软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(4):863−879 [doi: 10.13328/j.cnki.jos.004426]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

不确定关系数据属性级溯源表示与概率计算*

王 梁[1], 周光焱[1], 王黎维[2], 彭智勇[1]

[1](武汉大学 计算机学院,湖北 武汉 430072)

[2](武汉大学 国际软件学院,湖北 武汉 430079)

通讯作者: 王黎维, E-mail: liwei.wang@whu.edu.cn; 彭智勇, E-mail: peng@whu.edu.cn

摘 要: 传统的数据库应用中,数据往往被假定是精确可用的,而实际中数据普遍存在不确定性.以往许多利用溯源信息追踪数据不确定性的方法往往关注元组或单一属性存在不确定性的情况,通过对元组添加唯一变量标识,用变量标识所表示的溯源信息构造布尔表达式计算结果元组概率.当元组中多个属性存在不确定性时,对元组进行标识不能帮助用户快速而准确地找到造成不确定性的源属性值.定义属性表达式,并通过属性表达式构造溯源表达式.利用该溯源表达式不仅可以准确地追溯不确定性产生的具体位置,同时还可以实现结果元组的概率计算.为保证概率计算结果的正确性,提出溯源表达式的转换算法.通过分析影响结果元组概率计算效率的因素,还提出构建共享路径表的方法,在构建过程中对原子析取式进行预计算,以提高概率计算的效率.实验部分将该方法与现有的元组级溯源信息表示方法在时间代价和空间代价方面进行比较,验证其可行性和有效性.此外,实验部分还对利用共享路径加快结果元组概率计算的有效性进行了评估.

关键词: 不确定性;属性表达式;溯源表达式;概率计算;共享路径

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 王梁,周光焱,王黎维,彭智勇.不确定关系数据属性级溯源表示与概率计算.软件学报,2014,25(4):863−879. http://www.jos.org.cn/1000-9825/4426.htm

英文引用格式: Wang L, Zhou GY, Wang LW, Peng ZY. Attribute level lineage and probabilistic computation of uncertain data. Ruan Jian Xue Bao/Journal of Software, 2014,25(4):863−879 (in Chinese). http://www.jos.org.cn/1000-9825/4426.htm

Attribute Level Lineage and Probabilistic Computation of Uncertain Data

WANG Liang[1], ZHOU Guang-Yan[1], WANG Li-Wei[2], PENG Zhi-Yong[1]

[1](Computer School, Wuhan University, Wuhan 430072, China)

[2](International School of Software, Wuhan University, Wuhan 430079, China)

Corresponding authors: WANG Li-Wei, E-mail: liwei.wang@whu.edu.cn; PENG Zhi-Yong, E-mail: peng@whu.edu.cn

Abstract: In the traditional database applications, data is generally considered to be accurate and available. However, data uncertainty often occurs in the real world. Most of current methods usually use provenance information to track data uncertainty while placing focus on the uncertainty with tuple level rather than attribute level. Their main idea is to identify a tuple with a variable, and then construct Boolean expression based on provenance information to compute the probability of a tuple. For the tuple with lots of uncertain attributes, these methods can not help users rapidly and correctly identify the source of uncertainty. In this paper, attribute expressions are defined and used to construct the lineage expression for each result tuple. With the lineage expression, the new method can not only accurately traces the location where the uncertainty takes place, but also computes the probability of the result tuple. Meanwhile, the exchange algorithm of the lineage expression is proposed to guarantee the correctness of the probability computation. In order to improve the efficiency of the probability computation, a method is also provided to construct share paths, and compute the probability of atomic

* 基金项目: 国家自然科学基金(61232002, 61202033); 国家高技术研究发展计划(863)(2012AA011004); 教育部博士点新教师基金(200804861067); 湖北省自然科学基金(2011CDB448); 武汉大学博士研究生自主科研基金(2012211020207)

收稿时间: 2012-09-26; 修改时间: 2013-03-22; 定稿时间: 2013-05-03

disjunctions during the period of constructing share paths. Experiments are performed to compare tuple level lineage expressions with the existing methods on both time and cost. The results show the feasibility and validity of the proposed method, and further verify the validity of utilizing share paths to speed up the probability computation.

Key words: uncertainty; attribute expression; lineage expression; probabilistic computation; share path

在传统的数据库应用中,数据往往被假定是精确可用的,然而数据的不确定性却普遍存在于科学数据管理、近似查询处理、隐私保护等应用中[1].为了实现对不确定数据的管理,人们使用概率对数据的不确定性进行建模,并在传统的关系模式中引入概率维,定义了概率数据库[2].基于概率数据库进行查询时,结果元组中除了满足条件的数据外,还应包含数据的不确定性,即相应的概率.然而,只有在数据库操作过程中所涉及的各元组相互独立时,结果元组的概率才能正确地计算得出[3].由于数据的依赖关系普遍存在(即使原始数据不存在依赖关系,关系操作过程中也可能会产生),元组间存在依赖时,结果元组的概率计算成为概率数据库研究的重要问题.针对这个问题,文献[4]证明了在源元组相互独立时,基于溯源信息进行查询处理,结果元组概率计算的正确性.直观上,溯源信息是数据的派生信息,它记录了数据的来源和演化过程[5],从而可以提供概率计算时所需的独立/依赖关系.

以往的研究[3−4,6−11]大多只关注不确定性关系数据的元组级溯源表达,对于属性级溯源信息仅考虑单一属性存在不确定性的情况.这些工作的本质在于,利用唯一的变量标识单个元组,通过唯一变量的集合构造布尔表达式,以表达各种数据库操作.通过布尔表达式不仅可以溯源,而且可以计算结果元组的概率.然而,当单个元组中存在多个不确定属性且元组也存在不确定性时,现有的溯源方法并无法反映元组和属性之间的包含关系,导致结果元组的概率无法正确计算出来.具体来讲,若直接用唯一变量对不确定的元组及属性都进行标识,则无法通过变量判断属于某个元组的属性值是哪些,而属性取某个值的概率是在元组存在的前提下讨论才有意义.此外,仅对元组进行变量标识,不能帮助用户快速而准确地找到造成不确定性的源属性值.下面以一个车祸调查为例来具体地加以说明.

例 1:图 1 是一个车祸调查的实例.图 1(a)是目击调查表,记录了目击者看到的肇事车辆的品牌、驾驶员年龄的大致范围以及驾驶员衣服的颜色.其中,张三在作笔录时,不能确定车辆品牌一定是本田或现代,驾驶员衣服的颜色也可能不是黄色或橙色,因此对于张三的描述有一个可信度,作为该元组的可信度.图 1(b)是监控录像表,记录了车祸发生的那段时间摄像头拍到的各个品牌车辆的车牌号.由于视频拍摄问题,拍到的本田车的车牌号有两个可能取值.此外,由于肇事车辆可能是套牌车,因此对于摄像头拍到的车牌号可能是一个假的,故存在元组的不确定性.根据现有的元组级溯源方法,用唯一变量标识每条元组[2,3,5−10,12,13],通过可能世界模型[1]对不确定性进行建模.在每个可能世界实例中,各元组的各属性都是确定的取值.当进行表连接、选择或投影等操作时,若对结果元组进行溯源,只能追溯到其不确定性产生的元组,无法定位其不确定性产生的具体属性.此外,直接用唯一变量来标识各不确定属性和元组,虽然可以追溯不确定性到元组的具体属性,但通过各变量无法反映出元组和属性之间的关联关系,当利用溯源信息进行结果元组概率计算时可能会出错.例如,对于查询语句“select *S*1.目击者,车辆品牌,驾驶员衣服 from(select 目击者,车辆品牌 from 目击调查表) as *S*1,(select 目击者,驾驶员衣服 from 目击调查表) as *S*2 where *S*1.目击者=*S*2.目击者;”,用唯一变量 *a* 标识源元组(张三,本田|现代,25~30,黄色|橙色),用 *b* 和 *c* 分别标识 *a* 对应元组中的属性值“本田”和“黄色”.假设结果元组(张三,本田,黄色)是由来自中间结果表 *S*1 中的元组 *t*1 和 *S*2 中的元组 *t*2 连接得到的,则 *t*1 元组的存在概率是 *a* 存在的条件下属性“车辆品牌”为“本田”的概率,即 0.7×0.8.根据现有的溯源表示方法,用 $a\times b$ 表示 *t*1 的溯源信息,这样可以利用溯源信息来计算 *t*1 的概率,即 $P(a\times b)=P(a)\times P(b)=0.7\times 0.8$.同理可得 *t*2 的溯源信息为 $a\times c$.元组 *t*1 和 *t*2 做连接所得结果元组的溯源信息即为$(a\times b)\times(a\times c)$,该元组的存在概率为 *a* 对应元组存在的条件下属性“车辆品牌”为“本田”且属性“驾驶员衣服”为“黄色”的概率,即 $P(a)\times P(b)\times P(c)=0.7\times 0.8\times 0.3=0.168$.而直接利用溯源信息计算得到的概率值为 $P(a)\times P(b)\times P(a)\times P(c)=0.1176$.由此可见,在利用溯源信息进行概率计算时,需要识别属于同一元组的不同属性值,进而进行相应的转换以保证结果概率计算的正确性.

