Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 3, March 2016, pg.06 – 16

Roll of Relational Algebra and Query Optimizer in Different Types of DBMS

Vandana Bhagat[1], Dr. Arpita Gopal[2]

[1]Computer Department, Sinhgad Institute of Management Vadgaon(Bk), Pune, India

[2](Computer Department) Sinhgad Institute of Business Administrator and Research, Kondhwa(Bk.), Pune, India

Abstract— The data is a vital part of all the organization. Database is the software used to manage that data to make the retrieval of required data faster and accurate. For fast processing, the query has to be fired on the database in such a way that it should follow the shortest path to reach to the target data and which can manage its processing to reduce the unnecessary fetching of the records from the tables. It should enough intelligent that it can manage parallel execution of different query pieces or arrangement of those pieces in such a way so that the execution will be faster and required to fetch minimum records from the table to get the result. This task is done by Query Optimizer. The relational algebra required to create a query execution plan which then analyzed by the optimizer to select the shortest and most efficient plan for better query execution. Query execution is also depends on the data storage strategy used. There are multiple ways by which data can be stored in the memory. It gives added advantage for better query performance. The paper includes the steps of query execution and different techniques used to optimize the query for faster execution. It also shows the flow of query optimization. After that paper includes the introduction of different types of data storage procedures and query optimization for each type of storage.

Keywords— Columnar database, hybrid database, optimization, query processing, relational algebra, row oriented database.

I. INTRODUCTION

Database is a collection of information organized in such a way that a computer program can quickly select desired pieces of data. Traditional databases are organized by fields, records, and files. A field is a single piece of information, a record is one complete set of fields and a file is a collection of records. To access information from a database, a database management system (DBMS) is required. This is a collection of programs that enables to enter, organize, and select data in a database. There are different types of databases present which stores different type of data in different ways for different purpose. They are Row oriented storage, Column oriented storage and Hybrid storage. Each storage type gives better performance of query execution and needs different type of optimization technique. Market includes many types of DBMSs. But the researcher has considered only three types DBMS according to the data storage strategy used.

II. TYPES OF DATABASE MANAGEMENT SYSTEMS

A. Row oriented Databases:

In this type of database all the attributes of a particular record are added first and then goes to the next record insertion. Therefore, row oriented systems are more efficient when many columns of a single row are required to be fetched at the same time. When row size is relatively small the entire row can be retrieved in a single disc seek. In row oriented approach all the columns of the table must be accessed which are not required. This is because its touple-at-a-time processing paradigm. As a result, analytical and BI queries like aggregation are most often read significantly more data than is needed to satisfy the request. [10]

These types of databases are mostly used in case of OLTP where there are large number of short on-line transactions are carried out.

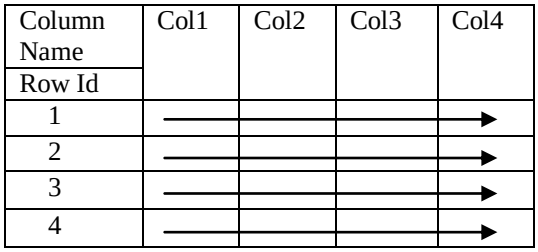


Figure1: Row Oriented storage

B. Column Oriented Database:

This database serializes all of the values of a column together then the values of the next column and so on. Column stores are especially useful for analytical workload which is due to the fact that analytical applications are largely attribute-focused rather than entity-focused. Therefore, only a small number of columns in the table might be of interest for a particular query. The problem in the column oriented database is record insertion and update process. These requires reading every column in the record, determining where the new value belongs in that set, inserting it in a right place then building the required linking information to be able to retrieve it and insert it correctly into the original record format. [10]

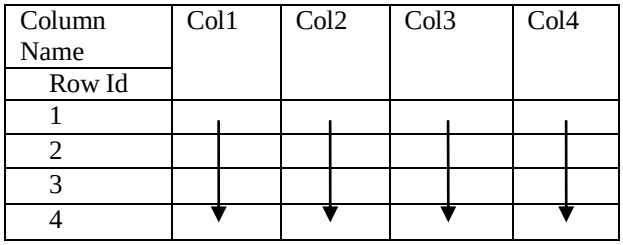


Figure2: Column Oriented storage

*C. **Hybrid database:***

Each type of database has its own pros and cons. In this section we will consider the use of advantages of different types of databases. The database which collectively gives the advantages of multiple types of databases can be considered as Hybrid Database. There are multiple types of Hybrid databases present in the market. Each type differentiates itself by giving different types of advantages of two or more distinct databases. There are many types of Hybrid databases exist. Researcher has concentrated on Row-Column hybrid database for the study.

