非确定性数据查询①

Uncertain Data Queries

朱素英　谢　东　(湖南人文科技学院 计算机科学技术系 湖南 娄底 417000)

摘　要： 数据库技术中尽管有完整性约束技术用来维护数据的确定性，但有些情况可能不满足完整性约束。本文提出了非确定性数据的确定性查询概念，给出的查询重写方法能有效地进行数据库 SQL 查询语句，它可以违反一系列的约束条件，重写查询去查找相应的与约束一致的数据。

关键词： 关系数据库 非确定性数据 查询重写

1　引言

数据库完整性约束(integrity constraint,IC)作为数据库模式的一部分，给出了数据及其联系应具有的制约和依赖规则，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。目前在完整性约束的研究方面，主要是发展一系列的约束关系来尽可能保证每一个数据库是合法和一致的。然而，现实中的数据是错综复杂的，不可避免地存在非确定数据(uncertain data)等情况[1]。一个普遍的现象是：现实世界的一个实体(entity)在数据库中常常是对应多个非确定的数据。例如，当集成分布式数据源时，不同的数据源在集成前满足各自本地数据库的完整性约束，但集成后却可能包含冲突的元组。

本文采用一种对于包含有错误和例外的数据库的合适的方法，用户可以在查询时假定一组完整性约束，返回所有符合约束的确定性结果，重写这个查询到另外一个能返回确定性结果的 SQL 查询，给出了能返回合适的确定性结果的重写 SQL 查询方法。

2　非确定性数据

对于给定的约束，数据库可能是非确定性的，主要有如下原因[1]：

(1) 完整性约束不是 SQL 标准的一部分，不能被 DBMS 维护；

(2)当数据从多个满足约束(如键约束)的数据源被集成时，可能就不满足约束(如在每个数据源有相同的键值)；

(3)新的完整性约束加入先前已经存在的遗留数据库后，遗留数据库的数据可能不满足新的完整性约束；

(4)用户希望在查询时“软”约束能被满足，但不需要在更新数据时约束失效；

(5)数据库的确定性约束机制没有被建立。

例如，在一个 SCM 系统从多个数据源集成数据，一个供应商 S1 有 2 个帐户余额：10K 和 50K，产生了冲突的信息。给定一个查询，要求返回余额大于 20K 的供应商。在普通的标准查询中，S1 将出现在返回结果中。然而，在直观上并不知道供应商 S1 的帐户余额大于 20K 的确定性有多大，因为 S1 的帐户余额可能是 10K。对于现实世界的实体，数据库应该只有一个唯一的元组对应，且每个属性值也是确定的。

非确定性管理在数据集成中是非常基础的[2,3]，但没有得到更多的注意。在一个集成系统中，每个单独数据源的数据库实例是满足完整性约束的，然而在全局模式却与源数据非确定，因此在全局模式下需要放松完整性约束或者删除完整性约束。

非确定性数据库的一个策略是数据清洗[4]，它识别和纠正数据中的错误，恢复数据库到确定性状态。但数据清洗技术是半自动的，需要人为干预，且处理代价昂贵。非确定性数据往往是有用的，不可能为了保存数据的确定性而修改数据，导致有用的数据丢失。对于一些任务，一个用户希望采用不同的清洗策略，

① 基金项目:湖南省自然科学基金(07JJ6113,07JJ3119);湖南省教育厅科研基金(08B040);湖南省重点建设学科
收稿时间:2008-12-08

或者希望保留所有的数据，甚至非确定性数据。并且，大多数数据在数据库中仍然是一致的,因而完整性约束不能强行控制集成系统的数据完整性。

例 1：表 1 的包含客户及其余额的关系表 customer(custkey，acctbal),假定键是 custkey，注意元组违反了键约束(可能数据从不同数据源被集成)。考虑查询返回关于余额大于 1000 的客户。

Q1: select custkey from customer where acctbal>1000

表 1　非确定性关系表 customer

	Custkey	acctbal
T1	C1	2000
T1	C1	100
T3	C2	2500
T4	C3	2200
T5	C3	2500

Q1 得到{c1,c2,c3,c3}，这不能被考虑为一致的。因为 C1 可能有余额低于 1000(元组 t2)，然而 c1 被包括满足 Q1 的元组 t1 中；c3 出现在结果中 2 次，由于 custkey 为关系的键，不能在结果中有重复值。

