华南理工大学

South China University of Technology

本科毕业设计（论文）说明书

不确定性数据的查询分析处理

学　　院　软 件 学 院

专　　业　软 件 工 程

学生姓名　区钺坚

指导教师　杜　卿

提交日期　**2010 年 6 月 5 日**

华 南 理 工 大 学

毕 业 设 计 （论文） 任 务 书

兹发给 软件学院 06 级 4 班学生 区钺坚 毕业设计（论文）任务书，内容如下：

1.毕业设计（论文）题目：不确定性数据的查询分析处理

2.应完成的项目：

（1）2010 年 3 月 1 日前拟定提纲并提交开题报告

（2）2010 年 5 月 15 日前完成论文初稿

（3）进行与论文题目相关的调研工作并收集相关的资料

（4）2010 年 6 月 1 日前参考外文文献资料并提交外文翻译译文

3.参考资料以及说明：

（1）崔斌，卢阳. 基于不确定数据的查询处理综述[J] .计算机应用.2008，28(11): 2729-2731

（2）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[J].计算机学报，2009，(01) :1-16

（3）李建中，于戈，周傲英. 不确定性数据管理的要求与挑战.中国计算机学会通讯.2009，5(4):6-15

（4）申德荣，于戈，寇月，聂铁铮.可能世界内数值型不确定数据匹配模型.计算机应用研究.2008-7：2607-2611

（5）R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S.Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In VLDB, pages 876~887, 2004.

（6）Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In Proceedings of the 31st international conference on Very large data bases (VLDB'05), pages 922~933.VLDB Endowment, 2005.

（7）HUA M,PEI J,ZHANG W,et al. Ranking queries on uncertaindata:A probabilistic threshold approach[C] .Proceedings of the 2008 ACMSIGMOD International Conference on Management of Da-ta. New York: ACMPress, 2008, :673-686 .

4.本毕业设计（论文）任务书于 2010 年 2 月 5 日发出，应于 2010 年 6 月 10 日前完成，然后提交毕业考试委员会进行答辩。

专业教研组（系）、研究所负责人_____审核_____年___月___日

指导教师_____签发 _____年___月___日

毕业设计（论文）评语：

毕业设计（论文）总评成绩：

毕业设计（论文）答辩负责人签字：

年　　月　　日

摘 要

不确定性数据是许多应用固有的，这些应用包括经济、军事、物流、金融、电信等等。由于这些应用的重要性和不确定性数据的快速增长积累，分析大量的不确定性数据已经成为一个重要的任务并且吸引着数据库社区越来越多的关注。不确定性数据的种类是丰富多彩的，例如关系数据库、流数据和移动物体。根据场景和数据特征的不同，研究者研究出许多不同的数据模型。所有这些模型的核心思想都来源于可能世界模型。每个可能世界模型包含大量的可能世界实例，这些可能世界实例的概率总和为 1。然而，由于可能世界实例的数量远远大于不确定数据库的规模，这就使得从所有可能世界实例中得到中间结果然后合并得出最终结果的方法是不可行的。应用传统的查询方法来处理不确定数据的查询会使得结果集出现偏差从而无法满足用户的需求。因此，必须要使用排序和剪枝等启发式技术来减少计算开销从而提高效率。

本论文首先介绍了各种各样的数据模型；其次，介绍了基于不确定性数据的 Top-k 和 skyline 查询，包括查询的定义、一些例子以及算法；然后，介绍了一些不确定性数据的处理；最后，介绍了两种索引技术——PTI 和 U-Tree，PTI 是用来管理一维不确定性数据的，U-Tree 则扩展了 PTI，使之能够用来对多维不确定对象进行索引。

关键词：不确定性数据 数据模型 Top-k 查询 skyline 查询 PTI 索引 U-Tree 索引

Abstract

Uncertain data are inherent in some important applications, inclusive of economy, military, logistic, finance and telecommunication, etc. Due to the importance of those applications and the rapidly increasing amount of uncertain data collected and accumulated, analyzing large collections of uncertain data has become an important task and has attracted more and more interest from the database community. Uncertain data has many different styles, such as relational data, streaming data, and moving objects. According to scenarios and data characteristics, lots of data models have been developed. All those models come from the core possible world model that contains a huge number of the possible world instances with the sum of probabilities equal to 1. However, the number of the possible world instances is far greater than the volume of the uncertain database, making it infeasible to combine medial results generated from all of possible world instances for the final query results. Using traditional querying methods on uncertain data will bias the answer set, and hence cannot satisfy users' needs. Thus, some heuristic techniques, such as ordering, pruning, indexing, must be used to reduce the computation cost for the high efficiency.

This paper firstly introduces all kinds of data models; secondly, we introduces Top-k queries and skyline queries based on uncertain data, including their definitions, some examples and algorithms; then the processing of uncertain data is introduced; at last we introduces two kinds of indexing technology, PTI and U-Tree, PTI is used for managing one-dimension uncertain data while U-Tree extends PTI to index multidimensional uncertain objects.

Keyword: uncertain data, data model, top-k queries, skyline queries, PTI, U-Tree

目 录

第一章 引言

1.1 不确定性数据的研究背景

近年来，在许多现实的应用中，例如传感器网络与射频识别电子标签、互联网数据、基于位置服务、 电信服务、 数据挖掘应用、金融服务，数据的不确定性普遍存在，不确定性数据(uncertain data)扮演着关键角色。随着技术的进步和人们对数据采集和处理技术理解的不断深入，不确定性数据得到了广泛的重视。不确定数据在一些重要应用领域中是固有存在的，如传感器网络和移动物体追踪。在不确定数据上使用传统的查询方法会使查询结果出现偏差，不能满足用户的需求。由于真实世界应用中数据固有的不确定性，以及相关应用系统需求的增加，针对不确定数据管理的研究已经成为数据库技术研究中一个新的热点。不确定性数据的表现形式多种多样，它们可以以关系型数据、半结构化数据、流数据或移动对象数据等形式出现。不确定性数据的产生原因比较复杂。可能是原始数据本身不准确或是采用了粗粒度的数据集合，也可能是为了满足特殊应用目的或是在处理缺失值、数据集成过程中而产生的。

不确定性数据管理技术的典型框架：模型定义、预处理与集成、存储与索引和查询分析处理。而查询分析处理是不确定性数据管理的最终目标。查询类型非常丰富，例如关系查询操作、数据世系、XML 处理、流数据查询、top-k 查询、Skyline 查询、OLAP 分析、数据挖掘等。尽管可以分别针对各个可能世界实例计算查询结果，再合并中间结果以生成最终查询结果，但由于可能世界实例的数量远大于不确定数据库的规模，该方法并不可行。因此，必须采用排序、剪枝等启发式技术优化处理，以提高效率。另外，由于输入数据具有不确定性，查询结果也往往是近似结果。

不确定性数据查询分析处理的研究工作在很多方面都得到了很好的发展，包括数据集成、索引技术、半结构化数据、世系分析、关系代数处理、数据流分析、top-k 查询、Skyline 查询、联机分析处理和数据挖掘等。

1.2 不确定性数据的分类

目前，有关非确定数据的定义和分类还没有严格的定义，Trio 中将数据分为 exact 和 inexact 两种[1]。有关 inexact 数据的描述又有多种，如不确定的数据、概率数据、模糊集数据、近似数据、不完备数据和不精确数据等。Motro[2]将不确定信息分为不确定和不精确两类。不确定是指属性值的可信性，概率是指属性取某一值的概率。不完备的数据是指有信息丢失。不精确数据是指数据的取值可能是集合中的数据之一。

文献[2]中除了将非确定的数据定义为不确定和不精确数据外，还包括不完备、模糊、不一致、不明确。其中除了不明确为语义模糊概念外，其他都涵盖了 Trio 中的定义。

申德荣等人[3]把不确定数据分为七类：range、or_set、probability、unknown、negative、vague、fuzzy。其中的前五类属于数值型的不确定性。

1.3 不确定性数据产生的原因

不确定性数据的产生原因比较复杂。周傲英等人[4]认为原因有原始数据本身不准确或是采用了粗粒度的数据集合，也可能是为了满足特殊应用目的或是在处理缺失值、数据集成过程中而产生的。HUA 等人[5]认为不确定性数据在各个应用中出现的原因包括：不完整的数据、设备的缺陷、数据传输延迟或者丢失。

1.4 不确定性数据的查询分析处理的关键问题与挑战

与传统的确定性数据相比，不确定性数据引入了概率维，而这个概率的引入使得查询、分析、处理的各个环节都都面临着巨大的挑战。李建中等人[6]提到了五方面的挑战。

1) 差异显著的数据模型

传统数据模型无法准确描述不确定性数据，可能世界（Possible World）模型是描述不确定性数据的通用模型。该模型包含若干个可能世界实例，在各个实例中，一部分元组存在，剩余元组不存在。可能世界实例的发生概率等于实例内元组的概率乘积和实例外元组的不发生概率的乘积之积。所有可能世界实例的发生概率之和等于 1。

2) 急剧攀升的问题复杂度

管理不确定性数据所面对的最直接的挑战，就是相对于数据库规模呈指数倍的可能世界实例的数量。“罗列可能世界实例，计算基于该实例的查询结果，整合各实例的查询结果生成最终的答案”的处理方式显然是不可行的，迫切需要结合各种剪枝、排序等技术以快速计算查询结果。

3) 非同一般的概率维

不确定性数据比确定性数据多了一个概率维，这就导致了查询、索引、处理工程、处理结果都大受影响。概率维度不是普通的维度，它的出现改变了传统的数据处理模式。

首先，部分查询定义可能拥有概率参数，例如 Pt-k 查询（一种 top-k 查询）需要一个概率参数 p，仅返回成为 top-k 成员的概率超过 p 的元组集合[5]。

其次，传统的索引技术（例如 B+树、R 树等）无法有效索引不确定性数据，需要开发新的索引技术。

再次，处理过程需要充分考虑概率因素，许多算法在执行过程中会优先考虑高概率的元组。最后，查询结果也会包含概率信息。

4) 多样的数据形态

半结构化数据（XML）、流数据、多维数据和空间数据等数据形式都会受到概率引入

的影响。

5) 丰富的查询类型

在不确定性数据环境下，由于引入了概率维度，查询的种类反而会增加。元组的概率维度值从侧面反映了该元组的重要程度，因而影响着查询的定义。

1.5 不确定性数据的查询分析处理的内容与目标

与在确定数据上查询不同，不确定数据上的研究工作将概率引入到数据模型中来衡量不确定对象成为结果集中元素的可能性。不确定性数据的表现形式多种多样，它们可以以关系型数据、半结构化数据、流数据或移动对象数据等形式出现。由于问题定义和数据模型的不同，不确定数据上的查询类型也多种多样。目前，根据应用特点与数据形式差异，研究者已经提出了多种针对不确定数据的数据模型。这些不确定性数据模型的核心思想都源自于可能世界模型。可能世界模型从一个或多个不确定的数据源演化出诸多确定的数据库实例，称为可能世界实例，而且所有实例的概率之和等于 1。尽管可以首先分别为各个实例计算查询结果，然后合并中间结果以生成最终查询结果，但由于可能世界实例的数量远大于不确定性数据库的规模，这种方法并不可行。因此，必须运用排序、压缩、剪枝、验证等启发式技术设计新型算法，以提高效率。本文介绍了不确定性数据的查询分析处理各个方面的知识。包括以下几个方面。

1) 不确定性数据的分类、模型以及不确定性数据产生的原因。
2) 基于不确定性数据的 Top-k 查询和 p-skyline 查询
3) 各种查询分析处理技术。
4) PTI 和 U-Tree 两种索引技术。

目前，基于不确定性数据的研究比较多，这方面的资料也比较多。但是，很多资料都是就不确定性数据的某一个方面进行阐述的，要从整体上把握不确定性数据需要阅读大量的文献。文献[4]提出了不确定性数据管理技术的典型框架：模型定义、预处理与集成、存储于索引和查询分析处理，并且介绍了不确定性数据各个方面的知识，是一篇综述不确定性数据管理技术研究的文章。但是，文中说得比较简单，如果没有一定的基础是很难理解的。本文的主要目的就是让读者能够快速的了解上面介绍到的内容，从而尽快从整体上把握不确定性数据。因此，本文尽量把别人的研究成果解释得详细一些，能够举例子说明的就举例子说明，需要计算的则把计算过程也描述出来，对于他们提出的一些定理能够证明的则给出证明。

第二章 不确定性数据的模型

根据应用场景的不同，数据模型是多种多样的，目前就已经提出了多种数据模型，其中可能世界模型(possible world model)[7,8]是最核心的，并衍生出多种应用相关的模型，特别是针对关系型数据、半结构化数据、流数据和多维数据的模型。

