Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.342 – 347

REVIEW ARTICLE

A Review on Partitioning Techniques in Database

Abhay Kumar[1], Jitendra Singh Yadav[2]

[1]Department of Computer Science and Engineering, JECRC University, Jaipur, India

[2]Department of Computer Science and Engineering, JECRC University, Jaipur, India

[1] abhaykumar.it@jecrc.ac.in; [2] jitendra.yadav@jecrcu.edu.in

Abstract — Data is most important in today's globe as it helps organizations as well as persons to take out information and use it to make various decisions. Data generally stocked in database so that retrieving and maintaining it becomes easy and manageable. All the operations of data handling and maintenance are done using Database Management System. Data management is much monotonous task in growing data environment. Partitioning is possible solution which is partly accepted. Partitioning provides user-friendliness, maintenance and impulsive query performance to the database users. In this paper, brief review of methods of partitioning and helps to reduce the wait in response time. Paper shows the positive result with partitioning methods.

Keywords— Database partitioning; Dbms Redefinition; Range Partitioning; Hash Partitioning; List Partitioning

I. INTRODUCTION

Partitioning allows the table to be declustered into the smaller parts called as partition and each partitioned table has its own name and characteristics. Composite key is the secret to the partition which consists of one or more column that decides the partition.

Users need to follow suggestions while partitioning the tables because it contains large data size, for example a table which stores the historical data and which requires dissimilar type of storage devices to store its data need to partition. Partition enhances the performance of accessing the data from the table.

While partitioning an index it includes few ideas which should be integrated initially, avoid rebuilding the entire index, perform maintenance without overthrow entire index and reduce the impact of index skew are the other two suggestions. For partitioning index-structured tables, composite key column should be primary key. Secondary indexes can be partitioned and OVERFLOW data segments are equally partitioned in Index-structured tables. Table partitioning provides scalability, availability and manageability without having database control.

Partition for performance focuses on partition trim and partition wise joins. Partition trim provides the partitioned data without querying the entire database, for example if table containing 10 years of historical data then query requesting a single day will only access a single partition instead of 50 partitions. Partition wise join decomposes big join into smaller join and ensures the performance.

A. Partitioning policy

We present three basic data-partitioning strategies. We can say there are n disks, $A_0,\ldots, A_{n-1}$,across which the data are to be partitioned.

Round-robin: the relation is scanned in any order and the i^{th} tuple is sent to the disk numbered Ai mod n. The round-robin scheme ensures an even distribution of tuples across disks; that is, each disk has approximately the same number of tuples as do the others.

Hash Partitioning: In this one or more attributes from the given relation's schema are designated as the partitioning attributes. A hash function is chosen whose range is {0, 1, ..., n-1}. Each tuple of the original relation is hashed on the partitioning attributes. If hash function returns i, then the tuple is placed on the disk Ai.

Range Partitioning: An entity-set that is partitioned by range is partitioned in such a way that each partition contains rows for which the partitioning term value lies within a given range. Ranges should be adjacent but not overlapping, and are label using the VALUES LESS THAN operator.

Further the partitioning can be studied as, using Range partitioning method it separates the data according to range of values of the partitioning key, for example partition of August 2010, this will partitioning key values from 1st August 2010 to 31st August 2010.Each partition has 'VALUES LESS THAN' clause and MAX VALUE is defined to compare with the highest value.
Hash partitioning record the data according to hash algorithm. It evenly distributes the data across devices.
List Partitioning provides the partition of a set of discrete values. If a table is having data of business centers across the globe then list partitioning separates according to the country.

Using the partition technique a table can be partitioned as:
- Single-Level partitioning
- Composite partitioning

Composite partitioning provides the mixture of the basic distribution. It molder the partition into sub partitions. Composite partition includes
- Composite Range- Range partitions
- Composite Range –Hash partitions
- Composite Range-List partitions
- Composite List-Range partitions
- Composite List-Hash partitions
- Composite List-List partitions

In count to basic partitioning strategies, partitioning expansion are provided
- Manageability Extensions
- Composite key Extensions

Manageability extension provides interval partitioning and partition advisor whereas Interval partitioning is an extension of range partition. If a table follows monthly partition then after end of month, a new partition is generated.

