SKYLINE QUERIES OVER INCOMPLETE MULTIDIMENSIONAL DATABASE

Ali A. Alwan[1], Hamidah Ibrahim[2], Nur Izura Udzir[3] and Fatimah Sidi[4]

[1]Universiti Putra Malaysia (UPM), Malaysia, ali83_upm@yahoo.com
[2]Universiti Putra Malaysia (UPM), Malaysia, hamidah@fsktm.upm.edu.my
[3]Universiti Putra Malaysia (UPM), Malaysia, izura@fsktm.upm.edu.my
[4]Universiti Putra Malaysia (UPM), Malaysia, fatimacd@fsktm.upm.edu.my

ABSTRACT In recent years, there has been much focus on skyline queries that incorporate and provide more flexible query operators that return data items which are dominating other data items in all attributes (dimensions). Several techniques for skyline have been proposed in the literature. Most of the existing skyline techniques aimed to find the skyline query results by supposing that the values of dimensions are always present for every data item. In this paper we aim to evaluate the skyline preference queries in which some dimension values are missing. We proposed an approach for answering preference queries in a database by utilizing the concept of skyline technique. The skyline set selected for a given query operation is then optimized so that the missing values are replaced with some approximate values that provide a skyline answer with complete data. This will significantly reduce the number of comparisons between data items. Beside that, the number of retrieved skyline data items is reduced and this guides the users to select the most appropriate data items from the several alternative complete skyline data items.

Keywords: skyline queries, flexible query operators

INTRODUCTION

The preference queries are significant and mostly used in various application domains, like multi-criteria decision making applications (Chee-Yong C., et al, 2006(a); Chee-Yong C., et al, 2006(b); Man L., Y., & Nikos M., 2007), where many criteria are involved in the query statement to select the most suitable answer that fit the user requirements. Another domain that applied the preference queries is the decision support system or recommender system, where these systems combine various interests to help users by recommend a strategic decision. Restaurant finder (Mohamed F. M., & Justin J. L., 2009) are typical example that show the importance of preference queries. Furthermore, E-commerce environment is also a significant area that involves preference queries. For example helping customer to make a trade off between the price, quality, and efficiency of the products to be purchased. Due to the importance of preference query results that obviously appeared in many database applications, a various number of preference evaluation techniques have been proposed in the literature (Stephan B., et al, 2001; Donald K., et al, 2002; Jan C., et al, 2003; Ilaria B., et al, 2006; Man L. Y. & Nikos M., 2007).

One of the popular and most frequently used of preference query type is the skyline queries. Many approaches which applied the concept of skyline technique have been proposed (Stephan B., et al, 2001; Donald K., et al, 2002; Jan C., et al, 2003; Ilaria B., et al, 2006; Man L. Y. & Nikos M., 2007). Most of the previous techniques assumed that all the values of attributes (dimensions) are present and the database state is complete (no missing values) for all data items of the database (Stephan B., et al 2001; Kian-Lee T., et al, 2001; Donald K., et al, 2002; Jan C., et al, 2003; Jian P., et al, 2005; Parke G., et al, 2005; Ilaria B., et al, 2006; Man L. Y. & Nikos M., 2007; Man L. Y. & Nikos M., 2009). However, this assumption is not always true particularly in the real world with large database and high number of dimensions (attributes) as some values may be missing. Further, the incompleteness of data introduces new challenges in the skyline queries. The missing values will influence negatively on the process of finding skyline data items leading to loss the transitivity property, cyclic dominance, and high overhead, due to exhaustive comparison between the data items to determine which data items are the skyline. Besides that, the skyline answers become insignificant in some cases as the size of skyline answers increased dramatically when the missing rate is high (Mohamed E. K., et al, 2008).

Another important issue is that the skyline data items are retrieved answers according to the current state of the database which is incomplete without imputing the missing values. In many cases users are more concern about the values in these missing dimensions, and the obtained skyline answers may not satisfy the users' demands. Therefore, an approach is needed to estimate these missing values in the skyline data items. These estimated values for the missing dimensions (attributes) in the skyline answers help in the process of identifying the skyline answers by further minimizing the size of the skyline answers. Most importantly, the estimated values help users to select the most suitable data items from several alternative skyline data items.

