文章编号：0253-374X(2014)11-1718-07 DOI:10.11908/j.issn.0253-374x.2014.11.015

缓存模式下的轮廓查询优化方法

黄震华[1]， 张 波[2]， 张佳雯[1]， 向 阳[1]
(1. 同济大学 电子与信息工程学院，上海 201804；
2. 上海师范大学 信息与机电工程学院，上海 200234)

摘要：研究在缓存模式下，利用一组预存储的临时轮廓快照集来高效等价重构用户新提交的轮廓查询 Q，并提出一种新颖的缓存模式下轮廓查询处理算法（APSQCM）来完成该任务. APSQCM 算法由两阶段组成，第一阶段捕获 Q 与缓存中临时轮廓快照间的内在关联，进而获取能够用来等价重构 Q 的所有轮廓基；而在第二阶段中，APSQCM 算法使用轮廓基来快速产生 Q 的正确结果集. 实验结果表明，APSQCM 算法具有有效性和实用性.

关键词：轮廓查询；缓存模式；等价重构；查询优化
中图分类号：TP311.13 **文献标志码**：A

An Optimization Method for Skyline Query on Cache Model

HUANG Zhenhua[1]， *ZHANG Bo*[2]， *ZHANG Jiawen*[1]， *XIANG Yang*[1]
(1. College of Electronics and Information, Tongji University, Shanghai 201804, China; 2. College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China)

Abstract: This paper studies the equivalent reconstruction of the newly proposed skyline query using a group of existing skyline snapshots under the database cache model. We present a novel two-phase algorithm for processing skyline query on cache model (APSQCM) to support this reconstruction. In the first phase, the APSQCM algorithm captures the inherent relationship between the newly proposed skyline query and the existing ones, and obtains all the skyline radixes which can be used to equivalently reconstruct the newly proposed skyline query. And in the second phase, the APSQCM algorithm produces the correct result of the newly proposed skyline query from the skyline radixes. Furthermore, we present detailed theoretical analyses and extensive experiments that demonstrate our method is both efficient and effective.

Key words: skyline query; cache model; equivalent rewriting; query optimization

近年来，轮廓查询技术成为数据库学科的一个研究重点和热点[1]，这主要因为它在物联网、智能决策系统、信息推荐以及数据可视化等领域有着广泛的应用. 文献[2]首次将轮廓查询引入数据库领域，并提出两个正确性经过严格证明的查询算法：块嵌套循环算法(block nested loops，BNL)和划分占领算法 (divide and conquer，DC). 假定对象全集为 ψ，其中 BNL 算法通过 $O(|\psi|^2)$ 次对象间的比较来找出返回完整的轮廓对象集，而 DC 算法使用递归分区的方法来获取查询结果. 文献[3]从理论上给出 BNL 算法和 DC 算法的时间开销，并提出一种基于外部排序的轮廓槽算法 (lattice skyline algorithm，LSA)，LSA 的时间复杂度可降为 $O(|\psi|\log|\psi|+k|\psi|)$，其中 k 为维数. 文献[4]基于 ZBtree 索引树[5]来进一步优化 LSA 算法的处理效率，使其时间复杂度降为 $O(\sqrt[k]{|\psi|}\log|\psi|+k|\psi|)$.

在 2006 年的超大数据库(very large data bases：VLDB)国际会议上，文献[6]首次将轮廓查询与地理信息系统(geographic information system：GIS)技术相结合，提出遥感轮廓查询的概念. 基于空间几何领域的有效性质[7]，文献[6]给出一个时间复杂度为 $O(|\varphi|^2\gamma+\psi^{1/2})$ 的算法，其中 ψ 为空间数据对象集，γ 为搜索方案数，φ 为遥感点集中的凸包顶点数. 文献[8]考虑电子商务领域数据的模糊不完整特性，并给出一个时间复杂度为 $O(|\psi|^2)$ 的算法来处理这类数据的轮廓查询. 文献[9]针对谷歌文件系统 (google file system，GFS)的云计算环境，提出一

收稿日期：2013-07-21
基金项目：国家自然科学基金(61272268，61103069)；教育部新世纪优秀人才支持计划(NCET-12-0413)；国家“九七三”重点基础研究发展规划(2014CB340404)；霍英东教育基金会高等院校青年教师基金(142002)；同济大学中央高校基本科研业务费专项资金
第一作者：黄震华(1980—)，男，理学博士，副教授，主要研究方向为大数据分析、数据挖掘等，E-mail：huangzhenhua@tongji.edu.cn

个对数级时间复杂度的分布式轮廓计算算法(skyline computation for google file system, SCGFS). 文献[10]在面向服务架构(service-oriented architecture, SOA)下，基于轮廓查询技术来有效选择 Web 服务. 文献[11]针对无线传感器网络，给出一个能量有效的轮廓查询处理算法(skyline query on wireless sensor networks, SQWSN). 文献[12]基于增强型平衡树(enhanced balanced tree, B^+)索引树结构，设计一种面向多租户数据库的轮廓查询方法(multi-tenant database oriented skyline, MDOS). 文献[13]提出一种对不确定移动对象进行连续概率轮廓查询的有效算法(continuous skyline computation, CSC).

