An Empirical Evaluation of LIKE Operator in Oracle

Manoj Kumar Gupta[1] and **Pravin Chandra**[2]

Submitted in November 2010 ; Accepted in February in 2011

Abstract - ***In database systems, user makes query and that query will be responded by the DBMS. Generally, there are a variety of methods for computing the response of the given query. It is the responsibility of the query processor to transform the query as entered by the user into an equivalent query that can be computed more efficiently. Query optimization is the process to find a good strategy or best query evaluation plan for processing a query. In the today's competitive environment, query optimization is one of the important criteria based on which one can compare the available commercial RDBMSs. The objective of this study is to discuss about techniques used by the Oracle for optimizing the queries and to present a comparative study of the various costs involve to execute the LIKE Operator based queries. This comparative study is based on empirical study done on Oracle 8i, Oracle 9i and Oracle 10g.***

Index Terms - ***Query Optimization, Oracle, Cost Control, LIKE Operator, Pattern Matching in DBMS***

1. INTRODUCTION

Query optimization is one of the important issues in database systems. A query may be expensive in terms of cost of execution if it is not optimized. In centralized database management systems, an efficient query processor would try to minimize the utilization of computing resources, such as storage space and processor time. In distributed environment, apart from the storage space and processor time; the costs of communication delays, setups and transmission have to be minimized. *Total cost* and *response time* are the good measures to compare the cost of queries in terms of resource consumption. [DB01]

Oracle is one of the most popular and efficient commercial RDBMS. Oracle claims that it uses both rule-based query optimizer and cost-based optimizer. The goal of the cost-based optimizer is the best *throughput (i.e.* the least amount of resources necessary to process all rows accessed by the SQL statement). Also, Oracle claims to optimize a statement with the goal of best *response time (i.e.* the least amount of resources necessary to process the first row accessed by a SQL statement). In general, it uses the cost-based approach. Oracle Corporation is continually improving its cost-based optimizer. The rule-based approach is available for backward compatibility with legacy applications. [WO01]

The objective of this study is to present the comparative performance of LIKE Operator of query optimizers used in Oracle 8i, 9i and 10g by using large volume of hypothetical data.

[1]*Associate Professor, Rukmini Devi Institute of Advanced Studies, Delhi*

[2]*Professor, University School of Information Technology (USIT) and Controller of Examinations, GGSIP University, Delhi*

E-mail: pc_ipu@yahoo.com and manojgupta5@yahoo.com

2. METHODOLOGY

This study is based on hypothetical data which is generated using an algorithm to generate random strings of variable length. The table is populated with 11 columns and 10^5 records. The table contains the strings based on all alphabets and blank space of maximum of 100 characters in each column.

For the analysis, the queries based on LIKE predicated are executed on the table (both without index and with index) on different versions of Oracle. This study broadly covers the three versions of Oracle, i.e., Oracle 8i, Oracle 9i, and Oracle 10g. The query execution plan and response time are observed and analyzed with help of different tools of Oracle.

3. THEORETICAL ASPECT OF QUERY OPTIMIZATION

Query optimization is the process to derive a number of query-evaluation plans to execute the query and selects the most efficient plan. It is the responsibility of the query optimizer to come up with a least-cost query-evaluation plan that computes the same result as the given relational-algebra expression (or, at least, is not much costlier than the least-costly way). There are several optimization criteria that have to be taken into the consideration at the time of optimization. [GH01] [GH02]

To find the least-costly query-evaluation plan, the optimizer needs to generate alternative plans that produce the same result as the given expression, and to choose the least-costly one. Generation of query-evaluation plans for and expression involves three steps:

1. Generating logically equivalent expressions using equivalence rules
2. Annotating resultant expressions to get alternative query plans
3. Choosing the cheapest plan based on estimated cost

a. **Rule-Based Optimizer:** Rule-based optimizer generates the equivalent optimal query evaluation plan by using the *equivalence rules* for the given relational algebraic query. It generates expressions equivalent to a given expression by means of *equivalence rules* that specify how to transform an expression into a logically equivalent one. *The optimization based on equivalence rules is very expensive in space and time*. [SK01]

b. **Cost-Based Optimization:** A cost-based optimizer generates a range of query-evaluation plans from the given query by means equivalence rules, and chooses one with the least cost. In general, with *n* relations, there are (2(n-1))! / (n-1)! different join orders. Brute-force method introduce large overhead in the optimization process because it will evaluate the cost of each evaluation plan separately, compare their costs and selects the least cost query-evaluation plan. This overhead can be reduced by applying the theory of ***dynamic programming,*** which can also be used for finding optimal query-evaluation plan optimistically. [SK01] The cost of an operation depends on the size and order statistics of its inputs. To estimate the cost of an operation some statistics about database relations are required, which are stored in database-system catalogs. The statistics has to be updated every time a relation is modified so that accurate statistics can be maintained. The updation of statistics may incur a substantial amount of overhead. [GH01] [GH02]

c. **Heuristics-Based Optimization:** The cost of optimization is the major drawback of cost-based optimization even with dynamic programming. The number of choices can be reduced by using *heuristics* that must be made in a cost-based fashion which will reduce the cost of optimization. Some systems use only heuristics; others combine heuristics with partial cost-based optimization. [GH01] [GH02]

d. **Materialized Views:** Materialized view is one of the concepts which can also be used for query optimization. A materialized view is a view whose contents is computed and stored, which can be used to speed up query processing e.g. indices. If base relations are modified then incremental maintenance is needed to efficiently update these views. In query optimization, materialized views are treated just like regular relations. [GH01] [GH02] Most database systems provide tools to help the database administrator with index and materialized view selection. These tools examine the history of queries and updates, and suggest indices and views to be materialized. The Microsoft SQL Server Database Tuning Assistant, the IBM DB2 Design Advisor, and the Oracle SQL Tuning Wizard are examples of such tools.

4. QUERY OPTIMIZATION IN ORACLE

A large variety of processing techniques are supported by Oracle in its SQL processing engine. SQL processing engine has four main components: parser, optimizer, row source generator, and SQL execution engine (*figure 1*).

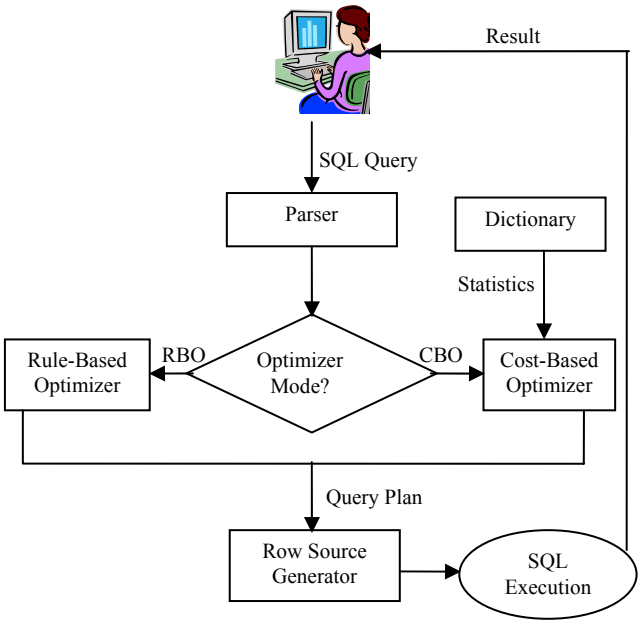

Figure 1: SQL Processing Architecture (adapted from [WO01])

The syntax and semantics analysis of the SQL statements is done by parser. The optimizer uses internal rules and/or costing methods to determine the most efficient way of producing the result of the query. The optimizer returns an optimal query plan for execution. The Oracle provides two types of optimizers: cost-based optimizer (CBO) and rule-based optimizer (RBO). The execution plan for the SQL statement is generated by the row source generator with the help of the query plan received from the optimizer. The execution plan is a collection of row sources structured in the form of a tree. Each row source returns a set of rows for that step. Each row source produced by the row source generator is executed by the SQL execution engine to produce the results of the query.

