华南理工大学

本科毕业设计（论文）翻译

英文原文名 Efficient Processing of Top-k Queries in Uncertain Databases

中文译名 高效处理不确定性数据库的 Top-k 查询

班　　级 软件 06 级 4 班
姓　　名 区钺坚
学　　号 200630555196
指导教师 杜　　卿
填表日期 2010 年 5 月 6 日

二〇一〇年五月

英文原文版出处： Florida State University, Tech. Rep.,2008

译文成绩：＿＿＿＿＿＿　　　　指导教师签名：＿＿＿＿＿＿

译文：

摘要

本文介绍了一种新颖的多项式时间的不确定性数据 Top-k 查询，所用的模型是被广泛应用的 x-relation 模型。每个 x-relation 包含许多 x-tuple，每个 x-tuple 可以包含一个或者多个元组，但实例化的时候最多出现一个元组。就时间和空间的开销而言，我们的研究结果比很多著名的基于不确定数据库的算法要优秀。对于单一选择的情形，我们的算法快几个数量级。

Ⅰ 引言

由于许多应用都要求对不确定性或者模糊数据进行管理，因此不确定性数据库得到了许多关注。这样的应用例子有：数据集成、数据清洗、移动物体和传感器数据管理。

a)不确定性数据模型：在 TRIO[1]系统，一个不确定性数据集，或者叫做 x 关系(x-relation)，包含许许多多的 x 元组(x-tuple)。每一个 x 元组包含了很多个带有概率的可选物，这个概率就代表被选中的可选物的离散概率分布。x 元之间假设是互相独立的。在这一篇论文当中，我们同样是采用 x 关系模型的，我们增加一个用于排名元组的分数属性。更加精确的说，每一个元组 t 都包含四个部分：一个唯一的标识符 id(t)，一个分数 s(t)，一个代表 t 出现在数据库中概率的置信度 p(t)以及其他属性 A(t)。一个 x 元组 T 就是有限的元组的集合，所受的约束为 $\sum_{t_i \in T} p(t_i) \leq 1$。这些 ti 就叫做 T 的可选物（alternative）。一个 x 元组代表了 T 在随机实例化数据库中可能值的离散概率分布，例如，对于 i=1,...,|T|,T 取 ti 的概率为 p(ti)，或者不出现，不出现的概率为 $1-\sum_{i=1}^{d} p(t_i)$。我们定义不确定性数据库 Ɗ 为两两不相交的 x 元组的集合。我们使用 D 来表示 Ɗ 中的所有元组的集合，令 $|D| = \sum_{T \in D} |T| = N$。为了不失一般性，我们假设所有在 D 中的分数都是不同的。

一个不确定性数据库 D 被实例化为可能世界，假设 x 元组[1]是相互独立的，令 W 为所有可能世界的集合。因此，D 使用一种简洁的方式来表示在上 W 的概率分布。请看图 1 表示的例子。

tuples	$s(t)$	$p(t)$
t_1	100	0.5
t_2	92	0.4
t_3	80	0.6
t_4	70	0.3

x-tuples	
τ_1	$\{t_1, t_4\}$
τ_2	$\{t_2\}$
τ_3	$\{t_3\}$

world W	$\Pr[W]$
$\emptyset$	$(1-p(t_1)-p(t_4))(1-p(t_2))(1-p(t_3)) = .048$
$\{t_1\}$	$p(t_1)(1-p(t_2))(1-p(t_3)) = .12$
$\{t_2\}$	$p(t_2)(1-p(t_1)-p(t_4))(1-p(t_3)) = .032$
$\{t_3\}$	$p(t_3)(1-p(t_1)-p(t_4))(1-p(t_2)) = .072$
$\{t_4\}$	$p(t_4)(1-p(t_2))(1-p(t_3)) = .072$
$\{t_1, t_2\}$	$p(t_1)p(t_2)(1-p(t_3)) = .08$
$\{t_2, t_4\}$	$p(t_2)p(t_4)(1-p(t_3)) = .048$
$\{t_1, t_3\}$	$p(t_1)p(t_3)(1-p(t_2)) = .18$
$\{t_3, t_4\}$	$p(t_3)p(t_4)(1-p(t_2)) = .108$
$\{t_2, t_3\}$	$p(t_2)p(t_3)(1-p(t_1)-p(t_4)) = .048$
$\{t_1, t_2, t_3\}$	$p(t_1)p(t_2)p(t_3) = .12$
$\{t_2, t_3, t_4\}$	$p(t_2)p(t_3)p(t_4) = .072$

