Top-k Query Processing Techniques in Uncertain Databases: A Review

Cherry Khosla
Lovely Professional University,
Punjab, India

Parveen Kakkar
DAV Institute of Engineering and
Technology, Jalandhar, India

ABSTRACT
Many applications involving large databases with uncertain data require various techniques to rank queries. Ranking queries (often called as top-*k*) are useful in answering most important query answers in various domains such as web search, managing sensor data, location tracking, data mining tasks and multimedia. In this survey paper, we describe and classify different top-*k* processing techniques in probabilistic databases and their implications.

Keywords
Uncertain database, ranking queries, sliding window, possible world, top-*k* query.

1. INTRODUCTION
Traditional databases store relatively static data or solid facts on which queries are executed and answers are reflected on the current state of the database. Over the past few years, we have observed some applications that do not fit into this model and querying prototype. Such applications consist of large databases or uncertain databases with probabilistic data. In these databases, information occurs in the form of streams, in other words they are called as Data Stream Management System (DSMS). A data stream is a real- time, continuous, ordered sequence of items. They are ordered either by explicit time stamp or implicit arrival time [1]. Table-1summarises the difference between traditional database management system and data stream management system. In probabilistic databases, there can two different types of top-k problems. Some of them consist of scoring function to rank function; others typically consist of set of tuples each of which is associated with a numeric value that represents a probability. Both the dimensions of top-k queries are not treated equally, score is measured in an ordinal scale and probability is considered in cardinal sense. The databases along with probability of each tuple together make a *possible world*. A ranked query fetches *k* data objects in the database having highest probability value. The probability value acts as estimation for the object based on some characteristics. Uncertainty in databases can be because of incomplete data, loss in data transfer etc [2] [3]. Most common types of data uncertainties in databases are tuple-level and value-level uncertainties.

1.1 Tuple-level Uncertainty
In tuple-level uncertainty, the values or attributes are known but tuples are uncertain. We are not sure whether the tuple is correct or not. Corresponding to each uncertain tuple, there are two possible worlds, one of them includes tuple and other does not.

1.2 Value-level Uncertainty
Here, the tuples certainly exists, while the attribute value is however uncertain. We are not sure about the values that an attribute of a tuple can take. It can have multiple values thereby showing uncertainty.

tuples	score	$p(t)$
t_1	v_1	$p(t_1)$
t_2	v_2	$p(t_2)$
⋮	⋮	
t_N	v_N	$p(t_N)$

rules	
τ_1	$\{t_1\}$
τ_2	$\{t_2, t_4\}$
⋮	⋮
τ_M	$\{t_5, t_8, t_N\}$

Fig 1. Tuple-level uncertainty

tuples	score
t1	{(90, 0.5), (75, 0.5)}
t2	{(82, 0.6), (60, 0.4)}
t3	{(85,1)}

world W	Pr [W]
{t1=90, t2=82, t3=85}	0.5 X 0.6 X 1
{t1=90, t3=85, t2=60}	0.5 X 1 X 0.4
{t2=82,t3=85, t1=75}	0.6 X 1 X 0.5
{t3=85,t2=60, t1=75}	1 X 0.4 X0.5

Fig 2. Example of possible world in tuple-level uncertainty

tuples	score
t_1	$\{(v_{1,1}, p_{1,1}), (v_{1,2}, p_{1,2}), \ldots, (v_{1,s_1}, p_{1,s_1})\}$
t_2	$\{(v_{2,1}, p_{2,1}), \ldots, v_{2,s_2}, p_{2,s_2})\}$
⋮	⋮
t_N	$\{(v_{N,1}, p_{N,1}), \ldots, (v_{N,s_N}, p_{N,s_N})\}$

