南京航空航天大学

论文集

（二〇〇三年）第10册

信息科学与技术学院

（第2分册）

南京航空航天大学科技部编

二〇〇四年三月

信息科学与技术学院

042 系

目　　录

序号	姓名	职称	单位	论文题目	刊物、会议名称	年、卷、期	类别
1	李立言 秦小麟	硕士 正高	042	空间数据库中连接运算的处理与优化	中国图像图形学报	2003.08.07	
2	黄曙荣 秦小麟	硕士 正高	042	时空数据库索引机制研究	计算机科学	2003.30.06	
3	王辉 陆荣国 秦小麟	硕士 正高 正高	042 外单位 042	UML在航空电子系统中的应用研究	航空电子技术	2003.00.04	
4	钟勇 秦小麟	博士 正高	042	Adaptive Data Protection Mechanism in Intrusion Tolerant Multilevel Secure	2003年第二届亚洲软件基础研讨会		
5	包磊 秦小麟	博士 正高	042	The Oveview of Grey Spatio-temporal Data Types	2003年第二届亚洲软件基础研讨会		
6	陆军 张育平	硕士 副高	042	基于构件的软件体系结构实现技术	计算机工程与应用	* 2002.00.04	
7	肖亚军 张育平	硕士 副高	042	基于CORBA构件的软件体系结构模型	计算机工程	* 2002.28.10	
8	郑扬 张育平 李勇	硕士 副高 硕士	042	应用软件构架研究	计算机应用研究	* 2001.00.08	
9	常煜芬 张育平	硕士 副高	042	中间件技术研究	计算机应用研究	* 2001.00.10	
10	戈敏霞 张育平	硕士 副高	042	Visual C++中的自动化客户端	计算机应用研究	* 2001.00.10	
11	苏畅 张育平	硕士 副高	042	利用ASP和关系数据库实现网络MIS系统	计算机应用与软件	2003.00.11	
12	宋懿伟 张育平	硕士 副高	042	The Research of CORBA PSS and The Desigh & Implemation of PSS Prototype	2003年第二届亚洲软件基础研讨会交流		
13	皮德常 秦小麟	中级 正高	042	Technologies About Mining Tendency Association Rule	2003年第二届亚洲软件基础研讨会交流		
14	王立松 丁秋林	中级 正高	042	Study and improvement of MLS Relational Data Model	南航学报——英文版	2003.20.02	
15	王立松 陈兵 毛勇 章清	中级 副高 副高 副高	042 042 外单位 外单位	监狱管教信息系统的设计与实现	计算机工程	2003.29.02	
16	吴洁 丁秋林	副高 正高	042	Atram- An Analysis Tool of Requirement and Architecture Management	2003年IEEE SMC会议文集	2003.01.00	
17	袁家斌 陶宝祺 石景山	中级 正高 中级	042 智能所 校办	Application of Smart Card Security technology in manufacturing	材料处理技术（英国）	2003.139.00	
18	陈慧萍 王建东 张有东	博士 正高 博士	042	An Intrusion Detection Model Based on Data Mining and Fuzzy Logic	2003年第二届亚洲软件基础研讨会交流		
19	张有东 王建东 陈慧萍 叶飞跃	博士 正高 博士 博士	042	The Multiple Statistic Analysis of Association Rule Mining	2003年第二届亚洲软件基础研讨会交流		
20	凌方 王建东	硕士 正高	042	MP^M中的等值完全折取范式	广西师范大学学报——自然科学版	2003.21.01	
21	皋军 王建东	硕士 正高	042	关联规则挖掘算法更新与拓展	计算机工程与应用	2003.39.35	
22	毛宇光 韩波 朱梧槚	副高 初级 正高	042	A Relational Calculus Based on Medium Logic	2003年第二届亚洲软件基础研讨会交流		
23	毛宇光 韩波 周勇 朱梧槚	副高 初级 中级 正高	042	Logic Foundation of In Complete Information Database	2003年第二届亚洲软件基础研讨会交流		

序号	姓名	职称	单位	论文题目	刊物、会议名称	年、卷、期	类别
24	张志政 毛宇光	硕士 副高	042	一种基于语义的模糊关系数据库除操作	小型微型计算机系统	2003.24.04	
25	邬丽云 毛宇光 张志政 韩波	硕士 副高 硕士 初级	042	神经网络数据挖掘工具在外汇预测中的应用	计算机科学	2003.30.04	
26	武立福 毛宇光	硕士 副高	042	一种改进的多级安全关系数据模型	计算机应用	2003.23.07	
27	武立福 毛宇光	硕士 副高	042	一种改进的强制存取控制模型	计算机科学	2003.30.08	
28	金仁康 毛宇光	硕士 副高	042	关于中介命题演算系统MP^M范式的研究	青岛大学学报	2003.16.增刊	
29	金仁康 毛宇光	硕士 副高	042	中介逻辑谓词演算系统MP^M的消解原理	计算机科学	2003.30.08	
30	张仕 毛宇光	硕士 副高	042	关系数据库到XML的模型转换	计算机应用研究	2003.20.增刊	
31	单峰 毛宇光 宫宁生	硕士 副高 博士	042 042 外单位	基于两级结构神经网络的汇率预测	现代信息技术理论与应用	2003.00.00	
32	薛芹 毛宇光	硕士 副高	042	基于COM/DCOM的备份系统的设计与实现	现代信息技术理论与应用	2003.00.00	
33	宫宁生 单峰 朱梧槚	博士 硕士 正高	042	AB Neural Network Model and Applicetions	2003年第二届亚洲软件基础研讨会交流		
34	朱朝晖 李斌 肖奚安 陈世福 朱梧槚	副高 副高 正高 正高 正高	042 外单位 外单位 外单位 042	A Representation Theorem for Recovering Contraction Relations Satisfying WCI	理论计算机科学	2003.290.00	
35	朱朝晖 肖奚安 周勇 朱梧槚	副高 正高 中级 正高	042 外单位 042 042	Normal conditions for inference relations and injective Models	理论计算机科学	2003.309.00	
36	徐敏 朱梧槚	中级 正高	042	A novel interface of search engine	2003年第二届亚洲软件基础研讨会交流		
37	周勇 朱梧槚	中级 正高	042	广义粗集的公理化	南京理工大学学报	2003.27.03	
38	周勇 谢红梅	中级 硕士	042	An Argumentation Framwork for ordered Logic Programs	2003年第二届亚洲软件基础研讨会交流		
39	韩波 毛宇光 张仕 徐洁磐	初级 副高 硕士 正高	042 042 042 南大	Cbase数据库查询重写模块的设计与实现	小型微型计算机系统	2003.24.07	
40	陈兵	副高	042	Study on Seamless and Multilevel VPN	南航学报——英文版	* 2002.00.02	
41	陈兵	副高	042	基于GSS-API的SOCKS V5认证机制研究	南理工学报	2003.00.12	
42	谭晓阳	中级	042	研究式教学法在“计算机网络”教学中的应用	南航学报——社科版	2003.增刊	
43	顾其威 项阳 章璐	正高 硕士 硕士	外单位 042 042	远程弱视治疗系统的设计与实现	计算机工程与应用	2003.00.11	
44	顾其威 卢朝阳 王巍	正高 硕士 硕士	外单位 042 042	LLC3通信平台与TCP/IP互联网关设计	小型微型计算机系统	2003.24.07	
45	谭晓松 顾其威	硕士 正高	042 外单位	一个多媒体网络教学平台通信模型的设计与实现	计算机应用研究	2003.10.00	
46	胡珊 顾其威	硕士 正高	042 外单位	企业电子邮件系统安全性技术研究及实现	小型微型计算机系统	2003.24.01	

