International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 1 | ISSN : 2456-3307

Skyline Computations with Partially Ordered Domains using Indexing Method

Y. Gopi [*1], Nallagonda Anitha[2]
[*1]Assistent Professor, Department of MCA, St. Mary's Group of Institutions, Guntur, Andhra Pradesh, India
[2]PG Students, Department of MCA, St. Mary's Group of Institutions, Guntur, Andhra Pradesh, India

ABSTRACT

Efficient processing of skyline queries with partially ordered domains has been intensively addressed in recent years. To further reduce the query processing time to support high-responsive applications, the skyline queries that were previously processed with user preferences similar to those of the new query contribute useful candidate result points. Hence, the answered queries can be cached with both their results and the user preferences such that the query processor can rapidly retrieve the result for a new query only from the result sets of cached queries with compatible user preferences. When caching a significant number of queries accumulated over time, it is essential to adopt effective access methods to index the cached queries to retrieve a set of relevant cached queries for facilitating the cache-based skyline query computations. In this paper, we propose an extended depth-first search indexing method (e-DFS for short) for accessing user preference profiles represented by directed acyclic graphs (DAGs), and emphasize the design of the e-DFS encoding that effectively encodes a user preference profile into a low-dimensional feature point which is eventually indexed by an R-tree. We obtain one or more traversal orders for each node in a DAG by traversing it through a modified version of the depth-first search which is utilized to examine the topology structure and dominance relations to measure closeness or similarity. As a result, e-DFS which combines the criteria of similarity evaluation is able to greatly reduce the search space by filtering out most of the irrelevant cached queries such that the query processor can avoid accessing the entire data set to compute the query results. Extensive experiments are presented to demonstrate the performance and utility of our indexing method, which outperforms the baseline planning techniques by reducing 37% of the computational time on average.

Keywords : Indexing Methods, Query Processing, Multi-Dimensional Databases, Data Management.

I. INTRODUCTION

Given a dataset containing multidimensional data points, a preference query retrieves a set of data points that could not be dominated by any other points. Nowadays, preference query has emerged as a considerably important tool for multi-preference analysis and decision making in real-life. Skyline query is considered to be the most important branch of preference query. While preference query depends upon a general dominance definition, skyline queries explicitly considers total or partial orders at different dimensions to identify dominance. Given a set of data points D, a skyline query returns an interesting subset of points of D that are not dominated (with respect to the attributes of D) by any points in D. A data point p_1 is said to dominate another point p_2 if p_1 is at least as good as p_2 on all attributes, and there exists at least one attribute where p_1 is better than p_2. Thus, a skyline query essentially computes the subset of "optimal" points in D, which has many applications in multi-criteria optimization problems.

A skyline query is classified as static if all the partially ordered domains remained unchanged at query time; otherwise, if a user can specify a different partially ordered domain to reflect his preference at query-time, it is considered a dynamic skyline query.

There has been a lot of research on the skyline query computation problem, most of which are focused on data attribute domains that are *totally ordered*, where any two values are comparable. Usually, the best value for a totally ordered domain is either its maximum or minimum value or a totally ordered domain can be represented as a chain. In our work, regarding totally ordered domains, we assume the smaller value is more preferred. Many approaches are proposed to handle skyline queries with only totally ordered domains and divided into two categories according to whether rely on any predefined index over the dataset. The category of techniques that do not rely on any predefined index include **BNL** [4], **D&C** [4], **SFS** [27], **LESS** [21], **Salsa** [3] and **OSP**

However, in many applications, some of the attribute domains are *partially ordered*

Such as interval data (e.g. temporal intervals), type hierarchies, and set-valued domains, where two domain values can be incomparable. Since a partial order satisfies inreflexivity, asymmetry and transitivity, a partially ordered domain can be represented as a directed acyclic graph (DAG). A number of recent research works [10, 42] has started to address the more general skyline computation problem where the data attributes can include a combination of totally and partially ordered domains. **SDC⁺** [10] is the first index method proposed for the more general skyline query problem, which is an extension of the well-known **BBS** index method [38] designed for totally ordered domains. **SDC⁺** employs an approximate representation of each partially ordered domain by transforming it into two totally ordered domains such that each partially ordered value is presented as an interval value. The state-of-the-art index method for handling partially ordered domains is **TSS** [42], which is also based on **BBS**. Unlike **SDC⁺**, **TSS** uses a precise representation of a partially ordered value by mapping it into a set of interval values. In this way, **TSS** avoids the overhead incurred by **SDC⁺** to filter out false positive skyline records.

