ISSN 1000-1239/CN 11-1777/TP
45(8): 1379-1385, 2008

# 支持第三级存储器的查询优化方法的研究

刘宝良　李建中　高　宏
(哈尔滨工业大学计算机科学与技术学院　哈尔滨　150001)
(liubaoliang@gmail.com)

# Study of Query Optimization Methods for Data on Tertiary Storage

Liu Baoliang, Li Jianzhong, and Gao Hong
(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)

**Abstract** The management of DBMS on tertiary storage is becoming more and more important with the development of applications, not only because tertiary devices are used to archive data, but also the amount of data that application has to deal with is increasing rapidly. The cost model and query optimization method of current disk based database management system can't deal with massive data on tertiary storage. A cost model which can evaluate relational operations for tertiary resident data is proposed. The cost of various relational operations can be deduced through the cost definitions of several basic data accessing pattern and the costs of two pattern combination operators. To further improve query efficiency, multiple relation copies are stored on the tertiary storage with different organization methods. The cost model can also evaluate the cost of the same relational operation on different relation copies. Two query optimization methods are also proposed, which can not only choose the most efficient implementation algorithm for relational operators, but also choose the most I/O efficient copy of the relation on tertiary storage. The experimental results show that query efficiency for tertiary resident data can be greatly improved by adapting the proposed cost model and the query optimization methods. The introduction of relation copies demonstrates the feasibility of improving query efficiency at the cost of using more storage space.

**Key words** cost model; query optimization; tertiary storage; relational operation; data access pattern

**摘　要** 目前的关系数据库代价模型及查询优化算法无法处理保存在第三级存储器中的海量数据.提出了估算第三级关系代数操作的代价模型,通过定义若干基本数据访问模式及两种模式合成方法的代价,导出关系代数操作的代价.提出了针对第三级存储器的查询优化方法,该方法不仅可以选择最高效的关系代数操作实现算法,而且可以选择I/O代价最小的关系副本,从而提高查询效率.实验结果表明,应用提出的代价模型及查询优化方法后可以显著地提高第三级存储器上数据的查询效率.关系副本的引入充分证明了用存储空间换取查询执行时间的策略的可行性.

**关键词** 代价模型; 查询优化; 第三级存储器; 关系代数操作; 数据访问模式

**中图法分类号** TP311.13

商业及科研应用积累的数据已经达到T字节($1TB=10^{12}B$)甚至P字节($1PB=10^{15}B$)数量级.如此庞大的数据通常存放在以机械手磁带库为代表的第三级存储设备中.目前的数据库系统虽然能够高效地处理磁盘中的数据,然而却无法直接管理第三级存储器上的数据.尽管针对磁盘数据的查询处理与查询优化方法得到了充分的研究并提出了若干高效实用的查询优化方法,第三级存储器上的关系操

**收稿日期**:2007-04-22;**修回日期**:2008-02-18

**基金项目**:国家自然科学基金项目(60273082);国家"八六三"高技术研究发展计划基金项目(2002AA444110);国家"九七三"重点基础研究发展规划基金项目(G1999032704);黑龙江省自然科学基金项目(zjg03-05)

作代价模型和基于代价的查询优化方法的研究却很少见.针对第三级数据的组织特点与访问特点,本文首次提出适用于第三级存储器的关系操作代价模型,并以此模型为基础,提出了3种针对第三级存储器的查询优化方法.

为叙述清晰起见,我们称需要机械手臂对脱机的存储介质进行加载才能访问数据的设备为第三级存储器,例如机械手光盘库和机械手磁带库.保存在第三级存储器上的数据为第三级数据.操作第三级数据的关系操作为第三级关系操作.估算第三级操作代价的模型为第三级代价模型.对处理第三级数据的查询进行优化的查询优化方法为第三级查询优化方法.以往关于第三级查询优化的研究要么是简单照搬磁盘查询优化方法处理第三级数据,要么是对机械手臂进行调度,效果都不理想.