序号	姓名	职称	单位	论文题目	刊物、会议名称	年、卷、期	类别
47	胡珊 顾其威	硕士 正高	042 外单位	无人值守变电站智能化远程图像监控系统	计算机工程	2003.29.02	
48	项阳 顾其威 梁平	硕士 正高 副高	042 外单位 外单位	JAVA技术在远程弱视治疗系统中的应用研究	计算机工程	2003.29.12	
49	项阳 顾其威	硕士 正高	042 外单位	A Distributed Framework of Web-Based Telemedicine System	publication date:26-27 june 2003 on page(s): 108-113 ei 检索会议交流	2003.06.00	
50	项阳 顾其威	硕士 正高	042 外单位	Telemedicine for Amblyopia Treatment Based on Internet	南航学报——英文版	2003.00.06	
51	项阳 顾其威	硕士 正高	042 外单位	Regulation Best Effort Flows on Per Domain Basis	Proceedings of the 2003 International Conference on Computer Networks and Mobile Conputing		
52	马维华	副高	042	基于UART的SPI扩展方法	小型微型计算机系统	2003.24.03	
53	马维华	副高	042	嵌入式系统在热电厂流量积算仪中的应用	电力系统自动化	2003.23.04	
54	马维华 陶国正	副高 硕士	042	基于差分传输的多机通信可靠性	计算机应用研究	2003.20.01	
55	刘志波 陈松灿	硕士 正高	042	基于单类分类器的击键认证系统的设计与实现	2003年第四届中国生物识别学术会议交流		
56	刘志波	硕士	042	.NET中统一的存储过程调用方法	计算机应用	2003.23.11	
57	张道强 陈松灿	博士 正高	042	A novel multi-valued BAM model with improved error-correcting capability	电子科学学刊——英文版	2003.20.03	
58	张道强 陈松灿 潘志松 谭可人	博士 正高 博士 硕士	042	Kernel-Based Fuzzy Clustering Incorporating Spatial Constraints for Image Segmentation	2003年国际机器学习与控制论会议交流		
59	张道强 陈松灿	博士 正高	042	Kernel-Based Fuzzy and Possibilistic c-means clustering	2003年国际人工神经网络会议		
60	张道强 陈松灿	博士 正高	042	推广的多值指数双向联想记忆模型及其应用	软件学报	2003.14.03	
61	张道强 陈松灿	博士 正高	042	Clustering Incomplete Data Using Kernel-Based Fuzzy C-means Algorithm	神经处理快报（讯）	2003.18.06	
62	潘志松 陈松灿	博士 正高	042	具有认知能力的入侵检测系统模型	南航学报	2003.35.02	
63	潘志松 陈松灿 张道强	博士 正高 博士	042	一种基于人工免疫原理的入侵检测系统模型	数据采集与处理	2003.18.01	
64	潘志松 陈松灿 张道强	博士 正高 博士	042	HYBRID NEURAL NERWORK AND C4.5 FOR MISUSE DETECTION	2003年国际机器学习与控制论会议交流		
65	王敏 王士同 吴小俊	博士 正高 副高	042 江南大学 外单位	新模糊形态学联想记忆网络的初步研究	电子学报	2003.31.05	
66	杨凯 高航	硕士 副高	042	基于模板匹配技术的眼球定位系统的研究和实现	电子与信息学报	2003.25.增刊	
67	赵国安 高航	硕士 副高	042	由W3100A构成嵌入式网关的家庭智能系统	单片机与嵌入式系统应用	2003.00.04	
68	赵国安 高航	硕士 副高	042	H.263 IMPLEMENTATION IN EMBEDDED SYSTEM	2003年第二届亚洲软件基础研讨会交流		
69	徐涛 蔡宇新	副高 硕士	042	LOCALIZATION OF OBJECT (SPINE) IN MEDICAL IMAGE USING ACTIVE SHAPE MODELS	南航学报——英文版	2003.20.02	
70	徐涛 夏烨	副高 硕士	042	多媒体通信网络中基于优先队列的色调度算法的改进	南航学报	2003.35.04	
71	徐涛 邢汉承 骆明	副高 正高 硕士	042 东大 042	用形态学边缘检测算子实现图像的浮雕显示	数据采集与处理	2003..18.03	

此为试读，需要完整PDF请访问：www.ertongbook.com

序号	姓名	职称	单位	论文题目	刊物、会议名称	年、卷、期	类别
72	徐涛 邢汉承	副高 正高	042 东大	Localization of a Linear Structure Object in Medical Images Based on Hidden Markov Model	第十六届（IEEE）基于计算机的医学系统会议交流		
73	蔡宇新 徐涛	硕士 副高	042	基于ASM的图像中二维物体的定位方法研究	计算机应用	2003.23.增刊	
74	蔡宇新 徐涛	硕士 副高	042	面向对象系统分析和设计在数据库系统开发中的应用策略	计算机应用研究	2003.00.增刊	
75	席鹏程 徐涛	硕士 副高	042	DE-NOISING AND RECOVERING IMAGES BASED ON KERNEL PCA THEORY	2003年第二届亚洲软件基础研讨会交流		
76	兰灵	硕士	042	工作流执行逻辑的实现	航空计算技术	2003.33.03	
77	方坤 徐涛	硕士 副高	042	一种基于中间件的分布式系统的设计与实现	电子与信息学报	2003.25.增刊	
78	章勇 丁秋林	副高 正高	042	一种新的证据合成法则	东南大学学报——自然科学版	2003.33.增刊	
79	秦莉 章勇	硕士 副高	042	利用Servlet构件三层Web体系结构的研究和应用	电子与信息学报	2003.25.增刊	
80	秦莉 章勇	硕士 副高	042	在JAVA中使用连接池提高访问数据库效率	2003年中国电子学会第九届青年学术年会		
81	万麟瑞 胡宏 孙红星	副高 硕士 硕士	042	面向构件的软件开发方法学研究	小型微型计算机系统	2003.24.03	
82	郁春波 万麟瑞 茅春华	硕士 副高 硕士	042	基于UML/APN集成建模技术的ISCM模型研究	计算机应用研究	2003.20.06	
83	茅春华 万麟瑞 郁春波	硕士 副高 硕士	042	基于UML/ADL集成建模方法的集成计划模型研究	计算机应用研究	2003.20.07	
84	金永明 万麟瑞	硕士 副高	042	网络计费数据采集方法与应用实现	航空计算技术	2003.33.01	
85	戴群 陈松灿 张本柱	初级 正高 硕士	042	Improved CBP Neural Network Model with Applications in Time Series Prediction	神经网络快报（讯）	2003. 18. 03	
86	黄元元	中级	042	Binary trademark retrieval using shape and spatial feature	光学工程国际会议论文集	2003.5286.00	
87	黄元元	中级	042	Binary trademark retrieval using entropy and moments	光学工程国际会议论文集	2003.5286.00	
88	施元庆 李俊	硕士 正高	042	Kerberos系统的分析和改进方法	计算机应用研究	* 2002.精扩本（下）	
89	顾霄雷 李俊 孙传伟	硕士 正高 硕士	042 042 011	Linux操作系统实时内核实现	计算机应用研究	2003.增刊	
90	黄志球 沈国华 王传栋	正高 初级 硕士	042	THE RESEARCH ON DATA WAREHOUSE SYSTEM FOR ENGINEERING TEST	2003年第二届亚洲软件基础研讨会交流		
91	牛家浩 黄志球 张静 刘佳	硕士 正高 硕士 硕士	042	基于抽象语法树的软件度量工具的设计与实现	计算机应用	2003.23.10	
92	陶剑青 黄志球 沈国华	硕士 正高 初级	042	一个通用报表工具的设计与实现	计算机工程与设计	2003.24.14	
93	张卓莹 杨映南	副高 副高	042 外单位	ISO VT技术	中国科技发展精典文库	2003.00.下	
94	周良	中级	042	多媒体教学实践点滴	南航学报——社科版	2003.05.23	

序号	姓名	职称	单位	论文题目	刊物、会议名称	年、卷、期	类别
95	潘锦基 周良 丁秋林	硕士 中级 正高	042	基于J2EE的物料信息系统设计与研究	计算机工程与应用	2003.39.26	
96	吕嘉 周良 伍贝妮 丁秋林	硕士 中级 硕士 正高	042	“推”式技术在B2C电子商务模式中的应用研究	计算机应用研究	2003.00.增刊	
97	陈丽 周良	硕士 中级	042	XML数据集成方法研究	计算机应用研究	2003.00.增刊	
98	方圆圆	初级	042	空间滚柱凸轮曲面建模及虚拟样机计算处理方法	机械设计与制造	2003.00.05	

第 8 卷(A 版) 第 7 期
2003 年 7 月

中国图象图形学报
Journal of Image and Graphics

Vol. 8(A), No. 7
Jul. 2003

空间数据库中连接运算的处理与优化

李立言 秦小麟

(南京航空航天大学计算机科学与工程系,南京 210016)

摘 要 空间数据库的性能问题严重制约了它的应用与发展.由于空间连接运算是空间数据库中最复杂、最耗时的基本操作,因此其处理效率在很大程度上决定了空间数据库的整体性能.尽管目前已经有许多空间连接算法,但空间连接运算的代价估计和查询优化仍然有待进一步研究.众所周知,大部分空间连接算法都是基于 R 树索引实现的,如果参与空间连接运算的关系上没有索引或只有部分索引,那么就需要使用特殊的算法来处理.另外,各种算法的代价评估模型需要一个相对统一的计算方法,实践证明,根据空间数据库的实际情况,使用 I/O 代价来估计算法的复杂性较为合理.在此基础上,针对复杂的空间查询中可能出现多个关系参与空间连接运算的情况,故还需要合理地应用动态编程算法来找出代价最优的连接顺序,以便最终形成一个通用的算法框架.通过对该算法框架的复杂性分析可以看出,在此基础上实现的空间数据库查询优化系统将具有较高的时空效率,并且能够处理非常复杂的空间查询.

