第37卷 第14期 计 算 机 工 程 2011年7月
Vol.37 No.14 Computer Engineering July 2011

·软件技术与数据库· 文章编号：1000—3428(2011)14—0056—03 文献标识码：A 中图分类号：TP311

基于聚类的空间数据多比例尺索引树

邹志文，费洪哲，李 根

(江苏大学计算机学院，江苏 镇江 212013)

摘 要：针对现有空间对象多尺度索引结构聚簇性不高的问题，在R树索引的基础上提出一种基于聚类的空间数据多比例尺索引结构。利用树的层次结构反映空间数据的多比例尺特性，用k-means算法对相同等级的空间对象进行聚类分组，减少空间区域覆盖和重叠。实验结果表明，该方法与基于四叉树的多比例尺索引相比，能有效提高空间数据多比例尺显示的性能。

关键词：多比例尺；R树；聚类算法；空间索引；地理信息系统

Multi-scale Index Tree of Spatial Data Based on Clustering

ZOU Zhi-wen, FEI Hong-zhe, LI Gen

(College of Computer, Jiangsu University, Zhenjiang 212013, China)

【Abstract】Due to the problem about clustering of multi-scale index structure for the existing space object is not high, this paper represents a cluster-based multi-scale spatial data indexing structure based on R-tree index, which uses hierarchical tree to reflect the multi-scale features of spatial data, and uses k-means algorithm to cluster groups on the same level of spatial objects, reducing coverage and overlapping regions of space. Experiments result shows that compared with other ways, the algorithm has a distinct superiority in the speed of multi-scale display of spatial data.

【Key words】multi-scale; R-tree; clustering algorithm; spatial index; Graphic Information System(GIS)

DOI: 10.3969/j.issn.1000-3428.2011.14.017

1 概述

随着地理信息系统(Graphic Information System, GIS)的迅速发展，多比例尺技术已经在GIS领域得到了广泛的应用。在GIS的地图显示过程中，为了减少数据量，提高数据存取效率，地图显示的数据量随地图比例尺的变化而变化。当以大比例尺显示地图时，内容比较详细，传输的数据量大；当比例尺变小时，内容变得更简略，一些次要要素被剔除。空间数据多比例尺索引结构决定了分析和查询地图的显示速度，目前较有代表性的是基于四叉树和R树的空间数据多比例尺索引结构[1-3]。

四叉树系列是基于空间划分的一类索引机制，存储空间需求量小，查询效率高，但在管理海量空间数据时效率太低[4]。文献[5]提出的Alternative Reactive树在R树的基础上增加了一个权值维，使二维对象通过三维最小包围框建立索引，但对海量空间数据查询时性能不理想。文献[6]对R树进行改进，实现了空间数据多尺度表达的索引结构，利用树的深度反映空间分辨率的变化很好地支持了空间数据的多比例尺特性，但无法有效减少空间对象之间的区域重叠。

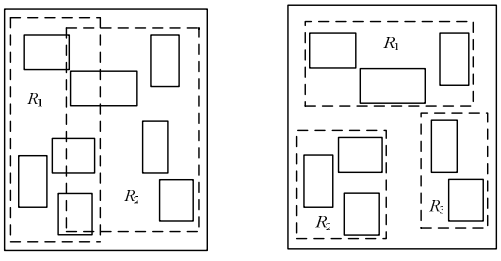
本文提出了一种基于聚类的空间数据多比例尺索引结构——CMIT(Cluster-based Multi-representation Index Tree)，减少了空间区域的重叠和覆盖，提高了空间数据的查询效率，同时又降低了构造索引树的时间复杂度。

2 空间对象聚类

在空间索引结构的建立过程中，需要将空间上相邻的对象归为一组结点，相邻关系的定义直接影响索引树结点间的区域重叠和区域覆盖面积。在R树家族中，Hilbert R-Tree索引通过Hilbert曲线定义空间对象邻近关系，是提高查询效率最有效的空间索引之一[7-8]。

Hilbert曲线的聚类特性较好，在构建索引树时，获得了比经典R树更小的区域覆盖和区域重叠面积，具有更高的查询效率。但是由于Hilbert曲线不能完全反映出空间实体的分布情况，因此在将高维空间映射到一维空间时，空间位置上相邻对象的Hilbert值并不一定相近。尤其在数据分布不均匀的情况下，存储空间率越高，结点之间的区域重叠面积越大，对索引结构的性能影响越大[9]。

