SKYLINE查询解析

黄震华[1,2],向 阳[1],林 琛[3],孙圣力[4]
(1. 同济大学计算机科学与工程系,上海 200092;2. 同济大学嵌入式系统与服务计算教育部重点实验室,上海 200092;
3. 复旦大学计算机与信息技术系,上海 200433;4. 北京大学软件与微电子学院,北京 102600)

摘 要: 现有的研究工作只考虑如何对单个底层关系表进行skyline计算,即它们假定用户所提交的skyline查询不涉及任何传统的关系操作,并且所有skyline维度均落入同一个关系表中.显然,在实际应用中,由于这种假设的不成立,使得在多数情况下用户查询的效率极其低下.基于此,将skyline计算作为一个特殊的关系操作符,研究它与传统关系操作符间执行顺序变换的等价规则.从而,利用这些等价变换规则,通过改变skyline操作符与传统关系操作符之间的执行顺序来有效提高查询的效率.同时,给出充分的理论证明来论证所给等价变换规则的正确性,并通过实验验证其有效性.

关键词: 数据库; skyline查询; 关系操作; 等价变换规则; 查询优化
中图分类号: TP311.13 **文献标识码:** A **文章编号:** 0372-2112 (2009) 08-1639-07

Parsing Skyline Queries

HUANG Zhen-hua[1,2], XIANG Yang[1], LIN Chen[3], SUN Sheng-li[4]
(1. *Department of Computer and Technology, Tongji University, Shanghai 200092, China*;
2. *The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China*;
3. *Department of Computer and Information Technology, Fudan University, Shanghai 200433, China*;
4. *School of Software and Microelectronics, Peking University, Beijing 102600, China*)

Abstract: The existing works only consider how to efficiently process skyline computation for a single table. That is, they assume the issued skyline queries do not involve any traditional relational operator. Clearly, in most real applications, the query efficiency is extremely low because of this unreasonable assumption. Motivated by these facts, we regard skyline computation as a special relational operator and study the equivalence transformation rules of implementation order of it and traditional relational operators. Then based on these equivalence transformation rules, we can efficiently improve the query performance. Moreover, we present sufficient theoretical proofs to demonstrate the correctness of the proposed equivalence transformation rules. The extensive experiments also show that the after-transforming solutions markedly outperform the before-transforming counterparts.

Key words: database; skyline query; relational operator; equivalence transformation rules; query optimization

1 引言

Skyline查询处理是近年来数据库技术领域的一个研究重点和热点.这主要是因为skyline查询处理在许多领域有着广泛的应用,如:多标准决策支持系统[1],城市导航系统[2],数据挖掘和可视化[3]以及用户偏好查询[4]等.

现有的相关工作[1~15]均只考虑如何在单个关系表上进行有效的skyline计算(注:为了容易区分,本文将skyline计算与传统关系操作结合的查询称为skyline查询.由于现有的文献中所提的skyline查询不涉及任何传统关系操作,因此,它相当于本文所提的skyline计算的概念),即它们假定用户所提交的查询不涉及任何传统的关系操作(如选择和卡氏积等),并且所有skyline维度均落入同一个关系表中.然而在实际应用中,skyline查询包含关系操作的情况经常会发生,例如,用户只需要某个特定范围内的查询结果集,或者某些skyline查询需要访问多个底层关系表才能够返回最终的结果.因此,当这些情况发生时,现有的研究工作不能够有效地解析用户提交的查询,从而无法在逻辑层面上,通过适当地调整skyline计算与传统关系操作之间的执行顺序来有效地返回skyline查询的结果集.

例1(skyline查询涉及关系积操作).系统存在两个关系表 T_1 和 T_2,如图1所示.

收稿日期:2008-09-03;修回日期:2009-03-19
基金项目:国家自然科学基金(No.70771077);国家863高技术研究发展计划(No.2008AA04Z106);上海市委科研计划项目(No.08DZ112300);同济大学青年优季人才基金(No.0800219093)

关系表 T_1				关系表 T_1			
元组标识	A	B	C	元组标识	D	E	C
t_{11}	9	4	4	t_{21}	8	3	4
t_{12}	3	2	4	t_{22}	9	4	4
t_{13}	10	1	4	t_{23}	12	2	6
t_{14}	12	3	6	t_{24}	17	6	6
t_{15}	18	2	6				
t_{15}	7	2	6				

图 1 关系表 T_1 和 T_2

本文约定元组在各维度上的取值越小，那么元组在该维度就越优. 假定此时用户发出 skyline 查询 $SKYQ(T_1 \times T_2, ABDE)$，即用户需要考察维度 $ABDE$ 上的 skyline 对象集合. 不难看出，该查询涉及底层两个关系表 T_1 和 T_2. 由于现有的研究工作只考虑单表的情况，而且据我们所知，目前还没有 skyline 计算与传统关系操作之间执行顺序变换及其正确性证明的相关文献，因此为了能够正确返回该查询的结果集，现有的方法均要通过两个阶段来进行：①首先对关系表 T_1 和 T_2 进行积操作，产生 24 个元组；②然后，对这 24 个元组进行维度 $ABDE$ 上的 skyline 计算，并将结果集返回给用户. 显然，当关系表 T_1 和 T_2 的元组个数以及它们所包含的维度个数较多时，现有的方法的效率极其低下. 因此有必要研究 skyline 计算与关系操作间执行顺序变换的有效规则，并基于这些等价变换规则来提高用户查询的效率.