图 1 一个不确定数据库和它的所有可能世界的例子

我们区分两种情形。在单可选的情形（例如，x 元组 T2）中，每个 x 元组只有一个可选物；在多选情形（例如，x 元组 T1），有多于一个可选物给 x 元组选择。

b）在不确定数据库在的 Top-k 查询：本论文在不确定数据集合下研究查询处理的问题，并且特别的我们关注在[2]定义的 Top-k 查询。

定义 1 （不确定 Top-k 查询（U-Topk））：令 $\mathcal{D}$ 为一个不确定数据库，它的可能世界实例空间为 $\mathcal{W}$。对于任意的 W∈$\mathcal{W}$，如果|W|<k，定义 Ψ(W) = ϕ。令 T 为任意 k 元组集合。对于一个在 $\mathcal{D}$ 上的 U-Topk 查询的结果为 T*，$T^* = \text{argmax}_{T} \sum_{W \in \mathcal{W}, \Psi(W)=T} \Pr[W]$。结可以被任意解开。

对于图 1 中的例子，U-Top2 查询结果是{t1，t2}，它的概率为 0.08+0.12=0.2，由可能世界{t1,t2}和{t1,t2,t3}贡献。

定义 2 （不确定 k-Ranks 查询（U-kRanks））：令 $\mathcal{D}$ 为一个不确定数据库，它的可能世界实例空间为 $\mathcal{W}$。对于任意的 W∈$\mathcal{W}$，令 ψ_i(W) 为得分排在第 i 的元组，$1 \le i \le |W|$。在 $\mathcal{D}$ 上的 U-kRanks 查询就是一个向量 ($t_1^*,\ldots,t_k^*$)，这里 $t_i^* = \arg\max_t \sum_{W \in \mathcal{W}, \psi_i(W)=t} \Pr[W]$，i=1,...,k。结可以被任意解开。

对于图 1 中的例子，U-2Ranks 查询结果为(t1,t3):t1 排名第一的概率为 0.12+0.08+0.18+0.12=0.5，t3 排名第二的概率为 0.18+0.048+0.072=0.3。

对于单选择情形，我们为 U-Topk 和 U-kRanks 查询提供解决方案，在 x 关系模型下两者都非常快，并且使用极小的空间。图 2 给出了在 x 关系模型下这个算法的

渐近的比较。对于多选情形的研究出现在这篇文章[3]的完整版本。

	U-Topk		U-kRanks	
	时间	空间	时间	空间
我们的	n log k	k	nk	k
[2]	nk	k^2	n^2k	nk

II 算法

我们存储一个关系数据库表的所有 N 个元组在 D，这些表称作元组表。我们把 x 元组的信息存储在一个 x 表。通过使用一个哈希映射，给定一个元组 t 的 id，那么 t 的所有可选物的得分和指置信度的值就可以在 O(1)时间获得。

为了处理一个 Top-k 查询，我们按照得分的降序来检索元组，今早结束检索，只要我们肯定那些未被检测的元组没有可能影响到查询结果。我们定义扫描深度，用 n 表示，扫描深度 n 就是保证结果正确所必须要扫描的元组的最小数目。更正式的：

定义 3 扫描深度（scan depth)：假设在不确定数据库 D 里面有 $t_1,...,t_N$ 这些已经排序的元组。对于 U-Topk 和 U-kRanks 查询，扫描深度 n 就是最小的 n 使得下面的描述成立：对于任意的 D'，D'的前 n 个元组与 D 在相同的排序准则是一样的，例如：t1...,tn，在 D'上的查询结果与在 D 上的查询结果是一样的。