然而，现有这些研究工作最大的不足是，它们均以硬盘上的基础数据集为输入参数，即它们直接在海量的硬盘数据上执行轮廓查询操作. 因此，当硬盘数据量和维空间规模增大时，轮廓查询的时间开销将呈指数级增长. 随着大容量廉价内存的出现，使得利用现有数据库管理系统的缓存模式来缩短响应时间成为克服现有轮廓查询方法低效率的首选技术. 基于以上两个事实，本文在数据库缓存模式下，利用一组预存储的临时轮廓快照集合 $T=\{\xi_1,\cdots,\xi_2\}$ 来高效等价重构用户新提交的轮廓查询 Q，并提出一种新颖的缓存模式下轮廓查询处理算法(algorithm for processing skyline query on cache model: APSQCM)来实现该任务. APSQCM 算法由两阶段组成，第一阶段捕获 Q 与缓存中临时轮廓快照间的内在关联，进而获取能够用来等价重构 Q 的所有轮廓基；而在第二阶段中，算法使用轮廓基来高效产生 Q 的正确结果集. 实验结果表明，APSQCM 算法具有有效性和实用性.

1　轮廓查询等价重构问题

定义 1(支配关系). 假定 p 和 r 是两个具有 u 个维度的对象，记维空间为 $U=\{d_1,\cdots,d_u\}$. 如果它们满足下列两个条件，那么称 p 在 U 上支配 r：① $\forall i\in[1,u]$, $p[i]\leqslant r[i]$；② $\exists j\in[1,u]$, $p[j]<r[j]$.

为了简单起见，把 p 在 U 上支配 r，记为 $r\prec_U p$，在不发生混淆的情况下，省去字符‘U’，而写成 $r\prec_p$. 并且把 p 不在 U 上不支配 r，记为 $r\not\prec_U p$.

定义 2(轮廓查询). 假定 ψ 是具有 k 个维度对象全集，记维度集合为 $U=\{d_1,\cdots,d_u\}$. 那么 ψ 上的轮廓查询可表示为：$Q(\bar{s},\prec_U p)$：$-\varphi(\bar{z},\bar{x})$，其中 $\varphi(\bar{z},\bar{x})$ 为定义在 ψ 上的非负原子的合取，$\bar{x}$ 为自由变量构成的集合，$\bar{s}$ 为 $\bar{x}$ 中自由变量的置换，而 $\bar{z}$ 为由存在量词限制的约束变量构成的集合.

当将原子划分为约束原子(即至少包含一个约束变量的原子)和自由原子(即不包含任何约束变量的原子)时，轮廓查询 Q 可以改写为：$Q(\bar{s},\prec_U)$：$-\varphi_f(\bar{x})\wedge\varphi_b(\bar{z},\bar{x})$，其中 φ_b 为约束原子的合取，而 φ_f 为自由原子的合取. 不难看出，Q 的核心查询 Q_c 具有如下形式：$Q_c(\bar{s})$：$-\varphi_f(\bar{x})\wedge\varphi_b(\bar{z},\bar{x})$.

例 1　假定存在轮廓查询 $Q(p,g,\prec_U)$：$-t(p,c)\wedge f(p)\wedge s(s,c,g)$，则它的各成分可描述如下：$\bar{s}=[p,g]$，$\bar{x}=\{p,g\}$，$\bar{z}=\{c,s\}$；$\varphi_b(\bar{z},\bar{x})=t(p,c)\wedge s(s,c,g)$，$\varphi_f(\bar{x})=f(p)$.

定义 3(轮廓查询等价). 假定存在定义在非空对象集合 ψ 上的两个轮廓查询 Q 和 Q'，如果对于所有的数据库 D 均有 $Q^D=Q'^D$ 成立，那么称 Q 和 Q' 等价.

不失一般性，记 V 为若干轮廓查询的集合，且 V 中的所有轮廓查询均定义在 ψ 上. 同时，如果 V 中存在某一特定的轮廓查询 r，它的查询语句头部为算子 v，那么称 v 定义在集合 V 上，并将 v 称为快照算子. 为了简便，用 r^{D_V} 表示 V 中查询 r 在数据库 D 上所返回的结果集，而简称 V 为轮廓快照集合.

定义 4(最大等价重构集合). 假定存在轮廓查询 Q 和轮廓快照集合 V. 令 $A=\{Q'\mid Q'^{D_v}=Q^D\}$，即 A 为定义在 V 上的所有 Q 的等价重构构成的集合. 如果一个重构集合 x 满足如下 3 个性质，那么称 x 是 Q 的最大等价重构集合：① $x\subseteq A$；② 对于 x 中的每个重组 r，A 中均存在一个重组 r'，使得 $r=_V r'$ 成立；以及③不存在任何重组集合 J，使得 $J\subseteq x$ 且具有性质②.

