REPORT

BY: AKSHAY JAIN
REG. No. 17BCS002

1. Introduction

1. Scope of the Topic:

The topic for this report is **Query Optimisation and the Various Techniques.** There has been extensive work on Query Optimisation since 70's and the scope of this topic is beyond imaginable. The work started due to the increasing amount of data being generated with every passing year and the tasks and industries using softwares, automation and digitisation of records.

The records were then stored in Data Base Management Systems and the DBMS started evolving with an endpoint then as RDBMS (Relational Data Base Management System). As the data grew larger and larger the computers then with simple processors needed a language to retrieve data and SQL (Structured Query Language) came into picture. With time SQL made a major impact in Information Technology sector. The queries executed using SQL on larger databases started taking significant amount of time which drew the attention of researchers to reduce the time and the best way to do so was the optimise the queries being written.

The papers selected by me were pioneer in the Query Optimisation and were used by tech giants like **MICROSOFT** who was one of the contributor towards SQL deploying the MySQL which is still the most used software in the globe.

Major Problem areas where the papers have contributed with the application:

- In the 90's the computation power was not that good as it is today. Today you can query a database with 1000000 rows in around 1 second using SQLite or Big Query but then querying 1000000 rows nearly took more than 2 hours, that is a huge time as people had to literally wait for the output results. These papers contributed a lot in bringing down the time constraints.
- Looking the time, the technology industry realised the need of faster processors and computation power, which lead to helping them testing the computational power of processors by testing various queries on them.
- One of this paper was key for Microsoft for developing MySQL and stepping into the database industry.
- These papers also introduced Artificial Intelligence to the world of Database by using Heuristics and Cost Functions in solving the Query Optimisation Problem.

2. Literature Study:

The earlier authors as described* by the researchers primitively worked on the development on queries and focussing on how the queries are being addressed. There prime focus on query optimisation using various algorithms and using parallel processing as a technique.

The early 80s had the sight of introducing multiple queries for better results and so the lookout towards the time consumption were not addressed due to enhanced results by multiple queries. As the databases sizes grew multiple queries started taking significant amount of time and then the query processing time was noticed.

At first the basic solution for decreasing the time was to minimise the data size and remove the unnecessary rows while retrieving results. This approach was soon discarded as this was a manual approach and could not fit with maximum databases.

Later Query Optimisers were built which used algorithms and multi-threading concepts to address the query processing time. This approach was successful for independent data sets but not with relational data sets. Also, this approach dependent the time complexity of the algorithm which were basically NP-Hard and required exponential amount of time.

Since the problems were NP-Hard so solving them in polynomial amount of time would require pre-defined knowledge or also known as **Heuristics** which enhanced the query optimisers and build the main area of research then. Also, it was noted the older techniques of writing the query contributed a lot towards increasing the query processing time.

*In my three research papers, none of the authors clearly or explicitly stated or cited the problem with the older techniques and so I am referring this literature to my own research using the ACM Library with paper of Query Optimisers in late 70s and early 80s.

2. New proposed Solutions - Research papers

2.1 Paper 1: An Overview of Query Optimisation in Relational Systems, Year of Publication: 1998, PODS '98 Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems

The author **Surajit Chaudhari** in this explains the foundations and present sampling of significant work in this area. The key concept explained by him his Query Optimiser and Query Execution Engine. The author describes Query Optimiser as a difficult search problem. As mentioned by him, in order solve this difficult problem we need to provide:

- A space of plans (search space)
- A cost estimation technique so that the cost may be assigned to each plan in the search space. Intuitively, this is an estimation of the resources needed for the execution of the plan.
- An enumeration algorithm that can search through the execution space.

His way of addressing the problem was using the Artificial Intelligence technique of heuristics.

He uses the System R optimiser as the model to explain his research.

