IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. III (May-Jun. 2014), PP 43-48
www.iosrjournals.org

Evaluation of Bitmap Index Compression using Data Pump in Oracle Database

Shivam Dwivedi[#1], Dr. Savita Shiwani[#2]
[#1]*(Computer Science& Engineering/Suresh Gyan Vihar University, Jaipur 302025, India)*
[#2]*(Associate Professor, Suresh Gyan Vihar University, Jaipur)*

Abstract: *Bitmap index is most commonly used technique for efficient query processing and mostly in the Data warehouse environment. We review the existing technologies of Compression and introduce the bitmap index compression through data pump. According to conventional wisdom bitmap index is more efficient for minimum unique value. But through data pump it doesn't require either bitmap index is created on high degree of cardinality or low degree of cardinality. In this paper, we propose data pump utility for release the disk space in database after deletion of records. Bitmap index point the old location even after deletion of records from table, This utility doesn't release disk space. We have implemented data pump for compression, to release the space and change the index pointing location. Data pump which is often used for logical backups in oracle database. Finally we review the bitmap index which commonly used for industrial purpose and discuss open issues for future evolution and development.*

Keywords: *Bitmap Index, Compression, Data Pump, Query Performance.*

I. Introduction

For selection of some records in oracle database. The simplest way evaluating a query is to scan all data records to specify condition. Full table scan sometimes affect the query performance [6,7]. A typical query condition to calculate the number of distinct values in particular column which require full table scan. Indexes accelerate the searching procedure such as variation with Bitmap (Chan &Loannidis, 1998), Bitmap Index for Data Warehouse (Kurt Stockinger and Kesheng Wu, 2003) and B-Trees or kd-Trees (Comer, 1979; Gaede & Guenther, 1998). In database as number of column increases, the number of index possible combination also increases. Bitmap index is widely used in the data warehouse environment [5,6,9]. Data pump allows efficient compression [3,4,5] of bitmap index after deletion. In this chapter we discuss bitmap index on large table which contains at least millions of records. When user applies bulk deletion from the database and even after fire commit .the index is still pointing to before location. Data pump is utility which basically import export the database object, in industrial application which supports huge amount of data and allow the DML's very frequently. Large table in database contains the huge amount and occupy the space in Giga bytes. Even the bulk deletion from table

Index pointing location doesn't change due to the by default behavior of oracle and table and index object still occupying the same space as before deletion. For industry it's drawback that object is still occupying space after deletion. Our objective behind this study is to release the disk space after deletion of records from table.

II. Bitmap Index

Bitmap index is most efficient index in oracle database. This index provides benefit in the query performance [6]. The name bitmap index was used by O'Neil and colleagues (O'Neil, 1987; O'Neil & Quass, 1997). Bitmap index introduced as an improvement of B-Tree index. Important feature of bitmap index is it contains key values and bitmaps. Key values and bitmaps are arranged as array in binary files. Bitmap index basically assign key to each distinct values. Degree of cardinality also matters for performance of bitmap index. Normally bitmap index contains small size for low degree of cardinality and maximum size for high degree of cardinality [8]. High degree of cardinality refers as bitmap index on the scientific values column, which contain maximum number of unique values. Low degree cardinality refers as minimum number of unique values like index on salary column. But our research doesn't matter with the cardinality issues.

2.1 Bitmap Index Structure

Bitmap index [1,2] has most efficient structure for the single dimensional queries as well as multi dimensional queries[10]. Logical operations are evaluated with queries, where Individual bitmap value assigns to each data value. Computer hardware also supports bitmap logical operation [5]. Normally these operations performed on the bit values which refer data. Here bitmap index with three different values represents the three unique column value

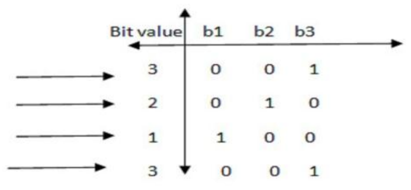


Fig. 1.bitmap index representation for unique column values

Data Pump

Data pump utility used for backup of logical entity in the database. Only process in oracle database which performs backup of logical entities. This tool transfer database objects at very high speed from one location to another location. One schema object can transfer to another schema. Data pump also includes the export and import facility, where user can export their data and preserves it as a backup for future aspects. For taking backup through data pump we need to create a directory, where oracle logically writes on that particular location and physical location also required for storage of log file and dump file. Through data pump following entities backup possible-

2.2 Database

For this utility data pump export the full database, it will not include the schema in it only the Meta data as a part of object which are usually present in data dictionaries.