2 SKYLINE 计算操作符

定义 1(支配关系) 假定 p 和 r 是两个具有 v 个维度的元组，记维度集合为 $V=\{d_1,\cdots,d_v\}$. 如果它们满足下列两个条件，那么我们称 p 在 V 上支配 r：① $\forall i\in[1,v], p[i]\leqslant r[i]$；② $\exists j\in[1,v], p[j]<r[j]$.

不难看出，如果元 p 在 V 上支配 r，那么在 v 个指标的组合上，p 的各指标均不比 r 差，并且至少在一个指标上比 r 优. 为了简单起见，我们把 p 在 V 上支配 r，记为 $r\prec_V p$，在不发生混淆的情况下，省去字符‘V’，而写成 $r\prec p$. 并且把 p 不在 V 上不支配 r，记为 $r\not\prec_V p$.

定义 2(skyline 集合) 假定 AD 是具有 k 个维度对象全集，记维度集合为 $V=\{d_1,\cdots,d_v\}$. 那么 AD 上的 skyline 集合 $\nabla^V(AD)$ 可表示为：$\nabla^V(AD)=\{p\mid p\in AD\wedge\neg\exists r\in AD, p\prec_V r\}$.

同样，在不发生混淆的情况下，省去字符‘V’，而将 $\nabla^V(AD)$ 写成 $\nabla(AD)$.

注意，AD 上的 skyline 计算返回的结果为 $\nabla(AD)$.

3 变换 SKYLINE 操作符与传统关系操作符间的执行顺序

由于现有的工作均没有考虑 skyline 查询中涉及传统关系操作时，如何通过 skyline 操作符与各传统关系操作符之间执行顺序的变换来有效返回查询结果集；另一方面，根据数据库理论[16]，我们知道，适当地改变各操作符的执行顺序能够显著地提高查询的效率. 因此，本文首次研究 skyline 操作符与传统关系操作符间执行顺序变换的等价规则，并通过等价变换规则来有效缩减查询的实际开销. 图 2 给出文章中接下来常用到的一些符号及其含义.

符号	含义
∇	skyline 操作符
σ	选择操作符
π	投影操作符
$\times$	卡氏积操作符
$\triangleright\triangleleft$	连接操作符
$\cup$	并操作符
$\cap$	交操作符
$-$	差操作符

图 2 符号及其含义

规则 1—变换 ∇ 与 σ 间的执行顺序：

当 skyline 查询涉及关系选择操作时，为了得到正确的结果集，现有的方法先对底层关系表进行 skyline 计算(即实施 skyline 操作符)，然后，通过选择操作来获取满足选择条件的元组. 显然，当底层关系表较大时，这种方法的效率极其低下.

为了能够提高 skyline 查询涉及关系选择操作时的效率，本文从捕获选择条件所满足的性质出发来正确变换 skyline 操作符与选择操作符之间的执行顺序. 定理 1 表明，只要用户所给出的选择条件(满足如下性质，那么就能够正确变换它们之间的执行顺序：$\forall p, r\in AD, \lambda(p)\wedge p\prec r\Rightarrow\lambda(r)$，其中 AD 为有限元组全集.

定理 1 假定元组集为 AD，那么有：$\forall p, r\in AD, \lambda(p)\wedge p\prec r\Rightarrow\lambda(r)\Leftrightarrow\sigma_\lambda(\nabla(AD))=\nabla(\sigma_\lambda(AD))$，其中 σ_λ 为具有条件 λ 的关系选择操作.

例 2 在例 1 的表 T_1 中，假定 $V=\{A,B\}$. 此时用户只想返回 skyline 集合中在 B 上的取值小于 2 的元组，即 $\sigma_\lambda: t.B<2$. 现有的方法将分两个阶段来返回正确的结果：①求出 T_1 中在 V 上的 skyline 集合 $\nabla(T_1)=\{t_{12},t_{13}\}$；②从 $\nabla(T_1)$ 中获取在 B 上的取值小于 2 的元组，即 $\sigma_\lambda(\nabla(T_1))=\{t_{13}\}$. 然而我们发现，由于 σ_λ 满足条件 $O_1.B<2\wedge O_1\prec O_2\Rightarrow O_2.B<2$，因此，可以先对 T_1 进行选择操作，获取满足 σ_λ 的所有元组，即 $\sigma_\lambda(T_1)=\{t_{13}\}$；然后只需在数据集 $\sigma_\lambda(T_1)=\{t_{13}\}$ 上执行 skyline 计算即可，结果为 $\nabla(\sigma_\lambda(T_1))=\{t_{13}\}$.