Recently, a new index method called **ZB-tree** [33] has been proposed for computing skyline queries for totally ordered domains which has better performance than **BBS**. The **ZB-tree**, which is an extension of the **B⁺-tree**, is based on interleaving the bit-string representations of attribute values using the Z-order to achieve a good clustering of the data records that facilitates e client data pruning and minimizes the number of dominance comparisons.

Given the superior performance of **ZB-tree** over **BBS**, one question that arises is whether we can extend the **ZB-tree** approach to obtain an index that has better performance than the state-of-the-art **TSS** approach, which is based on **BBS**. Since the **ZB-tree** indexes data based on bit string representation, one simple strategy to enhance **ZB-tree** for partially ordered domains is to apply the well-known bit vector scheme [9] to encode partially ordered domains into bit strings. We refer to this enhanced **ZB-tree** as **CHE+ZB**. We also combine the encoding scheme in **TSS** with **ZB-tree** to be another variant of **ZB-tree** named **TSS+ZB**. Our experimental evaluation shows that while **CHE+ZB**, **TSS+ZB** and **TSS** have comparable performance, the performance of **CHE+ZB** and **TSS+ZB** is often suboptimal as the bit vector encoding scheme does not always pro-duce good data clustering and effective data pruning.

Since partially ordered domains are typically used for categorical attributes to rep-resent user preferences (e.g., preferences for colors, brands, airlines), we expect that the partial orders for representing user preferences are not complex, densely connected structures. As an example, consider the partial order shown in Figure 1.1 representing a user's preference for car brands. The partial order shown has a simple structure consisting of one minimal value (representing the top preference for Ferrari), one maximal value (representing the least preference for Yugo), and two chains: the left chain represents the user's preference for German brands (with Benz

being preferred over BMW) which are incomparable to the right chain representing the user's preference for Japanese brands (with Toyota being preferred over Honda).

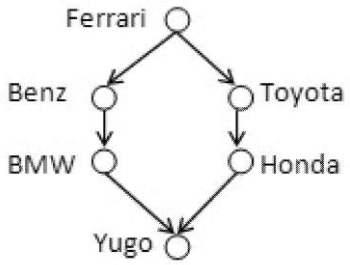


Figure 1. : Partial order representing a user's preference on car brands.

In our work, we introduce a new indexing approach, called **ZINC** (for Z-order Indexing with Nested Codes), that combines **ZB-tree** with a novel *nested encoding scheme* for partially ordered domains. While our nested encoding scheme is a general scheme that can encode any partial order, the design is targeted to optimize the encoding of commonly used partial orders for user preferences which we believe to have simple or moderately complex structures. The key intuition behind our proposed encoding scheme is to organize a partial order into nested layers of simpler partial orders so that each value in the original partial order can be encoded using a sequence of concise, "local" encodings within each of the simpler partial orders. Our experimental results show that using the nested encoding scheme, **ZINC** significantly outperforms all the other competing methods.

II. Related Work

2.1 Skyline Queries with Totally Ordered Domains

After skyline query processing is introduced into database area by [4], researchers de-vote e ort on processing skyline queries with totally ordered domains where the best value for a domain is either its maximum or minimum value.

2.1.1 NL, BNL

The first algorithm for processing skyline query is the simple *Nested-Loops* algorithm (**NL** algorithm). It compares every data point with all the data points (including itself), and as a result it can work for any orders. However, obviously **NL** is costly and inefficient. In [4], a variant of **NL** is proposed called *Block Nested-Loops* algorithm (**BNL** algorithm), which is significantly faster and is an a-block-one-time algorithm rather than a-point-one-time as **NL**. **BNL** achieves the e client processing by a good memory management. The key idea is to maintain in main memory a window, which is used to keep incomparable data points. When a data point t_i is read from input, t_i is com-pared to all data points of the window. Based on the comparison, t_i is either discarded, put into the window or put into a temporary file which is allocated in disk and will be considered as input in the next iteration of the algorithm. At the end of each iteration, we can output a part of data points in the window that have been compared to all the data points in the temporary file. These points are not dominated by any other point and do not dominate any points that will be considered in following iterations. Be exactly, these output points are the points that are inserted into the window when the temporary file is empty. Thus, **BNL** achieves the effect of"a-block-one-time". In the best case, the most preferred objects fit into the window and only one or two iterations are needed. Meanwhile, **BNL** has considerable limitations to its performance. First, the performance of **BNL** is an acted very much by the discarding effectiveness which **BNL** cannot affect at all. Furthermore, there is no guarantee that **BNL** will complete in the optimal number of passes.