By default, optimizer mode is cost-based optimizer with the goal of *best throughput.* Also, Oracle can optimize a statement with the goal of *best response time*. In general, Oracle uses the cost-based approach but it also supports rule-based approach. *Oracle Corporation is continually improving the cost-based optimizer and adding the new features which will work only with the cost-based optimizer.* The rule-based approach is available for backward compatibility with legacy applications [WD01].

The cost-based architecture is shown in *figure 2*. By using the parsed query, the query transformer to determines if it is advantageous to change the form of the query so that it enables generation of a better query plan. *Four different query transformation techniques are employed by the query transformer: view merging, predicate pushing, sub-query unnesting, and query rewrite using materialized views.* [GH01] [GH02] Any combination of these transformations might be applied on the received parsed query.

The estimator generates three different types of measures: selectivity, cardinality, and cost. The selectivity, which represents a fraction of rows from a row set, lies in the value range 0.0 to 1.0. There are several types of cardinality measures: effective, join, distinct, and group cardinality. The cost represents units of work or resource used in performing an operation. The cost-based optimizer uses disk I/O, CPU usage, and memory usage as units of work. The plan generator computes the cost of different possible plans for a given query and selects the one that has the lowest cost.

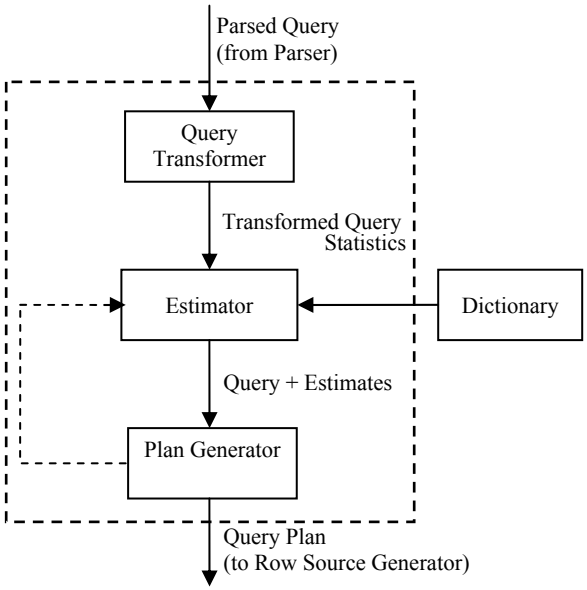

Figure 2: Architecture of Cost-Based Optimizer (adapted from [WO01])

In Oracle, large set of processing techniques is used in one engine. Many different join and index methods, parallel query, etc. e.g., nested loops, sort-merge, and hash joins, anti-joins, semi-joins, B-tree, bitmap, reverse, functional, and domain indexes, clusters and hash clusters, index-organized tables, nested tables, materialized views, partitioning, join indexes, index joins, index skip-scan, etc. are used. No single technique is best for everything. Oracle provides many different ones, and the optimizer determines which the best for each individual query is. [WD01]

5. EXPERIMENT

5.1 The Environment

All the experiments are performed on the same machine in which Microsoft Windows 2000 Advanced Server Version 5.0 OS with Service Pack 4 is installed. The experiments are performed on three different versions of Oracle that is Oracle 8i Release 8.1.7.0.0, Oracle 9i Release 9.0.1.0.0 and Oracle 10g Release 10.1.0.0.0

5.2 The Queries

A table consisting of 11 columns of VARCHAR2 data type is used for the experiment. Each column can store a string of maximum 100 characters. The table is populated with 100000 records. The following 8 queries are used in the experiments for observing the EXPLAIN PLAN and SQL Trace results. Experiments are performed using both without index and with index.

