Sequence Join

SONG Xiaomei
School of Software, Tsinghua University,
Beijing, China
YONYOU Software Co., Ltd.,
Beijing, China
songxm@lreis.ac.cn

ZENG Xiaoqing
Changsha University of Science &Technology
Changsha, China
YONYOU Software Co., Ltd.,
Beijing, China
zengxqa@ufida.com.cn

YE Xiaojun
School of Software, Tsinghua University,
Beijing, China
liulm@tsinghua.edu.cn

XIE Dong
YONYOU Software Co., Ltd.,
Beijing, China
xd@yonyou.com

Abstract—This article put forward a new join technology for tables of relational database, named sequence join. Sequence join make a tuplematch other one tuple according to the disk address. The technology of the specific method can help solve these problems: the number of columns in a relational table is too large and exceeds the maximum number of columns that a relational database can support; after a table is longitudinally divided into several sub-tables by columns, a lot of time need to be spent in reading this data using traditional joining techniques. This article described the implementation of sequence join in detail and did three sets of experiments to prove the effectiveness and efficiency of sequence join at last.

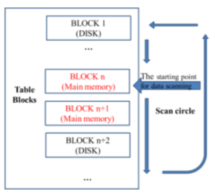
Key words-join, relational database

I.INTRODUCTION

In relational query processing, the join operator is one of the most time-consuming and data-intensive operations, many domestic and foreign Scholars join operation conducted a lot of research and discussion. According to the ANSI standard, SQL JOIN has given five types: INNER, FULL OUTER, LEFT OUTER, RIGHT OUTER and CROSS. These types of joins are mainly implemented by Nested Loops Join [1-4], Sort-Merge Join [5-7], Hash Join [8-10]. These join methods are based on two points: 1, thought of Cartesian product join, which is also named Nested Loops Join [11-12]; 2, unpredictable data storage address.

Every traditional join is based on Cartesian product join. Cartesian product join is realized and named CROSS JOIN. It takes a tuple from driving table to join a tuple from matching table sequentially and recursively. Traditional join operations have been implemented on Cartesian product join. They differ mainly in output format with different filters. Its time complexity is O(n^2) and indexing mechanism has been used to enhance the performance of query in the traditional relational database. Indexing mechanism really makes a tuple of data-driven tables skip some tuples from matching table data to improve join performance. However, when the amount of data is relatively large and we need to retrieve all the tuples, join operations are still performance bottleneck in queries [13].

Using traditional database, users cannot know the exact data storage address on disk. Traditional database always optimized query performance by avoiding more IO access. It does not always sequentially in turn access each tuple from the first tuple stored on disk. In 1980, Kim proposed a "rocking" thought [14]. It takes last block of last cycle as the first block of the next cycle in a query, thus avoiding a block I/O operation. In fact, as shown in Figure 1-a, when queries of other sessions may have taken some blocks in memory, the current query will take blocks loaded in memory as the starting point for data scanning. What’s more, updated tuples would have a different address in blocks. As shown in Figure 1-b, UPDATE operations are composed of DELETE and INSERT operations. The updated tuple would be inserted in a new space which may be on another block. The old tuple is not deleted from disk actually and just masked for the subsequent process. This is very helpful for rollback in a transaction.



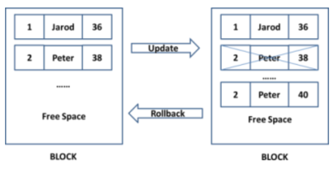
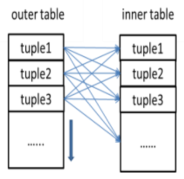


Figure 1-a Block scanningFigure 1-b Update and Rollback



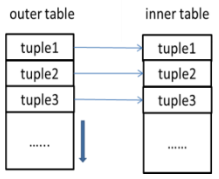


Figure 2. Nested Loops JoinFigure 3. Sequence Join

Applications of relational database used in the mainstream today have turned from OLTP to OLAP [15-16] such as decision support systems. Read-only operations get more and more attentions and read-only scenarios based relational database application are constantly increasing; on the other hand, a lot of data in the warehouse is never updated or modified after it is loaded by ETL (Extraction, Transformation and Loading) operations. So the traditional database optimization technology mentioned above is not suitable for the read-only scene. Make the technology disable, and we may find out other new techniques those are more suitable for data applications, such as the problem of too many columns in a table.

