Case study on Oracle rowcache internals, cached non-existent objects and a describe bug

By Tanel Poder (http://www.tanelpoder.com)

I got a question regarding Metalink note **296235.1** about a describe bug which causes objects to "disappear" when they are described when database is not open.

The question was:

> *If you describe any DBA_* Views (Data dictionary views) in a mount stage, than you will not be able to describe those views even after opening the database. Do you know what happens behind the scenes here?*

I did take a look into this and looks like the issue comes from a deficiency in OPI describe call codepath which incorrectly sets a non-existent flag in dictionary cache when database is closed.

With this post I hope to shed a little light into core parts of Oracle's dictionary cache (also called rowcache) internals and also show how, armed with some internals knowledge, it is possible to reason why some bugs happen and what components are involved. This in turn should help us to diagnose complex problems (potentially involving bugs) better and avoid or work around them too.

Let's look into a test case (Oracle 10.2.0.3 on Linux 32bit):

```
  SQL> startup mount;
  ORACLE instance started.

  Total System Global Area  595591168 bytes
  Fixed Size                  1263128 bytes
  Variable Size             163580392 bytes
  Database Buffers          423624704 bytes
  Redo Buffers                7122944 bytes
  Database mounted.
  SQL>
  SQL> desc dba_tables
  ERROR:
  ORA-04043: object dba_tables does not exist
```

See... I can not describe a table (or view in this case) as Oracle database is not open. This is expected behavior.

So, let's open up the database then and describe the same view again:

```
SQL> alter database open;

Database altered.

SQL> desc dba_tables
ERROR:
ORA-04043: object dba_tables does not exist
```

What?! Still no view like this? I know it exists as I just was able to use it 5 minutes ago!

Let's try a select:

```
  SQL> select count(*) from dba_tables;
  select count(*) from dba_tables
                       *
  ERROR at line 1:
  ORA-00942: table or view does not exist
```

Still nothing. Let's see if we can query any other views:

```
SQL> select count(*) from dba_objects;

  COUNT(*)
----------
     51024
```

...other views seem to work. Hmm... As all other views except the DBA_TABLES work ok then I assume this earlier failed describe changed something for DBA_TABLES access. As the describe happened when database was not open, this change could not have happened inside database - thus it must be somewhere in memory. Dictionary cache (rowcache) is first thing what comes into mind. So, let’s flush it:

```
SQL> alter system flush shared_pool;

System altered.

SQL> desc dba_tables
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                     NOT NULL VARCHAR2(30)
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 TABLESPACE_NAME                                    VARCHAR2(30)
 CLUSTER_NAME                                       VARCHAR2(30)
 IOT_NAME                                           VARCHAR2(30)
...
SQL>
```

Nice, now it works again! So, flushing the shared pool (which includes dictionary cache) helped us to resolve the issue.

So, what happened here?

First, we need to think how Oracle looks up objects from database. You all know there's a data dictionary in SYSTEM tablespace. Whenever you parse a statement which references any database objects, then Oracle process will go to dictionary cache and get that object's information from there. This is assuming the object's info is cached in dictionary cache. If it isn't, a cache miss happens, which causes a recursive SQL statement (or many) to be fired against data dictionary base tables (such OBJ$, TAB$, SEG$, etc...) and the result is used for populating relevant objects in dictionary cache.

So, the dictionary cache acts as a cache for existing objects in data dictionary. Makes very much sense performance wise, as otherwise you would need to always execute the same recursive SQL again when parsing statements.

But the dictionary cache also caches names of some non-existing objects!
Doesn't sound sane? Read these two links about the reasons why Oracle needs non-existent object markers:

- http://www.jlcomp.demon.co.uk/faq/non_exist.html
- http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/general008.htm

In addition to above, there is one more reason for caching non-existent objects. It's just performance. Let say an application tries to access a non-existent table APPS.T. The first access causes some recursive SQL to be fired in order to get that table's information fro data dictionary tables. However as this table doesn't exist, an error is returned back to application, but also, an entry for table APPS.T is kept in row cache with EXISTS=N flag, which says such table doesn't exist in the database. So, the next time this application tries to access the same table, Oracle can return the error immediately without having to rerun all the recursive SQLs. When the non-existent rowcache entry is flushed out the shared pool, the recursive SQLs must be used again.

So, back to the issue. Let's check what happens in rowcache when we try to access an existing object. I'm using a simple script rowcache.sql for this purpose (it uses V$ROWCACHE_PARENT as its source, you can download it from http://www.tanelpoder.com/files/scripts/rowcache.sql).

