Dbvisit
THE SMART ALTERNATIVE

What a DBA Should Know or How to Know Your Way Around Database

Vit Spinka

Agenda

- Dictionary views and actual dictionary tables
- Files, rowids and TTS
- Clusters and partitions
- IOTs
- Nested and hidden columns
- Multitenant / pluggable databases

Vit Spinka

- Working with Oracle Database since 8*i*
- Oracle Certified Master
- Principal developer of Dbvisit Replicate
- … which gets its data by parsing Oracle redo logs

- @vitspinka
- vit.spinka@dbvisit.com
- This presentation for download at http://vitspinka.cz/download.html

Dbvisit

- HQ in New Zealand, US subsidiary, partners throughout the world
- Used in 110+ Countries
- Database Replication is our playground
- Worldwide leader in DR solutions for Oracle Standard Edition
- Product Engineers with “real world” DBA Experience
- Regular presenters at Oracle events such as OOW, Collaborate and NZOUG
- Passionate about Oracle Technology

Trusted in 110+ countries. . . By 1000+ companies

Dictionary views

Dbvisit
THE SMART ALTERNATIVE

- Usual way to gain information about persistent database objects: query dictionary views

- DBA_DATA_FILES
  Describes files in database: file_id, relative_fno
- DBA_OBJECTS
  Tables, indexes, partitions: object_id, data_object_id
- DBA_TABLES
  Tables - indications only: cluster_name, iot_type

Dictionary views

Dbvisit
THE SMART ALTERNATIVE

- Usual way to gain information about persistent database objects: query dictionary views

- DBA_TAB_COLS
  Columns in a table: column_id, segment_column_id, internal_column_id
- DBA_PDBS
  Containers: pdb_id, con_uid, dbid, con_id
- DBA_EDITIONS
  PL/SQL Editions

Dictionary views

- And for most of the views, there are three (four now) flavors
- USER - owner = user
- ALL - user has privileges
- DBA - all in this database
- CDB - all in all pluggable databases (12c multitenant)

Dictionary tables

- However, sometimes you want / need to go deeper to better understand what is going on
- Or to obtain information hidden from DBA views
- Their DDL is easily available (?/rdbms/admin/sql.bsq) and is often well commented

```
create table obj$                                                  /* object table */
( obj#          number not null,                                  /* object number */
  /* DO NOT CREATE INDEX ON DATAOBJ#  AS IT WILL BE UPDATED IN A SPACE
   * TRANSACTION DURING TRUNCATE */
  dataobj#      number,                                 /* data layer object number */
  owner#        number not null,                              /* owner user number */
  name          varchar2("M_IDEN") not null,                    /* object name */
  namespace     number not null,          /* namespace of object (see KQD.H): */
 /* 1 = TABLE/PROCEDURE/TYPE, 2 = BODY, 3 = TRIGGER, 4 = INDEX, 5 = CLUSTER, */
                                                        /* 8 = LOB, 9 = DIRECTORY, */
  /* 10 = QUEUE, 11 = REPLICATION OBJECT GROUP, 12 = REPLICATION PROPAGATOR, */
                                                /* 13 = JAVA SOURCE, 14 = JAVA RESOURCE */
                                                /* 58 = (Data Mining) MODEL */
```

Dictionary tables

- SYS.OBJ$ - objects
- SYS.TAB$, SYS.COL$ - tables and columns
- SYS.IND$, SYS.ICOL$ - indexes and columns
- SYS.CDEF$, SYS.CCOL$ - constraints and columns
- SYS.TS$, SYS.FILE$ - tablespaces and files
- SYS.EDITION$ - PL/SQL editions
- SYS.CONTAINER$ - PDBs

Files

- Until Oracle7, a single file number (up to 1023) was enough to identify a datafile since birth to drop (today known as “absolute file number”)
- Oracle8 brought higher limit of files in a database
- Oracle8 brough TTS (transportable tablespaces)
- Solution: add “relative file number”, guaranteed to be unique in a single tablespace only
- As long as you have less then 1023 datafiles and don’t import TTS, absolute and relative file numbers will match. But don’t depend on it in your scripts.

Rowid

- Oracle7 has what we now call “restricted rowid” - block, row in block, (absolute) file number
- Oracle8 introduced “extended rowid” - data object id, relative file number, block, row in block (and big datafile changes it again a bit)
- Relative file number is limited to 1023 - now that’s the limit for number of datafiles in a single tablespace
- Rowid is not unique: relative file number is not unique, so having many datafiles and/or use of TTS can lead to identical rowids (small chance if random - but high chance if you export-import your own tablespace)
- Rowid is not unique: rows in cluster share datablocks (cf. next slides)

Object id and data object id

Dbvisit
THE SMART ALTERNATIVE

- Object id, the most known id, present in dba_objects
- Data object id identifies the segment
- Ordinary table is created with data object id = object id
- Table gets new data object id when it's truncated
- (and things like exchange partition)

- And base object id: link to "parent" object (e.g. index -> table)

Clustered tables

- Two or more tables share the same blocks
- And thus they share the same segment
- And thus they have the same data_object_id
- But they still keep distinct object_id
- sys.tab$ has column tab# (table number in cluster or NULL if not clustered)

```
SQL> select obj#, dataobj#, tab# from sys.tab$ where …
      OBJ#    DATAOBJ#     TAB#
---------- ---------- ----------
    482583     482581         1
    482584     482581         2
    482585     482581         3
```