Query No.	SQL Statement
1	`SELECT C1 FROM T1 WHERE C1 = 'VMPNODLHHCRWPEPSABCDXDKQYKWYDGFSFTHNNQGDQ';`
2	`SELECT C1 FROM T1 WHERE C1 LIKE 'VMPNODLHHCRWPEPSABCDXDKQYKWYDGFSFTHNNQGDQ%';`
3	`SELECT C1 FROM T1 WHERE C1 LIKE '%VMPNODLHHCRWPEPSABCDXDKQYKWYDGFSFTHNNQGDQ%';`
4	`SELECT C1 FROM T1 WHERE C1 LIKE '%VMPNODLHHCRWPEPSABCDXDKQYKWYDGFSFTHNNQGDQ';`
5	`SELECT C1 FROM T1 WHERE C1 = 'ABC';`
6	`SELECT C1 FROM T1 WHERE C1 LIKE 'ABC%';`
7	`SELECT C1 FROM T1 WHERE C1 LIKE '%ABC%';`
8	`SELECT C1 FROM T1 WHERE C1 LIKE '%ABC';`

Table 1: List of SQL Statements used for Experiments

6. COST COMPARISON AMONG ORACLE 8i, 9i AND 10g

The costs of the queries are compared on the basis following 5 parameters:

CPU Time = CPU Time in seconds executing
Elapsed Time = Elapsed Time in seconds executing
Disk = Number of physical reads of buffers from disk
Query = Number of buffers gotten for consistent read
Current = Number of buffers gotten in current mode (usually for update)

The various measured costs of above mentioned eight queries are represented in the Table 2 to Table 9 and Figure 3 to 10 respectively. The overall measured cost of all eight queries is represented in Table 10 and Figure 11.

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	900	8900	3480	7460	400
With Index (in 8i)	0	500	2	3	0
Without Index (in 9i)	1200	12900	10361	15964	1200
With Index (in 9i)	0	100	2	3	0
Without Index (in 10g)	600	17500	0	7521	0
With Index (in 10g)	0	2800	0	3	0
With Bitmap Index (in 9i)	0	0	0	3	400
With Bitmap Index (in 10g)	0	6000	0	3	0

Note: *** - denotes value of respective column is multiplied by 100**

Table 2: Cost Comparison of Query 1

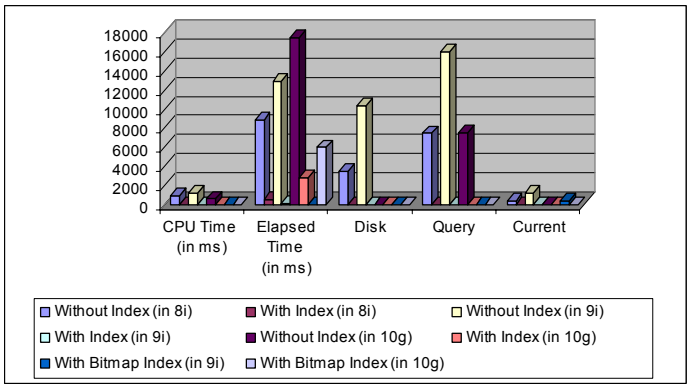

Figure 3: Cost Comparison of Query 1

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	800	8600	3480	7460	400
With Index (in 8i)	0	0	0	3	0
Without Index (in 9i)	600	12500	10361	15964	1200
With Index (in 9i)	0	0	0	3	0
Without Index (in 10g)	600	600	0	7521	0
With Index (in 10g)	0	600	0	3	0
With Bitmap Index (in 9i)	100	100	0	3	400
With Bitmap Index (in 10g)	0	400	0	3	0

Note: * - *denotes value of respective column is multiplied by 100*

Table 3: Cost Comparison of Query 2

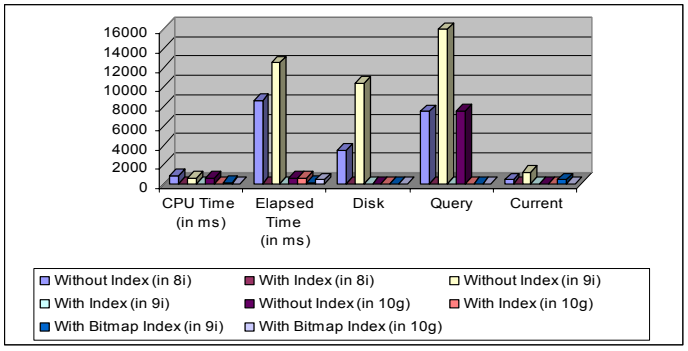

Figure 4: Cost Comparison of Query 2

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	1200	8600	3480	7461	400
With Index (in 8i)	0	2800	876	880	600
Without Index (in 9i)	300	12500	10361	15965	1200
With Index (in 9i)	100	2300	1812	1816	600