This paper presents an approach for retrieving skyline queries in incomplete database where some values of one or more dimensions do not exist in the database. Our approach utilizes the bitmap representation to divide the initial database into a set of distinct clusters in which every cluster stored the data items that have missing values at the same dimension(s). Thus, a significant number of unnecessary pairwise comparisons between data items would be avoided. Moreover, a set of groups is created for each cluster depending on the highest value in any of the dimensions of the data item. This process aims to reduce the number of comparisons in each cluster in retrieving the cluster skyline. To reduce the size of skyline answers and the number of comparisons between the selected cluster skyline of the whole database, a set of virtual skyline named *k-dom* is derived based on the common complete dimensions among the clusters and these virtual skylines are inserted at the top of every cluster.

Since the user may obtain a set of skyline data items which have more than one missing values for a given query operation, therefore a strategy to impute the missing values by replacing them with some plausible values in the skyline results is required. In this work, a strategy is proposed to estimate the missing values by employing the relationship between the dimensions (attributes) of the database and the current available values in the set of skyline results and later optimize these results to further reduce the size of the result by retrieving only the most relevant data items. We believe that this strategy can improve the performance of processing skyline queries in incomplete database system since prevents many unnecessary comparisons between data items, reduces the size of the skyline results, and provides complete answers to the user.

This paper is organized as follows. In the next section, the previous works related to this research are presented. Then, the basic definitions, and notations, which are used in the rest of the paper, are set out. The following section describes our proposed approach with some examples to clarify the approach. Conclusions and further research are presented in the final section.

RELATED WORKS

Many types and variations of skyline preference evaluation techniques of preference queries have been described in the database literature. Most of these skyline techniques aim to improve the search performance by terminating the process of searching the data items as early as possible in obtaining the "best" answer that satisfies the conditions as indicated in the submitted query. In the following we present the most important types of preference queries that utilize skyline techniques in complete and incomplete databases.

The first work of skyline preference evaluation technique of query processing in the database field is proposed by Stephan B., et al. (2001). They have proposed two algorithms, namely: BNL (Block-Nested-Loop) and DC (Divide-and-Conquer). BNL produced the skyline data items by repeatedly read the set of data items, and when a data item *p* is read from the input; *p* is compared to the other items in the dataset. The second algorithm, DC, divides the dataset into two equivalent sets. Then, it finds the skyline sets and combines the output of the two skyline sets to further perform comparison to eliminate those data points which are dominated by the skyline data items.

Kian-Lee T., et al. (2001) presents two incremental algorithms (Bitmap and Index) to produce the skyline data items of the database. Donald K., et al. (2002) proposed an online full space skyline computation technique NN (Nearest Neighbor) to collect the skyline data items. Jan C., et al. (2003) proposed SFS (Sort-Filter-Skyline). SFS employed the concept of presorting on BNL Skyline technique (Stephan B., et al., 2001) in order to produce the skyline query data items in an efficient manner and well behaved in a relational setting.

Dimitris P., et al. (2003) proposed BBS (Branch-Bound-Skyline) method to find the set of skyline data items. BBS is aim to optimal the I/O costs. Parke G., et al. (2005) proposed an external algorithm which is called LESS (Linear Elimination Sort for Skyline) which is inspired from SFS (Jan C., et al., 2003). LESS works on non-indexed data and does not require any additional preprocessing steps. Furthermore, Ilaria B., et al. (2006) presents an algorithm named SaLSa (Sort and Limit Skyline algorithm) by exploiting the concept of SFS method to pre-sort the data points first and then selects a subset of dataset to examine the skyline data items.

However, in incomplete database systems, query processing is challenging as in many cases a significant part of query answer may be neglected from the final answer due to the missing values in some dimensions (attributes). In addition, preference queries have not received much attention in incomplete database applications in which to evaluate the query, exhausted comparison needs to be performed in order to determine the best data items in the database that meet the query conditions. Preference queries in incomplete database are fundamentally different than the conventional preference queries in complete database because the transitivity property of preference techniques is no longer hold.

