第 33 卷　第 22 期　　计　算　机　工　程　　2007 年 11 月
Vol.33　No.22　　Computer Engineering　　November 2007
·软件技术与数据库·　文章编号：1000—3428(2007)22—0066—02　文献标识码：A　中图分类号：TP393

非一致性数据库的概率查询重写

谢　东，杨路明，蒲保兴，刘　波
(中南大学信息科学与工程学院，长沙 410083)

摘　要：结合概率数据库技术，以元组匹配所产生的聚类为基础，提出了一种新的基于聚类的非一致性数据的概率方法。基于可信聚类，给出了基本的查询重写技术，在有聚集的查询中，考虑了合适的元组概率、区间值、期望值。在不进行程序预处理的情况下，“重写”能被商业数据库系统有效地优化和执行，采用不一致性数据的区分度和数据库大小去理解其适应性，并使用了 TPC-H 基准的数据和查询。实验显示了该方法的有效性。

关键词：关系数据库；非一致性数据库；查询重写；聚类概率

Probabilistic Query Rewriting in Inconsistent Databases

XIE Dong, YANG Lu-ming, PU Bao-xing, LIU Bo
(School of Information Science and Engineering, Central South University, Changsha 410083)

【Abstract】 This paper combines with original probabilistic databases based on the clusters which is produced by tuple matching techniques, and proposes a new probabilistic approach based on clustering in inconsistent databases. It analyzes a basic technique for queries rewriting based on believable cluster, considers the appropriate tuple probabilities, interval values and expectation values. The approach needs not procedural pre-processing, so the rewritten queries can be efficiently optimized and executed by commercial database systems. In order to understand the flexibility of the approach for considering distinguishing degrees of inconsistency and database sizes, the experiments use the data and queries of the TPC-H specification, and results show that the approach is efficient.

【Key words】 relation database; inconsistent database; query rewriting; cluster probability

1　概述

数据库完整性约束(integrity constraint, IC)作为数据库模式的一部分，有效地保证了数据的完整性和有效性，使数据符合现实世界的实体规则。然而，现实世界的一个实体(entity)在数据库中常常对应多个不一致的数据。对于给定的约束，数据库可能是非一致性数据库(inconsistent database, IDB)[1]。如一个SCM系统从多个数据源集成数据，会产生冲突的信息，如相同的供应商有不同的名字等。数据清洗能识别和纠正错误，使数据库恢复到一致性状态。但不会为了保存数据的一致性而修改数据，导致有用的非一致性数据丢失。

表 1 显示了关系*buyer*存储客户的账户余额。其中，*buyid*是关系主键；*cluster*属性表示聚类(同一聚类表示同一客户)；*pr*属性表示元组概率；*name*和*acctbal*属性分别表示客户姓名和余额。在关系*buyer*中，聚类bc_1和bc_2内分别有 2 个元组，聚类内元组的概率和为 1。给定查询账户余额小于 60 的客户。

表 1　非一致性 *buyer* 关系

	cluster	*buyid*	*province*	*acctbal*	*pr*
s_1	bc_1	b_1	p_1	50	0.9
s_2	bc_1	b_2	p_2	10	0.1
s_3	bc_2	b_1	p_2	60	0.6
s_4	bc_2	b_2	p_2	5	0.4
s_5	bc_3	b_2	p_2	1	1

Q_1为select *cluster*,*pr* from *buyer* where *acctbal*<60。Q_1得到$\{(bc_1,0.9),(bc_1,0.1),(bc_2,0.4),(bc_3,1)\}$，聚类$bc_1$和$bc_2$都是可能的结果，$bc_3$是必然的结果。如果$bc_1$出现了 2 次，则不是一致性结果。可以对聚类分组求解元组概率，并得到一致性结果，即

$\{(bc_1,1),(bc_2,0.4),(bc_3,1)\}$

聚类方法把数据集合分成若干类，使每个类内部的数据尽量相似，而属于不同类的数据也尽量不同[2]。概率数据模型扩展了关系数据模型，在数据库的每个元组中引入概率标示属性来表示该元组的不确定性[3]。IDB[4~6]扩展了早期的研究[1]。一阶查询能有效地转换成SQL，并考虑了聚集查询重写[5]。本文工作最初的动机是来自文献[5~6]，文献[5]考虑了概率数据库查询重写。文献[6]分析了聚类内多个潜在的非一致性元组的概率，发展了一种候选数据库和聚类代表元组的概念，对聚类概率进行计算和评价，没有考虑查询重写。

