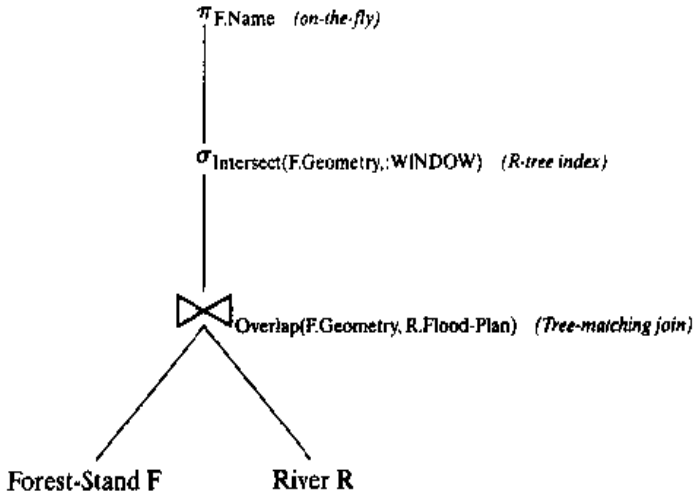


图5-6 执行策略：查询计算计划

5.3.2 混合体系结构中的分解与归并

在基于混合体系结构的空间数据库管理系统中，一个查询可以分解为空间部分与非空间部分。各个子查询首先分别在专门用于空间或非空间优化的单独的模块中进行优化，最后再归并起来以提供所期望的结果。分解基于三个标准：1)给每个空间谓词创建一个子查询；2)给每个与空间谓词通过连接词相连的非空间谓词创建一个子查询；3)给所有非空间谓词创建一个子查询。

考虑图5-7所示的查询树，有很多方法可以将这个查询树分解成空间与非空间部分。图5-8展示出分解的两种方式。每种分解方式的性能不同，软件中用于决定最优分解的那一层必须整合到查询优化器中。在分解之后就必须考虑子查询的调度问题，例如，子查询是串行处理还是并行处理，这和它们之间的耦合方式有关。

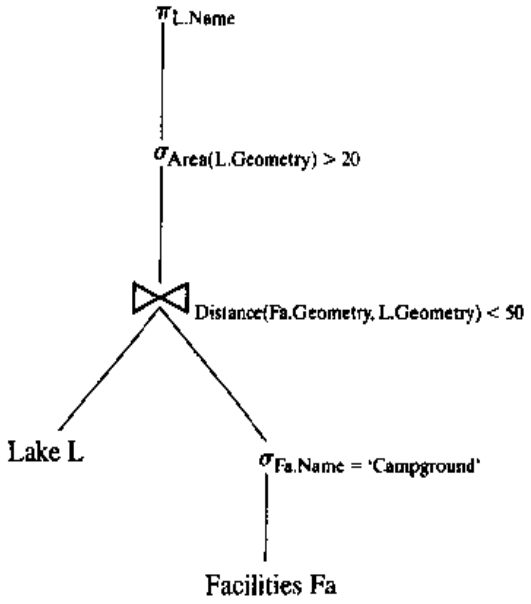


图5-7 执行策略：查询树

5.4 分布式空间数据库系统

下面简要介绍一类相对较新的数据库管理系统，称为分布式数据库管理系统（distributed database management systems, DDMS）。计算机网络的高速发展和不断增加的网络互连是DDMS快速增长的一个原因。DDMS是一组物理上分布的数据库集合，这组数据库集合由数据库管理软件进行管理。DDMS体系结构非常适用于SDB，因为空间数据是由不同组织采集的，而将数据库集中复制到一个站点也是非常困难的。

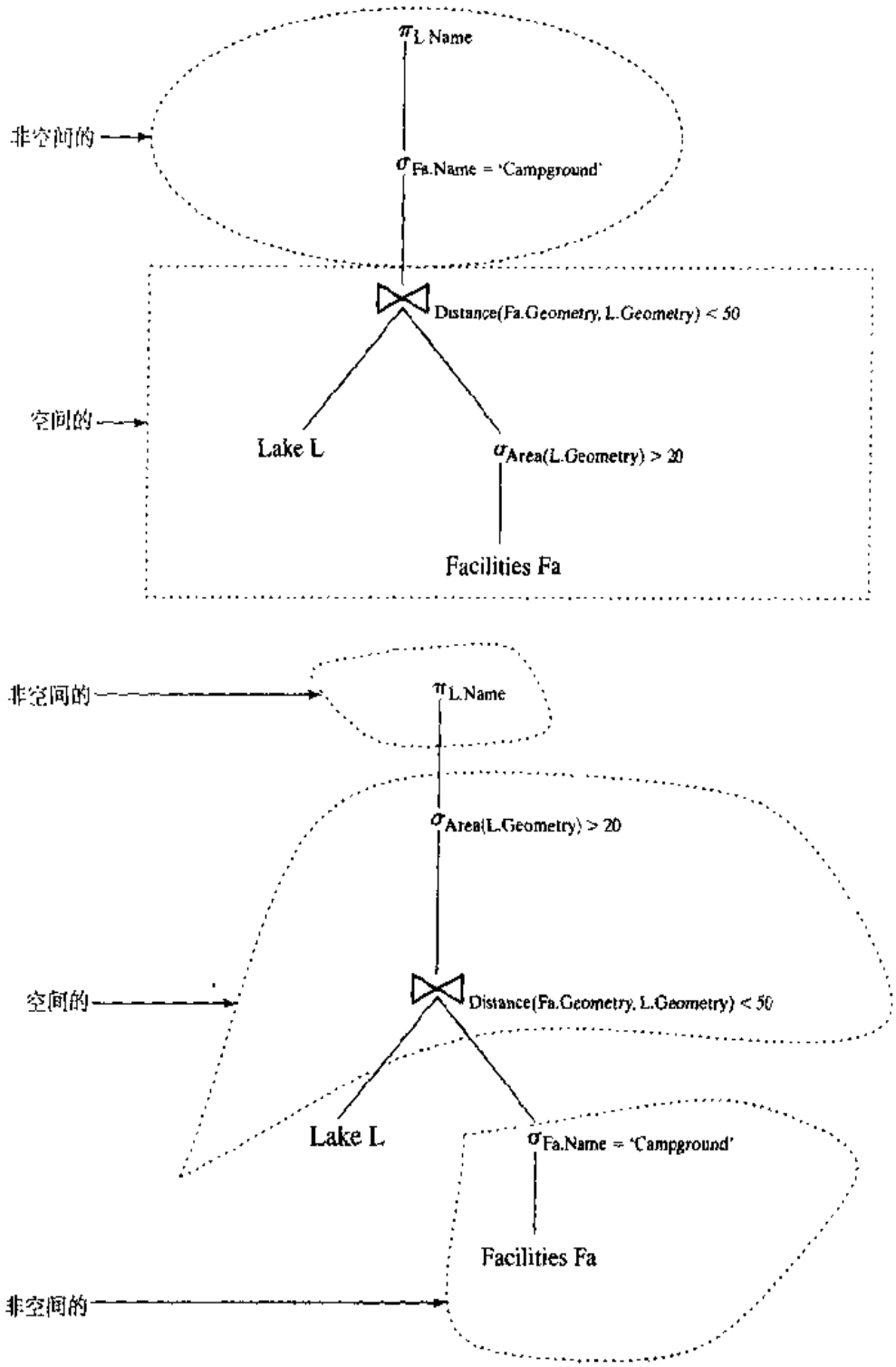


图5-8 两种分解方式

例如，假定一个郡的所有农田都遭受到各种各样的作物疾病，保险公司要评估该郡农民的受灾情况。进一步假定赔付数额和作物疾病类型有关。保险公司可以访问国土资源部维护的农田数据库以及由农业部建立和维护的作物疾病传播数

字地图。于是可以将这两个部门的数据库作为分布式数据库的一部分，而不必从两个不同的政府机构获得数据拷贝，然后就可以用以下方式进行查询。

```
SELECT   F.id,   D.Name
FROM     Farms F, Disease_Map D
WHERE    Intersects(F.Boundary, D.Boundary)
```

那如何处理这个查询呢？这个查询要求将Farms关系发送到Disease_Map关系所在的站点，或者将Disease_Map关系传送到Farm关系所在的站点，再或者将Farm关系和Disease_Map关系都传送到保险公司的数据库。尽管这些选项不会影响到CPU代价和I/O代价，但数据的传输代价会有很大的变化，这和采用了上述三种策略中的哪一种有关。在分布式数据库中，有一个特殊的连接操作称为半连接（semijoin），这种操作在一些情况下可以极大减少数据传输的代价。在给出半连接操作的例子之前，我们先简要描述两种分布式体系结构。

5.4.1 分布式DBMS体系结构

根据如何在不同的DBMS相关进程之间划分功能，可以采用两种体系结构：客户-服务器系统或协同服务器系统。

在客户-服务器系统中，有一个或多个客户进程和一个或多个服务器进程，客户进程可以发送查询至任意服务器进程。客户端负责用户界面，服务器管理数据并执行事务。这样，客户进程可以运行在PC上并将查询发送至在主机上运行着的服务器上。

协同服务器系统包括一组数据库服务器，每个服务器都能运行针对本地数据的事务，通过合作执行跨多个服务器的事务。

客户-服务器系统比协同服务器系统更常用，原因在于：客户-服务器系统容易实现，能充分利用昂贵的服务器，并有一定的历史原因。但是当简单查询必须跨越多个服务器时，就要求客户端的复杂度和功能达到一定标准，这将导致其功能与服务器有所重复。消除客户端与服务器的差别就变成了协同服务器系统。

客户-服务器系统提供了优化每个模块并减少数据传输量的可能性，所以在独立实现GIS应用和海量数据处理的领域，它已成为最令人感兴趣的系统。因特网

（即全球计算机之间的连接）提供驻留应用的基础设施，因此目前大部分应用都是基于客户-服务器的。

5.4.2 半连接操作

在半连接操作中，可利用如下手段减少数据传输代价：1) 只将连接属性和主码从站点1发送到站点2，2) 只将有关元组从站点2发送到站点1。例如，考虑图5-9所示的关系FARM和DISEASE_MAP，现在考虑如下策略。

FARM

FID（10字节）	OWNER_NAME（10字节）	FARM_BOUNDARY（2000字节）	FARM_MBR（16字节）

DISEASE_MAP

MAP-ID（10字节）	DISEASE_NAME（20字节）	DISEASE_BOUNDARY（2000字节）	D_MBR（16字节）

图5-9 两个分布关系：FARM关系有1000个元组，DISEASE_MAP关系有100个元组

1) 投影关系FARM的FID和FARM_MBR并传送到DISEASE_MAP所在的站点。传输的字节数为(10+16) × 1000=26 000字节。

2) 将收到的关系与关系DISEASE_MAP基于属性FARM_MBR和D_MBR进行连接。假定空间连接操作选择DISEASE_MAP的10个元组。将这10个元组的所有属性传送到关系FARM所在的站点。传输的字节数为 (10+20+2000+16) × 10=20 460字节。

3) 在关系FARM所在的站点，将关系FARM和DISEASE_MAP的元组连接。假定所有农田都遭受某一种作物疾病的侵害。现在将FID、OWNER_NAME和DISEASE_NAME传送到保险公司的站点。在这种情况下总共传输 (10+10+20) × 1000 = 40 000字节。在习题5.18中，读者可以验证采用半连接策略传输的字节数比每个关系的大小都要少。

5.4.3 基于Web的空间数据库系统

直到最近，专题地图（森林、城市、湿地、冰雪覆盖、等等）、遥感图像等空间数据产品仅限于少数的研究实验室和政府部门使用，普通大众对空间数据的访问则

局限于纸质地图的形式。不断增长的需求和因特网的易用性推动了基于Web的地理信息系统（Web-based Geographic Information Systems, WGIS）的发展，这样在因特网上共享空间数据产品就变得容易了。最初，开发工作主要集中于使用公共网关接口（Common Gateway Interface, CGI）包装独立的GIS软件。最近，在网络技术和编程环境（例如Java applets, ActiveX控件）等方面的进展促进了更加复杂的WGIS和应用的发展。图5-10展示了Minnesota大学开发的一个WGIS体系结构（称为MapServer）。该系统的设计和实现采用标准的三层体系结构，系统核心组件概述如下。

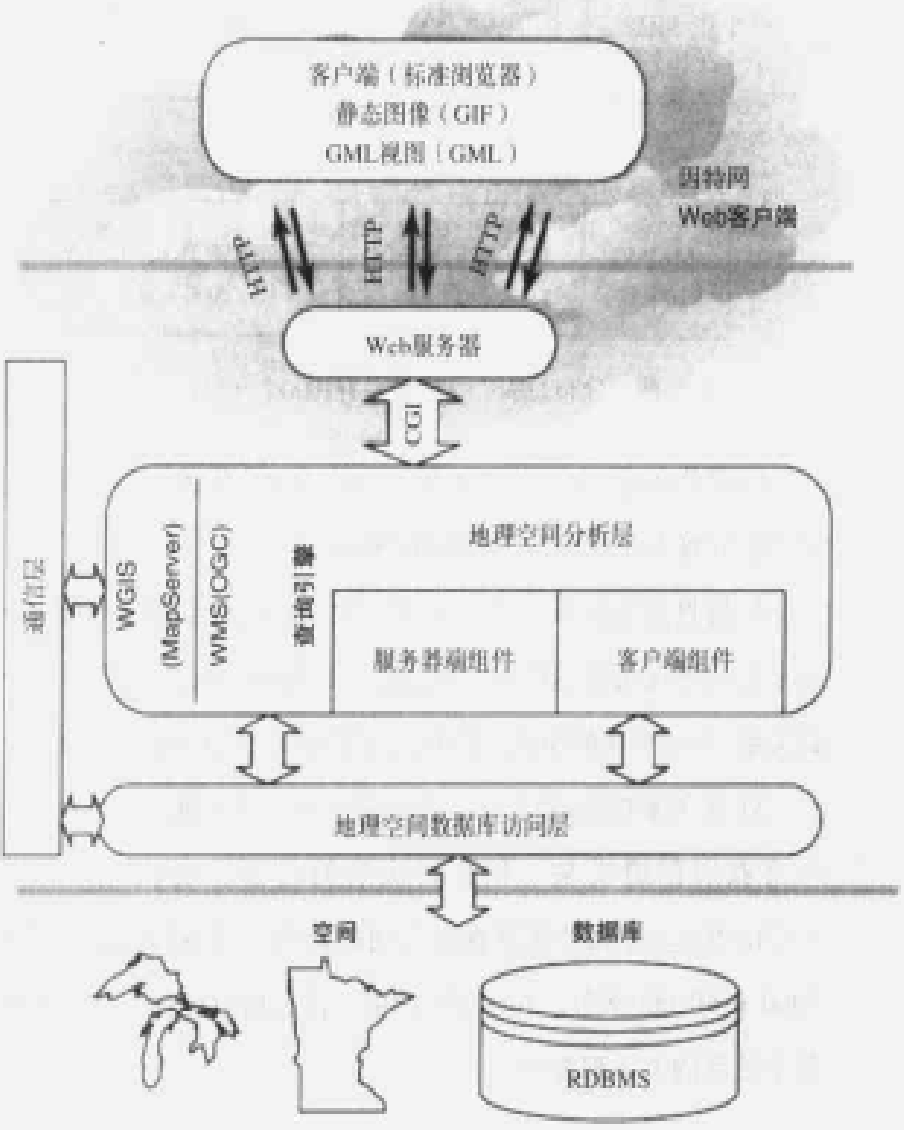


图5-10 Web GIS体系结构

1. 第一层：客户端

客户端通常是任意一个标准的Web浏览器。前端系统由一些超文本标记语言

(Hyper Text Markup Language, HTML)文档组成，这些文档描述用户与服务器交互的图形环境。这些文档由标准HTML标记和JavaScript元素组成。浏览器根据用户输入构造URL(Universal Resource Locator，统一资源定位器)，打开连接服务器的超文本传输协议(Hyper Text Transfer Protocol, HTTP)，并显示从服务器获得的结果。

2. 第二层：应用服务器

应用服务器是WGIS的核心组件，应用服务器由多级组成。

第一级：CGI模块

CGI（公共网关接口）模块是应用服务器上的一个组件，用于响应客户端提交的HTTP请求并将URL解码成标号（例如CGI变量）。分析解析后的标号可以判别客户端的请求是可视化一组空间层（矢量、栅格），或是查询（空间、非空间），或是地理空间分析（过滤、频谱特征分析等等）。根据需要执行的任务，请求被传送到相应的子系统。从子系统获得的结果以Web浏览器能理解的形式提交给客户端。

第二级：地理空间分析系统

核心的地理空间分析功能在这一级中实现。它对CGI模块传入的请求进行语法分析，并向通信层发送输入数据的请求。WGIS是一个分布式系统，这意味着空间数据不必与应用服务器位于同一台服务器上。此外，数据甚至可以不驻留在某一台服务器上，而是分布在多台服务器上。

第三级：通信系统

通信系统负责与通信组中可用的地理空间数据库服务器进行会话。它通过地理空间数据库系统识别并访问需要的数据集，然后将结果返回给地理空间分析系统。

3. 第三层：地理空间数据库访问系统

地理空间数据库访问系统（GeoSpatial Database Access System, GSDAS）基于开放的体系结构，它不受某种特定的RDBMS或文件格式的约束。该层由一系列自定义的开发接口构成，如通用二进制BIP (band interleaved by the pixel) 访问模块，访问标准图像格式的公用领域库（如GeoTIFF），访问专用格式的由厂商提供的库（例如，用于Arc/Info shape文件的shapelib库以及用于ERDAS IMAGINE文件的“eimg”库）。

对于非空间信息（即属性数据）来说，当前的库由对MySQL和ORACLE RDBMS进行的本地访问组成。GSDAS能够识别已知格式并调用相应的模块，这样就可以在某个应用中无缝地访问这些异构的数据源。

可以使用一个简单的配置文件开发Web应用。配置文件定义了CGI变量和空间数据层之间的关系，用于明确地控制应用的各个方面，例如哪些层需要画出来，如何显示这些层，以及如何查询这些层。该通用体系结构易于扩展并提供了强大的Web应用开发环境。该系统也可以自然地扩展以支持最新的标准，比如 Web地图服务器 (Web map server, WMS) 和地理标记语言 (geographic markup language, GML)。

WMS规范对客户端请求地图的方式以及服务器描述其持有数据的方式进行了标准化。任何标准浏览器（或者客户端系统）都可以请求兼容WMS规范的服务器以获得如下所示的一项或多项服务：地图图像（GetMap）、服务级元数据（GetCapabilities）和特定要素的可选信息（GetFeatureInfo）。这些请求以URL的形式提交给服务器。根据用户与客户端系统的交互方式，URL由一组标准参数组成（见[OGIS, 2000]，如宽度、高度、外包框、空间参照系等等）。兼容WMS规范的系统（例如前面描述的WGIS）产生适当的数据集并将该数据集返回给客户端用于可视化／处理。通常，数据集可以转换成标准图像格式，例如GIF、TIFF和JPEG，这些图像格式都是符合行业标准的图像文件格式，关于行业标准的图像文件格式的描述可以参见[Murray and van Ryper, 1999]。这种静态的表示方式给客户端的处理和查询带来了诸多限制。最近，OGC提出了GML标准以缓解这些问题并加强了不同WGIS之间的互操作性。

GML采用可扩展标记语言(eXtensible Markup Language, XML)进行编码，用于传输和存储地理信息，这些地理信息包括地理要素的几何信息和属性信息。GML的最初版本遵循OGC“Simple Feature”规范。GML支持对应于点(Point)、线串(Linestring)、线环(LinearRing)、多边形(Polygon)、多点(MultiPoint)、多线串(MultiLineString)、多多边形(MultiPolygon)和几何体集合（GeometryCollection）的几何元素。GML还提供用于编码坐标的坐标元素以及定义空间范围的框元素。GML表示方式的主要优点在于可以构建真正的可互操作的分布式GIS。

5.5 并行空间数据库系统

并行是DBMS的主要发展趋势。随着数据的快速增长和使用Web浏览器访问数据库的需求的增加，查询的快速响应时间变得极为重要。早期关于并行系统昂贵的价格和脆弱性的争论已经失去意义。常规存储器、处理器和磁盘的价格下跌很快，现在个人专用的电脑（PC）都可以连接在一起以模拟并行环境。并行的实现与并发不同，因为并发是在一台串行执行的机器上模拟并行环境以便多个用户可以同时访问系统。

评估并行系统有两个重要的度量标准：线性加速和线性扩展。线性加速意味着如果硬件数量加倍（处理器、磁盘等从x到$2x$），则完成任务的时间减半。线性扩展意味着如果硬件大小加倍，则完成大小为$2x$的任务所需的时间与原系统完成大小为x的任务所需的时间一样。尽管初看起来，线性加速和线性扩展很容易从串行系统推广到并行系统，但是仍存在着一些降低性能的因素。其中一部分因素如下所示：

- **启动**：如果一个并行操作被划分为数千个小任务，那么启动每个处理器的时间占总处理时间的绝大部分。
- **干扰**：当不同的处理器都试图访问共享资源时就会导致速度下降。
- **扭斜**：如果处理器间的负载分布不平衡，那么并行系统的效率就会大大降低，因为处理时间与最慢的工作所需的时间相关。稍后我们将讨论负载平衡的策略，这样可以减少某些类型的并行体系结构所存在的扭斜问题。

与串行情况一样，并行空间数据库系统的需求与传统的关系数据库的需求是有区别的。最根本的区别在于空间操作既是CPU密集型又是I/O密集型的。此外，SDB是通过高级的、空间可用的声明性语言（例如OGC SQL）来访问的，它们比传统SQL具有更多的基本操作。在描述如何并行实现空间范围查询和连接查询操作之前，我们先简要概述并行数据库系统可用的体系结构。

5.5.1 硬件体系结构

并行数据库系统中有三类主要的资源：处理器、主存模块和二级存储（通常是磁盘）。并行DBMS不同的体系结构就是按这些资源互相作用的方式来分类的。三

种主要的体系结构为共享内存（shared-memory, SM）、共享磁盘（shared-disk, SD）和无共享（shared-nothing, SN），如图5-11所示。

SN体系结构如图5-11a所示。每个处理器只与供其访问的主存和磁盘单元相关。每一组主存和磁盘单元称为一个结点（node），连接这些结点的网络负责结点之间的信息交换。因为将资源共享最小化，所以这种体系结构倾向于将处理器之间的冲突最小化，而这是SM和SD体系结构中的基本问题。其结果是SN体系结构的扩展性比其他两种体系结构要好得多，其线性加速和线性扩展的能力已经经过实践的检验。但如何在不同结点之间平衡负载就成了很困难的任务，特别是在数据高度扭斜的情况下更是如此。数据可用性也会成为一个严重的问题，因为当一个处理器失效时，对应磁盘上的数据也就不可用。这种体系结构也要求更频繁地重组DBMS的代码。

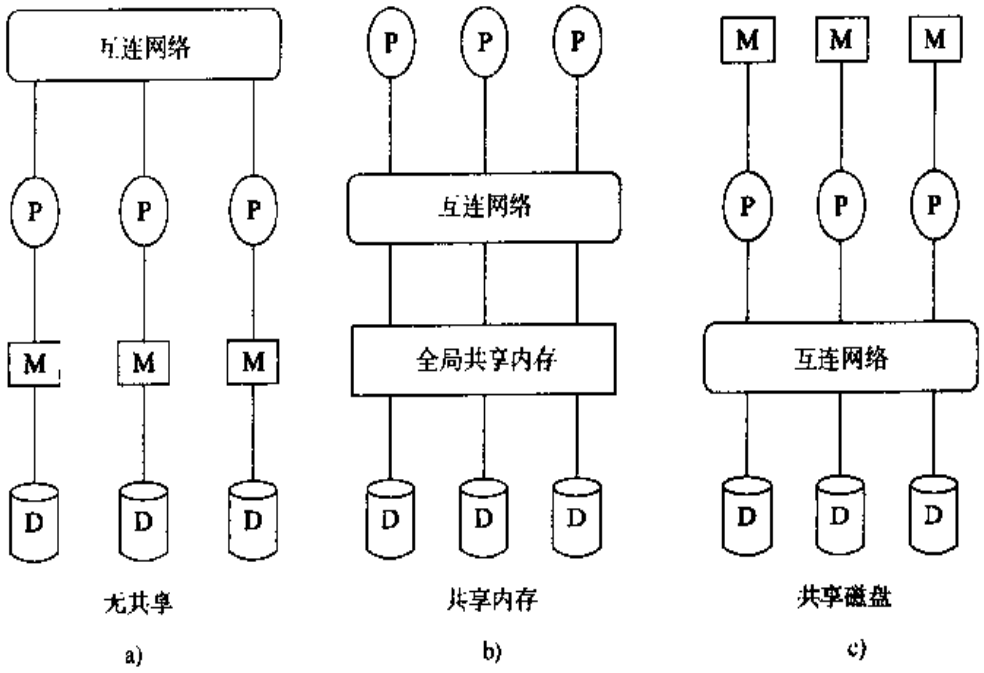


图5-11 并行体系结构选项

SN体系结构比较流行，一些商业并行数据库系统和原型系统都基于该体系结构。但是在SDB中，并行算法的设计和实现中的通信代价以及动态负载平衡都是非常重要的问题，因为需要处理的是大而复杂的对象、或是常常扭斜的数据分布。当前，人们主要关注于最小化I/O所需的代价，因此该领域内大量的研究都基于SD体系结构，甚至基于更简单的单处理器多磁盘的系统，目的则是为了减少通信代价。例如，

Paradise对象关系数据库系统是一个并行的地理空间DBMS，它采用共享磁盘体系结构。并行范围查询要求在运行时进行工作迁移，以便在空间数据对象大小各异和形状复杂的情况下，系统也能够达到较好的负载平衡。SN体系结构的缺点在于：为了在运行时获得动态负载平衡，必须在处理器之间复制数据，而复制就会减少用于存储空间数据的主存总量。SM体系结构使运行时的工作迁移更加容易，因为所有处理器可以平等地访问所有数据。

在SM体系结构中，多个CPU通过一个交互网络相连，并能够访问一个公共的、系统范围的主存，系统中的所有磁盘也是如此。采用SM体系结构可以减少通信的开销，并且很容易实现处理器同步。由于每个处理器都能平等地访问任意一部分数据，所以该体系结构很适合于使负载保持平衡。但是，随着处理器数目的增加，不同处理器对SM和磁盘的频繁访问会导致网络出现瓶颈。由于上述这些原因，加上数据库应用通常都是数据密集型的，所以该体系结构的扩展性很差。

在SD体系结构中，每个处理器都有一个只能被该处理器直接访问的专用主存，但所有处理器都能直接访问系统中所有的磁盘。减少资源共享就会减少SM体系结构中争用网络带宽这个主要问题。该体系结构也因此更具扩展性，但同时它也丧失了SM体系结构在主存方面的优点。与SM体系结构中的原因一样，这时保持数据负载平衡就相对简单了。

5.5.2 并行查询计算

对数据库应用的并行查询计算可以在不同的级别处理。在系统级，并发查询可以由不同的处理器并行处理以增加系统的吞吐量，这称为*查询间并行*。在下一级，同一查询中的不同操作可以由不同的处理器并行处理，这个过程称为*操作间并行*。在更低一级，同一操作可以由不同的处理器来并行处理，这称为*操作内并行*。我们只讨论操作内并行。

操作内并行可以通过函数分块或数据分块来达到。函数分块采用与串行情况不同的特殊数据结构（比如分布式数据结构）和算法。数据分块技术将数据分割到不同的处理器，并在每个处理器上独立执行串行算法。通过将数据“分簇”就可以实现数据分块。

1. 空间分簇

分簇的基本问题陈述如下：给定一组原子数据项、N个磁盘和一组查询，在考虑磁盘容量限制的前提下，将数据项分割到这N个磁盘，使给定查询集的响应时间最小化。理想情况下，响应时间应该为串行响应时间除以处理器数目。一些分簇问题，例如对于所有范围查询的情况，没有办法达到理想响应时间（特殊情况除外）。此外，许多类型的分簇问题属于NP-hard的，通常使用启发式解决方案。不同的分簇技术适用于不同的查询类型和不同的查询集。对于空间数据来说，被访问的数据类型（点、线或多边形）也会影响分簇方法的选择。一般来说，MBR可以作为空间扩展对象的近似。

一些启发式方法可用于解决分簇问题。根据要求在什么时候分割并分配数据，分簇方法可以分为两类：静态负载平衡（在计算处理之前分块并分配数据），动态负载平衡（在运行时进行上述工作）。如果静态分簇后的数据负载平衡很差，那么动态负载平衡可以通过在处理器间传送空间对象来改善数据负载平衡。由于数据分布的高度不一致，以及空间数据的大小与范围差异很大，所以为了实现较好的加速，通常将静态分簇和动态负载平衡结合使用。动态负载平衡通常需要进行数据复制（在多个磁盘之间复制数据），因为本地处理的代价通常小于为扩展对象传输数据的代价。下面回顾一下一种常见的采用空间分块函数的静态分簇方法。

2. 采用空间分块的静态分簇

静态负载平衡一般通过空间分块函数来实现，空间分块函数能够在不同磁盘间系统地分布数据。例如，假定二维点数据分布在坐标系的右上象限，而且，在象限上用网格将二维空间划分成许多单元。用一对数(x, y)来标识某个单元，单元标识符(x, y)的第一个坐标x由0开始从左到右依次增加。第二个坐标y从下到上依次增加。例如(0, 0)表示最下一行最左边的那个单元。

空间分块函数为每个单元分配一个磁盘ID。由N个磁盘组成的磁盘组，其磁盘ID从0到$N-1$。在二维空间中，函数f的定义如下：

$$f: Z^+ \times Z^+ \rightarrow [0, 1, ..., N-1]$$

其中Z^+为正整数空间，有N个磁盘，从0开始。图5-12展示了4个例子，该图将8×8的正象限映射到磁盘号从0到7的八个磁盘上。Z曲线和Hilbert曲线在第4章中描

述过。下面描述并行SDBMS的一个实际应用。

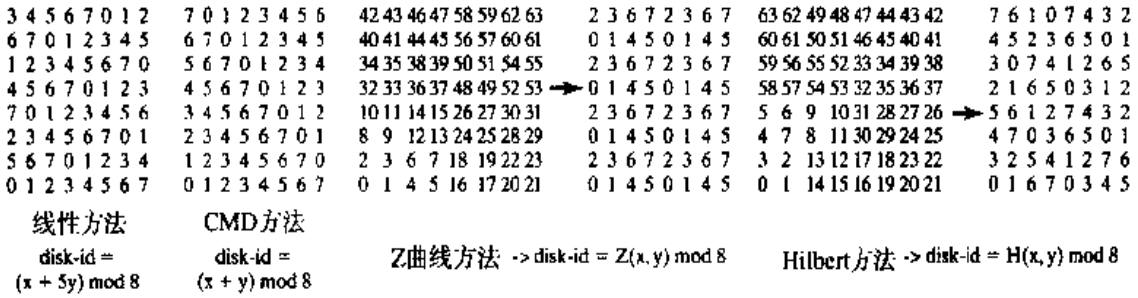


图5-12 不同数据分配方法示例

5.5.3 应用：实时地形可视化

实时地形可视化系统（例如分布式交互模拟系统）是一个虚拟的环境。和其他的虚拟环境、可视化系统和分布式交互模拟系统一样，实时地形可视化系统可以使用户浏览该系统并与计算机生成的三维地理环境实时交互。这种系统有三个主要的组件：交互单元、3D图形单元和SDBMS单元。图5-13展示了典型飞行模拟器中的地形可视化系统的不同组件。系统的SDBMS组件包含存储完整地理数据库的二级存储以及存储和模拟器当前位置相关的数据的主存。图形引擎从SDBMS组件接收空间数据并将空间数据转换成3D对象，然后发送到显示单元。

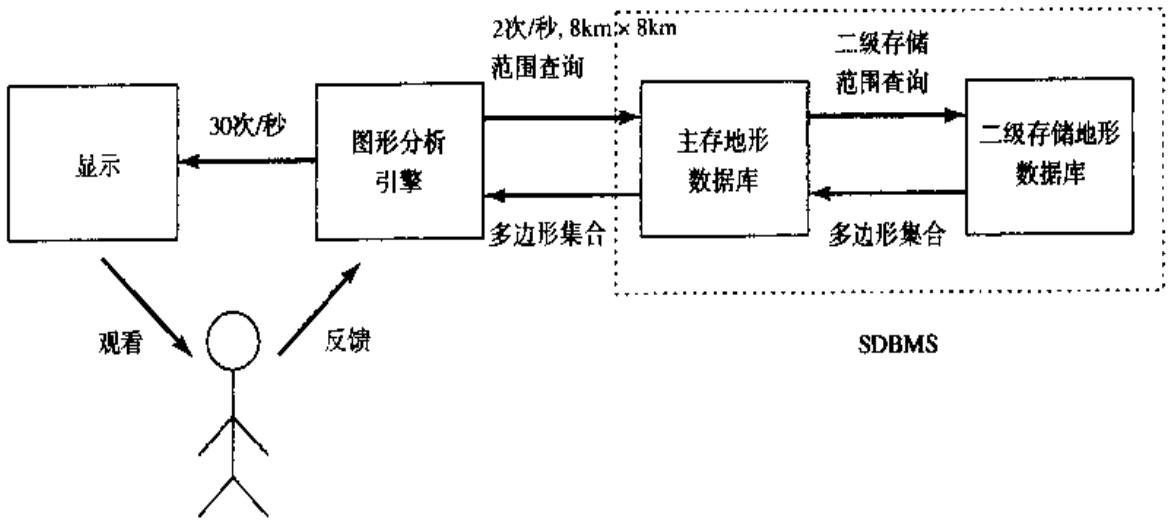


图5-13 地形可视化系统的组件

当用户在地形上移动时，地图的可见部分随时发生变化，应该将在给定位置和用户视点可见的空间对象的子集提供给图形引擎。图形引擎将用户视点转换成范围

查询并将查询发送到SDBMS单元。SDBMS单元检索出当前用户位置周围8km × 8km (km=kilometer, 千米)区域的地理描述，并把该描述发回给图形引擎。例如，图5-14展示了一个多边形地图和一个范围查询。地图中的多边形以虚线的方式显示。范围查询用矩形表示，范围查询的结果以实线的方式显示。SDBMS单元从主存中检索空间数据并计算与用户当前视点的几何交，然后将结果发送回图形引擎。该操作的频率取决于用户在地形上移动的速度。例如，在飞行模拟器的地形可视化中，每秒要生成两次范围查询，这样计算几何交的时间不到半秒。这个应用中的地图包含数万个多边形（也就是数百万条边），范围查询的区域可能占整个地图的20%~30%，这就要求数以百万计的几何交计算在半秒内完成。为了满足响应时间的限制，SDBMS常常在主存中高速缓存一部分空间数据。主存数据库反过来会查询二级存储数据库以得到需要高速缓存的数据子集。由于高速缓存效率很高，所以这个操作的频率应该很低。

图5-14 示例多边形地图和范围查询

现有的针对范围查询问题的串行解决方案一般不能直接用于解决空间范围查询问题，这是因为许多应用要求很高的性能。例如，为了解决范围查询问题中对响应时间的限制（例如图5-13中的半秒），必须使用并行处理以提供所需的性能。

利用基于分簇的数据分块和负载平衡，将解决GIS范围查询问题的串行算法并行化，这种方法可以满足对范围查询高性能的需求。图5-15描述了该方案的步骤。最初，外包框（指查询窗口）被广播至所有处理器，然后每个处理器对本地的一组多边形执行串行GIS范围查询算法。当静态分块未能在处理器间平均分配负载的时候，动态负载平衡（dynamic load balancing, DLB）技术可用于将空间对象传送给空闲处理器。

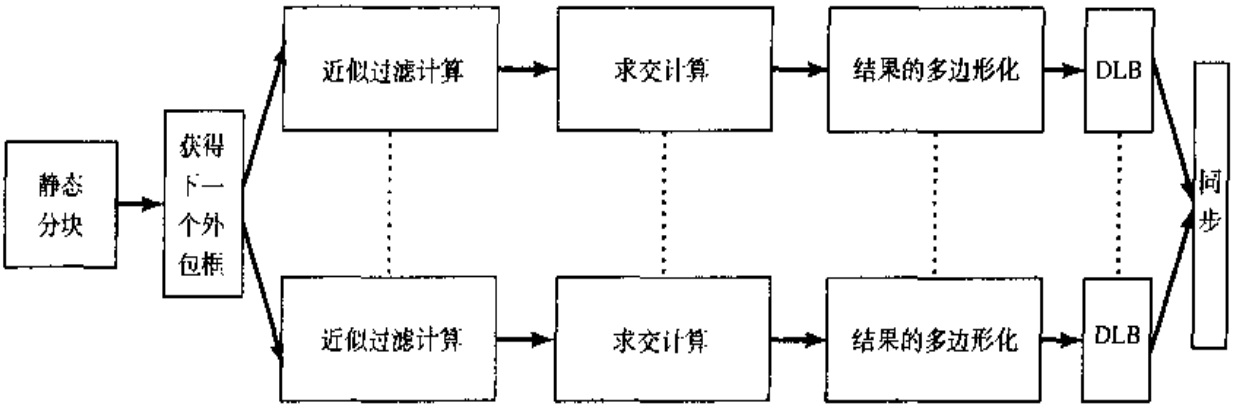


图5-15 并行模式的不同模块

有些技术（例如空间数据预处理）可用于减少范围查询问题的串行代价。注意，后续的范围查询可能与之前的范围查询在空间上产生交叠，所以利用这一点也可以减少范围查询处理的代价。在这种情况下，新的范围查询可以看作是前一个查询的增量。因此，增量范围查询的方法可用于解决这个问题，而增量范围查询可以表示为一个或多个更小的范围查询的组合。

还可以采用预先计算结果的方式来解决范围查询的问题。在数据上建立一个好的网格，计算所有空间对象与网格单元的交并将结果存储在主存中。因为每个范围查询都是网格单元的某种组合，因此可以检索组成范围查询的每个网格单元的交的结果并发送到图形引擎。另一方面，在使用数据分块方法的情况下，大对象可以分解成更小的对象以提高不同处理器之间的负载平衡，从而提高解决方案的效率。

但是这两种方法都会导致图形引擎工作总量的增加，因为它必须在相同的时间内处理更多的对象。图形引擎绘图的代价也随着多边形的增加而增加。此外，分解对象需要更多的内存来储存对象。另一方面，在范围查询之后将各个小对象归

并成一个对象会增加SDBMS组件的工作总量，因为小对象的合并增加了工作量。

例如，图5-16显示了将多边形数据分块成更小集合的不同组合方式。这些组合可以分成四类：I表示没有数据分割。II表示将多边形集合划分成多边形的子集，而每个多边形就是原子单位，不能再对这些多边形再进行分割。与第II类相反，第III类在不同处理器间分割单个多边形/外包框的区域。第IV类方式对单个多边形和外包框的区域和边都进行分割。第III类和第IV类与第II类相比，其潜在优势是可以获得更好的负载平衡和更少的处理器空闲，从而减少并行计算时间。但是要注意，第III类和第IV类方式可能增加工作总量，也可能增加将结果再归并为大多边形的工作。

分割外包框的选择 \ 分割多边形数据的选择	无分割	多边形的子集	小多边形的子集	边的子集
无分割	I	II	III	IV
分割成多个小框	III	III	III	IV
分割成多条边	IV	IV	IV	IV

图5-16 处理器间多边形/外包框分割的可选方案

5.6 小结

DBMS中的查询以某种声明性的形式表达。DBMS软件通常有一些组件负责以高效的方式执行查询。

空间查询操作可以分为四类：点、范围、空间连接和空间聚集。复杂SQL的空间组件通常由这四类中的元素构成。

空间数据库管理系统中的查询处理具有与众不同的特性，即空间查询既是CPU密集型又是I/O密集型的。对于空间查询处理来说，采用过滤-精炼模式将CPU和I/O代价最小化。

对于给定的一个查询，存在许多常见的执行查询的策略。查询优化器是DBMS

中的一个组件，它负责生成执行查询的计划并选择最优或近似最优的计划。查询优化有两个众所周知的技术：逻辑转换和基于代价的优化。

并行与分布是DBMS发展的主要趋势，这两种技术用于解决数据的快速增长和万维网的广泛使用而产生的问题。随着人们使用因特网进行空间数据的分布和处理的需求的增长，促使OGIS提出相关规范，例如WMS和GML。WGIS兼容了这些标准，因而增强了互操作性并促进了协同系统的发展。并行的空间查询处理也引起了人们对前沿技术（如空间实时地形可视化）的浓厚兴趣。

5.7 参考书目

5.1 有关过滤－精炼范型和空间算子代数的内容，请参见[Brinkhoff and Kriegel, 1994; Guting, 1994a]。

5.1.6 在重要的空间连接操作方面，其许多算法已经在相关文献中进行过介绍，例如[Brinkhoff et al., 1993; Patel and DeWitt, 1996]以及[Lo and Ravishankar, 1996; Mamoulis and Papadias, 1999; Patel and DeWitt, 2000; Song et al., 1999]。对于输入都未进行索引的情况，[Arge et al., 1998]介绍了空间连接问题的过滤步骤。[Faloutsos et al., 2000]讨论调整两组点之间空间连接选择性的法则。关于空间距离连接的处理，参见[Shin et al., 2000]。

5.1.7 关于最近邻居搜索的更多讨论，请参阅[Seidl and Kriegel, 1998]和[Goldstein and Ramakrishnan, 2000]。对于“K个最近对查询”，该查询结合了连接、最近邻居查询并从两个数据集中发现K对距离最近的对象，见[Corral et al., 2000]。[Korn and Muthukrishnan, 2000]讨论了逆最近邻居的概念，其中的最近邻居指的是那些将查询点作为其最近邻居的点。

5.2 查询优化是SDB环境中的另一个尚待研究的领域。一些重要研究进展见[Hellerstein and Stonebraker, 1993]和[Chaudhuri and Shim, 1996]。[Theodoridis et al., 2000b]讨论了在使用基于R树结构的情况下，评估选择和连接查询的代价的分析模型。对空间数据库中的选择性评估，可参见[Acharya et al., 1999]。[Leutenegger and Lopez, 2000]则讨论缓冲对R

树性能的影响。

5.3 对空间索引结构分析的详细介绍，可参考[Faloutsos and Kamel, 1994; Pagel et al., 1993a; Theodoridis et al., 1998]。

5.4.3 关于WGIS的更多介绍见[Vatsavai et al., 2000]。该论文提出了一个新颖的保持负载平衡的客户/服务器体系结构，该体系结构比典型的以服务器为中心或以客户为中心的系统有更好的性能。

5.5 关于并行GIS的更详细介绍见[Shekhar et al., 1998]。

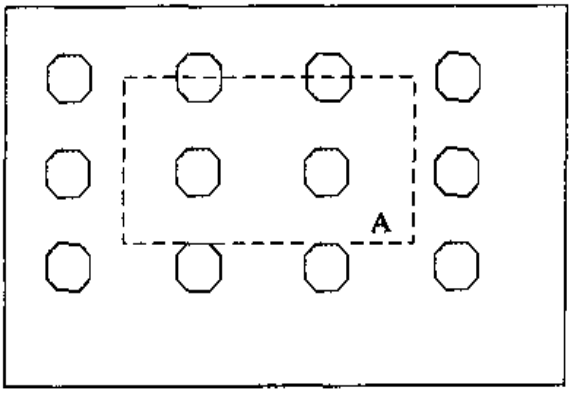
5.8 习题

1. 给定由12个八边形组成的数据空间（见下图），计算下列处理范围查询的代价：

(i) 直接处理。

(ii) 使用过滤-精炼策略处理。

假定空间上没有索引，处理每个多边形的代价等于其边数。对于过滤步骤，假定每个多边形用其MBR表示。



2. 如果将八边形换成六边形，结果会怎样？

3. 与传统关系数据库相比，SDB的查询优化有什么特殊之处？

4. 考虑空间连接中的树匹配策略和带索引的嵌套循环策略，为以下情况选择合适的空间连接策略：

(i) 交叠连接：数据集都没有R树索引。

(ii) 交叠连接：一个数据集有R树索引。

(iii) 交叠连接：两个数据集都有R树索引。

5. 如果有连接谓词而不是交叠，那么空间连接可采用哪个连接策略?

