How to cite this paper:

Yonis Gulzar, Ali A. Alwan, Norsaremah Salleh, & Imad Fakhri Al-Shaikhli. (2017). Skyline query processing for incomplete data in cloud environment in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference on Computing & Informatics (pp 567-576). Sintok: School of Computing.

SKYLINE QUERY PROCESSING FOR INCOMPLETE DATA IN CLOUD ENVIRONMENT

Yonis Gulzar[1], Ali A. Alwan[2], Norsaremah Salleh[3], Imad Fakhri Al-Shaikhli[4]

[1]International Islamic University Malaysia, Malaysia, yonis.gulzar@live.iium.edu.my
[2]International Islamic University Malaysia, Malaysia, aliamer@iium.edu.my
[3]International Islamic University Malaysia, Malaysia, Norsaremah@iium.edu.my
[4]International Islamic University Malaysia, Malaysia, imadf@iium.edu.my

ABSTRACT Many research works have been conducted focusing on processing skyline queries on databases. Recently, some approaches have been proposed to address the issue of skyline queries for a partially complete database in which data item values might not be presented (missing). However, these approaches are tailored for centralized database and accessed only one table to identify the skylines. Nevertheless, in many contemporary database applications, this is might not be the case, particularly for a database with incomplete data and many tables spread over various remote locations such as cloud environment. Applying skyline approaches designed for centralized database directly on cloud databases is undesirable due to the prohibitive cost of transferring the amount of data from one datacenter to another during skyline process. An approach is needed taking into consideration the unique features of cloud environment when processing skyline queries on a database with incomplete data. This paper proposes an approach that evaluates skyline queries in a database with partially incomplete data over the cloud. The approach aims at reducing the number of pairwise comparisons that needs to be conducted between data items and the amount of data transferred in identifying skylines. Several experiments over synthetic and real datasets have been conducted to evaluate the performance of our approach. The result shows that our approach outperforms the previous approach in terms of a number of pairwise comparisons and amount of data transferred.

Keywords: skyline queries; incomplete data; preference queries; query processing; cloud computing; distributed database.

INTRODUCTION

Skyline queries have been a formidable area of research in database community for many years. Skyline set comprises of a set of non-dominated data items which called (skylines) in a given database. Given two data items *p* and *q*, it can be said that *p* dominates *q* if and only if *p* is better than *q* in all dimensions and *p* is not worse than *q* in at least one dimension. For example, assume a passenger is looking for a flight to travel between two cities on a particular day which is the cheapest in price and the shortest in time duration. Among the list of fights available, skyline queries would return only those flights that are the cheapest in price and the shortest in time duration.

Many approaches based on skyline concept have been proposed in the literature concentrating on different types of databases such as complete and incomplete. For the complete database, the focus is given on shrinking the searching space and reducing the number of pairwise comparisons among the data items. In contrast, for incomplete databases, new challenges are introduced in processing skyline queries which the issue of *cyclic dominance* and *losing the transitivity property* of skyline technique. Therefore, applying skyline approach designed for complete data is impractical and can incur high cost due to the exhaustive pairwise comparisons between data items (Ali A Alwan, Hamidah Ibrahim, Nur Izura Udzir, & Fatima Sidi, 2016a; Bharuka & Kumar, 2013; Gulzar, Alwan, Salleh, Al Shaikhli, & Alvi, 2016; Khalefa et al., 2008). Hence, several attempts have been introduced resolving the issue of incomplete data when processing skyline queries (Ali A Alwan, Hamidah Ibrahim, Nur Izura Udzir, & Fatima Sidi, 2016a; Bharuka & Kumar, 2013; Gulzar, Alwan, Salleh, Al Shaikhli, & Alvi, 2016; Khalefa et al., 2008). However, with the exception of (Alwan et al., 2016a; Bharuka & Kumar, 2013; Gulzar et al., 2016; Khalefa et al., 2008), all of these attempts assumed that the database is centralized and only one table needs to be accessed to identify the skylines.