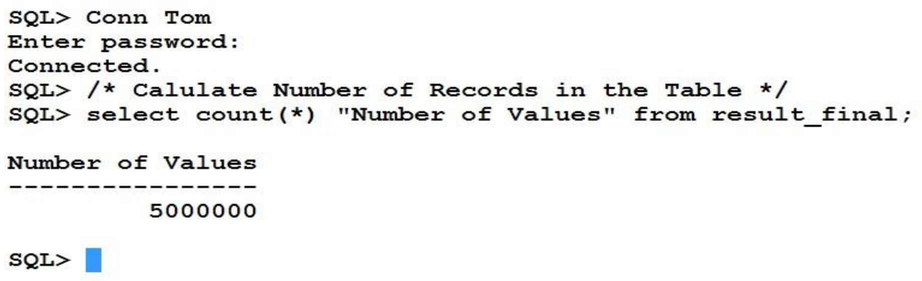
2.3 Schemas

Export the full schema, if requirement of schemas object then not necessary to export the full database then user can export only their own schema. To export other's schemas object user required privileges.

III. Remap Schema

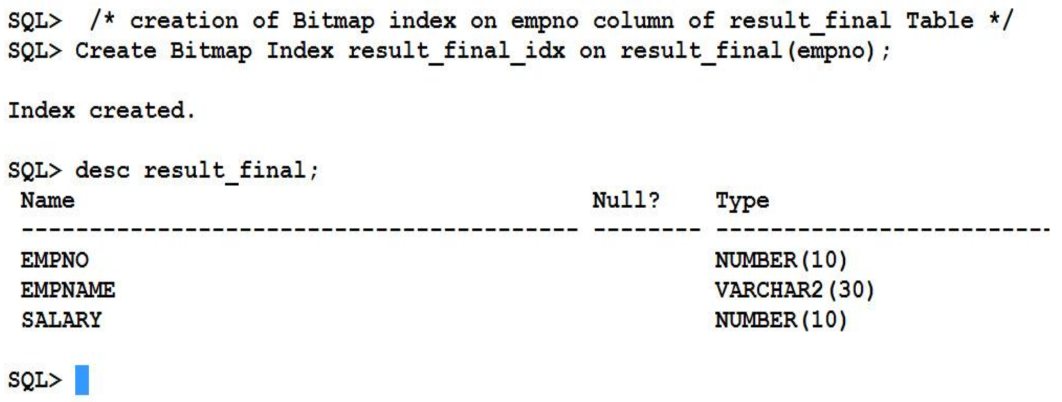
Oracle database provides remap schema to designate the user's object where to import either in same schema or the other's schema. Remap schema mostly used for industrial purpose to locate the user's data from one location to other server location. This feature supports by the data pump Import to relocate the object data

Step-1.1 To prove the compression of Bitmap index or release disk space after deletion of records we have table test_final which contains five million records.