目击者	车辆品牌	驾驶员年龄	驾驶员衣服	可信度
张三	(本田,0.8)|(现代,0.2)	25~30	(黄色,0.3)|(橙色,0.7)	0.7
李四	(本田,0.6)|(现代,0.4)	28~35	(橙色,0.6)|(红色,0.4)	0.8

(a) 目击调查表

车辆品牌	时间段	车牌	可信度
本田	15:00~15:05	(沪H73,0.7)|(沪H13,0.3)	0.9
现代	15:00~15:05	沪H35	0.8

(b) 监控录像表

Fig.1 The traffic accident investigation

图 1 车祸调查

对于一个结果元组,为了能够准确地追溯到产生该元组的不确定属性,同时又可以正确计算出结果元组的概率,本文定义属性表达式,并通过其构建溯源表达式.利用该溯源表达式,可以准确地追溯数据的来源.在概率计算方面,本文从正确性和有效性两方面考虑:首先,提出溯源表达式的转换算法,使得结果元组的概率可以正确地计算;其次,提出构建共享路径表的方法来避免概率重复计算,并在构建共享路径的过程中对原子析取式进行概率预计算.实验结果表明:属性级溯源信息的存储代价与元组级溯源信息的存储代价相比并无太多增长,但却可以使用户准确地追溯不确定性产生的具体位置.同时,利用属性级溯源信息计算结果元组概率的时间代价也并无过多增长.而采用优化方案后,结果元组概率计算的时间开销又可极大地降低.

本文的贡献主要有以下两点:

第一,定义属性表达式,用以构建结果元组的溯源表达式.当利用该溯源表达式进行概率计算时,通过文中提出的转换算法来保证概率计算的正确性.该转换算法的时间复杂度仅与溯源表达式自身的复杂程度相关.多数情况下,溯源表达式不会过于复杂,当溯源表达式数量较多时,其转换的时间代价接近线性增长.

第二,提出构建共享路径表的方法,并对原子析取式的概率进行预计算来加快结果元组的概率计算.构建索引的过程中,散列溯源表达式的空间复杂度仅与溯源表达式自身的复杂程度相关,与溯源表达式的规模无关.而构建共享路径表的空间代价仅与溯源表达式集合中含有的共享路径个数相关,且共享路径表可以增量地更新.

本文第 1 节是相关工作的介绍.第 2 节定义属性表达式,并通过其构建溯源表达式.第 3 节定义溯源树来表示溯源表达式,并提出溯源树上的转换算法,保证概率计算结果的正确性.第 4 节提出构建共享路径表的方法,在构建共享路径的过程中,对原子析取式进行概率预计算,以提高结果元组概率计算的效率.第 5 节是实验评估.第 6 节总结全文,并展望未来的研究工作.

1 相关工作

在概率数据库中,最常用的模型是可能世界模型[1].可能世界空间由一系列可能世界实例组成,每一个可能世界实例对应一个确定性数据库.对于概率数据库中某元组,其不确定属性在某个可能世界实例中是满足约束条件的确定值.目前,已有很多研究针对关系型数据扩展了可能世界模型,包括 Probabilistic ?-table[14], Probabilistic or-set table[15],Probabilistic or-set-? table[6,16]等.其中,Probabilistic ?-table 模型仅考虑元组级不确定性;Probabilistic or-set table 模型仅考虑属性级不确定性;而 Probabilistic or-set-? table 模型则融合了前面两个模型,既考虑元组级不确定性,又考虑属性级不确定性.

文献[17]对概率数据库中存在的关联关系进行了讨论,从元组级来看,元组之间可能有关联关系,如共存关系、互斥关系等.从属性级来看,属性之间也可能存在关联关系.当元组之间不存在生成规则时,则称元组是独立的.同样,当属性之间不存在生成规则时,则称属性是独立的.此外,属性值可能是离散的,也可能是连续的.

概率数据库中的数据由于存在不确定性,为了对不确定性进行追踪,可以考虑数据溯源技术.数据溯源是评估数据质量和数据可靠性的重要方法,其基本思想是:在数据管理和使用的过程中,记录数据的来源和演化的整个过程,并支持对数据的来源和数据演化过程的查询.在数据出现异常时,溯源信息可以用来追溯根源,核实取证,进而评价数据的质量和正确性,甚至进行数据的恢复[12,18−20].文献[19]引入 how-provenance 的溯源表示法,它

不仅可以表示数据的来源,还反映出其派生过程.对数据进行溯源时,文献[20]提出逆过程的方法,不预存溯源信息,而是在溯源的过程中通过逆函数实时计算.该方法可以降低细粒度溯源信息的存储代价,但若没有高效的逆过程,该方法等价于重新执行查询.随着存储设备成本的降低,时间代价相对于空间代价更为重要,因而当前大部分研究均采用标记的方法,把溯源信息存储下来.

在利用溯源技术来计算目标数据概率方面,文献[4]最早提出把数据的不确定性和溯源相结合,并实现了第一个不确定性溯源关系数据库 TRIO.该系统不仅跟踪数据的溯源信息,也表达数据的不确定性.文献[7]探索了在不确定性关系数据库中如何利用溯源信息进行概率值的计算问题,提出把查询和概率计算分开处理的思想.这样,在用户不关注查询结果的概率时,可节约计算概率值的开销.文献[8]中利用溯源信息来计算元组概率,通过how-provenance来表示溯源信息,进而计算元组的概率.随着数据集成技术的发展,现实应用中,单个数据项的溯源信息可能达到几十兆字节甚至更大,在不影响查询结果的情况下,采取近似溯源信息表示策略追踪若干最重要的溯源路径将更合理、更适用[9].文献[9]定义了不确定性溯源信息的保守近似和多项式近似查询策略,详细讨论了这两种近似策略产生的误差,提供了创建、理解和处理近似不确定性溯源信息的基本算法.由于元组之间可能存在依赖关系,文献[10]提出利用连接树(junction tree)来存储元组的概率以及具有直接依赖关系的元组的联合概率,通过它来计算任意两个元组之间的联合概率.连接树是概率图模型(PGM)的一种等价表示,文献[13]基于概率图模型提出了一种不确定性数据溯源的表示方法,以实现从表示溯源信息的布尔公式到贝叶斯网的等价转换.

2 溯源表达式

当前,有关概率数据库上溯源技术的相关研究大部分集中在源元组相互独立、属性也相互独立且概率值为离散值的情况.在该情况下,无需考虑源元组或属性之间复杂的依赖关系;但当元组中多个属性存在不确定性时,其溯源表达和结果元组概率计算的问题仍没有得到解决.因此,本文的方法也是假定源元组和属性均相互独立,概率值为离散值.本文基于 Probabilistic or-set-? table 模型,定义含有溯源信息的概率数据库如下:

定义 1(含溯源信息的概率数据库). 含有溯源信息的概率数据库 D 是一个三元组(R,I,L),其中,R 表示该数据库中所包含的关系表集合,它们既含有元组级不确定性,又含有属性级不确定性;I 是 R 中不确定性元组的某个属性取特定值时所对应的标识集合,该标识记作 $t\rightarrow[m].n$,称为属性表达式,表示元组 t 的第 m 个属性的第 n 个可能取值;L 是从 I 到幂集 2^I 的一个溯源函数映射.

在概率数据库 D 中,若某关系表是源表,则其中的元组不含来自于其他表中的数据,对应元组不存在溯源信息.若源表中元组包含不确定属性,则结果表中每条元组对应一个属性表达式集合,通过属性表达式可以追溯到源表中的属性.当属性和元组同时存在不确定性时,记元组的概率为 $P(X)$.而属性取某个值的概率是在元组存在的前提下,因此记为 $P(Y|X)$.对于元组的一个可能实例,其概率为 $P(XY)=P(X)\times P(Y|X)$.由于需要利用溯源信息进行概率计算,因此属性表达式 $t\rightarrow[m].n$ 对应的可能实例概率表示为 $P(t\rightarrow[m].n)=P(t)\times P([m].n)$,它表示元组 t 存在,且 t 的第 m 个属性取第 n 个可能值的概率.其中,$P(t)$表示元组 t 的存在概率,$P([m].n)$表示元组 t 的第 m 个属性取第 n 个可能值的概率.

为了解决例 1 中提到的问题,需要在概率数据库对溯源信息重新定义,使得对于各种基本数据库操作得到的结果表中的元组均可以准确地追溯其不确定性产生的来源,同时可以正确计算其存在概率.例 2 给出了概率数据库中的 3 张基本关系表,对于在其上进行各种类型数据库操作得到的结果元组,本文将定义其溯源信息的表示形式以及如何利用这些信息计算元组概率.

例 2:图 2 包含 3 张关系表,每个表均有两个属性,每个属性均有多个可能取值,每个可能取值对应一个概率.每个表的最后一列“exist”对应的是元组的存在概率,对图 2 的讨论将贯穿全文.

$R_1(A,B)$			
ID	A	B	Exist
1	{(15,0.7),(18,0.3)}	{(23,0.8),(25,0.2)}	0.8
2	{(15,0.8),(13,0.2)}	{(21,0.6),(26,0.2)}	0.9

$R_2(A,C)$			
ID	A	C	Exist
3	{(15,0.8),(18,0.2)}	{(23,0.7),(25,0.3)}	0.8
4	{(15,0.6),(16,0.4)}	{(41,0.6),(42,0.2)}	0.7

$R_3(C,D)$			
ID	C	D	Exist
5	{(23,0.8),(24,0.2)}	{(17,0.7),(25,0.3)}	0.9
6	{(19,0.3),(16,0.7)}	{(31,0.6),(52,0.2)}	0.7

Fig.2 Uncertain relational table R_1, R_2, R_3

图 2 不确定性关系表 R_1,R_2,R_3

图 2 中的不确定性关系表 R_1,R_2,R_3,它们既含有属性级不确定性,也含有元组级不确定性.