6. 树转换：画出下列查询的查询树。

(i) City表中的哪个城市与River表中的某条河流距离最近（见3.5节）？给出相应的SQL语句。

(ii) 列出GDP比加拿大高的国家。

(iii) 列出在某一方面（湖泊／邻国）有与法国一样多的国家。

7. 应用树转换规则为上面的每个查询树生成几个等价查询树。

8. 考虑一个复合查询，包括一个空间连接和两个空间选择。但该策略不能进行空间连接。利用湖泊和设施的最初的索引。此外，读写文件1和文件2是否会增加代价？设计一个能够同时处理连接和其下两个选择的可选策略。

9. 本章重点介绍交叠谓词的点选择、范围查询选择和空间连接的策略。请问这样是否足以支持SQL3/OGIS中空间查询的处理（见第3章）？

10. 先以交叠谓词作为过滤条件，然后加上一个精确的几何操作，如何以这样的方式来处理不同的拓扑谓词（例如，inside、outside、touch、cross）？

11. 采用下列一个存储方式：网格文件、带Z序的B+树，设计用于最近邻居查询的高效一遍算法。

12. 研究XML在因特网和网络商务环境中的发展变化。重绘图1-5以加入XML数据管理的最新趋势。

13. 比较最近邻居查询的两遍算法与一遍算法的计算代价。在什么情况下选择一遍算法更好?

14. 比较下面几对概念：

(i) 并行与分布式数据库

(ii) 分簇与动态负载平衡

(iii) 无共享与共享磁盘体系结构

(iv) 客户服务器与协同系统

15. 什么是GML？它对于空间数据库和GIS有何意义?

16. 扩展最近邻居查询的一遍算法以解决K个最近邻居的查询。

17. 考虑5.4节中Farm与Disease-map之间的空间连接查询，采用图5-9中的表定

义。计算下列三种策略的通信代价（传送的字节数）：

(i) 半连接。

(ii) 将Farm关系传送到另一个站点。

(iii) 将Disease-map传送到另一个站点。

半连接策略是否总是比其他两个策略的代价更小呢?

18. 研究图5-12中基于空间分区函数的静态分簇技术。评估用于下列查询的每种方法获得的I/O加速，假定每个单元(x, y)位于独立的磁盘块上：

(i) 行查询，比如最下一行中的单元。

(ii) 列查询，比如最左一列中的单元。

(iii) 范围查询，比如最左两列最下四行的单元。

第6章
空间网络

空间网络数据库（spatial network databases, SNDB）是空间数据库的重要组成部分，它构成许多重要应用的核心，这些应用包括：运输规划、空中交通管制、水电煤气设施、电话网络、城市管理、河道运输以及灌溉渠管理。到目前为止，我们已经讨论了空间对象及其相互关系，这些关系是基于邻近性（proximity）的概念产生的。显然，空间上的邻近性决定相邻对象的行为。不过，对于空间网络数据库来说，起根本作用的是基于连通性（connectivity）的关系。

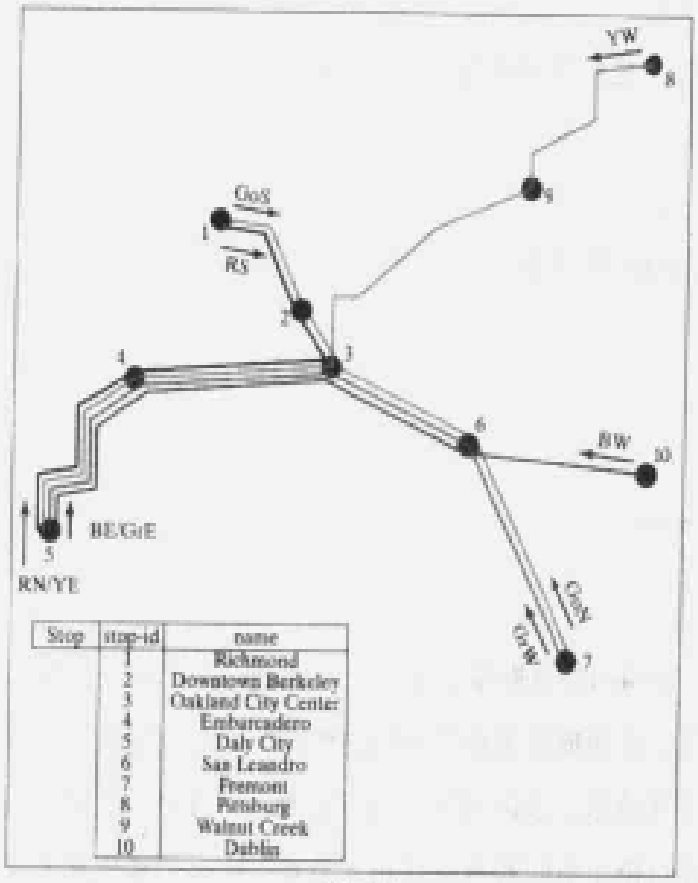
要实现由邻近性向连通性的观念上的转变，就要求对SNDB另行看待。本章将针对空间网络应用的特定要求，重点讨论如何改进前面介绍的各种数据库技术（数据模型、查询语言和索引）来实现这些要求。

6.1节通过介绍两个网络应用的例子，引出一些本章内容涉及的概念。6.2节对图的概念和图的操作进行简单介绍；6.3节揭示标准查询语言不适合常见的图查询的原因，并且介绍两种标准查询语言的扩展以弥补这种不足；6.4节扼要叙述降低I/O

代价的图的算法；最后，6.5节重点介绍优化标准图操作的方法。

6.1 网络数据库示例

这里我们以两个典型的空间网络应用（一个铁路网和一组河流水系）为例，并且在本章中会利用这两个例子来说明各部分的内容。首先，通过介绍这两个网络应用中的一些典型查询，我们引出数据库管理的相关问题：数据建模、查询语言和索引。



Stop	stop-id	name
	1	Richmond
	2	Downtown Berkeley
	3	Oakland City Center
	4	Embarcadero
	5	Daly City
	6	San Leandro
	7	Fremont
	8	Pittsburg
	9	Walnut Creek
	10	Dublin

a) BART

b) 河流网

图6-1 空间网络的两个例子

1. 铁路网络

BART铁路系统主要负责加州北部的旧金山以及邻近地区的铁路业务。它有5条铁路干线和10条运营线路（每条线路对应一个方向）。这些运营线路彼此相互交叉地通过大部分城区以及郊区城镇。所有铁路干线交汇在旧金山城区，并向四周不同方向辐射，图6-1a是一张BART系统的简化图。下面列出的是一些与该系统有关的典型查询。

1) 找出Yellow West（YW）线沿途的车站数。

2) 列出可以从Berkeley镇直接抵达的所有车站。

3) 列出连接Berkeley镇和Daly市的运营线路。

4) 找出Bule West（BW）线的最后一个车站。

2. 河流网

第二个例子涉及美国的两个主要水系：密西西比水系和科罗拉多水系，它们分别由密西西比河和科罗拉多河及其各自的主要支流组成。图6-1b是这两个水系的示意图。同样，我们列出几个与这些水系有关的查询。

1) 列出密西西比河在明尼苏达州境内的所有直接或间接支流的名称。

2) 列出科罗拉多河的所有直接支流。

3) 如果河流P1溢洪，哪些河流会受到影响。

6.2 概念数据模型、逻辑数据模型和物理数据模型

作为开始，本节将从数据库设计过程入手，这在第2章中已有所介绍。空间网络应用首先在概念层次上建模，然后再映射成逻辑模型。

为了构建空间网络应用模型，我们将重点放在广为采用的图（graph）的抽象数据类型上。空间网络最重要的概念是对象之间的连通性，它可以在一个图论框架中简洁地表达出来。抽象数据类型图可以嵌入到逻辑模式中（例如，对象-关系、SQL3等），而象形图（pictogram）则用于概念数据模型（例如，ER模型、UML），它通过隐含基于图的联系来简化概念模型图。习题9就是采用象形图方法的一个练习，读者可以自行解决。

6.2.1 逻辑数据模型

1. 图的基本概念

一个图$G = (V, E)$是由一个有限顶点集V和顶点之间的边集E组成的。因此，边集

E是顶点集V上的一个二元关系。如果构成边集的各个顶点对是有序的，那么图G就是有向的（directed）；否则该图是无向的（undirected）。顶点和边有时也分别称为结点（node）和链接（link）。有序顶点对的第一个顶点称为前驱（predecessor）或者源（source），第二个顶点称为后继（successor）、目的（destination）或汇点（sink）。

在BART例子中，结点是铁路系统中的车站，而链接表示车站之间的直接连接。结点不一定是空间中的点。在河流网例子中，图的结点表示的是河流，而不是河流的端点。如果某条河流流入另一条河流，那么可以用链接来表示这种关系。河流网图的不同表示适用于不同的应用。BART系统可以根据应用的需要构建为有向图或者无向图模型，而河流网的最自然表示方式是有向图。

图的结点和链接有时要添加标号（label）和权重（weight），以便表示附加的信息。例如，可以为铁路系统中的结点加上名字或者地理坐标（也可以将两者都加上去），车站之间的距离可以作为边的权重。

如果两条边共享一个结点，那么它们是邻接的（adjacent），一系列邻接边组成一条路径（path）。例如，序列（v_0，v_1），（v_1，v_2），…，（v_{n-2}，v_{n-1}），（v_{n-1}，v_n）表示一条路径，因为每条边都与前一条边或者后一条边有一个公共结点。如果端点v_0和v_n是同一个结点，那么这条路径称为一个环（cycle）。河流网中没有环，而在铁路系统中，一条往返旅行线路构成一个环。

2. 图的基本操作

构建成图模型的应用有许多通用的操作。我们列出其中的一些操作，这是在较高层次上用面向对象的概念对[Bailey，1998]中的图模型进行扩展而得到的操作。这里我们假定读者熟悉基本的面向对象术语和Java语法，一些重要图类的更详细的Java接口描述可以参见[Bailey，1998]的著作。不过，需要强调的是，所有操作都可以用不同方法来实现，这取决于应用和图的存储结构。

图有三个基本类（或者实体）：Graph（图类）、Vertex（顶点类）和Edge（边类）。对于每个类，我们列出相应的一组通用操作，也叫方法。

```
public class Graph
{
        public   void   add(Object label);
        //label表示增加的顶点
```

```
        public  void  addEdge(Object v1, Object v2, Object label);
        //在顶点v1和v2之间增加一条边
        public Object delete(Object label);
        //删除顶点
        public Object deleteEdge(Object v1, Object v2);
        //删除顶点v1和v2之间的边
        public Object get(Object label);
        //返回顶点的标号
        public Edge getEdge(Object v1, Object v2);
        //返回连接顶点v1和v2的边
        public Object get-a-Successor(Object label);
        //返回顶点的邻接结点
        public Iterator getSuccessors(Object label);
        //返回所有邻接结点
        public Iterator getPredecessors(Object label);
        //返回所有前驱结点
        public boolean isDirected();
        //如果是有向图，则返回true
}
```

Vertex和Edge是另外两个重要的类。大多数图类算法要反复同顶点和边打交道，通过标记来记录已被访问过的顶点和边，因此直接在类中提供这些操作是很有用的。在图的接口中Vertex是可见的，这个类被声明为公用的：

```
public class Vertex
{
        public Vertex(Object label)
        //这个类的构造函数，创建一个具有标号的结点
        public Object label()
        //返回该顶点的标号
        public boolean visit()
        //标记顶点已访问过
        public boolean isVisited()
        //如果顶点已被访问过，则返回true
}
```

与Vertex类相似，Edge类也被声明为公用的，这主要是由于在图类中用到了*getEdge*方法，使得Edge类在图类的接口中是可见的。下面只列出Edge类的一些基本方法。读者可以列出自己认为重要的Edge类的其他方法。

```
public class Edge
{
        public Edge(Object v1, Object v2, Object label, boolean directed)
        //这个类的构造函数，若directed为true，则是有向图

        public Object start()
        //返回边的第一个结点

        public Object end()
        //返回边的第二个结点
}
```

有些系统包含一些其他的数据类型，比如[Guting, 1994b;Shekhar and Liu, 1997]中提到的路线（route）或路径（path）。通俗地说，一条路径是一个序列，这个序列从一个结点开始，到另一个结点结束、还以遍历顺序包含其他结点和边。BART例子中的运营线路就是路径。路径的重写（rewrite）操作就是对结点和边的子集进行一组变换，产生另一条路径，例如，rewrite（edge→, path P）表示将路径P的所有边去掉。路径的评估操作是将路径中结点和边的属性（例如边长）进行聚集，产生一个标量（即数值）。

6.2.2 物理数据模型

光有Graph、Node和Edge类还不够，它们只是许多图算法（如连通性、最短路径等）所需的基本要素。能获得一致认可的用于空间网络建模的抽象数据类型集尚在酝酿之中，希望不久见到这种标准。邻接矩阵（Adjacency-matrix）和邻接表（Adjacency-list）是人们熟悉的两种主存数据结构，可以用于图的实现。在邻接矩阵中，行和列表示图的顶点，矩阵项的取值为1或0，这要看两个顶点之间是否有边，如果有边则为1，否则为0，如图6-2b所示。如果是无向图、那么矩阵是对称的。邻接矩阵结构可以快速回答对边的查询，例如，“边（u, v）是否在图G中”。

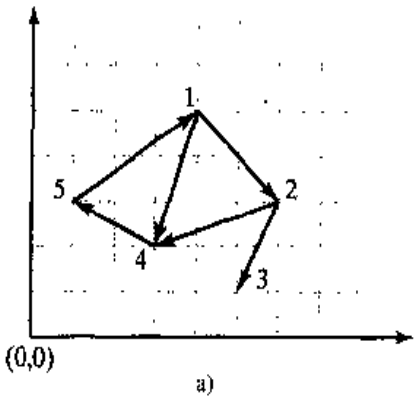
对于涉及枚举图的顶点的查询，例如，“找出v的所有邻近顶点”，利用邻接表结构可以高效完成。邻接表数据结构是一个指针数组，数组的每个元素对应图中的一个顶点，而指针则指向该顶点的一个直接后继顶点表，如图6-2c所示。

我们在前面提到过，主存的数据结构不适合于数据库应用，这是因为数据库中的数据量太大，无法一次全部放到主存中。因此，在把图算法部署到DBMS（数据库管理系统）之前，必须确保它们在I/O操作中的效率。

有向图可以在关系模型上实现，这里分别用关系R和S表示图的结点和边，图6-2d

给出了Node关系*R*和Edge关系*S*。我们注意到，在关系*S*中，边可以按第一个属性值作物理上的聚集。关系*R*中包括了结点的坐标值，关系*S*中则包括了结点之间的距离（权重）。

图6-2e是图的一种非规范化表示，通常用于加快最短路径的计算速度。这种结点表的表达方式中包含坐标值、后继结点列表和前驱结点列表。此外，图还可以用其他的表形式进行表示。例如，在公共交通应用中，可以用一个结点表以及一个线路表来反映结点和线路之间的关系。



a)

目的

源	1	2	3	4	5
1	0	1	0	1	0
2	0	0	1	1	0
3	0	0	0	0	0
4	0	0	0	0	1
5	1	0	0	0	0

b) 邻接矩阵

1	→	2 4
2	→	3 4
3	→	N
4	→	5
5	→	1

c) 邻接表

结点（R）

id	x	y
1	4.0	5.0
2	6.0	3.0
3	5.0	1.0
4	3.0	2.0
5	1.0	3.0

边（S）

源	目的	距离
1	2	$\sqrt{8}$
1	4	$\sqrt{10}$
2	3	$\sqrt{5}$
2	4	$\sqrt{10}$
4	5	$\sqrt{5}$
5	1	$\sqrt{18}$

d) 结点关系和边关系

id	x	y	后继	前驱
1	4.0	5.0	(2,4)	(5)
2	6.0	3.0	(3,4)	(1)
3	5.0	1.0	( )	(2)
4	3.0	2.0	(5)	(1,2)
5	1.0	3.0	(1)	(4)

e) 非规范化结点表

图6-2 图的三种不同表示

我们再看一下BART和河流网这两个空间网络例子。有关BART的信息记录在三个关系中。第一个关系是Stop，它有两个属性：stopid和name，这两个属性分别对应车站的标识和名字。第二个关系是DirectedRoute，它也有两个属性：number和name，分别表示运营线路号和线路名。表6-1显示了这两个关系的相应表格。最后一个关系是RouteStop，如表6-2所示，它有三个属性：routenumber、stopid和rank。其中，routenumber是线路号，对应关系DirectedRoute；stopid是车站标识，对应关系Stop；rank是车站在运营线路中的位置。

河流网有两个关系：River和FallsInto，如表6-3所示。*riverid*和*name*是关系River的两个属性，分别对应河流的标识号和名字。关系FallsInto也有两个属性：source和dest，它们是River关系的外码，其中，source所对应的河流将流入dest对应的河流。

表6-1 BART系统的Stop表和DirectedRoute表

Stop	stopid	name
	1	Richmond
	2	Downtown Berkeley
	3	Oakland City Center
	4	Embarcadero
	5	Daly City
	6	San Leandro
	7	Fremont
	8	Pittsburg
	9	Walnut Creek
	10	Dublin

DirectedRoute	number	name
	1	Red South
	2	Red North
	3	Gold South
	4	Gold North
	5	Yellow West
	6	Yellow East
	7	Blue West
	8	Blue East
	9	Green West
	10	Green East

表6-2 BART系统的RouteStop表

RouteStop	routenumber	stopid	rank
	1	1	1
	1	2	2
	1	3	3
	1	4	4
	1	5	5
	2	5	1
	2	4	2
	2	3	3
	2	2	4
	2	1	5
	3	1	1
	3	2	2
	3	3	3
	3	6	4
	3	7	5
	4	7	1
	4	6	2
	4	3	3
	4	2	4
	4	1	5
	5	8	1
	5	9	2
	5	3	3
	5	4	4
	5	5	5
	6	5	1
	6	4	2
	6	3	3
	6	9	4
	6	8	5
	7	10	1
	7	6	2
	7	3	3

（续）

RouteStop	routenumber	stopid	rank
	7	4	4
	7	5	5
	8	5	1
	8	4	2
	8	3	3
	8	6	4
	8	10	5
	9	7	1
	9	6	2
	9	4	3
	9	5	4
	10	5	1
	10	6	2
	10	10	3
	10	7	4

表6-3 河流网的River关系和FallsInto关系

River	riverid	name
	1	Mississippi
	2	Ohio
	3	Missouri
	4	Red
	5	Arkansas
	6	Platte
	7	Yellowstone
	8	P1
	9	P2
	10	Y1
	11	Y2
	12	Colorado
	13	Green
	14	Gila
	15	G1
	16	G2
	17	GI1
	18	GI2

FallsInto	source	dest
	2	1
	3	1
	4	1
	5	1
	6	3
	7	3
	8	6
	9	6
	10	7
	11	7
	13	12
	14	12
	15	13
	16	13
	17	14
	18	14

在BART例子中，各条列车线路所对应的所有路径在关系RouteStop中进行实体化，而在河流网的关系中，只枚举出网络图的顶点和边（见图6-3）。

6.3 图的查询语言

我们在第3章中提到过，为了查询空间数据，必须对通用查询语言（如RA和SQL92）的功能进行扩展。在这一节，我们将看到，很多重要的图操作无法用RA和SQL92表达。确定从某一给定结点是否可以到达图的所有结点，这是一个重要的图操作。如果不对图作某些假定，基于关系代数的查询语言就不能表达一些重要的图查询。

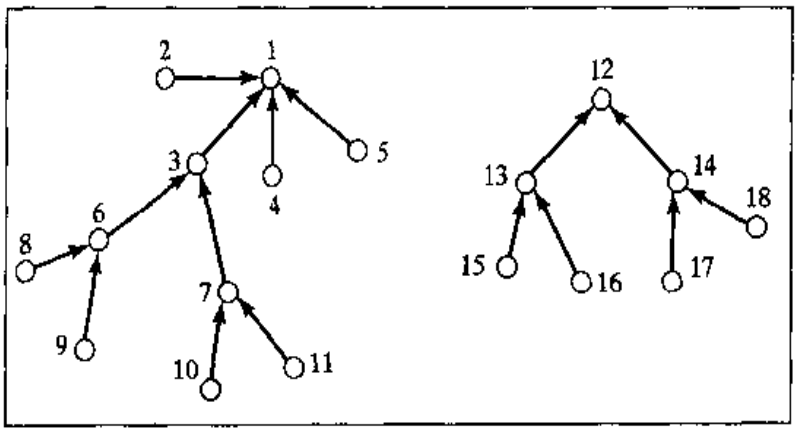


图6-3 河流网例子的图模型

6.3.1 关系代数的缺陷

一个重要的图操作是确定图的传递闭包（transitive closure）。图G（V，E）的传递闭包G^*是满足下列条件的图，它与G有相同的顶点集V，但它的边集则由G的所有路径组成。我们来看图6-4所示的例子。G（见图6-4a）的传递闭包如图6-4c所示，在G^*中新增加了4条边，分别对应G的一条路径。例如，在原图G中，可以从结点1到达结点3，所以G^*就有了新边（1，3）。

与G和G^*对应的关系分别由图6-4c和6-4d表示。用RA操作不可能从G推导出G^*的关系，除非能得到有关G的结构的附加信息。特别是，为了推导图的传递闭包，必须知道其最长路径的长度。下面是从G推导出传递闭包G^*的RA操作序列。

1	*Rename*		$G_1 = G \text{ and } G_2 = G.$
2	T_1	=	$\Pi_{G_1.source,G_2.dest}(G_1 \bowtie_{G_1.dest=G_2.source} G_2).$
3	T_2	=	$G \cup T_1.$
4	T_3	=	$G \bowtie_{G.dest=T_2.source} T_2.$
5	T_4	=	$\Pi_{T_3.source,T_3.dest} T_3.$
6	G^*	=	$T_2 \cup T_4.$

在推导传递闭包的过程中需要两个连接操作，这是因为图G的最长路径通过了2个中间结点。如果得不到最长路径的信息，则RA操作就无法确定何时才能生成传递闭包。注意下面的公式：

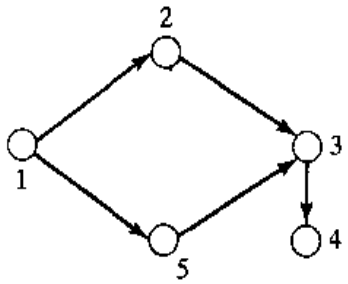
$$G' = G' \cup (G' \bowtie G)$$

规范地说，对于给定一个关系R，它的传递闭包X是下面这个递归方程的最小点

集解。

$$X = X \cup (R \bowtie X)$$

SQL3提出用一个递归操作RECURSIVE来处理传递闭包操作。在介绍RECURSIVE操作之前，我们先讨论一个与之相似的操作CONNECT，该操作已用于SQL92的许多实现中。

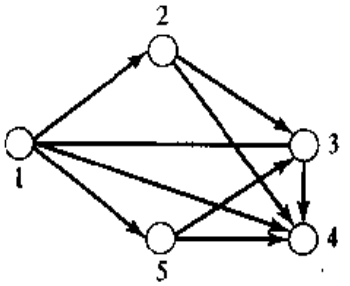


a) 图G

R

源	目的
1	2
1	5
2	3
3	4
5	3

b) 图G的关系形式



c) 图G的传递闭包G*

X

源	目的
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

d) 传递闭包的关系形式

图6-4 关系*R*和它的传递闭包*X*

6.3.2 SQL CONNECT 子句

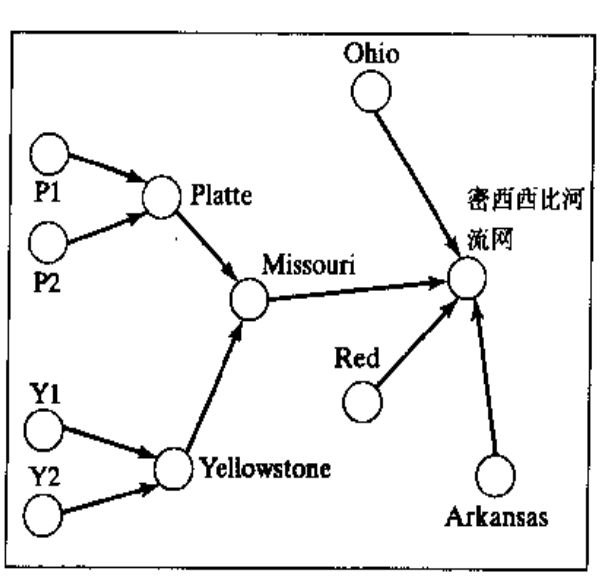
CONNECT子句可用于遍历一个有向无环图（directed acyclic graph，DAG），由START WITH子句和CONNECT BY子句来控制图的遍历。START WITH子句指定根结点，CONNECT BY子句定义父行（parent row）与子行（child row）之间的关系，用PRIOR操作符来决定搜索的方向。河流网就是一个有向无环图，其中，每条边的方向由河的流向决定。利用河流网，我们来介绍，如何用SQL中CONNECT BY子句来表达一些与之相关的重要查询。

1) 列出riverid等于1的河流（密西西比河）的所有直接或间接支流的riverid。

```
SELECT    source
FROM      FallsInto
CONNECT   BY PRIOR source = dest
START     WITH dest = 1
```

注释 这条查询的中间结果和最后输出如图6-5所示。查询从START WITH开始。首先选择表FallsInto中dest字段为1的所有行，查询结果如图6-5b所示。下一步使用CONNECT子句中的条件source = dest，把满足该条件所有河流（即它们流入选定河流）加到输出结果中。递归地重复这个过程，直到层次中的所有结点都被访问过为止。CONNECT子句用宽度优先搜索（BFS）来遍历DAG。6.4节将对BFS进行更详细的讨论。

注意 关键字PRIOR决定搜索方向。本例中，返回的是dest为1的结点的所有子结点。换一个角度，如果把上述查询中的CONNECT子句改为CONNECT BY source = PRIOR dest，那么返回的将是dest为1结点的所有父结点。



a) 密西西比河流网（Y1=Bighorn河，Y2=Powder河，P1=Sweet河，P2=Big Thompson河）

SOURCE
Ohio
Missouri
Red
Arkansas

1) 第一级

SOURCE
Ohio
Missouri
Platte
Yellowstone
Red
Arkansas

2) 第二级

SOURCE
Ohio
Missouri
Platte
P1
P2
Yellowstone
Y1
Y2
Red
Arkansas

3) 最终结果

b) CONNECT子句的顺序

图6-5 SQL的CONNECT子句操作

2) 列出密西西比河的所有直接或间接支流的名字。

```
SELECT    name
FROM      River
WHERE     riverid IN
(
```

```
SELECT   source
FROM     FallsInto
CONNECT  BY PRIOR source = dest
START    WITH dest IN
(
SELECT   riverid
FROM     River
WHERE    name = 'Mississippi'
)
)
```

注释 这是一个双重嵌套查询。由于CONNECT子句是在WHERE子句前处理，所以增加了复杂性。

3) 列出科罗拉多河的所有直接支流。

```
SELECT   name
FROM     River
WHERE    riverid IN
(
SELECT   source
FROM     FallsInto
WHERE     Level ≤ 1
CONNECT  BY PRIOR source = dest
START    WITH dest IN
(
SELECT   riverid
FROM     River
WHERE    name = 'Colorado'
)
)
```

注释 在前一个查询中，我们已经指出WHERE子句是在CONNECT子句之后处理的。由此我们可以选择DAG的级别。在查询中，我们可以通过在WHERE子句中使用Level来选择DAG的级别。由于本查询只需列出密西西比河的直接支流，所以我们设定Level≤1。

4) 如果河流P1会溢洪，那么将会有多少条河流受到影响。

```
SELECT   COUNT(source)
FROM     FallsInto
CONNECT  BY source = PRIOR dest
START    WITH source IN
(
```

```
SELECT   riverid
FROM     River
WHERE    name = 'P1'
)
```

注释 我们可以通过调整CONNECT子句中的PRIOR关键字的位置来改变搜索方向。

6.3.3 BART系统的查询示例

现在我们给出一些BART系统的查询例子，这些查询基于表6-1和表6-2中所示的关系。表6-2清楚地给出了所有的运营线路、线路中的车站和车站在线路中的顺序。

1) 在Yello West（YW）线路上有多少个车站。

```
SELECT   COUNT(R.stopid)
FROM     RouteStop R, DirectedRoute D
WHERE    R.routenumber = D.rnumber AND
         D.name = 'Yellow West'
```

注释 由于RouteStop表所包含的是线路号而不是线路名，因此为了回答上述查询，我们必须把RouteStop关系和DirectedRoute关系进行连接。

2) 按字母顺序列出Red North线路上的所有车站。

```
SELECT   S.name
FROM     Stop S, RouteStop R, DirectedRoute D
WHERE    D.number = R.routenumber AND
         R.stopid = S.stopid AND
         D.name = 'Red North'
ORDER    BY name
```

注释 车站和线路的名字分别记录在Stop表和DirectedRoute表中，车站和线路之间的关系由RouteStop表给出。因此，本查询需要两个连接：RouteStop与DirectedRoute之间的连接以及RouteStop与Stop之间的连接。ORDER BY子句表示结果以字母顺序列出。

查询结果 本查询的输出见下表。

Daly City
Downtown Berkeley
Embarcadero
Oakland City Center
Richmond

3) 列出Red South上的第二个车站。

```
SELECT  S.name
FROM    Stop S, RouteStop R, DirectedRoute D
WHERE   D.number = R.routenumber AND
        R.stopid = S.stopid AND
        R.rank= 2 AND
        D.name = 'Red South'
```

注释 本查询与前一个查询相似，只是WHERE子句中多了一个约束条件：R.rank = 2。

4) 列出Blue West上的最后一个车站。

```
SELECT  S.name
FROM    Stop S, RouteStop R, DirectedRoute D
WHERE   S.stopid = R.stopid AND
        R.routenumber = D.number AND
        D.name = 'Blue West' AND
        R.rank =
(
SELECT  Max(R1.rank)
FROM    RouteStop R1, DirectedRoute D1
WHERE   D1.number = R1.routenumber AND
        D1.name = 'Blue West'
)
```

注释 这是一个嵌套查询。

5) 列出连接Downtown Berkeley与Daly City的线路号。

```
SELECT  R1.routenumber
FROM    RouteStop R1, RouteStop R2, Stop S1, Stop S2
WHERE   R1.routenumber = R2.routenumber AND
        R1.stopid = S1.stopid AND
        R2.stopid = S2.stopid AND
        S1.name = 'Downtown Berkeley' AND
        S2.name = 'Daly City'
```

注释 在该查询中，我们首先将关系RouteStop和它自己连接起来，这种

连接叫做自连接。然后在自连接关系的两边连接Stop关系。最后对满足条件的行（同时出现Downtown Berkeley和Daly City的行）在routenumber属性上作投影操作。

6) 列出从Downtown Berkeley能直达的所有车站。

```
SELECT   Distinct(S2.name)
FROM     RouteStop R1, RouteStop R2, Stop S1, Stop S2
WHERE    R1.routenumber = R2.routenumber AND
         R1.stopid = S1.stopid AND
         R2.stopid = S2.stopid AND
         S1.name = 'Downtown Berkeley' AND
         R1.stopid <> R2.stopid
```

注释 本查询的SQL表达式与前一个查询相似，RouteStop表在属性routenumber上作自连接。然后在自连接表的两边对Stop表作连接操作。

查询结果 本查询的输出见下表。

Daly City
Embarcadero
Fremont
Oakland City Center
Richmond
San Leandaro

6.3.4 趋势：SQL3的递归

SQL3提供了计算一个关系的传递闭包的结构。例如，在图6-4中，关系*R*的传递闭包*X*可以用下面的语句推导出来：

```
1)  WITH  RECURSIVE  X(source, dest) AS
2)                   (SELECT source, dest FROM R)
3)                   UNION
4)                   ( SELECT R.source, X.dest
5)                    FROM R, X
6)                    WHERE R.dest = X.source);
7)  SELECT  * FROM X
```

注释 通过使用递归关系模式之后的关键字短语WITH RECURSIVE，就可以定义出递归关系，或者其值是利用递归方式计算出来的关系。规范的定义如下所示：

```
WITH RECURSIVE <关系模式> <涉及递归关系的查询>
```

第1 行定义了递归关系X；第2 行用关系R来初始化关系X，这是因为关系X包含关系R；第4 行到第6 行执行递归操作，递归操作可以用下面的传递规则来表示：

$$X(a,\ b) \text{ and } X(b,\ c) \rightarrow X(a,\ c)$$

第7行列出关系X的所有元组。

6.3.5 趋势：SQL3的网络ADT

现在我们讨论如何构建空间网络所要用到的基本ADT。利用Oracle8的对象关系特性，就可以创建出对应于顶点、边和图的ADT。

首先，创建一个新的类型vertexType，它有3个属性：*vid*、*x*、*y*。*vid*是顶点的唯一标识，而*x*和*y*是该顶点相应的空间位置。

图G (V, E)的边集E是顶点集V上的一个二元关系，因此，edgeType的两个属性（即*source*和*dest*）可以声明为引用vertexType，若想保持关系的规范化，就必须这样做。edgeType的属性*weight*是边的权重。

现在，我们可以创建对应于vertexType和edgeType的表。这两个表创建完之后，就可以用SQL中的INSERT子句向表中插入数据。例如，下面的语句将向Vertex表中插入2个元组。

```
INSERT INTO Vertex VALUES(1,3.0,4.0)
INSERT INTO Vertex VALUES(2,0.0,1.0)
```

由于属性*source*和*dest*引用vertexType，所以向Edge表插入数据会略微复杂一些。下面的SQL语句插入一条边，它的*weight*为10，*source*和*dest*的ID分别为1和2。

```
INSERT  INTO Edge
SELECT  10, ref(v), ref(w)
FROM    Vertex v, Vertex w
```

创建完成对应于图顶点和边的表之后，我们就可以把它们组合在一起来创建图的ADT。给定一个图$G = (V, E)$，它的2个属性*vset*和*eset*分别对应于V和E。在类型graphType中，还有一个成员函数getsuccessor，它把vertexType的实例作为参数接受，该参数被称为结点，并返回一个结点的邻接表，函数的返回类型是一

个顶点集合。我们建议读者创建一个新的ADT：vsubset，并且完成其成员函数的定义。如果在查询中要用到getSuccessor函数，则PRAGMA子句是必不可少的，它表明该函数不会改变数据库的状态。与Vertex和Edge一样，我们可以用SQL中的CREATE TABLE子句来创建对应graphType的表。

现在，就可以用如下语句向表Graph中插入数据：

```
INSERT  INTO Graph
SELECT  ref(v), ref(e)
FROM    Vertex v, Edge e
```

如果使用级联点注，并且熟悉引用信息，那么对表Graph的查询就很自然了。例如，“找出source id为1并且dest id为2的边的权重”这个查询可以用下面的SQL语句来表示：

```
SELECT  distinct(g.eset.weight)
FROM    Graph g
WHERE   g.eset.source.vid = 1 AND
        g.eset.dest.vid = 2
```

6.4 图的算法

在6.3节中，我们已经讨论了各种扩展SQL的方法，以满足图查询的特别需求，其中专门描述了SQL CONNECT操作和SQL3 RECURSIVE操作。在这一节，我们将详细描述用于实现这些操作的各种算法。

空间网络上常用的查询可以分为三类：单遍扫描查询、连接查询和网络分析查询。单遍扫描查询包括点查询和范围查询，例如：get、getEdge、get-a-successor、getSuccessors以及路径评估操作。图的连接查询是多个空间网络的图形叠加，例如，找出不相交的并且两者之间的距离在50米之内的一对高速公路和城市道路。网络分析查询是一组基于传递闭包的查询，其中包括最短路径、连通性确定、最短游历路线、定位和分配等。例如，ESRI的网络引擎支持下面的网络分析操作：分配（根据阻抗和需求给网络各部分安排资源），游历（预先指定一系列站点，确定能一次访问每个站点的最短路径），说明（产生描述一条线路的一系列说明）。大多数单遍扫描查询和连接查询可以用第5章中介绍的技术来处理，但这些技术不能用

来处理网络分析查询。在这一节，我们将重点介绍处理网络分析查询的算法，特别是介绍最短路径问题和图遍历问题的算法，因为这些算法可用于处理大多数网络分析查询，同样也用于处理单遍扫描查询和连接查询。我们首先给出一些基本算法，这些算法只适用于可以将整个空间网络放在主存中的情况。然后描述适用于大型空间网络的层次算法，这时网络不能全部放在主存中。

6.4.1 路径查询处理

路径查询处理（path-query processing）是空间网络应用中的一个重要组成部分。对导航、线路规划和交通管理的支持，本质上可以归结为：提供基于相关应用标准的路径选项（path option）。一个常用的图操作就是确定道路网中两个点A和B之间的最短路径，这里所指的“最短”可以是距离最短、旅行时间最短或者其他用户指定的约束。

路径计算可以分为三类：单对（single pair）、单源（single source）和所有对（all pairs）。

- **单对** 给定一个图$G = (V, E)$和N中的顶点u与v，找出u与v之间的最优路径。单对的一个特例就是最短路径问题。
- **单源** 给定一个源结点u，找出从u到G中所有可达结点之间的最优路径。这就是所谓的部分传递闭包（partial transitive closure）问题。
- **所有对** 在G中找出V的所有结点对u和v之间的最优路径。这是有关传递闭包的问题。

图遍历（graph traversal）算法是所有路径查询的计算基础，它沿着图的边，通过从一个结点到另一个结点的遍历来搜索路径。前面我们已经提到过，路径搜索是一个递归的操作，因而要不断把结点的邻接表从磁盘读到内存缓冲区中。所以，为了使图操作的查询处理更加快速、有效，必须对图算法进行特别的设计，以使其I/O代价达到最小。

6.4.2 图遍历算法

图遍历算法是所有路径计算算法的基础。常用的图遍历算法例子有：广度优先

（breadth-first）、深度优先（depth-first）和Dijkstra算法。下面我们简单地描述这三种算法，更详细的内容可以参见[Jiang, 1991]。

1. 广度优先搜索（BFS）

给定一个图G以及G中的一个源结点v，BFS算法访问所有从v可以到达的结点。算法首先访问源结点v的所有直接邻居。一个结点的直接邻居就是该结点的邻接表中的元素。然后算法递归地访问直接邻居的邻接表，如此循环下去。如果边关系的元组可以按其源结点的值进行物理簇集，则邻接表的大部分成员可以从同一个磁盘页面中找到，这样就可以大大降低I/O代价，并改进算法的整体效率。BFS算法的伪码在下面列出。具体的例子见图6-6，这里从结点1开始，第一次迭代访问结点(2, 4)，下一次迭代结点(3, 5)。

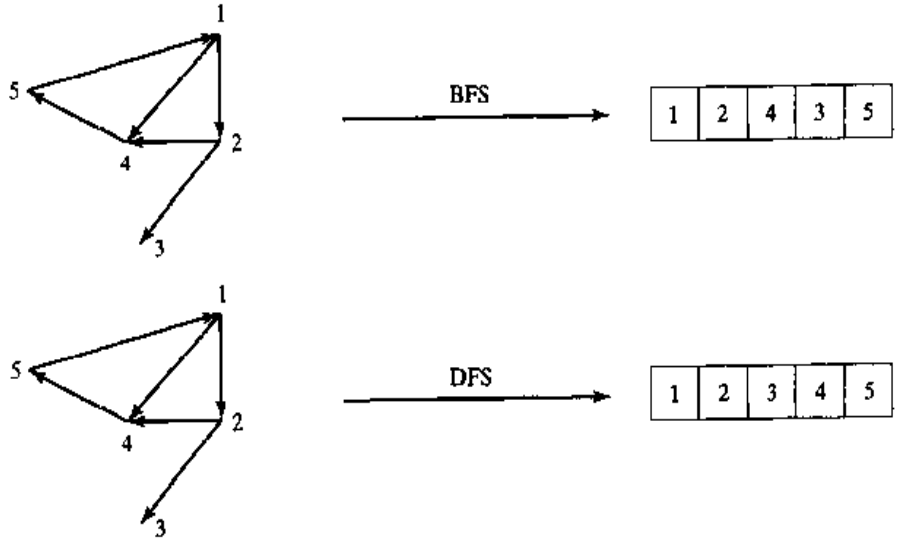


图6-6 BFS和DFS算法的结果（源结点为1）

```
procedure BFS(G, v)
{       var u, w: integer;
        visited := {v };
        if nonsink[v] then queue := v ;

        foreach w in queue do
                foreach u in adj – list_w do
                if u ∉ visited then
                    {  visited := visited ∪ { u };
                       if nonsink[u] then
                          queue := queue • u;
                    }
}
```

v是指源结点，图遍历从该结点开始。*nonsink*数组用来确保算法不会对没有直接邻居的结点进行循环处理，即这些结点是汇点(sink)。符号“•”代表插入队列操作。

2. 深度优先搜索（DFS）

与BFS算法正好相反，DFS算法先访问源结点的一个直接邻居，然后，在访问其他直接邻居之前，递归地访问其后继邻居。如此一来，DFS算法是先沿着边走完一条“路径”，然后再返回到顶层去走其他的“路径”。DFS算法的伪码如下所示。

```
procedure DFS(G, v)
        visited := {v };
        if nonsink[v] then VISIT(v) ;
}

procedure Visit(w)
{       var  u, : integer;
        foreach u in adj – list_w do
                if u ∉ visited then
                    {  visited := visited ∪ { u };
                       if  nonsink[u] then VISIT(u)
                    }
}
```

DFS的例子见图6-6，该例子从结点1开始，顺序访问结点2、3、4和5，其他访问顺序还有(1, 4, 5, 2, 3)、(1, 2, 4, 5, 3)。

3. Dijkstra算法

Dijkstra算法可以用来解决单源（部分传递闭包）问题，其伪码如下所示：

```
procedure Dijkstra(G(V, E), v);
{
        var: u, v, w integer;
        foreach u in  V do {C(v, u) = inf; C(v, v) = 0; path(v, u) := null}
        frontierSet := [v ]; exploredSet := emptySet;
        while not_empty(frontierSet) do
        {       select w from frontierSet with minimum C(v, w);
                frontierSet := frontierSet- [w]; exploredSet := exploredSet + [w];
                if (u = d) then  terminate
                else { fetch( w.adjacencyList);
                       foreach < u, C(w, u) > in w.adjacencyList
                       if C(v, u) > C(v, w) + C(w, u) then
                       {
                         C(v, u) := C(v, w) + C(w, u);
                         path(v, u) := path(v, w) + (w, u) ;
                                  if u ∉ frontierSet ∪ exploredSet then
                                  frontierSet := frontierSet + [u];
```

```
                }
            }
        }
}
```

过程的输入v是源结点。Dijkstra算法可以计算出从源结点到所有可到达结点之间的最短路径。下面列出算法的步骤。

1) 设代价$C(v, v)$等于0，其他的所有代价设为无穷大，即$C(v, u) = \inf$。$path(u, v)$设为null。

2) 设frontierSet只包含源结点v，把exploredSet初始化为空集。

3) 只要frontierSet不为空，那么执行下面的计算。

4) 从frontierSet选出一个结点w，使得从源结点v到w有最短路径。在第一次迭代时，源结点是被选出的结点，因为代价$C(v, v)$为0，而其他的代价为无穷大。

5) 把被选出的结点w从frontierSet中移除，并添加到exploredSet中。

6) 如果w的邻接表不在主存缓冲区中，那么就从辅存中读入w的邻接表。

7) 对于邻接表中的每个元素$<u, C(w, u)>$，检查它是否满足条件：$C(v, u) > C(v, w) + C(w, u)$（即存在一条从v到u并且经过w的最短路径），如果满足该条件就执行更新操作。

8) 将从源结点v到u的路径的代价更新为$C(v, u) = C(v, w) + C(w, u)$；同时，$path(v, u) = path(v, w) + path(w, u)$。

9) 如果u不在frontierSet或者exploredSet中，那么就把它加到frontierSet中。

现在我们来看图6-2a中的图。这里，边的代价表示Edge表（见图6-2d）中的距离。考虑如何确定从结点1到结点5的最短路径问题，Dijkstra算法在第一次迭代中将检查边(1,2）和(1,4)，然后把c(1,2)设为distance(1,2)，把c(1,4)设为distance(1,4)。

因为distance(1,2)<distance(1,4)，所以算法在下一个迭代时将选择结点2。第2次迭代时检查边(2,3)和(2,4)，把c(1,3)修改为distance(1,2)+distance(1,3)。由于路径(1,2),(2,4)的代价比c(1,4)大，所以不修改c(1,4)。又由于c(1,4)<c(1,3)，所以在第三次迭代时将选择结点4，在这次迭代中把c(1,5)更新为distance(1,4)+distance(4,5),也就是等于路径(1,4),(1,5)的长度。