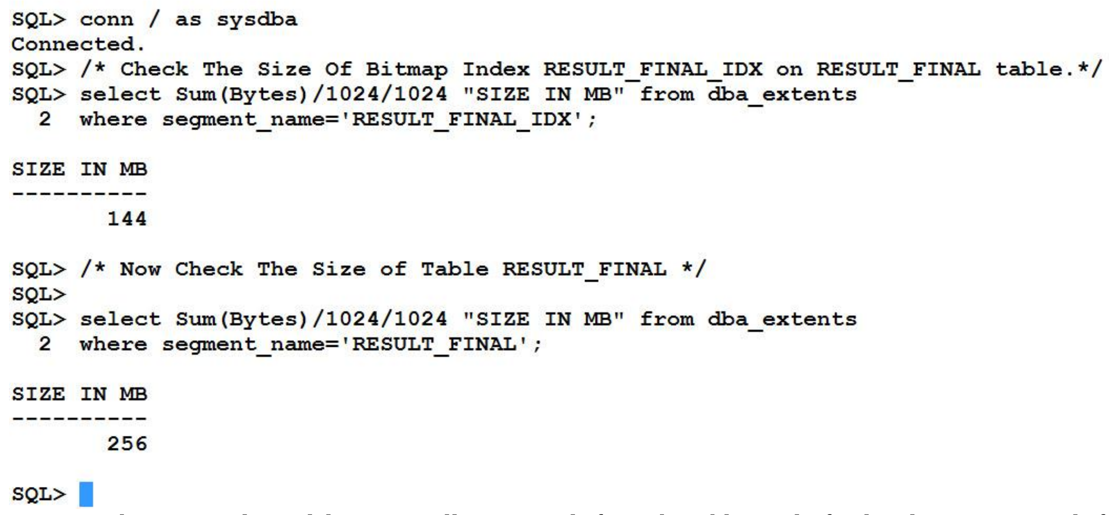
```
SQL> Conn Tom
Enter password:
Connected.
SQL> /* Calulate Number of Records in the Table */
SQL> select count(*) "Number of Values" from result_final;

Number of Values
----------------
         5000000

SQL>
```

Step-1.2 Now create bitmap index on empno column of table result_final which contains five million records.



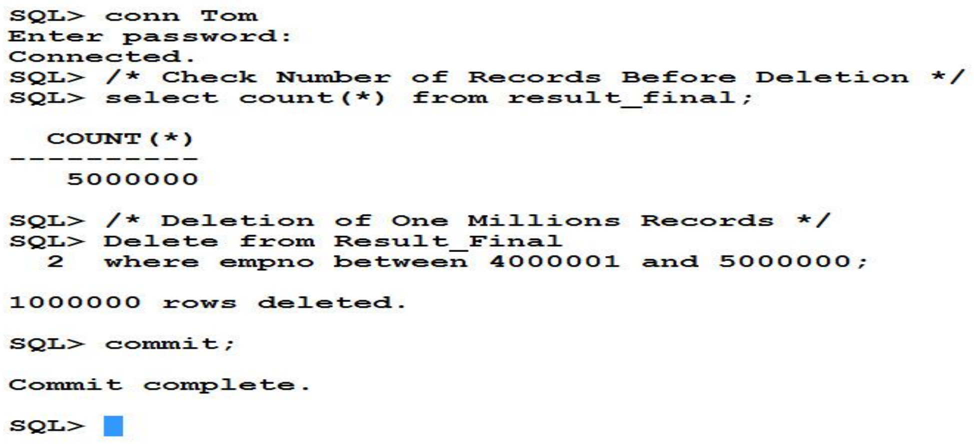
```
SQL>  /* creation of Bitmap index on empno column of result_final Table */
SQL> Create Bitmap Index result_final_idx on result_final(empno);

Index created.

SQL> desc result_final;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
 EMPNO                                              NUMBER(10)
 EMPNAME                                            VARCHAR2(30)
 SALARY                                             NUMBER(10)

SQL>
```

Step- 2. Check how much space is occupied by the Bitmap index and table result_final.