规则 2—变换 ∇ 与 π 间的执行顺序：

假定用户需要获取两个维度组合上的 skyline 集合，记这两个维度组合分别为 V 和 U，并且有 $V\supset U$，即

V 是 U 的父空间.此时,为了得到正确的结果集,现有的方法分别对这两个维度组合进行操作,且它们的输入数据集均为底层的关系表 AD,显然,当 AD 所包含的元组个数较大时,这种方法的效率极其低下.然而,接下来的定理表明,我们只需经过如下 4 个步骤就可以获取正确的结果集:①首先,获取 V 上的 skyline 集合,即 $\nabla^V(AD)$;②然后,获取 U 上的种子 skyline 集合 $Seed = \nabla^U(\nabla^V(AD))$;③接着,获取 AD-$Seed$ 中,在 U 上与 $Seed$ 中的元组取值相同的元组集合 $Rep = \{p \mid p \in O\_Dom - Seed \wedge \exists r \in seed, \forall i \in U, r[i] = p[i]\}$,我们称 Rep 为 U 上的影子 skyline 集合;(4)最后,返回 V 上的 skyline 集合 $\nabla^V(AD)$以及 U 上的 skyline 集合 $Seed \cup Rep$ 给用户.

定理 2　假定有限元组全集为 AD,如果维度组合 V 和 U 满足 $V \supset U$,那么有 $\nabla^U(AD) = \nabla^U(\nabla^V(AD)) \cup \{p \mid p(AD - \nabla^V(AD) \wedge \exists r \in (\nabla^U(\nabla^V(AD)), \forall i \in U, r[i] = p[i]\}$.

推论 1　假定有限元组全集 AD 在各维度上均不存在重复值,那么,如果维度组合 V 和 U 满足 $V \supset U$,则有 $\nabla^U(AD) = \nabla^U(\nabla^V(AD))$.

例 3　在例 1 的关系表 T_1 中,若 $V = \{A, B\}$以及 $U = \{A\}$.现有的方法分别用一样的策略对这两个维度组合进行计算,得:$\nabla^V(T_1) = \{t_{12}, t_{13}\}$,以及 $\nabla^U(T_1) = \{t_{12}\}$.然而,我们可以在 $\nabla^V(T_1) = \{t_{12}, t_{13}\}$的基础上,首先获取 U 上的种子 skyline 集合 $Seed = \nabla^U(\nabla^V(T_1)) = \{t_{12}\}$,然后,获取 U 上的影子 skyline 集合 $Rep = \varnothing$.因此,$\nabla^U(T_1) = Seed \cup Rep = \{t_{12}\}$.

规则 3—变换 ∇ 与 $\times$ 间的执行顺序:

假定用户发出的 skyline 查询涉及底层两个关系表的卡氏积积操作,那么现有的方法均通过两个阶段来进行:①对这两个关系表执行卡氏积积操作;②然后,在所得到的中间表上进行 skyline 计算.显然,当这两个关系表较大,并且它们所包含的维度个数较多时,这种方法的效率极其低下.然而,接下来的定理表明我们只需要先多这两个关系表单独进行 skyline 计算,然后将所得到的两个 skyline 集合执行卡氏积积操作即可.

定理 3　假定存在两个有限元组全集 AD^1 和 AD^2,它们上的 skyline 维度分别为 V 和 U;那么有:$\nabla^{V+U}(AD^1 \times AD^2) = \nabla^V(AD^1) \times \nabla^U(AD^2)$,$\times$ 为关系积操作.

例 4　在例 1 中,假定 skyline 查询涉及两个关系表 T_1 和 T_2 的积操作,那么为了得到正确的结果集,现有的方法通过如下两个步骤来实现:①先对关系表 T_1 和 T_2 执行积操作,得到一个含有 26 个元组的中间数据集 T;②然后对 T 执行维度组合 $ALL = \{A, B, C, D, E, C\}$上的 skyline 计算.而根据定理 3,可以先对 T_1 和 T_2 这两个关系表单独执行的 skyline 计算,其中,对于 T_1,可得它上面的 skyline 集合 $\nabla^{\{A,B,C\}}(T_1) = \{t_{12}, t_{13}\}$,同样可得 $\nabla^{\{D,E,C\}}(T_2) = \{t_{21}, t_{23}\}$;接着,对 $\nabla^{\{A,B,C\}}(T_1)$与 $\nabla^{\{D,E,C\}}(T_2)$执行积操作即可.

规则 4—变换 ∇ 与$\triangleright \triangleleft$间的执行顺序:

假定用户发出的 skyline 查询涉及底层两个关系表 T_1 和 T_2 的关系连接操作,那么现有的方法均通过两个阶段来返回正确的查询结果集:①首先,对这两个关系表执行连接操作;②然后,在所得到的中间表上进行 skyline 计算.显然,当这两个关系表较大,并且它们所包含的维度个数较多时,这种方法的效率极其低下.

定理 4 表明,如果 skyline 维度只属于其中的一个关系表,那么 skyline 计算与自然连接操作之间执行顺序就可以等价变换.假定关系表 T_1 包含所有的 skyline 维度.此时,我们可以先执行 T_1 上的 skyline 计算,然后将所得到的结果集与 T_2 进行连接操作,从而,可以显著提高查询效率.

定理 4　假定存在两个有限元组全集 AD^1 和 AD^2,其中 AD^1 包含所有的 skyline 维度,并记 skyline 维度组合为 V,那么有:$\nabla^V(AD^1 \triangleright \triangleleft AD^2) = \nabla^V(AD^1) \triangleright \triangleleft AD^2$,$\triangleright \triangleleft$为关系连接操作.

