A Model for Processing Skyline Queries over a Database with Missing Data

Ali A. Alwan[1,a], Hamidah Ibrahim[2,b], Nur Izura Udzir[2,c]

[1]Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur, Malaysia
[2]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, UPM, Serdang, Selangor, Malaysia
[a]aliamer@iium.edu.my, [b]hamidah.ibrahim@upm.edu.my, [c]izura@upm.edu.my

ISSN: 2231-8852

ABSTRACT

Skyline queries provide a flexible query operator that returns data items (skylines) which are not being dominated by other data items in all dimensions (attributes) of the database. Most of the existing skyline techniques determine the skylines by assuming that the values of dimensions for every data item are available (complete). However, this assumption is not always true particularly for multidimensional database as some values may be missing. The incompleteness of data leads to the loss of the transitivity property of skyline technique and results into failure in test dominance as some data items are incomparable to each other. Furthermore, incompleteness of data influences negatively on the process of finding skylines, leading to high overhead, due to exhaustive pairwise comparisons between the data items. This paper proposed a model to process skyline queries for incomplete data with the aim of avoiding the issue of cyclic dominance in deriving skylines. The proposed model for identifying skylines for incomplete data consists of four components, namely: Data Clustering Builder, Group Constructor and Local Skylines Identifier, *k*-dom Skyline Generator, and Incomplete Skylines Identifier. Including these processes in the proposed model has optimized the process of identifying skylines in incomplete database by reducing the necessary number of pairwise comparison through eliminating the dominated data items as early as possible before applying the skyline technique.

Keywords: skyline queries, incomplete data, preference queries, query processing

1. Introduction

Skyline queries incorporate and provide a flexible query operator that returns data items (skylines) which are not being dominated by other data items in all dimensions (attributes) of the database. In recent years, there has been tremendous emphasize on developing a database management system that incorporate and support more flexible query operators that return data items which dominate other data items in all attributes (dimensions). This type of query operations is named preference queries as they prefer a data item *p* over the other data item *q* if and only if *p* is better than *q* in all dimensions and not worse than *q* in at least one dimension. The preference queries are significant and mostly used in various application domains, like multi-criteria decision making applications, decision support system, multiobjective application and recommendation system, Hotel recommender (Michael, 2007), recipe finder (Alwan, et al., 2011) and restaurant finder (Al Kukhun, 2008; Mohamed, et al., 2009; Levandoski, et al., 2010) are typical examples that show the importance of preference queries. Furthermore, E-commerce environment is also a significant area that utilizes preference queries. For example, helping customer to make a tradeoff between the price, quality, and efficiency of the products to be purchased (Katerina, 2008).

Due to the importance of preference queries that obviously appeared in many database applications, various number of preference evaluation techniques have been proposed in the literature including but not limited to, top-*k* (Surajit and Luis, 1999), skyline (Börzsönyi, et al., 2001; Kossmann, et al., 2002; Chomicki, et al., 2003; Pei, et al., 2005; Yuan, et al., 2005; Godfrey, et al., 2005; Bartolini, et al., 2006; Jongwuk, et al., 2009), *k*-dominance (Chee-Yong, et al., 2006a), top-*k* dominating (Man & Mamoulis, 2007), and k-frequency (Chee-Yong, et al., 2006b).

One of the popular and most frequently used preference query types is the skyline queries. The main issue in the skyline queries is to find skylines by shrinking the search space as small as possible by focusing only on the set of data items that may potentially be the skylines.

