Rank Queries in Probabilistic Data Integration Systems for Digital Library Federation

Peng Pan, Qizhong Li,YuQing Sun
School of Computing Science and Technology, ShanDong University
ppan,lqz,sun_yuqing @sdu.edu.cn

Abstract

In digital library federation, a challenge is how to exchange and share information on heterogeneous digital libraries. The mediator-based approach of data integration provide a uniform view by creating the semantic relationship between sources and mediated schema. Since automatic matching produce probabilities,it's necessary to take the probability of mappings into consideration. Further more, for tuples' probabilities arise as an additional ranking dimension, both tuple probabilities and scores be factored in the interpretation of top-k queries in probabilistic databases.In this paper, after proposing a architecture of data integration for DLF, we demonstrate the probabilities for the schema mappings and query answerings. Then, having discussed the opt_U-kRank algorithm, we describe the revised_U-kRank algorithm which may improve the efficiency of execution. Finally, based on the experiment results of two U-kRank algorithms, we discuss the effect of probabilities distribution on executions..

1. Introduction

Digital Library(DL) is a virtual knowledge center which provides intelligent search services for those super large scale, distributed heterogeneous information to build a sharable and extensible knowledge system in networked environment ([1] [2]).

It is impossible for any library to be self-contained,because no single library can ever contain all the materials and information its users can ever need. It's frequent that users have to access more than one digital library when requesting for the desired resources. As each DL is autonomic, heterogeneous , the query has to be submitted over different DL's schemas. To simplify the process, the concept of Digital Library Federation(DLF) has been proposed [3].

978-1-4244-2511-2/08/$25.00 ©2008 IEEE

In DLF, in order to achieve inter-operation , we will have to face the challenge on how to exchange and share information among heterogeneous DLs. The technique of data integration[4] [5] may be the best solution, which aggregates distributed heterogeneous data sources and provides a uniform view to users. By creating the semantic relationships between the schemas of sources and mediator [6], the mediator-base approach maps all the sources to an uniform domain of concept,.

Since each DL of a DLF may frequently change it's states(join in, active, deactive, or quit) and data (including changing the types and the quantities), it's tedious, time-consuming and error-prone to manually specify schema matches and mappings. Therefore, faster and less labor-intensive integration approaches and tools supporting automatic matchs are developed[7]. However, since the users in reality are not skilled enough to provide precise mappings and at the same time the scale of the data prevents generating and maintaining precise mappings, such as in integrating data of the web scale , the mappings are frequently inaccurate [8] along with their probabilities.

In a data integration system,while a query is submitted over the schema of mediator, the query would give birth to no less than one query reformulations with certain probability. As each tuple of the results has different probability, we may adapt top-k algorithm to get the most approximate k answers. In a DLF, end-users are more interested in the most important (top-k) query answers in the potentially huge answer spaces.

In probabilistic databases, tuples' probabilities arise as an additional ranking dimension that interacts with tuples' scores. Both tuple probabilities and scores need to be factored in the interpretation of top-k queries in probabilistic databases.

The paper is organized as follows. Section 2 discusses related work . Section 3 proposes a data integration architecture for DLF. Focused on the probability for the schema mappings, Section 4 describes the query answering. Based on [9], section 5 expands on the discussion on the rank algorithm over uncertainty data. Section 6 analysis the experiment. Section 7 conclude

2. Related work

The research on probabilistic data at present focus on the probabilistic database and on the management of probabilistic data in general. The topic related probabilistic mappings just starts off in recent years [11] [12]. [13] used the top-k schema mappings obtained by a semi-automatic mapper to improve the precision of the top mapping. [14] combines IR and machine learning fields to find suitable mapping candidates. However, it seems that they didn't talk about the relationships between mappings and uncertainty as a whole. [8] presents two possible semantics for probabilistic mappings: BY-TABLE and BY-TUPLE, proposes the query complexity algorithms for answering queries in the presence of approximate schema mappings, and describes an algorithm for efficiently computing the top-k answers to queries in such a settings.