## 1 相关工作

第三级查询优化是解决第三级数据管理的关键问题之一.早在1995年,Sarawagi就对PostgreSQL中的查询处理进行扩充,使其支持机械手光盘库中的数据查询[1].首先按照传统的优化方法选择代价最小的执行方案,然后将这个查询分解成若干可以独立执行的子查询,每个子查询需要处理的所有数据位于同一张光盘中.查询调度器调整子查询的执行顺序,使查询当前驱动器中数据的子查询优先执行,以降低磁盘换入换出的额外开销. Yu等人采用查询预执行及批处理的方法提高查询的执行效率[2],并在Paradise[3]系统中验证了其方法的有效性.在Paradise系统中,保存在机械手磁带库中的所有数据在磁盘中都有一个对应的代理标识,查询首先在磁盘中的代理数据上执行一遍,得到数据的真实访问模式.再根据访问模式确定优化的调度方法.对于并发执行的查询,根据访问模式对其进行统一调度,以减少切换磁带及磁带定位的代价.文献[4]研究了磁头定位对查询操作的性能影响,并提出了两种改善磁头定位效率的连接算法.

## 2 第三级代价模型

本节首先定义基本数据访问模式,然后阐述基本数据访问模式的结合方法,并举例说明利用基本数据访问模式定义关系操作的方法.最后给出针对不同存储介质计算数据访问模式代价的方法.

### 2.1 第三级代价模型

**定义1.** 数据区域. $B(w,n,off,D,T_S,T_R,s)$ 数据区域 $B$ 是指存储在设备 $D$ 上包含 $n$ 个长度为 $w$ 的元组的连续存储区域. $B$ 在 $D$ 上的起始位置为 $off$,设备 $D$ 的定位速度为 $TS$,读写速度为 $TR$. $s=0$ 表明数据区域所在的存储介质是联机的,$s=1$ 表明数据区域所在的存储介质是脱机的.关系可由一个或多个数据区域构成.不失一般性,本文假定关系由一个数据区域构成.

通过对多种关系操作过程的详细分析,可以得出关系操作具有以下几种主要的基本访问模式:

**定义2.** 单次顺序扫描 $s\_s(B)$. 按照存储顺序访问数据区域 $B$ 上的每一个元组仅一次.

**定义3.** 多次顺序扫描 $m\_s(B, m, d)$. 按照存储顺序访问数据区域 $B$ 上每个元组 $m$ 次. $d$ 表示两次相邻访问之间磁头的运动方向,如果 $d=uni$ 表明相邻的两次数据扫描磁头的运动方向相同. $d=bi$ 表明相邻的两次数据扫描磁头的运动方向相反.如果存储介质为磁带并且 $d=uni$,则在相邻扫描之间需要倒带.

**定义4.** 交互扫描 $nest(B|_1^m, O[, d])$. $m$ 个内游标访问数据区域 $B_i(1\leqslant i\leqslant m)$. 如果 $O=ran$,则外游标随机地选择内游标中的数据;如果 $O=seq$,则外游标顺序地访问内游标的数据.外游标随机选取数据的条件与数据的分布有关,由于不能假定数据的分布特性,通常认为数据分布是完全随机的.在 $O=seq$ 的情况下,如果 $d=uni$ 表明相邻两次扫描中外游标按照相同的顺序依次选择每个内游标;如果 $d=bi$,表明在相邻的两次选择某一内游标时外游标的运动方向相反.

### 2.2 基本数据访问模式的合成

**定义5.** ＋合成算子. 设 $p_1,\cdots,p_n$ 为数据访问模式,$p_1+\cdots+p_n$ 运算表明访问模式 $p_i$ 先于访问模式 $p_{i+1}$ 执行,并且 $p_i$ 与 $p_j$ 访问位于同一设备的数据区域.

**定义6.** ×合成算子. 设 $p_1,\cdots,p_n$ 为数据访问模式,$p_1\times\cdots\times p_n$ 运算表明访问模式 $p_i$ 与访问模式 $p_j$ 并发执行($1\leqslant i,j\leqslant n$, $i\neq j$),并且 $p_i$ 与 $p_j$ 访问位于不同设备的数据区域.

