Journal of Computer Science

Review

Processing Skyline Queries in Incomplete Database: Issues, Challenges and Future Trends

Yonis Gulzar, Ali A. Alwan, Norsaremah Salleh and Imad Fakhri Al Shaikhli

Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia

Corresponding Author:
Ali A. Alwan
Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, Kuala Lumpur 53100, Malaysia
Email: aliamer@iium.edu.my

Abstract: In many contemporary database applications such as multi-criteria decision-making and real-time decision-support applications, data mining, e-commerce and recommendation systems, users need query operators to process the data aiming at finding the results that best fit with their preferences. Skyline queries are one of the most predominant query operators that privileges to find the query results that return only those data items whose dimension vector is not dominated by any other data item in the database. Because of their usefulness and ubiquity, skyline queries have been incorporated into different types of databases such as complete, incomplete and uncertain. This paper attempts to survey and analyze the previous works proposed to process skyline queries in the incomplete database. The discussion focuses on examining these approaches highlighting the strengths and the weaknesses of each work. Besides, we also discuss in detail the current challenges in processing skyline queries in the incomplete database and investigate the impact of incomplete data on skyline operation. A summary of the most well-known works has been reported to identify the limitations of these works. Some recommendations and future work directions have been drawn to help researchers investigate the unsolved problems related to skyline queries in a database system.

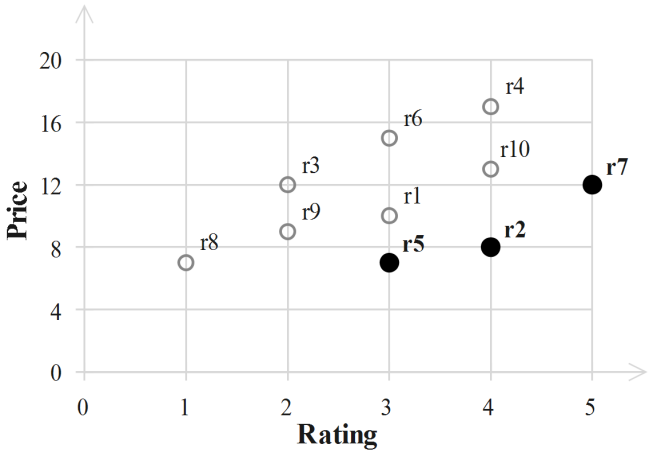
Keywords: Preference Queries, Skylines, Skyline Queries, Algorithms, Incomplete Data, Database

Introduction

Prior to the first introduction of the skyline operator into database community by Borzsony *et al*. (2001), there was a problem called *maximum vector problem* or *Pareto optimum* that has been addressed by (Bentley *et al*., 1978). Skyline queries attempt to return a set of data items S from a database, where S contains only those data items whose dimension values are not dominated by any other data item exist in the database. For instance, suppose there are two data items i and j comprise of a set of dimensions (attributes) belong to a database, D. Assume the data item, i belongs to S and D, while j belongs to D only. The data item i can be in the skyline set, S if and only if i has better values than j in all dimensions or at least it has value not worse than j in one dimension. The following restaurant database example in Fig. 1 clarifies the concept of skyline queries in database systems. Assume the database table contains the details of 10 restaurants, in which each restaurant consists of two dimensions. The first dimension represents the rating of each restaurant given by customers while the second dimension indicates the price of the food served in each restaurant. Fig. 1(B) illustrates the representation of the restaurant database example in 2-Dimensions space. Assume a user is looking for the best restaurant in the city where the price is the cheapest and the restaurant rate is the highest. Based on skyline definition and from the database example, it can be seen that restaurants $r8$, $r9$, $r3$, $r1$ and $r6$ are partially dominated by $r5$ or $r2$ either based on the first dimension (price) or the second dimension (rate). While restaurants $r4$ and $r10$ are partially dominated by restaurant $r7$ only. It can be noticed for restaurant $r8$ and $r5$ their price values are cheapest among all other restaurants but $r8$ do not have the better rating than any of the other restaurants. Therefore, $r8$ is dominated by $r5$. Similarly, restaurant $r7$ has higher price value compare with $r1$ and $r9$, however, it has better rating value compare with any other restaurant. Hence, restaurant $r7$ is not dominated by any other restaurant in the restaurant database. As demonstrated in Fig. 1(A), the result of applying skyline technique on restaurant database example will retrieve $r5$, $r2$ and $r7$ as the skylines of the restaurant database.

ID	Rating	Price
r1	3	10
r2	4	8
r3	2	12
r4	4	17
r5	3	7
r6	3	15
r7	5	12
r8	1	7
r9	2	9
r10	4	13

(A)



(B)

Fig. 1: Example of skyline query

Due to the powerful characteristics and the tremendous benefits of skylines queries, it has gained a formidable attention by many researchers in the area of database for the last decade. From the literature, skyline queries have been widely used in different real world database applications. The list includes but not limited to multi-criteria decision-making system, personalized systems, recommendation system, data mining, decision support system (Chan *et al*., 2006a; 2006b; Yiu and Mamoulis, 2007; Kontaki *et al*., 2008), hotel recommender (Godfrey *et al*., 2005), restaurant finder (Mokbel and Levandoski, 2009; Kukhun *et al*., 2008), and data stream (De Matteis *et al*., 2016). In these systems various (contradict) interests might be combined helping users by recommending a strategic decision. The list also includes e-commerce (Godfrey *et al*., 2005) as skyline queries could benefit customers to make the best choice through the trade-off between the price, quality and efficiency of the products to be purchased. Due to this viable utilization of skyline operator, skyline queries have been incorporated in numerous contemporary database applications. Skyline queries have been embedded in many databases types such as relational, web, crowd-sourcing, probabilistic, uncertain and incomplete databases.

Two significant concerns have been raised by the researchers since the introduction of skyline queries into the database community. The first concern is how to derive the skylines with the intention of shrinking the searching space. While the second concern is to avoid scanning the whole database and limiting the search to data items that have high potential to be in the skyline results. Many solutions that rely on the concept of skyline technique have been suggested aiming at deriving skylines. These include Divide-and-Conquer (D&C), Block Nested-Loop (BNL) (Borzsony *et al*., 2001), Bitmap and Index (Tan *et al*., 2001), Sort Filter Skyline (SFS) (Chomicki *et al*., 2003), Nearest Neighbor (NN) (Kossmann *et al*., 2002), Linear Elimination Sort Skyline (LESS) (Godfrey *et al*., 2005), Branch and Bound Skyline (BBS) (Papadias *et al*., 2003) and Sort and Limit Skyline algorithm (SaLSa) (Bartolini *et al*., 2006).

