基于惰性聚类分裂的动态R树实现方法*)

雷小锋 谢昆青 韩 亮 金星星

(北京大学智能科学系/视觉与听觉国家重点实验室 北京 100871)

摘 要 R*树是目前公认查询效果很好的R树变体,但是其构造代价较原始R树增加数倍,对于插入删除和更新频繁的空间数据效果不好。为此,本文提出一种基于惰性聚类分裂技术的R树动态实现方法(LR树)。惰性聚类分裂技术是在对象插入节点导致溢出时不立即进行分裂,而是尝试将其插入到邻近的未满节点中,直到邻近节点均已满时,再利用聚类技术进行节点分裂,在邻近节点和分裂节点之间重组入口项。LR树在确保查询性能的前提下,大大降低了构造代价,并且大幅提高了索引结构的空间利用率。最后的分析和实验证明了LR树的高效性。

关键词 R树,惰性聚类分裂

A Novel Implementation of Dynamic R-tree Based on Lazy Splitting and Clustering

LEI Xiao-Feng XIE Kun-Qing HAN Liang JIN Xing-Xing

(Department of Intelligence Science/National Laboratory on Machine Perception, Peking University, Beijing 100871)

Abstract R* tree is the most popular variation of R-tree, but not suitable for the situation of frequent insertion and deletion because its construction cost is many times than original R-tree. Therefore, a novel method of dynamic R-tree is proposed based on lazy splitting and clustering, called LR-tree. Where, lazy splitting does not split the node when a node is overflowed, but tries to insert it into the neighboring and not full node. Until all the neighbors are full, clustering technique is used to split nodes and reorganize the entries between the node and its neighboring nodes. LR-tree reduces the construction overheads with the guarantee of query performance, and improves the space utilization of index structure.

Keywords TR-tree, Lazy splitting, Clustering

1 前言

R树[1]索引结构是目前最为流行的多维索引结构,广泛应用于各种原型系统研究和商业应用中。R树是一棵深度平衡的动态树结构,其查找类似于B树。R树的查询性能很大程度上受到节点覆盖(coverage)和交叠(overlap)两个因素的影响。过大的覆盖会导致很大的空白区域(dead space),造成查询命中率的下降;过度的交叠则使查询操作需要搜索很多个路径,最坏时需要遍历所有的路径。

1.1 R树变体

原始R树通过二次分裂算法实现节点分裂。为了保证节点的空间利用率,在算法中会无视数据的分布,将一部分入口项简单地分配给某个分组,导致很糟糕的结果,并且很难补救。

R*树[2]通过引入强制重插入技术弥补了R树的上述缺陷,并且综合考虑节点覆盖、交叠以及目录矩形周长等参数优化插入和分裂算法。R*树改善了查询性能,提高了空间利用率,也大大增加了索引的构造代价。Hilbert R树[3]利用Hilbert曲线对空间数据进行一维线性排序以获得面积周长优化的节点。文[4]中提出Compact R树改进了分裂算法,使得空间利用率几乎达到100%,构建开销低于R树,检索性能与R树相仿。文[5]中提出cR树,将传统的两路分裂改造为基于聚类技术的多路分裂,获得较好的树结构,但存储利用率较低。

其他R树变体还很多,如利用最小边界凸多边形的CP树,适于主存索引的DR树,以及借鉴了位图索引思想的位图R树,这些变体在文[6]中有很好的综述。此外,很多研究关注R树批量操作技术,包括静态批量加载和动态批量插入。其中,静态批量加载在数据已知且相对静态时,对数据进行有效的预处理以提高加载速度,主要算法包括Packed R树、Hilbert Packed R树、STR压缩算法、TGS压缩算法、OMT压缩算法等;动态批量插入的目标是高效地将新数据集批量插入到现有树结构中,主要算法包括STLT算法、SCB算法等[7]。在很多批量操作技术中都利用聚类技术对数据进行集簇划分,以优化节点之间的重叠与覆盖。