**定义7.** 访问模式空间 $P$. 由基本数据访问模式及基本数据访问模式经由＋或×运算构成的合成模式构成的集合.×运算的优先级高于＋运算.

表1给出了几种典型关系操作的数据访问模式:

**Table 1 Cost Model of Relational Operations**

**表 1 关系操作的代价模型**

| Relational Operation | Data Access Pattern | Alias |
|---|---|---|
| $Select(U) \rightarrow W$ | $s\_s(U) \times s\_s(W)$ | |
| $NL\text{-}join(U,V) \rightarrow W$ | $s\_s(U) \times m\_s(U.n, uni, V) \times s\_s(W)$ | |
| $hash\_build(U) \rightarrow H\|_1^n$ | $s\_s(U) \times nest(H\|_1^n, ran)$ | $build\_hash(U,H)$ |
| $hash\text{-}join(U,V) \rightarrow W$ | $build\_hash(U,H) + build\_hash(V,H') + s\_s(H) + s\_s(H') \times s\_s(W)$ | |
| $attr\_sep(U) \rightarrow H\|_1^n$ | $s\_s(U) \times nest(V\|_1^n, s\_s, seq)$ | $attr\_sep(U,H)$ |
| $ASJ(U,V) \rightarrow W$ | $attr\_sep(U,H) + attr\_sep(V,K) + hash\_join(H1,K1,C) + s\_s(H2 \leqslant i \leqslant n) + s\_s(K2 \leqslant i \leqslant n) \times s\_s(W)$ | |
| $CDT\text{-}GH(U,V) \rightarrow W$ | $build\_hash(U,H) \times s\_s(V) \times s\_s(W)$ | |
| $NDT(U,V) \rightarrow W$ | $build\_hash(U,H) \times s\_s(V) \times s\_s(W)$ | |

### 2.3 基本数据访问模式的代价

访问外存数据的代价主要由磁头定位时间与数据传输时间构成,第三级存储器的脱机数据的访问还包括将存储介质装载到驱动器中的时间.存储介质的具体访问代价的计算详见文献[5].

数据区域 $B(w,n,off,D,T_S,T_R,s)$的访问代价根据存储介质及联机脱机的不同,可以表示为

$$T(B)=\begin{cases} w\times n/T_R, & \text{if } D=\text{disc},\\ w\times n/T_R+s\times T_{\text{oload}}, & \text{if } D=\text{optic},\\ w\times n/T_R+|\ cur-off\ |/T_S+s\times T_{\text{tload}}, & \text{if } D=\text{tape}. \end{cases}$$

单次顺序扫描 $s_s(B)$的访问代价显然等于数据区域 $B$ 的访问代价.

多次顺序扫描 $m_s(B, m, d)$的代价为

$$T(m\_s)=\begin{cases} m\times w\times n/T_R, & \text{if } D=disc,\\ m\times w\times n/T_R+s\times T_{\text{oload}}, & \text{if } D=optic,\\ m\times w\times n/T_R+|\ cur-off\ |/T_S+ & \\ \quad (m-1)\times w\times n/T_S+s\times T_{\text{tload}}, & \\ \quad \text{if } D=tape, d=uni,\\ m\times w\times n/T_R+|\ cur-off\ |/T_S+ & \\ \quad s\times T_{\text{tload}}, & \text{if } D=tape, d=bi. \end{cases}$$

交互扫描 $nest(B|_1^m, O[,d])$的代价为

$$T(nest)=\begin{cases} m\times w\times n/T_R, & \text{if } D=disc,\\ m\times w\times n/T_R, & \text{if } D=optic,\\ m\times w\times n/T_R+m\times(\frac{1}{2}\times m\times w\times n)/ & \\ \quad T_S+s\times T_{\text{tload}}, & \text{if } D=tape, O=r,\\ m\times w\times n/T_R+|\ cur-off\ |/ & \\ \quad T_S+(m-1)\times w\times n/T_S+s\times T_{\text{tload}}, & \\ \quad \text{if } D=tape, O=s, d=uni,\\ m\times w\times n/T_R+|\ cur-off\ |/T_S+s\times T_{\text{tload}}, & \\ \quad \text{if } D=tape, O=s, d=bi. \end{cases}$$