Some of the explanations used by author using figures are:

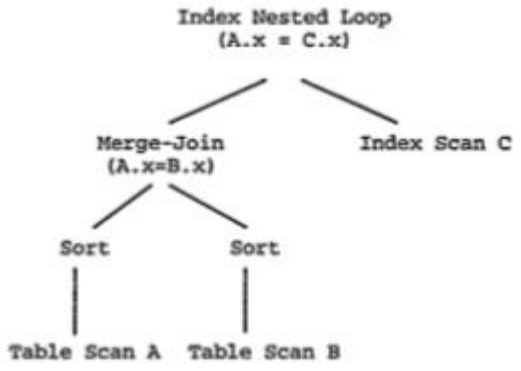


Figure 1. Operator Tree

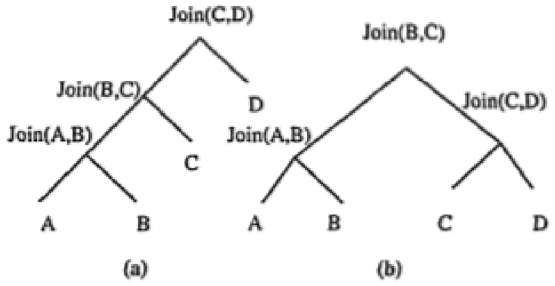


Figure 2. (a) Linear and (b) bushy join

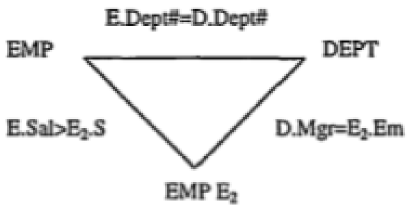
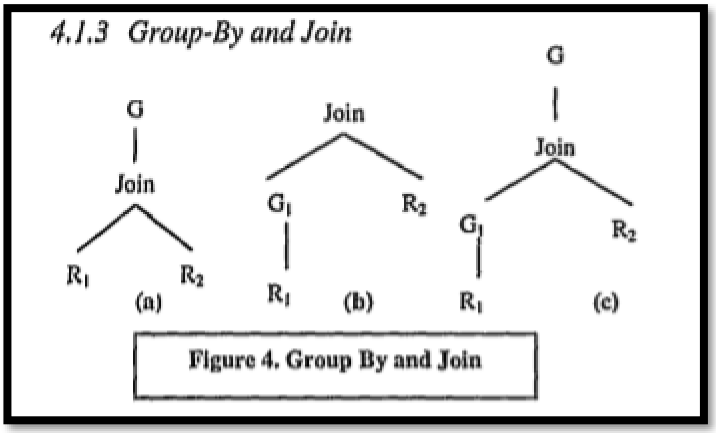


Figure 3. Query



4.1.3 Group-By and Join

Figure 4. Group By and Join

2.2 Paper 2: Query Optimisation, Year of Publication: 1996, ACM(Association of Computing Machinery) Computing Surveys

The author **Yannis E. Ioannidis** explained in his paper that a problem can be solved in many ways or in his words "A destination can be reached through different ways and modes and a person choses the way and mode which suits them and is most optimised for them"

Similarly, given a query, there are many plans that a Data Base Management System can follow to process and execute the query to derive an answer. All the plans that are available varies on different factors like cost, amount of time, availability of resources and etc. The goal of a Query Optimiser is to find a plan that needs the least amount of time with the cost being in the reach and practical.

The author describes the model having a Query Parser, Query Optimizer, Code Generator or Interpreter and Query Processor