The following running example depicts the process of skyline technique. Here, we assume a database consisting of 8 data items with 3 dimensions as illustrated in Fig. 1. The concept of skyline technique is working based on the process of pairwise comparison. This is achieved by comparing the values of the corresponding dimensions in the two data items in order to identify the skyline. From the database example the skylines are the data items t_1 and t_2 (smaller values are preferable) as these data items are not being dominated by the other data items. Since t_1 is better than t_2 in dimensions d_1 and d_3, while t_2 dominates t_1 in dimension d_2, thus both of them are skylines as none of them completely dominate each other. In Figure 1 the shaded rows represent the skylines of the database.

id	d_1	d_2	d_3
t_1	4	2	2
t_2	7	1	3
t_3	9	5	4
t_4	8	6	3
t_5	13	5	10
t_6	6	11	9
t_7	10	7	12
t_8	12	15	11

Fig. 1. Result of skyline technique

Several preference evaluation approaches which applied the concept of skyline have been proposed. The main issue concern for these approaches is reduces the searching space and the processing cost of finding skylines by terminating the process of finding skylines as early as possible. The searching space is determined by the number of pairwise comparisons that needs to be performed between the data items in identifying skylines. That means higher number of pairwise comparisons results into larger searching space and vice versa.

Most of the previous techniques assumed that all values of attributes (dimensions) are available and complete for all data items of the database (Börzsönyi, et al., 2001; Kian-Lee, et al., 2001; Kossmann, et al., 2002; Chomicki, et al., 2003; Pei, et al., 2005; Yuan, et al., 2005; Godfrey, et al., 2005; Yufei, et al., 2006; Bartolini, et al., 2006; Zhenhua & Wei, 2006; Michael, et al., 2007; Jongwuk, et al., 2009; Man & Mamoulis, 2009; Zhenhua, et al., 2010). However, this assumption is not necessary to be the case in the real world database particularly for databases with high number of dimensions as some values may be missing. Thus, the incompleteness of data introduces new challenges in processing skyline queries. The missing values will influence negatively on the process of identifying skylines leading to high overhead, due to exhaustive pairwise comparisons between the data items in determining the skylines. Besides, the incompleteness of data leads to the loss of the *transitivity property* of skyline technique which is held on all existing skyline techniques. This further leads to *cyclic dominance* between the data items as some data items are incomparable with each other and thus no data item is considered as skyline (Khalefa, et al., 2008). Thus, an approach is needed to avoid the issue of incomplete data and avoid the unnecessary pairwise comparisons between data items.

This paper presents a model for identifying skylines in incomplete database. This model consists of four components, namely: Data Clustering Builder, Group Constructor and Local Skylines Identifier, *k*-dom Skyline Generator, and Incomplete Skylines Identifier.

The rest of the paper is organized as follows. In Section 2, the previous works related to this research are reported. In Section 3, the basic definitions and notations, which are used in the rest of the paper are set out. In Section 4, we present the proposed model. Conclusion and the future work are presented in the final section, 5.

2. Related Works

There are many variations of skyline techniques in the literature, each focused on improving the efficiency and the performance of skyline process. The searching space is the most important critical factor that affects the performance of skyline process, thus most of the researches in this area concentrate on strategies that derive skylines with the aim at pruning the searching space as low as possible. Although many skyline approaches have been reported in the literature, but most of them failed to cover cases where the database contains incomplete data in which some missing values are present in one or more dimensions.

The work in (Börzsönyi, et al., 2001) is the first work that addresses the issue of skyline queries in the database field. In their work two different algorithms have been presented to generate the skylines, namely: Block-Nested-Loop (BNL) and Divide-and-Conquer (D&C).

(Kian-Lee Tan et al., 2001) present two incremental algorithms (Bitmap and Index) to produce the skylines of the database. Bitmap method utilizes the bitmap representation indicating the dimension with missing value as 0 bit and 1 otherwise and then perform the *and* bitwise operation.

Later, (Chomicki, et al., 2003) proposed Sort-Filter-Skyline (SFS) algorithm. SFS employed the concept of presorting the BNL skyline technique (Börzsönyi, et al., 2001) in order to return the skyline query results in a relational setting. Moreover, skyline process of SFS is less expensive.

While, Kossmann, et al., (2002) introduced an online skyline algorithm, Nearest Neighbor (NN), which computes the skylines continuously. NN method utilized the D&C scheme to split the indexed database into small regions, in order to prune the searching space recursively. However, NN method might become quite expensive operation when the number of involved dimensions is increased. Furthermore, the type of database significantly influenced on the quality and the size of the skylines. Branch-Bound-Skyline (BBS) which is proposed by Papadias, et al., (2003) is another attractive skyline technique that is inspired from NN technique. The aim of BBS is to achieve optimal cost of I/O that is needed to retrieve the skylines.