**算法 1.** DynProgOp.

输入:$R=\{R_1, R_2, \cdots, R_k\}$,关系统计信息查询 $Q$;

输出:查询执行计划.

```
Begin
① For i=1 To k Do
②   If R_i on disks Then
③     S_Map.put(R_i, MinAcs(R_i), Cost(MinAcs(R_i)))
④   Else
⑤     For j=1 To |Ĉ(R_i)| Do
⑥       S_Map.put(R_i, MinCover(R_i), Cost(MinCover(R_i)))
⑦ For i=1 To S_Map.length-1 Do
⑧   For j=i+1 To S_Map.length Do
⑨     T_i=S_Map.getTable(i)
⑩     T_j=S_Map.getTable(j)
⑪     If T_i.tableName ! =T_j.tableName Then /*Different table*/
⑫       E = new Entry(Join(T_i, T_j), AcsMthd(T_i), AcsMthd(T_j), MinJoinCost(T_i, T_j))
⑬       C_Map.put(E)
⑭       C_Map.setMaxEntrySize(2)
        /*now we are joining two tables*/
⑮ While S_Map.isEmpty()==FALSE Do
⑯   Costmin=∞
⑰   Foreach C in C_Map Do
⑱     T_Map.add(All tables in S_Map which have join attribute with C)
```

```
⑲  If T_Map.isEmpty()==TRUE Then
⑳    T_Map=all copies in S_Map which
     are not copies of relations in C
     /* Catestian product */
㉑  Foreach T in T_Map Do
㉒    If MinJoinCost(T, C)<MinJoinCost
     (C, T) Then
㉓      If MinJoinCost(T, C)<Costmin
       Then
㉔        Costmin=MinJoinCost(T, C)
㉕        T_min=AcsMthd(T)
㉖        E=new Entry(Join(T_min, C),
         AcsMthd(T), C, Costmin)
㉗    Else
㉘      If MinJoinCost(C, T)<Costmin
       Then
㉙        Costmin=MinJoinCost(C, T)
㉚        T_min=AcsMthd(T)
㉛        E=new Entry (Join(C, T_min),C,
         AcsMthd(T), Costmin)
㉜    C_Map.add(E)
㉝      S_Map.removeAllCopies(T_min.
       tableName)
㉞    C_Map.setMaxEntrySize(C_Map.
     getMaxEntrySize()+1)
㉟    C_Map.removeEntries(C_Map.
     getMaxEntrySize()-1)
㊱ Return C_May.getMinCostEntry()
End
```

+合成算子的操作数位于同一设备,需要串行执行,因此经过+算子合成访问模式的代价为

$$T(p_1+\cdots+p_n)=\sum_{i=1}^{m}T(p_i).$$

×合成算子的操作数位于不同设备,可以并发访问,因此经过×算子合成的访问模式的代价为

$$T(p_1\times\cdots\times p_n)=\mathrm{Max}\{(T(p_1),\cdots,T(p_n)\}.$$

## 3 第三级数据组织结构

**定义 8.** 关系副本. 设关系 $R$ 具有属性 $a_1,\cdots,a_n$, 设 $G$ 为集合$\{a_1,\cdots,a_n\}$的一个非空子集,如果 $G_1,\cdots G_m$是集合$\{a_1,\cdots,a_n\}$的一个划分,则称集合$\{C_1(rowid, G),\cdots, C_m(rowid, G_m)\}$为关系 $R$ 的一个副本,$C_i(1\leqslant i\leqslant m)$为副本中的元素. 关系 $R$ 的所有副本构成的集合称为关系 $R$ 的副本集,表示为 $\hat{C}(R)$. $\hat{C}(R)$中包含的副本个数成为副本集的势,表示为$|\hat{C}(R)|$.

**定理 1.** 关系副本集中所有副本的所有元素中的每个元组间具有一一对应关系,且元组的逻辑顺序相同.

证明. 显然.