属性表达式 01→[1].2 表示 ID 为 01 的元组的属性 A 取值为 18,它可以转化为概率求解表达式,即

$$P(01\rightarrow[1].2)=P(01)\times P([1].2)=0.8\times 0.3.$$

在概率数据库 D 中,若某关系表是非源表,则其中的元组来自于其他表中的数据,对应元组存在溯源信息.由于 L 中只记录了元组对应源头的属性表达式,为了能够对这些元组进行溯源,还需要记录数据库操作的过程,因此定义溯源表达式如下:

定义 2(溯源表达式). 由属性表达式通过析取(∨)、合取(∧)操作构成的表达式称为溯源表达式.溯源表达式 $\lambda=t\rightarrow[m].n\Theta\lambda'$,其中,$\Theta$表示析取或合取操作符,$\lambda'$为空或为溯源表达式.

对于非源表中元组包含的属性,其可能来自于源表或中间结果表.若直接参与结果元组派生的表是中间结果表而非源表,则需要寻找其对应的源表属性,因而根据其属性取值来源的不同,对应溯源表达式也有所区别,下面具体加以讨论.

2.1 来自源表的溯源表示

对于结果元组是直接来自源表的情况,根据关系操作的不同,定义结果元组的溯源表达式如下:

(1) 选择

在 $D=(R,I,L)$上,对 R 中某个关系表的一个选择查询 Q,若查询结果 W 不为空,则$\forall w\in W$,其溯源表达式为 $\lambda=t\rightarrow[i_1].k_1\wedge\ldots\wedge t\rightarrow[i_n].k_n$.其中,$t$ 是源表中的某条元组,$i_1,\ldots,i_n$ 对应选择条件涉及的不确定属性,$k_1,\ldots,k_n$ 是属性 $i_1,\ldots,i_n$ 的某个可能取值.w 的概率为源元组 t 存在的情况下,属性 $i_1,\ldots,i_n$ 取特定值的概率,即

$$P(t\rightarrow[i_1].k_1\wedge\ldots\wedge t\rightarrow[i_n].k_n)=P(t)\times(P([i_1].k_1)\times\ldots\times P([i_n].k_n)).$$

对于选择条件未涉及的属性列 h,在结果元组中仍为不确定属性,每个可能取值的概率值没有发生变化,其溯源表达式$\lambda'=t\rightarrow[h].k$,其中,$k$ 为属性 h 的某个可能取值.

(2) 投影

在 $D=(R,I,L)$上,对 R 中某个关系表上的某一属性进行投影查询 Q,若查询结果 W 不为空,则

$$\forall w\in W,t_1\rightarrow[i_1].k_1=t_2\rightarrow[i_2].k_2=\ldots=t_m\rightarrow[i_m].k_m,$$

其溯源表达式为$\lambda=t_1\rightarrow[i_1].k_1\vee\ldots\vee t_m\rightarrow[i_m].k_m$.投影操作的结果往往要执行去重操作,去重操作会涉及到结果元组的所有属性,所以最终结果转化成为一个可能的实例集合,各个属性不再具有不确定性,仅存在元组级不确定性.对相同的结果元组进行去重操作,其概率为参与合并的结果元组中至少有一个存在的概率.对于可能结果实例 w,其概率为 $P(t_1\rightarrow[i_1].k_1\vee\ldots\vee t_m\rightarrow[i_m].k_m)=1-\prod_{j=1}^{m}(1-P(t_m)\times P([i_j].k_m))$.

(3) 连接

在 $D=(R,I,L)$上,对 R 中某两个关系表的一个连接查询 Q,若查询结果 W 不为空,则$\forall w\in W$,其溯源表达式为 $\lambda=t_1\rightarrow[i_1].k_1\wedge t_2\rightarrow[i_2].k_2$.其中,$t_1,t_2$ 是做连接的两个元组.连接属性的值要相等才会对应产生结果元组,因此,结果元组的概率等于对应做连接的两个表中相应的元组存在且元组的连接属性取特定值的概率的乘积.对于可能结果实例 w,其概率为 $P(t_j\rightarrow[i].k_i\wedge t_{j'}\rightarrow[i'].k_{i'})=P(t_j)\times P([i].k_i)\times P(t_{j'})\times P([i'].k_{i'})$.

(4) 聚集

常见的聚集操作包括求和、求平均、求最值、计数等.在 $D=(R,I,L)$上,对 R 中某个关系表的一个求和、求平均、求最值操作 Q,若结果 W 不为空,则$\forall w\in W$,其溯源表达式为

$$\lambda=(t_1\rightarrow[i_1].k_1\wedge\ldots\wedge t_n\rightarrow[i_n].k_n)\wedge(\neg t_1'\wedge\ldots\wedge\neg t_n'),$$

其中,$t_1',\ldots,t_n'$ 是求和、求平均、求最值时该分组中存在但未出现的元组,$t_1,\ldots,t_n$ 则是该分组中存在且出现的元组.对于计算操作,不涉及属性级,仅判断一个元组是否存在,因此本文不考虑此种情况.

此外,对于并、交和差操作也仅涉及元组级别,因此只需进行元组级溯源,无需定义相关的溯源表达式.

2.2 来自中间表的溯源表示

若进行数据库操作的表不为源表,则表中元组 t 是一个溯源表达式.对于源表中不确定属性,若参与数据库操作,则结果元组中该属性是确定属性,因此,t 中的不确定属性来自于源表.对于 t 中的某不确定属性,其某个取值对应的溯源表达式为 $t\rightarrow(t'\rightarrow[m].n)$,$t'$是源表中的元组.

由于本文是为了实现对不确定性的溯源,当中间结果元组的属性为确定属性时,则其参与派生的结果元组的相应属性不存在溯源表达式.若结果元组 t 某属性 i 的第 j 个可能取值的溯源表达式为 $t\rightarrow[i].j=t\rightarrow(t'\rightarrow[m].n)$,则元组 t 的第 i 个属性取第 j 个可能值的概率等于元组 t'的第 m 个属性取第 n 个可能值的概率,即 $P([i].j)=P([m].n)$.所以,对于中间结果元组 t,它存在且第 i 个属性取第 j 个可能值的概率为

$$P(t\rightarrow(t'\rightarrow[m].n))=P(t\rightarrow[i].j)=P(t)\times P([i].j)=P(t)\times P([m].n).$$

例 3:对于关系代数操作$\pi_A(\sigma_{A>14}(R_1)\bowtie R_2\bowtie R_3)$,其结果元组如图 3 所示.假设 R_{12} 是$\sigma_{A>14}(R_1)\bowtie R_2$ 得到的中间表,R_{123} 是$\sigma_{A>14}(R_1)\bowtie R_2\bowtie R_3$ 得到的中间表,则 R_{12} 中属性 A 为确定属性,其参与派生的 R_{123} 中结果元组 A 属性不存在溯源表达式.对于 ID 为 33 的结果元组来说,各属性均为确定属性,其元组对应的溯源表达式为

$$\lambda=((01\rightarrow[1].1\wedge03\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)\wedge05\rightarrow[1].1)\vee((02\rightarrow[1].1\wedge03\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)\wedge05\rightarrow[1].1).$$

对于溯源表达式$(01\rightarrow[1].1\wedge03\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)$,其概率计算表达式为

$$((P(01)\times P([1].1))\times(P(03)\times P'([1].1)))\times P'([2].1)=((0.8\times0.7)\times(0.8\times0.8))\times0.7.$$

$R'_{123}(A)$

ID	A
33	15
34	18

Fig.3 Results table $R'_{123}(A)$

图 3 结果关系表 $R'_{123}(A)$

3 结果元组的概率计算

对于聚集操作,其溯源往往涉及到分组中的所有元组,对于分组中其结果元组的概率计算是#P-hard 问题[3],可以通过近似算法来降低计算复杂度.本文主要讨论涉及选择、投影及连接操作时,结果元组的溯源表达及概率计算问题.

在利用溯源表达式计算结果元组概率时,中间结果元组之间可能存在依赖关系,导致结果元组概率计算出错.因此,考虑把溯源表达式用树结构来表示,然后通过对树的转换操作来消除中间结果元组间的依赖关系,从而保证概率计算的正确性.表示溯源表达式的树型结构称为溯源树,下面给出溯源树的定义.

定义 3(溯源树). 根据溯源表达式的运算次序构建的二叉树称为溯源树,其中,

(1) 树中叶子节点为数据节点;

(2) 非叶子节点有运算节点和从属节点两种类型,运算节点包括合取操作(∧)和析取操作(∨)节点,从属节点指箭头(→)节点.

例 4:对于例 3 中的溯源表达式:

$\lambda=(05\rightarrow[1].1\wedge((03\rightarrow[1].1\wedge01\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)))\vee(05\rightarrow[1].1\wedge((03\rightarrow[1].1\wedge02\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)))$,

溯源树如图 4 所示.