Combing data items before applying skyline technique on cloud databases result into transferring huge amount of data from datacenters. This approach is extremely undesirable as it leads to a prohibitive cost due to transferring a large amount of data which incur high processing cost. It also leads to a large number of unnecessary pairwise comparisons between data items. Less attention has been paid for databases with partially incomplete data resides in cloud environment where different databases containing related information located at different sites. Due to the large volume of data, the cloud provides a platform for users to store data over the cloud and these data can be remotely accessed at any time from everywhere. Processing skyline queries for a database with incomplete data in cloud context might not be as easy as in centralized context.

In this paper, an approach for processing skyline queries for a database with incomplete data in cloud environment is proposed. In this context, database tables are spread over different sites and remote access needs to be conducted during skyline process. In this paper, we assume that database tables are divided horizontally and stored in different sites. Our approach consists of three components, namely: (i) identifying the skylines of each relation in all datacenters, (ii) joining Skylines of all relations, and (iii) identifying Global Skylines. In the first phase, the process of the skyline is performed on each of the relation that is involved in the query to eliminate the dominated data items that are guaranteed not skylines.

The rest of the paper is structured as follows. The previous works related to this research are presented in the next section. The basic definitions and notations, which are used throughout the paper and the detail of the proposed approach, are explained in sections followed. Experiment results are reported in next section. The conclusion is outlined in the last section.

RELATED WORK

There is a tremendous research effort have been devoted highlighting the issue of skyline queries in database systems. In the following, the relevant works in both complete and incomplete databases are presented and discussed. The first work that introduced skyline queries into database community is contributed by (Borzsony et al., 2001). Two algorithms have been introduced by (Borzsony et al., 2001), namely, BNL and D&C to process skyline queries in the complete database. Later, many researchers have proposed many algorithms to process skyline queries in a complete database, this includes LESS (Godfrey, Shipley, & Gryz, 2005), BSS (Papadias, Tao, Fu, & Seeger, 2003), SkyTree (Lee & Hwang, 2014). All of those algorithms benefit from the concepts of BNL and D&C techniques by either applying the concept of partition proposed in D&C or applying the sorting technique to improve BNL approach. These techniques are designed to process skyline queries in the centralized complete database.

In contrast, other works proposed algorithms to identify skylines in an incomplete database. This includes, BUCKET and Iskyline (Khalefa et al., 2008), RSSSQ (Arefin & Morimoto, 2012), Baseline, VP, *k*ISB (Miao et al., 2013), (Alwan et al., 2016a) proposed a framework that inspired by the work proposed in (Khalefa et al., 2008). Likewise (Gulzar et al., 2016) proposed an approach that is inspired by the work in (Bharuka & Kumar, 2013). However, these works assumed that the database is centralized and data items are saved in one table. Furthermore, there has been several skyline approaches proposed for distributed complete databases such as SFSJ (Vlachou, Doulkeridis, & Polyzotis, 2011), Iterative (Sun, Wu, Li, & Tung, 2008), SKYJOG (Zhang, 2016). These approaches assume that data are partitioned vertically or horizontally and presented in more than one table. To evaluate skyline queries, join operator needs to be involved combining the data items of the tables before applying skyline technique. However, it is impractical to directly apply these approaches on incomplete distributed databases due to the issue of *cyclic dominance* and *losing* transitivity property of skyline technique.

To the best of our knowledge, the latest work that raised the issue of processing the skyline queries in incomplete distributed databases is contributed by (Alwan, Ibrahim, Udzir, & Sidi, 2016). But, this work is limited to only the case of database tables are vertical partitioning where the attributes (dimensions) of the table are located on different sites and remote access needs to be performed during skyline process. Besides, the architecture of cloud is quite different from distributed environment.

PRELIMINARIES

This section gives some necessary definitions and annotations that are related to skylines queries in a cloud computing database with incomplete data. These definitions and notations are important to explain the details of our proposed approach. Our approach has been developed in the context of relational databases with partially incomplete data, *D*. A relation of the database *D* is denoted by *R* (*d*1, *d*2, ..., *dm*) where *R* is the name of the relation with *m*-arity and *d* = (*d*1, *d*2, ..., *dm*) is the set of dimensions.