A. U-Topk 查询

定义 D_i 为不确定性数据，当 D 限制为 D_i={t1,...ti}，对于 i=1,...,N, $D_i=\{T'|T'=T\cap D_i, T\in D\}$。对于图 1 中的数据库，这意味着，$D_1=\{T_1^{'}=\{t_1\}\}$，$D_2=\{T_1^{'}=\{t_1\},T_2^{'}=\{t_2\}\}$，$D_3=\{T_1^{'}=\{t_1\},T_2^{'}=\{t_2\},T_3^{'}=\{t_3\}\}$，$D_4=\{T_1^{'}=\{t_1,t_4\},T_2^{'}=\{t_2\},T_3^{'}=\{t_3\}\}$。我们用 W|$D_i$ 来表示从 D_i 生成的一个可能世界实例，它的概率为 Pr[W|D_i]。对于 $i\ge k$，令 S_i 表示从 D_i 生成的包含 k 个元组的可能世界，即，$S_i=\arg\max_{|W|=k}\Pr[W\mid D_i]$，令 ρ_i=Pr[Si|D_i]。我们的算法框架是：一个个元组检索，当 i 从 k 递增到 N 的过程中，计算 S_i。最后取 S_i 作为结果集，此时 S_i 对于的 ρ_i 是最大的。这个框架的正确性由下面这个引理保证。

引理 1：$\Pr[\Psi(W\mid D)=T^*]=\max\{\rho_i\mid k\le i\le N\}$

证明：

令 $i^*=\max\{i\mid t_i\in T^*\}$。显然，$\Pr[\Psi(W\mid D)=T^*]=\Pr[\Psi(W\mid D_{i^*})=T^*]=\rho_{i^*}$，因此，$\Pr[\Psi(W\mid D)=T^*]\le\max\{\rho_i\mid k\le i\le N\}$。

另一方面，考虑任意 T'，令 $i'=\max\{i\mid t_i\in T'\}$，根据定义，对于任意的 i'有：$\Pr[\Psi(W\mid D)=T^*]\ge\Pr[\Psi(W\mid D)=T']=\rho_{i'}$。因此可以得出：

$\Pr[\Psi(W \mid D) = T^*] = \max\{\rho_i \mid k \le i \le N\}$

使用引理 1 使得我们不需要枚举所以可能世界并且计算最大的总概率，我们只需要计算最大的 ρ_i 以及与其对应的 S_i，而这个 S_i 就是 U-Topk 查询的结果。因此，U-Topk 查询的问题就转化为对于 i=k,k+1,...,N，计算 ρ_i 以及与其对应的 S_i。实际上，只要我们确定剩下的 ρ_i 不可能大于目前已经找到的 ρ_i，那么我们就可以停止处理了。然而，我们仍然需要一个有效的算法来计算 S_i 和 ρ_i，同样需要 一个方法来告诉我们扫描深度是否已经达到。

引理 2： 对于一个单一选择的数据库 D 和任意的 $k \le i \le N$，Si 包含了 Di 中置信度最大的 k 个元组，并且：

$$\rho_i = \prod_{tj \in S_i} p(t_j) \bullet \prod_{tj \in D_i \setminus S_i} (1 - p(t_j))$$

证明：因为 Pr[W|D_i]是两个因子的乘积，在 W 中的所有元组出现的概率以及剩余的所有元组不出现的概率，当 W 包含 k 个置信度最大的元组时这两个因子都会取得最大值。只要知道了 S_i，那么 ρ_i 也就知道了。

接下来我们给出这种情形的扫描深度的特征。

引理 3：对于一个单一选择的数据库 D 和一个 U-Topk 查询，扫描深度就是最小的 n 使得：

$$\max_{1 \le i \le n} \rho_i \ge \prod_{1 \le i \le n} \max\{p(t_i), 1 - p(t_i)\} \qquad (1)$$