II.THOUGHT OF THE ALGORITHM

According to the above standpoint, this article presents a new database join technology – sequence join. This join technology can no longer be based on the Cartesian connection technology, and can only be used for read-only tables. According to the actual disk address, tuples from driving- and matching table are extracted one by one and joined together. This can avoid that a tuple from driving table scans all the tuples from the matching table. Its time complexity is only O(n), thereby increasing the speed of data extraction. In addition, for some aggregation operations against few columns it can reduce memory consumption and further improve computing speed.

At present, the existing SQL join methods are all based on the method of NLJ. Each tuple of driving table (outer table) matche seach tuple of matching table (inner table), for example, in Figure 1. Its time complexity is O(n2) and its efficiency is very low without doubt when the amount of data is large.

Sequence join method put forward in this article processes no longer based on the nest loop join method. If tuplesare inserted one by one and they would never be deleted or modified, and disable these optimization measures of concurrent read, sequence join takes the tuple of the driven table to match the same offset tuple of matching table, which is shown in Figure 3.

The physical table date is stored on disk randomly. But his logical structure can be formatted and controlled. This article uses the logical page to organize disk data. The logical structure of the disk data is organized as pages shown in Figure 4. Where:

1) Page Header Data: a block header of the table file, which contains some general information, such as starting and ending position of the free space, the starting and ending position of the item pointer, and the size of the remaining space, etc.
2) Linp: tuple items, which point to describe tuple information including the tuple location, size, etc.
3) Free space: it is the unallocated space (free space); the newly inserted tuple date would be allocated space sequentially from the tail of the free space team, while its corresponding Linp item would be allocated from the header of the free space team.
4) Tuple: it represents the actual data.

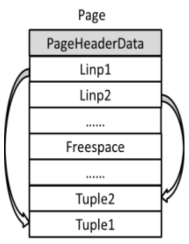


Figure 4. logical page

Sequence join process must be run in this premise: after logical page is loaded with data, it is no longer deleted or modified; disable the optimization for starting read from a memory page firstly. Driving table extract tuples sequentially from the file page; update the offset parameters (initial value is 0); extract tuples with the same offset matching from matching table; output the joining tuple. The specific operation process is shown as follows:

Step One: Define and initialize the global variable Outer Offset and Inner Offset (their values are set to 0); they are used to mark the tuple ID offset from driving table and matching table in the logical file blocks. Define and initialize the global variable Outer Tup Num and Inner Tup Num (their values are set to 0); they are used to mark the tuple number of current file page of driving table and matching table. If driving table successfully read the first logical file block. Turn to Step Two; otherwise, turn to Step Three.

Step Two: After reading the first logical file block, driving table takes the first logical file block as the current data page by the metadata information. Outer Offset's value is initialized to 0, Outer Tup Num get the number of tuples from Page Header Data structure. Turn to Step Four.

Step Three: Driving table sequentially scans logical file blocks and obtains the current page. From the Page Header Data information of last logical file block, the next logical file block is obtained and taken as the current data page of driving table. The value of Outer Offset is initialized to 0; Outer Tup Num is set as the number of tuples from the Page Header Data structure. Turn to Step Four.

Step Four: Determine whether Outer Offset is greater than Outer Tup Num. If it is, go to Step Three; otherwise continue. Driving table gets Outer Linp from the Page Header Data structure of the current data page. Outer Linp is the Linp structure and has the tuple ID offset whose value is Outer Offset. If Outer Linp is NULL which means the end of query, exit; other whiles, Outer Offset is plus 1 and process turn to Step Five.

