A STUDY OF VARIOUS PARAMETERS AND TECHNIQUES APPLICABLE IN DATABASE AND QUERY OPTIMIZATION

Md. Mustfa Alam ¹, Dr. Narendra Kumar², Dr. Saket Bihari Singh³

¹Research Scholar, Magadh University, Bodh-Gaya
²Assistant Professor, Deptt. of Computer App. and IT, A. M. College, Gaya
³Professor (Retd.), Mathematics, Magadh University, Bodh-Gaya

ABSTRACT

Today business environment is changing its nature of operations day by day and it is largely dependent on the database that serves as a backbone for any types of enterprise business need and industrial activities, such as production, sales, advertising, marketing, different sorts of reports, charts and other business requirements. The distributed database, complexities of database design and large numbers of attributes of a particular operation, dynamic SQL queries, etc., are proving costlier than before. Access of data in different forms and from different sources in a optimized or tuned environment results into low consumption of computing resources. Writing SQL queries in an optimized way does not always proved beneficial because the database specific query execution plans carry greater importance, therefore, database server specific optimization is also equally important while we consider swift query execution. Database design is also one of the most important factor while we ponder about the query performance and reasource consumption issues in any Database Management System.

In this paper, we studied, examined and explained what sorts of DBMS tools, techniques and logical parameters are applicable to enhance the query execution in a distributed database environment.

Keywords: Query Optimization, Database Design, Query Optimizer, Execution Plans, Database Server, Sub-Queries, Query Response Time, Database Optimization, Logical Design, Crystal Design, Dynamic Queries, Resource Consumption, Joins.
[1] Introduction to Database and Query Optimization:

It could either be database optimization or SQL-Queries tuning, the purpose of both the techniques are same, i.e., reducing response time and consumption of less computing & network resources. The optimization refers here to develop and choose most suitable and efficient execution-plan for optimal response time at minimum cost. There are a number of optimization techniques and logical parameters to tune and optimize the SQL-Queries and Database frequently used in many database servers for data access in a variety of business applications.

For the database optimization, many internal database server tools and techniques (options) are available. Some of them functions intrinsically while others have to be administered by the database administrator to enhance query performance. For the query response-time enhancement, the programmer level roles become more important because of the fact that the nature of query largely affects the query-response time and resource consumption. Also the design of database has a great impact on the performance of database server, and hence query response-time, computing resource and band width.

However, the performance issues also vary greatly with internal operational procedures of various Database Servers, such as, Oracle, MS-SQL Server, Ingress, Delphi, MaxDB, MySQL, etc. Due to the different operational levels of different database servers, one cannot be assure about the performance of optimized structure on one database server will results into the same performance as and when we migrate data to the other database server. The optimization techniques and query tuning both can be discussed and analyzed separately for the better understanding and results.

Many other parameters also influence the Query response-time and resource consumption, such as database growth, change in the tables and/or database configurations, changes in join parameters, cardinality, update in indexes, constraints, etc. Besides, iterative query structures, supply of dynamic queries by the users, network band width, etc., also plays important roles over the performance of database.
Fig. 1. A Typical Working of Database Performance Tuning and Optimization

Source:” http://commerce-data.com/services/additional-services/performance-tuning-and-optimization”

[2] Database Optimization Techniques: The database optimization techniques and procedures are analyzed; however, techniques, procedures and their related outcome may vary greatly with various database servers. Database optimization or tuning can be achieved in a variety of ways. These are discussed as below:

2.1 Data Modeling: The database design starts with the conceptual and logical data modeling in which all the relations, attributes and relationships keys are identified and modeled irrespective of database technology. Prior to these conceptual modeling is done where non-technical terms are used with high-level of data constructs. Conceptual data modeling is a pure architectural description of a business database, The purpose of all the data sorts of modeling is to minimize time and resource consumption. The following pointes are very imperative to be discussed and analyzed.

2.1.1 Conceptual Data Modeling: Conceptual data modeling deals with the wide business concept, further scope, organization of functions and business rules, etc. In aggregate form, we can says
what a DBMS system for a particular company/enterprise will contain. In this section of modeling, the following points can be mention out here:

- **EAR (Entity-Attribute-Relationship):** The main objective of conceptual data modeling is to identify and establish the organizational entities, their attributes and relations among them.
- **EAR** is entirely independent from hardware technologies, DBMS System and its specifications, etc.