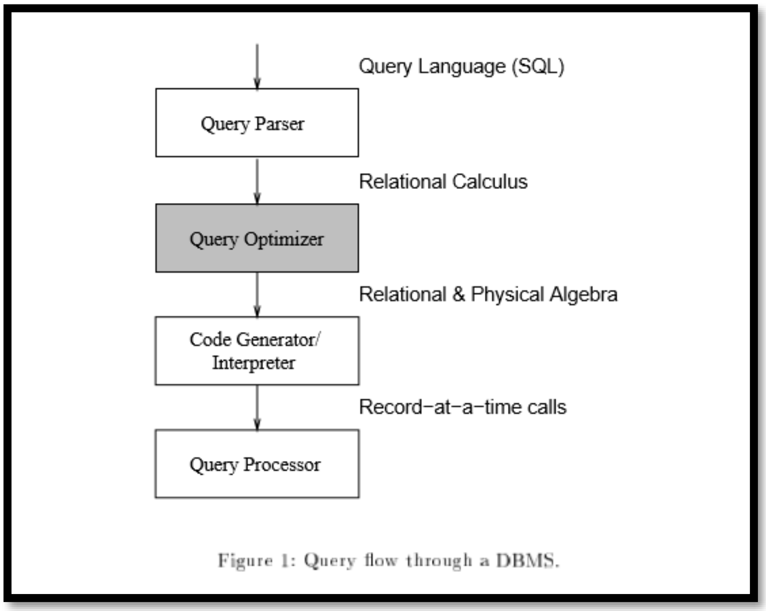


Figure 1: Query flow through a DBMS.

The author explains the architecture related to the Query Optimiser which here is represented by the below figure.

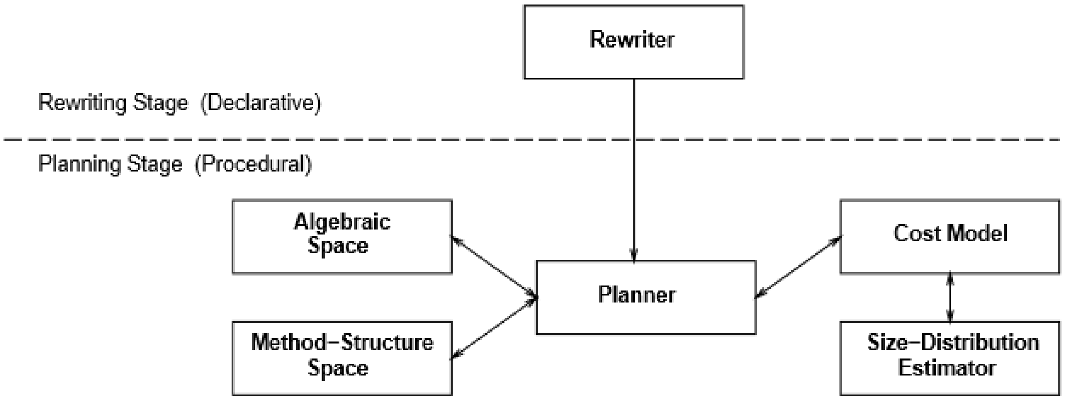

Rewriter
Rewriting Stage (Declarative)
Planning Stage (Procedural)
Algebraic Space
Method−Structure Space
Planner
Cost Model
Size−Distribution Estimator

Figure 2: Query optimizer architecture.

The main focus of this paper is to use Dynamic Programming Approach and Randomised Algorithm to create a Query Optimiser which suits well for relational Data Base Management Systems.

Later on the author have discussed the environment of parallel databases and distributed databases.

Some of the Query Optimisation Techniques explicitly mentioned in the research paper are Semantic Query Optimisation, Global Query Optimisation, Parametric or Dynamic Query Optimisation.

2.3 Paper 3: Query Optimisation Techniques – Tips for Writing Efficient and Faster SQL Queries, Year of Publication: 2015, International Journal of Scientific and Technology Research

The author **Jean Habimana** has focussed his research towards how the queries are written so that on execution they are efficient and are faster. He mentioned few tips which are really needed for daily query writing to enhance the query processing time. He suggests no query optimiser is better than replacing our own written queries with the optimised one.