2.1 可能世界模型

可能世界模型[7,8]从一个不确定性数据库演化出很多确定的数据库实例(称为可能世界实例)，而且所有实例的概率之和为 1。每个可能世界实例出现的概率为该实例中所有元组的概率与不在该实例中的元组不出现的概率的乘积。不确定性数据的种类较多，例如关系型数据、半结构化数据、流数据、移动对象数据等，尽管存在许多与数据类型紧密相关的数据模型，但是这些模型最终都可以转化为可能世界模型。可能世界模型是在不确定性数据管理领域，最常用的模型。

考虑下面的表格所示的一个简单例子。

表 2-1 不确定性表

元组	概率
t1	0.4
t2	0.3
t3	0.5

上表描述了一个不确定性表包含 3 个元组，每个元组是否出现在某个可能世界实例都有一个概率。因为 3 个元组，每个元组都可能出现或者不出现在某个可能世界实例，所以总共有 8 个可能世界实例。如下表：

表 2-2 可能世界实例

可能世界	概率
PW1={∅}	0.21
PW2={t1}	0.14
PW3={t2}	0.09
PW4={t3}	0.21
PW5={t1,t2}	0.06
PW6={t1,t3}	0.14
PW7={t2,t3}	0.09
PW8={t1,t2,t3}	0.06

从表 2-2 可以看出：

1) 所有实例的概率之和为 1。可以通过将概率这一列的值加起来验证一下：

0.21+0.14+0.09+0.21+0.06+0.14+0.09+0.06=1

2) 每个可能世界实例出现的概率为该实例中所有元组的概率与不在该实例中的元组不出现的概率的乘积。

例如，可能世界 PW1 的概率为 t1、t2、t3 都不出现的概率的乘积:

p(PW1)=(1-t1)*(1-t2)*(1-t3)=0.6*0.7*0.5=0.21

再如，可能世界 PW5 的概率为 t1、t2 出现但 t3 不出现的概率的乘积：

p(PW5)=t1*t2*(1-t3)=0.4*0.3*0.5=0.06

3) 每个元组在出现在所有可能世界的概率总和等于它本身的概率。如：

p(t1)=p(PW2)+p(PW5)+p(PW6)+p(PW8)=0.14+0.06+0.14+0.06=0.4

p(t2)=p(PW3)+p(PW5)+p(PW7)+p(PW8)=0.09+0.06+0.09+0.06=0.3

p(t3)=p(PW4)+p(PW6)+p(PW7)+p(PW8)=0.21+0.14+0.09+0.06=0.5

Jian 等人[9]介绍了从可能世界模型派生出的概率数据库(2.2)和不确定对象(2.3)两种模型，并且给出一种方法相互转换这两种模型。

2.2 概率数据库模型

概率数据库（probabilistic database）：包含了有限数目的不确定性表。

不确定性表 T（uncertain table）：包含一系列的元组，这些元组的出现概率 Pr(t)>0。

生成规则（generation rule）：明确了一系列的互斥元组，这些元组的概率加起来小于等于 1，并且在一个可能世界实例，每个生成规则最多只能出现一个元组。

假设所有的元组最多只能出现在一个生成规则 R，对于那些没有出现在任何 R 的元组，让它门构成一个规则 Rt，因此，一个不确定性数据表是由一系列生成规则构成的。

一个生成规则的概率（probability）由这个规则里面所有元组的概率和，表示为

$$\Pr(R)=\sum_{t\in R}\Pr(t)$$

一个生成规则的长度(length)就是这个规则的元组的数目，表示为|R|。

单一规则（singleton rule）：长度为 1 的生成规则，即|R|=1。

多元组规则（multi-rule）：长度大于一的生成规则，即|R|>1。

依赖元组：出现在多元组规则的元组。

独立元组：不是依赖元组的元组。

一个可能世界 W 就是 T 的一个子集，使得对每一个生成规则 $R\in\Re_T$：如果 Pr(R)=1 则 |R∩W|=1；如果 Pr(R)＜1 则|R∩W|≤1。

ω 表示所有可能世界实例。

对于一个不确定表 T，它的生成规则集合为 $\Re_T$，他的所有可能世界实例的数目就是那些概率为 1 的生成规则的长度与那些概率不为 1 的生成规则的长度加 1 的和的乘积。表示为：

$$|w|=\prod R\in\Re_T,\Pr(R)=1|R|\prod R\in\Re_T,\Pr(R)<1(|R|+1)$$

可能世界的存在概率（existence probability）：每个可能世界出现的概率。公式为：

$$\Pr(W) = \prod_{R \in \Re_T, |R \cap W| = 1} \Pr(R \cap W) \prod_{R \in \Re_T, |R \cap W| = \phi} (1 - \Pr(R))$$

假设在一个概率数据库里面有下面这个不确定性表 T：

表 2-3 不确定性表

元组	概率
t1	0.3
t2	0.4
t3	0.5
t4	1.0
t5	0.8
t6	0.2

它的生成规则为（R1=t2⊕t3，R2=t5⊕t6）。

根据生成规则的概率公式可知：

Pr(R1)=Pr(t2)+Pr(t3)=0.4+0.5=0.9<1

Pr(R2)=Pr(t5)+Pr(t6)=0.8+0.2=1.0

这两个生成规则的长度都是 2，因为它们都包含两个元组，属于多元组规则。t2,t3,t5,t6 属于依赖元组，因为它们出现在多元组规则里面；t1,t4 属于独立元组，因为它们不是依赖元组。

因为 R1 的概率小于 1，所以 t2 和 t3 这两个元组可以有一个或者一个都没有出现在某个可能世界实例；因为 R2 的概率为 1，所以 t5 和 t6 这两个元组有且仅有一个出现在某个可能世界实例。

所有可能世界实例及其概率如下表：

表 2-4 所有可能世界实例

可能世界实例	概率
W1={t1,t2,t4,t5}	0.096
W2={t1,t2,t4,t6}	0.024
W3={t1,t3,t4,t5}	0.12
W4={t1,t3,t4,t6}	0.03
W5={t1,t4,t5}	0.024
W6={t1,t4,t6}	0.006
W7={t2,t4,t5}	0.224
W8={t2,t4,t6}	0.056
W9={t3,t4,t5}	0.28
W10={t3,t4,t6}	0.07
W11={t4,t5}	0.056
W12={t4,t6}	0.014

可以看出每个可能世界实例都包含 t4 以及 R2 规则中的一个元组 t5 或者 t6，因为 t4 和 R2 的概率都是 1。

2.3 不确定对象模型

不确定对象（uncertain object）由一个在数据空间 D 的概率密度函数(PDF)f 来描述。一般的，对于在 D 中的任意点 u， $f(u)\geq 0$ 并且 $\int_{u\in D} f(u)du=1$。

一个不确定对象 U 所包含的实例的数目表示为|U|=l。

令 X1,...,Xn 为 n 个不确定对象。一个可能世界 W={x1,...,xn}包含了各个不确定对象的一个实例，xi 就是 Xi($1\leq i\leq n$)的一个实例。一个可能世界实例的存在概率为 $\Pr(W)=\prod_{i=1}^{n} f_i(x_i)$，fi 就是 Xi 的概率密度函数。

显然，对于一个可能世界 W，Pr(W)>0，进一步， $\sum_{W\in\omega}\Pr(W)=1$。

2.4 针对关系型数据

针对关系模型的扩展最为常见，包括 Probabilistic ?-table[10]、Probabilistic or-set table[11] 、Probabilistic or-set-? table[11] 、Probabilistic c-table[12] 等。

Probabilistic ?-table 就一个针对关系型数据的模型。它以一个独立的概率字段表示元组的概率，且各元组之间独立。一个特定的数据库实例(也即可能世界实例) 的概率等于其所包含的元组的概率乘积和其所不包含的元组的不发生概率的乘积。

Probabilistic ?-table 能够描述存在级不确定性，而 Probabilistic or-set table 则倾向于描述属性级不确定性。在 Probabilistic or-set table 中，元组的属性值被描述为多个候选值之间的“或”关系，可视为离散概率密度函数。

Probabilistic or-set-? table[10,11] 则是上述两种模型的综合体。部分学者也将 probabilistic or-set-? table 命名为 x-relation，它包含若干 x-tuple (无存在级不确定性) 或者 maybe x-tuple (有存在级不确定性)[13,14]。

Probabilistic c-table 模型的定义与 Probabilistic or-set Table 模型比较类似，不同之处在于它是从 c table 衍生出来的。

2.5 针对半结构化数据模型

半结构化数据模型(semistructured data model) 能有效描述缺乏严格模式结构的数据。半结构化数据通常可以用文档树来描述。Dekhtyar 等人提出了一种管理概率半结构化数据(probabilistic semistructured data) 的方法，该方法以关系数据库技术为基础，支持丰富的代数查询。更多的工作则是直接以文档树形式描述不确定性半结构化数据，例如 p-文档(p-document model) 、概率树模型、PXML 模型、PrXML 模型等。

2.6 数据流模型

在数据流模型中，数据到达的速度极快、数据规模极大，仅能够开发一次扫描算法，使用有限内存在线计算查询结果。在不确定性数据流(Uncertain Data Stream，或 Probabilistic Data Stream) 中，各元组具有不确定性。

2.7 针对多维数据模型

OLAP 提供了一种多维数据分析手段，能够快速得到复杂的查询统计结果。OLAP 中数据立方(Data Cube) 的基本元素是 cuboid。在确定性多维数据模型中，各个事实(fact) 必定属于某一个立方体中。但对于处理不精确数据的应用而言，各事实可能无法被准确地定位到立方体中。例如，考虑一个有关汽车销售的多维数据模型，它包括两个维度：city 与 automobile，分别表示购车城市与车体型号。city 维度是一个三级层次结构，国家→省→市。若仅仅知道某辆“奔驰车”是从“浙江北部城市”购买的话，由于“浙江北部城市”包含多个城市，该条记录是不确定性数据，无法存放到事实表中去。

2.8 概率数据库和不确定对象模型的相互转换

在离散的情形下，不确定对象模型和概率数据库模型是等价的。可以使用下面的方法来将不确定对象转化为概率表。对于不确定对象 U 中的每一个实例 u，创建一个元组 tu,它的概率为 f(u)。对每一个不确定对象 U={u1,...,ul},创建一个生成规则 $R_U = t_{u1} \oplus ... \oplus t_{ul}$ 。

在所有的情形，一个概率表总是可以使用一系列的不确定对象来表示的。方法如下：为不确定表 T 中的每一个元组 t 创建一个实例 ut,它的概率密度函数为 f(ut)=Pr(t)。对于一个生成规则 R: $t_{r1} \oplus ... \oplus t_{rm}$,创建一个不确定对象 U_R，它包含所有与元组 $t_{r1},...,t_{rm}$ 对应的实例 $u_{tr1},...u_{trm}$。进一步，如果 $\sum_{i=1}^{m} \Pr(t_{ri}) < 1$ 则创建另外一个实例 $u_{\varnothing}$，它的概率密度函数为 $f(u_{\phi}) = 1 - \sum_{i=1}^{m} \Pr(t_{ri})$，然后把 $u_{\varnothing}$添加到不确定对象 U_R。由于任意两个生成规则里面没有相同的元组，所有构建出来的所有不确定对象不会有共同的实例。

第三章 不确定性数据的查询

查询分析处理是不确定性数据管理的最终目标。查询类型非常丰富，例如 top-k 查询、p-Skyline 查询、流数据查询等。尽管可以分别针对各个可能世界实例计算查询结果，再合并中间结果以生成最终查询结果，但由于可能世界实例的数量远大于不确定数据库的规模，该方法并不可行。因此，必须采用排序、剪枝等启发式技术优化处理，以提高效率。另外，由于输入数据具有不确定性，查询结果也往往是近似结果。

3.1 基于不确定性数据的 Top-k 查询

面向确定性数据库的 top-k 查询的定义非常清晰：返回 ranking 函数值最大的 k 个元组。但是在不确定性数据库上根据应用的不同可以存在多种定义方法。目前，基于不确定性数据的 Top-k 查询主要有 U-Topk[15]、U-kRanks[15]、PT-k[5]、Pk-topk[16]四种。

接下来的四个小节将介绍这四种查询，包括它们的定义、举例说明以及相关的算法。对这四种查询的举例都是基于下面这个不确定性表的。

考虑下面这个不确定性表 T，它包含六个元组，每个元组都有一个分数和概率属性。

表 3-1 不确定性表

元组	分数	概率
t1	95	0.3
t2	90	0.4
t3	80	0.5
t4	78	1.0
t5	87	0.8
t6	70	0.2

生成规则（t2⊕t3，t5⊕t6）

因为 t2 和 t3 是互斥的，并且 p(R2,3)=p(t2)+p(t3)=0.4+0.5=0.9<1，所以 t2 和 t3 两个元组最能有一个或者没有出现在某个可能世界实例。