Interval partitioning process can be seen with single level partitions technique
- Interval- Range
- Interval-Hash
- Interval –List

Partition advisor is part of SQL mentor which recommends a partitioning strategy by SQL store and SQL Tuning set. Composite key extension extends in defining the partition key like Reference partitioning and virtual column based partitioning is part of key extensions. Implicit column based provides partitioning even if composite key is absent physically in the table. The composite key can be defined by expression, using one or more existing column. Reference partitioning allows the partitioning of two relations related to one another by referential integrity.

This paper focuses on partitioning theory. The rest of the paper is structured as follows. In the section II, three research paper are connected to database partitioning is discussed. Section III consists of testing conducted during the study of the topic. Section IV shows the conclusion based on the testing and ends the paper with future scope.

II. LITERATURE SURVEY

'Divide and conquer' rule is used to reduce the complexity which results in increasing the performance of the database. Near uniform range partition (NURP) approach [4] is based on range partitioning.

Traditionally uniform range partitioning algorithm were used for partitioning the tables but to speed up the partitioning of table, current partitioning techniques are studied and three range partitioning strategies are added. To manage the data in partitions, more strategies are involved so that these strategies will automate the partition. NURP-I is used to maintain the data equally in each partition. Uniform range partition distributes data unequally and Adjustment stage adjusts the data by splitting and merging. NURP-II uses a single query which helps to increase the execution speed. NRUP-III starts automating the partition when data is increasing gradually. NRUP is efficiently used for partitioning of large databases which leads into a good advantage of this method because this automates the partitioning of large database. Near-uniform range Partitioning Algorithm (NPA) [7] is used as increased partitioning approach for huge amount of data in real-time data warehouse. The primary work includes new test of data warehouse and aimed for range partitioning using NPA.
This paper also focused on the efficiency of the data warehouse. NPA also focuses on multilevel partition of table. It checks small tables to find the complexity such that it helps in increasing the performance. As amount of data increases, real time partition is done by new partitioning plan.
Shin obi [3] horizontal partitioning helps to improve the query performance. It most probably focuses on clustering of physical data and improves the performance by frequently accessed indexing data. This paper deals with the algorithms which optimally partition the table and manage the partition on different disk. This paper uses index partitioning approach for dynamic query workload from traffic monitoring application.
Partitioning to a data warehouse [4] was discussed previously during Exchange Partition method with ETL (Extraction Transform Load).

III. APPROACHES

Database Partitioning can be done by using different ways.
3.1. export/import
3.2. Dbms Redefinition method
3.3. Exchange method
3.4. Partition Advisor

The export command example:
"expdp scott/tiger tables=EMP,DEPT directory=TEST dumpfile=EMP_DEPART.dmp"

The import command example:
"Impdp scott/tiger tables=EMP,DEPT directory=TEST dumpfile=EMP_DEPART.dmp"

The Other approach based on Dbms Redefinition method.
This method has five basic steps:
3.2.1 Create a sample schema
3.2.2 Create a partitioned interim table
3.2.3 Start the redefinition process
3.2.4 Create Constraints and indexes
3.2.5 Complete the redefinition process.

This process creates model schema which goes under partitioning process. The next is to create partitioned intervening table with the number of partitions. With this intervening table, we can start redefinition process. First will check whether redefinition is possible or not by using Dbms_Redefinition.Can_redef_table (USER,'Table_name').
If the redefinition is possible then perform the redefinition process. If redefinition process fails then use Dbms_Redefinition.Abort_redef_table to stop the process. After the completion of this step, copy the table dependencies and tally the table. Last step is to complete the redefinition process. Use user_tables to ensure the number of partitions

inside the table.
The entire Dbms_Redefinition process is as follows:

i. create a sample table
ii. create a portioned interim_table
iii. Exec Dbms_redefination.Can_redef_table(User,'table');

```
Begin
 Dbms_redefination.start_redef_table
(
username=>user,
original_table=>'table',
int_table=> 'interim_table'
);
END;
/
Declare
num_errors PLS Integer;
Begin
DBMS_Redefination.cons_origin_param,true,true,true,true,num_errors);
END;
/
Begin
Dbms_redefination.start_redef_table
(
username=>user,
original_table=>'table',
int_table=> 'interim_table'
);
END;
/
exec dbms_stats.gather_table_stats(User,'interim_table', cascade => True);

Begin

Dbms_redefination.start_redef_table

(
username=>user,
original_table=>'table',
int_table=> 'interim_table'
);
END;
/
drop table interim_table;
```

DBMS_REDEFINITION process has two advantages. We can make online redefinition. Partitioning can be completed by keeping database up and running. This one is the best advantages over all methods.
Third approach is Exchange Partition. Database should not be online for this approach as DDL operations are performed in this approach.
This approach has IV steps:
3.3.1 Create a sample schema
3.3.2 Create a partitioned interim table
3.3.3 Exchange Partition
3.3.4 Split Partition
First we create a test schema. Then we create suitable partition table as a destination table. Other two processes include Exchange partition and split partition. Exchange partition method exchanges the partition between source and destination

table whereas Split partition divides a partition into small partitions.