2　APSQCM 算法

2.1　获取构造最大等价重构的轮廓基

定义 5(反转规则). 给定一个轮廓快照算子 $v\in V$：$v(\bar{s},\prec_U)$：$-\varphi_1(\bar{z}_1,\bar{x}_1)\wedge\cdots\wedge\varphi_l(\bar{z}_l,\bar{x}_l)$，可以获取与它相关的一组反转规则[14]，过程如下：使用不同的 Skolem 函数替换 v 规则体中的每个约束变量，并令 ρ 为每个约束变量到 Skolem 函数的映射，那么可以获取 l 个反转规则 $\rho(\varphi_i(\bar{z}_i,\bar{x}_i))$：$-v(\bar{s})$，其中 $i=1,2,\cdots,l$. 反转规则的左半部 $\rho(\varphi_i(\bar{z}_i,\bar{x}_i))$ 称为

规则头部,而它的右半部 $v(\bar{s})$ 称为规则体. 对于一个反转规则的头部 $\rho(\varphi_i(\bar{z}_i,\bar{x}_i))$ 来说,如果它至少包含一个 Skolem 函数,那么称它为约束规则头,否则称为自由规则头;此外,对于规则头部的每个变量 y,如果 y 只出现在 Skolem 函数中,那么称 y 为约束变量,否则称为自由变量.

例 2 假定 V 中含有 6 个轮廓快照算子:$v_1(u,t,k,\prec_U):-s(u,t)\wedge c(u,k)$,$v_2(s,u,z,\prec_U):-t(s,c)\wedge d(c,u)\wedge f(z)$,$v_3(s,\prec_U):-t(s,c)\wedge f(s)$,$v_4(u,t,\prec_U):-s(u,t)\wedge b(u,t)$,$v_5(u,c,\prec_U):-d(c,u)$,$v_6(s,u,\prec_U):-t(s,c)\wedge d(c,u)\wedge s(u,t)\wedge f(s)\wedge p(s,u)$. 那么根据定义 5,可以获取与它们相关的反转规则(令 $f_1(s,u,z)$、$f_2(s)$ 和 $f_3(s,u,c)$ 为 Skolem 函数).

$I_{11}:s(u,t):-v_1(u,t,k)$,$I_{12}:c(u,k):-v_1(u,t,k)$;

$I_{21}:t(s,f_1(s,u,z)):-v_2(s,u,z)$,$I_{22}:d(f_1(s,u,z),u):-v_2(s,u,z)$;$I_{23}:f(z):-v_2(s,u,z)$;

$I_{31}:t(s,f_2(s)):-v_3(s)$,$I_{32}:f(s):-v_3(s)$;

$I_{41}:s(u,t):-v_4(u,t)$,$I_{42}:b(u,t):-v_4(u,t)$;

$I_{51}:d(c,u):-v_5(c,u)$;

$I_{61}:t(s,c):-v_6(s,u,c)$,$I_{62}:d(c,u):-v_6(s,u,c)$,$I_{63}:s(u,f_3(s,u,c)):-v_6(s,u,c)$,$I_{64}:f(s):-v_6(s,u,c)$,$I_{65}:p(s,u):-v_6(s,u,c)$.

当获取所有反转规则之后,将它们划分为 n 个等价类 $R_1,\cdots,R_n$,使得落入每个等价类 R_i 中的反转规则具有相同的头部名 φ_i. 为了简便,称来自等价类 R_i 的规则头部为 φ_i 的镜像,而称 φ_i 为该头部的逆镜像.

定义 6(局胚) 给定一个轮廓查询 Q,它最大等价重构的一个局胚 p 为满足如下 5 个性质的反转规则头部组成的序列:① p 中的各规则头部来自不同的等价类;② 令 p_q 为与 pn 中各原子相对应的 Q 的子目标,那么存在一个映射 ϑ,它将 p_Q 中的每个自由变量映射为 p 中的自由变量;③ p_Q 中相同的约束变量在 p 中对应一致的 Skolem 函数;④ 对于 Q 的每个目标 g,如果它的一个约束变量在 p 中存在一个 Skolem 函数与之对应,那么 g 有一个 p 中镜像与之对应;以及⑤令 Q_B 为 p_q 中约束原子构成的集合,而 B 为 p 中约束规则头构成的集合,那么 $Q_B\supseteq B$.

如果局胚 p 包含 Q 中所有子目标的镜像,那么称它为一个最大等价重构轮廓基. 另外,如果局胚 p 是一个最小局胚,那么从 p 中移除任何一个原子之后它将不再成为一个局胚. 不失一般性,将空序列 $\varnothing$ 看成是一个特殊的局胚.