例 5　在例 1 中,假定 skyline 查询涉及两个关系表 T_1 和 T_2 的连接操作,并且 skyline 计算的维度组合为 $V = \{A, B\}$.那么为了得到正确的结果集,现有的方法通过如下两个步骤来实现:①先对关系表 T_1 和 T_2 执行连接操作,得到一个含有 12 个元组的中间数据集 T;②然后对 T 执行 V 上的 skyline 计算.而根据定理 4,我们可以先对 T_1 执行 V 上的 skyline 计算,可得它上面的 skyline 集合 $\nabla^V(T_1) = \{t_{12}, t_{13}\}$;接着,我们将 $\nabla^V(T_1)$与 T_2 执行连接操作即可.

规则 6—变换 ∇ 与$\cup$间的执行顺序:

假定某一时刻,系统中存在两个同构关系表 T_1 和 T_2 的 skyline 集合 $\nabla^V(T_1)$和 $\nabla^V(T_2)$,此时,用户想获取这两个关系表合并之后的 skyline 集合.为了能得到正确的结果集,现有方法首先需合并 T_1 和 T_2 这两个同构的关系表,然后再在 $T_1 \cup T_2$ 上执行 skyline 计算.显然,当这两个关系表较大,这种方法的效率极其低下.然而,我们可以利用系统中的存在的两个 skyline 集合 $\nabla^V(T_1)$和 $\nabla^V(T_2)$来提高 $T_1 \cup T_2$ 上 skyline 计算的效率,定理 5 给出了等价变换规则的正确性.

定理 5　假定存在两个同构关系表 AD^1 和 AD^2,skyline 查询涉及的维度组合为 V,那么有:$\nabla^V(AD^1 \cup AD^2) \subseteq \nabla^V(AD^1) \cup \nabla^V(AD^2)$,$\cup$为关系并操作.

例 6　在例 1 中,假定存在一个与 T_1 同构的关系表 T_3,它包含 5 个元组,如图 3 所示,用户关心的维度

组合为 $V=\{A,B\}$.此时,系统中存在 T_1 和 T_3 的 skyline 集合 $\nabla^V(T_1)=\{t_{12},t_{13}\}$ 和 $\nabla^V(T_3)=\{t_{33}\}$.当用户想进一步查看 T_1 和 T_3 这两个表合并之后的 skyline 集合时,现有方法需首先合并这两个表,得到含有 11 条元组的中间表 $T_1\cup T_3$,然后再在 $T_1\cup T_3$ 上执行 skyline 计算,得到最终结果集 $\nabla^V(T_1\cup T_3)=\{t_{12},t_{13}\}$.然而,根据定理 5,我们可以先对 $\nabla^V(T_1)$ 和 $\nabla^V(T_3)$ 求并集,得 $\nabla^V(T_1)\cup\nabla^V(T_3)=\{t_{12},t_{13}\}$,然后,在 $\nabla^V(T_1)\cup\nabla^V(T_3)$ 上执行 skyline 计算即可,得 $\nabla^V(\nabla^V(T_1)\cup\nabla^V(T_3))=\{t_{12},t_{13}\}$.

元组标识	A	B	C
t_{31}	12	4	4
t_{32}	9	4	4
t_{33}	3	2	4
t_{34}	12	7	6
t_{35}	28	9	6

图 3 关系表 T_3

规则 7—变换 ∇ 与 $\cap$/- 间的执行顺序:

由于 ∇ 与 $\cap$ 间执行顺序变换的策略和 ∇ 与 - 间执行顺序变换的策略相同,因此,为了简单而不是一般性,我们将 $\cap$ 和 - 用通配符 $\oplus$ 表示,并将两个关系表 T_1 和 T_2 执行 $\oplus$ 操作之后的所得到集合称为 T_1 和 T_2 的 $\oplus$ 集.

假定某一时刻,系统中存在两个同构关系表 T_1 和 T_2 $\oplus$ 集的 skyline 集合 $\nabla^V(AD^1\oplus AD^2)$,此时,用户想获取这两个关系表 skyline 集合的 $\oplus$ 集 $\nabla^V(AD^1)\oplus\nabla^V(AD^2)$.为了能够得到正确的结果集合,现有的方法首先需要对 T_1 和 T_2 分别执行 skyline 计算,然后在对所得到的两个 skyline 集合执行 $\oplus$ 操作.显然,当这两个关系表较大,这种方法的效率极其低下.然而,我们可以利用系统中的存在的 skyline 集合 $\nabla^V(AD^1\oplus AD^2)$ 来提高获取 $\nabla^V(AD^1)\oplus\nabla^V(AD^2)$ 的效率,定理 6 给出了等价变换规则的正确性.

定理 6 假定存在两个同构关系表 AD^1 和 AD^2,skyline 查询涉及的维度组合为 V,那么有:$\nabla^V(AD^1)\oplus\nabla^V(AD^2)\subseteq\nabla^V(AD^1\oplus AD^2)$,$\oplus$ 代表关系交操作或关系差操作.

