International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-5, October 2013

Database Partitioning: A Review Paper

Mayur Sawant, Kishor Kinage, Pooja Pilankar, Nikhil Chaudhari

Abstract*— Data management is much tedious task in growing data environment. Partitioning is the best possible solution which is partially accepted. Partitioning provides availability, maintenance and improvised query performance to the database users. This paper focuses the three key methods of partitioning and helps to reduce the delay in response time. Paper also investigates the composite partition strategies which includes the date, range and hash partitions. The paper shows the encouraging result with partitioning methods and basic composite partition strategies.

***Index Terms*— Database partitioning, Dbms_Redefinition, Range Partitioning, Hash Partitioning, List Partitioning**

I. INTRODUCTION

Partitioning allows the table, index and index-organized table to be decomposed into the smaller parts called as partitions. Each Partition has its own name and optionally has its characteristics.

Partition key is the secret to the partition. It comprises of one or more column that decides the partition. Any table can be partitioned except those CLOB (Character Large Object) and BLOB (Binary Large Object) data types.

Users need to follow suggestions while partitioning the tables. Tables greater than 2GB, tables which stores the historical data and table which requires different types of storage devices to store its data need to partition. Partition offers enhanced performance.

When to partition an index includes few suggestions which includes firstly, avoid rebuilding the entire index. Perform maintenance without invalidating entire index and reduce the impact of index skew are the other two suggestions. For partitioning index-organised tables, partition key column should be primary key. Secondary indexes can be partitioned and OVERFLOW data segments are equi-partitioned in index-organised tables.

System Partitioning provides scalability, availability and manageability without having database control. Benefits of partitioning include performance, manageability and availability.

Partition for performance focuses on partition pruning and partition wise joins. Partition pruning provides the partitioned data without querying the entire database. If table containing 3 years of historical data then query requesting a single week would only access a single partition instead of 156 partitions. Partition wise join decomposes big join into smaller join and insures the performance.

Partition for availability simply follows the 'divide and conquer' approach to manage the data. It helps in maintenance operation where we need mission critical tables.

Storing different partitions at different location enhances availability issue.

A. Partition Strategies

Oracle partitioning offers three fundamental basic partition strategies

- Range
- Hash
- List

Using the partition strategies a table can either be partitioned as a single list or as a composite partitioned table.

- Single-Level partitioning
- Composite partitioning

Range partitioning separates the data according to range of values of the partitioning key. For partition of July 2013, the partitioning key values from 1st July 2013 to 31st July 2013. Each partition has 'VALUES LESS THAN' clause and MAXVALUE can be defined to compare with the highest value.

Hash partitioning maps 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 centres across the globe then list partitioning separates according to the country.

Composite partitioning provides the combination of the basic distribution. It decomposes 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 addition to basic partitioning strategies, partitioning extensions are provided

- Manageability Extensions
- Partitioning key Extensions

Manageability extension provides interval partitioning and partition advisor. Interval partitioning is an extension of range partition. A new partition automatically creates when the data exceed than the last partition. If a table follows monthly partition then after 30 days, a new partition is generated.

Interval partitioning can be seen with single level partitions

- Interval- Range
- Interval-Hash
- Interval -List

Manuscript published on 30 October 2013.
*Correspondence Author(s)
Mayur Mahadev Sawant, Department of Information Technology, MIT College of Engineering , Pune, India.
Dr. Kishor Kinage, Professor, Department of Information Technology, MIT College of Engineering , Pune, India
Pooja Shashikant Pilankar, Department of Computer, Ramrao Adik Institute of Technology, Mumbai, India.
Nikhil Anil Chaudhari,, Department of Information Technology, MIT College of Engineering , Pune, India.

Partition advisor is part of SQL advisor. Partition advisor can recommend a partitioning strategy by studying workload, SQL cache and SQL Tuning set.

Partitioning key extension extends in defining the partition key. Reference partitioning and virtual column based partitioning fall under key extensions. Virtual column based partitioning provides partitioning even if partitioning key is not present physically in the table. The partition key can be defined by expression, using one or more existing column. Metadata is used to store the expression. Reference partitioning allows the partitioning of two tables related to one another by referential integrity.

This paper focuses on partitioning concepts. The rest of the paper is organised as follows. In the section II, three papers related to database partitioning is discussed. Section III consists of experiments conducted during the study of the topic. Section IV shows the conclusion based on the experiments and concludes the paper with future scope.

