基于范围语义的非一致性数据库聚集查询

谢 东，吴 敏

(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非聚集查询技术，发展普通的一致性查询应答的语义，提出非一致性数据库的范围语义，给出基于一致性查询范围的实际聚集查询重写方法，返回基于这种语义的聚集查询范围值。通过对初始分组属性和键属性聚集得到候选结果集，再过滤与初始分组属性值相冲突和不满足初始查询的元组。对于最小值，过滤掉可能不一致的元组，去掉可能值，得到一致性值；对于最大值，得到可能的最大候选值，实验基于 TPC-H 基准采用不同的参数进行研究。研究结果表明，聚集属性和投影属性数量、数据库以及初始查询的结果集对重写查询的负载有显著的影响，但非一致性数据比例和共享共同键值元组数的影响很小。尽管重写查询比初始查询的执行时间长，但是可以接受。

关键词：关系数据库；非一致性数据库；聚集查询
中图分类号：TP311 文献标识码：A 文章编号：1672−7207(2008)04−0810−06

Aggregation queries based on range semantics in inconsistent databases

XIE Dong, WU Min

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: Based on the query technique without aggregation in inconsistent databases, the common semantics of consistent query answering was developed to present the range semantics and a practical rewriting approach for aggregation queries based on the range of consistent answer, which returned to the range values. The candidate set was obtained by aggregating the original grouping and key attributes, and filtering the tuples that conflicted with the original grouping attribute values and were dissatisfied with the original query. For the minimum, the approach filtered possible inconsistent tuples and discarded the possible values to obtain the consistent values. For the maximum, the possible greatest candidate values were obtained. In the experiment, TPC-H was used to study the performance with different references. The results show that the overload of rewritten query is obviously affected by some aspects such as the number of aggregation and projection attributes, the size of database and result sets of the original query, but the overload has little effect on the proportion of inconsistent data and the number of tuples with the same key values. The running time of rewritten queries is longer than that of the original queries, but the overload is reasonable and the approach is effective.

Key words: relational database; inconsistent database; aggregation query

完整性约束(integrity constraint，IC)有效地保证了数据的完整性。然而，现实世界的一个实体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,

收稿日期：2007−10−25；修回日期：2008−01−28
基金项目：湖南省自然科学基金资助项目(07JJ6113，07JJ3119)；湖南省教育厅科研基金资助项目(07C832，07C385)
通信作者：谢 东(1971−)，男，湖南安化人，博士后，从事数据管理和非一致性数据库研究；电话：13657381116；E-mail: lgxzy@163.com

IDB)[1]，例如，数据从多个满足键约束的独立数据源被集成[2]，可能就不满足约束。数据清洗[3]能识别和纠正数据中的错误，恢复数据库到一致性状态，但这种技术是半自动的，且处理代价昂贵。非一致性数据往往是有用的，不可能为了保存数据的一致性而修改数据，导致有用的数据丢失，因而IC不能被强行来控制集成系统的数据完整性。现有的技术都是基于一致性数据库的，不支持在IDB上得到的不包含非一致性数据的“净化”查询结果。替代数据清洗的一种方法是一致性查询应答(Consistent query answering, CQA)[1]，CQA保持源数据不改变，在查询时解决非一致性，识别一致性数据。

在一致性数据环境中，标准聚集查询能获得确定的数值。由于在非一致性数据库中元组数值的不确定性，聚集查询不能返回一致性结果，因此，需要对CQA语义进行调整[4−5]，在范围语义下，关系的数字属性的最大值和最小值的计算可以被一阶查询表示。查询应答集[6]返回的最小区间值包含在候选数据库中获得的聚集函数值。ConQuer系统[7−10]发展了一种聚集查询重写框架，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，考虑了正负值的区间。Hippo系统[11−12]产生元组冲突图放到内存中去计算一致性结果，前提是数据库中只存在较少的冲突元组，产生的冲突超图足够小，能存储到主存中。问题是若数据库中存在大量的冲突元组，则冲突图加载到内存中会造成额外的系统负载。相对于查询重写，Infomix系统[13−14]能处理所有的一阶查询和更多的完整性约束类型，但其重点是基于稳定模型语义的可表达性和获取正确的逻辑结果，而不是计算的有效性和可扩展性，不能产生有效的解决结果。