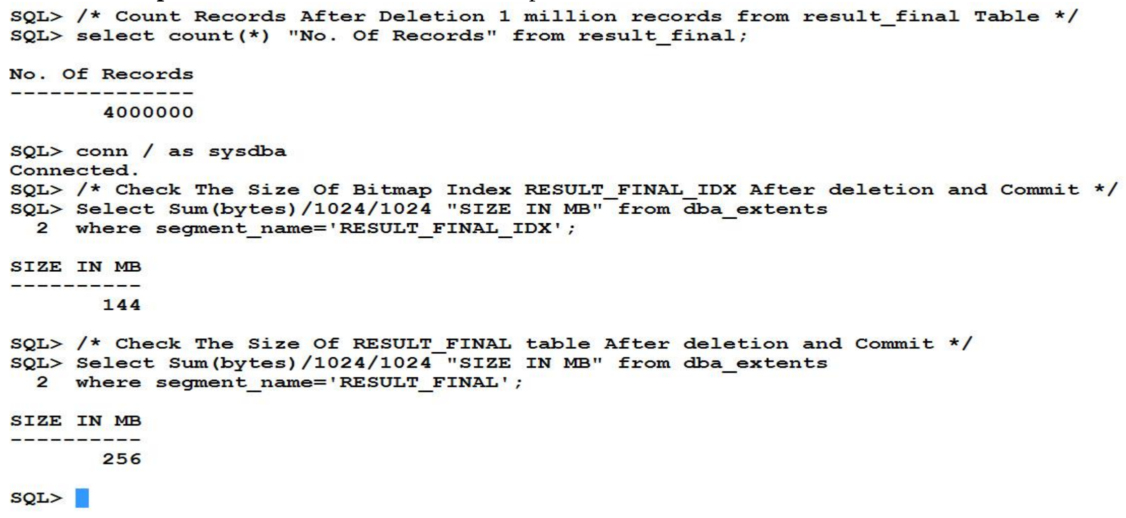
```
SQL> conn / as sysdba
Connected.
SQL> /* Check The Size Of Bitmap Index RESULT_FINAL_IDX on RESULT_FINAL table.*/
SQL> select Sum(Bytes)/1024/1024 "SIZE IN MB" from dba_extents
  2  where segment_name='RESULT_FINAL_IDX';

SIZE IN MB
----------
       144

SQL> /* Now Check The Size of Table RESULT_FINAL */
SQL>
SQL> select Sum(Bytes)/1024/1024 "SIZE IN MB" from dba_extents
  2  where segment_name='RESULT_FINAL';

SIZE IN MB
----------
       256

SQL>
```

Step-3. To show our utilities delete one million Records from the table result_final and commit records for permanent changes applying in database.



```
SQL> conn Tom
Enter password:
Connected.
SQL> /* Check Number of Records Before Deletion */
SQL> select count(*) from result_final;

  COUNT(*)
-----------
   5000000

SQL> /* Deletion of One Millions Records */
SQL> Delete from Result_Final
  2  where empno between 4000001 and 5000000;

1000000 rows deleted.

SQL> commit;

Commit complete.

SQL>
```