因为 t5 和 t6 是互斥的，并且 p(R5,6)=p(t5)+p(t6)=0.8+0.2=1.0，所以 t5 和 t6 两个元组有且仅有一个元组出现在每个可能世界实例。

而因为 t4 的概率为 1，所以 t4 必须出现在每一个可能世界实例。

假设按照分数从高到低来排名，则：t1>t2>t5>t3>t4>t6。

所有可能世界实例及其概率如下表：

表 3-2 所有可能世界实例

可能世界实例	概率	按分数排名的Top2
W1={t1,t2,t4,t5}	0.096	t1,t2
W2={t1,t2,t4,t6}	0.024	t1,t2
W3={t1,t3,t4,t5}	0.12	t1,t5
W4={t1,t3,t4,t6}	0.03	t1,t3
W5={t1,t4,t5}	0.024	t1,t5
W6={t1,t4,t6}	0.006	t1,t4
W7={t2,t4,t5}	0.224	t2,t5
W8={t2,t4,t6}	0.056	t2,t4
W9={t3,t4,t5}	0.28	t5,t3
W10={t3,t4,t6}	0.07	t3,t4
W11={t4,t5}	0.056	t5,t4
W12={t4,t6}	0.014	t4,t6

3.2 U-Topk 查询

3.2.1 U-Topk 查询的定义

文献[15]提出了 U-Topk 和 U-kRanks 两种基于不确定性数据的查询，其中 U-Topk 查询的定义如下：

定义 1： Uncertain Top－k Query (U-Topk)：令 D 为一个不确定数据库，它的可能世界空间为 PW={PW1,...,PWn}.令 T={T1,...Tm}为一系列长度为 k 的元组矢量，对于每个 Ti∈T:（1）Ti 中的所有元组是按照得分函数 F 排名的，（2）Ti 是非空可能世界 $PW(T^i) \subseteq PW$ 的 Top-k 查询结果。基于 F 的 U-Topk 查询返回 T* ∈ T，其中 $T^* = \arg\max_{T^i \in T}(\sum_{w \in PW(T^i)}(\Pr(w)))$。

U-Topk 查询返回一个长度为 k 的元组矢量，它在所有可能世界中的发生概率最大。当我们要求 Top-k 返回的所有元组都来自同一个可能世界的时候，使用 U-Topk 查询是适合的。

3.2.2 U-Topk 查询举例

考虑 3.1 节的例子：

p(t1,t2)=p(W1)+p(W2)=0.12

p(t1,t5)=p(W3)+p(W5)=0.144

p(t1,t3)=p(W4)=0.03

p(t1,t4)=p(W6)=0.006

p(t2,t5)=p(W7)=0.224

p(t2,t4)=p(W8)=0.056

p(t5,t3)=p(W9)=0.28

p(t3,t4)=p(W10)=0.07

p(t5,t4)=p(W11)=0.056

p(t4,t6)=p(W12)=0.014

因为<t5,t3>在所有可能世界实例出现的概率总和为最大，所有 U-Top2 返回的结果为<t5,t3>，它们出现在 W9。从这个例子可以看出，U-Topk 返回的结果集中的元组必须同时出现在某个可能世界实例。另外，这些元组在所有元组的排名不需要在前 k 个位置，在这里，t5 排在第三，t3 排在第四，都不是在前二。

3.2.3 U-Topk 查询算法分析

3.2.3.1 U-Topk 查询算法 1

文献[15]基于可能世界语义提出了解决 Top-k 查询的不确定数据模型。在该模型中，每个元组属于数据库的概率被称为置信度，产生规则是任意的逻辑公式用来确定有效的世界，每个可能世界是元组的联合。通过假设世界中存在的元组可以根据元组的置信度以及产生规则计算世界的概率。该文献提出了基于搜索空间的方法来处理 U-Topk 查询与 U-kRanks 查询。各元组首先按照 ranking 函数从大到小进行排序，然后不断地构造搜索空间，缩小空间的范围，最终获得查询结果。为了说明算法，引入了搜索状态 s_l 表示一个长度为 l 的元组向量，根据得分函数在一个或多个可能世界中成为 top-l 结果。假设 d 是目前从数据库中访问的元组数，s_l 的概率 P (s_l)=Pr(s_l ∧ ¬I (s_l，d))，其中 I (s_l，d) 表示在已访问的元组中而不在 s_l 中的元组。将 s_l 转变为 s_{li}，下标 i 表示该向量最后的可见元组所在的位置。

对于 U-Topk 查询，使用 OPTU-Topk 算法，该算法的基本思想： 1)设置一个以搜索状态概率优先级排序的队列 Q，初始化时插入 $s_{0,0}$，概率 P ($s_{0,0}$)=1。2） 当 Q 不为空时不断执行下面操作：从 Q 中弹出 $s_{l,i}$；如果 $l=k$ 时返回 s_{li}，否则根据 i 和 d 的比较情况选择下一个要访问的元组 t；分别对 t 可以加入和不加入 s_{li} 两种情况，将 $s_{l+1,\ i+1}$ 和 $s_{l,\ i+1}$ 插回 Q。

3.2.3.2 U-Topk 查询算法 2

文献[17]在基于可能世界语义上提出一种高效的方法来处理 U-Topk 和 U-kRanks 两种 Top-k 查询。

为了简化问题，作者首先假设各个元组是独立的，也就是如果有 n 个元组则有 2^n 个不同的可能世界实例。

为了处理 Top-k 查询，按照排名逐个检索元组，并且尽可能快的停止检索。这里作者提出了扫描深度(scan depth）的概念，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。

定义 2 ：扫描深度（scan depth）：假设在不确定数据库 D 里面有 t1,...,tN 这些已经排序的元组。对于 U-Topk 和 U-k Ranks 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，在 D'上的查询结果与在 D 上的查询结果是一样的。

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 $i \geq k$，令 Si 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 ρi=Pr[Si|$D_i$]。

作者提出的算法框架是：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 ρi 是最大的。

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 ρi 以及与其对应的 Si，而这个 Si 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 ρi 以及与其对应的 Si。实际上，只要我们确定剩下的 ρi 不可能大于目前已经找到的 ρi，那么我们就可以停止处理了。

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：

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

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。

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：

$$\max_{1 \le i \le n} \rho_i \ge \prod_{1 \le i \le n} \max\{p(t_i), 1 - p(t_i)\} \qquad (3\text{-}1)$$

证明：

我们首先证明，当(3-1)发生的时候，必须要再检索元组了。这是因为(3-1)的左边是检索了 n 个元组之后找到的最好的答案；而(3-1)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(3-1)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(3-1)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(3-1)的左边从不变小并且(3-1)的右边从不变大，当我们检索了 n 个元组的时候，(3-1)

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(3-1)没有成立的话。因为(3-1)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(3-1)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)。

3.2.3.3 算法 1 与算法 2 的比较

对于各个元组都是独立的情况下，算法1与算法2的比较如下：

表 3-3 两种U-Topk查询算法比较

	时间	空间
算法1	nk	k^2
算法2	n log k	k

3.3 U-kRanks 查询

3.3.1 U-kRanks 查询的定义

文献[15]提出了 U-Topk 和 U-kRanks 两种基于不确定性数据的查询，其中 U-kRanks 查询的定义如下：

定义 3： Uncertain k Ranks Query (U-kRanks)：令 D 为一个不确定数据库，它的可能世界空间为 PW={PW_1,...,PWn}.对于 i=1...K,令 $\{x_i^1,...,x_i^m\}$ 为一系列的元组，每个元组 x_i^j 在一个基于得分函数 F 的可能世界 $PW(x_i^j)\subseteq PW$ 中排名 i.一个基于 F 的 U-kRanks 查询返回 $\{x_i^*;i=1...k\}$，$x_i^*=\arg\max_{x_i^j}(\sum_{PW(x_i^j)}(\Pr(w)))$。

U-kRanks 查询返回在各个级别中出现的总概率最大的元组。这些元组不一定是最可能成为 Top-k 矢量的，它们也不一定都出现在同一个可能世界，并且同一个元组有可能在结果集里面出现多次。这类查询适合那些对元组来自不同世界没有限制的查询。

3.3.2 U-kRanks 查询举例

考虑 3.1 节的例子：

各个元组在所有可能世界实例排名第一的概率总和分别为：

p(t1)=p(w1)+p(w2)+p(w3)+p(w4)+p(w5)+p(w6)=0.3

p(t2)=p(w7)+p(w8)=0.28

p(t3)=p(w10)=0.07

p(t4)=p(w12)=0.014

p(t5)=p(w9)+p(w11)=0..336

p(t6)=0

因为 t5 在所有可能世界实例排名第一的概率总和最大，所以这里返回 t5。注意，t6 在所有可能世界实例排名第一的概率总和为 0，因为 t4 的分数比 t6 高，并且 t4 的概率为 1，所以在所有可能世界实例中，t6 都不可能排名第一。

各个元组在所有可能世界实例排名第二的概率总和分别为：

p(t1)=0

p(t2)=p(w1)+p(w2)=0.12

p(t3)=p(w4)+p(w9)=0.31

p(t4)=p(w6)+p(w8)+p(w10)+p(w11)=0.188

p(t5)=p(w3)+p(w5)+p(w7)=0.368

p(t6)=p(w12)=0.014

因为 t5 在所有可能世界实例排名第二的概率总和最大，所以这里返回 t5。注意，t1 在所有可能世界实例排名第二的概率总和为 0，因为 t1 在所有元组的排名为第一，所以 t1 不可能在某个可能世界实例中排名第二的。

因此，U-2Ranks 的返回结果为<t5,t5>。

从结果可以看出：U-kRanks 查询返回结果包含 k 个元组，但这 k 个元组不一定会出现在某个可能世界实例中，并且某个元组可以在结果集出现多次，如本例的 t5。

3.3.3 U-kRanks 查询算法分析

3.3.3.1 U-kRanks 查询算法 1

文献[15]除了给出 U-Topk 算法（见 3.2.3）外，还给出 U-kRanks 查询算法。对于 U-kRanks 查询使用 OPTU-kRanks 算法，该算法的基本思想是：在计算排名 i 时，对于一个新来的元组 t，计算其在所有可能世界在排名 i 上出现的概率 Pt,i；如果 Pt,i 比目前答案的概率大，并且比将未见元组考虑进来时的概率也大，那么 t 是结果集中排名为 i 的元组。

3.3.3.2 U-kRanks 查询算法 2

文献[17]除了介绍给出 U-Topk 算法以外，还介绍了使用一个动态编程的算法来处理 U-kRanks 查询。这个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 $r_{0,0}=1$。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：

$$x(j) = \arg\max_{j \le i \le N} \{p(t_i) \bullet r_{i-1,j-1}\} \qquad (3\text{-}2)$$

这样剩下的任务就是计算 $r_{i,j}$ 了，它由下面的公式得出：

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

公式(3-3)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 t_i 时，对于 j=0,1,...,min{i,k}，算法使用(3-3)来计算 $r_{i,j}$。根据（3-2），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，需要的空间为 O(k)。

最后，得出扫描深度 n 有如下特性，使得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

$$\sum_{j \le i \le n} \{p(t_i) r_{i-1,j-1}\} \ge \sum_{0 \le l \le j-1} r_{n,l} \qquad (3\text{-}4)$$

证明：因为(3-4)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 $t_1,...,t_n,t'_{n+1},...,t'_N$,(3-4)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(3-4)，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)。

3.3.3.3 算法 1 与算法 2 的比较

对于各个元组都是独立的情况下，算法 1 与算法 2 的比较如下：

表 3-4 两种U-kRanks查询算法比较

	时间	空间
算法1	n^2k	nk
算法2	nk	k

3.4 PT-k 查询

3.4.1 PT-k 查询定义

文献[5]提出了 PT-k 查询。

给定一个阈值 p,PT-k 返回所有在可能世界实例中成为 top-k 的总概率超过阈值 p 的元组。其公式表示如下：

$$Answer(Q,p,T)=\{t \mid t\in T, \Pr_Q^k(t)\geq p\}$$

要理解这条公式的意义，还必须要了解作者提出的其他相关的概念。

一个 Top-k 查询表示为 Q^k（P，f），包含一个谓词 P，一个等级排序函数 f，和一个整型 k>0。

Q^k（W）表示 Top-k 在可能世界 W 上查询 Q 返回的元组集合，它包含 k 个元组。

PT-k 在一个不确定表 T 的查询包括一个 Top-k 查询 Q 和一个概率阈值 p(0<p≤1).对于每个可能世界 W，Q 被应用并且 k 个元组的集合 Q^k（W）被返回。