例 7 与例 6 类似,假定存在一个与 T_1 同构的关系表 T_3,用户关心的维度组合为 $V=\{A,B\}$.此时,系统中存在 T_1 和 T_3 交集的 skyline 集合 $\nabla^V(T_1\cap T_3)=\{t_{12}\}$.当用户想进一步查看 T_1 和 T_3 所对应 skyline 集合的交集 $\nabla^V(T_1)\cap\nabla^V(T_3)$ 时,现有的方法需要首先对 T_1 和 T_2 分别执行 skyline 计算,得 $\nabla^V(T_1)=\{t_{12},t_{13}\}$ 和 $\nabla^V(T_3)=\{t_{33}\}$,然后再在对 $\nabla^V(T_1)$ 和 $\nabla^V(T_3)$ 进行交操作,得 $\nabla^V(T_1)\cap\nabla^V(T_3)=\{t_{12}\}$.然而,根据定理 6,我们可直接对 $\nabla^V(T_1\cap T_3)$ 执行 skyline 计算即可,得 $\nabla^V(\nabla^V(T_1\cap T_3))=\{t_{12}\}$.

4 实验评估

在这一节中,我们通过具体的实验来评估子空间 skyline 计算与关系操作间执行顺序变换前(即现有的方法)后(即我们的方法)的性能.实验数据由文献[1]给出的数据生成器来产生.文献[1]中的数据生成器产生的元组在各个维度上是相互对立的,因此根据文献[17]可知,对于一个具有 N 个维度的关系表 T,它 skyline 集合基数的期望值为 $\overline{SKY(T,N)}=(\log|T|)^{N-1}/(N-1)!$,而获取 skyline 集合的理论时间值为 $\overline{TIME}=\sum_{i=1}^{|T|}(\overline{SKY(T,N)}/(i-1)\times\overline{SKY(T,N+1)})$.

实验中用到的每一类查询语句由两个操作组成,一个是 skyline 计算,另一个是传统的关系操作符,并且选择率取 0.1,投影率(投影维度与总维度的比值)取 0.5.为了简单而不失一般性,所有表维度数据格式均为符点型.实验分为两组完成:(1)表基数不变,目标维数逐渐增大;(2)目标维数不变,表基数逐渐增大.实验环境为:PIV 2.4G CPU,1G 内存,160G 硬盘,Windows XP 平台.所有程序均在 JBuilder 9 中调试通过.

4.1 实验一:查询时间随维度个数变化

这一组实验中,我们产生三个表 T_1、T_2 和 T_3,其中 T_1 和 T_2 的表结构相同,用来做 skyline 计算与并/交/差操作的实验;skyline 计算与选择以及投影操作的实验在 T_1 上进行;而 skyline 计算与积和连接操作的实验在 T_1 和 T_3 上进行.T_1、T_2 和 T_3 的表基数均为 5×10^5.维度个数分别取 2,4,6,8.为了方便显示实验结果图,我们把本文的方法和现有的方法分为记为"Our"和"Traditional".图 4(a)~(g)给出了实验的结果.

从图 4 可以看出,skyline 计算与关系操作之间执行顺序变换之后的时间开销比变换之前有着显著的提高.例如,在图 5(a)中,skyline 计算与选择操作变换之前,现有的方法需要对整个数据集进行 skyline 计算,而变换之后,我们的方法只需要对满足条件的元组集合进行 skyline 计算,因此,查询性能有着显著的提高.另一方面,我们发现,随着维度个数的增大,它们之间的性能提高程度越来越明显.例如在图 5(c)中,当表的维度个数为 2 时,skyline 计算与积操作变换前的时间开销为 104.4 秒,而 skyline 计算与积操作变换之后的时间开销为 22 秒,即此时我们方法的时间为现有方法的时间的 21.1%;然而当表的维度个数为 8 时,skyline 计算与积操作变换之前的时间开销为 1666.8 秒,而 skyline 计算与积操作变换之后的时间开销为 256 秒,即此时我们

方法的运行时间为现有方法的时间的15.4%.