在构建Hilbert R-Tree索引过程中，为了提高空间利用率，机械地将Hilbert值最小的M个对象索引记录(M为叶子结点的扇出系数)填充叶子结点，会使距离较远的对象也放入同一个叶子结点，明显增加了区域重叠和区域覆盖面积。如图1所示，叶子结点扇出系数为4，把所有的对象分成3组存储在3个叶子结点中明显比分成2组存储在2个叶子结点中更合理。



(a)Hilbert曲线分组 (b)聚类算法分组

图1 空间对象分组对比

基金项目：国家自然科学基金资助项目(60773049)；江苏省研究生科研创新计划基金资助项目(CX07B_125z)；江苏省中小企业技术创新计划基金资助项目(BC2008140)；镇江市社会发展计划基金资助项目(SH2008028)

作者简介：邹志文(1968－)，男，副教授，主研方向：空间数据库；费洪哲、李 根，硕士研究生

收稿日期：2010-12-17 **E-mail：**lgujs@126.com

3　CMIT 索引

CMIT 索引是基于 R 树索引的一种改进，通过引入聚类算法将空间对象分组，对同一组中的对象根据其 Hilbert 值进行排序，依次存储到叶子结点中。同时利用 R 树的层次结构将同等级的空间对象存储在相同的层次，树的深度表示分辨率维的变化，实现空间对象的多比例尺表示。在构建索引树时，适当提高叶子结点包含记录的最大数目，增加存储在连续磁盘空间的相邻对象，在地图显示过程中能提高对象检索性能，加快地图显示速度。

3.1　CMIT 索引结点构成

空间对象索引记录存储在叶子结点中，其表示是(*MBR*,*ObjID*,*H*)，其中，*MBR* 表示空间对象的最小外包矩形；*ObjID* 是数据库中空间对象的标识符；*H* 表示空间对象的显示等级。中间结点存储结点的索引记录，其表示是(*MBR*, *ptrChild*, *nodeType*)，其中，*MBR* 表示所指向的子树的对象的最小外包矩形；*ptrChild* 指向子结点的指针；*nodeType* 表示所指向的子结点的类型。

3.2　CMIT 索引的定义

对于 m 个空间对象集合 $\{obj_1,obj_2,\mathrm{L}\ ,obj_m\}$，假设将集合中的对象分成 n 个不同的显示等级，即 n 种分辨率子视图 $L_1,L_2,\mathrm{L}\ ,L_n$，不妨设 $L_1<L_2<\mathrm{L}\ <L_n$。CMIT 索引主要有以下特性：

(1)每个结点包含索引记录的个数介于 m 与 M 之间，除非是根结点。

(2)叶子结点只包含空间对象索引记录，中间结点只包含结点索引记录。

(3)对于索引树同一层次的叶子结点，左边结点内索引记录对应对象的 Hilbert 值小于右边结点索引记录对应对象的 Hilbert 值。

(4)允许叶子结点出现在任意层，且索引树的同一层只能存储相同等级的空间对象索引记录。

(5)树的深度反映空间实体对象不同的显示分辨率，从根结点层开始，树的深度越大，分辨率越高。

图 2 是包含 3 种分辨率视图的 CMIT 索引结构示意图。与 R 树相比，CMIT 索引的结构发生了变化，叶子结点可以出现在任意层次；树的每一层代表相应分辨率下的实体对象，深度越大，对应的实体对象分辨率越高。

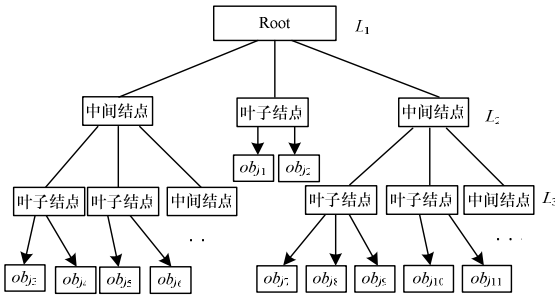


图 2　CMIT 索引结构

3.3　空间数据对象组织

聚类是将数据分成多个类，使属于同一个类的对象之间具有较高的相似度，而不同种类对象之间的差别较大。将聚类算法应用于空间数据的分类，就是把空间上相邻和查询上相关的对象存储在相同的硬盘页面，以实现对空间数据及相邻数据的快速索引。本文提出的 CMIT 索引使用 k-means 算法对空间对象分组，使空间相邻的对象分在同一组，以提高索引的性能。