6.4.3 单对(v,d)最短路径的Best-first算法

Best-first搜索是一个启发式框架，它通过使用领域相关的语义信息来提高算法的速度。A*是Best-first搜索算法的一个具体例子，它使用一个评估函数$f(v, d)$来低估结点v和d之间的最短路径的代价。没有评估函数的Best-first搜索算法与Dijkstra算法没有太大区别。Best-first搜索算法的伪码如下所示:

```
procedure Dijkstra(G(V, E), v, d, f);
{
        var: u, v, w integer;
        foreach u in V do {C(v, u) = inf; C(v, v) = 0; path(v, u) := null}
        frontierSet := [v]; exploredSet := emptySet;
        while not_empty(frontierSet) do
        {   select w from frontierSet with minimum (C(v, w)+ f(w, d));
            frontierSet := frontierSet- [w]; exploredSet := exploredSet + [w];
            if (u = d) then terminate
            else { fetch( w.adjacencyList);
                   foreach < u, C(w, u) > in w.adjacencyList
                   if C(v, u) > C(v, w) + C(w, u) then
                   {
                     C(v, u) := C(v, w) + C(w, u);
                     path(v, u) := path(v, w) + (w, u) ;
                                 if u ∉ frontierSet ∪ exploredSet then
                                 frontierSet := frontierSet + [u];
                   }
                 }
        }
}
```

当循环选中目的结点d为*frontierSet*的最佳结点后，过程就终止。如果从s到d的最短路径有更少的边，或者$f(v,d)$比v到d之最短路径的实际代价要小，则过程可以立即结束。在许多情况下，无需为找到最短路径而检查所有结点，而且，评估函数还能提供额外信息，将最短路径的搜索聚焦到目的结点上，减少所需检查的结点数。我们来考虑图6-2a中从结点1到结点5之间最短路径的确定问题。Best-first的A*方法采用欧氏距离作为评估函数，在第一次迭代中选出结点4用于以后的迭代，而不是选择结点2。在下一次迭代中，它找到一条到结点5的路径。

6.4.4 趋势：层次策略

层次算法把一个大的空间图分解成一个边界图和一系列分片图，这些图都比原来的图要小。对于图的查询处理，由于有些图太大，不能全部放到主存缓冲区中，

此时层次图特别有助于减少I/O代价和主存缓冲的需求量。

使用层次算法来计算最短路径的基本思想是：把原来的图分解成一系列更小的分片图和一个汇总图（叫做边界图）。通过适当地构建边界图，就可以把一个对原图的最短路径查询分解为一系列对更小图的最短路径查询。

层次图将原图分为两级表示。其中低一级的图是由原图的一系列分片组成，而高一级的图包含了边界结点（boundary node, BN），所以高级图也称为边界图（BG）。边界结点是指这样一组结点：它们在两个或两个以上分片中有一个邻居，即$N_i \in BN \Leftrightarrow \exists E_{ik}, E_{jk} \mid FRAG(k) \neq FRAG(j)$。边界图中的边称为边界边，一个分片的边界结点构成边界图中的一个团（clique），也就是说它们是完全连通的。与边界边相关的代价是穿过边界结点间之分片的最短路径的代价。每条边界边都与一个分片标识符相关联，边界路径即是穿过边界图的最短路径。

定理1 $BG.SP(s,d)\ rewrite\ [Edge\rightarrow] = G.SP(s,d)\ rewrite^{\ominus}\ [BoundaryNode \rightarrow BoundaryNode, InteriorNode \rightarrow, Edge \rightarrow]\ if\ s, d \in BG$

证明 考虑G中从s到d的一条优化路径G.SP(s,d)。从这条路径开始，重写操作删除所有的边和内部（即非边界）结点，结果路径p_g是一条由边界结点组成的路径。现在考虑一对邻接的边界结点，它们共享一个公共的分片及其内部结点，并且因为在G中从s到d的路径G.SP（s,d）是最优的，所以在该分片中，这对边界结点之间的路径也是最优的。因此，对应该分片中最短路径的边界结点之间存在边界边，这样在边界图中存在p_g。此外，由于边界图中所有边对应于相应分片中的最短路径，并且在边界图中没有比从s到d的路径p_g更短的路径，所以p_g是BG中从s到d的最短有向路径BG.SP(s,d)。这可以用经过边界图的最短路径中边界边的数目进行归纳来证明。

定理2 $p_g = (G.SP(s,d)\ rewrite\ [BoundaryNode \rightarrow BoundaryNode, InteriorNode \rightarrow, Edge \rightarrow]) \in P_{set}.where\ P_{set} = \{p_{ij}\ rewrite[Edge\rightarrow] \mid p_{ij} = BG.SP(b_i, b_j), b_i \in BoundaryNodes(Fragment(s)) b_j \in BoundaryNodes(Fragment(d))\}$

证明 考虑G中从s到d的一条最优路径。在路径中的第一个边界结点b_i和最后一个边界结点b_j之间的子路径是最优的路径。因此根据定理1我们可以得出$p_g \in P_{set}$。

⊖ 在该情况下，Rewrite[Guting, 1994b]丢弃路径中所有的内部结点和边，仅保留边界结点。

层次寻径算法

下面的算法给出了一个在层次图中寻找路径的模板。

```
//BG - 边界图
//情况1：源s和目的d都是边界结点
boundaryPath = BG.GetPath(s,d);
path = ExpandBoundaryPath(boundaryPath);
//情况2：源s是本地结点，目的d是边界结点
c = INFINITY;
for each bN in BoundaryNodes(Fragment(s)) do
      if ((cpc = Fragment(s).SPC(s, bN) + BG.(bN,d)) < c) {
            c = cpc;
            minB = i;
      }
path = Fragment(s).SP(s, minB) + ExpandBoundaryPath(BG.SP(minB, d);
//情况3：源s是边界结点，目的d是本地结点
//与情况2相似，但是源和目的相反
//情况4：源s和目的d都是本地结点
c = INFINITY;
for each sBN in BoundaryNodes(Fragment(s)) do
            for each dBN in BoundaryNodes(Fragment(d)) do
                  if ((cpc=Fragment(s).SPC(s,sBN) + BG.SPC(sBN,dBN)
                  + Fragment(d).SPC(dBN,d))< c) {
                        c = cpc;
                        minSB = sBN;
                        minDB = dBN;
            }
if (Fragment(s) == Fragment(d) && c > Fragment(s).SPC(s,d))
      path = Fragment(s).SP(s,d);
else
      path = Fragment(s).SP(s, minSB) + ExpandBoundaryPath(BG.SP(minSB, minDB))
      + Fragment(d).SP(minDB, d);
```

为了计算出优化路径，数据库将执行下列查询。

- `SP(s,d)`：这个查询返回给定图（例如G、BG或者分片图）中结点s和d之间的最短路径。
- `SPC(s,d)`：查询给定图中从源结点s到目的结点d的最短路径的代价。
- `BoundaryNodes(f)`：获得分片f的边界结点集的查询。
- `Fragment(n)`：获得内部结点n所对应的分片标识符的查询。
- `ExpandBoundaryPath(boundaryPath)`：扩展通过边界图的路径，并且通过对边界路径的各边执行`ExpandBoundaryEdge(boundaryEdge)`查询

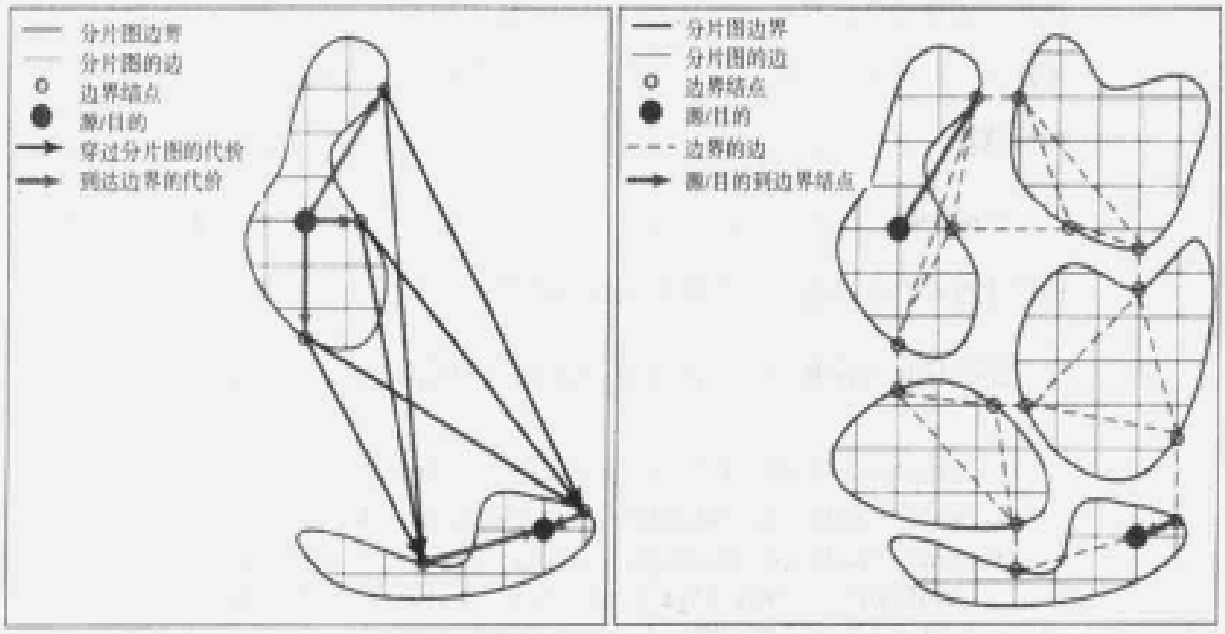
来获得G中对应的路径。

- ExpandBoundaryEdge(boundaryEdge)：扩展边界边，并且通过计算在分片中边界边的两个端点之间的最短路径来获得G中对应的路径。

层次算法包括三个步骤：在边界图中找到相关的边界结点对；计算边界路径；扩展边界路径。确定最短路径的第一步就是计算符合以下条件的边界结点，即最短路径通过这些边界结点离开源分片，然后进入目的分片。如果源和目的都是边界结点，那就不需要再计算。如果源结点是内部结点，而目的结点是边界结点，则可以用下面的方法来找出边界结点（最短路径通过它离开源分片）：首先查询分片图，以便得到从源结点到该分片中所有边界结点之间的路径的代价；然后再查询边界图，得到从源结点所属分片中的所有边界结点到目的结点之间的最短路径的代价。具有最低聚集代价的源-边界-目的确定了适合的边界结点。源结点是边界结点而目的结点是内部结点的情况与前面的相似，但是源结点和目的结点的作用与前面那种情况相反。如果源结点和目的结点都是内部结点，那么通过查询分片图，可以得到从内部结点到分片的所有边界结点之间的最短路径的代价，然后就可以获得适合的边界结点对，如图6-7a所示。下一步我们可以查询边界图，以便计算所有选定的边界结点对之间的最短路径，如图6-7b所示。具有最低聚集代价的路径确定了边界-结点对。一旦确定了适合的边界-结点对，就可以通过查询边界图来确定边界结点对之间的最短路径，如图6-8a所示。最后一步是，通过查询最短路径所要经过的分片图来扩展边界路径。边界路径上的邻接结点构成源-目的结点对，可以在此之上执行分片的最短路径查询，如图6-8b所示。

引理1 层次寻径算法可以找到从s到d之间的最优路径[Jing et at.,1998]。

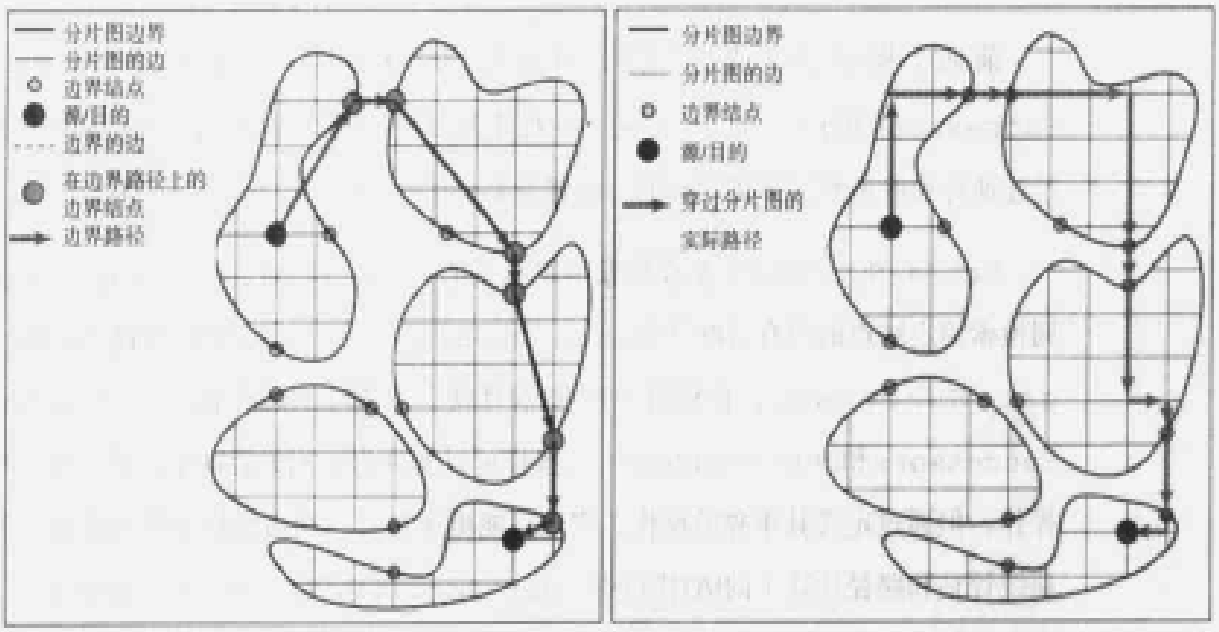
证明 最短路径(s,d)或都在分片图内，或者至少有一个是边界结点。若是前一种情况，则可以通过调用frag[i].GetPath(s,d)来获得。需要注意的是，即使源结点和目的结点在同一个分片中，两点之间的最短路径也可能离开该分片。若是后一种情况，最短路径(s,d)可以分为三部分：从s到第一个边界结点b_i的路径，从b_i到最后一个边界结点b_j的路径，以及从b_j到d的路径。根据定理2可知，从b_i到b_j的路径在P_{rt}中。



a) 第一步：找出源或目的到他们边界结点的代价

b) 第二步：用穿过分片图的代价来决定哪些边界结点一定在路径上

图6-7 寻径的例子（一）



a) 第三步：找出通过边界图的路径

b) 第四步：使用边界路径的信息找出最短路径

图6-8 寻径的例子（二）

6.5 趋势：空间网络存取方法

SQL92支持递归查询，它在网络上使用CONNECT子句，网络可以构建成有向无环图模型。为了支持传递闭包操作，在SQL3中包含了RECURSIVE子句。查询优化

器通过选择合适的算法来优化查询中的传递闭包操作。对不同类型的路径计算，优化器将考虑选择不同的算法集。在对象关系数据库和面向对象数据库中，将一个路径计算算法与一个应用查询相匹配的能力是很重要的。

在6.2节中，我们定义了一个Graph类的接口，它包括了许多重要的图操作。这些操作或*存取方法*的高效实现是快速评价任何图算法的先决条件。

路径计算查询和网络数据库的一般管理，都需要高效支持下列查询：

1. Create: <list of node records> → Network
2. get: <node-id, Network> → node properties
3. add: <node-id, node-properties, Network> → Network
   addEdge: <edge, edge-properties, Network> → Network
4. delete: <node-id, Network> → Network
   deleteEdge: <edge, edge-properties, Network> → Network
5. get-Successors: <node-id, Network> → list of <node-id, node-properties> of successors
6. get-a-Successor: <node-id, successor-id, Network> → node-properties of the successor

前四个操作不仅适用于图，也适用于许多常见的数据结构，而Get-Successors和get-a-successor操作是图应用所特有的，因为这些应用要通过连通性和遍历顺序来访问数据，而邻近关系在网络操作中起着次要的作用。

Get-a-Successor操作检索给定结点的一个后继结点，get-Successors则检索给定结点的所有后继结点。Get-a-Successor主要用于路由评估查询，而get-Successors主要用于图搜索算法，例如Dijkstra算法。尽管get-Successors和get-a-Successor操作可以实现为对相关后继结点的一系列*Find*操作，但通过定义其单独的操作，实现可能更为有效。网络的许多路径查询（包括路线评估和路径计算）的I/O代价中，get-Successors和get-a-Successor操作占了很大一部分。

为了有效实现图算法，图数据结构的常见的表格表示方法是非规范化的结点表（见图6-9）。本节后面的讨论都是基于这种表示形式的。

6.5.1 网络操作的I/O代价度量

我们在第5章已经介绍了如何计算点查询和范围查询的期望代价，并且推导出

一种独立于数据结构的度量方法来计算I/O代价，这种度量方法可用来比较和对比不同的索引和簇集方法（例如，R树和Z序）。

对于空间网络来说，光有点查询和范围查询的I/O度量方法显然是不够的。这是因为网络中的关系是受连通性而不是受接近性关系影响的。例如，在河流网例子中，Yellowstone河更接近科罗拉多河，但它却流入到密苏里河，即它与密苏里河相连接。

就网络操作的I/O代价而言，一种简单并且直观的度量方法与非分离（unsplit）边的数目有关。如果一条边的两个端点位于同一个磁盘页中，那么它是非分离的，如果一条边的两个端点位于不同磁盘页中，就说它们是分离（split）的。

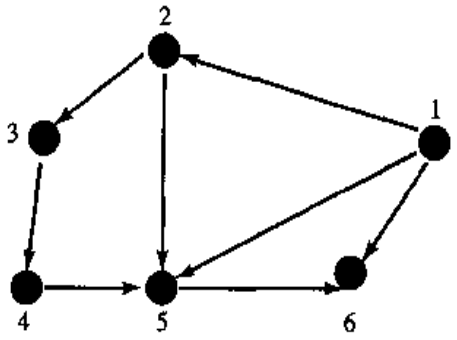
考虑图6-9所示的图以及它的结点关系和边关系。结点关系的属性包括结点ID、*x*和*y*的坐标值（未在图中给出）。边关系的属性有源、目的和权重。权重属性可以用来表示边的空间、时间和统计信息，如距离、旅行时间和边的使用频率。例如，在一个交通管理的路线评估算法中，访问主路的次数要比访问邻接辅路的次数大得多。

假定结点关系是按结点ID属性值进行物理聚集（如图6-9所示），并且一个磁盘页可以容纳node关系的两个元组，即结点1和2在同一个磁盘页中，另外有两个磁盘页分别包含结点(3,4)和结点(5,6)。这时，检索结点1的邻近结点表的操作`get-Successors(1)`就要遍历一个非分离边(1,2)和两个分离边(1,5)、(1,6)。如果我们可以簇集使非分离边的数目达到最小的结点，那么就可以减少网络操作的I/O代价。

我们来正式地定义连通性剩余率*CRR*（Connectivity Residue Ratio）：

$$CRR = \frac{\text{非分离边总数}}{\text{边总数}}$$

可以看出，最大化CRR将会最小化像`get-a-Successor()`这样的网络操作的平均I/O代价，并且也常常能减少`get-Successors()`的I/O代价。作为示例，考虑刚才提及的分页模式的CRR。对于图6-9给出的页((1,2),(3,4),(5,6))，它的CRR为3/8=0.375。如果每个页面可以容纳三个结点，那么就可以获得更高的CRR。例如，CRR((1,2,3),(4,5,6))为4/8=0.5；CRR((1,5,6),(2,3,4))为5/8=0.625。



结点

nid	x	y	后继	前驱
1	—	—	(2,5,6)	()
2	—	—	(3,5)	(1)
3	—	—	(4)	(3)
4	—	—	(5)	(3)
5	—	—	(6)	(2,1)
6	—	—	()	(1,5)

图6-9 图以及它的非规范化的表

6.5.2 减少磁盘I/O的图分区方法

在第4章中，我们介绍了图分区，这是在计算连接索引中两个关系的空间连接的语境中介绍的。图分区问题就是按照边上的代价(权重)把图结点划分到指定大小的子集中，使得所有分割边的代价总和最小。分割边（cut edge）是指两个结点位于不同分区的边。这里，子集的大小由磁盘页的尺寸来确定。

这样一来，将*CRR*最大化的一种方法就是用图分区算法来将有权重的图划分为不同分区。目前已经有各种公共领域的软件包，这些软件包可以快速地将一个非常大的图（有上百万个结点）划分为不同分区。图6-10显示了美国明尼苏达州明尼阿波利斯市的主要公路和街道。图6-11显示了分割边(虚线)和非分离边(实线)，它们位于各自的CCAM数据页中，CCAM用于表示明尼阿波利斯市的道路图。CCAM将在下一节中描述。它采用最小分割边的图分区方法。

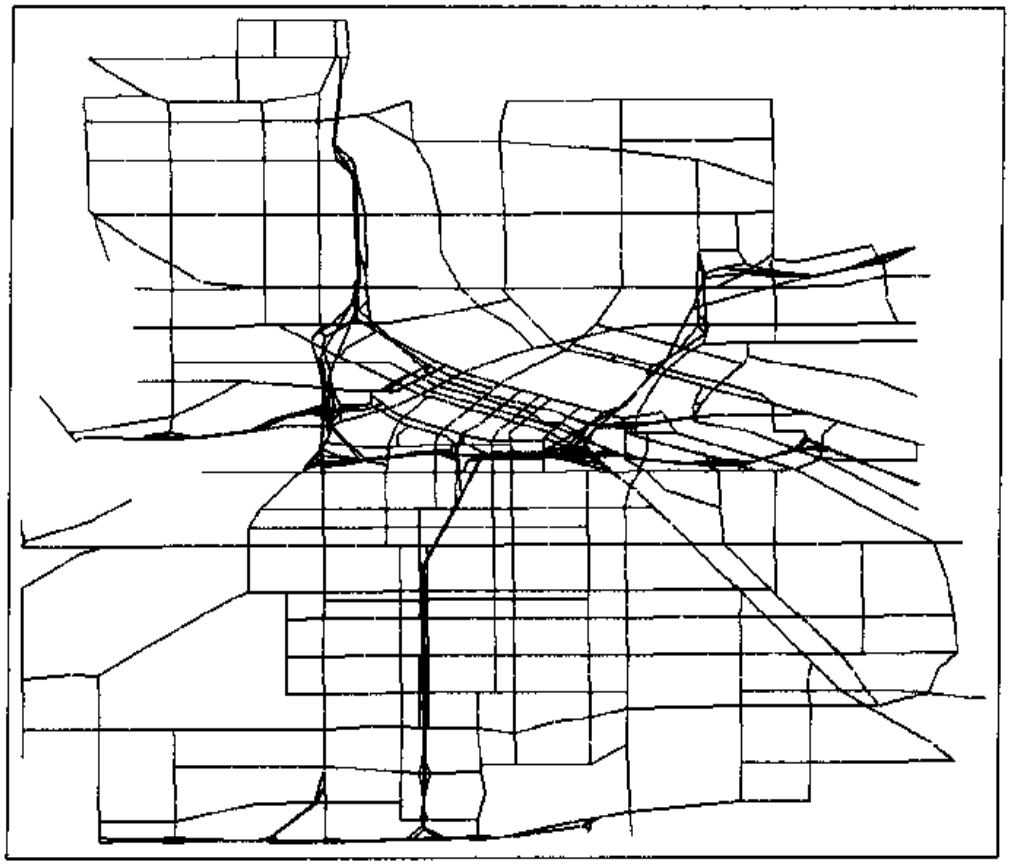


图6-10 明尼阿波利斯市的主要公路

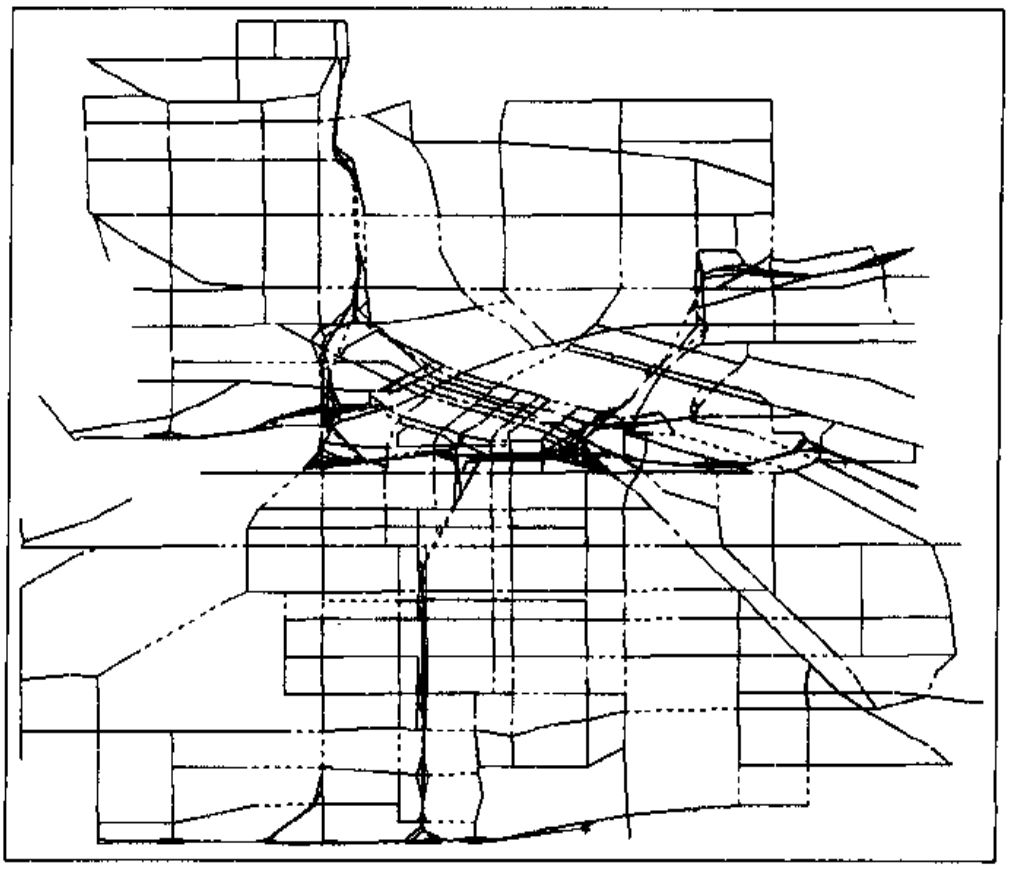


图6-11 明尼阿波利斯市的主要公路的CCAM分页中的分割边

6.5.3 CCAM: 一种连接性聚集的空间网络存取方法

CCAM通过图的分区来对网络中的结点进行簇集。此外，可以采用一种辅助的二级索引来支持Find()、get-a-Successor()和get-Successors()操作。可以针对具体的应用来选择二级索引，有些采用带Z序的B+树。为了适于应用，还可以创建其他存取方法（如R树和网格文件）作为CCAM的二级索引。在本节中，我们来描述实现各种网络操作所采用的文件结构和过程。

按连接性聚集的数据文件

对于每个结点来说，一条记录保存了结点数据、坐标值、后继列表和前驱列表。一个后继列表（前驱列表）包括一组引出（引入）边，每条边用其终止结点（起始结点）的结点ID和关联的边的代价表示。后继列表也被称为邻接表，并用于网络计算。在做Insert()和Delete()操作时，前驱列表可用于更新后继列表。结点x的邻居列表是这样一组结点，其结点ID出现在x的后继列表或前驱列表中。我们注意到，保存结点的记录没有固定的格式，这是因为后继列表和前驱列表的尺寸随结点的不同而有所变化。

CCAM通过最大化CRR的图分区方法把结点分配到数据页中，尽量让每个数据页保持至少半满的状态。数据文件的记录不是按结点ID值进行物理排序的。若不对结点进行重命名来使结点ID中包含磁盘页信息，就无法创建一个主索引，并且在更新操作会有一些额外开销。因此，可以在数据文件头上创建一个二级索引，数据文件中的每条记录对应一个索引项。

如果网络嵌入到地理空间中，则每个结点的坐标(x,y)也保存在记录中。因而，在坐标(x,y)上的空间索引方案就可以作为二级索引，这样的二级索引可以支持空间数据库上的点查询和范围查询。

例子 图6-12给出了一个示例网络和它的CCAM。图6-12的左半部是一个空间网络，其中用结点ID（一个整数）和地理坐标（一对整数）来标注各结点。为了简化这个例子，结点ID取一个整数，用来表示坐标(x,y)的Z序。例如，坐标为(1,1)的结点，其结点ID为3。连接结点的实线表示边，虚线表示将空间网络划分为数据页。如果边(u,v)的两个结点u和v位于不同的分区中，那么在该边上存在一个切割(*cut*)。

本例划分为四个分区：(0,1,4,5)、(2,3,8,9)、(6,7,12,13)和(10,11,14,15)。图6-12中的右半部是数据页以及二级索引。我们看到，通过使用图分区方法，结点被CCAM聚集到数据页中。位于相同分区中的结点保存在同一个数据页中，它们在物理上不按结点ID值排序。按结点ID排序的二级索引有助于Find()操作，本例中的二级索引是一个B+树，它基于结点坐标(x,y)的Z序。

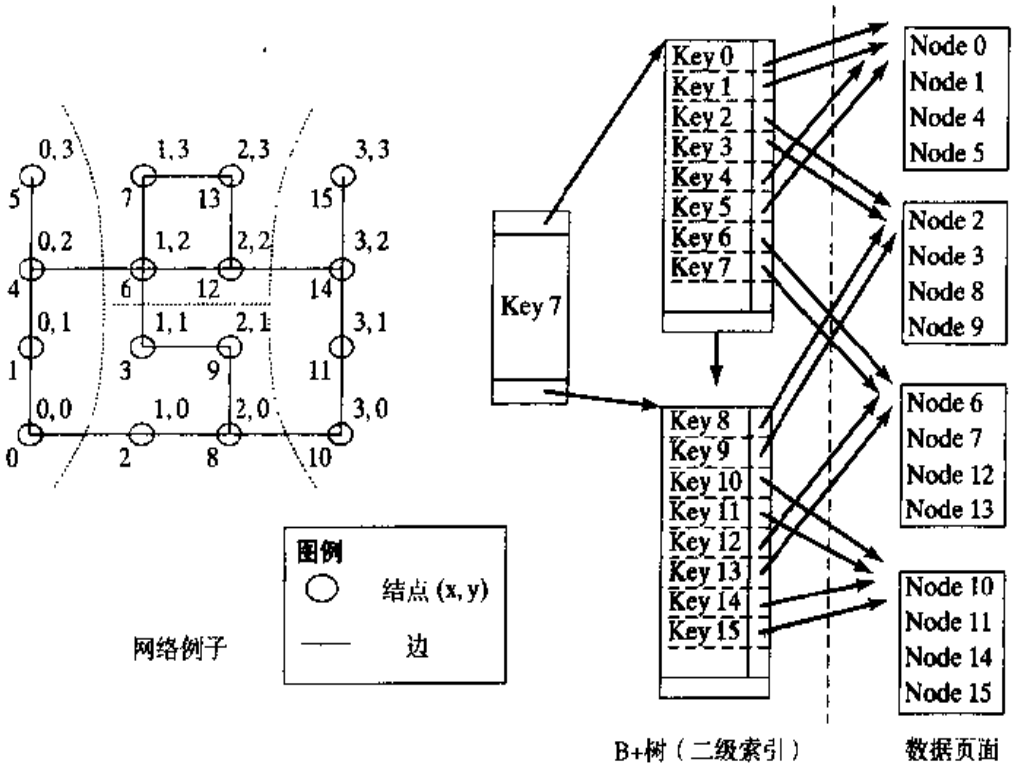


图6-12 一个示例网络的聚集和保存（key表示空间顺序）

6.6 小结

空间网络（例如道路图）是空间数据库应用中发展最快的一种。空间网络数据通常被建模为图，其结点是嵌入空间中的点。对于路径评估和最短路径计算，空间网络是通过get-a-Successor()和get-successors()操作来访问的。这些操作的高效实现通常是基于结点之间的连通性，而不是基于结点之间的欧几里得距离。

关系数据库可以用结点表和边表来表示图，可以执行简单的操作，例如get-Successor()。然而，基于关系代数的查询语言（如SQL2）对许多图操作（如最短路径）就无能为力了，这些操作要用到传递闭包。近来，数据库查询语言增加了传递闭包操作。专用的GIS产品（如ESRI网络引擎）也已支持空间网络数据和计算。

查询语言中增加的图操作影响到许多数据库组件（如查询处理策略、存储和访问方法）。传递闭包的处理可以采用BFS或者DFS策略，而计算部分传递闭包更多地是用Dijktra算法。最短路径计算还有其专用的策略（A*的，层次的）。CRR是一种与数据结构无关的度量，其最大化的存储和访问方法可以提高图操作（如get-a-Successor()）的I/O效率。专门的聚集方法CCAM采用最小分割的图分区算法来聚集结点记录，以使CRR最大化。

6.7 参考书目

6.1 对图数据模型的详细讨论见[Guting, 1994b]。在Arc/Info的NETWORK扩展中实现的网络模型可以参见[ESRI,1991]。重要图类的Java接口在[Bailey, 1998]中有详细描述。

6.2 [Mannino and Shapiro,1990]和[Biskup et al.,1990]深入讨论网络的扩展查询语言。[van der Lans,1992]详细讨论CONNECT子句；[Ullman and Widom,1999]对SQL3中RECURSIVE子句做了全面描述。

6.3 [Jing et al.,1998]和[Jiang,1991]重点讨论I/O代价的图算法。

6.4 [Shekhar and Liu,1997]描述用于空间网络数据库的CCAM存取方法。

6.8 习题

1. 欧几里得空间是空间图的一个独特方面，空间图嵌入在欧几里得图内，欧几里得图还提供了方向和度量方面的性质。因而，结点和边就有像left-of、right-of、north-of等这样一些关系。市区的交叉路口不准“左转弯”的限制就是这样一个例子。

考虑采用由结点和边组成的常规图模型为道路图建模。在模型中可以用结点代表道路交叉口，用边表示连接交叉口的路段。但遗憾的是，这时很难构建转弯限制（比如禁止左转弯）模型。寻径算法（例如，A*,Dijkstra）会考虑一个结点的所有邻接结点，却无法遵守这种转弯限制。解决该问题的方案是，给结点和边增加一些属性，并修改寻径算法使之关注这些属性。另一种方法是重新定义结点和边，以便在图的语义中包含转弯限制，而无需修改寻径算法。

建议读者重新定义结点和边，这样就不必更改图的寻径算法。

2. 研究能提供地理信息服务（如寻径）的因特网站点（比如mapquest.com），列出实现这些地理信息服务所需的数据类型和操作。

3. 基于图6-1和6-2中的表，用SQL表达下述查询：

(a) 找出Red North路线上的最后一个车站。

(b) 哪条路线把Richmond和San Leandro连接起来？

(c) 找出Downtown Berkeley和Dublin之间的车站数目。

4. 与包含多边形/线对象的空间数据库相比，空间网络有什么特别之处？

5. 我们已经了解了一些图操作（如空间网络中的传递闭包）。这些图操作是否与包含点、线、多边形对象的空间数据库相关？例如，考察GIS的再分类操作，对于给定一个海拔图，生成一个海拔超过600米的山区图。注意，山区内的邻接多边形要合并，以消除公共边界。

6. 给定一个由世界各国行政边界构成的多边形地图，完成下面工作：

(a) 对以下查询进行分类，看它是否属于传递闭包、最短路径或其他分类。

(1) 若从希腊到中国，最少要经过几个国家。

(2) 找出从陆路就能抵达美国的所有国家。

(b) 用SQL3/递归查询写出上面这些查询。

7. 在本章中，大部分讨论主要集中在空间网络的图操作上（如传递闭包、最短路径）。OGIS的拓扑和度量操作是否与空间网络相关？列出一些要点以及OGIS的拓扑和度量操作。

8. 用一个适当的图表示河流网，以便准确回答下面的查询：如果某条支流溢洪，哪些河流会受到影响。

9. 针对空间网络的概念建模，扩展第2章所讨论的象形图集。

10. 给出相似于图6-2e河流网例子的非规范化结点表。

11. 通过为Graph类增加一个Path子类，扩展空间网络的逻辑数据模型。然后，列出一些有用的操作和应用领域，再列出Graph类的其他一些有意义的子类。

12. 道路网常常带有转弯限制，例如，通常不允许从居住小区的道路左转到相交的公路上。请给出一个道路网图模型为转弯限制进行建模，标识出图中

的结点和边。思考如何计算有转弯限制的最短路径。

13. 比较和对比CCAM与R树，对于每项基本的图操作，指出CCAM与R树谁有更高效的存储方法，并解释理由。

14. 比较计算最短路径的Dijktra算法和Best-first算法。

15. 什么是最短路径问题的层次算法？该算法应该在什么情况下使用？

16. 比较图的邻接表和邻接矩阵表示法。为下列图分别创建邻接表和邻接矩阵：

    (i) 图6-4中的图G

    (ii) 图6-4中的图G*

    (iii) 图6-3中的河流网

17. 什么是图的传递闭包？为图6-2a中的图构建传递闭包。

18. 什么是CRR？对图6-12中的图，计算CCAM分页模式的CRR值。为采用以下几何分页模式的页面计算CRR：页面1包含(0, 1, 2, 3)，页面2包含(4, 5, 6, 7)，页面3包含(8, 9, 10, 11)，页面4包含(12, 13, 14, 15)。

19. 回忆本书在2.1.5节讨论的地图再分类操作，讨论在实现该操作时，是否会用到SQL3的递归和OGIS操作？

20. 比较广度优先算法、Dijktra算法和Best-first算法。在什么情况下，Dijktra算法可以简化为广度优先算法？在什么情况下，Best-first算法会简化为广度优先算法？

21. 考虑使用空间网络表示公路地图，公路的交点表示为结点，两个相邻交点之间的公路段表示为边。已知公路段的中心点坐标。另外商店表中有商店名称、地址和位置信息；顾客表中有顾客姓名、地址和位置信息。对以下网络分析问题，说明如何使用最短路径算法：

    (i) 对于一个给定的顾客，根据沿公路网的行驶距离查找最近的商店。

    (ii) 对于一个给定的商店S和一个顾客集合，查找从S出发，访问所有顾客后回到S的最低价的路径。

    (iii) 为每个商店确定在其服务范围内的顾客集合，通过利用最近商店使得为所有顾客服务的代价的总和最小。

    (iv) 为一个新商店选择合适的位置，使其能够为距离当前商店最远的顾客提供服务。

第7章

空间数据挖掘简介

数据挖掘（data mining）是一个相对较新并且迅速发展的领域，本章将对该领域相关的重要概念进行简单介绍。当然，我们将主要关注对空间数据的挖掘（mining），但这里讨论的技术适用于许多不同类型的数据集，包括时态数据库、多媒体数据库和文本数据库。

数据挖掘是发现隐藏在大型数据库中有意义的、潜在有用的信息模式（pattern）的过程。“挖掘”这一比喻意在传达这样一种概念：模式是隐藏在大型数据库中，等待人们去发现的珍贵信息宝藏（nugget）。数据挖掘作为从大量数据中获取信息的一种方法，已经在商业领域获得广泛应用，这些数据是公司多年来收集并悉心保存的。

如果说数据挖掘是从大型数据库中提取模式的技术，那么最大型的数据库就要有一个功能强大的空间组件。例如，系统地绘制（mapping）整个地球表面地图的

地球观测卫星，每天收集大约1TB的数据。其他大型空间数据库还包括美国人口普查数据库、气象数据库和气候数据库。空间数据库的挖掘需求不同于经典的关系数据库的挖掘需求，特别是空间自相关性（spatial autocorrelation）这一概念，即相似的对象趋向于在地理空间中进行聚集，它是空间数据挖掘所特有的概念。

完整的数据挖掘过程是许多子过程的组合，其中每个子过程都值得进行深入研究。重要的子过程有数据提取(data extraction)、数据清理(data cleaning)、特征选取（feature selection）、算法设计和调整以及当算法应用于特定数据时对所得到的输出结果进行分析。对于空间数据来说，尺度(scale)的问题，即以何种聚集级别进行数据分析，也是非常重要的。众所周知，在空间分析中，按不同聚集级别进行的相同实验有时会产生互相矛盾的结果。本章重点讨论数据挖掘算法的设计，特别是描述如何扩展经典的数据挖掘算法来对空间自相关这一性质进行建模。此时，理解空间数据挖掘与空间数据分析之间的区别是非常重要的。顾名思义，空间数据分析涵盖了一个较广的技术领域，它处理空间对象的空间特征和非空间特征。另一方面，空间数据挖掘技术常常源于空间统计、空间分析、机器学习和数据库技术，并被定制以便分析海量数据集。本章对新兴的空间数据挖掘领域进行介绍，这一领域通常是建立在人们熟知的空间分析和空间统计技术基础之上的。有关空间分析和空间统计更规范的介绍可参见[Bailey and Gatrell, 1995]、[Cressie, 1993]、[Fischer and Getis, 1997]、[Goodchild, 1986]及[Fotheringham and Rogerson, 1994]。

7.1节介绍数据挖掘的过程，并列举一些与数据挖掘相关的著名技术。7.2节介绍空间自相关这一重要概念，并介绍如何计算空间自相关并将它集成到经典的（classical）数据挖掘技术中。7.3节讨论分类技术并介绍PLUMS模型。7.4节讨论关联规则发现技术。7.5节讨论各种聚类技术。7.6节讨论空间孤立点检测技术。

7.1 模式发现

数据挖掘是发现隐藏在大型数据库中、潜在有意义且有用的信息模式的过程。模式可以是概括性的统计信息，如平均值、中值或数据集的标准偏差，或者是简单的规则，如“沿岸地产比内地平均要贵40%”。

有一个大家津津乐道的模式，是从一家全国性零售商的交易数据库中发现的：“在下午购买尿片的顾客一般同时也会买啤酒”，这个模式现在已经成为数据挖掘箴

言的一部分了。这是一个意料之外并且非常有趣的发现，商家利用这一发现，通过重新安排货架，实现了盈利增长。因此数据挖掘包括一组技术，用来产生假设，然后借助标准统计工具验证其有效性（validation）和正确性（verification）。例如，如果一家商店有1000种商品（中等数量），要找出其中哪两种商品是相关的（即通常“被一同购买”），需进行大约500 000次相关性检查。如果交易量很大，那么计算代价会非常昂贵。关联规则算法将过滤掉大部分的相关性检查，大量减少最后的计算。数据挖掘的目标是：使用计算机算法能够快速、自动地查找局部的（local）并且潜在地具有高实用性(high-utility)的模式。

7.1.1 数据挖掘过程

数据挖掘的整个过程如图7-1所示。在一个典型的场景中，领域专家（domain

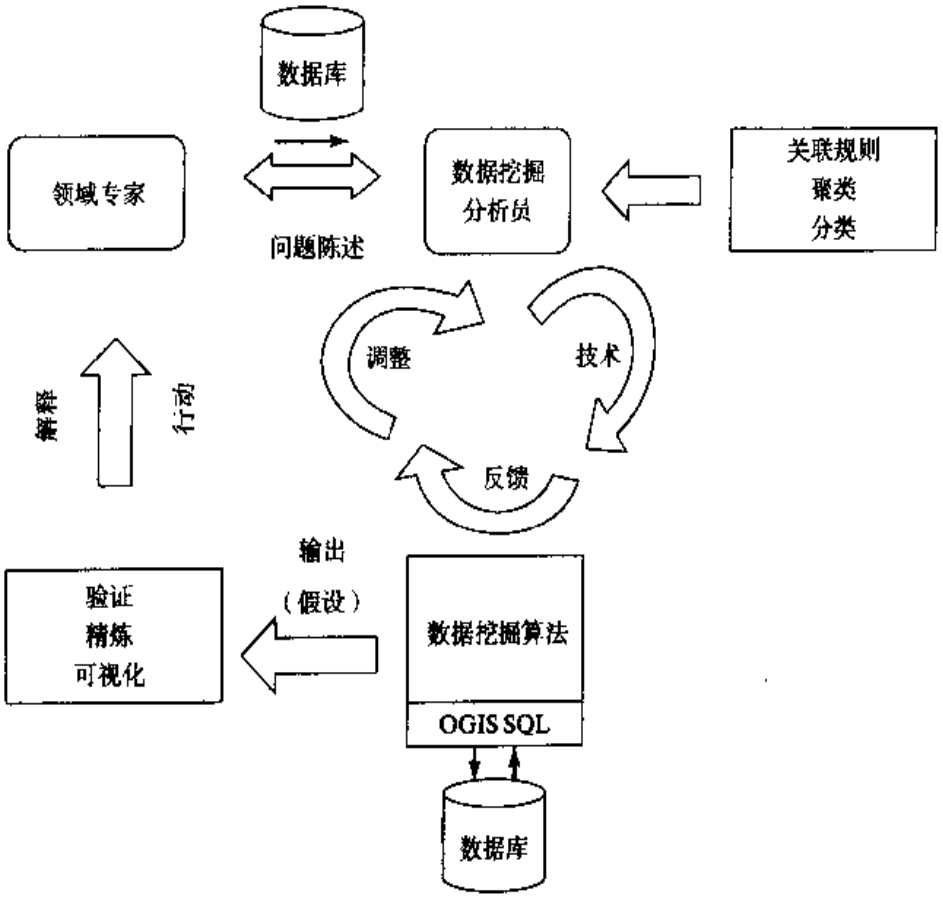


图7-1 数据挖掘过程。数据挖掘过程需要领域专家与数据挖掘分析员进行密切交互。挖掘过程的输出是一组假设（模式），这些结果可以用统计工具进行严格验证，可用GIS可视化地表现出来。最后，分析员可以解释这些模式，制定并推荐合适的方案

expert, DE）向数据挖掘分析员(data-mining analyst, DMA)咨询以解决某个特定问题。例如，一位城市执法部门的管理者可能希望弄清楚该市当年犯罪率过高原因。这位领域专家拥有数据库的访问权限，该数据库能为DMA提供解决这一特定问题的线索。通过不断迭代，领域专家和数据挖掘分析员最终在问题陈述（problem statement）上达成一致，其解决方案为最初问题提供一个令人满意的答案。