Partitions

- Table has one object_id
- Table is split into partitions and subpartitions - and each of them is a segment
- And each of them is again an object - and thus has object_id and data_object_id
- (sys.obj$.type# = 2 = TABLE, 19 = TABLE PARTITION, 34 = TABLE SUBPARTITION)
- Link to parent table: tabpart$.bo# (base object id) or obj$.subname (name of the parent table)

An IOT has more ids

- Object id of index
- Object id of table
- Object id of overflow segment
- Object id of mapping table
- And for each of those corresponding data object id (and with partitions…)
- Link to parent table: obj$.bo# (base object id), tab$.bobj# and as usual ind $.bo#
- (And see flags and property to see if a table is IOT, if it has an overflow segment, if it has mapping table)

An IOT has more ids

```
SQL> select obj#, tab$.bobj#, name from tab$, obj$ …
      OBJ#      BOBJ# NAME
---------- ---------- ------------------------------
    476774     476773 SYS_IOT_OVER_476773
    476773     476774 IOTTABLE
```

Columns

- There are three id’s
- col#: column id - for internal columns, this is 0. The value usually presented to the user.
- intcol#: internal column id. Unique and thus set for each column.
- segcol#: position in segment. 0 for columns not stored (e.g. virtual or SYS_NC_ROWINFO$)

col#

- An id that user expects
- Zero for internal columns (note that these still count towards the 1000 column limit) like SYS_NC_OID$
- Or can be same for a user-facing column (ADT) and it's internal data

```
 COL# SEGCOL#    INTCOL# NAME
----- ------- ---------- ----------------------
    1       1          1 CUSTID
    2       0          2 ADDRESS
    2       2          3 SYS_NC00002000003$
```

intcol#

- Internal id that is unique
- And thus can be used to create indexes and constraints on ADT
- Still, Oracle uses both col# and intcol# in other tables (even icol$ and ccol$)

segcol#

- Segment column id
- Some columns are present in the dictionary, but not stored - nested table column, virtual specified by user, virtual created by Oracle (e.g. extended stats), pseudo nc_rowinfo$ used to access XMLType value
- A LONG value must be always the last column in a table - so it will have the highest segcol#
- Cluster keys are always first columns in the table (as they are actually stored in a different table)

Editions

Dbvisit
THE SMART ALTERNATIVE

- Edition-based redefinition introduces editions (“versions”) of PL/SQL, views and synonyms
- Creates many “adjunct schemas” and the editioned objects are owned by these sys users (SYS_%)
- Uses obj$.spare3 (base=“the normal and expected” user) and user$.spare2 (edition id for adjunct schemas)
- Views are built on top of _CURRENT_EDITION_OBJ instead of OBJ$
- This view resolves editions and shows just the current one
- So you don’t see the other editions’ objects in usual DBA views (but there are %_AE views for All Editions)

Editions

- Each version of object has a different object id

```
SELECT object_id, object_name, object_type, edition_name
FROM   user_objects_ae
ORDER BY object_name, edition_name;
 OBJECT_ID OBJECT_NAME          OBJECT_TYPE         EDITION_NAME
---------- -------------------- ------------------- ----------------------
     72613 CREATE_EMPLOYEE      PROCEDURE           RELEASE_V1
     72614 CREATE_EMPLOYEE      PROCEDURE           RELEASE_V2
     72609 EMPLOYEES            VIEW                RELEASE_V1
     72610 EMPLOYEES            VIEW                RELEASE_V2
     72607 EMPLOYEES_PK         INDEX
     72608 EMPLOYEES_SEQ        SEQUENCE
     72606 EMPLOYEES_TAB        TABLE
```

RAC

- RAC is a shared data architecture
- This means it does not really affect data stored in the database
- Undo and redo is locally written by each node, but accessible to all nodes (for recovery)
- So the real issue is with memory-based v$ views, as they no longer show complete picture
- So “always” use gv$ views (but beware of few traps like gvlog / gvlogfile / gv$thread)

Multitenant (CDBs)

Dbvisit
THE SMART ALTERNATIVE

- Multitenant brings one more layer of naming resolution
- All of the things discussed so far are true within a single PDB
- For data related queries the simplest solution is to think at PDB level (and connect to a PDB)
- For queries spanning PDBs - all CDB views have con_id column, which is *current* container id
- If you replug databases, this con_id will change; then use con_uid (“unique id”), kind of DBID at container level
- Note that dictionary tables are stored in PDBs themselves (with some link magic to CDB tables)

Multitenant - global views

- CDB views use magic similar to RAC views
- They query dictionary tables across all PDBs

```
SELECT "OWNER","OBJECT_NAME","SUBOBJECT_NAME","OBJECT_ID",
"DATA_OBJECT_ID","OBJECT_TYPE","CREATED","LAST_DDL_TIME",
"TIMESTAMP","STATUS","TEMPORARY","GENERATED","SECONDARY",
"NAMESPACE","EDITION_NAME","SHARING","EDITIONABLE",
"ORACLE_MAINTAINED","CON_ID" FROM CONTAINERS("SYS"."DBA_OBJECTS")
```

- But beware that CONTAINERS do not include PDB$SEED

Multitenant - global data

- Absolute file numbers are still global
- OMF puts files into different directories (one per PDB)
- Tablespace names are not unique; use DB:TBS syntax in RMAN
- And because of that CONTAINERS clause, CDB_DATA_FILES won't show PDB$SEED files

Thank you - Q&A

Like RacAttack, try RepAttack
http://tinyurl.com/njhyln6

info@dbvisit.com
@dbvisit