CQA的聚集查询计算复杂性是一个挑战性问题，大多数聚集函数的计算是NP完全问题，也显示一些聚集查询能被有效地计算[4−5]。在计算资源有限的情况下，聚集查询基于函数依赖，由一个标准的聚集操作符应用(MIN(A), MAX(A), COUNT(*), COUNT(A), SUM(A)和AVG(A)) 组成。

本文作者在已有非一致性关系数据库的非聚集查询技术的基础上，提出非一致性数据库的范围语义，给出基于一致性查询范围的实际聚集查询重写方法，返回基于这种语义的聚集查询的范围值。实验基于TPC-H决策支持基准[15]采用不同的参数(数据尺寸、非一致性数据的比例以及违反键约束的元组个数)进行研究，结果表明该方法是有效的。

1　相关概念

一个数据库模式是具有有限元数的关系中有限元组的集合。CQA在关系数据库中，采用固定的关系模式$S=(U, R, B)$。其中：U表示可能的无限数据库域，R表示一组数据库谓词，B表示一组数据库内置谓词(=，≠，>，≥，<，≤)。这个模式决定了一阶谓词逻辑语言$L(S)$。一个数据库实例D可以看作是数据库元组$R^*(c_1, \cdots, c_n)$的有限集合(R^*是R的 1 个谓词，$c_1, \cdots, c_n$是U中的常数)。完整性约束C是$L(S)$中的语句，基于模式的一阶查询语言。因此，一个查询是$L(S)$的一阶形式$Q(x_1, \cdots, x_n)$，其中，$x_1, \cdots, x_n$是自由变量，$n\geq 0$。若每个$D'\in \mathrm{Rep}(D, C)$：$D'|Q(t')$，则元组$t'=(t_1, \cdots, t_n)$是一致性结果。即当$x_1, \cdots, x_n$分别取值$t_1, \cdots, t_n$时，若$n=0$，则查询是布尔查询。对于每个$D'\in \mathrm{Rep}(D, C)$，若$D'|=Q(t')$，则查询为真，否则为假。

定义 1　非一致性数据库[1]。设I是数据库D的 1 个实例，对于D上的任意完整性约束C，若$\forall I \models C$，则D是一致，否则D是非一致性数据库。

定义 2　软约束(Soft constraint, SC)。用户给定的一组完整性约束Σ，没有建立在可能的非一致性数据库D上，而是仅仅用于用户查询D时，把Σ作为查询一致性数据的考虑因素，则这种约束称为软约束。

定义 3　对称差(Distance[1])给定数据库实例I和I'，则 2 个数据库实例的对称差$\Delta(I, I')=(I-I')\cup(I'-I)$。

定义 4　候选数据库(Repaired/Candidate database，CDB[1])。给定一组完整性约束Σ和数据库实例I，若$I'|\Sigma$，则I'是I关于Σ的 1 个CDB，且不存在$I^* \models \Sigma$和$\Delta(I, I')\supset \Delta(I, I^*)$。

I'是关系中满足Σ的I的子集，与I有最小的区别。$\mathrm{Rep}(I, \Sigma)$ 表示I关于Σ的 CDB。

定义 5　一致性查询应答(CQA[1])。设I是模式R上 1 个可能不满足一组完整性约束Σ的数据库实例。给定 1 个查询Q，若对于I关于Σ的每个CDB I_{REP}，都有$t\in Q(I_{\mathrm{REP}})$，则t是关于Σ的一致性结果，表示为$t'\in Q_{\mathrm{CQA}}(I, \Sigma)$。若为布尔查询，则$Q_{\mathrm{CQA}}(I, \Sigma)$=true(false)。