III. PREVIOUS STUDY

A. Global query processing and optimization in CORDS Multi Data base, Qiang Zhu, Per-Ake Larson

This paper introduces a new technique to process global query in multi data base system. Multi data base system integrates information from autonomous pre-existing local database managed by heterogeneous local database management system in distributed environment.

Paper includes challenges of multi database. For such type of data base following special types of techniques are introduced in the paper for query optimization and improved performance.

Query Sampling: A global query is first decomposed into several local queries that can be performed on local relevant data bases.

Query Probing: It discovers missing information from local data base.

Piggy back Method: The idea of this technique is to ride some side intervals piggy back on the processing of user query to get desirable information with just slight additional cost.

B. Introduction to Query Processing and Optimization, Michael L. Rupley, Jr.

This paper has introduced the basic concepts of query processing and query optimization in the relational data base domain. Query processing and some algorithms and rule-sets utilized to produce more efficient queries. The paper also discussed the implementation plan to extend the capabilities of my Mini-Data base Engine program. Some query optimization techniques and algorithms are listed in this paper.

C. Query Optimization, Thomas Neumann

The researcher in this study has given the basic idea about the different concepts of Relational algebra. The paper gives the knowledge about logical and physical algebra. Different concepts like Text book Query Optimization, Join Ordering, Accessing the Data, Physical Properties, Query Rewriting, Self Tuning are also discussed.

D. "A tour through hybrid column/row-oriented DBMS schemes ", by DANIEL ABAD Thursday, September 3, 2009

This article has discussed about the immerging concept of Hybrid Data base System and shows how that approach will be beneficial for the companies in the coming years. It has given three approaches for doing the same like,

PAX

Fractured Mirrors

Fine-grained hybrids

The author has also given the advantages and disadvantages of each approach.

E. Daniel J. Albadi, Samuel R. Madden, Nabil Hachmen 8 (2008) in their paper "Column-Store Vs. Row-Store: How Different Are They Really?" has compared performance of commercial row-store under the variety of different configurations with a column store and showed that the row-store performance is significantly slower on a recently proposed data warehouse benchmark. Authors have analyzed the performance difference and show that there are some important differences between the two systems at a query executor level. They have concluded this by demonstrating impact on performance of a variety of column oriented query execution techniques like vectorized query processing, compression and new join algorithm which has been introduced in this paper. The paper concludes that though in some extend column-store has many advantages on row-store, change must be made to both the storage level and query executor to fully obtain the benefit of column oriented approach.

IV. QUERY

A request for information from a data base is a query. It is a piece of code that is sent to a data base in order to get information back from the data base.[18]. It is a "question" that you ask the data base. The result of the query is the information that is returned by the data base management system. Queries are usually constructed using SQL (structured query language) which resembles a high-level programming language. [20]

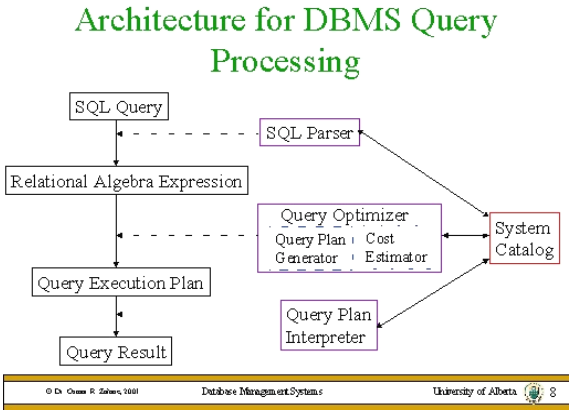


Figure3: Query execution/ Processing [27]