在 APSQCM 算法第一阶段中,使用过程 1 来返回能够构成查询最大等价重构的所有轮廓基. 过程 1 的处理步骤如算法 1 所示. 当过程 1 选择轮廓查询 Q 的一个子目标 φ_i 时,函数 1 就被调用来获取与之相关的合法的最小局胚. 不难看出,由于函数 1 只返回满足定义 6 中给出的 5 个性质的最小局胚,因此,它能够缩减大量不能够成为等价重组的候选查询,从而显著降低了时间开销. 函数 1 的处理步骤如算法 2 所示.

算法 1:过程 1

输入:包含 n 个原子的轮廓查询 Q,反转规则的 n 个等价类 $R_1,\cdots,R_n$;

输出:能够构成查询最大等价重构的所有轮廓基组成的集合 A;

方法:

1. $A\leftarrow\{\varnothing\}$;
2. While A 集合中还存在某个未成为轮廓基的局胚 p Do
3. 选择一个 Q 的子目标 φ_i,使得 φ_i 在 p 中不存在任何镜像;
4. If M_i 未被计算 Then
5. $M_i\leftarrow$函数 1(φ_i,$\{R_1,\cdots,R_n\}$); /* 函数 1 见算法 2 */
6. If $M_i=\varnothing$ Then
7. $A\leftarrow A-\{p\}$;
8. If $M_i\neq\varnothing$ Then
9. For M_i 中的每个最小局胚 m Do
10. $A\leftarrow A\cup(\{p\cup m\}$;
11. $A\leftarrow A-\{p\}$;
12. 返回 A;

算法 2:函数 1

输入:轮廓查询 Q 及其子目标 φ_i,反转规则的 n 个等价类 $R_1,\cdots,R_n$;

输出:包含 φ_i 镜像的所有最小局胚构成的集 M_i;

方法:

1. $M_i\leftarrow\varnothing$;
2. For R_i 中的每个反转规则 I Do
3. If I 的规则头部 φ' 满足定义 6 中的性质②和

⑤ Then

4. $M_i \leftarrow M_i \cup \{I$ 的头部 $\varphi'\}$；

5. While M_i 集合中还存在某个未成为最小局胚的规则头序列 h Do

6. If Q 存在一个约束变量 v 与 h 中的 Skolem 函数 $f(Z)$ 对应，且 q 存在一个子目标 φ_t，它在 h 中没有镜像 Then / * 即定义 6 中的性质③ * /

7. For R_t 中的每个规则头 x Do

8. If 序列 $h \cup \{x\}$ 满足定义 6 中的性质②、④和⑤ Then

9. $M_i \leftarrow M_i \cup \{h \cup \{x\}\}$；

10. $M_i \leftarrow M_i - \{h\}$；

11. 返回 M_i；

例 3　对于例 1 中的轮廓查询 Q 和轮廓快照集合 V，以及例 2 中的反转规则，能够得到 7 个等价类，见表 1. 在该例中，过程 1 总共只调用 9 次函数 1，并且获取 8 个胚源，见表 2. 注意，为了简便而不失一般性，将 Skolem 函数 $f_1(s, u, z)$、$f_2(s)$ 和 $f_3(s, u, c)$ 分别简写为 f_1、f_2 和 f_3.

表 1　七个等价类

Tab.1　Seven equivalence classes

等价类	规则头部名	规则
R_1	s	I_{11}, I_{41}, I_{63}
R_2	c	I_{12}
R_3	t	I_{21}, I_{31}, I_{61}
R_4	d	I_{22}, I_{51}, I_{62}
R_5	f	I_{23}, I_{32}, I_{64}
R_6	b	I_{42}
R_7	p	I_{65}

表 2　过程 1 获取的胚源

Tab.2　The embryoes obtained by Procedure 1

调用序号	函数 1 返回的最小局胚集合	是否胚源
1	$\{t(s, f_1), d(f_1, u)\}; \{t(s, c)\}$	否
2	$\{t(s, f_1), s(u, t), d(f_1, u)\}; \{t(s, f_1), s(u, f_3), d(f_1, u)\}$	否
3	$\{t(s, c), s(u, t)\}; \{t(s, c), s(u, f_3)\}$	否
4	$\{t(s, f_1), s(u, t), d(f_1, u), f(z)\}; \{t(s, f_1), s(u, t), d(f_1, u), f(s)\}$	是
5	$\{t(s, f_1), s(u, f_3), d(f_1, u), f(z)\}; \{t(s, f_1), s(u, f_3), d(f_1, u), f(s)\}$	是
6	$\{t(s, c), s(u, t), d(c, u)\}$	否
7	$\{t(s, c), s(u, f_3), d(c, u)\}$	否
8	$\{t(s, c), s(u, t), d(c, u), f(z)\}; \{t(s, c), s(u, t), d(c, u), f(s)\}$	是
9	$\{t(s, c), s(u, f_3), d(c, u), f(z)\}; \{t(s, c), s(u, f_3), d(c, u), f(s)\}$	是