关键词 数据库(520·4050) 空间数据库 空间连接运算 R 树索引 动态编程 查询优化

中图法分类号: TP311.13 **文献标识码**: A **文章编号**: 1006-8961(2003)07-0732-06

Processing and Optimization of Join Operation in Spatial Database

LI Li-yan, QIN Xiao-lin

(*Dept. of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing* 210016)

Abstract The performance problem of spatial database limits its application and development seriously. Spatial join is the most complex and time-consuming operation in spatial database system. Its efficiency determines the performance of the whole spatial database system to a great extent. Although there are many spatial join algorithms already, cost estimation and query optimization of spatial join operation need further study. Most spatial join algorithms are implemented based on R-tree index, but if the corresponding relations have no indices, or only have partly indices, special algorithms should be used to handle the situation. Cost estimation models of each algorithm need a relatively uniform calculation method. Considering the characteristic of spatial database, it's reasonable to use I/O cost to estimate the complexity of each algorithm. Based on the above approaches, and because complex spatial queries may include multiple relations for spatial join, dynamic programming algorithm should be used to choose the proper join order which has minimum cost. It becomes a universal algorithm framework. Through the complexity analysis of the algorithm framework, the spatial database query optimization system implemented base on this approach will have better spatial and temporal efficiency, and can handle very complex spatial queries.

Keywords Spatial database, Spatial join operation, R-trees index, Dynamic programming, Query optimization

0 引 言

空间连接运算是空间数据库最重要的基本操作之一.它是根据某个空间谓词(例如"相交"、"包含")把来自两个关系的数据合并起来的一种操作,例如,"找出长江流经的所有省份的名称",这就是一个以"相交"为空间谓词的空间查询.

目前,空间连接处理一般分为如下两个步骤:(1)筛选(filter),该步是首先使用简单的数据结构

基金项目:国家自然科学基金资助项目(49971063);江苏省自然科学基金资助项目(BK2001045)

收稿日期:2002-08-06;**改回日期**:2003-02-27

来近似表示空间对象,其比较常用的数据结构是MBR(Minimum bounding rectangle),然后对这种简化的对象进行空间连接运算,其所得的结果集称为候选集;(2)细化(refinement),它是以筛选出的结果集为输入来逐一对真实的空间对象进行空间比较操作,看看是否真正符合原来的空间谓词.

由于细化步骤一般都具有很大的时间开销,因此在做查询优化时,应该尽量推迟这一步骤,而是先利用其他的限制条件来减小候选集,以减少时间开销.例如,"找出长江流经的所有人口多于4 000万的省份的名称",就可以先计算出各个省份与长江的MBR相交的候选集,然后从中过滤掉不符合"人口多于4 000万"这一条件的元组,最后再对真正的空间对象进行相交运算即可得到所需的结果.在这个例子中,由于前两个步骤是可以交换的,因此需要对筛选步骤的代价进行评估,以确定合适的执行顺序.

基于MBR的R树索引是筛选步骤中最常用的数据结构.本文将分别针对参与连接运算的两个关系上有两个、一个或者没有R树索引的情况给出相应的空间连接算法及其代价评估模型.

1　研究背景

R树是最常用的空间存取方法,它可以看成是B树在多维空间上的扩展[1~3].同B树一样,R树是一种可动态调整的平衡树,且每个R树结点包含若干个〈*ptr*,*mbr*〉对.对于叶结点,*ptr*是指向某个空间对象的指针,*mbr*则是对应于该对象的最小外框;对于中间结点,*ptr*指向某个子结点,而*mbr*则是该子结点所包含的所有空间对象的最小外框.

基于R树的空间连接算法[1]最早由Brinkhoff等人提出.如果参与连接运算的两个关系上都有R树索引,那么该算法将同步进行深度优先遍历,同时排除与MBR不相交的子树.由于该算法具有很高的效率,因此至今仍是最重要的空间连接算法之一.Huang等人提出了一种基于广度优先搜索的改进型R树连接算法[4],该算法当缓存空间足够大时,具有更高的效率.

当仅有一个关系上有R树索引时,可以使用传统的嵌套索引循环(indexed nested loop)连接算法,即通过循环搜索R树索引来逐个匹配无索引关系中的每个元组,但这种方法只适用于关系非常小的时候,在通常情况下效率极低.为此,Lo和Ravishankar提出了种子树连接算法(seeded tree join)[5],该算法首先以已有的R树为种子来对没有索引的关系建立一棵类R树索引,然后调用R树连接算法来完成连接运算.

如果参与连接运算的两个关系上都没有索引,则可以使用空间哈希连接算法(spatial hash-join)[6].该算法是使用采样信息来划分第1个关系,同时创建若干个"桶(buckets)";然后把第2个关系按照同样的桶来划分.这样划分完成后,就可以在范围相同的桶上执行连接操作了.

Patel和DeWitt提出了另一种基于哈希算法的空间连接算法,称为"基于划分的空间归并连接(partition based spatial merge join)"算法[7],该算法首先将空间进行正规划分,然后将参与连接的两组空间对象映射到分区上,再使用空间扫描算法把覆盖了同样区域的分区组连接起来.由于某些对象可能会占用多个分区,所以需要把连接运算的输出结果进行排序,以便删除重复的元组.而这种排序操作可能会带来很大的I/O开销.

把以上这些空间连接算法组合起来,再针对不同的情况使用相应的算法,就可以处理空间数据库中的各种连接操作了,然而,如果参与连接运算的关系数目多于两个,则连接顺序的选择就变得十分重要.这是因为不同的连接顺序所产生的时间和空间代价相差很大,而连接顺序的组合会随关系数目的增加成指数增长的缘故.为了确定合适的连接顺序,首先需要估计各个算法的查询代价,尽管实际的查询代价往往会受到数据分布情况的影响,但是代价估计的目的不是准确的计算代价大小,而是帮助选择一个物理查询计划.由此可见,只要估计方法能够反映各个查询计划的相对代价大小就可以达到目的.有了算法的查询代价后,接下来还需要找到总代价最小的连接顺序.为了避免穷举法的低效和爬山法的盲目性,本文使用了动态编程算法,该算法是采用"自底向上"的策略,由于在计算较大子表达式的计划时,只考虑每个子表达式的最佳计划,因此具有较高的效率,而且一般都能找到最优计划.

2　典型算法及代价评估

2.1　R树连接算法

R树连接算法的核心是对两棵R树同步进行深度优先遍历,当两个结点的*mbr*相交时,则递归

地进入下一层扫描，直到叶结点相交，输出结果为止[1]，其算法的伪代码如下：

算法 1 R 树连接算法

```
r_tree_join(Node I1, Node I2)/* I1、I2—R 树的结点 */
{
    for(I1 中的每个元素 E1)
      for(I2 中的每个元素 E2)
        if (overlap(E1.mbr, E2.mbr))
          if (I1,I2 是叶结点)
            /* 输出到结果集 */
            output(E1.ptr, E2.ptr);
          else if (I1 是叶结点) {
            /* 从磁盘或缓冲区读入页面 */
            read_page(E2.ptr);
            r_tree_join(E1.ptr, E2.ptr);
          } else if (I2 是叶结点) {
            read_page(E1.ptr);
            r_tree_join(E1.ptr, E2.ptr);
          } else {
            read_page(E1.ptr);
            read_page(E2.ptr);
            r_tree_join(E1.ptr, E2.ptr);
          }
}
```

换个角度看，R 树连接算法实际上是一系列的窗口查询(window queries)，其分别由 I_1 和 I_2 结点的元素作为数据和查询窗口.

