ORACLE 11G INVISIBLE INDEXES

Inderpal S. Johal, Data Softech Inc.

INTRODUCTION

In this document we will work on another Oracle 11g interesting feature called Invisible Indexes. This will be very helpful for DBA when a new application is moved to production and there are lots of indexes been created on a table. Oracle optimizer will not use Invisible index unless OPTIMIZER_USE_INVISIBLE_INDEXES initialization parameter is set to TRUE at the session or system level. The default value for this parameter is FALSE.

BENEFIT OF INVISIBLE INDEXES

1. Sometime you find that creating an Index for certain Application query or operation is beneficial. But the new index can degrade the effect of some other Application queries. In such scenario, you can create Invisible index and use them for required query by specifying the OPTIMIZER_USE_INVISIBLE_INDEXES to TRUE at session level. By Default this parameter is FALSE and so Oracle optimizer will ignore all invisible indexes.
2. There are lots of unused indexes in most of the application. They might be on some big tables and so you cannot afford to delete them and then recreate them. We can use Oracle 11g Invisible index feature to make them unavailable for Oracle optimizer. If user are experiencing any performance issue, you can make it visible or else drop it. Please Note that unlike Unusable indexes, Invisible indexes are maintained for all DML statements.

INVISIBLE INDEXES CREATION

Step 1 : Identify the Table on which you want to create the Invisible Index

```
SQL> desc indy_test
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USERID                                    NOT NULL NUMBER
 USERNAME                                  NOT NULL VARCHAR2(100)
 USERSTATUSID                                       NUMBER
 COUNTRY                                            VARCHAR2(20)
 STATE                                              VARCHAR2(20)
```

Step 2 : Create the Invisible Index on Userid column of Indy_test table

```
SQL> create index idx_userid_invisible on indy_test(userid) invisible;
Index created.
```

Step 3 : Check the Data Dictionary to see if the index is Invisible. This is new column ntroduced in 11g

```
SQL>select table_name,index_name,visibility from user_indexes where
table_name='INDY_TEST';
TABLE_NAME           INDEX_NAME                     VISIBILIT
-------------------- ------------------------------ ---------
INDY_TEST            IDX_USERID_INVISIBLE           INVISIBLE
```

INVISIBLE INDEXES AND ORACLE OPTIMIZER

Step 4 : 11g has new initialization parameter OPTIMIZER_USE_INVISIBLE_INDEXES which will enable or disable Oracle optimizer to use invisible indexes.

```
SQL> show parameter invisible
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
optimizer_use_invisible_indexes      boolean     FALSE
```

Step 5 : Check if the OPTIMIZER_USE_INVISIBLE_INDEXES=FALSE, then how Oracle query optimizer works for invisible indexes

```
SQL> explain plan for
  2  select *  from indy_test where userid=585;
Explained.
```

Step 6 : Explain Plan shows that Indexes is not used if I am selecting only one row from almost 150K rows. This is because Index is Invisible and initialization parameter shown above is false.

```
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
PLAN_TABLE_OUTPUT
--------------------------------------------
| Id  | Operation          | Name       | Rows  |
--------------------------------------------
|   0 | SELECT STATEMENT   |            |     1 |
|   1 |  TABLE ACCESS FULL | INDY_TEST  |     1 |
--------------------------------------------
7 rows selected.
```

Step 7 : Let's change the initialization parameter OPTIMIZER_USE_INVISIBLE_INDEXES to TRUE and so the behaviour of Oracle Optimizer

```
SQL> alter session set optimizer_use_invisible_indexes=TRUE;
Session altered.
```

Step 8 : Verify the Parameter setting.

```
SQL> show parameter invisible
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
optimizer_use_invisible_indexes      boolean     TRUE
```

Step 9 : Create the Explain plan to find out the Execution plan

```
SQL> explain plan for
  2  select * from indy_test where userid=585;
Explained.
```

Step 10 : Explain Plan shows that Optimizer is now using the Invisible Index

```
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |     1 |
|   1 |  TABLE ACCESS BY INDEX ROWID| INDY_TEST           |     1 |
|   2 |   INDEX RANGE SCAN           | IDX_USERID_INVISIBLE |     1 |
--------------------------------------------------------------------------
8 rows selected.
```