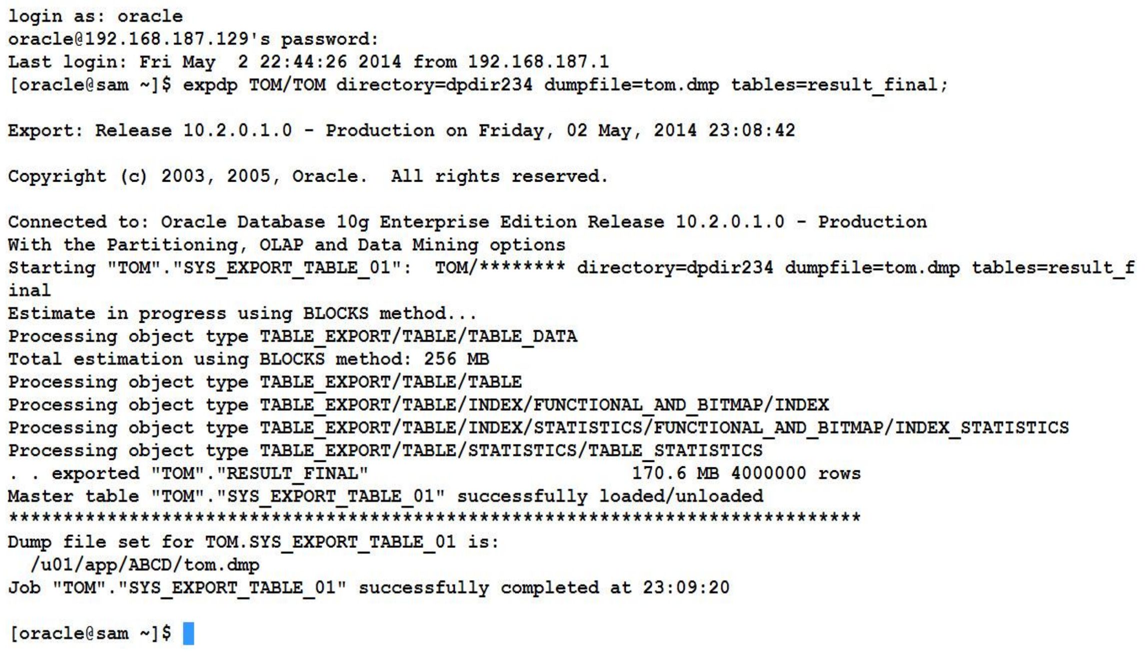
Step-3.1 Now check the size of Bitmap index result_final_idx and table result_final.



```
SQL> /* Count Records After Deletion 1 million records from result_final Table */
SQL> select count(*) "No. Of Records" from result_final;

No. Of Records
--------------
       4000000

SQL> conn / as sysdba
Connected.
SQL> /* Check The Size Of Bitmap Index RESULT_FINAL_IDX After deletion and Commit */
SQL> Select Sum(bytes)/1024/1024 "SIZE IN MB" from dba_extents
  2  where segment_name='RESULT_FINAL_IDX';

SIZE IN MB
----------
       144

SQL> /* Check The Size Of RESULT_FINAL table After deletion and Commit */
SQL> Select Sum(bytes)/1024/1024 "SIZE IN MB" from dba_extents
  2  where segment_name='RESULT_FINAL';

SIZE IN MB
----------
       256

SQL>
```

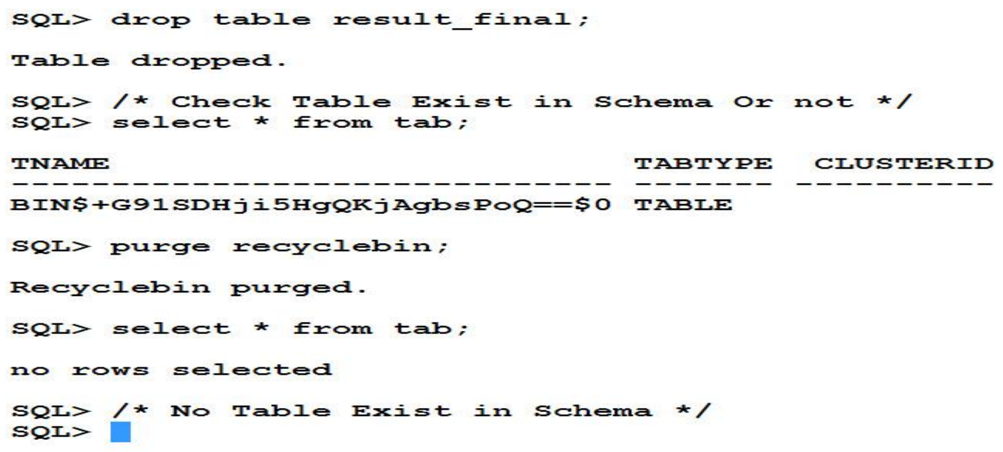
The above results completely showing bitmap index and table result_final still occupying same disk space even after deletion of records and applying commit statement. Now for release the disk space we use data pump utility.

Step- 4. Export Data pump Utility result_final table for compression of objects and release the disk space.