对于一个元组 t∈T，t 的 Top-k 概率就是对于所有的$W\in\omega$,t 出现在 Q^k（W）的概率的和。表示为：

$$\Pr_{Q,T}^k(t)=\sum_{W\in\omega,t\in Q^k(W)}\Pr(W)$$

3.4.2 PT-k 查询举例

考虑 3.1 节的例子：

各个元组在所有可能世界实例成为 top2 的概率总和分别为：

p(t1)=p(w1)+p(w2)+p(w3)+p(w4)+p(w5)+p(w6)=0.3

p(t2)=p(w1)+p(w2)+p(w7)+p(w8)=0.4

p(t3)=p(w4)+p(w9)+p(w10)=0.38

p(t4)=p(w6)+p(w8)+p(w10)+p(w11)+p(w12)=0.202

p(t5)=p(w3)+p(w5)+p(w7)+p(w9)+p(w11)=0.704

p(t6)=p(w12)=0.014

假设给定的阈值 p 为 0.35 则，PT-2 返回的结果为<t2,t3,t5>

假设给定的阈值 p 为 0.39 则，PT-2 返回的结果为<t2,t5>

假设给定的阈值 p 为 0.5 则，PT-2 返回的结果为<t5>

从本例可以看出，PT-k 查询返回的结果所包含的元组个数与阈值 p 有关，p 越大，那么符合条件的元组就越少，返回结果的元组个数就越少。

3.4.3 PT-k 查询的算法

文献[5]提出了 PT-k 查询的定义，并且给出了一个高效的算法：首先，根据谓词 P 去掉不确定表 T 中不符合 P 的元组并且对各个元组按照得分函数 F 进行排序；然后对 P(T)中的每个元组 t 计算它的压缩统治集以及该统治集的子集概率；接着，计算 t 成为 Top-k 的概率 $Pr^k(t)$，如果 Pr^k (t)超过阈值 p 则输出 t；最后利用 t 来对未计算的元组进行剪枝，如果剩下未计算的所有元组都不能满足阈值 p 则退出循环。

在介绍该算法的时候，作者首先假设各个元组是独立的，并且已经按照得分函数进行排列。如果各个元组不是独立的，那么可以通过把产生规则中的元组压缩为一个规则元组；如果各个元组不是按照得分函数进行排序的，那么可以按照某种方法对一个元组的压缩统治集进行排序。

要理解该算法，除了要知道 3.4.1 提到的各个概念之外，还必须要理解统治集、规则元组压缩、通过共享前缀来减少扫描、剪枝技术等概念及其相关的定理、推论。

考虑在一个不确定表 T 进行 Top-k 查询 Q^k(P，f)。满足查询谓词的元组集合表示为

$$P(T)=\{t \mid t \in T \wedge P(t)=true\}$$

P(T)同样是不确定表，它里面的各个元组依然包含概率，进一步，T 中的生成规则 R 除去不在 P(T)中的元组，然后映射到 P(T)。因此，PT-k 查询就是从 P(T）中找出 Top-k 概率超过阈值的元组。

3.4.3.1 统治集

P(T)包含了所有满足查询的元组、元组的概率以及生成规则。去除不在 P(T)的元组不会影响 Top-k 查询结果。因此，$Answer(Q,p,T)=Answer(Q,p,P(T))$。我们在 Top-k 查询的时候只需要考虑 P(T)。

在 P(T)中的一个元组 t 是否成为 Top-k 查询结果只受那些排列在 t 前面的元组的影响。

统治集（dominant set）：对于 P(T)中的一个元组 t，它的统治集就是 P(T)中所有排在 t 前面的元组的集合。表示为 $S_t=\{t' \mid t' \in P(T) \wedge t' \prec_f t\}$

定理 3： 统治集属性(The dominant set property)：对于一个元组 t∈T，$Pr^k_{Q,T}(t)=Pr^k_{Q,S_t}(t)$。

3.4.3.2 基本的案例

在基本的案例当中，作者假设所有的元组是独立的，L=t1……tn 是 P(T)中的所有元组并且是按照得分函数进行排序的。

一个元组出现在一个可能世界的第 j 个位置当且仅当它的统治集有 j-1 个元组出现。

位置概率(position probability)：在一个可能世界中一个元组出现在第 j 个位置的概率，表示为 Pr(ti, j)。

子集概率 Pr(Sti, j)：在一个可能世界中 Sti 出现 j 个元组的概率。

$\Pr(\phi,0)=1$，且 $\Pr(\phi, j)=0$ （$0<j\le n$）。

一个元组出现在一个可能世界的第 j 个位置的概率等于该元组出现的概率乘以该元组的统治集有 j-1 个元组出现在该可能世界的概率，公式为：

$$\Pr(t_i, j)=\Pr(t_i)\Pr(S_{t_i}, j-1) \tag{3-5}$$

显然，一个元组 ti 的 Top-k 概率为这个元组出现在第 1,2,3...k 个位置的概率和，表示公式为：

$$\Pr^k(t_i)=\sum_{j=1}^{k}\Pr(t_i, j)=\Pr(ti)\sum_{j=1}^{k}\Pr(S_{ti}, j-1) \tag{3-6}$$

显然，当 $i\le k$ 的时候，一个元组 ti 成为 Top-k 的概率等于它本身的概率，也就是 $\Pr^k(t_i)=\Pr(t_i)$。

定理 4： 对于 $1\le i, j\le|T|$，有：

1) 元组 ti 的统治集 Sti 出现 0 个元组的概率为 ti-1 不出现的概率与 ti-1 的统治集出现 0 个元组的概率的乘积，公式为：

$$\Pr(S_{ti},0)=\Pr(S_{ti-1},0)(1-\Pr(t_{i-1}))=\prod_{j=1}^{i-1}(1-\Pr(t_{i-1}))$$

2) 元组 ti 的统治集 Sti 出现 j 个元组的概率为 ti-1 出现的概率与 ti-1 的统治集出现 j-1 个元组的概率的乘积与 ti-1 不出现的概率与 ti-1 的统治集出现 j 个元组的概率的乘积的和，公式为：

$$\Pr(S_{ti}, j)=\Pr(S_{ti-1}, j-1)\Pr(t_{i-1})+\Pr(S_{ti-1}, j)(1-\Pr(t_{i-1}))$$

运用定理 2 来计算各个元组成为 Top-k 的概率要求各个元组是独立的，时间复杂度为 O(kn)，n 是不确定表 T 里面的元组的个数。

定理 4 是一种基本的情形，所有的元组都是独立的，如果元组之间不是独立的，那么在计算元组 t 的 Top-k 概率时就要考虑 t 与生成规则 R 的关系了。

3.4.3.3 规则元组的压缩

在计算元组 t 的 Top-k 概率时，t 与生成规则 R 的关系可以细分为下面四种：

1) 生成规则 R 的所有元组都排列在 t 后面。根据定理 1，R 可以被忽略。
2) 生成规则 R 的所有元组都排列在 t 前面。此时 R 被叫做完全（complete）尊重 t，

在一个可能世界中 R 最多有一个元组出现。根据定理 1，可以把 R 中的所有元组看作一个规则元组（rule-tuple)t_R，t_R 的概率为 Pr(R)。

3) t 排列在 R 的元组之间（此时 R 被叫做开放（open）尊重 t)，并且 t 不是 R 的元组。这时候可以把 R 的元组分为两部分，t 前面的部分叫做 R_{left}，t 后面的部分叫做 R_{right}。与 2）类似，可以把 R_{left} 部分压缩成一个规则元组 t_{left}，它的概率为 Pr（R_{left}），而 R_{right} 则可以被忽略。

4) t 排列在 R 的元组之间，并且 t 是 R 的元组。因为 t 出现了，所有 R 中的其他元组都不能出现。根据定理 1，这时在计算 $\Pr^k(t)$ 的时候只需要考虑排在 t 前面的并且不在 R 中的元组，也就是 $S_t - \{t' | t' \in R\}$。

根据 2）可以得出推论 1：

推论 1： 规则元组压缩（Rule_tuple compression)：对于一个元组 $t \in P(T)$，如果 $\forall t' \in R$，$t' \prec_f t$ 那么 $\Pr^k_{Q,T}(t) = \Pr^k_{Q,T(R)}(t), T(R) = (T - \{t \mid t \in R\}) \cup \{t_R\}$，tR 就是 R 中统治 t 的元组，$\Pr(t_R) = \Pr(R)$，其他 T 中的生成规则依然留在 T(R).

根据 4）可以得出推论 2：

推论 2： 规则中的元组（Tuple in rule)：对于一个元组 t∈R，P(t）=true 并且|R|>1，$\Pr^k_{Q,T}(t) = \Pr^k_{Q,T'}(t)$，T'为不确定表，$T' = (S_{t_i} - \{t' | t' \in R\}) \cup \{t\}$。

元组 t 的压缩统治集（compressed dominant set）就是将 St 中所有的依赖元组压缩为规则元组或者移除（与 t 在同一规则）后的结果集，表示为 T(t)。

因此，对于一个元组 $t \in P(t)$，所有在 $T(t) \cup \{t\}$ 的元组都是独立的，于是 $\Pr^k_{Q,T}(t) = \Pr^k_{Q,T(t) \cup \{t\}}(t)$，这样就可以根据定理 4 来计算 $\Pr^k$(t)，并且只需要扫描一次 T(t)。

我们可以根据得分函数将 P(T)里面的所有元组进行排序，得到一个有序链表 L。对于每个元组 ti,扫描一次它前面的元组，得到一个压缩的统治集 T(ti)，在这个统治集里面所有的元组都是独立的，这样我们就可以使用定理 4 来计算 $\Pr^k$(ti)了，此时只需考虑 $T(t_i) \cup \{t_i\}$ 集合里面的元组。这样，计算所有元组的 Top-k 概率的实际复杂度为 $O(n^2)$，n 是不确定表的元组的个数。

3.4.3.4 通过共享前缀来减少扫描

两个压缩统治集可以通过共享前缀而减少计算子集概率的开销，而这种开销在 Top-k 查询的时候是比较主要的。

公式 3-6 表明，使用子集概率 $\Pr^k$(Sti,j)来计算 $\Pr^k$(ti)的时候，在 Sti 中的元组的顺序如何并没有关系。这就使得我们可以通过排序不同元组的压缩统治集从而使得对应的子集概率值能够尽可能多的被共享。

文中提出了两种方法来排序元组：

1) 积极的方法（aggressive method)：把所有独立的和完整的规则元组放在开放的规则

元组之前，而开放规则中压缩成的规则元组 tr 则按照开放规则中未被压缩的第一个元组 t 的先后来排序，t 出现的越后，则 tr 排到越前。

2) 懒惰的方法（lazy method）：尽可能多的利用 T(ti-1)中的共有前缀，剩下的元组按照 1）中的方法来排序。

假设 L 为 P(T)中所有元组的有序链表，L(ti)表示 T(ti）中所有元组的有序链表，Prefix(L(ti),L(ti+1))为 L(ti)和 L(ti+1)的最长的共同前缀，那么，总共需要计算的子集概率的数目为：

$$Cost = \sum_{i=1}^{n-1}(|L(t_{i+1})| - |\Pr efix(L(t_i), L(t_{i+1}))|) \qquad (3\text{-}7)$$

作者通过实验证明，懒惰方法总比积极方法要好。由于懒惰法比积极法尽可能多的利用了前一个元组的统治集的计算结果，所以需要计算的元组要比较少。

3.4.3.5 剪枝技术

一个 PT-k 查询只对那些 Top-k 概率超过阈值的元组感兴趣，因此可以通过剪枝的方法来避免检索所有的元组是否符合查询谓词。作者给出了四条定理来进行剪枝。

定理 5： 通过成员概率来剪枝(Pruning by membership probability) $\Pr^k(t) \le \Pr(t)$。进一步来说，对于一个独立元组 t，如果 $\Pr^k(t) < p$，那么：

对于任意没有统治 t 的独立元组 t'，并且 $\Pr(t') \le \Pr(t)$，并且 $\Pr^k(t') < p$；

对任意多元组生成规则 R 使得 t 的排序高于 R 中的所有元组并且 $\Pr(R) \le \Pr(t)$ 以及对于任意 $t'' \in R$ 有 $\Pr^k(t'') < p$

定理 6： 通过相同生成规则的其他元组来剪枝 (Pruning by tuples in the same rule) 对于出现在同一多元组生成规则 R 中的元组 t 和 t'，如果 t' 没有统治 t 并且 $\Pr(t) \ge \Pr(t')$ 并且 $\Pr^k(t) < p$ 那么 $\Pr^k(t') < p$。

定理 7： 通过总共的 Top-k 概率来剪枝 (Pruning by total top-k probability) 令 A 为一系列成为 Top-k 概率超过阈值 p 的元组的集合。如果 $\sum_{t\in A}\Pr^k(t) > k - p$，那么对于每个没有在 A 中的元组 t' 有：$\Pr^k(t') < p$。