定义 6　聚集查询q表示为如下形式(其中G是一组分组属性，$\mathrm{agg}_1(e_1), \cdots, \mathrm{agg}_n(e_n)$是聚集表达式)：

SELECT G, $\mathrm{agg}_1(e_1)$ e_1, $\cdots$, $\mathrm{agg}_n(e_n)$ e_n FROM R [WHERE SC] GROUP BY G

由于考虑的是非一致性数据，完整性约束不能建立在违反约束的数据库上，且 SQL 查询可以有属性在 GROUP BY 语句中，但没有出现在 SELECT 语句中，

因此，给出如下假定。

假定 **1** 假定研究的完整性约束均为软约束。

假定 **2** 假定查询中 GROUP BY 语句中属性也出现在 SELECT 语句中。

定义 **7** 区间(Interval[4])给定一组完整性约束 Σ 和数据库实例 I，若聚集查询 q 在每个候选数据库上返回到 $v\in[a, b]$，则 q 在 I 上的一致性结果是区间 $[a, b]$($-\infty<a\leq b<\infty$)，表示为 $I\models q_{\Sigma}\in[a, b]$。若 $[a, b]$ 是一致性结果，则 a 为下界，b 为上界。若没有子区间是一致性结果，则 $[a, b]$ 是最优的一致性结果，其中：a 最大下界(greatest-lower-blound-answer, $g_{\text{lb-answer}}$)，b 为最小上界(least-upper-blound-answer, $l_{\text{ub-answer}}$)。

对聚集表达返回范围(边界值)，不是准确的数值。基于范围语义，需要调整 CQA 的语义，定义 8 给出了范围保证分组属性 G 的每个值是一致性结果。

定义 **8** 一致性查询范围(Range of consistent answer, ROCA)。设 D 是一个数据库，q_G 是一个聚集查询，q 是 q_G 去掉聚集后的查询，Σ 是一组查询约束。对于每个候选数据库 I，若 t 是 q 在 D 上的一致性结果，且每个聚集值 v 满足 $g_{\text{lb-answer}}$(最大下界)$\leq v\leq l_{\text{ub-answer}}$(最小上界)，则 $(t, g_{\text{lb-answer}}, l_{\text{ub-answer}})$ 是 q_G 在 D 上的一致性查询范围。

2 聚集查询重写

例 **1** 给定表示客户余额的关系 R(*custkey, nation, mkt, balance*)，分别表示客户号，国别，市场和余额，存在 $t_1, \cdots, t_6$ 元组的实例 $I=\{(c_1, n_1, b_1, 1\ 000), (c_1, n_2, b_1, 100), (c_2, n_2, b_1, 2\ 000), (c_2, n_2, b_2, 200), (c_3, n_2, b_2, 3\ 000), (c_3, n_1, b_2, \text{NULL})\}$，考虑如下查询：

SELECT *nation*, SUM(*balance*) FROM *R* WHERE *mkt*= 'b_1' GROUP BY *nation*

I 有如下的CDB：$I_1=\{t_1, t_3, t_5\}$，$I_2=\{t_1, t_3, t_6\}$，$I_3=\{t_1, t_4, t_5\}$，$I_4=\{t_1, t_4, t_6\}$，$I_5=\{t_2, t_3, t_5\}$，$I_6=\{t_2, t_3, t_6\}$，$I_7=\{t_2, t_4, t_5\}$，$I_8=\{t_2, t_4, t_6\}$。每个CDB是一致的，尽可能与 I 接近。其余额总和分别为 6 000，3 000，4 200，1 200，5 100，2 100，3 300，300，因而不是一致性结果。根据范围语义，可以描述范围为 $300\leq sumbal\leq 6\ 000$。假定关系 R 有 $2n(n>0)$ 个元组(每个元组存在另外一个冲突元组)，则有 2^n 个候选数据库。SUM(*pay*)在可能的指数级的CDB中将产生指数级的值(0 到 2^{n+1})，是不可计算的。