2.1.2 D&C

Divide-and-Conquer algorithm (**D&C** algorithm) [4, 32], as its name indicates, takes a divide-and-conquer strategy. It recursively divides the whole space into a set of partitions, skylines of which are easy to compute. Then, the overall skyline could be obtained as the result of merging these intermediate skylines.

2.1.3 SFS, LESS, Salsa, OSP

Sort-Filter-Skyline algorithm (**SFS** algorithm) proposed in [27] performs an additional step of pre-sorting before generating skyline points. In this step the input is sorted in some topological sort compatible with the given preference criteria so that a dominating point is placed before its dominated points. The second step is almost the same as the procedure of **BNL**, except that in **SFS** when a point is inserted into the window during a pass, we are sure that it is a most preferred point since no point following it can dominate it. **SFS** is guaranteed to work within the optimal number of passes since **SFS** can control the discarding effectiveness. Optimized algorithms, *Linear Elimination Sort for Skyline* (**LESS** algorithm) and *Sort and Limit Skyline algorithm* (**Salsa** algorithm), are derived from **SFS** in [21] and [3]. Finally, the *Object-based Space Partitioning* (**OSP** algorithm), which is proposed in [53], performs skyline computation in a similar manner, except for that organizes intermediate skyline points in a *left-child/right-sibling* tree, which accelerates the checking of whether the currently read point could be dominated by some intermediate skyline point.

All of the above methods do not rely on any predefined index structure over the dataset. They all require at least one scan through the data source, making them unattractive for producing fast initial response time. Another set of techniques [45, 31, 39, 33] are proposed which require that the dataset are already indexed before skyline evaluation and generally produce shorter response time.

Bitmap, Index

The *Bitmap* method is proposed in [45]. This technique encodes in bitmaps all the information needed to decide whether a data point belongs to the skyline. In specific, whether a given data point could be dominated can be identified through some bit-wise operations. This is the first technique utilize the efficiency of bit-wise operations. Meanwhile, the computation of the entire skyline is expensive since it has to retrieve the bitmaps of all data points. Also, because the number of distinct values in domains might by high and the encoding method is simple, the space consumption might be prohibitive. Another method, called *Index* method, is also proposed in [45]. It partitions the entire data into several lists, indexes each list by a B-tree and uses the trees to find the local skylines, which are then merged to a global one.

The skyline query computation suffers a high cost in high dimensions with partially ordered domains. In our previous work, we proposed a cache-based framework called Caching Support for Skyline computations (CSS) which uses a cache to store user preference profiles and skyline results such that CSS does not have to access the entire data set for calculating the skyline results for a new query. We have concluded that such a cache based approach improves upon existing methods and is especially well-suited for interactive applications that require a fast response time.

Disadvantages:

No effective access methods to index the cached queries for efficiently retrieving a set of relevant cached queries for the skyline query computations.

III. Proposed System

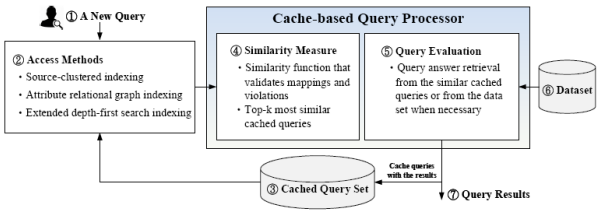
In addition to the source-clustered and the attribute relational graph (ARG) indexing methods, we propose an extended depth-first search indexing method (e-DFS) for accessing user preference profiles of the cached queries. We first perform the e-DFS encoding that effectively encodes a user preference profile into a low-dimensional feature point which is eventually indexed by an R-tree. We then obtain one or more traversal orders for each node in a DAG by traversing it through a modified version of the depth-first search which is utilized to examine the topology structure and dominance relations to measure closeness or similarity. The system

framework using indexing methods. When a new query q is requested, one access method (i.e., an indexing method) searches for the similar user preference profiles from the cached query set. Next, the system performs a similarity evaluation to compute the similarity scores only on the cached queries selected by the access method to measure the level of similarity with respect to the new query q, and then evaluates the new query q based on the top-k similar cached queries (s. If q cannot be answered given the cached queries, the system directly accesses the data set to compute the query result for q. Finally, the system outputs skyline query results to the user and caches the queries with the user preference profiles and the results.