From the literature, most of these solutions assumed that data item values are present and dimensions (attributes) are always populated with some values (Borzsony *et al*., 2001; Yiu and Mamoulis, 2007; Godfrey *et al*., 2005; Tan *et al*., 2001; Chomicki *et al*., 2003; Kossmann *et al*., 2002; Bartolini *et al*., 2006; Yuan *et al*., 2005; Chomicki *et al*., 2005; Pei *et al*., 2005; Huang and Wang, 2006; Lee *et al*., 2009; Yiu and Mamoulis, 2009). Nonetheless, it is argued that this assumption is not necessary to be always true, particularly for a database with a high number of dimensions and a tremendous amount of data (Alwan *et al*., 2016; Khalefa *et al*., 2008). In many of the contemporary databases application such as crowd-sourcing, temporal, spatial, probabilistic, uncertain and big data databases, it is most likely that some values are missing due to many reasons. Some of the reasons include imprecise (incomplete) data entry which might be omitted by the user when working on the database. Besides, inaccurate data from heterogeneous data sources, in most of the online databases the data come from different remote resources such as sensor. As a result, the received data may miss some values in one or more attributes (dimensions). Last but not the least, integrating heterogeneous schemes in which the mediator systems combine the attribute values of local schemas to obtain a universal global scheme. The process of data integration may lead to missing some values in one or more attributes (dimensions) as some attributes may not appear in all local schemas. These are some of the typical examples that demonstrate the causes that lead change the database to be incomplete. Thus, the incompleteness of data introduces new challenges in processing user queries on the databases and particularly skyline queries, which involve the whole data items due

to the process of pairwise comparison between the values of the database dimensions to identify the skylines. The database is said to be *incomplete* if and only if it contains at least a data item with missing values in one or more dimensions. It has been reported that the missing values have a direct negative impact on processing skyline queries and in many cases, resulting high overhead, due to exhaustive pairwise comparisons between the data items. Most importantly, the incompleteness of data leads to the loss of the *transitivity property* of skyline technique which is held on all existing skyline techniques applied on complete data. 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 a skyline (Alwan *et al*., 2016; Khalefa *et al*., 2008). We provide a formal definition for the incomplete database as follows.

Incomplete Database

A database is said to be incomplete if and only if values of some dimensions are not present (missing). Otherwise, it is complete.

With these complicated challenges, it is impractical to directly apply skyline approaches designed for the complete database on a database with partial incomplete data due to the prohibitive cost and the exhaustive pairwise comparisons that need to be conducted between data items. Most importantly, it is highly unlikely to obtain the correct set of skylines from the database. This is because a skyline process might suffer from the issue of *cyclic dominance* and losing the transitivity property of skyline which results into incorrect results or in the worst case might return an empty set of skylines. The following running example clarifies further these challenges. For instance, we assume that a tourist is looking to rent a car somewhere in a city with lower rent cost per hour, less consumption and a good rating given by other users (tourists). A car renting database consists of three data items with three dimensions, namely: $d_1 = rph$, $d_2 = cpl$ and $d_3 = r$, where *rph* represents the rent price per hour, while *cpl* indicates the consumption rate per litter and lastly *r* denotes the rating value of each car by users. We also assume that there are some data items with missing values, which means that some dimension values are not present. Thus, the data items of the database are as follows. c_1 (5, 17, -) c_2 (6, -, 5) c_3(-, 20, 3). The symbol (-) denotes that the value is missing in that dimension of the data item. Based on the common dimensions with non-missing values it can be concluded that c_1 dominates c_2 as c_1 is better than c_2 (lower is better) at first dimension *rph*, while c_2 dominates c_3 as c_2 had a better rating than c_3 (third dimension, *r*). However, comparing c_1 with c_3 demonstrate that c_3 dominates c_1 in the second dimension, *cpl*. Hence, car c_1 does not dominate c_3 which therefore means the dominance relation is not transitive. Furthermore, car c_3 is dominating c_1, which means that the dominance relation is *cyclic*. From this example, all three cars are being dominated by one another and therefore the skyline process fails to decide the best car for the tourist.

In this study, we are focusing on examining and discussing the previous approaches proposed for processing skyline queries with incomplete data in three different databases, namely: centralized, distributed and cloud databases. These approaches have been analyzed thoroughly describing the strengths and the weakness of these approaches. Furthermore, we also discuss the current research trends related to skyline queries in some other environments and highlights the most issues concern skyline queries in these contexts.

The rest of the paper is organized as follows. Section 2 gives the basic definitions and notations, which are related to the concept of skyline techniques and used throughout the paper. In Section 3, the related literature of skyline queries in incomplete database over various contexts is reported and discussed. Besides, we also summarize the related works covered in this survey paper highlighting some critical aspects of these works. A discussion on the raised challenges due to the imperfection of data is given in Section 4. Furthermore, we included some significant future work directions that can be undertaken based on the works presented in this study. Finally, Conclusions are presented in Section 5.

Definition, Notations and Concept of Skyline

This section provides a set of necessary definitions and notations that facilitate the process of understanding the concept of skyline queries in incomplete databases. This includes definitions related to the dominance theory of the skyline, the skyline queries, incomplete database, comparable, distributed databases and dynamic databases. These terms are further explained as follows.

A relation of the database D is denoted by R (d_1, d_2, ..., d_m) where R is the name of the relation with m-arity and $d = (d_1, d_2, ..., d_m)$ is the set of dimensions.

Definition 1 Dominance

Given two data items p_i and $p_j \in D$ database with d dimensions, p_i dominates p_j (the greater is better) (denoted by $p_i \succ p_j$) if and only if the following condition holds: $\forall d_k \in d, p_i.d_k \geq p_j.d_k, \Lambda \exists d_l, \in d, p_i.d_l > p_j.d_l$.

Definition 2 Skyline Queries

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

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 $a_i \not\varepsilon a_j$).

Definition 5 Dynamic Database

A database *D* is said to be dynamic if its contents are persistently change through many update operations such as DELETE, UPDATE or INSERT.

In dynamic databases processing skyline is quite challenging due to the continuous update of the data in the database. This frequent change might impact negatively on the validity of the skyline and force users to re-compute the skyline process whenever the data is updated. Re-evaluating the skylines by accessing the whole database is an exhaustive process that incurs longer processing time and higher overhead for the skyline process. Therefore, this approach should be avoided and provide a solution that partially scans the database when there is an update on the contents of the database.