The work in (Godfrey, et al., 2005) proposed an algorithm named Linear Elimination Sort for Skyline (LESS), which is inspired from SFS (Chomicki, et al., 2003). LESS works on non-indexed data and does not require any additional pre-processing steps. LESS essentially adopts the benefits of BNL (Börzsönyi, et al., 2001) and SFS (Chomicki, et al., 2003). However, LESS algorithm requires some pre-processing processes like utilizing index structure such as sorted list to reduce the query operation cost. Bartolini, et al., (2005) highlighted on the issue of skyline queries by proposing Sort and Limit Skyline algorithm (SaLSa) algorithm which exploited the concept of SFS method in order to pre-sort the data items first and then progressively selects a subset of data items to examine the skylines. SaLSa executes the skyline queries on-top of the system; therefore, it is not required to understand the skyline semantics.

The work proposed by (Ken, et al., 2010) focus on the process of skyline queries in a database that is sorted in a Z-order (Z-curve). They argued that Z-order properties facilitated and improved the process of test dominance relationship among data items. For this purpose they proposed an efficient framework named Z-Sky that manipulates skyline queries and updates the skylines on Z-order curves.

However, preference queries have not received much attention in incomplete database applications in which to evaluate the query, exhaustive comparisons need to be performed in order to determine the relevant data items in the database. Preference queries in incomplete database are fundamentally different from the conventional preference queries in complete database because the transitivity property of preference techniques such as top-*k* and skyline is no longer hold. For that reason, developing an efficient algorithm to process preference queries in incomplete database is important.

In the following we present and discuss the preference queries proposed by previous researches in incomplete databases. The focus given by the researchers in this environment is highlighted.

The first work that tackled the issue of skyline queries in incomplete database is contributed by Khalefa, et al., (2008). They proposed two algorithms for handling the skyline queries in incomplete data, namely: *Bucket* and *Iskyline*. The *Bucket* algorithm is straightforward and divides the data items into distinct buckets based on their bitmap representation. Then, the conventional skyline algorithm is utilized on each bucket to identify the local skylines. Finally, the local skylines of each bucket are compared to each other to derive the final skylines. The *Iskyline* algorithm is an optimized version for *Bucket* by conducts two optimization techniques that reduce the number of local skylines in every node.

However, *Iskyline* is time consuming as in each node there are many pairwise comparisons that need to be performed to find the local skylines.

The work in (Haghani, et al., 2009) also studied the issue of top-*k* queries over continuous streaming uncertain and incomplete database systems. In their work two algorithms have been introduced to retrieve the top-*k* query answers, namely: Sorted List Algorithm (SLA) and Early Aggregation Algorithm (EAA).

Mohamed, et al., (2010) addresses the issue of ranking top-*k* queries in uncertain and incomplete database by introducing a probabilistic model that works under partial orders concept. Under the proposed model several algorithms have been described with the aim of pruning the searching space and filtering the database to efficiently process the query.

Later, Bharuka, & Kumar, 2013a) also discussed the issue of skyline queries in incomplete data. They have proposed an approach named Sort-based Incomplete Data Skyline (SIDS). SIDS assumed that input data is pre-sorted in non-increasing order to derive the skylines. To process the skyline queries, SIDS employs the round-robin to read the values of the dimensions and determines the next best value in that dimension to be chosen for processing. However, increasing the number of dimension influence negatively on the performance of the skyline process. Besides, sorting data items incur more cost on the execution of the skyline queries.