There are many research on Top-k over probabilistic data. [15] describes a novel approach, which computes and ranks efficiently the top-k answers to a SQL query on a probabilistic database. Without considering any other scoring factor, it takes probabilities as only rank factor. [17] proposes novel solutions to speed up the probabilistic ranked query (PRank) over the uncertain database. It introduces two effective pruning methods, spatial and probabilistic, to help reduce the PRank search space.[9] addresses the scores-uncertainty interaction of top-k queries in probabilistic databases. The interaction between the concepts of “most probable” and “top-k” is materialized using two new top-k query semantics (1)U-Topk query reports a k-length tuple vector with the maximum probability of being top-k across all database possible worlds (2) U-kRanks query reports a set of k tuples where each tuple is the most probable tuple to appear at some rank 1 … k across all database possible worlds. The Revised_U-kRank algorithm proposed in this paper is based on the opt_U-kRank algorithm.

3. A data integration architecture for Digital Library Federation

Figure 1 is an information integration model for a DLF. The global schema provides the concept standards agreed in the federation. Each DL exports it's own sources by mapping the local schema and global schema, and the users may submit query over the global schema by DLF portal. The metadata repository provides the metadata standards in common, and the query engine is in charge of receiving user query request , optimizing and executing query plan. A wrapper is a software module that is specific to a data source, that translates data from the source to a form that is used by the query processor of the system, translate the query to the source's interface on it's schema and retrieve the sources. A mediator is a software module that takes input as a set of data produced by either a wrapper or another mediator, and produces output as another set of data, namely the one corresponding to the results of a given query.

There are three approaches to create the mappings in traditional integration system: local-as-view (LAV), global-as-view (GAV) [10] and GLAV[16]. LAV is easy to express the source with the complex query process, while GAV is likely to cope with the query , but inconvenient to express and maintain the source schema. GLAV tends toward both of their characters.

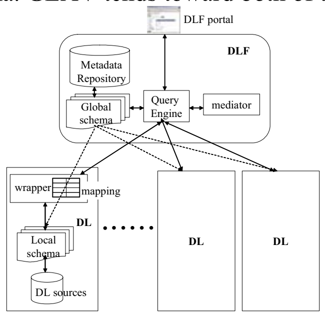

Figure 1. The data integration architecture for Digital Library Federation

In a DLF, considering the autonomy of each DL, the LAV approach is the appropriate candidate to create the mappings for this paper, with the following formalization in general:

$\forall x(s(x) \rightarrow q_g(x))$ (while $q_g \sqsupseteq s$)

$\forall x(s(x) \equiv q_g(x))$ (while $q_g \sqsupseteq s$)

where s is an element of S which is the local schema, expressed by the logic theory over a related alphabet A_S, and q_g is a query over G of the arity of g.

4. Probability for the schema mappings

4.1. The probability of mappings

If the schema mappings are created automatically, it's highly probable that no less than one candidate with a probability would appear.[8] divides the distribution of tuples into two types:

1) BY-TABLE: all the tuples follow the same schema mapping

2) BY-TUPLE: in the source S, there are more than one subset of tuples grouped by different mappings.

In this paper our discuss has these limitations:

1) We focus on the BY-TABLE mapping, and the BY-TUPLE mapping is out of this paper's discussion.

2) Our discussion is only about the relational data model, of which a schema is considered as a set of relations, and a relation as a set of attributes.

3)Only select-project-join (SPJ) queries in SQL are considered.

4)The LAV mappings is in the form of which a single relation of T is expressed by only one projection query of S.

