The Best Oracle 12c New Features

Collaborate 2013

Rich Niemiec, Rolta (www.rolta.com)

(Thanks: Andy Mendelsohn, Debbie Migliore, Maria Colgan, Penny Avril, Jacob Niemiec , & Lucas Niemiec)

How Much Data …

- 2004 monthly internet traffic >1E; 2010 it was 21E/month.
- In 2012, **2.5E data created every day** (about 1Z=1000E per year)
- June 2012 – Facebook has **100P Hadoop cluster**
- Facebook: **500T** processed daily – (210T/hr DWHSE scanned)
- A Single Jet Engine **– 20T/hour** –same rate as Facebook!
- Gmail has **450 million users**
- Wal-Mart – 1 million customer transactions/hour (2.5P DB)
- Large Hadron Colider produced **13P in one year**
- Business data **doubles every 1.2 years**
- 19% of $1B **companies have >1P of data** (31% in 2013)
- 2011 – First **Exabyte tape library** from Oracle
- Decoding Human Genome took 10 yrs; Now takes a week!

Overview – 12c

- Know the Oracle!
- Start Me Up – Using Memory Target, The Buffer Cache & The Result Cache
- Invisible Columns (12c) & virtual columns (11g)
- Multiple indexes on the same Column (12c) & Invisible Indexes (11g)
- Adaptive Execution Plans (12c) & Adaptive Cursor Sharing & Bind Peeking (11g)
- Runaway query Management (12c)
- Change Table Compression at import Time (12c) & (Partition Compression – 11g)
- Create Views as Tables (12c)
- Online Move Partition (12c) & Interval Partitioning (11g)
- Partial Indexes for Partitioned Table (12c)
- Pluggable Databases (12c)
- Enhanced DDL Online (12c)
- Exadata and Big Data (In-Database MapReduce in 12c)
- Consolidated Database Replays & Better Reporting (12c)
- Automatic Diagnostics Repository (12c)
- Security Enhancements (12c)
- Other 12c New Features

Know the Oracle

"I admire risk takers. I like leaders – people who do things before they become fashionable or popular. I find that kind of integrity inspirational."

LAWRENCE J. ELLISON | *Chairman & Chief Executive Officer, 2003*

Oracle Firsts – *Innovation!*

1979 First commercial SQL relational database management system
1983 First 32-bit mode RDBMS
1984 First database with read consistency
1987 First client-server database
1994 First commercial and multilevel secure database evaluations
1995 First 64-bit mode RDBMS
1996 First to break the 30,000 TPC-C barrier
1997 First Web database
1998 First Database - Native **Java** Support; Breaks 100,000 TPC-C
1998 First Commercial RDBMS ported to **Linux**
2000 First database with **XML**
2001 First RDBMS with **Real Application Clusters** & First middle-tier database cache
2004 First **True Grid Database**
2005 First **FREE Oracle Database** (10g Express Edition)
2006 First **Oracle Support for LINUX Offering**
2007 **Oracle 11g Released!**
2008 Exadata V1 Server Announced (Oracle buys BEA)
2009 Oracle buys Sun – Java; MySQL; Solaris; Hardware; OpenOffice
2010 Oracle announces **MySQL Cluster 7.1, Exadata, Exalogic, America's Cup Win**
2011 X2-2 Exadata, ODA, Exalytics, SuperCluster, Big Data, Cloud, Social Network
2012 X3-2 Exadata, Expanded Cloud Offerings, **Solaris 11.1**
2013 Oracle12c Released! Oracle X3-8 Exadata, Acquisitions (Acme Packet…etc.)!

Testing the Future Version

Version 12.1.0.0.1 of the Database

Oracle Database 12c Release 1: Upgrade Paths

Direct Upgrade Path

Source Database	Target Database
10.2.0.5 (or higher)	12.1.x
11.1.0.7 (or higher)	12.1.x
11.2.0.2 (or higher)	12.1.x

In-Direct Upgrade Path

Source Database	Upgrade Path for Target Database	Target Database
7.3.3.0.0 (or lower)	7.3.4.x --> 9.2.0.8	11.2.x
8.0.5.0.0 (or lower)	8.0.6.x --> 9.2.0.8	11.2.x
8.1.7.0.0 (or lower)	8.1.7.4 --> 9.2.0.8	11.2.x
9.0.1.3.0 (or lower)	9.0.1.4 --> 9.2.0.8	11.2.x

Database Upgrade Assistant (DBUA)

- More automation during the upgrade process
- Additional validation steps (also for on-line)
- Post upgrade more automated as well
- Better status as to specific component success
- Post upgrade fix-it scripts to help automate future needs
- Parallel upgrade takes advantage of multiple CPU cores
- Express Edition Upgrade to others (since 11g)

Database Upgrade Assistant (DBUA)

- DBUA checks before the upgrade:
  - Invalid user accounts or roles
  - Invalid data types or invalid objects
  - De-supported character sets
  - **Adequate resources** (rollback segments, tablespaces, and free disk space)
  - Missing SQL scripts needed for the upgrade
  - Listener running (if Oracle Enterprise Manager Database Control upgrade or configuration is requested)
  - Oracle Database software linked with Database Vault option. If Database Vault is enabled, Disable Database Vault before upgrade.

The New Version – Life is Good!

$ sqlplus ***/***

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Oct 30 11:21:04 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.

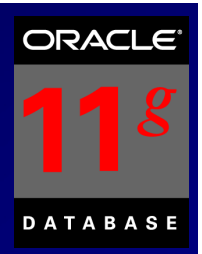
```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.0.1 - 64bit Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> sho sga

Total System Global Area  626327552 bytes
Fixed Size                  2276008 bytes
Variable Size             524289368 bytes
Database Buffers           92274688 bytes
Redo Buffers                7487488 bytes
SQL>
```

MEMORY_TARGET
&
Automatic Memory Management

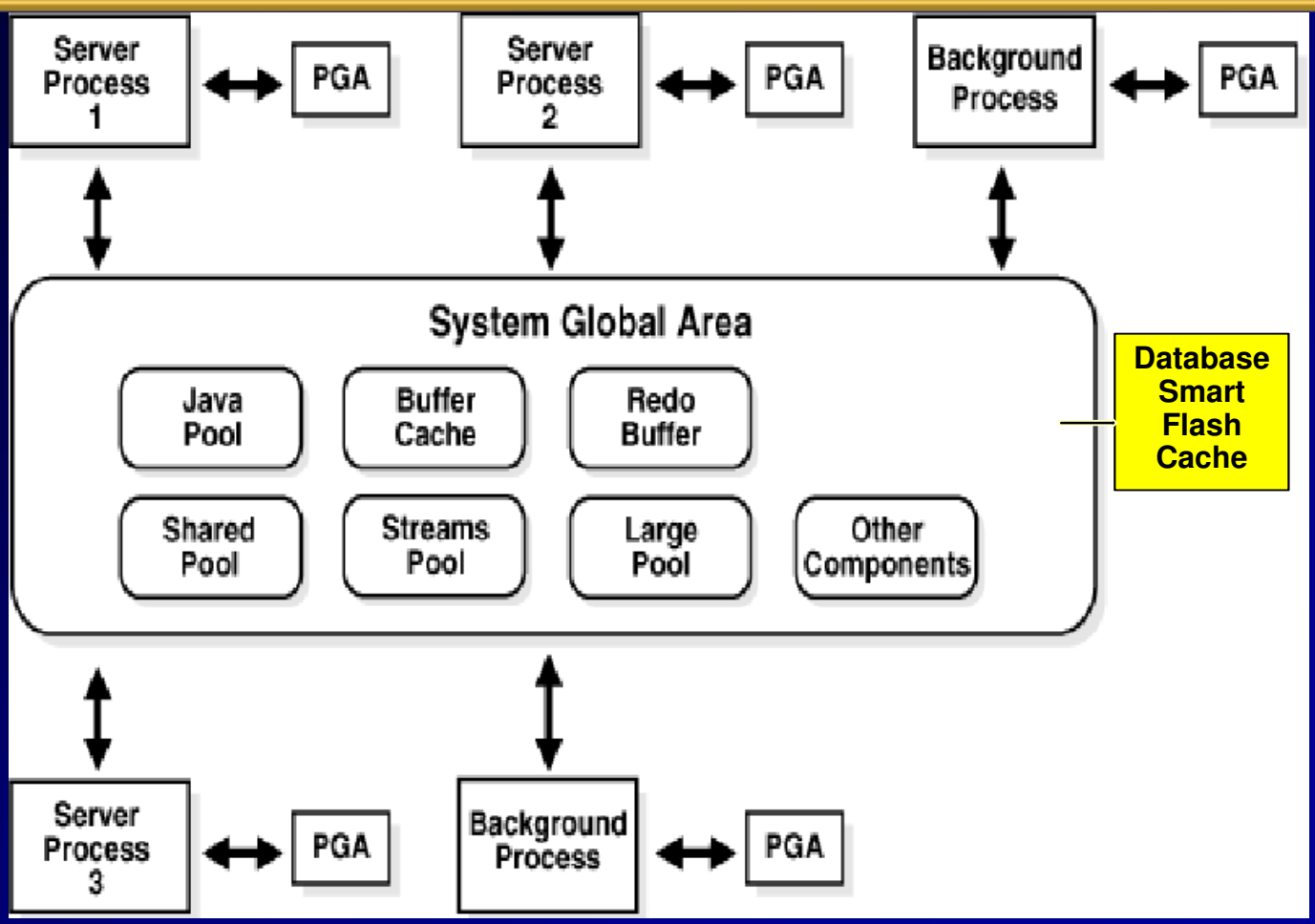
Automatic Memory Management (AMM)
MEMORY_TARGET in 11g



- First there was some Automatic Memory Mgmt - 9i
  - SGA_MAX_SIZE introduced in 9i – Dynamic Memory
  - No more Buffers – DB_CACHE_SIZE
  - Granule sizes introduced - _ksm_granule_size
- Then came SGA_TARGET – 10g
  - Oracle Applications recommends setting this for SGA
  - Set minimums for key values (Data Cache / Shared Pool)
- Now there is MEMORY_TARGET – 11g
  - SGA + PGA all in one setting; Still set minimums

SGA, PGA, MEMORY_TARGET, and Database Smart Flash Cache

(cache multiple devices w/o volume manager)



Database Smart Flash Cache (Solaris/Oracle Linux) – L2 cache set 2-10x SGA:

```
DB_FLASH_CACHE_FILE = /dev/sda, /dev/sdb, /dev/sdc
DB_FLASH_CACHE_SIZE = 32G, 32G, 64G
```

Manually Sized SGA (use SGA_TARGET) PGA_AGGREGATE_LIMIT (New 12c)

Component	Initialization Parameter
Log buffer	LOG_BUFFER (pfile only since 10g)
Keep Pool	DB_KEEP_CACHE_SIZE
Recycle Pool	DB_RECYCLE_CACHE_SIZE
Block caches	DB_nK_CACHE_SIZE

Program Global Area (now in MEMORY_TARGET):

Aggregate PGA PGA_AGGREGATE_TARGET (11g)

New PGA Limit **PGA_AGGREGATE_LIMIT (12c)**

Moving from SGA_TARGET to: MEMORY_TARGET (set minimums)

ALTER SYSTEM SET SGA_TARGET=200M;
ALTER SYSTEM SET PGA_AGGREGATE_TARGET=100M;
ALTER SYSTEM SET PGA_AGGREGATE_LIMIT=140M;

SQL> sho parameter target

NAME	TYPE	VALUE
memory_max_target	big integer	360M
memory_target	big integer	360M
pga_aggregate_target	big integer	100M
sga_target	big integer	200M

The Virtual Column

The Virtual Column

- The value of the virtual column is a derived expression.
  - Can be derived from columns of the same table or from constants
  - Can include SQL or user-defined PL/SQL functions
- Virtual column DATA is NOT PHYSICALLY STORED.
- You CAN NOT explicitly write to a virtual column
- You CAN create a PHYSICAL index (result is function-based index) or partition on a virtual column <unlike a computed column in SQL Server or other databases>
- If you UPDATE columns of a virtual column and it has an index, then it will be computed on the UPDATE vs. on the SELECT (very important from a tuning standpoint).
- Index Organized and External Tables can NOT have virtual columns.

The Virtual Column


```
create table emp_rich
  (empno number(4),
   sal    number(7,2),
   yearly_sal generated always as (sal*12),
   deptno number(2));
```

Table created.

```
insert into emp_rich(empno, sal, deptno)
                select empno, sal, deptno from scott.emp;
```

14 rows created.

The Virtual Column

select * from emp_rich;

EMPNO	SAL	YEARLY_SAL	DEPTNO
7369	800	9600	20
7499	1600	19200	30
7521	1250	15000	30
7566	2975	35700	20
7654	1250	15000	30
7698	2850	34200	30

...

Multiple *Types* of Indexes on the *Same Column*

(Using the Invisible Index even more)

Multiple Types of Indexes on the Same Column(s)

- Create MORE than one index on a column
- Set only ONE index to VISIBLE
- Ok to have ONE + any Function Based Index (exception)
- Great to use different types of indexes for ***batch, query, or data warehousing at different times.***
- Some restrictions apply…for a give column(s)
  - You can not create a B-tree AND B-tree cluster index
  - You can not create a B-tree and an index-organized table (IOT)
- **All indexes ARE MAINTAINED during DML**
  - **DML could be slow if TOO MANY indexes are created**
- Great for ***variable*** workloads!

Multiple Types of Indexes on the Same Column(s)

Basic SELECT to DEPT Table:

```
SELECT * FROM DEPT;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

Multiple Types of Indexes on the Same Column(s)

Create Unique Index… Can NOT insert duplicate!

```
create unique index dept_unique1 on dept(deptno);
Index created.


insert into dept(deptno) values (10);
insert into dept(deptno) values (10)
*
ERROR at line 1:
ORA-00001: unique constraint (SYS.DEPT_UNIQUE1) violated
```

Multiple Types of Indexes on the Same Column(s)

Make Index Invisible… *Still* can NOT insert duplicate!

```
alter index dept_unique1 invisible;
Index altered.


SQL> insert into dept(deptno) values(10);
insert into dept(deptno) values(10)
*
ERROR at line 1:
ORA-00001: unique constraint (SYS.DEPT_UNIQUE1) violated
```

Multiple Types of Indexes on the Same Column(s)

Check the Indexes Views - Index is Invisible

```
select a.table_name, a.index_name,
       b.column_name, a.uniqueness, a.visibility
from   user_indexes a, user_ind_columns b
where  a.index_name = b.index_name
and    a.table_name = 'DEPT';
```

```
TABLE_NAME INDEX_NAME      COLUMN_NAME  UNIQUENESS   VISIBILITY
---------- --------------- ------------ ------------ ----------
DEPT       DEPT_UNIQUE1    DEPTNO       UNIQUE       INVISIBLE
```

Multiple Types of Indexes on the Same Column(s)

Make Index Visible again:

```
alter index dept_unique1 visible;
Index altered.

select a.table_name, a.index_name,
       b.column_name, a.uniqueness, a.visibility
from   user_indexes a, user_ind_columns b
where  a.index_name = b.index_name
and    a.table_name = 'DEPT';

TABLE_NAME INDEX_NAME      COLUMN_NAME  UNIQUENESS   VISIBILITY
---------- --------------- ------------ ------------ ----------
DEPT       DEPT_UNIQUE1    DEPTNO       UNIQUE       VISIBLE
```

Multiple Types of Indexes on the Same Column(s)


```
create index dept_normal on dept(deptno);
create index dept_normal on dept(deptno)
                                 *
ERROR at line 1:
ORA-01408: such column list already indexed
```

Make FIRST Index Invisible & can now create SECOND index:

```
alter index dept_unique1 invisible;
Index altered.

create index dept_normal on dept(deptno);
Index created.
```

Multiple Types of Indexes on the Same Column(s)

Check the Indexes Views – TWO Indexex *on the same column*:

```
select a.table_name, a.index_name,
       b.column_name, a.uniqueness, a.visibility
from   user_indexes a, user_ind_columns b
where  a.index_name = b.index_name
and    a.table_name = 'DEPT';
```

TABLE_NAME	INDEX_NAME	COLUMN_NAME	UNIQUENESS	VISIBILITY
DEPT	**DEPT_UNIQUE1**	DEPTNO	UNIQUE	**INVISIBLE**
DEPT	**DEPT_NORMAL**	DEPTNO	NONUNIQUE	**VISIBLE**

Multiple Types of Indexes on the Same Column(s)

Try to make both Indexex Visible… *ERROR*!