2.2　基于轮廓基构造最大等价重构

定义 7(同胚关系). 给定一个轮廓基 U，如果 U 中的两个原子 φ 和 φ' 满足如下任何一个条件，那么称 φ 和 φ' 满足同胚关系，并记为 $\varphi \otimes \varphi'$：① φ 和 φ' 包含相同的 Skolem 函数 $f(Z)$，并且它们对应查询 Q 中相同的约束变量；以及②存在 t 个原子 $\varphi_1, \varphi_2, \cdots, \varphi_t$，使得 $\varphi \otimes \varphi_1, \varphi_1 \otimes \varphi_2, \cdots, \varphi_{t-1} \otimes \varphi_t, \varphi_t \otimes \varphi'$ 成立.

不难看出，同胚关系是个等价关系，并且它将 U 划分为 m 个不相交的组 $\{G_1, G_2, \cdots, G_m\}$. 对于每个分组 $G_i (1 \leqslant i \leqslant m)$，记 I_i 为规则头与 G_i 中某原子相同的反转规则构成的集合，并记 C_i 为 I_i 中所有规则体构成的集合. 根据反转规则的定义，可知 C_i 只包含快照算子. 另外，添加一个空快照 NULL 到 C_i 中，且 NULL $\wedge v \equiv v$.

例 4　对于表 2 中的胚源 $e = \{t(s, f_1), s(u, t), d(f_1, u), f(z)\}$，根据同胚关系的定义将它划分为 3 个组 $G_1 = \{t(s, f_1), d(f_1, u)\}$、$G_2 = \{s(u, t)\}$ 以及 $G_3 = \{f(z)\}$，从而有：$I_1 = \{t(s, f_1) :- v_2(s, u, z), d(f_1, u) :- v_2(s, u, z)\}$、$I_2 = \{s(u, t) :- v_1(u, t, k), s(u, t) :- v_4(u, t)\}$ 以及 $I_3 = \{f(z) :- v_2(s, u, z)\}$，因此，可得：$C_1 = \{$NULL, $v_2(s, u, z)\}$、$C_2 = \{$NULL, $v_1(u, t, k), v_4(u, t)\}$ 以及 $C_3 = \{$NULL, $v_2(s, u, z)\}$.

定义 8(可构造轮廓快照组). 假定将轮廓基 W 划分为 m 个不相交的组 $\{G_1, G_2, \cdots, G_m\}$，并且记 $H = \{v_1, v_2, \cdots, v_m\}$ 为一组轮廓快照算子，其中 $v_i \in C_i$. 如果 H 满足如下 5 个性质，那么称 H 为可构造轮廓快照组：

① $T_{1 \leqslant i \leqslant m}(S_i^b \cup S_i^f) = a(Q)$；

② $\forall i \neq t, S_i^b \cap S_t^b = \varnothing$；

③ $\forall i \neq t, S_i^f \cap S_t^f = \varnothing$；

④ $\forall v_i \in H, \sigma' \in N_i \Rightarrow \sigma' \in S_i^f$；

⑤ $\forall v_i \in H, \eta_i(B_i) \cap E_i = \varnothing$，其中集合 $B_i = \{x \in z_b(S_i^b) \mid \eta_i(x) \in z_f(\xi_i(v_i))\}$，集合 $E_i = \{y \in z(g) \mid g \in z_f(\xi_i(v_i)) \wedge \neg \exists x \in z_f(Q), y = \eta_i(x)\}$.

基于定义 8，可以有效构造轮廓查询 Q 的最大等价重构集合的过程，如例 5 所示.

例 5　对于例 1 中的查询 Q，例 3 中的每个胚源相应的轮廓快照集合 $\{M_1, , \cdots, M_m\}$ 见表 3. 对于第 1 个胚源，只有 $\{v_1, v_2\}$ 是可构造轮廓快照组，而轮

廓快照算子 v_4 不能包含在可构造轮廓快照组中,这主要是因为它不满足定义 8 中的性质⑤,即 $\eta_4(B_4)\bigcap E_4=\{t\}$. 对于第 2 和第 5 个胚源,$\{v_1, v_2\}$和$\{v_6\}$为可构造轮廓快照组. 然而,$\{v_1, v_6\}$不能成为可构造轮廓快照组是因为它不满足性质③,即 $S_1^{\mathrm{f}}\bigcap S_6^{\mathrm{f}}\neq\varnothing$;$\{v_2, v_3\}$不能成为可构造轮廓快照组是因为它不满足性质②,即 $S_2^{\mathrm{b}}\bigcap S_3^{\mathrm{b}}\neq\varnothing$. 对于第 3、4、7 和 8 个胚源,只有$\{v_6\}$是可构造轮廓快照组. 因此,对所有 8 个胚源来说,总共可以获取 2 个可构造轮廓快照组,即$\{v_1, v_2\}$和$\{v_6\}$. 对于$\{v_1, v_2\}$,它产生查询 $p_1(s, u, k, \prec_{w_1\cup w_2}): -v_1(u, t, k, \prec_{w_1})\wedge v_2(s, u, s, \prec_{w_2})$,从而相应的包含重构为 $r_1(s, u, \prec_w): -p_1(s, u, k, \prec_w)$;而对于$\{v_6\}$,它产生查询 $p_2(s, u, \prec_{w_6}): -v_6(s, u, c, \prec_{w_6})$,从而相应的包含重构为 $r_2(s, u, \prec_w): -p_2(s, u, \prec_w)$.