现在，数据挖掘分析员必须决定要用哪种技术或者是哪些技术的组合来解决领域专家的问题。例如，数据挖掘分析员可能觉得在分类（classification）框架下该问题可以得到最好的解决。在这种情况下，目标就是建立一个模型，把犯罪率作为其他社会经济变量的函数来预测犯罪率。一旦选定了合适的技术，就要选择一个合适的数据挖掘算法来实现这项技术。对于分类，数据挖掘分析员可以采用线性回归（linear regression）而不是决策树，因为分类属性是连续取值的。

在理想情况下，数据挖掘算法应使用SQL(对于空间数据库使用OGIS SQL)直接存取数据库数据，不过，这里有一个常见的耗时操作，即把数据库数据转换为算法要求的格式。挖掘技术以及合适算法的选择同样是一个非确定性的、反复迭代的过程。例如，算法大多要求对用户定义的参数进行调整，因为在绝大多数情况下，无法事先判定设置何种参数可以适用于特定数据库。

数据挖掘算法的典型输出是一个假设（hypothesis），其形式可以是模型参数（如在回归算法中）、规则或标号。因此，算法的输出是潜在的模式。下一步就是模式的验证、精炼和可视化。对于空间数据来说，这一步骤通常在GIS软件的帮助下完成。数据挖掘过程的最后一部分是对模式的解释，如果可能的话，应给出采取恰当动作的建议。例如，对于前面的例子，挖掘的结论可能是“高犯罪率与城市经济状况的下降直接相关”。这样，执法部门的管理者可以将结果发给相关的政府官员。或者，挖掘的结果也可能表明，某些邻近街区犯罪活动过高是导致犯罪率偏高的原因，这些街区称为“犯罪高发区”（热点区域），于是执法机构可以加强警察在这些街区的巡逻力度。

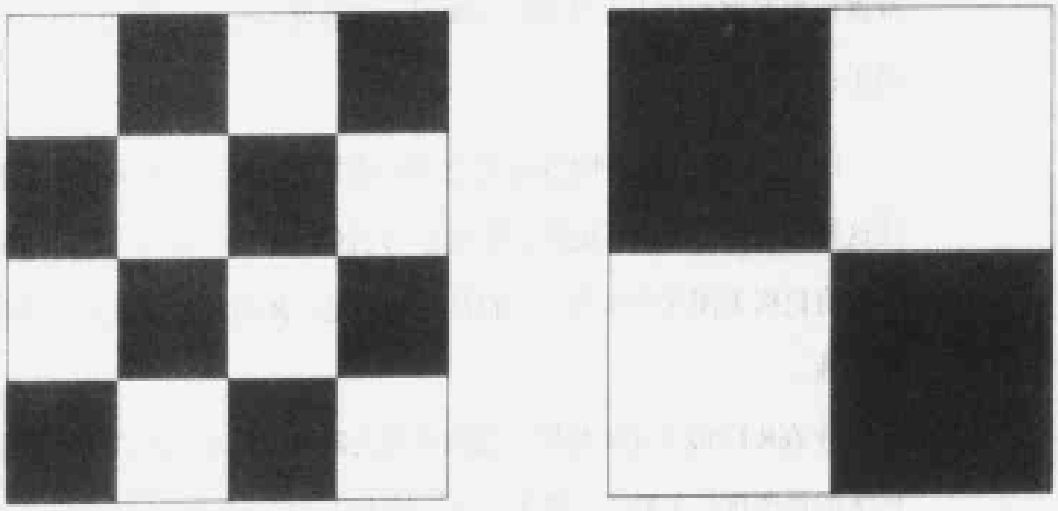
7.1.2 统计学和数据挖掘

上面所描述的整个数据挖掘过程看起来与统计学非常相似！那么，它们之间的

区别在哪儿呢？一种观点是，在应用严格的统计学工具之前，把数据挖掘作为过滤（考察性的数据分析）步骤。这个过滤步骤的作用是逐个地处理大量数据，并产生一些潜在有意义的假设，最终可以使用统计学工具对这些假设进行验证。这有点像在进行范围查询时使用R树检索MBR。R树和MBR可以提供快速过滤，搜索满足一个范围查询的潜在候选空间。不同之处是，R树可以保证查询的完备性，即不会错误地排除不应排除的候选空间。这一概念适用于某些数据挖掘技术（如关联规则），但并非全部（如聚类就不适用）。有关数据挖掘和统计学之间差别的详细讨论见[Hand, 1999]。

7.1.3 将数据挖掘作为搜索问题

数据挖掘是从大型数据库中搜索有意义（interesting）和有用（useful）模式的过程。数据挖掘算法可以搜索一个很大的模式空间，搜索的结果是候选模式，这些模式可以分为有意义或者有用的模式，或者既有意义又有用。例如，把一个4×4的图像中所有像素分为两类，黑色的和白色的（如图7-2所示），总共有2^{16}种可能组合。如果限定每个2×2的块只能归入一个类（黑色的或白色的），则组合的个数就减少到2^4个。尽管这一限定比较苛刻，但并非完全没有道理。图像中大多数的相邻像素



a) b)

图7-2 一个数据挖掘算法的搜索结果。a) 一个可能的模式，总共有2^{16}种可能模式。b) 如果限定每个2×2的块只能归入一个类，则可能的模式总数就减少到2^4个。在其他一些信息的基础上，数据挖掘算法可以快速发现“最理想的”模式

属于相同的类，这在高分辨率的图像中尤其明显。这是空间自相关性导致的结果，其他许多尺寸较大的特征也会有这样的结果。

7.1.4 空间数据挖掘的独特性

经典数据挖掘与空间数据挖掘的区别同经典统计学与空间统计学之间的区别相似。统计学分析中的一个基础假设是，采样数据应该是独立生成的，像连续抛硬币，或掷骰子。在对空间数据进行分析时，采样独立性的假设一般都不成立。实际上，空间数据趋于高度自相关。例如，具有相似特征、职业和背景的人群通常会住在相同的街区；一个地区的经济是很相似的；自然资源、野生动植物和气候的变化是随空间渐变的。事实上，相似事物在空间上聚集的性质是非常基本的，以至于地理学家将其作为地理学的第一定律：每一个事物都与其他事物相关，但邻近事物间的相关性比距离较远的事物间的相关性要大得多[Tobler, 1979]。在空间统计学中，统计学中有一个领域专门对空间数据进行分析，称为空间自相关(spatial autocorrelation)。

7.1.5 历史上著名的空间数据探测案例

空间数据挖掘是对潜在有用的模式进行自动化搜索的过程。有一些著名的空间数据挖掘案例实际上在计算机发明很久之前就发生了。[Griffith, 1999]提供了一些例子：

1) 1855年，当亚细亚霍乱席卷整个伦敦的时候，一位流行病学家在地图上标出受这种疾病袭击的所有地区，发现这些地区组成了一个簇，而簇的质心是一台水泵。在政府机构关闭这台水泵后，霍乱开始消退。随后，科学家证实这种疾病具有水源传播性。

2) 在R.Lenz（使用地图）发现所有大陆可以拼在一起成为一块陆地（就像一个巨大的拼图玩具）后，就提出了冈瓦纳大陆（Gondwanaland）理论，即所有的大陆原本是一整块陆地。随后的化石研究为这一假设提供了更多证据。

3) 1909年，一群牙科医师发现，美国科罗拉多州Springs地区居民的牙齿格外健康，他们把这归功于当地饮用水中高含量的天然氟化物。研究人员随后证实氟化物对抑制龋齿十分有效。现在美国所有的市政当局都保证在饮用水中添加氟化物。

空间数据挖掘的目标是自动发现上述这些相关性，专家可以检查这些相关性来进一步验证其有效性和正确性。

7.2 空间数据挖掘的动机

7.2.1 应用领域示例

下面介绍一个贯穿本章的例子，用来说明空间数据挖掘中的不同概念。我们已有美国俄亥俄州伊利湖岸两块沼泽地的相关数据，用来预测（predict）一种在沼泽地饲养的鸟类——红翅黑鹂 (agelaius phoeniceus)的空间分布。这两块沼泽地分别是Darr和Stubble，数据采集时间是1995年和1996年的4月到6月。

将一个均匀格网覆盖在两块沼泽地上，然后在每个单元或者像素上记录不同类型的测量值，每个像素的大小为5×5米。在每个单元上记录7种属性的值，见表7-1。当然，领域知识对于确定一个属性重要与否是非常关键的。例如，我们认为Vegetation Durability（植物韧性）比Vegetation Species（植被种类）更重要，因为关于红翅黑鹂筑巢习性的知识表明：巢穴位置的选择更大程度上依赖于植物结构以及该植物不受风和水波的影响的特性，而和植物种类无关。

表7-1 用于预测红翅黑鹂巢穴位置的习性变量

（注意，这里有6个独立变量和1个依赖变量，而且，依赖变量是二值的）

属 性	类 型	角 色	说 明
Vegetation Durability(VD)	序数	独立变量	10～100
Stem Density(SD)	数值	独立变量	茎干数/平方米
Stem Height(SH)	数值	独立变量	超出水面的高度（单位：厘米）
Distance to Open Water(DOP)	数值	独立变量	单位：米
Distance to Edge(DTE)	数值	独立变量	单位：米
Water Depth(WD)	数值	独立变量	单位：厘米
Red-winged Blackbird	二值	依赖变量	在该单元格中是否有红翅黑鹂的巢穴

我们的目的是构造一个模型来预测沼泽地中鸟巢的位置。通常使用一部分数据构造模型，这部分数据称为学习数据或者训练数据，而剩余的数据则称为测试数据，用于测试模型。例如，后面我们会说明如何用1995年Darr湿地的数据构造模型，然后用1996年Darr湿地或者1995年Stubble湿地的数据进行测试。在学习数据中，所有

的属性用于构造模型，而在测试数据中则需要隐藏一个属性（本例中是鸟巢位置）。基于从1995年Darr湿地数据中获得的知识以及测试数据中的独立属性的值，我们期望预测出1996年Darr湿地和1995年Stubble湿地中鸟巢的位置。

本章我们重点关注三个独立属性，即*vegetation durability*、*distance to open water*以及*water depth*。三个变量的显著性通过经典统计分析建立。这些变量的空间分布和1995年Darr湿地实际的鸟巢位置在图7-3中给出。图7-3b、c、d中的灰度用于表示每个属性的值，色调愈暗表示值愈大。每幅图中具有非零值的像素数目标识在图的下方。例如，图中有85个鸟巢位置和5372个可筑巢位置（像素）。从这些图中可以看出空间数据固有的两个重要性质。

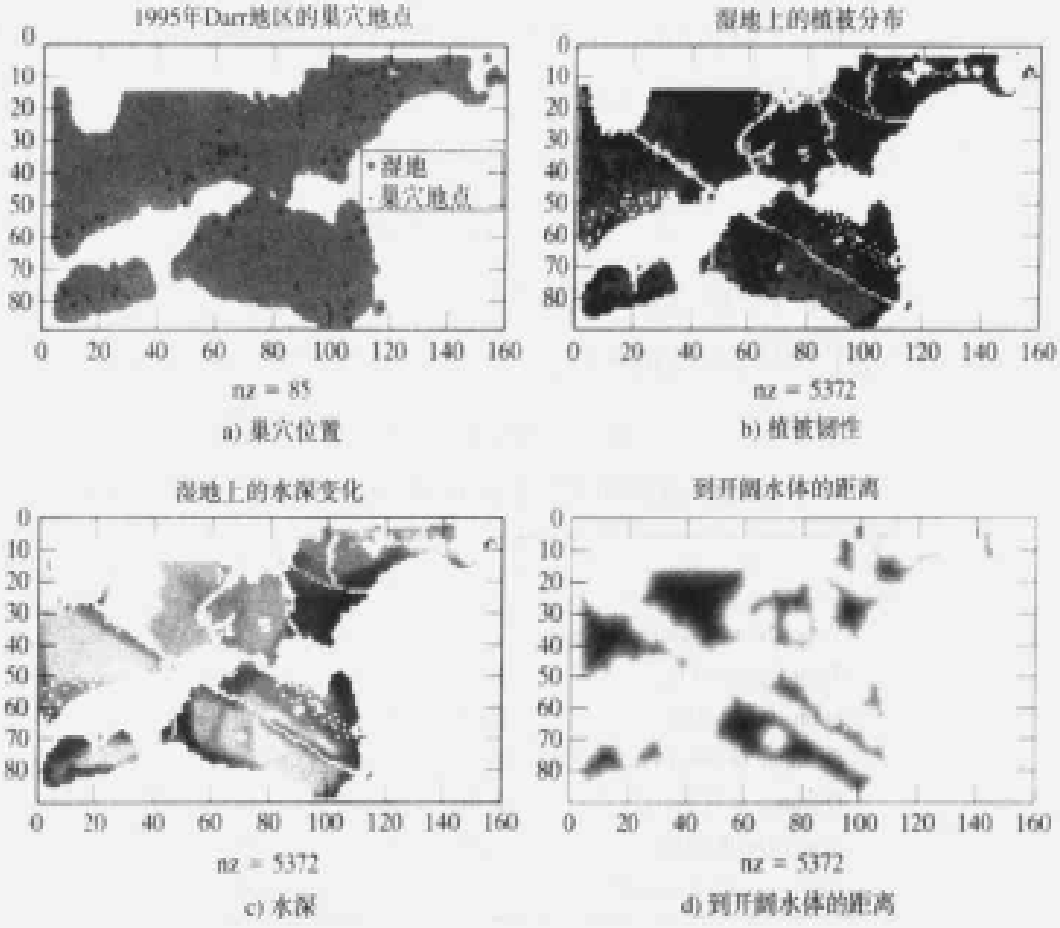


图7-3 Darr湿地，1995年。a) 学习数据集：沼泽地的几何形状和红翅黑鹂巢穴的位置。b) 植被韧性在沼泽地上的空间分布。c) 水深的空间分布。d) 到开阔水体距离的空间分布

1) 空间位置上的属性值在整个空间上呈渐变分布。尽管这是显而易见的，但是经典的数据挖掘技术明确或者隐含地假设数据是独立生成的。例如，图7-4给出独

立产生数据的情况下属性的空间分布。这种空间上的平滑性质称为空间自相关性。

2) 有时，属性的空间分布表现出与全局趋势矛盾的局部趋势，图7-3b非常生动地说明了这种情况。这里，*vegetation durability*的空间分布在湿地的西部非常粗糙，与整个湿地上植被韧性均一的总体表现形成鲜明对比。因此，空间数据不仅不是独立的，而且也不是一致分布的。

现在我们来说明如何量化空间自相关性和空间异质的概念。

7.2.2 空间形态和自相关的度量

以前提到过，空间可以视为连续的和离散的。在大多数的地球科学数据集中，空间连续性是很普遍的，但是以连续的形式表示数据通常比较困难，因为在连续空间中存在着无穷多的样本。另一方面，在离散空间中仅枚举有限数量的样本。在连续空间中，位置通过坐标来标识，而在离散空间中，位置则标识为对象。空间统计学主要用于发现地理信息。一般情况下，术语“地统计”（geostatistics）与连续空间相对应，而“空间统计学”（spatial statistics）则与离散空间对应。

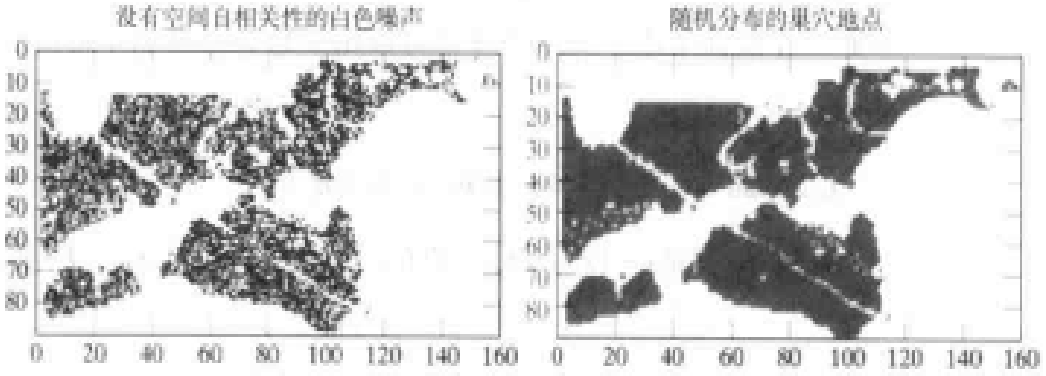


图7-4 满足经典回归的随机分布假设的空间分布

一般用中心（centrality）、散度（dispersion）和形状（shape）来刻划一个点集的空间形态。

形心（mean center）是一个平均位置，它通过计算X坐标的平均值和Y坐标的平均值得到。形心也称为一个空间分布的重心。对于一些空间应用，例如人口中心，通常用加权形心来进行度量。加权形心的计算方法如下：每个点的坐标乘以其权重

（如该区的人口数量）的总和与总权重之和的比率。中心度量的应用有几种方式，它可以用于简化复杂对象（例如，为了避免大量的存储需求和边界数字化的复杂度，一个地理对象可由其中心来表达），也可以为计划的活动标识出最有效的位置（例如，一个物流中心可以定位在中心点以便使抵达它的行程最短）。

散度（dispersion）用来度量围绕其中心分布的扩散程度，常用的散度和变化性的度量有范围、标准差、方差和协方差。用于地理分布的散度量测通常采用地理对象权值与对象之间接近度的比率的累加来计算。形状是多维的，没有一个度量可以表现形状的所有维，许多形状度量都是基于形状周长与同面积圆周长的比较而得出的。

在地理空间应用中有一种现象非常普遍：某一位置的事件经常会受到其邻近位置事件的影响。空间依赖性（spatial dependence）可以定义为“一个变量呈现出与测量的空间位置之间的距离函数有相似（或相反）取值的倾向”。空间自相关性可用于度量空间依赖性。

空间自相关性是空间采样变量经常表现出的一种性质。例如，两个互相邻近的位置的温度值非常相近。类似地，土壤肥沃程度随空间渐变，降雨量、气压等也是如此。在统计学中，有一些度量可以用来量化这种相互依赖，其中之一就是Moran's I。

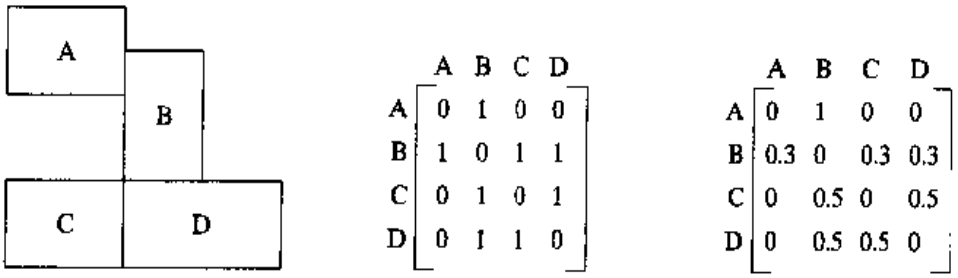
1. Moran's I: 一种空间自相关性的全局度量

给定一个变量$x = \{x_1, ..., x_n\}$，它在n个位置采样，Moran's I系数定义为：

$$I = \frac{zWz'}{zz'}$$

这里$z = \{x_1 - \bar{x}, ..., x_n - \bar{x}\}$，$\bar{x}$是$x$的均值，$W$是$n \times n$的行规范化的邻接矩阵，$z'$是$z$的转置。图7-5给出了一个空间网格、其邻接矩阵和行规范化的邻接矩阵的例子。图7-5b 和7-5c中标识为C的行在B列和D列有两个非零项，表示图7-5a中B和D都与C相邻。

注意，Moran's I系数不但依赖于变量x的不同值，而且依赖于它们的排列方式。例如，图7-6所示的两个3 × 3图的Moran's I系数是不同的，即使其像素值的集合相同也是如此。读者可以试着计算它的Moran's I系数来确认这一点。表7-2给出了4邻域关系和8邻域关系的Moran's I系数。



a) 地图　　b) 布尔值W　　c) 行规范化W

图7-5 a) 空间网格。b) 邻接矩阵。c) 行规范化的邻接矩阵

表7-2 预测红翅黑鹂巢穴位置的解释性变量的Moran's *I* 系数

解释性变量	4邻域	8邻域
到岸边的距离	0.7606	0.9032
到开阔水体的距离	0.7342	0.8022
水深	0.6476	0.7408
基干高度	0.7742	0.8149
基干密度	0.6267	0.7653
植被韧性	0.3322	0.4851

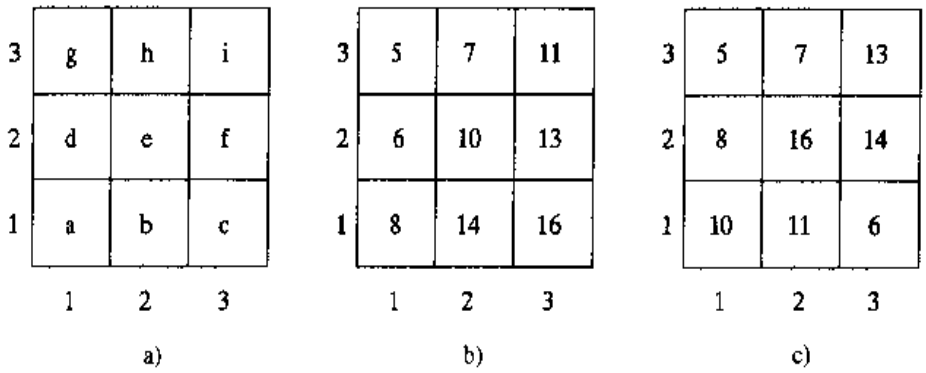


图7-6 Moran's *I*系数。图像b和c的像素值集合相等，但它们拥有不同的Moran's *I* 系数

2. 空间自相关性的局部指标

随着高分辨率图像数据的普及和GPS设备越来越多地用于标记野外采样位置，空间自相关性存在的事实也不断地凸现出来。因而，空间统计学家经常使用空间自相关性的局部度量来追踪空间依赖性如何在同一空间层的不同区域发生变化。局部自相关性在不同位置有较大变化预示着空间异质的存在，这一点可以从图7-3b的植被韧性图层中看出。在位置i的局部Moran's I度量定义为：

$$I_i = \frac{z_i}{s^2}\sum_j \frac{W_{ij}}{z_j}, i \neq j$$

其中 $z_i = x_i - \bar{x}$，s是x的标准差。例如，表7-3中Moran's I系数为：

$$I_{75} = \frac{(75-55.82)}{675.32}(71+85+61+63-4(55.82)) = 1.6109$$

表7-3 16×16灰度图像

40	41	39	44	52	64	74	67	63	63	57	47	48	59	62	50
48	50	51	45	78	82	92	109	115	98	83	74	70	76	95	70
40	40	41	46	86	92	79	97	123	107	115	110	101	83	78	56
40	39	38	47	74	103	82	89	94	91	115	121	113	104	88	56
45	44	46	51	82	98	74	72	59	71	83	83	83	103	106	64
50	43	44	44	67	88	74	59	45	85	107	88	70	97	115	75
48	40	41	41	71	85	98	82	51	86	118	91	66	86	100	78
48	45	40	47	98	95	89	71	52	81	110	71	45	46	54	53
52	48	56	61	103	91	85	75	63	72	94	57	37	35	36	39
48	48	79	78	40	45	51	61	64	58	58	48	53	47	40	43
37	45	47	41	26	25	28	29	31	33	35	37	56	57	46	47
27	28	29	28	27	26	27	29	28	28	29	32	44	45	40	47
29	26	24	27	29	28	27	27	27	28	28	34	41	38	38	47
28	27	25	27	27	27	26	27	27	28	27	38	53	47	36	48
25	27	26	25	28	34	31	27	27	28	28	34	45	48	38	48
25	26	27	28	34	39	32	29	27	29	28	31	37	41	41	47

7.2.3 空间统计模型

统计模型通常用于表示随机变量的观测值。随后可以用这些模型来根据概率理论进行估计、描述和预测。

1. 点过程

点过程是用于点模式中点的空间分布的一种模型。一些自然过程可以建模为空间点模式。森林中树木的位置，城市中加油站的位置都是点模式的例子。一个空间点过程的定义为：$Z(t) = 1; \forall t \in T$ 或者 $Z(A)=N(A), A \subset T$，其中$Z(.)$和T都是随机的。这里T是索引集（$T \subset \Re^d$），$Z(.)$是空间过程。从广义上说，空间点过程可以分为随机过程和非随机过程。实际的点模式经常与随机模式（由泊松过程产生）比较，比较时采用空间点到其最近邻点之间的平均距离。对于一个随机模式来说，平均距离的期望值为 $\frac{1}{2*\sqrt{\text{密度}}}$，其中“密度”是单位面积内的平均点数。对于一个实际的过程来说，如果该距离落入特定区间，那么可以推断出该模式由随机过程产生，否则由非随机过程产生。由非随机过程产生的模式可以是聚类的（聚集模式），也可以是均匀排列的（规则模式或非聚类模式）。

2. 格

一个格D可以记为$Z(s)$：$s \in D$，其中，索引集D是观测数据的空间位置上的一个可数集。这里，格指的是规则的或者不规则的空间位置的可数集合。定义在人口普查区上的人口普查数据是格的一个例子。一些空间分析函数（例如，应用于格的空间依赖性、空间自回归、马可夫随机场）可以用于格模型。

3. 地统计学

地统计学处理空间连续性的分析，空间连续是空间数据集固有的特征。地统计提供一组统计工具来构建空间变化模型和未采样位置的属性插值（预测）模型。空间变化可以使用变量图（variogram）进行分析。空间自相关性的数量和组成都可以通过变量图进行描述，变量图概括度量值对的差与相应点对距离之间的关系。通过采样点的已知值，空间插值（预测）技术就可以用来估计非采样点的值，这里假设问题域缺乏地理稳定性，而且不考虑全局模型。Kriging是一个地统计中众所周知的估计过程，它使用已知值（在采样位置）和一个半变量图（通过数据估计得到）来确定未知的值。比起传统的用于空间数据集的插值方法（如加权平均、最近邻域），Kriging方法提供更好的估计，因为它考虑了空间自相关性。

7.2.4 数据挖掘的三位一体

数据挖掘是一个多学科的交叉领域，有许多从数据中抽取模式的新颖方法。然而，如果要给数据挖掘技术贴上标签（label），那么最无争议的三个是分类（classification）、聚类（clustering）和关联规则（association rule）。在详细描述这些技术之前，先给出一些典型的应用这些技术的例子。

1. 位置预测和专题分类

分类的目标是基于一个关系的其他属性值来估计该关系的某个属性的值。许多问题都可以表达为分类问题。例如，基于其他属性值（*vegetation durability*、*water depth*）来确定一片湿地中鸟巢的位置就是个分类问题，有时候也称为位置预测（location prediction）问题。类似地，预测可能的犯罪活动频繁发生的区域也可以归为位置预测问题。零售商为超市选址实质上就是求解一个位置预测问题的过程。房地产业界众所周知的“位置就是一切”口号就是该问题的真实体现。

在专题分类中，其目标是在卫星接收器所记录下的“光谱信号”值的基础上，将卫星图像的像素分成不同类别（如水体、城市、农村、森林）。专题分类的问题在于它具有很强的空间连接性，因为在多数情况下，图像中相邻的像素属于同一个类。因此，如果像素尺寸小于空间特性的尺寸的话，卫星图像很自然地表现出高度的空间自相关性。

2. 确定属性之间的相互作用

对于大量、连续产生并存储在数据库中的数据而言，快速的模式探测是数据挖掘的动机之一。最简单、可能也是最著名的数据挖掘技术之一是发现一个关系中属性内部之间联系。例如，在超市数据分析语境中，形如$X \rightarrow Y$的模式意味着那些购买商品X的人们也很有可能购买商品Y。在空间数据库语境中，我们有这样的规则：*is_close(house, beach)→is_expensive(house)*，即靠近海滨的房屋价格会比较昂贵。在鸟类的生活习性的语境中，我们会获得这样的规则：*low vegtation durability→high stem density*。在7.4.1节中，我们将讨论Apriori这种最常用的关联规则发现算法。

由于计算效率的原因，对于更严格的统计相关性分析，通常使用关联和关联规则选择特征的候选子集。

3. 热点标识：聚类和孤立点

我们在7.1.1节曾经提到，执法部门采用热点分析来确定辖区内犯罪率异常高的区域。他们通过记录每次犯罪的地点，然后使用孤立点检测和聚类技术来确定犯罪密度较高的区域。孤立点检测和聚类也可以用于确定鸟巢位置的热点和癌症的疾病聚类。

另一个使用空间点聚类的实际例子是确定服务站点的位置。例如，假定一家轿车公司拥有其所有客户的地理位置信息，并且想要开设新的服务中心来满足其客户的专门需要，那么可以使用聚类方法来确定服务中心的最佳位置。

聚类是非监督学习的例子，即事先没有标号内容或标号个数的知识。因此，聚类算法就必须“更加努力”地工作来确定可能的聚类。后面我们会讨论两种聚类方法。K-medoid是一种确定性（deterministic）聚类算法，其中每条记录只放在一个聚类中。而另一方面，或然性（probabilistic）聚类则指出每条记录属于任一聚类的

概率。

孤立点检测的目标通常是发现数据点的一个“小”子集，这些数据点常被视为噪声、错误、偏差或异常。通俗地说，孤立点是一些与其余数据集显得不一致的观测点。标识孤立点可能会发现意料之外的知识，它有大量的实际应用领域，如信用卡欺诈、运动员表现分析、投票舞弊、破产倒闭、天气预测及热点检测等。

以最简单形式讲，热点是研究空间中与空间总体行为相比较显得非常突出的区域。因此，可以通过可视地检测数据在地图上的分布或者通过阈值检测标识出热点。例如，当区域的属性值（如犯罪率）与平均值的标准差至少达到两倍时，这些区域就可以标识为热点。从空间自相关性的角度来看，热点位置具有高局部空间自相关性。

以下三节的内容介绍数据挖掘中三种主要的方法，即分类、关联规则、聚类和孤立点检测。

7.3 分类技术

简单地讲，分类就是找到一个函数：

$$f: D \rightarrow L$$

其中，f 的域 D 是属性数据的空间，L 是标号的集合。例如，在我们所举的鸟类生活习性领域例子中，D 是三维空间，由*vegetation durability*、*water-depth*和*distance to open water*组成。集合 L 由两个标号组成：有巢和无巢。分类问题的目标是根据给定的有限子集$Train \subset D \times L$来确定合适的函数 f。分类的成功与否由函数 f 应用到测试数据集*test*的准确度来判定，测试数据集*test*与训练数据集*train*不相交。分类问题是预测性建模（predictive modeling），因为只有在来自集合D的数据给定时，f 才可用于预测标号 L。

有很多技术可用于解决分类问题。例如，在最大或然率分类中，目标是要完全指定联合概率分布$P(D, L)$，这通常是应用贝叶斯定理来完成，遥感分类就选用这种方法。在商务世界中，决策树分类器（decision-tree classifier）因其易于使用而广为采用。决策树分类器将属性空间（这里是D）划分为区域，然后为每个区域分配一个标号。神经网络（neural network）通过计算具有非线性边界的区域来概化决

策树分类器。另外一种常用方法是使用回归分析（regression analysis）方程来构建D和L之间相互作用的模型，例如，采用线性方程$y = mx + c$在线性回归分析中建立类边界的模型。

本章重点介绍扩展传统数据挖掘技术，融入空间自相关性这个空间数据的与众不同的关键特性。我们采用线性回归作为原型，说明如何扩展分类方法对空间自相关性进行建模。之所以选择线性回归分析来阐述空间分类，是因为这种方法广为人知，并且空间回归是空间统计学领域中研究最深入的空间分类方法。也可以扩展其他的分类技术（如神经网络、决策树、规则）。

7.3.1 线性回归

当类变量是实数值时，计算条件期望比计算条件概率更适合。因此，分类的目标是要计算

$$E[C|A_1, ..., A_n]$$

还可以使用一种更熟悉的记法，用Y代替C，X_i代替A_i，并且假设所有的属性由标准随机变量同一地（identically）和独立地（independently）产生，则线性回归方程为：

$$E[Y|\mathbf{X} = \mathbf{x}] = f(\alpha + \beta\mathbf{x})$$

这里，$\mathbf{X} = (X_1, ..., X_n)$。该表达式等价于更常见的表达式$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$。训练数据可以用于计算参数向量$\beta$，向量$\beta$则可以用于计算测试数据集中类属性的值。函数$f$将$\alpha + \beta\mathbf{x}$映射为0到1区间上的一个数，以返回概率。通常$f$采用一个对数函数（如分对数、概率）。

7.3.2 空间回归

如前所示，当变量在空间中引用时，它们就会表现出空间自相关性。因此，上述对随机变量的一致独立的分布（i.i.d.）的假设就不适用于空间数据语境了。空间统计学家已经提出很多方法来扩展考虑了空间自相关性的回归技术。最简单也最直观的方法是借助邻接矩阵W修改回归方程。因此，空间自回归的回归（SAR）方程为：

$$\mathbf{Y} = \rho W\mathbf{Y} + \mathbf{X}\beta + \varepsilon$$

SAR方程的求解过程无疑要比经典回归方程复杂得多，因为方程右边有了ρWY项。还要注意，W矩阵的大小与数据样本大小的平方成正比。幸运的是，W矩阵中非零的项极少，可以充分利用这个事实，使用稀疏矩阵技术来加速求解过程。

7.3.3 模型评估

我们已经讨论过两种通用的分类问题求解模型，它们分别是线性回归和SAR。现在我们介绍评估模型性能的标准方法，并解释这种评估标准方法为什么不适于空间数据挖掘情况。

对于像nest（有巢）或no-nest（无巢）这两种分类问题来说，可能会有四种结果出现。例如，nest被正确预测到的情况称为true-positive（TP，真-正）；模型预测到有巢而实际为no-nest的情况称为false-positive（FP，假-正）。类似地，no-nest被正确预测到的情况称为true-negative（TN，真-负）；而no-nest被预测为nest则称为false-negative（FN，假-负）。图7-7显示这四种组合。

分类的目标是基于其他属性的值来预测特定属性的条件概率。因此，分类技术的输出结果是概率。将概率值转换为实际类标号的方法是选择一个截止概率（cut-off probability）b，然后将所有那些预测概率值大于截止概率b的记录标记为一个类标号，例如将一个记录标记为nest，而剩余的记录则标记为no-nest。通过改变截止概率b，我们可以得出两个不同分类器彼此相对的运转状况的有效评估。因此，给定一个截止概率b，对应的真-正率（TPR(b)）和假-正率（FPR(b)）的定义为：

$$TPR(b) = \frac{TP(b)}{TP(b) + FN(b)}$$

$$FPR(b) = \frac{FP(b)}{FP(b) + TN(b)}$$

如果将待考虑的两个分类器的TPR与FPR绘制为曲线，则曲线在对角线$TPR=FPR$之上的分类器对于该特定的数据集来说是更合适的模型。这些曲线称为接收器运行特征（receiver operating characteristics, ROC）曲线。我们在1995年

Darr训练数据集和1995年Stubble测试集上比较过经典回归和SAR模型。图7-8的结果清楚地说明通过包括空间自相关性ρWY项，回归模型在学习和预测能力上有很大提高。

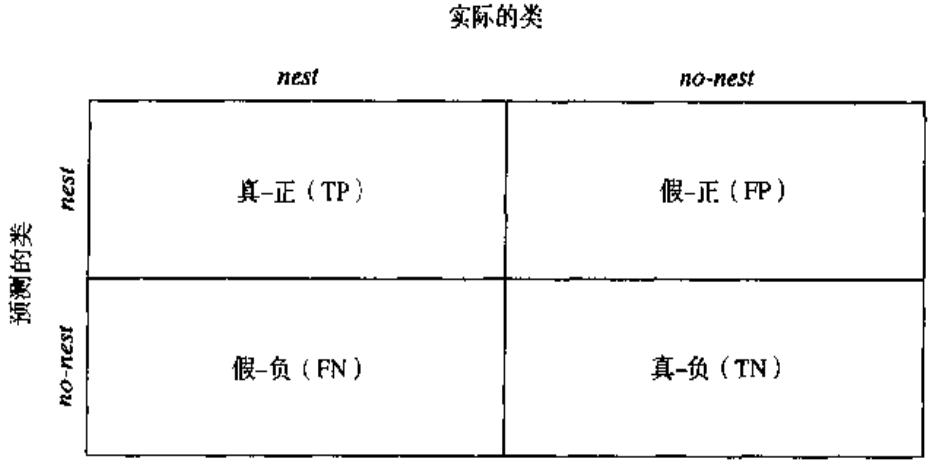


图7-7 两类预测问题的四种可能输出结果

上面提到的模型评估技术在空间数据语境下尚不完善。考虑图7-9所示的例子，这里的目标是预测标记为A的位置。ROC曲线不能区分预测7-9c所示位置的模型与预测7-9d所示位置的模型，即使图7-9d中的预测位置比图7-9c的预测更接近于实际的位置。根据观测到的这种情况，人们可以设计一种新的框架来求解位置预测问题中的两类空间分类问题，下一节我们会阐述这种框架。

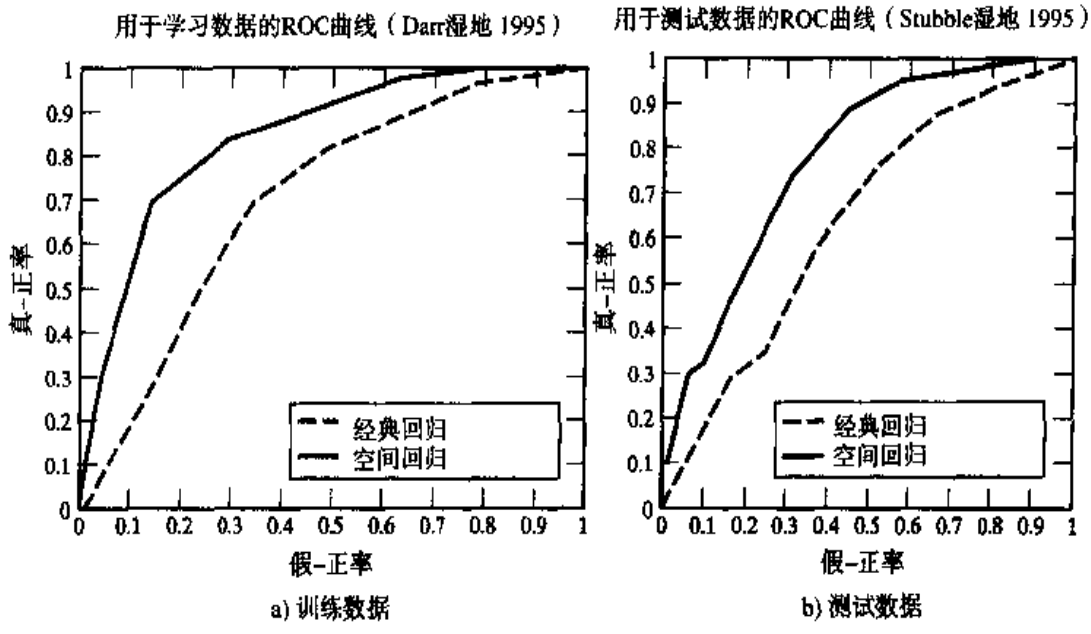


图7-8 ROC曲线。a) 利用1995年Darr湿地数据建立的经典模型与SAR模型的ROC曲线的比较。b) 利用1995年Stubble湿地数据构建的两种模型的比较

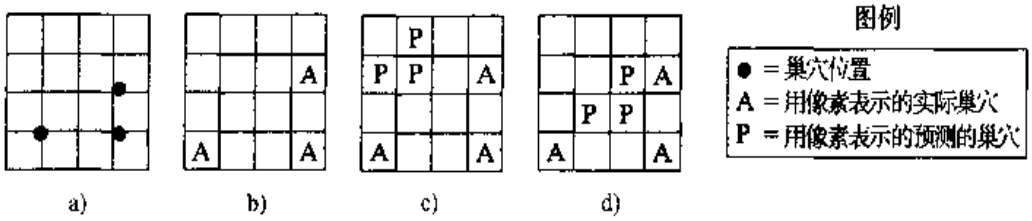


图7-9 空间数据情况下ROC曲线的问题。a) 鸟巢的实际位置。b) 实际有鸟巢的像素。c) 通过模型预测到的位置。d) 用另一种模型预测到的位置。

预测d在空间上比c更精确，经典的分类

准确性度量无法捕获这种差别

7.3.4 采用图相似度预测位置

位置预测问题是鸟巢位置预测问题的概化，它能够从其他领域（包括犯罪预防和环境管理）的类似问题中抽取出本质特征。这个问题的形式化定义如下。

给定：

1) 基础地理空间G的空间框架S由一组位置$\{s_1, ..., s_n\}$组成。

2) 一个解释性函数集合f_{x_k}：$S \rightarrow R^k$，$k = 1, ..., K$，其中R^k是解释性函数的可能取值范围。

3) 一个依赖类变量f_C：$S \rightarrow C = c_1, ..., c_M$

4) 一个用于参数α的值，表示空间准确度的相对重要性。

寻找：分类模型：$\hat{f}_C : R^1 \times ... R^k \rightarrow C$。

目标：最大相似度$(map_{s_i \in S}(\hat{f}_C(f_{x_1}, ..., f_{x_k})), map(f_C))$

$$= (1-\alpha)\text{classification\_accuracy}(\hat{f}_C, f_C)$$

$$+ (\alpha)\text{spatial\_accuracy}(\hat{f}_C, f_C)$$

约束：

1) 地理空间S是一个多维的欧氏空间[⊖]。

2) 解释性函数$f_{x_1}, ..., f_{x_k}$和依赖类变量f_C的值可能并不独立于邻近空间位置的对应值（即存在空间自相关性）。

⊖ 整个地球表面不能建模为一个欧氏空间，但就某个局部来说是近似可以的。

3) 解释性函数的域R^t是一维的实数域。

4) 依赖变量的域$C = \{0, 1\}$。

上面的公式强调位置预测的两个重要方面。它明确地表明：1) 数据样本可能表现出空间自相关性；2) 目标函数（即，图相似性度量）是分类准确度与空间准确度的结合。依赖变量f_c与预测变量$\hat{f}_c$之间的相似度是“传统分类”准确度同依赖表示方法的“空间分类”准确度的结合。调整项α控制着**空间准确度**的重要程度，并通常依赖于具体领域。当$\alpha \to 0$时，图相似性度量接近于传统的分类准确度度量。从直观上看，α捕捉空间数据所表现出的空间自相关性。

7.3.5 马可夫随机场

基于贝叶斯分类器的马可夫随机场（Markov Random Field, MRF）用MRF和贝叶斯规则来估计分类模型$\hat{f}_c$。相互依赖关系由一个无向图（即一个对称的邻域矩阵）来表示的一组随机变量称为马可夫随机场。马可夫性质是指一个变量只依赖于其邻域，并且独立于其余所有的变量。位置预测问题可以在这样的框架下建模，它假设不同位置s_i的类标号$l_i = f_c(s_i)$组成一个马可夫随机场。换句话说，如果$W(s_i, s_j) = 0$，则随机变量l_i和l_j相互独立。