1.2 本文工作

实际上,在R树的动态构造过程中,对象插入哪个节点以及节点如何分裂在当时的空间分布下通常是合理的,随着对象不断地插入,其空间分布发生变化,导致原来的决定变得不合理,这是动态R树构造必然会面临的问题。关键是在索引构造中提供修正决定的机制。本文提出一种基于惰性聚类分裂技术的动态R树实现方法,即LR树,更进一步放宽节点邻近性要求,而在后续处理中提供改善邻近性的机制。LR树在确保查询性能的前提下,大大降低了构造代价,并且大幅提高了R树的空间利用率。

本文第2节阐述了LR树的基本思路。第3节介绍K-

*)基金项目:国家自然科学基金项目(40235056)。雷小锋 博士后,主要研究方向为时空数据库与时空数据挖掘,时空推理和认知,机器学习;谢昆青 教授,博导,主要研究方向为智能信息处理模式识别;韩 亮 硕士研究生,主要方向为移动对象数据库;金星星 硕士,主要方向为数据库与数据挖掘。

Means 聚类算法及其距离测度。第 4 节讨论基于惰性聚类分裂的 LR 树构造方法。第 5 节提供了详细的实验评估。最后是结束语。

2 LR 树的基本思路

LR 树针对 R 树插入和分裂算法进行改进，其结构和查询算法完全与 R 树一致。由于采用了惰性聚类分裂技术处理插入和分裂算法，因此称为 LR 树(Lazy R-tree)。

回顾 R 树的构造，索引结构需要满足如下的性质：深度平衡、半满、对象邻近性。原始 R 树为了兼顾空间利用率，往往会有一部分对象的存储违反邻近性原则。而 R* 树在节点溢出时首先将一定比例的对象从节点中删除并重新插入到树中，即强制重插入技术，从而改善了树结构的邻近性，获得很好的查询性能，却付出重插入的沉重代价，如图 1 所示。

既然在动态构造中只能近似地保证邻近性，如果在后续处理中提供改善邻近性的机制，则可以放宽邻近性要求。这就是 LR 树的基本思路。具体采用的惰性聚类分裂技术包含惰性分裂和聚类分裂。其中，惰性分裂指的是在节点溢出时并不立即进行分裂，而是将待插入对象插入邻近的未满节点中。聚类分裂是当插入节点及其邻近节点均已满时采用聚类技术重组入口项，以改进邻近性。

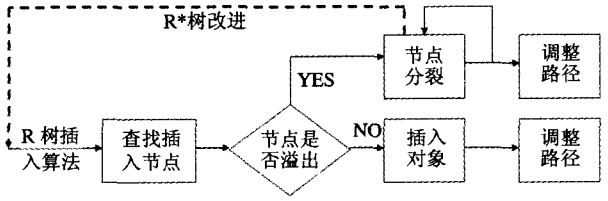

图 1 R 树插入算法及 R* 树改进

3 聚类算法

常见的聚类方法在文[8]中有系统的综述。本文采用通用的 K-Means 算法，其时间复杂度为 $O(k*n*d*t)$，k 为聚类数目，n 为对象数，d 为数据维数，t 为算法迭代次数。算法首先从 n 个对象中任意选择 k 个对象作为类簇中心，然后在每次迭代中将每个对象按照距离最近原则分配到对应的类簇，并更新类簇中心。迭代一直进行到类簇中心不再发生变化为止。利用 K-Means 算法对 R 树节点入口项聚类时，需要定义类簇中心和入口项之间的距离测度。对于 d 维空间中的 n 个点组成的类簇，将其中心定义为均值。

