Identifying Skylines in Dynamic Incomplete Database

GHAZALEH BABANEJAD, HAMIDAH IBRAHIM,
NUR IZURA UDZIR, FATIMAH SIDI, GHONCHEH BABANEJAD
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
Serdang, Selangor D.E
MALAYSIA
Ghazaleh.babanejad@gmail.com, hamidah.ibrahim@upm.edu.my, izura@upm.edu.my,
fatimah@upm.edu.my, Goncheh.babanejad@gmail.com

ALI A.ALWAN
Department of Computer Science
International Islamic University Malaysia
MALAYSIA
aliamer@iium.edu.my

Abstract: - Nowadays in database systems finding the best results that meet the preferences of users is the most important issue. Skyline queries will present the data items that are not being dominated by the other items in a database. Most of the operations assume the database is complete which means there are no missing values in the database dimensions. In reality, databases are not complete especially for multidimensional database. Missing values have a negative effect on finding skyline points. It changes the native of dominance relation, leads to cyclic dominance and unsatisfying the transitivity property of skylines. This problem becomes more severe in dynamic database in which new items are inserted or items are deleted or updated from the database. Besides, most of the works that handled the incomplete issue assumed that items are static. In this paper we propose the new approach which finds the most relevant data items that meet user's preferences for dynamic incomplete databases.

Key Words: - Skyline queries, Preference queries, Incomplete database, Dynamic database.

1 Introduction

Choosing the best option between different choices based on preferences of users is not easy. Finding the best and accurate answer is one of the most important issues in real life. As an example consider a user who wanted to go for holiday and the user's preferences for choosing a hotel are: (i) hotel that is near to a beach (ii) hotel with cheap price. But we know that hotels that are near to a beach are expensive compared to those which are far away from a beach.

SQL is one of the traditional database systems that returns results that match exactly with the preferences, then it will not return results which are near to the preferences. Hence several preference evaluation techniques have been proposed, namely: top-k, multiobjective method, k-dominance, k-frequency, top-k dominance, skyline, ranked skyline, k-representative dominance, distance based dominance, and ∈-skyline [1-8], as shown in Figure 1. This work focuses on skyline queries.

Databases which are used for skylines can be divided into two categories: Complete and Incomplete databases. Most of the existing approaches assume that the databases are complete and static. It is obvious that in reality databases are not complete and their data items keep on changing. For example consider sensors which are allocated in the forests for record the humidity, temperature and other dimensions for checking and preventing disaster. There are so many changes in each of these dimensions where new data are added while some of them are deleted or updated. There are also some sensors which could not send signals which lead to incomplete data.

In this paper we propose an algorithm named IDS that attempt to find skyline points in dynamic incomplete databases with minimum comparison,

i.e. it compares only necessary data items and discarded the rest of the data items.

The rest of this paper organized as follows:

Section 2 presents the related works. In section 3 we discuss the problem formulation. in section 4 we propose the IDS algorithm. And in the last part of this paper, we conclude and provide the future directions.