Here is the exchange partition approach-

Create sample schema

Create partitioned interim table

```
ALTER TABLE DATATIMEZONEDIMERED_PARTITION

EXCHANGE PARTITION

DATATIMEZONEDIMERED_PART_813

WITH TABLE DATATIMEZONEDIMERED

WITHOUT VALIDATION

UPDATE GLOBAL INDEXES;


ALTER TABLE DATATIMEZONEDIMERED_PARTITION

SPLIT PARTITION DATATIMEZONEDIMERED_PART_MAR

AT (TO_DATE ('20/02/2010','DD/MM/YYYY'))

INTO (PARTITION DATATIMEZONEDIMERED_PART_RS)

UPDATE GLOBAL INDEXES;

EXEC DBMS_STATS.GATHER_TABLE_STATS (USER,'DATATIMEZONEDIMERED_PARTITION',
CASCADE=>TRUE);
```

Partition advisor uses the SQL access advisor which provides important details about additional indexes and materialized view which leads to recuperate the system performance. Partition advisor gives the partitioning plan to enhance the throughput.Three ways are used to implement partition advisor in which one of them is Enterprise Manager which provides simple interface for the SQL Access Advisor (Advisor Central > SQL Advisor >SQL Access Advisor). DBMS_ADVISOR and DBMS_SQLTUNE package.

Procedure used by DBMS_ADVISIOR-

Create task

Reset task

Delete previous STS workload task link

Delete previous STS

Create STS

Select all statement into cursor cache

Load statement into STS

Link STS workload parameters

Set task parameters

EXECUTE task

Generate a script

SELECT DBMS_ADVISOR.GET_TASK_SCRIPT ('ABHAY_KUMAR') As a script

From dual

IV. CONCLUSIONS

Here I have discussed 4 methods in last section. These methods are used to partition a database by using different ways. All the above methods are good in the respective environment. Dbms_Redefinition method is best suited when partitioning to be done online whereas Partition exchange comes in the picture when we want sub-partition offline while SQL Access Advisor can be used when the GUI based approach is needed. Import-Export approach used when we deal with the large data.

REFERENCES

[1] Scaling to Infinity: Partitioning in Oracle Data Warehouses, SageLogix, Inc., White Paper

[2] A NET 2000 Ltd., **"Data** Scrambling Issues‖, A White Paper(2010), Website, October 10/2012, http://www.datamasker.com/data scrambling issues.pdf

[3] Eugene Wu and Samuel Madden, Partitioning Techniques for Fine-grained Indexing, 978-1-4244-8960-2/11, 2011 IEEE

[4] Wen Qi, Jie Song and Yu-bin Bao, Near-uniform Range Partition Approach for Increased Partitioning in Large Database, IEEE, 978-1-4244-5265-1/10, 2010

[5] http://docs.oracle.com/cd/E11882_01/server.112/e25523/partition.htm.

[6] http://books.google.co.in/books?hl=en&lr=&id=AfL0tYzOrEC&oi=fnd&pg=PP2&dq=references+for+data+partitioning+technique&ots=Uv_rM9mE6&sig=2vS7hkhkpFMfrslp7R7dvAnYeWs#v=onepage&q=references%20for%20data%20partitioning%20technique&f=false

[7] Jie Song and Yubin Bao, NPA: Increased Partitioning Approach for Massive Data in Real-time Data Warehouse, IEEE, 978-1-4244-7585-8/10, 2010

[8] A NET 2000 Ltd., **"Data** Sanitization techniques‖, A White Paper(2010), Website, September 21,2012, http://www.datamasker.com/datasanitization_whitepaper.pdf.

[9] A. S. Talwadker, "Survey of performance issues in parallel database systems," J. Comput. Small Coll., vol. 18, pp. 5–9, June 2003. [Online]. Available: http://dl.acm.org/citation.cfm? id=770818.770823

[10] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, "Bigtable: A distributed storage system for structured data," ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, p. 4, 2008.