贝叶斯规则可以利用特征值向量X和邻域类标号向量L_i预测l_i，如下所示：

$$Pr(l_i \mid X,\ L_i) = \frac{Pr(X \mid l_i,\ L_i) Pr(l_i \mid L_i)}{Pr(X)} \tag{7.1}$$

$L_i = L \setminus l(s_i)$表示位置s_i邻域内类标号的集合，但不包括s_i处的类标号。

一个求解过程可以利用训练数据通过检查类标记的频数与空间框架中位置总数的比率来计算$Pr(l_i \mid L_i)$，L_i表示s_i邻域内类标号集合（但不包括s_i处的类标号）。$Pr(X \mid l_i,\ L_i)$可以使用训练数据集中观测值的核函数来估算。由于要估算更为复杂的分布，所以为了保证估算的可靠性，相对于非空间环境中贝叶斯分类器所需的数据集，这里需要更大的训练数据集。如果可用的训练数据集不够大，那么关于$Pr(X \mid l_i,\ L_i)$的假设就非常有用。通常假设特定位置周围所有邻居的影响是均一的。为了保证计算效率，可以假定只有局部解释性数据$X(s_i)$以及邻域标号集L_i与预测类标号$l_i = f_c(s_i)$有关。通常还假设邻居之间的所有相互作用可以通过类标号变量内的相互作用来表

达。许多领域也采用特定的参数概率分布形式，从而导出更简单的求解过程。此外，根据Hammersley-Clifford定理，通过局部定义的MRF限定吉布斯分布（Gibbs distribution），这经常会使问题求解变得更容易。

SAR和MRF贝叶斯分类器都可以构建空间语境下的模型，因而被各个研究团体用在与空间数据集有关的分类问题上。现在，我们在概率框架下比较这两种建模空间语境的方法。

概率框架下的SAR与MRF比较

本节中我们用简单的概率框架来比较SAR和MRF。假设类 $l_i \in (c_1, c_2, ..., c_M)$ 是离散的，并且对于位置 s_i 的类标号估算 $\hat{f}_{c_i}(s_i)$ 是一个随机变量。另外，在未具体指定生成模型时，我们假设特征值(X)是常量。假设SAR的模型参数是常数（例如，β 是常数向量，ρ 是常数）。最后，假设空间框架是规则的网格。

首先要注意，基本的SAR模型可以重写为如下形式：

$$y = X\beta + \rho Wy + \varepsilon$$

$$(I - \rho W)y = X\beta + \varepsilon$$

$$y = (I - \rho W)^{-1}X\beta + (I - \rho W)^{-1}\varepsilon = (QX)\beta + Q\varepsilon \tag{7.2}$$

对于我们建模的特定问题而言，$Q = (I-\rho W)^{-1}$、β和ρ是常量。变换特征向量X为QX的结果可以视为一个空间平滑操作。就特征空间变换来看，SAR模型类似于线性对数模型。换句话说，SAR模型假设在特征空间变换中的类是线性可分的。

图7-10给出两组数据集盐和辣椒的特征值空间分布。在该特征上定义了c_1和c_2两个类。接近2的特征值映射到c_2，而接近1和3的特征值映射到c_1。在原始的特征空间中，这些类不是线性可分的。局部空间平滑可以消除特征值中盐和辣椒的空间模式，以进行特征值分布的变换。在图7-10的上半部，很少有3这个值，由于其邻居多数是1，因此平滑地将它们修改为接近1。SAR在这组数据上执行得很好，因为在变换空间中类是线性可分的。然而，图7-10下半部分给出了一组不同的空间数据集，这里局部平滑不能使类线性可分。假如$Q = (I-\rho W)^{-1}$未能使类线性可分，那么即便在变换后的特征空间中，线性分类器也不能划分这些类。

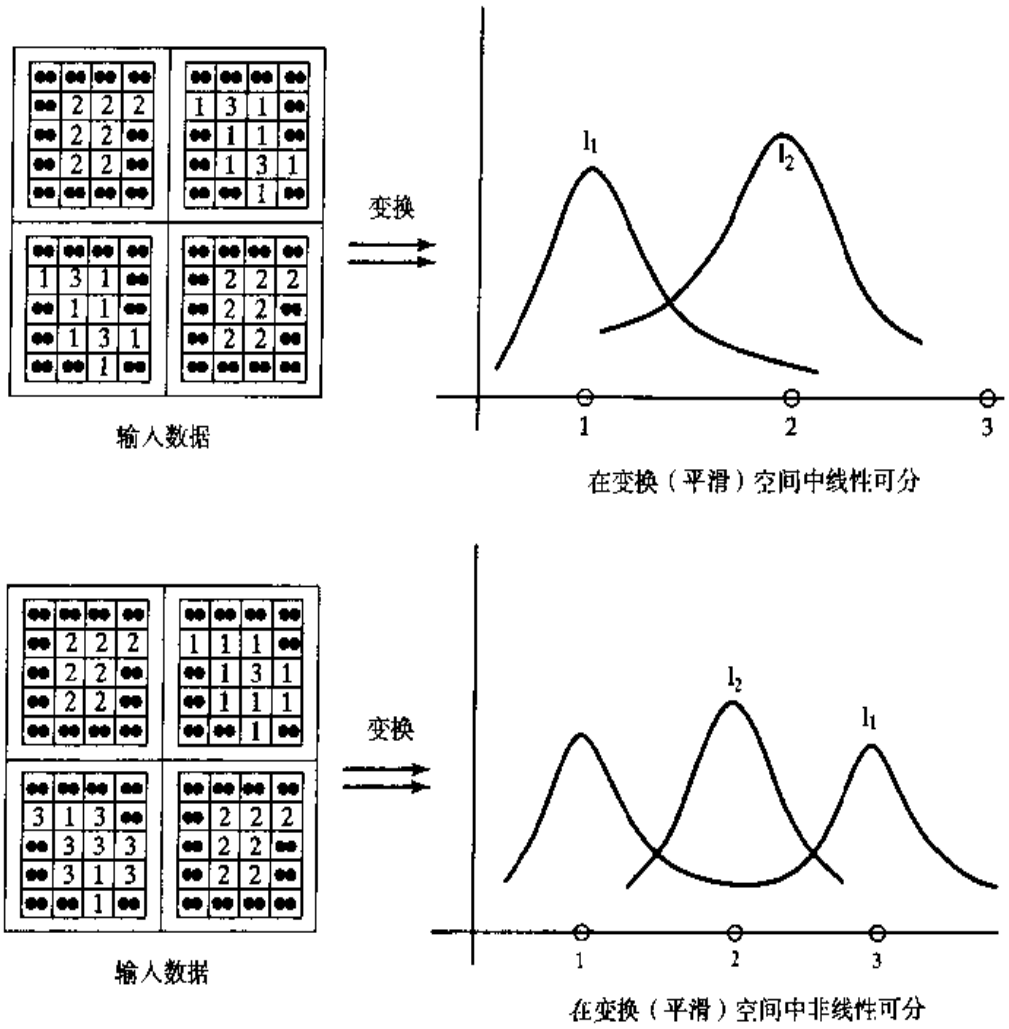


图7-10 隐含盐和辣椒空间模式的空间数据集

虽然MRF和SAR分类具有不同的形式，但是它们具有共同的目标，即计算后验概率分布：$p(l_i|X)$，l_i为$l(s_i)$的缩写。不过，这两种模型中后验概率的计算随假设的不同而不同。对于MRF而言，后验概率的计算使用贝叶斯规则。而在对数回归中，后验分布直接拟合数据。对于对数回归来说，类标号映射$C=\{l(s_i),\forall i\}$的概率由下式给出：

$$Pr(C|X)=\prod_{i=1}^{N}p(l_i|X) \tag{7.3}$$

对数回归与MRF之间一个重要的区别在于，对数回归假设邻近类别间不存在依赖关系。给定对数模型，在位置s_i其二元类标号取得第一个值c_1的概率为：

$$Pr(l_i|X)=\frac{1}{1+\exp(-Q_iX\beta)} \tag{7.4}$$

其中，邻近标号的依赖关系是通过W矩阵来实施，（Q_i中）的下标i标记矩阵Q的

第i行。这里我们利用了y可以重写为方程7.2的事实。

为了发现MRF公式与对数回归公式之间的局部关系（对于两类的情况$c_1=1, c_2=0$），对$C=\{0,1\}$在s_i点处，可推导出以下公式：

$$\begin{aligned} & Pr((l_i=1)|X, L_i) \\ &= \frac{Pr(X|l_i=1, L_i)Pr(l_i=1, L_i)}{Pr(X|l_i=1, L_i)Pr(l_i=1, L_i)+Pr(X|l_i=0, L_i)Pr(l_i=0, L_i)} \\ &= \frac{1}{1+\exp(-Q_i X\beta)} \end{aligned} \tag{7.5}$$

这意味着：

$$Q_i X\beta = \ln\left(\frac{Pr(X|l_i=1, L_i)Pr(l_i=1, L_i)}{Pr(X|l_i=0, L_i)Pr(l_i=0, L_i)}\right) \tag{7.6}$$

最后这个方程表明，空间依赖性是由W项通过Q_i引入。更重要的是，该方程还表明，在拟合β时我们同时试图拟合特征的相对重要性以及类标号的相对频率$\left(\frac{Pr(l_i=1, L_i)}{Pr(l_i=0, L_i)}\right)$。相反，在MRF公式中，我们明确地在类别的前项中为相对频率建模。最后，这个关系表明，我们在对数回归中建立了关于类条件分布的分布假设。对数回归和对数SAR模型属于更一般的指数族。指数族由下面公式给出：

$$Pr(u|v) = e^{A(\theta_v)+B(u,\pi)+\theta_v^T u} \tag{7.7}$$

u和v分别是位置和标号。该指数族包括许多常见的分布，如高斯分布、二项分布、贝努利分布和泊松分布等。参数θ_v和π控制分布的组成。方程7.6意味着这种类别条件的分布来自于指数族。此外，分布$Pr(X \mid l_i=1, L_i)$和$Pr(X \mid l_i=0, L_i)$在所有情况下都较平均值（如协方差、偏度、峰度等）更适配，因此在差$ln(Pr(X \mid l_i=1, L_i)) - ln(Pr(X \mid l_i=0, L_i))$中，高阶项抵消掉，而在方程7.6左边留下方程7.7中的线性项（$\theta_v^T u$）。

7.4 关联规则发现技术

关联规则是形如$X \rightarrow Y$的模式。在数据挖掘中，一个最为著名的关联规则模式是：尿片→啤酒。关联规则是一种较弱的相关性形式，因为关联规则无法发现负关联。例如，规则“豆腐$\xrightarrow{\text{购买}}$牛肉”（购买豆腐的人不大会购买牛肉）可能是真命题，但不

能将它看成是关联规则。使用概率论的术语来表述，关联规则$X \rightarrow Y$是条件概率$P(Y \mid X)$的一种表示。

一个关联规则可以特征化为两个参数：支持度（support）和置信度（confidence）。设$I = \{i_1, i_2, ..., i_l\}$是一组项集，$T = \{t_1, t_2, ..., t_n\}$是一组事务集，其中每个$t_i$是$I$的一个子集。设$C$是$I$的一个子集，则$C$对$T$的支持度是那些包含$C$的事务的数量：$\sigma(C) = \{t \mid t \in T, C \subset t\}$。注意，$\sigma(C)$表示该语境中集合$C$的基数（cardinality）而非关系代数操作符。$i_1 \rightarrow i_2$当且仅当以下两个条件满足时才成立：

支持度：i_1和i_2的出现至少占整个事务集的$s\%$：$\dfrac{\sigma(i_1 \wedge i_2)}{|T|}$

置信度：在i_1出现的所有事务中，至少有$c\%$的事务包含i_2：$\dfrac{\sigma(i_1 \wedge i_2)}{\sigma(i_1)}$

例如，考虑一个字母集合$I=\{A, B, C, D, E, F\}$和一个单词的事务集$T=\{ABC, ABD, BDE, CEF\}$，在事务集中单词的顺序无关紧要（即$ABC=BCA=CAB$）。表7-4给出三个规则$A \Rightarrow B$，$B \Rightarrow C$，$F \Rightarrow E$的支持度和置信度。另外一个例子（见图7-11）给出了一家电子商店在某一时刻销售状况的快照，还给出了具有较高支持度的项集和具有较高置信度的关联规则。下面我们来介绍快速发现大型数据库中关联规则的算法：*Apriori*。

7.4.1 Apriori：计算频繁项集的算法

Apriori算法可能是最广为人知的发现频繁项集的算法。频繁项集是满足预先定义的支持度阈值的项的集合。该算法利用支持度度量的一个简单且基本的单调性质：如果一个项集具有较高的支持度，则其所有子集也具有较高的支持度。Apriori算法的框架表示如下：

```
频繁项集:=∅;
k:=1;
While CandidateSet_k ≠ ∅ do
        为CandidateSet_k中的每一个项集创建一个计数器
        forall 数据库中的事务 do
                对在事务中出现的CandidateSet_k中的项集
增加其计数;
        Level_k := CandidateSet_k中所有超过支持度阈值的元素
        频繁项集 := 频繁项集 ∪ Level_k;
        CandidateSet_k+1 := 所有k +1项集
```

满足其k项子集在Level中

k = k+1;

end

表7-4 三个规则的支持度和置信度

规则	支持度	置信度
A⇒B	0.50	1.0
B⇒C	0.25	0.33
F⇒E	0.25	1.0

项

Car CD Player	D
Car Alarm	A
TV	T
VCR	V
Computer	C

频繁项集

支持度	项集
100%(6)	A
83%(5)	C, AC
67%(4)	C, T, V, DA, DC, AT, AV, DAC
50%(3)	DV, TC, VC, DAV, DVC, ATC, AVC, DAVC

事务

1	DAVC
2	ATC
3	DAVC
4	DATC
5	DATVC
6	ATV

置信度 = 100%的关联规则

D ⟶ A (4/4)	D ⟶ A (4/4)	VC ⟶ A (3/3)
D ⟶ C (4/4)	D ⟶ A (3/3)	DV ⟶ A (3/3)
D ⟶ AC (4/4)	D ⟶ A (3/3)	VC ⟶ A (3/3)
T ⟶ C (4/4)	D ⟶ A (4/4)	DAV ⟶ A (3/3)
V ⟶ A (4/4)	D ⟶ A (3/3)	DVC ⟶ A (3/3)
C ⟶ A (5/5)	D ⟶ A (3/3)	AVC ⟶ A (3/3)

置信度≥80%的关联规则

C ⟶ D (4/5)	A ⟶ C (5/6)	C ⟶ DA (4/5)

图7-11 事务数据库、频繁项集和高置信度规则的例子

Apriori首先发现所有频繁的（即大于支持度阈值）1项集（单项）。第二步是组合所有的频繁项集，形成2项集$CandidateSet_2$。算法然后评估这个集合以搜索频繁2项集。以后依次类推：频繁2项集组合形成3项集，直到集合$CandidateSet_i$为空为止。

当所有频繁项集都计算出来之后，下一步是搜索满足最小置信度要求的规则，方法如下：给定一个频繁项集{*ABC*}，检验所有可能的规则是否满足置信度参数*c*。例如，对于如下每个规则：

$$\{AB\}\rightarrow\{C\}$$

$$\{BC\}\rightarrow\{A\}$$

$$\{CA\}\rightarrow\{B\}$$

检查其置信度度量，那些超过阈值c的规则即是合法的关联规则。这些规则的置信度可以通过频繁项集的支持度计算出来。

生成空间关联规则有两种方法。第一种方法的焦点是空间谓词而不是项；第二种方法将事务概念泛化以包括邻域，将关联规则的概念泛化为同位规则(colocation rule)。

7.4.2 空间关联规则

空间关联规则是根据空间谓词而不是根据项来定义的。一个空间关联规则是形式如下的规则：

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \rightarrow Q_1 \wedge \ldots \wedge Q_m$$

这里，P_i和Q_j中至少有一个是空间谓词。例如，规则

$$is\_a\ (x,\ country) \wedge touches\ (x,\ Mediterranean) \xrightarrow{s\%,\ c\%} is_a(x,\ wine\text{-}exporter)$$

（即，靠近地中海的国家通常是葡萄酒出口国）是一个支持度为s和置信度为c的关联规则。表7-5给出一个根据1995年Darr湿地数据所发现的关联规则的例子。由于关联规则用于分类属性，因此对于数据集为数值型的应用来说就很受限制。这是因为从数值到分类数据的变换涉及到一个离散化过程，在大多实例中这会有某些随意性。例如，在Darr湿地的例子中，何谓high Stem-Height（高的茎干高度）？

表7-5 从1995年Darr湿地数据中发现的空间关联规则的例子

空间关联规则	支持度	置信度
$Stem\_height(x, high) \wedge Distance\_to\_edge(x, far) \rightarrow Vegetation\_Durability(x, moderate)$	0.1	0.94
$Vegetation\_Durability(x, moderate) \wedge Distance\_to\_water(x, close) \rightarrow Stem\_Height(x, high)$	0.05	0.95
$Distance\_towater(x, far) \wedge Water\_Depth(x, shallow) \rightarrow Stem\_Height(x, high)$	0.05	0.94

7.4.3 同位规则

同位规则试图将关联规则泛化为空间索引的点集合数据集。在空间与非空间关联之间有几个关键区别，包括：

1) 在空间数据的环境中，没有事务的概念，因为数据嵌入到连续空间中。把空间分区成事务会导致高估或低估所感兴趣的度量（如支持度或置信度）。

2) 空间数据库中项集的规模比较小，即在空间情况下项集中的项数远小于非空间情况下的项数。例如，在零售业中，处理动辄有上万个项数的不同项的情况非常普遍，而对空间数据集来说，这种情况就很少出现，空间项一般不超过几十个。这意味着候选集生成的代价不再是Apriori算法的支配因素，而邻域的枚举（例如，频繁项集的实例）在整体的计算代价中占主导地位。

3) 在多数情况下，空间项是连续变量的离散化版本。例如，在美国，一般把那些平均年收入大于5万美元的区域称为高收入区域。

在这种空间关联规则发现方法中，事务的概念被邻域所替代，我们通过一个例子来说明这个方法。同位模式发现过程根据位置图寻找频繁的空间事件类型的同位子集（见图7-12）。例如，对动植物生活习性的分析可以得出捕猎肉食动物的物种、共生物种和具有燃烧源的火灾事件之间的同位性。读者会发现，对图7-12中的地图进行分析来寻找同位模式是非常有趣的事情。在图的下方，可以注意到该地图中有两个大小为2的同位模式。

7.5 聚类

聚类（clustering）是一个在大型数据库中发现“群”或“簇”的过程。与分类不同，聚类并不涉及群数或群标号的先验信息，因而，在聚类中没有训练或测试数据的概念，这就是将聚类称为是无指导学习（unsupervised learning）的原因。

簇是基于一种用于确定数据库中每对元组之间关系的“相似性”标准而形成的。相似的元组通常分在一个组中，然后再标记这个组。例如，卫星图像的像素一般根据光谱信号进行聚类。通过这种方式，不需要太多的人为干预就可以将遥感图像快速分段。当然，领域专家还必须对聚类结果进行检查、验证，而且可能的话还要精

炼聚类。在1996年美国总统大选中出现过一个人口分段的著名例子，政治专家们指出“足球妈咪”[⊖]（Soccer Moms）将是决定选举结果的人群，随后各主要政党对足球妈咪大献殷勤。在犯罪分析和疾病追踪中，聚类是另一种确定“热点”的技术。

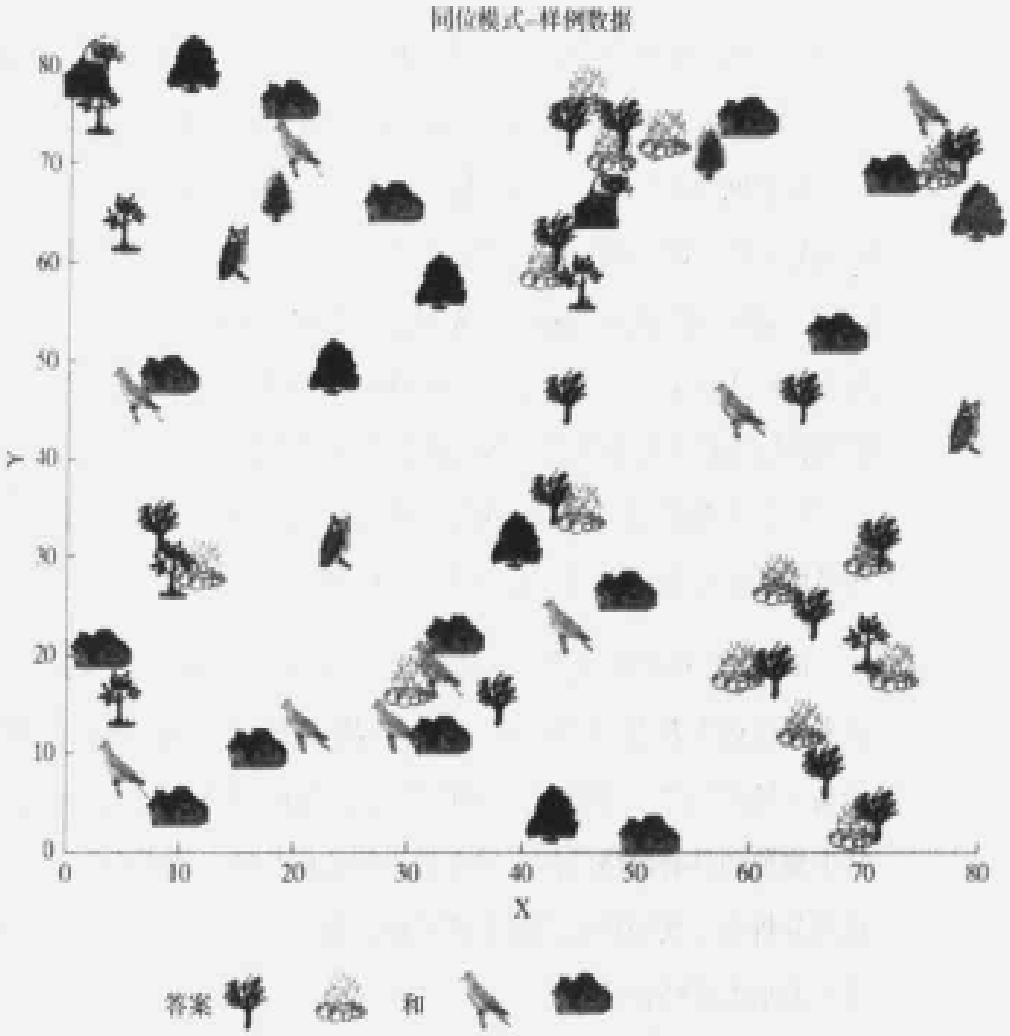


图7-12 同位模式举例

统计学中的聚类是人们熟知的一种技术，数据挖掘的作用是要提升聚类算法，使之能够处理目前非常常见的海量数据集。数据库的大小是表中记录的数量和每条记录的属性数量（维数）的函数。除了容量之外，数据类型（数值、二值、分类、序数）也是确定要采用的算法的重要决定性因素。

在多维属性空间中，框定聚类问题是很方便的。给定由m个变量描述的n个数据

⊖ “足球妈咪”是指美国的一批中产阶级妇女，周末因带小孩踢足球而常聚在一起。她们通常持有比较开通的政治观点。

对象，每个对象可以表示为m维空间中的一个点，这时聚类可以简化为从一组非均匀分布点中确定高密度的点群。在多维空间中搜索潜在的群组则需要首先选择合理的相似性标准。

例如，在美国人口普查数据中，对郡的聚类可以基于以下四个属性：失业率、人口、人均收入和平均寿命，将具有相似属性值的郡聚集到一起。

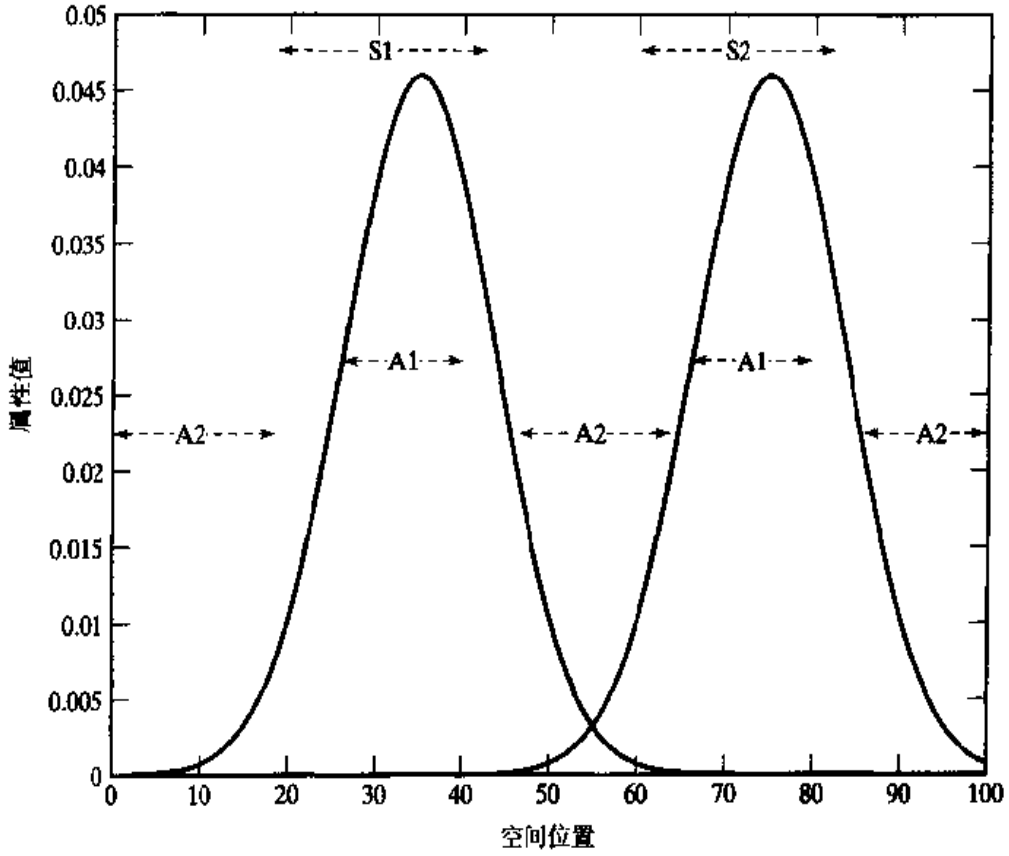


图7-13 空间聚类的两种解释。如果目标是确定主导周围环境（所谓影响力）的位置，则聚类是S1和S2；如果目标是确定均一属性值的区域，则聚类是A1和A2

当所处理的属性数据具有物理（如地理）空间参照时，聚类问题有两种解释。考虑图7-13的曲线，其属性值（例如，人口密度）变化是x轴所示的位置的函数。那么什么是聚类，又如何解释聚类呢？例如，如果我们的目标是通过图中的属性值变化来辨识热点，例如，通过用一个随地域变化的属性值的方差，来度量从一组城市里选出的中心的城市及受其影响的区域，则我们在图7-13中寻找标记为S1和S2的空间聚类。另一方面，如果我们的目标是确定图中属性均一表示的群组，则我们寻找标识为A1和A2的聚类。虽然第二种解释从本质上讲是非空间的，但是由于在属

性数据中可能存在空间自相关性，故存在空间方面的问题。所标识的聚类应该是空间同质的而非“混杂的”。聚类问题的这两种解释可以形式化定义如下：

定义1 给定：一组空间对象（例如点）的集合$S=\{s_1, ..., s_n\}$和一个在S上判定得到的实数值非空间属性f（例如，$f: S \rightarrow R$）。

寻找：S的两个不相交子集C和$NC = S - C$，其中$C=\{s_1, ..., s_k\}$，$NC=\{nc_1, ..., nc_l\}$，并且$k < n$。

目标：$\min_{C \subset S} \sum_{j=1}^{l} | f(nc_j) - \sum_{i=1}^{k} \frac{f(c_i)}{dist(nc_j, c_i)} |^2$

其中：$dist\ (a, b)$是欧氏距离或其他距离度量。

约束：

1) 数据集符合中心位置理论，即假设中心城市的影响随距离的平方增加而衰减。

2) 至多有一个非空间属性。

定义2 给定：1) 一个空间对象（如点）的集合$S=\{s_1, ..., s_n\}$和一组定义在S上的实数值非空间属性f_i，$i=1, ..., I$（即对于每个i，有$f_i: S \rightarrow R$）；2) S上的邻域结构E。

寻找：K个子集$C_k \subset S$，$k=1, ..., K$

目标：$\min_{C_k \subset S} \sum_{s_i, s_j \in C_k, s_j \in C_k} dist(F(s_i), F(s_j)) + \sum_{i,j} nbddist(C_i, C_j)$

其中：1) F是f_i的笛卡儿积，$i=1, ..., n$；2) $dist\ (a, b)$是欧氏距离或其他距离度量；3) $nbddist\ (C, D)$是C和D中属于E的点的数量，即映射到不同聚类的邻居对。

约束：对于所有$k=1, ..., K$，有$|C_k|>1$。

聚类算法的分类

在许多领域中，聚类分析是最常用的数据分析技术之一，因此出现了大量聚类算法。对这些算法进行分类是非常有用的。根据定义聚类时所采用的技术，聚类算法可以分为以下四类：

1) 层次的（hierarchical）聚类方法以所有模式作为单一聚类开始，然后连续执行分裂和合并，直到满足某个终止标准为止。最后产生一个聚类树，该树称为树状图(dendogram)。可以在不同层次上切割树状图来产生所期望的聚类。可以将层次算

法进一步划分为融合（agglomerative）方法和分裂（divisive）方法。一些典型的层次聚类算法包括：采用层次方法的平衡迭代归约和聚类（BIRCH）、采用代表点的聚类（CURE）以及采用链的健壮聚类（ROCK）。

2) 分区（partitional）聚类算法以每个模式作为单一聚类开始，迭代地重新分配数据点到每个聚类，直至满足某个终止标准为止。这些方法适合用于发现球形的聚类。*K-means*和*K-medoids*是常用的分区算法。平方误差是分区聚类中最常用的标准函数。近来出现的这类算法包括：围绕中心点的划分（PAM）、聚类大型应用（CLARA）、基于随机搜索的大型应用聚类（CLARANS）以及期望最大化（EM）。

3) 基于密度的（density-based）聚类算法基于某一区域中数据点的密度来尝试发现聚类。这些算法将聚类看作数据空间中对象的密集区域。一些基于密度的聚类算法包括：带噪声的应用的基于密度的空间聚类（DBSCAN）和基于密度的聚类（DENCLUE）。

4) 基于网格的（grid-based）聚类算法首先将聚类空间离散量化为有限数量的单元格，然后在离散的空间上执行所要求的操作。包含的点多于特定数量的单元格被认为是密集的，将密集的单元格连接起来形成聚类。基于网格的聚类算法主要用于分析大型的空间数据集。一些基于网格的聚类算法包括：统计信息基于网格的方法（STING）、STING+、WaveCluster、BANG聚类和CLIQUE。

有时候，这些不同类别的聚类算法之间的差别会变弱，有些算法甚至可以分属于多个类别。例如，CLIQUE既可以归为基于密度的算法，也可以归为基于网格的聚类方法。

下面我们来阐述两种大家熟知的聚类方法，即*K-medoid*算法和使用EM算法的混合分析。我们还将简单讨论如何根据空间数据的特殊性质修改EM算法。

7.5.1 K-medoid聚类算法

虽然*K-medoid*技术可以很容易地推广到高维空间中，但是我们仍然要重点考虑二维空间R^2中的点。给定二维空间R^2中n个数据点的一个集合P，$P=\{p_1, p_2, \ldots, p_n\}$，*K-medoid*聚类的目标是将数据点划分为$k$个聚类，使得下述目标函数最小化：

$$J(M) = J(m_1, \ldots, m_k) = \sum_{i=1}^{k} \sum_{p \in C_i} d(p, m_i)$$

在$J(C)$中，m_i是聚类C_i的代表点。如果m_i被限制为P的一个成员，则称之为一个中心点（medoid）。另一方面，如果m_i是聚类点的平均、并且不一定是P的成员，则称之为平均点（mean）。因此*K-mean*与*K-medoid*方法是密切相关的。虽然*K-mean*算法更广为人知，但我们仍将焦点放在*K-medoid*方法上，因为与中值不同，中心点对孤立点不敏感。

*K-medoid*将K个聚类特征化，P中每个点属于其最近的中心点。由于我们将周围空间限制为R^2，所以距离函数d是普通的欧氏距离：

$$d(p, m_i) = ((p(x) - m_i(x))^2 + (p(y) - m_i(y))^2)^{\frac{1}{2}}$$

这样，*K-medoid*方法将聚类问题转变成一个搜索问题。搜索空间X是由P的所有k项子集M构成的集合（即$|M|=k$），目标函数是$J(M)$。X可以建模为一个图，图的结点是X中的元素。如果$|M_1 \cap M_2|=k-1$，则结点M_1与M_2邻近（即它们之间的差别为且仅为一个数据点）。

*K-medoid*算法包括下述步骤：

1) 选择X中任意一个结点M_0。

2) 迭代地从当前结点M_i移动到邻近结点M_{i+1}，使得$J(M_{i+1})<J(M_i)$。将当前中心点m替代为P中的一个数据点$p \in P$，形成从当前结点到邻近结点的移动。因而，$M_{i+1} = M_i \cup \{p\} - \{m\}$。

3) 当对于所有的邻近结点有$J(M_{i+1}) \geq J(M_i)$时结束。

其中第二步是算法的核心。从一个结点移动到其邻近结点有很多选择，表7-6列出一些可选方案。表中包括相关文献中提及的每个选项的名称、移动策略以及是否该选项可以保证局部最优。由于仅仅利用了邻近结点，故所有的选项都是局部搜索的实例。

表7-6 聚类中局部搜索的四种选项

局部搜索	从M_i到$M_{i+1} = M_i \cup \{p\} - \{m\}$的移动策略	能保局部最优
Global hill climbing(HC)	移动到最佳邻居	是
Randomized HC	移动到最佳的已采样的邻居	否

（续）

局部搜索	从M_i到$M_{i+1}=M_i\cup\{p\}-\{m\}$的移动策略	确保局部最优
Local HC	一发现新邻居就移动过去	是
Distance-restricted HC	移动到指定距离之内的最佳邻居	否

7.5.2 聚类、混合分析和EM算法

K-medoid（或*K-mean*）方法的缺陷在于它生成硬性划分的聚类，即每个数据点唯一地分配给一个且仅一个聚类。由于事先不知道实际的聚类情况，因此这可能是一种严重的局限。在统计学文献中，聚类问题经常按照混合模型（mixture model）重构。在混合模型中，假设数据是由一个概率分布序列产生的，每个分布产生一个聚类。于是，目标变为确定每个概率分布的参数及其在整个混合分布中的权值。在混合模型中，数据库的每个实例以不同的隶属度（grade of membership）归属于所有的聚类，隶属度通过混合模型中某个分布的权值来量化。因此，混合模型框架比*K-medoid*方法更灵活。通常，每个概率分布表现为正态分布，难点在于确定每个分布的均值、方差和权值。正态性假设并不像表面所看到的那样受限制，因为统计学定理保证任何概率分布都可以表示为有限个正态分布的总和。

1. 有限混合示例

考虑图7-14中的4×4灰度图像。假定我们期望将像素集合划分为两个聚类A和B，每个聚类建模为一个高斯分布。有限混合模型问题就是计算参数μ_A、μ_B、σ_A、σ_B、p_A、p_B。

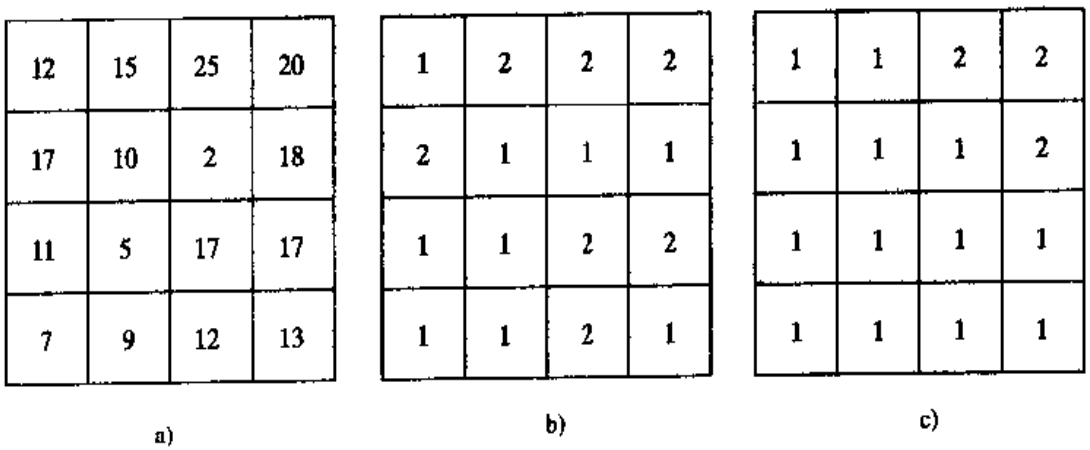


图7-14 a）4×4灰度图像。b）使用EM算法产生的图像的标号。c）使用邻域EM算法对同样的图像产生的标号。注意，空间平滑是通过修改目标函数来实现的

此时，假设每个像素的聚类隶属关系在图7-14b中给出，那么所有的参数可以很容易地计算出来。例如，

$$\mu_A = \frac{12+10+2+18+11+5+7+9+13}{9} = 9.7$$
$$\sigma_A = \frac{(12-\mu_1)^2+(10-\mu_1)^2+\ldots+(13-\mu_1)^2}{8} = 4.7$$
$$p_A = \frac{9}{16}$$

用类似的方法可以计算出，$\mu_B = 17.6$、$\sigma_B = 4.1$和$p_B = \frac{5}{16}$。计算给定像素属于某个聚类的概率只要使用贝叶斯定理即可计算出来。例如，给定一个像素值x，它属于聚类A的概率为：

$$P(A|x) = \frac{P(x|A)p_A}{P(x)} = \frac{P(x|A)p_A}{P(x|A)p_A + P(x|B)p_B} = \frac{N(x,\mu_A,\sigma_A)p_A}{N(x,\mu_A,\sigma_A)p_A + N(x,\mu_B,\sigma_B)p_B},$$

其中，

$$N(x,\mu_A,\sigma_A) = \frac{1}{\sqrt{(2\pi)}\sigma_A}\exp^{\frac{-(x-\mu_A)^2}{2\sigma^2}}$$

这里，聚类标号和分布参数都是未知的，我们仅仅知道有两个聚类，每个聚类建模为高斯分布。乍看起来，这个问题似乎是不可解的，因为有太多的未知内容，其中包括每个像素的聚类标号和聚类的分布参数。实际上，这类问题可以使用EM算法来求解。与*K-medoid*算法类似，EM算法也是一种迭代算法，起初它猜测分布参数的估计值；然后，根据给定的初始参数计算数据的期望值，这个新的期望数据值则用于计算分布参数的最大似然估计（见附录中最大似然估计的简短讨论）。这个过程不断迭代，直到满足一些收敛条件为止。虽然收敛速度比较慢，但EM算法保证每次迭代后都会使最大似然估计得到改善。EM算法的步骤如下：

1) 猜测初始的模型参数：u_A^0、$\sum_A^0$、p_A^0、u_B^0、$\sum_B^0$、p_B^0。

2) 对于每次迭代j，计算数据对象x属于聚类A和B的概率。

$$P(A|x)=\frac{p_A^j P^j(x|A)}{P^j(x)} \quad P(B|x)=\frac{p_B^j P^j(x|B)}{P^j(x)}$$

3) 在新的估计基础上更新混合模型的参数：

$$p_A^{j+1}=\frac{1}{n}\sum_x P(A|x) \qquad p_B^{j+1}=\frac{1}{n}\sum_x P(B|x)$$

$$\mu_A^{j+1}=\frac{\sum_x xP(A|x)}{\sum_x P(A|x)} \qquad \mu_B^{j+1}=\frac{\sum_x xP(B|x)}{\sum_x P(B|x)}$$

$$\sigma_A^{j+1}=\frac{\sum_x P(A|x)(x-\mu_A^{j+1})^2}{\sum_x P(A|x)} \qquad \sigma_B^{j+1}=\frac{\sum_x P(B|x)(x-\mu_B^{j+1})^2}{\sum_x P(B|x)}$$

4) 计算对数估计 $E_j=\sum_x \log(P^j(x))$。如果对于某些固定的终止条件ε，满足$|E_j-E_{j+1}|\leq\varepsilon$，则停止；否则，设置$j=j+1$。

2. 邻域EM算法

细心的读者可能会发现，EM算法完全忽略了像素的空间分布，仅仅利用了像素值。因此，如果我们重新安排图7-14a中所示的像素值，EM算法仍然会产生同样的聚类标号和同样的分布参数值[⊖]。我们知道，这样的求解方法并没有考虑空间数据固有的空间自相关性质。正如我们在前面提到的那样，空间参照数据的搜索空间是概念属性空间和物理（地理）空间的组合，而空间自相关性质意味着聚类在实际空间中应该是渐变的。

为了使EM算法具有空间敏感性，我们首先遵循[Ambroise et al., 1997]提出的方法。

步骤一 混合模型的EM算法等价于如下目标函数的优化：

$$D(c,\mu_k,\sigma_k,p_k)=\sum_{k=1}^{2}\sum_{i=1}^{n}c_{ik}\log(p_k N(x_i,\mu_k,\sigma_k))-\sum_{k=1}^{2}\sum_{i=1}^{n}c_{ik}\log(c_{ik}),$$

这里$c=c_{ik}$，$i=1,\ldots,n$，而$k=1,\ldots,K$定义了一个模糊分类，代表数据点x_i对于聚类k的隶属度。c_{ik}满足约束（$0<c_{ik}<1$，$\sum_{k=1}^{2}c_{ik}=1$，$\sum_{i=1}^{n}c_{ik}>0$）。注意，我们有两个聚类$k=1,2$和n个数据点。

⊖ 实际上，由于初始参数的随机性，每次运行EM算法可能会得到不同的结果。

步骤二 为了考虑空间自相关性，我们引入新的项

$$G(c)=\frac{1}{2}\sum_{k=1}^{2}\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ik}c_{jk}w_{ij}$$

这里$W=(w_{ij})$是前文定义的邻接矩阵（contiguity matrix）。

新的“空间加权”的目标函数为：

$$U(c,\mu,\sigma)=D(c,\mu,\sigma)+\beta G(c)$$

这里$\beta\geq 0$是控制数据集的空间同质性的参数。

步骤三 除了新参数c（是一个$n\times 2$矩阵）之外，其他所有参数的计算与前面相同。c_{ik}的计算公式如下：

$$c_{ik}^{m+1}=\frac{p_k^m N(x_i,\mu_k,\sigma_k)\exp\{\beta\sum_{j=1}^{n}c_{jk}^{m+1}w_{ij}\}}{\sum_{l=1}^{2}p_l^m N(x_i,\mu_l^m,\sigma_l^m)\exp\{\beta\sum_{j=1}^{n}c_{jl}^{m+1}w_{ij}\}}$$

在每次迭代m中，c_{ik}^{t}可以使用固定点迭代模式求出。

我们已经使用邻域EM（NEM）算法在鸟类数据集上进行了实验。假定用两个聚类对应于鸟巢的存在和不存在。当$\beta=0$时，NEM简化为经典的EM算法。改变β参数的结果在图7-15中给出。结果表明在聚类算法中考虑空间信息会使结果得到极大改善（比较图7-15b与7-15a），但是过于强调空间信息则会导致过于平滑和准确度下降。