因此，考虑采用查询重写方法。查询重写是采用一阶查询重写方法对大量易处理的查询进行重写，重写的查询也是一阶的，可以被 SQL 表示，因此，CQA 计算能被相同的数据库查询引擎直接重用。由于重写查询能在 PTIME 和独立于数据的途径下得到，因而，这种方法在计算复杂度时是 PTIME 的。

t_1，t_2 和 t_3 是满足初始查询的元组，ROCA是由 n_1 和 n_2 的最值组成。计算ROCA，必须得到每个客户余额的上下边界值，然后求和。8 个CDB的余额区间值分别为 $\{(n_1, 1\ 000), (n_2, 2\ 000)\}$, $\{(n_1, 1\ 000), (n_2, 2\ 000)\}$, $\{(n_1, 1\ 000), (n_2, 0)\}$, $\{(n_1, 1\ 000), (n_2, 0)\}$, $\{(n_1, 0), (n_2, 2\ 100)\}$, $\{(n_1, 0), (n_2, 2\ 100)\}$, $\{(n_1, 0), (n_2, 100)\}$ 和 $\{(n_1, 0), (n_2, 100)\}$。因此，ROCA是 $\{(n_1, 0, 1\ 000), (n_2, 0, 2\ 100)\}$。在每个CDB中，*nation* 为 n_1 的客户总余额介于 0 和 1 000 之间，1 000 在 I_1，I_2，I_3 和 I_4 中，0 在其他CDB中；*nation* 为 n_2 的客户总余额介于 0 和 2 100 之间，2 100 在 I_5 和 I_6 中，0 在其他CDB中。采用如下的重写查询得到ROCA。

```
WITH cand AS(
    SELECT nation, custkey, MIN(balance)
    minb,MAX(balance) maxbal
    FROM R WHERE mkt = 'b1'
    GROUP BY nation,custkey ),
min_cand1 AS(
    SELECT custkey FROM cand
    GROUP BY custkey
    HAVING COUNT(*)=1),
min_cand2 AS(
    SELECT custkey FROM min_cand1 m
    WHERE NOT EXISTS(SELECT *
                FROM R c
                WHERE c.custkey=m.custkey
                AND (c.mkt<>'b1' OR
                    c.mkt IS NULL))),
min_cand3 AS(SELECT nation,minb
    FROM cand c, min_cand2 m
    WHERE c.custkey=m.custkey),
max_cand AS(
    SELECT nation,SUM(minb) minb,
    SUM(maxbal) maxbal FROM cand
    GROUP BY nation)
SELECT m2.nation,(CASE WHEN m1.nation IS
    NULL THEN 0 ELSE m1.minb END)
    minbal,maxbal
FROM max_cand m2 LEFT OUTER JOIN
    min_cand3 m1 ON m2.nation =m1.nation
```

算法 1 给出了无连接聚集查询重写算法，基本思

想如下：

a. 最小值由于存在可能值(即冲突元组的最小值为 0)，必须要把可能值排除掉，得到必然值(一致性元组)，然后求最大值。首先，对初始查询加入键属性，对初始分组属性和键属性进行聚集求最大值和最小值，得到 *cand* 结果集。

b. 由于冲突元组的最小值为 0，*min_cand1* 用于剔除 *cand* 中的初始分组属性值相冲突的元组。在 SC 的条件选择谓词中，可能有不满足选择条件的情况，若选择条件相反(采用 NSC 表示)和选择条件属性值为空(采用 ISNULL(SCA)表示)，则表明对于键属性值来说，由于存在多个选择分组属性值，可能是不满足初始查询的元组出现，因此，其最小值也为 0，需要在 *min_cand1* 的过滤基础上，再过滤掉不满足初始查询的元组，得到 *min_cand2* 结果集。

c. 对过滤产生的最小值 *min_cand2* 结果集与候选结果集进行连接，产生过滤的初始分组属性的最小值 0。由于最大值是可能的最大候选值，因此，对 *cand* 结果集的初始分组属性聚集求和产生的 *max_cand3* 结果集中的最大值是确定的，没有被 *min_cand2* 过滤的最小值是确定的。

d. 最后 *max_cand* 结果集左连接 *min_cand3* 结果集，若 *min_cand3* 的初始分组属性为空，则返回为 0，否则，返回确定的最小值和最大值。