II. LITERATURE SURVEY

Near-uniform range partition (NURP) approach [1] is based on range partitioning. 'Divide and conquer' rule is used to minimize the complexity and increase the performance of the database. Traditionally used uniform range partitioning algorithm is used for partitioning. To speed up the partitioning, current partitioning technique is studied and three efficient range partitioning strategies are added.

To balance the data in partitions, three more strategies are used. The aim behind these strategies is to automate the partition. NURP-I is used to distribute the data equally in each partition. Uniform range does so but not equally. Adjustment stage checks the each partition and adjusts the data by splitting and merging. NURP-II uses a single query rather than looping which helps to increase the execution speed. NRUP-III automates the partition when data is increasing. NRUP is efficiently used for partitioning on large databases. Advantage of this method is to automate the partitioning of large database.

Near-uniform range Partitioning Algorithm (NPA) [2] is used as increased partitioning approach for massive data in real-time data warehouse. The primary work includes new challenges of data warehouse and aiming for range partitioning using NPA. This paper also focuses on the increased partitioning and efficiency of the data warehouse.

NPA also focuses on multilevel partition. It checks small tables which help to find out the complexity and thus increasing the performance. As the data grows on increasing, real time partitioning is done by new partitioning plan. The same concept can be used for star schema of data warehouse.

Shinobi [3] helps to improve the query performance by using horizontal partitioning. It focuses on cluster the physical data and improves the performance by frequently accessed indexing data. This paper presents design algorithms which optimally partition the table and manage the partition. This paper uses index partition approach for real-world query workload from traffic monitoring application.

Partitioning to a data warehouse [4] is discussed with Exchange Partition method during ETL (Extraction Transform Load). The exchange partition and basic algorithm helps ETL process to work out in simpler ways.

III. APPROACHES

Database Partitioning can be done by using 5 different methods.

3.1. expdp/impdp
3.2. Dbms_Redefinition method
3.3. EXCHANGE_PARTITION method
3.4. Partition Advisor

Import-Export commands are used to partition a table. Two steps included in this approach. First step is to export the data from non-partitioned table. Second is to import it in the partitioned table. Fig.1 and fig.2 listed below show the query for the import-export command.

```
expdp username/password TABLES=non_partitioned_table
DIRECTORY=directory1 DUMPFILE=dump_file1.dmp
```

Fig.1. Export command

```
impdp username/password
REMAP_TABLE=non_partitioned_table:partitioned_table
DIRECTORY=directory1 DUMPFILE=dump_file1.dmp
TABLE_EXISTS_ACTION=APPEND
```

Fig.2. Import command

The second 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 sample schema which goes under partitioning process. The next is to create partitioned interim table with the number of partitions. With this interim table, we can start online redefinition process. First we have to check whether redefinition is possible or not by using Dbms_Redefinition.Can_redef_table (USER, 'Table_name').

If the redefinition is possible then start the redefinition process. If redefinition process fails due to some reason then use Dbms_Redefinition.Abort_redef_table to abort the process. After successful completion of this step, copy the table dependencies and synchronize the table. Last step is to complete the redefinition process. Use user_tables to check the number of partitions inside the table. Fig.3 shows the entire Dbms_Redefinition process.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

```
Create a sample table

create a partioned interim_table

EXEC Dbms_Redefinition.Can_Redef_Table(USER, 'table');

BEGIN
 DBMS_REDEFINITION.start_redef_table(
   uname     => USER,
   orig_table => 'table',
   int_table  => 'interim_table');
END;
/

DECLARE
num_errors PLS_INTEGER;
BEGIN
DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(USER,
'table','interim_table',
  DBMS_REDEFINITION.CONS_ORIG_PARAMS, TRUE, TRUE,
TRUE, TRUE, num_errors);
END;
/

BEGIN
 dbms_redefinition.sync_interim_table(
   uname     => USER,
   orig_table => 'table',
   int_table  => 'interim_table');
END;
/

EXEC DBMS_STATS.gather_table_stats(USER, 'interim_table',
cascade => TRUE);

BEGIN
 dbms_redefinition.finish_redef_table(
   uname     => USER,
   orig_table => 'table',
   int_table  => 'interim_table');
END;
/
drop table interim_table ;
```

Fig.3 Dbms_Redefinition Process

DBMS_REDEFINITION process provides us two advantages. We can make online redefinition. Partitioning can be done by keeping database up and running. This is one of the best advantages over all methods.