这个定理是基于这么一个事实的：所有元组成为 Top-k 的概率的总和为 k。

用反证法来证明定理 5：假设对于某个不属于 A 的元组 t'有：$\Pr^k(t') \ge p$，那么

$\sum_{t\in A}\Pr^k(t) + \Pr^k(t') > (k-p) + p = k$,即

$\sum_{t\in A}\Pr^k(t) + \Pr^k(t') > k$

这就违背了上面提到的基本事实，于是定理 7 成立。

定理 8： 一个紧凑的停止条件（a tight stopping condition)。令 t1,……,tm,……,tn 为等级排序后的元组。假设 L=t1,……,tm 已经被读取了，令 LR 为开放尊重 tm+1 的规则的集合。对于任意 ti(i>m),有

如果 ti 不是 LR 中的元组，那么 ti 的 Top-k 概率 $\Pr^k(t_i) \le \sum_{j=0}^{k-1} \Pr(L, j)$

如果 ti 是 LR 中的元组，那么 ti 的 Top-k 概率 $\Pr^k(t_i) \le \max_{R \in LR}(1-\Pr(t_{R_{left}}))\sum_{j=0}^{k-1} \Pr(L - t_{R_{left}}, j)$

证明：显然，对于任意的不确定表 T1,T2，如果 $T_1 \subseteq T_2$，那么 $\sum_{j=0}^{k-1} \Pr(T_2, j) \le \sum_{j=0}^{k-1} \Pr(T_1, j)$。

对于第一项，考虑 ti 的压缩统治集 T(ti)，因为 $L \subseteq T(t_i)$，所以 $\sum_{j=0}^{k-1} \Pr(T(t_i), j) \le \sum_{j=0}^{k-1} \Pr(L, j)$。

于是：

$$\Pr^k(t_i) = \Pr(t_i)\sum_{j=0}^{k-1} \Pr(S_{t_i}, j) = \Pr(t_i)\sum_{j=0}^{k-1} \Pr(T(t), j) \le \sum_{j=0}^{k-1} \Pr(T(t), j) \le \sum_{j=0}^{k-1} \Pr(T(t), j)$$

对于第二项，设 ti 出现在一个 open 规则 R∈LR.因为 $\Pr(t_i) + \Pr(t_{R_{left}}) \le \Pr(R) \le 1$，所以 $\Pr(t_i) \le 1 - \Pr(t_{R_{left}})$。对于 ti 的压缩统治集 T(Ti)，$(L - t_{R_{left}}) \subseteq T(t_i)$，因此：

$$\Pr^k(t) = \Pr(t)\sum_{j=0}^{k-1} \Pr(T(t), j) \le (1-\Pr(t_{R_{left}}))\sum_{j=0}^{k-1} \Pr(L - t_{R_{left}}, j)$$

定理 8 提供了还没有被检测的元组的上边界。如果两个上边界都低于概率阈值 p，那么所有还没有被检测的元组都不用检测了。

3.5 Pk-topk 查询

3.5.1 Pk-topk 查询定义

文献[16]借鉴了文献[5]提出的 PT-k 查询，提出了 Pk-topk 查询。

Pk-topk 查询返回在所有可能世界实例中成为 top-k 的总概率最大的 k 个元组。

Pk-topk 查询与 PT-k 语义很类似，不过没有阈值 p，返回的结果总是 k 个元组。

3.5.2 Pk-topk 查询举例

考虑 3.1 节的例子：

各个元组在所有可能世界实例成为 top2 的概率总和分别为：

p(t1)=p(w1)+p(w2)+p(w3)+p(w4)+p(w5)+p(w6)=0.3

p(t2)=p(w1)+p(w2)+p(w7)+p(w8)=0.4

p(t3)=p(w4)+p(w9)+p(w10)=0.38

p(t4)=p(w6)+p(w8)+p(w10)+p(w11)+p(w12)=0.202

p(t5)=p(w3)+p(w5)+p(w7)+p(w9)+p(w11)=0.704

p(t6)=p(w12)=0.014

根据 PT-k 的定义，在所有可能世界实例中，最可能成为 Top-2 的两个元组为 t2 和 t5，因此返回结果为<t2,t5>。

从本例可以看出，Pk-topk 的计算与 PT-k 类似，不过就是没有阈值 p，Pk-topk 返回的

结果总是包含 k 个元组的。

3.6 p-skyline 查询

3.6.1 p-skyline 查询定义

Borzsonyi 等人[18]给出了确定性数据库中 skyline 查询的定义：给定一个确定性的 n 维数据集合 D，任一点 d 可被表示为(d.D1,...,d.Dn)。Skyline 查询返回数据集合 S，$S \subseteq D$ 则 $\forall u \in S$ 不存在其它点 v，满足（1）对于任一维度 i($1 \le i \le n$)，$u.D_i \le v.D_i$ (2)存在一个维度 j($1 \le j \le n$)使得 $u.D_j < v.D_j$。

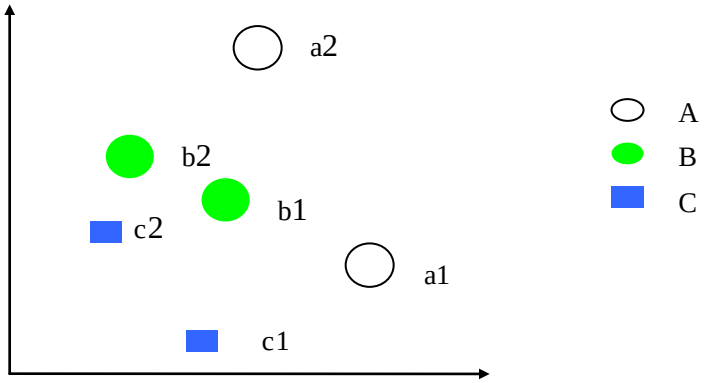
不确定性数据库会衍生出很多可能世界实例,各元组在各可能世界实例中可能是 skyline 点，也可能不是 skyline 点。Pei[19]给出了不确定数据库中 p-skyline 查询的定义：

给定一个概率阈值 p(0≤p≤1)，p-skyline 查询的返回结果就是所有成为 skyline 点的概率超过 p 的不确定对象的集合。

3.6.2 p-skyline 查询举例

假设有不确定对象 A,B,C，他们各自包含 2 个不确定实例，这些实例出现的概率为 1/2，并且有且仅有一个实例出现在某个可能世界中。假设数据集合为二维的，各个不确定实例的关系如下图。

图 3-1 不确定实例的关系



所有可能世界实例及其概率见下表：

表 3-5 所有可能世界实例

可能世界实例	概率	skyline结果
W1={a1,b1,c1}	0.125	{a1,b1}
W2={a1,b1,c2}	0.125	{a1,b1}
W3={a1,b2,c1}	0.125	{a1,b2}
W4={a1,b2,c2}	0.125	{a1,b2}
W5={a2,b1,c1}	0.125	{a2}
W6={a2,b1,c2}	0.125	{a2}
W7={a2,b2,c1}	0.125	{a2}
W8={a2,b2,c2}	0.125	{a2}

考虑可能世界实例 W1:根据 skyline 的定义，a1 不能成为 skyline 点，因为存在实例 a1，它在各个维度（即 X 和 Y)的取值都比 c1 要大。b1 成为 skyline 点，因为 b1 在 Y 方向的取值比 a1,c1 要大；a1 成为 skyline 点，因为 a1 在 X 方向的取值比 b1,c1 要大。

对 W2,W3,W4 的分析方法类似。

对于 W5,W6,W7,W8 这四个可能世界实例，只有 a2 成为 skyline 点。因为 a2 在各个维度的取值都比 b1,b2,c1,c2 要大。

由于各个可能世界实例都包含不确定对象 A 的实例（a1 或者 a2），所有 A 成为 skyline 点的概率为 1。B 成为 skyline 点的概率为前四个可能世界实例的概率总和，即 0.5。C 成为 skyline 点的概率为 0，因为 C 所有实例（c1,c2）都不能成为 skyline 点。

因此，当阈值 p 的值小于 0.5 的时候，p-skyline 查询返回 A 和 B；当当阈值 p 的值大于 0.5 的时候，p-skyline 查询返回 A。

3.6.3 针对 p-skyline 查询的算法

文献[19]同时提出了两种解决方法:自下而上方法(bottom up method) 和自上而下方法(top down method)，分别采用不同的定界、剪枝、精化等启发式规则进行迭代处理。

自底向上的方法引入了两个特殊实例 Umin 和 Umax,它们分别表示取 U 的所有实例在各维上的最小值和最大值所组成的实例。我们可以限定 U 的概率为 Pr(Umin）≥Pr(U）≥Pr(Umax)。使用下面四种剪枝策略可以加快查询：1)如果 Pr(Umin)<p，那么 U 不是 p-skyline 元素;如果 Pr(Umax）≥p，那么 U 在 p-skyline 中；2）假设对于 U 中任意实例 u，Pr+(u)和 Pr-(u）分别是 u 的概率上限和概率下限，如果 Pr+(u)的平均值小于 p，那么 U 不在 p-skyline 中;如果 Pr-(u)的平均值大于等于 p，则 U 是 p-skyline 的元素；3）假设 U 和 V 是两个不同的对象，如果 U 的一个实例 u 满足 Vmax→u，那么 Pr(u）=0；4）假设 U 和 V 是两个不同的对象，U'是 U 的一个子集并且 Umax'→Vmin，如果 $(|U-U'|/|U|)\times\min_{u\in U'}\{pr(u)\}$ 小于 p，则 V 不在 p-skyline 中。为了减少每个不确定对象实例概率的计算，可以将对象的实例分层，概率按层次递减。基于上述策略，自底向上的方法使用堆和 Rtree 等结构组织算法。

自顶向下的方法为每个不确定对象建一棵划分树。划分树是二叉树，建造过程类似于 kd-tree。页节点包含实例及最小限界矩形，中间节点维护后代的最小限界矩形。对于划分树的一个节点 N，其最小限定矩形的左下角和右上角的顶点分别用 Nmin 和 Nmax 表示。N 中任意一个实例 u 的概率可以限定为 Pr(Nmin）≥Pr(u）≥Pr(Nmax)。进一步如果不确定对象 U 的划分树有 n 个叶节点 $N_1,\ldots,Nn$，那么 U 的概率可用不等式 $\frac{1}{|U|}\sum_{i=1}^{n}|N_i|\Pr(N_{ima})\le\Pr(U)\le\frac{1}{|U|}\sum_{i=1}^{n}|N_i|\Pr(N_{imi})$ 限定。剪枝策略:1）假设 N 是不确定对象 U 的划分树的一个节点，如果存在一个对象 V 使得 Vmax→Nmin，那么 N 可以被剪枝;2）假设 N 是不确定对象 U 的划分树的一个节点,如果 Pr(Nmin)=Pr(Nmax),那么 N 可以被剪枝;3)假定 p 是给定的极限值，如果不确定对象 U 的划分树有 n 个叶节点 N1,...,Nn，如果

$\frac{1}{|U|}\sum_{i=1}^{n}|N_i|\Pr(N_{i\max}) \le \Pr(U) \le \frac{1}{|U|}\sum_{i=1}^{n}|N_i|\Pr(N_{i\min})$ 中最左侧表达式的值大于等于 p，那么 U 是 p-skyline 元素，如果最右侧表达式的值小于 p，那么 U 不在 p-skyline 中。

3.7 针对四种 Top-k 查询的算法

Jin 等人[16]最早提出了面向滑动窗口模型的查询处理方法来处理各种 Top-k 查询。他们提出了一种针对 Top-k 查询的框架。首先，可以针对各种 Top-k 查询设计 compact set，各个 compact set 含有一部分数据。这个 compact set 具有两个特性：(1) 能够计算 Top-k 查询结果； (2) 能够增量维护。但是 compact set 仍然不足以回答滑动窗口模型，因此，可以将多个 compact sets 进行压缩，降低空间复杂度与时间复杂度。他们提出的 SCSQ-buffer 策略在时间复杂度与空间复杂度上均是优秀的。

第四章 不确定性数据的分析处理

4.1 世系分析

数据的世系(lineage 或者 provenance) 是指数据产生并随着时间推移而演变的整个过程[20]。现有工作大多针对确定性数据库，但很多应用(特别是超大规模应用) 往往需面对不精确数据集合，而非精确的数据集合，包括科学数据管理、传感器数据管理、近似查询处理、隐私保护等[21]。

Trio 项目最早研究如何整合不确定性与数据的世系[21]。该项目定义了 ULDB (Uncertainty Lineage Database)，面向世系分析的不确定性数据库，并证明它是完全的(complete)；提出了执行关系操作的算法[22,23]。

当数据的世系比较复杂时，如何计算查询结果的概率就成为一件困难的事情。当不考虑概率因素时，可以为某一个数据世系设计多种查询计划，并且得到相同结果。但是当存在概率因素时，不同查询计划返回的查询结果的概率值却可能不同。其原因是在设计查询计划的过程中未考虑到数据的相关性特征，导致重复计算[23]。文献[24]提出了一种数据计算与概率计算解偶的技术，使两者可以分开计算，这样一方面概率值可以采用传统的关系型数据库方法进行计算，另一方面，如果用户并不关注查询结果的概率，则可以节约计算概率值的开销。