在 R 树连接算法中，由于每个结点都可能会被多次存取，因此内存缓冲区管理策略对算法的代价影响很大. 假设内存页面缓冲采用最近最少使用法 LRU(Least recently Used)策略，则 R 树连接算法的代价可表示为[2]：

$$C_{RJ}=T_1+T_2+(N_A(I_1,I_2)-T_1-T_2)P(M) \quad (1)$$

其中，T_1、T_2 表示这两棵 R 树所占用的内存页面数；$N_A(I_1,I_2)$表示算法所存取的 R 树结点的总数；$P(M)$表示所要存取的结点不在缓冲区的概率，M 是缓冲区的大小. 关于 $N_A(I_1,I_2)$和 $P(M)$的详细计算方法请参见文献[2].

2.2 种子树连接算法

设空间对象集 A、B 参与连接运算，其中，A 上有 R 树索引 R_A，B 上无索引. 种子树连接算法可分为如下 4 个步骤：

(1) 采种(seeding)过程，根据系统参数(主要考虑对象数目和内存大小)计算采种的高度 k，把 R_A 顶部的 k 层拷贝作为种子，其中最底层结点中的项称为槽(slots). 拷贝完成后，再把槽中的指针清空.

(2) 生长过程，把 B 中的每个对象插入到种子树中，从根节点开始，先找到合适的槽，如果是第 1 次插入此槽，则需要分配一个新的结点；然后通过调整槽指针指向该结点来把对象插入到该新结点中；如果槽已经生成了子结点，则视此结点为 R 树根结点；最后按照 R 树的插入算法把对象插入 R 树中合适的位置. 槽下面的这种 R 树称为生成子树(grown subtree).

(3) 清理过程，按照槽的子结点的 mbr，逐层向上调整种子中结点的 mbr，如果槽中没有子结点，则删去此槽，同时相应调整上层的 mbr.

(4) 树匹配(tree matching)过程，此过程相当于用 R 树连接算法来处理 R_A 和生成的种子树.

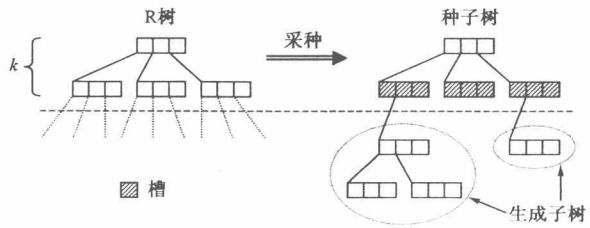


图 1 种子树的构造

种子树并不是一棵严格的 R 树，由于插入各个槽的对象数目可能相差很大，很可能导致生成子树的高度不相等，所以种子树不是一棵平衡树，但是由于基于深度优先的树匹配算法并不要求参与运算的树是平衡树，所以仍然适用. 种子树的优点是提高了树匹配的效率，由于种子树与提供种子的 R 树相应的各个结点之间有相同或相似的范围，所以匹配时的命中率很高，而与结点相交的比例则很低. 由于种子树连接算法在生长过程中占用内存很大，因此如果树的大小超出了内存缓冲区的大小，那么就只有把一部分写入磁盘，需要时再读出. 为了避免插入对象时频繁地读写磁盘，可以把将要插入某个槽的对象先存储到对应的临时文件中，这样当所有的对象都插入后，再把各个槽对应的临时文件读出来生成 R 树，以便成为该槽的生成子树. 这种方法由于仍然要求每个槽至少分配到一个内存页面，才能有效避免缓冲区摆动(buffer thrashing)，因此应该通过修正系统参数来影响槽的数目，以满足需要.

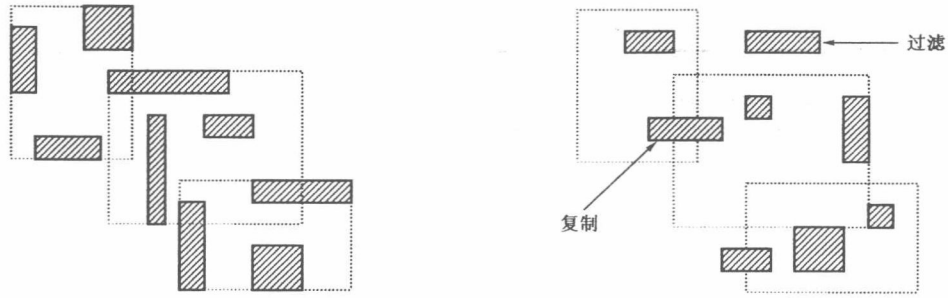
为了估计种子树算法的代价，可首先假设系统内存足够大，这样虽能够避免缓冲区摆动，但是不一定能容纳整个种子树，因此需要在生长阶段使用临

时文件. 在采种阶段,算法的代价为 $s_A \cdot T_A$,即从 R_A 树上拷贝 k 层所存取的页面数,其中 s_A 表示 R_A 树中前 k 层结点所占的比例;在生长阶段,B 中的每个空间对象需要被读出并写到某个槽对应的临时文件中,最后还要读出来创建生成子树,其存取代价为 $3P_B$;清理阶段本来没有存取操作,不过清理后的整个种子树还要写回磁盘,其代价为 T_B;树匹配阶段的代价约为 T_A+T_B,即两棵树的所有页面都被读入的代价. 这样,种子树连接算法的总代价[3]可表示为

$$C_{STJ} = (1 + s_A)T_A + 3P_B + 2T_B \qquad (2)$$

2.3　空间哈希连接算法

空间哈希连接算法是由基于传统关系模型上的哈希连接算法扩展而来的[6]. 设两个空间对象集 A、B 参与连接运算,运算时首先根据系统参数(主要考虑对象数目和内存大小)来计算桶的数目,然后对 A 进行采样,以确定各个桶的初始范围,再把 A 中的对象插入到各个桶中,同时根据需要扩充桶的范围;最后把 B 中的对象插入到同样的桶中,如果某个空间对象覆盖了两个以上的桶区域,则要再复制相应份数的该对象,并将其插入到每个相交的桶中,这样不与任何桶相交的对象就被过滤掉了,而且这种桶划分的质量直接影响空间哈希连接算法的效率. 由于采样的不确定性,无法保证 A 被均匀划分(即各个桶中所包含的对象数目接近相等),且由于 A 和 B 中对象的空间分布情况不同,更无法保证 B 被均匀划分,因此改进采样和桶划分的质量是提高算法效率的关键.



(a) A 划分为 3 个桶　　(b) B 中对象插入这 3 个桶

图 2　空间哈希连接的划分过程

划分完成以后,A 和 B 就可以在相同的桶区域上直接进行连接运算. 当某个桶中的 A 或 B 的对象子集能够全部装入内存时,则只需对这两个集合进行单次扫描,即可完成这个桶中的连接运算,但是如果假设不能成立,那么就只好先对桶中的 A 或 B 的对象子集建立 R 树索引,再使用嵌套索引循环连接算法进行连接运算,但由于这样做的代价很高,因此应该通过修正系统参数来增加桶的数目,以尽量避免这种情况.

如果要估计空间哈希连接算法的代价,则需假设内存足够大,即能够容纳单个桶中的 A 或 B 的对象子集. 在空间对象集 A 的划分阶段,采样代价为 cD,其中,D 为桶的数目,c 是一个常数,表示随机 I/O 的代价,采样后对 A 的划分代价为 $2P_A$,即 A 中每个对象读写一次的代价;在空间对象集 B 的划分阶段,读取代价为 P_B,写入代价为 $(r_B-f_B)P_B$,其中 r_B 表示重复对象数,f_B 表示过滤对象数;连接阶段代价为 $P_A+(r_B-f_B)P_B$,即写回磁盘的空间对象全部读取一次的代价. 空间哈希连接算法的总代价[3,8]为

$$C_{HJ} = cD + 3P_A + (1 + r_B - f_B)P_B \qquad (3)$$

3　基于动态编程的查询优化

空间数据库常常要处理多个空间关系的连接,其最关键的问题是确定空间连接的顺序. 动态编程算法提供了一种自底向上的优化策略,它能够以较高的效率从可能的查询计划中找出代价较小的计划. 由于动态编程算法没有考虑全局优化,因此可能在某些情况下找不到最优计划,但是通过适当的改进(例如 Selinger 风格优化[9]),就可以在大多数情况下找到代价最优的计划.

