Middle-East Journal of Scientific Research 22 (11): 1658-1662, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.22.11.21538

Skyline Operator on Join Datas

T. Nalini and M. Shakila

Department of Computer Science and Engineering, Bharath University, Chennai, India

Abstract: We propose to extend database systems by a Skyline operation. This operation filters out a set of interesting points from a potentially large set of data points. A point is interesting if it is not dominated by any other point. For example, a hotel might be interesting for somebody traveling to Nassau if no other hotel is both cheaper and closer to the beach. We show how SQL can be extended to pose Skyline queries, present and evaluate alternative algorithms to implement the Skyline operation and show how this operation can be combined with other database operations (e.g., join and Top N).

Key words: Indexes · Information management · Data storage systems · Partitioning algorithms · Algorithm design and analysis · Merging · SSPL · Big data · Skyline · Pruning

INTRODUCTION

Skyline operator is one of the most interesting set of points impartial enabling to find the best of the hotels in nearby and also the cost and The cheapest of all the given places. Here we impose also in the is one rhythms on these set databases and find with the domination of each other in accordance with the query in fulfilling the need. In computing the skyline property in this set of interesting points we take into consideration set monostatic point which detects your cheapest and nearest hotel and favorite of all. As with the comparison with the other implementation we here use the three different variants on these skyline operators such as the best-nested-loop variant, divide and conquer and finally the two skyline operators in in accordance with the set of queries. In computing the better of the two different heights of the building we have the two variant height attitudes which may tell us the best of the query. of the best operating query to choose the nearest one this is one of the best optional to find the vector detecting problem factor and this is graphical representational of the skyline properties. This type of finding the best one is the skyline [1-10]. So this skyline helps us to find in finding the optional one. We have the set of interesting points in detecting the best of all. This set of all interesting hotels are known as the skyline .and suppose we are in choice of detecting the cheapest and nearest hotels in a place. In consideration we have taken the whole set of points into the interesting set of vector points and this set of queries are taken into consideration as a skyline [11-20]. The skyline then is taken as the SQL databases is evaluated and calculated the most nearest and cheapest of all the set of queries.

Related Work

Index Based Algorithm: Index-based skyline algorithms utilize the reconstructed data-structures to avoid scanning the entire data set [21-26]. Tan et al. make use of bitmap to compute skyline of a table T A1; A2; ...; Ad. Given a tulle x ¼ x1; x2; ...; ad 2 T, x is encoded as a b-bit bit-vector, b ¼ Pd. This paper devises pruning operation on the candidate positional indexes and the mathematical analysis for pruning is presented in this paper. The experimental results show that SSPL has a significant advantage over the existing skyline algorithms. Dominance relationship between tulles is defined on skyline. 8t1; t2 2 T, t1 dominates t2 (denoted by t1 t2 set to 0, bit jib to bit kid are set to1. let Bisk represent the bit file corresponding to the jet bit in the it attribute Ai. It is given that a tuple x ¼ x1; x2; ...; xd Þ 2 T and xi is the ji Þth smallest value in Ai . Let A ¼ BS1j1 &BS2j2 &... &BSdjd where & represents the bitwise and operation. And let B ¼ BS1j1 jBS2j2 ... jBSdjd 1 where j represents the bitwise or operation. If there is more than a single one-bit in C ¼ A&B, x is not a skyline tuple.Otherwise, x is a skyline

Corresponding Author: T. Nalini, Department of Computer Science and Engineering, Bharath University, Chennai, India.

tuple. Kossmann *et al*. [22] propose NN algorithm to process skyline query. NN utilizes the existing methods for nearest neighbor search to split data space recursively. By a preconstructed R-tree, NN first finds the nearest neighbor to the beginning of the axes. Certainly, the nearest neighbor is a skyline tuple.