7.5.3 大型空间数据库聚类的策略

现在我们讨论如何利用第4章介绍的空间索引结构来提高*K-medoid*算法的伸缩性（scale）。

假设某空间数据库中包含n个点，并且无法同时在主存中驻留所有点，我们再做如下附加假设：

1) 在空间数据库管理系统中已有可以使用的空间索引结构，例如R树或者Z序。

2) c是存储在一个磁盘页面中的平均点数。

3) k是聚类的数量。

4) *K-medoid*算法代价主要受步骤二（计算$J(M_{mi})-J(M_i)$）的影响。

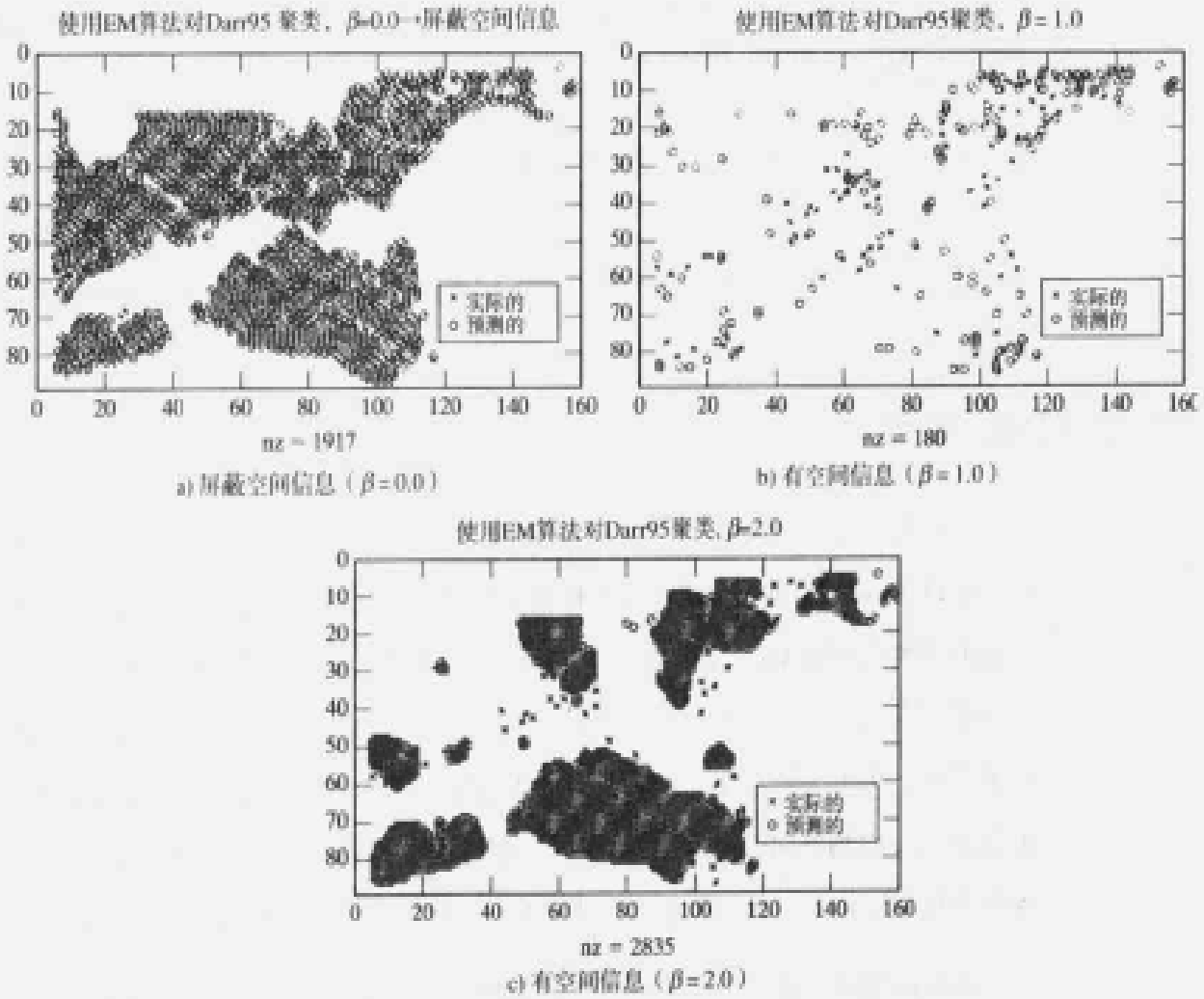


图7-15 使用邻域EM算法。a) 由于预期的聚类没有考虑任何空间信息，所以结果不佳。b) 考虑空间信息（$\beta=1.0$）使结果有很大改善。c) 过于强调空间信息（$\beta=2.0$）再次导致糟糕的结果

1. 通过R树采样

R树的叶结点对应着一个与MBR关联的点的集合。我们可以通过从每个叶结点选择一个样本点（如平均值或中心点）得到n个点的代表样本。一种常用的样本点选择方法是选择到MBR质心最近的数据点。因此，算法需要聚类的不是n个点，而是平均，即n/c个点。

2. 仅选择相关的聚类

一种计算$J(M_{i+1})-J(M_i)$的方式是：对所有的非中心点点循环，再度计算距离。对于大型数据库来说，这项策略引发的代价是令人难以接受的。幸运的是，仅有那些在$M_{i+1}=M_i\cup\{p\}-\{m\}$中与旧中心点$m$和新中心点$p$相关的非中心点才会对计算

$J(M_{i-1}) - J(M_i)$有贡献。因此，只把m和p的聚类点读入主存中。这种方法的高效性基于聚类中的点的高效检索。

我们观察到，与中心点相关的*Voronoi*多边形包含其聚类点，这可以作为一种高效检索某一中心点的聚类点的方法。于是，一个查询区域为Voronoi多边形的范围查询可以检索到中心点m的所有聚类点。使用R*树或者Z序索引可以高效地处理这样的查询。

7.6 空间孤立点检测

通俗来讲，全局孤立点就是在数据集中与其他数据点表现不一致的观测点[Barnett和Lewis，1994]，或者大大地偏离其他观测点以至于怀疑它是由不同的机制生成的观测点[Hawkins，1980]。确定全局孤立点的可能会发现意料之外的知识，确定全局孤立点在许多领域中有大量的实际应用，如信用卡欺诈、运动员表现分析、投票违规、恶劣天气预测。本节讨论的重点是空间孤立点，即与其邻域表现不一致的观测点。检测空间孤立点在地理信息系统和空间数据库的很多应用中非常有用。这些应用领域包括交通、生态、公共安全、公众健康、气候和基于位置的服务。

我们将空间数据集建模为一个空间参照对象的集合，如房屋、道路和交通传感器。空间对象根据其被度量的属性有两种不同类型的维，即空间的和非空间的。空间参照对象的空间属性包括位置、形状和其他的几何或拓扑性质。空间参照对象的非空间属性包括交通传感器标识符、制造商、所有者、年龄以及测量材料。空间参照对象的空间邻域是基于空间维的一个空间数据的子集，如位置。空间邻域可以基于空间属性（如位置）使用空间关系（如距离或邻接）进行定义。空间参照对象之间的比较基于非空间属性进行。

空间孤立点是与其空间邻域中其他空间参照对象的非空间属性值有极大不同的空间参照对象。通俗来说，空间孤立点是非空间属性值的局部不稳定，或者相对于邻居表现极端的空间参照对象（即使它们与整个总体没有大的区别）。例如，在一个新兴大都市的老社区中，一栋新房屋就是基于非空间属性房屋年代的空间孤立点。

我们通过一个例子来说明全局孤立点与空间孤立点的检测方法之间的区别。在图7-16a中，*X*轴是一维空间中数据点的位置，*Y*轴是每个数据点的属性值。全局孤

立点检测方法忽略每个数据点的空间位置，而拟合非空间属性值的分布模型。使用这种方法检测到的孤立点是数据点G，它有极大属性值7.9，超过图7-16c所示的阈值$\mu+2\sigma = 4.49+2 \times 1.61=7.71$。该检测假设属性值是正态分布的。

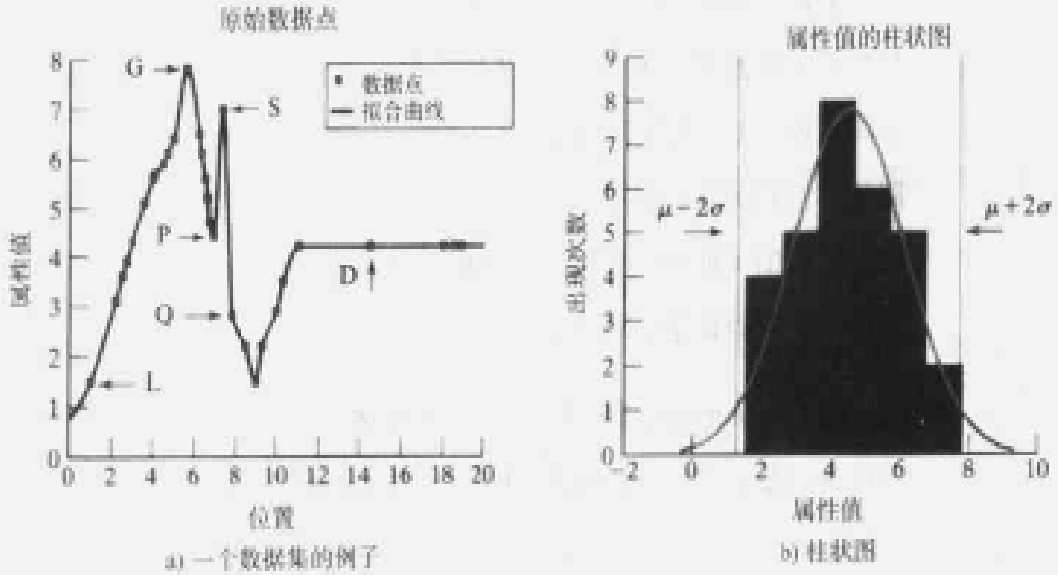


图7-16 孤立点检测的数据集

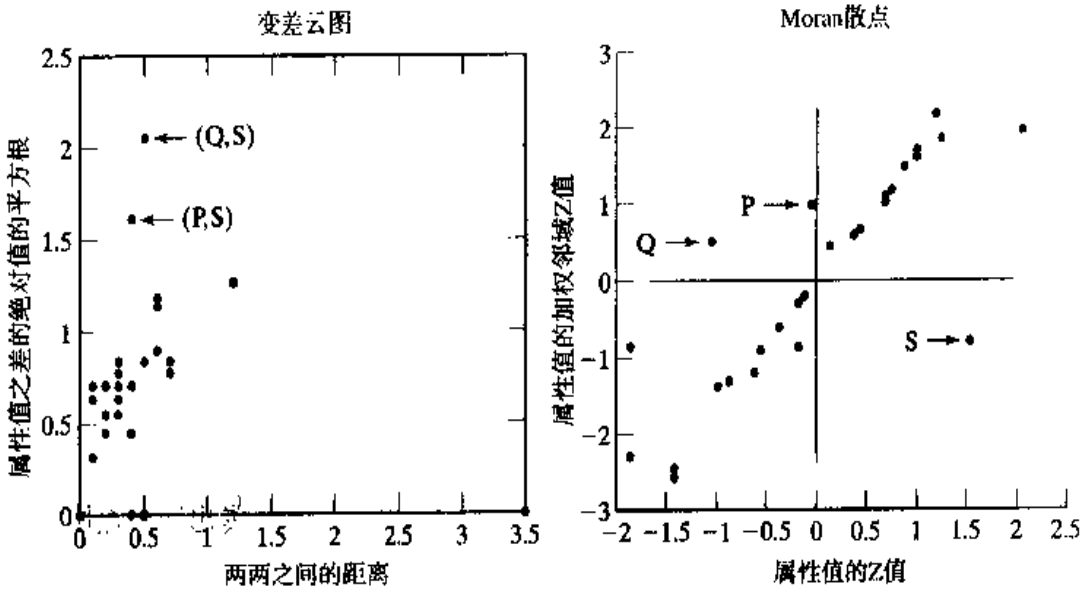
检测空间孤立点的测试将空间属性与非空间属性分隔开来。空间属性用于刻画位置、邻域和距离；非空间属性维用于比较空间参照对象与其邻居。在相关的空间统计学文献中提供了两种测试，即图形测试（graphical test）和定量测试（quantitative test）。图形测试基于空间数据的可视化，该方法强调空间孤立点，例如变差云图（variogram cloud）和Moran散点图。定量方法提供了一个精确测试，将空间孤立点与其他数据区分开来。[Luc, 1994]中提出的散点图是定量方法族中有代表性的技术。

变差云图显示数据点及其邻域的关系。对于每一对位置，绘出其位置上属性值差的绝对值的平方根与位置间的欧氏距离。在表现出强空间依赖性的数据集中，属性差的方差随着位置间距离的增加而增大。互相接近而属性差很大的位置可能就是一个空间孤立点，即使在非空间参照情况下检查数据集时，两个位置上的属性值看起来很合理。图7-17a是图7-16a所示的例子数据集的变差云图，该图显示了左边的两个位置对(P, S)和(Q, S)位于主要对群的上方，所以这两个位置对可能与空间孤立点有关。由于点S同时出现在位置对(P, S)和(Q, S)中，故S可以确定为一个空间孤立

点。然而，空间孤立点检测的图形测试受到很大限制，因为缺乏区分孤立点的精确标准。另外，变差云图需要不少的后处理工作，即突出显示位置对以便从其邻居中分离出空间孤立点，特别是当存在多个孤立点或密度变化较大时更是如此。

[Luc, 1995]中提出的Moran散点图是绘制归一化属性值（$Z[f(i)] = ((f(i)-\mu_i)/\sigma_i)$）与归一化属性值（$W \cdot Z$）的邻域平均，这里$W$是行归一化（如$\sum_j W_{ij}=1$）的邻域矩阵（$W_{ij}>0$，当且仅当满足$neighbor\ (i, j)$时）。图7-17b的左上和右下象限表明不同值的空间关联：低属性值被高属性值的邻居所包围（如点P和Q）、高属性值被低属性值包围（如点S）。因此，我们可以确定出那些被异常高或低的邻居属性值包围数据点，它们就是空间孤立点。

定义 Moran的孤立点是位于Moran散点图左上和右下象限中的数据点。这些点可以通过公式$(Z[f(i)])\times(\sum_j(W_{ij}Z[f(j)]))<0$来确定。



a) 变差云图　　b) Moran散点图

图7-17 检测空间孤立点的变差云图和Moran散点图

[Luc, 1994]中提出的散点图在X轴显示属性值，在Y轴显示邻域中属性的平均值，然后利用最小二乘回归线来确定空间孤立点。散布方向为右上表示空间自相关性（邻接值倾向于相似）为正；散布方向为左上表示空间自相关性为负。残量为点P（位置为(X_p, Y_p)）到回归线$Y=mX+b$的竖直距离（Y轴方向），即残量$\varepsilon = Y_p - (mX_p + b)$。当标准残量$\varepsilon_{标准} = \dfrac{\varepsilon-\mu_\varepsilon}{\sigma_\varepsilon}$大于3.0或小于-3.0时，标志着该点可能是空间孤立点，

这里μ_ε和σ_ε是误差项ε的分布的均值和标准差。在图7-18a中，对于图7-16a所示的数据集，散点图绘出属性值与邻近区域内属性值的平均。点S被证明距离回归线最远，因此该点为空间孤立点。

定义 散点图中的孤立点是散点图中到最小二乘回归线有很大标准残量误差的数据点。假设误差符合正态分布，则常用$\varepsilon_{标准}=\left|\frac{\varepsilon-\mu_\varepsilon}{\sigma_\varepsilon}\right|>\theta$进行检验。到回归线$Y=mX+b$的标准残量$\varepsilon_{标准}=\frac{\varepsilon-\mu_\varepsilon}{\sigma_\varepsilon}$大于$\theta$或小于$-\theta$的结点就被视为可能的空间孤立点。$\mu_\varepsilon$和$\sigma_\varepsilon$是误差项$\varepsilon$的分布的均值和标准差。

一个位置也可以通过函数$S(x)=[f(x)-E_{y\in N(x)}(f(y))]$与其邻域行比较，这里$f(x)$是位置$x$的属性值，$N(x)$是$x$的邻居集合，$E_{y\in N(x)}(f(y))$是$x$的邻居的平均属性值。统计函数$S(x)$表示位于$x$处的传感器的属性值与$x$的邻居的平均属性值之差。

如果属性值$f(x)$符合正态分布，则空间统计量$S(x)$也符合正态分布。对于正态分布$f(x)$，一个常用的空间孤立点检测方式可以描述为：空间统计量$Z_{s(x)}=\left|\frac{S(x)-\mu_s}{\sigma_s}\right|>\theta$。对于属性值为$f(x)$的每个位置$x$，$S(x)$是位置$x$处的属性值与$x$的邻居的平均属性值之差，$\mu_s$是$S(x)$的平均值，$\sigma_s$是所有观测点上$S(x)$的标准差。$\theta$的选择和指定的置信度水平有关。例如，若置信度水平为95%，则$\theta\approx2$。

图7-18b利用图7-16a的数据集给出空间统计量$Z_{s(x)}$方法的可视化结果。X轴是一维空间中数据点的位置，Y轴是针对每个数据点的空间统计量$Z_{s(x)}$的值。可以很容易地观察到点S的$Z_{s(x)}$值超过3，故将S点标识为空间孤立点。注意，S的两个邻近点P和Q的$Z_{s(x)}$值接近-2，因为在它们的邻域中存在空间孤立点。

现在我们给出一个应用领域的空间孤立点案例研究。图7-19显示了一个传感器装置的网络，这些传感器安装在美国明尼苏达州明尼阿波利斯-圣保罗（双子城）城区周围的州际高速公路上。900个装置（表示为多边形）用来测量规定间隔情况下特定高速公路路段的交通量和占用率。此时邻域概念自然要按照图连通性而不是欧氏距离进行定义。我们的目标是基于每个装置的交通状况度量（如交通量）的值来确定哪些装置是孤立点。

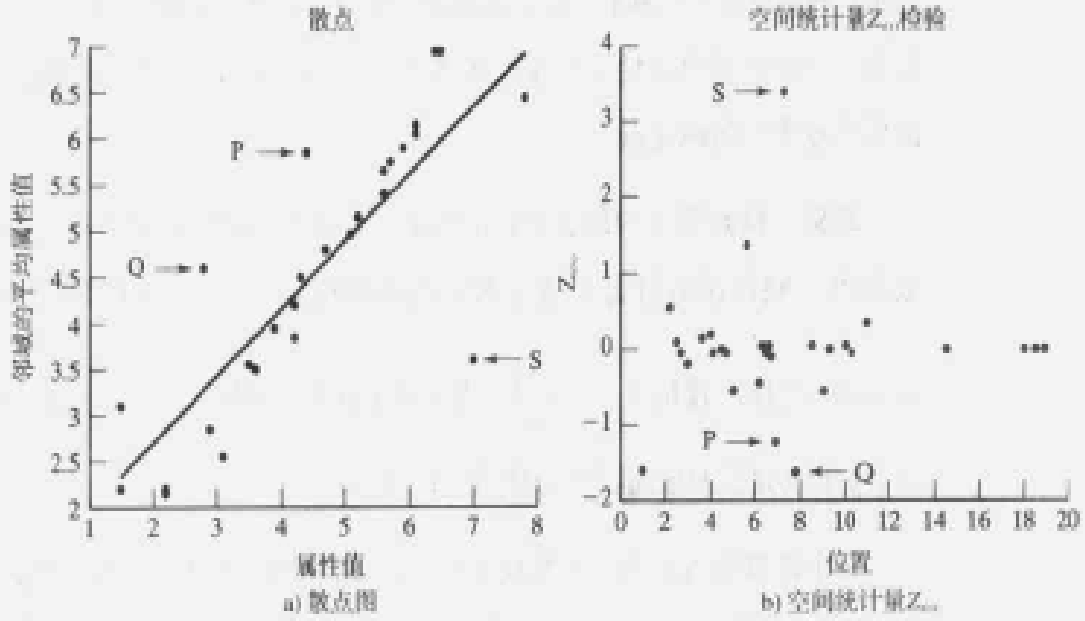


图7-18 检测空间孤立点的散点图和空间统计量Z(d)

图7-20中给出三种我们所要考虑的邻域定义。我们考虑时空邻域，是因为把时间和空间一起使用对于发现空间孤立点非常关键。在图7-20中，如果空间图中的s_1和s_3连接到s_2，则$\{s_1, t_1\}$和$\{s_3, t_1\}$是$\{s_2, t_2\}$的空间邻居。如果t_1、t_2、t_3是连续时间段，则数据点$\{s_2, t_3\}$是$\{s_2, t_2\}$的时序邻居。另外，基于时空序列，我们将邻域定义为一个时空邻域。在图7-20中，如果空间图中的s_1和s_3连接到s_2，并且t_1、t_2、t_3是连续的时间段，则(s_1, t_1)、(s_2, t_2)、(s_3, t_3)、(s_2, t_3)是(s_2, t_2)的时空邻域。

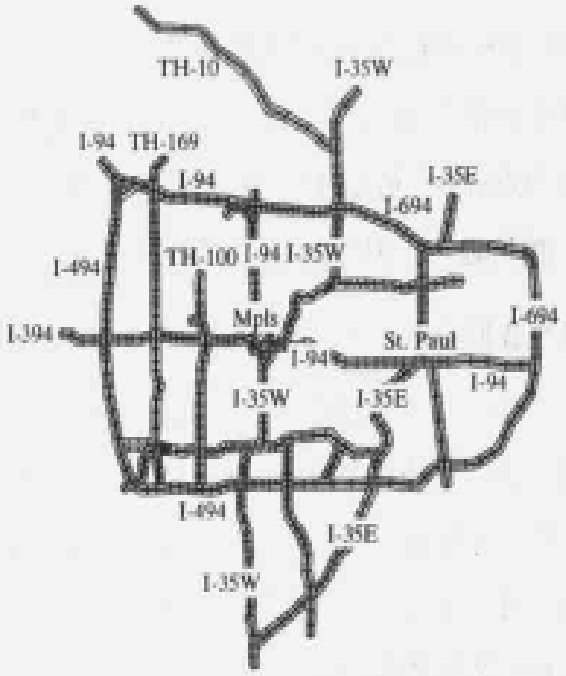


图7-19 交通传感器装置构成的网络

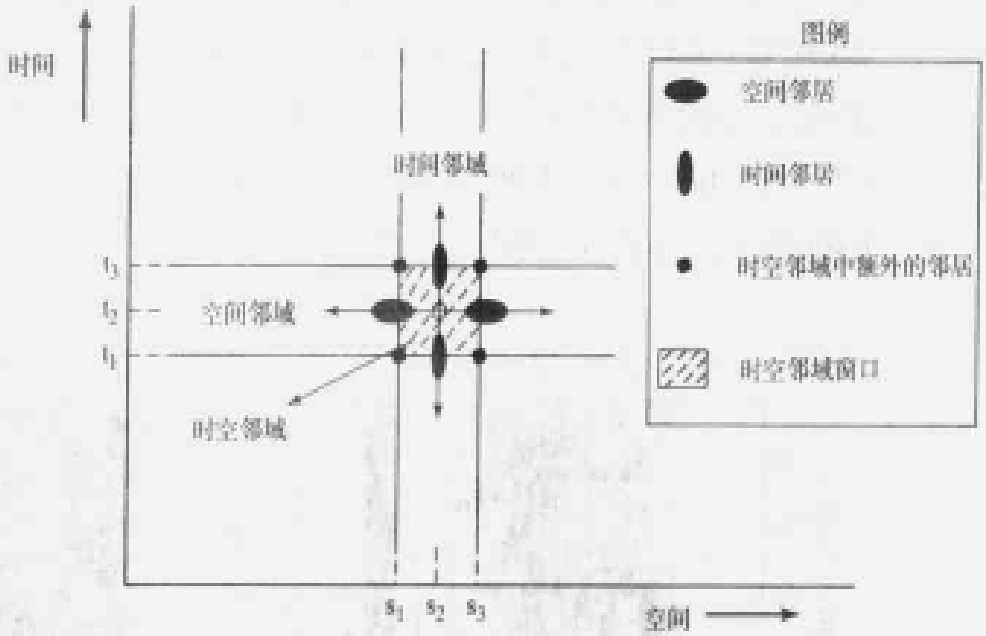


图7-20 空间和时间邻域

下一个需要考虑的任务是选择检验统计量来探测孤立点。在这个应用中，我们使用$S(x)=[f(x)-E_{w(x)}(f(y))]$，这里$f(x)$是$x$的属性值（如容量）。如果$f(x)$符合正态分布，则可以证明$S(x)$也符合正态分布（见练习）。如果$Z$分值（z-score）满足如下条件，则认为数据点$x$是一个孤立点。

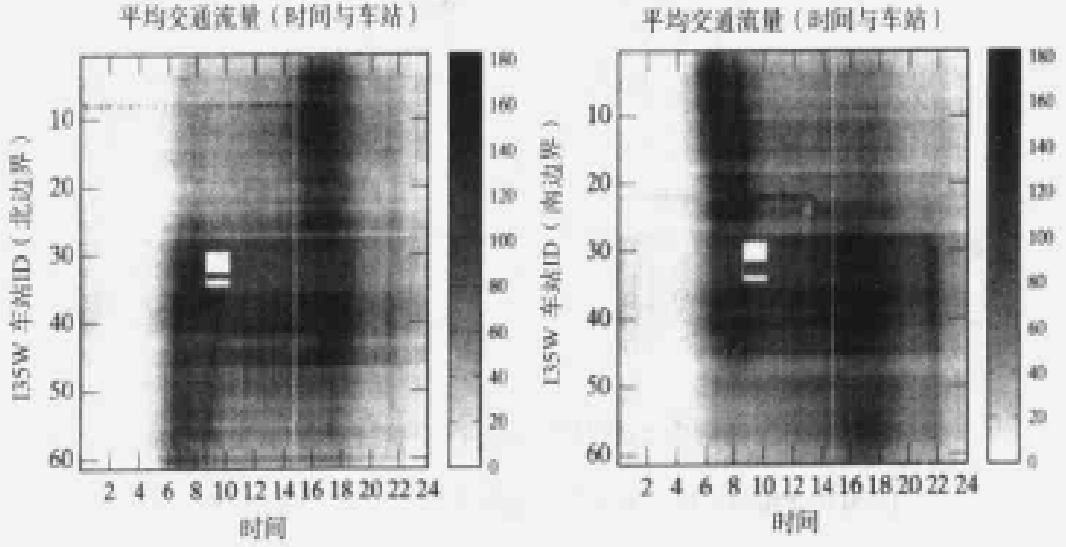
$$\frac{S(x)-\mu_s}{\sigma_s}>\theta$$

θ的选择和指定的置信度区间有关。例如，置信度区间为95%，则$\theta=2$。

孤立点检测的第三个也是最后一个任务是，设计和应用“高效”算法来计算检验统计量并应用孤立点探测检验。这是一个重大的任务，因为常见的数据集太大，不能全都放置在主存中。例如，在交通数据中大约有1000个传感器，它们每五分钟发出一个读操作。对于六个月的时间帧，假设每个读操作产生100字节的数据，则数据集的大小近似为$100\times12\times24\times180\times1000=5$GB。因此，使用高效I/O的算法来发现孤立点就变得非常迫切了。

以下的例子中说明$Z_{w(x)}$方法在双子城交通数据集上的有效性。图7-21给出一个交通流量孤立点的例子。图7-21a和b分别是1997年1月21日I-35W北边界和南边界的交通流量图。X轴显示一整天的5分钟时间段，Y轴是安装在高速公路上的传感器标号，从北

端的1号装置到南端的61号装置。从图a和b中可以很容易地观察到下午2点45分的异常白线和X轴上上午8点20分到上午10点以Y轴上站点29到34所围成的白色矩形。下午2点45分的异常白线是时间孤立点，而白色矩形是时空孤立点。二者都表现出缺失数据。另外，图7-21a中装置9与其邻近站点比较，出现不一致的交通流量，故该装置被检测为空间孤立点。装置9可能是失效的传感器。



a) I-35W北边界　　b) I-35W南边界

图7-21　交通流量数据中的空间孤立点

7.7 小结

数据挖掘是一个发展迅猛的领域，是数据库管理、统计学、人工智能等领域的交叉学科。数据挖掘提供半自动化的技术来挖掘海量数据中的未知模式。

空间数据挖掘是数据挖掘中用于快速分析空间数据的一个很好的研究领域。空间数据挖掘对主要的科学挑战（包括全球气候变化和基因组的研究）都有潜在的影响。

地理学第一法则清楚地概括出空间数据挖掘的独特特征：所有事物相互间都有关系，而距离近的事物之间的相互关系比距离远的事物更紧密。这说明经典数据挖掘的独立相等分布（independence and identically distributed, IID）随机变量的标准假设对于空间数据的挖掘并不适用。空间统计学家造就出空间自相关性这个词来捕获空间数据的这种性质。

在数据挖掘中，重要的技术包括：关联规则、聚类、分类和回归。在用于挖掘空间数据之前，这些技术都需要进行适当的修改。总的来说，有两种修改方法，可以使这些技术适用于空间数据。一种方法是修正基于idd假设的基本统计模型，另一种方法是修改驱动搜索的目标函数来包括空间项。空间自回归的回归技术是第一种方法的一个实例，NEM算法是后者的一个实例。

7.8 附录：贝叶斯演算

概率理论提供了一种处理不确定性的数学框架，它也是数据挖掘的基石。因为在数据挖掘中，我们试图基于一个虽是大型、但有限的数据库来概括我们的发现。

给定一组事件Ω，概率P是满足如下两个定理的从定义域Ω到值域[0, 1]的一个函数：

1) $P(\Omega)=1$。

2) 如果A和B是相斥事件（如两粒骰子的投掷），则

$$P(AB) = P(A)P(B)$$

7.8.1 条件概率

条件概率的概念是数据挖掘的核心概念。条件概率$P(A|B) = \alpha$表示在事件B发生情况下，事件A发生的概率是α。因此，如果事件B发生并且其他任何事件与A无关，则$P(A) = \alpha$。

概率演算的基本规则如下：

$$P(AB) = P(A|B)P(B) = P(B|A)P(A)$$

简单地说，这个方程表明联合概率$P(AB)$是条件概率$P(A|B)$与边缘概率$P(B)$之积。该规则经过简单处理就生成贝叶斯定理：

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

在贝叶斯规则的语境中，$P(A|B)$称为*后验概率*，$P(B)$称为*先验概率*。贝叶斯规则允许概率反演（inversion），它是分类的基础。例如，可以根据利用训练数据上

计算得到的概率来计算测试集数据的概率。

7.8.2 最大似然

假设已知一个符合正态分布$N(\theta)$的随机变量A，$\theta=(\mu, \sigma)$是分布的均值和标准差。概率理论的目标是研究假设θ固定时样本空间中A的或然性。在统计学中，我们使用贝叶斯定理转化问题，研究A固定时参数θ的或然性。正态分布（或任何其他分布）$N(A, \theta)$（作为θ而不是x的函数）是θ的似然函数。然后我们尝试选择θ，该θ有产生数据A的最大似然。这个观点将统计问题与微分学联系起来。

7.9 参考书目

7.1 数据挖掘是一个发展迅猛的领域，是数据库管理、人工智能和统计学等领域的交叉学科。空间数据挖掘是数据挖掘中一个重要且不断发展的领域。[Han et al., 1997]设计了一个空间数据挖掘的系统原型。

7.2 Darr和Stubble湿地的数据由[Ozesmi和Ozesmi, 1999]收集。经典数据挖掘技术，如对数回归和神经网络，可应用于预测鸟巢位置。

空间统计学家研究空间数据已经很多年了。他们构造了一系列术语（如空间自相关性和空间异质）来表示空间数据的独特特性。在[Cressie, 1993]中可以找到一些空间统计方法很好的思想。

7.3 经典数据挖掘技术的概述参见[Han和Kamber, 2000]。空间回归在[Anselin, 1988; LeSage, 1997]中有详细讨论。LeSage还在网上提供了一个很好的可用于不同空间回归模型的MATLAB工具箱。

7.4 [Agrawal和Srikant, 1994]引入了*Apriori*算法。[Koperski和Han, 1995]实现了已知最早的空间数据扩展。

7.4.3 有关空间同位模式的更详细的讨论可参见[Shekhar和Huang, 2001]。

7.4.5 在[Shekhar等, 2002]中可以找到经典方法（回归方法、贝叶斯方法）的详细描述及其空间扩展（SAR、MRF）。这篇论文基于概率和实验的框架对这些模型进行了比较。

7.5 我们关于*K-medoid*算法的讨论参考了[Estivill-Castro和Murray, 1998]。

EM算法可参考[Dempster et al., 1977]，其对于空间数据的扩展则可参考[Ambroise et al., 1997]。[Ordonez和Cereghini, 2000]给出了EM算法的高效SQL实现以在大型数据库中进行聚类。对于基于小波变换的空间聚类参见[Sheikholeslami et al., 1998]。[Wang et al., 1997]讨论了一个层次统计信息的基于网格的方法用于聚类和面向区域的查询。

7.10 习题

1. 考虑如下关于不同城市中娱乐设施的数据库（见表7-7）。

表7-7 设施数据库

CityID	Facilities
1	a,b,e
2	b,c,d
3	c,e
4	d,c
5	d,e
6	a,c,e
7	a,b,c,e
8	a,b,c,d,e

(a) 计算项集$\{a, b\}$、$\{c\}$和$\{a, b, c\}$的支持度。

(b) 计算关联规则$\{a, b\}\rightarrow\{c\}$的置信度。

(c) 计算关联规则$\{c\}\rightarrow\{a, b\}$的置信度。

(d) 为何置信度不对称而支持度对称？

(e) 从下表中抽取支持度超过30%、置信度超过70%的空间关联规则。X表示数据库中的湖泊（湖泊总数为100）。对于每个规则，写出其支持度和置信度。

空间谓词	计　数
near(X,forest)	45
inside(X,state_park)	90
adjacent(X,federal_land)	50
near(X,forest) and inside(X,state_park)	30
near(X,forest) and adjacent(X,federal_land)	20
near(X,forest) and inside(X,state_park) and adjacent(X,federal_land)	10

规　则	支持度	置信度
lake(X)⇒near(forest)		
lake(X) and inside(X,state_park)⇒near(X,forest_land)		

（续）

规　则	支 持 度	置 信 度
lake(X) and inside(X,state_park)⇒adjacent(X,federal_land)		
lake(X) and inside(X,state_park) and adjacent(X,federal_land)⇒near(forest)		

(f) 利用*K-medoid*算法计算$J(M_{i+1})-J(M_i)$时，为何仅需要将$M_{i+1}=M_i\cup\{m_n\}-\{m_o\}$中的非中心点$m_o$（旧中心点）和$m_n$（新中心点）从主存中取出？

提示 所有的非中心点满足以下四种情况之一：

(i) $p\ni C_{m_o}\wedge\exists m\in M_i$，使得$d(p,m)<d(p,m_o)\Rightarrow p\in C(m)\ in\ M_{i+1}$。

(ii) $p\ni C_{m_o}\wedge\forall m\in M_i$，$d(p,m)<d(p,m_o)\Rightarrow p\in C(m_o)\ in\ M_{i+1}$。

(iii) $p\ni C_{m_o}\wedge\exists m_1\in M_i$，使得$d(p,m_1)<d(p,m_n)\Rightarrow p\in C(m_1)\ in\ M_{i+1}$。

(iv) $p\ni C_{m_o}\wedge\exists m_1\in M_i$，使得$d(p,m_1)>d(p,m_n)\Rightarrow p\in C(m_n)\ in\ M_{i+1}$。

(g) 假设所有的聚类大小相同，利用以上的方法会得到什么样的性能改善？

2. 考虑具有N个特征和T个事务的数据集，可以枚举出多少不同的关联，可以发现出多少个不同的关联规则？

3. 对于以下情况应该使用哪种数据挖掘技术？

(a) 天文学家希望确定天空中一个未知对象是否是某个种类的星系（如，双跨排架星系）。

(b) 气象学家希望预测感恩节的天气（温度和降雨）。

(c) 设计购物中心的城市规划人员想知道人们会同时光顾哪些类别的商店。

(d) 政策分析人员希望根据过去二十年的投票历史对城市进行分组。

(e) 为了规划警察巡逻的路线，公安部门希望标识出城市地图上的热点区域。

(f) 流行病学家希望预测蓝尼罗河病毒的扩散和运动状况。

(g) 医生希望确定空间位置对癌症发病率是否有影响。

(h) 自然资源规划人员希望使用遥感图像估计松树林分的总面积。

(i) 法官希望根据新的人口普查数据重新划分国会选区。

4. 比较和对比以下概念：

(a) 关联规则与统计相关。

(b) 自相关与交叉相关。

(c) 分类与位置预测。

(d) 热点与聚类。

(e) 聚类与分类。

(f) 关联与聚类。

5. 考虑以下9个点的集合：(0, 0), (0, 1), (1, 1), (1, 0), (2, 3), (5, 5), (5, 6), (6, 6), (6, 5)。

(a) 假设所有的点属于一个聚类。计算聚类的平均值和中心点。

(b) 作为聚类的最有代表性的点比较平均值和中心点。使用从代表点到聚类中所有其他点的平均距离作为比较的量度。

(c) 考虑这种情景：前四个点属于一个聚类，后四个点属于另一个聚类。为这些聚类计算代表点的平均值。余下的点(2, 3)应该分配给哪个聚类?

6. 相对于挖掘关系数据，空间数据挖掘有何特殊之处？将空间特征实体化作为经典数据挖掘算法或模型的输入是否合适?

7. 相对于统计学，空间统计学有何特殊之处?

8. 在以下空间特征中，哪个特征表现出正的空间自相关性？为什么？（有没有物理上的原因或科学上的原因？）

坡度、含水量、温度、土壤类型、人口密度、年降水量（雨水、雪）

9. 如下空间点函数属于正空间自相关、无空间自相关和负空间自相关这三种类别的哪一种：

(a) $f(x, y) = 1$

(b) $f(x, y) = \begin{cases} 1, |x + y| \text{为奇数} \\ 0, \text{其他} \end{cases}$

(c) $f(x, y) = (x-x_0)^2 + (y-y_0)^2$

(d) $f(x, y)$是[0, 1]区间中的随机数

10. 讨论如下来自市场销售数据分析专家关于挖掘数值数据集的断言："如果谨慎地选择特征，则人们仅需要线性回归这一种数据挖掘技术就可以了"。

提示 特征选择可以简化模型。

11. 计算图7-14a所示的灰度图像的Moran's *I*和图7-6b、7-6c中的矩阵。

12. 比较以下概念：

(i) 空间孤立点与全局孤立点

(ii) SAR与MRF贝叶斯分类器

(iii) 同位与空间关联规则

(iv) 分类准确度与空间准确度

13. 选择出相关的空间特征之后，空间数据既可以通过定制方法（如，同位、空间孤立点检测、SAR）挖掘，也可以通过经典方法（如，关联规则、全局孤立点、回归）挖掘。比较这些方法，并说明你会在什么情形下使用哪种方法。

14. 空间孤立点检测计算可以建模为一个空间自连接查询。使用$Z_{(x)}$方法编写检测空间孤立点的SQL3或OGIS表达式。假设给定如下的关系表：

(a) neighbor(location, location)

(b) observation(sensor-id, location, value)

15. 通过*K-medoid*算法标识如下点集的两个聚类：

(i) (0,0)，(1,1)，(1,0)，(0,1)，(6,6)，(7,6)，(6,7)，(7,7)

(ii) (0,0)，(1,1)，(1,0)，(0,1)

16. 将频繁查询同时检索的记录聚类到共同的磁盘页面是空间数据库的一个重要存储问题。在本章中讨论的聚类方法能否对选择该问题适用的空间存储方法提供帮助?

17. 定义规模依赖（scale-dependent）和规模独立（scale-independent）的空间模式，并举例说明。

18. 研究7.3.3和7.3.4节中讨论的空间准确度的概念。根据预测的鸟巢位置和实际的鸟巢位置的地图，提出空间准确度的度量。分类算法如何使用提出的度量?