Fig 3. Value-level uncertainty

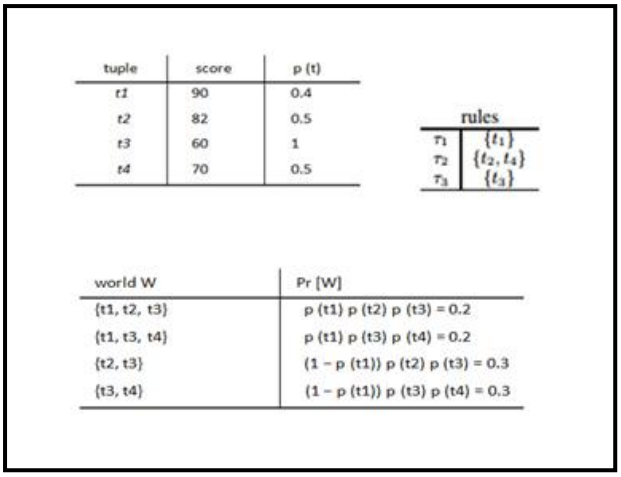

Fig 4. Example of possible world in value-level uncertainty

Table 1 Difference between DBMS and DSMS

Difference between DBMS and DSMS		
	DBMS	**DSMS**
Type of data	Persistent Data	Stream data
Access	Random	Sequential , One pass
Processing Model	Query Driven [Pull-based]	Data-Driven [Push-based]
Type of query	Fixed	Adaptive

2. RANKED QUERIES IN CONVENTIONAL DATABASES

In traditional SQL queries where the data facts are static, the query optimization can be applied. The retrieval of the data can be optimized using WHERE clause, JOINS, and other optimization techniques [4]. Due to its importance, much of previous work of the top-k queries focuses on coherent methods to retrieve the query answers. Chang et al. [5] was the first to propose Onion Technique to perform ranked search. In this technique special indexing structure for linear optimizations is defined. It returns top-N records according to the weighted sum of attribute value. Gang Luo et al. [6] provided SOA index for linear optimization of queries for a given data stream. They reduced both the index storage and maintenance overheads and keeping only subsets of tuples using count based sliding window. Top-X System was introduced by Fagin et al. [4] in his survey on top-k queries on relational databases. The proposed system was based on XML databases. In XML databases, multiple indexes are built for different content and/or structure conditions. Top-X is built to process the multiple indexes to retrieve the top-k objects. The system implements inexpensive sorted access to ranked lists. Ranked lists are used in Top-X that maintains different orders for document corpus. Lists are implemented as relational tables indexed using B+- Trees. Guo et al. [7] defines an XRank system in which ranks are produced for keyword queries over hyperlinked XML documents. Keyword searching over XML does not always return an entire document; but can be a deeply nested XML element. As a result ranking has to be done at the granularity of XML elements instead of entire XML documents. XRank system handles the challenges that come while evaluating keyword search queries over hierarchical XML documents by applying conjunctive keyword query semantic. The XRank takes input as XML/HTML document and computes the *ElemRank* . The *ElemRank* are then combined with ancestor information to generate an index structure called as Hybrid Dewey Inverted List (HDIL). The query evaluator evaluates the queries using HDIL and returns the ranked results.

3. RANKING QUERIES IN UNCERTAIN DATABASE

According to the data models proposed by Mohamed A. Soliman Ihab F. Ilyas [8], there are three different classes of ranking queries: Uncertain Top Rank (UTop-Rank), Uncertain Top Prefix (UTop-Prefix) and Uncertain Top Set (UTop-Set). UTop-Rank(i, j) can be used to find out the most probable athlete to end up in some competition. UTop-Prefix can be used in market analysis to find out the most likely product. In this survey of top-*k* queries, taxonomy is introduced to classify the top-*k* query processing techniques according to the different design dimensions. The design dimensions are: *Query Model, Data Access Methods, Implementation level, Query and data Uncertainty, Ranking Functions* [9]. Jeffrey Jestes et al. [10] proposed different ranks based on expectation, median and other statistics and derived expected ranks, median ranks and quantile ranks correspondingly. Expected ranks define a ranking framework that depends on the ranks of tuples across all possible worlds. The rank of each tuple across all possible worlds are represented by their mean(also referred as expectation) .Xi Zhang, Jan Chomicki[11] defined Global-Top*k* which returns *k* highest-ranked tuples according to their probability of being in *possible world.* A naïve method is generating al possible aggregate values and their probabilities and arranging them according to their aggregate values. But this method is impractical as the size of possible world space is exponential. Globa-top-k method minimize the search space by two-level method.(1) group state level. (2) G-x-tuple top-k query processing. In the survey of uncertain data algorithms and applications by Charu C. Aggarwal et al. [12] defined the concept of indexing uncertain data. There are many domains in which data is updated periodically in the index, therefore current values cannot be known exactly. Hence different kinds of queries which can be resolved using index structures: Range queries, nearest neighbor queries and aggregate queries. Tao Chen et al. [13] have proposed the method to optimize the top-k queries by sharing the results of the queries, which gives instant combination of tuples over a sliding window. The dependent tuples cannot be true at the same time; therefore possible worlds are generated by combining the tuples based on some correlations. These correlations are called as generation rules. Two different kinds of generation rules: Independent and Mutually exclusive. Jianjun et al.[14] developed NiagaraCQ to address the problem of scalability of internet in continuous queries. NiagaraCQ groups the continuous queries which shares similar structures thereby reducing the I/O cost significantly. Furthermore grouping can elimination the invocations of large number of queries. The rset of the paper is organized as follows. In section 4 Continuous query languages is briefly described. Semantics related to the sliding windows are described in section 5 and properties of ranking queries are described in section 6.

