HIGH DIMENSIONAL DATA COMPUTATION USING ZINC EXPERIMENTS

K Venkata Raju[1], A Vijaya Kumar[2], MD Mounika[3],K Sandeep[4]

AssocProfessor, Department of Computer Science & Engineering, K L University, Guntur, Andhra Pradesh, India.[1]
AsstProfessor,Department of Computer Science & Engineering, K L University, Guntur, Andhra Pradesh, India.[2]
Student,Department of Computer Science & Engineering, K L University, Guntur, Andhra Pradesh, India.[3]
Student, Department of Computer Science & Engineering, K L University, Guntur, Andhra Pradesh, India.[4]

ABSTRACT

Skyline[1] is an important operation in many applications to return a set of interesting points from a potentially huge data space. Given a table, the operation finds all tuple's that are not dominated by any other tuple's. It is found that the existing algorithms cannot process skyline on big data efficiently. This paper presents a novel skyline algorithm SSPL[1] on big data. SSPL utilizes sorted positional index lists which require low space overhead to reduce I/O cost significantly. We present a new indexing method named ZINC[2] (for Z-order indexing with Nested Code) that supports efficient skyline computation for data with both totally and partially ordered attribute domains. By combining the strengths of the Z-order indexing method with a novel nested encoding scheme to represent partial orders, ZINC[2] is able to encode partial orders of varying complexity in a concise manner while maintaining a good clustering of the PO domain values. Our experimental results have demonstrated that ZINC[2] outperforms the state-of-the-art TSS technique for various settings.

Index Terms: ZINC, SDC+, ZB-Tree, Skyline Computation.

INTRODUCTION

Data mining is one of the important step in KDD process (Knowledge Discovery and Database). It's the process of extracting data from huge data set.Data mining is about processing data and identifying patterns and trendsso that you can decide. Data mining principles have been around for many years, but, with the advent of *big data*, it is even more prevalent.Big data is caused the size of the information is very large. It is no longer enough to get relatively simple and straightforward statistics out of the system with large data sets.Sky line is one of the important operation in many applications to return important points from large database. Skyline has attracted extensive attention and many algorithms are proposed. A set of skyline algorithms[1], such as Bitmap, NN, BBS, SUBSKY, and ZBtree, utilize indexes to reduce the explored data space and return skyline results. For providing skyline computation process on each data set traditionally used index-based algorithms utilize the preconstructed data-structures to avoid scanning the entire data set. It preconstructsdata-structures with low space overhead. By the data-

IJAER

structures, the algorithm only involves a small part of table to return the skyline[4] results. Index-based algorithms have serious limitations and the used indexes can only be built on a small and selective set of attributecombinations. Nowadays, big data is used commonly in scientific research and business application.

People will expect to get results quickly and they do not want to wait for several hours. For that we propose a novel skyline algorithm on big data, skyline with sorted positional index lists (SSPL), to return skyline results efficiently[1]. The algorithm utilizes the preconstructeddata-structures which require low space overhead toreduce I/O cost significantly. SSPL consists of two phases: obtaining the candidate positional indexes (phase 1) and retrieving the skyline results (phase 2). In phase 1, SSPL first retrieves the sorted positional index lists {L1; L2; . . . ; Lm} involved by skyline criteria {A1;A2; . . .;Am} in a round-robin fashion. A mathematical analysis is proposed to compute scan depth d of the lists in phase 1. It is guaranteed that the candidate positional indexes corresponding to the skyline results are contained in the first d elements in {L1; L2; . . . ; Lm}. In phase 1, SSPL performs pruning on any candidate positional index retrieved from {L1; L2; . . . ; Lm} to discard the candidate whose corresponding tuple is not skyline result. This paper proposes general rules and mathematical analysis for pruning operation. Phase 1 ends when there is a candidate positional index seen in all lists of {L1; L2; . . . ; Lm}. In phase 2, SSPL exploits the obtained candidate positional indexes to compute skyline results by a selective and sequential scan on the table. At first glance, the sorted positional index lists for SSPL are similar to the sorted column files and. However, the most significant idea for SSPL is its pruning operation. Unlike the sorted column files which are used to support sorted retrievalmainly, the sorted positional index lists are the data structuresto facilitate pruning and reduce the candidate tuples significantly. Although SSPL is an approximate method to obtain skyline results[4], its probability of correctness is extremely high. The extensive experiments areconducted on two sets of terabyte synthetic data and a set of gigabyte real data, and the experimental results show that compared to the existing algorithms; SSPL involves up to six orders of magnitude fewer tuples, and obtains up to three orders of magnitude speedup.Skyline Sorted Positional Index List algorithm have serious limitations and it fails to process the sequential execution in data sets.