Table 1. The schema DLF_DIRECTORY

DLF_DIRECTORY							
Title	Publisher	Date	Format	Creator	subject	Type	Source

Table 2. An instance of DL_DIRECTORY

DL_DIRECTORY						
Btitle	**issuer**	**Bdate**	**carrier**	**author**	**Bsubject**	**category**
Chinese History	Thread Binding Books Publishing House	2006	575p;26cm	HaiShan Xu	Chinese History	monograph
The open empire A history of China to 1600	JiangSu People's Publishing House	2007	391p;23cm	Hansen Valerie	Chinese History	series
A panorama of Chinese history and geography	SiChuan University Press	2008	234p;26cm	HuiQiang Wang	historical geography	monograph
Chinese History	People's China	2006	14 Vol;23cm	ChangYun Shen	general history	series
The 36 unsettled cases of Chinese History	HuBei People's Publishing House	2007	264p;24cm	SiXin Xiang	Chinese History	series
The enviroment and society of Chinese History	Sanlian Bookstore	2007	576p;23cm	LiHua Wang	environment	monograph

Table 3. A probabilistic schema mapping between DL_DIRECTORY and DLF_DIRECTORY

Possible Mappings	Probability
{(Btitle,Title),(issuer,Publisher),(author,Creator),(Bsubject,Subject),(category,Type)}	0. 6
{(Bsubject,Title),(issuer,Publisher),(author,Creator), (Bsubject,Title),(category,Type)}	0. 3
{(Btitle,Title),(issuer,Creator),(author,Publisher),(Bsubject,Type), (category,Subject)}	0. 1

4.2. Query answering

When a query is proposed over the schema of G, several reformulations would be formed in term of variant mappings on the schema of S, and the probability of each returned tuple is unequal.

Based on example 1, the following query is running on the digital library federation site.

select Title from DLF_DIRECTORY

order by Relevance("Chinese History")

In term of mappings, the original query on the digital library DL_1 can be reformulated into the forms as following:

Select BTitle from DL_DIRECTORY (probability is 0.6)

Select BSubject from DL_DIRECTORY(probability is 0.3)

Select BTitle from DL_DIRECTORY (probability is 0.1)

Executing these query, we get the result seen in Table 4. As a result of occurrence both in 1) and 3), the item 'The open empire A history of China to 1600' gets the sum of the two reformulation's probabilities, and the item 'Chinese History' gets the sum of 1.0 with the occurrence in the answer of each reformulation.

Table 4. The answers of Q over DLF_DIRECTORY

t1	Chinese History	1
t2	The open empire A history of China to 1600	0. 7
t3	The 36 unsettled cases of Chinese History	0. 7
t4	The enviroment and society of Chinese History	0. 7
t5	A panorama of Chinese history and geography	0. 7
t6	general history	0. 3
t7	entironment	0. 3
t8	historical geography	0. 3

5. Top-k query answering

In order to find the most probable top-*k* answers in uncertain databases, the interaction between the concepts of "most probable" and "top-*k*" be considered in top-k query. Here we adopt the U-kRanks query defined in [9] to solve this: A top-*k* query that reports a set of k tuples, where each tuple is the most probable tuple to appear at some rank 1… k across all database possible worlds.

5.1 The U-kRanks query's definition

Following is the definition of U-kRanks query in[9]:

Let D be an uncertain database with possible instances I={I_1,I_2,….I_n}, and $\sum_{i=1}^{n}\Pr(Ii)=1$ .For i=1….k,let{ $t_i^1,\ldots t_i^m$ } be a set of tuples, where each tuple t_i^j appears at rank i in a non empty set of database instances I(t_i^j) based on scoring function F. A U-kRanks query based on F, return { t*; i=1….k}, where t*= $\operatorname{argmax}_{t_i^j} \cdot \sum_{\xi\in I(t_i^j)}\Pr(\xi)$.

It's clear that each tuple is a clear winner at its rank over all worlds.

ach tuple is a clear winner at its rank over all worlds.