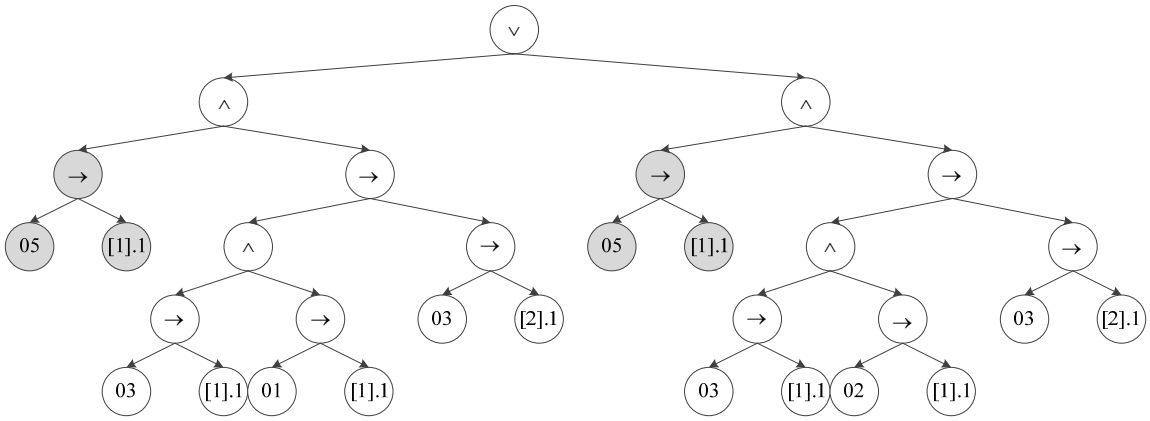


Fig.4 Lineage tree

图 4 溯源树

对于元组相互独立的表,进行连接操作得到的结果元组之间可能会有依赖关系产生.此时进行投影操作,会由于去重操作导致结果元组来自多个有依赖关系的元组.对于一棵溯源树,若其对应的关系代数中含有连接操作,同时含有投影操作,则需要对溯源树进行检查,若存在依赖关系,则执行转换操作.

在通过溯源表达式进行概率计算时,部分概率的重复计算使得结果不正确.对应到溯源树中,从根节点出发,沿不同的路径可以到达同一个属性表达式对应的子树.如图 4 中灰色节点构成的子树,对应属性表达式05→[1].1,从根节点开始沿不同的路径可以找到这样的子树,因此会存在概率的重复计算.为了避免概率的重复计算而导致结果出错,本文提出如图 5 所示的转换算法.

算法 1. *exchange_lineage*(λ).
Input: Original lineage expression λ;
Ouput: Transformed lineage expression λ'.
1: t=*disjunction_extract*(λ);
2: while (t)
3: V=*split*(t);
4: for every λ_i in V do
5: if not *atom*(λ_i) then
6: λ_i=*transform*(λ_i);
7: for every λ_i in V do
8: λ_j=*uncompare*(λ_i,V);
9: if λ_j is null then
10: continue;
11: if (*compare*(λ_i,λ_j)) then
12: λ''_{n+1}=*common_exact*(λ_i,λ_j);
13: λ'_{n+1}=*uncommon_transform*(λ_i,λ_j);
14: $\lambda_{n+1} = \lambda''_{n+1} \wedge \lambda'_{n+1}$;
15: *delete*(λ_i,λ_j);
16: if *unatom*(λ'_{n+1}) then
17: $\lambda_{n+1} = \lambda''_{n+1} \wedge$ *exchange_lineage*(λ'_{n+1});
18: *insert*(λ_{n+1});
19: else goto 9;
20: λ'=*combine*(V);
21: return λ';

Fig.5 Transform algorithm for lineage expressions

图 5 溯源表达式转换算法

算法 1 是一个结果元组溯源表达式的转换,其中,*disjunction_extract*(λ)表示深度优先遍历溯源树,遇到析取

操作节点就不再遍历其子树,以析取操作节点为根的子树,就是对应要找的析取式.如图 4 所示,溯源树根节点是析取符号,则把整棵溯源树放入 t,溯源树的子树可能还包含有析取式,算法 1 的第 17 行通过递归调用自身对其所包含的析取式进行转换.$split(t)$表示分解析取式 t,得到 t 的各析取项集合;$atom(\lambda_i)$表示λ_i 是属性表达式;$transform(\lambda_i)$表示把λ_i 箭头右边的溯源表达式与箭头左边的项中属于同一个源元组的子表达式的属性进行合取;$uncompare(\lambda_i,V)$表示找出 V 中未与λ_i 比较过的一个项;$compare(\lambda_i,\lambda_j)$表示判断$\lambda_i,\lambda_j$ 是否有公共子项,若有,则返回 ture;$common_exact(\lambda_i,\lambda_j)$表示提取$\lambda_i,\lambda_j$ 公有的子项;$uncommon_transform(\lambda_i,\lambda_j)$表示把$\lambda_i,\lambda_j$ 的非公有子项以析取符号∨相连;$delete(\lambda_i,\lambda_j)$表示从集合 V 中删除λ_i,λ_j;$unatom(\lambda'_{n+1})$ 表示 λ'_{n+1} 中包含的非公有子项有非属性表达式;$insert(\lambda_{n+1})$表示把λ_{n+1} 插入集合 V 中;$combine(V)$表示用 V 中各项替换初始的溯源表达式中对应的部分.

在利用布尔表达式计算结果元组概率时,传统方法通过避免产生有依赖关系的中间结果元组来保证结果元组概率计算的正确性.例如,元组 a,b 均与元组 c 做连接,得到 $a\wedge c$ 和 $b\wedge c$.该中间结果元组有公共的源元组 c,因此存在依赖关系.若对连接结果在某属性上进行投影去重仅得到 1 条结果元组 d,则 d 的溯源表达式为$(a\wedge c)\vee(b\wedge c)$,通过转换得到$(a\vee b)\wedge c$.即,先选择 a,b 中的一条元组,再与 c 做连接,这样得到的结果不变,同时也可保证概率计算的正确性.在本文中,溯源表达式即对应布尔表达式,属性表达式对应布尔表达式中的变量,因此在溯源表达式转换过程中,也采用这样的方法提取公共项,对应算法 1 中的第 8 行~第 14 行.但一条元组中可能多个属性参与数据库操作,如例 4 中 ID 为 33 的元组的溯源表达式$(05\rightarrow[1].1\wedge((03\rightarrow[1].1\wedge01\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)))\vee(05\rightarrow[1].1\wedge((03\rightarrow[1].1\wedge02\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)))$中,由于 ID 为 03 的源元组中的两个属性 $03\rightarrow[1].1$,$03\rightarrow[2].1$ 都参与数据库操作,对应传统方法中布尔表达式包含的同一个变量,所以在表达式转换时,需要把属于同一个元组的多个属性进行合并,式中$(03\rightarrow[1].1\wedge01\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)$转换后变为 $01\rightarrow[1].1\wedge03\rightarrow([1].1\wedge[2].1)$,该部分对应于算法 1 中的第 4 行~第 6 行.转换前的概率 $P((03\rightarrow[1].1\wedge01\rightarrow[1].1)\rightarrow(03\rightarrow[2].1))=P(03\rightarrow[1].1\wedge01\rightarrow[1].1)\times P([2].1)=(P(03)\times P([1].1))\times(P(01)\times P'([1].1))\times P([2].1)$,其中,为了区分两个不同元组第 1 个属性的第 1 个可能取值的概率,分别用 $P([1].1)$,$P'([1].1)$来表示.转换后的概率 $P(01\rightarrow[1].1\wedge03\rightarrow([1].1\wedge[2].1))=(P(01)\times P([1].1))\times(P(03)\times P'([1].1)\times P([2].1))$,与转换前的相等.

算法 1 递归地对溯源树进行转换,主要包括两个转换,分别对应算法 1 中的第 4 行~第 6 行和第 8 行~第 14 行.通过前面的分析可知,这两个转换都可以保证概率计算的正确性,因此通过算法 1 的转换,可以保证结果元组概率计算的正确性.

假定一个溯源表达式的计算代价为 $O(m)$,结果元组的规模为 n,m 是溯源表达式包含的属性表达式的个数,与元组规模 n 无关,则对于一个结果元组集合,其溯源表达式转换的时间复杂度为 $O(mn)$.在一般情况下,溯源表达式不会过于复杂,因而,当 n 较大时,随着结果元组规模的扩大,结果元组溯源表达式的转换时间呈线性增长.

为了便于结果元组的概率计算,转换后的溯源表达式均存储下来.由于转换的过程是把溯源表达式中某些公共项提取出来,所以转换后的路径表达式的存储代价不会大于转换前.假定一条元组的溯源表达式存储代价为 s,则通过算法 1 的转换,增加的存储开销不超过 $O(ns)$.

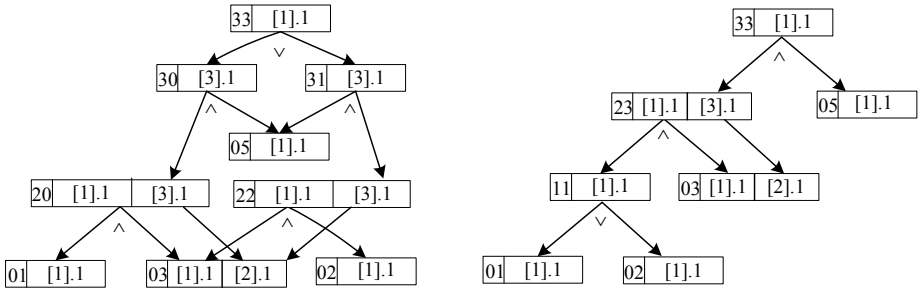
例 5:对于例 3 中的溯源表达式$\lambda=(05\rightarrow[1].1\wedge((03\rightarrow[1].1\wedge01\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)))\vee(05\rightarrow[1].1\wedge((03\rightarrow[1].1\wedge02\rightarrow[1].1)\rightarrow(03\rightarrow[2].1)))$,溯源路径图如图 6(a)所示,图中只显示出要用到的部分属性的溯源路径.可以看到,结果元组 33 的概率是由中间结果元组 30,31 的概率计算得到的,而 30,31 有共同的源属性,两个元组之间不独立,计算概率时会有部分概率重复计算,所以得到结果元组 33 的概率就会变大.根据算法 1,对溯源表达式按运算层次进行分割,得到:

$$\{(05\rightarrow[1].1\wedge(03\rightarrow[1].1\wedge01\rightarrow[1].1))\rightarrow(03\rightarrow[2].1),(05\rightarrow[1].1\wedge(03\rightarrow[1].1\wedge02\rightarrow[1].1))\rightarrow(03\rightarrow[2].1)\}.$$

对集合中各项箭头右边的溯源表达式与箭头左边的项中属于同一个源元组的子表达式的属性进行合取,得到:

$$\{05\rightarrow[1].1\wedge(01\rightarrow[1].1\wedge03\rightarrow([1].1\wedge[2].1)),05\rightarrow[1].1\wedge(02\rightarrow[1].1\wedge03\rightarrow([1].1\wedge[2].1))\}.$$

提取公因式后,得到$(05\rightarrow[1].1\wedge03\rightarrow([1].1\wedge[2].1))\wedge(01\rightarrow[1].1\vee02\rightarrow[1].1)$,其溯源路径图如图 6(b)所示.