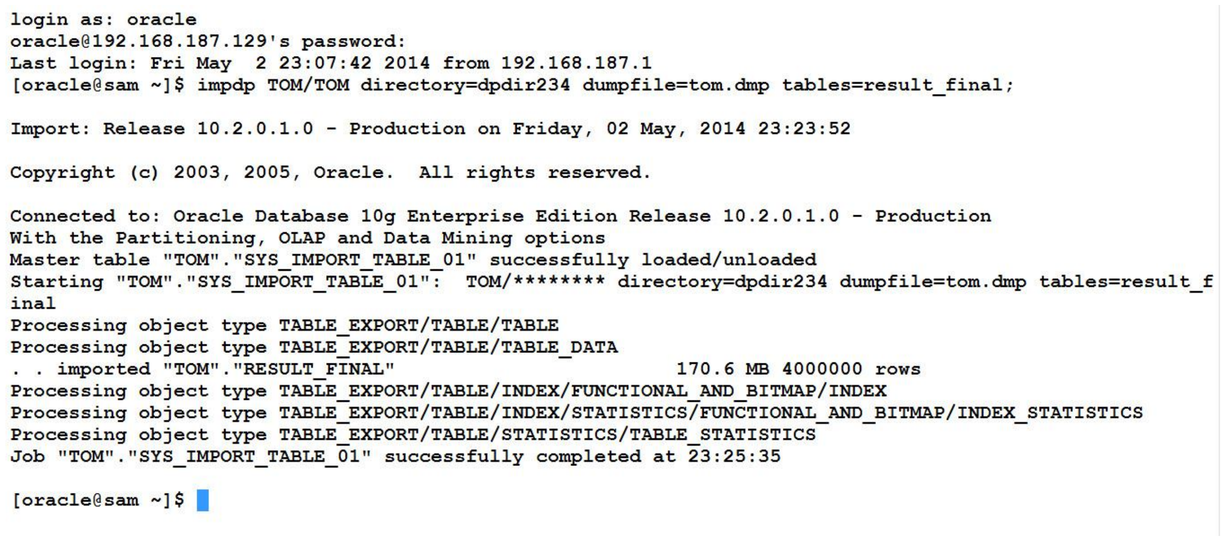
```
login as: oracle
oracle@192.168.187.129's password:
Last login: Fri May  2 22:44:26 2014 from 192.168.187.1
[oracle@sam ~]$ expdp TOM/TOM directory=dpdir234 dumpfile=tom.dmp tables=result_final;

Export: Release 10.2.0.1.0 - Production on Friday, 02 May, 2014 23:08:42

Copyright (c) 2003, 2005, Oracle.  All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
Starting "TOM"."SYS_EXPORT_TABLE_01":  TOM/******** directory=dpdir234 dumpfile=tom.dmp tables=result_f
inal
Estimate in progress using BLOCKS method...
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 256 MB
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/INDEX/FUNCTIONAL_AND_BITMAP/INDEX
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_AND_BITMAP/INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
. . exported "TOM"."RESULT_FINAL"                        170.6 MB 4000000 rows
Master table "TOM"."SYS_EXPORT_TABLE_01" successfully loaded/unloaded
******************************************************************************
Dump file set for TOM.SYS_EXPORT_TABLE_01 is:
  /u01/app/ABCD/tom.dmp
Job "TOM"."SYS_EXPORT_TABLE_01" successfully completed at 23:09:20

[oracle@sam ~]$
```

Step-4.1 Before applying Import Data pump utility, drop result_final Table to release space.