5.2 The probability of states

After getting m tuples from one database D by the scoring function F, several states would be formed. We denote some states by $s_{m,l}$, including two substates: s_l that represents l(l<=m) tuples occurring in the $s_{m,l}$

and $s_{\neg l}$ that represents rest tuples(m-l) not in $s_{m,l}$. As a result, the probability of $s_{m,l}$ is expressed by $\rho(s_{m,l})=\Pr(s_l \cap s_{\neg l})$. For example, {t1,t2,t3,t4} are a set of tuples which are retrieved by a scoring function F, and to one state s(4,3)=<t1,t2,t4> , it's probability can be computed, where $\rho(s_{4,3}) = \Pr(\{t1,t2,t4\} \cap \neg \{t3\})=\Pr(t1 \wedge t2 \wedge \neg t3 \wedge t4)= \Pr(t1)*\Pr(t2)*\Pr(\neg t3)*\Pr(t4)$.

5.3 Discussion on the opt_U-kRanks algorithm

When a new tuple is retrieved, it is used to extend all states causing all possible ranks of this tuple to be recognized.

Let t be a tuple seen after retrieving m tuples from the score-ranked stream(also named as vectors). Let $P_{t,i}$ be the probability that tuple t appears at rank i, based on scoring function F. It follows from previous state definition that $P_{t,i}$ is the summation of the probabilities of all states with length i whose tuple vectors end with t, provided that t is the last seen tuple from the database. In other words, we can compute $P_{t,i}$, for i = 1 . . .m, as soon as we retrieve t from the database.

For each rank i, we need to remember the most probable answer obtained so far. Since it's not feasible to determine what is the final answer till the end of retrieval. we should decide when can we conclude an answer for each rank i as early as possible . In [9], Soliman etc. propose a algorithm applied in opt_U-kRanks query and analyze the condition for termination of rank i . Based on [9], we expand the discuss of when to report the answer of rank i.

In this paper,we illustrate the opt_U-*k*Ranks query using trees . Suppose that the score-ranked tuple vectors is (t1,t2....tn). As depicted in Figure 2 , each node represents a state for probability, and the value of each node represents the length of the state,i.e. the rank position. The symbol of each edge between node *i* and *j* denotes that by adding a new tuple state(occurence and non-occurence), the state *i* may change to another state *j*. The path from root node to one node i like *t1t2....ti* just is the expression of state *i* .

To be able to report an answer for rank i, we should decide when can we report an answer for each rank i. As Figure 2 depicted, the nodes enclosed by dashed rectangles denote the states of rank 3, and those by dashed oval denote the states whose ranks are less than 3 when the new tuple is t4. In Figure 2, we found that the states whose edge marked with 't5' are all from the states whose value are two and whose other edges are marked with t4 or ¬t4. This means that an new tuple t will appear at rank i only if we extend states with length i − 1 by appending t to their tuple vectors.As a result, we get the set of states with length three whose tuple vectors end with t4 . They are {t1,t2,¬t3,t4},{t1,¬t2,t3,t4} and {¬t1,t2,t3,t4}, and we may compute the probability that t4 appears at rank three, based on a scoring funciton F as following:

$P_{4,3} = \sum \rho(s_{4,3}) = \Pr(t1 \wedge t2 \wedge \neg t3 \wedge t4) + \Pr(t1 \wedge \neg t2 \wedge t3 \wedge t4)+\Pr(\neg t1 \wedge t2 \wedge t3 \wedge t4) = \Pr(t1)*\Pr(t2)*\Pr(\neg t3) * \Pr(t4) + \Pr(t1)*\Pr(\neg t2) *\Pr(t3)*\Pr(t4) + \Pr(\neg t1)*\Pr(t2)*\Pr(t3)*\Pr(t4)$

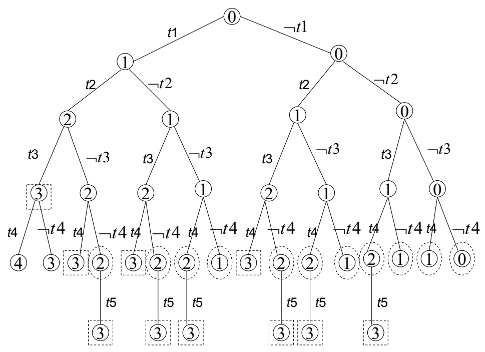

Figure 2. The tree for probability of states

Based on Figure 2, we compute a inequation applied to the termination conditon for reporting an anwer for rank i.