(a) 转换前的溯源路径图 (b) 转换后的溯源路径图

Fig.6 Lineage path

图 6 溯源路径图

4 优化方案

在传统数据库中,溯源信息只用来追溯结果元组的来源;而在概率数据库中,溯源信息不仅用来对结果元组进行溯源,还要能够利用它来进行概率计算.通过算法 1 的转换后,虽然可以保证概率计算的正确性,但计算一组结果元组的概率时,总的计算开销可能会很高.在现实应用中,人们常常需要在结果表中查找概率大于某个给定阈值的元组,此时,需要利用溯源表达式实时地计算各元组的概率.由于溯源表达式可能含有相同的部分(即共享路径)而导致概率重复计算,所以本文考虑构建溯源表达式的共享路径表,从而避免共享路径概率的重复计算.若计划树是不安全的,为保证概率计算的正确,需要对溯源表达式进行转换.转换后的溯源表达式的共享路径可能会发生改变,因此把转换前后的溯源表达式都存储下来.若计划树是安全的,则无需对溯源表达式进行转换.

4.1 散列溯源表达式

为了查找溯源表达式的共享路径,首先用溯源树表示.树中属性表达式均包含在以箭头“→”为根的子树中.

例 6:例 5 中,转换后的溯源表达式为(05→[1].1∧03→([1].1∧[2].1))∧(01→[1].1∨02→[1].1),其溯源树如图 7 所示.

为了降低查找开销,首先比对各个结果表中对应的关系代数操作.当关系代数中含有相同的源表时,溯源表达式可能含有共享路径,具体分为两种情况:

(1) 关系代数表达式仅含有一个相同的源表,且对该表上同一属性都做投影操作;

(2) 关系代数表达式有多个相同的源表之间做连接操作.

对于情况(1),对一个表的某属性做投影,该属性上若有相同的属性值,则结果元组会进行去重操作,结果元组的溯源表达式中会包含析取式,因此可能会有共享路径存在;对于情况(2),若做连接的源表中相同表的个数大于两个,则可能存在共享路径.这一点与元组级溯源信息不同,对于元组级溯源信息,做连接的源表中相同表的个数大于 1 个即存在共享路径.例如,对于图 1 中的几个关系表,关系代数操作 $R_1 \bowtie R_2$,包含溯源表达式 03→[1].1∧ 01→[1].1,而对于 $R_1 \bowtie R_2 \bowtie R_3$,包含转换后的溯源表达式 03→([1].1∧[2].1)∧01→[1].1∧05→[1].1.由于 R_1,R_2 的连接结果若与其他表中元组做连接时,R_2 中会有两个属性参与连接操作,使得最终的溯源表达式与仅有 R_1,R_2 做连接所得结果元组的溯源表达式不存在共享路径.而对于传统的溯源表示方法,$R_1 \bowtie R_2$ 对应的是 03∧01, $R_1 \bowtie R_2 \bowtie R_3$ 对应的是 03∧01∧05,存在共享路径 03∧01.

在查找共享路径时,本文借鉴 Trie 树[21]的思想,把合取式用类似 Trie 树的形式表示.因为 Trie 树表示字符串时,字符串中各字符之间是有序的,溯源表达式中的合取式对应于连接操作,表在做连接时也是有一定顺序的,所以考虑把合取式中各项散列到不同的桶中,并按照合取式自左到右的顺序,用指针把这些桶连接起来,形成一个树型结构.而溯源表达式中的析取式对应于投影去重操作,析取式中各项之间没有固定顺序,故无法用类似

Trie 树的形式表示.下面给出桶的前驱和后继的定义.

定义 4(桶的前驱/后继). 对于指针 *l* 所指向的桶 *a*,*l* 所关联的桶 *b* 称为桶 *a* 的前驱,桶 *a* 称为桶 *b* 的后继.

与 Trie 树不同的是,该树型结构中每个节点包含的不是一个字符,而是一个属性表达式或属性表达式的析取式.除此之外,根节点不为空,也包含一个属性表达式或属性表达式的析取式.向桶中散列时,对于析取式的各析取项,若包含合取式,则对合取式进行散列.散列时,根据溯源表达式对应的溯源树,按照深度优先遍历算法把各项依次散列到相应的桶中,并标记该项对应的结果元组 ID.在散列过程中,根据散列的顺序把这些桶用指针依次连接起来.散列溯源表达式的算法如图 8 所示.

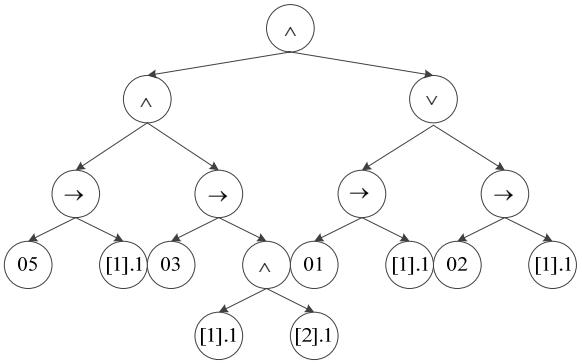


Fig.7 The lineage tree of the lineage expression

图 7 溯源表达式对应的溯源树

算法 2. *depthfirst_hash*(*u*).
Input: lineage tree *u*;
1: *p*=*node*(*u*).
2: if (*conjunctive*(*p*) && *left_atom*(*p*))
3: *hash*(*p.left*);
4: else if (*conjunctive*(*p*) && !*left_atom*(*p*))
5: *depthfirst_hash*(*p.left*);
6: if (*conjunctive*(*p*) && *right_atom*(*p*))
7: *hash*(*p.right*);
8: else if (*conjunctive*(*p*) && !*right_atom*(*p*))
9: *depthfirst_hash*(*p.right*);
10: if (*disjunctive*(*p*) && *atom_all*(*p*))
11: *hash_all*(p_i);

Fig.8 Hashing the lineage expression

图 8 散列溯源表达式

在算法 2 中,*node*(*u*)表示取当前溯源树 *u* 的根节点,*conjunctive*(*p*)表示 *p* 对应的是合取操作,*left_atom*(*p*)表示以 *p* 的左孩子节点为根的子树是属性表达式,*hash*(*p.left*)表示把 *p* 的左孩子中对应的属性表达式散列到相应的桶中,*disjunctive*(*p*)表示 *p* 对应的是析取操作,*atom_all*(*p*)表示以 *p* 的孩子节点为根的子树中均为属性表达式,*hash_all*(*p*)表示把以 *p* 为根的子树对应的溯源表达式散列到相应的桶中.

算法 2 通过深度优先遍历来进行散列,其时间复杂度与溯源树本身的规模相关,而溯源树的规模与溯源表达式包含的属性表达式个数 *m* 相关.对于包含 *n* 个结果元组的集合,其溯源表达式散列的时间复杂度为 *O*(*mn*).在一般情况下,溯源表达式不会过于复杂,因而,当 *n* 较大时,随着结果元组规模的扩大,结果元组溯源表达式的转换时间呈线性增长.

在最坏情况下,溯源表达式没有共享路径且均为合取式,此时,对于溯源表达式中每一个属性表达式均需要一个桶.假设一个溯源表达式中包含的属性表达式个数为 *m*,对于 *n* 个结果元组的集合,其空间复杂度为 *O*(*mn*).

例 7:把例 6 中的溯源树进行散列,散列结果如图 9 所示.

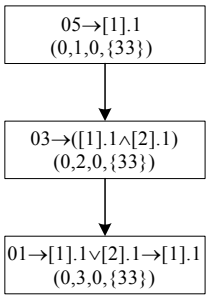


Fig.9 The hash result of a lineage expression

图 9 溯源表达式的散列结果

在图 9 中,矩形代表一个桶,桶中包含两项:

第 1 项表示散列到该桶的溯源表达式;

第 2 项是一个四元组,四元组中前两项分别是该桶所在树中从根节点开始到该节点对应的溯源表达式被共享的次数和共享的路径长度;

第 3 项是一个标志位,表明该项是否被写入共享路径表中;

第 4 项为包含该溯源表达式的结果元组 ID 的集合.

4.2 构建共享路径表

散列完所有的溯源表达式后,构建共享路径表,表中记录了结果元组 ID、共享路径的概率值、共享路径长度、路径共享次数.对于同一个结果元组,它对应的溯源表达式可能有多条共享路径.由于对溯源表达式散列是按照其运算次序进行的,此时若存在多个共享路径,它们对应的溯源树的子树集合属于包含关系.因此在进行概率计算时,先计算较短的共享路径的概率值,再利用计算结果计算较长路径的概率值.构建共享路径表的算法如图 10 所示.

