# DATABASE OPTIMIZATION TECHNIQUES

**Kavya Kumar Vallurupalli**

## Introduction

Along with the usability, flexibility, planning and the view of a web application, one of the main standard that marks an application as a good application is its performance. Optimization of performance assures the improvement of an application without making major changes to the initial base. Since web applications are mixtures of both the client side and the sever side, identifying the correct areas to optimize performance will help in achieving maximum performance. Individual categories can also be targeted for performance optimization. One such area where performance can be optimized are the database systems.

## What is a database and why do we need it in a web system?

A database is a collection of data that can be accessed for searching for information, modifying the data, extracting data, creating new data or deletion of data. The history of databases began from a simple database management system to Sql based relational databases to Object Query Language and finally the unstructured databases with the creation and identification of networks and hierarchal co relation between the data.

An appropriate data storage ensures the smooth running of the application. Apart from reducing the time taken to access and perform operations on the data, an optimized database minimizes the scope of error by improving the concurrency and consistency of the data. A web system relies on this data to maximize the efficiency of its purpose.

## Database Optimization Techniques

### Structured Databases

### 1. Indexing in A Database:

Indexing speeds up the process of accessing data by reducing the number of block access required to acquire the data. The schema that stores the indexes is independent of the data stored and its sorting order. It only affects the execution speed.

The main types of indexing methods:

**Clustered**:

The data block in sorted sequentially and only one clustered index is created for the data block. Usually the sorting is done in the row order among the data stored in the table/ data block.

**Un clustered**:

The data is sorted in an arbitrary order, but the indexes contain a logical order. When a query is asked the indexes guide the query to look at the leaf index which contains the pointer to the data that is required. The indexes are the non-primary key columns. There can be more than one non-clustered index per table.

Another way to increase the efficiency of an index is to create a covering index. Incase a query is to be repeated regular, a covering index can be created which contains all the columns referenced in that query. This eliminates the need to navigate to the actual data.

**2. Partitioning of Data**

The method of storing data has a significant effect on the time taken to access the data. One way to do this is to partition the data into smaller segments allowing the amount of data to be accessed by a query to decrease.

**Vertical Partitioning:**

The tables are partitioned column wise. This reduces the overhead time taken to access the data by reducing the amount of irrelevant data that is access when the query is run. For example, in an employee database only the employee number is needed in most cases. Splitting the table vertically and storing the employee details in another table reduces the number of logical reads since the data that is access is less.

There are two types of vertical partitioning:

1)Normalization

It is the process of removing redundant data and putting them in other tables. Norm forms are used to eliminate redundancy. There are 4 main forms. 1NF,2NF,3NF and BCNF.

2)Row- Splitting

Dividing the table into fewer columns vertically and they can be grouped using the Primary key or a Unique key that allows correspondence between the rows in both the tables.

**Horizontal Partitioning:**

The tables are partitioned horizontal. Each of the table has the same number of columns. But the number of rows is reduced significantly. For example, storing the sales report data of a company month wise. This assures that the overhead time is reduced since the amount of data to be accessed is less.

**3.Use of Transact – SQL**

An extension to SQL which includes procedures and allows in built processing like date manipulation, data processing and changes to the basic SQL statements. The major ability of a TSQL is to be able to pass parameters.

One way to improve optimization is to move the TSQL codes to the database server allowing them to act as a central point. This allows code refactoring to be possible, which in turns allows the TSQL to take advantage of the indexes. Queries can be rewritten to improve the performance of the database. It will need an idea about query optimization [2] and execution engine internals.

For Example: -

SELECT order Name FROM order WHERE orderId IN (12,13,14,15);

Can also be written as follows:

SELECT orderName FROM order WHERE orderId = 12

OR

orderId=13

OR

orderId=14

OR

orderId=15;

Analysis:-

Both the queries return the same results. But the query syntax can affect the execution plan of the query optimizer even though the queries logically specify the same result.

Using TSQL helps to lay the base to apply optimization techniques to enhance performance.

Best Practices of TSQL:

1. Do not use 'Select *' since it may lead to retrieval of irrelevant columns that adds expense to the data retrieval step.