Let S_m^j be the set of states with lenght j whose tuple vectors end with t_m, namely $\sum \rho(s_{m,j})$ .Let $P_{m,j}$ be the probability at rank j for tuple t_m.

Since the probability of an event is no more than 1, it is clear that $P_{5,3} <= S_4^2 = \sum \rho(s_{4,2})$ in Figure 2. Without losing generality, we may conclude that for any tuple t_m, the inequation $P_{m,k} <= S_{m-1}^{k-1} = \sum \rho(s_{m-1,k-1})$ (k is a rank position) is always true. If we want tuple t_m is the right answer for rank k, we should guarantee that $P_{m,k}$ is the biggest among all. Based on above discussions, we may deduce that although $P_{m,k} > S_m^{k-1}$ stands for $P_{m,k} > P_{m+1,k}$, which is the conditon given in [9], it's not enough to guarantee $P_{m,k}$ bigger than $P_{m+2,k}$ or....or $P_{n,k}$ (n>i). Here we give a sufficient condition $P_{m,k} > \sum_{j<k} S_m^j$,which is more sound than described above.

Next, we simplify the inequation.

$P_{m,k} = \sum \rho(s_{m-1,k-1}) * \Pr(t_m)$

As Figure 2 depicted, we have:

$S_m^{k-1} = \sum \rho(s_{m-1,k-1}) * \Pr(\neg t_m) + \sum \rho(s_{m-1,k-2})$ $* \Pr(t_m)(m>1,k>2)$

$S_m^{k-2} \ldots S_m^1$, S_m^0 and son on.

$S_m^0 = \sum \rho(s_{m-1,0}) * \Pr(\neg t_m)$

$P_{m,k} > \sum_{j<k} S_m^j$ can be formulated as $P_{m,k} >$ $\sum_{j<k} S_m^j = S_m^{k-1} + \ldots S_m^1 + S_m^0$

when k>=2 we have the following:

$P_{m,k} > \sum_{j<k} S_m^j$

$\Leftrightarrow \sum \rho(s_{m-1,k-1}) * \Pr(t_m) > \sum \rho(s_{m-1,k-1}) * \Pr(\neg t_m) + \sum \rho(s_{m-1,k-2}) * \Pr(t_m) + \sum \rho(s_{m-1,k-2}) * \Pr(\neg t_m) + \ldots \sum \rho(s_{m-1,0}) * \Pr(t_m) + \sum \rho(s_{m-1,0}) * \Pr(\neg t_m) =$

$\sum \rho(s_{m-1,k-1}) * (1 - \Pr(t_m)) + \sum \rho(s_{m-1,k-2}) + \ldots \sum \rho(s_{m-1,0})$ (k>=2)

$\Leftrightarrow \Pr(t_m) > \frac{1}{2}(1 + \sum_{\xi=0}^{k-2} \sum \rho(s_{m-1,\xi}) / \sum \rho(s_{m-1,k-1}))$ (k>=2)

$\Leftrightarrow \Pr(t_m) > \frac{1}{2}(1 + \sum_{\xi=0}^{k-2} S_{m-1}^{\xi} / S_{m-1}^{k-1})$(k>=2)

$\sum \rho(s_{m-1,0}) * \Pr(t_m) > \sum \rho(s_{m-1,0}) * (1 - \Pr(t_m))$(k=1)

$\Leftrightarrow \Pr(t_m) > \frac{1}{2} \sum \rho(s_{m-1,0}) = \frac{1}{2} S_{m-1}^0$ (k=1)

Summing up above, we may get a approximate condition that $\Pr(t_m)$ is no less than 1/2 to evaluate whether t_m is the appropriate answer for rank i.

