Oracle Database 12cR2- New feauture

# Architecture new feature

## Multitenant Container Database

### Benefits

DBA resource costs are reduced with:

* No application change and very fast provisioning: A new database can be provisioned quickly. A clone of a populated database can be created quickly. A populated database can be quickly unplugged from its CDB on one platform and quickly plugged into a CDB on a different platform. A non-CDB can quickly be plugged into a CDB.
* Fast upgrade and patching of the Oracle Database version: The cost (time taken and human effort needed) to upgrade many PDBs is the cost of upgrading a single Oracle Database occurrence. You can also upgrade a single PDB by unplugging it and plugging it into a CDB at a different Oracle database version.

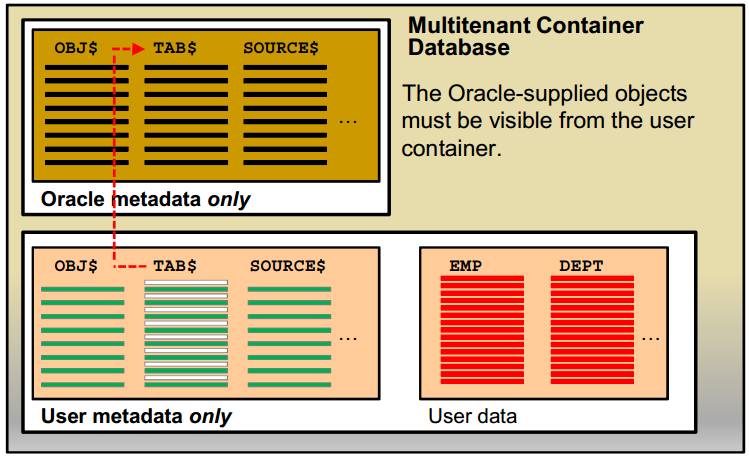
The multitenant architecture maintains:

* Secure separation of duties: The administrator of an application can do all the required tasks by connecting to the particular PDB that implements its back end. However, someone who connects to a PDB cannot see other PDBs. To manage PDBs as entities (for example, to create or drop or unplug or plug one), the system administrator needs to connect to the CDB. For these specific tasks, new privileges need to be granted.
* Isolation of applications that may not be achieved manually unless by using Database Vault, for example. A good example of isolation is dictionary separation enabling Oracle database to manage the multiple PDBs separately from each other and from the CDB itself.

Ensures full backwards-compatibility with non-CDBs

Fully operates with RAC and Data Guard

### OverView



One of the goals of the multitenant architecture is that each container has a one-to-one relationship with an application.

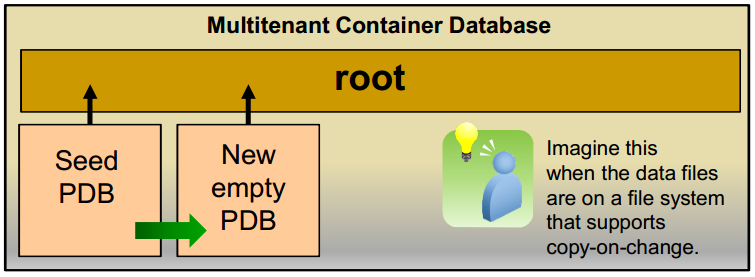
Separating the metadata is the first step, the second is allowing the application or users inside the “user” container to access the Oracle-supplied objects.

**Naming**

The Oracle-supplied objects reside in a container called the root container (named CDB$ROOT).

The user container is called a pluggable database (PDB) and has the name you give it when creating it or plugging it into the CDB.

**Provisioning a Pluggable Database**



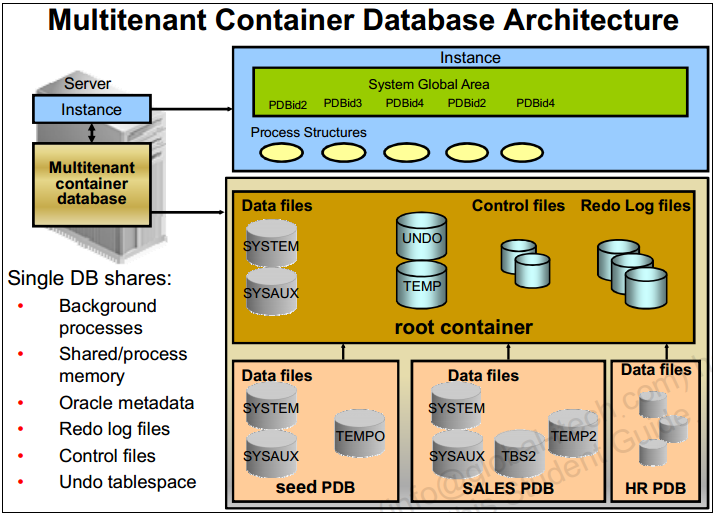
There are four methods to provision pluggable databases:

* Create a new PDB from PDB$SEED pluggable database: for example, for a brand new application implementation.
* Create a new PDB from a non-CDB: plug the non-CDBs in a CDB as PDBs, as part of migration strategy. It is also a good way to consolidate the non-CDBs into a CDB.
* Clone a PDB from another PDB into the same or another CDB: an example of this method is application testing.
* Plug an unplugged PDB into a CDB, for example, instead of upgrading a multitenant container database from one release to another, you can unplug a pluggable database from one Oracle Database release, and then plug it in to a newly created multitenant container database from a higher release

**Fast intra-CDB db link**

* With a multitenant architecture that holds several PDBs, these once separated non-CDBs may now reside in a single instance, sharing memory, disk, and CPU resources, but maintaining application separation.
* These databases shared data by using database links. The database link still works, but now because the “link” communication does not leave the instance, the link is very fast.

### Architecture



At the physical level, the CDB has a database instance and database files, just as a non-CDB does.

* The redo log files are common for the whole CDB. The information it contains is annotated with the identity of the PDB where a change occurs. Oracle GoldenGate is enhanced to understand the format of the redo log for a CDB. All PDBs in a CDB share the ARCHIVELOG mode of the CDB.
* The control files are common for the whole CDB. The control files are updated to reflect any additional tablespace and data files of plugged PDBs.
* The UNDO tablespace is common for all containers.
* A temporary tablespace common to all containers is required. But each PDB can hold its own temporary tablespace for its own local users.
* Each container has its own data dictionary stored in its proper SYSTEM tablespace, containing its own metadata, and a SYSAUX tablespace.
* The PDBs can create tablespaces within the PDB according to application needs.
* Each data file is associated with a specific container, named CON\_ID.

### Containers

Two types of containers in V$CONTAINERS:

* The root container:
  + The first container created at CDB creation
  + Mandatory
  + Oracle system-supplied common objects and metadata
  + Oracle system-supplied common users and roles
* Pluggable database containers (PDBs):
  + A container for an application:
    - Tablespaces (permanent and temporary)
    - Schemas / objects / privileges
    - Created / cloned / unplugged / plugged
  + Particular seed PDB:
    - PDB$SEED provides fast provisioning of a new PDB
  + Limit of 253 PDBs in a CDB including the seed
  + Limit of 1024 services in a CDB

SQL> select CON\_ID, DBID, NAME, OPEN\_MODE, CREATE\_SCN from V$CONTAINERS;

CON\_ID DBID NAME OPEN\_MODE CREATE\_SCN

---------- ---------- ---------------- ---------- ---------

1 1978024075 CDB$ROOT READ WRITE 0

2 4001565720 PDB$SEED READ ONLY 1594430

3 4177599030 PDB1 MOUNTED 1779882

What belongs to the CDB and not to a specific container?

* Control files and redo log files

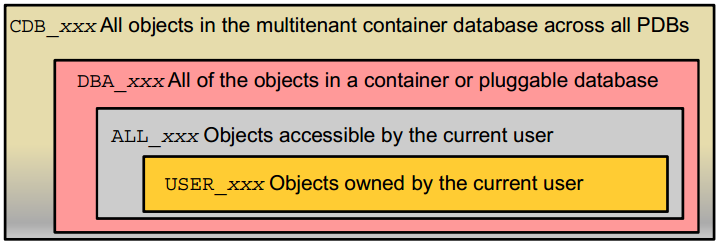
What is in the root that does not exist in PDBs?

* An UNDO tablespace for each instance of a RAC database
* Oracle-supplied metadata
* Shared Oracle-supplied data
* CDB views providing information across PDBs
* CDB resource manager plan allowing resource management between PDBs within a CDB

What is in a PDB that is not in the root nor in another PDB?