动态编程的基本思想是对一个代价表进行填充,尔后只保留推出最终结论所需的最少信息. 假设有 n 个关系 $R_1, R_2, \cdots, R_n$ 参与空间连接,则需要为包含这些关系中的两个到 n 个关系的每一个"可能"的子集构建一个表项,"可能"是指这个子集中的关系能够通过空间谓词连接起来,换句话说,若以关系

此为试读,需要完整PDF请访问: www.ertongbook.com

为顶点,以空间连接谓词为边来构造一个查询图,则此图的每个含有两个及两个以上顶点的连通子图应占一个表项. 每个表项 Q 代表一种查询策略,它包含如下 3 种信息:

(1) 这个子集中的全部关系连接的最小代价(Q. cost);

(2) 达到最小代价的连接计划(Q. plan);

(3) 这些关系连接结果的估计大小(Q. size).

最小代价可以按照上一节给出的公式来估计,而连接结果的估计大小则与连接算法、连接次序无关,只与数据的分布情况有关. 假设数据为正态分布,则连接结果的大小 S(A,B)[2,3]可表示为

$$S(A,B) = S_B \cdot S_A \cdot (r_A + r_B)^2 \qquad (4)$$

其中,r_A 表示集合 A 中空间对象的 *mbr* 的平均边长相对值,其取值范围被标准化到[0,1];S_B,S_A 分别为集合 A、B 中元素个数.

综合以上结论,即可实现基于动态编程的空间连接查询优化算法,其框架如下:

算法 2 空间连接查询优化算法

```
optimize(Query Q, int n)
/*Q—逻辑查询计划,其中包括关系集{R1,…,Rn};n—关系数 */
{
  for (each {A,B} in {R1,…,Rn} and {A,B}可连接) {
  /* 所有具有连接谓词的关系对 */
    if (A,B 有索引) {
      Q{A,B}.cost=CRJ(A,B);/* 使用式(1)*/
      Q{A,B}.plan=[RJ,A,B];/* 使用广义表保存查询计划 */
    } else if (A 有索引) {
      Q{A,B}.cost=CSTJ(A,B);/* 使用式(2)*/
      Q{A,B}.plan=[STJ,A,B];
    } else if (B 有索引) {
      Q{A,B}.cost=CSTJ(B,A);
      Q{A,B}.plan=[STJ,B,A];
    } else {
      Q{A,B}.cost=CHJ(A,B);/* 使用式(3)*/
      Q{A,B}.plan=[HJ,A,B];
    }
    Q{A,B}.size=S(A,B); /* 使用式(4)*/
  }
  for (i=3 to n)
    for (each {T1,…,Ti} in {R1,…,Rn} and {T1,…,Ti}可连接) {
    /* 所有具有连接谓词的关系集 */
      Q{T1,…,Ti}.cost=MAX; Q{T1,…,Ti}.plan=NULL;
      for (each 分解 Q{T1,…,Ti}→{Q1,Q2} and {Q1,Q2}可连接) {
      /* 把关系集分解为两个具有连接谓词的关系子集 */
        if (|Q1|=1 and Q1 中的关系 R 有索引) {
        /* 单关系子集,且该关系上有索引 */
          {Q1,Q2}.cost=Q2.cost+CSTJ(Q1,Q2);
          {Q1,Q2}.plan=[STJ,Q1.plan,Q2.plan];
        } else {
          {Q1,Q2}.cost=Q1.cost+Q2.cost+CHJ(Q1,Q2);
          {Q1,Q2}.plan=[HJ,Q1.plan,Q2.plan];
        }
        if ({Q1,Q2}.cost<Q{T1,…,Ti}.cost) {
        /* 只保留已知的代价最小的查询计划 */
          Q{T1,…,Ti}.cost={Q1,Q2}.cost;
          Q{T1,…,Ti}.plan={Q1,Q2}.plan;
          Q{T1,…,Ti}.size=S(Q1,Q2);
        }
      }
    }
}
```

该算法步骤为:先计算所有两个关系连接的代价,然后逐渐增加关系数,当计算 i 个关系的连接代价时,由于从两个到 $i-1$ 个关系的连接代价都已经算出,因此只需将其分解即可;同时由于中间关系都没有索引,所以只有当分解出的某项只含有一个关系,并且该关系上有索引时,才能使用种子树连接算法,否则就只能使用哈希连接算法.

该算法中用到了 C_{RJ}、C_{STJ}、C_{HJ} 和 S 这 4 个估计函数,分别对应于前面所描述的 4 个公式. 这 4 个函数均以两个关系作为输入参数,通过对这两个关系进行统计信息的分析,即可计算出连接的代价或大小. 其中,原始关系的统计信息由空间数据库直接提供,而中间关系的统计信息则需要通过估计函数来间接计算得到.

例如,设有 3 个关系 R_1、R_2、R_3 进行空间连接,3 个关系之间都有连接谓词,其中,T_1 上没有索引,R_2、R_3 上有 R 树索引. 在寻找最佳连接顺序时,首先计算这 3 个关系两两连接的代价,即得到中间关系 $R_{1,2}$、$R_{2,3}$ 和 $R_{1,3}$ 的估计信息,其中,计算 $R_{1,2}$ 和 $R_{1,3}$ 时,使用种子树连接算法;而计算 $R_{2,3}$ 时,则使用 R 树连接算法,同时每个中间关系的大小也要估算出来;接下来即可利用这 3 个中间关系来计算 R_1、R_2、R_3 连接的总代价,其中,计算 $R_{2,3}$ 与 R_1 连接时,使用

空间哈希连接算法;而计算 $\boldsymbol{R}_{1,2}$ 与 $\boldsymbol{R}_3$、$\boldsymbol{R}_{1,3}$ 与 $\boldsymbol{R}_2$ 连接时,则使用种子树连接算法;最后根据总代价的大小,就可以找到最佳的连接策略.

动态编程算法的时空复杂度与输入数据之间的关系有关,对于算法 optimize 而言,时空开销最大的情况就是任意两个输入关系之间都有连接谓词,即连接关系图是一个完全图.在这种情况下,算法的第 i 层需要在代价表中存储 $C(i,n)$ 个表项,其总的空间开销为

$$C_{\text{space}}=C(2,n)+C(3,n)+\cdots+C(n,n)=2^n-n-1$$

同样,在算法的第 i 层需要进行分解,其时间开销为

$$C(i,n)\cdot(C(1,i)+C(2,i)+\cdots+C(i-1,i))/2$$
$$=C(i,n)\cdot(2^{i-1}-1)$$

而总的时间开销为

$$C_{\text{time}}=C(2,n)+\sum_{i=3}^{n}(C(i,n)\cdot(2^{i-1}-1))$$
$$=\sum_{i=2}^{n}(C(i,n)\cdot(2^{i-1}-1))$$

4 结　论

本文讨论了空间连接运算的处理和优化过程中的关键技术问题,总结了基于 R 树索引的几种空间连接算法及相应的代价模型,并使用动态编程的思想把这些算法整合起来,给出了实现空间查询优化的算法框架.

通过设计与实现一个基于 Realms[10] 的空间分析模块,并集成进 PostgreSQL 中,从而实现了一个空间分析数据库原型系统 SADBS(Spatial analysis database systems).在此基础上,笔者等正在研究和实现本文所描述的空间查询优化技术,目前已经完成了 R 树索引模块与空间分析模块的集成,而且各种空间连接算法和动态编程算法也有了初步实现,正在进一步完善之中.

参 考 文 献

1 Brinkhoff T, Kriegel H P, Seeger B. Efficient processing of spatial joins using R-trees[A]. In: Proceedings of the 1993 Association for Computing Machinery Special Interest Group International Conference on Management of Data [C]. Washington, D. C. USA, 1993: 237～246.

2 Huang Y W, Jing N, Rundensteiner E A. A cost model for estimating the performance of spatial joins using R-trees[A]. In: Proceedings of Ninth International Conference on Scientific and Statistical Database Management[C]. Olympia, Washington USA, 1997: 30～38.

3 Mamoulis N, Papadias D. Integration of spatial join algorithms for processing multiple inputs[A]. In: Proceedings of the 1999 Association for Computing Machinery Special Interest Group International Conference on Management of Data [C]. Philadelphia, Pennsylvania USA, 1999: 1～12.

4 Huang Y W, Jing N, Rundensteiner E A. Spatial joins using R-trees: Breadth first traversal with global optimizations[A]. In: Proceedings of 23rd International Conference on Very Large Data Bases[C]. Athens, Greece, 1997: 396～405.