算法 3. *depthfirst_sharepath*(*v*).
Input: bucket *v* without prior bucket;
1: *q*=*node*(*v*).
2: if (*q.count*!=0 && *q.flag*!=0)
3: *save(q)*;
4: *q.flag*=0;
5: *depthfirst_sharepath*(*v.nextchild*);
6: else
7: *depthfirst_sharepath*(*v.parent.nextchild*);

Fig.10 Constructing a share path table

图 10 构建共享路径表

在算法 3 中,*q.count* 表示相应溯源表达式的共享次数,*q.flag* 表示相应溯源表达式相关信息是否被写入共享路径表中,*save*(*q*)表示把桶 *q* 对应的相关信息存入共享路径表中,*v.nextchild* 表示某个未被处理过的桶 *v* 的后继,*v.parent* 表示桶 *v* 的前驱.在散列溯源表达式时,每个被散列的桶的标志位均被置为 0(这里未在算法 2 中详细写出).写入共享路径表后,标志位被置为 1.这样,再有更新时,可以增量地把更新的共享路径信息写入共享路径表.

算法 3 是扫描一个没有前驱的桶的过程,由算法 2 的空间代价分析可知,其最大空间复杂度为 $O(mn)$,故扫描这些散列的桶来构建共享路径表的最大时间复杂度不会超过 $O(mn)$.

若构建共享路径表,则每次计算结果元组概率时都要先扫描共享路径表,查找是否有共享路径存在.因为查找共享路径也要消耗一定的时间,所以并不是任何情况下都会用到共享路径.对于一组要计算概率的元组,其总的计算开销与含有共享路径的溯源表达式的比例以及共享路径的长度有关.下面给出共享路径长度的定义.

定义 5(共享路径长度). 共享路径中包含的合取或析取操作符的总个数称为共享路径长度.

对于 n 个结果元组,假定共有 n' 条共享路径,每个结果元组概率的计算开销为 $cost(t)$,路径共享次数为 $count$,长度为 l 的共享路径概率计算开销为 $cost(l)$,扫描共享路径表的时间开销为 $cost(table)$,则利用共享路径计算这 n 个元组概率的总开销为

$$cost=\sum_{i=1}^{n}cost(t_i)+\sum_{i=1}^{n}cost(table_i)-\sum_{j=1}^{n'}count_j\times cost(l_j) \tag{1}$$

在利用共享路径计算概率时,若要使得计算开销有所降低,需满足扫描共享路径表的总开销小于利用共享路径节约的概率计算开销.由公式(1)可知,在一组溯源表达式中,与未使用共享路径的概率计算开销相比,若要使得计算效率提升,需满足 $\sum_{i=1}^{n}cost(table_i)-\sum_{j=1}^{n'}count_j\times cost(l_j)<0$. 而当 $n>>n'$ 时,可能会导致优化后的概率计算代价大于优化前的概率计算代价.

基于上述分析,可以通过公式(1)来确定是否利用共享路径来计算概率.同时,当有新的结果元组产生时,无

需即时更新索引,当新增元组导致利用共享路径计算结果元组概率的效率低于某个阈值δ时,再执行更新索引的操作.

4.3 原子析取式预计算

在利用共享路径加快概率计算时,对于溯源表达式是析取式的情况,根据第 4.1 节中提到的,若各析取项均为属性表达式,则直接把该溯源表达式散列到一个桶中.下面首先给出原子析取式的定义.

定义 6(原子析取式). 各析取项均为属性表达式的析取式称为原子析取式.

在图 9 中,没有后继的桶中包含项 01→[1].1∨02→[1].1,它是一个原子析取式,不对其进行散列查找共享路径.然而,原子析取式的概率计算代价相对较大,假设 $cost(\times)$表示乘法操作的时间代价,$cost(-)$表示减法操作的时间代价,$cost(r)$表示从数据库中取出属性或元组概率值的时间代价.对一个原子析取式 $x\vee y$,在求概率 $P(x\vee y)=1-(1-P(x))\times(1-P(y))$时,步骤如下:

(1) 2 次取数据操作,得到 $P(x),P(y)$的概率;

(2) 2 次减法操作,计算 $1-P(x),1-P(y)$的值;

(3) 1 次乘法操作,计算$(1-P(x))\times(1-P(y))$的值;

(4) 1 次减法操作,计算 $1-(1-P(x))\times(1-P(y))$的值.

计算 $P(x\vee y)$总的时间开销为 $2\times cost(r)+3\times cost(-)+cost(\times)$.对于 $x\wedge y$ 来说,在求概率 $P(x\wedge y)=P(x)\times P(y)$时,仅需要 2 次取数据操作,得到 $P(x),P(y)$的概率,然后执行 1 次乘法操作,总的时间开销为 $2\times cost(r)+cost(\times)$.对于有 n 个项的析取式 $x_1\vee\ldots\vee x_n$,其概率计算的时间开销为 $n\times cost(r)+(n+1)\times cost(-)+(n-1)\times cost(\times)$.而有 n 个项的合取式 $x_1\wedge\ldots\wedge x_n$,其概率计算的时间开销为 $n\times cost(r)+(n-1)\times cost(\times)$.由此可见,在其他条件相同的情况下,相同项的析取式概率计算比合取式的时间开销多了$(n+1)\times cost(-)$.

基于此,预计算原子析取式的概率值.通过修改算法 2,在第 10 步判断条件之后,增加计算原子析取式概率的步骤.在溯源树中,原子析取式对应子树的根节点为析取操作节点,在此节点中记录原子析取式对应的概率值.

5 实验评估

元组级溯源信息可以追溯不确定性产生的来源,同时又可用来计算结果元组的概率值.然而,该溯源信息只能在元组级别追溯不确定性,并不能准确定位不确定性产生的具体位置.本文提出的属性级溯源方法可以准确地追溯产生不确定性的具体属性,同时通过算法 1 可以保证其结果元组概率值的正确计算.对于元组级溯源方法和优化前后的属性级溯源方法,将从结果元组概率计算的时间代价和总的空间代价两方面进行比较,以表明本文方法的有效性和可行性.该部分将在第 5.1 节中详细加以介绍.此外,为了验证第 4.2 节中对利用共享路径进行概率计算的时间开销分析的正确性,第 5.2 节将会对影响时间开销的各个因素分别进行测试.

实验环境如下:Intel(R) Core(TM) i5-2320 3.00GHz 处理器,4GB 内存,Windows 7 旗舰版 64 位操作系统,使用 postgreSQL 9.1.3 存储不确定性数据,使用 Java 语言编写程序.

本文实验的数据集基于 TPC-H 基准数据集[22]中的 Customer 表、Lineitems 表以及 Orders 表.在实验中,对这 3 个表进行分割,仅取出所需的属性列,并随机生成属性级、元组级的概率值.实验中用到的实验数据见表 1 和表 2.

Table 1 Parameters of the optimized scheme

表 1 优化方案的参数

评估方案	Customer 表	Orders 表	Lineitems 表
总体优化方案	[1000,5000]	[1000,5000]	[1000,5000]

Table 2 Parameters of share paths

表 2 共享路径测试的参数

评估方案	C_O_L 表	C_O_L_5 表	路径共享次数	共享路径数	共享路径长度
共享路径-1	20 000	—	1	[10,10000]	1
共享路径-2	20 000	—	[10,10000]	1	1
共享路径-3	0	20 000	1	10 000	(1,15)
共享路径-4	[2000,10000]	—	1	[1000,5000]	1

在表 2 中,C_O 表是 Customer 表,Orders 表是做连接时得到的表,C_O_L 表是 Customer 表、Lineitems 表、Orders 表做连接时得到的表,C_O_L_5 表是 C_O_L 表做 0~4 次自连接时得到的表.共享路径-1 对应的是其他参数不变、共享路径数不同对概率计算效率影响的实验.共享路径-2 对应的是其他参数不变、路径共享次数不同对概率计算效率影响的实验.共享路径-3 对应的是其他参数不变、共享路径长度不同对概率计算效率影响的实验.共享路径-4 对应的是路径共享次数和共享路径长度不变、不同规模的结果元组对概率计算效率影响的实验.由第 4.2 节的代价分析可知,含有共享路径的溯源表达式的比例会对使用共享路径后的计算效率产生影响.为此,当元组规模发生变化时,改变共享路径数量 n',使得 n'与 n 之间的比例保持不变.

5.1 优化方案评估

对于表 1 中的 3 张表,分别标注各元组中的不确定属性,利用属性级溯源信息对选择、投影和连接得到的结果元组进行标注.为了验证利用属性级溯源信息计算结果元组概率的有效性,我们将其时间开销与传统的元组级溯源信息计算结果元组概率的时间开销进行比对.通过把表 1 中的 3 张含有不确定属性值的源表用可能世界模型建模,对于每个可能世界实例中的元组,其属性值都是确定值.对每条源元组进行标注,通过选择、投影和连接得到的结果元组的溯源信息则为元组级溯源信息.对于属性级溯源信息,通过构建共享路径表,在构建过程中对原子析取式进行预计算,以比较优化前后概率计算的时间开销.元组级溯源方法以及优化前后的属性级溯源方法的概率计算开销如图 11 所示.