To the best of our knowledge the only work that tackled the issue of skyline queries in incomplete database is contributed by Mohamed E. K., et al., (2008). Mohamed E. K., et al., (2008) proposed *Iskyline* algorithm that handles the skyline queries in incomplete relational database by dividing the initial database into distinct nodes depending on the missing dimensions and then applying the conventional skyline technique to retrieve the local skyline in every cluster. *Iskyline* method conducts two optimization techniques that reduce the number of local skyline in every cluster. However, *Iskyline* is time consuming as in each node there are many pairwise comparison need to be performed to find the local skyline. Most importantly, large amount of missing data in the skyline results does not give any insight to help user in selecting the most appropriate data item.

PRELIMINARIES

Our approach has been developed in the context of relational databases. A database consists of a finite set of relations (R_1, R_2, …, R_m). A relation is denoted by $R(A_1, A_2, \ldots, A_n)$ where R is the name of the relation with n-arity and A_i's are the attributes (dimensions) of R. We denote the missing value in the attribute A by a (*) to indicate the incomplete attributes. For example the data item a(4, *, 7) indicates that the first and third attributes have values 4, 7, respectively. While the second attribute is unknown (*). To divide the initial database into a set of comparable data items we employed the principle of *bitmap representation* that collects the whole data items which have missing values in the same attribute into the same cluster. For example data item *a*(4, *, 6) and *b*(*, 4, 3) are represented by the bitmaps *P*.*a*=101 and *P*.*b*= 011, respectively. Our approach is applicable for the complete and incomplete databases. Due to space limitation, only incomplete database is used in the examples.

OUR PROPOSED METHOD

Our technique produces the set of skyline queries that are not dominated by any other data items in the whole database. Splitting the data items into set of clusters would avoid the problem of cyclic dominance by neglecting the incomplete dimensions (attributes). To further reduce the number of comparisons between data items in each cluster and save a significant amount of processing time, each cluster is further divided into set of groups. The groups are created recursively depending on the highest value in any of the dimensions of that data item. If the highest value in the remaining data items of a cluster is less than or equal to the lowest value of the created group, then, we stop the process of creating groups as no other data items may be considered as a skyline. For example data items *a*(*, 4, 3, 8) and *b*(*, 3, 8, 2) are collected in the same group as the highest value in both data items is 8. The aim of this step is to avoid an exhaustive comparison between data items and eliminate the problem of cyclic dominance. Figure 1 shows three different clusters that are created based on our example database. *d*1, *d*2 and *d*3 consist of the dimensions (attributes) of a data item in the database.

A*i*	*d1*	*d2*	*d3*
A1	*	5	3
A2	*	1	1
A3	*	3	3
A4	*	6	6
A5	*	2	6
A6	*	4	5
A7	*	3	2
A8	*	5	5
A9	*	6	4
A10	*	2	1
A11	*	2	2
A12	*	3	5

Cluster 1 = 011

B*i*	*d1*	*d2*	*d3*
B1	4	*	5
B2	3	*	2
B3	7	*	6
B4	5	*	3
B5	2	*	1
B6	2	*	2
B7	7	*	4
B8	5	*	5
B9	1	*	3
B10	7	*	5
B11	3	*	3
B12	3	*	5

Cluster 2 = 101

C*i*	*d1*	*d2*	*d3*
C1	2	3	*
C2	3	6	*
C3	4	5	*
C4	1	3	*
C5	2	2	*
C6	6	3	*
C7	1	1	*
C8	5	3	*
C9	6	4	*
C10	2	1	*
C11	5	4	*
C12	3	5	*

Cluster 3 = 110

Figure 1: Clusters

Now, we create a set of groups for each cluster by gathering the data items which have the same highest value in any of the complete attributes. Figure 2 illustrates the concept of grouping technique. The shaded data items are the local skyline of the cluster.

