Tactful Management Research Journal
ISSN: 2319-7943
Impact Factor : 2.1632(UIF)

IMPORTANCE OF DATABASE TUNING IN ORACLE 9I FOR E-BANKING

Anusha Suthersan

Ramanad Arya D.A.V College
Bhandup(e)

Abstract : The tremendous advances and the aggressive infusion of information technology has brought in a paradigm shift in banking operations. The adoption of E-banking becomes a necessity for the banking which wishes to maintain its share of market and retain its customers..E-banking is source of transactions for Investment banks, since it overcomes the geographical bars between the countries. In such a banking environment data storage and maintenance becomes the prime factor for all major transactions. Data is growing at a tremendous rate , However the increase in data is, in itself, a minor problem, but the increase in the percentage of the unstructured data in the overall data volume , is what is concerning everyone in banking sector Capital market firms have been transforming organically to handle the big data over years .

This paper discusses the key transformation that investment banking firms like Barclays Capital (India) are undergoing to handle big data, Database tuning for the use of big data technology helps business as usual to work smoothly

Key words:E-banking, b-tree index, Bitmap index, Function-based indexes, Table Clusters.

INTRODUCTION

Data is growing at a tremendous rate with an overall increase in digital universe with petabytes of data flowing from messages, emails, and blogs and so on. According to the studies conducted by top market research firm ,the big data market is expected to grow from US$3.2 billion in 2010 to us$16.9 billion in 2015.The next big challenge for capital markets will be the velocity of data, considering the frequency of data generation & delivery in financial services industries like investment banking and the need to be still analyzed in the real time .Responding to these trends, Capital markets firm are setting aside a bigger pie of IT spend on big data technologies

According to the report by a major analyst firm, using big data analytics in the field of customer intelligence in the investment banking sector in UK has reaped benefits of £554 million and is expected to go up to £5,275 million by the end of 2017.Investment banking in UK are spending 4% of their gross operations surplus on big data R&D.Capital market are also seeking competitive advantages by exploring the plethora of information from all possible sources and by transforming organically to handle the big data

One of the key transformation taking places in the capital market to handle big data is Re-modeling of data models to make better, faster and less risky decisions. The traditional relative database management system is tuned using Index for the faster retrieval of data .The choice of index and their setup depends on type of tables used for data storage in database. This helps the bank to generate reports with accurate figures, with in their time constraint. Automation of the reports for business as usual helps the bank to speed up their day to day task.

1.1 Objectives of the Study

The objectives of the study are to:

•Present the current scenario of Database Management & Tuning in E-banking for Investment banks

•Tuning the logical structure of the Database using INDEX
•Different scenarios for implementing the INDEX.

1.2 Methodology

The study has been conducted mainly on the basis of secondary information for this purpose, Capital Market strategy, rules and regulations, Database management system have been discussed. Various seminar papers and summary of discussions in those seminars, taskforce report of research organizations and some periodicals ,national and international journals, newspaper, magazines and Barclays Capital (Investment bank) Database management system has been analyzed. Some bank specialists have also been personally interviewed in order to collect some primary information used in this study. Besides this, Internet has been used as another source of information.

2. E-Banking (Database Tuning)

A Database is an organized collection of data. For all banks, it is very important to aggregate and manage vast amount of data and accurately reporting their financial positions to both regulatory agencies and the general public.

Data's are stored inside the tables in a Database, the table size increases as data increases.

The performance rate goes down as table size increases; therefore it is very important to tune the database for better performance Tuning using different Indexes on oracle9i server has been analyzed here.

Tuning the Logical Structure

Oracle includes numerous data structures to improve the speed of Oracle SQL queries. Taking advantage of the low cost of disk storage, Oracle includes many new indexing algorithms that dramatically increase the speed with which Oracle queries are serviced. This study explores the internals of Oracle indexing; reviews the standard b-tree index, bitmap indexes, function-based indexes, table clusters and demonstrates how these indexes may dramatically increase the speed of Oracle SQL queries.