Lastly, to the best of our knowledge the work in (Bharuka, & Kumar, 2013b) is the most recent work that contributed in the area of skyline queries in incomplete data. They have introduced *Incomplete Data Frequent Skyline*, *IDFS* algorithm to process the skyline queries. The idea of IDFS relies on the concept of skyline frequency (Chee-Yong, et al., 2006b) to sort the skylines with missing values. Skyline frequency value of data item indicates that the data item is considered as a skyline in k dimensions where k is the number of dimensions with known values. Therefore, data item which has a high frequency called *top-k frequent* skyline and will be return in the skyline set. However, IDFS assumed that data items should be presorted in order to identify the skylines which also incur more processing time in deriving skylines.

3. Preliminaries

In this section, we provide the definitions and notations that are related to skyline queries in incomplete database which are necessary to clarify our proposed approach. Our approach has been developed in the context of incomplete multidimensional relational databases, D. A relation of the database D is denoted by $R(d_1, d_2, \ldots, d_m)$ where R is the name of the relation with m-arity and $d = \{d_1, d_2, \ldots, d_m\}$ is the set of dimensions.

Definition 1 Skyline: skyline technique retrieves the skyline, S in a way such that any skyline in S is not dominated by any other data items in the dataset.

Definition 2 Skyline queries: select a data item p_i from the set of dataset D if and only if p_i is as good as p_j (where $i \neq j$) in all dimensions (attributes) and *strictly* in at least one dimension (attribute). We use $S_{skyline}$ to denote the set of skyline data items, $S_{skyline} = \{p_i \mid \forall\ p_i, p_j \in D, p_i \succ p_j\}$.

Definition 3 Incomplete Database: given a database $D(R_1, R_2, ..., R_n)$, where R_i is a relation denoted by $R_i(d_1, d_2, ..., d_m)$, D is said to be *incomplete* if and only if it contains at least a data item p_j with missing values in one or more dimensions d_k (attributes); otherwise, it is *complete*.

Definition 4 Comparable: Let the data items a_i and $a_j \in R$, a_i and a_j are comparable (denoted by $a_i \, \varepsilon \, a_j$) if and only if they have no missing values in at least one identical dimension; otherwise a_i is incomparable to a_j (denoted by $a_i \not\varepsilon \, a_j$).

4. The Proposed Model of Skyline Queries

The proposed model for identifying skylines in incomplete database consists of four components, namely: Data Clustering Builder, Group Constructor and Local Skylines Identifier, *k*-dom Skyline Generator, and Incomplete Skylines Identifier as explained below. The major tasks of the proposed model are to identify the skylines from incomplete database. Fig. 2 illustrates the proposed model of skyline queries in incomplete database.

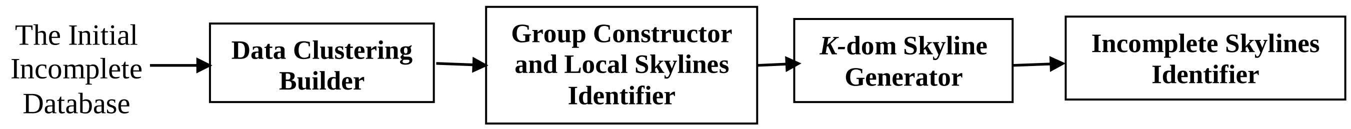
The Initial Incomplete Database
Data Clustering Builder
Group Constructor and Local Skylines Identifier
K-dom Skyline Generator
Incomplete Skylines Identifier

Fig. 2. The proposed model for processing skyline queries in a database with missing data

4.1. Data Clustering Builder

This component analyzes the initial database in order to identify the number of dimensions with missing values in each data item. Then, it divides the database into distinct clusters based on the common dimension(s) with missing values in each data item. The data items that are grouped in the same cluster are easily compared to each other without losing the transitivity property and avoiding the failure in the test dominance. Fig. 3 shows our running example database which will be used throughout this paper in clarifying the phases of the proposed approach. For simplicity the example that we have chosen is limited to a single missing value in each data item. However, our approach is capable of handling various numbers of missing values. The symbol (*) is used to represent the missing values in the data items.

id	***d1***	***d2***	***d3***
a_1	*	5	8
a_2	*	2	2
a_3	*	4	9
a_4	*	6	8
a_5	*	8	4
a_6	*	4	3
a_7	*	1	2
a_8	*	2	3
a_9	*	6	9
a_{10}	*	3	1
a_{11}	*	8	5
a_{12}	*	9	5