对于 d 维空间中的 n 个矩形 $R_1,\cdots,R_n$ 组成的类簇，每个矩形表示为 $R_i(P_{iL},P_{iH})$，其中 P_{iL} 是矩形左下角点，P_{iH} 是矩形右上角点，则其中心定义为：

$$\bar{P}\left(\frac{\sum_{i=1}^{n}(P_{iL}.x_1+P_{iH}.x_1)/2\times Area(R_i)}{\sum_{i=1}^{n}Area(R_i)},\cdots,\frac{\sum_{i=1}^{n}(P_{iL}.x_d+P_{iH}.x_d)/2\times Area(R_i)}{\sum_{i=1}^{n}Area(R_i)}\right)$$

d 维空间中两点的距离测度直接采用其欧氏距离。对于 d 维空间中的两个形状，其相似性测度定义为包含两个形状的最小外接矩形的对角线长度。同理，两个类簇之间的相似性测度定义为包含两个类簇的最小外接矩形的对角线长度。

4 LR 树构造

传统 R 树的对象插入过程为：(1)找到插入的叶节点。(2)若叶节点未满，则将新的入口项[mbr, oid]加入到与叶节点对应的页面中，调整父节点目录矩形。(3)若对象插入的叶节点已满，则出现分裂。创建一个新的节点，并且将 $M+1$ 个入口项在两个节点间分布。LR 树在寻找插入目标叶节点时，直接利用聚类算法中所定义的距离测度，选择最相似(或最近)的节点，具体算法 ChooseSubtree(略)。通过迭代或递归使用 ChooseSubtree 可以找到目标叶节点。LR 树最关键的内容是入口项插入算法和节点分裂算法。

4.1 惰性分裂技术

在入口项插入算法中采用惰性分裂技术，在节点溢出时不立即进行分裂，而是将待插入对象插入邻近的未满叶节点中。该技术延迟了节点的分裂，进而减少分裂次数并降低索引的构造代价，故称为惰性分裂。

算法 1(插入对象到叶节点—InsertEntry)

输入：叶节点 leaf，入口项 e，邻近的兄弟叶节点数目 k

输出：无

(1) 如果叶节点 leaf 未满，则将入口项 e 插入到叶节点对应的页面中，返回；

(2) 如果叶节点 leaf 已满，则找到其兄弟节点中与 leaf 节点最近的 k 个叶节点；

(3) 按距离顺序寻找 k 个叶节点中未满的节点 kleaf；

(4) 如果找到，将入口项 e 插入到 kleaf 节点对应的页面中，并返回；

(5) 如果未找到，则在 leaf 和 k 个邻近节点上执行聚类分裂算法进行入口项的重组。

4.2 聚类分裂技术

惰性分裂尽量避免分裂，导致 R 树结构邻近性的恶化，而聚类分裂则利用聚类技术处理节点分裂，从而提供改善邻近性的机制。当插入节点及其邻近节点均已满时，则必须分裂，采用聚类技术在这些节点中重新组织入口项，就是聚类分裂。

算法 2(叶节点分裂算法 C-Split)

输入：叶节点 leaf，数据对象 e，k 个邻近的兄弟叶节点

输出：将 leaf 和 k 个叶节点中的入口项重组得到的 $k+2$ 或更多的节点

(1) 利用 K-Means 算法对 $M*(k+1)+1$ 个入口项进行聚类，输出 $k+2$ 或更多类簇；

(2) 更新 leaf 和 k 个兄弟叶节点中的入口项；

(3) 根据最后一个类簇中的入口项新建一个叶节点，插入到 leaf 节点的父节点下。

算法的时间代价主要包括：$n*$ 选择子树 $+$ $n1*$ 插入入口项 $+$ $n2*$ 聚类分裂，其中 $n1+n2=n$，$n2=[n/(M*k)]$，聚类分裂代价主要是节点的磁盘 IO，聚类的时间复杂度为 $O(k*n*d*t)$，实际中迭代次数 t 很小。当 $k=0$ 时，LR 树就是 R 树，当 $k\geqslant[n/M]$ 时，则 LR 树相当于对静态数据的聚类重组。惰性聚类分裂技术可以保证：(1)聚类分裂可以获得入口项组织的优化，减少节点的覆盖和交叠，提高查询性能；(2)惰性分裂技术延迟节点分裂，减少分裂次数，提高索引的构造性能；(3)空间利用率大幅提高。实际中，邻近节点数目 k 的增加会更大地延迟并减少节点分裂，空间利用率也会提高，同时也带来聚类分裂代价的增加。