To maintain optimal performance, drop indexes that an application is not using. You can find indexes that are not being used by using the ALTER INDEX MONITORING USING FUNCTIONALITY over a period that is representative of your workload. This monitoring feature records whether an index has been used. If you are deciding whether to create new indexes to tune statements, then you can also use the EXPLAINPLAN statement to determine whether the optimizer chooses to use these indexes Oracle uses indexes to avoid the need for large-table, full-table scans and disk sorts, which are required when the SQL optimizer cannot find an efficient way to service the SQL query. I begin our look at Oracle indexing with a review of standard Oracle b-tree index methodologies.

2.1 Indexing Correct Tables and Columns

The important guidelines for creating the index:-

We have to create an index if we want to retrieve less than 15% of the rows in a large table. The percentage varies greatly according to the relative speed of a table scan and how the distribution of the row data in relation to the index key. The faster the tables scan, the lower the percentage, the more clustered the row data, the higher the percentage.

To improve performance on joins of multiple tables, index columns used for joins.

Primary and unique keys automatically have indexes, but you might want to create an index on a foreign key.

Small tables do not require indexes. If a query is taking too long, then the table might have grown from small to large.

Some columns are strong candidates for indexing. Columns with one or more of the following characteristics are candidates for indexing:

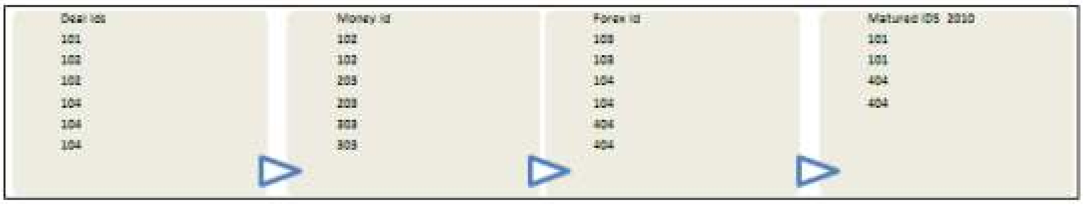
Values are relatively unique in the column.

There is a wide range of values (good for regular indexes).
There is a small range of values (good for bitmap indexes).
The column contains many nulls, but queries often select all rows having a value. In this case, use the

following phrase: [Where col_y > -9.99 * power(10,125)]

2.2 B-Tree Indexes

The oldest and most popular type of Oracle indexing is a standard b-tree index, which excels at servicing simple queries. The b-tree index was introduced in the earliest releases of Oracle and remains widely used in oracle server B-tree indexes are used to avoid large sorting operations. For example, a SQL query requiring 10,000 rows to be presented in sorted order will often use a b-tree index to avoid the very large sort required to deliver the data to the end user.



Deal Ids	Money Id	Forex Id	Matured IDS 2010
101	102	103	101
102	102	103	101
102	203	104	404
104	203	104	404
104	303	404	
104	303	404	

While b-tree indexes are great for simple queries, they are not very good for the following situations:

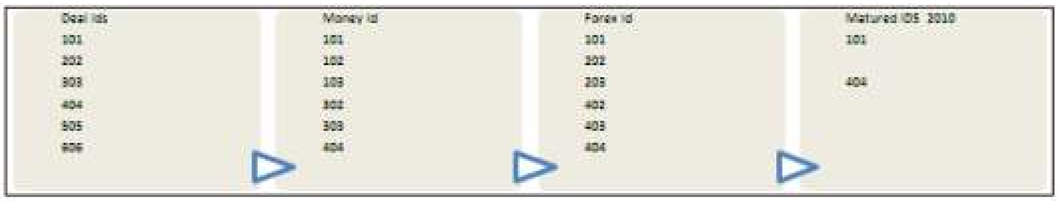
Low-cardinality columns: columns with less than 200 distinct values do not have the selectivity required in order to benefit from standard b-tree index structures.

No support for SQL functions: B-tree indexes are not able to support SQL queries using Oracle's built-in functions. Oracle9i provides a variety of built-in functions that allow SQL statements to query on a piece of an indexed column or on any one of a number of transformations against the indexed column.

