IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-9 E-ISSN: 2347-2693

Tuning a Multi-Million Row Table

Anmol Sharma

Available online at: www.ijcseonline.org

Received: 27/Aug/2016 Revised: 09/Sept/2016 Accepted: 25/Sept/2016 Published: 30/Sep/2016

***Abstract*—** Every organization faces the issue of slow performance with their database on regular basis. In this scenario, a DBA is required to tune the database or find the root cause of the sluggishness. Tuning a database after it has been setup is a highly tedious task and requires a lot of expertise, in-depth knowledge of the architecture and underlying functionality.
So, it's better to deal with the problem from the starting, i.e. structuring the Database in such a manner, that it yields good performance. This will make the database more stable and reduce the effort required to manage the database.
So, presented below is a method to achieve better performance regardless of the quality of the hardware on which the database is hosted. Below method of database structuration made my life easier when I was asked to design the structure of an organization's database from scratch and this entire paper is based on that method.

Note— This method works best if you have a main mega table which needs to be queried again and again, but due to its flexibility this method works well in almost every environment.

Keywords- Database Optimization, Database Structuration, Performance Tuning, Table Structuration.

I. INTRODUCTION

THIS document describes a procedure that can be used to tune a table or an entire database. Emphasis is laid on the structure, the way the database stores & uses data & files.

So, it's better to lead with an example which will help in creating a better understanding of the concept.

Why is Performance tuning required?

- The first question that needs to be answered is, "Is it really necessary to tune a table"
- Well, a table which is queried frequently needs to be tuned in such a manner that the retrieval rate is as minimal as possible. Favorably in micro seconds.
- So, a table which provides a slow retrieval rate i.e. in multiple seconds or even worse, in minutes, should always be considered for tuning.

Sample Data:

Service Area Code	Phone Numbers	Preferences	Opstype	Phone Type
13	98272xxxxx	0	D	2
12	98279xxxxx	0	D	2
11	98272xxxxx	0	A	1
13	98275xxxxx	0	A	4

Table 1. Sample Data

These are just a few lines of sample data, an actual table that needs to be tuned will contain millions of rows of similar data.

***Note*—** The Sample data used in the test case had approx. 30 million rows.

II. PREPARING THE STRUCTURE

A. *Database*

The first thing that is required is obviously a database.

B. *Selecting Target Column*

Secondly, a column is required to be selected based on which the database will be searched frequently, usually that particular column will contain unique values and can be used as a primary key.

This particular column will be used to partition the table and for creating an index.

Based on the sample data, the 'Phone_numbers' column is used for partitioning and indexing purposes.

C. *Structure*

The structure is highly critical as it will decide how you utilize the system resources for storage[1].
Here we are going to store each partition in a different tablespace.

Note— create partitions in such a manner that all the partition holds equivalent amount of data. Create as many partitions as required.

Sample Data— In the sample data, 31 partitions were created in a manner that all partitions hold a minimum of 50,000 and a maximum 150,000 rows.

D. Creating a Tablespace[2]

- create tablespace ts1 datafile '/oracle/new/ts1.dbf' size 50m autoextend on next 50m maxsize 200m;

- create tablespace ts2 datafile '/oracle/new/ts2.dbf' size 50m autoextend on next 50m maxsize 200m;

E. Creating a Partitioned Table

- create table table1 (service_area_code number, phone_number number, preferences varchar(20), opstype varchar(10), phone_type number) partition by range (phone_number) (partition p1 values less than (7000000000)tablespace ts1, partition p2 values less than (7100000000) tablespace ts2,

  _

  partition p31 values less than (10000000000) tablespace ts31);

Partition type **"RANGE"** has been utilized here as it was the most suitable for this type of data. Other partitioning methods like 'LIST', 'HASH' and 'COMPOSITE' can also be used but their suitability will depend on the data which will be stored in those partitions. In this case the results achieved with composite (range with hash sub-partition) were not favorable. Composite partitioning dampened the performance by huge margin and it made the execution plan even more difficult to analyze.

Note— Adding separate tablespaces and datafiles will surely improve performance but to get the most out of your Database structure, the datafiles must be distributed among various storage media which will further enhance the I/O throughput thus improving the overall performance of the database.