算法 **1**. RewriteAgg(q, Σ)

输入：聚集查询 q；软约束 Σ

输出：重写查询 Q：

```
WITH cand AS(
    SELECT G,key,MIN(e1) AS
        min_e1, MAX(e1) AS max_e1,…,
        MIN(en) AS min_en,MAX(en) AS
max_en
    FROM R WHERE SC GROUP BY G, key),
min_cand1 AS(SELECT key
    FROM cand GROUP BY key
    HAVING COUNT(*)=1),
min_cand2 AS(
    SELECT key FROM min_cand1 m
    WHERE NOT EXISTS( SELECT * FROM R
            WHERE R.key =m.key AND
            (NSC OR ISNULL(R.SCA)))),
min_cand3 AS(
    SELECT G,min_e1,…, min_en
    FROM cand c,min_cand2 m
    WHERE c.key =m.key),
max_cand AS(
    SELECT G, SUM(min_e1) AS min_e1,…,
    SUM(max _en) AS max _en,SUM(max_e1)
    AS max_e1,…, SUM(max _en) AS max _en
    FROM cand GROUP BY G)
SELECT m2.G,(CASE WHEN m1.G IS NULL
    THEN 0 ELSE m1.min_e1 END) min_e1,
    max_e1,…, CASE WHEN m1.G IS NULL
    THEN 0 ELSE m1.min_en END) min_en,
    max_en
FROM max_cand AS m2 LEFT OUTER JOIN
    min_cand3 AS m1 ON m2.G=m1.G
```

3　实验分析

评价重写方法的实验环境为Intel Celeron 2.53 GHz，内存为 512 MB，Windows XP professional，SQL Server2005。实验采用TPC-H规范查询中的Q_1和Q_6[15]，Q_1和Q_6的投影和聚集属性数量分别为 10，8 和 1，1。采用TPC-H查询建议的标准设置，生成不同尺寸的非一致性数据，并考虑如下参数。

a. 数据尺寸 s：考虑 s=0.1，0.5，1 和 2 GB(1 GB 数据库约有 800 万元组)。

b. 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，实验的 n 取 2。

c. 数据库非一致性的百分比 p：完整的数据库每个关系有相同的 p，实验的 p 采用 0，0.01，0.05，0.10，0.20 和 0.50。

图 1 所示为p=5%和n=2 的 1GB数据库的计算负载。采用$(T_r-T_o)/T_o$(T_r表示重写查询时间，T_o表示初　始查询时间)公式来计算，Q_1的重写查询负载为 9.89，

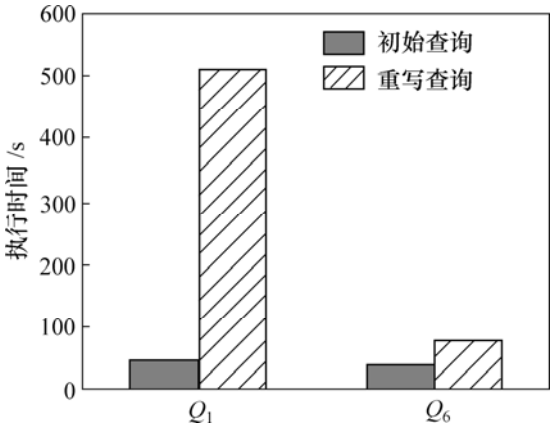

图 **1**　Q_1和Q_6的执行时间

Fig.1　Running time of Q_1 and Q_6

Q_6的重写查询负载为 0.88。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的范围值的计算需要更多的时间。由于Q_1具有更多的聚集属性和投影属性，因此，Q_1的重写查询执行时间比Q_6的长。若删除Q_1的所有分组，则得到 590 万个元组的结果集，而其他所有查询的结果集大小在 12 万个元组以下。算法需要对中间的结果集进行计算，首先产生候选集*cand*，然后，产生*min_cand1*，*min_cand2*，*min_cand3*和*max_cand*，由于中间结果集很大，因此，产生了较大的查询负载。