4. CONTINUOUS QUERY LANGUAGE

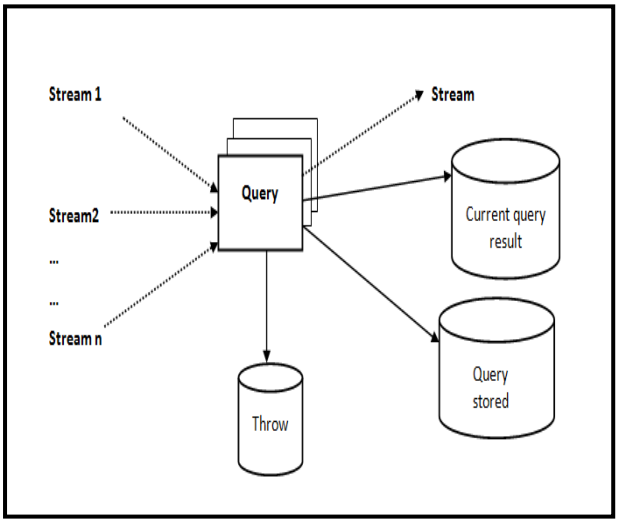

Fig 5. Architecture of CQL

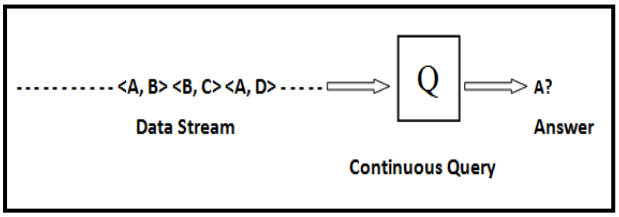

Fig 6. A continuous query Q over single data stream

A Continuous query Language (CQL) is an expressive SQL-based language for executing stream queries using streams and stored relations. DSMS queries are different from traditional database queries, for e.g. in SQL where a query returns a data from tables stored in the database. A DSMS query does not only return a set of tuples, but a potentially an infinite stream of tuples. These stream queries are termed as continuous queries (CQ). The continuous queries run indefinitely or until they are terminated. A CQL query is similar to the select statement in SQL. The semantics of continuous queries are based on two data types- stream and relations and three different classes of operators that can be defined over these types: Stream-to-relation and relation-to-stream and relation-to-relation. [15] [16]

According to the architecture defined in figure-7, a stream of queries run at an instant of time. Some of them give results which are stored in *current query result*, and others are stored in *query stored* module for future reference. The stream queries which are of no use can be thrown. [17]

4.1 Stream-to-relation operators

They are based on the concept of sliding window over a stream. The window at that any point of time contains a snapshot of the portion of the stream. Syntactically window operators are specified using a window specification language SQL-99.