5 Lo M L, Ravishankar C V. The design and implementation of seeded trees: an efficent method for spatial joins [J]. IEEE Transactions on Knowledge and Data Engineering, 1998, 10(1): 136～152.

6 Lo M L, Ravishankar C V. Spatial hash-joins [A]. In: Proceedings of the 1996 Assciation for Computing Machinery Special Interest Group International Conference on Management of Data[C]. Montreal, Cananda, 1996: 247～258.

7 Patel J M, DeWitt D J. Partition based spatial merge join[A]. In: Proceedings of the 1996 Association for Computing Machinery Special Interest Group International Conference on Management of Data[C]. Montreal, Cananda, 1996: 259～270.

8 Koudas N, Sevcik K C. Size separation spatial join [A]. In: Proceedings of the 1997 Assciation for Computing Machinery Special Interest Group International Conference on Management of Data[C]. Tucson, Arizona USA, 1997: 324～335.

9 Garcia-Molina H, Ullman J D, Widom J. Database system implementation [M]. Upper Saddle River, New Jersey USA: Prentice Hall, 2000: Chapter 7, Section 6.

10 Güting R H, Schneider M. Realms: A foundation for spatial data types in database systems[A]. In: Proceedings of the 3rd International Symposium on Large Spatial Databases [C]. Singapore, 1993: 14～35.

李立言　1978 年生,南京航空航天大学硕士研究生.研究方向为空间数据库、主存数据库.

秦小麟　1953 年生,硕士学位,现任南京航空航天大学教授,博士生导师.当前的研究方向为空间分析 DBMS、时空数据库、GIS、安全数据库等.

时空数据库索引机制研究*)

黄曙荣[1,2]　秦小麟[1]

(南京航空航天大学信息科学与技术学院　南京210016)[1]　(盐城工学院计算机工程系　江苏盐城224003)[2]

Research of Index Mechanisms in Spatio-temporal Databases

HUANG Shu-Rong[1,2]　QIN Xiao-Lin[1]

(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing Jiangsu 210016)[1]

(Department of Computer Engineering, Yancheng Institute of Technology, Yancheng Jiangsu 224003)[2]

Abstract　Spatio-temporal database manages the large amount of spatial objects that change over time. It is necessary to query the spatio-temporal objects of the past and the current and to anticipate the future of spatio-temporal objects. It is important to design an efficient index mechanism for accessing the spatio-temporal data efficiently. The paper analyzes the features of the spatio-temporal objects, studies the methods of spatio-temporal index mechanisms, classifies the index mechanisms, and discusses the key technologies of spatio-temporal indexes. And it also presents the index methods of STADBS that we are studying.

Keywords　Spatio-temporal databases, Spatio-temporal objects, Spatio-temporal indexes, STAM

1　引言

时空数据库(Spatio-temporal Database)是指存储随着时间发生变化的空间对象的数据库系统,这里的变化包括对象的位置和/或范围的变化。时空数据库的应用涉及到全球变化(如气候、陆地的变化)、交通运输(交通监控、智能传输系统)、社会(人口统计、健康)及多媒体系统等多个方面。

因为涉及到时间因素,时空数据库需要管理大量的带有长期时间信息的空间数据,为了对这些数据进行有效存取,对时空数据库索引机制的研究显得尤为重要。但迄今为止,对时空数据库的研究仍处于起步阶段,且主要集中于对时空模型和查询语言的研究,在时空索引方面的研究还很有限。一方面,对空间索引的研究成果可以支持对点、任意形状的对象及光栅数据的查询操作,但没有考虑到时空应用中的时间特性;另一方面,已提出的时间存取方法只对有效时间(valid-time)和/或事务时间(transaction-time)进行索引以支持时间数据库的索引,没有考虑到时空对象的空间属性。

本文分析了时空对象的特点,研究了时空索引机制的方法并进行了分类,讨论了时空索引中的关键技术,并对我们正在研制的时空分析数据库管理系统 STADBS 中的索引方法作了介绍。

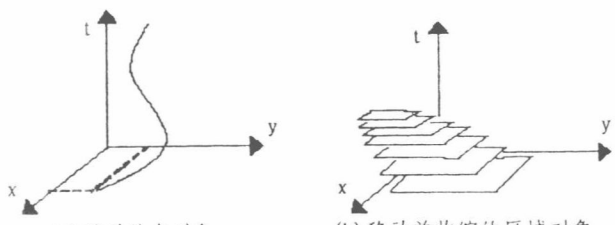
2　时空对象(spatio-temporal object)

2.1　时空对象的概念

空间位置和/或范围随着时间的变化而发生变化的空间对象称为时空对象(也称为移动对象),如正在飞行的飞机、高速公路上行驶的小汽车、随着时间的变化其边界范围在扩大或缩小的森林等等。有大量的应用需要存储和管理时空对象,查找对象在过去、现在的位置和范围,推测其在未来某一时间戳(time-stamp)或时间间隔(time-interval)的行为。

时空数据库存储的是随时间变化的空间对象。在空间数据库中定义了的三种基本空间对象:点(points)、线(lines)和区域(regions)。点表示空间范围为零或只关心其空间位置,而不是它的大小的空间对象,例如在一张大比例的地图中的一个城市;线即空间曲线,可描述在空间中移动的设施或者空间连线,例如道路、河流、电话线、电线等;具有一定空间范围的对象称为区域,例如一个国家行政区、一片湖泊等。因此,目前对时空数据类型的研究主要集中于移动点(moving points)和移动区域(moving regions)这二种基本时空对象数据,由于线(曲线)本身就是运动的抽象或者投影,可转换为对移动点轨迹(trajectory)的研究。

二维参考空间(x,y)中的点对象随时间变化情况可用三维参考空间的曲线表示,区域对象的变化不仅包括空间位置的改变,还包括形状的变化,如扩张、收缩等。图1(a)、(b)分别给出了二维参考空间中移动点和移动区域的例子(第三维是时间维)。本文讨论的是二维参考空间中的空间对象,同样的讨论也可扩展到三维或更高维的空间。



(a) 移动的点对象　(b) 移动并收缩的区域对象

图1　二维参考空间中的移动对象

2.2　时空对象的表示

时空对象 O_i 可用记录$\langle o_id, p_i(t), e_i(t), t\rangle$来表示[1]。其中 o_id 是对象标识符,它唯一标识每个对象 O_i;$p_i(t)$和 $e_i(t)$分别对应于对象 O_i 在时间 t 的位置和范围。在各种时空数据

*)国家自然科学基金资助项目(69973022)。黄曙荣　硕士研究生,研究方向为时空对象索引技术。秦小麟　教授,博士生导师,当前研究兴趣为空间分析数据库、时空数据库、GIS 等。

库的应用中一般允许在任意时间戳插入和/或删除对象，对象的生命期(lifetime)从对象插入时开始到被删除时结束，在生命期内的对象称为活动的(alive)，否则为不活动的(non-alive)。时空数据库在时间戳 t 的状态 S(t)包括所有在时间戳 t 活动的对象，可将 S(t)看成是时空对象在时间戳 t 的空间位置和范围的快照。因为时空数据库存储的是随时间变化的空间对象，理论上讲它保存了所有的时空状态 S(t)，并可对任意时间戳的 S(t)进行查询，这意味着时空数据库中时空对象数据的删除仅是逻辑删除，即已删除对象的记录仍保存在数据库中(成为不活动的对象)。

2.3 时空对象的时间维特点

时空数据库可看成是空间数据库在时间维上的扩展，在分析时空数据时应考虑到其时间维的特点。

(1)时间信息的变化总是单调递增的。

(2)在时间数据库中，定义了两个时间维度：有效时间和事务时间。有效时间是指现实世界中事件发生的时间，事务时间指信息存储到数据库中的时间。支持事务时间数据库的特点是各种变化的发生时间是按递增顺序的，且通常是对数据库的当前状态操作，这种数据库记录了对象变化的整个过程(历史)。时空数据库也应支持有效时间、事务时间或双时间(bitemporal)。

(3)数据库的主要目标是准确反映现实世界，空间对象随时间的递增，其变化频率的快慢决定了数据库表示时空数据的方式，可分为离散(discrete)和连续(continuous)两种情况。离散情况下，空间对象的变化频率慢，可用时空状态 S(t)的快照(snapshot)集表示时空数据，在时空对象发生变化时更新数据；连续情况下的空间对象变化频率快，其空间位置和范围随时间的变化情况可分别表示为时间函数 $p_i(t)$、$e_i(t)$，且只在对象变化的特征(如运动的速度、方向等)改变时才更新数据。