A query is processed in four general steps:[21]

A. Scanning and Parsing
B. Query Optimization or planning the execution strategy
C. Query Code Generator (interpreted or compiled)
D. Execution in the runtime database processor

A. Scanning and Parsing

1) When a query is first submitted (via an applications program), it must be scanned and parsed to determine for appropriate syntax. Scanning is the process of converting the query text into a tokenized representation.
2) The tokenized representation is more compact and is suitable for processing by the parser. This representation may be in a tree form.
3) The Parser checks the tokenized representation for correct syntax.
4) In this stage, checks are made to determine if columns and tables identified in the query exist in the database and if the query has been formed correctly with the appropriate keywords and structure.
5) If the query passes the parsing checks, then it is passed on to the Query Optimizer.[21]

B. Query Optimization or Planning the Execution Strategy

1) Query can be executed in different ways.
2) Each operation in the query (SELECT, JOIN, etc.) can be implemented using one or more different Access Routines.
3) The goal of the query optimizer is to find a reasonably efficient strategy for executing the query using the access routines.
4) Optimization typically takes one of two forms: Heuristic Optimization or Cost Based Optimization[21]

C. Query Code Generator (interpreted or compiled)

1) Once the query optimizer has determined the execution plan (the specific ordering of access routines), the code generator writes out the actual access routines to be executed.

2) With an interactive session, the query code is interpreted and passed directly to the runtime database processor for execution.[21]

D. Execution in the runtime database processor

1) At this point, the query has been scanned, parsed, planned and (possibly) compiled.
2) The runtime database processor then executes the access routines against the database.
3) The results are returned to the application that made the query in the first place.
4) Any runtime errors are also returned.
5) Before going in depth of Optimization we need to understand different relational algebra operations. Then only the exact query plan can be evaluated.

V. RELATIONAL ALGEBRA

It is similar to normal algebra except it uses relations as values instead of numbers and the operations and operators are different. One needs to know about relational algebra to understand query execution and optimization in a relational DBMS. Some advanced SQL queries requires explicit relational algebra operations like outer join. SQL is declarative, which means that you tell the DBMS what you want, but not how it is to be calculated. Relational algebra is more procedural than SQL. Relational algebra is mathematical expressions.[22]

A. The basic Operations of Relational Algebra are

1) Union: R1 U R2 (union) is the relation containing all tuples that appear in R1, R2, or both. [24]

2) Insertion: R1 n R2 (intersection) is the relation containing all tuples that appear in both R1 and R2. [24]

3) Difference: R1 - R2 (set difference) is the relation containing all tuples of R1 that do not appear in R2.[24]

4) Rename (ρ): A rename is a unary operation written as $\rho_{a/b}(R)$ where the result is identical to R except that the b attribute in all tuples is renamed to an attribute. This is simply used to rename the attribute of a relation or the relation itself. [16]

5) Projection (π): Chooses a subset of the columns in a relation, and discards the rest.

R2 = project(R1,D1,D2,...Dn)

That is, from the tuples in R1 we create a new relation R2 containing only the domains D1,D2,..Dn.This specifies the specific subset of columns (attributes of each tuple) to be retrieved.[25]

6) Selection (σ)

Selects tuples from a relation whose attributes meet the selection criteria, which is normally expressed as a predicate.

R2 = select(R1,P) That is, from R1 we create a new relation R2 containing those tuples from R1 that satisfy (make true) the predicate P. [25]

B. Relational Algebra: Derived Operators

1) *Join:* It combines attributes of two relations into one.