* Application tablespaces
* Local temporary tablespaces
* Local users and local roles
* Non-shared local metadata
* PDB resource manager plan allowing resource management within PDB

### Oracle Database basic view



SQL> SELECT view\_name FROM dba\_views WHERE view\_name like 'CDB%';

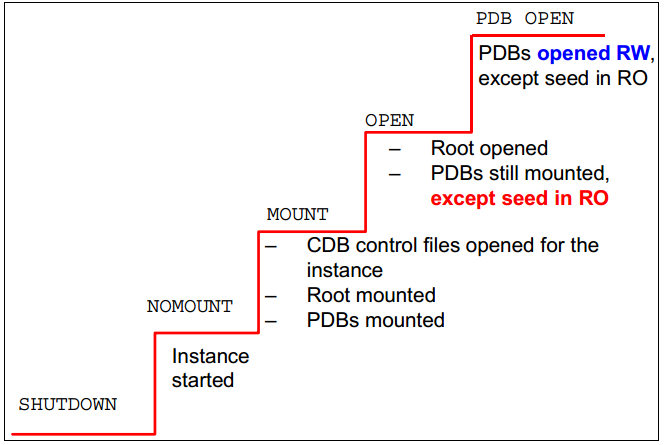
* CDB\_pdbs: All PDBS within CDB
* CDB\_tablespaces: All tablespaces within CDB
* CDB\_users: All users within CDB (common and local)

DBA dictionary views providing information within PDB:

SQL> SELECT table\_name FROM dict WHERE table\_name like 'DBA%';

### Startup và Shutdown Database

**Startup CDB**:



**Shutting Down CDB**

SQL> CONNECT sys@CDB1 AS SYSDBA

SQL> SHUTDOWN IMMEDIATE

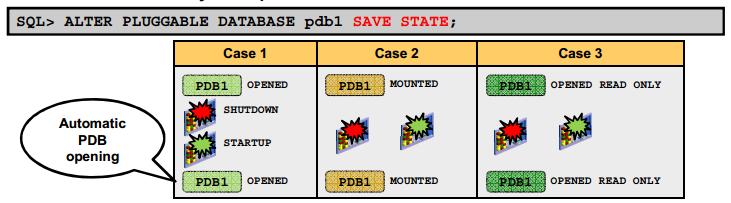
* All PDBs closed (no new specific message)
* CDB closed
* CDB dismounted
* Instance shut down

**Close PDB**

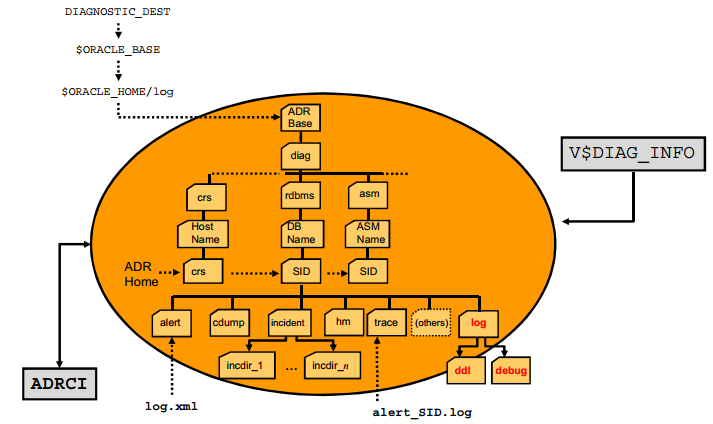
SQL> CONNECT sys@PDB1 AS SYSDBA

SQL> SHUTDOWN IMMEDIATE

**Automatic PDB Opening**



### Admin log in database



**Alert log file**

Location*: $ORACLE\_BASE/diag/rdbms/<db\_name>/<SID>/trace*

The alert file of a database is a chronological log of messages such as the following:

* + Any nondefault initialization parameters used at startup
  + All internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occurred
  + Administrative operations, such as the SQL statements *CREATE, ALTER*, *DROP*

*DATABASE*, and *TABLESPACE*; and the Enterprise Manager or SQL\*Plus statements *STARTUP, SHUTDOWN, ARCHIVE LOG,* and *RECOVER*

* + Several messages and errors relating to the functions of shared server and dispatcherprocesses
  + Errors during the automatic refresh of a materialized view

**DDL Log File**

The DDL log is created only if the ENABLE\_DDL\_LOGGING initialization parameter is set to TRUE

* + DDL log contains one log record for each DDL statement.
  + Two DDL logs containing the same information:
    - XML DDL log: *log.xml* written to

*$ORACLE\_BASE/diag/rdbms/<dbname>/<SID>/log/ddl*

* + - Text DDL: *ddlsid.log* written to

*$ORACLE\_BASE/diag/rdbms/<dbname>/<SID>/log*

**Debug Log File**

Debug log contains warnings about conditions, states, or events that do not inhibit correct operation of an Oracle Database component.

* + The log is intended for use by Oracle Support when diagnosing a problem.
  + It is included in incident packaging service (IPS) incident packages.
  + It is written to

*$ORACLE BASE/diag/rdbms/<db name \_name>/<SID>/debug.*

**New ADRCI Command**

Show log

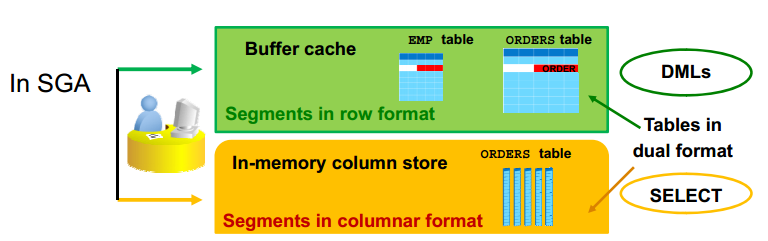
$ adrc

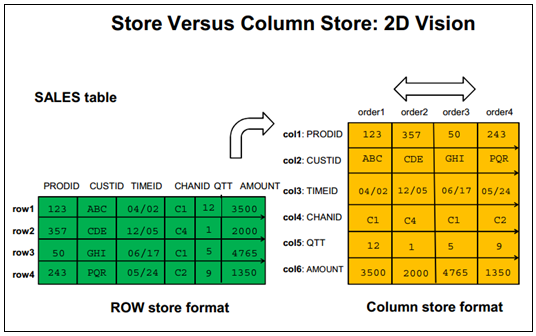
adrci> SHOW LOG;

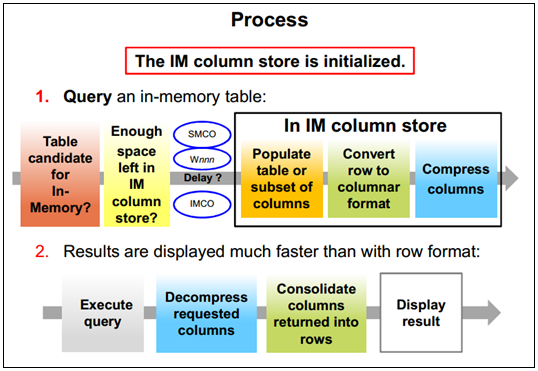
## In-Memory Column Store

### Concept

* + The In-Memory Column Store feature enables objects (tables, partitions, and other types) to be stored in memory in a new format known as the columnar format. This format enables scans, joins, and aggregates to perform much faster than the traditional on-disk format, thus providing fast reporting and DML performance for both OLTP and DW environments.
  + This is particularly useful for analytic applications that operate on **few columns returning many rows** rather than for OLTP that operates on few rows returning many columns. **The DBA must define** the segments that are to be populated into the in-memory column store (IM column store), such as hot tables, partitions, and more precisely the more frequently accessed columns.
  + The in-memory columnar format does not replace the on-disk or buffer cache format It is a consistent copy of a table or of some columns of a table converted to the new columnar format that is independent of the disk format and only available in memory. Because of this independence, applications are able to transparently use this option without any changes. For the data to be converted into the new columnar format, a new pool is requested in the SGA. The pool is the **IM column store**.
  + There are three main advantages:
    - Queries run a lot faster: All data can be populated in memory in a compressed columnar format. No index is required and used. Queries run at least 100 times faster than when fetching , p data from the buffer cache, thanks to the columnar compressed format.
    - DMLs are faster: Analytics indexes can be eliminated by being replaced by scans of the IM column store representation of the table.
    - Arbitrary ad hoc queries run with good performance, because the table behaves as if all columns are indexed.







### Deploying IM Column Store

* Verify the database compatibility value.

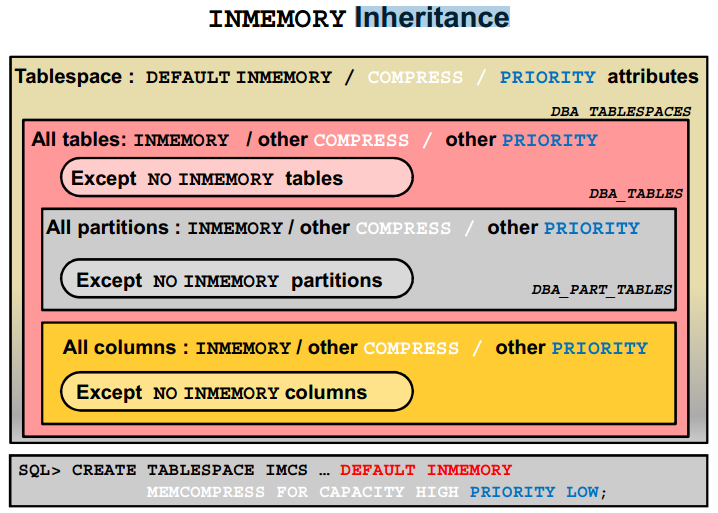
COMPATIBLE = 12.1.0.0.0

* Configure the IM column store size.

INMEMORY\_SIZE = 100G

* Enable/disable a whole segment

SQL> CREATE TABLE large\_tab (c1 …) INMEMORY; Dual format  
SQL> ALTER TABLE sales NO INMEMORY; Row format only  
SQL> ALTER TABLE t1 INMEMORY ; Dual format



### Priority

By default, the population of an in-memory table or partition or subpartition or even column depends on the PRIORITY value set for the table or partition or subpartition or column.

* The NONE value means an “on demand” population when the data is queried only. The default is PRIORITY NONE when no PRIORITY is defined.
* There are four priority levels: LOW, MEDIUM, HIGH, and CRITICAL. If one of these levels is specified, in-memory objects are populated in priority order at instance startup on the next IMCO cycle (default 2 minute cycles).
  + IMCO initiates background population and repopulation of in-memory enabled objects, queuing population tasks.
  + The population tasks are queued to SMCO background process for execution and thus may have to wait for available worker processes (Wnnn). SMCO dynamically spawns slave processes (Wnnn) to implement these tasks.