DEFINITION 1 INCOMPLETE DATABASE: given a database *D* (*R*1, *R*2, ..., *Rn*), where *Ri* is a relation denoted by *Ri* (*d*1, *d*2, ..., *dm*), *D* is said to be *incomplete* if and only if it contains at least a data item *pj* with missing values in one or more dimensions *dk* (attributes); otherwise, it is *complete.*

DEFINITION 2 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 $pi \succ pj$) if and only if the following condition holds: $\forall\, d_k \in d, p_i.d_k \geq p_j.d_k \wedge \exists d_l, \in d, p_i.d_l > p_j.d_l$.

DEFINITION 3 SKYLINE QUERIES*:* Select a data item *pi* from the set of *D* database if and only if *pi* is as good as *pj* (where $i \neq j$) in all dimensions (attributes) and *strictly* better than p_j *in* at least one dimension (attribute). We use *Sskyline* to denote the set of skyline data items, $Sskyline = (pi\ \forall\ pi, pj\ \in D, pi\ \succ\ pj)$.

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

THE PROPOSED APPROACH

This section explains the detail steps of the proposed approach for processing skyline queries. The approach focuses on processing skyline queries with the aim of reducing the number of pairwise comparisons and the amount of data transferred during skyline evaluation. For this purpose, we attempt to ensure that many dominated data items reside in different datacenters are eliminated from further processing before applying skyline technique. This will help to avoid many unnecessary pairwise comparisons between data items while preserve the transitivity property and avoids the issue cyclic dominance. The proposed approach of processing skyline queries in a database with partially incomplete data in the cloud is elaborated in Figure 1. The steps of determining the skylines of each relation include sorting data and construing array, filtering data, identifying local skylines, retrieving local skylines, combining local skylines of databases and identifying global skylines. Combining the skylines of each relation is performed to identify the candidate skylines. Finally, further comparisons are performed on the combined data items to derive the final global skylines. These phases are explained in details as follow.

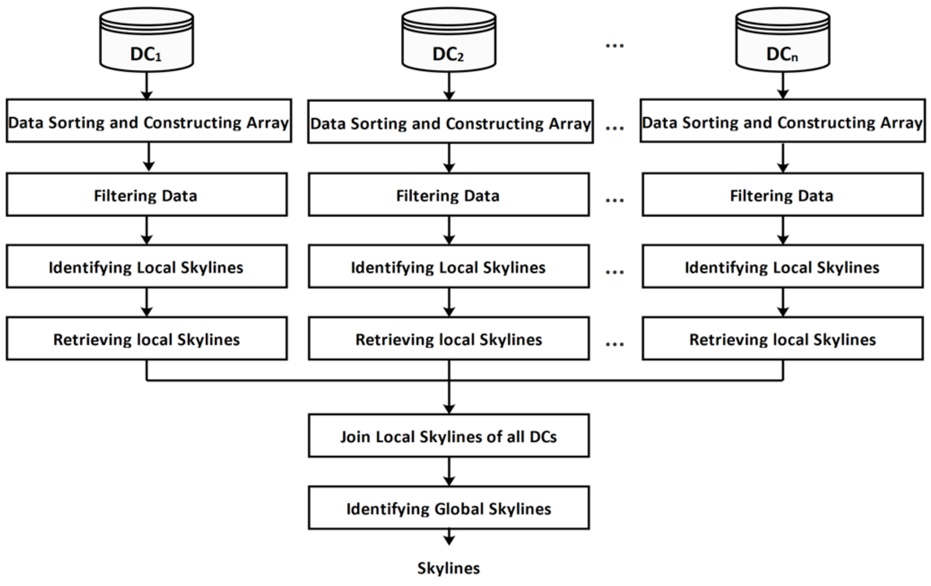

DC1
DC2
DCn
Data Sorting and Constructing Array
Filtering Data
Identifying Local Skylines
Retrieving local Skylines
Join Local Skylines of all DCs
Identifying Global Skylines
Skylines

Figure 1. The Proposed Approach

i. Identifying the skylines of each relation in all datacenters

This phase attempts to separately identify the skylines of each relation located in all involved datacenters aiming at discarding all dominated data items from the join operation. Doing so results into propagating only the most candidate data items into the next phases. This helps to avoid joining many dominated data items by performing data filtration which further lead to eliminating the unnecessary pairwise comparisons between data items and reduce the amount of data transferred significantly. The details process of this phase is elaborated in the following subsections.