Cluster 1 = 011

id	***d1***	***d2***	***d3***
b_1	8	*	4
b_2	3	*	2
b_3	7	*	8
b_4	5	*	3
b_5	2	*	4
b_6	2	*	1
b_7	5	*	5
b_8	2	*	4
b_9	1	*	3
b_{10}	4	*	5
b_{11}	3	*	2
b_{12}	1	*	8

Cluster 2 = 101

id	***d1***	***d2***	***d3***
c_1	1	3	*
c_2	3	6	*
c_3	8	5	*
c_4	1	5	*
c_5	3	3	*
c_6	6	9	*
c_7	8	4	*
c_8	2	1	*
c_9	5	9	*
c_{10}	6	8	*
c_{11}	9	9	*
c_{12}	3	8	*

Cluster 3 = 110

Fig. 3. Clusters

Fig. 4 illustrates the algorithm of clustering data in our proposed model. We first read a data item from the initial database to determine the corresponding cluster of the data item (step 2). If the corresponding cluster has been created, i.e. the bitmap representation of the data item fits the bitmap representation of an existing cluster (step 3), then the data item is inserted into that cluster (step 4). Else, a new cluster is created based on the bitmap representation of the data item (step 7) and the data item is inserted into it (step 8). The above steps are repeated until all data items in the database are examined. In this step we ensure that data items with the same bitmap representation are gathered in the same cluster. The number of clusters created varies depending on the number of dimensions or combination of dimensions with missing values in the database. In addition, the number of data items in the whole clusters is equal to the number of data items in the initial database.

```
Algorithm 1
Input: Initial incomplete database, $D$
Output: A set of clusters, $C = \{C_1, C_2, \ldots, C_p\}$
 1. BEGIN
 2.     FOR each data item $a_i$ of $D$ DO
 3.         IF the bitmap representation of $a_i$ == the bitmap representation of an existing
 4. cluster $C_j$ THEN
 5.             Insert $a_i$ into $C_j$
 6.         ELSE
 7.             BEGIN
                    Based on the bitmap representation of $a_i$ create a cluster, $C_k$
 8.                 Insert $a_i$ into $C_k$
 9.             END
10. END
```

Fig. 4. The algorithm for clustering data

4.2. Group Constructor and Local Skylines Identifier

The aim of this component is to further classify the collected data items in each cluster into groups and then identify the local skylines in each cluster. The groups are constructed based on the highest value of any of the dimensions in the cluster. Then, local skylines are derived from the created groups. This component attempts to reduce the number of pairwise comparisons that needs to be performed in identifying skylines. Fig. 5 illustrates the concept of grouping technique. The shaded data items are the local skyline of the cluster.

Group 1			
id	***d1***	***d2***	***d3***
a_{12}	*	9	5
a_9	*	6	9
a_3	*	4	9
Group 2			
a_4	*	6	8
a_{11}	*	8	5
a_5	*	8	4
a_1	*	5	8

Cluster 1

Group 1			
id	***d1***	***d2***	***d3***
b_7	8	*	7
b_{12}	1	*	8
b_1	8	*	4
Group 2			
b_7	5	*	5
b_4	5	*	3
b_{10}	4	*	5

Cluster 2

Group 1			
id	***d1***	***d2***	***d3***
c_{11}	9	9	*
c_6	6	9	*
c_9	5	9	*
Group 2			
c_3	8	5	*
c_{10}	6	8	*
c_{12}	3	8	*
c_7	8	4	*

Cluster 3

Fig. 5. Groups and local skylines

4.3. *k*-dom Skyline Generator

The essential component of the proposed model of skyline queries in incomplete database is *k*-dom skyline generator. The main function of this component is to derive a set of virtual skylines formed out of the local skylines for each cluster. Then, the derived *k*-dom skylines are combined to produce one global *k*-dom skyline. The global *k*-dom skyline is inserted at the top of each cluster to prevent the dominated data items from further processing.