By examining the concept of skyline queries the following characteristics can be concluded. (i) no monotone ranking function is required to identify the skylines as each dimension (attribute) is evaluated separately, (ii) the skylines are derived by merely exploit the actual existing values of the data, (iii) the scale of dimension does not affect skyline results, (iv) incorporating skyline operator into SQL query processor is extremely simple and easy, (v) skyline queries can be considered a qualitative approach, which means the focus is higher on the quality of the result rather than the quantity (vi) Skyline operator carries out multi-criteria query operations in an easy way, (vii) it exhibits the transitivity property that leads to avoid many unnecessary pairwise comparisons and (viii) it is used as an effective operator for the aggregate operation (Chan *et al*., 2006a; 2006b; Yiu and Mamoulis, 2007; Alwan *et al*., 2016; Kontaki *et al*., 2010).

Nevertheless, despite the positive aspects, skyline queries have some limitations and weaknesses as listed below: (i) The size of skyline set is uncontrolled. This means the most extreme worst case is that all the data items in the database might be returned as skylines. (ii) The computation complexity of skyline process is highly impacted by the database size and the number of dimensions that consists of the database. This means when the volume of the database and/or the number of dimensions are high, then the searching space will be wider and causing large number of pairwise comparisons. (iii) With the immense size of skylines, the user needs to examine many skylines in order to determine the most appropriate one to be chosen. (iv) It might not be easy to identify the skylines when the database is incomplete or imprecise due to the issue of *cyclic dominance* and losing the *transitivity property* of skyline. (v) It is impractical to pre-compute the skylines to save the processing time when data item values are frequently changing and thus reapply the skyline techniques on the whole database to identify the new skylines should be performed (Alwan *et al*., 2016; Khalefa *et al*., 2008; Gulzar *et al*., 2016).

Related Literature of Skyline Queries on Incomplete Databases

Database with incomplete data has been considered as a rich area of research in the database community for many years. It has been readdressed in the recent years and becomes a common problem in many real-life contemporary database applications such as crowd-sourcing, web, cloud and big data databases. In this context, query processing is challenging as in many scenarios there is a significant part of the query result that may be omitted due to the missing values in some dimensions. Besides, for skyline queries the incompleteness of the data might incur a high cost to derive the skylines and the results produced will be incomplete and might be useless and eventually misguide the user. For these reasons a large number of various solutions relying on skyline techniques are proposed in the database literature tackling the issue of processing skyline queries on incomplete data. These approaches focus mainly on the issue of cyclic dominance and losing the transitivity property of skyline due to the missing values in the database. Besides, the previous works have also addressed the issue of enhancing the efficiency and the performance of skyline process by reducing the searching space and avoid scanning the whole database in order to determine the skylines. This has been achieved by developing strategies attempt to avoid the unnecessary pairwise comparisons between data items and eliminate many dominated data items before applying skyline process. In the following, we present and discuss the previous existing approaches proposed for processing skyline queries in the incomplete database including centralized, distributed and cloud computing databases. The focus given by the researchers in these environments is highlighted.

The work in (Khalefa *et al*., 2008) is the first work contributed towards raising the issue of incomplete data on processing skyline queries. Three different algorithms have been introduced, namely: *Replacement*, *bucket* and *ISkyline*. The idea of *replacement* algorithm is very trivial and relies on substituting the missing values of the incomplete dimensions with $-\infty$ to make the database complete. Then, applying the conventional skyline method on the database to determine the skylines. This solution is not effective and might be impractical in many cases. Therefore, *bucket al*gorithm has been introduced to overcome issues raised by *replacement* technique whereby data items are divided into discrete buckets according to

their bitmap representation in which dimension with missing value denoted as 0, otherwise denoted as 1. Each bucket holds the data items of the same pattern and missing values found in the same dimension(s). Then, utilizing traditional skyline algorithm on each bucket to defining the local skylines of each bucket. Finally, the local skylines of each bucket are further compared against local skylines of other buckets in order to determine the final skylines. However, bucket approach introduced many unnecessary pairwise comparisons and propagate many dominated data items for further processing. Hence, *ISkyline* has been introduced aiming at optimizing the skyline process over the incomplete data by eliminating many dominated data items which result into avoiding lots of unnecessary pairwise comparisons. The idea of *ISkyline* relies on exploiting the concept of bucketing designed for buck*et al*gorithm and apply two optimization techniques (virtual and shadow skylines) that significantly decrease the number of local skylines in every node which in turn reduce the number of pairwise comparisons. However, *ISkyline* might suffer from exhaustive pairwise comparisons when the number of dimensions increases, the missing rate is high and/or the size of the database is very large. These factors result into generating a large number of buckets and perform many pairwise comparisons to identify the skylines. Besides, *ISkyline* has been designed to fit with a centralized database and it might not be an effective solution for distributed or cloud databases.

The next work that also concentrates on skyline query processing in incomplete database is reported in Arefin and Morimoto (2012). They proposed an algorithm named Replacement Based Skyline Sets Queries, as RBSSQ to handle skyline operation on incomplete database. RBSSQ consists of two stages: *Data Processing* and *Skyline Sets Computation*. In the first stage, they processed the data by employing the concept of replacement algorithm proposed in (Khalefa *et al*., 2008) in order to substitute the missing values and change the database status from incomplete to complete. Though, unlike *replacement* algorithm, they used the value that is outside the domain values depending on the nature of the dimensions to replace missing values. If smaller values are preferred for processing then all missing values will be replaced by larger values than the domain value or vice versa. Then, the final skylines are computed in *Skyline Sets Computation* stage by exploiting oracle function. They argued that RBSSQ is more secure than any other previously proposed approaches by not disclosing the individual values of data items, followed by producing the aggregate values of data items. Like *bucket* and *ISkyline*, RBSSQ has been designed to work on centralized incomplete database.

Bharuka and Kumar (2013) proposed *Sort-based Incomplete Data Skyline* (SIDS) algorithm for evaluating the skyline over incomplete data. The algorithm employed a sorting function to pre-sorted data as an input in descending order for each dimension. Then, the first data item of the first dimension is chosen and compared with the remaining data items present in the same dimension. The process is repeated on first data item from the second dimension and so on. By doing so, all data items are being selected from processing in a round-robin fashion. The main idea behind processing data items in a round-robin order is to prune the dominated data items in early stages of skyline process to reduce the amount of data items to be involved which in consequence minimize the number of comparisons performed to identify the skylines. The process starts by considering all data items as candidate skylines and then iteratively remove those dominated data items from the candidate set. The data items that are yet to be pruned and are further processed indicates that this data item is presented in all dimensions and will be defined as a skyline. Note that SIDS works fine over a database with a small volume of data, but when it comes to a database with a huge amount of data, it might result into increasing the execution time of the skyline operation. This is because SIDS algorithm lacks the concept of utilizing the parallel execution that might speed up the process of skyline computation. SIDS has been designed to work on an incomplete database with single access and only one database relation will be accessed to run the skyline query.