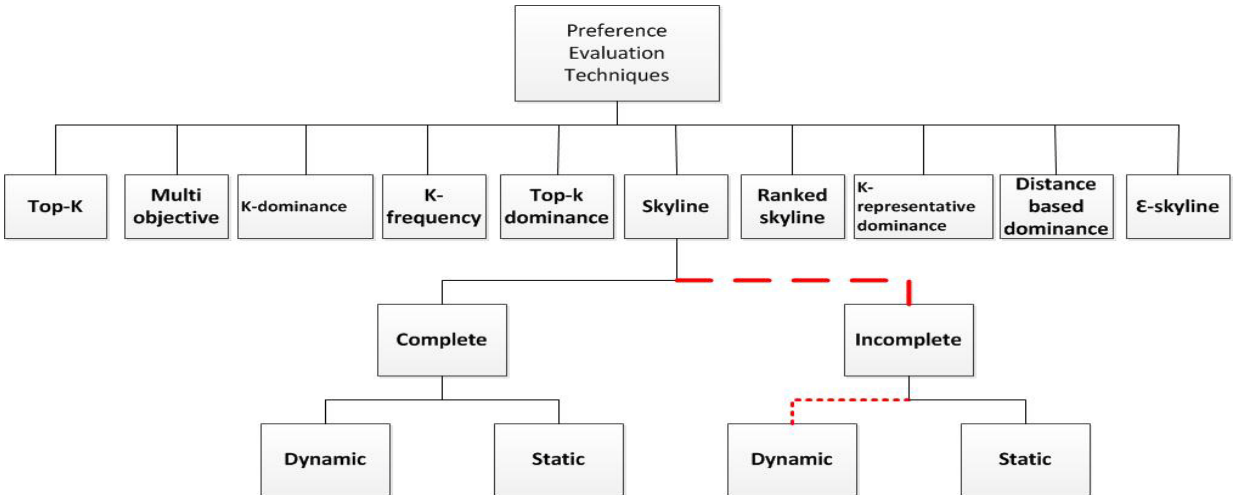

Fig. 1. Preference Evaluation Techniques

2 Related work

The based use of skyline queries is applied in databases. Several algorithms exist for skyline queries like BNL, D&C and algorithm using B-tree that is proposed by Borzsony et al. [6]. For these algorithms no preprocessing is needed. There are also other algorithms which proposed presorting and indexing [2-5].
The algorithms was proposed for certain environment like partially-ordered [7], high-dimensional [1], [8-9], sliding window [10-11], mobile ad-hoc networks [12], web information systems [13], and data mining [14].
There are algorithms which are proposed specially for dynamic database, most of them are based on top-k dominant [15] and k-dominant skyline [16].

Based on the literatures that we have analyzed, there is no work done for skyline queries in dynamic incomplete database. However in reality most of the databases are incomplete and their data keep on changing.

There are three algorithms which focused on incomplete static databases. In this work we have chosen Khalefa et al.'s algorithm [17] as a base to overcome the incomplete dynamic drawbacks.

The reasons for choosing Khalefa et al.'s algorithm [17], ISkyline, are:

(1) In ISkyline, there are several steps and in each step a set of different candidate skylines is produced. These include local skylines, shadow skylines, and virtual points. These steps are used as points to identify the items that are not needed for comparison whenever the database is changed.[18]
(2) In SIDS [19] whenever the database is changed, comparisons against all the existing items of the database need to be performed because the algorithm does not keep track the potential/candidate skylines.[18]
(3) In Incoskyline [20] finding new skylines after the database is changed will incur more comparison compared to ISkyline [17] since there is no domination history that can assist in finding the new skylines.[18]

3 Problem Formulation

3.1 Preliminaries

In this work we consider that bigger values are better than smaller. In this section we will cover the basic definitions which are related to the skyline queries for dynamic incomplete database.

ISBN: 978-1-61804-276-7

3.3.1 Definition 1: Dominance Relation
Given a database D with items P_i, i = 1, 2, …, m, and n dimensions $d = \{d_1, d_2, \ldots, d_n\}$. Let two d-dimensional items $P_k = \{U_1, U_2, \ldots, U_n\}$ and $P_l = \{S_1, S_2, \ldots, S_n\}$, P_k dominates P_l denoted by $P_k \succ P_l$ (greater values are preferred) if and only if the following condition holds:

$$\forall\, i \in d\,,\ U_i \geq S_i \wedge \exists\, j \in d,\ U_j > S_j$$

3.3.2 Definition 2: Skyline Query
Given a set of items D, an item $P_i \in D$ is a skyline if there is no other items $P_j \in D$ that dominates P_i. Items that are not dominated by other items in the database D are considered as skylines. Skylines have the transitivity property that means if P_i dominates P_k and P_k dominates P_l it leads to P_i dominates P_l [6].

3.3.3 Definition 3: Incomplete Database
A database *DI* is incomplete if and only if, it contains at least a data item with missing values in one or more dimensions. There are many reasons for missing values in databases like mistake in data entry, inaccurate data from heterogeneous data sources, and integrating heterogeneous schemes [19].

3.3.4 Definition 4: Dynamic Database
A database *DD* is said to be dynamic if the items in the database keep on changing in which new items are inserted, existing items are deleted and updated.