**定义 9.** 最小关系覆盖. 设关系属性集$\{a_1,\cdots,a_n\}$及其子集$\{a_i,\cdots,a_j\}(1\leqslant i\leqslant j\leqslant n)$, $\{a_1,\cdots,a_n\}$的一个划分$\{C_1,\cdots,C_m\}$,每个 $C_i(1\leqslant i\leqslant m)$是$\{a_1,\cdots,a_n\}$的一个子集. 如果划分$\{C_1,\cdots,C_m\}$的一个子集$\{C_l,\cdots,C_k\}$包含所有$\{a_i,\cdots,a_j\}$的属性,则称$\{C_l,\cdots,C_k\}$为$\{a_i,\cdots,a_j\}$的覆盖. 如果去掉$\{C_l,\cdots,C_k\}$中任何一个元素,其均不能构成$\{a_i,\cdots,a_j\}$的覆盖,则称$\{C_j,\cdots,C_k\}$为$\{a_i,\cdots,a_j\}$的最小覆盖. 对应于划分$\{C_l,\cdots,C_k\}$的关系副本称为属性集$\{a_i,\cdots,a_j\}$的最小关系覆盖.

## 4 第三级查询优化方法

### 4.1 动态规划近似算法

基于动态规划方法的近似优化算法能够在大多数情况下返回最优执行方案,即使没有找到最优执行方案,其返回的执行方案与最优执行方案也比较接近. 并且随着连接个数增多,该算法也具有很好的执行效率.

动态规划近似算法的形式化描述如算法 DynProgOp.

**算法 2.** GreedyOpt.

输入: $R=\{R_1, R_2,\cdots, R_k\}$, 关系统计信息查询 $Q$;

输出: 查询执行计划.

```
Begin
① For i=1 To k Do
②   If R_i on disks Then
③     S_Map.put (MinAcs(R_i), cost
      (MinAcs(R_i)))
④   Else
⑤     For j=1 To |Ĉ(R_i)| Do
```

⑥ $S\_Ma.put(R_i, MinCover(R_i), cost(MinCover(R_i))$

⑦ $Costmin=\infty$

⑧ For $i=1$ To $S\_Map.length-1$ Do

⑨ For $j=i+1$ To $S\_Map.length$ Do

⑩ $T_i=S\_Map.getTable(i)$

⑪ $T_j=S\_Map.getTable(j)$

⑫ If $T_i.tableName$ ! $=T_j.tableName$ and $MinJoinCost(T_i, T_j)<Costmin$ Then

⑬ $Costmin=MinJoinCost(T_i, T_j)$

⑭ $E_{min}=$new Entry(Join($T_i$, $T_j$), $AcsMthd(T_i)$, $AcsMthd(T_j)$, $Costmin$)

⑮ $S\_Map.removeAllCopies(E_{min}.getLeftTable().tableName)$

⑯ $S\_Map.removeAllCopies(E_{min}.getRightTable().tableName)$

⑰ While $S\_Map.isEmpty()$==FALSE Do

⑱ $Costmin=\infty$

⑲ Foreach entry $T$ in $S\_Map$ Do

⑳ If $MinJoinCost(E_{min}, T)<Costmin$ Then

㉑ $Costmin=MinJoinCost(E_{min}, T)$

㉒ $T_{min}=T$

㉓ $E_{min}=$new Entry(Join($E_{min}$, $T_{min}$), $E_{min}$, $AcsMthd(T_{min})$, $Costmin$)