Furthermore, Miao *et al*. (2013; Gao *et al*., 2014) proposed a set of algorithms with the intention of processing skyline queries in incomplete database. This includes *baseline algorithm*, *Virtual Point based algorithm* (VP) and *k-iSkyband algorithm* (*k*ISB); these algorithms employed the concept of the work proposed in (Khalefa *et al*., 2008). The first algorithm, *baseline algorithm* divides the initial data items into different clusters to be spread over many buckets based on the bitmap representation of the data item. Then identify a set of data items named candidate *k*Skyband, C*k*SB which represent those non-dominated data items of every bucket. Finally, identify the global skylines of the database using global *k*Skyband, G*k*SB technique which returns selected data items after cross domination tests between C*k*BS data items from other buckets. *Virtual Point*, (*VP*) algorithm has been proposed to overcome the inefficiency of G*k*SB algorithm. The idea of virtual points is as follows: A set of virtual points are generated and chosen to be compared with cross buckets instead of comparing all data items of each C*k*SB. However, creating virtual points for each bucket may increase the redundancy of the virtual points that leads to an extra memory usage. Thus, *k*ISB algorithm has been suggested to avoid the issue of virtual points redundancy and helps to prune unwanted C*k*SB as early as possible before performing cross domination tests.

Gulzar *et al*. (2016; 2017a) proposed a model to process skyline queries for centralized incomplete database relies on the concept of SIDS approach introduced in (Papadias *et al*., 2003). The proposed model comprises four phases in which the first phase

scan, the initial data items in order to sort them in descending order based on each dimension to keep those data items on top of the list that have a high probability to be part of skyline result. Then, indices of data items are stored in constructed arrays to keep track of dimension values of each data item. In the second phase, the sorted data items are read according to round-robin fashion for each constructed array (number of arrays constructed is equal to total number of dimensions with no missing values). During scanning the presence of data items is counted until each data item of the database is scanned at least once. In the third phase, eliminating those dominated data items before applying skyline technique. Elimination is conducted based on the count value of each data item in which data items with count less than two will be eliminated and excluded from further processing. Lastly, the remaining data items in the candidate set will be compared against each other in order to determine the final skylines. This is achieved by comparing each data item in the candidate set with each other and the non-dominated data items of the candidate list will be reported as skylines.

Moreover, Zhang *et al*. (2016) proposed a framework named Compared One by One, (COBO) that exploits a skyline technique called ISSA to compute the skyline. The framework carries out the process of the skyline in the stages, namely: *Pruning compared list* and *reducing expected comparison times*. *Pruning compared list* involves the idea of *bucket al*gorithm introduced in (Khalefa *et al*., 2008) to prune the unwanted data items from each partition before comparing crossed data items. The second stage of the proposed framework *reducing expected comparison times* computes the total sum of complete dimensions of all non-dominated data items from each bucket and sorts those data items in increasing order (smaller value considered as best). This technique minimizes the number of comparisons and eventually reduces the processing time.

Alwan *et al*. (2016; 2014) proposed a framework for handling skyline queries in an incomplete centralized database. The proposed framework consists of four components, namely: Data clustering builder, group constructor and local skylines identifier, *k*-dom skyline generator and incomplete skyline identifier. In the clustering builder, the database is evaluated to categorize the number of the dimensions with missing values. Then, the database is divided into different clusters according to the bitmap representation of the data item. To avoid losing the transitivity property the data items that are grouped in one cluster can be easily compared against each other. In group constructor and local skylines, identifier compound the clusters are grouped based on the highest value of any dimensions in the cluster and then local skylines are derived. The main function of *k*-dom generator 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, which will be inserted at the top of each cluster to prevent the dominated data items from further processing. At the end, in the incomplete skyline identifier compound, the non-dominated local skyline points are further compared to ensure that the return data items are the skyline of the entire database. The main aim of this framework is to reduce the pairwise comparisons and reduce the searching space.

The work in Lee *et al*. (2016) addressed the issues of processing skyline queries in an incomplete database. Two different algorithms have been proposed to process skyline queries in a database with incomplete data, namely: Baseline algorithm, BUCKET and Sorting-based Bucket skyline Algorithm (SOBA). BUCK*ET al*gorithm utilizes the same concept applied in *ISkyline* (Khalefa *et al*., 2008). Despite producing the correct skyline results, BUCK*ET al*gorithm takes too many unnecessary pairwise comparisons in order to determine the skylines. SOBA involves two optimized techniques: *Bucket level orders* and *point lever orders* aiming at reducing the domination tests between data items, minimize the size of skyline set and increases the efficiency of skylines processing over incomplete data. The idea of SOBA relies on using BUCK*ET al*gorithm to get local skylines from all partitions. Then, employs two optimization techniques, *bucket-level optimization* and *point-level optimization*. In *bucket level optimization*, the buckets are sorted in ascending order so that the data items compared across buckets with smaller common subspace to avoid unnecessary domination tests. While in *point-level optimization* data items of each bucket are rearranged in descending order by computing the sum of all available values of each data items. Doing so helps SOBA to be more effective in eliminating dominated data items as early as possible and to avoid the extra pairwise comparisons among data items. This work has been proposed to be used in the centralized database.

The work contributed by Alwan *et al*. (2017) focus on the issue of skyline query processing by proposing an approach for processing skyline queries in a distributed database with incomplete data. In their work, they assume that data is present in more than one relation and data is divided horizontally between relations. The proposed approach encompasses three phases, namely: Identify the skylines of each relation, joining the skylines of the relations and determining the final skylines. The first phase contains three sub-phases aiming at identifying the skylines of each involved relation. This includes (i) clustering data which intends to distribute the data items into distinct clusters based on their bitmap representation. Then, in the second sub-phase, (ii) identifying local skylines of each cluster. Lastly, in the third sub-phase, (iii) deriving *k*-dom skyline where a set of virtual data items are created by selecting the highest values of each non-missing dimension present in different clusters. These virtual data items are used to eliminate those local skylines across the clusters that not contributing in the final skylines. Doing so helps to reduce the number of data items to be processed in the next phases and in turn

decrease the number of pairwise comparisons. Then, after finding the final skylines of each relation, the skylines from all relations are further aggregated to the main relation where skyline query was submitted. Before finding the global skyline of all relations, the local skylines are merged in one relation using JOIN operation. Lastly, the global skylines are determined by carrying out the domination test between data item.