证明：

我们首先证明，当（1）发生的时候，必须要再检索元组了。这是因为（1）的左边是检索了 n 个元组之后找到的最好的答案；而（1）的右边是 Pr[W|D_i]的上边界，对于任意的 W 和任意的 i>n。

其次，我们证明如果（1）不成立，那么我们肯定还没有到达扫描深度。这个条件是紧凑的。这保证了我们的算法不会检索多余 n 个元组。我们首先证明下面的断言：如果我们检索了 k 个置信度大于等于 1/2 的元组，那么（1）必定成立。考虑第一次检索了 k 个这样的元组，例如检索完 t_s。因为这 k 个在 D_s 置信度最大的元组必定是置信度大于等于 1/2 的 k 个元组。结合引理 2，我们有：$\max_{1 \le i \le s} \rho_i \ge \rho_s = \prod_{1 \le i \le s} \max\{p(ti), 1 - p(ti)\}$。进一步地，因为（1）的左边从不变小并且（1）的右边从不变大，当我们检索了 n 个元组的时候，（1）依然成立。

我们构造另外一个不确定数据库 D'，它的前 n 个元组与 D 是一样的，而其余元组的置信度为 1，我们讨论一定可以从 D'中找到比 D 更好的 U-Topk 结果如果（1）没有成立的话。因为（1）没有成立，所以在 D 和 D'的前 n 个元组中必定有 l<k 个元组置信度大于等于 1/2。因为 D'中剩余的所有元组的置信度都是 1，把这 l 个检索过和 k-l 个未检索过的元组放在一起得到一个比当前从 D 中获得的 Top-k 结果要好

的结果，它的概率为 $\prod_{1\le i\le n}\max\{p(t_i),1-p(t_i)\}$。因此，根据定义，我们还没有达到扫描深度。

使用引理 2 和引理 3，很容易得到一个处理 U-Topk 查询的高效算法。这个算法一个个元组的检索，维护目前检索过的 k 个置信度最大的元组，使用引理 2 来计算每一个 ρ_i。我们可以使用一个大小为 k 的堆来实现这个目标，每个元组的时间开销为 O(log k)。同时，它维护了（1）的右边，这样就可以在检索了 n 个元组之后停止检索。这对于每个元组来说都可以在常数时间内完成。因此，我们可以总结：

定理 1：对于一个单选择的不确定数据库，我们的算法能够在检索了 n 个元组之后得出 U-Topk 查询结果，它的时间开销为 O(n log k)，空间需求为 O(k)。

B.U-kRanks 查询

这一节我们使用一个动态编程的算法来考虑 U-kRanks 查询。我们的算法是基于下面这个简单的知识的：一个元组 t_i 排名在第 j 个位置的概率依赖于前面的 i-1 个元组有 j-1 个元组出现，而不管这 j-1 个元组是什么。

令 D 为一个单选择的不确定数据库。对于 $1\le j\le i\le N$，令 $r_{i,j}$ 为 Di 中一个可能世界有 j 个元组的概率，即，$r_{i,j}=\sum_{|W|=j}\Pr[W\mid D_i]$。我们同样定义 r0,0=1。显然，ti 在随机生成的可能世界中排名 j 的概率为 $p(t_i)\bullet r_{i-1,j-1}$。因此，在不确定数据库 D 中的 U-kRanks 查询结果为 $t_{X(j)}$，使得对于 j=1,...,k，有：

$$x(j)=\arg\max_{j\le i\le N}\{p(t_i)\bullet r_{i-1,j-1}\} \qquad （2）$$

现在我们剩下的任务就是计算 ri,j 了，它由下面的公式得出：

$$r_{i,j}=\begin{cases} p(t_i)r_{i-1,j-1}+(1-p(t_i))r_{i-1,j}, & if \quad i\ge j\ge 0; \\ 1, & if \quad i=j=0; \\ 0, & otherwise. \end{cases} \qquad （3）$$