Sorting Data and Constructing Array

This step is responsible to analyze the initial incomplete data stored in the table and attempts to sort the data items based on the existing values belong to each dimension in non-ascending order. Then a set of arrays is constructed storing the value of the dimension d_i of each sorted data items. Hence, the number of arrays constructed mainly depends on the number of dimensions with no-missing value. This step helps in shrinking the searching space which further leads to reduce the number of pairwise comparisons between data items in the next phases.

Filtering Data

This phase is one of the most significant phases in introducing the local skylines of each involved table. This phase is responsible for eliminating the dominant data items before applying skyline technique. This is achieved by scanning the whole data items in each array in sequential order using round robin fashion. The scanning process ends when all data items have been read at least once. It might happen that some data items are read more than once. Therefore, a counter is needed to count the number of read of each data item. The idea behind using the counter is to sort the data items according to their count values in decreasing order. Hence, the data items with the highest count score have a higher potential to be in the skylines set. Besides, it also helps in eliminating a large number of dominated data items. The outputs of this process are a list of data items with their corresponding count values.

Identifying Local Skyline

The main aims of this step are to identify the non-dominated data items with a high potential to be in the skyline set. While excluding data items that have no contribution to be in the final skyline set. Therefore, this leads to eliminate the dominated data items that have count less than two from the join process and to prevent the dominated data items from being transferred from one site to another during the skyline evaluation process.

Retrieving Local Skylines

This stage applies the skyline technique over the non-dominated data items to identify the skylines of each relation which named local skylines. This process is conducted in parallel on all participated relations separately spread over many datacenters to limit the number of data items to be joined later. Since the eliminated data items are not the skylines of the relation, thus they are also not the skylines once joined with the data items of the other relation(s). Thus, we can safely conclude that the eliminated data items would not be part of the final skylines. This step assists in reducing the number of pairwise comparisons that needs to be performed in retrieving skylines. Moreover, the amount of data items to be transferred from one datacenter to another to evaluate the final skylines also decreased. The process starts by reading the first data item in the candidate set and then compared with the remaining data items. The read data item named *processing data item*, *p*, while the data item which is to be compared with *p* called *candidate data item q*. During the comparison process if *p* dominates *q* then *q* will be immediately eliminated from the candidate set. Else if neither *p* dominates *q* and nor *q* dominates *p*, then *q* will remain in the candidate set for further processing. However, if *q* dominates *p* then *p* will only not be removed immediately; rather it will remain until the end of the iteration process. This is because *p* may have good potential to eliminate other data items and helps to sustain transitivity property and solves the issue of cyclic dominance. This process continues until all remaining data items are processed. It should be noted that no two data items are compared more than once. We argue that this process results in avoiding many unnecessary pairwise comparisons between data items. The output of this step is the set of the local skylines of each sub-relation to be joined to form the final skylines.

ii. Joining Skylines of all Relations

This phase intent to combine the identified local skylines of the involved relations together in one relation which is resides in one datacenter. It should be noted that the output of this phase is a set of data items with the high potential to be in the skyline set. Thus, many unnecessary pairwise comparisons have been avoided and only limited number of data items has been propagated into the next phase.

iii. Identifying Global Skylines

This section explains the details step of the last phase in our proposed approach of processing skyline queries in a database with partially incomplete data in a cloud environment. It tries to determine the final skyline set which contains those data items that are not dominated by other data items in all involved relations. The sub-phases of phase A (Identifying the skylines of each relation) of our proposed approach will be performed after the join operation is performed on the joined local skylines. If the joined data item is not dominated by the other data items in the candidate skylines set, then it is retrieved as part of the final skyline. Otherwise, it is removed from the candidate skyline sets and will not be part of the final skylines. In this process, we guarantee that the final skylines are the skylines of the relations in all cloud datacentres and no other data items might dominate the identified final skylines.