表 3 胚源及其相应的轮廓快照集合

Tab.3 Embryoes and their skyline snapshots sets

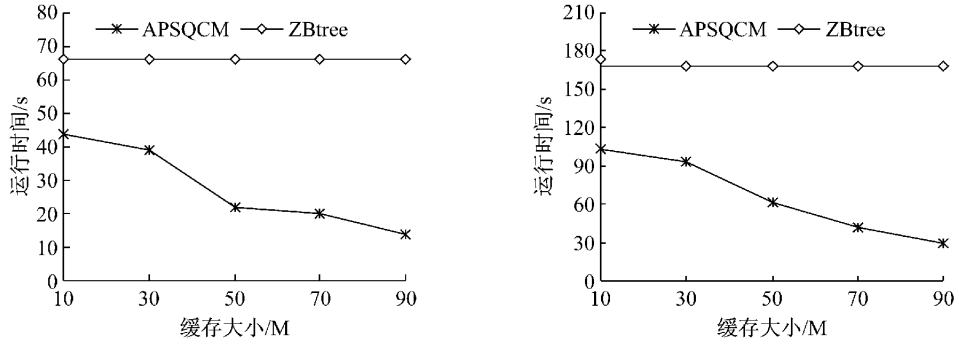
胚源	相应的轮廓快照集合
1	$M_1=\{\mathrm{NULL}, v_2\}$, $M_2=\{\mathrm{NULL}, v_1, v_4\}$, $M_3=\{\mathrm{NULL}, v_2\}$
2	$M_1=\{\mathrm{NULL}, v_2\}$, $M_2=\{\mathrm{NULL}, v_1, v_4\}$, $M_3=\{\mathrm{NULL}, v_3, v_6\}$
3	$M_1=\{\mathrm{NULL}, v_2\}$, $M_2=\{\mathrm{NULL}, v_6\}$, $M_3=\{\mathrm{NULL}, v_2\}$
4	$M_1=\{\mathrm{NULL}, v_2\}$, $M_2=\{\mathrm{NULL}, v_6\}$, $M_3=\{\mathrm{NULL}, v_3, v_6\}$
5	$M_1=\{\mathrm{NULL}, v_6\}$, $M_2=\{\mathrm{NULL}, v_1, v_4\}$, $M_3=\{\mathrm{NULL}, v_5, v_6\}$, $M_4=\{\mathrm{NULL}, v_2\}$
6	$M_1=\{\mathrm{NULL}, v_6\}$, $M_2=\{\mathrm{NULL}, v_1, v_4\}$, $M_3=\{\mathrm{NULL}, v_5, v_6\}$, $M_4=\{\mathrm{NULL}, v_3, v_6\}$
7	$M_1=\{\mathrm{NULL}, v_6\}$, $M_2=\{\mathrm{NULL}, v_6\}$, $M_3=\{\mathrm{NULL}, v_5, v_6\}$, $M_4=\{\mathrm{NULL}, v_2\}$
8	$M_1=\{\mathrm{NULL}, v_6\}$, $M_2=\{\mathrm{NULL}, v_6\}$, $M_3=\{\mathrm{NULL}, v_5, v_6\}$, $M_4=\{\mathrm{NULL}, v_3, v_6\}$

3 实验评估

本节通过具体实验来评估 APSQCM 算法的有效性. 与 APSQCM 算法进行性能比较的是目前轮廓查询效率最高的区域平衡树算法(zone balanced tree, Zbtree)[4]. 实验环境为: i5-3 210M 2.5G CPU,2G 内存,500G 硬盘,Windows 7 操作系统. 所有实验代码在 Java 编译器中运行通过.

实验数据由文献[2]给出的数据生成器来产生. 文献[2]中的数据生成器产生的对象有两种类型:独立分布数据集和反相关分布数据集. 同时,为了让实验更具可靠性,产生 100 个轮廓查询,然后对运行时间取平均值. 实验评估分三组进行:(1)在第一组实验中,将基础数据量和对象维数分别固定为 500M 和 8 维,而缓存大小从 10M 到 90M 间变化;(2)在第二组实验中,将基础数据量和缓存大小分别固定为 1G 和 100M,而对象维数从 4 维到 12 维间变化;(3)在第三组实验中,将对象维数和缓存大小分别固定为 8 维和 80M,而基础数据量从 200M 到 1G 间变化.