The IMCO background process queues population tasks for objects with priority other than NONE only. The queue is drained in priority order, from CRITICAL to LOW.

If the population runs out of IM column store space, all tables with higher priority take precedence over those with lower priority. The statement that does not populate data into IM column store is committed in order to not break any application semantics and uses the buffer cache.

### Interaction

Work well with:

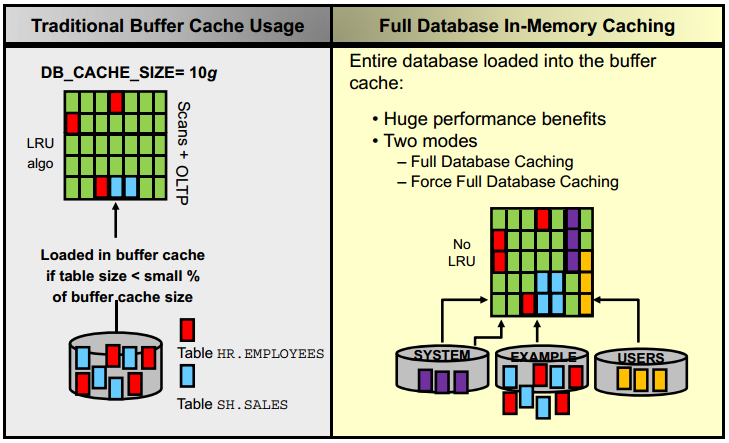
* Rac
* In-Memory PX
* Data Pump.

## Full Database In-Memory Caching

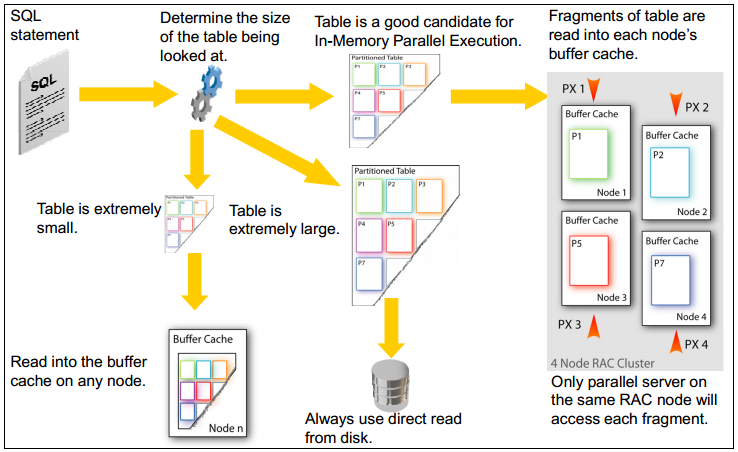
### Concept

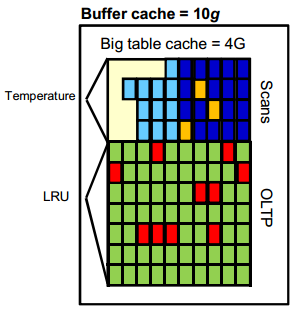
The Full Database In-memory Caching feature enables an entire database to be cached in memory when the database size (sum of all data files, SYSTEM tablespace LOB CACHE files , LOB CACHE files minus SYSAUX, TEMP) is smaller than the buffer cache size. Caching and running a database from memory leads to huge performance benefits. Two modes can be used:

* + Full Database Caching: Implicit default and automatic mode in which an internal calculation determines if the database can be fully cached for an instance. NOCACHE LOBs are not cached in Full Database Caching but in Force Full Database Caching mode even NOCACHE LOBs are cached.
  + Force Full Database Caching: Neither Full Database Caching nor Force Full Database Caching forces or prefetches data into memory. **Workload must access the data first for them to be cached**. It considers the entire database as eligible to be completely cached in the buffer cache. This mode requires the DBA to execute the *ALTER DATABASE FORCE FULL DATABASE CACHING* command. This mode takes precedence over Full Database Caching mode. To revert to traditional caching, use the *ALTER DATABASE NO FORCE FULL DATABASE CACHING* command



### Automatic Big Table Caching





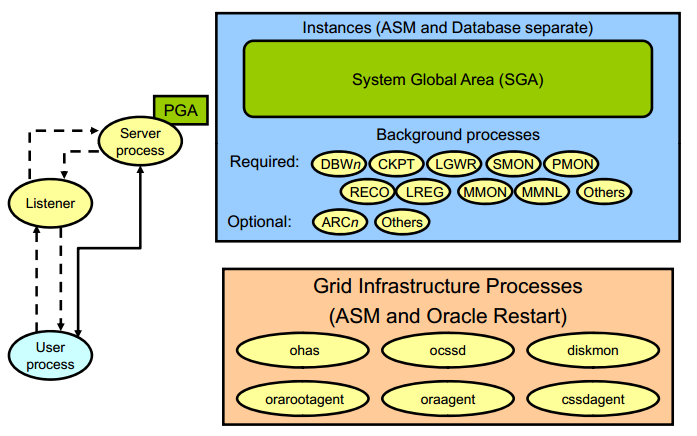
Automatic Big Table Caching enhances the in-memory parallel query capabilities of the Oracle Database in both single instance and Oracle RAC environments. An optional section of the buffer cache, called the big table cache, is used to store data for parallel table scans.

If a large table is approximately the size of the combined size of the Big Table Cache of all instances, the table is partitioned and cached, or mostly cached, on all instances. With in memory PX, this could eliminate most disk reads for queries on the table, or the database could intelligently read from disk only for those portions of the table that do not fit in the Big Table Cache.

## Processes Architecture

The processes in an Oracle Database system can be divided into three major groups:

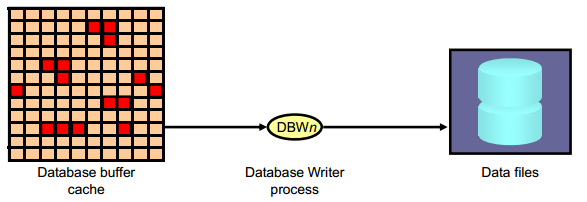
* + User processes that run the application or Oracle tool code
  + Oracle Database processes that run the Oracle Database server code (including server processes and background processes)
    - Server processes created on behalf of each user’s application can perform one or more of the following:
      * Parse and run SQL statements issued through the application.
      * Read necessary data blocks from data files on disk into the shared database buffers of the SGA (if the blocks are not already present in the SGA).
      * Return results in such a way that the application can process the information
    - To maximize performance and accommodate many users, a multiprocess Oracle Database system uses some additional Oracle Database processes called background processes. An Oracle Database instance can have many background processes.
  + Oracle daemons and application processes not specific to a single database
    - Networking listeners
    - Grid Infrastructure daemons



The background process performs functions instead of corresponding processing calls. It controls in and out, providing parallel processing mechanisms for increased efficiency and reliability. Depending on the configuration, Oracle instance has Background processes such as:

* + Database Writer process (DBWn)
  + Log Writer process (LGWR)
  + Checkpoint process (CKPT)
  + System monitor process (SMON)
  + Process monitor process (PMON)
  + Recoverer process (RECO)
  + Listener registration process (LREG)
  + Manageability monitor process (MMON)
  + Manageability monitor lite process (MMNL)
  + Job queue coordinator (CJQ0)
  + Job slave processes (Jnnn)
  + Archiver processes (ARCn)
  + Queue monitor processes (QMNn)

### Database Writer (DBWn)



The Database Writer process (DBWn) writes the contents of buffers to data files.

* + The DBWn processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk.
  + The *DB\_WRITER\_PROCESSES* initialization parameter specifies the number of DBWn processes. The maximum number of Database Writer processes is 100. If it is not specified by the user during startup, Oracle Database determines how to set *DB\_WRITER\_PROCESSES* based on the number of CPUs and processor groups.

The DBWn process writes dirty buffers to disk under the following conditions:

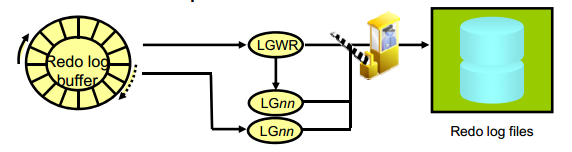
* + When a server process cannot find a clean reusable buffer after scanning a threshold

number of buffers, it signals DBWn to write. DBWn writes dirty buffers to disk asynchronously while performing other processing.