I create a new table and flush shared pool after it, to get rid of rowcache entries:

```
SQL> create table new_table (a int);

Table created.

SQL> alter system flush shared_pool;

System altered.

SQL> @rowcache new_table

no rows selected
```

So, now let’s access that table:

```
SQL> select * from new_table;

no rows selected

SQL> @rowcache new_table

      INDX       HASH ADDRESS  EXIST CACHE_NAME           KEY
---------- ---------- -------- ----- -------------------- --------------------------------
       198      41101 3E9F8AA4 Y     dc_objects           3D00000009004E45575F5441424C4500
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000
```

Oracle has fetched data from data dictionary and populated a rowcache entry for object new_table, the field EXISTENT=Y, thus this object exists.
See the contents of my rowcache.sql script to understand how this lookup is done. The object name string is available the V$ROWCACHE_PARENT.KEY column in raw form.

Reading first KEY line backwards:

- "4E45575F5441424C45" - is a hex representation of "new_table" characters
- "0900" means 9 (in little endian form on my test platform), which is the object name character length.
- 3D000000 in the beginning (again in little endian form) is the schema ID this cached object belongs.

We can easily get the corresponding schema name with a query:

```
SQL> select user from dual;
```

```
USER
------------------------------
TANEL

SQL> select username from dba_users where user_id = to_number('3D', 'XX');

USERNAME
------------------------------
TANEL
```

Now, let’s see what happens when accessing a **non-existent object**. First let’s check if there are any rowcache entries cached for MYTABLE (I know it doesn't exist):

```
SQL> @rowcache mytable

no rows selected

SQL> select count(*) from mytable;
select count(*) from mytable
                     *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> @rowcache mytable

      INDX       HASH ADDRESS  EXIST CACHE_NAME           KEY
---------- ---------- -------- ----- -------------------- --------------------------------
      1387      54401 3CA834E8 N     dc_objects           3D00000007004D595441424C45000000
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000

      1454      59615 3CA81F38 N     dc_objects           0100000007004D595441424C45000000
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000
```

You see! There are two rowcache objects created with EXISTENT=N, which means object with this name does *not* exist in database.

So, why are there two entries with different hash values? This indicates that there are two different objects involved for some reason. The reason is the PUBLIC synonym namespace and it can be seen from two different schema ID values in beginning of KEY fields (3D and 01). 3D belongs to schema TANEL and 01 is PUBLIC schema (btw, SYS is 0).

When I run select * from mytable without a schema qualifier, first a rowcache lookup is done in my current schema (and as the lookup failed, a non-existent object was created for that). And second, another lookup was performed in PUBLIC pseudo schema, which also failed and caused a non-existent marker to be generated. The hash values for same object name are different because the hashing function takes also the schema name as its input (and dblink name and object namespace too as a matter of fact).

Let's see if we get only one rowcache entry when referring to a table with qualified schema name:

```
SQL> alter system flush shared_pool;

System altered.

SQL> @rowcache mytable

no rows selected

SQL> select count(*) from tanel.mytable;
select count(*) from tanel.mytable
                           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> @rowcache mytable

      INDX       HASH ADDRESS  EXIST CACHE_NAME           KEY
---------- ---------- -------- ----- -------------------- --------------------------------
       179      54401 4312EC34 N     dc_objects           3D00000007004D595441424C45000000
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000
```

As predicted, only one non-existent marker was created (as the other lookup for public schema was not done).

Ok, now that we've explored some rowcache internals, let's get back to the original problem. Objects seem to disappear if they are described when database is not open. I'm pasting the test case here again:

```
SQL> startup mount;
ORACLE instance started.

Total System Global Area  595591168 bytes
Fixed Size                  1263128 bytes
Variable Size             163580392 bytes
Database Buffers          423624704 bytes
Redo Buffers                7122944 bytes
Database mounted.
SQL>
SQL> desc dba_tables
ERROR:
ORA-04043: object dba_tables does not exist

SQL> alter database open;

Database altered.

SQL> desc dba_tables
ERROR:
ORA-04043: object dba_tables does not exist


SQL> select * from dba_tables;
select * from dba_tables
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

As you see, the dba_tables object seems to be gone from the database.

Now let’s walk through this test case with using the rowcache.sql script to see what's happening in rowcache:

```
SQL> startup mount;
ORACLE instance started.

Total System Global Area  595591168 bytes
Fixed Size                  1263128 bytes
Variable Size             163580392 bytes
Database Buffers          423624704 bytes
Redo Buffers                7122944 bytes
Database mounted.