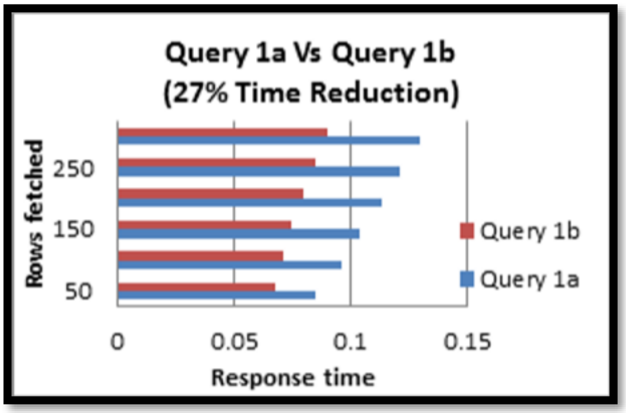
Some of the tips mentioned by the author are:

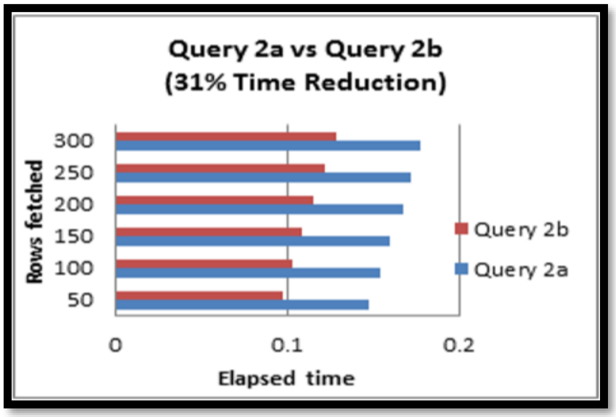
1. Use column names instead of * in a SELECT statement.
2. Avoid including a HAVING clause in a SELECT statement.
3. Eliminate unnecessary DISTINCT conditions.

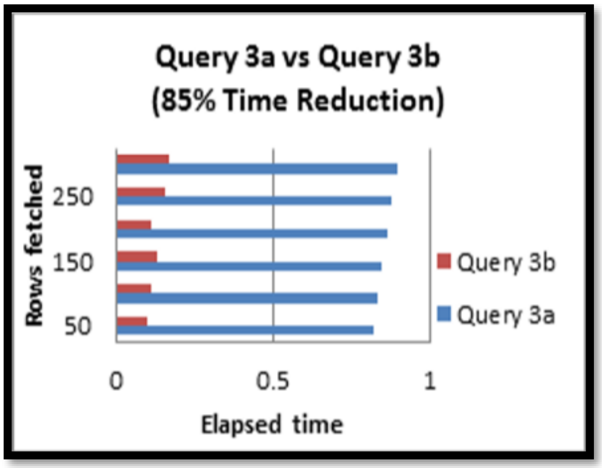
4. Un-nest sub queries.
5. Consider using an IN predicate when querying an indexed column.
6. Use EXISTS instead of DISTINCT when using table joins that involves tables having one-to-many relationships
7. Try to use UNION ALL in place of UNION.
8. Avoid using OR in join conditions
9. Avoid functions on right hand side of the operator.
10. Remove any redundant mathematics.

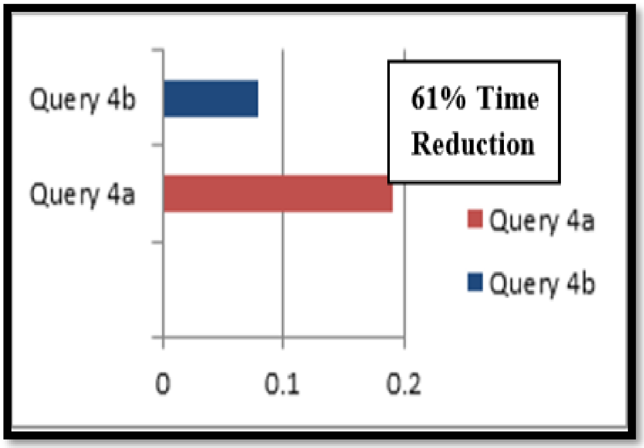
The author has compared the written query and the optimised and provided the proof with graphs.