第一组实验的结果如图 1 所示,其中图 1a 和 1b 分别显示随缓存大小变化时,APSQCM 算法和 ZBtree 算法在独立分布数据集和反相关分布数据集上的轮廓查询运行时间.



a 独立分布数据集 b 反相关分布数据集

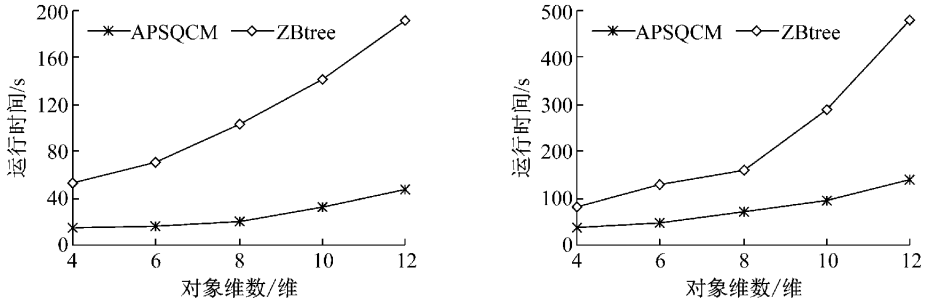
图 1 运行时间随缓存大小变化

Fig.1 Runtime vs. size of cache

从图 1 可以看出,APSQCM 算法的运行时间在各种实验设置下均显著优于 ZBtree 算法. 例如,在

反相关分布数据集上(图 1b),当缓存大小为 90M 时,ZBtree 算法的运行时间为 167.3s,而本文的 APSQCM 算法仅为 29.1s,即 APSQCM 算法的时间开销仅为 ZBtree 算法的 17.4%.另一方面,笔者发现,ZBtree 算法对于不同的缓存大小设置,它的运行时间均是相同的,这是因为 ZBtree 算法以硬盘上的基础数据集为输入参数,与缓存的设置无关.

第二组实验的结果如图 2 所示,其中图 2a 和 2b 分别显示随对象维数变化时,APSQCM 算法和 ZBtree 算法在独立分布数据集和反相关分布数据集上的轮廓查询运行时间.



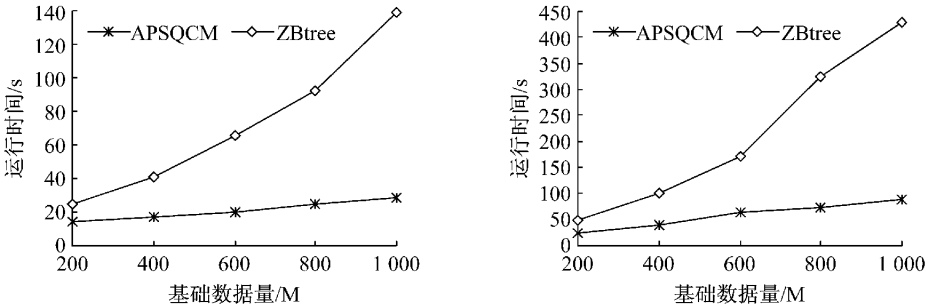
a 独立分布数据集　　b 反相关分布数据集

图 2　运行时间随对象维数变化

Fig.2　Runtime vs. dimensionality

从图 2 可以看出,APSQCM 算法的运行时间在各种实验设置下均显著优于 ZBtree 算法.例如,在独立分布数据集上(图 2a),当对象维数为 12 维时,ZBtree 算法的运行时间为 191.5 s,而本文的 APSQCM 算法仅为 47.8 s,即 APSQCM 算法的时间开销仅为 ZBtree 算法的 25.0%.另一方面,笔者还可以观察到,随着对象维数的增加,ZBtree 算法的运行时间呈指数级增长,而 APSQCM 算法的运行时间成线性增长,例如,在反相关分布数据集上(图 2b),当对象维数为 4 维时,ZBtree 算法和 APSQCM 算法的运行时间分别为 82.5 s 和 37.1 s,此时,APSQCM 算法的时间开销为 ZBtree 算法的 45.0%;而当对象维数为 12 维时,ZBtree 算法和 APSQCM 算法的运行时间分别为 480.3 s 和 138.5 s,此时,APSQCM 算法的时间开销仅为 ZBtree 算法的 28.8%.