EXPERIMENTAL SETTINGS

Several experiments have been conducted over different datasets to evaluate the performance of the proposed approach. We have compared our approach against the most recent work, SIDS (Sort-based Incomplete Data Skyline), contributed by (Bharuka & Kumar, 2013). Since skyline technique is a CPU intensive and needs a high number of pairwise comparisons between data items. Therefore, this work concentrates on measuring the efficiency of the proposed approach with respect to the number of pairwise comparisons and amount of data transfer between datacenters. These are considered as most influenced parameters in processing skyline queries. The number of pairwise comparisons has been counted with respect to number of dimensions, and database size. These two metrics are measured by varying the number of dimensions, the number of dimensions with missing values, and the database size. In our experiments, we assumed that the database is fragmented vertically into three tables situated on three different datacenters and user has submitted the query into datacenter 1. Two different datasets have been used in the experiment namely: synthetic (correlated) and real dataset (NBA and MovieLens). Table 1 summarizes the parameter setting for synthetic and real datasets.

Table 1. The Parameter Setting of Synthetic and Real Datasets

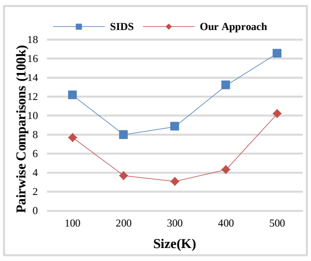
Dataset Name	Parameter Settings			
	No. of dimensions (d)	No. of dimensions with missing values (d')	Dataset Size (KB)	Data Centers
MovieLens	4	3	400, 800, 1200, 1600, 2000	3
NBA	6-18	5-17	40,80,120,160,200	3
Correlated	4-12	3-11	100, 200, 300, 400, 500	3

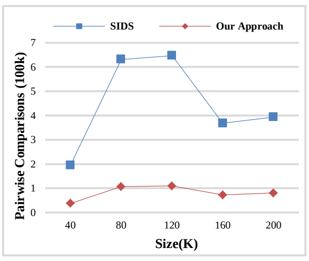
Cardinality

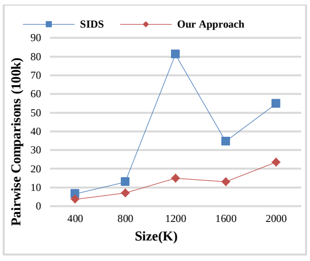
The experiment reports in this section attempt to investigate the impact of the database cardinality on processing skyline queries. In this set of the experiment the database size is variable and the number of dimensions is fixed. Figure 2 illustrates the results obtained from real and synthetic datasets. Figure 2.a shows the number of pairwise comparisons derived for correlated dataset. The number of dimensions is 8 and the size of the database is varying from 100KB to 500KB. Figure 2.b depicts the experiment results of NBA dataset. In this dataset, the number of dimensions is fixed to 18 and dataset size varies between 40KB to 200KB. Figure 2.c presents the experiment results of MovieLens dataset where database size varies from 400KB to 2000KB and the number of dimensions is 8. From the results, it is observed that our approach outperforms SIDS and database cardinality has no significant impact on the performance of our approach. This is due to applying the concept of data filtration and local skyline identifier that helps in reducing the number of pairwise comparisons.

Dimensionality

This set of experiment investigates the impact of a number of dimensions on skyline technique. We fixed the size of the dataset while varying the number of dimensions. Figure 3 depicts the results obtained from real and synthetic datasets. Figure 3.a presents the experiment results of NBA dataset where a number of dimensions are varying from 4 to 18 while dataset size is 120KB. Figure 3.b describes the experiment results of the correlated dataset where a number of dimensions are varying from 4 to 12 and dataset size is fixed to 200KB. It can be concluded that our approach introduced a lower number of pairwise comparisons and steadily outperform SIDS approach. It is also noticed that increasing number of dimensions has a reasonable impact on skyline process, which leads to a larger number of pairwise comparison in identifying the skylines.

