**INEQUALITY JOINS**

Inequality joins report with their comparative analysis

**Group 27**

02/27/2016

**Table of Contents**

[**ABSTRACT** 3](#_Toc446507717)

[**INTRODUCTION** 3](#_Toc446507718)

[**INEQUALITY SELF JOIN** 3](#_Toc446507719)

[**INEQUALITY JOIN** 4](#_Toc446507720)

[**OPTIMIZATION** 4](#_Toc446507721)

[**CONCLUSION** 4](#_Toc446507722)

[**BIBLIOGRAPHY** 4](#_Toc446507723)

# **ABSTRACT**

Minibase is a database management system intended for educational use. It has a parser, optimizer, buffer pool manager, storage mechanisms and a disk space management system. The goal is not just to have a functional DBMS, but to have a DBMS where the individual component can be studied and implemented. Here MiniBASE Java is used to experiment with join operations – a fundamental database operation that requires both memory and time for its successful completion.

Join operation combines related tuples from same or different relations on different attributes schemes. Inequality joins, which join relational tables on inequality conditions are used in various applications. In this report, we have introduced fast inequality join algorithms with one or two predicate conditions. We put columns to be joined in sorted arrays and we use permutation arrays to encode positions of tuples in one sorted array w.r.t. the other sorted array. We have implemented a centralized version of these algorithms on top of PostgreSQL, and a distributed version on top of Spark SQL. We have compared against well-known optimization techniques for inequality joins and show that our solution is more scalable and several orders of magnitude faster. and compare its performance with sort-merge join and nested loop join with inequality predicates. It is evident from results that IEJoin outperforms other techniques for both single and two predicate inequality joins.

# **INTRODUCTION**

Joins are one of the most crucial operators in a database. Based on the join predicates and the algorithm used for the join, the result generation may range from a few seconds to few days. Thus, careful analysis of the the join techniques is essential for implementing the most efficient algorithm for the relations and the predicates in consideration.

Minibase supports Nested loop join and Sort-Merge Join [3]. To optimize the performance of selections and projection on attributes, these operations are implemented while reading data from files. To access data from the files containing records, Iterators are used which can return data in desired sorted order.

Join is generally performed by doing a selection or projection on a Cartesian product operation. This operation is time consuming and hence some databases use indices such as B+trees. Using B+trees may not result in full performance as the index is unclustered.

As an extension to the existing algorithms, we implement the space efficient inequality join ( IE Join). Two variants of the IE join exist: one for the generic two predicate join between two tables and the other one for two predicated self join. These queries contain predicate of the form where are attributes in relation R & S respectively and op is an inequality operator in {<,>,}..

# **INEQUALITY SELF JOIN**

Self-join queries join table to itself based on join attribute defined in query. For such join with same table, we have implemented the algorithm. It takes self-join inequality query as input and returns a set of result pairs. The algorithm sorts the join attribute list, computes permutation array which map the occurrences of records in one sort order to the occurrences of those records in another sorted order. The permutation array is scanned sequentially and a track of the scanned records is maintained in a bit array which allows for space efficient calculation of the joins.  It also sets an offset variable to distinguish inequality operators with or without equality.

Pseudo code –

Complexity – Time complexity of IESelfJoin in O() & space complexity is o(n).

# **INEQUALITY JOIN**

Inequality join algorithm takes a query with two inequlity join conditions as input and returns a set of result pairs. It first sorts the attribute values to be joined, computes permutation array for both the tables by proceeding in the same way as in case of two predicate IE self-join. Two offset arrays are also created for both join attributes in the respective predicates. These offset arrays map the position of records sorted on X attribute in R table to the record sorted on X’ attribute in S table where X maps to X’. Finally, a bit array is maintained as in case of self-join for computing the join tuples. The offset arrays along with the permutation arrays help in accessing the records according to constraints imposed by the specified predicates.

Pseudo code –

Complexity – Time complexity of IESelfJoin in O() & space complexity is O(m + n).

# **OPTIMIZATION**

# **CONCLUSION**

# **BIBLIOGRAPHY**

[1] Gehrke, Ramakrishnan: “Database Management Systems”

[2] Z. Khayyat, W. Lucia, M. Singh, M. Ouzzani, P. Papotti, J. Quiane-Ruiz, N. Tang, P. Kalnis. [Lightning Fast and Space Efficient Inequality Joins.](https://williamlucia.github.io/publications/) In: Proceedings of the 42nd International Conference on Very Large Data Bases (VLDB), pp. 2074-2085, New Delhi, India (2016)

[3] <http://research.cs.wisc.edu/coral/mini_doc/joins/j.html>