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 10g)	3400	3500	0	7522	0
With Index (in 10g)	3200	4000	0	886	0
With Bitmap Index (in 9i)	500	500	5	2077	400
With Bitmap Index (in 10g)	3400	3900	0	1008	0

Note: * - *denotes value of respective column is multiplied by 100*

Table 4: Cost Comparison of Query 3

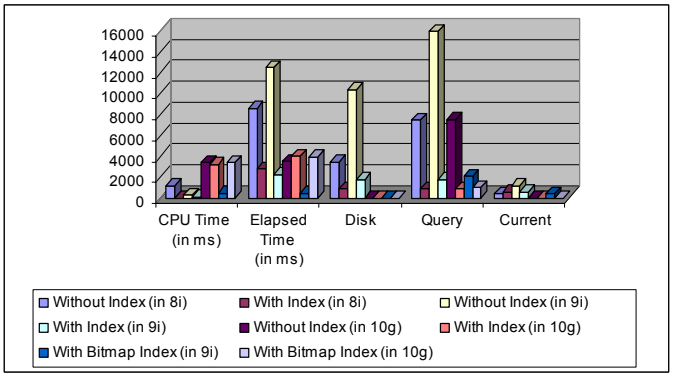

Figure 5: Cost Comparison of Query 3

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	800	8600	3480	7460	400
With Index (in 8i)	100	200	0	879	600
Without Index (in 9i)	700	12800	10361	15964	1200
With Index (in 9i)	100	100	0	1815	600
Without Index (in 10g)	600	600	0	7521	0
With Index (in 10g)	400	400	0	885	0
With Bitmap Index (in 9i)	500	500	0	2077	400
With Bitmap Index (in 10g)	300	600	0	1007	0

Note: * - *denotes value of respective column is multiplied by 100*

Table 5: Cost Comparison of Query 4

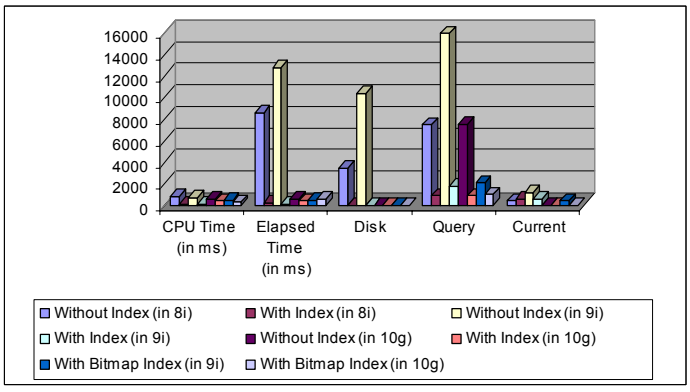

Figure 6: Cost Comparison of Query 4

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	800	8700	3480	7460	400
With Index (in 8i)	0	0	0	3	0
Without Index (in 9i)	1200	12300	10361	15964	1200
With Index (in 9i)	0	0	0	3	0
Without Index (in 10g)	300	300	0	7521	0
With Index (in 10g)	0	0	0	3	0
With Bitmap Index (in 9i)	0	0	0	3	400
With Bitmap Index (in 10g)	0	0	0	3	0