Nearest Neighbour Algorithm: Next, the data space is partitioned by the nearest neighbor to several subspaces. The subspaces that are not dominated by the nearest neighbor are inserted into a to-do list. While the to-do list is not empty, NN removesone of the subspaces to performsameprocess calculation is called*constraint-based*. These two types of queries use quite different query processing strategies. constraint-free query the key to efficient query processing is to reduce the number of datapoints to be accessed subset SKY T of T, in which 8t1 2 SKY T, 9t2 2 T, t2 t1. The subspaces will incur duplicates, NN exploits the methods: Laisser-faire, Propagate, Merge and Fine-grained Partitioning, to eliminate duplicates.

To sum up, because of the prohibitive precipitation cost and space overhead, index-based algorithms have serious limitations. It is much expensive for bitmap algorithm to perform preconstruction and computation of the skyline results. The bit-vector length of each encoded tulle in bitmap algorithm equals the sum of cardinalities of all attributes. If some attributes have high cardinalities, the space overhead for storage is large. Besides, for checking whether each tulle in table is a part of skyline, bitmap algorithm has to retrieve the corresponding bit-transposed files involving all tulles. For tree-based algorithms, In the size of skyline criteria is typically small, the combination of the attributes over which the queries are posed can be quite large. Given a table with M attributes and skyline criteria involves not more below the threshold in size, LD&C directly computes the skyline results of the partition. At last, LD&C invokes DD&C to merge the skyline results of partitions. FLET first determines a virtual tulle t1 before execution. During scanning the input, any tulle dominated by t1 is discarded directly. If there occurs a tulle t2 that dominates t1, t1 is replaced by t2 and after scanning algorithm LESS to improve SFS. Similar to SFS, LESS first sorts the table in certain order compatible with the skyline criteria. LESS integrates sorting and skyline processing. It has all of SFS's benefits without additional disadvantages and consistently outperforms SFS. LESS also has BNL's advantages, but effectively none of its disadvantages. LESS does not need the bookkeeping overhead and it requires much less cost for sorting than SFS because many tulles are discarded by EF buffer. LESS is invulnerable to how the table is ordered originally.

Bartolini *et al*., develop SalSa algorithm based on SFS to exploit the sorting of a table to order tuples so that only a subset of table needs to be examined for computing skyline results. SalSa first sorts the table in certain order as in SFS. It denotes by U all unread tuples in table. Tepresents all tuples in the table. Each time a new tuple p is read from U, p is compared against the current skyline tuples in memory buffer as in BNL. SalSa makes use of a stop point pstop to check whether it can terminate reading tuples. When the current tuple retrieved from U is the update of pstop . It is guaranteed that SalSa terminates if all tuples in U are inserted into memory buffer, this might trigger dominated by pstop and memory buffer keeps the skyline results.The current skyline algorithms have to scan the entire table at least once to return the skyline results. There are many other skyline algorithms in different applications, such as personalized skyline [2], metric skyline [10], distributed skyline [11], [13], [21]. In this paper, we focus on skyline query on a standalone computer. The algorithm proposed in this paper combines the advantages of index- U based algorithms and generic algorithms andvercomEeheirdisadvantages.

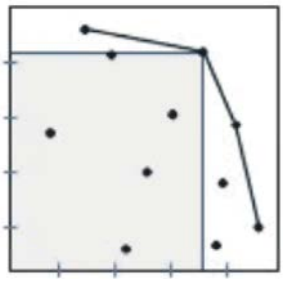

Fig. 1: Skylinre restaurants

Problem Statement: Given tuple number n in table T and size m of skyline criteria, the expected number s of skyline results under component independence is s ¼ Hm_1; n, here Hm;n is the mth order harmonic of n. For any n > 0,H0;n ¼ 1. For anym > 0, Hm;0 ¼ 0. For any n > 0 and m Hm;n is inductively defined as: Hm;n ¼ Pn i¼1Hm_1;I i, Hm;n can be approximated as: While skyline query processing in a constraint-free space has been well studied [4, 5, 25, 21, 17, 19, 2] to the best of our knowledge, this paper is the first effort on relative skyline query processing in a constrained spaceSkyline is an important operation in many applications to return a set of interesting points from a potentially huge data are maintained such that the expansion can continue froma