4.2 数据流分析

为处理数据流，必须设计单遍扫描算法，仅消耗少量内存，且能实时监测查询结果。近期的研究工作拓展到了不确定性数据流上，即流上的各元组都是不确定性元组。可以充分借鉴面向确定性数据流的工作成果，以设计针对不确定性数据流的新方法，包括计算 F2 的 pAMS 方法[25]、计算相异元素个数的 pFM 方法[25]和生成聚类的 ECF 方法[26]等。滑动窗口模型是数据流上的重要模型，仅考虑最近的 N 个元组，金澈清等人提出了在滑动窗口上计算 top-k 查询的新方法[16]。最新的研究工作还包括对事件数据流的处理，特别是在 RFID 环境中[27]。

4.3 非确定性数据库中空值处理

虽然非确定性数据库中引入了置信度、可能性集合等概念，但是并不能完全解决空值问题，比如，在一个记录病人信息的数据库中，病人信息表的一个属性“是否有小孩”，这个属性值对我们来说是不可知的(除病人提供)，置信度和可能性集合在这里失效了。如果在非确定性数据库沿用关系数据库中的空值概念，又至少存在以下不足：

1)在语义上模糊，而且根据空值的定义，在非确定性数据库中空值缺乏表现多个可能取值的能力；

2)在对空值进行检索中，可能会造成信息的丢失和不完整；

3)由于空值的未知性，在很多商业应用中，空值往往是被忽略的，这也造成很多决策信息的不完整和不精确。所以需要一个更为系统的方法来处理空值，这也是本文的主要研究内容。

姜小华等人[28]利用 NK（不可知）、NV（不可用）、NVK（不可知也不可用）和 NE（相对空）来取代传统数据库中用 NULL 来表示空值的方法，使得概念上更加清晰、查询和检索更加精确、更真实地反应了现实世界的情况。

NK、NV、NVK 和 NE 的应用举例：

表 4-1 空值处理的例子

ID	姓名	年龄	孩子	丈夫名字	出生日期	地址
01	张三	28	NVK	NV	1910-5-12	NK
02	李四	NK	NK	王五	NK	NE

上表完全摆脱了空值，取而代之的是 NV、NK、NVK 和 NE，这样在语义上非常清晰。从表 4-1 知道张三肯定没有结婚，因为其丈夫名字属性值为 NV 的。也很容易知道李四的年龄和出生日期是不可知的。如果要检索所有未结婚的女员工，只需检索丈夫名字属性值为 NV 就行了，因为只有未婚女性的丈夫名字为不可用的。在用空值表示的时候是无法表达这么明确的意义而且无法检索出符合实际的结果。如果从表 3 检索所有年龄小于 30 的员工返回的是员工 01 和 02，因为员工 02 的 AGE 是 NK 的，不能被忽略掉。

4.4 关系代数处理

关系代数处理研究工作开展得较早，从 20 世纪 80 年代后期开始到现在一直在延续。首先，在各关系表中添加属性字段以描述元组的不确定性；然后，数据库接受类 SQL 语言的查询语句，并且进行处理。这种类 SQL 语句往往可以转化为标准的 SQL 语句进行查询。查询结果中还包括概率字段以描述查询结果的准确性。

Andritsos 等人提出了查询重写技术来处理查询[29]。Sen 等人则利用构建图模型的方法来处理 SQL 语句[30]。数据库上的查询评价问题看作是基于概率图模型上的推断问题。文中应用概率图模型来刻画元组间的相关关系,并应用概率图模型来分解联合概率分布,通过分解后因子表达式中的条件概率分布计算并返回查询概率。要完整地计算出查询结果有时是非常困难的。研究结果表明，一个含连接操作的查询要么相对于概率数据库在多项式复杂度的时间内被计算出来,要么相对于概率数据库是#P-完全问题[31,32]。

4.5 数据挖掘

数据清洗过程会丢失数据的部分特征，降低查询结果质量，因此如何在不确定性数据上直接做挖掘算法就显得很重要。聚类技术和分类技术都可以用于数据清洗。数据挖掘能从一堆纷繁芜杂的数据中抽取有用知识。当原始数据的精确度不高时，传统方法需要首先对数据进行清洗，利用插值、采样、回归、标准化等方法提高数据质量，然后再运行聚类算法。然而，数据清洗过程会丢失数据的部分特征，降低查询结果质量，因此如何在不确定性数据上直接做挖掘算法就显得很重要。聚类技术得到了最广泛的研究，包括 k-means 算法的改进版本 UK-means 算法[33]、基于密度的 FDBSCAN 算法[34]。

文献[33]首先将不确定性数据的数据挖掘作为一个新的研究方向提出来，并以数据聚类为例根据经典的 Kmeans 聚类方法提出针对不确定性数据的 U Kmeans 聚类方法。文献[5]提出的方法简单地用两个不确定性对象之间距离的期望值代替实际值，从而丢失了对象位置在其不确定区间内的分布信息。况且 Kmeans 聚类方法具有不适宜发现非凸形状簇、对噪声和离群点敏感等缺点，因此文献[33]提出的方法实用性有限。

文献[34]在著名的基于密度的聚类方法DBSCAN[35]的基础上考虑数据的不确定性,提出针对不确定性数据的 FDBSCAN 聚类方法。该方法虽然利用到概率分布信息,但是在对对象的不确定区间进行代价较高的离散化抽样计算后，只简单地将计算得到的核心对象概率和密度可达概率是否大于 0.5 分别作为对象是否是核心对象和是否可达的判断标准，这对正确聚类结果的获得都是不利的。

许华杰等人[36]提出的聚类方法 PDBSCAN 充分利用对象位置在其不确定区间内的概率分布信息定义及计算核心对象概率和密度可达概率,并采用概率索引技术提高聚类算法的效率。PDBSCAN 聚类算法有以下特点：

1）对概率核心对象和概率密度可达的计算并不像有的文献那样生硬地将两个不确定性对象(区域) 之间的距离用单个值(如距离的期望值) 来代替，而是利用两个不确定性对象之间的距离的最小值和最大值作为限定范围，并考虑不确定性在该范围上的概率分布；

2）算法在判断概率核心对象和概率密度可达时考虑不确定性对象概率分布，允许用户在计算精度和效率之间进行权衡，设置概率阀值 p，而不是简单地将概率是否大于 0.5 作为核心对象和密度可达的判断标准；

3）通过 R 树和概率阀值索引 PTI 这两种索引方法提高计算效率。

文献[36]通过分析比较得出：PDBSCAN 聚类算法在有效性以及效率方面都比 FDBSCAN 聚类算法要优秀。

第五章 索引技术

索引技术是数据管理技术的重要内容。有效的索引能够大幅提高查询效率。关系型数据库往往采用 B+树及其变种为一维数据建立索引；在多维数据管理领域或时间-空间数据管理领域，广泛使用 R 树以及其变种进行索引。这些索引技术均能够大幅提高查询处理速度。同理，在处理不确定性数据中也需要关注索引问题。在某些查询任务中，例如 top-k 查询，元组的概率值也非常重要，因此需要针对概率维度创建一维索引，此时传统索引技术有效。但传统的索引技术无法解决所有问题。当各元组的取值必须通过概率分布函数描述，且概率分布函数无法预先指定时，传统的索引技术就无法胜任了。目前较好的方法有概率阈值索引技术（PTI）[37]和 U-Tree 技术[38]。前者针对一维数据，后者针对多维空间数据。

5.1 概率阈值索引(PTI)技术

在移动对象数据库等应用中，物体在某时刻的位置可能并非确定，仅知道在一个较大的区域之内，一般用概率密度函数(pdf) 描述。R 树以及其变种是为高维数据创建索引的重要手段，一种方法是以 pdf 的覆盖范围作为该物体的实际空间尺寸，并建立索引。但是这个方法的可行性不高，原因在于 pdf 所覆盖的范围可能很大，而物体最可能出现的地方却是其中的一小块地方。构建索引必须考虑 pdf 的分布特征。

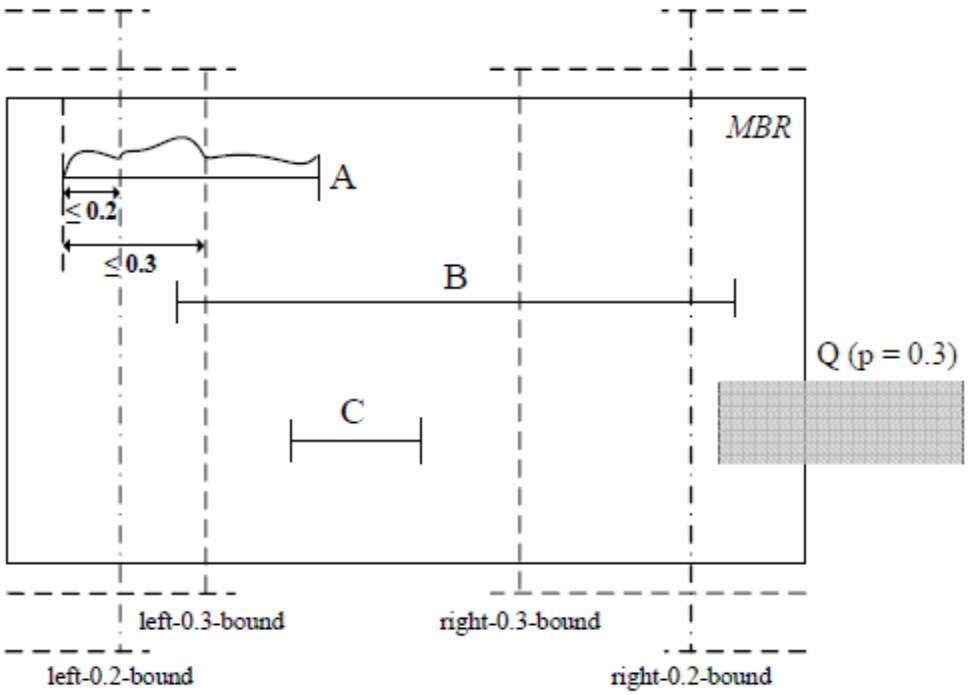
5.1.1 PTI 的定义

概率阈值索引(Probability Threshold Indexing, PTI) [37]能够实现对一维数据的索引。假设数据库中各元组的属性值对应一个一维区间[a,b]，可以设置多个 x-bound. 各个 x-bound 由两根线组成，在线的左边或右边，其总概率均不超过 x，令 Mi 表示第 i 个元组的 MBR，Mi.lb(x)和 Mi.rb(x)分别表示 x-bound 的左边和右边值，Li 和 Ri 分别表示最左边和最右边的值，fi 表示该元组的概率密度函数,则我们有 $\int_{Li}^{Mi.lb(x)} fi(y)dy \le x$ 和 $\int_{Mi.rb(x)}^{Ri} fi(y)dy \le x$。该做法能够避免一些额外的计算。

5.1.2 PTI 应用举例

考虑下图所示例子：

图 5-1[37] PTI应用举例



上图以一维间隔的形式描述了在一个更大的 MBR Mj 中的三个孩子 MBR(A,B,C)。同样显现了 2 个 x-bound，一个为 0.2-bound，另外一个为 0.3-bound。上图表明一个 x-bound 就是一对直线，在 MBR 的每个间隔最多有 x 跨过这两条直线。

上图还描述了 A 的不确定性 pdf，从中可以看出 $\int_{L_A}^{Mj.lb(0.2)} fi(x)dx \le 0.2$，并且 $\int_{L_A}^{Mj.lb(0.3)} fi(x)dx \le 0.3$。对于间隔 B，在右边 0.3-bound 的约束为 $\int_{Mj.rb(0.3)}^{R_B} fi(x)dx \le 0.3$。间隔 C 没有跨过 0.2-bound 和 0.3-bound，所以它满足这两个 x-bound。

一个范围查询 Q 通过内部结点进行测试。如果没有 x-bound 的帮助，那么在进行 Q 的时候必须要（1）检测哪一个 MBR（A,B,C）与 Q 的间隔有交集，（2）对于符合条件的 MBR(这里是 B)，进一步检测 B 所指向的结点，直到到达叶子级别，（3）在叶子级别计算间隔的概率。

有了 x-bound 的帮助能够减少很多不必要的计算。上例中，因为 right-0.2-bound 的右边的概率不会超过 0.2，而 Q 要求返回在某个范围出现概率超过 0.3。

一般而言，给定 MBR Mj 的一个 x-bound，如果下面的两个条件成立的话，我们能够消除对 Mj 的进一步计算。

1) [a,b]没有与 Mj 的左边以及右边的 x-bound 相交，也就是 b<Mj.lb(x)或者 a>Mj.rb(x)成立。

2) p≥x.