Consider the procedure of SSPL we have to extract the efficient features of big datawith computation and other features like data assessment we have to introduce Z-order data set comparison for efficient big data computation.

For example a set of data records D a skyline query returns the interesting subset of records of *D* that are not dominated (with respect to the attributes of *D*) by any records in *D*. A data record *r*1 is said to dominate another record *r*2 if *r*1 is at least as good as *r*2 on all attributes, and there exists at least one attribute here *r*1 is better than *r*2.

IJAER

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 (TO)*, where the best value for a domain is either its maximum or minimum value. However, in many applications, some of the attribute domains are *partially ordered (PO)*such as interval data (e.g. temporal intervals), type hierarchies, and set-valued domains, where two domain values can be incomparable. A number of recent research work has started to address the more general skyline computation problem where the data attributes can include a combination of TO and PO domains.

The first method that proposed for the more general skyline query problem is SDC+, which is an extension of BBS index method for totally ordered domains. SDC+ works 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.

A new index method has been proposed for computing skyline queries for TO domains called ZB-tree[3]. It has better performance than BBS. It is the extension of B+ -trees, 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 efficient data pruning and minimizes the number of dominance comparisons.

RELATED WORK

INDEX BASED ALGORITHM:

Index-based skyline algorithms utilize the preconstructed data-structures to avoid scanning the entire data set.

Tan[3]make use of bitmap to compute skyline of a table T(A1;A2; . . .;Ad). Given a tuple x = (x1; x2; …. .. ;xd) ∈ T, x is encoded as a b-bit bit-vector, b = $\sum_{i=1}^{d}$ki (ki is the cardinality of Ai). We assume that xi is the j_i^{th} smallest value in Ai, the ki-bit bit-vector representing xi is set as follows: bit 1 to bit j_i- 1 are set to 0, bit j_i to bit k_i are set to 1. The encoded table is stored as bit-transposed files , let BS_{ij} represent the bit file corresponding to the j^{th}bit in the i^{th} attribute Ai. It is given that a tuple x = (x1; x2; . . . ; xd) ∈ T and x_i is the (ji)th smallest value in Ai. Let A = $BS1_{j1}$&$BS2_{j2}$& . . . &BSd_{jd} where & represents the bitwise and operation. And let B = $BS1_{(j1-1)}$| $BS2_{(j2-1)}$| . . . | $BSd_{(jd-1)}$ where j represents the bitwise or operation. If there is more than a single one-bit in C = A&B, x is not a skyline tuple. Otherwise, x is a skyline tuple.

Kossmann propose NN algorithm to processskyline 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. Next, the data space is partitioned by the nearest neighbor to several subspaces. The

IJAER

subspaces that are not dominated by the nearest neighbor are inserted intoa to-do list. While the to-do list is not empty, NN removes one of the subspaces to perform the same process recursively. During the space partitioning, overlapping of the subspaces will incur duplicates, NN exploits themethods: Laisser-faire, Propagate, Merge and Fine-grained Partitioning[5], to eliminate duplicates.

THE SSPL ALGORITHM

This section first introduces the data-structures required by SSPL then describes the overview of theSSPL algorithm next shows how to perform pruning followed that presents the implementation and analysis of SSPL and finally introduces how to extend SSPL to cover other cases .