Third approach is Exchange Partition. Database should be offline for this approach as DDL operations are processed in this approach. This approach has 4 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 sample schema. Then we create appropriate partition table as a destination table. Next two processes include Exchange partition and split partition. Exchange partition method exchanges the partition between source and destination table. Split partition divides a partition into small partitions. Fig.4 shows the entire Exchange Partition approach.

```
create sample schema

create partitioned interim table

ALTER TABLE DATETIMEZONEDIMRED_PARTITION
  EXCHANGE PARTITION
DATETIMEZONEDIMRED_PART_B13
  WITH TABLE DATETIMEZONEDIMRED
  WITHOUT VALIDATION
  UPDATE GLOBAL INDEXES;

ALTER TABLE DATETIMEZONEDIMRED_PARTITION
  SPLIT PARTITION DATETIMEZONEDIMRED_PART_JAN
AT (TO_DATE('21/01/2013', 'DD/MM/YYYY'))
  INTO (PARTITION DATETIMEZONEDIMRED_PART_PQ,
      PARTITION DATETIMEZONEDIMRED_PART_RS)
 UPDATE GLOBAL INDEXES;

EXEC DBMS_STATS.gather_table_stats(USER,
'DATETIMEZONEDIMRED_PARTITION', cascade =>
TRUE);
```

Fig. 4 Partition Exchange Process

Partition advisor uses the SQL access advisor which is introduced in 10g. It provides important details about additional indexes and materialized view which leads to improve the system performance. Partition advisor gives the partitioning schemes to enhance the throughput.

Three ways are used to implement partition advisor. 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. Fig. 5 shows the procedure using DBMS_ADVISOR.

```
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 to Task

set STS workload parameters

set task parameters

Execute task

gernerate a Script

SELECT DBMS_ADVISOR.get_task_script
('MAYUR_SAWANT') AS script
FROM   dual;
```

Fig. 5 DBMS_ADVISOR procedure

IV. CONCLUSIONS

We have discussed 4 methods in last section. These methods are used to partition a database by using different approaches. All the above methods are best in the respective environment.

Dbms_Redefinition method is best suited when partitioning to be done online. SQL Access Advisor can be used when the GUI based approach is needed. Partition exchange comes in the picture when we want sub-partition offline. Import-Export approach can be used when we are dealing with the large data. Table 1 shows the comparison of all methods.

TABLE 1 Comparison of partitioning approaches

Methods	Online	Offline	GUI	Large Data
Import-Export	-	Yes	-	Yes
Dbms_Redefinition	Yes	-	-	-
Partition Exchange	-	Yes	-	-
Partitioning Advisor	-	-	Yes	-

V. ACKNOWLEDGMENT

We are thankful to Mr Prathamesh Chavan for his genuine guidance for the data management.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

REFERENCES

1. 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
2. Jie Song and Yubin Bao, NPA: Increased Partitioning Approach for
3. Massive Data in Real-time Data Warehouse, IEEE, 978-1-4244-7585-8/10, 2010
4. Eugene Wu and Samuel Madden, Partitioning Techniques for Fine-grained Indexing, 978-1-4244-8960-2/11, 2011 IEEE
5. Scaling to Infinity:Partitioning in Oracle Data Warehouses, SageLogix, Inc., White Paper

AUTHOR PROFILE

Mayur Mahadev Sawant received his B.E. degree in Information Technology from Mumbai University.

Currently he is doing Master Of Engineering from MIT College Of Engineering, Pune University. His research area includes Database Partitioning, Keystroke Biometrics and Mouse Dynamics.

Dr. Kishor Kinage graduated from S. S. G. M. College of Engineering Shegaon in 1989 and completed his post graduation from V. J. T. I., Mumbai in 1998. He completed his PhD from NMIMS University Mumbai, in 2011.

Currently he is working as Professor in Information Technology at MIT College of Engineering Pune. His special fields of interest include Biometrics, face recognition, Geographic Information Systems.

Pooja Shashikant Pilankar received her B.E. degree in Computer Science from Solapur University.

Currently she is doing Master Of Engineering from Ramrao Adik Institute of Technology, Mumbai University. Her research area includes Database and Java.

Nikhil Anil Chaudhari received his B.E. degree in Information Technology from Pune University.

Currently he is doing Master Of Engineering from MIT College Of Engineering, Pune University. His research area includes Database Partitioning, Wireless Sensor Network.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)