Wang *et al*. (2017) have also highlighted the issue of processing skyline queries over massive incomplete data. They proposed an approach named Skyline Preference Query (SPQ) which operate on a large volume of a database with incomplete data. The idea of SPQ is as follows. It first attempts to prioritize specific dimensions based on user's given preference. Then, based on user prioritization it divides the initial set of data items, D into two distinct cluster c_1, c_2 where c_1 is called strict cluster and c_2 is called the loose cluster. In both clusters c_1 and c_2, the data items are further classified into different smaller clusters based on their bitmap representation to avoid the issue of cyclic dominance. Then, the data items of the strict cluster are sorted in a non-ascending order based on each non-missing dimension and stores the values of the *id* dimension of the data item in an array. The number of arrays constructed to store sorted data items depends on the number of prioritized dimensions selected by the user. SPQ employs the same concept of processing data introduced in (Bharuka and Kumar, 2013). Thus, at the end of the process result set named SSRS contains the data items that are not dominated by any data items. Similarly, local skylines are obtained from the loose cluster and are saved in RSRS. Global skylines are determined either by non-empty set obtained by intersecting the SSRS and RSRS nor by comparing data items of SSRS and RSRS with each other. The data items with high probability to be part of final skyline will remain for further processing, while other data items are eliminated. High probability depends upon the less missing late. In other words, the less missing rate data items are most likely to be in the final skyline set.

Gulzar and Alwan (2017b) proposed an approach for processing skyline queries in cloud databases with partial incomplete data inspired by the work introduced in (Khalefa *et al*., 2008). The idea of the proposed strategy relies on removing the unnecessary data items through filtration before applying skyline process. Furthermore, utilizing the concept of parallel processing of data filtering by simultaneously access the involved data centres to speed up the process identifying skylines. The approach is designed to work on a vertically distributed database where data is vertically partitioned stored in many data centres at different locations and remote access is needed to transfer data from one datacentre to another. The approach consists of three stages, namely: Identifying skylines of each relation in all datacentres, joining skylines of all relations and identifying global skylines. A sorting function has been employed similar to BNL (Borzsony *et al*., 2001) and SIDS (Bharuka and Kumar, 2013) in order to sort the data items in descending order before running skyline process. In the first stage, data is read and sorted in descending order based on dimensions with non-missing values. A set of arrays is constructed containing the indices of the sorted data items based on non-missing dimensions, then, these arrays are accessed and scanned in round-robin fashion counting the occurrences of each data item in the constructed arrays. This process stops when each data item has been read at least once. Then, in the next step eliminate those data items with the count less than 2, (count < 2). Doing so helps in removing many dominated data items which are worthless in determining the skylines while keeping those high potential data items to be in the skyline set. Lastly, the remaining data items will be further processed in order to identify the local skylines of each relation. The process is carried out in parallel on all involved relations located at different data centres. Then, these local skylines are joined with their corresponding data items combining the local skylines of all relations to generate the final skylines. Finally, the combined local skylines of the relations are further processed to determine the final skylines of the database. However, the proposed approach needs a predefined sorting function that sorts data items before conducting the skyline and *join* operations. Besides, in the worst case, the skyline process may scan the entire database to find the skylines.

Lastly, the work introduced in Babanejad *et al*. (2017) highlights the issue of processing skyline queries in incomplete dynamic database. They proposed an algorithm named *DInSkyline* identifying skylines in dynamic database where data are most often change due to the continuous update operations on the database. *DInSkyline* is an intuitive approach, able to compute skylines in dynamic incomplete database without re-scanning the whole data when database is updated. The proposed solution has two steps. In the first step, compute the skylines of the initial database before any update. Then in the second step, computes skylines of those updated data items. The algorithm employs the concept of bitmap representation to divide the data items over different clusters, then applying the convention skyline techniques on each cluster to retrieve the local skylines of each cluster. During the skyline process on the initial data, a set of lists is created. This includes *Bucket Dominated* (BDD) list which keeps the dominated local skylines. The second list which named *Bucket Dominating* (BDG), attempts to save the dominating local skylines. The last list which called, *Domination History* (DH) that keeps the record of data items dominated by final skylines. When new data items are added to the database, their skylines will be derived separately without considering the existing data. Then, compare those skylines with data items in *BDG* to determine the final skylines of the updated data. Besides, for the delete operation, if deleted data items present in *BDG*, then the dominated data items of deleted skylines are taken back from *DH* list and those data items are compared with data items present in *BDG* to identify the final skylines.

Table 1: Summary of previous skyline approaches in incomplete database systems

Approach name	Technique	Computation space	Access method	Missing rate (%)	No. of dimensions	Database behavior	Context	Data set type
ISkyline (Khalefa *et al*., 2008)	Clustering	Full space	Non-index	10- 90	2-2650	Static	Centralized	Real, synthetic
RBSSQ (Arefin and Morimoto, 2012)	Clustering	Full space	Index	10-40	5	Static	Centralized	Synthetic
SIDS (Bharuka and Kumar, 2013)	Sorting	Full space	Index	10-90	2-6040	Static	Centralized	Real, synthetic
Baseline, VP, *k*ISB (Miao *et al*., 2013; Gao *et al*., 2014)	Clustering	Full space	Non-index	10-80	5-2560	Static	Centralized	Real, synthetic
Framework (Gulzar *et al*., 2016; 2017a)	Sorting	Full space	Index	< 50	3-18	Static	Centralized	Real, synthetic
ISSA (Zhang *et al*., 2016)	Sorting	Full space	Index	25-50	5-6040	Static	Centralized	Real, synthetic
Incoskyline (Alwan *et al*., 2016; 2014)	Clustering	Full space	Non-index	< 50	3-21	Static	Centralized	Real, synthetic
SOBA (Lee *et al*., 2016)	Clustering	Full space	Non-index	< 50	10-706	Static	Centralized	Real, synthetic
Jincoskyline (Alwan *et al*., 2017)	Clustering	Full space	Non-index	< 50	6-18	Static	Distributed	Real, synthetic
Model (Gulzar *et al*., 2017b)	Sorting	Full space	Index	< 50	3-17	Static	Cloud	Real, synthetic
SPQ (Wang *et al*., 2017)	Sorting	Partial	Index	<20	2-16	Static	Centralized	Real, synthetic
DInSkyline (Babanejad *et al*., 2017)	Sorting, clustering	Full space	Index	<50	3- 20	Dynamic	Centralized	Real

To the best of our knowledge, this paper has examined the most notable related works which have been published in the most well-known journals found in the major databases such as Scopus, Springer and ScienceDirect. Therefore, this survey helps interested researchers in the area of skyline queries in incomplete database in figuring out the concept of skyline queries and guide them to explore more future opportunities relevant to skyline queries in incomplete database.