Fig. 6 shows the mapped virtual skyline after deriving the virtual skyline *k*-dom and applying the mapping policy. For example *k*-dom in Cluster 1 comes from the mapping of cluster skyline of Clusters 2 and 3 by considering the highest value in each dimension. That means local skyline Cluster2 and Cluster 3 produced the *k*-dom (*, 9, 7). The mapping process will significantly reduce the number of comparisons. Instead of comparing two virtual skylines (b_7 and c_{11}) independently with the local skyline of Cluster 1, we insert this *k*-dom data item into the Cluster 1 and start the skyline process to eliminate the dominating local skyline from further processing. Notice that, the value of the first dimension in the *k*-dom is *, this is because in Cluster 1 the incomplete dimension is the first dimension and will not be used in the comparison process. While, for dimensions 2 and 3 we replaced the * symbol with the highest value of that dimension. Also notice, because a_9 is better and not worse than *k*-dom, a_9 will be further considered to be as a global skyline. *k*-dom removes a_{12} from further processing because *k*-dom is better than a_{12} in all the complete dimensions.

Inserting *k*-dom to local skyline of cluster 1			
k-dom	*	9	7
a_9	*	6	9
a_{12}	*	9	5

Cluster 1

Inserting *k*-dom to local skyline of cluster 2			
k-dom	6	*	5
b_7	8	*	7
b_{12}	1	*	8

Cluster 2

Inserting *k*-dom to local skyline of cluster 3			
k-dom	8	9	*
c_{11}	9	9	*
c_6	6	9	*

Cluster 3

Fig. 6. Effect of *k*-dom on the candidate skyline

4.4. Incomplete Skylines Identifier

This component further compares the non-dominated local skylines. The incomplete skylines identifier aims to ensure that any incomplete global skyline is a skyline over the entire database. Besides, it also ensures that any eliminated local skylines are not part of the global skylines. The final result is retrieved after conducting pairwise comparisons among the candidate skylines to determine which data item should be displayed to the user. For this purpose, *Algorithm* 2 as presented in Fig. 7 can be utilized in which the input is a set of candidate skylines produced by the previous phase while the output is a set of final skylines. Given a set of data items, each data item in the set is analyzed (step 2). For each data item a_l of the set (step 3), if the data item a_k dominates a_l (step 5), then a_l is deleted from the set (step 6). Else, if the data item a_l dominates the data item a_k (step 7), then a_k is deleted from the set (step 8). This process is repeated for all data items in the set. Finally, the remaining data items of the set are inserted into the set of skylines (step 10).

```
Algorithm 2
Input: A set of data items, $I = \{a_1, a_2, ..., a_n\}$
Output: A set of skylines, $S$
1. BEGIN
2.     FOR each $a_k$ in $I$ DO
3.         FOR each $a_l$ where $k <> l$ in $I$ DO
4.         BEGIN
5.             IF data item $a_k \succ a_l$ THEN
6.                 Delete $a_l$ from $I$
7.             ELSE IF data item $a_l \succ a_k$ THEN
8.                 Delete $a_k$ from $I$
9.         END
10.    Insert data items of $I$ to $S$
11. END
```

Fig. 7. The skylines algorithm

Fig. 8 shows the global skyline of our example database. The shaded data item is the final global skyline of the example database. From the result we conclude that there are no data items in any of the clusters that are better than the global skyline. This is due to the process of pairwise comparisons that has been performed in the earlier stage between the local skylines of the clusters and the *k*-dom. Apparently, *k*-dom skyline has help in preventing the dominated data items from further processing and this is result into reduces the number of pairwise comparisons between data items.

Global skyline			
c_{11}	9	9	*
a_9	*	6	9
b_7	8	*	7
b_{12}	1	*	8

Fig. 8. The final global skyline

5. Conclusions

Skyline queries have received high interest from database community due to its great benefits on various types of database applications. It is also known that skyline query is an expensive operation and need exhaustive pairwise comparisons between data items to derive the skylines. In this paper we have proposed a model that attempts to process skyline queries in incomplete data with the aim of reducing the searching space by avoiding the unnecessary comparisons between the data items. The clustering and grouping with the concept of virtual *k*-dom skyline are used during the process of deriving the set of skylines for a given query operation.