改变 1 GB数据库的p和n，研究数据库的非一致性比例的影响。图 2 所示为p从 0 到 0.5，n＝2 的运行结果。p对初始查询和重写查询的运行时间只有很小的影响。当改变p时，Q_6结果集的大小不变，因为所有数据库大小相同。

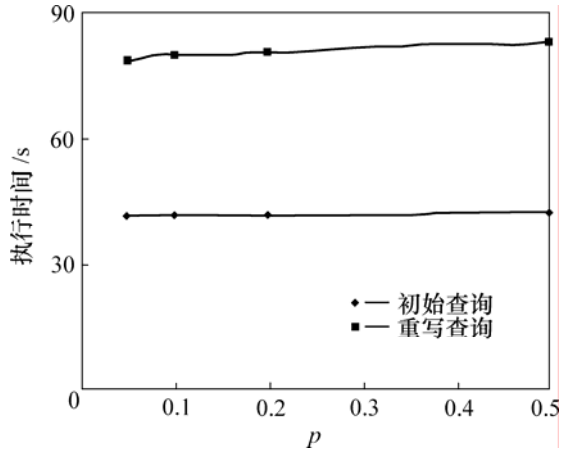

图 **2** 非一致性数据比例p不同时Q_6的执行时间

Fig.2 Running time of Q_6 at different p

图 1 和图 2 所示结果表明，重写查询的负载主要由初始查询的结果(不考虑分组)与聚集属性和投影属性数量控制。

图 3 所示为改变n值和固定p=0.1 的Q_6的运行时间。n在任何 2 个可供选择重写的运行时间的影响是很小的。图 4 所示为不同尺寸下Q_6的查询重写的执行时间(n=2)。采用 100 MB，500 MB，1 GB，1.5 GB和 2 GB的数据库。为了保证一致性的比较，数据库的违反约束的元组总数保持常数。给出 40 万个非一致性元组，则p值 2.5，3.3，5，10 和 50 分别对应 2 GB，1.5 GB，1 GB，500 MB和 100 MB数据库。可以看到，运行时间随数据库的大小呈线性变化。