Step Five: Driving table obtain stuple initial address and tuple length according to the structure of Outer Linp and finally extracts tuple data. If matching table reads the first logical file block; turn to Step Six; otherwise turn to Step Seven.

Step Six: Matching table gets the first logical file block through metadata information as the current data page matching table. The value of Inner Offset is initialized to 0; Inner Tup Num is set as the number of tuples from the Page Header Data structure. Turn to Step Eight.

Step Seven: Matching table sequentially scans logical file blocks and obtains the current page. From the Page Header Data information of last logical file block, the next logical file block is obtained and taken as the current data page of matching table. The value of Inner Offset is initialized to 0; Inner Tup Num is set as the number of tuples from the Page Header Data structure. Turn to Step Eight.

Step Eight: Determine whether Inner Offset is greater than Inner Tup Num. If it is, go to Step Seven; otherwise continue. Driving table gets Inner Linp from the Page Header Data structure of the current data page. Inner Linp is the Linp structure and has the tuple ID offset whose value is Inner Offset. If Inner Linp is NULL which means the end of this read circle, go to Step Three; other whiles, Inner Offset is plus 1 and process turn to Step Nine.

Step Nine: Driving table obtain stuple initial address and tuple length according to the structure of Outer Linp and finally extracts tuple data. A new tuple that consists of tuples of driving and matching table is formed and output. Go to Step Three.

III.COMPUTATIONAL EXPERIMENTS AND RESULTS

The thought of sequence join raised in section two has been implemented in Postgre SQL(Version 9.3.4). To test the performance of sequence join, we have carried out three sets of experiments which were comparative experiments between sequence join and inner join. All experiments were solved on a 2.53 GHz Pentium processor with 1.93 GB of RAM running with Windows XP as the operating system.

These three sets of experiments were designed based on different size of data whose numbers are 1thousand~ 10 thousands, 10 thousands~100 thousands and 1 million ~ 10 millions especially.

From the Figure 5a-c, we can find out : 1)the query time would be almost linear with respect to the he amount of data;2)sequence join is significantly faster than Inner Join; 3) With the increasing amount of data, the trend line slope of sequence join almost does not change, while the trend line slope of Inner Join becomes larger and larger.

So it is obvious that the sequence join is better than Inner Join from the point of query efficiency.