5.2 U-Tree 索引技术

PTI 只适用与一维空间并且这些方法的可行性是有限的，因为：（1）它们不能支持对象有任意的 pdf，（2）由于隐藏了复杂性保证，它们可能会引起实际运行很大的开销。

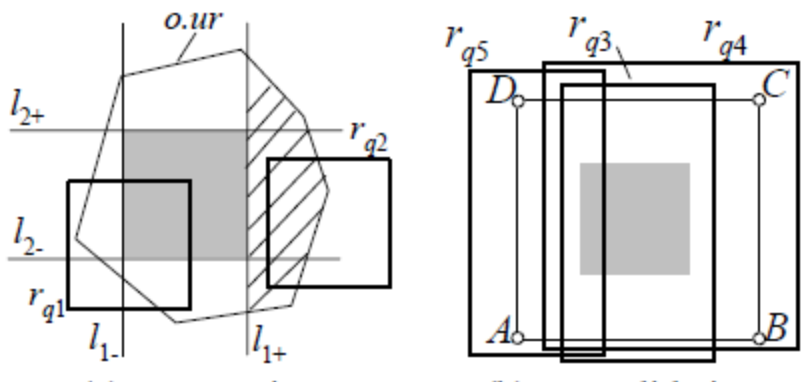
文献[38]提出了基于 R*-tree 的不确定对象索引策略 U-tree，其固有的良好的动态结构可以使数据对象以任何次序更新或插入，而且对不确定数据本身的概率密度函数 pdf 没有任何限制，实验结果表明，U-tree 具有良好的动态更新性能和域查询性能，域查询的磁盘 I/O 和 CPU 计算时间可以得到最大程度的优化。

5.2.1 概率约束区域（PCR）

U-tree 索引技术是基于概率约束区域（Probabilistically Constrained Region，PCR）这个概念的。这一节首先介绍 PCR 的概念以及它的特点。

对象 o 的 PCR 表示为 o.pcr(p)，p 的取值范围为[0,0,5]。下图(a)描述了一个二维的例子。

图 5-2[37] PCR 举例



(a)用于剪枝　　（b）用于验证

多边形表示不确定区域 o.ur，o.pcr(p)为图中灰色部分，它由四根线构成 l_{1-},l_{1+},l_{2-},l_{2+}。l_{1+}把 o.ur 分为左右两部分，o 出现在右边部分的概率为 p。相似的，o 出现在 l_{1-}左边，l_{2-}下面，l_{2+}上面的概率分别为 p。显然，o 出现在 l_{1-}和 l_{1+}（或者 l_{2-}和 l_{2+}）之间的概率为 1-2p。

使用 PCR 可以用于剪枝或者验证一个对象是否满足查询，而不必计算出这个对象出现的精确的概率。

考虑上图（a），假设 p=0.2，r_{q1},r_{q2} 分别为两个概率范围查询 q1,q2 的查询区域，它们的概率阈值分别为 p_{q1}=0.8,p_{q2}=0.2。对象 o 不能满足 q1，因为 o 出现在阴影部分的概率为 0.2，而 r_{q1} 没有与阴影部分相交，这就使得 r_{q1} 部分的概率不超过 1-0.2=0.8。再加上 r_{q1} 没有完全包含 o.pcr(0.2)，所以 o 出现在 r_{q1} 的概率必定小于 0.8，从而得知对象 o 不能满足 q1。O 不能满足 q2，因为 o 出现在阴影部分的概率为 0.2，而 r_{q2} 出现在 l_{1+}右边并且没有与 o.pcr(0.2)相交。

图（b）展示了如何利用 PCR 来进行验证一个对象。矩形 ABCD 为对象 o 的最小边界

区域(MBR)，它是能够包含 o.ur 的最小的矩形，表示为 o.MBR。考虑 q3,q4,q5 查询，它们的查询区域分别为 r_{q3},r_{q4},r_{q5}，概率阈值分别为 0.6,0.8,0.2。o 必定满足 q3，因为 r_{q3} 完全包含 l_{1-}和 l_{1+}之间的部分，而 o 出现在这个区域的概率为 1-0.2-0.2=0.6。同理，o 必定满足 q4 查询，因为 r_{q4} 完成包含 l_{1-}的右边部分，而 o 出现在这部分的概率为 1-0.2=0.8；o 必定满足 q5 查询，因为 r_{q5} 完成包含 l_{1-}的左边部分，而 o 出现在这部分的概率为 0.2。

上面的五个查询使用的剪枝、验证准则是不同的。

观察数据 1：对于一个搜索区域为 r_q，概率阈值为 p_q 的概率范围查询 q 有下面几个剪枝、验证准则：

1) 对于 p_q>0.5，如果 rq 没有完全包含 o.pcr(1-p_q)，那么可以移除对象 o 了。
2) 对于 p_q≤0.5，剪枝的条件是 r_q 与 o.pcr(p_q)不相交。
3) 对于任意的 p_q，如果对于所有 i∈[1,d]使得 r_q 完全覆盖 $o.pcr_{i-}(\frac{1-p_q}{2})$ 和 $o.pcr_{i+}(\frac{1-p_q}{2})$之间的 o.MBR 部分，那么对象 o 就肯定能够满足 q。
4) 对于 p_q>0.5，验证准则是对于所有 i∈[1,d]，r_q 完全包含 o.MBR 中 o.pcr_{i-}(1-p_q)的右边部分或者 o.pcr_{i+}(1-p_q)的左边部分。
5) 对于 p_q≤0.5，验证准则是对于所有 i∈[1,d]，r_q 完全包含 o.MBR 中 o.pcr_{i-}(p_q)的左边部分或者 o.pcr_{i+}(p_q)的右边部分。

上面的五个准则中，前面 2 个是用于剪枝的，后面三个是用于验证的。根据 p_q 与 0.5 的关系，一次只能应用三个准则到一个对象。如果 p_q>0.5，那么规则 1、4、3 可用，首先尝试应用规则 1 来进行剪枝，如果对象被剪枝了，那么规则 4 和 3 就不必用来验证了；如果 p_q≤0.5，规则 2、3、5 可用，依次使用规则 2、5、3 来进行剪枝和验证。

5.2.2 U-catalog 的引入

如果我们能够预先计算出所有 p∈[0,0.5]对应的 PCR，那么观察数据 1 的效率就能够最大化。因为所有 p∈[0,0.5]对应的 PCR 有无数个，所以效率最大化是不可行的。我们可以预先定义一些对各个对象都比较常用的 p，让它们构成一个 U-catalog。U-catalog 中的值按照升序表示为 p1,p2,…,pm，其中 m 为 U-catalog 的大小。m 的值越大，那么剪枝、验证的能力就越强，CPU 时间的开销就越少，但 I/O 和空间的开销就越大。

由于 U-catalog 的大小有限，给定某个阈值 p_q，与它对应的 PCR 可能不存在。这时候就需要以一种保守的方法来应用观察数据 1 中的剪枝、验证准则。为了能够剪枝对象 o，所选择的 PCR 必须要能够使得我们验证即使使用一个小于等于查询阈值 p_q 的概率，o 依然不能出现在查询区域 rq。为了能够验证对象 o，我们需要证明 o 出现在区域 r_q 的概率大于等于 p_q。

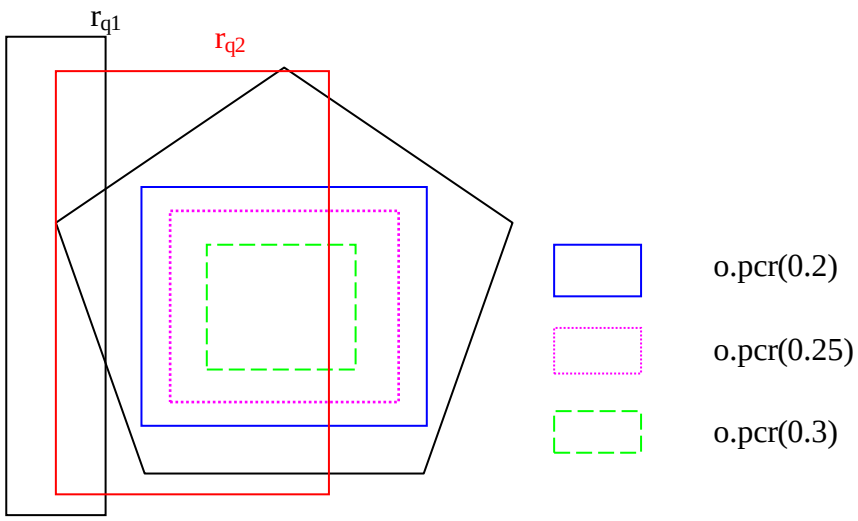
观察数据 2：对于一个搜索区域为 r_q，概率阈值为 p_q 的概率范围查询 q 有下面几个剪

枝、验证准则：

1) 对于 p_q>1-pm，如果 r_q 没有完全包含 o.pcr(p_j)，p_j(1≤j≤m)为 U-catalog 中不小于 1-p_q 的最小的那个值，那么可以移除对象 o 了。
2) 对于 p_q≤1-pm，剪枝的条件是 r_q 与 o.pcr(p_j)不相交，p_j(1≤j≤m)为 U-catalog 中不大于 p_q 的最大的那个值。
3) 对于任意的 p_q，如果对于所有 i∈[1,d]使得 r_q 完全覆盖 o.pcr$_{i-}$(p_j)和 o.pcr$_{i+}$(p_j)之间的 o.MBR 部分，p_j(1≤j≤m)为 U-catalog 中不大于（1-p_q）/2 的最大的那个值。那么对象 o 就肯定能够满足 q。
4) 对于 p_q>0.5，验证准则是对于所有 i∈[1,d]，r_q 完全包含 o.MBR 中 o.pcr$_{i-}$(p_j)的右边部分或者 o.pcr$_{i+}$(p_j)的左边部分，p_j(1≤j≤m)为 U-catalog 中不大于 1-p_q 的最大的那个值。
5) 对于 p_q≤0.5，验证准则是对于所有 i∈[1,d]，r_q 完全包含 o.MBR 中 o.pcr$_{i-}$(p_j) 的左边部分或者 o.pcr$_{i+}$(p_j)的右边部分。

考虑下图所示的例子，U-catalog 为{0.1,0,2,0,3,0.4,0.5}，假设有两个概率范围查询，它们的查询区域分别为 r_{q1} 和 r_{q2}，概率阈值分别为 0.25 和 0.75。

图 5-3 U-catalog 举例



这里 U-catalog 的大小为 5，也就是 m=5，pm=0.5。

对于第一个查询，因为 p_q=0.25<1-pm=0.5，因此先用准则 2 来进行剪枝。p_j(1≤j≤m)为 U-catalog 中不大于 p_q 的最大的那个值，也就是 p_j=0.2。因为 o.pcr(0.2)没有与 r_{q1} 相交，因此对象 o 可以被剪枝。

对于第二个查询，因为 p_q=0.75>1-pm=0.5，因此先用准则 1 来进行剪枝。p_j(1≤j≤m)为 U-catalog 中不小于 1-p_q=0.25 的最小的那个值，也就是 p_j=0.3。因为 r_{q2} 没有完全包含 o.pcr(0.3)，因此对象 o 可以被剪枝。

5.2.3 保守功能盒(CFB)

尽管 PCR 提供了一个高效的方法来剪枝、验证对象，但是它们是不适合用来做索引的，因为在索引结构里面的每一项都需要记录 m 个 PCR，m 就是 U-catalog 的大小。存储一个 PCR 需要 2d（d 为维度）个值，因此每一项需要包含最小 2d*m 个值，这就导致节点扇出随着维度 d 的增加而迅速下降。使用保守功能盒（Conservative Functional Box）能够解决这个问题。对于每个对象 o，使用 $o.cfb_{out}$ 和 $o.cfb_{in}$ 两个函数，它们分别叫外部的和内部的保守功能盒。对于每个 $p_j(1\leqslant j\leqslant m)$，$o.cfb_{out}(p_j)$返回一个 d 维的包含 $o.pcr(p_j)$的盒。类似的，$o.cfb_{in}(p_j)$返回一个 d 维的包含在 $o.pcr(p_j)$的盒。