19. 基于空间对象（例如，国家边界）的形状考虑空间模式。例子模式包括大多数常见或不常见的形状的标识。讨论如何形式化形状的概念并发现这种模式。

20. 考虑图7-12中的样本数据和同位模式。讨论从样本数据发现同位模式的计算方法。如何定义同位规则的条件概率？如何定义同位规则的支持度？（提示 考虑邻居关系上的空间连接。）

21. 考虑多个空间孤立点的检测问题。讨论如何扩展7.6节中讨论的技术。注意，利用7.6节讨论的某些检验方法，一个真正的空间孤立点可以使得其邻居被标识为空间孤立点。

第8章

空间数据库发展趋势

本章我们讨论一些新出现的与空间信息系统相关的数据库问题，重点介绍栅格数据库管理、基于内容的检索和空间数据仓库。

卫星遥感影像以及对地形图和专题地图扫描产生大量的栅格数据，我们要求空间数据库管理系统（SDBMS）能有效地支持场（field）模型实体。数据仓库处理大量从可操作的和遗留的系统获得的数据。数据仓库负责清洗和转换这些数据，以便反映出其变化和趋势。这样的数据分析用于决策支持系统，而决策支持系统对于许多组织机构来说都是至关重要的。在这些数据仓库中，由于数据的历史特性，其数量非常巨大。这种数据通常是作为多维数据进行建模和处理的。尽管数据仓库查询所处理的数据量很大，但查询响应时间必须很低才能支持交互的和迭代的数据分析，以便完成数据挖掘和发现过程。

8.1节介绍重要的栅格图像数据操作，还要介绍栅格数据的存储和索引模式。8.2节讨论基于内容的检索（content-based retrieval, CBR）。8.3节介绍应用于空间数据的数据仓库术语和概念。

8.1 支持场实体的数据库

空间数据库管理系统必须支持建模为场的空间实体，这样的实体包括温度、高程、降雨量和气压等。这些实体的一个基本共性是它们都是“连续的”（continuous），即如果两个点在空间上彼此靠近，那么这两个点的场值也是相近的。

理论上，一个场模型可以通过求场函数的逆变换为一个对象模型。例如，如果f是温度场，那么所有温度超过40°C的区域对象可以描述为

$$f^{-}(x) = \{x \in Domain \mid f(x) \geq 40\}$$

从计算的观点来看，计算（非线性的）函数的逆函数是极其困难的，更实用的方法是直接处理场实体。

然而，在数据库中怎样表示一个场函数呢？场的连续性的一个重要推论是：可以用一个相对较小的平均场值的采样代表整个函数。例如，在图8-1所示的场函数中，其网格（grid）上的每个单元格（cell）指定一个颜色（或灰度），这是根据该单元格内有限采样点的平均值而确定的。场函数的连续性确保采样均值的分布可以很好地表示原函数。这样，就可以利用代表值的矩阵来表示场函数。在GIS中，这样的矩阵称为栅格（raster），而栅格的每个单元格则称为像素（pixel），像素是图像元素（picture element）的缩写。

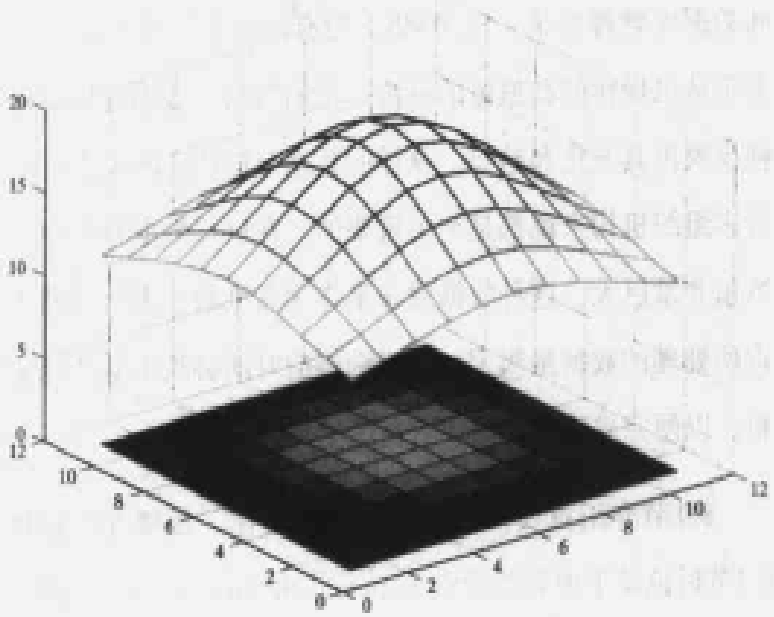


图8-1 一个连续函数及其栅格表示

数据库必须要支持栅格数据的另一个原因是，卫星采集的数据、航空像片、数字高程地图（DEM）已经是栅格形式的了。

8.1.1 栅格与图像操作

地图代数（map algebra）是对栅格分析所做的大量操作进行组织的一种系统框架，最先由Tomlin [Tomlin，1990]提出。它最初是为地图绘制团体而开发的，但现在已经演化为一种卓越的处理栅格的语言，许多GIS商业软件产品都支持某种形式的地图代数。但要记住，地图代数是一个用于栅格分析而不是栅格查询的框架。人们熟知的图像代数（image algebra）与地图代数类似，但更正式一些，它是图像处理操作的集合。后来，有人尝试借用图像代数的技术对地图代数作形式化和功能的扩展，便出现了所谓的地理代数（Geo-Algebra）。

代数是一种数学结构，包括两个不同元素集合：（Ω_o，Ω_o）。Ω_o是操作数（operand）的集合，Ω_o是操作（operation）的集合。代数需要满足许多公理，但其中最重要的是闭合性，即对操作数进行操作的结果必定在Ω_o内。一个简单的代数的例子是具有加法和乘法操作的自然数集合。

在地图代数中，操作数是栅格矩阵，而操作可以分为4类：局部的（local）、聚焦的（focal）、区域的（zonal）和全局的（global）。下面将对这几类操作分别进行描述。

1. 局部操作

局部操作将一个栅格映射到另一个栅格上，以便新栅格中每个单元格取值仅依赖于它在原栅格中单元格的值。图8-2是一个局部操作的例子。

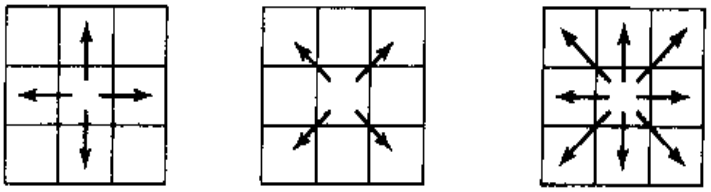
图8-2 局部操作的示例：阈值化

根据原栅格中单元格的值低于（或高于）用户定义的某个值，给栅格的每个单

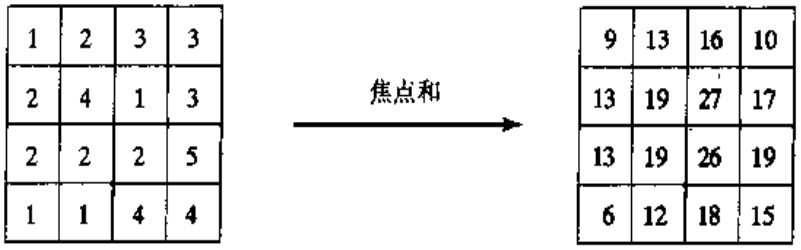
元格赋值为0或1。在图8-2中，值小于3的所有单元格设为0，而值大于等于3的单元格的值为1。这种操作就是所熟知的阈值化（thresholding）。例如在DEM中，阈值化操作可以用于识别超过某个高度的单元格。

2. 聚焦操作

聚焦操作中，新栅格单元格的值依赖于原栅格中相应单元格以及邻近单元格的值。如图8-3a所示，构成单元格的邻域通常有三种定义。若用国际象棋来比喻这些邻居，则这些邻居可称为车、象和后。在图8-3b中，新栅格的一个单元格值是原栅格中“后”的邻居的所有单元格的值的总和。聚焦操作实例包括场的坡度、梯度和连续性等。图8-3b给出了一种聚焦操作，即“后”的邻居的聚焦求和。



a) 车位、象位和后位邻域



b) 一个焦点操作的例子：焦点和

图8-3 单元格的邻域和一个聚焦操作例子：聚焦求和

3. 区域操作

在区域操作中，新栅格中单元格的值是原图层中相应单元格的值以及其他单元格的值的一个函数，它们出现在由另一个栅格指定的同一个区域上。如图8-4所示，新栅格中左上角单元格的值（12）是原栅格中由区域图指定的A区域中所有单元格的值的总和。这个信息也可以用一个表来表示，每个区域对应表的一行，该行列出该区域中所有单元格的总和。如上例所示，区域不一定是连续的，也无需覆盖输入地图的所有单元格。

4. 全局操作

在全局操作中，新栅格中单元格的值是位置的函数，或者是原栅格或其他栅格

上所有单元格的值的函数。图8-5是一个全局操作的例子。原栅格给出了源*S1*和*S2*的位置，新栅格的中每个单元格记录它到最近一个源的距离。水平和垂直方向上的相邻单元格间隔一个距离单位，而在对角线上的两个相邻单元格的距离为 $\sqrt{2}$ 倍单位距离。

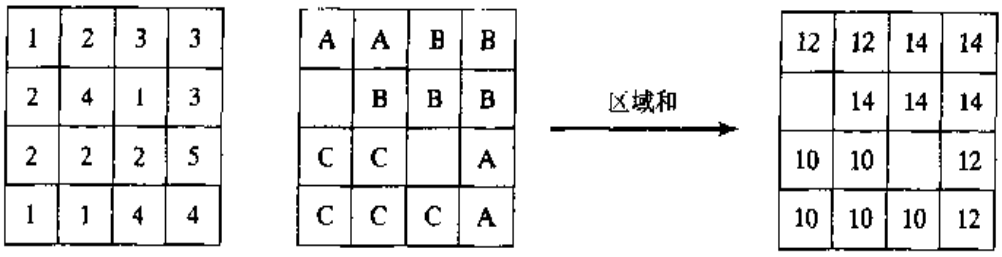


图8-4 区域操作示例：区域求和

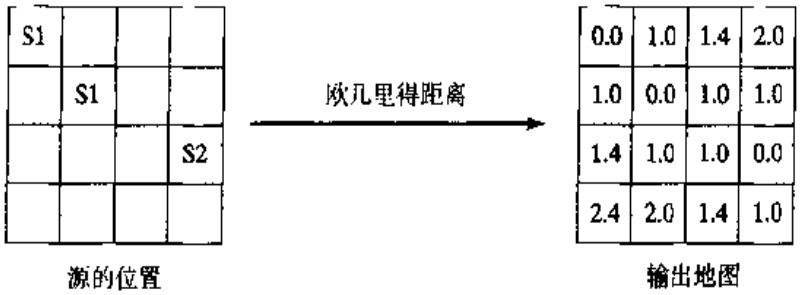


图8-5 全局操作示例

5. 图像操作

除了上面介绍的四类操作外，还有另外两个专用于图像处理的操作，这两个操作是剪裁（trim）和切片（slice）操作。剪裁操作沿坐标轴提取原栅格的一个子集，如图8-6所示。切片操作专用于医学图像领域，它把一个高维栅格图像投影为一个或一系列低维的栅格图像。如图8-7的例子所示，一个三维栅格图像沿Z轴方向投影为三个二维栅格图像构成的序列。

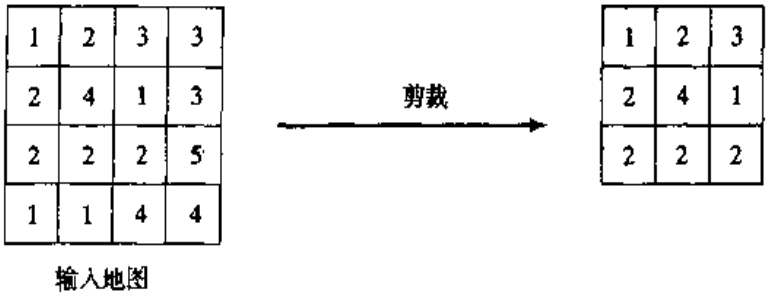


图8-6 剪裁操作：保留维数

8.1.2 存储和索引

对于查询和其他数据库的支持，把栅格图像存储在数据库中而不是作为文件系统中的文件是很重要的。由于栅格图像比其他属性占用的磁盘块空间分片更多，因此必须为栅格图像设计专门的存储策略。例如，如果像存储其他属性那样存储栅格图像，就会极大增加在关系表中查找元组的时间，因为磁盘块因子（每个磁盘块的记录数）过小。

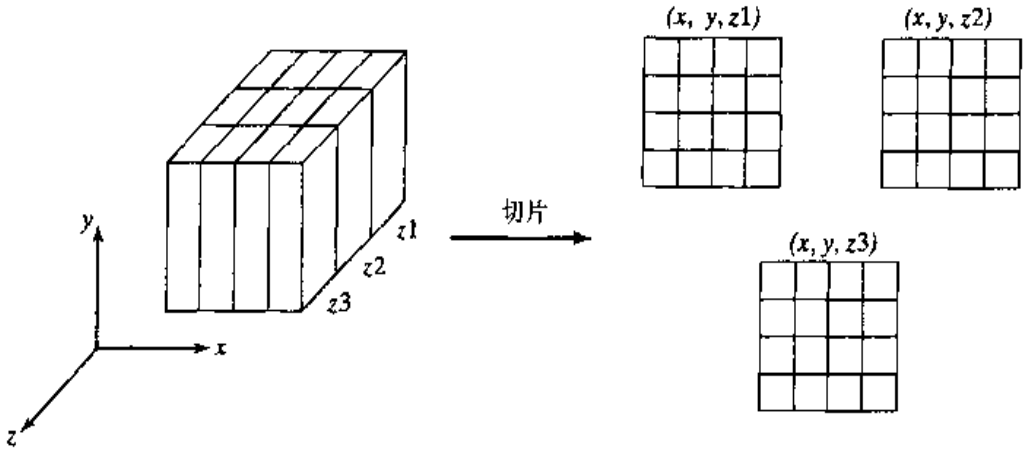


图8-7 切片操作：降低维数

处理此问题的一种策略是只把有关栅格阵列的重要信息连同其他属性一起存储在一个表中，而把实际的原始栅格图像存储到另外一个表中。为了访问原始的栅格图像，第一个表要包含指向存储图像的表的外码。

对于非常大的栅格阵列来说，一种有效的策略是把阵列分解成若干数据片，然后将这些数据片作为独立的元组分别存储到表中。这种策略对于那些只涉及部分原始数据的查询操作特别有效，如剪裁操作。

例如，考虑图8-8a所示的栅格图像，如果查询区域A想要访问阵列的一部分，那么，将栅格图像划分到磁盘页的方法不同会产生截然不同的结果。假定栅格图像是一个16 000×16 000大小的阵列，一个像素需要一个字节的存储空间，那么磁盘页面大小为16Kb=16 000字节，而查询区域A的大小为700×800。

1. 线性分配

在这个默认的策略中，阵列的每一行分别对应一个磁盘页面，也就是说，原阵

列被分成大小为1 × 16 000的子阵列。为了检索那些与查询区域A交叠的元素，需要进行700次磁盘访问，因为查询区域A相当于35个磁盘页面，显然这样的代价太高了。

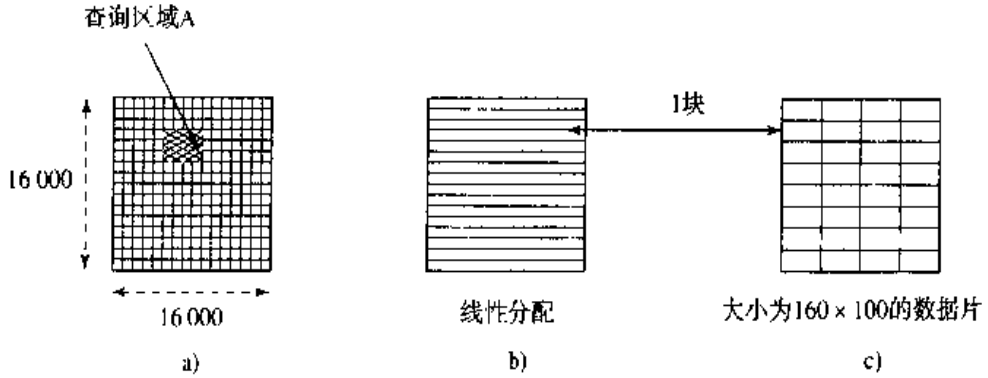


图8-8 存储矩阵的不同策略

2. 分片

在这种策略中，将阵列分成大小为160 × 100的子阵列。因此，假定查询区域恰好与数据分片策略一致，则磁盘页面的访问次数为5 × 8 = 40。因而就这个特定查询来说，这种分片策略大大降低了页面访问数量。显然，不同的查询范围需要不同的最优分片策略，分片策略和参数可以通过跟踪查询的统计来进行调整。此外，还可以在数据片上建立R树索引以更快地访问与查询区域匹配的数据片。

另一种策略建立在数据压缩的基础上。在一个常见地理区域上的一组图像可以用一组更小的正交图像来近似表示，各个图像可以采用诸如游程编码、向量量化和分波段编码等技术进行压缩。压缩可以减少存储一组图像所需的磁盘空间。

8.2 基于内容的检索

从数据库的观点来看，地图代数操作由于处理的是*数据分析*而不是*数据查询*，所以只能处于次要地位。数据库的关键挑战是从数据库中高效地检索出那些满足用户查询指定谓词的图像。搜索谓词可以基于元数据或内容。元数据就是使用简单数据类型（例如数字、文本）对一幅图像所做的描述，例如，每个栅格的数据项中通常有坐标值（纬度，经度）、地名、日期、传感器类型等信息。可以使用基于元数据的谓词来检索图像，处理这类查询可以采用在第5章中讨论的技术。相反，一幅图像的内容是指完整的图像语义，这远远超过了元数据中记录的信息所能表达的范围，例如图像中对象的颜色、纹理以及空间配置等。考虑这样一个搜索谓词，它要

求找出所有与一幅给定（例子）图像相似的图像。给定图像中有一条沿湖泊并穿越城区的高速公路。这类查询通常称为基于内容的检索，因为查询的结果取决于隐含在图像中的内容和关系。下面列举几个有重要意义的基于内容查询的例子：

1) 找出所有背景为海洋，前景为海滩的图像。

2) 定位一幅位于明尼苏达州，并且附近有树木的河流的图像。

3) 找出距芝加哥方圆100英里以内，并位于芝加哥西南方的包含湖泊的所有州立公园的图像。

4) 找出形状类似于女士皮靴或泪滴的国家。

5) 找出这样的一对分子或大陆，它们的形状刚好可以拼在一起，就像七巧板那样。

其中第三个查询对于空间数据库来说非常重要，因为它涉及到所有三类空间谓词：即topological的（湖泊被森林包含）、directional（芝加哥的西南方）和metric（距离芝加哥方圆100英里以内）。最后两个查询是基于形状的，OGIS的数据模型尚不能对它建模。在空间约束基础上查询图像数据库的能力显然是一项重要的任务，也是当前迫切需要研究的一个课题。

基于内容检索的通用方法由五个步骤组成，如图8-13所示。这里我们主要讨论该方法所用到的一些基本概念。

8.2.1 拓扑相似性

拓扑关系已在第2章和第3章中详细讨论过了。在那两章中，我们指出拓扑关系可以用Egenhofer的九交模型来表示，九交模型是第3章中提及的OGIS对SQL进行扩展的理论基础。对于多边形对象，可以直观地定义关于拓扑关系的“邻域”（neighborhood）概念。

如果T_1 (T_2)不经过任何一个中间拓扑关系就能连续变换成T_2 (T_1)，就说两个拓扑关系T_1与T_2是相邻的。这样Disjoint（相离）与Touch（相接）相邻，而Disjoint与Overlap（交叠）就不相邻，Disjoint首先要转为Touch，然后才能转换为Overlap。拓扑关系的相邻图如图8-9所示。形式上，两个关系T_1与T_2之间距离$D_{top}(T_1, T_2)$为：

$D_{top}(T_1, T_2)$等于相邻图上T_1和T_2之间最短路径中边的数目。

8.2.2 方位相似性

和拓扑关系的概念类似，可以定义方位（directional）关系，众所周知的8个方位如下：

{北（N），西北（NW），西（W），西南（SW），南（S），东南（SE），东（E），东北（NE）}

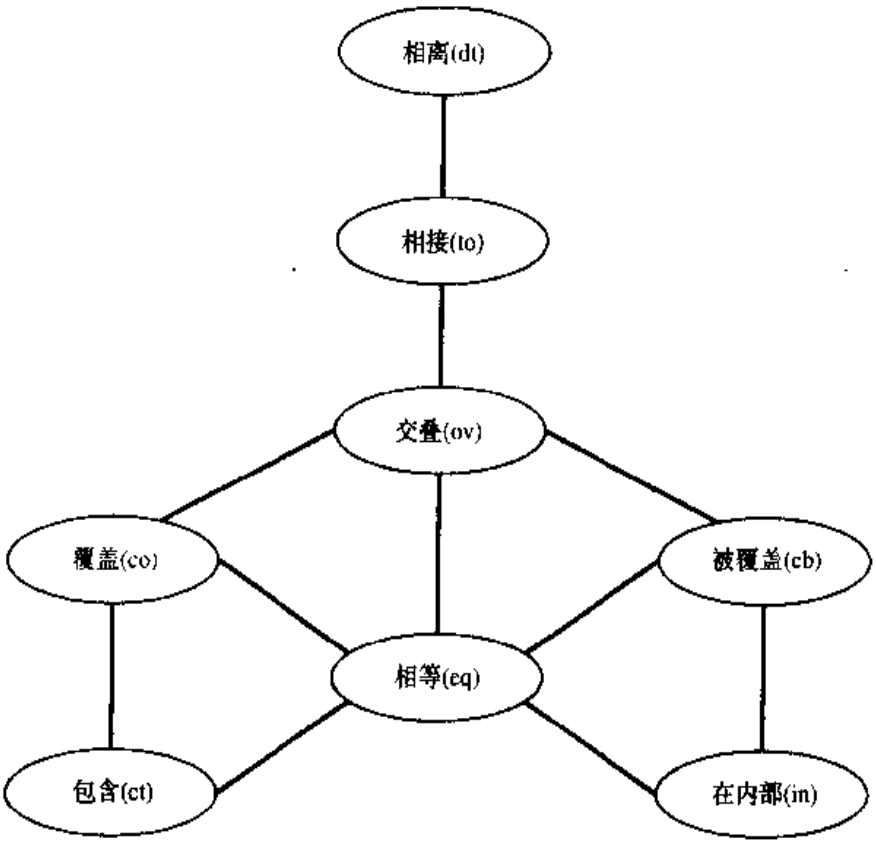


图8-9 拓扑相邻图 [Ang et al.，1998]。两个关系的距离是图中它们之间的最短路径

唯一不同的是，对于线和多边形这类扩展的对象，很难界定它们的方位关系，例如，西北到哪里结束，北又从哪里开始？对于基于内容的检索来说，更愿意采用一种方位关系的模糊概念方法。

其中一个方法是使用如图8-10所示的相邻图。图中的实线可以用来计算两个结点之间的距离，虚线则表示两个结点之间的距离为零。因此，两个方位关系D_1与D_2之间的距离$D_{dir}(D_1, D_2)$定义为：

$$D_{dir}(D_1, D_2)\begin{cases} \text{图上的最短路径} & \text{如果}D_1\text{和}D_2\text{用实线连接} \\ 0 & \text{如果}D_1\text{和}D_2\text{用虚线连接} \end{cases}$$

8.2.3 距离相似性

从某种意义上说，基于距离的相似性是三类空间关系（拓扑的、方位的和度量的）中最不重要的，在拓扑关系和方向关系保持不变的情况下更是如此。例如，观察图8-11所示的图像，图像P至少在视觉上与图像Q更相似而不是与图像R相似，尽管单就距离而言，P与R更接近。

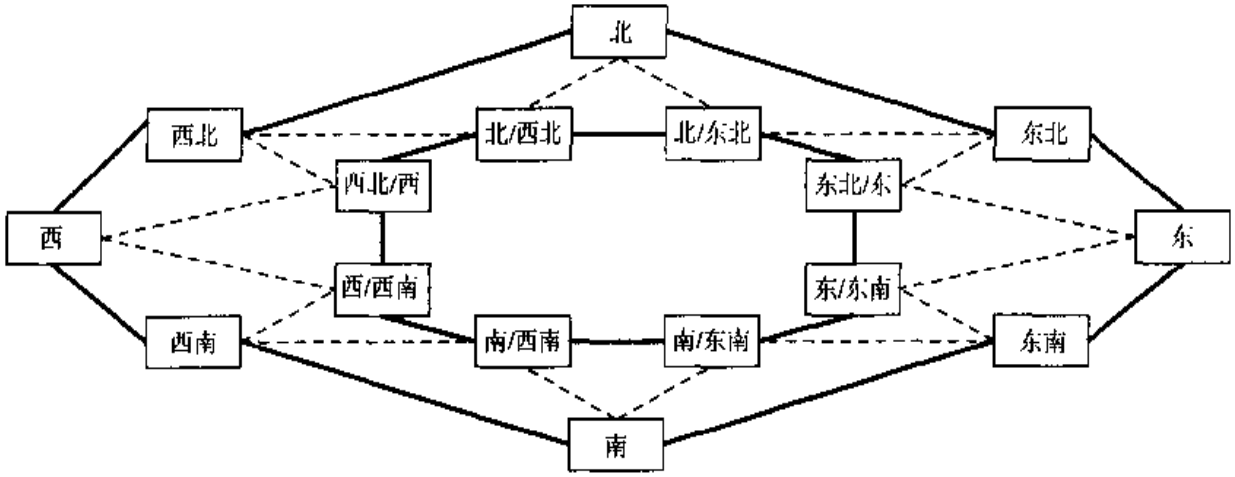


图8-10 方位邻近图[Ang et al., 1998]。两个关系的距离是图中它们之间的最短实线路径

对于距离相似性，我们将采用对象质心之间的标准欧氏直角距离。

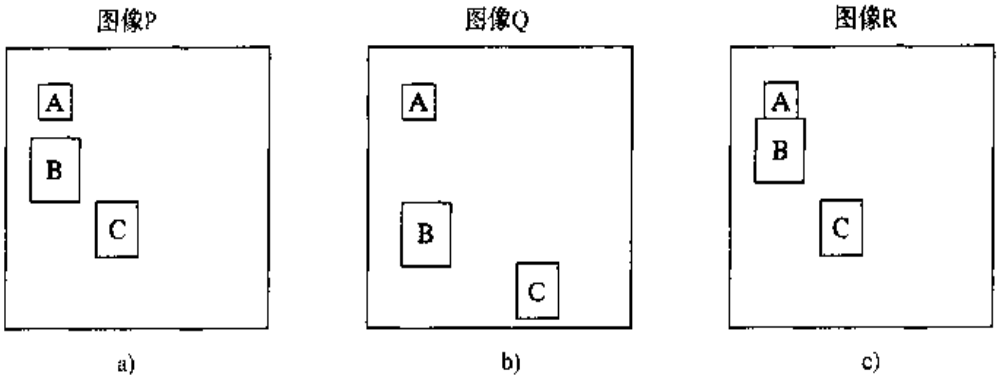


图8-11 视觉角度：将distance作为最小区分空间的谓词

8.2.4 属性关系图

属性关系图（attribute relational graph, ARG）是刻画对象及其关系信息的完

全连通图。参考图8-12所示的图像和它的ARG，ARG的结点标有对象的标号，两个结点之间的边标有两个结点间的关系信息。例如，结点O_1与O_2之间的边标有（D，61，5.2），这表明O_1与O_2之间的拓扑关系为相离（disjoint），它们之间的角度为61°（随下标递增顺序测量），它们之间的距离为5.2个单位。

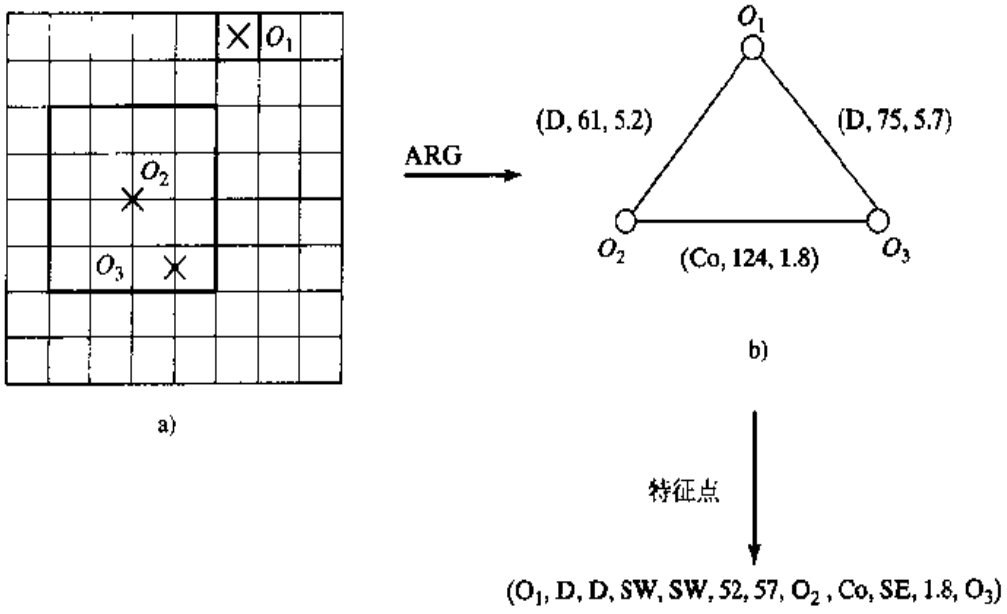


图8-12 图像和它的ARG[Petrakis and Faloutsos，1997]。
ARG被映射为一个N维的特征点

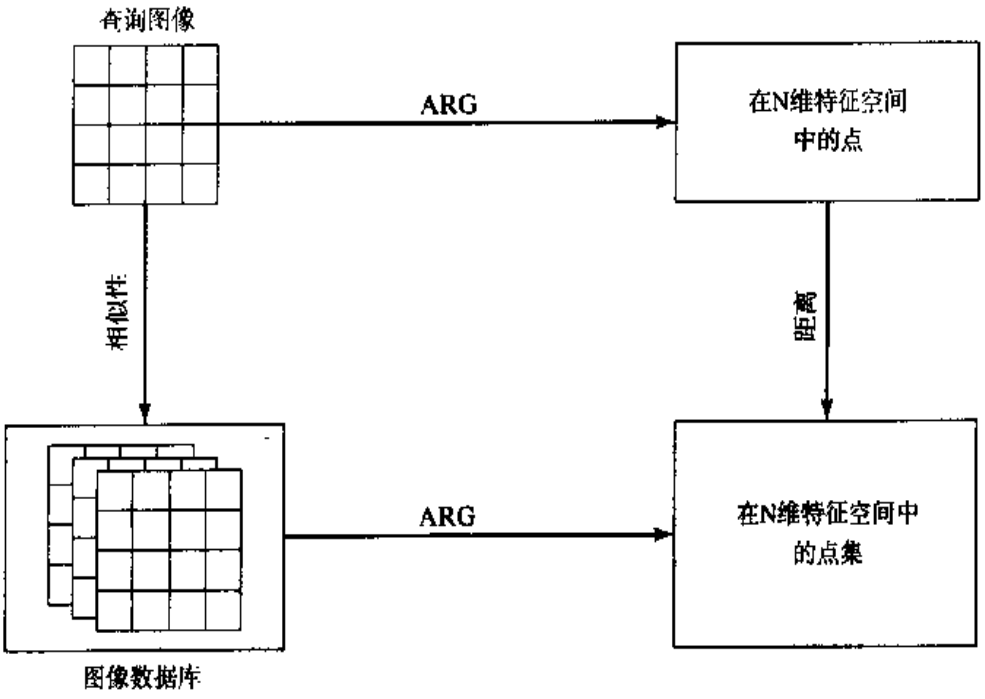


图8-13 基于内容检索的通用方法

为每幅图像创建ARG之后，ARG要映射到特征空间中的一个多维点。特征空间中的点按照某种预先指定的顺序进行组织：首先是第一个对象，其后是该对象与所有其他对象之间的关系；然后是第二个对象，以及第二个对象与随后所有对象之间的关系。在这个阶段，对象之间的方位角也转换为方位谓词。例如，O_1与O_2之间的角度61°就映射为西南方位（见图8-13）。

8.2.5 检索步骤

1) 将图像数据库中的所有图像映射为多维特征空间中的点，这种映射是通过首先在数据库中为每幅图像构建一个ARG来完成的。

2) 对应于图像间每种相似性准则来定义一个距离度量。相似性准则可以是图像中对象的形状、亮度、颜色、纹理和空间关系等。如果搜索是基于多个相似性准则来完成的，那么最终的距离度量是各个部分的聚集（例如，总和）。该任务通常是由领域专家来完成。

3) 建立一种聚集和索引多维特征点的存储方法（如R树）。

4) 将所查询的图像映射到特征空间中的一个点或一个区域，然后选取与查询点靠近或者位于查询区域范围内的点。这个步骤可能要用到最近邻居查找算法或者范围查询处理算法。

5) 返回与选择点所对应的图像作为结果。

8.3 空间数据仓库概述

数据仓库（data warehouse, DW）是一个决策支持技术的集合，以便帮助知识工作者（经理人、管理者、分析者等）作出更准确、更迅速的决策。数据仓库包含大量信息，这些信息来自各种独立数据源，通常分别由运行的数据库独立维护。通常，运行数据库最适于进行联机事务处理（OLTP），在这种情形下，一致性和可恢复性至关重要。事务一般很小并且只是根据主码访问少量的记录。运行数据库维护当前状态的信息。与此相反，数据仓库则要维护历史信息，它是为联机分析处理（OLAP）而设计的，OLAP需要查询大量的聚集数据来检测趋势和异常。维、度量和聚集层次是数据仓库中的核心概念。维用来定义语境的度量，可以将维看成是决定度量（因变量）的自变量（或键）。聚集层次有两种，即维度幂集层次

（dimension power-set hierarchy）和分维概念层次(per-dimension concept hierarchy)。为进行复杂分析，数据仓库中的数据通常被建模为多维数据立方体。例如，在人口普查的数据仓库中，年龄组、收入类型、人种和时间（年）都是所关心的维。数据立方体是一个在N维空间中泛化group-by SQL查询的聚集操作符。

空间数据仓库除了包含非空间数据外，还包含地理数据（如卫星图像、航空照片等）。空间数据仓库的例子有美国的人口普查数据集、EOS的卫星影像库、Sequoia 2000，以及公路交通测量资料。传统数据仓库与空间数据仓库之间的主要差别在于结果的可视化。传统数据仓库OLAP的结果通常以文本和数字的报表或电子表格的形式表现，而空间数据仓库的结果则可能是一幅地图。另一个区别与对几何数据类型（例如点、线、面）聚集操作符的选择和标准化有关。无论是现有的数据库还是新近的地理空间数据标准，都未讨论过这个问题。本节的后面部分将概述数据仓库的重要操作及其在空间语境中的对应操作。

8.3.1 聚集操作

数据立方体包含一些立方格，每个立方格代表层次的某一级别。聚集函数计算每个立方格内一组给定值的统计信息。具体的聚集函数有：求和（sum）、求均值（average）和求质心（centroid）。按照[Gray et al.，1997]的观点，聚集函数可以分为三类，即分布的（distributive）、代数的（algebraic）和整体的（holistic）。在本节里，我们来定义这些函数并举一些GIS领域的例子。表8-1列出了针对不同数据类型的所有聚集函数。

表8-1 聚集操作

数据类型	聚集函数		
	分布的函数	代数的函数	整体的函数
数值集合	Count、Min、Max、Sum	Average、Standard Deviation、MaxN、MinN	Median、MostFrequent、Rank
点、线、多边形集合	Convex Hull、Geometric Union、Geometric Intersection	Centroid、Center of mass、Center of gravity	Nearest neighbor index、Equi-partition

1. 分布的

如果存在这样的函数G，通过把G应用到N+1维立方格的F值上就可以计算出F

的一个用于N维（数据）立方格的值，此时称聚集函数F是分布的。例如，当N=1时，$F(M_{ij}) = G(F(C_j)) = G(F(R_i))$，其中$M_{ij}$代表一个二维矩阵中的元素，$C_j$指矩阵的每一列，$R_i$指矩阵的每一行。考虑图8-14所示的聚集函数Min和Count。在第一个例子里，F=Min，则G=Min，因为$Min(M_{ij}) = Min(Min(C_j)) = Min(Min(R_i))$。在第二个例子里，$F$=$Count$，$G$=$Sum$，因为$Count(M_{ij}) = Sum(Count(C_j)) = Sum(Count(R_i))$。其他的分布聚集函数还包括Max和Sum。注意，在计算聚集函数时要忽略空值元素。

分布的GIS聚集操作包括凸包（convex hull）、几何并（geometric union）和几何交（geometric intersection）。一个点集Q的凸包是一个最小的凸多边形P，此时Q中的每个点要么在P边界上要么在P内部。简单来说，凸包就是由围绕所有钉子拉紧的橡皮筋所构成的形状。几何交是个二元操作，它涉及两组几何区域，并返回由这两个区域共同覆盖的区域集合。集合并也是涉及两组几何区域的二元操作，返回至少被其中一个区域所覆盖的区域集合。对于所有这些聚集，运算符先聚集所计算的子集区域，然后计算最终的结果。

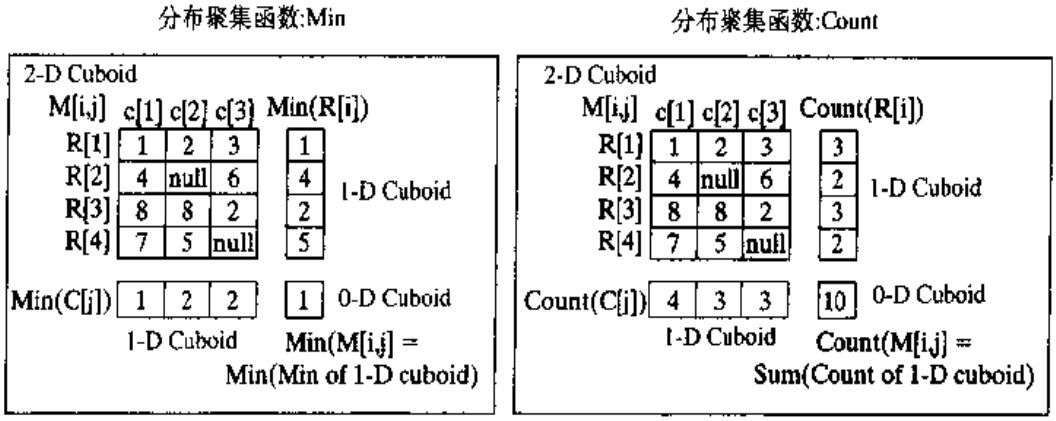


图8-14 分布聚集函数的计算

2. 代数的

如果N维立方格的F可以用一定数目的N+1维立方格的聚集来计算，就说聚集函数F是代数的。均值（average）、方差（variance）、标准差（standard deviation）、max$N$和min$N$等操作都是代数的。例如，图8-15给出了矩阵M的均值和方差计算。二维矩阵M中元素的均值可以通过求一维子立方体（例如，行或列）值的总和sum以及元素计数count而得到。方差可以从行或列的计数、求和（即$\sum_i X_i$）以及平方和（即$\sum_i X_i^2$）导出。类似的技术也可用于其他的代数函数。

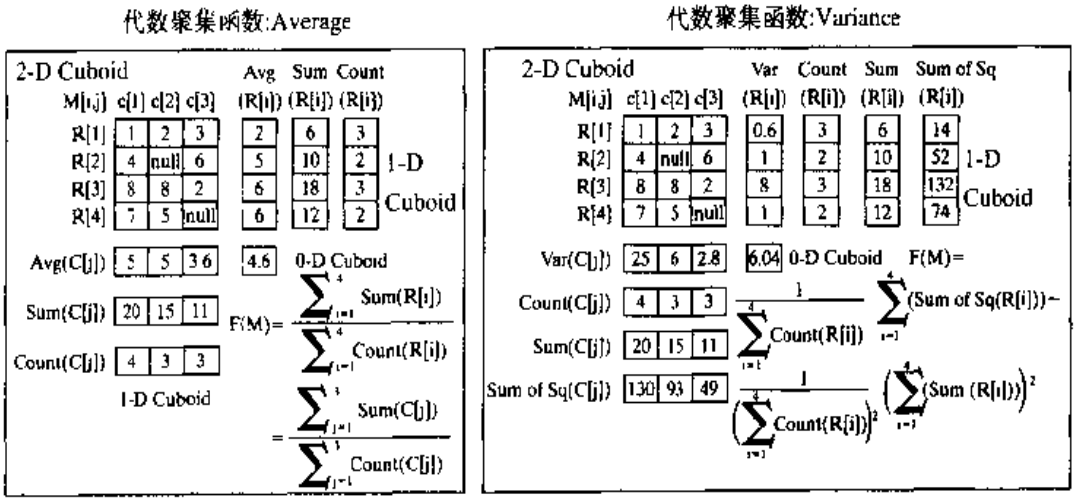


图8-15 代数聚集函数的计算

在GIS中，代数聚集操作是一个中心计算。n个几何点的中心 $\vec{V}'=(V_x',V_y')$ 可以定义为 $Center=\frac{1}{n}\sum\vec{V_i}$，$C_x=\frac{\sum V_{xi}}{n}$，$C_y=\frac{\sum V_{yi}}{n}$。中心（center）和个数（count）用来计算下一层的结果。代数聚集函数的其他例子还有质心（center of mass）和重心等。

3. 整体的

如果N维立方格的F值不能用常数个N+1维立方格的聚集来计算，那么聚集函数F是全局的。整体聚集函数的例子有Median（中值）、MostFrequent（最大频率）和Rank（秩）。

整体的GIS聚集操作包括均分（equipartition）和最近邻居指数（nearest-neighbor index）。点集的均分生成一条线L，并且L的两侧有相同数量的点对象。最近邻居指数用于度量某一个空间域中对象的聚类程度。如果一个空间域具有相似值趋于聚在一起的性质，那么该域的最近邻居指数就比较高。一旦添加了新数据，最近邻居关系中的许多元组都可能会改变，因此，最近邻居指数是整体的。均分线也可能因为新点的加入而改变。为了计算各个维层次上的均分或最近邻居指数，需要有基本的数据。

聚集函数的计算难度是分等级的。分布的聚集函数可以基于低一级维度的值来计算；代数的聚集函数也可以利用低一级维度值的聚集来计算。即使是对于占用大量磁盘存储空间的数据集，也可以使用少量的内存缓冲区计算分布的和代数的聚集函数，但整体的聚集函数在所有维层次上都需要基本数据来计算结果。

8.3.2 几何聚集的例子

图8-16给出了几何求并聚集操作的结果。基本表Election-Base有下列属性：State Name、Governor Political Party（GPP）、majority Legislator Political Party（LPP）、Boundary和Delegates。其中属性Boundary是一个外码，指向另外一个描述用几何多边形表示州界的表。根据基本表及其对应的地图，我们提出表8-2中列出的5个查询：GQ1、GQ2、GQ3、GQ4和GQ5。如果邻近的多边形有与GROUP-BY子句相同的属性值，则Geometric-Union-by-Continuous-Polygon操作符就会把这些多边形合并成一个大的多边形。这些查询的效果与GIS的地图再分类操作一样，例如，查询GQ1得到的结果与通过属性GPP和LPP对Base-Map（基本图）再分类相同的结果一样。图8-16的左边部分是这些查询对应的图形解释，区域$Q1$，$Q2$，...，$Q8$的边界可以从基图中较小区域$P1$，$P2$，...，$P12$的几何并中派生出来。例如，查询$Q2$表示$P2$和$P3$集合的几何并，因为区域$P2$和$P3$在分组属性集<LPP，GPP>上具有相同的值。同样，地图Map-L2-A中的区域$A1$、$A2$、$A3$和地图Map-L2-B中的区域$B1$、$B2$、$B3$可以从地图Map-L1（或Base-Map）中较小区域的几何并中派生出来。例如，地图Map-L2-A中的区域A_1是地图Map-L1中区域$Q1$和$Q4$的几何并集。

表8-2 地图再分类的SQL查询

GQ1	SELECT GPP, LPP, Geometric-Union-by-Continuous-Polygon (Boundary) FROM Election-Base **GROUP BY** GPP, LPP
GQ2	SELECT GPP, Geometric-Union-by-Continuous-Polygon (Boundary) FROM Election-L1 **GROUP BY** GPP
GQ3	SELECT LPP, Geometric-Union-by-Continuous-Polygon (Boundary) FROM Election-L1 **GROUP BY** LPP
GQ4	SELECT GPP, Geometric-Union-by-Continuous-Polygon (Boundary) FROM Election-Base **GROUP BY** GPP
GQ5	SELECT LPP, Geometric-Union-by-Continuous-Polygon (Boundary) FROM Election-Base **GROUP BY** LPP

8.3.3 聚集层次

CUBE操作符综合了直方图（histogram）、交叉表（cross-tabulation）、上卷（roll-up）、下钻（drill-down）和汇总（subtotal）等操作，它是简单聚集函数的N维泛化。图8-17所示的是在三个维上进行聚集的概念。这三个维分别为Year（年）、

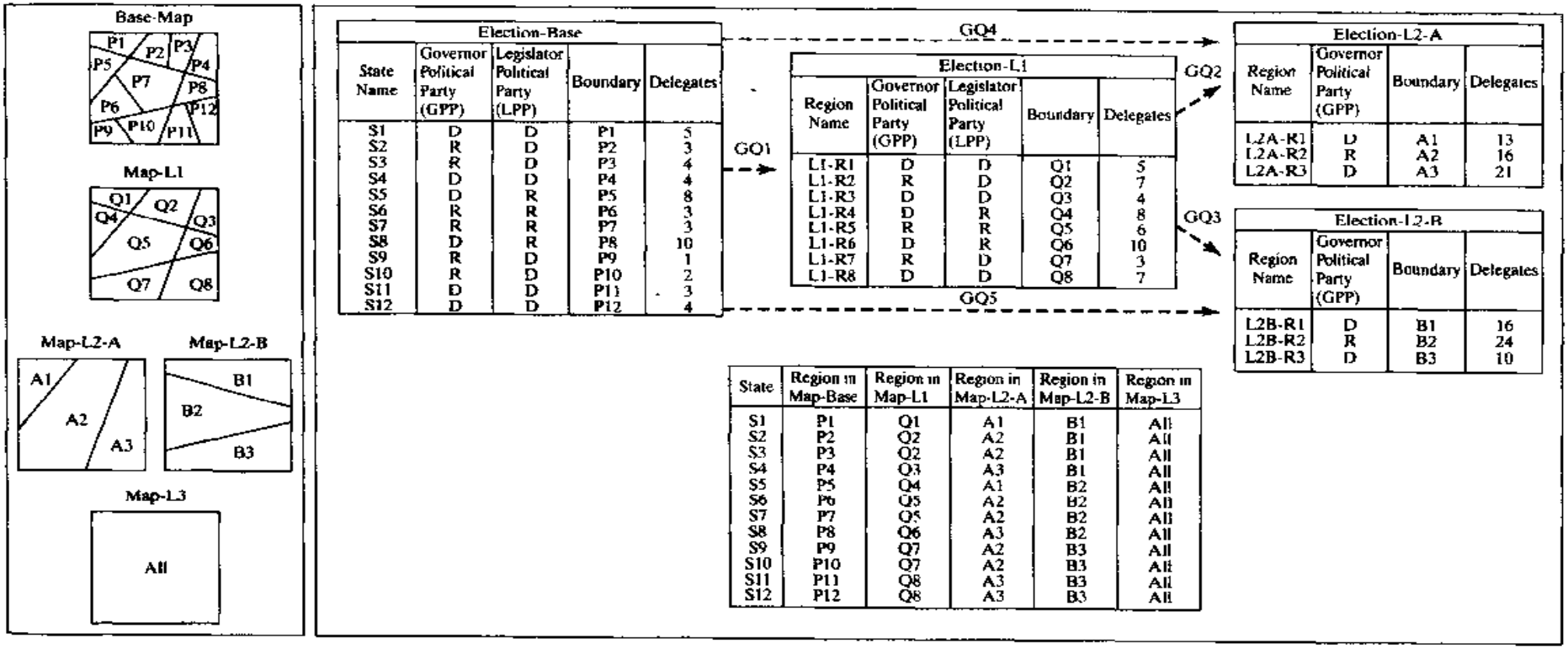


图8-16 GIS聚集函数的几何并的例子

Company（公司）和Region（地区），用销售额进行度量。零维数据立方体是一个表示总计的点。一维数据立方体共有三个：Group-by Region、Group-by Company和Group-by Year。三个二维数据立方体是一些交叉表，它们是这三个维的一个组合。三维数据立方体是由三个相交的二维交叉表组成的立方体。图8-18给出经cube操作之后三维数据立方体中全部元素的表的形式。创建一个数据立方体需要生成一个聚集列的幂集。