R3 = join(R1,D1,R2,D2)

Given a domain from each relation, join considers all possible pairs of tuples from the two relations, and if their values for the chosen domains are equal, it adds a tuple to the result containing all the attributes of both tuples (discarding the duplicate domain D2). [25]

2) *Natural Join:*

If the two relations being joined have exactly one attribute (domain) name in common, then we assume that the single attribute in common is the one being compared to see if a new tuple will be inserted in the result. [25]

3) *Semijoin:*

The left semijoin is joining similar to the natural join and written as R ⋉ S where R and S are relations. The result of this semijoin is the set of all tuples in R for which there is a tuple in S that is equal on their common attribute names.[25]

4) *Antijoin:*

The antijoin, written as R ▷ S where R and S are relations, is similar to the semijoin, but the result of an antijoin is only those tuples in R for which there is no tuple in S that is equal on their common attribute names. [25]

5) *Division:*

The division is a binary operation that is written as R ÷ S. The result consists of the restrictions of tuples in R to the attribute names unique to R, i.e., in the header of R but not in the header of S, for which it holds that all their combinations with tuples in S are present in R. [25]

C. Physical Algebra Join

1) Nested Loop Join

The nested loops join, also called nested iteration, uses one join input as the outer input table and one as the inner input table. The outer loop consumes the outer input table row by row. The inner loop, executed for each outer row, searches for matching rows in the inner input table.[25]

A nested loop join on relations R1 (with N domains) and R2 (with M domains), considers all |R1| x |R2| pairs of tuples.

```
R3= join(R1,Ai,R2,Bj)
for each tuple t in R1 do
 for each tuple s in R2 do
  if t.Ai = s.Bj then
   insert(R3, t.A1, t.A2, ..., t.AN,
    s.B1, ..., s.B(j-1), s.B(j+1), ..., s.BM)[24]
```

2) Index Join

An index join exploits the existence of an index for one of the domains used in the join to find matching tuples more quickly.

```
R3= join(R1,Ai,R2,Bj)
for each tuple t in R1 do
 for each tuple s in R2 at index(t.Ai) do
   insert(R3, t.A1, t.A2, ..., t.AN,
    s.B1, ..., s.B(j-1), s.B(j+1), ..., s.BM)
```

We could choose to use an index for R2, and reverse the order of the loops. The decision on which index to use depends on the number of tuples in each relation.[24]

3) Block wise Nested Loop Join

A block-nested loop is an algorithm used to join two relations in a relational database.

This join algorithm improves on the simple nested loop join by only scanning R2 once for every group of R1 tuples. For example, one variant of the block nested loop join reads an entire page of R1 tuples into memory and loads them into a hash table. It then scans R2, and probes the hash table to find R2 tuples that match any of the tuples in the current page of R1. This reduces the number of scans of R2 that are necessary. Much faster, but requires memory. For further improvement we can use hashing for equi-joins.[24]

4) Sort Join

If we don't have an index for a domain in the join, we can still improve on the nested-loop join using sort join.

R3= join(R1,Ai,R2,Bj)

- Merge the tuples of both relations into a single list
- List elements must identify the original relation
- Sort the list based on the value in the join domains Ai and Bj
- All tuples on the sorted list with a common value for the join domains are consecutive

Pair all (consecutive) tuples from the two relations with the same value in the join domains [24]

D. Physical Algebra: Aggregation

- Aggregation sorted input: Aggregates the input directly. It is fast but requires sorted input[27]
- Aggregation quick sort: Sorts chunks of input with quick sort, merges sorts[27]
- Aggregation Heap Sort : Slower sort, but longer runs[27]
- Aggregation Hybrid Hash: Partitions like a hybrid hash join.[27]

VI. NEED OF RELATIONAL ALGEBRA IN QUERY OPTIMIZATION

One problem while execution of the query is, it does not specify a query execution plan. A relational algebra expression is procedural. It can give an associated query execution plan. The solution to the problem is to Convert SQL query to an equivalent relational algebra and evaluates it using the associated query execution plan.

Query optimizer uses heuristic algorithms to evaluate relational algebra expressions.

This involves:

- Estimating the cost of a relational algebra expression
- Transforming one relational algebra expression to an equivalent one. To transform a relational expression into another equivalent expression we need transformation rules that preserve equivalence.

Query optimization is a function in which multiple query plans for satisfying a query are examined and that query plan is selected which gives efficient query execution in minimum cost. These plans can be done in number of ways therefore the final plan may or may not be absolute. [5]

Typically the resources which are measured are CPU path length, amount of disk buffer space, disk storage service time, and interconnect usage between units of parallelism. The set of query plans examined is formed by examining possible access paths (e.g., primary index access, secondary index access, full file scan) and various relational table join techniques (e.g., merge join, hash join, product join).