As the future work directions for our work in this paper, we plan to evaluate our developed model over some real and synthetic datasets to investigate the impact of the missing rate, the size of dataset and the number of dimensions of the data items In addition, we also plan the examine our proposed model for incomplete distributed databases where by database tables are spread over various sites. In this context the result of the skyline queries needs to be collected from different tables located at different locations.

References

Chee-Yong, C., Jagadish, H.V., Kian-Lee T., Anthony K.H. T., and Zhang, Z.(2006(a)). On High Dimensional Skylines. *In the 10th International Conference on Extending Database Technology (EDBT06),* Munich, Germany, March 2006, pp. 478-495.

Chee-Yong C., Jagadish, H.V., Kian-Lee T., Anthony K.H. T., and Zhang, Z. (2006(b)). Finding *k*-dominant Skylines in High Dimensional Space. *In the ACM SIGMOD International Conference on Management of Data*, Chicago, IL, USA , June 2006, pp. 503-514.

Kian-Lee, T., Pin-Kwang, E., and Beng, C. O. (2001). Efficient Progressive Skyline Computation. In the 27th International Conference on Very Large Data Bases (VLDB 27), September 2001, pp. 301-310, Man L. Y. & Mamoulis, N. (2007). Efficient Processing of top-k Dominating Queries on Multi-Dimensional Data. *In the 33rd International conference on Very Large Data Bases (VLDB 33)*, Vienna, Austria, September 2007, pp. 483-494.

Al Kukhun, D., Soukkarieh B., Lopez-Ornelas E., and Sedes, F. (2008). LA-GPS: A Location-aware Geographical Pervasive System. *In the 24th International Conference on Data Engineering Works (ICDEW'08)*, Cancun, Mexico, April 2008, pp. 160-163.

Alwan, A. A., Ibrahim, H.., Tan C. Y., Sidi, F., and Udzir, N. I. (2011). Preference Evaluation of Preference Queries Techniques over a High Multidimensional Database. *In the 3rd International Conference on Networked Digital Technologies (NDT11)*, Macau, China, July 2011, pp. 212–223. Springer.

Jongwuk, L., Gae-won, Y., and Seung-won, H. (2009). Personalized top-k Skyline Queries in High-Dimensional Space. *Journal of Information Systems*, vol. 34, no. (1) (2009), pp. 45-61.

Papadias, D., Tao,. Y., Fu, G., and Seeger, B. (2003). An Optimal and Progressive Algorithm for Skyline Queries. *In the International Conference on Management of Data*, San Diego, California, USA, June 2003, pp. 467-478.

Kossmann, D., Ramsak, F., and Rost, S. (2002). Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. *In the 28th International Conference on Very Large Data Bases (VLDB 28),* Hong Kong, China, August 2002, pp. 275-286.

Haghani P., Michel S., and Aberer K. (2009). Evaluating top-k Queries over Incomplete Data Streams. *In the 18th ACM conference on Information and Knowledge Management (CIKM'09)*, Hong Kong, China, November, 2009, pp. 877-886.

Bartolini, I., Ciaccia, P., and Patella, M. (2006). SaLSa: Computing the Skyline without Scanning the Whole Sky. *In the 15th International Conference on Information and Knowledge Management (CIKM'06),* Arlington, Virginia, USA, November 2006, pp. 405-414.

Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2003). Skyline with Presorting. *In the 19th International Conference on Data Engineering (ICDE03),* Bangalore, India, March 2003, pp. 717-816.

Pei, J., Jin, W., Ester, M., and Tao, Y. (2005). Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces. *In the 31st International Conference on Very Large Data Bases (VLDB 31),* Trondheim, Norway, September 2005, pp. 253-264.