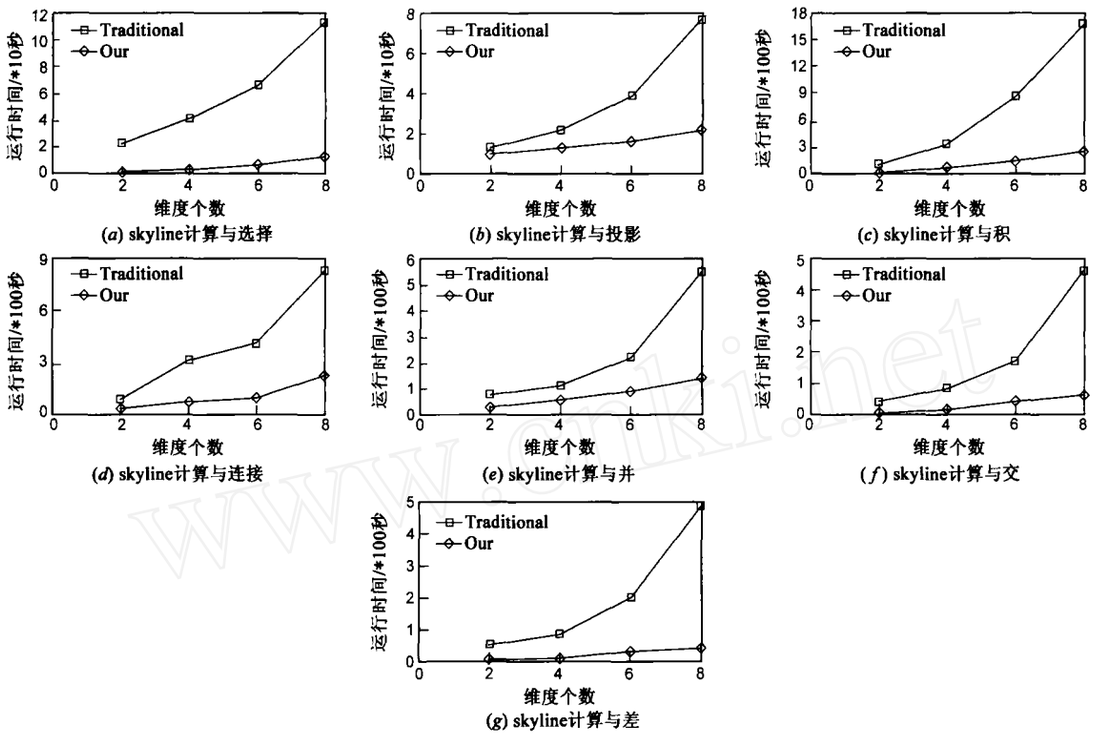


图4　运行时间随维度个数变化

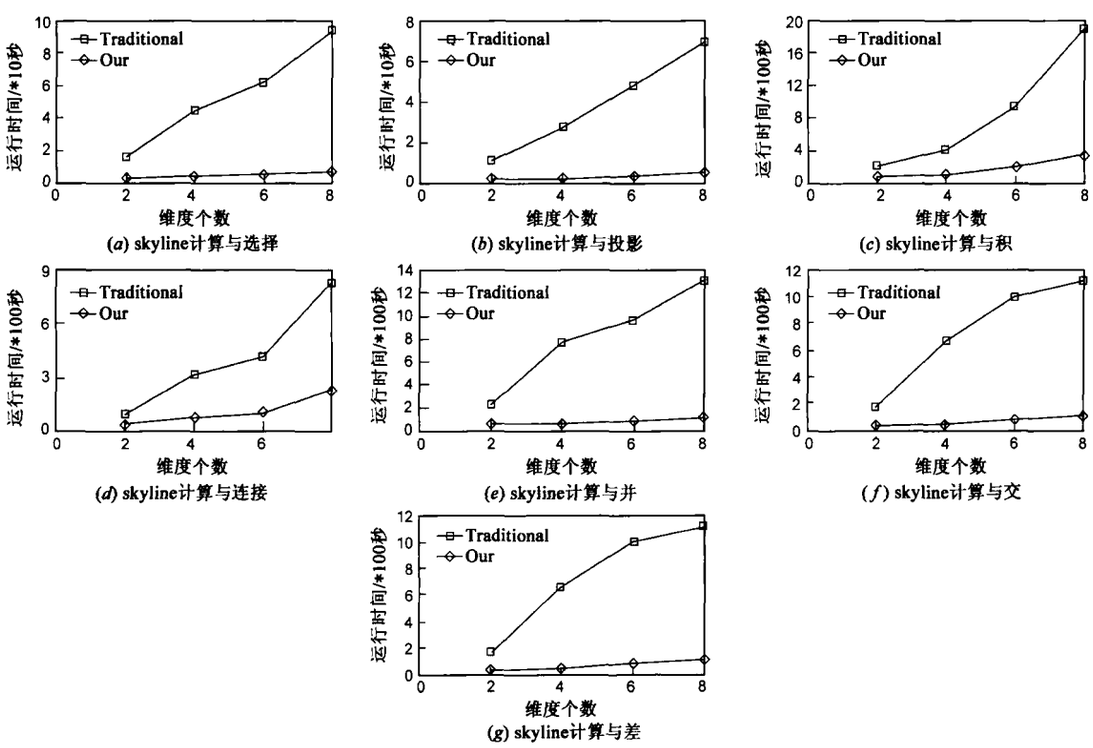


图5　运行时间随表基数数变化

4.2　实验二:查询时间随表基数变化

与实验一相同,在这一组实验中,我们产生三个表 T_1、T_2 和 T_3,其中 T_1 和 T_2 的表结构相同,用来做 skyline计算与并/交/差操作的实验;skyline 计算与选择以及投影操作的实验在 T_1 上进行;而 skyline 计算与积和连接操作的实验在 T_1 和 T_3 上进行.维度个数数固定为

6. T_1、T_2 和 T_3 的表基数分别取 2×10^5、4×10^5、6×10^5 和 8×10^5. 为了方便显示实验结果图,我们把本文的方法和传统的方法分为记为"Our"和"Traditional". 图 5(a)~(g)给出了相应的实验结果.

这组实验的实验结论与图 4 中的实验结果类似,即 skyline 计算与关系操作之间执行顺序变换之后的时间开销比变换之前有着显著的提高. 并且随着表基数的增大,它们之间的性能提高程度越来越明显. 例如,在图 5(c)中,当表的基数为 2×10^5 时,skyline 计算与积操作变换之前的时间开销为 212.5 秒,而 skyline 计算与积操作变换之后的时间开销为 7.56 秒,即此时我们方法的时间为现有方法的时间的 35.7%;然而,当表的基数为 8×10^5 时,skyline 计算与积操作变换之前的时间开销为 1892.4 秒,而 skyline 计算与积操作变换之后的时间开销为 346.1 秒,即此时我们方法的时间为现有方法的时间的 18.3%.