3.3.5 Definition 5: Skyline Query in Dynamic Incomplete Database
Given an incomplete database, *DI*, the set of skylines based on *DI*, *S*, and its new state, *Dnew*, $P_i \in Dnew$ is a skyline if there is no item in the new state that dominates P_i. Finding the set of skylines in the new state, *Dnew*, should incur the least number of comparisons between the data items in the *Dnew* which will indirectly incur the least possible time.

The data item $P_j \in DI$ may have missing values in one or more of the dimensions. The initial dataset *DI* may change to a new state, *Dnew*, due to the following operations:

(1) Insert Operation: $Dnew = DI \cup D_{<insert>}$ where $D_{<insert>}$ is the new set of items to be inserted into the initial database *DI*.
(2) Delete operation: $Dnew = DI - D_{<delete>}$ where $D_{<delete>}$ is the set of data items to be deleted from the initial database *DI*.
(3) Update operation: $Dnew = (DI - D_{<delete>}) \cup D_{<insert>}$ where an update operation is considered as a delete operation followed by an insert operation.

The algorithms which are proposed by Khalefa et al. [17], Alwan et al. [20], and Bharuka et al. [19], are applicable to find skyline points in the initial database *DI*, and if new items are added to the database *DI*, the whole dataset needs to be analysed. We prove that IDS algorithm can decrease the number of comparisons for finding skyline points over dynamic incomplete database.

4 The Proposed Algorithm

As mentioned earlier the missing values have negative effect on finding skyline points. Incomplete items are non-transitive and lead to the cyclic dominance. Hence we will not have any skyline points as all the items with missing values may dominate each other [17].

Our proposed algorithm, IDS, attempt to find skyline points over dynamic incomplete database. This is shown in Figure 2. To prove that the IDS algorithm is correct we apply the new dataset over the ISkyline algorithm to produce skyline points,[S', and compare the result with our algorithm, if the results are equal then we can conclude that our IDS algorithm is correct.

Now we will briefly illustrate the ISkyline algorithm. For more detail about this algorithm readers may refer to [17]. First the bucket algorithm is applied to the database *DI* and every item is categorized in the related bucket based on the missing dimension(s). Then in every bucket the local skylines are identified by performing pairwise comparison between items of each bucket. Next, Virtual Points (VPs) and Shadow Skylines (SSs) are derived items which are dominated by VP are removed from the bucket and are considered as Shadow Skylines while the remaining items are considered as candidate skylines. After that, comparison between the shadow skylines and the candidate skylines are performed which then

produced the skyline points, S. In every step we keep track of the domination relations and store them in a domination history table (HD).

For an insert operation, first we should perform the bucket algorithm for the new items, $D_{<insert>}$, and then find the temporary local skylines between their items, TLS. Then we do the comparison between TLS and S to produce, S', (Figure 3).