空间对象聚类分组算法描述如下：

输入　空间对象集合 $S=\{R_1,R_2,\mathrm{L}\ ,R_n\}$

输出　集合 S 划分为数个子集 S_i，$S_i\mathrm{I}\ S_j=\varnothing,i\neq j$ 且 $S_1\mathrm{U}S_2\mathrm{UL}\ \mathrm{U}S_k=S$

Step1　计算最大分组数目 $k_{\max}$，即保证 $n/k_{\max}\geqslant m$，m 为结点必须包含的子对象最小数目。聚类分组数目 k 初始化为 2。

Step2　从集合 S 中选择 k 个对象分别赋给 k 个集合，作为集合的种子对象，将各个对象间的距离之和最大作为种子选择的原则，将各个对象的中心作为集合的中心。

Step3　用 k-means 算法作为剩余对象选择所属分组的原则，即将对象加入到距离集合中心最近的集合中，加入后更新所在集合的中心。重复 Step3，直到所有对象都分配到各个集合中为止。

Step4　以覆盖区域、重叠面积和外围周长作为评估指标计算评估函数，得到评估值 W_k。

Step5　k 值加 1，如果 $k\leqslant k_{\max}$，转 Step2。

Step6　从评估集合 $\{W_2,W_3,\mathrm{L}\ ,W_{k_{\max}}\}$ 中选出评估值最小的划分聚类个数，算法结束。

3.4　CMIT 索引树生成算法

假设已经对需要建立索引的空间对象进行等级划分，分成 n 个等级，分别用集合 $objSet_i$ 表示，$1\leqslant i\leqslant n$，i 指对象的等级。对 $objSet_i$ 中的对象用聚类算法进行分组，若分成 k 组，则用集合 $groupSet_j$ 表示，$1\leqslant j\leqslant k$，j 为组下标。叶子结点和中间结点能包含的最大索引记录为 M。空间对象的索引记录用集合 X 表示，中间结点和叶子结点的索引记录用集合 Y 表示，集合 X 和集合 Y 初始值为空。CMIT 索引生成算法如下：

CreateCmitIndex(F)

输入　空间对象集合 F

输出　CMIT 索引结构的根结点

变量　*IntlNodeSet* 是中间结点集合；*LeafNodeSet* 是叶子结点集合；X 为叶子结点的索引记录；Y 为中间结点的索引记录

```
Begin
for( i=n; i>0; i--){
Divided the spatial objects in objSet_i into k groups by the cluster algorithm;
for( j=1; j<=k; j++){
LeafNodeSet ≡ FormIndexEntry (groupSet_j,M, 0);
for(item ∈ LeafNodeSet)
X≡X+itemGenertEntryNode();
}
X≡X+Y;
Y≡NULL;
IntlNodeSet ≡ FormIndexEntry(X, M, 1);
X≡NULL;
for(item ∈ LeafNodeSet)
Y≡Y+itemGenertEntryNode();
}
while(|Y|>1) {
IntlNodeSet ≡ FormIndexEntry(Y, M, 1);
Y≡NULL;
for(item ∈ LeafNodeSet)
```

```
Y≡Y+itemGenertEntryNode();
}
The only node in IntlNodeSet is RootNode;
return RootNode;
End
```

CreateCmitIndex 函数中调用了 FormIndexEntry，其主要功能是产生结点的索引记录并按 Hilbert 索引值的大小组织新的结点。FormIndexEntry 函数的描述如下：

FormIndexEntry(*X*, *M*, *flag*)

输入 索引记录的集合 *X*，结点的扇出系数 *M*，标识该函数生成中间结点集合还是叶子结点的集合 *flag*

输出 结点集合 *NodeSet*

变量 *indexList* 为链表结构，用于存储按 Hilbert 值从小到大排序的结点的索引记录

```
Begin
for (x ∈ X){
Computer the Hilbert value for x;
indexList.Insert(x);
}
while (indexList !=null){
if (flag==0)
Create leaf node tempNode;
else
Create non-leaf node tempNode;
n=min(len(indexList), M);
tempNode≡indexList.GetItems(1,n);
Nodeset≡Nodeset+tempNode;
indexList.DeleteItems(1,n);
}
return NodeSet;
End
```