㉔ $S\_Map.removeAllCopies(E_{min}.tableName)$

㉕ Return $E_{min}$ and $Costmin$

End

在算法中,$MinAcs(R)$ 是磁盘上关系 $R$ 的最小代价访问路径,可能是索引扫描等.$MinCover(R)$为第三级存储器上关系 $R$ 的最小关系覆盖. $Join(R, S)$表示关系 $R$ 与 $S$ 的逻辑连接操作.$MinJoinCost(R,S)$表示两个关系的最小连接代价,可能是磁盘连接,也可能是第三级存储器上关系连接.$AcsMthd(T)$表明在当前的最小连接代价下关系 $T$ 的访问方式.$S\_Map$ 包含当前所有尚未处理的关系及其副本.$C\_Map$ 包含目前已经优化的子查询.$Entry$ 记录了子查询的逻辑表示和最小代价的实现方法.$EntrySize()$表示 $Entry$ 中的子查询包含的关系个数.$S_Map.removeAllCopies(tableName)$函数将所有关系名为 $tableName$ 的所有副本从 $S\_Map$ 删除.$C_Map.removeEntries(size)$将关系个数少于 $size$ 的所有子查询计划删除.算法的第①~⑥步初始化 $S\_Map$,即注册关系的所有副本,以便在后面选择访问代价最低的副本.第⑦~⑭步计算所有具有连接关系的关系副本的连接,并将其添加到 $C\_Map$ 中.第⑮~㉟步自低向上地计算 $k(k\geqslant 3)$个关系连接的最小代价,并通过 $k$ 个关系连接的最小代价导出 $k+1$ 个关系连接的最小代价.

### 4.2 贪心近似算法

效率更高的优化算法是采用贪心策略.贪心近似算法比上面的动态规划算法更粗略.贪心算法的基本思想是首先找出两个关系做连接的最小代价执行方案,然后"贪婪"的选择这样的关系,该关系加入到此执行方案后使新的执行方案在目前所有可行的执行方案中代价最小.

贪心近似算法的形式化描述如算法 GreedyOpt.

算法第①~⑥步将所有关系及其副本加入到 $S\_Map$ 中,第⑦~⑯步选择所有 2 个关系的连接中,连接代价最小的关系.然后第⑰~㉕步每次循环为执行方案扩充一个关系,该关系是目前优化器的最优选择.

## 5 实验结果

实验环境为 Pentium Ⅲ的 PC 机,主频为 1.13GHz,内存为 256MB,硬盘 40GB,转速为 7200 转/秒.光驱为 Toshiba DVD-ROM SD-C230.操作系统为 Windows XP SP2,开发环境为 JDK 1.5,eclipse 3.2M5. 我们在 XXL[6] 程序包(版本为 1.1Beta 1)的基础上实现了第三级代价模型及第三级查询优化方法.使用的实验数据集是面向数据仓库性能的标准测试数据集[7](2.1 版本).实验数据集的总大小为 1GB. 查询采用 TPC-H 标准查询 Q5,Q7,Q8 和 Q9.

为了验证关系冗余存储对查询的效率的影响,为数据集中 LINEITEM 关系添加多个关系副本,加入副本后关系 LINEITEM 的模式为如表 2 所示:

**Table 2 Data Organization Method of LINEITEM(Schema of Relation Copy)**

**表 2 关系 LINEITEM 的组织方式(关系副本的模式)**

| Copy id | Copy Schema |
|---|---|
| 1 | *LINEITEM*(*l_orderkey*, *l_partkey*, *l_suppkey*, *l_linenumber*, *l_quantity*, *l_extendedprice*, *l_discount*, *l_tax*, *l_returnflag*, *l_linestatus*, *l_shipdate*, *l_commitdate*, *l_receiptdate*, *l_shipinstruct*, *l_shipmode*, *l_commit*) |
| 2 | *LINEITEM_1*(*rowid*,*l_orderkey*), *LINEITEM_2*(*rowid*, *l_partkey*), *LINEITEM_3*(*rowid*, *l_suppkey*), *LINEITEM_4*(*rowid*, *l_linenumber*), *LINEITEM_5*(*rowid*, *l_quantity*), *LINEITEM_6*(*rowid*, *l_extendedprice*), *LINEITEM_7*(*rowid*, *l_discount*), *LINEITEM_8*(*rowid*, *l_tax*), *LINEITEM_9*(*rowid*, *l_returnflag*), *LINEITEM_10*(*rowid*, *l_linestatus*), *LINEITEM_11*(*rowid*, *l_shipdate*), *LINEITEM_12*(*rowid*, *l_commitdate*), *LINEITEM_13*(*rowid*, *l_receiptdate*), *LINEITEM_14*(*rowid*, *l_shipinstruct*), *LINEITEM_15*(*rowid*, *l_shipmode*), *LINEITEM_16*(*rowid*, *l_commit*) |
| 3 | *LINEITEM_1*(*l_orderkey*, *l_partkey*, *l_suppkey*, *l_quantity*, *l_extendedprice*, *l_discount*), *LINEITEM_2*(*l_tax*, *l_returnflag*, *l_linestatus*, *l_shipdate*, *l_commitdate*, *l_receiptdate*, *l_shipinstruct*, *l_shipmode*, *l_commit*) |