previous state. In addition, as described in whenspace. A subset of attributes is designated as skyline criteria, on which the dominance relationship between tuples is defined. Given two tuples p and q in a table, p dominates q if, among skyline criteria, p is not larger than q in allattributes and strictly smaller than q in at least one attribute. Skyline finds all tuples that are not dominated by any other tuples. Skyline algorithms, such as Bitmap, NN, BBS, SUBSKY and ZBtree, utilize indexes to reduce the explored data space and return skyline results. Because of the prohibitive pre-computation cost and space overhead to cover the attributes involved in skyline on big data, index-based algorithms have serious limitations and the used indexes can only be built on a small and selective set of attribute combinations. In the next section, we report the results of our experiment tal evaluation.

Experimental Setup: All methods proposed in this paper wereimplemented using Microsoft Visual C++ V6.0, including(1) the improved sort-merge join based skyline methods by(i) using R-tree MBRs (noted as SMJS1), (ii) com-paring with joined skylines during the join process (notedas SMJS2); (2) the block nested-loop join based skylinemethod (noted as NLJS); and (3) the naive-based skyline algorithm (noted as Naive). Using the data generator provided by www.tpc.org, we generated several types of TPCD benchmark tables: Customer(each tuple has 44 bytes), Order((each tuple has 84 bytes)) and Part (each tuple has60 bytes) 5 . For each type of table, we generated data setswith different sizes (from 10, 000 to 1, 000, 000 tuples). respectively, with respect to different cardinalities.

In both cases, the size differences are evident and become larger with more cardinalities and more participating relations. The computation of joined skylines with aggregate constraints uses less time due to the smaller size of input tablesafter the aggregation. The run time comparisons of differentmethods for computing joined skylines with/without aggregate constraints with respect to different sizes of the joinedtable C O (with totally 10 descriptive attributes) SUM" aggregate as in both cases, our proposed methods run much faster than Naive method. In particular, SMJS2 finishes first and SMJS1 takes less time than.

As mentioned earlier, in the conventional setting of static data, here is a large body of work for both single-source skyline processing [2, 3, 1] and multiple source skyline-join processing [12, 16]. These methods assume that the data is unchanging during query execution and focus on computing a single skyline rather than continuously tracking skyline changes. Recently, several algorithms have been developed to track skyline changes over data streams. These methods continuously monitor the changes in the skyline according to the arrival of new tuples and expiration of old ones. Data stream skyline processing under the sliding windowmodel is addressed in [7] and [14]. An important issue that needs to be addressed here is the expiration of sky line objects. To tackle this issue, Tao et al. present the Eager algorithm [14] that employs an event list, while Lin et al. propose a method (StabSky) that leverages dominance graphs [7]. Both these methods memorize the relationshipbetween a current skyline object and its successor(s). Once skyline objects expire, their successor(s) can be presented as the updated skyline without any added computation. The above-mentioned approaches focus on skyline queries in which the skyline attributes belong to a single stream,thus rendering them inapplicable to the problem of comput ing skyline-joins over multiple streams. In this paper, we demonstrate the novel Layered Skyline-window-Join (LSJ)operator; this operator is ?rst of its kind for answering skyline-window-join (SWJ) queries over data streams.

NLJS. The relationship between run time and the dimensionality of the joined table. are respectively. The joined table is obtained from the joining of Oof 500,000 records and P of 100,000 records. Each time we choose 2, 3, 4 and 5 dimensions per table respectivelyto participate join operation, thus the joined table has thedimensionality of 4, 6, 8 and 10 respectively. It shows the same ranks of run time as the test of run time with respect to different sizes of joined tables.