Note: * - denotes value of respective column is multiplied by 100

Table 6: Cost Comparison of Query 5

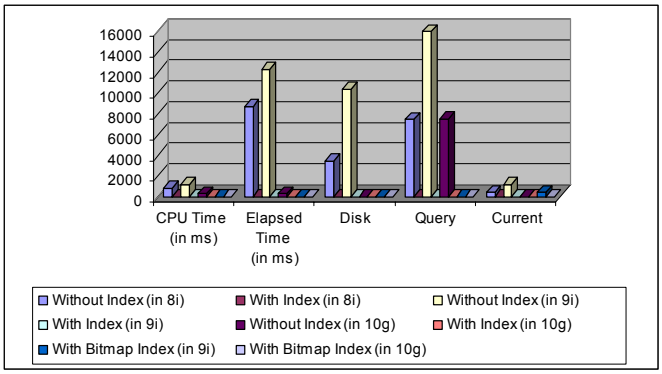

Figure 7: Cost Comparison of Query 5

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	600	8300	3480	7461	400
With Index (in 8i)	100	200	0	880	600
Without Index (in 9i)	1000	12800	10361	15965	1200
With Index (in 9i)	100	100	0	1816	600
Without Index (in 10g)	600	700	0	7522	0
With Index (in 10g)	100	600	0	886	0
With Bitmap Index (in 9i)	0	0	0	3	400
With Bitmap Index (in 10g)	0	400	0	4	0

Note: * - denotes value of respective column is multiplied by 100

Table 7: Cost Comparison of Query 6

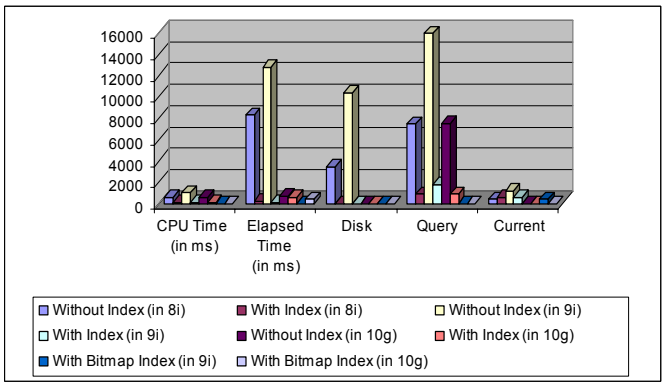

Figure 8: Cost Comparison of Query 6

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	800	7000	3481	7476	400
With Index (in 8i)	800	700	0	896	600
Without Index (in 9i)	1500	13400	10362	15979	1200
With Index (in 9i)	300	300	0	1831	600
Without Index (in 10g)	3200	3600	0	7538	0
With Index (in 10g)	3400	4200	0	902	0
With Bitmap Index (in 9i)	600	600	0	2077	400
With Bitmap Index (in 10g)	2900	3200	0	1024	0

Note: * - denotes value of respective column is multiplied by 100

Table 8: Cost Comparison of Query 7

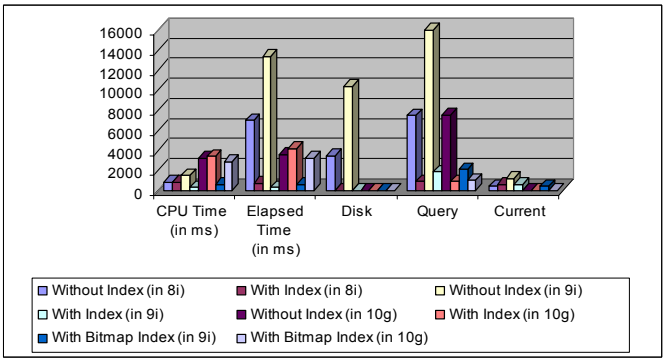

Figure 9: Cost Comparison of Query 7

Cost in	CPU Time (in ms)*	Elapsed Time (in ms)*	Disk	Query	Current*
Without Index (in 8i)	500	8800	3480	7460	400
With Index (in 8i)	300	300	0	880	600
Without Index (in 9i)	1700	12600	10361	15965	1200
With Index (in 9i)	300	300	0	1816	600
Without Index (in 10g)	400	600	0	7522	0

With Index (in 10g)	300	1100	0	886	0
With Bitmap Index (in 9i)	200	300	0	2077	400
With Bitmap Index (in 10g)	300	1000	0	1008	0