Katerina F., Evaggelia P. (2008). BITPEER: Continuous Subspace Skyline Computation with Distributed Bitmap Indexes. *In the International Workshop on Data Management in Peer-to-Peer Systems (DaMaP'08),* Nantes, France, March 2008, pp. 35-42.

Ken C. K. L., Wang-Chien L., Baihua Z., Huajing L., and Yuan T. (2010). Z-Sky: An Efficient Skyline Query Processing Framework Based on Z-order. *The Very Large Data Bases Journal*, vol. 19, no. (3), (2010), pp. 333-362.

Börzsönyi, S., Kossmann, D., and Stocker, K. (2001). The Skyline Operator. *In the 17th International Conference on Data Engineering (ICDE 2001),* Heidelberg, Germany, April 2001, pp. 421-430.

Surajit C. and Luis G. (1999). Evaluating Top-k Selection Queries. *In the 25th International Conference on Very Large Data Bases (VLDB 25),* Edinburgh, Scotland, September 1999, pp.397-410.

Yuan, Y., Lin. X., Liu,. Q,, Wang, W., Xu Y. J,, and Zhang, Q. (2005). Efficient Computation of the Skyline Cube. *In the 31st International Conference on Very Large Data Bases (VLDB 31)*, Trondheim, Norway, September 2005, pp.267-278.

Man, L. Y. and Mamoulis, N. (2009). Multi-Dimensional top-k Dominating Queries. *The Very Large Data Bases Journal*, vol. 18, no. (3), (2009), pp. 695–718.

Khalefa, M. E., Mokbel, M. F., and Livandoski, J. J. (2008). Skyline Query Processing For Incomplete Data. *In the 24th International Conference on Data Engineering (ICDE 2008)*, Cancun, Mexico, April 2008, pp. 556-565.

Mohamed, F. M., Justin, L. (2010). Toward context and preference-aware location-based services. *In the 8th International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE09)*, RI, USA, August 2010, pp. 25-32.

Godfrey, P., Shipley, R., and Gryz, J. (2005). Maximal Vector Computation in Large Data Sets. *In the 31st International Conference on Very Large Data Bases (VLDB31)*, Trondheim, Norway, September 2005, pp. 229-240.

Michael, D. M., Jignesh M. P., and William I. G. (2007). Efficient Continuous Skyline Computation, *International Journal of Information Science*, 177, 17, (2007), 3411-3437.

Yufei, T., Xiaokui, X., and Jian, P. (2006). SUBSKY: Efficient Computation of Skylines in Subspaces. *In the 22nd International Conference on Data Engineering, (ICDE 2006)*, Atlanta, Georgia, USA, April 2006, pp. 65-74.

Zhenhua, H., and Wei, W. (2006). A Novel Incremental Maintenance Algorithm of SkyCube. *In the 17th International Conference of Database and Expert Systems Applications (DEXA 2006)*, Kraków, Poland, September 2006, pp. 781-790.

Zhenhua H., Shengli S., and Wei W. (2010). Efficient Mining of Skyline Objects in Subspaces over Data Streams. *Knowledge and information systems journal*, vol. 22, no.(2) (2010), pp. 159-183.

Levandoski, J. J, Mokbel, F. M., Khalefa, M. E. (2010). FlexPref: A Framework for Extensible Preference Evaluation in Database Systems. *In the 26th International Conference on Data Engineering, (ICDE2010)*, Long Beach, California, USA, March 2010, pp. 828-839.

Mohamed A. S., Ihab F. I. and Shalev B. (2010). Supporting ranking queries on uncertain and incomplete data. *Very Large Data Base Journal*, vol. 19, no.(4), 2010, pp. 477 – 501.

Bharuka, R. and Kumar, S. P. (2013a). Finding skylines for incomplete data. *In the 24th Australasian Database Conference, ADC '13*, Adelaide, Australia, February 2013, pp. 109-117.

Bharuka, R. and Kumar, S. P. (2013b). Finding superior skylines points from incomplete data. *In the 19th International Conference on Management of Data (COMAD)*, Ahmedabad, India, December 2013, pp. 35-44.