3 时空索引的分类

参照 Y. Theodoridis 等在文[2]中对时空索引的描述，结合目前在时空索引方面已取得的研究成果，时空索引可作如下分类：

(1)*按处理的数据类型分类* 主要的时空数据类型包括点数据和区域数据。一部分索引只处理点数据[3,4]或移动点的轨迹，支持轨迹查询[5~9]；处理区域数据的存取方法既能处理点数据又能处理非点数据，因为点对象可看成是大小为零的区域对象，如 3D R-树、RT-树、HR-树、PPR-树及 MV3R-树等[10~14]。

(2)*按数据加载的方式分类* 空间索引根据索引创建时数据是以批处理方式全部载入还是动态插入/删除生成索引结构分为静态的(static)和动态的(dynamic)两类，但在时空数据库中，时间维的存在意味着所有对过去时间戳的空间数据插入或更新均是无意义的(无论是静态载入还是动态插入)。因此，时空索引可分为静态的、按时间顺序的(chronological)和动态的三种类型。静态类型和按时间顺序类型均不允许对过去时间戳的数据插入或更新，前者是以批处理方式载入数据(如3D R-树)，后者按时间戳的递增顺序动态插入数据(RT-树、HR-树、PPR-树等)；动态类型允许动态插入数据，且可对过去时间戳的数据插入或更新。

(3)*按数据集的更新情况分类* 根据空间对象是静止/移动及时间是静态/动态的不同，时空索引可按数据集的更新情况分为增长的(growing)、变化的(evolving)和完全动态的(full-dynamic)三类。增长类型是指对象静止不动，而时间是动态递增的[10]；时间是静态的，而对象是移动的，这样的索引属变化类型；支持时间动态变化情况下的移动对象称为完全动态类型(full-dynamic)，目前研究的索引结构大多是支持时间动态变化的[11~14]。

(4)*按时空对象的近似表示分类* 部分索引结构(如3D R-树)采用 R-树中用最小边界矩形(MBRs)表示空间对象的方法表示时空对象。由于时空对象独有的移动特性，这种表示方法会导致大量的无效空间和 MBRs 的重叠，并不能很好地支持时空查询，文[7,12,13]中各自提出了不同的时空对象近似表示方法。

(5)*按是否支持特定的时空查询分类* 索引的目的是有效支持用户定义的各种查询操作。基本的空间查询包括选择查询(selecttion queries)、连接查询(join queries)和最近邻居查询(nearest-neighbor queries)。时空索引除了要支持这三种基本查询外，还应支持时间查询和复杂的时空查询，如时间片查询(timeslice queries)(时间片指特定的时间戳或时间间隔)、轨迹查询(trajectory queries)以及历史查询(history queries)等：RT-树索引结构仅支持基本的时空查询，HR-树、PPR-树支持对时间戳的有效查询，3D R-树和 MV3R-树可同时支持时间戳、时间间隔的查询；STR-树[8]、TB-树[9]以及 TPR-树[7]可支持对移动点的轨迹查询。

(6)*按支持的时间维度分类* 时空数据库应至少包含一维时间信息，时空索引可分为支持有效时间、事务时间和双时间的索引机制。目前大多数的时空索引均支持事务时间，文[15]中提出了支持双时间查询的索引结构 R^{st}-树。

(7)*按离散/连续的情况分类* 目前大多数的索引结构均是基于离散情况提出的(如3D R-树、HR-树、RT-树以及PPR-树等)，可以查询时空对象过去及当前的状态；连续情况下可根据时空对象位置和/或范围随时间变化的函数建立索引，不仅可以查询对象过去、当前的位置和/或范围，还可根据对象当前的信息推测出它在将来某个时间戳的行为，文[5~9]讨论了支持连续情况下查询的索引结构。

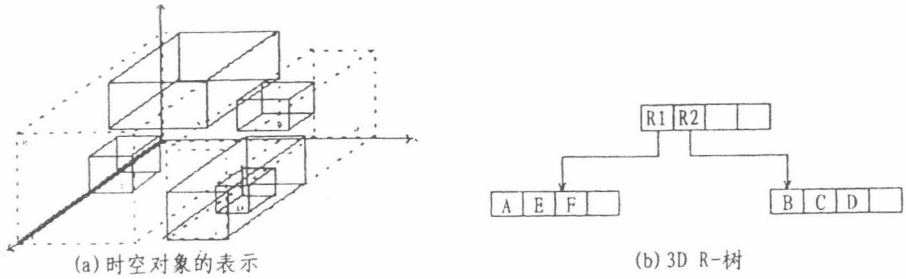
4 离散情况下的时空索引机制

离散情况下的时空存取方法(STAM)主要分为以下三类：

(1)*将时间维看成是另一个空间维度*

将时间看成是时空对象另一维度的空间信息，这样不仅简单，且可直接使用传统的空间存取方法(如 R-树、四叉树)处理 d+1维的空间信息(d 是参考空间的维度)。文[10]提出的3D R-树将二维参考空间中的最小边界矩形转换成三维空间的最小边界矩形(MBRs 的高度对应于对象的生命期)，以表示位置及范围均不随时间发生变化的时空对象，并使用 R-树索引结构存储时空数据。图2给出了时空对象实例及其对应的3D R-树。

3D R-树实现较容易，可有效地支持时间片查询，缺点是没有考虑到时间维的特有属性，生命期长或/和位置变化大的对象，都会对应较长/大的三维 MBRs，引起3D R-树中结点 MBRs 的大量重叠，会降低查询的效率。



(a) 时空对象的表示　　(b) 3D R-树

图2　用 R-树结构存储三维的 MBRs

(2) 对象的时间信息与空间信息一起保存

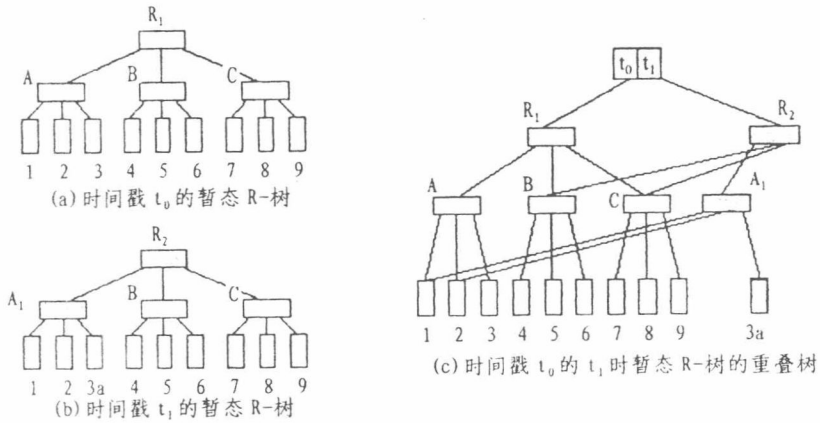
以 MR-树和 RT-树[11]为代表的存取方法是将时间信息和空间信息一起存储到 R-树结点中，无论是叶结点还是非叶结点中的实体，都以〈S,T,P〉格式存储，其中 S 表示空间信息(MBRs)，T 是时间信息(时间间隔)，P 为指向子树或具体对象的指针。这里 $T=(t_i,t_j)$，$i\leqslant j$，t_i 表示当前时间戳，t_{j+1}是下一个时间戳($t_i\leqslant t_{j+1}$)，如果对象在时间戳 t_i 到 t_{j+1}期间空间位置不变，则实体的空间信息 S 保持不变，而时间信息由 T 更新为 T'(T'=(t_i,t_{j+1}))；一旦对象的空间位置发生变化，则原实体保持不变，产生新的实体〈S',T',P'〉，(S' 是对象新的空间位置，T'=(t_{j+1},t_{j+1}))插入到 RT-树中。

这种结构较适用于对象移动较少的时空数据库，如果数据库中对象的移动量较大，实体数目的增长会很可观。另外，每个实体中均包含对象实例的空间信息和时间信息，插入和分裂策略很难同时满足对对象的空间和时间属性的查询要求(如插入溢出时按哪个属性分裂结点)。