本文结合原有的概率数据库技术，以元组匹配所产生的聚类为基础，提出了一种新的基于可信聚类的非一致性数据的元组概率方法，并给出了一种基于可信聚类的基本查询重写技术。

2　基本概念

定义 1 非一致性数据库[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$\forall I \models IC$，则D是一致性，否则D是非一致性数据库。

定义 2 聚类(cluster)[6]　把关系R分成k个元组子集$C_i(1\leq i\leq k)$，C_i包括多个概率为pr_{ij}的元组t_{ij}，满足如下条件，

基金项目：湖南省教育厅科研基金资助项目(05C671)；中南大学重点创新基金资助项目(ZB018)

作者简介：谢　东(1971－)，男，博士研究生，主研方向：数据库信息系统和数据管理；杨路明，教授；蒲保兴，副教授；刘　波，博士研究生

收稿日期：2006-12-25　**E-mail：**lgxzy@163.com

称为聚类。聚类内元组的个数称为聚类基数$|C_i|$。

$C_1 \cup C_2 \cup \cdots \cup C_k = R$

$C_i \cap C_t = \varnothing \wedge i \neq t \wedge 1 \le i,t \le k$

$t_{ij} \in C_i \wedge pr_{i1} + \cdots + pr_{ij} = 1 \wedge 1 \le i \le k \wedge 1 \le j \le |C_i|$

定义 3 可信聚类(believable cluster, BC) C_i是关系R上的一个基数大于 1 的聚类，t_i是聚类C_i内的任意一个元组，Q是R上的一个查询。如果$t_i \notin Q(R) \wedge \exists\, t_j \in Q(R) \wedge i \neq j$，那么聚类$C_i$是可信的，称为非完全可信聚类；如果$\forall\, t_i \in Q(R)$，则称为完全可信聚类；如果$t_i \notin Q(R)$，则称为不可信聚类。非完全和完全可信聚类统称为可信聚类。

本文方法不存在不可信聚类。引入概率标记 *pr* (probability)属性表示该元组的不确定性，范围为(0,1]。

定义 4 可信聚类概率(probability of believable cluster, POBC) 设关系R是非一致的，t是可信聚类c_i内的元组，那么R上查询Q的聚类概率为

$pr = \sum t|_t \in {}_{ci}\, pr(t)$

如果聚类是完全可信聚类，则概率为 1。

3 查询重写

在可信聚类等概念基础上讨论无连接的查询，通过例子说明重写策略和方法。

Q_1可以得到$\{(bc_1,0.9),(bc_1,0.1),(bc_2,0.4),(bc_3,1)\}$。其中，$bc_1$是完全可信聚类(因为该聚类内的 2 个元组全部在结果集中)；bc_2是非完全可信聚类；bc_3也是完全可信聚类。然而，得到聚类bc_1的 2 个结果集并不能表示bc_1的概率。Q_2进行了聚类分组概率计算，得到可信聚类概率一致性结果，即$\{(bc_1,1),(bc_2,0.4),(bc_3,1)\}$。其中，$bc_1$其可信聚类概率为 1；$bc_2$可信聚类概率为 0.6；$bc_3$其可信聚类概率为 1。

算法 1 给出了得到无聚集可信聚类概率的查询重写。与初始查询不同的是，只要对聚类分组就可得到可信聚类概率。

算法 1

输入 查询 Q:select *cluster* from R where W。

输出 Q 的查询重写 Q'，即

Q':select *cluster*,sum(*pr*) as *pr* from R where W group by *cluster*

其中，*pr* 是关系中聚类内元组的概率；*cluster* 是关系的聚类标记。

考虑有无连接的聚集查询重写，在表 1 中，关系*buyer*中返回客户各省的余额合计。Q_2:select *province*,sum(*acctbal*) from *buyer* group by *province*。Q_2得到$\{(p_1,50),(p_2,80)\}$。由于关系*buyer*中的元组s_1可能不会出现，因此p_1的最小值是 0，最大值是 50。聚类内元组事件是“排他”的，s_3和s_4在同一聚类内不能同时出现，而元组s_5是确定元组，必须出现，p_2的最小值是 6，最大值是 71。笔者采用一种基于数学期望的聚集查询方法，如Q_2得到的期望值为$\{(p_1,45),(p_2,40)\}$。

