An Efficient Approach for Processing Skyline Queries in Incomplete Multidimensional Database

Ali A. Alwan1 · Hamidah Ibrahim2 · Nur Izura Udzir2 · Fatima Sidi2

Received: 29 August 2015 / Accepted: 18 January 2016 / Published online: 12 February 2016 © King Fahd University of Petroleum & Minerals 2016

Abstract In recent years, there has been great attention given to skyline queries that incorporate and provide more flexible query operators that return data items (skylines) which are not being dominated by other data items in all dimensions (attributes) of the database. Many variations in skyline techniques have been proposed in the literature. However, most of these techniques determine skylines by assuming that the values of all dimensions for every data item are available (complete). But this assumption is not always true particularly for large multidimensional database as some values may be missing (not applicable during the computation). In this paper, we proposed an efficient approach for processing skyline queries in incomplete database. The experimental results show that our proposed approach has significantly reduced the number of pairwise comparisons and the processing time in determining the skylines compared to the previous approaches.

Keywords Skyline · Skyline queries · Preference queries · Incomplete database · Multidimensional database · Query processing

1 Introduction

In recent years, there has been much interest to design and develop database management systems that incorporate and provide more flexible query operators that limit the result set to the most relevant one which best meet the user preferences (criteria). The preference queries are significant and widely used in a variety of application domains, such as multi-criteria decision making [1–5], where many criteria are involved in the query statement to select the most suitable answer that fits the user’s demands. Another domain that applied the preference queries is the decision support system which helps to combine various interests in recommending a strategic decision. Other domains include e-commerce environments like trade-off between the price, quality, and efficiency of the products to be purchased; personal preferences of users who request a web service such as hotel recommender [6], restaurant finder [7,8], and recipe recommender [9]; and peer-to-peer network database [10]. Due to the importance of preference queries that obviously appeared in many real-life database applications as mentioned above, a various number of preference evaluation techniques have been proposed in the literature [1–3,5,7,11–26].

One of the predominant and frequently used types of preference query is the skyline query [1–3,5,10–12,16,17,19,21]. Skyline queries attempt to find the set of non-dominated data items (skylines) in a given multidimensional database. For instance, suppose a tourist is seeking for a hotel in a specific area that is near to a beach and at the same time is cheap in price. Among the set of available hotels, skyline queries would return only those non-dominated hotels that meet the tourist’s preferences. Many preference query approaches based on the concept of skyline technique have been proposed in the literature [1,2,5,6,11,12,15,17,21,24,27–33], but most of them assumed that all values of the dimensions
are available, i.e., the database state is complete with no missing values. However, this assumption is not always true and realistic in the real world, particularly applications with large database and high number of dimensions as some values may for some reasons missing.

The incompleteness of database introduces new challenges in processing the skyline queries as applying the skyline technique directly on a large incomplete database is impractical and can lead to prohibitive cost. Moreover, the missing values will influence negatively on the process of finding skylines due to the exhaustive pairwise comparisons between the incomplete data items. Besides that, it also leads to loss the transitivity property and the cyclic dominance [28]. In addition, as the size of the skylines increases dramatically while the rate of missing values is large, the results do not give any insight to the user [28].

For example, consider the following scenario where a tourist seeking for a restaurant in a city that is the nearest to his current location, cheap in price, with large number of available tables, and at the same time with the highest rate. A restaurant \( r_1 \) is represented with four dimensions \((d_1, p_1, t_1, r_1)\) where \( d_1, p_1, t_1, \) and \( r_1 \) represent the distance, price, number of available tables, and the restaurant rate, respectively. We assume the restaurant database consisting of 3 records with missing values, namely \( r_1(\ast, 5, 4, 9) \), \( r_2(5, \ast, \ast, 7) \), and \( r_3(3, \ast, 5, \ast) \). The symbol \((\ast)\) is used to represent the missing values in the records. Based on the common dimensions with non-missing values, \( r_1 \) dominates \( r_2 \) as \( r_1 \) is better than \( r_2 \) in the fourth dimension (restaurant rate) (greater is better), while \( r_2 \) dominates \( r_3 \) as \( r_2 \) is better than \( r_3 \) in the first dimension (distance). However, comparing \( r_1 \) against \( r_3 \) shows that \( r_3 \) dominates \( r_1 \). Thus, \( r_1 \) does not dominate \( r_3 \) which therefore means the dominance relation is not transitive. Based on the transitivity property when \( r_1 \) dominates \( r_2 \), and \( r_2 \) dominates \( r_3 \), hence, as a consequence, \( r_1 \) dominates \( r_3 \), which does not happen in our example. However, \( r_3 \) dominates \( r_1 \) which means that the dominance relation is cyclic. From this example, all these three restaurants are being dominated, and thus, no skylines will be revealed.

In this paper, we present an approach for processing skyline queries in incomplete database in which some values of the data items do not exist (missing). Our approach utilizes the bitmap representation to divide the initial database into a set of distinct clusters whereby each cluster gathers the data items that have missing values on the same dimension(s). Clustering the initial database avoids a significant number of unnecessary pairwise comparisons between the data items. Then, a set of groups is created from each cluster based on the upper-bound value of the cluster and the lower-bound value of a recently created group which act like a threshold value in identifying the group members. This process aims at reducing the number of pairwise comparisons in each cluster in determining the local skylines. A set of data items named \( k\)-dom is derived from the local skylines based on the common dimensions with non-missing values. These \( k\)-dom skylines are merged as a single global \( k\)-dom skyline that is inserted at the top of every cluster to prevent any dominated local skylines from further processing. The global \( k\)-dom skyline is then compared to the local skylines of the cluster to produce the candidate skylines. This process further reduces the number of candidate skylines. Finally, pairwise comparisons are performed between these candidate skylines to derive the final skylines. Intensive experiments over different types of dataset have been carried out to evaluate the proposed approach.

The rest of the paper is structured as follows. In Sect. 2, the previous works related to this work are presented and discussed. In Sect. 3, the basic definitions and notations, which are used throughout the paper, are set out. Section 4 describes our proposed approach for processing skyline queries in incomplete database which consists of four main phases. Examples are also given to clarify the phases. The experiment results are illustrated in Sect. 5. Conclusion and further research direction are presented in the final Sect. 6.

2 Related Works

Many types and variations in skyline techniques have been described in the database literature. Most of these skyline techniques focused on improving the efficiency of skyline query processing. The process of skyline is expensive due to the exhaustive pairwise comparisons between data items to determine the skylines. The searching space is the most important factor that influences the performance of skyline process, and thus, most researchers in this area concentrate on strategies that derive skyline query answers by pruning the searching space as small as possible. Although many skyline techniques have been reported in the literature, but most of them do not handle cases where the database contains incomplete data items. In the following, we present the related approaches for processing skyline queries in both complete and incomplete databases.

The first work on skyline query processing in the database field is proposed by Stephan et al. [21]. They have proposed two algorithms, namely Block-Nested-Loop (BNL) and Divide-and-Conquer (DC). BNL algorithm produces skylines by repeatedly scanning 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 database. The second algorithm, DC, divides the database into two equivalent sets. Then, it finds the local skyline sets and combines the output of the two local skyline sets to further perform comparison to eliminate those local skylines which are dominated by the global skylines. DC does not require any preprocessing process on the initial database such as sorting.