Basically there are two types of optimization. [5]

A. Logical optimization: It generates a sequence of relational algebra to solve the query.

B. Physical optimization: It is used to determine the means of carrying out each operation.

C. Heuristic Query Optimization

A query can be represented as a tree data structure. Operations are at the interior nodes and data items (tables, columns) are at the leaves.

The query is evaluated in a depth-first pattern.

An overall rule for heuristic query optimization is to perform as many select and project operations as possible before doing any joins.[3]

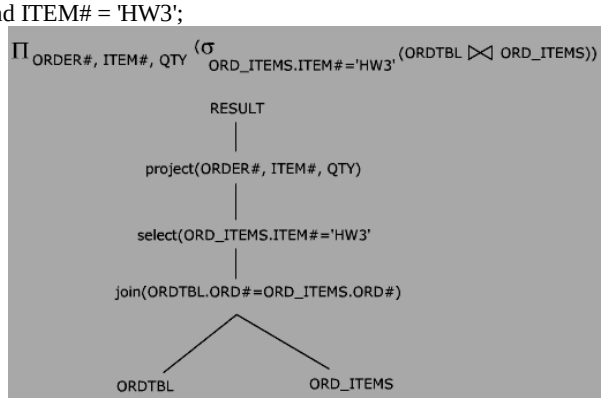
D. Systematic (Cost based) Query Optimization

Several Cost components to consider:

1) Access cost to secondary storage (hard disk)
2) Storage Cost for intermediate result sets
3) Computation costs: CPU, memory transfers, etc. for performing in-memory operations.
4) Communications Costs to ship data around a network. e.g., in a distributed or client/server database.

In general, it performs the selections first, and then the join and finally the projection.[3]

The steps of query optimization are explained below

Step 1 Casting into some Internal Representation:	Representation of each SQL query into some internal representation (Query Tree) which is more suitable for machine manipulation. Select ORDDATE, ITEM#, QTY from ORDTBL, ORD_ITEMS where ORDTBL.ORD# = ORD_ITEMS.ORD# and ITEM# = 'HW3';  Query Tree for the above query[28]
Step 2 Conversion into Canonical Form	The optimizer makes use of some transformation rules for sequencing the internal operations involved. • **Rule-1:** (A JOIN B) WHERE restriction_A AND restriction_B --- (A WHERE restriction_A) JOIN (B WHERE restriction_B) **Note:** Restriction when applied first, cause elimination and hence better performance. • **Rule 2:(**A WHERE restriction_1) WHERE restriction_2 --- A WHERE restriction_1 AND restriction_2 **Note:** Two restrictions applied as a single compound one instead applying the two individual restrictions separately. • **Rule 3:** (A[projection_1])[projection_2] --- A[projection_2] **Note:** If there is a sequence of successive projections applied on the same relation, all but the last one can be ignored. i.e., The entire operation is equivalent to applying the last projection alone. • **Rule 4:** (A[projection]) WHERE restriction --- (A WHERE restriction)[projection] **Note:** Restrictions when applied first, cause eliminations and hence better performance.[28]

Step 3 Choosing Candidate Low-level Procedures	In this step, the optimizer decides how to execute the transformed query. Here factors such as existence of indexes or other access paths, physical clustering of records, distribution of data values etc. are considered.[28]
Step 4 Generation Query Plans and Choosing the Cheapest	In this last step, query plans are generated by combining a set of candidate implementation procedures.[28]

VII. QUERY EXECUTION IN ROW, COLUMN ORIENTED AND HYBRID DATABASE

Query optimizer uses different strategies depending upon the storage of the data. Queries can be differentiated as Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP).

OLTP: It contains large number of short on-line transactions (INSERT, UPDATE, and DELETE). It works on current and detail data. It is used to control and run fundamental business tasks.

OLAP: It contains relatively low volume of transactions. Queries are often very complex and involve aggregations. It is used to help with planning, problem solving, and decision support.