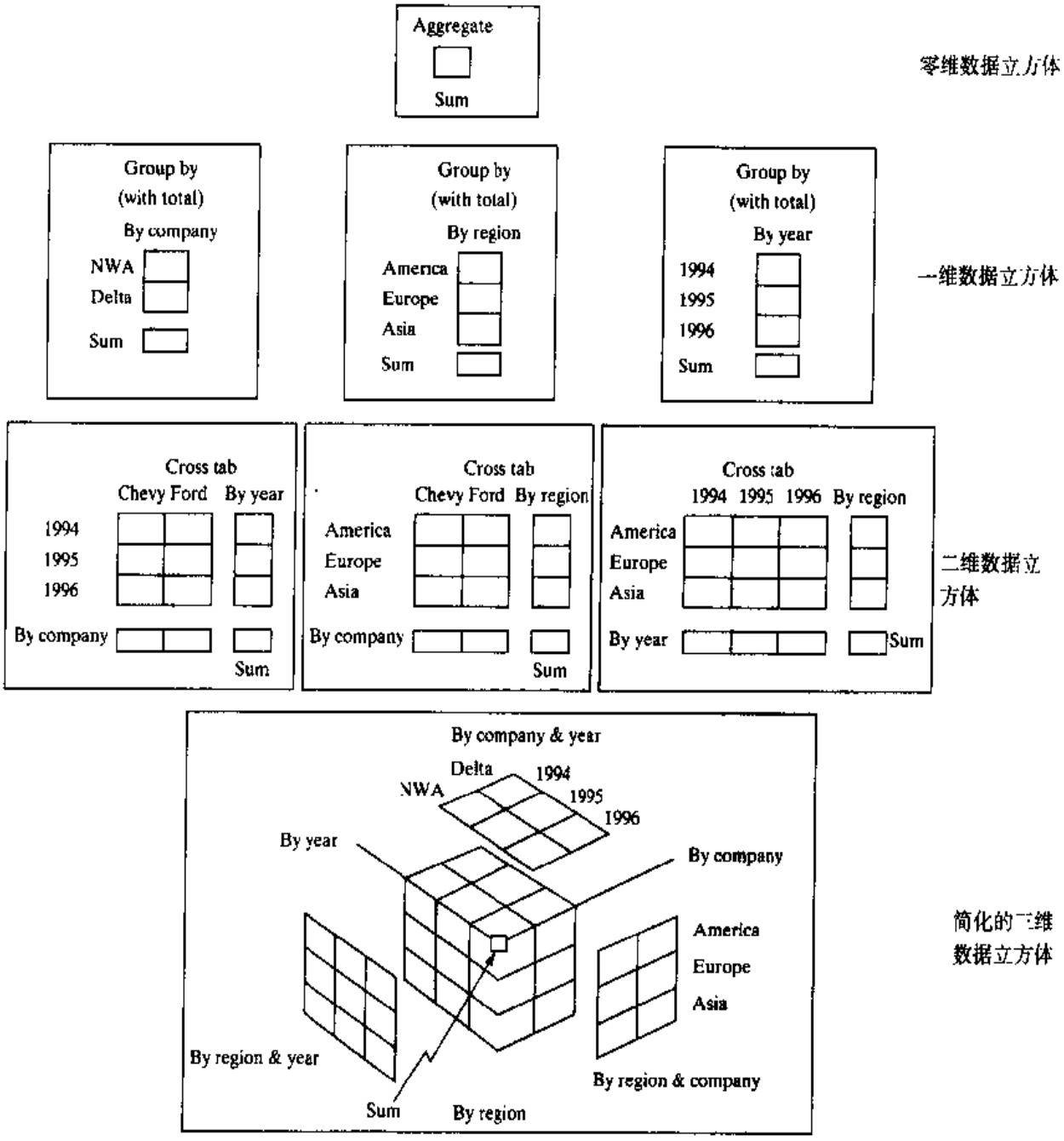


图8-17 零维、一维、二维和三维数据立方体

图8-17中各个子空间数据立方体的表格视图在图8-19中给出。图8-19中所有表的并是利用cube操作符生成结果表。图8-17中标记为“Aggregate”的零维子空间数据立方体在图8-19中用表“SALES-L2”表示。图8-17中标记为“By Company”的一维子空间数据立方体在图8-19中用表“SALES-L1-C”表示；而图8-17中标记为“By Company & Year”的二维数据立方体在图8-19中用表“SALES-L0-C”表示。利用类似的方法，读者可以自己建立其余子空间数据立方体和表之间的对应关系。

表8-3 GROUP-BY查询列表

Q1	SELECT 'ALL', Year, Region, SUM(Sales) FROM SALES-Base group-by Year,Region
Q2	SELECT 'ALL', 'ALL', Region SUM(Sales) FROM SALES-L0-A group-by Region
Q3	SELECT 'ALL', 'ALL', 'ALL' SUM(Sales) FROM SALES-L1-A
Q4	SELECT 'ALL', 'ALL', Region, SUM(Sales) FROM SALES-Base group-by Region
Q5	SELECT 'ALL', 'ALL', 'ALL', SUM(Sales) FROM SALES-Base

CUBE操作符可以由一系列使用GROUP-BY操作符和聚集函数的SQL查询来建模。图8-19中每个箭头均用一个SQL查询表示。表8-3列出对应于图8-19中标记为$Q1$，$Q2$，...，$Q5$的五个箭头的查询语句。例如，表8-3中的查询$Q1$用“Year”和“Region”聚集“Sales”，并生成图8-19中的表“SA LES-L0-A”。

带有CUBE操作符的GROUP-BY子句指定分组属性，该属性也应该出现在SELECT子句中，以便使每个函数应用到一组元组所产生的结果值与分组属性的值保持一致。图8-18的左上角显示的就是一个带CUBE操作符的SQL查询。另外，SQL还提供了用于GROUP BY子句的ROLLUP操作符。

8.3.4 哪些地方用到聚集层次

为支持OLAP，数据立方体提供了下列操作符：roll-up、drill-down、slice和dice以及pivot。我们现在给出这些操作符的定义：

- Roll-up（上卷）：提高抽象层次。该操作符对一个或多个维进行泛化，并对相应的度量进行聚集。例如，图8-19中的表SALES-L0-A是表SALES-Base在“Company”维上的一个上卷。许多SQL的商业实现都把ROLLUP操作符作为GROUP BY子句的扩展来支持。例如，对表SALES-Base的ROLLUP（Year，Region）操作将返回表SALES-L0-C（聚集掉Region）和SALES-L1-C（聚集掉Year和Region）的并集。

立方体查询

```
SELECT  Company, Year, Region,
        Sum(Sales) AS Sales
FROM  SALES
GROUP BY  CUBE Company, Year, Region
```

SALES

Company	Year	Region	Sales
NWA	1994	America	20
NWA	1994	Europe	15
NWA	1994	Asia	5
NWA	1995	America	12
NWA	1995	Europe	6
NWA	1995	Asia	18
NWA	1996	America	7
NWA	1996	Europe	20
NWA	1996	Asia	17
Delta	1994	America	15
Delta	1994	Europe	3
Delta	1994	Asia	12
Delta	1995	America	9
Delta	1995	Europe	14
Delta	1995	Asia	20
Delta	1996	America	15
Delta	1996	Europe	8
Delta	1996	Asia	16

基本表

立方体

Data Cube

Company	Year	Region	Sales
ALL	ALL	ALL	232
NWA	ALL	ALL	120
Delta	ALL	ALL	112
ALL	1994	ALL	70
ALL	1995	ALL	79
ALL	1996	ALL	83
ALL	ALL	America	78
ALL	ALL	Europe	66
ALL	ALL	Asia	88
NWA	1994	ALL	40
NWA	1995	ALL	36
NWA	1996	ALL	44
Delta	1994	ALL	30
Delta	1995	ALL	43
Delta	1996	ALL	39
NWA	ALL	America	39
NWA	ALL	Europe	41
NWA	ALL	Asia	40
Delta	ALL	America	39
Delta	ALL	Europe	25
Delta	ALL	Asia	48
ALL	1994	America	35
ALL	1994	Europe	18
ALL	1994	Asia	17
ALL	1995	America	21
ALL	1995	Europe	20
ALL	1995	Asia	38
ALL	1996	America	22
ALL	1996	Europe	28
ALL	1996	Asia	33
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

通过立方体操作生成的结果表

图8-18 数据立方体的一个例子

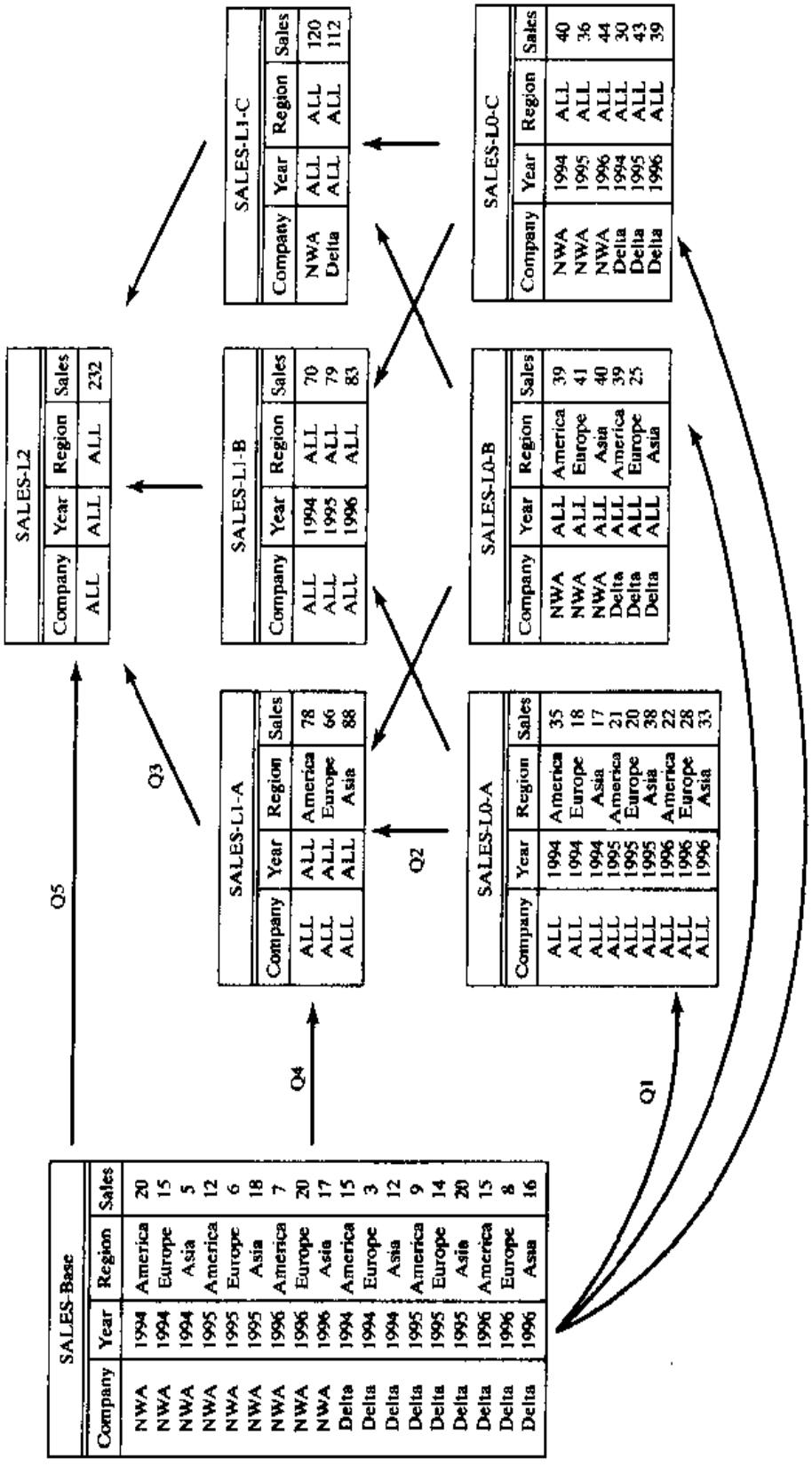


图8-19 使用group-by的一个例子

- Drill-down（下钻）：降低抽象层次或增加细节。它特化一个或少数几个维，表示低层次的聚集。例如，图8-19中的表SALES-L0-A就是表SALES-L1-A在“Year”维上的下钻。
- Slice和dice（切片和切块）：选择和投影。在一个维上切片类似于在该维上下钻一级，但其显示的条目数量要受切片命令所指定的数目的限制。切块操作很像一维以上的切片。例如在二维的显示中，切块意味着对行和列同时进行切片。

表8-4显示的是在图8-19中表SALES-L2的“Year”维上，对值“America”切片的结果。

表8-5显示的是在图8-19中表SALES-L2的“Year”维上对值“1994”和在“Region”维上对值“America”进行切块的结果。

表8-4 在“Region”维上对值“America”切片

Company	Year	Region	Sales
ALL	ALL	America	78

表8-5 在图8-19中，表SALES-L2的“Year”维上对值“1994”切块并在“Region”维上对值“America”切块

Company	Year	Region	Sales
ALL	1994	America	35

- Pivoting（旋转）：重定向数据的多维视图。它表示不同交叉表布局中的度量。该函数经常在电子表格软件例如，微软的Excel中出现。

8.4 小结

近年来，随着遥感技术的发展，栅格形式的空间数据越来越多。由于大部分遥感数据可能永远不能转换成矢量格式，所以栅格数据库就变得极其重要。栅格数据可以用场模型来建模。地图代数提供了一套操纵栅格数据集的操作符。数据分片是最常用的存储栅格数据集的方法。

我们可以通过元数据或内容来查询栅格数据项。由元数据（如记录日期、传感器类型、空间位置）指定的查询可以使用前面几章中讨论的技术进行处理；而基于内容的检索（CBR）则需要新的技术来定义和处理。CBR查询通常查找最相似的栅

格数据项而不是检索满足所有查询谓词的栅格数据项。栅格数据项之间的空间相似性根据栅格数据的属性关系图（ARG）来确定。所谓属性关系图，就是表示代表各个栅格数据项的空间对象及其空间关系（拓扑关系、方向关系、距离关系）的模型。CBR查询可以通过检索与目标ARG最相似的ARG进行处理。如何高效处理CBR查询仍然是一个研究热点。

空间数据仓库是为高效支持统计聚集汇总查询而设计的一种特殊的数据库。聚集操作可以分为三组，即分布的、代数的和整体的。即使在数据集过大而无法全部装入内存的情况下，也可以通过使用少量的内存缓冲，对数据集进行一次扫描，从而计算出分布聚集函数和代数聚集函数。数据仓库通常处理一个事实表和一组维表，其中事实表与所有的维表都有关系，有特定意义的汇总信息可以使用不同维的子集来表示。给定*N*维，就有2的*N*次幂个有意义的汇总表，这些表都可以使用带有group by子句的SQL查询来计算。CUBE操作符是SQL group by子句的一个泛化，这样就可以通过一个语句来指定所有的2的*N*次幂个汇总表的计算。CUBE操作符生成的汇总表可以使用诸如旋转、上卷、下钻和切片和切块等操作来进行研究分析。

8.5 参考书目

8.1 矢量-栅格的二分在[Couclelis，1992]中有详细讨论。用于制图建模的地图代数在[Tomlin,1990]中有所介绍；而有关图像处理的图像代数可参考Ritter的[Ritter et al., 1990]；关于动态空间建模所用的空间代数在[Takeyama and Couclelis,1997]中有所阐述。要了解数据库的全貌，见[Baumann,1994]。

8.2 近来，基于内容的检索受到广泛关注。[Yoshitaka and Ichikawa,1999]对多媒体数据库CBR的近期发展进行了介绍。要了解空间（CBR）的全貌，可参考[Papadias et al., 1999]和[Ang et al.,1998]。对于属性关系图的索引，请参见[Petrakis and Faloutsos，1997]。GIS的基于Spatial-Query-by-Sketch的系统在[Egenhofer,1997]中有所介绍。基于纹理的航空图像检索可以在[Ma and Manjunath,1998]中找到相应的介绍。

8.3 数据立方体和其上的聚集操作的概念可以在[Gray et al.,1997]中找到。[Sarawagi and Stonebraker,1994]详细讨论了在二级和三级存储媒介存储多维数组的内容。[Keim,1999]中介绍了三维空间数据库的相似性搜索。

[Barclay et al.,2000; Shekhar et al., 2001b]中介绍了空间数据仓库的一个例子。

8.6 习题

1. 假设R是图8-2a 所示的栅格层。请使用FocalMean操作来计算一个新的栅格层R_{lp}。其中，FocalMean的定义如下：

$$R_{lp}(x) = \frac{1}{|nbd(x)|} \sum_{y \in nbd(x)} R(y)$$

这里，x和y是像素，$nbd(x)$是x的邻接像素的集合。对$nbd(x)$使用“后”（Queen）邻居模板。新栅格层R_{lp}是原始栅格层R的一个“模糊”图像，这就是FocalMean操作有时也被称为低通过滤器的原因。

2. 计算一个新的栅格层R_{hp}，使得

$$R_{hp} = R - R_{lp}$$

R_{hp}保留原始栅格层R的边缘信息（如果有的话）。它被称为高通过滤器，这是二元局部操作的一个例子。

3. 下面两个栅格图显示的是每个像素的含氮量和土壤类型。计算每种土壤类型的平均氮含量，并说明该操作是局部操作、聚焦操作、区域操作还是全局操作。

1	2	3	3
2	4	1	3
2	2	2	5
1	1	4	4

含氮量

A	A	B	B
	B	B	B
C	C		A
C	C	C	A

土壤地图

4. 栅格层R表示家庭（H）和银行（B）的位置。假定每个家庭与这两家银行都有业务往来。一份有关各家庭最近三十天内去银行的调查结果表明：一个家庭与最近银行的业务往来占与银行全部业务往来的60%。请问在最近的三十天内，各家庭到银行花费的平均距离是多少？假设水平相邻和垂直相邻的两点之间的距离是一个距离单位，对角线相邻的两点之间的距离为1.4个距离单位。

H	H	B	H
H	H	H	H
H	H	H	H
H	B	H	H

银行和家庭的位置

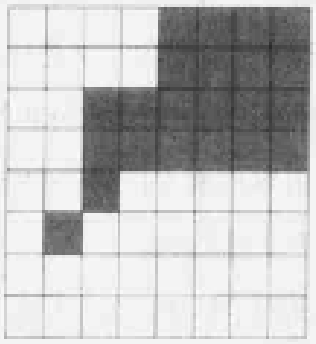
5. 从数据库的角度分析，先trim（剪裁）后slice（分片）和先slice后trim的两个操作顺序中，哪一种顺序更有效？假定操作符从右到左进行判定。

6. 对于下面所示的二值栅格图像，分别计算出采用下面三种数据结构所需要的存储空间。假定存储每个数字或指针需要一个字节的空间。

   • 矩阵表示

   • 游程编码

   • 四叉树



二值图像

7. 矢量与栅格的二分概念可以用英语中的“what”和“where”来比较。请讨论之。

8. 在栅格DBMS中，拓扑是隐式的还是显式的？

9. 采用如下所示的分辨率，计算存储20km × 20km矩形区域的栅格表示所需要的存储空间。

   (a) 10m × 10m大小的像素。

   (b) 1m × 1m大小的像素。

   (c) 1cm × 1cm大小的像素。

10. 研究OGIS提出的栅格数据模型。并与本章讨论的地图代数做比较。

11. 定义代数系统的“闭合性”概念，评价OGIS关于矢量空间数据模型的“闭

合性”。

12. 用于地理空间应用（例如，人口普查）的数据仓库有什么特别之处？

13. 比较和对比：

(a) 度量与维度。

(b) 数据挖掘与数据仓库。

(c) 上卷与下钻。

(d) SQL2 GROUP BY聚集查询与带有CUBE操作符的GROUP BY聚集操作。

(e) CUBE与上卷。

14. 将下列的空间聚集函数按照分布的、代数的、整体的函数类别进行划分：mean、medoid、autocorrelation（Moran's I）、minimum bounding orthogonal rectangle以及7.4节的minimum bounding circle、minimum bounding convex polygon、spatial outlier detection tests。

15. 地图代数通常是定义在带有标量特征的栅格图之上，也就是说栅格的每个单元格都有一个标量值。请把地图代数推广到具有矢量特征的栅格图，以便应用于风速图和进行洪水分析的高程坡度图。

16. 一些基于内容检索方面的研究者认为ARG类似于实体关系模型，你同意吗？请简单举例并解释。

17. 对于带有无固定形状现象（例如云、烟的气溶胶）的栅格图可以采用ARG对其建模吗？请解释。

18. 任意一对空间对象通常都存在一些空间关系，例如距离。对所有空间关系建模会导致团，即ARG图中出现大量的边。请你提出一些方法来简化ARG，使基于内容的检索更加有效。

19. 考虑使用基于内容的查询来检索满足给定边界形状（例如泪滴、女士靴子、鸭子）的国家。讨论如何对空间对象的形状及形状之间的相似性进行建模。

20. 考虑图8-19，写出对应于各个表之间未标注的边的SQL聚集查询。

21. 研究一个电子表格软件，如微软的Excel，找出旋转操作。创建一个示例数据集并说明旋转操作的效果。

22. 研究人口普查数据集，并将其建模为一个数据仓库。列出所有的维、度量、聚集操作和聚集层次。

23. 给定一个表FACT（属性包括routenumber、stopid、rank、fare-collected、number-boarded和number-disboared），它是对表6-2中给出的RouteStop表的扩展。请使用数据仓库操作和SQL来处理下面的报表：

(i) 使用站点汇总的收费情况。

(ii) 使用线路汇总的收费情况。

(iii) 统计各站点的平均货物活动（装货+卸货）。

(iv) 每条线路上的最繁忙的车站。

参考文献

Acharya, S., Poosala, V., and Ramaswamy, S. (1999). Selectivity estimation in spatial databases. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadephia, Pennsylvania, USA*, pages 13–24. ACM Press.

Adam, N. and Gangopadhyay, A. (1997). *Database issues in Geographical Information Systems*. Kluwer Academics.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In Bocca, J. B., Jarke, M., and Zaniolo, C., editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann.

Albrecht, J. (1998). Universal analytical gis operations - a task-oriented systematization of data-structure-independent gis functionality. In *Geographic information research: transatlantic perspectives, M. Craglia and H. Onsrud (Eds.)*, pages 557–591. Tylor & Francis.

Ambroise, C., Dang, V., and Govaert, G. (1997). Clustering of spatial data by the EM algorithm. *Quantitative Geology and Geostatistics*, 9:493–504.

Ang, C., Ling, T., and Zhou, X. (1998). Qualitative spatial relationships represenation and its retrieval. In *Database and Expert Systems Applications(DEXA)*, volume 1460, pages 65–77. Springer-Verlag.

Anselin, L. (1988). *Spatial econometrics: Methods and models*. Kluwer.

Aref, W. and Samet, H. (1994). A cost model for query optimization using R-trees. In *The Proceedings of the Second ACM Workshop on Advances in Geographic Information Systems, ACM Press*.

Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., and Vitter, J. S. (1998). Scalable sweeping-based spatial join. In *VLDB '98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24–27, 1998, New York City, New York, USA*, pages 570–581. Morgan Kaufmann.

Asano, T., Ranjan, D., Roos, T., Wiezl, E., and Widmayer, P. (1997). Space filling curves and their use in the design of geometric data structures. *Theoretical Computer Science*, 181(1):3–15.

Bailey, D. (1998). *Java structures*. WCB McGraw-Hill.

Bailey, T. C. and Gatrell, A. (1995). *Interactive spatial data analysis*. Longman Scientific & Technical.

Barclay, T., Slutz, D. R., and Gray, J. (2000). Terraserver: A spatial data warehouse. *SIGMOD Record*, 29(2):307–318.

Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. John Wiley, New York, 3rd edition.

Baumann, P. (1994). Management of multidimensional discrete data. *The VLDB Journal*, 3(4):401–444.

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*. ACM Press.

Biskup, J., Rasch, U., and Stiefeling, H. (1990). An extension of sql for querying graph relations. *Computer Languages*, 15(2):65–82.

Blyth, T. and Robertson, E. (1999). *Basic Linear Algebra*. Springer-Verlag.

Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The unified modeling language user guide*. Addison Wesley.

Brinkhoff, T., Kriegel, H., Schneider, R., and Seeger, B. (1994). Multi-step processing of spatial joins. In *Proceedings of ACM SIGMOD Int. Conf. on Management of Data*, pages 197–208. ACM Press.

Brinkhoff, T., Kriegel, H., and Seeger, B. (1993). Efficient processing of spatial joins using r-trees. In *Proceedings of ACM SIGMOD Int. Conf. on Management of Data*, pages 237–246. ACM Press.

Brinkhoff, T. and Kriegel, H.-P. (1994). The impact global clustering on spatial database systems. In *Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB '94)*. Morgan Kaufmann.

Brodeur, J., Bedard, Y., and Proulx, M.-J. (2000). Modelling geospatial application databases using uml-based repositories aligned with international standards in geomatics. In *ACM-GIS 2000, Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems, November 10–11, 2000, Washington, D.C., USA*, pages 39–46. ACM.

Chakrabarti, K. and Mehrotra, S. (1999). Efficient concurrency control in multidimensional access methods. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadephia, Pennsylvania, USA*, pages 25–36. ACM Press.

Chaudhuri, S. and Shim, K. (1996). Optimization of queries with user-defined predicates. In *VLDB '96, Proceedings of 22nd International Conference on Very Large Data Bases*, pages 87–98. Morgan Kaufmann.

Chou, H.-T. and DeWitt, D. J. (1985). An evaluation of buffer management strategies for relational database systems. In *VLDB '85, Proceedings of 11th International Conference on Very Large Data Bases*, pages 127–141. Morgan Kaufmann.

Chrisman, N. (1997). *Exploring geographic information systems*. John Wiley and Sons.

Clementini, E. and Felice, P. D. (1995). A comparison of methods for representing topological relationships. *Information Sciences*, 3.

Clementini, E., Felice, P. D., and van Oosterom, P. (1993). A small set of formal topological relationships suitable for end-user interaction. In *Lecture Notes in Computer Science, Vol. 692*, pages 277–295. SSD '93, Springer-Verlag.

Corral, A., Manolopoulos, Y., Theodoridis, Y., and Vassilakopoulos, M. (2000). Closest pair queries in spatial databases. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA*, pages 189–200. ACM.

Couclelis, H. (1992). People manipulate objects (but cultivate fields): beyond on the raster-vector debate in gis. In *Theories and Methods of Spatio-temporal Reasoning in Geographic Space*, pages 65–77.

Cressie, N. A. (1993). *Statistics for spatial data*. Wiley Series in Probability and Statistics.

de La Beaujardiere, J., Mitchell, H., Raskin, R. G., and Rao, A. (2000). The nasa digital earth testbed. In *ACM-GIS 2000, Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems, November 10–11, 2000, Washington, D.C., USA*, pages 47–53. ACM.

Delis, V., Hadzilacos, T., and Tryfona, N. (1994). An introduction to layer algebra. In *Proceedings of Advances in GIS Research*. Taylor and Francis.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data using the EM algorithm. *Journal of the Royal Statistical Society*, 39(B).

den Bercken, J. V., Seeger, B., and Widmayer, P. (1997). A generic approach to bulk loading multidimensional index structures. In *VLDB '97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25–29, 1997, Athens, Greece*, pages 406–415. Morgan Kaufmann.

Egenhofer, M. (1991a). Extending SQL for geographical display. *Cartography and Geographic Information Systems*, 18(4):230–245.

Egenhofer, M. (1991b). Reasoning about binary topological relations. In *SSD '91, Advances in Spatial Databases, Second International Symposium*, pages 143–160. Springer-Verlag.

Egenhofer, M. (1994). Spatial SQL: A Query and Presentation Language. *IEEE TKDE*, 6(1):86–95.

Egenhofer, M. (1997). Query Processing in Spatial-Query-by-Sketch. *Journal of Visual Languages and Computing*, 8(4):403–424.

Egenhofer, M., Frank, A. U., and Jackson, J. P. (1989). A topological data model for spatial databases. In *SSD '89, Design and Implementation of Large Spatial Databases, First Symposium*, pages 47–66. Springer-Verlag.

Elmasri, R. and Navathe, S. (2000). *Fundamentals of database systems*. Addison Wesley & Benjamin Cummings.

ESRI (1991). *Network analysis: Modeling network systems*. Environmental Systems Research Institute, Inc.

Estivill-Castro, V. and Murray, A. (1998). Discovering associations in spatial data—an efficient medoid based approach. In *Research and Development in Knowledge Discovery and Data Mining (PAKDD-98)*, volume 1394, pages 110–121. Springer.

Faloutsos, C. and Kamel, I. (1994). Beyond uniformity and independence: Analysis of r-tree using the concept of fractal dimension. In *Proceeding 13th ACM-SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 4–13. ACM Press.

Faloutsos, C. and Roseman, S. (1989). Fractals for secondary key retrieveal. In *Proceedings of the ACM conference on Principles of Database Systems*, pages 247–252.

Faloutsos, C., Seeger, B., Traina, A. J. M., and Jr., C. T. (2000). Spatial join selectivity using power laws. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA*, pages 177–188. ACM.

Faloutsos, C. and Sellis, T. (1987). Analysis of object oriented spatial access methods. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, pages 426–439. ACM Press.

Fischer, M. and Getis, A. (1997). *Recent developments in spatial analysis*. Springer Verlag.

Fotheringham, A. S. and Rogerson, P. A. (1994). *Spatial Analysis and GIS*. Taylor and Francis.

Gaede, V. and Gunther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.

Goldstein, J. and Ramakrishnan, R. (2000). Contrast plots and p-sphere trees: Space vs. time in nearest neighbour searches. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10–14, 2000, Cairo, Egypt*, pages 429–440. Morgan Kaufmann.

Goodchild, M. F. (1986). *Spatial Autocorrelation*. CATMOG 47, GeoBooks.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997). Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1):29–53.

Griffith, D. (1999). Statistical and mathematical sources of regional science theory: Map pattern analysis as an example. *Papers in Regional Science*, 78(1):21–45.

Guting, R. (1994a). An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4):357–399.

Guting, R. (1994b). Graph DB: Modeling and querying graphs in databases. In *VLDB '94, Proccdings of 20th International Conference on Very Large Data Bases*. Morgan Kaufmann.

Guttman, A. (1984). R-tree: A dynamic index structure for spatial searching. In *SIGMOD '84, Proceedings of the ACM SIGMOD Conference*. ACM Press, ACM Press.

Hadzilacos, T. and Tryfona, N. (1997). An extended entity-relationship model for geographic applications. *SIGMOD Record*, 26(3).

Hagen, L. and Kahng, A. (1991). Fast spectral methods for ratio cut partitioning and clustering. In *Proceedings of IEEE International Conference on Computer Aided Design*, pages 10–13.

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Han, J., Koperski, K., and Stefanovic, N. (1997). Geominer: A system prototype for spatial data mining. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13–15, 1997, Tucson, Arizona, USA*, pages 553–556. ACM Press.

Hand, D. J. (1999). Statistics and Data Mining: Intersecting Disciplines. *SIGKDD Explorations*, 1(1):16–19.

Hawkins, D. (1980). *Identification of Outliers*. Chapman and Hall.

Hellerstein, J. and Stonebraker, M. (1993). Predicate migration: Optimizing queries with expensive predicates. In *SIGMOD '93, Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 267–276. ACM Press.

Herring, J. (1991). The mathematical modeling of spatial and non-spatial information in geographic information systems. In Mark, D. and Frank, U., editors, *Cognitive and Linguisitic Aspects of Geographic Space*, pages 313–350.

Jiang, B. (1991). Traversing graphs in a paging environment, bfs or dfs? *Information Processing Letters*, 37(3):143–147.

Jing, N., Huang, Y., and Rundensteiner, E. (1998). Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 10(3):409–432.

Kamel, I. and Faloutsos, C. (1993). On packing r-trees. In *CIKM'93, Proceedings of the Second International Conference on Information and Knowledge Management*, pages 490–499. ACM Press.

Kanth, K. V. R., Ravada, S., Sharma, J., and Banerjee, J. (1999). Indexing medium-dimensionality data in oracle. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadephia, Pennsylvania, USA*, pages 521–522. ACM Press.

Karypis, G., Aggarwal, R., Kumar, V., and Shekhar, S. (1998). hmetis home page. http://www-users.cs.umn.edu/ ~karypis/metis/hmetis/main.html.

Karypis, G. and Kumar, V. (1998). Metis home page. http://www-users.cs.umn.edu/~karypis/metis/metis/main.html.

Keim, D. A. (1999). Efficient geometry-based similarity search of 3d spatial databases. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadephia, Pennsylvania, USA*, pages 419–430. ACM Press.

Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307.

Khoshafian, S. and Baker, A. (1998). *MultiMedia and Imaging Databases*. Morgan Kaufmann.

Koperski, K. and Han, J. (1995). Discovery of spatial association rules in geographic information systems. In *Advances in Spatial Databases, 4th International Symposium(SSD '95)*, pages 47–66. Springer-Verlag.

Korn, F. and Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA*, pages 201–212. ACM.

Kornacker, M. and Banks, D. (1995). High-concurrency locking in R-trees. In *VLDB '95, Proceedings of 21th International Conference on Very Large Data Bases*. Morgan Kaufmann.

Kriegel, H.-P., Muller, A., Potke, M., and Seidl, T. (2001). Spatial data management for computer aided design. In *ACM SIGMOD International Conference on Management of Data*, page 614. ACM Press.

Laurini, R. and Thompson, D. (1992). *Fundamentals of spatial information systems*. Academic Press.

Lehman, P. and Yao, S. (1981). Efficient locking for concurrent operations on b-trees. *ACM TODS*, 6(4).

LeSage, J. (1997). Regression analysis of spatial data. *Journal of Regional Analysis and Policy (Publisher: Mid-Continent Regional Science Association and UNL College of Business Administration)*, 27(2):83–94.

Leutenegger, S. T. and Lopez, M. A. (2000). The effect of buffering on the performance of r-trees. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):33–44.

Lin, H. and Huang, B. (2001). Sql/sda: A query language for supporting spatial data analysis and its web-based implementation. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):671–682.

Lo, M. and Ravishankar, C. (1996). Spatial hash joins. In *ACM SIGMOD International Conference on Management of Data*, pages 247–258. ACM Press.

Luc, A. (1994). Exploratory Spatial Data Analysis and Geographic Information Systems. In Painho, M., editor, *New Tools for Spatial Analysis*, pages 45–54.

Luc, A. (1995). Local Indicators of Spatial Association: LISA. *Geographical Analysis*, 27(2):93–115.

Ma, W. and Manjunath, B. (1998). A texture thesaurus for browsing large aerial photographs. *Journal of the American Society for Information Science*, 49(7):633–48.

Mamoulis, N. and Papadias, D. (1999). Integration of spatial join algorithms for processing multiple inputs. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadephia, Pennsylvania, USA*, pages 1–12. ACM Press.

Mannino, M. and Shapiro, L. (1990). Extensions to query languages for graph traversal problem. *IEEE TKDE*, 2(3):353–363.

Merrett, T., Kimbayasi, Y., and Yasuura, H. (1981). Scheduling of page-fetches in join operations. In *Proceedings of the 7th International Conference on Very Large Data Bases*. Morgan Kaufmann.

Moon, B., Jagadish, H., Faloutsos, C., and Saltz, J. H. (1996). Analysis of the clustering properties of hilbert space-filling curve. Technical Report UMIACS-TR-96-20, CS dept., University of Maryland.

Murray, J. D. and van Ryper, W. (1999). *Encyclopedia of Graphics File Formats*. O'REILLY.

Nievergelt, J., Hinterberger, H., and Sevcik, K. (1984). The grid file: An adaptable, symmetric multikey file structure. *ACM Transactions on Database Systems*, 9(1):38–71.

OGIS (1999). Open GIS consortium: Open GIS simple features specification for SQL (Revision 1.1). In *URL: http://www.opengis.org/techno/specs.htm*.

OGIS (2000). Web map server interfaces implementation specification. In *URL: http:// www. opengis.org/techno/specs/00-028.pdf*.

OpenGIS (1998). *Open GIS simple features specification for SQL*. Open GIS Consortium, Inc., http://www.opengis.org.

Ordonez, C. and Cereghini, P. (2000). Sqlem: Fast clustering in sql using the em algorithm. *SIGMOD Record*, 29(2):559–570.

Orenstein, J. (1986). Spatial query processing in an object-oriented database. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*. ACM Press.

Orenstein, J. and Manola, F. (1988). PROBE spatial data modeling and query processing in an image database application. *IEEE Trans on Software Engineering*, 14(5):611–629.

Ozesmi, S. and Ozesmi, U. (1999). An artificial neural network approach to spatial habitat modeling with interspecific interation. *Ecological Modeling*, 116:15–31.

Pagel, B.-U., Six, H.-W., and Toben, H. (1993a). The transformation technique for spatial objects revisited. In *SSD, Lecture Notes in Computer Science, Vol. 2121*, pages 73–88. Springer-Verlag.

Pagel, B.-U., Six, H.-W., Toben, H., and Widmayer, P. (1993b). Towards an analysis of range query performance in spatial data structures. In *Proceedings of the Twelfth ACM SIGACT-SIGMODSIGART Symposium on Principles of Database Systems*, pages 214–221. ACM Press.

Papadias, D., Karacapilidis, N., and Arkoumanis, D. (1999). Processing fuzzy spatial queries: A configuration similarity. *International Journal of GIS*, 13(2):93–128.

Patel, J. and Dewitt, D. (1996). Partition based spatial-merge join. *Proceedings of ACM SIGMOD*. pages 259–270.

Patel, J. M. and DeWitt, D. J. (1996). Partition Based Spatial-Merge Join. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. ACM Press.

Patel, J. M. and DeWitt, D. J. (2000). Clone join and shadow join: two parallel spatial join algorithms. In *ACM-GIS 2000, Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems, November 10–11, 2000, Washington, D.C., USA*, pages 54–61. ACM.

Petrakis, E. and Faloutsos, C. (1997). Similarity searching in medical image databases. *IEEE TKDE*, 9(3):433–447.

Price, R., Tryfona, N., and Jensen, C. S. (2000). Modeling part-whole relationships for spatial data. In *ACM-GIS 2000, Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems, November 10–11, 2000, Washington, D.C., USA*, pages 1–8. ACM.

Ramakrishnan, R. (1998). *Database management system*. McGraw Hill.

Ritter, G., Wilson, J., and Davidson, J. (1990). Image algebra: An overview. *Computer Vision, Graphics and Image Processing*, 49:297–331.

Roussopoulos, N., Kelley, S., and Vincent, F. (1995). Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*. ACM Press.

Samet, H. (1990). *The design and analysis of spatial data structures*. Addison-Wesley.

Sarawagi, S. and Stonebraker, M. (1994). Efficient organization of large multidimensional arrays. In *Tenth International Conference on Data Engineering*, pages 328–336. IEEE Computer Society.

Schneider, R. and Kriegel, H.-P. (1991). The tr*-tree. A new representation of polygonal objects supporting spatial queries and operations. In *Lecture Notes in Computer Science, Vol. 553*, pages 249–264. SSD '93.

Scholl, M. O., Voisard, A., and Rigaux, P. (2001). *Spatial database management systems*. Morgan Kaufmann Publishers.

Seidl, T. and Kriegel, H.-P. (1998). Optimal multi-step k-nearest neighbor search. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2–4, 1998, Seattle, Washington, USA*, pages 154–165. ACM Press.

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. (1979). Access path selection in a relational database management system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 23–34. ACM Press.

Sheikholeslami, G., Chatterjee, S., and Zhang, A. (1998). Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB '98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24–27, 1998, New York City, New York, USA*, pages 428-439. Morgan Kaufmann.

Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., and Lu, C. (1999a). Spatial databases—accomplishments and research needs. *IEEE TKDE*, 11(1):45-55.

Shekhar, S., Coyle, M., Liu, D.-R., Goyal, B., and Sarkar, S. (1997). Data models in geographic information systems. *Communication of the ACM*, 40(4).

Shekhar, S. and Huang, Y. (2001). Discovering spatial co-location patterns: A summary of results. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases, SSTD 2001, Redondo Beach, CA, USA, July 12–15, 2001*, volume 2121 of *Lecture Notes in Computer Science*, pages 236–256. Springer.

Shekhar, S. and Liu, D.-R. (1997). A connectivity-clustered access method for networks and network computation. *IEEE TKDE*, 9(1):102–117.

Shekhar, S., Lu, C., Chawla, S., and Ravada, S. (1999b). Efficient join index based join processing: a clustering approach. *TR99-030 (Also to appear in IEEE TKDE)*.

Shekhar, S., Schrater, P. R., Vatsavai, R. R., Wu, W., and Chawla, S. (2002). Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on Multimedia*, 4(2):1–15.

Shekhar, S., Vatsavai, R. R., Chawla, S., and Burk, T. E. (1999c). Spatial pictogram enhanced conceptual data models and their translation to logical data models. *Integrated Spatial Databases: Digital Images and GIS. Lecture Notes in Computer Science*, 1737:77–104.

Shin, H., Moon, B., and Lee, S. (2000). Adaptive multi-stage distance join processing. *SIGMOD Record*, 29(2):343–354.

Silberschatz, A., Korth, H., and Sudarshan, S. (1997). *Database system concepts, 3rd Ed.* McGraw-Hill.

Song, J.-W., Whang, K.-Y., Lee, Y.-K., Lee, M.-J., and Kim, S.-W. (1999). Transformation-based spatial join. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, November 2–6, 1999*, pages 15–26. ACM.

Stonebraker, M. and Moore, D. (1997). *Object relational DBMSs: The next great wave*. Morgan Kaufmann.

Takeyama, M. and Couclelis, H. (1997). Map dynamics: integrating cellular automata and gis through geo-algebra. *International Journal of GIS*, 11(1):73–91.

Theodoridis, Y. and Sellis, T. (1996). A model for the prediction of r-tree performance. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 161–171 ACM Press.

Theodoridis, Y., Stefanakis, E., and Sellis, T. (1998). Cost models for join queries in spatial databases. In *Proceedings of the Fourteenth International Conference on Data Engineering*, pages 476–483. IEEE Press.

Theodoridis, Y., Stefanakis, E., and Sellis, T. (2000a). Efficient cost models for spatial queries using r-trees. *IEEE TKDE*, 12(1).

Theodoridis, Y., Stefanakis, E., and Sellis, T. K. (2000b). Efficient cost models for spatial queries using r-trees. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):19–32.

Tobler, W. (1979). Cellular geography. In *Philosophy in Geography, Eds., S. Gale and G. Olsson*. D. Reidel Publishing Company: Dordrecht, Holland.

Tomlin. C. (1990). *Geographic information systems and cartographic modeling*. Prentice-Hall.

Ullman, J. and Widom, J. (1999). *A first course in database systems*. Prentice-Hall.

van der Lans, R. F. (1992). *The SQL guide to Oracle*. Addison-Wesley.

Vatsavai, R. R., Burk, T. E., Wilson, B. T., and Shekhar, S. (2000). A web-based browsing and spatial analysis system for regional natural resource analysis and mapping. In *Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems ACMGIS*, pages 95–101.

Wang, W., Yang, J., and Muntz, R. R. (1997). Sting: A statistical information grid approach to spatial data mining. In *VLDB '97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25–29, 1997, Athens, Greece*, pages 186–195. Morgan Kaufmann.

Worboys, M. (1995). *GIS: A computing perspective*. Taylor and Francis.

Yoshitaka, A. and Ichikawa, T. (1999). A Survey on Content-Based Retrieval for Multimedia Databases. *IEEE TKDE*, 11(1):81–93.

Theodoridis, Y. and Sellis, T. (1996). A model for the prediction of r-tree performance. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 161–171 ACM Press.

Theodoridis, Y., Stefanakis, E., and Sellis, T. (1998). Cost models for join queries in spatial databases. In *Proceedings of the Fourteenth International Conference on Data Engineering*, pages 476–483. IEEE Press.

Theodoridis, Y., Stefanakis, E., and Sellis, T. (2000a). Efficient cost models for spatial queries using r-trees. *IEEE TKDE*, 12(1).

Theodoridis, Y., Stefanakis, E., and Sellis, T. K. (2000b). Efficient cost models for spatial queries using r-trees. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):19–32.

Tobler, W. (1979). Cellular geography. In *Philosophy in Geography, Eds., S. Gale and G. Olsson*. D. Reidel Publishing Company: Dordrecht, Holland.

Tomlin. C. (1990). *Geographic information systems and cartographic modeling*. Prentice-Hall.

Ullman, J. and Widom, J. (1999). *A first course in database systems*. Prentice-Hall.

van der Lans, R. F. (1992). *The SQL guide to Oracle*. Addison-Wesley.

Vatsavai, R. R., Burk, T. E., Wilson, B. T., and Shekhar, S. (2000). A web-based browsing and spatial analysis system for regional natural resource analysis and mapping. In *Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems ACMGIS*, pages 95–101.

Wang, W., Yang, J., and Muntz, R. R. (1997). Sting: A statistical information grid approach to spatial data mining. In *VLDB '97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25–29, 1997, Athens, Greece*, pages 186–195. Morgan Kaufmann.

Worboys, M. (1995). *GIS: A computing perspective*. Taylor and Francis.

Yoshitaka, A. and Ichikawa, T. (1999). A Survey on Content-Based Retrieval for Multimedia Databases. *IEEE TKDE*, 11(1):81–93.