因此认为对于 Q1，{c2,c3} 是确定性结果。理由是 c2 是单一元组,满足 Q1 和不违反键约束。即使 c3 出现了 2 次(违反键约束)，但 2 个元组满足 Q1。

重写了 Q1 去满足确定性结果：

```
Q2:select distinct custkey from customer c
where acctbal > 1000
    and not exists (select * from customer c'
        where c'.custkey=c.custkey
            and c'.acctbal <=1000)
```

重写查询 Q2 使用了 distinct 关键字保证返回正确的数量(c3 将只出现一次)，有一个嵌套子查询，目的是过滤掉那些满足 q1 但其它元组不满足的元组(本例过滤 c1，因为 c1 在 t2 的余额小于 1000)。

3　确定性查询

现有的数据库技术都是基于确定性数据库的，不支持在 IDB 上得到不包含非确定性数据的“净化”查询结果。非确定性被看作是一种逻辑现象，如果数据库违背完整性约束，那么它是非确定性的，冲突元组不能有效地表达数据非确定性的语义。

由于现实世界是一致的，非确定性数据不能对应现实世界的状态，因此不是确定的信息，需要在查询前得到纠正。然而，修改数据库会产生多个可能正确的数据库，因而需要考虑在这些数据库中如何获得可靠信息。因此，替代数据清洗的一种方法是保持源数据不改变，而在查询时解决非确定性，识别确定性数据。在非确定性数据环境下执行确定性查询，即使不知道正确的确定性数据，也能让查询结果包含确定的元组，用户查询能得到符合完整性约束和查询条件的确定性结果，称为确定性查询。

基于候选的概念，使用确定性查询结果的定义。候选是非确定性数据库的子集，正确地包括每个键值的一个元组，在候选的概念下是一致的。缺乏用户的输入和违反约束的元组是矛盾的，系统提高保留所有元组的选择，而不是强迫删除或者忽略这些非确定性元组。

候选说明数据库可能已经被清洗，能被用户去理解数据是潜在的非确定。如例 1 原始查询和已重写查询结果的不同，能检测到客户 c1 满足查询但不是确定性数据。说明这个数据将需要被清洗。

定义 1.设 I 是数据库 D 的一个实例，对于 D 上的一组完整性约束 Σ，如果 I 满足 Σ，则表示为 $I \vdash \Sigma$，否则表示为 $I \nvdash \Sigma$。

定义 2.确定性数据库(consistent database)。设 I 是数据库 D 的任意一个实例，对于 D 上的一组完整性约束 Σ，如果 $\forall I \vdash \Sigma$，则 D 是确定性数据库。

定义 3.非确定性数据库。设 I 是数据库 D 的一个实例，对于 D 上的一组完整性约束 Σ，如果 $\forall I \nvdash \Sigma$，则 D 是非确定性数据库。

定义 4.对称差(distance)。给定数据库 D 的实例 I 和 I’，则两个数据库实例的对称差 $\triangle(I,I') = (I - I') \cup (I' - I)$。

定义 5.候选数据库(candidate database,CDB)。给定数据库 D 的一组完整性约束 Σ 和数据库实例 I。如果数据库实例 $I' \vdash \Sigma$，则 I’ 是 I 关于 Σ 的一个候选数据库，且不存在 $I^* \vdash \Sigma$ 且 $\triangle(I,I') \supset \triangle(I,I^*)$。

I’是模式中满足完整性约束的 I 的子集，在集合包含与 I 有最小的区别。本文把 $Rep(I,\Sigma)$ 表示为 I 关于 Σ 的候选数据库。可以注意到，CDB 可能不是唯一的。

对于查询 Q 来说，如果每个 $D' \in Rep(D,IC)$：$D' \vdash Q(t')$，那么元组 $t'=(t_1,\cdots,t_n)$ 是确定性结果。即当 $x_1,\cdots,x_n$ 分别取得值 $t_1,\cdots,t_n$。如果 $n=0$，则查询是布尔查询，对于每个 $D' \in Rep(D,IC)$，如果 $D' \vdash Q(t')$，

那么查询为真，否则为假。

例 2：在图 1 中，有如下的候选：D_1^R ={t1,t3,t4}，D_2^R ={t1,t3,t5}，D_3^R ={t2,t3,t4}，D_4^R ={t2,t3,t5}。每个候选是一个确定性数据库，尽可能与图 1 的非确定数据库接近。

