第18卷 第4期
2008年4月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 18 No. 4
Apr. 2008

非一致性数据库的多关系聚集查询重写

张华兵,杨路明,谢 东,王佳宜
(中南大学 信息科学与工程学院,湖南 长沙 410083)

摘 要:基于非一致性关系数据库的选择连接查询技术,提出了基于非一致性数据库多个关系上的聚集查询重写方法。该聚集查询重写方法先通过查询出多关系上的一致性结果,然后进行分组聚集,返回聚集表达范围边界值。实验采用TPC-H决策支持基准进行性能研究,结果表明重写查询比初始查询的执行时间要长,但还是可以接受的,因此该方法是有效的。

关键词:关系数据库;非一致性数据库;聚集查询重写
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2008)04-0038-04

Multi-Relations Aggregation Query Rewritings in Inconsistent Databases

ZHANG Hua-bing, YANG Lu-ming, XIE Dong, WANG Jia-yi
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: Based on the technique of selective join query on inconsistent databases, proposes an approach, which can overcome the obstacles in the field of multi-relations aggregation query rewritings in inconsistent database. Can search the consistent results on multi-relations and cluster the results by means of grouping aggregation to return the extremum in the range of aggregation values. The experiment use TPC-H to study the capability. The result demonstrates that the rewriting query have a longer carrying-out time than the original query, but it's acceptable, so the method is operative.

Key words: relational database; inconsistent database; aggregation query rewriting

0 引 言

数据库一直通过数据完整性约束技术(integrity constraint, IC)来控制数据的完整性,数据库设计一直聚焦在发展一系列的约束关系来保证每一个数据库是合法和一致的。然而,在很多情况下不能用完整性约束强行控制数据的完整性,特别是在数据库更新频繁的情况下,检测约束的一致性的代价是非常昂贵的。因此,在特殊情况下,数据库往往也需要具有非一致性的特点。

从近年来的研究情况来看,IDB(inconsistent database)[1]已经成为研究热点,IDB和CQA(consistent query answer)[1]的概念已经被广泛接受。文献[2]第一个提出CQA的聚集范围的概念,但只考虑了聚集属性而没有考虑分组。Hippo系统[3]是CQA应用在大型数据库上的系统,采用的方法是将产生的元组冲突图放到内存中,产生一个Java程序的过程化方法来计算一致性结果。其缺点是限制了冲突图的数量,如果存在海量的冲突元组,则Hippo系统不能计算。其它CQA重写查询系统如Infomix系统[4]采用逻辑编程实现一致性结果的查询,计算代价较大。ConQuer系统[5,6]采用一阶查询重写方法对大量易处理的查询进行重写,有效地利用现有DBMS的功能,不要求程序的预处理和重加工,提高了查询效率,但它没有考虑SPJ聚集查询。

文中所要做的主要工作是在已有的选择连接(SPJ)查询的基础上,提出了基于非一致性数据库多个关系上的聚集查询重写方法。聚集查询重写方法先通过查询出多关系上的一致性结果,然后进行分组聚集。并根据CQA的聚集范围的概念,实现返回多个关系的聚集表达范围边界值。

1 相关概念

设R是一个关系,D是R上的一个数据库,给出一

收稿日期:2007-07-14
基金项目:湖南省教育科研基金(05C671);中南大学"大学生创新创业启航行动"重点资助创新项目(ZB018)
作者简介:张华兵(1977-),男,湖北武汉人,硕士研究生,研究方向为数据库和数据管理;杨路明,教授,研究方向为数据库信息系统。

组查询约束 $\sum$,数据库 D 在 $\sum$ 可以是非一致性的。D 的修改[1] D^R 是 R 上的一个例子,D^R 满足 $\sum$ 查询约束,D^R 最低程度地不同于 D。

定义 1:非一致性数据库(inconsistent database, IDB)[1]。设 I 是数据库 D 的任意一个实例,对于 D 上的任意 IC,如果任意 $I \in IC$,则 D 是一致性,否则 D 是非一致性的数据库。

定义 2:区间(interval)。给定一组完整性约束 $\sum$ 和数据库实例 I,如果聚集查询 q 在每个候选数据库上返回到值 $v[a,b]$,则 q 在 I 上的一致性结果是区间 $[a,b](-\infty < a \leqslant b < \infty)$,表示为 $I \models \sum q[a,b]$。如果 $[a,b]$ 是一致性结果,则 a 为下界,b 为上界。如果没有子区间是一致性结果,则 $[a,b]$ 是最优的一致性结果,a 为最大下界,b 为最小上界。