F. Adding Datafile

There are two paths you can take while adding datafiles if required:

Adding datafiles to all the tablespaces together:

- Alter tablespace ts1 add datafile '/oracle/new/ts1a.dbf' size 50m autoextend on next 50m maxsize 200m;

- Alter tablespace ts2 add datafile '/oracle/new/ts2a.dbf' size 50m autoextend on next 50m maxsize 200m;

_ _

- Alter tablespace ts31 add datafile '/oracle/ts31a.dbf' size 50m autoextend on next 50m maxsize 200m;

Note— Via this method you can simply add datafiles to tablespaces as per the requirement[3]. This helps in mitigating the error you might receive regarding insufficient space during data loading.

Adding datafile to a specific tablespace when required:

- Alter tablespace ts1 add datafile '/oracle/new/ts1a.dbf' size 50m autoextend on next 50m maxsize 200m;

Note— Check the database alert-log file for the name of the tablespace that requires the datafile.

G. Using Resumable Space Allocation

By default, Oracle database will always be in non-resumable mode which may cause an insufficient space error to crash the entire data loading process.
In resumable space allocation mode, rather than crashing, the process will stay halted. This will provide a chance to the database administrator to solve the problem and let the process resume as normal[4].

Enabling Resumable Space Allocation:

- Alter system enable resumable

- Alter system set resumable_timeout=3600 scope=both;

Now, in this case, the session will wait for an hour before throwing an error which provides the user ample time to solve the problem. Timing can be altered by changing the resumable_timeout parameter[5].

Here you can either add datafiles to all the tablespaces or you can check the alert log file and add the datafiles to a particular tablespace as required.

Note— Recommended option is to add datafile to a tablespace only when it's required. It will consume less space and your database will only have datafiles that are necessary which will reduce the complexity.

III. LODING DATA

In my case, while loading data, usually it was either in form of CSV files or another table from which data needs to be transferred to the partitioned table.

Note— This is not important to tuning, the data just needs to be inserted, doesn't matter which method and source is used.

A. Loading via CSV files

Control file:

OPTIONS(DIRECT=TRUE, ROWS=10000000, SKIP=1)

unrecoverable

load data

infile '/oracle/nccp/2082013_0.csv'

infile '/oracle/nccp/2082013_1.csv'

insert into table table1

fields terminated by ',' optionally enclosed by '"'

(service_area_code,phone_number,preferences,opstype,phone _type)

Using Sql loader:

sqlldr scott/tiger control=/oracle/a.ctl log=/oracle/a.log

sqlldr '"/as sysdba"' control=/oracle/a.ctl log=/oracle/a.log

Note— Using DIRECT clause enables direct loading which is quicker than the default method and using SKIP skips the first line as you usually have column names as the first line of a CSV file[6].

B. Loading from another table

Inset into table1 select * from table2;

Note— Specific columns can be loaded instead by inserting their names in the insert and select clause.

IV. OPTIMIZING

A. Indexing

As per the sample data, the index needs to be created on the phone_number column. A b-tree (default) index is preferred as the column contains unique values.

Two Variations in this type of index can be used:

- Un-partitioned Index
- Partitioned Index

Un-partitioned Index:

Create index index1 on tables1(phone_number);

or

A primary key can also be used as it will create an index automatically.

Partitioned index(Recommended):

Create index index1 on table1(phone_number) LOCAL;

or

Create index index1 on table1(phone_number) global partition by range(phone_number)
(partition p1 values less than (7000000000)tablespace i1,
partition p2 values less than (7100000000)tablespace i2,
_ _
partition p31 values less than (10000000000)tablespace i31));

Note— You can remove the tablespace clause and store the entire index in one tablespace or you can diversify the storage like the above example and then store it on different storage media if available which will further improve the I/O throughput and final output performance.

B. Optimizer & Statistics

The first that needs to be checked is the optimizer parameter[7].

- Show Parameter optimizer

This command can be used to check the value that the parameter currently holds.

It can contain 4 different values

- All_Rows
- First_Rows_10
- First_Rows_100
- First_Rows_1000

In my case the ALL_ROWS worked better than the others.

- Alter system set Optimizer_mode=First_Rows_10;

This will change the optimizer mode to rule based First_Rows_10.

Calculating Statistics:

This is important for the optimizer to perform its function, so whenever a change occurs in an important table, stats should always be calculated[8].