Group 1			
A*i*	*d1*	*d2*	*d3*
A4	*	6	5
A5	*	2	6
A9	*	6	4
Group 2			
A8	*	5	5
A6	*	4	5
A1	*	5	3
A12	*	3	5

Cluster 1

Group 1			
B*i*	*d1*	*d2*	*d3*
B3	7	*	6
B10	7	*	5
B7	7	*	4
Group 2			
B8	5	*	5
B12	3	*	5
B4	5	*	3
B1	4	*	5

Cluster 2

Group 1			
C*i*	*d1*	*d2*	*d3*
C9	6	4	*
C2	3	6	*
C6	6	3	*
Group 2			
C11	5	4	*
C3	4	5	*
C12	3	5	*
C8	5	3	*

Cluster 3

Figure 2. Groups and Local Skylines

Notice that, in Figure 2 some data items of the clusters are removed before applying the skyline technique. For example, A2, A3, A7, A10 and A11 are removed from Cluster 1 since to the highest value of these data items is less than the lowest value of the created group. Also, B2, B5, B6, B9 and B11 are removed from Cluster 2 for to the same reason. This process aims to reduce the number of data items before applying the process of pairwise comparison to find the cluster skyline data items. Next, we create a set of virtual skyline *k-dom* from the cluster skyline based on the complete dimensions between different clusters and put them at the top of every cluster. Then, we conduct a *mapping policy* among the virtual skyline data items by merging the *k-dom* data items as a single data item. The mapped *k-dom* is composed the cluster skyline. This process aims to eliminate as early as possible from the candidate cluster skyline before considering them in the final answer.

Figure 3 shows the mapped virtual skyline after deriving the virtual skyline *k-dom* and applying the mapping policy. For example *k-dom* in Cluster 1 comes from the mapping of cluster skyline of Clusters 2 and 3 by considering the highest value in each dimension. That means local skyline Cluster2 and Cluster 3 produced the *k-dom* (*, 4, 6). The mapping process will significantly reduce the number of comparisons. Instead of comparing two virtual skylines (B3 and C9) independently with the local skyline of Cluster 1, we insert this *k-dom* data item into the Cluster 1 and start the skyline process to eliminate the dominating local skyline from further processing. Notice that, the value of the first dimension in the *k-dom* is *, this is because in Cluster 1 the incomplete dimension is the first dimension and will not be used in the comparison process. While, for dimensions 2 and 3 we replaced the * symbol with the highest value of that dimension. Also notice, because A4 is better and not worse than *k-dom*, A4 will be further considered to be as a global skyline. *k-dom* removes A5 from further processing because *k-dom* is better than A5 in all the complete dimensions.

Inserting *k-dom* to local skyline of cluster 1			
k-dom	*	4	6
A4	*	6	5
A5	*	2	6

Cluster 1

Inserting *k-dom* to local skyline of cluster 2			
k-dom	6	*	5
B3	7	*	6

Cluster 2

Inserting *k-dom* to local skyline of cluster 3			
k-dom	7	6	*
C9	6	4	*
C2	3	6	*

Cluster 3

Figure 3. Effect of *k-dom* on The Candidate Skyline

Finally, after performing the mapping policy a set of candidate skyline is retrieved from each cluster. Then, we further conduct pairwise comparison to select the global skylines that are not dominated by any data items in the database. Figure 4 shows the global skyline of our example database.

Global skyline			
B3	7	*	6

Figure 4 The Global Skyline

From the result we conclude that there are no data items in any of the clusters that are better than the global skyline. Lastly, the missing values of the global skyline data items are imputed before returning them to the user. This process is achieved by referring to the current available data items and the relationship between data items to estimate the values of the missing dimensions.

CONCLUSION

The process of skyline is expensive due to the exhaustive pairwise comparison at the dimension (attribute) level between data items to select the skyline query results. The searching space is the most important critical factor that affects the performance and the speed of skyline process. In this paper we have presented and discussed skyline queries in incomplete database. We proposed an approach that manipulates the incomplete database to retrieve the skyline results. Our approach tends to perform the process of skyline queries with the aim of reducing the searching space by avoiding the unnecessary comparison between data items. The clustering and grouping with the concept of virtual data items *k-dom* are used during the process of generating the set of skyline answers for a given query operation. We intend to simulate our approach to measure the number of comparisons between data items and the time taken during the skyline process.