Note: ** - denotes value of respective column is multiplied by 100*

Table 9: Cost Comparison of Query 8

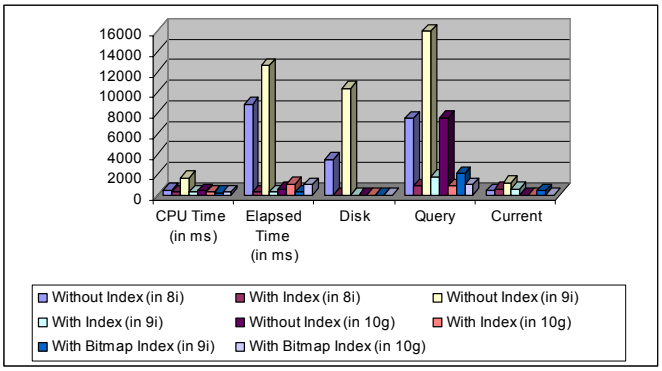

Figure 10: Cost Comparison of Query 8

Total Cost in	CPU Time* (in ms)	Elapsed Time* (in ms)	Disk	Query	Current#
Without Index (in 8i)	6400	67500	27841	59698	32000
With Index (in 8i)	1300	4700	878	4424	30000
Without Index (in 9i)	8200	101800	82889	127730	96000
With Index (in 9i)	900	3200	1814	9103	30000
Without Index (in 10g)	9700	27400	0	60188	0
With Index (in 10g)	7400	13700	0	4454	0
With Bitmap Index (in 9i)	1900	2000	5	8320	32000
With Bitmap Index (in 10g)	6900	15500	0	4060	0

Note: ** - denotes value of respective column is multiplied by 10,*
 # - denotes value of column is multiplied by 1000

Table 10: Comparison of Total Costs of All 8 Queries

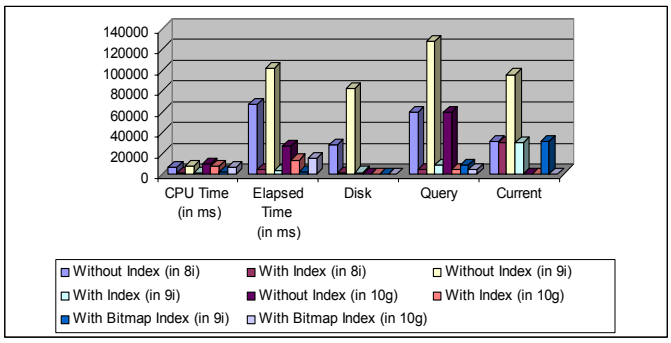

Figure 11: Comparison of Total Costs of All 8 Queries

7. SUMMARY AND CONCLUSIONS

The study is being completed taking into consideration the different parameters which is been mentioned in the objective of my study. It includes both descriptive and empirical study and the conclusions drawn are having far reaching implications. On the basis of the study I had concluded:

a. The cost of FIRST_ROW (response time) and ALL_ROWS (throughput) is same in all these three versions of Oracle. It might be due to in ALL_ROWS Oracle computes the statistics based on block available in SGA.
b. In case of without index table, exact match and pattern matching has same cost in all these three versions of Oracle. Because Oracle performs full table scan (compares with each row) if no index is available on the queried column(s).
c. In Oracle 9i, the cost of evaluation plan is almost double of the cost in Oracle 8i. This cost in Oracle 9i might be decreased if the size of RAM is increased so that the disk swapping can be reduced.
d. In Oracle 10g, estimated statistics and computed statistics shows the different results whereas in Oracle 8i and Oracle 9i both estimated and computed statistics shows the same results. It means Oracle 10g using different approach for estimating and computing statistics as compared to Oracle 8i and Oracle 9i.
e. In Oracle 10g, the cost of execution plan is less than the Oracle 9i but slightly greater than that of Oracle 8i. It may be due to that Oracle 10g is managing the memory more efficiently as compared to the Oracle 9i but Oracle 10g has more overhead than that of Oracle 8i.
f. In case of non-indexed table, CPU time and elapsed time in Oracle 8i is lesser than that of Oracle 9i. The Oracle 9i is basically the rewrite of the Oracle 8i with some more features which involve more overhead.
g. In case of indexed table, CPU and elapsed time in Oracle 9i is lesser than that of Oracle 8i. In Oracle 9i, the indexes are more optimized than that of Oracle 8i.
h. CPU time in Oracle 10g is more than that of Oracle 8i and Oracle 9i but the elapsed time is inversed. Oracle 10g is having the grid support. It assumes that all the queries are distributed queries therefore it requires more CPU time as compared to Oracle 8i and Oracle 9i.
i. In case of bitmapped indexed table in Oracle 9i, elapsed time is lesser than that of normal indexed table but the CPU time is inversed.
j. In case of bitmapped indexed table in Oracle 10g, CPU time is lesser than that of normal indexed table but the elapsed time is inversed.
k. Bitmapped indexes in Oracle 9i are more optimized than that of Oracle 10g.