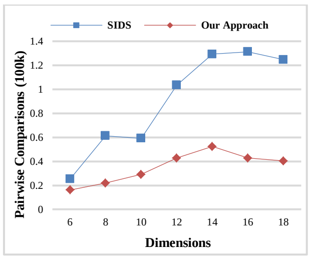


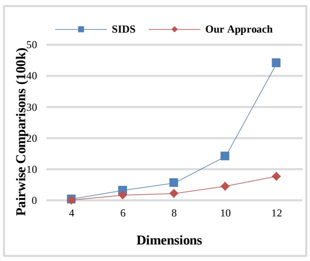




(a) Correlated (b) NBA (c) MovieLens

Figure 2: Cardinality Effect



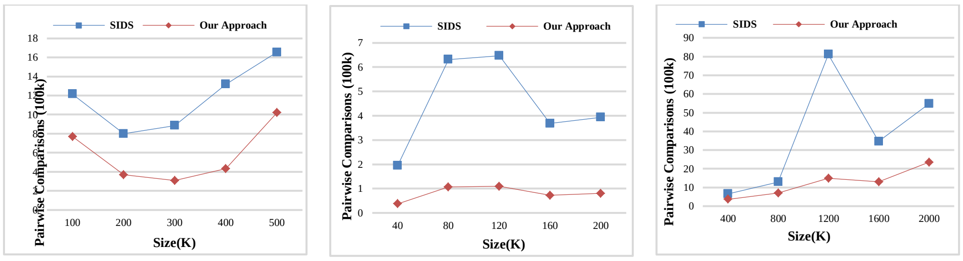


(a) NBA (b) Correlated

Figure 3: Dimensionality Effect

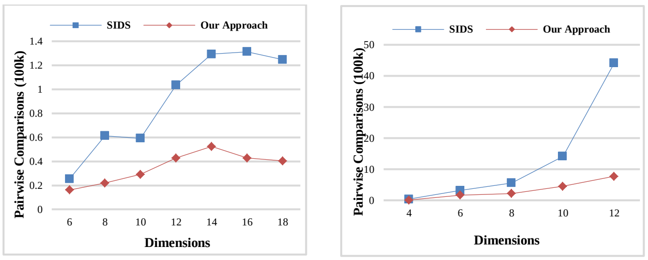
Data Transfer

This set of experiments emphasizes on examining the effect of dataset size on the amount of data transfer among datacenter. The amount of data transfer indicates the total amount of data items that needs to be transferred across the cloud datacentres to evaluate the skylines since it influences the performance of the skyline query process in a cloud environment. Figure 4a, b, and c depict the results of the proposed approach on synthetic, MovieLens, and NBA datasets, respectively. From the figures, we conclude that applying skyline technique on each datacenter separately and then transferring only local skylines to query submitted datacenter is far better than transferring all the data from each datacenter. Experiment results show that we have reasonable impact on skyline process, which leads to a larger number of pairwise comparison in identifying the skylines