4.2 Relation-to-stream operators

CQL has three different types of relation-to-stream operators: *Istream* (for "insert stream"), *Dstream* (for "delete stream"), *Rstream* (or "relation stream").

```
SELECT  top-k <what we want to see>
FROM <relevant tables, data-set, possible world>
WHERE <query constraints>
SLIDE <Window to be updated after some interval of time of tuples>
```

Fig 7. General top-k query

5. SEMANTICS OF SLIDING WINDOWS

A real time data stream is a sequence of data items that arrive some order and may be seen only once. Items may arrive in burst and only an excerpt of a stream is of interest at given point of time. For example one might be interested in gathering statistics of the packets processed by a router last day. This gives rise to the sliding window approach. The *sliding window* refers to the window of active data elements at a given instant of time [18] [19]. Different window models can be used to extract the data from a large data stream. These can be classified as:[6]

a) *Direction of movement:* Fixed starting and ending points define *fixed windows*. Moving starting and ending pints in either direction define *sliding window*. One fixed pint and other moving define *landmark windows*.

b) *Content definition:* Logical or *time-based widows* are based on time intervals. Physical or *count-based windows* can be defined in terms of number of tuples.

c) *Update interval: Jumping windows* are updated after every *k* ticks or after every *kth* arrival. If *k* is equal to window size, then they are referred to as *tumbling windows.*

The window constructs plays a vital role in processing stream queries. Jurgen et al. [20] proposed a window specification enhanced from the FROM clause of the traditional SQL queries. The BNF grammar for this specification is given in Fig 8 . To define a window over a stream

```
<from clause>  ::= FROM < stream reference> [ < window specification >] [{, <
                   stream reference > [ < window specification >]}....]
<window specification>      ::= WINDOW ([< window partition clause >] <
                               window frame clause >  [< slide clause>])
<window partition clause>   ::= PARTITION BY < column list >
<window frame clause>       ::= < window frame units >< window frame start >
<window frame units>        ::= ROWS | RANGE
<window frame start>        ::= < value specification > | UNBOUNDED
<value specification>       ::= < positive integer > [< units >]
<slide clause>              ::= SLIDE < value specification >
```

Fig 8. Sliding window BNF [20]

6. PROPERTIES OF RANKING QUERIES

Jeffrey Jestes et al.[10] proposed different fundamental properties which a ranking query should possess. These properties are: Exact-k, Containment, Unique rank, Value invariance, Faithfulness, and stability. Table 2 shows the analysis of the properties of different ranking queries.[4] [10]

a) Exact-k
A ranked query retrieves k data objects in the database having highest probability value. The top-k list should contain exactly k item

b) Containment
The top(k+1) list should contain all the elements of top-k.

c) Unique rank
The top-k list should contain each element exactly once. All the elements are assigned with a unique rank. Same rank should not be assigned to more than one element in the list.

d) Value Invariance
The scores defined over a function only determine the relative behavior of tuples. Changing the score without changing the order should not alter the top-k.

e) Stability
The stability defines the importance of an element in the top-k list. The element should not be removed from the list

It is important for any ranking query technique to satisfy all the properties to give efficient results in probabilistic databases

Table 2 Summary of top-k methods on the semantics of ranking query properties

Method	Exact-k	Containment	Unique rank	Value invariance	Stability
Global-Top*k*	✓	✗	✓	✓	✓
U-Topk	✗	✗	✓	✓	✓
U-kRanks	✗	✓	✗	✓	✗
E-Rank	✓	✓	✓	✓	✓
Expected	✓	✓	✓	✓	✓
Median	✓	✓	✓	✓	✓
Qauntile	✓	✓	✓	✓	✓
PT-k	✗	✗	✓	✓	✓

7. CONCLUSIONS

We have surveyed different top-k query processing techniques in both relational as well as probabilistic databases. We have studied different semantics and properties of different top-k query processing techniques. Also we have reviewed how stream queries work using continuous query languages.

We envision the following research directions to be important to pursue:

How ambiguities can be removed from the possible worlds: the possible worlds which are generated according to the generation rules applied on either tuple-level or value-level uncertainty may have ambiguous data.

How the Cost of sharing data among queries in the same sliding window can be reduced by using the results of the queries lying in the same frequency window

8. REFERENCES

[1] Lukasz Golab, M. Tamer Ozsu, M. Tamer Özsu . "Data Stream Management." Canada: Morgan & Claypool publishers, 2010.