Sorted Positional Index List

Given a table T, the positional index (PI) of $t \in T$ is i if t is the i^{th} tuple in T.We denote by T(i) the tuple in T with its PI = i, and byT(i)(j) the j^{th} attribute of T(i). The execution of SSPLrequires sorted positional index lists. Given a tableT(A1;A2; . . .;AM), we maintain a sorted positional indexlistLj for each attribute Aj($1 \leq j \leq M$). Lj keeps thepositional index information in T and is arranged inascending order of Aj,. That is $\forall i_{1,\, i2}(1 \leq i_1 < i_2 < n)$;

The sorted positional index lists are constructed as follows: First, table T is kept as a set of column files CS =f{C1; C2; . . . ; CM} . The schema of each column file Cj isCj(PI;Aj) ($1 \leq j \leq M$), here PI represents the positionalindex of the tuple in T and Aj is the corresponding attributevalue of T(PI). Then, each column file Cj is sorted inascending order according to Aj. Because SSPL onlyinvolves PI field of column files, the PI values in columnfiles are retained and kept as sorted positional index lists. Herewe compare the sorted positional index lists with theindexes used in tree-based algorithms briefly. SSPL constructsa sorted positional index list for each attribute, onlyM lists are needed. SSPL reduces the space overhead of data structures from exponential to linear.

More importantly,the processing of SSPL[1] can cover all attributes, ratherthan limited to a small and selective set of attributecombinations in tree-based algorithms.It is noted that read/append-only is an importantcharacteristic of big data, and update is performed inperiodic and batch mode. Therefore, sorted positional indexlists are worth pre-computing and will be used repeatedlyuntil the next update. And when update operation begins,sorted positional index lists can be updated by merging thecorresponding column files in big old data and relativelymuch smaller new data.

Skyline query processing has involved a lot of research. In this section we will assess the work that handles data with both TO and PO domains.

IJAER

Importance to ZB-tree, the state-of-the-art approach for data with only TO domains was BBS. The main goal is to map each PO attribute into an approximate representation consisting of a pair of TO attributes. The transformed data is then indexed using BBS. Due to the approximate representation, this approach requires post-processing of false positive skylines. Although this limitation is alleviated with some optimization technique to allow partial progressive skyline computation, the overhead of dominance comparisons can be high.

The another state –of –the –art approach is TSS. This approach is based on BBs. Unlike the BBS approach, TSS uses a presise representation by mapping each PO domain value into an ordinal number with respect to a topological orderingof the PO domain values and a set of interval values.

Another approach is skyline computation[1] for continuous data streaming with PO domains.The main goal is efficient skyline maintenance for streaming non-indexed data which is different from indexed based approach for static data.

Another recent approach is dynamic skyline querries which are sky line querries where the user preferences are specified at run time. Data with categorical attributes, the partial orders representing the user's value preferences for such attributes are given as part of the input skyline query.

BACKGROUND WORD

Given a table T(A1;A2; . . .;AM),∀ t ϵ T, let us denote by t[j] the jth attribute Aj of t. Without loss of generality,let a subset of attributes ASskyline={A1;A2; . . .;Am} be skyline criteria[1], and the dominance relationship between tuples is defined on ASskyline. For clarity, we assume that min condition only is used for skyline computation. However, the algorithm here can be extended to process any combination of condition (min or max).

Skyline query. Given a table T, skyline query returns asubset SKY (T) of T, in which ∀ t1 ϵ SKY (T), ∄ t2 ϵ T, t2 < t1.Given tuple number n in table T and size m of skylinecriteria, the expected number s of skyline results undercomponent independence is known. s ¼=Hm_1;n,here Hm;n is the mth order harmonic of n. For any n > 0,H0;n =1. For anym> 0, Hm;0 = 0. For any n > 0 andm > 0, Hm;nis inductively define as According to the computation formula of Hm;n, it isfound that the number of skyline results does not changesignificantly as the tuple number increases, while it is very sensitive to the size of skyline criteria. For example, given m= 3, when n increases from 10^5 to 10^9, s changes from 66 to 214. Given n = 10^9, when m increases from 2 to 5, s changes from 20 to 7,684. Although the absolute number of skyline results is large, its proportion among all tuples is rather small. For example, given m =5 and n = 10^9,s/n= 7.684×10^{-6}.Given tuple number n in table T and size m of skylinecriteria, the expected number s of

IJAER

skyline results undercomponent independence is known. s=Hm-1;n,hereHm;n is the mth order harmonic of n. For any n > 0,H0;n =1.