(a) Correlated **(b) NBA** **(c) MovieLens**

Figure 2: Cardinality Effect

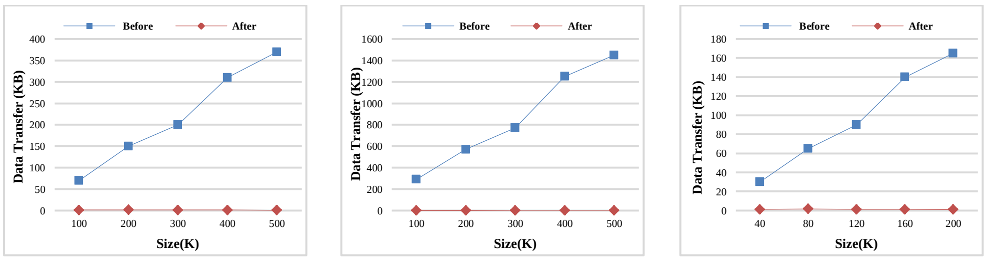


(a) NBA **(b) Correlated**

Figure 3: Dimensionality Effect

Data Transfer

This set of experiments emphasizes on examining the effect of dataset size on the amount of data transfer among datacenter. The amount of data transfer indicates the total amount of data items that needs to be transferred across the cloud datacentres to evaluate the skylines since it influences the performance of the skyline query process in a cloud environment. Figure 4a, b, and c depict the results of the proposed approach on synthetic, MovieLens, and NBA datasets, respectively. From the figures, we conclude that applying skyline technique on each datacenter separately and then transferring only local skylines to query submitted datacenter is far better than transferring all the data from each datacenter. Experiment results show that we have successfully saved 95% to 98% of data from being transferred. That, in turn, saves the network cost as well.



(a) Correlated **(b) MovieLens** **(c) NBA**

Figure 4: Amount of Data Transfer

CONCLUSION

This paper proposed an approach for processing skyline queries for partially complete database over cloud environment. We described the details of our approach and how it managed to identify the final skylines of the database. Several experiments have been conducted and the results show that our approach for handling skyline queries in incomplete cloud databases has significantly outperformed the previous approach.

ACKNOWLEDGMENTS

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

REFERENCES

Alwan, A. A., Ibrahim, H., Udzir, N. I., & Sidi, F. (2016a). An Efficient Approach for Processing Skyline Queries in Incomplete Multidimensional Database. *Arabian Journal for Science and Engineering*, 1-17.

Alwan, A. A., Ibrahim, H., Udzir, N. I., & Sidi, F. (2016b). Processing skyline queries in incomplete distributed databases. *Journal of Intelligent Information Systems*, 1-22.

Arefin, M. S., & Morimoto, Y. (2012). Skyline sets queries for incomplete data. *international Journal of Computer Science & Information Technology, 4*(5), 67-80.

Bharuka, R., & Kumar, P. S. (2013). *Finding skylines for incomplete data*. Paper presented at the Proceedings of the Twenty-Fourth Australasian Database Conference.

Borzsony, S., Kossmann, D., & Stocker, K. (2001). *The skyline operator.* Paper presented at the Proceedings of the 17th International Conference on Data Engineering.

Godfrey, P., Shipley, R., & Gryz, J. (2005). *Maximal vector computation in large data sets.* Paper presented at the Proceedings of the 31st International Conference on Very Large Data Bases.

Gulzar, Y., Alwan, A. A., Salleh, N., Al Shaikhli, I. F., & Alvi, S. I. M. (2016). A Framework for Evaluating Skyline Queries over Incomplete Data. *Procedia Computer Science, 94*, 191-198.

Khalefa, M. E., Mokbel, M. F., & Levandoski, J. J. (2008). *Skyline query processing for incomplete data.* Paper presented at the IEEE 24th International Conference on Data Engineering (ICDE).

Lee, J., & Hwang, S.-W. (2014). Scalable skyline computation using a balanced pivot selection technique. *Information Systems, 39*, 1-21.

Miao, X., Gao, Y., Chen, L., Chen, G., Li, Q., & Jiang, T. (2013). *On efficient k-skyband query processing over incomplete data.* Paper presented at the Conference on Database Systems for Advanced Applications.

Papadias, D., Tao, Y., Fu, G., & Seeger, B. (2003). *An optimal and progressive algorithm for skyline queries.* Paper presented at the Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data.

Sun, D., Wu, S., Li, J., & Tung, A. K. (2008). *Skyline-join in distributed databases.* Paper presented at the IEEE 24th International Conference on Data Engineering Workshop (ICDEW).

Vlachou, A., Doulkeridis, C., & Polyzotis, N. (2011). *Skyline query processing over joins.* Paper presented at the Proceedings of the 2011 ACM SIGMOD International Conference on Management of data.

Zhang, J. (2016). Efficient Skyline Query over Multiple Relations. *Procedia Computer Science, 80*, 2211-2215