# Topic: Analysis of Joins and Semi-joins in Centralized and Distributed Database Queries

The research compares Join operations and Semi joins for centralized and distributed query systems. Initially, the paper explains the basics of joins and semi joins for distributed and centralized systems. Continuing that an Experimental Analysis is conducted on both joins and semi joints systems with various circumstances. There is a graphical representation also included in the paper to get a better idea of the advantages and disadvantages both systems have.

For the extraction of data from the database, these joins and semi-join are the most commonly used operations hence having the optimum complexity and higher speed could be the best case here. Join operation performs simple concatenation of two tables or datasets whereas Semi Join is a kind of same but restricts the duplicate values to be added to the outcome.

With factors like Bytes Accessed, IO Cost, and Cost running the experiment for efficiency comparison for the centralized systems, the paper states that for cases of simple join operation it is extremely efficient in the centralized system compared to semi-join in the same system. Whereas on similar experimental analysis with a distributed system but with different factors like Response Time, Total Time, the semi-join operation was working more efficiently compared to the normal join operations.

After going through the whole paper the research is detailed and various scenarios were also considered like query optimizer and all. The graphical representation for the comparison stated out what could we expect from a centralized and distributed system in the context of join and semi joins. Query Optimizer tools were utilized to identify alternative query recommendations, which helped to understand how efficient these searches were and shed some insight on query variants that generated comparable results.

The conclusion was not up to the mark in the context of the accuracy and having an ample amount of research. Here the author mentioned that semi joins are better for the distributed systems but there was a sentence where he stated that for distributed systems simple joins perform better than semi joins without any basis and was not covered in the paper.

# A Survey on Distributed Deadlock and Distributed Algorithms to Detect and Resolve Deadlock

The Research puts light on the different algorithms and data structures that are being used in a Distributed Database System to Detect and Recover from a deadlock situation. In starting of the paper there is a small description about Distributed Data System and how there is the generation of deadlock. After that how we could prevent the deadlock and if created then how it could be solved or recovered. In the end, there are Edge chasing algorithm and B.M. Alom Algorithm which is deeply explained with its benefits and disadvantages.

Concurrency control is a technology used in DDBMS to maintain database consistency when transactions are running on the same database at the same time. One of the techniques is that transactions lock the data while operations are performed on the data. When two transactions lock data and request data locked by the corresponding transaction, it enters an infinite state, where two transactions wait for each other, which is called a deadlock.

Data structures such as WFG (Wait For Graph), probes, and tables are used for recovery each time a deadlock occurs. Different algorithms have been proposed using all these structures at the same time to detect deadlocks. Wait for Graphs are directional graphs that detect deadlocks after each cycle. Probes are messages that are connected to all transactions, waiting for each other to complete. A deadlock occurs when the message returns to the initial transaction that started the probe.

The EdgeChasing Algorithm is similar to the Probing method below and has the disadvantage that the algorithm will fail if the transaction that initialized the probe is not deadlocked.

B.M. Alom Algorithm uses both tables simultaneously and uses WFG to detect deadlocks in the system. One of the tables stores all the transactions involved in the creation of the deadlock, and the other stores the priority of each such transaction. In a deadlock, the system aborts the lowest priority transaction. If the priority of a transaction is constantly changing in the system, B.M. I found the algorithm to be very low.

The paper seems realistic and it could be implemented in the real world by using the algorithms described but it would be great if they had included some code snippets where we could have a demonstration of how the code will be for any programming language. There is also scarcity for the graphical representation as there is a lower number of images. It could be added to have a better understanding of the paper.