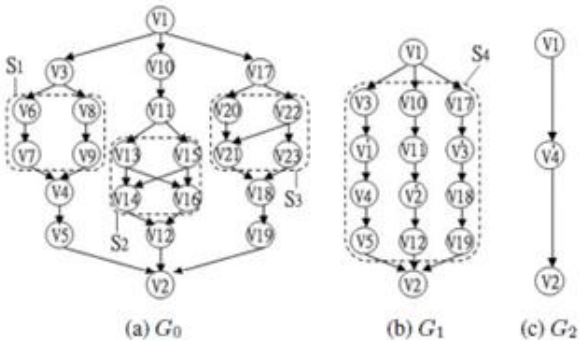

Figure 1: Partial order reduction process generation in Z-order datasets.

We represent a partial order by a directed graph *G* = (*V;E*),Where*V* and *E* denote, respectively, the set of vertices and edges in *G* such that given *v; v0 2 V* , *v* dominates *v0* if there is a directedpath in *G* from *v* to *v 0*. Given a node *v 2 V* , we use *parent*(*v*) (resp., *child*(*v*)) to denote the set of parent (resp., child) nodes of *v* in *G*. A node *v* in *G* is classified as a minimal node if *parent*(*v*) =Θ; and it is classified as a maximal node if *child*(*v*) = Θ;. We use *min*(*G*) and *max*(*G*) to denote, respectively, the set of minimal nodes and maximal nodes of *G*.

Given a partial order *G*0, the key idea behind nested encodingis to view *G*0 as being organized into nested layers of partial orders, denoted by *G*0 *! G*1 _ _ _ *!Gn*1*!Gn*, whereeach*Gi*is nested within a simpler partial order *Gi*+1, with the lastpartial order *Gn*being a total order. As an example, consider thepartial order *G*0 shown in Fig. 2, where *G*0 can be viewed as beingnested within the partial order *G*1 which is derived from *G*0by replacing three subsets of nodes *S*1 = *fv*6*; v*7*; v*8*; v*9*g*, *S*2 = *{v*13*; v*14*; v*15*; v*16}and*S*3 = *{v*20*; v*21*; v*22*; v*23*}* in *G*0 by three new nodes *v 0* 1, *v0* 2 and *v 0* 3, respectively, in *G*11. *G*1 in turn can be viewed as being nested within the total order *G*2 which is derived from*G*1 by replacing the subset of nodes *S*4 = *{v*3*; v 01; v*4*; v*5*; v*1 *v*11*; v02; v*12*; v*17*; v*03*; v*18*; v*1*}g* by one new node *v0*4 in *G*2. We refer to the new nodes *v*1, *v02*, *v0*3 and *v0*4 as *virtual nodes*; and eachvirtual node *v0j* in *Gi*+1 is said to *contain* each of the nodes in *Sj*that *v0j* replaces. By viewing *G*0 in this way, each node in *G*0 canbe encoded as a sequence of encodings based on the nested nodecontainments within virtual nodes.

PARTIALORDERREDUCTIONALGORITHM

Given an input partial order[10]*Gi*, algorithm PO-Reduce operates as follows: Let *S* = *{S*1*; _ _ _ Sk}* be the collection of regular regions in *Gi*; (2) If *S* is empty, then let *S* = *fS*1*g*, where *S*1 is an

irregular region in *Gi*that has the smallest size (in terms of thenumber of nodes) among all the irregular regions in*Gi*. (3) Create a new partial order *Gi*+1 from *Gi*as follows. First, initialize *Gi*+1 to be *Gi*. For each region *Sj*in *S*, replace *Sj*in *Gi*+1 with a virtualnode*vj*such that *parent*(*v0j*) = *parent*(*v*) with *v 2 min*(*Sj*) and *child*(*v0j*) = *child*(*v*) with *v 2 max*(*Sj*). (4) If *Gi*+1 is a total order, then the algorithm terminates; otherwise, invoke the PO-Reduce algorithm with *Gi*+1 as input.