第三组实验的结果如图 3 所示,其中图 3a 和 3b 分别显示随基础数据量变化时,APSQCM 算法和 ZBtree 算法在独立分布数据集和反相关分布数据集上的轮廓查询运行时间.



a 独立分布数据集　　b 反相关分布数据集

图 3　运行时间随基础数据量变化

Fig.3　Runtime vs. volume of basic data

与第一组和第二组实验一样,从图 6 可以看出,APSQCM 算法的运行时间在各种实验设置下均显著优于 ZBtree 算法.例如,在独立分布数据集上(图 3a),当基础数据量为 1G 时,ZBtree 算法的运行时间为 139.2 s,而本文的 APSQCM 算法仅为 28.7 s,即 APSQCM 算法的时间开销仅为 ZBtree 算法的 20.6%.另一方面,笔者发现,不管是独立分布数据集还是反相关分布数据集,随着基础数据量的增加,ZBtree 算法的运行时间呈指数级增长,而本文 APSQCM 算法的运行时间基本趋于平稳.这主要是因为 ZBtree 算法对基础数据量是指数敏感的,而 APSQCM 算法的大部分输入数据是从缓存的预存

储快照集合中获取的.例如,在反相关分布数据集上(图3b),当基础数据量为200M和1G时,ZBtree算法的运行时间分别为48.6s和427.5s,即基础数据量为1G时的时间开销是200M时的8.8倍;而当基础数据量为200M和1G时,APSQCM算法的运行时间分别为25.2 s和89.1 s,即基础数据量为1G时的时间开销仅为200M时的3.5倍.

4 结 论

轮廓查询技术是目前数据库学科的一个研究重点和热点.据了解,现有的研究工作直接在海量的硬盘数据上执行轮廓查询操作.因此,当硬盘数据量和维空间规模增大时,轮廓查询的时间开销将呈指数级增长.

本文在总结现有工作不足的基础上,借助当前主流数据库管理系统(如ORACLE、SQL SERVER和DB2等)的缓存模式来提高轮廓查询的运行效率,并提出正确性经过严格证明的两阶段优化算法APSQCM.APSQCM算法首先捕获新提交轮廓查询与缓存中轮廓快照间的内在关联,进而获取能够用来等价重构查询的所有轮廓基;然后使用轮廓基来高效产生Q的正确结果集.此外,本文在独立分别数据集和反相关分布数据集上进行三组实验,实验结果表明,APSQCM算法具有有效性和实用性.

参考文献:

[1] Ma L, Zhu M. Skyline query for location-based recommendation in mobile application [C]//Proceedings of International Conference on Web-age Information Management. Beidaihe: Springer, 2013: 236-247.

[2] Borzsonyi S, Kossmann D, Stocker K. The skyline Operator [C]//Proceedings of International Conference on Data Engineering. Heidelberg: IEEE Press, 2001: 421-430.

[3] Godfrey P, Shipley R, Gryz J. Maximal vector computation in large data sets [C]//Proceedings of International Conference on Very Large Data Bases. Trondheim: VLDB Endowment, 2005: 229-240.

[4] Lee K, Lee W, Zheng B, *et al*. Z-SKY: an efficient skyline query processing framework based on Z-order [J]. The VLDB Journal, 2010, 19(3): 333.

[5] Myllymaki J, Kaufman J. High-performance spatial indexing for location-based services [C]//Proceedings of International Conference on World Wide Web. Budapest: ACM, 2003: 112-117.

[6] Sharifzadeh M, Shahabi C. The spatial skyline queries [C]//Proceedings of International Conference on Very Large Data Bases. Seoul: VLDB Endowment, 2006: 751-762.

[7] Achlioptas D, Oghlan A, Tersenghi F. On the solution-space geometry of random constraint satisfaction problems [J]. Random Structures & Algorithms, 2011, 38(3): 251.

[8] Khalefa M, Mokbel M, Levandoski J. Skyline query processing for incomplete data [C]]//Proceedings of International Conference on Data Engineering. Cancun: IEEE Press, 2013: 556-565.

[9] Huang Z, Xiang Y. Improve the usefulness of skyline analysis in cloud computing environments [C]//Proceedings of International Symposium on Computational Intelligence and Design. Changsha: IEEE Press, 2009: 325-328.

[10] Alrifai M, Skoutas D, Risse T. Selecting skyline services for QoS-based web service composition [C]//Proceedings of International Conference on World Wide Web. Raleigh: ACM, 2003: 11-20.

[11] Wang G, Xin J, Chen L, *et al*. Energy-efficient reverse skyline query processing over wireless sensor networks [J]. IEEE Transaction on Knowledge and Data Engineering, 2011, 1(8): 1.

[12] Sun S, Huang Z, Li P. Skyline computation over multi-tenant database system [J]. Journal of Frontiers of Computer Science and Technology, 2011, 5(4): 289.

[13] Fu S, Dong Y, Tang Y, *et al*. Continuous probabilistic skyline queries for moving objects with uncertainty based on event [J]. Journal of Statistical Planning and Inference, 2011, 141(2): 924.

[14] Chena V, Chinchillib V, Richardsc D. Robustness and monotonicity properties of generalized correlation coefficients [J]. *Acta Automatica Sinica*, 2011, 37(7): 836.