公式（3）的正确性是显而易见的：为了从 D_i 中取得 j 个元组，要么就选择 t_i 然后从 D_{i-1} 中选择 j-1 个元组，要么不选择 t_i 然后从 D_{i-1} 中选择 j 个元组。

检索每个元组 ti 时，对于 j=0,1,...,min{i,k}我们的算法使用（3）来计算 $r_{i,j}$。根据（2），这同样保存了目前找到的最好的答案 x(j)。为了计算 $r_{i,j}$，只需计算 $r_{i-1,j}$，在计算过程中，我们的算法需要的空间为 O(k)。

最后，我们得出扫描深度 n 有如下特性，使得我们的算法能够在得出结果的时候停止，只需要从元组表里检索 n 个元组，这是最少的。

引理 4：对于一个单选择不确定数据库 D 和一个 U-kRanks 查询，扫描深度 n 就是最小的 n 使得对于 j=1,...,k 下面这条公式成立

$$\sum_{j\le i\le n}\{p(t_i)r_{i-1,j-1}\}\ge\sum_{0\le l\le j-1}r_{n,l} \qquad （4）$$

证明：因为（4）的左边是当前排在 j 位置最好的结果，有足够的证据证明，对

于任意 D'，假设它的元组为 $t_1,...,t_n,t'_{n+1},...,t'_N$,(4)的右边是 t'_i 排在第 j（j=1,...,k)个位置的概率的上边界，而这个上边界是可以获得的。

首先，对于任意 i>n，考虑 t'_i 成为从 D'中随机生成的世界第 j 个元组的概率。令 ξ_s 为$\{t'_{n+1},...,t'_{i-1}\}$中有 s 个元组出现的概率（如果 i=n+1 则 $\xi_0=1$)，有：

$$\Pr[\Psi j(W \mid D') = t_i^{'}] = p(t_i^{'})(\sum_{l=0}^{j-1} r_{n}, l \bullet \xi_{j-1-l}) \le \sum_{l=0}^{j-1} r_{n}, l \bullet \xi_{j-1-l} \le \max_{0 \le l \le j} r_{n,l}$$

最后一条不等式之所以成立时因为$\sum_{s=0}^{j-1} \xi_s \le 1$。因此，在得到正确的结果之前我们最多需要访问 n 个元组。

其次，我们证明对于任意的 j，总有一个 D'能够达到这个上边界。令 p(t'n+1)=……=p(t'N)=1，且 $l^* = \arg\max_{0 \le l \le j-1} r_{n,l}$。考虑元组 t'_{n+j-l^*}，它出现在 D'中随机生成的可能世界的第 j 个位置的概率为 r_{n,l^*}。因此，为了避免出错，我们最少需要访问 n 个元组。

因为对于每一个元组和$1 \le j \le k$ 我们可以在 O(k）时间内检查不等式（4），所以下面的定理成立。

定理 2： 对于一个单选择不确定数据库，我们的算法能够在检索 n 个元组之后得出 U-kRanks 查询的结果，时间开销为 O(nk)，空间开销为 O(k)。

III 总结

扩展的实验结果和多选情形的算法可以在完整版找到[3]。

IV 致谢

这些工作是在 Ke Yi 和 Feifei Li 在 AT&T 实验研究所的时候完成的。Ke Yi 的部分支持来自香港直接拨款资助(DAG07/08)。Feifei Li and 和 George Kollios 的部分支持来自 NSF grant IIS-0133825。

参考文献

[1] P. Agrawal, O. Benjelloun, A. Das Sarma, C. Hayworth, S. Nabar,T. Sugihara, and J. Widom, .Trio: A system for data, uncertainty, andlineage,. in VLDB, 2006.

[2] M. A. Soliman, I. F. Ilyas, and K. C. Chang, .Top-k query processing in uncertain databases,. in ICDE, 2007.

[3] K. Yi, F. Li, D. Srivastava, and G. Kollios, .Efficient processing of topk queries in uncertain databases,. Florida State University, Tech. Rep.,2008.