5 性能评估

为了评估 LR 树的性能，我们在 Java 环境下搭建统一的

(下转第 125 页)

件，生成进行服务调用的客户端代码片断。这里调用了遥感数据服务有3个应用系统：在应急响应系统中，针对应急事件，启动相应的预案，对警力、交通、120等资源予以调度，此系统通过WMS服务获得应急地点高分辨率航空影像，以辅助决策；数字绿化带系统用以支持城市绿化带建设过程中的规划、建设，包括绿化隔离地区的规划、土地、市政、人口、绿化等多种类型数据，访问WMS接口获得的遥感影像作为规划的底图；在空间数据建模系统中，遥感数据作为模型库的输入数据源之一，通过WCS接口获得多传感器、多平台、多时相、多光谱的遥感数据，在专业模型的支持下，可进行土地分类、洪水灾害评估等专业分析。

结束语 实践证明，在构建数字城市中，遥感数据具有天然的分布式与海量管理的特点，在一系列政策、标准、规范的保障下，构建统一的数据服务平台，进行数据服务的发布是可行的。对于一个多数据节点松耦合的数据服务系统，服务质量(QoS)是确保系统成功的一个重要因素，这就要求能在整个系统中实时地监控服务状态。应用程序可以根据服务质量的高低，采用一定策略选择服务质量最好的数据节点。另外，当前网格技术通过WSRF(Web Services Resource Framework，Web服务资源框架)规范，趋于与Web服务的融合，实现有状态的资源和无状态的服务的统一。如何在网格平台上实现数字城市遥感数据的共享，也是需要深入研究的问题。

参 考 文 献

1 科技部. 国家科技计划年度报告. 2005. http://www.most.gov.cn/ndbg/ndbg2004/2004.pdf
2 OGC. Web Map Service. http://portal.opengeospatial.org/files/?artifact_id=14416
3 OGC. Web Coverage Service(WCS), Version 1.0.0. http://portal.opengeospatial.org/files/?artifact_id=12582
4 OGC. Web Feature Service Implementation Specification. http://portal.opengeospatial.org/files/?artifact_id=8339
5 邬伦，刘宇，张晶，等. 地理信息系统——原理、方法和应用. 北京：科学出版社，2001
6 OGC. Web Registry Server. http://portal.opengeospatial.org/files/?artifact_id=1028
7 孙吉娟. 面向NSDI遥感影象数据库共享的元数据库和数据交换中心技术探讨. 测绘科学，2004，29(5)
8 NASA. Directory Interchange Format (DIF) Writer's Guide. http://gcmd.gsfc.nasa.gov/difguide/difman.html
9 NFGIS. 国家基础地理信息系统(NFGIS)元数据标准草案(初稿). http://nfgis.nsdi.gov.cn/nfgis/chinese/bz/mt0.htm
10 李军，彭凯，李琦，等. 基于数字北京的空间信息工程的建设与实践. 测绘科学，2005，30(1)

(上接第103页)

比较环境，并开发了R树、R*树索引算法。实验中使用两组数据集：(1)随机生成数据集：包含50000个矩形对象，坐标为[0.0，1.0]之间的随机浮点数。(2)真实数据集：包含16704个多边形对象的某市2001年的绿地图，Shape格式。

实验中，页面大小设为2kB，叶节点和中间节点容量为50，填充因子设为70%。构造索引时，对同一组数据以随机顺序插入树中，构造出100棵树结构计算平均构造时间和空间利用率。对于查询则一次运行1000次查询，取平均结果。实验在Intel 1.86G，512M内存的普通PC机上进行。