The above conclusions are based on hypothetical data. These may differ in actual environment with larger volume of data and/or different system configuration. The experiments are

performed using 256 MB RAM, therefore if larger amount of RAM is used than the result may be slightly varied.

Further, the results indicate that the query optimizer performance of Oracle 9i may only be due to a rewrite of the Oracle 8i query optimizer rather than any new algorithm's implementation (as the results do not differ by much). Some more features are added into the Oracle 9i. The indexes in Oracle 9i might be more optimized than that of Oracle 8i. Oracle 10g has grid support which is beneficial in distributed environment. Due the grid support, a lot of overheads are included in the Oracle 10g. Oracle 10g may be assuming all queries to be distributed queries, therefore it requires more CPU time. From the observations of the experiments, it has been noticed that the prefixed substring in LIKE operator requires 'full table scan' irrespective of whether index is used or not. Therefore, there is scope to improve the cost of LIKE '%.....%' by design of new implementations for this operator.

REFERENCES

[1]. [AE01] Aronoff, Eyal, et. el., *Oracle8 Advanced Tuning and Administration*, Tata McGraw Hill, Oracle Press Edition.

[2]. [CM01] Corer, Michel J, et. el., *Oracle8 Tuning*, Tata McGraw Hill, Oracle Press Edition.

[3]. [DP01] Darryl L. Presley, *Query and Application Tuning using Explain and TKPROF Utility*, October 1994

[4]. [DB01] Desai, B. C., *An Introduction to Database Systems*, Galgotia, 2000.

[5]. [EN01] Elmasri, E. and Navathe, S.B., *Fundamentals of Database Systems,* Pearson Education, 2004

[6]. [GH01] George Lumpkin, Hakan Jakobsson, *Query Optimization in Oracle 9i* , Oracle Corporation, February 2002

[7]. [GH02] George Lumpkin Hakan Jakobsson, *Query Optimization in Oracle Database 10g Release 2*, Oracle Corporation, June 2005

[8]. [PS01] Parida, R.A. and Sharma,V., *The Power of Oracle 9i*, Firewall Media, 2007

[9]. [RG01] Ramakrishnan, R. and Gehrke, J., *Database Management Systems,* McGraw-Hill, 2003

[10]. [RP01] R Powell, *Cost Based Optimizer - Common Misconceptions and Problems*, Jan 97 (URL: http://www.download-west.oracle.com)

[11]. [SR01] Schumacher, Robin, *Oracle Performance Troubleshooting with Dictionary Internals SQL & Tuning Scripts*, Shroff Publishers.

[12]. [SK01] Silberschatz, A., Korth, H.F. and Sudershan, S., *Database System Concepts*, McGraw-Hill, 2006

[13]. [SC01] Surajit Chaudhuri, An Overview of Query Optimization in Relational Systems, (URL:ftp://ftp.research.microsoft.com/users/ surajitc/pods98-tutorial.pdf)

[14]. [VT01] Vadim Tropashko, Performance Analysis of Optimizer Plans, (URL: www.oracle.com/technology/deploy/performance/pdf/ Performance_Analysis)

[15]. [WT01] http://whitepapers.techrepublic.com.com

[16]. [WC01] http://www.courses.csusm.edu

[17]. [WC02] http://cisnet.baruch.cuny.edu

[18]. [WD01] http://www.dba-oracle.com

[19]. [WD02] http://www.download-west.oracle.com

[20]. [WO01] http://www.oracle.com

[21]. [WO02] http://www.oracle-base.com

[22]. [WS01] http://www.sqlsummit.com

[23]. [WR01] http://redbook.cs.berkeley.edu