(3) 使用部分持久和重叠技术存储不同时间戳时空状态的方法

部分持久结构[16](partially persistent structure)将二维空间存取方法(如 R-树、四叉树或 k-d 树结构)转换为部分持久的，它在"逻辑"上存储过去的时空状态 S(t)，并只对当前时空状态数据更新。对时空状态 S(t)的历史查询可直接对时空状态在时间戳 t 的部分持久结构操作(可看成是对暂态 R-树的操作)，返回结构中在时间戳 t 的活动对象。该结构避免了3D R-树中长生命期对象引起的结点 MBRs 重叠问题，可有效地支持时间戳和时间间隔查询。

重叠(overlapping)技术的思想是：用二维空间存取方法为每个时间戳建立一棵对应的暂态树(如暂态 R-树)，并假设相邻时间戳的暂态树的区别不大，存在许多共同路径(路径上结点内空间对象没有随时间变化)。为了节省磁盘空间，相邻时间戳的暂态树间可共享这些路径，而那些包含空间对象变化的结点及路径可更新后复制到后续暂态树中。如图3所示，图3(a)、(b)分别对应时间戳 t_0和 t_1的暂态 R-树，如果在时间戳 t_1结点3数据更新(假设是结点3中一个空间对象被删除)，则时间戳 t_1的暂态 R-树与 t_0时的区别仅是创建了新的结点3a和到达结点3a 的新路径，其余结点和路径均与 t_0时相同，可共享，两棵暂态 R-树有重叠部分，见图3(c)。



(a) 时间戳 t_0的暂态 R-树

(b) 时间戳 t_1的暂态 R-树

(c) 时间戳 t_0的 t_1时暂态 R-树的重叠树

图3　暂态 R-树及重叠树

采用重叠的方法操作简单，可适用于多种空间存取方法(如 R-树、四叉树等)。文[12～14]提出了各种重叠的 R-树，其中 HR-树是最有代表性的索引结构；在文[17,18]中讨论了用重叠的线性四叉树存储随时间变化的光栅数据。重叠的方法在对象变化频繁的时空数据中对存储空间的要求较高(最终退化为相互独立的暂态树)。

文[12]提出的 PPR-树更好地结合了部分持久结构和重叠的方法，对长生命期时空对象的 MBRs 进行了分裂优化(减少无效空间)，可支持快速查询，减少对存储空间的要求。

5　连续情况下的时空索引机制

连续情况下的研究热点是如何对连续移动点的轨迹索引。

(1) 移动点过去轨迹的索引方法

对连续移动点过去轨迹的索引一般可转化为离散情况下的索引问题。例如，在平面上作直线运动的点对象，其运动轨迹可看成是若干线段的集合。对象的移动变化意味着在原来轨迹的末端插入新的线段。如果将每条线段都当作是一个空间对象(如点或区域对象)处理，则连续点的移动变化相当于

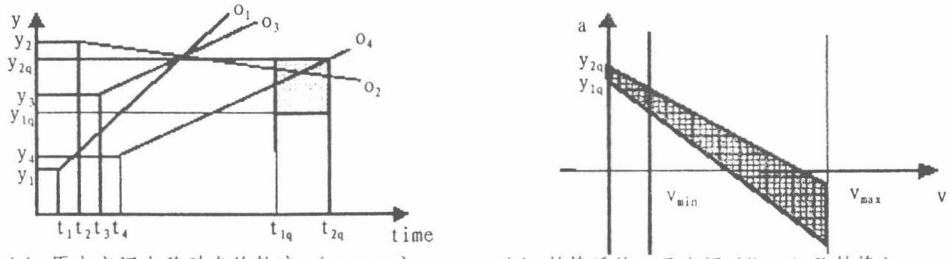
在数据库中插入和删除线段对象，这正是在离散情况下时空对象移动的特殊问题(即只有对象的插入和删除，对象的位置和范围不作变化)，可对离散情况下的时空存取方法进行扩展后创建索引，如STR-树、TB-树。

(2)支持移动点未来位置查询的索引方法

连续情况下移动点的运动轨迹可用时间函数(如$p_i(t)$)表示，根据$p_i(t)$及已知的某一时间戳移动点的位置来预测其未来位置。

创建索引的一种简单方法是假设点对象在d-维(d=1，2，3)空间移动，这些点未来的移动轨迹可当作d+1-维空间的线对象进行索引。如J. Tayeb等将一维空间中点的移动轨迹看成是(x，t)空间中的线，使用PMR-四叉树(PMR-quadtree)作为索引结构[6]。具体方法是将时间维划分为若干个相等的时间间隔ΔT，对每个ΔT重建新的PMR-树索引，并允许在ΔT内插入、删除和更新操作。这种索引结构的缺点是在树中存在许多具体数据的复制(如一条线段通常会存储在几个结点中)，且需要大量的存储空间和频繁的数据更新。

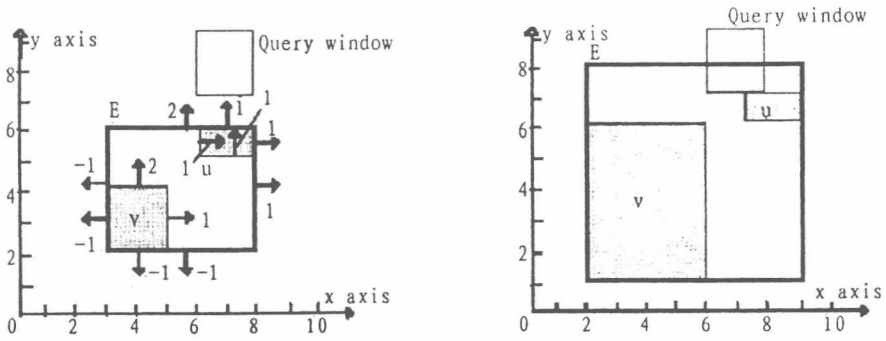
另一种建索引的方法是G. Kollios等提出的在二元空间中建索引的方法[5]，其主要思想源于几何算法中的二元转换：将原来空间(primal space)中的线(移动点的轨迹)转换为二元空间(dual space)中的点，用点存取方法(PAM)建索引，如图4所示。实验表明[5]，在二元空间中，基于k-d-树的存取方法优于R-树。



(a) 原来空间中移动点的轨迹 (y=vt+a)　　(b) 转换后的二元空间 (Hough-X 转换)

图4 二元空间转换

文[7]提出了直接在d-维空间对移动点建索引的结构-TPR-树(time-parameterized R-tree)：用速度矢量和MBRs两个参数表示移动对象，其中MBRs是以时间为参数的最小边界矩形，即它包围的是结点中当前时间的所有实体。在响应对未来任一时间戳的查询时，先计算这个未来时间戳的MBRs，在与查询窗口比较后返回相交区域的所有活动对象，如图5所示。



(a) 当前时间戳 t_0 的MBRs　　(b) 未来时间戳 t_1 的MBRs

图5 TPR-树中的MBRs

6 我们的工作

我们目前正在研制时空分析数据库管理系统STADBS，采用将时空子系统ROSTE (Robust Spatial-Temporal Extension)集成到可扩充数据库系统中的开发策略[19]。时空子系统ROSTE基于Realms[20]和主存技术，具有支持时空对象的表示、存储、管理、分析操作(尤其是拓扑分析)和索引机制等功能。

我们现已设计和实现了ROSTE中的时空对象存储管理子系统[21]及支持时空对象的空间分析操作算法[22]。在此基础上，正在设计和实现采用PPR-树作为ROSTE的索引结构，借鉴G. Kollios在文[13]中提出的MBRs优化分裂算法，研制基于Realms和主存的支持时空对象高效存取、查询的索引机制，以保障STADBS强大的时空分析功能的效率。

结束语 时空索引机制的研究工作现仍处于探索阶段，对于现已提出的各种时空索引机制，很难说哪一种明显优于其它的索引机制，这不仅需要大量实验的验证，还取决于不同应用系统的具体要求。另外，时空索引机制在有效支持各种复杂时空查询、支持分布式数据库的时空数据存取等方面还需要做进一步的研究工作。

参考文献

1 Manolopoulos Y, Theodoridis Y, Tsotras V J. Advanced Database Indexing. Kluwer Academic Publishers, 1999

2 Theodoridis Y, Sellis T, Papadopoulos A, Manolopoulos Y. Specifications for Efficient Indexing in Spatiotemporal Databases. STDBM, 1998

3 Agarwal P, Arge L, Erickson J. Indexing Moving Points. ACM PODS, 2000

4 Nascimento M, Silva J, Theodoridis Y. Evaluation of Access Structures for Discretely Moving Points. STDBM, 1999

此为试读，需要完整PDF请访问：www.ertongbook.com