* + DBWn writesbuffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

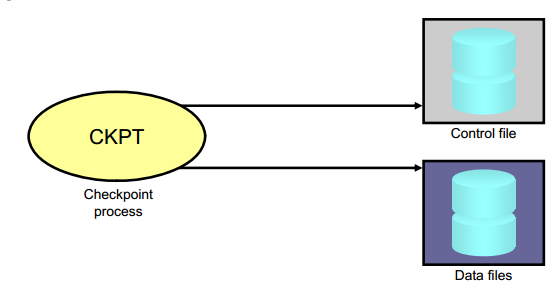
In all cases, DBWn performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system

### Log Writer Process (LGWR)



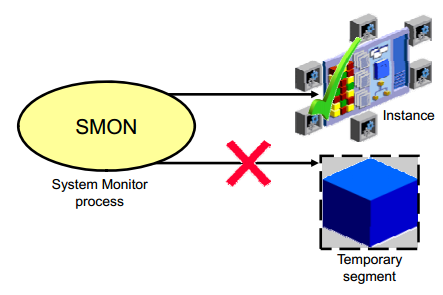
* + The Log Writer process (LGWR) is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.
  + LGWR starts and coordinates multiple helper processes that concurrently perform some of the work. LGWR handles the operations that are very fast, or must be coordinated, and delegates operations to the LGnn that could benefit from concurrent operations, primarily writing the redo from the log buffer to the redo log file and posting the completed write to the foreground process that is waiting.
  + Writes the redo log buffer to a redo log file on disk
    - When a user process commits a transaction
    - When an online redo log switch occurs
    - When the redo log buffer is one-third full or contains 1 MB of buffered data
    - Before a DBWn process writes modified buffers to disk
    - When three seconds have passed since the last write
  + Serves as coordinator of LGnn processes and ensures correct order for operations that must be ordered

### Checkpoint Process (CKPT)



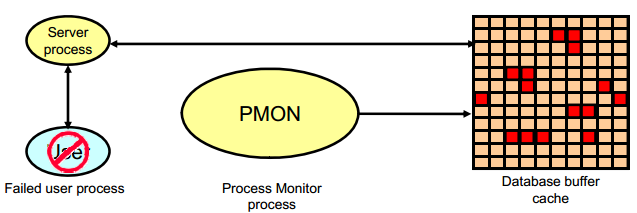
* + A checkpoint is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.
  + When a checkpoint occurs, Oracle Database must update the headers of all data files to record the details of the checkpoint. This is done by the CKPT process. And Signals DBWn to write blocks to disk

### System Monitor Process (SMON)



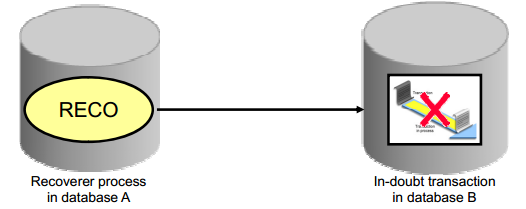
* + The System Monitor process (SMON) performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online

### Process Monitor Process (PMON)



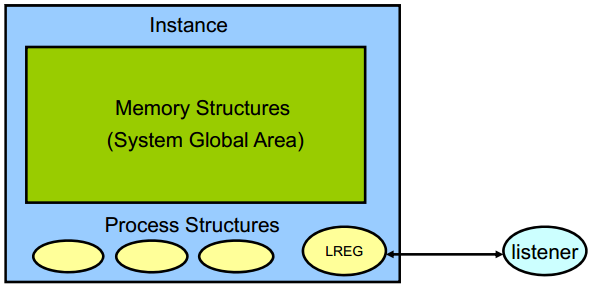
* + The Process Monitor process (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using
  + PMON periodically checks the status of dispatcher and server processes, and restarts any that have stopped running (but not any that Oracle Database has terminated intentionally).

### Recoverer Process (RECO)



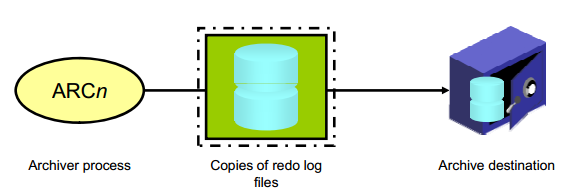
* + The Recoverer process (RECO) is a background process that is used with the distribute database configuration that automatically resolves failures involving distributed transactions
  + Automatically connects to other databases involved in indoubt distributed transactions
  + Automatically resolves all in-doubt transactions
  + Removes any rows that correspond to in-doubt transactions

### Listener Registration Process (LREG)



* + The Listener Registration process, LREG, registers information about the database instance and dispatcher processes with the Oracle Net Listener. LREG provides the listener with the following information:
    - Names of the database services
    - Name of the database instance associated with the services and its current and maximum load
    - Service handlers available for the instance, including their type, protocol addresses, and current and maximum load

### Archiver Processes (ARC*n*)



* + The Archiver processes (ARCn) copy redo log files to a designated storage device after a log switch has occurred. ARCn processes are present only when the database is in ARCHIVELOG mode and automatic archiving is enabled.
  + Can collect transaction redo data and transmit that data to standby destinations

# Database management new feature

## **Security**

### Auditing

Auditing, which means capturing and storing information about what is happening in the system, increases the amount of work the system must do. Auditing must be focused so that only events that are of interest are captured. Properly focused auditing has minimal impact on system performance. Improperly focused auditing can significantly affect performance.

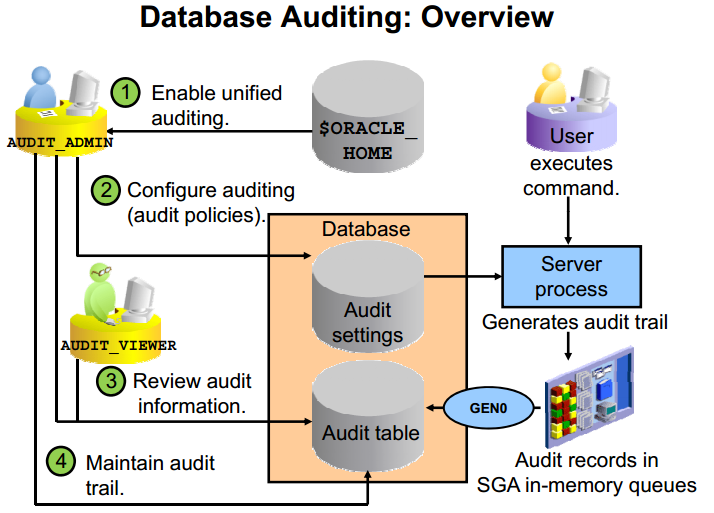
* **Mandatory auditing**: All Oracle databases audit certain actions regardless of other audit options or parameters. The reason for mandatory audit logs is that the database needs to record some database activities, such as connections by privileged users.
* **Standard database auditing**: Select the objects and privileges that you want to audit and create the appropriate audit policies.
* **Value-based auditing**: Extends standard database auditing, capturing not only the audited event that occurred but also the actual values that were inserted, updated, or deleted. Value-based auditing is implemented through database triggers.
* **Fine-grained auditing (FGA)**: Extends standard database auditing, capturing the actual SQL statement that was issued rather than only the fact that the event occurred.

Through the use of auditing policies, you can configure audit settings for the following activities:

* Logging on to the database and the use of privileges and roles
* Executing SQL statements against specific database objects
* Application context values
  + Utilities and features:
  + Oracle Data Pump
  + Oracle Database Real Application Security
  + Oracle Database Vault
  + Oracle Label Security
  + Oracle Recovery Manager
  + Oracle SQL\*Loader Direct Load

**Unified auditing**

**New feature**



Prior to Oracle Database 12c, audit records from various sources were stored in different locations. Oracle Database 12c supports unified auditing, in which all audit records are stored in a single audit table.

When you create a new Oracle Database 12c database, mixed mode auditing is enabled. This mode enables you to use the auditing features available before Oracle Database 12c and also the unified auditing features. Mixed mode auditing is enabled by default through the ORA\_SECURECONFIG predefined auditing policy for newly created databases.

A user must be granted one of the following roles to perform auditing:

* AUDIT\_ADMIN enables the user to:
  + Create unified and fine-grained audit policies
  + Execute the AUDIT and NOAUDIT SQL statements
  + View audit data
  + Manage the audit trail (table in the AUDSYS schema)
* AUDIT\_VIEWER enables the user to:
  + View and analyze audit data

**Unified Audit Implementation**

* Mixed auditing mode:
  + Allows smooth migration of existing databases to use the unified auditing features (pre 12c)
* Unified auditing mode:

Before populating the UNIFIED\_AUDIT\_TRAIL view:

* Enable unified auditing.
  + Shut down all processes and database instances.
  + cd $ORACLE\_HOME/rdbms/lib

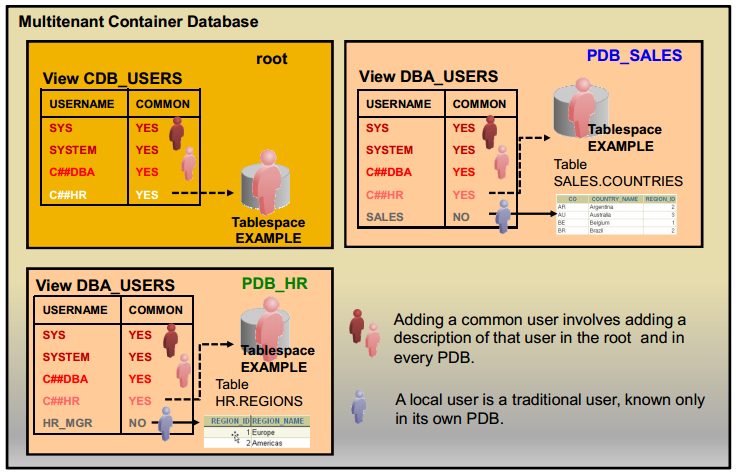
make -f ins\_rdbms.mk uniaud\_on ioracle ORACLE\_HOME=$ORACLE\_HOME

* + Start all Oracle processes of all instances.

Define a tablespace for the read-only audit table.

### Privileges

**User**



Local User: A user in a non-CDB maps to a local user in a PDB.

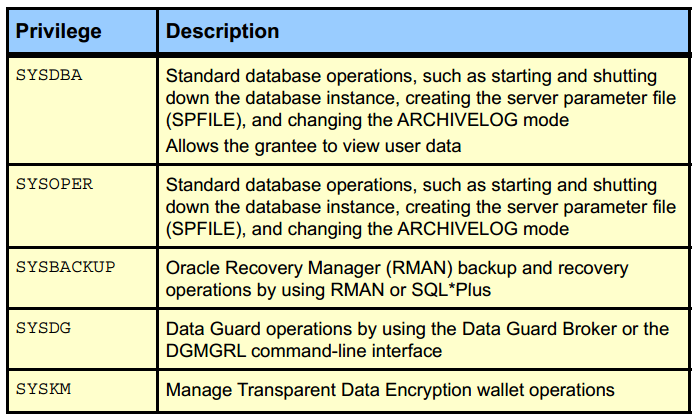
* + A local user is defined in the PDB’s own data dictionary—and so is not known outside of that PDB.
  + A local user can connect only to the PDB where it is defined.
  + A local user is specific to a particular PDB and owns a schema in this PDB.
  + According to the privileges granted, a user can work on the application data within the PDB or with other PDBs’ application using database links. Moreover, there cannot be any local users defined in the root.