SQL> @rowcache dba_tables

no rows selected
```

Right after startup there are no rowcache entries about dba_tables. This is expected. Let’s describe dba_tables now:

```
SQL> desc dba_tables
ERROR:
ORA-04043: object dba_tables does not exist


SQL> @rowcache dba_tables

      INDX       HASH ADDRESS  EXIST CACHE_NAME           KEY
---------- ---------- -------- ----- -------------------- ---------------------------------
         0      34505 4324D8B8 N     dc_objects           000000000A004442415F5441424C4553
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000

         1      36489 4324D1C8 N     dc_objects           010000000A004442415F5441424C4553
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000
```

As the dictionary lookups for describe failed (as database is closed), there are non-existent rowcache entries created for "DBA_TABLES" object. From the first bytes of KEY field we see (00 and 01) that the non-existent entries were created for DBA_TABLES both in SYS and PUBLIC schema.

Let's open up the database:

```
SQL> alter database open;

Database altered.

SQL> @rowcache dba_tables

      INDX       HASH ADDRESS  EXIST CACHE_NAME           KEY
```

```
---------- ---------- -------- ----- -------------------- --------------------------------
       494      34505 4324D8B8 **N**     dc_objects           000000000A004442415F5441424C4553
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000

       506      36489 4324D1C8 **N**     dc_objects           010000000A004442415F5441424C4553
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000


SQL> desc dba_tables
ERROR:
ORA-04043: object dba_tables does not exist
```

Even after db open, the non-existent markers are still in rowcache and this is exactly the reason why this DBA_TABLES object seems to be disappeared (remember, rowcache also acts as cache for non-existent object names). If cache explicitly says an object doesn't exist, Oracle doesn't look further (into dictionary tables).

As shared pool flush also flushes out not-locked rowcache entries, it can be used for making the lost objects reappear:

```
SQL> alter system flush shared_pool;

System altered.

SQL> @rowcache dba_tables

no rows selected

SQL> desc dba_tables
           Name                            Null?    Type
           ------------------------------- -------- --------------------
    1      OWNER                           NOT NULL VARCHAR2(30)
    2      TABLE_NAME                      NOT NULL VARCHAR2(30)
    3      TABLESPACE_NAME                          VARCHAR2(30)
    4      CLUSTER_NAME                             VARCHAR2(30)
    5      IOT_NAME                                 VARCHAR2(30)
    6      STATUS                                   VARCHAR2(8)
    7      PCT_FREE                                 NUMBER

[...snip...]
```

So, this problem only happens when *describing* objects when database is closed.
When I try to access the object by other means (like plain select), this problem doesn't happen, but the error message returned gives an important clue for understanding the bug:

```
SQL> startup mount;
ORACLE instance started.

Total System Global Area  595591168 bytes
Fixed Size                  1263128 bytes
Variable Size             163580392 bytes
Database Buffers          423624704 bytes
Redo Buffers                7122944 bytes
```

```
Database mounted.
SQL>
SQL> @rowcache dba_tables

no rows selected

SQL> select count(*) from dba_tables;
select count(*) from dba_tables
                     *
ERROR at line 1:
ORA-01219: database not open: queries allowed on fixed tables/views only


SQL> @rowcache dba_tables

no rows selected
```

See, the error message is different for a SELECT, it says database is not open and queries not allowed. Therefore it looks like the dictionary lookup wasn't even performed as Oracle knew the database was closed. And as a result the non-existent markers were not created (as dictionary lookup function kqrget() is the one responsible for that).

Let's see the describe again:

```
SQL> desc dba_tables
ERROR:
ORA-04043: object dba_tables does not exist


SQL> @rowcache dba_tables

      INDX       HASH ADDRESS  EXIST CACHE_NAME           KEY
---------- ---------- -------- ----- -------------------- --------------------------------
         0      34505 4324E478 N     dc_objects           000000000A004442415F5441424C4553
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000

         1      36489 4324DD88 N     dc_objects           010000000A004442415F5441424C4553
                                                          00000000000000000000000000000000
                                                          00000000010000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000000000000000000000000000
                                                          00000000


SQL>
```

We get a different error for describe, saying object doesn't exist. So it looks like the dictionary lookup was performed, but it failed to return the object as database was closed. Thus the non-existent record was populated by lookup function. And database open doesn't clear this out.

As a conclusion, I would say the bug comes from describe OPI call not checking whether the database is open, before trying to do a dictionary lookup. Thus it incorrectly creates the non-existent markers into dictionary cache. The describe code should be fixed that it would also return the *ORA-01219: database not open* error when user is attempting to describe a non-fixed object (I mention a non-fixed because, fixed V$ and X$ objects are hardcoded into Oracle kernel and don't need data dictionary lookups at all).