```
alter index dept_unique1 visible;
*
ERROR at line 1:
ORA-14147: There is an existing VISIBLE index defined on
   the same set of columns.
```

Only ONE index may be visible at a time

(except function-based indexes)

Multiple Types of Indexes on the Same Column(s)

Despite a unique index that's invisible, *uses visible index only*:

```
select deptno
from   dept
where  deptno=10;

DEPTNO
------
    10

-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |     1 |    13 |     1   (0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN | DEPT_NORMAL  |     1 |    13 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------

alter index dept_normal invisible;
Index altered.
```

Multiple Types of Indexes on the Same Column(s)

Make FIRST TWO Indexes Invisible & create THIRD index:

```
create index dept_reverse on dept(deptno) reverse;
Index created.

select a.table_name, a.index_name,
       b.column_name, a.uniqueness, a.visibility
from   user_indexes a, user_ind_columns b
where  a.index_name = b.index_name
and    a.table_name = 'DEPT';

TABLE_NAME INDEX_NAME      COLUMN_NAME  UNIQUENESS   VISIBILITY
---------- --------------- ------------ ------------ ----------
DEPT       DEPT_UNIQUE1    DEPTNO       UNIQUE       INVISIBLE
DEPT       DEPT_REVERSE    DEPTNO       NONUNIQUE    VISIBLE
DEPT       DEPT_NORMAL     DEPTNO       NONUNIQUE    INVISIBLE
```

Multiple Types of Indexes on the Same Column(s)

Now I create a Bitmap Index & Function-Based Index:

```
alter index dept_reverse invisible;
Index altered.

create bitmap index dept_bitmap on dept(deptno);
Index created.

create index dept_fb on dept(substr(deptno,1,1));
Index created.
```

OK to Create TWO VISIBLE indexes if one is a Function-Based Index!

Multiple Types of Indexes on the Same Column(s)

Check the Indexes Views – FIVE Indexex *on the same column*:

```
select a.table_name, a.index_name,
       b.column_name, a.uniqueness, a.visibility
from   user_indexes a, user_ind_columns b
where  a.index_name = b.index_name
and    a.table_name = 'DEPT';
```

TABLE_NAME	INDEX_NAME	COLUMN_NAME	UNIQUENESS	VISIBILITY
DEPT	**DEPT_UNIQUE1**	DEPTNO	UNIQUE	**INVISIBLE**
DEPT	**DEPT_REVERSE**	DEPTNO	NONUNIQUE	**INVISIBLE**
DEPT	**DEPT_NORMAL**	DEPTNO	NONUNIQUE	**INVISIBLE**
DEPT	**DEPT_BITMAP**	DEPTNO	NONUNIQUE	**VISIBLE**
DEPT	**DEPT_FB**	SYS_NC00004$	NONUNIQUE	**VISIBLE**

(Index types: NORMAL, NORMAL/REV, UNIQUE, BITMAP, FUNCTION-BASED NORMAL)

Multiple Types of Indexes on the Same Column(s)

Interesting note on Index Suppression:

```
alter index dept_bitmap invisible;
Index altered.

select /*+ index(dept dept_fb) */ deptno
from   dept
where  substr(deptno,1,1)=1;

    DEPTNO
----------
        10
-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |     5 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| DEPT |     1 |     5 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(TO_NUMBER(SUBSTR(TO_CHAR("DEPTNO"),1,1))=1)
```

Multiple Types of Indexes on the Same Column(s)

Interesting note on Index Suppression:

```
select /*+ index(dept dept_fb) */ deptno
from   dept
where  substr(deptno,1,1)='1';

    DEPTNO
----------
        10

-------------------------------------------------------------------------------
| Id  | Operation                           | Name    | Rows  | Bytes | Cost
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |         |     1 |     5 |     2
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DEPT    |     1 |     5 |     2
|*  2 |   INDEX RANGE SCAN                  | DEPT_FB |     1 |       |     1
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access(SUBSTR(TO_CHAR("DEPTNO"),1,1)='1')
```

ROLTA

12c

Adaptive Optimization

Adaptive Query Optimization

- Adaptive query optimization allows optimizer to adjust execution plan at run time when additional/better information is available.
  - **Adaptive Plans**: Different Join Methods (change NL to HASH) or Parallel Distribution
  - **Adaptive Statistics**: Dynamic stats, Auto Reoptimization, and SQL Plan Directives
- Adaptive Plans does not pick the final plan until execution time based on statistics collection. Information learned at execution time is used in future executions. You'll see the plan table output in the note section:

```
Note
-----------------------
- this is an adaptive plan
```

- **The 12c *Adaptive* Optimizer adapts plans based on not just the original tables stats, but also additional adaptive statistics**
- There are three types of Adaptive Statistics:
  - Dynamic Statistics (previously dynamic sampling in 10g/11g) or runtime statistics
  - Automatic Reoptimization or statistics generated after the initial execution
  - SQL Plan Directives direct optimizer to dynamic statistics & gets accurate cardinality

Adaptive Query Optimization

- **The Adaptive Optimizer adapts the execution plan based on stats collected at run time** (a sample of stats – *dynamic statistics*)… it makes runtime optimizations (perhaps changing join method)
- Adaptive Query Optimization is set ON by default. To turn it *OFF* set:

  OPTIMIZER_ADAPTIVE_REPORTING_ONLY = TRUE

  OPTIMIZER_FEATURES_ENABLE=12.1.0.1 (or higher)
- You can also set it for a given session:

```
SQL> alter session set optimizer_adaptive_reporting_only=false;
```

- Set it to TRUE for reporting only. You can then check to get notes about runtime optimizations, such as dynamic plans switching from NL to HASH joins.

  (Use DBMS_XPLAN.DISPLAY_CURSOR … use ADAPTIVE for format…)

```
SQL> sho parameter OPTIMIZER_ADAPTIVE_REPORTING_ONLY

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
optimizer_adaptive_reporting_only    boolean     FALSE
```

Adaptive Query Optimization

- Previously called **dynamic sampling** in 10g/11g, **Dynamic Statistics** was used only in absence of stats on one of the tables in a multi-table join; This is helpful when existing statistics are not sufficient.
- SQL Plan Management (SPM) builds SQL plan baselines to use a verified plan
- *In 12c, Adaptive SPM can be used by using the SPM Evolve Advisor (checks for better plans)*
- Adaptive query optimization uses runtime stats to get an adaptive plan that may be better than the default plan.
- **Automatic Reoptimization** – When actual stats (after query executes) vary greatly compared to the original plan statistics (when the original plan was created), the optimizer records the new statistics (actual vs. estimated) & applies them **next time (see below `Note` from Plan Table)**.

```
Note
-----------------------------------------
- statistics feedback used for this statement
```

Adaptive Query Optimization

- Reoptimization is called performance feedback This is improving SQL statements that are repeated & optimized over time.
- This allows the optimizer to choose PARALLEL and to set a degree. It is set to MANUAL by default. To turn it *on* set:

  PARALLEL_DEGREE_POLICY = ADAPTIVE
- Even if you DON'T set the parameter above, reoptimization based on statistics may influence degree of parallelism that the optimizer uses.
- Reoptimization creates SQL Plan Directives (information/instructions for optimizer to use to generate a more optimal plan *next time*).
  - SQL Plan Directives stored in SYSAUX tablespace & initially created in Shared Pool
  - SQL Plan Directives may be managed using the DBMS_SPD package
  - DBA_SQL_PLAN_DIRECTIVES will show directives/reasons (e.g. MISESTIMATE)

```
Note
-----------------------------------------------------
- dynamic sampling used for this statement (level=2)
- 1 Sql Plan Directive used for this statement
```

Runaway Query Management

Runaway Query Management

- Resource Manager now pro-actively manages problems queries and takes action based on settings for a given consumer group when:
  - CPU is exceeded
  - Physical I/O is exceeded (disk)
  - Logical I/O is exceeded (memory)
  - Elapsed Time is exceeded
- This can be automated!
- New views allow the DBA to see problem queries that are over the limit for each Consumer Group (can be set to automatically be terminated or can be switched to a new group with lower resources)
- Views are persisted in the AWR
- Must have the appropriate resources to manage this
- Can be set based on start of session or start of SQL or PL/SQL:
  - SWITCH_FOR_CALL resource plan directive

Runaway Query Management

(Oracle 12c DBA Guide example…)

Create a Resource plan Directive that kills any session that exceeds 60 seconds of CPU time

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
PLAN => 'DAYTIME',
GROUP_OR_SUBPLAN => 'OLTP',
COMMENT => 'OLTP group',
MGMT_P1 => 75,
SWITCH_GROUP => 'KILL_SESSION',
SWITCH_TIME => 60);
END;
/
```

Create a Resource plan Directive that switches sessions to the low_group if they exceed 10000 physical IO's or 2500M of data transferred. Session returns back to original group after bad query ends

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
PLAN => 'DAYTIME',
GROUP_OR_SUBPLAN => 'OLTP',
COMMENT => 'OLTP group',
MGMT_P1 => 75,
SWITCH_GROUP => 'LOW_GROUP',
SWITCH_IO_REQS => 10000,
SWITCH_IO_MEGABYTES => 2500,
SWITCH_FOR_CALL => TRUE);
END;
/
```

Runaway Query Management

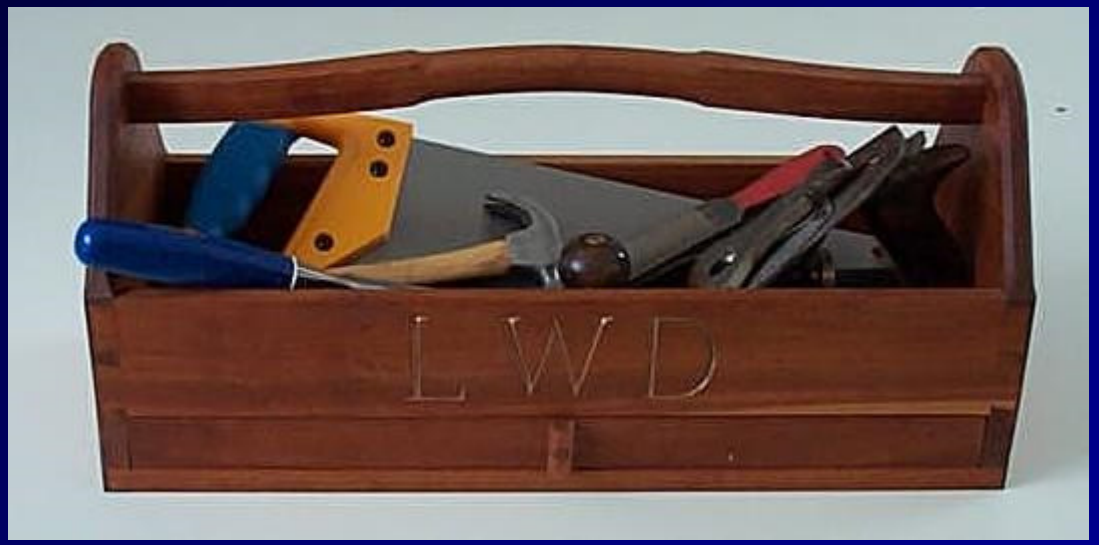
(Oracle 12c DBA Guide example…)

Check the statistics for sessions and consumer groups:

```
SELECT se.sid sess_id, co.name consumer_group, se.state,
       se.consumed_cpu_time cpu_time, se.cpu_wait_time,
       se.queued_time
FROM   v$rsrc_session_info se, v$rsrc_consumer_group co
WHERE  se.current_consumer_group_id = co.id;
```

SESS_ID	CONSUMER_GROUP	STATE	CPU_TIME	CPU_WAIT_TIME	QUEUED_TIME
113	OLTP_ORDER_ENTRY	WAITING	137947	28846	0
135	OTHER_GROUPS	IDLE	785669	11126	0
124	OTHER_GROUPS	WAITING	50401	14326	0
114	SYS_GROUP	RUNNING	495	0	0
102	SYS_GROUP	IDLE	88054	80	0
147	DSS_QUERIES	WAITING	460910	512154	0

Nice DBA Tools



Change Table Compression at Import Time
&
Data Pump Export View as a Table

(Also: No redo logging option of table load/Index creation)

Change Table Compression at Import Time

- Use impdp command line option (or use DBMS_DATAPUMP)
- Use the TABLE_COMPRESSION_CLAUSE:

```
TABLE_COMPRESSION_CLAUSE=NONE
TABLE_COMRPESSION_CLAUSE=NOCOMPRESS
TABLE_COMPRESSION_CLAUSE=COMPRESS BASIC
TABLE_COMPRESSION_CLAUSE=COMPRESS ROW STORE COMPRESS ADVANCED  (used for OLTP)

Warehouse compression (low is faster load):
TABLE_COMPRESSION_CLAUSE=COMPRESS COLUMN STORE COMPRESS FOR QUERY LOW
TABLE_COMPRESSION_CLAUSE=COMPRESS COLUMN STORE COMPRESS FOR QUERY HIGH
Archive compression (low is faster load):
TABLE_COMPRESSION_CLAUSE=COMPRESS COLUMN STORE COMPRESS FOR ARCHIVE LOW
TABLE_COMPRESSION_CLAUSE=COMPRESS COLUMN STORE COMPRESS FOR ARCHIVE HIGH

impdp hr TABLES=hr.employees DIRECTORY-dpump_dir1 DUMPFILE=hr.dmp
   TRANSFORM=TABLE_COMPRESSION_CLAUSE=NOCOMPRESS
```

- This is especially helpful for Exadata migrations where more compression options (HCC) are available.

Change Table Compression at Import & Create Views as Tables **Examples**

A Basic Example changing a table to COMPRESS:

```
$ impdp scott2/tiger TABLES=dept2
    TRANSFORM=TABLE_COMPRESSION_CLAUSE:compress:table

Master table "SCOTT2"."SYS_IMPORT_TABLE_01" successfully loaded/unloaded
Starting "SCOTT2"."SYS_IMPORT_TABLE_01":  scott2/******** TABLES=dept2
    TRANSFORM=TABLE_COMPRESSION_CLAUSE:compress:table
Processing object type TABLE_EXPORT/TABLE/TABLE
Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
. . imported "SCOTT2"."DEPT2"                    5.937 KB       4 rows
Processing object type TABLE_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Job "SCOTT2"."SYS_IMPORT_TABLE_01" successfully completed at Sat Mar 2 03:59:51 2013
      elapsed 0 00:00:12
```

A Basic Example creating views as tables:

```
$ impdp scott2/tiger VIEWS_AS_TABLES…
    VIEWS_AS_TABLES=schema.view_name:table, …
```

Create Views as Tables Example

Export a view as a table and then import it:

```
create view  emp_dept as
(select a.empno, a.ename, b.deptno, b.dname, b.loc
 from emp a, dept b
 where a.deptno=b.deptno);
View created.