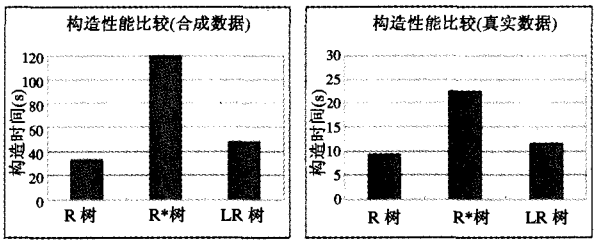

图2 不同索引结构的构造性能评估

索引结构的构造性能是衡量索引结构健壮性和可扩展性的重要内容，尤其在大量动态数据的管理中。从图2可以看出在构造代价上，LR树较原始R树高一些，较R*树则大大降低。从图3可以看出，LR树的查询性能与R*树在同一个等级上，在k最近邻居查询上表现甚至超出R*树。同样，在空间利用率上LR树的表现超出R*树，如图4所示。图4中右图表明，随着邻近节点数k的增加，空间利用率得到提高。

综上所述，LR树具有与R*树相当的查询性能，略高于原始R树的构造代价，以及很好的空间利用率。

结束语 本文提出了一种基于惰性聚类分裂技术的动态R树索引实现方法(LR树)。其关键思想是在节点溢出时并不立即进行分裂，而是将入口项插入到邻近的未满节点中，即惰性分裂。直到邻近节点均已满时才创建新的节点，并利用聚类技术重组入口项，即聚类分裂。LR树付出略高于R树的构造代价，可以获得与R*树相当的查询性能，并大大提高了空间利用率，适用于数据变化频繁的应用。

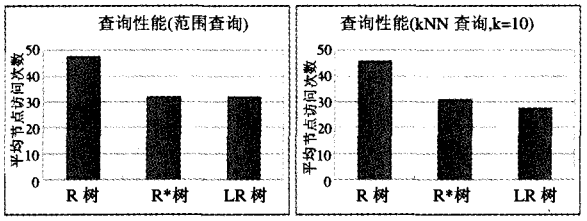

图3 不同索引结构的查询性能评估

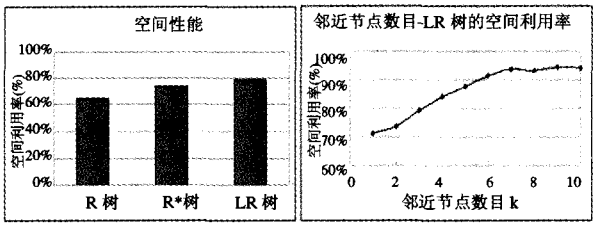

图4 不同索引结构的空间利用率评估

参 考 文 献

1 Guttman A. R-tree: A Dynamic Index Structure for Spatial Searching [C]. In: Proc. of 20th Int. Conf. on Very Large Data Bases, Morgan Kaufmann, 1984
2 Beckmann N, Kriegel H P, Schneider R, et al. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles [C]. In: Proc. of ACM Intl. Conf. on the Management of Data (SIGMOD), 1990. 322～331
3 Kamel I, Faloutsos C. Hilbert R-tree: An improved R-tree using fractals. In: Proc. of the 20th VLDB, 1994. 500～509
4 Huang P W, Lin P L, Lin H Y. Optimizing storage utilization in R-tree dynamic index structure for spatial database. Journal of Systems and Software, 2001, 55(2): 291～299
5 Brakatsoulas S, Pfoser D, Theodoridis Y. Revisiting R-Tree Construction Principles[C]. In: Proc. of 6th East European Conf. on ADBIS, Slovakia, 2002
6 Gaede V, Gunther O. Multidimensional access methods. ACM Computing Surveys, 1998, 30(2): 170～231
7 Bohm C, Berchtold S, Keim D A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia database. ACM Computing Surveys, 2001, 33(3): 322～373
8 Han J W, Kamber M. Data Mining: Concepts and Techniques [M]. U. S: Morgan Kaufmann Publishers, Inc, 2001