5.4 The Revised_U-kRank algorithm

Till now, we rewrite the algorithm for U-*k*Ranks query of [9] to make a quick determination as following:

```
Algorithm: Revised_U-kRank(source,k)
Require:
source: Score-ranked tuple stream
k: Answer length
Ensure: U-kRanks query answer
1: answer[ ]        //empty vector of length k
2: ubounds[ ]        //vector of length k initialized with 1's
3: reported←0 {No. of reported answers}
4: depth ← 1
5: space← {current set of states}
6: while ( source is not exhausted AND reported < k) do
7:{        t ← next tuple from source
8:         if t<1/2 then { depth←depth + 1;continue}
9:         for i=1 to min(k, depth) do
10:       {          = * Pr(tm)
11:                  if (answer[i] was previously reported) then continue
12:                  if ( > answer[i].prob) then
13:                  {          answer[i] ← t
14:                             answer[i].prob ←
15:                  }
16:                  else
17:                  {          Update ubounds[i] with = +...+
18:                             if (answer[i].prob > ubounds[i]) then
19:                             {          Report answer[i]
20:                                        reported←reported + 1}
21:                  }
22:        }
23:       depth←depth + 1
24:}
```

6. Experiment

We now present experimental results illustrating the discussion in Section 6. Experiments were performed on a 2GHz Pentium IV Windows desktop with 1 GB of RAM and 60 GB of hard disk space. The algorithm in Section ? were implemented in C, and they access database through cursor operations.

We take 50 tuples as tested tuple vectors, and run two algorithms respectively by using different parameter value from 1 to 10.Figure 3-5 are the results based on the experiments.

Since the two algorithms cost times mostly on computing the combination of probabilities, adding a new elements would increase the times consumed rapidly. Although we may get the same results by running the two algorithms, the Revised_U-kRank algorithm could avoid some unnecessary processes of computing the combinations on account of it's function to determine as soon as possible whether a tuples is qualified. Compared with the opt_U-kRank algorithm in [9], the Revised_U-kRank could improve efficiency in equal distributions of probabilitis. The result of Figure 3 verifies the discussion.

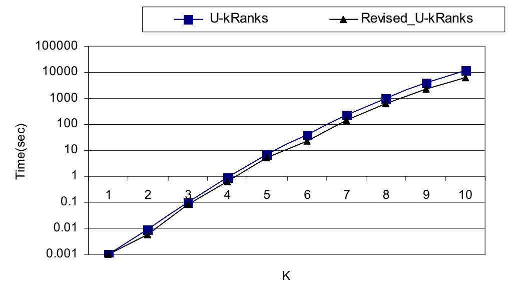

Figure 3. The time costs for two algorithms

The distributions of probabilites have an easily noticed effect on the execution time. If more tuples with higher probabilities queue in the front, the algorithm may find the element at rank i in shorter time and save the count of computing combination. On the other hand, if more tuples in the front of queue have with lower probabilities, the algorithm would have to further search for eligible candidates.This would increase the complexity of computing. In Figure 4, p represents the avearge probability value of first 20 elements. More bigger the value of p is, more elements with higher probabilities stand the front of the queue. This would increase the effiency of computing.

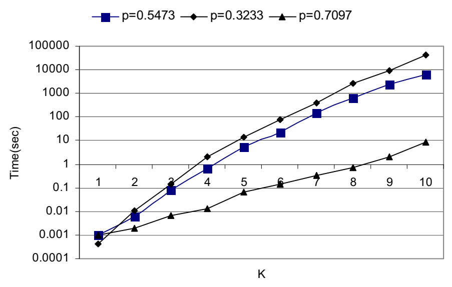

Figure 4. The effects of probabilities distributions on time

The condition $P_{m,k} > \sum_{j<k} S_m^j$ and Pr(t$_m$)>1/2 are so sufficient that a few tuples being the right candidate at rank i may be omitted. For example, suppose each element in tuple queues is less than 1/2, the two algorithm can't get the right answer undoubtedly. Therefore, it's more often that the results only contain partial elements from rank 1to rank j(j<k), and the rest elements(k-j) desired would be lost. Figure 5 depicts the effect of probabilistic distribution on final number of elements.