### 5.1 第三级代价模型的有效性验证

第 1 组实验通过对比关系操作的实际代价与第三级代价模型的估算代价来验证第三级代价模型的有效性.第三级代价模型对磁盘连接的估算误差结果如图 1 所示,第三级代价模型对第三级连接的代价估算误差的结果如图 2 所示.从图 2 中可以看出,第三级代价模型的估算误差大约在 15%,并且随着关系的增大估算误差具有下降趋势.

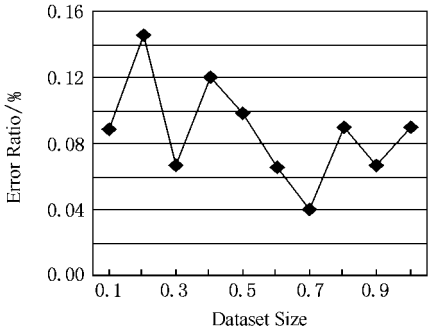


Fig. 1 The error ratio between the estimated value.

图 1 磁盘连接代价估算值与实际值的误差

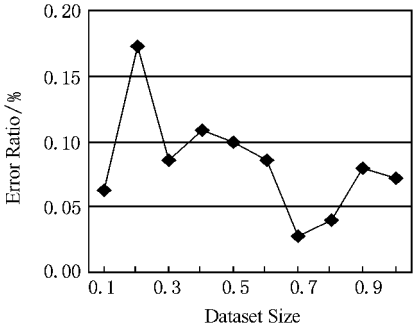


Fig. 2 The error ratio between the estimated and real value of disk join value and real value of tertiary join.

图 2 第三级连接代价估算值与实际值的误差

### 5.2 关系副本对查询效率的影响

第 2 组实验主要对比增加关系副本对查询效率的影响.为了验证增加关系副本对查询效率的影响,还利用 XXL 中的缺省的基于穷举的查询优化算法应用于没有关系副本的数据集上,以便对比查询执行结果.实验结果如图 3 所示.从图中可以看出引入关系副本后查询执行效率明显高于没有关系副本的查询.主要原因是优化算法能够找到满足查询的最小关系副本元素执行查询,从而减少了访问冗余关系属性的 I/O 代价.

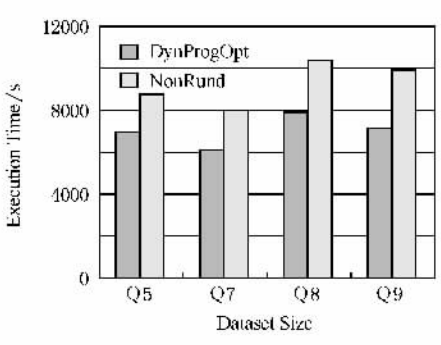


Fig. 3 Influence of relation copies on the query.

图 3 关系副本对光盘存储器上查询执行

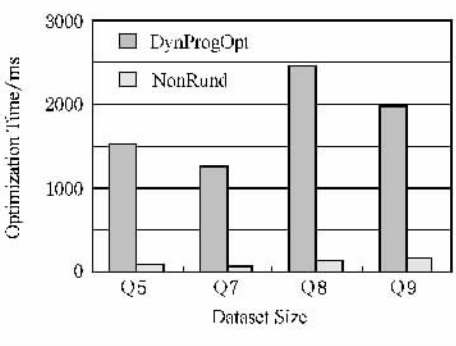


Fig. 4 Comparison of query optimization execution time.

图 4 查询优化算法执行时间比较效率的影响

### 5.3 第三级查询优化算法性能比较

图4列出了基于动态规划方法的查询优化算法与基于贪心策略的查询优化算法的性能比较结果.可以明显看出基于贪心策略的算法性能明显地好于基于动态规划的算法性能.