REFERENCES

Chee-Yong, C., Jagadish, H.V., Kian-Lee, T., Anthony, K.H. & Zhenjie, Z. (2006(a), March). *On high dimensional skylines*. Paper presented at the 10th International Conference on Extending Database Technology. Retrieved from http://www.comp.nus.edu.sg/~atung/publication/edbt06.pdf

Chee-Yong, C., Jagadish, H.V., Kian-Lee, T., Anthony K.H. & Zhenjie, Z. (2006(b), June). *Finding k-dominant skylines in high dimensional space*. Paper presented at the ACM SIGMOD International Conference on Management of Data. Retrieved from http://www.im.cjcu.edu.tw/~kdir-lab/paper/B02.pdf

Dimitris P., Yufei T., Greg F., & Bernhard S. (2003, June). *An optimal and progressive algorithm for skyline queries*. Paper presented at the International Conference on Management of Data. Retrieved from http://infolab.usc.edu/csci587/Fall2010/slides/ali-skyline.pdf

Donald, K., Frank, R., & Steffen, R. (2002, August). *Shooting stars in the sky: An online algorithm for skyline queries*. Paper presented at the 28th International Conference on Very Large Data Bases. Retrieved from http://www.dbis.ethz.ch/research/publications/43.pdf

Ilaria, B., Paolo, C., & Marco, P. (2006, November). *SaLSa: Computing the skyline without scanning the whole sky*. Paper presented at the 15th International Conference on Information and Knowledge Management. Retrieved from http://www-db.deis.unibo.it/research/papers/CIKM06.pdf

Jan, C., Parke, G., Jarek, G., & Dongming, L. (2003, March). *Skyline with presorting*. Paper presented at the 19th International Conference on Data Engineering. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1260846

Jian, P., Wen, J., Martin, E., & Yufei, T. (2005, September). *Catching the best views of skyline: A semantic approach based on decisive subspaces*. Paper presented at the 31st International Conference on Very Large Data Bases. Retrieved from http://www.vldb2005.org/program/paper/tue/p253-pei.pdf

Kian-Lee, T., Pin-Kwang, E., & Beng, C. O. (2001, September). *Efficient progressive skyline computation*. Paper presented at the 27th International Conference on Very Large Data Bases. Retrieved from http://www.vldb.org/conf/2001/P301.pdf

Man, L. Y. & Nikos, M. (2007, September). *Efficient processing of top-k dominating queries on multi-dimensional data*. Paper presented at the 33rd International Conference on Very Large Data Bases. Retrieved from http://www.vldb.org/conf/2007/papers/research/p483-yiu.pdf

Man, L. Y. & Nikos, M. (2009, June). Multi-dimensional top-k dominating queries. *The Very Large Data Bases Journal* 18 (3), 695–718. doi: 10.1007/s00778-008-0117-y

Mohamed F. M., & Justin J. L. (2009, June). *Toward context and preference-aware location-based services*. Paper presented at the 8th International Workshop on Data Engineering for Wireless and Mobile Access. Retrieved from http://www-users.cs.umn.edu/~justin/papers/MobiDE09.pdf

Mohamed E. K., Mohamed F. M. & Justin J. L. (2008, April). *Skyline query processing for incomplete data*. Paper presented at the 24th International Conference on Data Engineering. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4497464

Parke, G., Ryan, S. & Jarek, G. (2005, August). *Maximal vector computation in large data sets*. Paper presented at the 31st International Conference on Very Large Data Bases. Retrieved from http://www.vldb2005.org/program/paper/tue/p229-godfrey.pdf

Stephan, B., Donald, K., & Konrad, S. (2001, April). *The skyline operator*. Paper presented at the 17th International Conference on Data Engineering. Retrieved from http://www.dbis.ethz.ch/research/publications/38.pdf