Common User

* + A common user is defined in the root’s data dictionary.
  + Only common users can be defined in the root: Creating a common user allows the CDB administrator to create at once a user that is replicated in each PDB.
  + A common user is known, not only where it is defined in the root, but also in every PDB that belongs to the CDB.
  + A common user can perform administrative tasks specific to the root or PDBs, such as plugging and unplugging PDBs, starting up the CDB, or opening a PDB when granted the proper privileges.

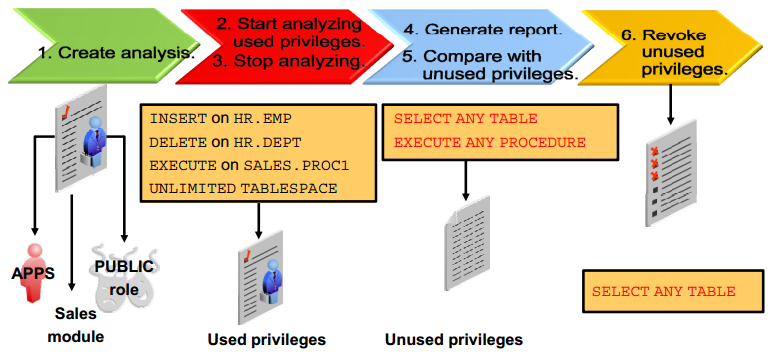
**Admin Account**

* + SYS:
    - Owns the data dictionary and the Automatic Workload Repository (AWR)
    - Used for startup and shutdown of the database instance
  + SYSTEM: Owns additional administrative tables and views
  + SYSBACKUP: Facilitates Oracle Recovery Manager (RMAN) backup and recovery operations
  + SYSDG: Facilitates Oracle Data Guard operations
  + SYSKM: Facilitates Transparent Data Encryption walletoperations



**Privilege Analysis**

* + Analyze used privileges to revoke unnecessary privileges.
  + Use DBMS\_PRIVILEGE\_CAPTURE package.



**Secure Roles**

Roles are usually enabled by default, which means that if a role is granted to a user, then that user can exercise the privileges given to the role. Default roles are assigned to the user at connect time.

It is possible to:

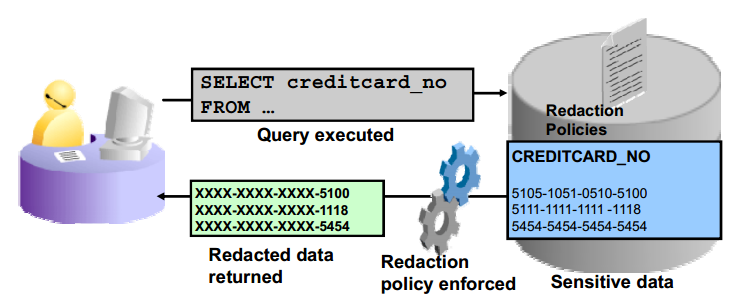
* + Make a role nondefault. The user must now explicitly enable the role before the role’s privileges can be exercised.
  + Have a role require additional authentication by using the IDENTIFIED clause to indicate that a user must be authorized by a specified method before the role is enabled with the SET ROLE statement. The default authentication for a role is None.
  + Create secure application roles that can be enabled only by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things such as the user’s network address, the program that the user is running, the time of day, and other elements needed to properly secure a group of permissions.
  + Administer roles easily using the Oracle Database Vault option. Secure application roles are simplified, and traditional roles can be further restricted.

SQL> SET ROLE vacationdba;

SQL> CREATE ROLE secure\_application\_role

IDENTIFIED USING <security\_procedure\_name>;

**Oracle Data Redaction**



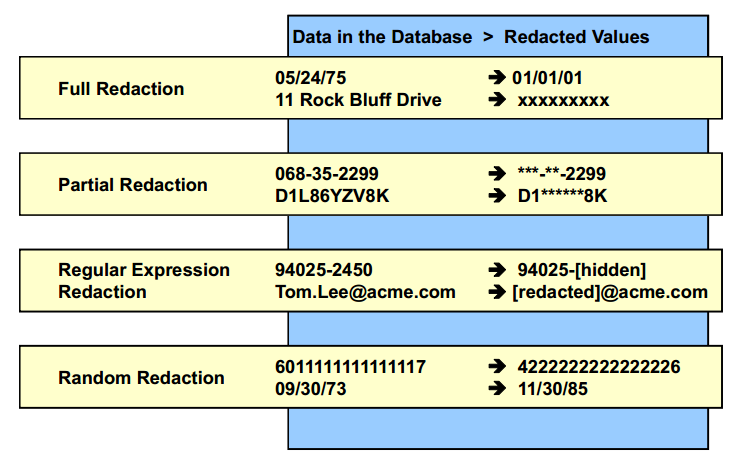
Currently, many Oracle Database customers prevent the display of sensitive data to end users by performing redacting in each application. Oracle Data Redaction moves this functionality from the application to the database. This approach has several benefits over redacting data in the application tier, and it is useful in a variety of application scenarios.

Oracle Data Redaction is a transparent, flexible, and simple solution. It modifies sensitive data columns contained in SQL query results dynamically right before the results are returned to applications. The columns are redacted according to flexible policies that provide conditional redaction. The policies are managed directly within the database. For maximum transparency, redaction preserves the returned column data type and formatting, and it does not alter the underlying data blocks on disk or in cache. Oracle Data Redaction is designed to be fast so that it can be used on production systems. In addition, it is embedded in the database management system; so no separate installation is required.

Operational activities that are not subject to redaction:

* + Backup and restore
  + Import and export
  + Patching and upgrades
  + Replication

Type of reduction: 4 type



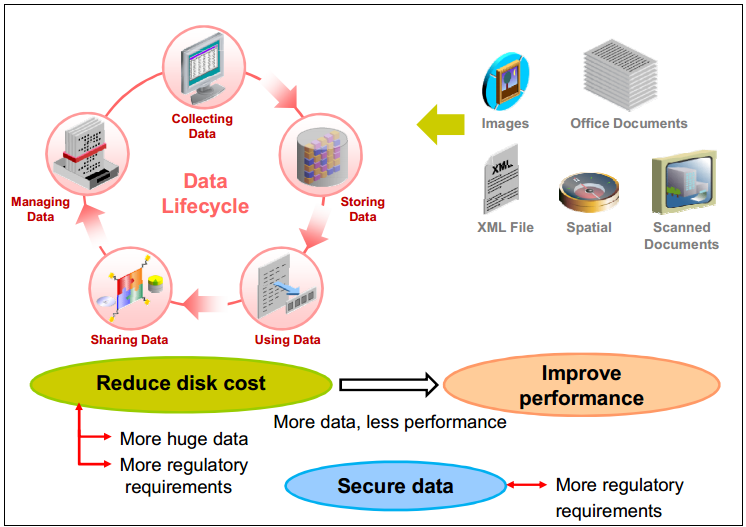
## **ADO and Storage**

### Information Lifecycle Management (ILM)

What is Information Lifecycle Management, also referred to as ILM? It is a strategy for managing business data over its lifetime to reduce storage costs, improve data access within the database, and adapt to regulatory requirements.

ILM is the practice of applying policies for the effective management of information throughout its useful life. ILM includes every phase of a “row.” It consists of the policies, processes, practices, and tools that are used to align the business value of information with the most appropriate and cost-effective IT infrastructure from the time information is conceived through its final disposition.

**ILM Challenges:**



Solution:

* + Automatic Data Optimization
  + In-Database Archiving

### **Automatic Data Optimization**

Oracle Database 12c includes activity tracking with Heat Map providing the ability to track and mark data as it goes through life cycle changes:

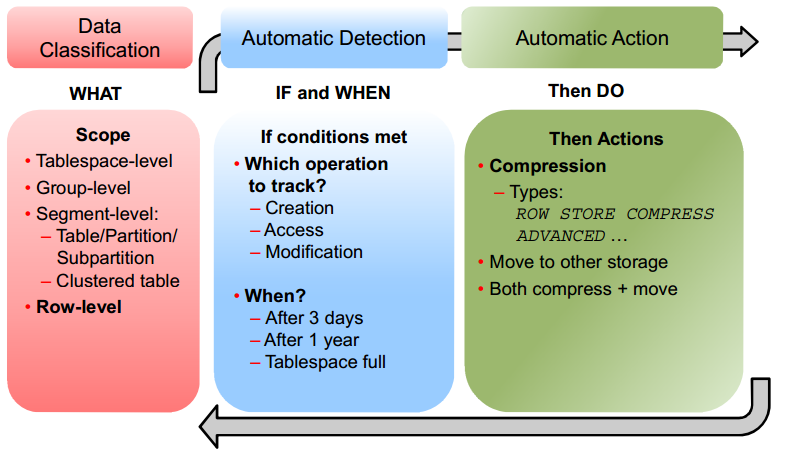
* + Data accesses at segment-level
  + Data modifications at block and segment levels

Block-level and segment-level statistics collected in memory are stored in tables in the SYSAUX tablespace.