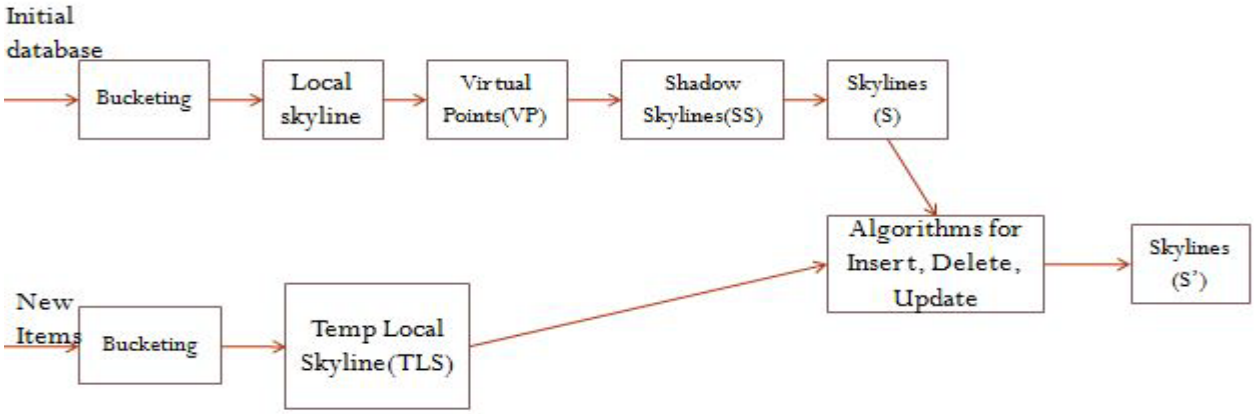

Fig. 2. IDS Framework

```
(Checking Temp Local Skylines with Skylines)
For i=0  to S.count
  For j=0  to  TLS.count
    If Qj ∈ TLS  dominates Pi ∈  S then
       Remove Pi from S and Insert Pi to SS and virtual point of Qj to VP
    End if
    If  Pi ∈ S dominates Qj ∈ TLS then
        Remove Qj from TLS and  Insert Qj into TSS and virtual point of Pi into VP
     End if
   End for
End for
(Checking Temp Local Skylines with Virtual points)
For  vp.index=0 to vp.count
         For  j=0 to  TLS.count
                  If VP dominates Qj then
                     Remove Qj from TLS and Insert into SS
                  End if
         End for
End for
(Checking Temp Local Skylines with Shadow Skylines)
For SS.index =0  to SS.count
         For j=0  to  TLS.count
                  If  SS  dominates  Qj  then
                           Remove Qj from TLS and insert to SS and virtual point of SS to VP
                  end if
         end for
 End for
(At the point, all elements of TLS should be added to Skyline list)
Merge (TLS,S)
```

Fig. 3. The algorithm for an insert operation

ISBN: 978-1-61804-276-7

```
(Check domination history)
If Pi ∈ D<delete> is dominated by local skyline then
        Delete Pi from the database
    If Pi is a local skyline then
        Retrieve all items dominated by Pi from the domination history table
        and save them in P-ret and delete Pi from the database, SS, and VP
          (Check P-ret items with each other)
            If Li ∈ P-ret is dominated by other items like Lj ∈ P-ret then
           Ignore Li and insert Li in the domination history table
                Else (Check P-ret with VP)
                  If VP dominates Lj
                      Insert Lj into SS and domination history table
                       Else (Check P-ret with SS)
                         If SS dominates Lj
                           Insert Lj into SS and the virtual points of SS into VP
                         Else
                           Insert Lj into Skylines (S')
                         End if
                    End If
            End If
End If
```

Fig. 4. Presents the algorithm for the insert operation

For deletion, the IDS algorithm will work as follows. Consider that $P_i \in D_{<delete>}$, based on the domination history (DH) if P_i is dominated by any items in local skylines part then P_i is deleted from the database. But if P_i is a local skyline, P_i is deleted from the database, virtual points and shadow skylines, and then the items which previously were dominated by P_i are retrieved from domination history and kept in P-ret. In the next step all items in P-ret, $L_i \in$ P-ret, are compared to each other. If an item, (L_i) dominates another item (L_j) then L_j should be ignored. Otherwise L_i is checked with VP. If VP dominates L_i then L_i will be the shadow skyline else L_i is checked with the shadow skylines. If shadow skylines do not dominate L_i then L_i is the skyline (Figure 4).

For the update operation, the delete algorithm is enforced which is then followed by the insert algorithm to produce the new skylines, *S'*.

We have applied the IDS algorithm over three different datasets with 150, 100 and 300 items and then compared it with ISkyline algorithm. Applied ISkyline over the initial dataset *DI* and counted the number of comparisons then added new items *Dnew* to *DI* and run ISkyline algorithm and calculated the number of comparisons. Then we run the IDS algorithm with *DI* and *Dnew* then counted the number of comparisons. We saw that the number of comparisons for IDS algorithm decreased. We can conclude that IDS algorithm can overcome the drawback in existing algorithms which is unable to handle dynamic incomplete databases.

5 Conclusion

This paper presents an algorithm, IDS, for solving the problem of skyline queries over dynamic incomplete database where multi-dimensional items have missing values and these items are changing over time due to inserting new items, deleting and updating existing items. Until now there is no algorithm which can find skylines over dynamic incomplete database.

References:

[1] Chan, C. Y., Jagadish, H. V., Tan, K. L., Tung, A. K., and Zhang, Z. (2006, June). Finding k-dominant skylines in high dimensional space.

In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (pp. 503-514).