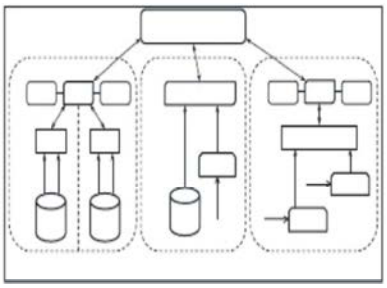

Fig. Skyline framework

Evaluate Space Efficiency: We evaluate the space usage in terms of the number of uncertain elements kept in SN,q against different settings. As this number may change as the window slides, we record the maximal value over the whole stream. Meanwhile, we also keep the maximal number of SKYN,q.

The First set of experiments is reported in Figure 4 where 4 datasets are used: Inde-Uniform (Independent distribution for spatial locations and Uniform distribution for occurrence probability values), Anti-Uniform, Anti-Normal and Stock Uniform. We record the maximum sizes of SN,q and SKYN,q .It is shown that very small portion of the 2-dimensional dataset needs to be kept. Although this proportion increases with the dimensionality rapidly, our algorithm can still achieve a 89% space saving even in the worst case, 5 dimensional anti correlated data. Size of SKYN,q is much smaller than that of candidates. Since the anti-correlated dataset is the mostchallenging, it will be employed as the default dataset in therespectively, with respect to different cardinalities. In both cases, the size differences are evident and become largerwith more cardinalities and more participating relations.

The computation of joined skylines with aggregate constraints uses less time due to the smaller size of input tables after the aggregation. The run time comparisons of different methods for computing joined skylines with/without aggregate constraints with respect to different sizes of the joined table C O (with totally 10 descriptive attributes) are depicted and. Here, we typically test the “SUM” aggregate as mentioned . In both cases, our proposed methods run much faster than Naive method. In particular, SMJS2 finishes first and SMJS1 takes less time than NLJS. The relationship between run time and the dimensionality of the joined table are illustrated respectively. The joined table is obtained from the joining of Oof 500,000 records and P of 100,000 records. Each time we choose 2, 3, 4 and 5 dimensions per table respectively to participate join operation, thus the joined table has the dimensionality of 4, 6, 8 and 10 respectively. It shows the same ranks of run time as the test of run time with respect to dfferentsizes of joined tables.

The study on skyline queries by considering relative network distances to multiple query points at the same time. SSPL is proven to be instance optimal in terms of the network search space over all algorithms where network distances are computed by expanding the searching region from query points without using pre-computed distance information. Our experiments confirmed that SSPL has the best performance consistently for various test settings. The path distance pruning approach, based on which SSPL is designed, can be applied to benefit other types of road network queries where network distance comparisonn is needed.

RESULT

In this paper, the efficient result of had been obtained in using the nested loop and divide and conquer method. The SSPL algorithms have been proposedfor processing multi-source relative skyline queries in road networks. It is not only the first effort to process relative skyline queries in road networks, but also the first study on skyline queries by considering relative network distances to multiple query points at the same time. SSPL is proven to be instance optimal in terms of the network search space over all algorithms where network distances are computed by expanding the searching region from query points without using pre-computed distance information. Our experiments confirmed that SSPL has the best performance consistently for various test settings. The path distance pruning approach, based on which SSPL is designed, can be applied to benefit other types of road network queries where network distance comparisonn is needed.

CONCLUSION

In this paper, we consider the problem of processing skyline query on big data. It is analyzed that the current skyline algorithms cannot perform skyline on big data efficiently.The purpose of skyline operator incorporating state-of-the-art join methods into skylinecomputation. The experiments on TPC-D datasets demonstrate the effciency and scalability of the proposed methods. We believe that this research does not only meaningfully extend the skyline operator to the multi-relationaldatabase systems, but also indicate the interesting topicssuch as joined skylines in the case of updated data and other types of aggregates.