Conceptual data modeling provide wide exposure of the working system, business rules and major entities and is usually performed by Database Architect and Business stakeholders. In case of OOM, UML, Document Based Databases, etc; all these modeling has to apply conceptual data modeling in anyway.

### 2.1.2 Logical Data Modeling:

In this modeling, the rules, data structures, validations, technical mapping, etc are described and defined for system implementation, irrespective of a specific database server. The logical data modeling basically provide a strong foundation that is to be implemented in physical data modeling. Attributes of data, their types and related precisions are defined at this modeling level. The normalization process are usually adopted till the 3\(^{rd}\) NF.

During logical modeling, no any sorts of keys are defined; however, the connector or normalizations details are verified for better execution and performance purposes. This phase is usually handled by Data Architect.

### 2.1.3 Physical Data Modeling:

This is the phase where Database Administrator comes into action for actual implementation of database using a specific database server. During the physical modeling, the visualization of database is done by replicating key attributes, triggers, indexes, ranges, constraints and others DBMS features. The main features of Physical modeling can be pointed out as below:

Md. Mustfa Alam Dr. Narendra Kumar Dr. Saket Bihari Singh
• This model is specific to a Project, however, may be integrated with other physical model as needed.
• Relationships and cardinality are conspicuously reflected. Though, relationships and key attributes might be changed as described in the logical model for better performance.
• This model is developed for a specific Database server technology. If implemented with other, it might need some major changes.
• The entire key attributes, such as primary key, foreign key, data ranges, indexes, access profile, authentication profiles, views, and other objects are defined during the implementation of database structure.
• Finally entities are converted into tables while relationships are as foreign keys. All the attributes are created as columns.

2.1.4 De-normalization during Physical Implementation: The normalization done during the early design processes might requires de-normalization or change in the normalized relationships during the actual physical modeling and implementation with a specific database server. Joining orders may also be changed or manipulated based on the physical environment.

2.2 Query Optimizer: Query optimizer is aim to generate most efficient, optimal and cost-effective execution plan for a specific SQL statement. The optimizer compare a number of execution plans, called candidate plans, and among them it choose best execution plan among them after calculating the cost of query based on various statistics.

2.2.1 Components and Operations: The query optimizer includes three basic components:

• **Query-Transformer:** The optimizer checks and determines whether it is useful to change or reconstruct the SQL-Queries to get better execution plan. If so, the optimizer calculates and chooses the most cost-effective alternative to generate execution plan.

• **Estimator:** This component makes an estimate of cost about each candidate plan that is based on the statistic available in data dictionary.
- **Plan-Generator:** Finally query optimizer compare all the candidate plans and select best and lowest-cost plan, called execution plan.

![Components of Query Optimizer](https://docs.oracle.com/database/121/TGSQL/tgsql_optcncept.htm#TGSQL192)

**Fig. 2.1** Components of Query Optimizer

Source: “https://docs.oracle.com/database/121/TGSQL/tgsql_optcncept.htm#TGSQL192”

Here it is imperative to discuss the following techniques used by the query optimizers to deal with the database and SQL-Query optimization.

2.2.2 **Cost-Based-Optimization:** Query optimizer is a part of database server, and it has access to a number of database tools, internal statistics, cost of previously accessed data, etc., therefore, it is always in better position to determine most efficient and optimal query plan than users or programmers to execute SQL-Statements.

**How it works:**
- Since SQL-Statements are non-procedural, the optimizer is free to unify, restructure, and process in its own selected order.
- Based on the statistics that has already been available about accessed data, the optimizer optimizes all the SQL – Statements.
The optimizer examines the following and determine the optimal execution plan:

- Various Access Methods
- Diverse Join Methods
- Multiple Join Orders
- Any possible transformation
- The query optimizer allocates a relative cost to each step of all possible plans in form of a numerical value. Then, optimizer factors all these relative cost to produce an overall cost estimate of plan. By doing so, the optimizer calculates cost of all candidate plans and choose most cost-effective one as execution plan.
- Cost of a query is usually measured with their factors, such as, Input / Output, microprocessor and communication needed.
- The recent versions of Oracle Database Server, such as, 12C, 18C, 21C, the optimizer might enforce different decision because of better information availability and more possibility of optimizer transformations.