2.3 Bitmap indexes

Oracle bitmap indexes are very different from standard b-tree indexes. In bitmap structures, a two-dimensional array is created with one column for every row in the table being indexed. Each column in table represents a distinct value within the bitmapped index. This two-dimensional array represents each value within the index multiplied by the number of rows in the table

The real benefit of bitmapped indexing occurs when one table includes multiple bitmapped indexes. Each individual column may have low cardinality. The creation of multiple bitmapped indexes provides a very powerful method for rapidly answering difficult SQL queries Oracle uses a specialized optimizer method called a bitmapped index merge to service this query. In a bitmapped index merge, each Row-ID, or RID, list is built independently by using the bitmaps, and a special merge routine is used in order to compare the RID lists and find the intersecting values. Using this methodology, Oracle can provide sub-second response time when working against multiple low-cardinality columns. For example as described in this figure



Deal Ids	Money Id	Forex Id	Matured IDS 2010
101	101	101	101
202	102	202	
303	103	203	404
404	302	402	
505	303	403	
606	404	404	

FUNCTION BASED INDEXES

One of the most important advances in Oracle indexing is the introduction of function-based indexing. Function-based indexes allow creation of indexes on expressions, internal functions, and user-written functions in PL/SQL and Java. Function-based indexes ensure that the Oracle designer is able to use an index for its query. Consequently, Oracle would perform the dreaded full-table scan. Examples of SQL with function-based queries might include the following (Select * from Matured_ids where trim(id) = 101); . Oracle always interrogates the where clause of the SQL statement to see if a matching index exists. By using function-based indexes, the Oracle designer can create a matching index that exactly matches the predicates within the SQL where clause. This ensures that the query is retrieved with a minimal amount of disk I/O and the fastest possible speed, consuming time & scan.

2.4 Using Table Clusters for Performance

A table cluster is a group of one or more tables that are physically stored together because they share common columns and usually appear together in SQL statements. Because the database physically stores related rows together, disk access time improves. To create a cluster, use the CREATE CLUSTER statement.

3. CONCLUSIONS

We can create indexes on columns to speed up queries. Indexes provide faster access to data for operations that return a small portion of a table's rows. When we use INDEX there is a significant reduction in I/O results in even faster processing of data within the Oracle Database

In general, create an index on a column in any of the following situations:

•The column is queried frequently.
•A referential constraint exists on the column.
•A UNIQUE key constraint exists on the column.

Although query optimization helps avoid the use of nonselective indexes within query execution, the SQL engine must continue to maintain all indexes defined against a table, regardless of whether queries make use of them. Index maintenance can present a significant CPU and I/O resource demand in any write-intensive application. In other words, do not build indexes unless necessary.

The more indexes, the more overhead is incurred as the table is altered. When rows are inserted or deleted, all indexes on the table must be updated. When a column is updated, all indexes on the column must be updated.

we must weigh the performance benefit of indexes for queries against the performance overhead of updates. For example, if a table is primarily read-only, we might use more indexes but, if a table is heavily updated, we must use fewer indexes.

4. REFERENCES

1.Dr.Himani sharma. (2011). E-banking. BANKERS' PERSPECTIVES ON E-BANKING,15(22490906), 2,3,4,5,6,7,8. Retrieved from http://www.publishingindia.com/uploads/samplearticles/njrim-sample-article.pdf
2.Abraham Silberschatz., Henry Korth F., & Sudarshan, S. (2006). Indexing and Hashing. In Database System Concepts (5th ed., pp. 481-522). New York, U.S: Mc Graw Hill.
3.Retrieved from http://www.dba-oracle.com/oracle_tips_bitmapped_indexes.html
4.Bitmap Index vs. B-tree Index: Which and When? (n.d.). Retrieved from http://www.oracle.com/technetwork/articles/sharma-indexes-093638.html
5.Oracle B-Tree Index: From the concept to Internals - Oracle - Oracle - Toad World. (n.d.). Retrieved from http://www.toadworld.com/platforms/oracle/w/wiki/11001.oracle-b-tree-index-from-the-concept-to-internals.aspx
6.Oracle PL/SQL Syntax & Examples. (n.d.). Retrieved from http://psoug.org/definition/CLUSTER.html
7.ORACLE-BASE - Oracle Function Based Indexes. (n.d.). Retrieved from http://oracle-base.com/articles/8i/function-based-indexes.php