$ expdp scott2/tiger VIEWS_AS_TABLES=emp_dept

Processing object type TABLE_EXPORT/VIEWS_AS_TABLES/TABLE
. . exported "SCOTT2"."EMP_DEPT"
   7.140 KB      14 rows
```

Create Views as Tables Example


```
SQL> rename emp_dept to emp_dept_view;
$ impdp scott2/tiger VIEWS_AS_TABLES=emp_dept

Processing object type
   TABLE_EXPORT/VIEWS_AS_TABLES/TABLE_DATA
. . imported "SCOTT2"."EMP_DEPT"
   7.140 KB       14 rows

select segment_name, segment_type
from   dba_segments
where  segment_name = 'EMP_DEPT';

SEGMENT_NAME          SEGMENT_TYPE
--------------------- ------------------
EMP_DEPT              TABLE
```

Compression History – Timeline (FYI Only)

- Index Compression since 8i
- Table Compression since 9i
  - No Additional License Requirement
  - Only for direct inserts
  - Compression Not Maintained with updates and normal inserts
  - Had to re-org table to re-compress over time.
- 11g Advanced Compression
  - Additional License Requirement
  - Compression Maintained with all DML activity
  - No re-orgs required after initial compression
- 11gR2 – Hybrid Columnar Compression (with Exadata)
- **12c – Change Table Compression at Import Time**

Partitioning: (FYI Only)

- Online Move Partition – 12c
- Partial Indexes for Partitioned Table – 12C
- WHAT ELSE IS NEW IN ORACLE 12c
  - Partition Maintenance Operation on Multiple Partitions (12c fyi)
  - Interval Reference Partitioning (12c fyi) (use in parent/passes to child)

Interval Partitioning – 11g

- This is a helpful addition to range partitioning where Oracle automatically creates a partition when the inserted value exceeds all other partition ranges. **11g also has Ref & Virtual Column Partitioning & Oracle 12c has Interval Ref Partitioning. (not covered here).**

There are the following restrictions:

- You can only specify one partitioning key column, and it must be of NUMBER or DATE type.
- Interval partitioning is NOT supported for index-organized tables.
- Interval Partitioning supports composite partitioning:
  - Interval-range *** Interval-hash *** Interval-list
- You can NOT create a domain index on an interval-partitioned table.

Partial Indexes for Partitioned Table – NO NO NO…

Create an index on a subset of the partitions of a table:

```
Create index dept_index on dept3 (deptno) local
  (partition d1 tablespace users,
   partition d2 tablespace users);
create index dept_index on dept3 (deptno) local;
                          *
ERROR at line 1:
ORA-14024: number of partitions of LOCAL index must equal that of the
underlying table


create partial index dept_index on dept3 (deptno) local
  (partition d1 tablespace users,
   partition d2 tablespace users);
create partial index dept_index on dept3 (deptno) local
       *
ERROR at line 1:
ORA-00901: invalid CREATE command
```

Partial Indexes for Partitioned Table … YES!


```
CREATE TABLE DEPT3
(DEPTNO NUMBER(2), DEPT_NAME VARCHAR2(30))
 INDEXING OFF
 PARTITION BY RANGE(DEPTNO)
 (PARTITION D1 VALUES     LESS THAN (10) indexing on,
  PARTITION D2 VALUES     LESS THAN (20) indexing on,
  PARTITION D3 VALUES     LESS THAN (MAXVALUE));

Table created.

SQL> create index dept3_partial on dept3 (dept_name)
       local indexing partial;

Index created.
```

(Local Index Partitions D1 & D2 will be usable – can create global index instead)

Online Move Partition

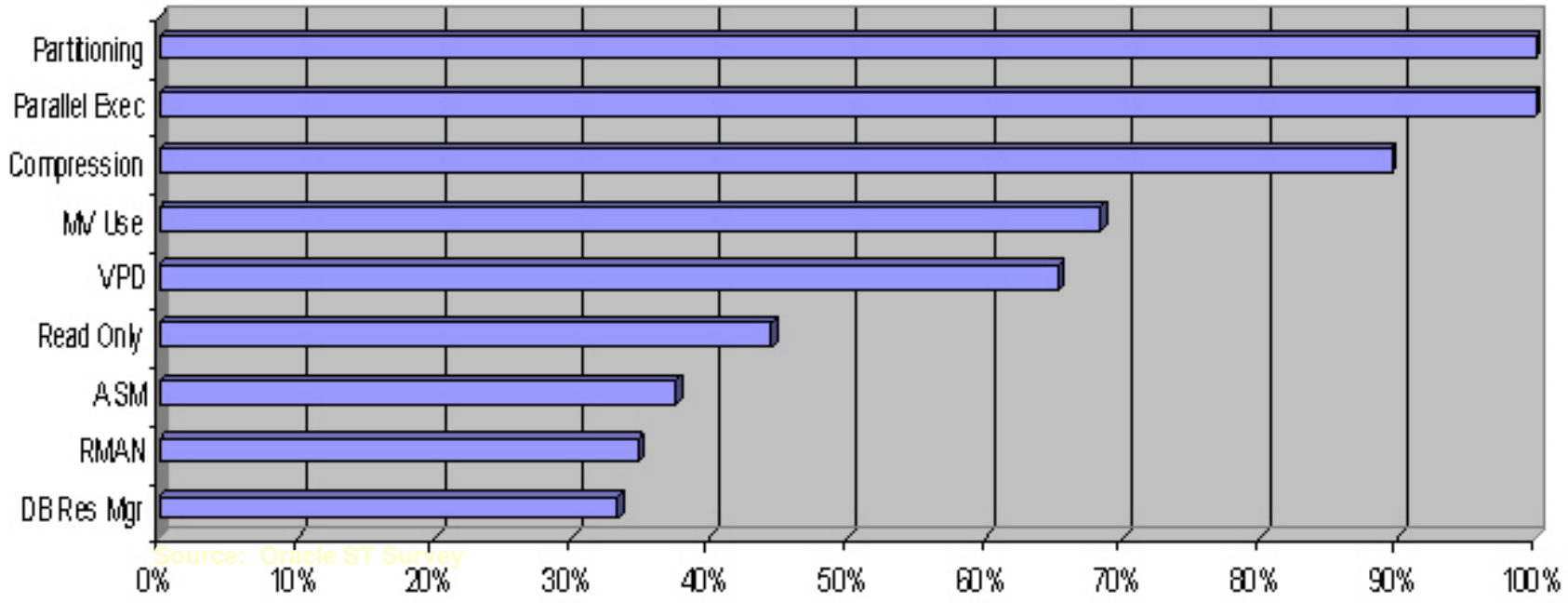
- You can now move partitions *real time*:
  - ALTER TABLE MOVE PARTITION…
- Now a non-blocking DDL!
- DML on the partition continue to run before/during/after the move!
- Global indexes are maintained as well.

```
alter table dept3 move partition d1 tablespace users;
alter table dept3
            *
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified or timeout
   expired


alter table dept3 move partition d1 tablespace sysaux online;
Table altered.
```

Large-Scale Data Warehouses*

Feature Usage



* *Oracle Survey*

Pluggable Databases
(Plug into the Power of the Database!)

Thanks: Penny Avril & Bryn Llewellyn

ORA-65052: statement involves operations with *different container scope*

ORA-65040: operation *not allowed* from within a pluggable database

ORA-65017: *seed pluggable database* may not be dropped or altered

Start with a Pristine Oracle System and Brand New Oracle Database

Non-CDB

Install New DB

Add User Data

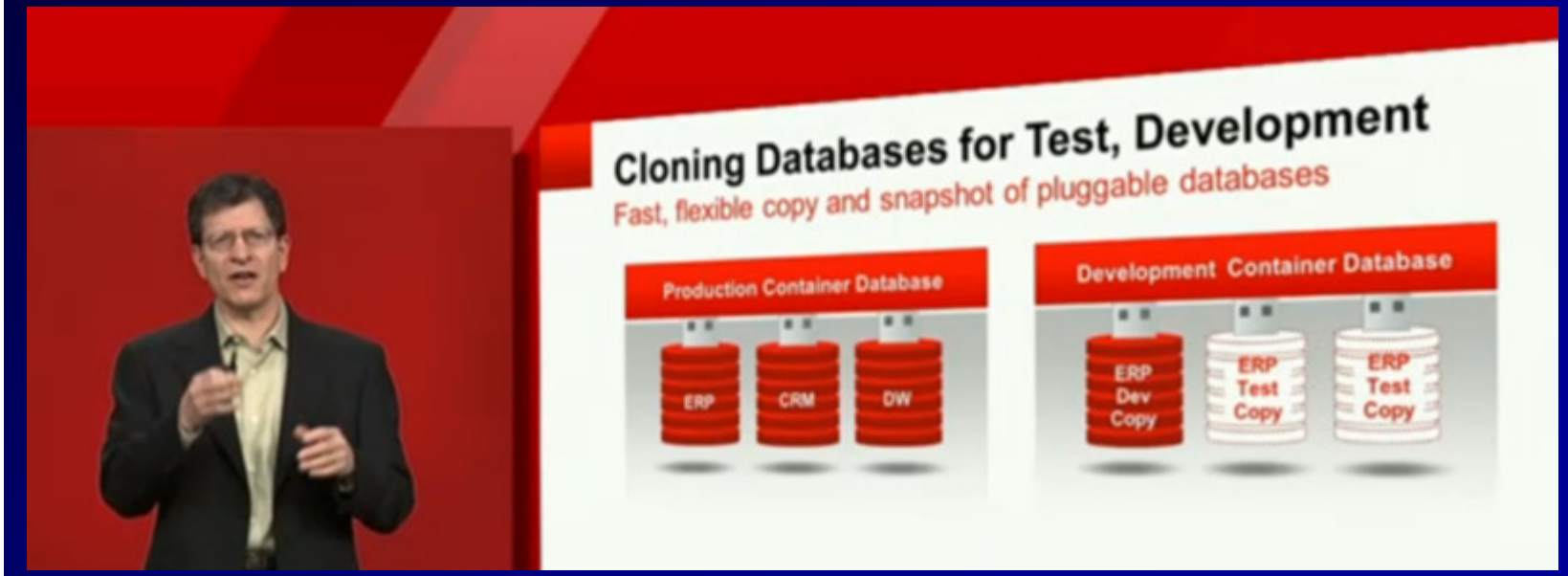
More Data

Pristine DB

Separate PDB

Keep Pristine DB Separated

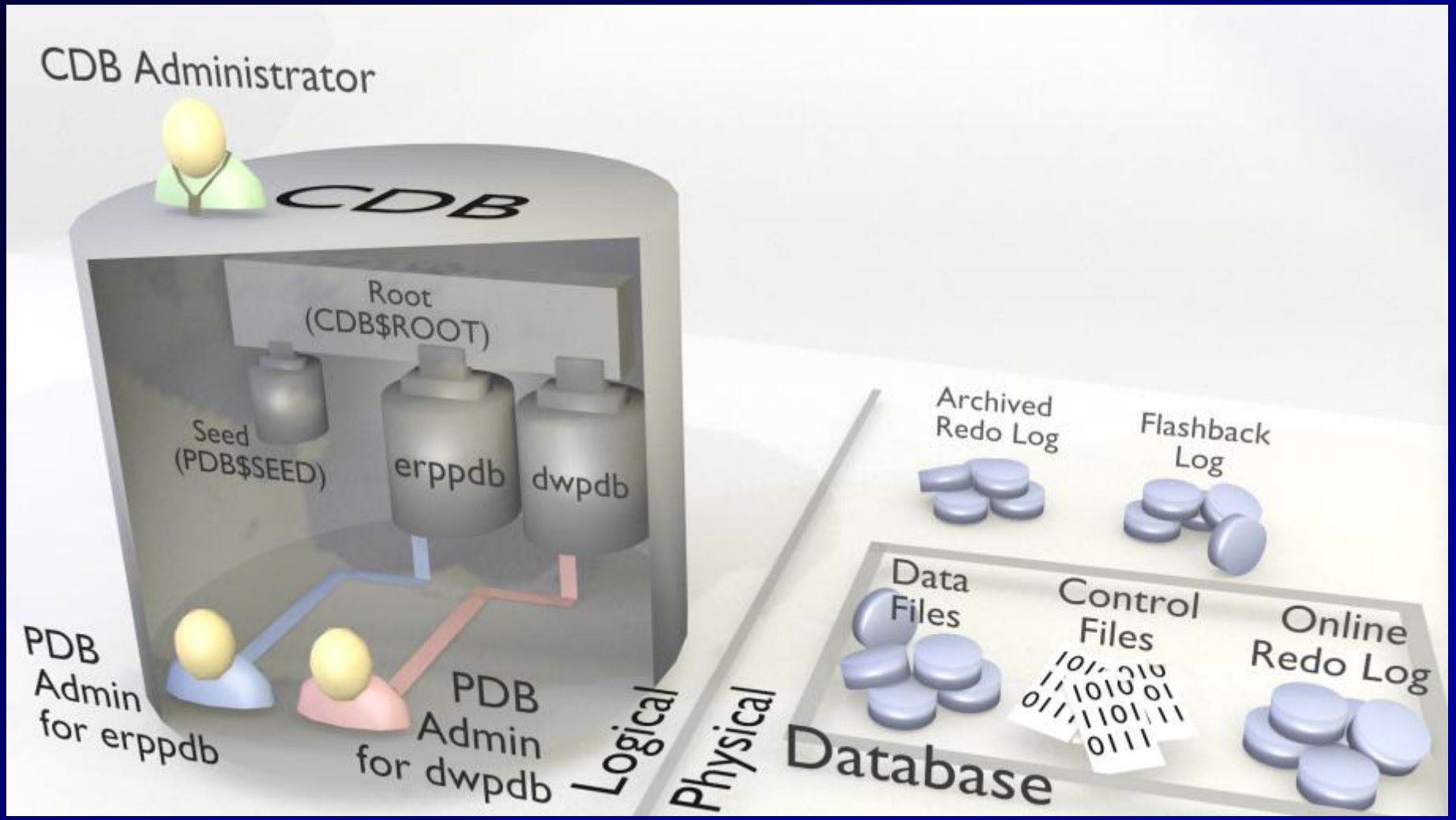
Pluggable Databases are Coming!



Pluggable Databases

- CDB = Container Database (has Root DB & also has a seed PDB)
- PDB = Pluggable Database (plugged into a CDB)
- Non-CDB = Original type of Database (neither a CDB or PDB)
- Why?: Can't consolidate 100's of database on one machine … too many resources required when you add the SGAs up! Enter PDBs.
  - Share: Big Data Sources, Acquisitions, Partners, Shared Research, Governments
- Quickly create a new database (PDB) or copy existing one (PDB)
- Move existing PDBs to new platform or location or clone it (snapshot)
- Patch/Upgrade PDB by plugging it into a CDB at a later version
- Physical machine runs more PDBs old way: Easier to manage/tune
- Backup entire CDB + any number of PDBs
- New syntax for commands: PLUGGABLE DATABASE

Pluggable Databases…



Is the database a CDB or non-CDB?