When a node *v* in a region *R* is being replaced by a virtual node *v0*, we say that *v* is *contained in v0* (or *v0 contains v*), denoted by *v R! v0*. Clearly, the node containment can be nested; for example, if *v* is contained in *v0*, and *v0* is in turn contained in *v00*, then *v* is also contained in *v00*. Given an input partial order *G*0, we define the *depth* of a node *v* in *G*0 to be the number of virtual nodes that contain *v* in the reduction sequence computed by algorithm PO-Reduce. As an example, consider the value *v*6 in Fig. 2 and let *R*0 = *fv*6*; v*7*; v*8*; v*9*g* and *R*1 = *fv*3*; v*1*; v*4*; v*5*; v*10*; v*11*; v0*2 *v*12*; v*17*; v0*3*; v*18*; v*19*g*.

Thus, given an input partial order *G*0, algorithm PO-Reduceoutputs[10] the following: (1) the partial order reduction sequence,*G*0*! G*1 _ _ _ *!Gn*1*!Gn*, where *Gnis* a totalorder; and (2) the node containment sequence for each node in *G*0. If a node *v*0 in *G*0 has a depth of *k*, we can represent the node containment sequence for *v*0 by *v*0*R!*0*v*1_ _ _ *R!k*1 *vk*, where each *vi* is contained in the region *Ri, i 2* [0*; k*).

PERFORMATION ANALYSIS

To evaluate the performance of our proposed ZINC, we conducted an extensive set of experiments to compare ZINC[2] against three competing methods: TSS and the two basic extensions of ZB-tree, namely, TSS+ZB and CHE+ZB. Our experimental results show that ZINC[2] outperforms the other three competing methods. Given that both TSS+ZB and CHE+ZB are also based on ZB-tree, the superior performance of ZINC demonstrates the effectiveness of our proposed NE encoding for PO domains.

Algorithms: We consider two variants of the main competingmethod, TSS: an unoptimized variant of TSS (denoted by TSS) and an optimized variant of TSS (denoted by TSS-opt). In TSS, the set of intervals associated with each data / index entry's PO value are stored explicitly with the entry, while in TSS-opt, the intervals associated with an entry are retrieved from a separate precomputed structure.

To compare the effectiveness of our proposed nested encoding scheme, we also introduced two variants of ZB-tree that are based on using different schemes to encode PO domains. The first variant, TSS+ZB, combines the TSS encoding scheme with the ZB-tree method. Each PO domain value *vp*of a data point is encoded into a bitstring based on its ordinal value *vt*in a topological sorting of the PO domain values. The inclusion of *vt*in the derivation of the data

IJAER

point's Z-address is important to ensure ZB-tree's monotonicity property. Each leaf node entry in TSS+ZB stores a data point *p* together with the interval set representation of each of*p*'s PO attribute values. In each internal node entry of TSS+ZB, besides storing the *minpt*and *maxpt*of the corresponding RZ-region (similar to what is done in ZB-tree), for each PO attribute *A*, a merged interval set for *A* is also stored which is the union of the interval sets for attribute *A* of the covered data points. In TSS+ZB, region-based dominance test is applied as follows: if (1) the Z-address of an intermediate skyline point *pi* dominates *minpt*of an internal node entry *ej*, and (2) the interval set of *pi* subsumes the
interval set of *ej*w.r.t. every PO dimension, then the region represented by *ej*is dominated by *pi* and is pruned from consideration.

Synthetic datasets: We generated three types of synthetic data-sets according to the methodology in. For TO domains, we used the same data generator as [8] to generate synthetic datasets with different distributions. For PO domains, we generated DAGs by varying three parameters to control their size and complexity: *height* (*h*), *node density* (*nd*), and *edge density* (*ed*), where *h 2 Z+*, *nd; ed 2* [0*;* 1]. Each value of a PO domain corresponds to a node in DAG and the dominating relationship between two values is determined by the existence of a directed path between them. Given *h*, *nd*, and *ed*, a DAG is generated as follows. First, a DAGis constructed to represent a poset for the powers et of a set of *h* elements ordered by subset containment; thus, the DAG has 2*h* nodes.Next, (1 - *nd*) *100% of the nodes (along with incident edges)are randomly removed from the DAG, followed by randomly removing (1 -*ed*) *100% of the remaining edges such that the resultant DAG is a single connected component with a height of *h*. Following the approach in [8], all the PO domains for a dataset are based on the same DAG. Table 2 shows the parameters and their values used for generating the synthetic datasets, where the first value shown for each parameter is its default value. In this section,default parameter values are used unless stated otherwise.