The candidate plans or statistical values may vary greatly with various Database Server also because of their different nature and techniques to create execution plan of a given query.

2.2.3 Rule-Based-Optimization: Database engines basically make use of certain rules to take decision about how to execute a query or stored procedure. This type of optimization is commonly known as heuristic optimization. For example, if there is an index, the RBO rules would use that index. The RBO was only optimization technique available in old versions of ORACLE while SQL-Server uses the same concept of optimization as Table Hints.

2.2.4 Adaptive-Query-Optimization: This is relatively new feature added in new versions of Oracle which enable query optimizer to make run-time alteration and adjustment to the execution plans. It is beneficial in case when the existing or available statistics are not enough to derive and determine an optimal execution plan.
2.3 **Programmer/Developer Level Optimization:** Various programmer level optimization techniques are also adopted which help to determine and create optimal plan. Some major techniques are:

- Specification of field-name(s) instead of * after Select.
- Use of indexes
- Maximum possible use of Where clause in place of Having
- Avoidance of linked sub-queries
- Use of temporary table wherever applicable.
- Avoid executing queries under loops.
- Use of wildcards
- Avoidance of DISTINCT and GROUP BY clause
  - Use of Exist instead of IN
  - Use of Multi-Column Indexes
  - Use of Partial Indexes
  - Avoidance of dynamic queries
- Avoidance of function placement at RHS of operator.

[3] **Automatic Query Optimization:** It is a database feature, particularly added in SQL-Server 2017, which provides imminent into possible performance problems of query execution. This database feature notifies programmer/developer as and when any kind of potential performance
concern is detected, and also lets us to make them correct or provide option to be automatically fix problems by database engine.

[4] Algebraic Query Optimization: Relational algebra is perhaps most important query optimization tool that has endless potentials. It’s a tool that has abundant potential and flexibility to model and write SQL queries which might prove ultimate and lets DBMS system to generate maximum possible optimal plans. We can analyze and convert equivalent relational-expressions to generate minimal equivalent algebraic expression.

Usually we apply following operators to extract results using database queries.

- Union
- Intersection
- Projection
- Join
- Selection
- Minus, etc.

The expression of relational algebra has very high impact on the performance issues of database queries. For better understanding it is necessary to explain the algebraic expression logic. Some of the expression are analyzed below:

4.1 Conjunctive Cumulative Selection Operations may be written as individual selections. This is commonly known as sigma cascade. For example:

\[ \sigma X_1 \cap X_2 (E) = \sigma X_1 (\sigma X_2 (E)) \]

When we apply X1 intersection X2, it is too expensive in term of data retrieval. We have option to filter out tuples satisfying condition X2, i.e., inner selection, and then we can apply X1 which is outer selection to the filtered out fewer tuples. The same can be implemented to two or more intersecting sections. Hence, we break a single condition into a string of selections or cascades. That is why we call it as sigma cascade.

4.2 Selection is always Commutative. For example:

\[ \sigma X_1 (\sigma X_2 (E)) = \sigma X_2 (\sigma X_1 (E)) \]
Sigma condition is always commutative in behavior and nature. This implies that applying first either X1 or X2 is immaterial. Hence it is always better and optimal to apply first that yields less number of tuples. This process save time factor and hence computing resources.

[5] Hazards:
Wrong procedure or logical mistakes may lead to many consequences over database performance. Some of the problems may have catastrophic effect that may cause huge loss of data and database server failure.

For example, suppose we write an inner query with the expected return of 0 to 1000, and after sometime it might be possible that the database grown massively and resulting inner query may return billions of records or data that may prove catastrophic effect in two ways:

- The Query Optimizer may not generate execution plan and raise error
  Or
- The database engine may shut down

Likewise several other kinds of performance issue may occur due to inefficient query and logical mistakes.

[6] Summary
The database optimization and query tuning is becoming more and more imperative and critical day by day due to increasing business data loads. The performance issues are largely being handled by database tools itself, such as query optimizer and resource pooling, however, some major responsibility like design and development of database depends on Database professionals, such as data architect, database developer and programmers. In this paper, we have studied and analyzed various processes and techniques to optimize the database and tune SQL-Queries. We also assessed that the algebraic expression has high potential in this regard. Perhaps most perfect and potential tool to optimize database performance is relational algebra. Using algebraic expression, there could be lots of potential applications in database optimization process and SQL tuning.
References / Bibliography