5 相关工作

S. Borzsonyi 等人[1]首次将 skyline 查询引进到数据库领域中,并提出两个可行的查询算法:BNL 算法以及 DC 算法. 假定对象全集为 AD. 其中 BNL 算法通过 $O(|AD|^2)$次对象间的比较来找出返回完整的 skyline 结果集,而 DC 算法使用递归分区的方法来获取查询结果. J. Chomicki 等人[2]在 BNL 算法的基础上提出一种先对象进行排序,再进行比较的查询方法 SFS,SFS 算法能够有效减少 BNL 算法中对象间比较的次数,然而,它增加了排序的时间开销. P. Godfrey 等人[3]从理论上给出在均匀分布情况下,BNL 算法、DC 算法以及 SFS 算法的时间开销,并提出一种基于外部排序的可行方法 LESS,算法 LESS 的时间复杂度可降为 $O(|AD|\log|AD|+k|AD|)$,其中 k 为 skyline 维度个数. D. Kossmann 等人[4]给出一种基于 R-树索引的计算方法 NN,NN 算法递归搜索区域的最近邻点,并且通过区域剪枝技术删除被最近邻点支配的所有对象,从而提高了获取查询结果的速度. D. Papadias 等人[5]指出 NN 算法缺陷,并提出一种基于排序 R-树节点的方法 BBS,BBS 算法克服了 NN 算法冗余比较 R-树节点的不足,而且比 NN 算法具有更强的剪枝能力. 实验评估表明,在通常情况下,BBS 算法具有最好的查询效率.

Y. Tao 等人[6]首次考虑在数据流上维护 skyline 对象,并以 R-树索引为基础,给出两个基于区域查询技术的有效维护方法:I-Eager 算法和 I-Lazy 算法. 文献[6]中的实验表明 I-Lazy 算法的性能相对优于 I-Eager 算法. X. Lin[7]等人考虑在数据流环境上有效维护最近 N 个 skyline 对象的问题,特别,他们进一步研究了 n-of-N skyline 查询问题,即返回最近 $n(\forall n\leq N)$个 skyline 对象问题. M. Sharifzadeh 等人[8]首次在空间数据库上研究 skyline 查询,并给出空间 skyline 查询的概念. 作者利用空间几何领域[16]中的有效性质来快速返回空间 skyline 查询的结果集,从而将空间 skyline 查询的时间复杂度从 $O(|P|^2Q)$降低至 $O(|S|^2C+P^{1/2})$,其中 P 为空间数据点集合,S 为搜索方案数,以及 C 为查询点集合 Q 中的凸包顶点数. 为了丰富现有数据库产品的查询引擎,M. Goncalves 等人[9]首次考虑将 skyline 查询技术和 *Top-k* 查询技术[17]相结合,提出 Hybrid-Naïve 操作符,并扩展了传统的 SQL 语句,让用户可以从 skyline 和 Top-k 两个角度来有效考察数据,从而使得用户可以更加准确捕获代考察的对象. Q. Li 等人[10]将 skyline 查询技术引进到时间序列研究邻域来克服维度危机问题. J. Pei 等人[11]等人考虑如何在不确定的数据集上进行有效的 skyline 查询计算,并给出一种新颖的概率 skyline 模型来解决当对象出现的概率不确定的情况. 当数据对象持续移动时,Z. Huang 等人[12]考虑持如何有效监控不同时刻上的 skyline 集合. W. Jin 等人[13]扩展 skyline 查询语义,提出 Thin Skyline 查询的概念. Thin Skyline 查询返回的结果集包括两个部分:(1)skyline 对象集合;以及(2)在 ε 距离内的最近邻对象. 作者给出两个有效的方法,即采样剪枝算法以及索引评估算法. 这两个算法基于统计信息来有效搜索 thin skyline 对象. 同时,作者给出一个基于微簇的高效算法 MBA 来对返回的 thin skyline 对象集合进行挖掘. MBA 算法在挖掘性能和可视化 thin skyline 对象两个方面具有显著的有效性和实用性. P. Wu 等人[14]首次研究分布式网络中的 skyline 查询,并以 CAN[18]体系架构为基础. Z. Huang 等人[15]首次考虑在多用户环境中,如何有效处理和优化多个 skyline 查询.

然而,值得注意的是,现有的相关工作均没有考虑传统关系操作(如选择、卡氏积以及连接等)存在于 skyline 查询的情况,而只是简单地假设用户的查询不涉及任何关系操作. 因此,在实际应用中,这些研究工作具有很大的局限性.