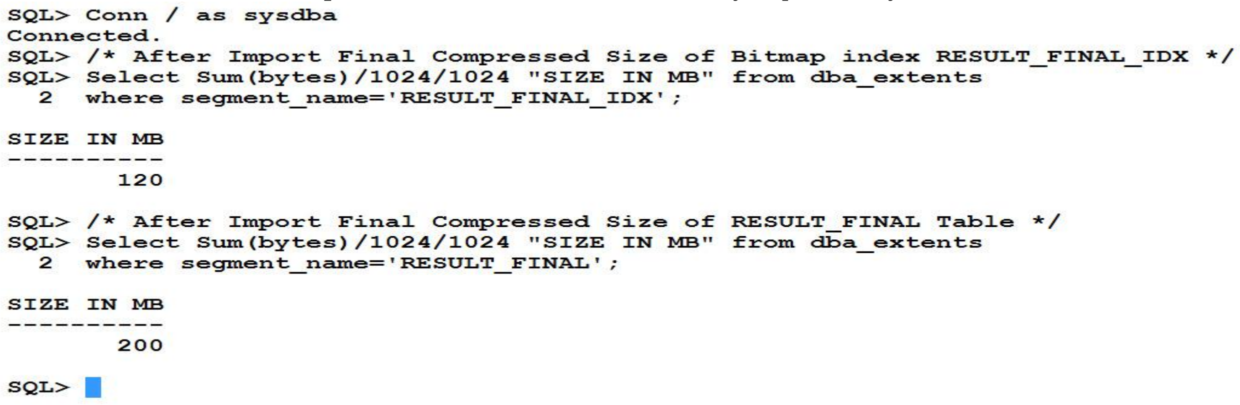
```
SQL> drop table result_final;

Table dropped.

SQL> /* Check Table Exist in Schema Or not */
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BIN$+G91SDHji5HgQKjAgbsPoQ==$0 TABLE

SQL> purge recyclebin;

Recyclebin purged.

SQL> select * from tab;

no rows selected

SQL> /* No Table Exist in Schema */
SQL>
```

Step-4.2 Now Import the result_final table

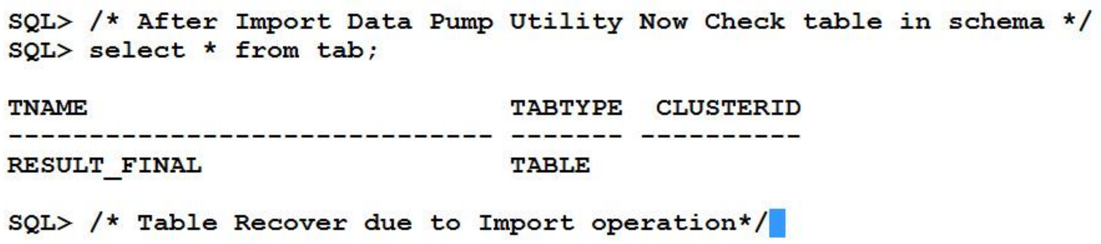


```
login as: oracle
oracle@192.168.187.129's password:
Last login: Fri May  2 23:07:42 2014 from 192.168.187.1
[oracle@sam ~]$ impdp TOM/TOM directory=dpdir234 dumpfile=tom.dmp tables=result_final;

Import: Release 10.2.0.1.0 - Production on Friday, 02 May, 2014 23:23:52

Copyright (c) 2003, 2005, Oracle.  All rights reserved.

Connected to: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
Master table "TOM"."SYS_IMPORT_TABLE_01" successfully loaded/unloaded
Starting "TOM"."SYS_IMPORT_TABLE_01":  TOM/******** directory=dpdir234 dumpfile=tom.dmp tables=result_f
inal
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "TOM"."RESULT_FINAL"                        170.6 MB 4000000 rows
Processing object type TABLE_EXPORT/TABLE/INDEX/FUNCTIONAL_AND_BITMAP/INDEX
Processing object type TABLE_EXPORT/TABLE/INDEX/STATISTICS/FUNCTIONAL_AND_BITMAP/INDEX_STATISTICS
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "TOM"."SYS_IMPORT_TABLE_01" successfully completed at 23:25:35

[oracle@sam ~]$
```

Step-4.4 Table result_final is recovered by Import Utility.



```
SQL> Conn / as sysdba
Connected.
SQL> /* After Import Final Compressed Size of Bitmap index RESULT_FINAL_IDX */
SQL> Select Sum(bytes)/1024/1024 "SIZE IN MB" from dba_extents
  2  where segment_name='RESULT_FINAL_IDX';

SIZE IN MB
----------
       120

SQL> /* After Import Final Compressed Size of RESULT_FINAL Table */
SQL> Select Sum(bytes)/1024/1024 "SIZE IN MB" from dba_extents
  2  where segment_name='RESULT_FINAL';

SIZE IN MB
----------
       200

SQL>
```



```
SQL> /* After Import Data Pump Utility Now Check table in schema */
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
RESULT_FINAL                   TABLE

SQL> /* Table Recover due to Import operation*/
```

Step-5 This is final step which shows the final Compression of Bitmap Index or release occupied disk space by the database objects Bitmap index and Table.