Table 1 summarizes the previous works relevant to skyline queries in incomplete database covered in this section. The table demonstrates the approach, the database behavior, database context, missing rate, computation space, the access method, the number of dimensions, the technique used to process the data and the types of database.

Discussion and Future Research Opportunities

From the works reported throughout this paper, we observed that processing skylines queries in incomplete database are considered as one of the significant challenges of data management in database applications. A variation of skyline solutions has been proposed aiming at manipulating skyline queries on a database with partially incomplete data. Most of the existing strategies exploit the concept of bitmap representation of data items and employ some sorting functions to sort the data before applying skyline operation. These two concepts lead to enhancing the skyline process via several aspects including decrease the number of pairwise comparisons between data items, avoid scanning the whole data items and reducing the search space. Most importantly, these concepts have also solved the issue of cyclic dominance and losing the transitivity property of skyline technique. This issue of cyclic dominance and losing the transitivity property have been addressed in nearly all of the previous works. This is due to the significant impact of these problems on the skyline process and the correctness of the skyline results. Some literature works have addressed the issue of skyline queries on more complicated database applications such as distributed and cloud databases. These architectures are quite different from centralized database and data might need to migrate from one relation to another remote location. This operation might result in incurring more overhead on processing skyline queries. Another issue which has been focused on by some previous works is processing skyline queries in incomplete dynamic databases where data are under persistent change through update operations. The dynamic behavior of the data adds more challenge and in the worst case might lead to re-compute the skyline process on the whole data. This is considered impractical in a database with huge volume due to the prohibitive cost of processing the entire database. Therefore, many efficient techniques have been introduced concentrating on handling the issue of the dynamic contents of the database when processing skyline queries and avoid scanning the entire database when it is updated.

In the following, we present and discuss some interesting research challenges and future work directions related to processing skyline queries that should be explored raising the issue of partial incomplete data. In these subsections, many research opportunities are explained which can be exploited by interested researchers in the database community.

Crowd-sourcing Databases

The crowd-sourcing database has become an area of increasing interest in the recent years by database community. Three important factors that highly influence data processing in crowd-sourcing databases are quality control, cost control and the latency control (Li *et al*., 2016). In recent years, many challenges related to data management in crowd database have been raised (Li *et al*., 2016). Some works focused on managing the

quality control which indicates the quality of the results obtained from the crowd. It is argued that crowd-sourcing, in some cases, yield imprecise results or the quality of the results is imperfect. This is due to many reasons such as worker quality and imperfection of the data (duplicated data, outdated data and missing data). Thus, processing skyline queries on crowd-sourcing database with imprecise data is non-trivial task. Hence, effective skyline techniques are required to ensure high quality of the skyline results obtained from the crowd. Another factor that affects processing skyline queries in crowd-sourcing database is the cost control. The crowd does not provide free services and user will be paid for each task given to the crowd. Therefore, it is an essential task to design a non-expensive skyline method to process imperfect data that guarantee high quality of result while managing reasonable cost and time latency. This might include developing an efficient approach perform prior pruning for dominated data items which have not contribute toward skyline results. Doing so helps reduce the monetary cost and speed up the skyline process. The last factor that also influences data processing in the crowd is latency control which indicates the time needed to complete the task by the crowd workers. Deriving the answers from the crowd may need excessive amount of time. This is due to several reasons such as worker availability, number of workers, quality of the workers and worker distraction. All these factors influence the time latency to generate the results from the crowd database. However, processing skyline queries seem to be quite challenging in crowd-sourcing incomplete databases due to the three optimization objectives that should be fulfilled: result quality, monetary cost and latency. Hence, we suggest that many research opportunities need to be discovered to investigate the impact of the incompleteness of the data, which ultimately impact the accuracy of the skyline results (Li *et al*., 2016). In addition, any solution design to process skyline queries on crowd-sourcing databases should give high consideration to other optimization factors, namely: cost and latency. Another important issue that should also be examined when processing skyline queries with incomplete data is values estimation. Since accomplishing user task by the crowd is an expensive process, therefore, we should exploit the available data in estimating the missing values instead of requesting them from the crowd. This will help reduce the amount of data items to be estimated by the crowd and in turns, reduce the monetary cost and the time latency to generate the predicted values. It is uneasy task to utilize the data available in the database ensuring minimum relative error between the estimated value and the real missing values. This is rather challenging for a database with independent data and high missing rate.

Dynamic Incomplete Databases

It is argued that processing skyline queries in dynamic incomplete database have not receive much attention by database researchers. Several techniques have been proposed in the literature focusing on the issue of the dynamic contents of the database and how missing values impact the skyline process. There are many contemporary examples where dynamic incomplete data can be found. This includes stock market where stocks are keep changing most frequently during the day. Temporal and spatial databases are typical examples for a database with dynamic incomplete data. Many research opportunities could be identified aiming at reducing the exhaustive pairwise comparisons when data items change with update operations such as insert, delete, or update. Frequent changes in values of data items might incur additional cost if skyline process is reapplied on the entire database. Issues such as cyclic dominance and transitivity property should also take into account when designing skyline strategies fit with dynamic incomplete databases.

Mobile Crowd-Sourcing Databases

In the era of ubiquitous computing, mobile phones become an indispensable item for individuals in the modern societies where everything is done over the Internet and almost every device is connected to the Internet and most of the data is coming from various resources. This dramatic development in using mobile devices pose new challenges for crowd-sourcing data management. Issues related to the limited power supply and capacity of the mobile devices raise new challenges for crowd-sourced data management. These factors have direct impact on processing user queries such as skyline queries and the quality and the latency in accomplishing the process might be deteriorated. Other factors such as spatial distance, might also influence the quality and the time latency due to the long distance that needs to be traversed to deliver the results (Li *et al*., 2016). Most importantly, it is also very challenging to control the cost of performing skyline queries crowd-sourcing mobile platforms. There is a wide range of research opportunities that could be exploited on mobile crowd-sourcing databases related to processing skyline queries. The focus should be given on issues relevant to the dynamic and incomplete contents of the database. In these contexts, imprecise and frequently change data are very common and it might not be easy to access remote data whenever needed. Therefore, effective solutions are needed in tackling these issues and to make sure optimization factors (cost, quality and time) are not compromised due to the incompleteness and the dynamic behavior of the databases. Caching data and value prediction should also be utilized when processing skyline queries as an alternative solution when data are inaccessible or the cost and time latency of fetching data

is very high. Nevertheless, value estimation for skylines with missing data is not an easy task on the mobile crowd-sourcing platform due to high demands of system resources. In the worst case the relative error might be unacceptable and lead to introduce inaccurate results that mislead the user. It is rather challenging to design an efficient estimation technique that complies with the issues relevant to the mobile crowd-sourcing database.