候选是被用来给出一个精确的意思去查询非确定数据库。特别地，一个 CQA 在非确定性数据库的每个候选上是被要求显示查询结果。

如果 I 是数据库 D 上符合一组完整性约束∑的一个候选数据库，则不存在 I⊂I*，且 I* ⊨∑和 I*⊆I。因为如果假定存在 I⊂I*，且 I* ⊨∑和 I*∈I。则存在元组 t∈I*且 t∉I，由于 I* ⊨∑，因此 I*不存在与元组 t 相冲突的元组 t'，即 t'∉I 且 t'∉I*，得到△(I,I')⊃△(I,I*)，根据定义 5，则 I 不是一个候选数据库。

定义 6.确定性查询。设 I 是模式 R 上一个可能不满足一组完整性约束∑的数据库实例。给定一个查询 q，如果对于 I 关于∑的每个候选数据库 Rep(I,∑)，都有 Rep(I,∑) ⊨q(t)，则 t 是关于∑的确定性结果，表示为 t∈CQA(q,I,∑)，CQA(q,I,∑)为查询 q 在 I 上关于∑的确定性查询应答。如果 q 为布尔查询，则表示为 CQA(q,I,∑)＝TRUE(FALSE)。

我们的方法计算 CQA 是基于查询重写。给出一个 SQL 查询 q 和一组键约束∑，将重写 q 到另外一个 SQL 查询 Q^C 进行检索。

定义 7.对于给定的关系 R 上的合取查询 q，把关系 R 的键属性表示为 KEY。q 的形式为：SELECT S FROM R [WHERE SC] [ORDER BY O]。

定义 8.对于给定的关系 R 上的合取查询 q，把其条件选择谓词形式表示为 SC={SC_1,SC_2,⋯,SC_n}，其条件选择谓词 SC 的否定形式表示为 NSC={NSC_1, NSC_2, ⋯ ,NSC_n}，其属性表示为 SCA={SCA_1, SCA_2,⋯,SCA_n}。

显然，元组的属性值可以采用 SQL 查询中的 IS NULL 和 IS NOT NULL 表示。为了便于表达，本文引入一个新的谓词 ISNULL 进行表示，代替 SQL 中的 IS NULL。

定义 9.对于给定的关系 R 上的合取查询 q，如果 R 的属性 A 存在空值，那么谓词 ISNULL(A)用于判断属性 A 的元组值是否为空。

本文给出了无连接的查询重写算法 1，基本思想如下：

(1) 对于给定的初始查询，重写查询在常见表表达式中基于初始查询加入了 DISTINCT 和键属性，用于消除投影于键属性和初始查询中的投影属性的重复元组。

(2) 在常见表表达式中的子查询中，采用谓词“R1.S<>R2.S”、NSC 和 ISNULL(SCA)进行判断，剔除了不满足初始查询的元组，主要是 3 种情况：a)键值相等但投影属性值不相同；b)键值相等但不满足条件谓词(否定形式)；c)键值相等但条件谓词中的属性为 NULL。最后在外查询中返回初始查询的非键属性值。

算法 1．Rewrite1(q,∑)

输入：无连接合取查询 q；软约束∑

输出：q 的重写查询 Q

BEGIN

cand←R 投影中的 SC 加入 DISTINCT 和 KEY 的查询子集；

filter←R 中 KEY 相等但投影属性值不相同或 NSC 或 ISNULL(SCA)的查询子集；

cand←cand－filter；

end←从候选集 cand 中对投影属性 S 的查询子集；

Q←cand+end；

END

4 结论

本文对于给定的一组键查询约束，重写了只返回确定性结果的 SQL 查询。提出了非确定性数据的确定性查询概念，给出的查询重写方法能有效地进行数据库 SQL 查询语句，它可以违反一系列的约束条件，重写查询去查找相应的与约束一致的数据。下一步的工作我们将考虑多关系连接的非确定性数据查询问题。

参考文献

1 Arenas M, Bertossi L, Chomicki J. Consistent Query Answers in Inconsistent Databases. PODS, 1999:68－79.

2 Chomicki J, Marcinkowski J, Staworko S. Hippo: A System for Computing Consistent Answers to a Class of SQL Queries. EDBT, 2004:841－844.

3 Fuxman A, Miller RJ. First-Order Query Rewriting for Inconsistent Databases. ICDT, 2005:337－351.

4 Dasu T, Johnson T. Exploratory Data Mining and Data Cleaning. New York: John Wiley, 2003.