Advantages

- It uses a new access method e-DFS
- An e-DFS encoding method to convert a DAG into a low-dimension point that preserves most of the preference orders.
- Access time to the cached queries greatly reduced.

Architecture



Modules:

Source-clustered Indexing

We introduce the first indexing method named source-clustered indexing. This method uses the node(s) in the first level of a DAG as the key(s) to search for similar user preference profiles. We use these nodes as the keys because the nodes in the first level dominate those in the second level; those in the second level dominate those in the third level, and so on. Therefore, the nodes in the first level are more important than those in the other levels.

Attribute Relational Graph Indexing

The source-clustered indexing method cannot efficiently handle the cached queries with complex user preference profiles, because the indexing structure simply uses the source nodes as the keys and the rest of the relations are not considered. On the other hand, the attribute relational graph (ARG) indexing structure maps a DAG to a corresponding ARG. The ARG indexing method uses the relations between vertices in a DAG as features, and converts a DAG to a multi-dimensional feature point.

Extended Depth-First Search Indexing

The ARG indexing method is designed to represent user preference profiles more effectively than the source-clustered indexing method, particularly when the number of vertices in a DAG is large. However, the encoding method of the ARG indexing may return high-dimensional feature points eventually indexed by a Tree that suffers the curse of dimensionality. Therefore, the ARG indexing method incurs high computational costs when searching for similar user preference profiles for new queries. In this section, we introduce a method, the extended depth-first search indexing algorithm (e-DFS for short), which utilizes a modified depth-first search algorithm to preserve the characteristics of the dominance relations in a DAG, while reducing the number of dimensions of the converted feature points. Similar to the ARG indexing method, the e-DFS indexing method adopts R-trees to store these feature points.

IV. CONCLUSION

We propose a new indexing method called extended depth-first search indexing method (e-DFS) for user preference profiles represented by DAGs to facilitate the access to the cached queries for efficient skyline query computation with partially ordered domains. The computation time of processing a new query is significantly reduced, because the query results are retrieved from the results of cached queries with compatible user preferences, which must be accessed

through an efficient access method to select a set of relevant cached queries for query processing.

V. REFERENCES

[1]. W. Balke, U. Guntzer, and C. Lofi. Eliciting matters controlling skyline sizes by incremental integration of user preferences. In DASFFA, pages 551-562, 2007.

[2]. W. Balke, U. Guntzer, and W. Siberski. Exploiting indi erence for customization of partial order skylines. In IDEAS, pages 80-88, 2006.

[3]. I. Bartolini, P. Ciacia, and M. Patella. E cient sort-based skyline evaluation. In TODS, volume 33(4), pages 1-49, 2008.

[4]. S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In ICDE, pages 421-430, 2001.

[5]. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. In JAIR, pages 135-191, 2004.

[6]. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Preference-based constrained optimization with cp-nets. In Computational Intelligence, vol-ume 20, pages 137-157, 2004.

[7]. C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In UAI, pages 71-80, 1999.

[8]. R. I. Brafman and C. Domshlak. Introducing variable importance tradeo s into cp-nets. In In Proceedings of UAI-02, pages 69-76. Morgan Kaufmann, 2003.

[9]. Y. Caseau. E cient handling of multiple inheritance hierarchies. In OOPSLA, pages 271-287, 1993.

[10]. C. Y. Chan, P. K. Eng, and K. L. Tan. Stratified computation of skylines with partially-ordered domains. In SIGMOD, pages 203-214, 2005.

[11]. C. Y. Chan, H. V. Jagadish, K. L. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In EDBT, pages 478-495, 2006.

[12]. S. Chaudhuri, N. Dalvi, and R. Kaushik. Robust cardinality and cost estimation for skyline operator. In ICDE, page 64, 2006.