Big Data

Most recently big data become a very hot research topic and has received formidable attention by lots of researchers in the database society. This focus comes due to the dramatic increase in the data volume in many contemporary database applications. New challenges related to query processing have been encountered with the rapid increase and huge amount of data. Therefore, applying existing techniques to process user queries might be impractical due to the prohibitive cost and the time complexity of such techniques. In this regard, applying traditional skyline techniques on big data is an expensive and impractical solution for many reasons. This is because big data has a very large volume of data and running skyline technique needs exhaustive search on the entire database. Therefore, introducing new skyline strategies that best fit with the unique characteristics of big data is deemed to be important. Such strategies should take into consideration avoiding scanning the whole database to determine the skylines and reduce the domination test process when deriving the skylines. Hence, some effective solutions have been suggested relying on generating a small proportion of data, then applying the query on the sample of data. A representative result will be produced and return to the user. However, such approaches should be further verified ensuring the correctness and the completeness of the representative results and accurately reflect the results of the entire database. In this regard, issues related to data imperfection and uncertainty should also be investigated. The issue of inaccurate and uncertain data adds new challenges when processing skyline queries on big data. Thus, it is also important to continue investigating and attempt to develop skyline techniques that work on big data with imprecise and uncertain data.

Conclusion

In this paper, we presented and examined previous works related to processing skylines queries in incomplete databases. The focus is given on examining the strengths and the weaknesses of the solutions designed for skyline queries on incomplete database. Issues such as cyclic dominance and losing transitivity property have been addressed. We also spot the main challenges faced due to the incompleteness of data when working with skyline queries. We conclude that type of database architecture such as distributed or cloud add rather challenges besides the imperfection of data. Current research issues and challenges have been discussed deliberately including dynamic, temporal and spatial databases when values are updated more frequently. This frequent change might result in losing the existing values and replaced with missing value. Some future research directions have been drawn including processing skyline queries in crowd-sourcing databases, dynamic databases, mobile crowd-sourcing databases and big data. Crowd-sourcing and big data have become a demanding and rich area of research in the recent years and many issues have been addressed and discussed. In all of these areas, the issues of incomplete data have been examined in depth. Many potential ideas have been discussed to resolve the issue of missing values in the skylines which let skylines to be incomplete and in most cases, misguide the user. Therefore, value estimation issues have been discussed to handle the missing values in the retrieved data items by estimating the values which change the skylines from incomplete to be fully complete.

Acknowledgment

This research is supported by the project FRGS15-205-0491, Ministry of Education, Malaysia

Author's Contributions

Yonis Gulzar: Design and preparation of the manuscript.

Ali A. Alwan: Design, preparation and review the contents of the manuscript.

Norsaremah Salleh: Development and review of the manuscript.

Imad Fakhri Al Shaikhli: Development of the manuscript.

Ethics

This article is the original contribution of the authors and is not published elsewhere. There is no ethical issue involved in this article.

References

Alwan, A.A., H. Ibrahim and N.I. Udzir, 2014. A framework for identifying skylines over incomplete data. Proceedings of the 3rd International Conference on Advanced Computer Science Applications and Technologies (SAT' 14).

Alwan, A.A., H. Ibrahim, N.I. Udzir and F. Sidi, 2017. Processing skyline queries in incomplete distributed databases. J. Intelligent Inform. Syst., 48: 399-420. DOI: 10.1007/s10844-016-0419-2

Alwan, A.A., H. Ibrahim, N.I. Udzir and F. Sidi., 2016. An efficient approach for processing skyline queries in incomplete multidimensional database. Arabian J. Sci. Eng., 41: 2927-2943.

Arefin, M.S. and Y. Morimoto, 2012. Skyline sets queries for incomplete data. Int. J. Comput. Sci. Inform. Technol., 4: 67-80.

Babanejad, G., H. Ibrahim, N.Z. Udzir, F. Sidi and A.A. Alwan, 2017. Deriving skyline points over dynamic and incomplete databases. Proceedings of the 6th International Conference of Computing and Informatics, Apri. 25-27, Sintok, pp: 77-83.

Bartolini, I., P. Ciaccia and M. Patella, 2006. SaLSa: Computing the skyline without scanning the whole sky. Proceedings of the 15th ACM International Conference on Information and Knowledge Management, Nov. 06-11, Arlington, Virginia, pp: 405-414. DOI: 10.1145/1183614.1183674

Bentley, J.L., H.T. Kung, M. Schkolnick and C.D. Thompson, 1978. On the average number of maxima in a set of vectors and applications. J. ACM, 25: 536-543. DOI: 10.1145/322092.322095

Bharuka, R. and P.S. Kumar, 2013. Finding skylines for incomplete data. Proceedings of the Twenty-Fourth Australasian Database Conference, Jan. 29-Feb. 01, Adelaide, pp: 109-117.

Borzsony, S., D. Kossmann and K. Stocker, 2001. The skyline operator. Proceedings of the 17th International Conference on Data Engineering, Apr. 2-6, IEEE Xplore Press, Heidelberg, Germany, pp: 421-430. DOI: 10.1109/ICDE.2001.914855

Chan, C.Y., H.V. Jagadish, K.L. Tan, A.K.H. Tung and Z. Zhang, 2006a. Finding k-dominant skylines in high dimensional space. Proceedings of the ACM SIGMOD International Conference on Management of Data, Jun. 27-29, Chicago, pp: 503-514. DOI: 10.1145/1142473.1142530

Chan, C.Y., H.V. Jagadish, K.L. Tan, A.K.H. Tung and Z. Zhang, 2006b. On high dimensional skylines. Proceedings of the International Conference on Extending Database Technology, (EDT'06), Springer, pp: 478-495. DOI: 10.1007/11687238_3

Chomicki, J., P. Godfrey, J. Gryz and D. Liang, 2003. Skyline with presorting. Proceedings of the 19th International Conference on Data Engineering, Mar. 5-8, IEEE Xplore Press, Bangalore, India. DOI: 10.1109/ICDE.2003.1260846

Chomicki, J., P. Godfrey, J. Gryz and D. Liang, 2005. Skyline with presorting: Theory and optimizations. Proceedings of the Intelligent Information Processing and Web Mining, (PWM' 05), Springer, pp: 595-604.