[2] Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2003, March). Skyline with presorting. In Proceedings of the International Conference on Data Engineering (Vol. 3, pp. 717-719).

[3] Kossmann, D., Ramsak, F., and Rost, S. (2002, August). Shooting stars in the sky: An online algorithm for skyline queries. In Proceedings of the 28th International Conference on Very Large Data Bases (pp. 275-286).

[4] Papadias D., Tao Y., Fu G., and Seeger B. (2005). Progressive skyline computation in database systems, ACM Transactions on Database Systems (Vol. 30, No. 1, pp. 41–82).

[5] Tan, K. L., Eng, P. K., and Ooi, B. C. (2001, September). Efficient progressive skyline computation. In Proceedings of the International Conference on Very Large Data Bases (Vol. 1, pp. 301-310).

[6] Borzsony, S., Kossmann, D., and Stocker, K. (2001). The skyline operator. In Proceedings of the 17th International Conference on Data Engineering (pp. 421-430).

[7] Chan, C. Y., Eng, P. K., and Tan, K. L. (2005, June). Stratified computation of skylines with partially-ordered domains. In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (pp. 203-214).

[8] Pei, J., Jin, W., Ester, M., and Tao, Y. (2005, August). Catching the best views of skyline: A semantic approach based on decisive subspaces. In Proceedings of the 31st International Conference on Very Large Data Bases (pp. 253-264).

[9] Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J. X., and Zhang, Q. (2005, August). Efficient computation of the skyline cube. In Proceedings of the 31st International Conference on Very Large Data Bases (pp. 241-252).

[10] Lin, X., Yuan, Y., Wang, W., and Lu, H. (2005, April). Stabbing the sky: Efficient skyline computation over sliding windows. In Proceedings of the 21st International Conference on Data Engineering (pp. 502-513).

[11] Y. Tao and D. Papadias. (2006). Maintaining sliding window skylines on data streams, IEEE Transactions on Knowledge and Data Engineering, TKDE (Vol. 18, No. 2, pp. 377–391).

[12] Huang, Z., Jensen, C. S., Lu, H., and Ooi, B. C. (2006, April). Skyline queries against mobile lightweight devices in MANETs. In Proceedings of the 22nd International Conference on Data Engineering (pp. 66-66).

[13] Balke, W. T., Güntzer, U., and Zheng, J. X. (2004). Efficient distributed skylining for web information systems. In Proceedings of the International Conference of Advances in Database Technology-EDBT 2004 (pp. 256-273).

[14] Jin, W., Han, J., and Ester, M. (2004). Mining thick skylines over large databases. In Knowledge Discovery in Databases: PKDD 2004 (pp. 255-266).

[15] Kontaki, M., Papadopoulos, A. N., and Manolopoulos, Y. (2010). Continuous processing of preference queries in data streams. In Proceedings of the International Conference on SOFSEM 2010 Theory and Practice of Computer Science (pp. 47-60).

[16] Cui, X. W., Dong, L. G., Zou, H., and An, X. M. (2011, July). Finding k-dominant skyline in dynamic data set. In Proceedings of the Seventh International Conference on Natural Computation (Vol. 3, pp. 1247-1250).

[17] Khalefa, M. E., Mokbel, M. F., Levandoski, J. J. (2008, April). Skyline query processing for incomplete data. In Proceedings of the 24th International Conference on Data Engineering (pp. 556-565).

[18] Babanejad, Ghazaleh and Ibrahim, Hamidah and Udzir, Nur Izura and Sidi, Fatimah and Aljuboori,Ali A.Alwan (2014)*Finding skyline points over dynamic incomplete database*. In: Malaysian National Conference of Databases 2014 (MaNCoD 2014), 17th September 2014, Serdang, Selangor.

[19] Alwan, A., Ibrahim, H., Udzir, N. I., and Sidi, F. (2013). Estimating missing values of skylines in incomplete dataset. In Proceedings of the Second International Conference on Digital Enterprise and Information Systems (DEIS2013) (pp. 220-229).

[20] Bharuka, R., and Kumar, P. S. (2013, January). Finding skylines for incomplete data. In Proceedings of the Twenty-Fourth Australasian Database Conference (Vol. 137, pp. 109-117).