2　聚集查询重写

现在先用一个例子简单说明 ConQuer 系统[5,6]采用的一致性查询的语义和重写策略。

例 1:图 1 中的表 order1、customer 均为非一致性数据中的两个关系。表 order1 存储了定单的相关信息,orderkey 属性为定单号,custfk 属性为客户号,是关系 order1 的外键。表 customer 存储了客户的货款的相关信息,custkey 属性为客户号,acctbal 属性为客户帐户余额。

表 order1

	orderkey	clerk	custfk
s1	01	ali	C1
s2	02	jo	C2
s3	02	ali	C3
s4	03	ali	C4
s5	03	pat	C2
s6	04	ali	C2
s7	04	ali	C3
s8	05	ali	C2

表 customer

	custkey	acctbal
t1	C1	2000
t2	C1	100
t3	C2	2500
t4	C3	2200
t5	C3	2600

图 1　一个非一致性关系

现在要求查询出客户余额大于 1000 的定单,根据 RewriteJoin(q, $\sum$)[5]算法,很容易得到所要求的一致性结果为{02,04,05}。

RewriteJoin(q, $\sum$)算法是多个非一致性关系的连接查询算法,多个非一致性关系的聚集查询将在此基础上推论。现以例 1 中的两个关系为例,逐步进行分析。对图 1 中的两个关系,考虑查询某个操作员所处理的所有帐户余额大于 1000 的定单的总和,给出初始查询。

```
q1: select o.orderkey,sum(acctbal)
    from order1 o,customer c where o.custfk = c.custkey and c.acctbal>1000 group by o.orderkey
```

执行这个查询,得到结果{(01,2000),(02,7300),(03,2500),(04,7300),(05,2500)}。这个结果不是所要求的一致性结果。原因如下:一是客户 C1 存在小于 1000 的帐户,所以定单 01 不应该是一致性查询结果;二是客户 C4 的帐户余额不能确定,这样定单 03 也不应该是一致性查询结果。所以这个结果不应该是题意所要求的。

2.1　连接查询

要得到题意所要求的聚集查询结果,首先必须在非一致性数据库上查询到符合条件的一致性结果。用 RewriteJoin(q, $\sum$)算法很容易得到一致性查询结果为{02,04,05}。在此基础上,再考虑查询那些处理余额大于 1000 的客户的定单的职员。这里给出考虑 order1.clerk 属性的查询 q2。

```
q2:with Candidates1 as (
    select distinct o.orderkey, o.clerk
    from customer c, order1 o where c.acctbal>1000 and o.custfk = c.custkey),
  Filter1 as (
    select o.orderkey
    from Candidates1 Cand join order1 o on Cand.orderkey = o.orderkey left outer join customer c on o.custfk=c.custkey where c.custkey is null or c.acctbal<=1000union all
    select orderkey from Candidates1 Cand group by orderkey having count(*) > 1)
  select clerk
  from Candidates1 Cand where not exists (select * from Filter1 F where Cand.orderkey = F.orderkey)
```

在这个查询中子查询 Candidates1 的查询结果为{(01,ali),(02,jo),(02,ali),(03,pat),(04,ali),(05,ali)},子查询 Filter1 不但将定单 01 和 03 的元组过滤掉,还将定单 02 的元组也过滤掉,因为处理定单 02 的职员可能是 jo 和 ali,所以包含定单 02 的元组不是 q1 的一致性查询结果。查询 q2 的最终结果为{ali,ali}。结合查询 q1 的结果分析,既符合帐户余额大于 1000 的,又由同一个职员处理的定单,只有定单 04 和 05。

现在要求在图 1 的两个关系表中,查询出帐户余额大于 1000 且由同一个职员处理的定单,并要求在查询结果中查出属性 orderkey、clerk、acctbal 的相关值。综合分析查询 q1 和 q2,在先不考虑聚集的情况下可以给出查询。

```
q3:with Candidates2 as (
    select o.orderkey,o.clerk,acctbal
    from customer c, order1 o where c.acctbal>1000 and o.custfk = c.custkey),
  Filter2 as (
    select o.orderkey
```

```
from Candidates2 Cand join order1 o on Cand.orderkey = o.orderkey left outer join customer c on o.custfk=c.custkey where c.custkey is null or c.acctbal<=1000union all
    select c1.orderkey
    from Candidates2 c1, Candidates2 c2 where c1.orderkey=c2.orderkey and c1.clerk<>c2.clerk
    group by c1.orderkey,c1.clerk having count(*) > 1)
  select orderkey,clerk,acctbal
  from Candidates2 Ca where not exists (select * from Filter2 F where Ca.orderkey = F.orderkey)
```