ADO allows you to create policies that use Heat Map statistics to compress and move data only when necessary. ADO automatically evaluates and executes policies that perform compression and storage tiering actions ADO policies automatically execute actions under predefined conditions:

* + Compress data only when it qualifies.
  + Move segments to other storage tiers under space pressure.

ADO can execute compression and data movement only if Heat Map is enabled. After being enabled, Heat Map automatically collects statistics to execute ADO actions after the statistics evaluation is completed.



**Compression**

Compression Scope

* + Tablespace
  + Group (LOBs)
  + Segment
  + Row

How Does It Work?

Typical database compression algorithms compress repeating values found within rows of data stored in a database. So for example, all the order dates within an order row will be compressed. However, as the rows are stored on disk in row format, there is a lot of non relevant information stored between each occurrence of the next value of an order date: between each order date, additional values are found for order ID, customer ID, product IDs, and so on.

Hybrid Columnar Compression uses a different technique to store the column values. Instead of storing in a row format, the data is effectively stored by column: for instance, all the order dates will be stored together, then all the order IDs, then all the customer IDs, and so on. This means that within a unit of compression, a much higher rate of repeating values is found, and a greater compression ratio can be achieved. Each compression unit can span multiple data blocks. The values for a particular column may or may not span multiple blocks. The compression is no longer bound to one data block.

Compression Type

* + ROW STORE COMPRESS BASIC or ADVANCED is used for rows inserted without using direct-path insert and updated rows, using the Advanced Compression option (ACO).
  + COLUMN STORE COMPRESS FOR QUERY LOW or HIGH provides a higher level of compression than ROW STORE compression. It works well when load performance is critical, frequent queries are run against the table, and no normal DML is expected.
  + COLUMN STORE COMPRESS FOR ARCHIVE LOW/ARCHIVE HIGH compression provides the highest level of compression and works well for infrequently accessed data, mostly for read-only data. It enables HCC. COLUMN STORE COMPRESS FOR ARCHIVE LOW/ARCHIVE HIGH on a heap table maps to MEDIUM for SecureFiles LOB segments.

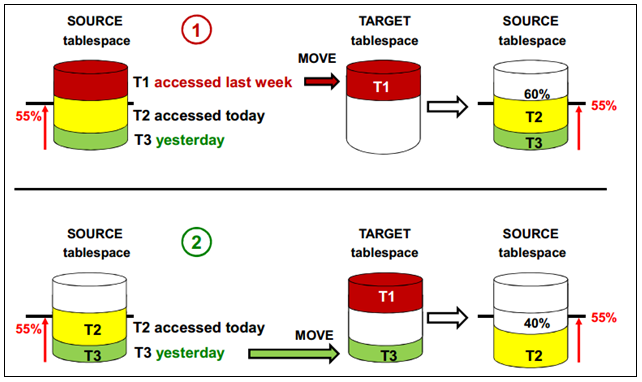
**Storage Tiering**

The second possible action is data movement to another storage tier, whether it is a lower cost storage tier, or a higher-performance storage tier with other compression capabilities such as HCC. The only possible scope for data movement is SEGMENT.

The tiering fullness threshold of the source tablespace depends on two parameters:

* + Objects with tiering policy will be moved if the tablespace they reside in becomes TBS PERCENT USED full (defaulted to 85).
  + Objects will be moved to the target tablespace until the source tablespace becomes TBS PERCENT FREE free (defaulted to 25).

Priority:



**Execution**

Segment-level policy

* + In maintenance window

Row-level policy

* + Regularly by MMON every 15 minutes

### In-Database Archiving

**How Does It Work?**

Typical database compression algorithms compress repeating values found within rows of data stored in a database. So for example, all the order dates within an order row will be compressed. However, as the rows are stored on disk in row format, there is a lot of non relevant information stored between each occurrence of the next value of an order date: between each order date, additional values are found for order ID, customer ID, product IDs, and so on.

Hybrid Columnar Compression uses a different technique to store the column values. Instead of storing in a row format, the data is effectively stored by column: for instance, all the order dates will be stored together, then all the order IDs, then all the customer IDs, and so on. This means that within a unit of compression, a much higher rate of repeating values is found, and a greater compression ratio can be achieved. Each compression unit can span multiple data blocks. The values for a particular column may or may not span multiple blocks. The compression is no longer bound to one data block.

**In-Database Archiving**

Tạo ra các bảng có khả năng nén dữ liệu

* + Default các row insert vào sẽ không được nén
  + Sau 1 thời gian, người sử dụng chủ động nén lại bằng câu lệnh nén, dữ liệu nén sẽ invisible với các truy vấn thông thường

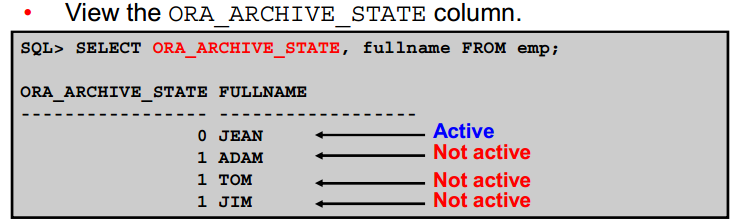
SQL> CREATE TABLE emp  
(EMPNO NUMBER(7), FULLNAME VARCHAR2(40),  
JOB VARCHAR2(9), MGR NUMBER(7))  
ROW ARCHIVAL;

* + Set rows in archive state

SQL> UPDATE emp SET ORA\_ARCHIVE\_STATE = 1  
WHERE empno < 100;

* + Set rows back in active state

SQL> UPDATE emp SET ORA\_ARCHIVE\_STATE = 0;



**Temporal Validity**

* + Sử dụng thêm 2 cột phụ để tính toán thời gian valid của dự liệu. Ngoài thời gian trên, khi thực thi lệnh valid, dữ liệu sẽ được nén lại, tích kiệm không gian lưu trữ
  + Tạo bảng:

SQL> CREATE TABLE emp  
( empno number, salary number, deptid number,  
name VARCHAR2(100),  
user\_time\_start DATE, user\_time\_end DATE,  
PERIOD FOR user\_time (user\_time\_start,user\_time\_end));

* + Enable tính năng valid:

SQL> exec DBMS\_FLASHBACK\_ARCHIVE.ENABLE\_AT\_VALID\_TIME('CURRENT')

* + Disable tính năng valid:

SQL> exec DBMS\_FLASHBACK\_ARCHIVE.ENABLE\_AT\_VALID\_TIME('ALL')

### **Advanced Index Compression**

Prior to this release, the only form of advanced index compression was low compression. Now you can also specify high compression. High compression provides even more space savings than low compression.

### **TDE Tablespace Live Conversion:**

You can now encrypt, decrypt, and rekey existing tablespaces with Transparent Data Encryption (TDE) tablespace live conversion. A TDE tablespace can be easily deployed, performing the initial encryption that migrates to an encrypted tablespace with zero downtime. This feature also enables automated deep rotation of data encryption keys used by TDE tablespace encryption in the background with zero downtime.

### **Fully Encrypted Database:**

Transparent Data Encryption (TDE) tablespace encryption is applied to database internals including SYSTEM, SYSAUX, and UNDO.

### **TDE Tablespace Offline Conversion:**

This release introduces new SQL commands to encrypt tablespace files in place with no storage overhead. You can do this on multiple instances across multiple cores. Using this feature requires downtime, because you must take the tablespace temporarily offline. With Data Guard configurations, you can either encrypt the physical standby first and switchover, or encrypt the primary database, one tablespace at a time.

## Oracle Database Admin New feature

### **Command history for SQL \* Plus:**

Pre 12cR2, this could be achieved through a workaround, now, the history command would do the magic for you.

### Online rename and relocation of an active data file

Unlike in the previous releases, a data file migration or renaming in Oracle database 12c R1 no longer requires a number of steps i.e. putting the tablespace in READ ONLY mode, followed by data file offline action. In 12c R1, a data file can be renamed or moved online simply using the ALTER DATABASE MOVE DATAFILE SQL statement. While the data file is being transferred, the end user can perform queries, DML and DDL tasks. Additionally, data files can be migrated between storages e.g. from non-ASM to ASM and vice versa.

* Rename a data file:

SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users01.dbf' TO '/u00/data/users\_01.dbf';

* Migrate a data file from non-ASM to ASM:

SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users\_01.dbf' TO '+DG\_DATA';

* Migrate a data file from one ASM disk group to another:

SQL> ALTER DATABASE MOVE DATAFILE '+DG\_DATA/DBNAME/DATAFILE/users\_01.dbf' TO '+DG\_DATA\_02';

* Overwrite the data file with the same name, if it exists at the new location:

SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users\_01.dbf' TO '/u00/data\_new/users\_01.dbf' REUSE;

* Copy the file to a new location whilst retaining the old copy in the old location:

SQL> ALTER DATABASE MOVE DATAFILE '/u00/data/users\_01.dbf' TO '/u00/data\_new/users\_01.dbf' KEEP;

You can monitor the progress while a data file being moved by querying the **v$session\_longops** dynamic view. Additionally, you can also refer the alert.log of the database where Oracle writes the details about action being taken place.

### DDL logging

There was no direction option available to log the DDL action in the previous releases. In 12cR1, you can now log the DDL action into xml and log files. This will be very useful to know when the drop or create command was executed and by who. The ENABLE\_DDL\_LOGGING initiation parameter must be configured in order to turn on this feature. The parameter can be set at the database or session levels. When this parameter is enabled, all DDL commands are logged in an xml and a log file under the $ORACLE\_BASE/diag/rdbms/DBNAME/log|ddl location. An xml file contains information, such as DDL command, IP address, timestamp etc. This helps to identify when a user or table dropped or when a DDL statement is triggered.