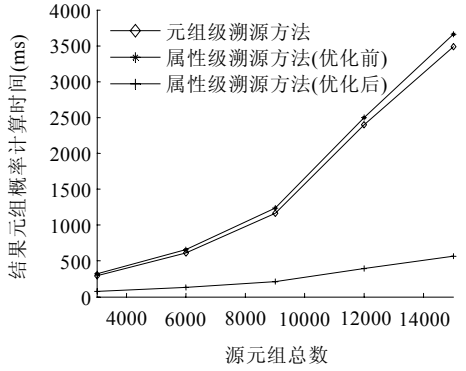


Fig.11 Comparison on probability computation overheads of result tuples

图 11 结果元组概率计算开销比对

图 11 是 Customer,Lineitems 和 Orders 这 3 张表做选择、连接和投影操作得到的结果元组在优化前后的概率计算开销对比.从图中可以看到,利用元组级溯源方法和未优化的属性级溯源方法进行概率计算的开销相差不大.这是因为,在计算概率时,其通过数据库操作读取概率值的次数是一样的,而概率值的计算相对于读取概率值来说代价较小.当结果元组数较少时,这部分开销可以忽略不计.例如,属性级溯源信息(01→[1].1∧03→[1].1)→(03→[2].1)∧05→[1].1 需要 3 次数据库扫描操作,分别找到 01,03 和 05 这 3 条元组,并返回其相应属性和元组的概率值.而相对应的元组级溯源信息 01∧03∧05,也需要 3 次数据库扫描操作,分别找到 01,03 和 05 这 3 条元组,并返回其相应元组的概率值.在利用属性级溯源信息进行概率计算时,同时涉及属性概率和元组概率,

相比利用元组级溯源信息计算概率的代价略大.随着源元组总数的增加,结果元组数也相应地增加,从而使得溯源表达式的个数也随之增加.因此,随着源元组总数的增加,利用属性级、元组级溯源信息进行概率计算的时间开销的差值也随之增大.对于属性级溯源方法,优化后的概率计算开销大为降低,且随着溯源表达式个数的增多,优化前后的概率计算开销差值也越大.这是因为,优化后的方案减少了从数据库读取概率值的次数,因而极大地提高了计算效率.

图 12 中,随着溯源表达式数量的增多,属性级溯源方法与元组级溯源方法的存储代价之差也越来越大.由于属性级溯源信息相较于元组级溯源信息的粒度更细,因此其存储代价相对较大.当通过优化方案计算结果元组概率时,需要把溯源表达式散列到桶中,利用桶中相关信息构建共享路径表,在加快概率计算的同时必然造成存储开销的增长.由图 12 可以看到,采用优化方案后,其存储代价大为增加.

图 13 是属性级溯源方法在采用优化方案前后的存储代价与元组级溯源方法存储代价的差值变化.由图 13 可以看出,随着源元组总数的增加,未采用优化方案增加的存储开销呈线性增长且增幅较小.采用优化方案后,增加的存储开销增幅较大但也近似呈线性增长.采用优化方案后,增加的存储开销之所以近似地呈线性增长,是因为一个包含 m 个属性表达式的溯源表达式,它将被散列到 m 个桶中.而对于不同溯源表达式包含的相同共享路径,将散列到相同的桶中,无需创建新桶.对于 n 个溯源表达式,若其包含的属性表达式的个数最多为 m,则在最坏情况下,即不存在共享路径且每个溯源表达式均包含 m 个属性表达式时,其存储代价的差值会随结果元组规模的扩大而同比例地增长.

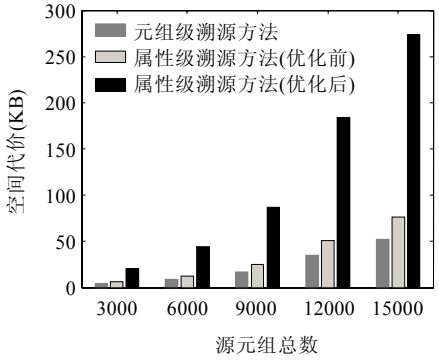


Fig.12 Comparison on storage overheads

图 12 存储代价比对

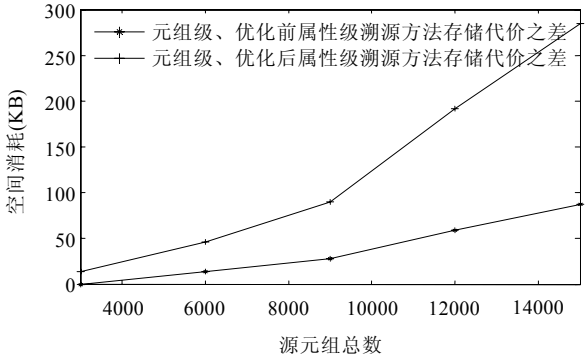


Fig.13 Increased storage overhead after optimizing

图 13 优化后增加的存储开销

综上所述,未优化的属性级溯源方法在计算结果元组概率时并不会造成计算开销过多地增长,其存储开销也近似呈线性增加.因此,利用属性级溯源信息构建溯源表达式是可行的.为了加快结果元组概率的计算,本文提出的优化方案通过牺牲存储代价以换取计算效率的极大提高.虽然采用优化方案后其存储开销极大增加,但相对于元组级溯源方法,其增加的存储开销增长幅度接近线性增长,因而该优化方案是可行而有效的.

5.2 共享路径测试

由第 4.2 节的代价分析可知,利用共享路径并不一定能够提高计算效率.在实际计算过程中,根据公式(1)来决定是否使用共享路径.

本节主要通过实验来验证其代价分析的正确性.由于含有共享路径的溯源表达式占全部溯源表达式的比例和共享路径长度对于概率计算的时间开销都有影响,而对包含共享路径的溯源表达式占全部溯源表达式的比例产生影响的因素有两个:共享路径数量和路径共享次数.针对这 3 个因素,我们分别进行实验测试.

首先,测试共享路径数量对其概率计算开销的影响.图 14 是表 2 中的第 1 个评估方案,即共享路径-1.它是在其他参数不变的情况下,测试共享路径数量对概率计算时间开销的影响.从图中可以看出:当共享路径数量过少时,利用共享路径计算结果元组概率的时间开销反而高于未优化的方法.随着共享路径数量的增多,其时间开销

逐渐降低.当共享路径有 5 000 个时,优化后的属性级溯源方法要优于元组级溯源方法.

其次,测试路径共享次数对其概率计算开销的影响.图 15 是表 2 中的第 2 个评估方案,即共享路径-2.在其他参数不变的情况下,当路径共享次数过少时,其优化后概率计算的时间开销大于优化前.随着路径共享次数的增多,其时间开销逐渐降低.当路径共享次数为结果元组总数的一半时,优化后的属性级溯源方法的概率计算时间开销远低于优化前,同时也比元组级溯源方法要高效.

最后,测试共享路径长度对其概率计算开销的影响.图 16 是表 2 中的第 3 个评估方案,即共享路径-3.对于 Customer 表、Lineitems 表、Orders 表多次做连接,得到多个不同的结果表.这些表之间元组的共享路径长度不同,测试优化前后属性级溯源方法以及元组级溯源方法的概率计算开销.从图 16 可以看出,随着共享路径长度的增加,利用共享路径进行概率计算所节约的时间开销就越多.

此外,当路径共享次数和共享路径长度不变时,测试不同规模的结果元组概率计算开销.其中,共享路径数量随结果元组总数的增加而增加,使得两者之间的比例不变.图 17 对应的是表 2 中的第 4 个评估方案,即共享路径-4.从图中可以看出,随着结果元组规模的扩大,利用共享路径进行概率计算所节约的时间开销就越多.

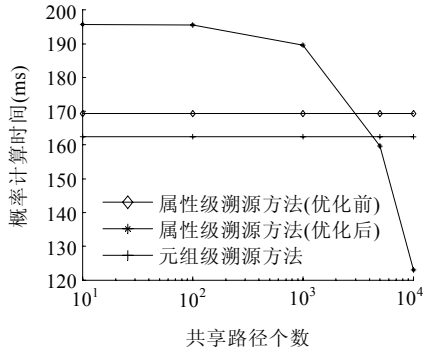


Fig.14 Probability computation overheads for different number of share paths

图 14 不同共享路径数的概率计算开销

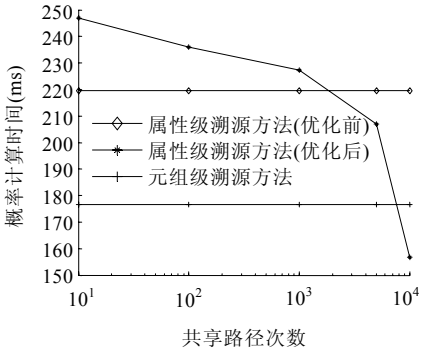


Fig.15 Probability computation overheads for different sharing number of paths

图 15 不同路径共享次数的概率计算开销

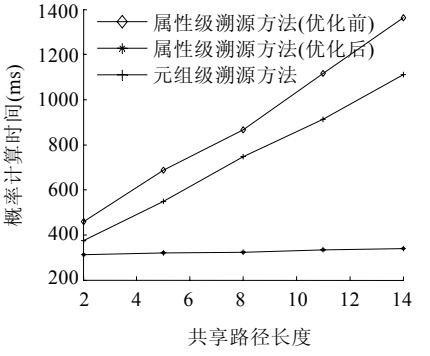


Fig.16 Probability computation overheads for different lengths of share paths

图 16 不同共享路径长度的概率计算开销

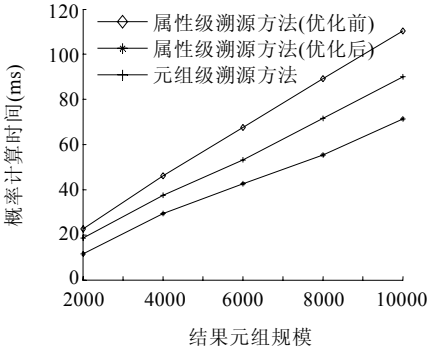


Fig.17 Probability computation overheads for different sharing number of tuples

图 17 不同元组规模的概率计算开销

6 总结与展望

在对不确定性关系数据进行溯源时,现有的元组级溯源信息表示方法在某些情况下不能准确地追溯数据

的来源,而对元组单一属性进行溯源的方法又存在一定的局限性.基于此,本文定义属性表达式,使其可以反映元组和属性之间的包含关系,并利用属性表达式构造溯源表达式.通过溯源表达式,可以追溯到产生结果元组的源属性.对于不确定性关系数据,溯源信息不仅用来追溯数据起源,还用来计算结果元组的概率.本文提出溯源表达式的转换算法,对元组中多个属性存在不确定性的情况,可以正确计算结果元组的概率.文中通过分析影响结果元组概率计算的因素,提出通过构建共享路径表的方法,并在构建过程中对原子析取式进行概率预计算,来提高概率计算的效率.第 5.1 节把本文提出的属性级溯源信息表示方法与传统的元组级溯源信息表示方法在时间代价和空间代价方面进行对比,验证了本文方法的有效性和可行性.第 5.2 节通过改变影响概率计算效率的参数,来测试不同情况下利用共享路径对概率计算效率的影响,验证了第 4.2 节中指出的计算结果元组概率的代价分析的正确性.