IV. Experiment Results

From the above results we have

1. 18 percent occupied disk space of Bitmap index RESULT_FINAL_IDX is released by data pump after deletion of one million records.
2. 22 percent occupied disk space of table RESULT_FINAL is released by data pump after deletion of one million records.

This is due to Data Pump import/ export operation performed on the table RESULT_FINAL.

V. Conclusion

In this section we present the experiment evaluation of Bitmap Index Compression and release occupied disk space of database objects like table and indexes after deletion of records. Industrial database frequently allows the bulk data insertion and deletion .In database deletion of millions records from the table doesn't release occupied disk space immediately. To utilize this disk space for useful records data rather than wasting it. In our study we used the data pump utility for the compression of Bitmap index and table. Data pump utility performed for the logical backups in database. Limitations of this compression technique through data pump that we need to drop the table for release the disk space. The Experimental results, it is observed that in compression of bitmap index or release the disk space is possible through the data pump utility.

Next steps in our research will be to release the disk space along with the deletion of records. Oracle database comprises the behavior to occupy the disk space due segment related issue. Because once segment grow for available records. It will not release the space even after deletion of records. This is of special interest for the case to compress the bitmap index and tablespace along with the DML'S (Data Manipulation Language).

Reference

[1]. E. O'Neil, P. O'Neil, K. Wu, Bitmap index design choices and their performance implications, Research Report, Lawrence Berkeley National Laboratory, 2007.

[2]. N. Koudas, Space efficient bitmap indexing, in: Proceedings of ACM Conference on Information and Knowledge Management (CIKM), 2000.

[3]. G. Navarro, E.S. de Moura, M. Neubert, N. Ziviani, R. Baeza-Yates, Adding compression to block addressing inverted indexes, Information Retrieval 3 (1) (2000) 49–77.

[4]. K. Wu, E.J. Otoo, A. Shoshani, Compressing bitmap indexes for faster search operations, in: Proceedings of International Confer- ence on Scientific and Statistical Database Management (SSDBM), 2002.

[5]. M. Stabno, R. Wrembel, RLH: Bitmap compression technique based on run-length and Huffman encoding, in: Proceedings of ACM International Workshop on Data Warehousing and OLAP (DOLAP), 2007.

[6]. T. Apaydin, G. Canahuate, H. Ferhatosmanoglu, A.S. Tosun, Approx- imate encoding for direct access and

query processing over compressed bitmaps, in: Proceedings of International Conference on Very Large Data Bases (VLDB), 2006.

[7]. T. Apaydin, G. Canahuate, H. Ferhatosmanoglu, A.S. Tosun, Approx- imate encoding for direct access and query processing over compressed bitmaps, in: Proceedings of International Conference on Very Large Data Bases (VLDB), 2006.

[8]. K. Wu, P. Yu, Range-based bitmap indexing for high cardinality attributes with skew, in: International Computer Software and Applications Conference (COMPSAC), 1998.

[9]. K.C. Davis, A. Gupta,Indexing in data warehouses: bitmaps and beyond, in: R. Wrembel, C. Koncilia (Eds.),Data Warehouses and OLAP: Concepts, Architectures and Solutions, Idea Group Inc., 2007, pp.179–202 ISBN 1-59904-364-5.

[10]. D. Rotem, K. Stockinger, K. Wu, Optimizing I/O costs of multi- dimensional queries using bitmap indices, in: Proceedings of International Conference on Database and Expert Systems Applica- tions (DEXA), Lecture Notes in Computer Science, vol. 3588, Springer, Berlin, 2005.