定义 5 *ROBC* (range of believable cluster)。设R是一个关系，C_i是查询Q上的可信聚类，A是可聚集的属性，则C_i在A上的范围值*ROBC*为：(1)如果C_i为完全可信聚类，$ROBC = \min(A) = \max(A)$；(2)如果C_i为非完全可信聚类，$0 \le ROBC \le \max(A)$；(3) 最小值$\min(A) = \min(A(t)), t \in C_i$；(4) 最大值$\max(A) = \max(A(t)), t \in C_i$；(5) 期望值 $expectvalue(A) = \prod_{t \in ci} A(t)$。

在有聚集的查询重写算法 2 中，最小值由于存在必然事件和可能事件，因此必须要把可能事件排除掉，得到必然事件的属性数值，以求得最大值和期望值。

算法 2

输入 一个分组查询 Q 为 select G, $AGG(e)$ as E from R where SC group by G。其中，G 为分组属性集合；$AGG(e)$为聚集操作符集合；SC 为合取选择谓词集合。

输出 (*cluster* 为关系的聚类标记)

```
with candidate as(
select G,cluster,min(e) as minb,sum(pr) as pr,max(e) as maxb,
sum(pr*e) as exceptvalue
from R group by G,cluster),
min_candidate as(
select G,min(minb) as minb
from candidate group by G having sum(pr)=1 or count(*)>1),
max_candidate as(
select G,sum(maxb) as maxb,sum(exceptvalue) as exceptvalue
from candidate group by G)
select m2.G,(case when minb is null then 0 else minb end) as minb,maxb, exceptvalue
from max_candidate m2 left outer join min_candidate m1 on m2.G=m1.G
```

其中，*cluster* 为关系的聚类标记。

4 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，SQLServer2005。查询重写采用Java实现，实验采用TPC-H规范[7]的第 1 查询和第 6 查询，去掉了avg和count 2 种聚集函数。使用TPC-H查询建议的标准设置，采用了数据生成工具UIS[8]生成不同尺寸的非一致性数据，并考虑数据库尺寸*sf*。实验考虑*sf*=0.1GB, 0.5GB,1GB。1GB数据库约有 800 万元组。聚类内元组数量为*n*。UIS工具产生大量的聚类，每个聚类有相同数量的元组。实验考虑*n*=1,2,5,25 的情况。*n*=1 是一致性数据库的情况，用于参照基数。

图 1 中Q_1和Q_6的执行参数是*sf*=1, *n*=3。重写查询Q_1和Q_6的执行时间与初始查询的执行时间相差较大。原因是Q_1有 6 个投影属性和 4 个聚集表达式，而Q_6有 2 个投影属性和 1 个聚集表达式。初始查询的order by和重写查询的order by 和group by的执行时间变得相当短。因此，重写查询的负载由投影属性和聚集表达式决定。

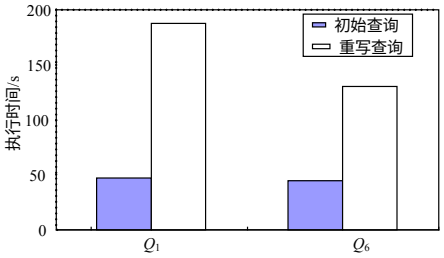

图 1 Q_1和Q_6的执行时间

图 2 在不同数据库尺寸下对方法进行了测试，显示了 *n*＝3,*sf*＝0.1,0.5,1,1.5,2 且有 order by 语句的重写查询的运行时间，运行时间随着数据库尺寸的增大呈线性递增。

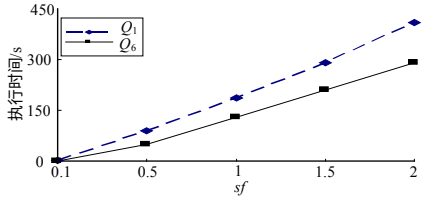

图 2 执行时间(*n*=3) (下转第 88 页)