OLTP queries work efficiently on Row oriented database while OLAP queries work efficiently on Column oriented database. Each type of database uses its own optimization techniques which are specific for either OLTP or OLAP queries. But when OLTP queries fired on column oriented database or OLAP queries fired on row oriented database, their performance decreased drastically. The Hybrid database gives combined advantage of both types of databases and gives good performance for any type of query.

Following are different type of optimization techniques which are specific for particular type of storage. Some techniques can be applied to any of the storage.

VIII. COMMON OPTIMIZATION TECHNIQUES

A. Symmetric multi-processor (SMP) or massive parallel processor (MPP) Technology:

In this approach the query is broken down into parts and processes them in parallel by different processors by replicating the query. Each copy works against portion of the DB which is usually horizontally partitioned. The same query is to run on each portion in parallel in a separate processor then intermediate results are combined to create the final query set as if the query was running once on the entire DB. [13]

B. Processing a sequence of interdependent queries:

It requires a multi-query optimization. Optimization plan is decided on the basis of the dependencies between queries. In which order they should be specified and which results should be stored, then each query is passed separately to the DB optimizer. [14]

C. Best View First (BVF):

Instead of constructing the global execution plan by adding the queries one by one (bottom-up approach), this approach uses a top-down approach. For every iteration, the most beneficial view best view MV is selected, based on a savings metric, and all the queries which are covered by best view and have not been assigned to another view yet, are inserted in the global plan. The process continues until all queries are covered.[15]

IX. OLTP QUERY OPTIMIZATION TECHNIQUES

A. Indexing: Database optimizers scan the appropriate index to identify any target rows faster than scanning the entire table. If the table is indexed, a binary search for files is carried out on the blocks rather than on the records.[13]

*B. **B Tree, B+ Tree***

X. OLAP QUERY OPTIMIZATION TECHNIQUES

A. Generalized projections (GP) query optimization technique:

GP captures aggregations, group by, projections with duplicate elimination, and duplicate preserving projections. The GP pushes down query trees for select-project-join queries which use any aggregate function. The pushing down technique will result in the removal of tuples and attributes early and consequently leading to smaller intermediate relations and reducing the query cost. [13]

B. Combining Rule-Based Approach (RBA) and Cost-Based Approach (CBA)

This algorithm is implemented using Java and generates new selection predicates over the dimension tables of a given OLAP query and therefore a new rewritten query is generated. These new predicates contribute in optimizing queries. [17]

C. Translating nested query expressions into an algebra extended with the GMDJ operator:

GMDJ is an operator with a simple and easy to optimize implementation. The algorithm uses GMDJs in place of joins, outer joins, combinations of aggregation/joins, or set difference. The algorithm only introduces additional joins when this is unavoidable. In all other cases, the algorithm avoids unnecessary supplementary tables and the number of algebraic operations is kept to the minimum. As the GMDJ operator can commute with projections, selections, joins, and other GMDJ operators, the GMDJ operator provides flexibility comparable to that of join reordering. [18]

XI. HYBRID DATABASE QUERY OPTIMIZATION

Hybrid database is meant to execute any type of query in the efficient way. It provides the advantages of both types of database avoiding their disadvantages. The Hybrid database system should have to have both row- and column-oriented execution engines. The optimizer has to know about the differences between row-storage and column-storage, and indexing code has to be updated appropriately. The execution engine and optimizer should already have knowledge about the difference between rows and columns and can act appropriately. It requires some knowledge about a query workload.

It is much easier to implement in a column-store that already supports early materialization (early materialization refers to the ability to reconstruct rows from column-storage early in a query plan) than in other types of systems. This is because early-materialization requires query operators to be able to handle input in both column and row-oriented format (it will be in column format if tuples haven't been materialized yet, otherwise it will be in row-oriented format). [11]

XII. CONCLUSION

Relational algebra is more operational and useful as internal representation for query evaluation plans. There are several ways of expressing a given query. The query optimizer should choose the most efficient query plan. Logical algebras describe only the general approach while physical algebra fixes the exact execution including run time characteristics. The knowledge about relational algebra is essential to understand query execution and optimization in a relational DBMS.