由于属性级溯源信息比元组级溯源信息的粒度更细,因此其存储开销必然较大.同时,本文仅考虑元组相互独立的情况,当元组之间存在关联关系时,其概率计算的复杂性将大为增加.而现实生活中,很多的信息存在关联关系,在利用元组表达时需要考虑元组之间的关联关系.例如:同一车牌的车不可能同时出现在两个地方;某篇论文的第一作者为学生时,其合作作者中很可能包含该学生的导师等.此外,在进行溯源信息的查询时,当溯源路径过长时,不能直观地看到数据的整个演化过程.基于此,未来的工作主要集中在 3 个方面:一是对于存在依赖关系的源元组,如何利用溯源信息来计算其结果元组的概率;二是如何对溯源信息进行压缩存储,在保证概率计算时间可接受的范围内,尽可能地降低存储代价;三是如何可视化显示溯源信息,对于不同的结果元组,若溯源信息存在共享路径,则突出显示其公共部分.

致谢 在此,我们向武汉大学珞珈图腾数据库实验室的同学和老师表示感谢.

References:

[1] Zhou AY, Jin CQ, Wang GR, Li JZ. A survey on the management of uncertain data. Chinese Journal of Computers, 2009,32(1): 1−16. [doi: 10.3724/SP.J.1016.2009.00001]

[2] Cavallo R, Pittarelli M. The theory of probabilistic databases. In: Stocker PM, Kent W, Hammersley P, eds. Proc. of the Very Large Data Bases. Brighton: Morgan Kaufmann Publishers, 1987. 71−81.

[3] Dalvi N, Suciu D. Efficient query evaluation on probabilistic database. In: Nascimento MA, Özsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB, eds. Proc. of the Very Large Data Bases. Toronto: VLDB Endowment, 2004. 864−875. [doi: 10.1007/s00778-006-0004-3]

[4] Benjelloun O, Sarma AD, Halevy A, Theobald M, Widom J. Databases with uncertainty and lineage. The Int'l Journal on Very Large Data Bases, 2008,17(2):243−264. [doi: 10.1007/s00778-007-0080-z]

[5] Simmhan YL, Plale B, Gannon D. A survey of data provenance in e-science. ACM SIGMOD Record, 2005,34(3):31−36. [doi: 10.1145/1084805.1084812]

[6] Sarma AD, Benjelloun O, Halevy A, Widom J. Working models for uncertain data. In: Liu L, Reuter A, Whang KY, Zhang J, eds. Proc. of the Int'l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006. [doi: 10.1109/ICDE.2006.174]

[7] Sarma AD, Theobald M, Widom J. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In: Alonso G, Blakeley JA, Chen ALP, eds. Proc. of the Int'l Conf. on Data Engineering. Cancún: IEEE Computer Society, 2008. 1023−1032. [doi: 10.1109/ICDE.2008.4497511]

[8] Gao M, He X, Jin CQ, Wang XL, Zhou AY. Recording how-provenance on probabilistic databases. In: Han WS, Srivastava D, Yu G, Yu H, Huang ZH, eds. Proc. of the Asia-Pacific Web Conf. Busan: IEEE Computer Society, 2010. 205−211. [doi: 10.1109/APWeb.2010.19]

[9] Ré C, Suciu D. Approximate lineage for probabilistic database. In: Jonker W, Petkovic M, eds. Proc. of the Very Large Data Bases. Auckland: VLDB Endowment, 2008. 797−808.

[10] Kanagal B, Deshpande A. Lineage processing over correlated probabilistic databases. In: Elmagarmid AK, Agrawal D, eds. Proc. of the Special Interest Group on Management of Data. Indianapolis: Association for Computing Machinery, 2010. 675−686. [doi: 10.1145/1807167.1807241]

[11] Amsterdamer Y, Deutch D, Milo T, Tannen V. On provenance minimization. In: Lenzerini M, Schwentick T, eds. Proc. of the Principles of Database Systems. Athens: Association for Computing Machinery, 2011. 141−152. [doi: 10.1145/1989284.1989303]

[12] Buneman P, Khanna S, Tan WC. Why and where: A characterization of data provenance. In: Bussche JV, Vianu V, eds. Proc. of the Int'l Conf. on Database Theory. London, Berlin, Heidelberg: Springer-Verlag, 2001. 316−330. [doi: 10.1007/3-540-44503-X_20]

[13] Yue K, Liu WY, Zhu YL, Zhang W. A probabilistic-graphical-model based approach for representing lineages in uncertain data. Chinese Journal of Computers, 2011,34(10):1897−1906. [doi: 10.3724/SP.J.1016.2011.01897]

[14] Fuhr N, Rölleke T. A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. on Information Systems, 1997,15(1):32−66. [doi: 10.1145/239041.239045]

[15] Lakshmanan LVS, Leone N, Ross R, Subrahmanian VS. Probview: A flexible probabilistic database system. ACM Trans. on Database Systems, 1997,22(3):419−469. [doi: 10.1145/261124.261131]

[16] Green TJ, Tannen V. Models for incomplete and probabilistic information. In: Ioannidis YE, Scholl MH, Schmidt JW, Matthes F, Hatzopoulos M, Böhm K, Kemper A, Grust T, Böhm C, eds. Proc. of the Extending Database Technology. Berlin, Heidelberg: Springer-Verlag, 2006. 278−296. [doi: 10.1007/11896548_24]

[17] Sen P, Deshpande A, Getoor L. PRDB: Managing and exploiting rich correlations in probabilistic databases. The Int'l Journal on Very Large Data Bases, 2009,18(5):1065−1090. [doi: 10.1007/s00778-009-0153-2]

[18] Gao M, Jin CQ, Wang XL, Tian XX, Zhou AY. A survey on management of data provenance. Chinese Journal of Computers, 2010, 33(3):373−389. [doi: 10.3724/SP.J.1016.2010.00373]

[19] Glavic B, Dittrich K. Data provenance: A categorization of existing approaches. In: Kemper A, Schöning H, Rose T, Jarke M, Seidl T, Quix C, Brochhaus C, eds. Proc. of the 12th Symp. of the German Informatics Society Section "Databases and Information Systems" (DBIS) on Database Systems in Business, Technology and Web. Aachen: Gesellschaft für Informatik, 2007. 227−241.

[20] Woodruff A, Stonebraker M. Supporting fine-grained data lineage in a database visualization environment. In: Gray WA, Larson PÅ, eds. Proc. of the Int'l Conf. on Data Engineering. Birmingham: IEEE Computer Society, 1997. 91−102. [doi: 10.1109/ICDE.1997.581742]

[21] Bentley JL, Sleator DD, Tarjan RE, Wei VK. A locally adaptive data compression scheme. Communications of the ACM, 1986, 29(4):320−330. [doi: 10.1145/5684.5688]

[22] Transaction Processing Council (TPC). TPC benchmark H: Standard specification, 2012. http://www.tpc.org/tpch

附中文参考文献:

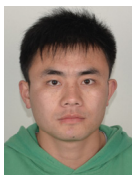
[1] 周傲英,金澈清,王国仁,李建中.不确定性数据管理技术研究综述.计算机学报,2009,32(1):1−16. [doi: 10.3724/SP.J.1016.2009.00001]

[13] 岳昆,刘惟一,朱运磊,张伟.一种基于概率图模型的不确定性数据世系表示方法.计算机学报,2011,34(10):1897−1906. [doi: 10.3724/SP.J.1016.2011.01897]

[18] 高明,金澈清,王晓玲,田秀霞,周傲英.数据世系管理技术研究综述.计算机学报,2010,33(3):373−389. [doi: 10.3724/SP.J.1016.2010.00373]

王梁(1988－),男,河南南阳人,博士生,主要研究领域为数据质量,数据溯源,不确定性数据管理.
E-mail: nywl@whu.edu.cn

王黎维(1980－),女,博士,副教授,CCF 会员,主要研究领域为数据质量,数据溯源,科学工作流.
E-mail: liwei.wang@whu.edu.cn



周光焱(1988－),男,学士,主要研究领域为数据库.
E-mail: zoegintama@gmail.com

彭智勇(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为复杂数据管理,Web 数据管理,可信数据管理.
E-mail: peng@whu.edu.cn