To enable DDL logging

SQL> ALTER SYSTEM|SESSION SET ENABLE\_DDL\_LOGGING=TRUE;

The following DDL statements are likely to be recorded in the xml/log file:

CREATE|ALTER|DROP|TRUNCATE TABLE

DROP USER

CREATE|ALTER|DROP PACKAGE|FUNCTION|VIEW|SYNONYM|SEQUENCE

### Temporary Undo

Each Oracle database contains a set of system related tablespaces, such as, SYSTEM, SYSAUX, UNDO & TEMP, and each are used for different purposes within the Oracle database. Pre Oracle 12c R1, undo records generated by the temporary tables used to be stored in undo tablespace, much similar to a general/persistent table undo records. However, with the temporary undo feature in 12c R1, the temporary undo records can now be stored in a temporary table instead of stored in undo tablespace. The prime benefits of temporary undo includes: reduction in undo tablespace and less redo data generation as the information won’t be logged in redo logs. You have the flexibility to enable the temporary undo option either at session level or database level.

Enabling temporary undo

To be able to use the new feature, the following needs to be set:

* Compatibility parameter must be set to 12.0.0 or higher
* Enable TEMP\_UNDO\_ENABLED initialization parameter
* Since the temporary undo records now stored in a temp tablespace, you need to create the temporary tablespace with sufficient spaceFor session level, you can use: ALTER SESSION SET TEMP\_UNDO\_ENABLE=TRUE;

Query temporary undo information

The dictionary views listed below are used to view/query the information/statistics about the temporary undo data:

* V$TEMPUNDOSTAT
* DBA\_HIST\_UNDOSTAT
* V$UNDOSTAT

To disable the feature, you simply need to set the following:

SQL> ALTER SYSTEM|SESSION SET TEMP\_UNDO\_ENABLED=FALSE;

### Restricting PGA size

Pre Oracle 12c R1, there was no option to limit and control the PGA size. Although, you set a certain size to PGA\_AGGREGATE\_TARGET initialization parameter, Oracle could increase/reduce the size of the PGA dynamically based on the workload and requirements. In 12c, you can set a hard limit on PGA by enabling the automatic PGA management, which requires PGA\_AGGREGATE\_LIMIT parameter settings. Therefore, you can now set the hard limit on PGA by setting the new parameter to avoid excessive PGA usage.

SQL> ALTER SYSTEM SET PGA\_AGGREGATE\_LIMIT=2G;

SQL> ALTER SYSTEM SET PGA\_AGGREGATE\_LIMIT=0; --disables the hard limit

### Database upgrade improvements

Whenever a new Oracle version is announced, the immediate challenge that every DBA confronts is the upgrade process. In this section, I will explain the two new improvements introduced for upgrading to 12c.

**Pre-upgrade script**

SQL> @$ORACLE\_12GHOME/rdbms/admin/preupgrd.sql

The above script generates a log file and a [pre/post]upgrade\_fixup.sql script. All these files are located under the $ORACLE\_BASE/cfgtoollogs directory. Before you continue with the real upgrade procedure, you should run through the recommendations mentioned in the log file and execute the scripts to fix any issues.

Note: Ensure you copy the preupgrd.sql and utluppkg.sql scripts from the 12c Oracle home/rdbms/admin directory to the current Oracle database/rdbms/admin location.

**Parallel-upgrade utility**

The database upgrade duration is directly proportional to the number of components that are configured on the database, rather than the database size. In previous releases, there was no direct option or workaround available to run the upgrade process in parallel to quickly complete the overall upgrade procedure.

The catctl.pl (parallel-upgrade utility) that replaces the legacy catupgrd.sql script in 12c R1 comes with an option to run the upgrade procedure in parallel mode to improve the overall duration required to complete the procedure.

The following procedure explains how to initiate the parallel (with 3 processes) upgrade utility; you need to run this after you STARTUP the database in UPGRADE mode:

cd $ORACLE\_12\_HOME/perl/bin

$ ./perl catctl.pl –n 3 -catupgrd.sql

The above two steps need to be run explicitly when a database is upgraded manually. However, the **DBUA** inherits the both new changes.

### Real-time ADDM analysis

Analyzing past and current database health statuses through a set of automatic diagnostic tools such as AWR, ASH and ADDM is part of every DBAs life. Though each individual tool can be used at various levels to measure the database’s overall heath and performance, no tool can be used when the database is unresponsive or totally hung.

When you encounter an unresponsive database or hung state, and if you have configured Oracle Enterprise Manager 12c Cloud Control, you can diagnose serious performance issues. This would give you a good picture about what’s currently going on in the database, and might also provide a remedy to resolve the issue.

The following step-by-step procedure demonstrates how to analyze the situation on the Oracle EM 12c Cloud Control :

* Select the Emergency Monitoring option from the Performance menu on the Access the Database Home page.This will show the top blocking sessions in the Hang Analysis table.
* Select the Real-Time ADDM option from the Performance to perform Real-time ADDM analysis.
* After collecting the performance data, click on the Findings tab to get the interactive summary of all the findings.

### Gathering statistics concurrently on multiple tables

In previous Oracle database editions, whenever you execute a DBMS\_STATS procedure to gather table, index, schema or database level statistics, Oracle used to collect stats one table at a time. If the table is big enough, then increasing the parallelism was recommended. With 12c R1, you can now collect stats on multiple tables, partitions and sub partitions concurrently. Before you start using it, you must set the following at the database level to enable the feature:

SQL> ALTER SYSTEM SET RESOURCE\_MANAGER\_PLAN='DEFAULT\_MAIN';

SQL> ALTER SYSTEM SET JOB\_QUEUE\_PROCESSES=4;

SQL> EXEC DBMS\_STATS.SET\_GLOBAL\_PREFS('CONCURRENT', 'ALL');

SQL> EXEC DBMS\_STATS.GATHER\_SCHEMA\_STATS('SCOTT');

### Enhanced Rapid Home Provisioning and Patch Management

## DataGuard

### Oracle Data Guard Database Compare:

This new tool compares data blocks stored in an Oracle Data Guard primary database and its physical standby databases. Use this tool to find disk errors (such as lost write) that cannot be detected by other tools like the DBVERIFY utility.

### Subset Standby:

A subset standby enables users of Oracle Multitenant to designate a subset of the pluggable databases (PDBs) in a multitenant container database (CDB) for replication to a standby database.

### Automatically Synchronize Password Files in Oracle Data Guard Configurations:

This feature automatically synchronizes password files across Oracle Data Guard configurations. When the passwords of SYS, SYSDG, and so on, are changed, the password file at the primary database is updated and then the changes are propagated to all standby databases in the configuration.

### Preserving Application Connections to An Active Data Guard Standby During Role Changes:

Currently, when a role change occurs and an Active Data Guard standby becomes the primary, all read-only user connections are disconnected and must reconnect, losing their state information. This feature enables a role change to occur without disconnecting the read-only user connections. Instead, the read-only user connections experience a pause while the state of the standby database is changed to primary. Read-only user connections that use a service designed to run in both the primary and physical standby roles are maintained. Users connected through a physical standby only role continue to be disconnected.

Oracle Data Guard for Data Warehouses:

The use of NOLOGGING for direct loads on a primary database has always been difficult to correct on an associated standby database. On a physical standby database the data blocks were marked unrecoverable and any SQL operation that tried to read them would return an error. Or, for a logical standby database, SQL apply would stop upon encountering the invalidation redo.

### Read/Write and Read-Only Instances

Read-write and read-only database instances of the same primary database can coexist in an Oracle Flex Cluster.

## Table and table partition maintenance enhancements

### Invisible columns

In Oracle 11g R1, Oracle introduced a couple of good enhancements in the form of invisible indexes and virtual columns. Taking the legacy forward, invisible column concepts has been introduced in Oracle 12c R1. I still remember, in the previous releases, to hide important data –columns from being displayed in the generic queries– we used to create a view hiding the required information or apply some sort of security conditions.

In 12c R1, you can now have an invisible column in a table. When a column is defined as invisible, the column won’t appear in generic queries, unless the column is explicitly referred to in the SQL statement or condition, or DESCRIBED in the table definition. It is pretty easy to add or modify a column to be invisible and vice versa:

SQL> CREATE TABLE emp (eno number(6), ename name varchar2(40), sal number(9) INVISIBLE);

SQL> ALTER TABLE emp MODIFY (sal visible);

You must explicitly refer to the invisible column name with the INSERT statement to insert the database into invisible columns. A virtual column or partition column can be defined as invisible too. However, temporary tables, external tables and cluster tables won’t support invisible columns.

### Multiple indexes on the same column

Pre Oracle 12c, you can’t create multiple indexes either on the same column or set of columns in any form. For example, if you have an index on column {a} or columns {a,b}, you can’t create another index on the same column or set of columns in the same order. In 12c, you can have multiple indexes on the same column or set of columns as long as the index type is different. However, only one type of index is usable/visible at a given time. In order to test the invisible indexes, you need to set the optimizer\_use\_use\_invisible\_indexes=true.

Here’s an the example:

SQL> CREATE INDEX emp\_ind1 ON EMP(ENO,ENAME);

SQL> CREATE BITMAP INDEX emp\_ind2 ON EMP(ENO,ENAME) INVISIBLE;

### Adding multiple new partitions

Before Oracle 12c R1, it was only possible to add one new partition at a time to an existing partitioned table. To add more than one new partition, you had to execute an individual ALTER TABLE ADD PARTITION statement to every new partition. Oracle 12c provides the flexibility to add multiple new partitions using a single ALTER TABLE ADD PARTITION command. The following example explains how to add multiple new partitions to an existing partitioned table:

SQL> CREATE TABLE emp\_part

(eno number(8), ename varchar2(40), sal number (6))

PARTITION BY RANGE (sal)

(PARTITION p1 VALUES LESS THAN (10000),

PARTITION p2 VALUES LESS THAN (20000),

PARTITION p3 VALUES LESS THAN (30000)

);

* Now lets add a couple of new partitions:

SQL> ALTER TABLE emp\_part ADD PARTITION

PARTITION p4 VALUES LESS THAN (35000),

PARTITION p5 VALUES LESS THAN (40000);

In the same way, you can add multiple new partitions to a list and system partitioned table, provided that the MAXVALUE partition doesn’t exist.