## 6 总 结

本文提出了适用于处理第三级存储器上关系操作的代价模型及查询优化方法.在仔细分析不同关系操作的数据访问模式基础上,得出若干种基本数据访问模式.数据模式的结合操作符可以将关系操作转变为复合数据访问模式.通过定义基本数据访问模式及数据模式结合操作符的代价,导出关系操作的代价.针对第三级存储器的查询优化方法利用提出的第三级代价模型对查询进行优化.第三级查询优化算法的一个显著特点就是能够从多个关系副本中选择I/O代价最小的一个副本执行查询.实验结果表明,本文提出的第三级数据代价模型可以有效地估算第三级关系操作的代价.本文提出的查询优化方法可以有效地优化针对第三级数据的查询操作.关系副本的引入充分证明了用存储空间换取查询执行时间的策略的可行性.

## 参 考 文 献


[1] Sarawagi S. Database systems for efficient access to tertiary memory [C] //Proc of the 14th IEEE Symp on Mass Storage Systems. Washington: IEEE Computer Society Press, 1995

[2] Yu J, DeWitt D. Query pre-execution and batching in paradise: A two-pronged approach to the efficient processing of queries on tape-resident raster images [C] //Proc of the 9th Int'l Conf on Scientific and Statistical Database Management. Washington: IEEE Computer Society Press, 1997

[3] DeWitt D, Kabra N, Luo J, *et al*. Client-server paradise [C] //Proc of the 20th Int'l Conf on Very Large Data Bases. San Franscisco: Morgan Kaufmann, 1994

[4] Li Jianzhong, Zhang Dongdong, Zhang Yanqiu. Join algorithms based on tertiary storage [J]. Journal of Software, 2003, 14(5): 947-954 (in Chinese)
(李建中,张冬冬,张艳秋. 基于三级存储器的Join算法[J]. 软件学报, 2003, 14(5): 947-954)

[5] Hass L M, Carey M J, Livny M. Seeking the truth about ad hoc join costs [J]. The VLDB Journal, 1997, 6(3): 241-256

[6] Jochen B, Dittrich J, Seeger B. javax.XXL: A prototype for a library of query processing algorithms [C] //Proc of the 2000 ACM SIGMOD Int'l Conf on Management of Data. New York: Association for Computing Machinary, 2000

[7] Transaction Processing Performance Council. DBGEN and QGEN and Reference Data Set [OL]. (2001) [2007-05-08]. http://www.tpc.org/tpch/



**Liu Baoliang**, born in 1976. Ph. D. His main research interests include query processing techniques for massive data.

**刘宝良**,1976年生,博士,主要研究方向为海量数据的查询处理技术.



**Li Jianzhong**, born in 1950. Professor and Ph. D. supervisor. His main research interests include database, parallel computing, *etc*.

**李建中**,1950年生,教授,博士生导师,主要研究方向为数据库、并行计算.



**Gao Hong**, born in 1966. Professor and Ph. D. supervisor. Her main research interests include database, data warehouse system, *etc*.

**高 宏**,1966年生,教授,博士生导师,主要研究方向为数据库、数据仓库.


**Research Background**

The management of DBMS on tertiary storage is becoming more and more important with the development of applications, not only because tertiary devices have been used to archive data, but also the amount of data that application has to deal with is increasing rapidly. Despite the decrease in secondary storage prices, the data storage requirements of these applications cannot be met economically using secondary storage alone. Tertiary storage offers a lower-cost alternative. But till now, commercial database systems are optimized for performance with primary and secondary storage. Tertiary storage, if included in the system, serves only as a backup medium or slow store from which data is first moved onto disks and then the DBMS operates on it as though it were disk resident data. Tertiary devices differ significantly from magnetic disks, in particular, they are not random access devices and media switch times are several orders of magnitude higher. Due to these differences, optimizations that work well for magnetic disks can result in disastrous performance on tertiary storage. So we need to study query optimizations required by tertiary storage system,