2. Use EXISTS for existence checks instead of the aggregate COUNT (). This reduces the lookup time since true is returned on the first match that is found using EXISTS as opposed to checking for all the values done by count.

3.Avoid joining of columns of two different data types. The lower data type is converted in such a case and is rendered useless for index creations.

4.Transactions must be short, and all the tables accessed in a TSQL but be done in the same order.

5.Avoid using LIKE for text search and opt for full text search.

6. If sorted data is not priority use UNION ALL instead of UNION since it is faster.

**4.Query Optimization:**

An optimizer should be able to handle a large amount of data. Optimization becomes the key for large data due to the joins, sub-joins, and grouping. This data cannot be cached at the client side. For such a case an effective paging query concept can be used to handle the large amounts of data. Sun et a [2] proposed a technique by using a paging query solution. All the data satisfying the query conditions are placed in the server memory. Only a part of this data is cached in the client side and is passed to the client.

In relational data bases, sub-queries can be used to handle queries which need more information to gather the data asked of it. Since they are co-related queries, ensuring that each query is optimized in its own will improve the overall optimization. Li et al., [5] proposed a method to enhance query optimization by using heuristic strategies. One of the strategies was to perform selection options to limit the number of tuples to be processed. Another one aimed at limiting the number of columns by projections. Distributed

query processing aims to tackle the data in distributed locations. Optimization in such cases can be divided into two aspects.

1. Achieve minimal cost including the CPU, I/O, and data transmission cost.

2.Achieve minimal processing time by increasing the possibility of parallel processing.

**Unstructured Data:**

One of the costly areas when it comes to building an application is storage. Often it surpasses the cost of the technology. Hence storage optimization can help in reducing cost efficiently.

One way to do this is to distribute data across cloud storages while reducing redundant data in turn reducing the data volume.

Meta data can be used to identify data and move it from tier 1 to tier 2 at an appropriate stage thus reducing the cost of the primary and backup storage. One such optimizer is the HPE storage optimizer than aims at intelligently controlling data volume and efficiently manage data across the entire enterprise.

A paper written by Peter Bueman [3] introduced a data model with a simple query language for transforming labeled tree structures. The paper concentrated on improving syntax for deep learning queries. Querying the tree by using some graph logic will aid to optimization of the query structure.

**Conclusion**

Apart from traditional database optimization techniques, many new techniques are being presented with the aim of providing an optimized storage system . While structured data can be stored in a standard form , with concentration of the storage design along with optimized query structures, optimization can be attained to a fair degree. But in case of unstructured data , since there is no definite form it can be stored it ,research is still ongoing . As of now storage options are being advocated for unstructured data .

Query optimization research is still an active field. Aims include the development of realistic cost estimates, the optimization of queries on databases with deductive or computational capabilities, and the simultaneous optimization of multiple queries and update transactions. Although a complex field, along with performance tuning, query optimization by far is the field which provides the most scope of performance enhancement.

Overall database optimization improves the performance of an application on both the client and server side as well as plays a significant part in reducing the cost heuristics.

**References**


[1] MATTHIAS JARKE. Query Optimization in Database Systems . web.eecs.umich.edu/~jag/eecs584/papers/qopt.pdf.

[2] D. Li, L. Han and Y. Ding, "SQL Query Optimization Methods of Relation Database System", Computer Engineering and Applications (ICCEA), (2010)

[3] Buneman, Peter, et al. "A Query Language and Optimization Techniques for Unstructured Data." *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data - SIGMOD 96*, 1996, doi:10.1145/233269.233368.

[4] Syedur Rahman, et al. Analyze Database Optimization Techniques.

[5] F. Sun and L. Wing, "Paging Query Optimization of Massive Data in Oracle 10g Database", Computer and Information Science and Service System (CSSS), IEEE International Conference, (2011).

[6] https://sqlperformance.com/2014/09/sql-plan/rewriting-queries-improve-performance

[7] https://en.wikipedia.org/wiki/Transact-SQL

[8] https://en.wikipedia.org/wiki/Transact-SQL

[9] http://www.differencebetween.net/technology/software-technology/difference-between-logical-and-physical-database-model/

[10] https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=1251&context=libphilprac

[11] Alireza Isfandyari Moghaddam. "Databases: From Paper-Based to Web-Based."