REFERENCES

1. Bartolini, I., P. Ciaccia and M. Patella, 2008. Efficient Sort-Based Skyline Evaluation, ACM Trans. Database Systems, 33(4): 31(1-31): 49.
2. Bartolini, I., Z. Zhang and D. Papadias, 2011. Collaborative Filteringwith Personalized Skylines, IEEE Trans. Knowledge Data Eng., 23(2): 190-203.
3. Bentley, J.L., H.T. Kung, M. Schkolnick and C.D. Thompson, 1978. On the Average Number of Maxima in a Set of Vectors and Applications, J. ACM, 25(4): 536-543.

4. Feiyi Wang, Fengmin Gong, Chandramouli Sargor, Katerina Goseva-Popstojanova, Kishor Trivedi, Frank Jou, 2012. Adaptive Intrusion Detection: a Data Mining Approach.
5. John McHugh, Alan Christie and Julia Allen, 1999. Software Engineering Institute, CERT Coordination Center" The Role of Intrusion Detection Systems.
6. Joomla cms, http://www.joomla.org/, 2011. Christian Bockermann, Martin Apel, Michael Meier Technische University¨at Dortmund 44221 Dortmund, Germany, Learning SQL for Database Intrusion Detection using Context-Sensitive Modelling", 2011.
7. Kun Bai, Hai Wang and Peng Liu, 2011. The School of Information Science and Technology, Pennsylvania State University," Towards Database Firewalls.
8., Mexing Le, Angelos Starou, Member, IEEE and Breit Byung Hoon Kang, 2011. Member, IEEE "Double Guard: Detecting Intrusions in Web Applications IEEE Transactions On Dependable and Secure Computing.
9. SANS, 2011. The Top Cyber Security Risks," http://www.sans.org/ to-cyber-security-risks/,
10. National Vulnerability Database, 2011. "Vulnerability Summary for 0-4332," http://web.nvd.nist.gov/ view/vuln/detail?vulnId= CVE-2010-4332,
11. Autobench, http://www.xenoclast.org/autobench/, 2011.
12. "Common Vulnerabilities and Exposures," http://www.cve.mitre. org/, 2011.
13. greensql, http://www.greensql.net/, 2011.
14. http://www.hpl.hp.com/research/linux/httperf/, 2011.
15. http://www.acme.com/software/http_load/, 2011.
16. Linux-vserver, http://linux-vserver.org/, 2011.
17. Metasploit, http://www.metasploit.com/, 2011.
18. Papadiase, D., Y. Tao, G. Fu and B. Seegar, 2005. Progressive Skyline Computation In Database Systems', ACM Trans. Database Systems, 30(1): 42-81.
19. Gray, J. and P.J. Shenoy, 2000. ' Rules of thumb in data engineering ", Proc. 16th int'l Conf.Data Eng.(IDC'00), pp: 3-1`2.
20. Lee, K.C., B. Zheng, H. Li and Y. Yian, 2007. Z-Sky: An Efficient Top-k Aggregation Of Ranked Inputs", ACM Trans.Database ZSystems, 32(3): 19-2.
21 Lee, K.C., W.C. Lee, B. Zheng and H. Li, 2010. An efficient skyline query processing framework based on z order", the VLDB, 19(3): 33-62.
22. Chomicki, J., P. Godfrey, J. Gryz and D. Liang, 2003. Skyline with presorting," Proc.19th Int l Conf,DataEng.(ICDE.03), pp: 717-719.
23. Gadfrey, P., 2004. 'Skyline Cardinality for Relational Processing",Foundations of Information and Knowledge System", springer berlin/Heidelberg, 2942: 78-79.
25. Kossmann, D., F. Ramsak and S. Rost, "Shooting Star in the Sky: An Online Algorithm for Skyline Queries", Proc. 28th int'l Conf. Very Large Data bases (VLDB, 02).
26. Zhu, L., Y. Tao and S. Zhou, mar, 2009. Distributed Skyline Retrival with low Bandwith Consumption,"IEEE Trans. Knowlegde Data Eng., 21(3): 384-400.