查询 q3 结果为{(04, ali, 2500), (04, ali, 2200), (04,ali,2600), (05,ali,2500)}。

2.2 聚集查询

在上一节中查询 q3 的查询结果上考虑聚集查询就相对容易了。文献[2]建议对聚集表达返回范围(边界值),而不是准确的数值。那么根据查询 q3 的结果,定单 04 对职员 ali 的的总帐户余额介于 2200 和 2600 之间。

现在讨论在图 1 的两个关系上通过查询重写,来得到定单 04 和 05 的帐户余额之和,对查询 q3 结果的 acctbal 属性做聚集查询。

```
q4: select orderkey,clerk,min(acctbal),max(acctbal)
    from Candidates2 Ca where not exists (select * from Filter2 F where Ca.orderkey = F.orderkey) group by orderkey,clerk
```

该查询的结果为:{(04,ali,2200,2600),(05,ali,2500,2500)},对定单 04 来说帐户 C3 对帐户余额之和贡献了上边界值 2200 和下边界值 2600;对定单 05 来说帐户 C2 对帐户余额之和贡献的边界值始终为 2500。所以根据前面的推理分析,给出有聚集的树查询重写算法(见图 2):

```
RewriteAgg(q,Σ)
给出查询 q 的形式:
Select Kroot, S, min(N),max(N)
from Rroot,R1, … ,Rm where KJ and NKJ and SC
group by Kroot, S
其中: Rroot 是连接图根结点的关系
KJ 是键与键的连接谓词
NKJ 是非键与键的连接谓词
SC 是选择的连接谓词
设: root 是 Rroot 的键属性
K1, … ,Km 是 R1, … ,Rm 的键属性
NSC 是 SC 的否定选择连接谓词
NSC 是 SC 的否定选择连接谓词
LOJ[5]是 q 的连接图的左连接
q 的查询重写 Qc 如下:
with Candidates as (
select Kroot, S from Rroot, R1, … ,Rm where KJ and NKJ and SC),
    Filter as (
    Select Kroot
    from Candidates C join Rroot on C.Kroot = Rroot.Kroot left outer join LOJ
    where KJ and (R1.K1 is null or … or Rm.Km is null or NSC)
    union all
    select Kroot
    from Candidates C1,Candidates C2 where KJ and NKJ
    group by Kroot,S having count(*)>1)
    select Kroot,S,min(N),max(N)
    from Candidates C where not exists (select * from Filter F where C.Kroot = F.Kroot)
    group by Kroot,S
```

图 2 有聚集的树查询重写算法

该算法的子查询 Candidates 与初始查询不同之处主要在于对 select 语句的改变,这种改变使得子查询 Candidates 能计算出一组候选集。

在至少一个修改[1]中,Filter 的第一个子查询返回的候选集不在 CQA 中。要确保子查询 Filter 考虑所有的候选集,必须在 Candidates 与连接图[5]根结点的关系(Rroot)之间有一个内部连接,关系 Rroot 与初始查询的其它关系是连接的(表达为 LOJ)。由于希望检测到键与非键连接不满足的情况,需要执行一个左连接而不是内连接。

在 Filter 第一个子查询的 where 句子中,首先它有一个键—键连接查询的表达式 KJ。这些连接不需要在左连接 LOJ 中考虑。如果一个候选满足键—键连接,那么它一定在每个修改中也满足它;其次,子查询先检查是否有空值在左连接中,以此来检查是否有元组不满足初始查询的连接。最后,它检查是否有一些初始查询的选择条件没有被满足(表达为 NSC)。

3 实验分析

评价重写方法的实验环境为 Intel Celeron 2.53GHz, 内存 512MB, Windows XP professional, SQL Server2005。查询重写采用 JAVA 实现,实验采用 TPC-H 规范的查询中的第 3 和第 12 查询,TPC-H 查询建议的标准设置。实验采用了一种数据生成工具 UIS 生成不同尺寸的非一致性数据,并考虑如下的参数:

数据尺寸 s。实验考虑 s=0.1GB、0.5GB、1GB、2GB。1GB 数据库约有 8 百万元组。

在数据库的非一致性部分,违反了键约束的 n 个元组共享一个共同的键值。如 $n=2$ 则每个键值在数据库的非一致性部分出现 2 个元组,实验的 n 值取 2。

表 1 给出在 FROM 语句中的关系数量、选择性(选择性高说明更多的元组满足查询)在 SELECT 语

句中属性的数量、聚集表达式中属性的数量。

表1　查询特点

	关系数量	投影属性数量	聚集属性数量
Q3	3	4	1
Q12	2	3	2

现在根据图3中的数据计算重写查询的负载,采用(Tr－To)/To(Tr表示重写查询时间,To表示初始查询时间)公式来计算,对于Q3重写查询负载是0.73,对于Q12重写查询负载是0.64。从图3上也可以看出重写查询比初始查询执行时间要长,这主要是因为在重写查询中多了Filter子查询,Filter中有较耗时间的2个连接查询。另外,因Q3语句中的关系数量比Q12要多,所以Q3的执行时间比Q12要长。虽然重写查询比初始查询执行时间要长,但是还在可接受的范围内,因此该查询重写方法是可行的。