```
SQL> SELECT NAME, CREATED, CDB, CON_ID
  2  FROM V$DATABASE;

NAME      CREATED   CDB      CON_ID
--------- --------- --- ----------
CDB1      19-FEB-12 YES         0
```

Pluggable Databases

- In a CDB: Only one CDB$ROOT (Root), only one PDB$SEED (Seed), plus any PDBs (up to 252 more – 253 including the seed) that you create or plug in.
- CDB Root has schemas, schema objects, data dictionary information about PDBs
- Seed database – Can't add any objects – only to create new PDBs (clone it to create others)
- PDB – appears to users/applications as if it was a non-CDB. Accessing a PDB is like accessing a non-CDB
- PDBs are how you split applications physically

Containers 0 - 254

- Entire CDB => Container ID = 0
- Root (CDB$ROOT) => Container ID = 1
- Seed (PDB$SEED) => Container ID = 2
- PDBs => Container ID = 3 to 254

(While in PDB1):

```
SQL> SHO CON_ID CON_NAME

CON_ID
------------------------------
3

CON_NAME
------------------------------
PDB1
```

(Connect to ROOT):

```
SQL> connect / as sysdba
SQL> SHO CON_ID CON_NAME
CON_ID
------------------------------
1

CON_NAME
------------------------------
CDB$ROOT
```

(integer overflow!)

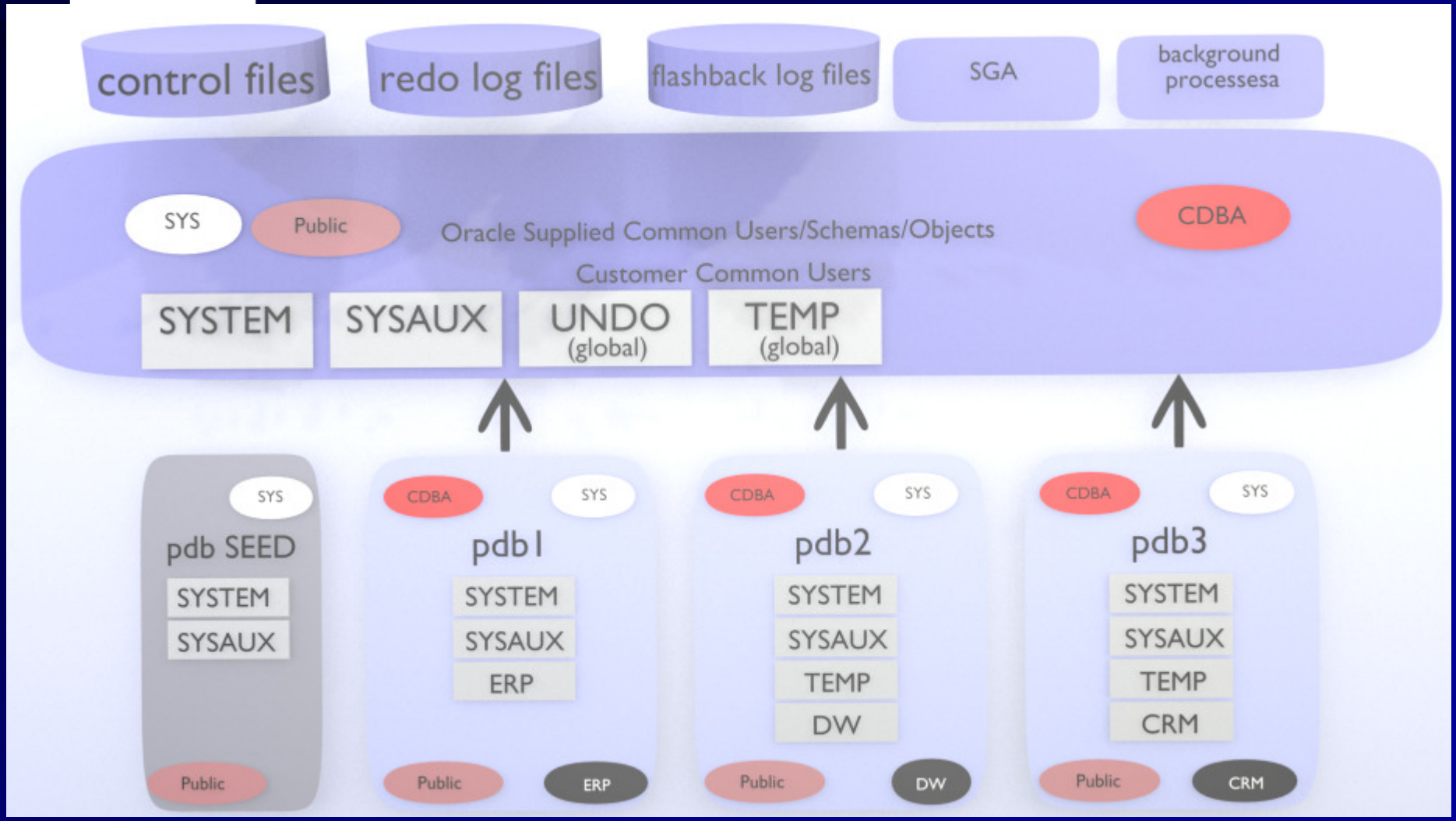
CDB or PDB created…

- Background Processes /SGA (shared by root & all PDBs)
- Character Set shared by root & all PDB's
- Redo shared by root and all PDB's
- Undo shared by root and all PDB's

- Temporary Tablespace – *can* create for each PDB
- Time Zones – *can* be set for each PDB
- Initialization parameters – *some can* be set by PDB

- *Separate* SYSTEM & SYSAUX for root & each PDB
- Data files *separate* for root & each PDB (same block size)

Pluggable Databases…



Query the PDBs


```
select name, open_mode, open_time
from v$pdbs;
```

```
NAME             OPEN_MODE  OPEN_TIME
---------------- ---------- ------------------------------
PDB$SEED         READ ONLY  23-FEB-13 05.29.19.861 AM
PDB1             READ WRITE 23-FEB-13 05.29.25.846 AM
PDB_SS           READ WRITE 23-FEB-13 05.29.37.587 AM
```

Pluggable Databases

- PDB is backward compatible with pre-12c database.
- **Common users** like SYS, SYSTEM connect to the CDB and also across all PDBS in which it has privileges (you can create your own *common users* as well). Common users create/plug/unplug PDBs.
- Privileged common users can even switch CDBs
- **Local users** are only in a SINGLE PDB (dwadm, erpadm …etc.)
- Listener, Service Name, ..etc. needed
- One CDB has same software version, Active Data Guard, RMAN Backups, Initialization parameters related to database level (character set for instance)
- Plug or unplug a PDB into a CDB. Plug it in to associate it with the CDB, consisting of the XML file describing the PDB and its files (database files and wallet file)

From the Oracle docs … create DB

CDB must have "*enable pluggable database*"

CREATE DATABASE newcdb

USER SYS IDENTIFIED BY sys_password USER SYSTEM IDENTIFIED BY system_password

LOGFILE GROUP 1 ('/u01/logs/my/redo01a.log','/u02/logs/my/redo01b.log') SIZE 100M BLOCKSIZE 512,

GROUP 2 ('/u01/logs/my/redo02a.log','/u02/logs/my/redo02b.log') SIZE 100M BLOCKSIZE 512,

GROUP 3 ('/u01/logs/my/redo03a.log','/u02/logs/my/redo03b.log') SIZE 100M BLOCKSIZE 512 MAXLOGHISTORY 1 MAXLOGFILES 16 MAXLOGMEMBERS 3 MAXDATAFILES 1024 CHARACTER SET AL32UTF8 NATIONAL CHARACTER SET AL16UTF16

EXTENT MANAGEMENT LOCAL

DATAFILE '/u01/app/oracle/oradata/newcdb/system01.dbf' SIZE 700M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED

SYSAUX DATAFILE '/u01/app/oracle/oradata/newcdb/sysaux01.dbf' SIZE 550M REUSE AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED

DEFAULT TABLESPACE deftbs DATAFILE '/u01/app/oracle/oradata/newcdb/deftbs01.dbf' SIZE 500M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED

DEFAULT TEMPORARY TABLESPACE tempts1 TEMPFILE '/u01/app/oracle/oradata/newcdb/temp01.dbf' SIZE 20M REUSE AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED

UNDO TABLESPACE undotbs1 DATAFILE '/u01/app/oracle/oradata/newcdb/undotbs01.dbf' SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED

ENABLE PLUGGABLE DATABASE

SEED FILE_NAME_CONVERT = ('/u01/app/oracle/oradata/newcdb/', '/u01/app/oracle/oradata/pdbseed/')

SYSTEM DATAFILES SIZE 125M AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED

SYSAUX DATAFILES SIZE 100M

USER_DATA TABLESPACE usertbs DATAFILE '/u01/app/oracle/oradata/pdbseed/usertbs01.dbf' SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;

Creating a PDB
Many ways to do it…

- Create a PDB by copying the seed PDB
- Create a PDB by cloning another PDB
- Create a PDB by using the XML metadata files and other files and plugging them into a CDB
- Create a PDB using a non-CDB (multiple ways)
  - Use DBMS_PDB to create an unplugged PDB
  - Create an empty PDB and use data pump to move data
  - Using GoldenGate replication to create

Create a PDB – fyi only…
(Parameters to possibly set)

- PATH_PREFIX = '/disk1/oracle/dbs/dwpdb/'
- This PATH_PREFIX clause restricts the location of files and directory object paths associated with a PDB to the /disk1/oracle/dbs/dwpdb directory.
- FILE_NAME_CONVERT = ('/oracle/dbs/', '/oracle/dwpdb/')

  This FILE_NAME_CONVERT clause generates file names for the new PDB in the /oracle/dwpdb directory using file names in the /oracle/dbs directory. This is when you want to move file location upon creation.
- SOURCE_FILE_NAME_CONVERT = ('/disk1/oracle/pdb1/', '/disk2/oracle/pdb1/'). This SOURCE_FILE_NAME_CONVERT clause uses the files in the /disk2/oracle/pdb1 directory instead of the /disk1/oracle/pdb1 directory. In this case, the XML file describing a PDB specifies the /disk1/oracle/pdb1 directory, but the PDB should use the files in the /disk2/oracle/pdb1 directory. *NONE* if location is correct.

Create a PDB - fyi only…

CREATE PLUGGABLE DATABASE dwpdb ADMIN USER dwadm IDENTIFIED BY *password*;

CREATE PLUGGABLE DATABASE dwpdb ADMIN USER dwadm IDENTIFIED BY *password* ROLES=(SELECT_CATALOG_ROLE, GATHER_SYSTEM_STATISTICS);

(PDB_DBA role is also granted in addition to the above specifically granted roles.)

CREATE PLUGGABLE DATABASE dwpdb ADMIN USER dwadm IDENTIFIED BY *password* STORAGE (MAXSIZE 10G MAX_SHARED_TEMP_SIZE 100M) DEFAULT TABLESPACE dw DATAFILE '/disk1/oracle/dbs/dwpdb/dw1.dbf' SIZE 2G AUTOEXTEND ON PATH_PREFIX = '/disk1/oracle/dbs/dwpdb/' FILE_NAME_CONVERT = ('/disk1/oracle/dbs/pdbseed/', '/disk1/oracle/dbs/dwpdb/');

Cloning a PDB


```
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1;
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1
  PATH_PREFIX = '/disk2/oracle/pdb2'
  FILE_NAME_CONVERT = ('/disk1/oracle/pdb1/',
  '/disk2/oracle/pdb2/');
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1
  FILE_NAME_CONVERT = ('/disk1/oracle/pdb1/',
  '/disk2/oracle/pdb2/')  STORAGE (MAXSIZE 2G
  MAX_SHARED_TEMP_SIZE 100M);
CREATE PLUGGABLE DATABASE pdb2 FROM
  pdb1@pdb1_link;
```

Create PDB from non-CDB (3 ways) (fyi only…)

- Use Oracle Data Pump with or without transportable tablespaces (11.2.0.3 – full transportable export). Create an empty PDB and then import into it.
- Use Oracle GoldenGate replication … replicate from non-CDB to PDB & fail over when replication catches up with non-CDB.
- Execute DBMS_PDB.DESCRIBE on a non-CDB in Oracle Database 12c Release 1 (12.1)… creates the .XML Metadata file. You can then use this with the database files to create a PDB (see next slide).

Use DBMS_PDB to create PDB from non-CDB *(fyi only)*

- Ensure non-CDB is in a transactionally-consistent state and place it in read-only mode.
- Generate an XML file (ncdb.xml) in /disk1/oracle directory:

```
BEGIN
  DBMS_PDB.DESCRIBE(
       pdb_descr_file => '/disk1/oracle/ncdb.xml');
END;
/
```

- Shutdown the non-CDB.
- Plug in the non-CDB, Access the PDB.
- Run the noncdb_to_pdb.sql script:
  - @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql
- Open the new PDB in read/write mode & Back up the PDB.

Unplug/Plug-in a 12.1 PDB …

CDB with 2 PDBs

Unplug PDB

CDB with 1 PDB

Plug into a Different CDB (12.2):

Plug in an Unplugged PDB


```
CREATE PLUGGABLE DATABASE dwpdb USING
  '/disk1/usr/dwpdb.xml' NOCOPY TEMPFILE REUSE;
CREATE PLUGGABLE DATABASE dwpdb USING
  '/disk1/usr/dwpdb.xml'
  SOURCE_FILE_NAME_CONVERT =
  ('/disk1/oracle/dw/', '/disk2/oracle/dw/')  NOCOPY
  STORAGE (MAXSIZE 4G
  MAX_SHARED_TEMP_SIZE 100M)  TEMPFILE
  REUSE;
```

Unplugging & Dropping PDBs


```
ALTER PLUGGABLE DATABASE dwpdb
  UNPLUG INTO '/oracle/data/dwpdb.xml';

DROP PLUGGABLE DATABASE dwpdb KEEP
  DATAFILES;

DROP PLUGGABLE DATABASE dwpdb
  INCLUDING DATAFILES;
```

Create and Manage with CC (Use 12c Cloud Control – OEM)

- Go to the CDB target & manage storage & objects.
- Under Provisioning & Patching – Provision (create or clone) or Unplug Pluggable databases
- You can create multiple PDBs at once
- View Job Details under Procedure Activity
- Go to CDB target (as a common user) and then look at PDB level … you can switch containers and refresh to look at specific PDB information.

Connecting to CDB/PDB
Using sqlplus…

- To connect to the root … must be a common user & must have create session privilege in the root.
- To connect to a PBD, must either be common user with local create session or local PDB user with create session.
- Use SQLPLUS /nolog … and then CONNECT
- Connect / as sysdba (to root, just as a non-CDB)

Moving between CDB/PDBs Switch Containers…


```
SQL> ALTER SESSION SET CONTAINER=PDB1;
Session altered.

SQL> alter session set container=CDB1;
ERROR:
ORA-65011: Pluggable database does not exist

ALTER SESSION SET CONTAINER=CDB$ROOT;
Session altered.

ALTER SESSION SET CONTAINER=PDB$SEED;
Session altered.

ALTER SESSION SET CONTAINER=pdb_ss; (not case sensitive)
Session altered.
```

DBA_CONTAINER_DATA

```
SQL> desc dba_container_data
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 USERNAME                                           VARCHAR2(128)
 DEFAULT_ATTR                                       CHAR(1)
 OWNER                                              VARCHAR2(128)
 OBJECT_NAME                                        VARCHAR2(128)
 ALL_CONTAINERS                                     VARCHAR2(1)
 CONTAINER_NAME                                     VARCHAR2(128)

SELECT * FROM DBA_CONTAINER_DATA;

USERNAME   D OWNER      OBJECT_NAM A CONTAINER_
---------- - ---------- ---------- - ----------
SYSTEM     Y                     Y
DBSNMP     Y                     Y
SYS        Y                     Y
SYSBACKUP  Y                     Y
```

Open/Close PDBs


```
SQL> ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN READ WRITE;
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE CLOSE; (shutdown)
Pluggable database altered.