De Matteis, T., S. Di Girolamo and G. Mencagli, 2016. Continuous skyline queries on multicore architectures. Concurrency Computation, 28: 3503-3522.

Gao, Y., X. Miao, H. Cui, G. Chen and Q. Li, 2014. Processing k-skyband, constrained skyline and group-by skyline queries on incomplete data. Expert Syst. Applic., 41: 4959-4974. DOI: 10.1016/j.eswa.2014.02.033

Godfrey, P., R. Shipley and J. Gryz, 2005. Maximal vector computation in large data sets. Proceedings of the 31st International Conference on Very Large Data Bases, Aug. 30-Sept. 02, Trondheim, Norway, pp: 229-240.

Gulzar, Y., A.A. Alwan, N. Salleh and I.F. Al Shaikhli, 2017. a model for skyline query processing in a partially complete database. Adv. Sci. Lett.

Gulzar, Y., A.A. Alwan, N. Salleh and I.F. Al Shaikhli, 2017b. Skyline query processing for incomplete data in cloud environment. Proceedings of the 6th International Conference on Computing and Informatics, (ICOCI' 17), Kuala Lumpur, Malaysia.

Gulzar, Y., A.A. Alwana, N. Salleh, I.F. AlShaikhli and S.I.M. Alvi, 2016. A framework for evaluating skyline queries over incomplete data. Procedia Comput. Sci., 94: 191-198. DOI: 10.1016/j.procs.2016.08.030

Huang, Z. and W. Wang, 2006. A novel incremental maintenance algorithm of skycube. Proceedings of the Conference on Database and Expert Systems Applications, (ESA' 06), Springer, pp: 781-790. DOI: 10.1007/11827405_7

Khalefa, M.E., M.F. Mokbel and J.J. Levandoski, 2008. Skyline query processing for incomplete data. Proceedings of the IEEE 24th International Conference on Data Engineering, Apri. 7-12, IEEE Xplore Press, Cancun, Mexico. DOI: 10.1109/ICDE.2008.4497464

Kontaki, M., A.N. Papadopoulos and Y. Manolopou, 2008. Continuous top-k dominating queries in subspaces. Proceedings of the Panhellenic Conference on Informatics, Aug. 28-30, IEEE Xplore Press, Samos, Greece. DOI: 10.1109/PCI.2008.45

Kontaki, M., A.N. Papadopoulos and Y. Manolopoulos, 2010. Continuous processing of preference queries in data streams. Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science, (PCS' 10), Springer, pp: 47-60.

Kossmann, D., F. Ramsak and S. Rost, 2002. Chapter 25 – Shooting stars in the sky: An online algorithm for skyline queries. Proceedings of the 28th International Conference on Very Large Data Bases, Aug. 20-23, IEEE Xplore Press, Hong Kong, pp: 275-286. DOI: 10.1016/B978-155860869-6/50032-9

Kukhun, D.A., B. Soukkarieh, E. Lopez-Ornelas and F. Sedes, 2008. LA-GPS: A location-aware geographical pervasive system. Proceedings of the IEEE 24th International Conference on Data Engineering Workshop, Apri. 7-12, IEEE Xplore Press, Cancun, Mexico. DOI: 10.1109/ICDEW.2008.4498308

Lee, J., G.W. You and S.W. Hwang, 2009. Personalized top-k skyline queries in high-dimensional space. Inform. Syst., 34: 45-61. DOI: 10.1016/j.is.2008.04.004

Lee, J., H. Im and G.W. You, 2016. Optimizing skyline queries over incomplete data. Inform. Sci., 361: 14-28. DOI: 10.1016/j.ins.2016.04.048

Li, G., J. Wang, Y. Zheng and M.J. Franklin, 2016. Crowdsourced data management: A survey. IEEE Trans. Knowledge Data Eng., 28: 2296-2319. DOI: 10.1109/TKDE.2016.2535242

Miao, X., Y. Gao, L. Chen, G. Chen and Q. Li *et al*., 2013. On efficient k-skyband query processing over incomplete data. Proceedings of the International Conference on Database Systems for Advanced Applications, (SAA' 13), Springer, pp: 424-439.

Mokbel, M.F. and J.J. Levandoski, 2009. Toward context and preference-aware location-based services. Proceedings of the Eighth ACM International Workshop on Data Engineering for Wireless and Mobile Access, Jun. 29-29, Providence, Rhode Island, pp: 25-32. DOI: 10.1145/1594139.1594150

Papadias, D., Y. Tao, G. Fu and B. Seeger, 2003. An optimal and progressive algorithm for skyline queries. Proceedings of the ACM SIGMOD International Conference on Management of Data, Jun. 09-12, San Diego, pp: 467-478. DOI: 10.1145/872757.872814

Pei, J., W. Jin, M. Ester and Y. Tao, 2005. Catching the best views of skyline: A semantic approach based on decisive subspaces. Proceedings of the 31st International Conference on Very Large Data Bases, Aug. 30-Sept. 02, Trondheim, pp: 253-264.

Tan, K.L., P.K. Eng and B.C. Ooi, 2001. Efficient progressive skyline computation. Proceedings of the Conference on Very Large Data Base, Sept. 11-14, San Francisco, pp: 301-310.

Wang, Y., Z. Shi, J. Wang, L. Sun and B. Song, 2017. Skyline preference query based on massive and incomplete dataset. IEEE Access, 5: 3183-3192. DOI: 10.1109/ACCESS.2016.2639558

Yiu, M.L. and N. Mamoulis, 2007. Efficient processing of top-k dominating queries on multi-dimensional data. Proceedings of the 33rd International Conference on Very Large Data Bases, Sept. 23-27, Vienna, pp: 483-494.

Yiu, M.L. and N. Mamoulis, 2009. Multi-dimensional top-k dominating queries. Varge Large Data Base J., 18: 695-718. DOI: 10.1007/s00778-008-0117-y

Yuan, Y., X. Lin, Q. Liu, W. Wang and J.X. Yu *et al*., 2005. Efficient computation of the skyline cube. Proceedings of the 31st International Conference on Very Large Data Bases, Aug. 30-Sep. 02, Trondheim, pp: 241-252.

Zhang, K., H. Gao, H. Wang and J. Li, 2016. ISSA: Efficient skyline computation for incomplete data. Proceedings of the International Conference on Database Systems for Advanced Applications, (SAA' 16), Springer, pp: 321-328.