- Exec dbms_stats.gather_table_stats ('USER_NAME','TABLE_NAME');

- Exec dbms_stats.gather_index_stats ('USER_NAME','INDEX_NAME');

- Exec dbms_stats.gather_Schema_stats ('SCHEMA_NAME');

Note— Calculating stats for the table and index will majorly improve performance.

C. Parallelism:

Using Parallel SQL will further improve the performance of your database sql queries.

Parallelism creates multiple processes that work on the same statement thus using more CPU cycles for the same statement which improve query performance majorly[9].

- Alter table table1 Parallel 4;

This will set the queries performed on table1 to be executed by 4 processes simultaneously.

- Select /*+parallel(10) */ * from table1;

This will improve performance by including 10 parallel processes but only for this particular query.

Note— Parallel hint won't always work in your favour, so judge the performance based on your experience with it, if it causes any slowdown consider excluding it.

Generally parallel processes do improve performance but sometimes it can trick the database into thinking that a full table scan is cheaper than using an index which will obviously cause major performance issues[10].

D. Execution Plan

Execution plan can be generated via Explain Plan which helps in studying the entire plan that the database is going to use while executing a specific command[11].

You can test the optimizations you have made by explaining plans for your query after making some changes. This way the best methods of optimizing can be determined.

- Explain plan for select * from table1;
- Select * from table (dbms_xplan.display);

or

- Set Autotrace on or traceonly (will show only the plan not the output).

V. CONCLUSION

In this paper I have put forward a method of designing the database and restructuring a major table in a much optimized manner.

My research & analysis have proven this method as a major asset. Although I would suggest that the example I have used should only be considered as a base.

Increase the size of the data-files slightly if required, otherwise this method will definitely provide major performance enhancements if you are willing to put in the effort to implement it.

Alongside improving performance, this method will also save space and help you organize your database in a much more convenient manner.

I will keep experimenting with my method further and update this paper as I learn more ways to improve it further. My future work in this stream will cover the following two issues:

- Enhancing the structure further, if possible
- Providing recommendations on tuning the database internally after this entire setup is complete to further enhance performance.

ACKNOWLEDGMENT

I would like to thank all those people who taught and inspired me throughout my life based on which today I am capable enough to present this paper.

I would also like to thank everyone who gave me the chance to work and trusted me enough to let me experiment. Without this, I would have never been able to learn what I know and succeed in the path that I have chosen for my life and career.

REFERENCES

[1] Partitioning in Oracle Database, http://www.oracle.com/technetwork/database/enterprise-edition/partitioning-11g-whitepaper-159443.pdf, Jul 4, 2014
[2] Create Tablespace, https://docs.oracle.com/database/121/SQLRF/statements_7003.htm#SQLRF01403 Aug 23, 2014
[3] Managing Tablespace and Datafiles, http://www.oracle-dba-online.com/tablespaces_and_datafiles.htm Aug 30, 2014
[4] Managing Resumable Space Allocation, http://docs.oracle.com/cd/B28359_01/server.111/b28310/schema002.htm#ADMIN11581 Sep 15, 2014
[5] Resumable Space Allocation, http://gavinsoorma.com/2009/06/resumable-space-allocation/ Oct 1,2014
[6] SQL Loader Concepts, https://docs.oracle.com/cd/B19306_01/server.102/b14215/ldr_concepts.htm Oct 27, 2014
[7] The Query Optimizer, http://docs.oracle.com/cd/B28359_01/server.111/b28274/optimops.htm Dec 20, 2014
[8] Performance Tuning Overview, http://docs.oracle.com/cd/E11882_01/server.112/e41573/perf_overview.htm#PFGRF025 Jan 24, 2015
[9] Database Performance Tuning, Guidehttps://docs.oracle.com/database/121/TGDBA/toc.htm Feb 25, 2015
[10] How to tune your Oracle database's performance, http://www.theregister.co.uk/2014/05/06/oracle_database_performance_workshop/ Mar 2, 2015
[11] Explain Plan Usage, https://oracle-base.com/articles/8i/explain-plan-usage Jul 15, 2015

Authors Profile

Anmol Sharma, a B.Tech graduate from Beant College of Engineering & Technology, Gurdaspur (143521) has worked on multiple institutional and organizational projects which involve research and innovation. He is currently working at Heuristics Informatics Pvt. Ltd. He specializes in database and related technologies and holds multiple technical certification like Oracle 10g & 12c OCP, Oracle SQL Expert and Oracle Exadata Implementation Specialist.