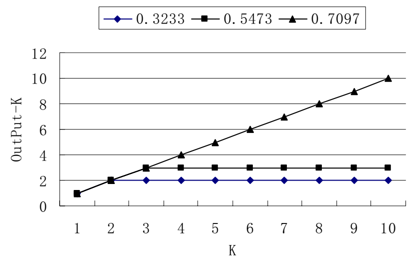

Figure 5. The effects of probabilities distributions on outputs

7. Conclusion and future works

This paper discusses the problems related to the probabilistic data integration for Digital Library Federation including architecture, probabilistic mappings, probabilistic queries, and Top-*k* queries. After proposing a architecture of data integration for DLF, we demonstrate the probabilities for the schema mappings and query answerings. Then, having discussed the opt_U-kRank algorithm, we describe the revised_U-kRank algorithm which may improve the efficiency of execution. Finally, based on the experiment results of two U-kRank algorithms, we discuss the effect of probabilities distribution on executions.

Many other problems need to be solved and furthered .Having only discussed the probabilities of BY-TALBE mappings, we would probe in the topic of BY-TUPLE mappings in the future, because it's more complex than BY-TABLE. We also need to further the U-kRank algorithms to get more efficiency.

8. References

[1] E.A.Fox. Source Book on Digital Libraries.Technical Report TR-93-35 Virginia Polytechnic Institute and State University,1993.

[2] S.P.Harter. What is a Digital Library? Definitions, Content, and Issues. In KOLISS ,1996.

[3] Birmingham B.,et al.EU-NSF digital library working group on interoperability between digital libraries. http://www.iei.pi.cnr.it/DELOS/NSF/interop.htm,2001-10-15

[4] Lenzerini, Maurizio, Data integration: A theoretical perspective ， In PODS, 2002. pp. 233-246.

[5] A. Halevy, A. Rajaraman, and J. Ordille, Data integration: The teenage years, In VLDB, 2006, pp. 9–16

[6] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández ， Schema Mapping as Query Discovery, In VLDB, 2000, pp. 77 – 88.

[7] E. Rahn, and P. A. Bernstein, A survey of approaches to automatic schema matching,Very Large Database J., 2001,10(4):334–350.

[8] Xin Dong, Alon Y. Halevy, and Cong Yu. Data integration with uncertainty. In VLDB ， 2007, pp.687-698

[9] M. Soliman, I. F. Ilyas, and K. Chen-Chaun Chang. Top-k query processing in uncertain databases. In ICDE, 2007 ,pp. 896-905

[10] Alon Y. Levy , Logic-based techniques in data integration, in MinkerJ (ed.) Logic-based artificial intelligence. 2000, pp. 575–595

[11] Dan Suciu, Nilesh Dalvi, Foundations of probabilistic answers to queries, In SIGMOD. 2005. pp. 963.

[12] Nilesh Dalvi, Dan Suciu, Management of probabilistic data: Foundations and challenges, In PODS 2007, pp. 1-12.

[13] Avigdor Gal, Managing uncertainty in schema matching with top-K schema mappings, 4090 LNCS,

Journal on Data Semantics VI - Special Issue on Emergent Semantics, 2006, pp. 90-114.

[14] Henrik Nottelmann, Umberto Straccia, Information retrieval and machine learning for probabilistic schema matching, Information Processing and Management, v 43, n 3, May, 2007, pp. 552-576.

[15] Re, C. Dalvi, N. Suciu, D. ,Efficient Top-k Query Evaluation on Probabilistic Data,In ICDE 2007 pp. 886-89

[16] M. Friedman, A. Levy, and T. Millstein, Navigational plans for data integration, In AAAI ,1999., pp. 67–73.

[17] Lian, X. and Chen, L. 2008. Probabilistic ranked queries in uncertain databases. In EDBT 2008, pp. 511-522.