Alter pluggable database open upgrade; (to migrate)
Pluggable database altered.
```

Open/Close PDBs


```
ALTER PLUGGABLE DATABASE PDB_SS, PDB1 CLOSE; (not in CDB)
ALTER PLUGGABLE DATABASE PDB_SS, PDB1 CLOSE
*
ERROR at line 1:
ORA-65040: operation not allowed from within a pluggable database

alter session set container=CDB$ROOT;
Session altered.

alter pluggable database ALL open read only; (from CDB)
Pluggable database altered.

ALTER PLUGGABLE DATABASE PDB_SS, PDB1 CLOSE;
Pluggable database altered.
```

Check PDB status…


```
select name, open_mode,  open_time
from   v$pdbs;

NAME       OPEN_MODE  OPEN_TIME
---------- ---------- ---------------------------
PDB$SEED   READ ONLY  11-MAR-13 09.29.18.284 PM
PDB1       MOUNTED    27-MAR-13 01.19.02.666 AM
PDB_SS     MOUNTED    27-MAR-13 01.19.02.985 AM
```

Open/Close PDBs


```
ALTER PLUGGABLE DATABASE PDB_SS, PDB1 open;
Pluggable database altered.

select name, open_mode,  open_time
from   v$pdbs;

NAME       OPEN_MODE  OPEN_TIME
---------- ---------- ---------------------------
PDB$SEED   READ ONLY  11-MAR-13 09.29.18.284 PM
PDB1       READ WRITE 27-MAR-13 01.26.32.905 AM
PDB_SS     READ WRITE 27-MAR-13 01.26.36.559 AM
```

Open/Close PDBs


```
alter pluggable database all except pdb1 close immediate;
Pluggable database altered.

select name, open_mode,  open_time
from   v$pdbs;

NAME       OPEN_MODE  OPEN_TIME
---------- ---------- ---------------------------
PDB$SEED   READ ONLY  11-MAR-13 09.29.18.284 PM
PDB1       READ WRITE 27-MAR-13 01.26.32.905 AM
PDB_SS     MOUNTED    27-MAR-13 01.29.47.225 AM

alter pluggable database pdb$seed close immediate;
alter pluggable database pdb$seed close immediate
                         *
ERROR at line 1:
ORA-65017: seed pluggable database may not be dropped or altered
```

Startup PDB


```
Startup pluggable database pdb1 open;(read/write)
Pluggable Database opened.
(or while in pdb1 just run STARTUP)

Startup pluggable database pdb1 open read only;
Pluggable Database opened.

Startup pluggable database pdb1 force; (closes/opens)
Pluggable Database opened.
(or while in pdb1 just run STARTUP FORCE)
```

Careful – New commands!


```
SQL> SHUTDOWN PLUGGABLE DATABASE PDB1;
SP2-0717: illegal SHUTDOWN option

SQL> STARTUP
Pluggable Database opened.

SQL> SHUTDOWN (also SHUTDOWN ABORT works)
ORACLE instance shut down.

select name, open_mode,  open_time
from   v$pdbs;

NAME       OPEN_MODE  OPEN_TIME
---------- ---------- --------------------------
PDB1       MOUNTED    27-MAR-13 01.50.25.345 AM
```

Query CDB before PDB1 startup…


```
SQL> connect / as sysdba
Connected.

select name, open_mode,  open_time
from   v$pdbs;

NAME         OPEN_MODE  OPEN_TIME
------------ ---------- ----------------------------
PDB$SEED     READ ONLY  11-MAR-13 09.29.18.284 PM
PDB1         MOUNTED    27-MAR-13 02.00.06.536 AM
PDB_SS       READ WRITE 27-MAR-13 01.41.58.049 AM
```

When you startup the CDB…


```
SQL> startup
ORACLE instance started.

Total System Global Area  626327552 bytes
Fixed Size                  2276008 bytes
Variable Size             524289368 bytes
Database Buffers           92274688 bytes
Redo Buffers                7487488 bytes
Database mounted.
Database opened.
```

```
select name, open_mode,  open_time
from   v$pdbs;
```

```
NAME        OPEN_MODE  OPEN_TIME
----------  ---------- ---------------------------
PDB$SEED    READ ONLY  27-MAR-13 02.04.46.883 AM
PDB1        MOUNTED
PDB_SS      MOUNTED
```

RMAN & other Nice Commands

alter pluggable database all open;

(great command!)

RMAN> alter pluggable database pdb1 close;

RMAN> restore pluggable database pdb1;

RMAN> recover pluggable database pdb1 until SCN 777070;

RMAN> alter pluggable database pdb1 open resetlogs;

srvctl add service … –pdb <pdb_name>

V$ Views…

- New views start with CDB_ are CDB only
- Dictionary/Performance views (V$) – show only PDB when queried from that PDB (isolation)
- Query performance views from root & will show all PDB's
- PDB's have container identifier – when you look at root… see all of the id's
- V$SESSION & V$INSTANCE have a CON_ID column for containers (& new V$IO_OUTLIER)

Where is Everything?

```
SELECT d.con_ID, p.PDB_NAME, d.FILE_ID, d.TABLESPACE_NAME,
       d.FILE_NAME
FROM   CDB_PDBS p, CDB_DATA_FILES d
WHERE  p.PDB_ID(+) = d.CON_ID
order  by d.con_id;
```

CON_ID	PDB	FILE_ID	TABLESPACE_NAME	FILE_NAME
1		6	USERS	/u01/app/oracle/oradata/cdb1/users01.dbf
1		4	UNDOTBS1	/u01/app/oracle/oradata/cdb1/undotbs01.dbf
1		3	SYSAUX	/u01/app/oracle/oradata/cdb1/sysaux01.dbf
1		1	SYSTEM	/u01/app/oracle/oradata/cdb1/system01.dbf
2	PDB$SEED	2	SYSTEM	/u01/app/oracle/oradata/cdb1/pdbseed/system01.dbf
2	PDB$SEED	5	SYSAUX	/u01/app/oracle/oradata/cdb1/pdbseed/sysaux01.dbf
3	PDB1	7	SYSTEM	/u01/app/oracle/oradata/cdb1/pdb1/system01.dbf
3	PDB1	8	SYSAUX	/u01/app/oracle/oradata/cdb1/pdb1/sysaux01.dbf
4	PDB_SS	9	SYSTEM	/u01/app/oracle/oradata/cdb1/pdb_ss/system01.dbf
4	PDB_SS	10	SYSAUX	/u01/app/oracle/oradata/cdb1/pdb_ss/sysaux01.dbf
4	PDB_SS	11	EXAMPLE	/u01/app/oracle/oradata/cdb1/pdb_ss/example.dbf

11 rows selected.

Map tables to PDBs…


```
SELECT p.PDB_ID, p.PDB_NAME, t.OWNER, t.TABLE_NAME
FROM   CDB_PDBS p, CDB_TABLES t
where  p.PDB_ID = t.CON_ID
AND    T.OWNER ='ORDDATA'
ORDER  BY t.TABLE_NAME;
```

```
    PDB_ID PDB_NAME   OWNER      TABLE_NAME
---------- ---------- ---------- ------------------------------
         2 PDB$SEED   ORDDATA    ORDDCM_ANON_ACTION_TYPES
         3 PDB1       ORDDATA    ORDDCM_ANON_ACTION_TYPES
         2 PDB$SEED   ORDDATA    ORDDCM_ANON_ATTRS
         3 PDB1       ORDDATA    ORDDCM_ANON_ATTRS
         3 PDB1       ORDDATA    ORDDCM_ANON_ATTRS_TMP
         2 PDB$SEED   ORDDATA    ORDDCM_ANON_ATTRS_TMP
         3 PDB1       ORDDATA    ORDDCM_ANON_ATTRS_WRK
         2 PDB$SEED   ORDDATA    ORDDCM_ANON_ATTRS_WRK
         …
```

Sanity check -what do I have …


```
select name, service_id, con_name, con_id
from   v$active_services
order  by 1;

NAME             SERVICE_ID CON_NAME            CON_ID
---------------- ---------- --------------- ----------
SYS$BACKGROUND            1 CDB$ROOT                 1
SYS$USERS                 2 CDB$ROOT                 1
cdb1                      6 CDB$ROOT                 1
cdb1XDB                   5 CDB$ROOT                 1
pdb1                      3 PDB1                     3
pdb_ss                    3 PDB_SS                   4

6 rows selected.
```

ALTER SYSTEM while in PDB

Effect of flushing shared pool or buffer cache at different levels

- ALTER SYSTEM FLUSH SHARED_POOL
- ALTER SYSTEM FLUSH BUFFER_CACHE
- ALTER SYSTEM SET USE_STORED_OUTLINES
- ALTER SYSTEM SUSPEND/RESUME
- ALTER SYSTEM CHECKPOINT
- ALTER SYSTEM KILL SESSION
- ALTER SYSTEM DISCONNECT SESSION
- ALTER SYSTEM SET initialization_parameter

(Great commands to run at the PDB level)

Able to modify initialization parameter for a given PDB…


```
SELECT NAME FROM V$PARAMETER
WHERE  ISPDB_MODIFIABLE = 'TRUE'
AND    NAME LIKE 'optim%';   (without condition – can set 147 parameters out of 357)
                                          (There were 341 parameters in 11gR2)
NAME
----------------------------------------
optimizer_adaptive_reporting_only
optimizer_capture_sql_plan_baselines
optimizer_dynamic_sampling
optimizer_features_enable
optimizer_index_caching
optimizer_index_cost_adj
optimizer_mode
optimizer_use_invisible_indexes
optimizer_use_pending_statistics
optimizer_use_sql_plan_baselines

10 rows selected.
```

Key ones modifiable: cursor_sharing, open_cursors, result_cache_mode, sort_area_size

Key ones NOT modifiable: shared_pool_size, db_cache_size, memory_target. pga…

Set PDB Resource Plans …

- Keep runaway PDBs from affecting other PDBs
- Allocate appropriate resource plans (between/within PDBs)
- Set min/max CPU / I/O / Parallelism / (Future: Memory / Network / I/O on non-Exadata)

alter system set RESOURCE_LIMIT = TRUE_CONTAINER = ALL
(dynamically enable resource limits for all containers)

alter system set RESOURCE_LIMIT = TRUE_CONTAINER = CURRENT
(dynamically enable resource limits for the root)

Set PDB Resource Plans …

- If 4 PDBs have 3 shares each, there are 12 shares total and each has 3/12 or 1/4th of the CPU resources.
- If 2 PDBs have 3 shares & 2 PDBs have 1 share, then the ones with 3 shares have 3/8ths of the CPU resources and are 3x more likely to queue parallel queries than the ones that have 1 share.
- CPU utilization_limit and parallel_server_limit percents also be set.

```
BEGIN DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE(
      plan => 'newcdb_plan',
      pluggable_database => 'pdb1',
      shares => 3,
      utilization_limit => 70,
      parallel_server_limit => 70);
END;
/
```

Resource Plan Queries…


```
SELECT PLAN, STATUS, COMMENTS
FROM   DBA_CDB_RSRC_PLANS
ORDER  BY PLAN;
```

```
PLAN                      STATUS              COMMENTS
------------------------- ------------------- --------------------
DEFAULT_CDB_PLAN          ACTIVE              Default CDB plan
ORA$INTERNAL_CDB_PLAN     ACTIVE              Internal CDB plan
```

```
SELECT PLAN, PLUGGABLE_DATABASE, SHARES,
       UTILIZATION_LIMIT, PARALLEL_SERVER_LIMIT
FROM  DBA_CDB_RSRC_PLAN_DIRECTIVES
ORDER BY PLAN;
```

```
                          Pluggable                              Utilization Parallel
Plan                      Database                    Shares        Limit     Limit
------------------------- ------------------------- ------ ----------- --------
DEFAULT_CDB_PLAN          ORA$DEFAULT_PDB_DIRECTIVE      1         100      100
DEFAULT_CDB_PLAN          ORA$AUTOTASK                             90      100
```

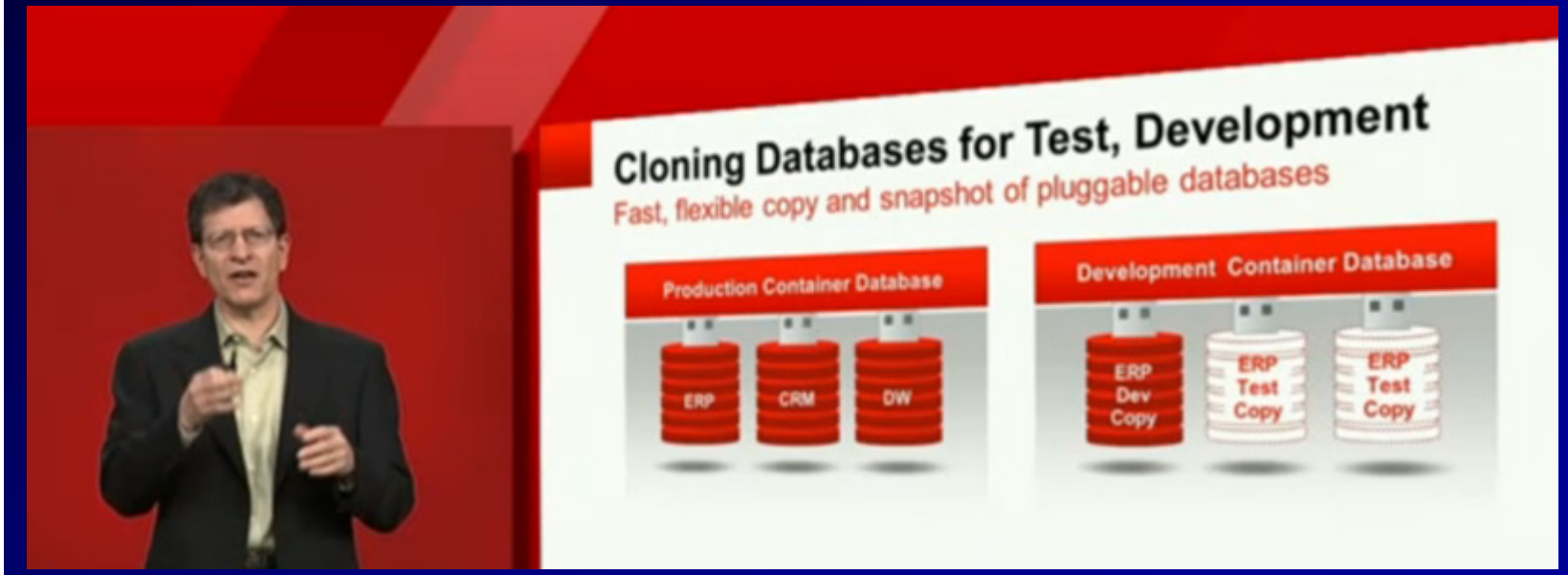
Check PDB History