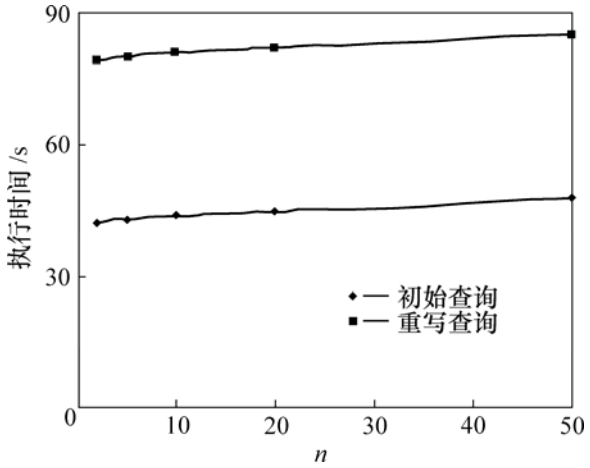

图 **3** 元组数n不同时Q_6的执行时间

Fig.3 Running time of Q_6 at different n

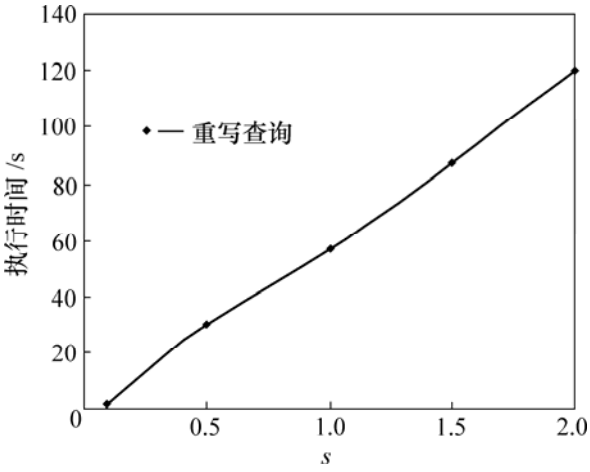

图 **4** 数据大小s不同时Q_6的执行时间

Fig.4 Running time of Q_6 at different s

4 结 论

a. 提出了非一致性数据库的范围语义，给出了基于一致性查询范围的实际聚集查询重写方法，返回基于这种语义的聚集查询的范围值。

b. 聚集属性和投影属性数量、数据库大小以及初始查询的结果集大小对重写查询的负载有显著的影响，但非一致性数据比例和共享共同键值元组数的影响很小。尽管重写查询比初始查询的执行时间要长，但是可以接受，方法是有效的。

c. 下一步的工作需要考虑多个关系连接下的聚集重写查询问题，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]//Proceedings of the ACM Symposium on Principles of Database Systems. New York: ACM Press, 1999: 68−79.

[2] 邹小兵, 蔡自兴, 魏世勇, 等. 基于分布式 Agent 的 ERP 子系统设计[J]. 中南工业大学学报: 自然科学版, 2003, 34(4): 424−427.
ZOU Xiao-bing, CAI Zi-xing, WEI Shi-yong, et al. Design of an ERP sub-system based on distributed agent[J]. Journal of Central South University of Technology: Natural Science, 2003, 34(4): 424−427.

[3] Dasu T, Johnson T. Exploratory data mining and data cleaning[M]. New York: John Wiley, 2003.

[4] Arenas M, Bertossi L, Chomicki J. Scalar aggregation in FD-inconsistent databases[C]//Proceedings of the International Conference on Database Theory. Berlin: Springer-Verlag, 2001: 39−53.

[5] Arenas M, Bertossi L, Chomicki J, et al. Scalar aggregation in inconsistent databases[J]. Theoretical Computer Science, 2003, 296(3): 405−434.

[6] Arenas M, Bertossi L, Chomicki J. Answer sets for consistent query answering in inconsistent databases[J]. Theory and Practice of Logic Programming, 2003, 3(4/5): 393−424.

[7] Fuxman A, Fazli E, Miller R J. ConQuer: efficient management of inconsistent databases[C]//Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2005: 155−166.

[8] Fuxman A, Miller R J. Towards inconsistency management in data integration systems[C]//Proceedings of the IJCAI Workshop on Information Integration on the Web. New York: ACM Press, 2003: 143−148.

[9] Fuxman A, Fuxman D, Miller R J. ConQuer: A system for efficient query answering over inconsistent databases[C]// Proceeding of the International Conference on Very Large Databases. New York: ACM Press, 2005: 1354−1357.

[10] Fuxman A, Miller R J. First-order query rewriting for inconsistent databases[C]//Proceedings of the International Conference on Database Theory. Berlin: Springer-Verlag, 2005: 337−351.

[11] Chomicki J, Marcinkowski J, Staworko S. Computing consistent query answers using conflict hypergraphs[C]//Proceeding of the International Conference on Information and Knowledge Management. New York: ACM Press, 2004: 417−426.

[12] Chomicki J, Marcinkowski J, Staworko S. Hippo: a system for computing consistent query answers to a class of SQL queries[C]//Proceedings of the International Conference on Extending Database Technology. Berlin: Springer-Verlag, 2004: 841−844.

[13] Eiter T, Fink M, Greco G, et al. Efficient evaluation of logic programs for querying data integration systems[C]//Proceedings of the International Conference on Logic Programming. Berlin: Springer-Verlag, 2003: 163−177.

[14] Lembo D, Lenzerini M, Rosati R. Source inconsistency and incompleteness in data integration[C]//Proceedings of the International Workshop on Knowledge Representation Meet Databases. Berlin: Springer-Verlag, 2002: 79−86.

[15] Transaction Processing Performance Council. TPC-H standard specification[EB/OL]. [2006−05−30]. http://www.tpc.org.