Step 11 : Now we will verify that if the Index is changed to Visible and initialization parameter is changed to FALSE, then How Oracle optimizer is working

```
SQL> alter session set optimizer_use_invisible_indexes=false;
Session altered.
```

Step 12 : Make index Visible using ALTER INDEX command

```
SQL> alter index IDX_USERID_INVISIBLE visible;
Index altered.
```

Step 13 : Verify the Status of Index in Data dictionary view

```
SQL> select table_name,index_name,visibility from user_indexes where
table_name='INDY_TEST';
TABLE_NAME           INDEX_NAME                     VISIBILIT
-------------------- ------------------------------ ---------
INDY_TEST            IDX_USERID_INVISIBLE           VISIBLE
```

Step 14 : Generate Explain Plan and you will see that Oracle optimizer is now using the index

```
SQL> explain plan for
  2  select * from indy_test where userid=585;
 Explained.
SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |     1 |
|   1 |  TABLE ACCESS BY INDEX ROWID| INDY_TEST           |     1 |
|   2 |   INDEX RANGE SCAN           | IDX_USERID_INVISIBLE |     1 |
--------------------------------------------------------------------------
8 rows selected.
```

INVISIBLE INDEXES AND DML OPERATION

Step 15 : Now we will verify that if Index is Invisible then any DML activity on this indexes column update the Index itself. So let's make the Index Invisible again using ALTER INDEX command

```
SQL> alter index IDX_USERID_INVISIBLE invisible;
```

Index altered.

Step 16 : Check the Existing Size of the Invisible Index

```
SQL> select sum(bytes) from user_segments where
segment_name='IDX_USERID_INVISIBLE';
SUM(BYTES)
----------
   4194304
```

Step 17 : Update the Indexed Column. This update will mark all existing Index entry as DELETED and insert new rows in the Index. So hopefully Index size should be doubled

```
SQL> update indy_test set userid=userid+100;
```

161262 rows updated.

```
SQL> commit;
```

Commit complete.

Step 18 : Check the Size of the Invisible Index again. You will see that any DML applied to Invisible Index column will continue to work with Invisible Index.

```
SQL> select sum(bytes) from user_segments where
segment_name='IDX_USERID_INVISIBLE';
SUM(BYTES)
----------
   7340032
```

Step 19 : Verify the effect of ALTER INDEX REBUILD command on the Invisible Index. You will see that Index Segemnt size is back as all rows marked as DELETE are gone.

```
SQL> alter index IDX_USERID_INVISIBLE rebuild;
```

Index altered.

```
SQL> select sum(bytes) from user_segments where
segment_name='IDX_USERID_INVISIBLE';
SUM(BYTES)
----------
   4194304
```

INVISIBLE INDEXES AND DDL OPERATION

Step 20 : Interesting point in Rebuilding Invisible Index. You can see below the Rebuild Invisible Index changes the Index status to Visible.

```
SQL> alter index IDX_USERID_INVISIBLE invisible;
Index altered.

SQL> select table_name,index_name,visibility from user_indexes where
table_name='INDY_TEST';
TABLE_NAME           INDEX_NAME                     VISIBILIT
-------------------- ------------------------------ ---------
INDY_TEST            IDX_USERID_INVISIBLE           INVISIBLE

SQL> alter index IDX_USERID_INVISIBLE rebuild;
Index altered.

SQL> select table_name,index_name,visibility from user_indexes where
table_name='INDY_TEST';SQL
TABLE_NAME           INDEX_NAME                     VISIBILIT
-------------------- ------------------------------ ---------
INDY_TEST            IDX_USERID_INVISIBLE           VISIBLE
```

INVISIBLE INDEX AND ORACLE HINTS

Step 21 : Another Point to Check is that if INDEX hint work to use the Invisible Index. You will see that it will not work and Oracle Optimizer continue to ignore it.

```
SQL> explain plan for
  2  select /*+ index(indy_test IDX_USERID_INVISIBLE */ *
  3  from indy_test where userid=585;
Explained.

SQL> select plan_table_output
  2  from table(dbms_xplan.display('plan_table',null,'BASIC ROWS'));
PLAN_TABLE_OUTPUT
--------------------------------------------
| Id  | Operation          | Name       | Rows  |
--------------------------------------------
|   0 | SELECT STATEMENT   |            |     1 |
|   1 |  TABLE ACCESS FULL | INDY_TEST  |     1 |
--------------------------------------------
7 rows selected.
```