为了能够使用 CFB 来进行查询处理，必须要适当的修改观察数据 2 中的剪枝、验证准则。

观察数据 3：对于一个搜索区域为 r_q，概率阈值为 p_q 的概率范围查询 q 有下面几个剪枝、验证准则：

1) 对于 $p_q>1-pm$，如果 r_q 没有完全包含 $o.cfb_{in}(p_j)$，$p_j(1\leqslant j\leqslant m)$为 U-catalog 中不小于 $1-p_q$ 的最小的那个值，那么可以移除对象 o 了。
2) 对于 $p_q\leqslant 1-pm$，剪枝的条件是 r_q 与 $o.cfb_{out}(p_j)$不相交，$p_j(1\leqslant j\leqslant m)$为 U-catalog 中不大于 p_q 的最大的那个值。
3) 对于任意的 p_q，如果对于所有 $i\in[1,d]$使得 r_q 完全覆盖 $o.cfb_{out}^{i-}(pj)$和$o.cfb_{out}^{i+}(pj)$之间的 o.MBR 部分，$p_j(1\leqslant j\leqslant m)$为 U-catalog 中不大于（$1-p_q$）/2 的最大的那个值。那么对象 o 就肯定能够满足 q。
4) 对于 $p_q>0.5$，验证准则是对于所有 $i\in[1,d]$，r_q 完全包含 o.MBR 中 $o.cfb_{out}^{i-}(pj)$的右边部分或者 $o.cfb_{out}^{i+}(pj)$的左边部分，$p_j(1\leqslant j\leqslant m)$为 U-catalog 中不大于 $1-p_q$ 的最大的那个值。
5) 对于 $p_q\leqslant 0.5$，验证准则是对于所有 $i\in[1,d]$，r_q 完全包含 o.MBR 中 $o.cfb_{in}^{i-}(pj)$ 的左边部分或者 $o.cfb_{in}^{i+}(pj)$的右边部分。

观察数据 3 稍微修改了观察数据 2 的几条准则，但依然是正确的，因为它放宽了限制条件。尽管使用 CFB 导致剪枝和验证能力降低了，但是它的空间开销变小了。这些节省的空间能够增加与之对应的索引结构的节点扇出，从而提高了查询性能。

作者给出了一个基于线性编程的计算 CFB 的方法。对于每个对象，它的 CFB 只需要在插入数据库的时候计算一次。

5.2.4 U-tree 的结构以及特点

U-tree 的一个中间项 e 包含一个指向其孩子节点的指针和两个 d 维的长方形 $e.MBR_{\perp}$和 $e.MBR_T$。$e.MBR_{\perp}$是 e 的子树中所有对象 o 的 $o.cfb_{out}(p1)$的 MBR，$e.MBR_T$ 是 e 的子树中所有对象 o 的 $o.cfb_{out}(pm)$的 MBR。p1 和 pm 分别为 U-catalog 中最小和最大值。$e.MBR(p1)=e.MBR_{\perp}$，$e.MBR(pm)=e.MBR_T$。需要注意的是，一个中间项 e 并没有包含任何它的子树中

的对象的关于 cfb_{in} 的信息。实际上一个 U-tree 是由 cfb_{out} 唯一决定的。尽管保存 cfb_{in} 的信息在非叶子节点可以减少查询开销，但这样做使得索引结构和更新算法变复杂了。

U-tree 是为了剪枝不包含任意结果的子树的，这种结构并没有加速验证过程，因为验证过程需要用到各个对象存储在叶子节点的具体信息。一个叶子项包含对象 o 的 $o.cfb_{out}$、$o.cfb_{in}$、o.ur 的 MBR 以及 o.ur 和 o.pdf 的磁盘地址。

U-tree 的特点：假设 e 为 U-tree 的一个中间项，o 为 e 中子树的任意对象，那么，对于任意 U-catalog 的 pj(1≤j≤m)，e.MBR(pj)总是包含 o.cfbout(pj)的。这个属性使得能够应用一个高效的算法来处理概率范围查询。

5.2.5 应用 U-tree 索引来进行概率范围查询

这里提供了一个观察数据来剪枝没有包含任何符合查询条件的对象。

观察数据 4：对于一个搜索区域为 r_q，概率阈值为 p_q 的概率范围查询 q，如果 rq 没有与 e.MBR(pj)相交，那么 e 的子树就可以被剪枝了。这里 pj 为 U-catalog 中满足 pj≤pq 的最大值。

应用 U-tree 索引结构进行概率范围查询的算法是这样的：

从根节点开始搜索，根据观察数据 4 来去除一些不合格的项。对每个剩下来的项，检索它的孩子节点，进行上述的操作直到到达一个叶子节点。对每个对象 o，首先使用观察数据 3 来进行剪枝或者验证。如果 o 既没有被剪枝，也没有被验证，那么就把它以及保存 o.ur 和 o.pdf 的物理地址添加到一个候选集 Scan。当 U-tree 中所有该检测的节点都检测过之后，开始处理 Scan。这一阶段，首先把 Scan 中的元素根据它们的物理地址编程一组组，然后对每个地址进行一次 I/O 操作，把所有相关候选对象的具体信息加载起来，接下来就可以计算它们的出现概率了。

第六章 总结

不确定性数据在越来越多的领域得到广泛应用，基于不确定性数据的研究正如火如荼的进行。不同的数据模型、不同的语义会派生出各种各样查询。各种应用要根据自身的特点选择数据模型以及查询 算法。目前存在多种多样的数据模型，其中应用最广泛的模型是可能世界模型，所有的模型都可以从这个模型派生。但是这个模型的可能世界实例远远大于不确定性数据的规模，因此需要使用排序和剪枝等启发式技术来减少计算开销从而提高效率。目前已经有很多针对不确定性数据的查询算法。

由于不确定性数据的应用领域越来越广泛，为了适应各种应用的场景，未来必定会出现更多的数据模型以及查询算法。

参考文献

[1] SARMA A D, BENJELLOUN O, HALEVY A, et al. Working models for uncertain data [EB /OL]. (2006203202). http: / / twikiedlab. cs.umass. edu /pub /.

[2] MOTRO A. Management of uncertainty in database systems [EB /OL]. (2005203215). http: / / ise. gmu. edu /～ami/ research /publications /pdf /modern94.

[3] 申德荣, 于 戈, 寇 月, 聂铁铮. 可能世界内数值型不确定数据匹配模型. 计算机应用研究. 2008-7：2607-2611

[4] 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. 计算机科学报，2009，(01)：1-16

[5] HUA M,PEI J,ZHANG W,et al. Ranking queries on uncertaindata:A probabilistic threshold approach[C] .Proceedings of the2008 ACMSIGMOD International Conference on Management of Da-ta. New York: ACMPress, 2008, :673-686 .

[6] 李建中，于 戈，周傲英. 不确定性数据管理的要求与挑战. 中国计算机学会通讯. 2009，5(4):6-15

[7] Abiteboul S，Kanellakis P，Grahne G. On the representation and querying of sets of possible worlds. ACM SIGMOD Re2cord，1987，16 (3) : 34248

[8] Green T J，Tannen V. Models for incomplete and probabilistic information. IEEE Date Engineering Bulletin，2006，29(1) : 17224

[9] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data.In Proceedings of the 2003 ACM SIGMOD international conference on Management of data(SIGMOD'03), pages 551{562, New York, NY, USA,2003. ACM Press.

[10] NORBERT FUHR and THOMAS RO¨LLEKE University of Dortmund. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems

[11] Lakshmanan L V S , Leone N , Ross R , Subrahmanian V S.ProbView : A flexible database system. ACM Transactions on Database Systems , 1997 , 22 (3) : 4192469

[12] Green T J , Tannen V. Models for incomplete and probabilistic information. IEEE Date Engineering Bulletin , 2006 , 29 (1) : 17224

[13] Barbara D , Garcia Molina H , Porter D. The management of probabilistic data. IEEE Transactions on Knowledge and Data Engineering , 1992 , 4 (5) : 4872502

[14] Benjelloun O , Sarma A D , Halevy A , Widom J . ULDBs :Databases with uncertainty and lineage/ / Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul , 2006 : 9532964

[15] Soliman M A，Ilyas I F，Chang K C. Top-k query processing in uncertain databases/ / Proceedings of the 23rd IEEE International Conference on Data Engineering. Istanbul，2007 :8962905

[16] Jin Che Qing，Yi Ke，Chen Lei，Yu Xu，Lin Xue Min. Sliding window Top-k queries on uncertain Streams. Proceedings of the VLDB Endowment，2008，1 (1) : 3012312

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

[18] Borzsonyi S, Kossmann D, Stocker K. The skyline operator// Proceedings of the 17th IEEE International Conferenceon Data Engineering. Heidelberg, 2001: 4212430

[19] Pei J, Jiang B, Lin X, Yuan Y. Probabilistic skylines on uncertain data// Proceedings of the 33rd International Conference on Very Large Data Bases. Vienna, 2007: 15226

[20] Woodruff A, Stonebraker M. Supporting fine grained datalineage in a database visualization environment// Proceedingsof the 13th IEEE International Conference on Data Engineering. Birmingham, 1997 : 912102

[21] Widom J. Trio : A system for integrated management of data，accuracy，and lineage// Proceedings of the 2nd BiennialConference on Innovative Data Systems Research. Asilomar，2005 : 2622276

[22] Benjelloun O，Sarma A D，Halevy A，Theobald M，WidomJ. Databases with uncertainty and lineage. The VLDB Journal，2008，17 (2) : 2432264

[23] Benjelloun O, Sarma A D, Halevy A, Widom J. ULDBs :Databases with uncertainty and lineage/ / Proceedings of the 32nd International Conference on Very Large Data Bases. Se2oul, 2006 : 9532964

[24] Sarma A D, Theobald M，Widom J. Exploiting lineage forconfidence computation in uncertain and probabilistic databases// Proceedings of the 24th IEEE International Confer2ence on Data Engineering. Cancun, 2008 : 102321032

[25] Sarma A D, Benjelloun O, Halevy A, Widom J. Workingmodels for uncertain data// Proceedings of the 22nd IEEE International Conference on Data Engineering. Atlanta, 2006: 7

[26] Imielinski O, Jr WL. Incomplete information in relational databases. Journal of ACM, 1984, 31 (4): 7612791

[27] Benjelloun O, Sarma A D, Halevy A, Widom J. ULDBs:Databases with uncertainty and lineage// Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul, 2006: 9532964

[28] 姜小华，罗 军. 非确定性数据库中空值处理. 计算机应用. 2008,28(12):235-237

[29] Andritsos P, Fuxman A, Miller R J. Clean answers overdirty databases: A probabilistic approach// Proceedings of the 22nd IEEE International Conference on Data Engineering. Atlanta, 2006: 30

[30] Sen P, Deshpande A. Representing and querying correlatedtuples in probabilistic databases// Proceedings of the 23rd IEEE International Conference on Data Engineering. Istanbul, 2007: 5962605

[31] Gradel E, Gurevich Y, Hirsch C. The complexity of queryreliability// Proceedings of the 17th ACM SIGACT2SIG2MOD2SIGART Symposium on Principles of Database Systems. Seat tle, 1998:

2272234

[32] Dalvi N, Suciu D. Efficient query evaluation on probabilistic databases// Proceedings of the 30th International Conferenceon Very Large Data Bases. Toronto, 2004: 8642875

[33] Chau M ,Cheng R ,Kao B ,et al . Uncertain Data Mining :An Example in Clustering Location Data [C] // The 10th Pacific Asia Conference on Knowledge Discovery and Data Mining. Singapore ,2006

[34] Kriegel H-P ,Pfeifle M. Density-based clustering of uncertain data [C] // The 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. Chicago ,2005

[35] Ester M,Kriegel H-P,Sander J,et al . A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise [C] // The 2nd International Conference on Knowledge Discovery and Data Mining. Portland ,1996

[36] 许华杰，李国徽，杨　兵，杜建强. 基于密度的不确定性数据概率聚类. 计算机科学. 2009, 36(5)

[37] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S.Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In VLDB, pages 876~887, 2004.

[38] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In Proceedings of the 31st international conference on Very large data bases (VLDB'05), pages 922~933.VLDB Endowment, 2005.

致谢

在本次论文设计过程中，指导老师杜卿对论文从选题，构思到最后定稿的各个环节给予细心指导，使我得以最终按时完成论文设计。在学习中，杜老师严谨的治学态度、丰富渊博的知识、敏锐的学术思维、精益求精的工作态度以及诲人不倦的师者风范是我终生学习的楷模。杜老师高深精湛的造诣与严谨求学的治学精神将永远激励着我。在此，谨向杜老师致以我衷心的感谢和崇高的敬意！

感谢我的父母，他们辛勤劳作，养育了我二十多年。

感谢跟我一起研究不确定性数据的两位同学，他们把自己好不容易搜集到的珍贵资料贡献了出来，并且毫无保留的与我分享他们的读书心得。

感谢我的项目经理，尽管项目很紧急而人手又不足，但他还是让我请了三个月长假，并且一直鼓励我把论文放在首位。感谢项目组的另外几位同事，因为我请假，他们的工作量增加了，但他们毫无怨言，并且按时完成了任务。

感谢与我住在同一个宿舍的另外三位同学，他们良好的生活习惯为我营造了一个舒适的生活环境，他们的勤奋感染了我，使我努力学习。

感谢所有曾经帮助过我的人。

最后，我要感谢在百忙中抽时间对本文进行审阅，评议的各位老师！