```
SELECT DB_NAME, CON_ID, PDB_NAME, OPERATION,
       OP_TIMESTAMP, CLONED_FROM_PDB_NAME
FROM   CDB_PDB_HISTORY
WHERE  CON_ID > 2
ORDER  BY CON_ID;
```

Sample output:

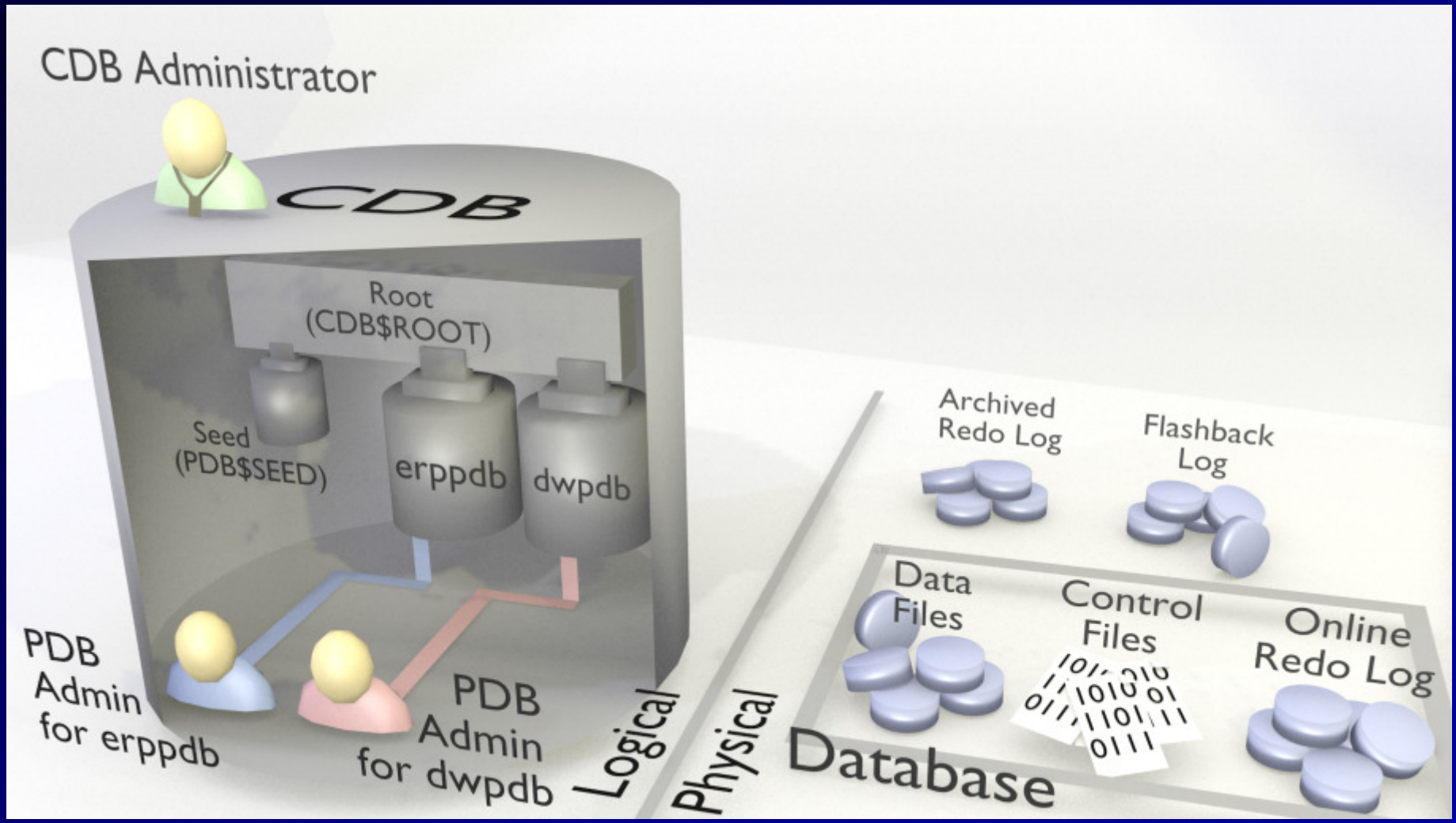
```
DB_NAME    CON_ID PDB_NAME        OPERATION        OP_TIMEST CLONED_FROM_PDB
---------- ------ --------------- ---------------- --------- ---------------
NEWCDB          3 PDB1            CREATE           01-APR-13 PDB$SEED
NEWCDB          4 PDB_SS          CREATE           01-APR-13 PDB$SEED
NEWCDB          5 PDB2            CLONE            02-APR-13 PDB1
```

Get Ready for Pluggable Databases!



This guy and his team working hard to make your life easier!

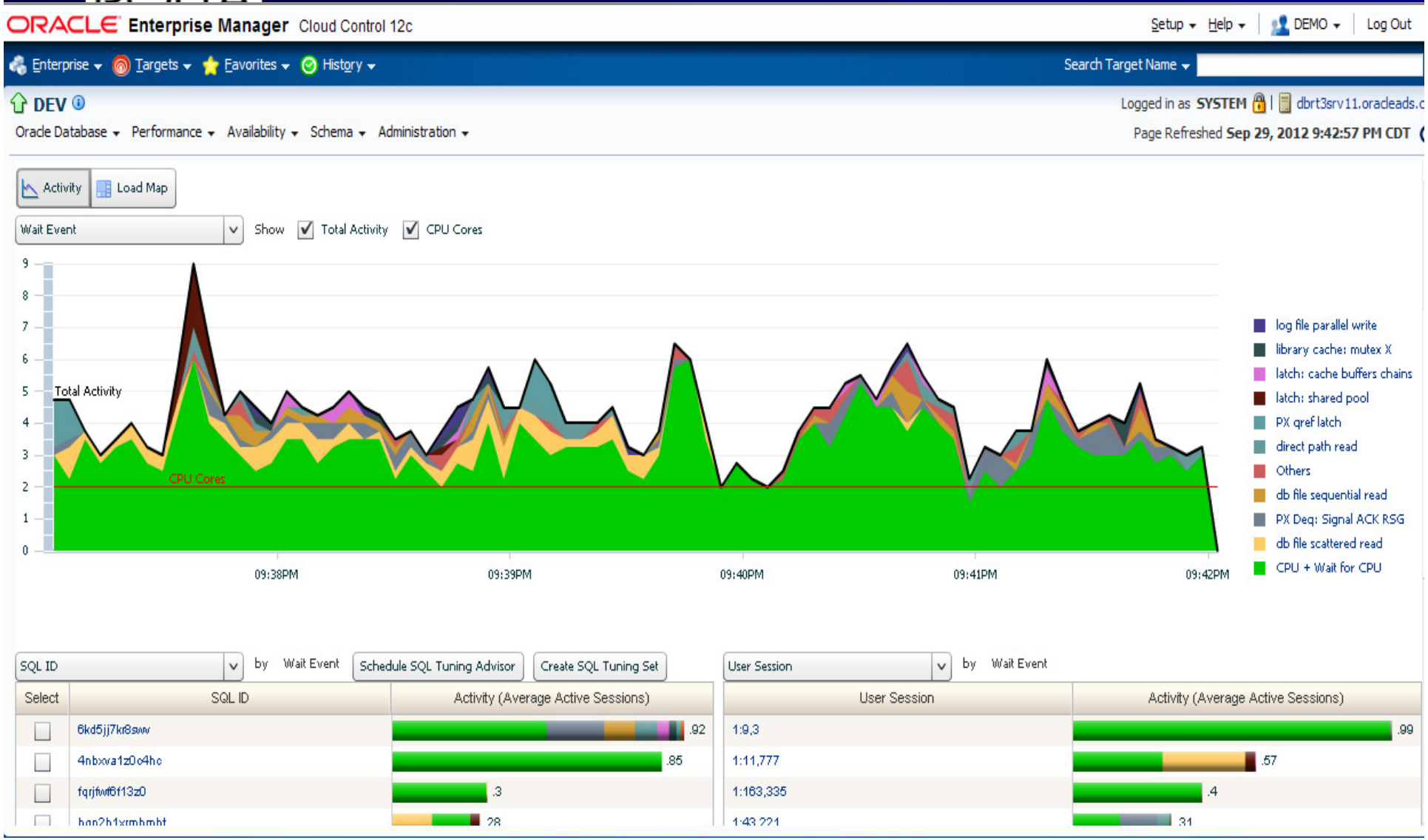
What is your System of the Future?



12c Cloud Control Manages 12c Database & RAC

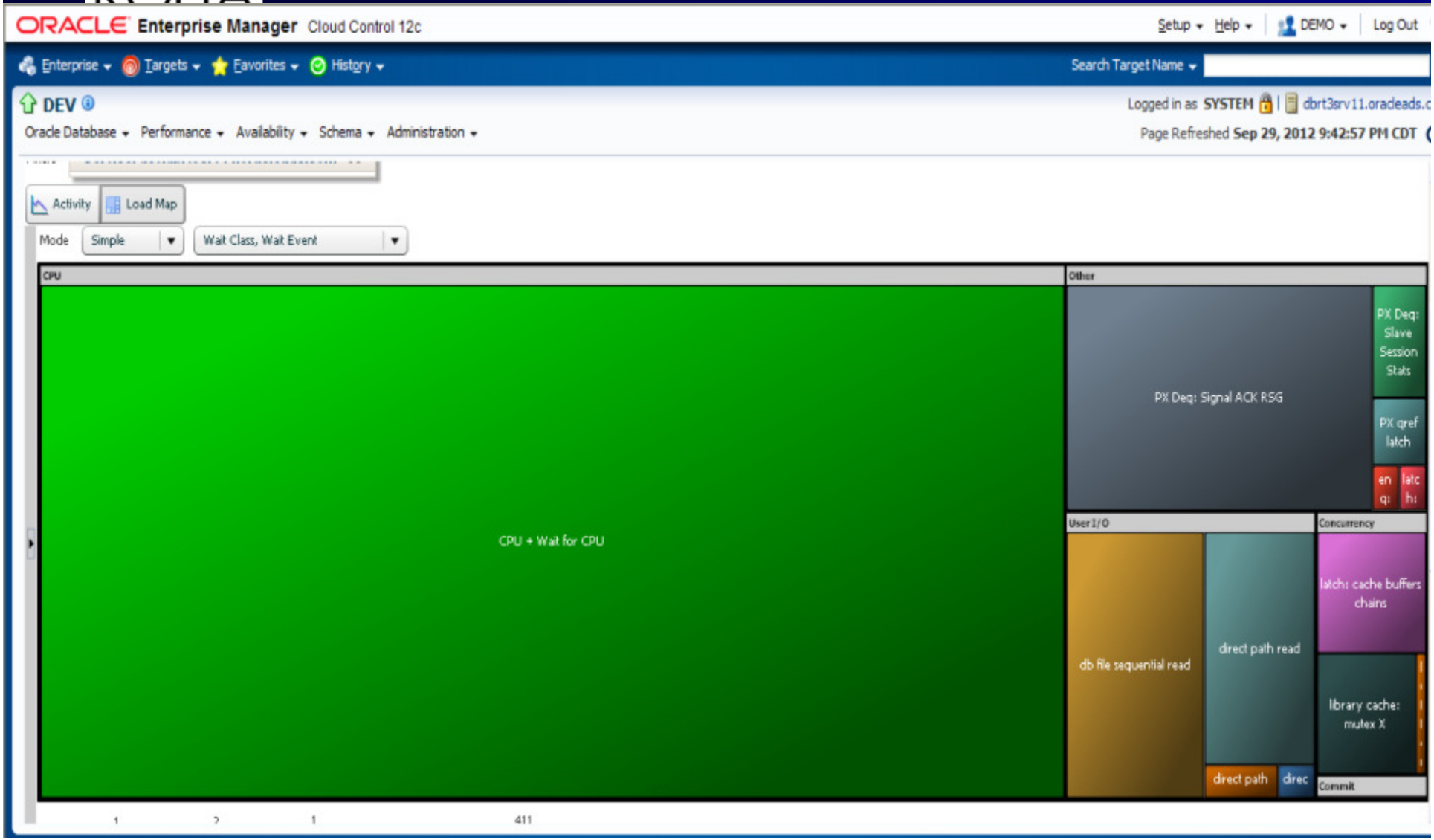
Wait Class – Top Dimensions By SQL ID (Scroll down to see SQL)

Cloud Control 12c



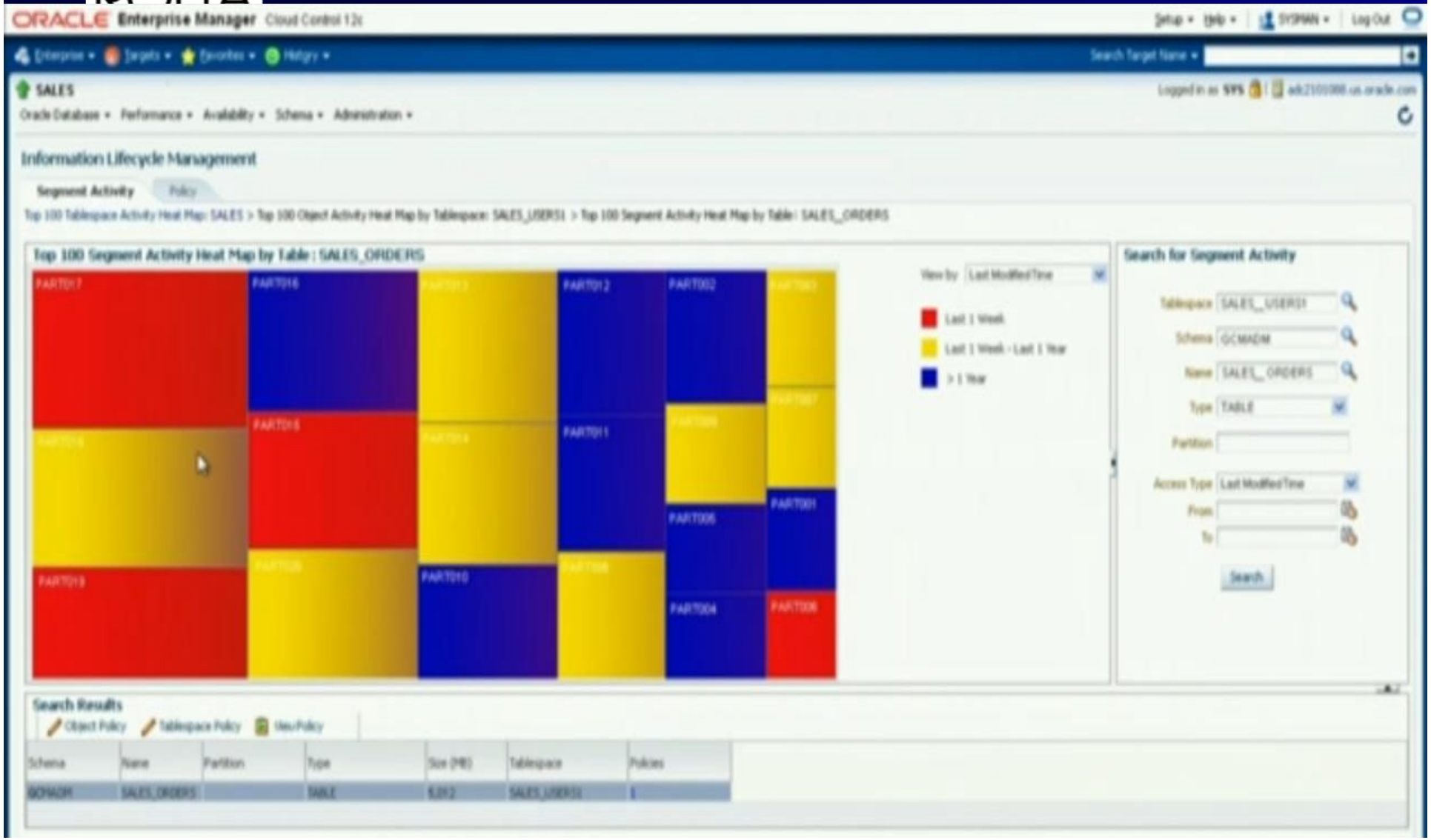
Wait Class – Top Dimensions By SQL ID (Click on LOAD MAP)

Cloud Control 12c



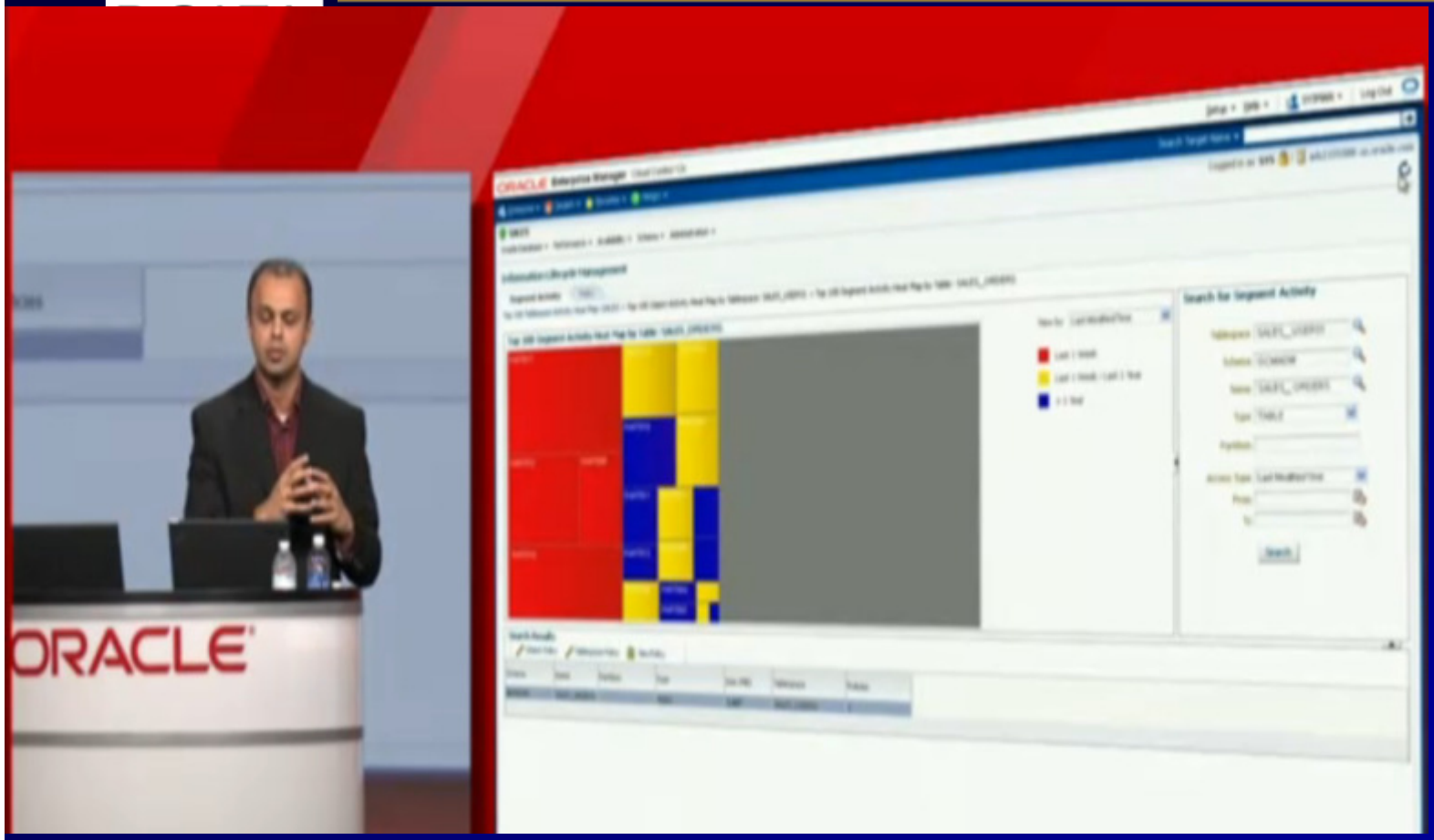
Replay OOW Keynote if you missed it…
Heat Map… A lot of cold data

Cloud Control 12c



Replay OOW Keynote if you missed it… (compress the cold data)

Cloud Control 12c



New in 12c Database:

- **Heat Map** tracks modifications of rows (block level), table, partition levels
- Automate **policy-driven** data movement and compression using **Heat Map**

12c

Nice Developer Tools/Improvements

DDL_LOCK_TIMEOUT – 11g
Enhanced DDL Capabilities – 12c

The DDL Lock Timeout

- DDL Statements (Create/Alter/Drop) require exclusive locks and thus sometimes fail due to bad timing.
- The parameter DDL_LOCK_TIMEOUT specifies the amount of time (in seconds) the DDL statement will wait for the lock before timing out and failing.
- The default value is 0, the max value is 100000 (27.77 hours).
- Example:

```
alter session set DDL_LOCK_TIMEOUT = 30
```

Session altered.

You can specify a lock timeout in seconds for FINISH_REDEF_TABLE

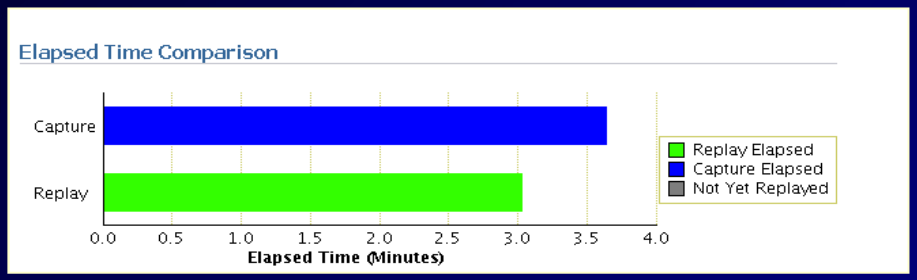
Enhanced DDL Online

- Many schema level DDL maintenance commands no longer have blocking locks. Less of an issue for online use while there are users using the objects. This DDL non-blocking operations include:
  - *DROP INDEX ONLINE*
  - *DROP CONSTRAINT ONLINE*
  - *SET UNUSED COLUMN ONLINE*
  - *ALTER INDEX VISIBLE*
  - *ALTER INDEX INVISIBLE*
  - *SET UNUSED COLUMN ONLINE*

*Can also now **move a Data File** while **Online** and is open and being accessed!*

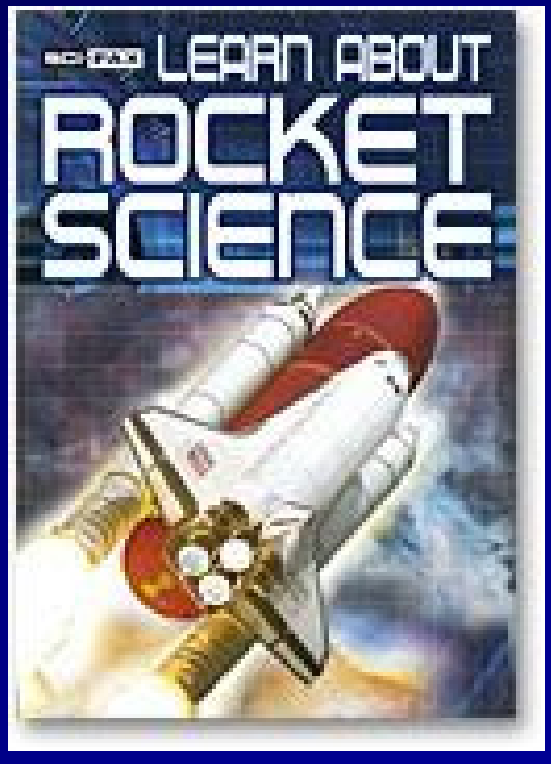
Real Application Testing!
Database workload capture and replay

Replay Options...



- Synchronized Replay
  - Exact Concurrency, commits & data divergence minimal
- Unsynchronized Replay
  - Not the same concurrency or commits
  - Data divergence can be large depending on load test performed
- Creates Report – Better Reporting in 12c
  - Data Divergence, Error Divergence, Performance Divergence
- NEW in 12c: Consolidated Database Replays 12c
  - Take multiple workloads on different databases and consolidate into a single replay (either manually with non-CDBs or with PDBs).

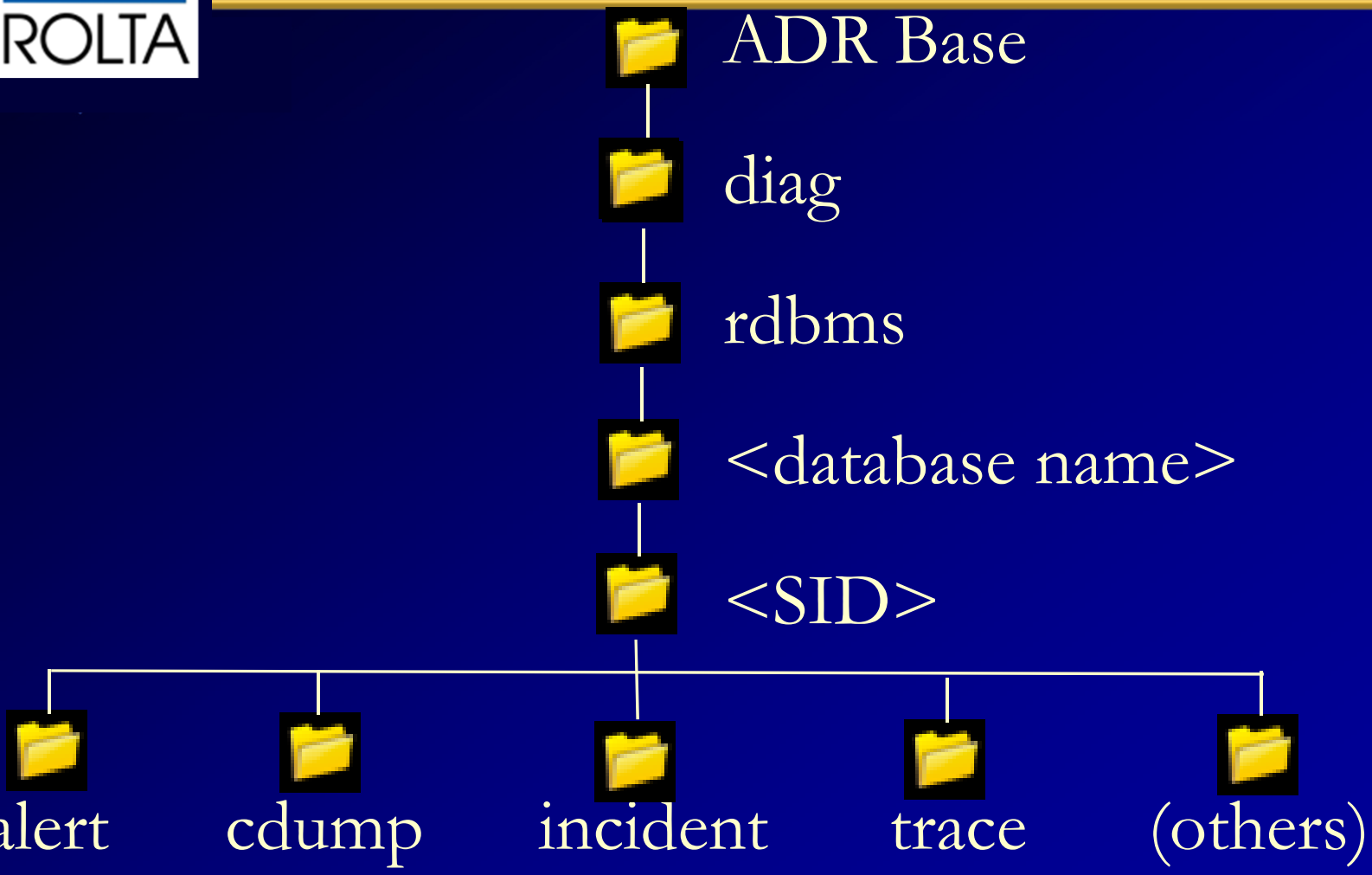
Automatic Diagnostic Repository (ADR)



Automatic Diagnostic Repository (ADR)

- Oracle includes a Fault Diagnosability Infrastructure to prevent, detect, diagnose, resolve issues related to bugs, corruption, etc.
- When a critical error occurs it is assigned an incident number and all diagnostic data tagged with this in ADR.
- ADR is a file based repository outside of the database
- ADR helps detect problems proactively
- ADR helps limit the damage of interruptions
- ADR helps reduce problem diagnostic time
- ADR simplifies Oracle Support / Customer interaction
- The ADR also contains Health Reports, Trace Files, Dump Files, SQL Test Cases and Data Repair Records

ADR Directory Structure for a Database Instance



Alert Log: /u01/app/oracle/diag/rdbms/cdb1/cdb1/trace
ORACLE_HOME: /u01/app/oracle/product/12.1.0/dbhome_1

ADR – V$ Diagnostic Info

12c – No changes (that I saw)


```
  1* select * from V$diag_info
SYS@sillgr2> /

   INST_ID NAME                           VALUE
---------- ------------------------------ ------------------------------------------------------------
         1 Diag Enabled                   TRUE
         1 ADR Base                       /u01/app/oracle
         1 ADR Home
         1 Diag Trace
         1 Diag Alert
         1 Diag Incident
         1 Diag Cdump
         1 Health Monito
         1 Default Trace
         1 Active Proble
         1 Active Incide

11 rows selected.

SYS@sillgr2>
SYS@sillgr2>
SYS@sillgr2>
```

```
  1* SELECT * FROM V$DIAG_INFO
SQL> /

 INST_ID NAME                      VALUE
-------- ------------------------ ------------------------------------------------------------
       1 Diag Enabled             TRUE
       1 ADR Base                 /u01/app/oracle
       1 ADR Home                 /u01/app/oracle/diag/rdbms/cdb1/cdb1
       1 Diag Trace               /u01/app/oracle/diag/rdbms/cdb1/cdb1/trace
       1 Diag Alert               /u01/app/oracle/diag/rdbms/cdb1/cdb1/alert
       1 Diag Incident            /u01/app/oracle/diag/rdbms/cdb1/cdb1/incident
       1 Diag Cdump               /u01/app/oracle/diag/rdbms/cdb1/cdb1/cdump
       1 Health Monitor           /u01/app/oracle/diag/rdbms/cdb1/cdb1/hm
       1 Default Trace File       /u01/app/oracle/diag/rdbms/cdb1/cdb1/trace/cdb1_ora_3045
       1 Active Problem Count     3
       1 Active Incident Count    17

11 rows selected.