The query optimizer is the component of a database management system that attempts to determine the most efficient way to execute a query. Cost-based query optimizers assign an estimated "cost" to each possible query plan, and choose the plan with the smallest cost. The knowledge of relational algebra and steps of query optimization in the query execution can defiantly improve the performance of the query.

The knowledge of relational algebra and query execution steps makes it possible to generate new optimization technique that can improve the query performance.

Different types of databases use different query optimization techniques. For OLTP queries the row oriented DBMS optimizer uses Indexing, joining, B Tree, B+ Tree etc while for OLAP queries the column oriented DBMS optimizer uses Materialized views, aggregations, compression techniques. Different types of optimizers optimize particular type of query. It gives bad performance for the queries which are not similar to its implementation strategy. Hybrid DBMS contains query engine and optimizer for both types of queries. Therefore it can give good performance for both types of query. It implements only essential features of each type of optimizer.

REFERENCES

[1] "What's new in Querying, Query Processing and Optimization in PBMS?" Ilaria Bartolini1, Paolo Ciaccia1, Marco Patella1, and Yannis Theodoridis2J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.
[2] "Query Optimization" Thomas Neumann September 9, 2007
[3] "Query Optimization", Dr. Karen C. Davis
[4] "Introduction to Query Processing and Optimization" , Michael L. Rupley, Jr.
[5] "An Overview of Query Optimization", copyright @2006 Pearson Addison- Wesley.
[6] "DataGuides: Enabling Query Formulation and Optimization inSemi structured Databases", Roy Goldman, Jennifer Widom
[7] "Global query processing and optimization in CORDS MultiDatabase" Qiang Zhu, Per-Ake Larson
[8] A tour through hybrid column/row-oriented DBMS schemes (author) Thursday, September 3, 2009
[9] Daniel J. Albadi, Samuel R. Madden, Nabil Hachmen 8 (2008) in their paper "Column-Store Vs. Row-Store: How Different Are They Really?"
[10] "Column-Stores vs. Row-Stores How Different are they Really? ", Daniel J. Abadi, Samuel Madden, and Nabil Hachem, SIGMOD 2008
[11] "A tour through hybrid column/row-oriented DBMS schemes", DANIEL ABAD, (THURSDAY, SEPTEMBER 3, 2009)
[12] Andrei Costea, Adrian Ionescu ,(August 2012) "Query Optimization and Execution in Vectorwise MPP"
[13] Choenni, S., Kersten, M., Van den Akker, J., and Saad, A , "On multi-query optimization"
[14] Researchers LadjelBellatreche, Arnaud Giacometti, Dominique Laurent, Patrick Marcel, HassinaMouloudi (2005) , "OLAP query Optimization: A Framework for Combining Rule-Based and Cost-Based Approaches"
[15] PanosKalnis and DimitrisPapadias , "Multi-query Optimization for On-Line Analytical Processing" (November, 2001)
[16] Michael L. Rupley(2008) , "Introduction to Query Processing and Optimization"
[17] "OLAP query optimization: A Framework for Combining Rule-Based and Cost-Based Approaches" Ladjel Bellatreche , Arnaud Giacometti , Dominique Laurent,Patrick Marcel , Hassina Mouloudi
[18] "Efficient Computation of Subqueries in Complex OLAP", Michael O. Akinde, Michael H. B¨ohlen

Links:
[19] http://www.webopedia.com/TERM/D/database.html
[20] http://wiki.answers.com/Q/What_is_a_database_query
[21] http://cisnet.baruch.cuny.edu/holowczak/classes/9440/queryprocessing/connolly/
[22] http://www.databasteknik.se/webbkursen/relalg-lecture/index.html
[23] http://en.wikipedia.org/wiki/Query_optimization
[24] http://www.cs.rochester.edu/~nelson/courses/csc_173/relations/algebra.html
[25] http://en.wikipedia.org/wiki/Relational_algebra
[26] http://krisvenky.tripod.com/id13.html
[27] http://www.slideshare.net/signer/query-processing-and-optimisation
[28] An Introduction to Database Systems, (Eighth Edition) By C.J. Date, A. Kannan, S. Swamynathan