3.5 多比例尺显示

对于空间信息索引结构，查询操作是 CMIT 索引在实际应用中最重要的算法。若空间对象共分 n 个显示等级，则将地图在逻辑上分为 n 个不同的层次，记为 $L_1, L_2, \mathrm{L}, L_n$，令 $L_1 < L_2 < \mathrm{L} < L_n$。每一层次的地图空间包含相应显示等级的实体对象，定义当前地图显示比例 S 和层次 L 之间满足函数关系：$L=F(S)$。对于地图查询操作 $Query=(R,S)$，R 为当前屏幕显示地图数据，设本次显示操作装载的目标对象集为 T，$T=\{obj \mid MBR(obj) \cup R \neq \varnothing \wedge L(obj) \geqslant F(S)\}$，其中，$obj$ 为空间数据对象；$MBR(obj)$为 obj 的最小外包矩形；$L(obj)$为 obj 的等级。CMIT 索引查询算法的具体步骤如下：

输入 查询区域 R，显示比例尺 S

输出 目标对象集 T

Step1 求出 S 所对应的层次 L，令 *pnode* 指向根结点，索引记录集合 *indexRd* 为空。

Step2 如果 *pnode* 为叶子结点且对应层次小于等于 L，则将与 R 相交的实体对象加入集合 T。如果 *pnode* 为中间结点，则将结点中与 R 相交的索引记录加入集合 *indexRd*。

Step3 若集合 *indexRd* 为空，转 Step4。从 *indexRd* 中取出一条索引记录 x，令 *pnode* 指向 x 对应的结点，转 Step2。

Step4 返回目标对象集 T，显示相应的实体对象。

4 实验结果

通过将本文提出的 CMIT 索引与基于四叉树的多比例尺索引(SuperMap 索引)进行比较，验证 CMIT 索引在地图显示过程中高效的检索性能。实验数据使用镇江市京口区地图，基础比例尺为 1∶10 000。实验的微机环境为 Pentium 4 3.0 GHz CPU，512 MB 内存。

分别用 CMIT 索引和 SuperMap 索引组织地图数据，建立空间数据索引结构。平移、放大和缩小是地图显示的基本操作，在实验中，通过这 3 种基本操作衡量数据检索性能。表 1 为基于 2 种索引的数据各自执行 3 种操作的最大时间开销的对比。对于放大和缩小操作，后者的时间开销是前者的 1.5 倍~3.1 倍，特别在缩小操作时，CMIT 索引的性能优势更大。表 2 为基于 2 种索引的数据各自执行 3 种操作的平均时间开销的对比，平均时间开销是通过对每种操作执行 100 次所获得的时间开销平均值，对于每一种操作，CMIT 索引的访问效率都具有明显的优势。

表 1 2 种索引的最大时间开销比较 ms

操作	CMIT 索引	SuperMap 索引
平移	75	124
放大	168	213
缩小	136	387

表 2 2 种索引的平均时间开销比较 ms

操作	CMIT 索引	SuperMap 索引
平移	54	104
放大	147	196
缩小	126	340

5 结束语

由于传统 R 树索引与地图显示的多比例尺表示特性不相适应，对于构建大数据量的索引，区域重叠和覆盖情况更加严重。本文提出了一种新的多比例尺索引结构 CMIT，实验结果表明，该索引减少了空间区域的覆盖和重叠，提高了空间数据索引效率。

参考文献

[1] Guttman A. R-tree: A Dynamic Index Structure for Spatial Search[C]//Proceedings of ACM SIGMOD International Conf. on Management of Data. Boston, USA: ACM Press, 1984: 47-57.

[2] 郭 薇, 郭 菁, 胡志勇, 等. 空间数据库索引技术[M]. 上海: 上海交通大学出版社, 2006.

[3] 李 军, 景 宁, 孙茂印. 多比例尺下细节层次可视化的实现机制[J]. 软件学报, 2002, 13(10): 2037-2043.

[4] 陈俊华, 宋关福, 李绍俊. 基于 RDBMS 的空间数据库的设计与实现[C]//中国 GIS 年会论文集. 成都: [出版者不详], 2001.

[5] von Oosterom P. Reactive Data Structures for Geographic Information Systems[M]. New York, USA: Oxford University Press, 1993.

[6] 邓红艳, 武 芳. 一种用于空间数据多尺度表达的 R 树索引结构[J]. 计算机学报, 2009, 32(1): 177-184.

[7] Kamel I, Faloutos C. Hilbert R-tree: An Improved R-tree Using Fractals[C]//Proceedings of the 20th International Conference on VLDB. Santiago, Chile: [s. n], 1994: 500-509.

[8] Beckmann N, Schneider R. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles[C]//Proceedings of ACM SIGMOD'90. New York, USA: ACM Press, 1990: 322-331.

[9] 何小苑, 闵华清. 基于聚类的 Hilbert R 树空间索引算法[J]. 计算机工程, 2009, 35(9): 40-42.

编辑 张正兴