SQL> This is 12c Query Output above!
```

Security Enhancements

Enhanced security of Audit Data with new

AUDIT_ADMIN role

- Also SYSBACKUP privilege (don't need SYSDBA for RMAN)
- Update strong user authentication using kerberos
- Simplified Vault administration

Oracle Database Security

Built over MANY years...

ORACLE DATABASE 11g

Oracle Audit Vault

Oracle Database Vault

DB Security Evaluation #19

Transparent Data Encryption

EM Configuration Scanning

Fine Grained Auditing (9i)

Secure application roles

Client Identifier / Identity propagation

Oracle Label Security (2000)

Proxy authentication

Enterprise User Security

Global roles

Virtual Private Database (8i)

Database Encryption API

Strong authentication (PKI, Kerberos, RADIUS)

Native Network Encryption (Oracle7)

Database Auditing

1977 Government customer

2007

Other 12c Features …

- Database Instance Smart Flash Cache Support for Multiple Devices (can access/combine) without the overhead of the local volume manager.
- Supports **In-Memory Jobs** & In-Memory Temporary Tablespaces
- Active Data Guard Security has in-memory table of failed login attempts
- **Heat Map** that tracks modifications of rows (block level), table, partition levels
- Automate policy-driven data movement and compression using Heat Map
- Move partitions while ONLINE with DML happening
- Improved query performance against OLAP cubes (especially Exadata)
- Automatic extended stats for groups of columns accessed together
- DBMS_STATS.GATHER_TABLE_STATS run on a partitioned table when CONCURRENT is set to TRUE will gather stats using multiple jobs concurrently
- Online statistics gathered during a bulk load (similar to rebuild index command)
- Flashback Data Archive (FDA) can be fully used on HCC tables on Exadata
- Enterprise Manager Database Express 12c ships with every database (NICE!)
- "Spot ADDM" triggered by high CPU or I/O into AWR Reports
- **Mask Data** At Source for testing & Oracle Masking templates for E-Business
- **Oracle Data Redaction** (prevents things like SSN from being displayed)

Other 12c Features …

- Full Transportable support & Point-in-time recovery for PDBs
- TRUNCATE TABLE …CASCADE (truncate child tables too)
- Data Pump No Logging Option for import
- No-echo of Encryption Passwords on expdp/impdp commands
- Sql*Loader Express Mode – no control file!
- **In-Database MapReduce (Big Data)**
- Update strong user authentication using kerberos & Simplified Vault administration
- Many Windows enhancements (if you must use Windoze)
- Fast Application Notification (FAN) gets improved with Application Continuity which helps recover incomplete requests without executing more than once.
- Real-Time Apply (redo) is now default for Data Guard vs. applying archive logs
- SQL Apply Support for Objects, Collections, XML Type, & SecureFiles LOBs
- Oracle Spacial is now Oracle Spacial & Graph – Enhancements include routing engine enhancements, caching of index metadata, vector performance, Asian address support (geocoding), raster algebra & analytics, enhance image processing
- Many ACFS, Oracle Multimedia, Oracle Text & Oracle XML enhancements
- **VARCHAR2(32767)** –not default/4K stored inline/>4K out of line(like a LOB)

12c Deprecated Features …fyi

(could be desupported in future releases)

- IGNORCASE argument of ORAPWD
- Single character options with SVRCTL (accepts full-word options now)
- *_SCHEDULER_CREDENTIALS

(This list will certainly change in the future…)

The Future: 8 Exabytes
Look what fits in one 12c Database!

2K – A typewritten page
5M – The complete works of Shakespeare
10M – One minute of high fidelity sound
2T – Information generated on YouTube in one day
10T – 530,000,000 miles of bookshelves at the Library of Congress
20P – All hard-disk drives in 1995 (or your database in 2010)
700P –Data of 700,000 companies with Revenues less than $200M
1E – Combined Fortune 1000 company databases (average 1P each)
1E –Next 9000 world company databases (average 100T each)
8E – Capacity of ONE Oracle12c Database (CURRENT)
12E to 16E – Info generated before 1999 (memory resident in 64-bit)
16E – Addressable memory with 64-bit (CURRENT)
161E – New information in 2006 (mostly images not stored in DB)
1Z – 1000E (Zettabyte - Grains of sand on beaches -125 Oracle DBs)
100TY - 100T-Yottabytes – Addressable memory 128-bit (FUTURE)

8 Exabytes: Look what fits in one 12c Database!

- **All databases of the largest 1,000,000 companies in the world (3E).**

or

- All Information generated in the world in 1999 (2E)

or

- All Information generated in the world in 2003 (5E)

or

- All Email generated in the world in 2006 (6E)

or

- 1 Mount Everest filled with Documents (approx.)

Bigger Data – Get Ready for it…

- Worldwide, data is growing rapidly*:
  - 2000: 800 Terabytes (10^{12})
  - 2006: 160 Exabytes (10^{18})
  - 2009: 500 Exabytes (just Internet)
  - 2012: 2.7 Zettabytes (10^{21})
  - 2020: 35 Zettabytes …?
- Data generated in ONE day*….?
  - Twitter: 7 TB
  - Facebook: > 10 TB

Brain: 2.8 x 10^{20} bits of Memory Space – John von Neumann, Harvard

Big data: The next frontier for innovation, competition, and productivity McKinsey Global Institute 2011

We are drowning in *data*, but thirsting for Information

**** Data collated from various online sources***

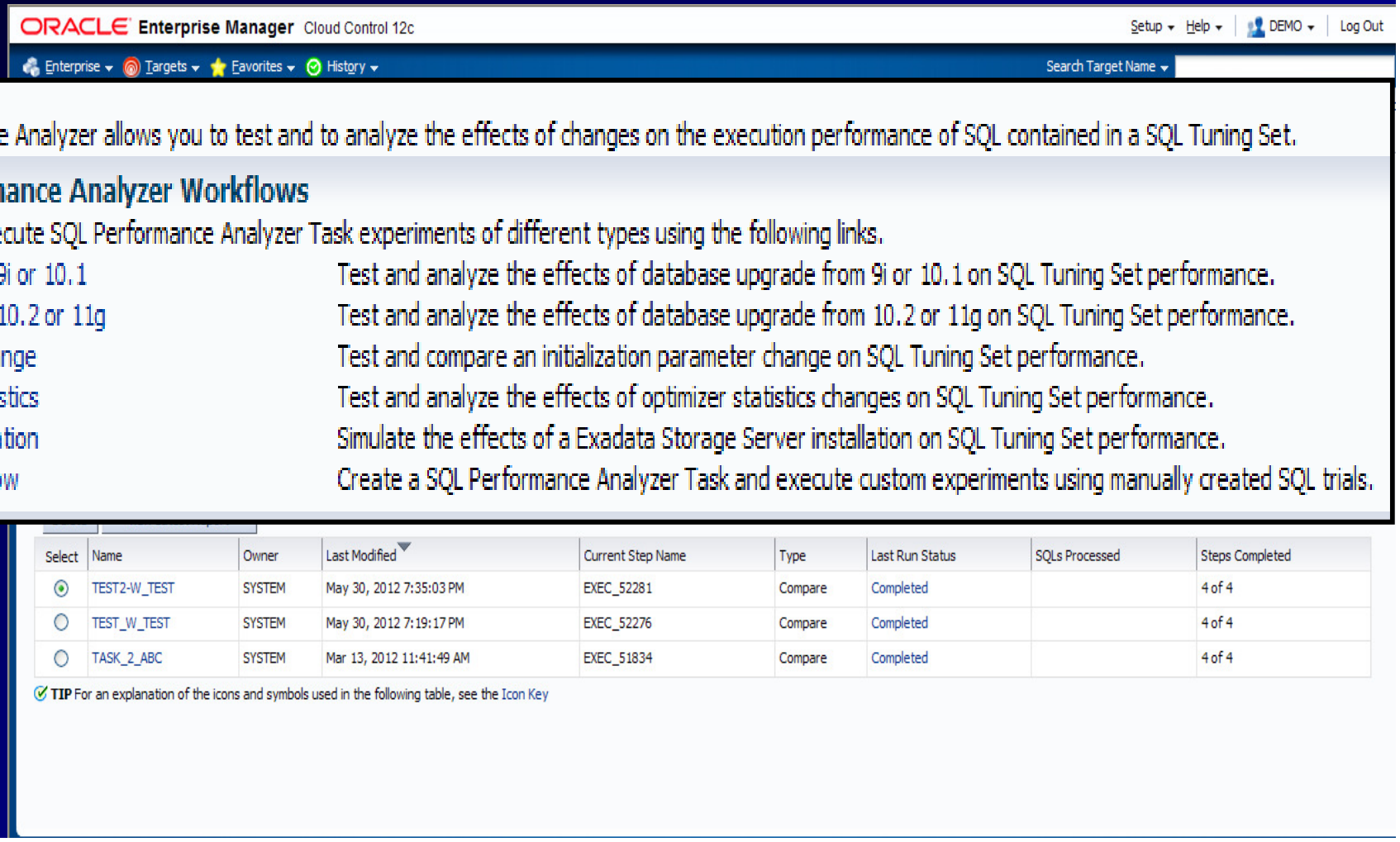
V$ Views over the years

Version	V$ Views	X$ Tables
6	23	? (35)
7	72	126
8.0	132	200
8.1	185	271
9.0	227	352
9.2	259	394
10gR1	340 (+31%)	543 (+38%)
10gR2	396	613
11gR1	**484 (+42%)**	**798 (+46%)**
11gR2	**525 (+33%)**	**945 (+54%)**
12cR1	**606 (+25%)**	**1062 (+33%)**

More SPEED Coming… Get Ready… This guy does not ever slow down!!

SQL Performance Analyzer 12c – Exadata Simulation

Cloud Control 12c



Exadata Simulation

Cloud Control 12c

ORACLE Enterprise Manager Cloud Control 12c

Setup ▾ Help ▾ DEMO ▾ Log Out

Enterprise ▾ Targets ▾ Favorites ▾ History ▾ Search Target Name ▾

DEV

Logged in as SYSTEM | dbrt3srv11.oracleads.c

Oracle Database ▾ Performance ▾ Availability ▾ Schema ▾ Administration ▾

Advisor Central > SQL Performance Analyzer > SQL Performance Analyzer Task: SYSTEM.TOP3 > SQL Performance Analyzer Task Report: SYSTEM.TOP3

Logged in As SYSTEM

SQL Performance Analyzer Task Report: SYSTEM.TOP3

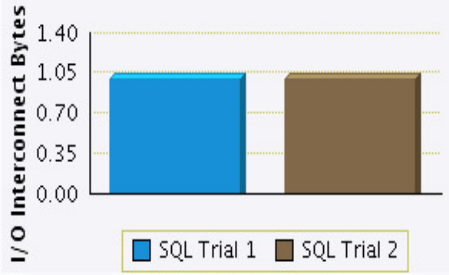
Save Mail

SQL Tuning Set Name TOP_ACTIVITY_1317427853462
STS Owner SYS
Total SQL Statements 1
SQL Statements With Errors 0

SQL Trial 1 INITIAL_SQL_TRIAL
SQL Trial 2 SECOND_SQL_TRIAL
Comparison Metric I/O Interconnect Bytes

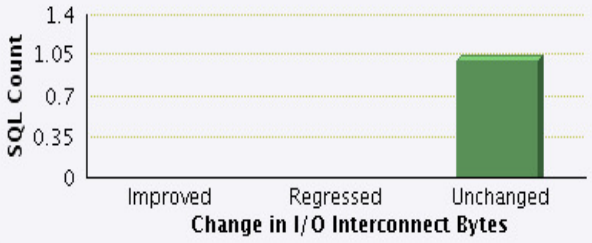
Global Statistics

Projected Workload I/O Interconnect Bytes



Improvement Impact 0%
Regression Impact 0%
Overall Impact 0%

SQL Statement Count



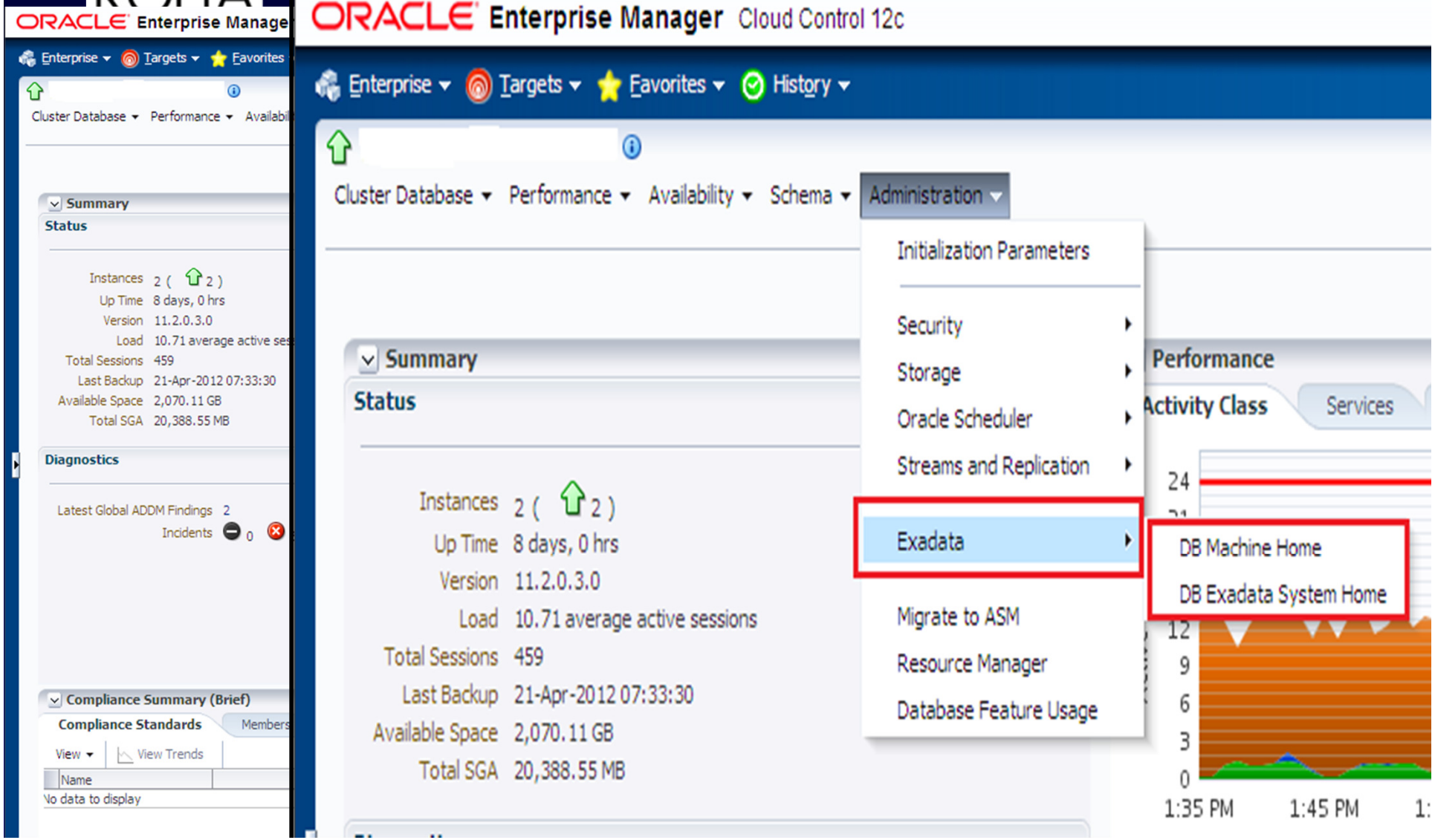
Top 10 SQL Statements Based on Impact on Workload

	SQL ID	Net Impact on Workload (%)	I/O Interconnect Bytes		Net Impact on SQL (%)
			SQL Trial 1	SQL Trial 2	
⇨	6mnkk3r01hn7n	0.000	0	0	0.000

TIP A '-' means that the value is not applicable.

Cloud Control 12c

Cloud Control 12c – Monitor Exadata



Summary – 12c Database

- Know the Oracle!
- Start Me Up – Using Memory Target, The Buffer Cache & The Result Cache
- Invisible Columns (12c) & virtual columns (11g)
- Multiple indexes on the same Column (12c) & Invisible Indexes (11g)
- Adaptive Execution Plans (12c) & Adaptive Cursor Sharing & Bind Peeking (11g)
- Runaway query Management (12c)
- Change Table Compression at import Time (12c) & (Partition Compression – 11g)
- Create Views as Tables (12c)
- Online Move Partition (12c) & Interval Partitioning (11g)
- Partial Indexes for Partitioned Table (12c)
- Pluggable Databases (12c)
- Enhanced DDL Online (12c)
- Exadata and Big Data (In-Database MapReduce in 12c)
- Consolidated Database Replays & Better Reporting (12c)
- Automatic Diagnostics Repository (12c)
- Security Enhancements (12c)
- Other 12c New Features