[2] Chonghai Wang, Li Yan Yuan, Jia Huai You, Osmar R. Zaiane, "On Pruning for Top-K Ranking in Uncertain Databases," in Proc. 37th International Conference on Very Large Data Bases, August 29th -September 3rd , 2011

[3] Xiang Lian, Lei Chen, "Probabilistic Ranked Queries in Uncertain Databases," in Proc. Extending Database Technology Conference, March 25-30, 2008.

[4] Graham Cormode , Feifei Li and KeYi, "Semantics of Ranking Queries for Probabilistic Data and Expected

Ranks," in Proc. ICDE IEEE 25th International Conference, 2009

[5]Yuan-Chi Chang, Lawrence Bergman, Vittorio Castelli, Chung-Sheng Li, Ming-Ling Lo, John R. Smith, "The onion technique: indexing for linear optimization queries," in Proc. ACM SIGMOD international conference on Management of data, Vol 29, pp 391-402, June 2000.

[6] Gang Luo , Kun-Lung Wu, Philip S. Yu, "SAO: A Stream Index for Answering Linear Optimization Queries," ICDE, pp 1302–1306, 2007

[7] Guo, L., Shao, F., Botev, C.,Shanmugasundaram, J, "Xrank: Ranked keyword search over xml documents," SIGMOD 16–27 (2003)

[8] Mohamed A. Soliman, Ihab F. Ilyas, "Ranking with Uncertain Scores," in Proc. IEEE International Conference on Data Engineering , 2009

[9] Ihab F. Ilyas, George Beskales, Mohamed A. Soliman. "A Survey of Top-k Query Processing Techniques in Relational Database Systems." ACM Computing Surveys(CSUR). Vol 40, Oct. 2008.

[10] Jeffrey Jestes, Graham Cormode, Feifei Li, and KeYi, "Semantics of Ranking Queries for Probabilistic Data", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERIN. Oct.2010.

[11] Xi Zhang, Jan Chomicki, "On the Semantics and Evaluation of Top-k Queries in Probabilistic Databases," Department of Computer Science and Engineering, University at Buffalo, SUNY, U.S.A.

[12] Charu C.Aggarwal and Philip S.Yu, "A Survey of Uncertain Data Algorithms and Applications," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERIN. vol. 21 pp.609-623, May. 2009.

[13] Tao Chen, Lei Chen, Member ,IEEE, M. Tamer Ozsu, Fellow ,IEEE, and Nong Xiao, "Optimizing multi-top-k queries over uncertain data streams," IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERIN. vol. 25 no. 8, pp.1814-1829, Aug. 2013.

[14] Chen, Jianjun, David J. DeWitt, Feng Tian, Yuan Wang, "NiagaraCQ: A scalable continuous query system for internet databases." ACM SIGMOD Record. Vol. 29. No. 2. ACM, 2000.

[15] Robert Kajic, "Evaluation of the Stream Query Language CQL," Thesis, Uppasala Universiy, May 2010.

[16] Arvind Arasu , Shivnath babu, jenifer widom, "The CQL continuous query language: semantic foundations and query execution" , VLDB Journal 2006.

[17] Shivnath Babu and Jennifer Widom, " Continuous Queries over Data Stream," Stanford University

[18] Jin Li ,David Maier, Kristin Tufte, Vassilis Papadimos, Peter A. Tucker, "Semantics and Evaluation Techniques for Window Aggregates in Data Streams," in Proc. ACM SIGMOD International Conference on Management of data, 2005.

[19] Mayur Datar, Aristides Gionis, Piotr Indyk, Rajeev Motwani, "MAINTAINING STREAM STATISTICS OVER SLIDING WINDOWS." Society for Industrial and Applied Mathematics. Vol. 31, No. 6, pp. 1794–1813, 2002

[20] Krämer, Jürgen, and Bernhard Seeger. "Semantics and implementation of continuous sliding window queries over data streams." ACM Transactions on Database Systems (TODS) 34.1 (2009)

[21] Nilesh Dalvi, Dan Suciu, "Efficient Query Evaluation on Probabilistic Databases", in Proc. 30th VLDB Conference, 2004.