Real dataset: We used a real dataset on movie ratings that isderived from two data sources, *Netflix* and *MovieLens*. Netflixcontains more than 100 million movie ratings submitted by more than 480 thousand users on 17770 movies during the period from 1999 to 2005. MovieLens contains more than 1 million ratings submitted by more than 6040 users on 3900 movies. Both these data sources use the same rating scale from 0 to 5 with a higher rating value indicating a more preferred movie. Our dataset consists of the ratings for 3098 of the movies that are common to both data sources.

We derived a PO attribute, named *movie preference*, for the 3098 movies as follows: a movie *mi* dominates another movie *mj*iff the average rating of *mi* in one data source is higher than that of *mj*, and the average rating of *mi* in the other data source is at least as high as that of *mj*. We also derived two TO attributes for each movie, named *average rating* and *number of ratings*, which

IJAER

represent, respectively, the movie's average rating (each value is between 0.00 and 5.00) and the total number of ratings that it has received over the two data sources. The number of distinct values for these two TO domains are 501 and 219800, respectively. For each of the TO domains, a higher attribute value is preferred.

CONCLUSION

This paper presents a novel skyline algorithm SSPL on big data. SSPL utilizes sorted positional index lists which require low space overhead to reduce I/O cost significantly. We present a new indexing method named ZINC (for Z-order indexing with Nested Code) that supports efficient skyline computation for data with both totally and partially ordered attribute domains. By combining the strengths of the Z-order indexing method with a novel nested encoding scheme to represent partial orders, ZINC is able to encode partial orders of varying complexity in a concise manner while maintaining a good clustering of the PO domain values. Our experimental results have demonstrated that ZINC[2] outperforms the state-of-the-art TSS technique for various settings.

REFERENCES

[1]Xixian Han, Jianzhong Li, "Efficient Skyline Computation on Big Data", IEEE Transactions On Knowledge And Data Engineering, Vol. 25, No. 11, November 2013.

[2] Bin Liu Chee Yong Chan, "ZINC: Efficient Indexing for Skyline Computation", The 37th International Conference on Very Large Data Bases,
August 29th September 3rd 2011, Seattle, Washington.*Proceedings of the VLDB Endowment,* Vol. 4, No. 3Copyright 2010 VLDB Endowment 21508097/10/12... $ 10.00.

[3] C.-Y. Chan, H.V. Jagadish, K.-L.Tan, A.K.H. Tung, and Z. Zhang, "Finding K-Dominant Skylines in High Dimensional Space," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), pp. 503-514, 2006.

[4] L. Chen and X. Lian, "Efficient Processing of Metric SkylineQueries," IEEE Trans. Knowledge Data Eng., vol. 21, no. 3, pp. 351-365, Mar. 2009.

[5] M.Gibas, G. Canahuate, and H. Ferhatosmanoglu, "Online IndexRecommendations for High-Dimensional Databases Using QueryWorkloads," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 2,pp. 246-260, Feb. 2008.

[6] P. Godfrey, "Skyline Cardinality for Relational Processing,"Foundations of Information and Knowledge Systems, vol. 2942,pp. 78-97, Springer Berlin/Heidelberg, 2004.

[7] P. Godfrey, R. Shipley, and J. Gryz, "Algorithms and Analyses forMaximal Vector Computation," The VLDB J., vol. 16, no. 1, pp. 5-28, 2007.

[8] J. Gray and P.J. Shenoy, "Rules of Thumb in Data Engineering,"Proc. 16th Int'l Conf. Data Eng. (ICDE '00), pp. 3-12, 2000.

IJAER

[9] K. Hose and A. Vlachou, "A Survey of Skyline Processing inHighly Distributed Environments," The VLDB J., vol. 21, no. 3,pp. 359-384, 2012.

[10] Y. Fang and C. Y. Chan. Efficient skyline maintenance for streamingdata with partially-ordered domains. In *DASFAA*, pages 322–336,2010.

IJAER