6 结论

现有的研究工作均没有考虑 skyline 计算与传统关系操作共同存在于用户所发出的查询的情况,因此,当 skyline 查询涉及选择、卡氏积以及连接等传统关系操作时,为了能够得到正确的查询结果,现有的方法均是按照用户查询中各操作的发出的顺序对底层关系表进行计算,因此,在实际应用中,这种执行方式的效率极其低下. 基于此,本文对用户给出的 skyline 查询进行有效地解析,在逻辑层面上,通过变换 skyline 计算和各传统关系操作间的执行顺序来有效提高 skyline 查询的效率. 我们从理论上证明各个等价变换规则的正确性. 另

一方面,我们通过具体的实验来评估本章所给出变换规则的有效性,实验结果表明,变换后的查询效率比变换前的查询效率有着显著的提高.

参考文献:

[1] S Borzsonyi, D Kossmann, K Stocker. The skyline Operator [A]. Proc IEEE ICDE '01 [C]. Heidelberg: IEEE Press, 2001.421-430.

[2] J Chomicki, P Godfrey, J Gryz, D Liang. Skyline with presorting: theory and optimization [A]. Proc ICIIS '05 [C]. Maryland: MIT Press, 2005.216-225.

[3] P Godfrey, R Shipley, J Gryz. Maximal vector computation in large data sets [A]. Proc. VLDB '05 [C]. Trondheim: VLDB Endowment, 2005.229-240.

[4] D Kossmann, F Ramsak, S Rost. Shooting stars in the sky: an online algorithm for skyline queries [A]. Proc VLDB '01 [C]. Roma: VLDB Endowment, 2001.311-322.

[5] D Papadias, Y Tao, G Fu, B Seeger. Progressive skyline computation in data systems [J]. ACM Transaction on Database Systems, 2005, 30(1): 41-82.

[6] Y Tao, D Papadias. Maintaining sliding window skylines on data streams [J]. IEEE Transaction on Knowledge and Data Engineering, 2006, 18(3): 377-391.

[7] X Liu, Y Yuan, W Wang, H Lu. Stabbing the sky: efficient skyline computation over sliding windows [A]. Proc IEEE ICDE '05 [C]. Tokyo: IEEE Press, 2005.502-513.

[8] M Sharifzadeh, C Shahabi. The spatial skyline queries. [A]. Proc VLDB '06 [C]. Seoul: VLDB Endowment, 2006. 751-762.

[9] M Goncalves, M Vidual. Preferred skyline: a hybrid approach between SQLf and skyline [A]. Proc. DEXA '05 [C]. Copenhagen: Springer Verlag, 2005.375-384.

[10] Q Li, L Lopez, B Moon. Skyline index for time series data [J]. IEEE Transaction on Knowledge and Data Engineering, 2004, 16(6): 669-684.

[11] J ei, B. Jiang, X Lin, Y Yuan. Probabilistic skylines on uncertain data [A]. Proc VLDB '07 [C]. Vienna: VLDB Endowment, 2007.641-652.

[12] Z Huang, W Wang. A novel incremental maintenance algorithm of SkyCube [A]. Proc DEXA '06 [C]. Kraków: Springer Verlag, 2006.781-790.

[13] W Jin, J Han, M Ester. Mining thick skylines over large databases [A]. In Proc. PKDD '04 [C]. Pisa: Springer Verlag, 2004.255-266.

[14] P Wu, C Zhang, Y Feng, B Zhao, D Agrawal, A Abbadi. Parallelizing skyline queries for scalable distribution [A]. Proc EDBT '06 [C]. Munich: Springer Verlag, 2006.112-130.

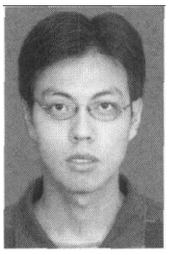
[15] Z Huang, J Guo, S Sun, W Wang. efficient optimization of multiple subspace skyline queries [J]. Journal of Computer Science and Technology, 2008, 23(1): 103-111.

[16] 阳国贵,吴泉源.对象关系数据库中一类特定连接谓词的有效处理方法[J].电子学报,2000,28(11):111 - 113.
Yang Guo-gui, Wu Quan-yuan. Researches on join index technology in ORDB [J]. Acta Electronica Sinica, 2000, 28(11): 111 - 113. (in Chinese)

[17] R Akbarinia, E Pacitti, P Valduriez. Best position algorithm for top-k queries [A]. In Proc. VLDB '07 [C]. Vienna: VLDB Endowment, 2007.495-506.

[18] 胡建强,郭长国,王怀民,邹鹏.一种基于 P2P 网络的 Web 服务发现方法[J].电子学报,2005,33(12):2503 - 2508.
Hu Jian-qiang, Guo Chang-guo, Wang Huai-min, Zou Peng. A P2P based distributed web services discovery method [J]. Acta Electronica Sinica, 2005, 33(12): 2503-2508. (in Chinese)

作者简介:



黄震华　男,1980 年生,博士,讲师,软件行业协会系统工程分会理事,CCF 会员.主要研究方向为数据库、数据仓库、OLAP 分析、数据挖掘与知识发现等.
Email: jukie.huang@gmail.com

向　阳　男,1962 年生,博士,教授.主要研究方向为数据库、数据仓库、数据挖掘、决策支持系统与语义网等.

林　琛　男,1982 年生,博士研究生,CCF 学生会员.主要研究方向为数据库、数据挖掘、信息检索与社会网络分析等.

孙圣力　男,1979 年生,博士,讲师,CCF 会员.主要研究方向为数据库、数据流挖掘与服务计算等.