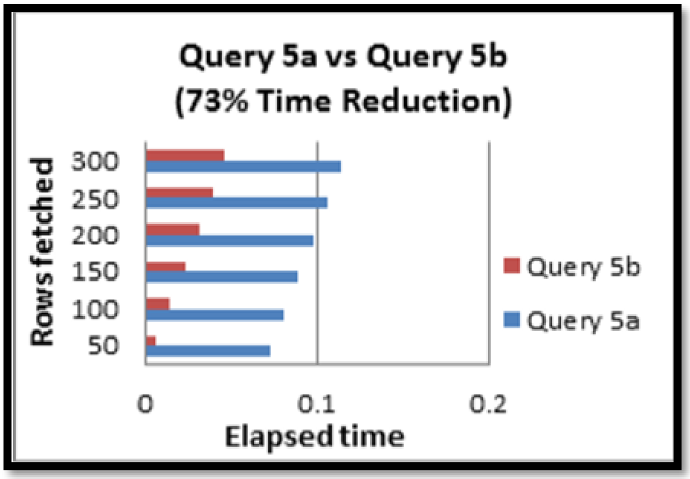
Some of the graphs I have shared below:

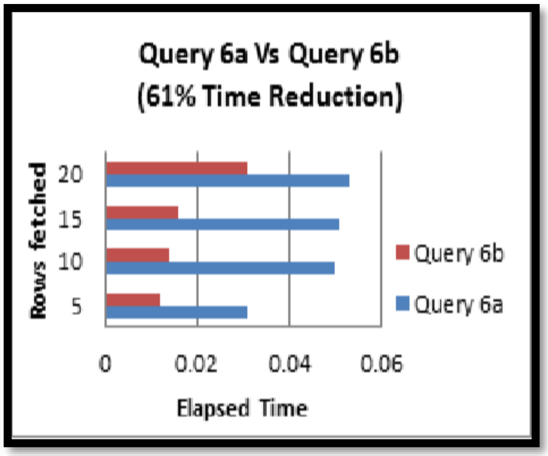


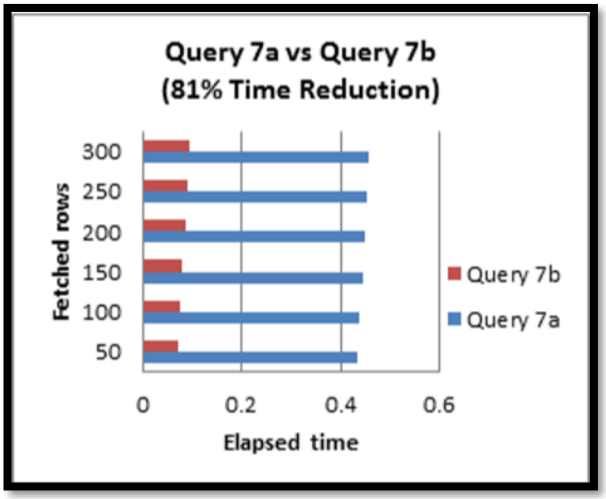


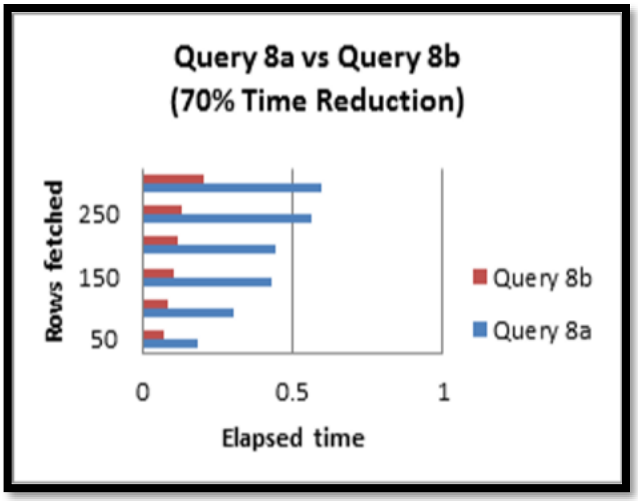


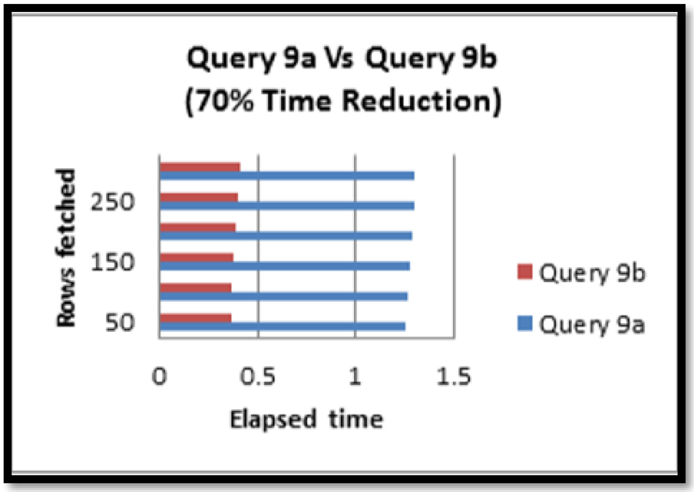


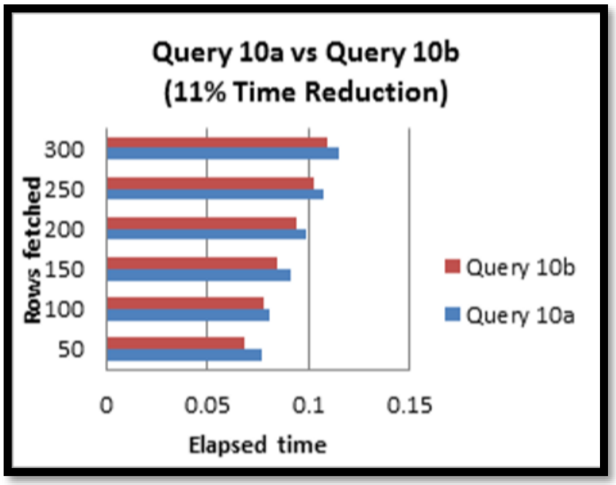

Query 5a vs Query 5b
(73% Time Reduction)
Rows fetched
300
250
200
150
100
50
0
0.1
0.2
Elapsed time
Query 5b
Query 5a


Query 6a Vs Query 6b
(61% Time Reduction)
Rows fetched
20
15
10
5
0
0.02
0.04
0.06
Elapsed Time
Query 6b
Query 6a


Query 7a vs Query 7b
(81% Time Reduction)
Fetched rows
300
250
200
150
100
50
0
0.2
0.4
0.6
Elapsed time
Query 7b
Query 7a


Query 8a vs Query 8b
(70% Time Reduction)
Rows fetched
250
150
50
0
0.5
1
Elapsed time
Query 8b
Query 8a


Query 9a Vs Query 9b
(70% Time Reduction)
Rows fetched
250
150
50
0
0.5
1
1.5
Elapsed time
Query 9b
Query 9a


Query 10a vs Query 10b
(11% Time Reduction)
Rows fetched
300
250
200
150
100
50
0
0.05
0.1
0.15
Elapsed time
Query 10b
Query 10a

3. Glossary/References

- ACM DIGITAL LIBRARY
- INTERNATIONAL JOURNAL OF SCIENTIFIC AND TECHNOLOGY
- INTERNATION JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCES
- INSTITUTE OF ELECTRONIC AND ELECTRICAL ENGINEERS
- HARVARD DAS LAB
- MICROSOFT RESEARCH LAB
- GOOGLE SCHOLAR
- STANFORD UNIVERSITY ARTICLES
- WIKIPEDIA