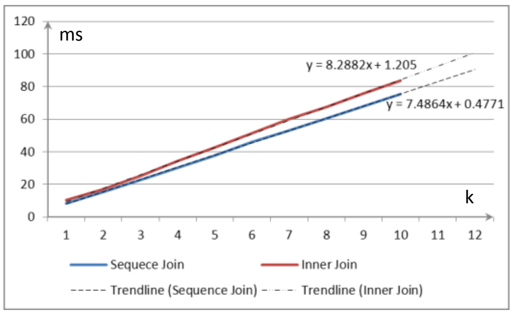


Figure 5-a.1 thousand ~ 10 thousands

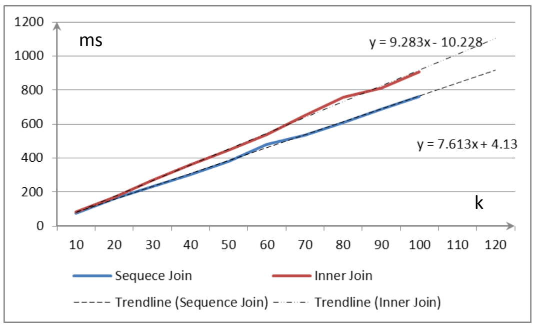


Figure 5-b.10 thousands ~ 100 thousands

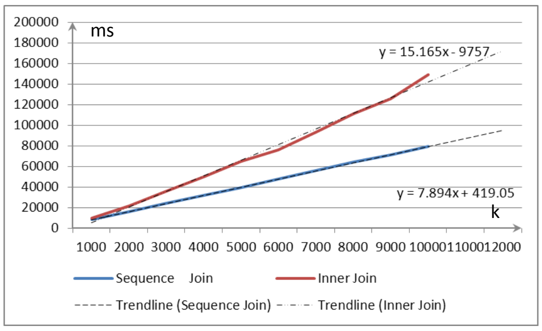


Figure 5-c.1 million ~ 10 millions

IV. Conclusions

The technology of sequence join based on read-only data, will make tuples of driving table and those of matching table sequentially join together according to offset tuple identifiers. First of all, sequence join technology no longer makes each tuple of driving table try to access all tuples from matching table, and only need to access the specified one tuple, which greatly reduces the time consumption and improves the efficiency of query. Secondly, sequence join technology supports vertical segmentation of a big table which has too many columns; it can overcome the table column number limit problem of the traditional database. Finally, the sequence join technology can ensure that common fields can be created into separate tables which can take only part of the data into memory to participate in the calculation such as aggregation computation. This also can be used to reduce the memory and CPU resources obviously.

Acknowledgement

This research is supported by the Beijing Postdoctoral Foundation under grant No. 2013ZZ-46. Its supports are gratefully acknowledged.

References

[1] Bornea, M.A.; Vassalos, V.; Kotidis, Y.; Deligiannakis, A. "Double Index NEsted-Loop Reactive Join for Result Rate Optimization", Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on, On page(s): 481 – 492

[2] Trifunovic K, Nuzman D, Cohen A, et al. Polyhedral-model guided loop-nest auto-vectorization[C]//Parallel Architectures and Compilation Techniques, 2009. PACT'09. 18th International Conference on. IEEE, 2009: 327-337.

[3] Zhou J. Nested Loop Join[M]//Encyclopedia of Database Systems. Springer US, 2009: 1895-1895.

[4] Minor M, Bergmann R, Görg S. Case-based adaptation of workflows[J]. Information Systems, 2014, 40: 142-152.

[5] Yang H, Dasdan A, Hsiao R L, et al. Map-reduce-merge: simplified relational data processing on large clusters[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007: 1029-1040.

[6] Albutiu M C, Kemper A, Neumann T. Massively parallel sort-merge joins in main memory multi-core database systems[J]. Proceedings of the VLDB Endowment, 2012, 5(10): 1064-1075.

[7] Liagouris J, Mamoulis N, Bouros P, et al. Efficient Management of Spatial RDF Data[R]. Technical Report TR-2014-02, CS Department, HKU, www. cs. hku. hk/research/techreps, 2014.

[8] Chen S, Ailamaki A, Gibbons P B, et al. Improving hash join performance through prefetching[J]. ACM Transactions on Database Systems (TODS), 2007, 32(3): 17.

[9] Blanas S, Li Y, Patel J M. Design and evaluation of main memory hash join algorithms for multi-core CPUs[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011: 37-48.

[10] Balkesen C, Teubner J, Alonso G, et al. Main-memory hash joins on multi-core CPUs: Tuning to the underlying hardware[C]//Data Engineering (ICDE), 2013 IEEE 29th International Conference on. IEEE, 2013: 362-373.

[11] Blasgen M W and Eswaran K P. Storage and access in relational databases. IBM Syst J, 1977, 16(4): 363~377

[12] ElMasri R and Navathe S. Fundamental of Database Systems. Benjamin /Cummings, Menlo Park, Calif, 1989. 542~553

[13] Wang L Z . Research on Hash Join Algorithm in DM Database[D]. Huazhong University of Science and Technology, 2012.

[14] Kim W. A new way to compute the product and join of relation. in: proceeding of SIGMOD. New York: ACM, 1980. 179~187

[15] Russakovsky A. Hopping over Big Data: Accelerating Ad-hoc OLAP Queries with Grasshopper Algorithms[J]. arXiv preprint arXiv:1310.0141, 2013.

[16] Do N. Application of OLAP to a PDM database for interactive performance evaluation of in-progress product development[J]. Computers in Industry, 2014, 65(4): 636-645.