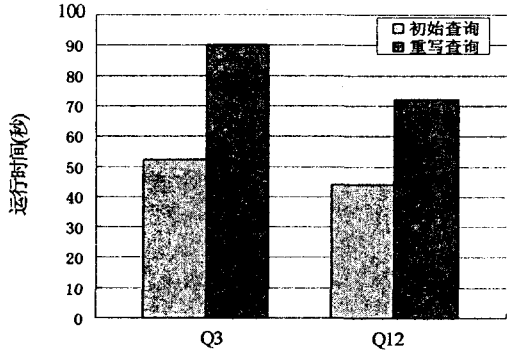

图3　查询Q3和Q12在1G数据库上的执行时间

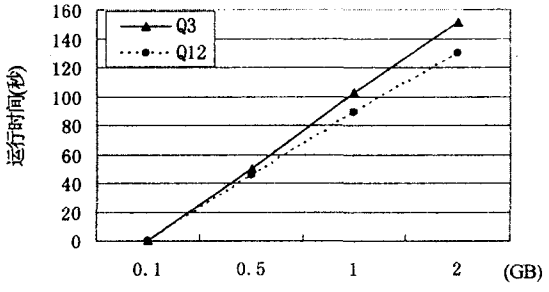

图4　Q3和Q12在不同大小的数据库上的执行时间

图4中所显示的是Q3和Q12在不同大小数据库(分别采用0.1GB、0.5GB、1GB、2GB的数据库)上运行的时间。重写查询Q3和Q12的运行时间随数据库的大小变化而变化,另外,因Q3中存在对几个不同属性进行分组操作,而Q12实际只对同一个属性进行分组操作,所以重写查询Q3执行时间比Q12要长。

4　结　语

基于非一致性关系数据库的选择连接(SPJ)查询技术已日趋成熟。文中在此基础上进一步扩展了SPJ的聚集查询重写策略,提出了基于非一致性数据库多个关系上的聚集查询重写方法RewriteAgg(q,Σ)。该方法先通过查询出多关系上的一致性结果,然后进行分组聚集,最终实现返回聚集表达范围边界值。RewriteAgg(q,Σ)方法成功实现了在具有复杂关系的非一致性数据库中进行SPJ聚集查询的目的。实验采用TPC－H决策支持基准进行性能研究,实验结果表明重写查询比初始查询的执行时间要长,但还在可以接受的范围之内,因此该方法是有效的。下一步的工作需要提高聚集重写查询执行性能,进一步扩展聚集查询的适用范围。

参考文献:

[1] Arenas M, Bertossi L, Chomicki J. Consistent Query Answers in Inconsistent Databases[C]// In PODS. Philadelphia, USA:[s.n.],1999: 68－79.

[2] Arenas M, Bertossi L, Chomicki J. Scalar Aggregation in FD－Inconsistent Databases[C]// In ICDT. London, UK:[s.n.],2001:39－53.

[3] Chomicki J, Marcinkowski J, Staworko S. Hippo: A System for Computing Consistent Answers to a Class of SQL Queries [C]// In EDBT. Heraklion, Crete:[s.n.],2004:841－844.

[4] Eiter T, Fink M, Greco G, et al. Efficient Evaluation of Logic Programs for Querying Data Integration Systems[C] // In I-CLP. Mumbai, India:[s.n.], 2003:163－177.

[5] Fuxman A, Fazli E, Miller R J. ConQuer: Efficient Management of Inconsistent Databases[C] // In ACM SIGMOD. Baltimore, Maryland:[s.n.],2005: 155－166.

[6] Fuxman A, Miller R J. First－Order Query Rewriting for Inconsistent Databases[C] // In ICDT. Edinburgh, UK:[s.n.],2005: 337－351.

(上接第39页)

hancement of echocardiograms via multiscale nonlinear processing[J]. IEEE Trans Medical Imaging, 1998, 17(4): 532－540.

[7] Mallat S, Hwang W L. Singularity detection and processing with wavelets[J]. IEEE Trans Information Theory, 1992, 38(2):617－643.

[8] Laine A F, Schuler S. Mammographic feature enhancement by multiscale analysis[J]. IEEE Trans Medical Imaging, 1994, 13(4):725－740.