### Online migration of table partition or sub-partition

Migration of a table partition or sub-partition to a different tablespace no longer requires a complex procedure in Oracle 12c R1. In a similar way to how a heap (non-partition) table online migration was achieved in the previous releases, a table partition or sub-partition can be moved to a different tablespace online or offline. When an ONLINE clause is specified, all DML operations can be performed without any interruption on the partition|sub-partition which is involved in the procedure. In contrast, no DML operations are allowed if the partition|sub-partition is moved offline.

Here are some working examples:

SQL> ALTER TABLE table\_name MOVE PARTITION|SUBPARTITION partition\_name TO tablespace tablespace\_name;

SQL> ALTER TABLE table\_name MOVE PARTITION|SUBPARTITION partition\_name TO tablespace tablespace\_name UPDATE INDEXES ONLINE;

The first example is used to move a table partition|sub-partition to a new tablespace offline. The second example moves a table partition/sub-partitioning online maintaining any local/global indexes on the table. Additionally, no DML operation will get interrupted when ONLINE clause is mentioned.

Important notes:

* The UPDATE INDEXES clause will avoid any local/global indexes going unusable on the table.
* Table online migration restriction applies here too.
* There will be locking mechanism involved to complete the procedure, also it might leads to performance degradation and can generate huge redo, depending upon the size of the partition, sub-partition.

### Online Conversion of a Nonpartitioned Table to a Partitioned Table

Nonpartitioned tables can be converted to partitioned tables online. Indexes are maintained as part of this operation and can be partitioned as well. The conversion has no impact on the ongoing DML operations.

### Online SPLIT Partition and Subpartition

The partition maintenance operations SPLIT PARTITION and SPLIT SUBPARTITION can now be executed as online operations for heap organized tables, allowing the concurrent DML operations with the ongoing partition maintenance operation.

### Online Table Move

Nonpartitioned tables can be moved as an online operation without blocking any concurrent DML operations. A table move operation now also supports automatic index maintenance as part of the move.

### Oracle Database Sharding

Sharding with Oracle Database 12c Release 2 (12.2) is an architecture for suitable online transaction processing (OLTP) applications where data is horizontally partitioned across multiple discrete Oracle databases, called shards, which share no hardware or software. The collection of shards is presented to an application as a single logical Oracle database.

### How to drop and truncate multiple partitions/sub-partitions

As part of data maintenance, you typically either use drop or truncate partition maintenance task on a partitioned table. Pre 12c R1, it was only possible to drop or truncate one partition at a time on an existing partitioned table. With Oracle 12c, multiple partitions or sub-partitions can be dropped or merged using a single ALTER TABLE table\_name {DROP|TRUNCATE} PARTITIONS command.

* The following example explains how to drop or truncate multiple partitions on an existing partitioned table:

SQL> ALTER TABLE emp\_part DROP PARTITIONS p4,p5;

SQL> ALTER TABLE emp\_part TRUNCATE PARTITONS p4,p5;

* To keep indexes up-to-date, use the UPDATE INDEXES or UPDATE GLOBAL INDEXES clause, shown below:

SQL> ALTER TABLE emp\_part DROP PARTITIONS p4,p5 UPDATE GLOBAL INDEXES;

SQL> ALTER TABLE emp\_part TRUNCATE PARTITIONS p4,p5 UPDATE GLOBAL INDEXES;

If you truncate or drop a partition without the UPDATE GLOBAL INDEXES clause, you can query the column ORPHANED\_ENTRIES in the USER\_INDEXES or USER\_IND\_PARTITIONS dictionary views to find out whether the index contains any stale entries.

### Splitting a single partition into multiple new partitions

The new enhanced SPLIT PARTITION clause in 12c will let you split a particular partition or sub-partition into multiple new partitions using a single command. The following example explains how to split a partition into multiple new partitions:

SQL> CREATE TABLE emp\_part

(eno number(8), ename varchar2(40), sal number (6))

PARTITION BY RANGE (sal)

(PARTITION p1 VALUES LESS THAN (10000),

PARTITION p2 VALUES LESS THAN (20000),

PARTITION p\_max VALUES LESS THAN (MAXVALUE)

);

SQL> ALTER TABLE emp\_part SPLIT PARTITION p\_max INTO

(PARTITION p3 VALUES LESS THAN (25000),

PARTITION p4 VALUES LESS THAN (30000), PARTITION p\_max);

### Merge multiple partitions into one partition

You can merge multiple partitions to a single partition using a single ALTER TBALE MERGE PARTITIONS statement:

SQL> CREATE TABLE emp\_part

(eno number(8), ename varchar2(40), sal number (6))

PARTITION BY RANGE (sal)

(PARTITION p1 VALUES LESS THAN (10000),

PARTITION p2 VALUES LESS THAN (20000),

PARTITION p3 VALUES LESS THAN (30000),

PARTITION p4 VALUES LESS THAN (40000),

PARTITION p5 VALUES LESS THAN (50000),

PARTITION p\_max (MAXVALUE)

);

SQL> ALTER TABLE emp\_part MERGE PARTITIONS p3,p4,p5 INTO PARTITION p\_merge;

If the range falls in the sequence, you can use the following example:

SQL> ALTER TABLE emp\_part MERGE PARTITIONS p3 TO p5 INTO PARTITION p\_merge;

### Rolling Back Redefinition

There is a new ROLLBACK parameter for the FINISH\_REDEF\_TABLE procedure that tracks DML on a newly redefined table so that changes can be easily synchronized with the original table using the SYNC\_INTERIM\_TABLE procedure.

The new V$ONLINE\_REDEF view displays runtime information related to the current redefinition procedure being executed based on a redefinition session identifier.

## Data Pump enhancements

This part of the section will focus on the important enhancements introduced in data pumps. There are quite a few useful additions, such as converting view into a table while exporting and turning off logging while import.

### Turn off redo log generation

The new TRANSFORM option introduced in data pumps import provides the flexibility to turn off the redo generation for the objects during the course of import. When DISABLE\_ARCHIVE\_LOGGING values is specified with the TRANSFORM option, redo generation for the objects in the context will be turned off during the entire import duration. This feature provides a great relief when importing large tables, and reduces the excessive redo generation, which results in quicker imports. This attribute applies to tables and indexes.

This example demonstrates this feature:

$ ./impdp directory=dpump dumpfile=abcd.dmp logfile=abcd.log TRANSFORM=DISABLE\_ARCHIVE\_LOGGING:Y

### Transport view as table

This is another improvement in the data pumps. With the new VIEWS\_AS\_TABLES option, you can unload the view data into a table. The following example describes how to unload views data into a table during export:

$ ./expdp directory=dpump dumpfile=abcd.dmp logfile=abcd.log views\_as\_tables=my\_view:my\_table

### Renaming Data Files During Import

## Useful Enchanment to developers

### Truncate table CASCADE

In the previous releases, there wasn’t a direct option provided to truncate a master table while it is referred to by the child tables and child records exist. The TRUNCATE TABLE with CASCADE option in 12c truncates the records in the master table and automatically initiates recursive truncate on child tables too, subject to foreign key reference as DELETE ON CASCADE. There is no CAP on the number of recursive levels as it will apply on all child, grand child and great grandchild etc.

This enhancement gets rid of the prerequisite to truncate all child records before truncating a master table. The new CASCADE clause can also be applied on table partitions and sub-partitions etc.

SQL> TRUNCATE TABLE <table\_name> CASCADE;

SQL> TRUNCATE TABLE <table\_name> PARTITION <partition\_name> CASCADE;

ORA-14705 error will be thrown if no ON DELETE CASCADE option is defined with the foreign keys of the child tables.

### Session level sequences

A new SESSION level database sequence can be created now in 12c to support the session level sequence values. These types of sequences are most useful and suitable on global temporary tables that have session level existence.

Session level sequences produce a unique range of values that are limited within the session, not across the sessions. Once the session ends, the state of the session sequences also goes away. The following example explains creating a session level sequence:

SQL> CREATE SEQUENCE my\_seq START WITH 1 INCREMENT BY 1 SESSION;

SQL> ALTER SEQUENCE my\_seq GLOBAL|SESSION;

The CACHE, NOCACHE, ORDER or NOORDER clauses are ignored for SESSION level sequences.

### ROW limiting for Top-N result queries

There are various indirect approaches/methods exist to fetch Top-N query results for top/bottom rows in the previous releases. In 12c, retrieving Top-N query results for top/bottom rows simplified and become straight forward with the new FETCH FIRST|NEXT|PERCENT clauses.

* In order to retrieve top 10 salaries from EMP table, use the following new SQL statement:

SQL> SELECT eno,ename,sal FROM emp ORDER BY SAL DESC

FETCH FIRST 10 ROWS ONLY;

* The following example fetches all similar records of Nth row. For example, if the 10th row has salary of 5000 value, and there are other employees whose salary matches with the Nth value, the will also be fetched upon mentioning WITH TIES clause.

SQL> SELECT eno,ename,sal FROM emp ORDER BY SAL DESC

FETCH FIRST 10 ROWS ONLY WITH TIES;

* The following example limits the fetch to 10 per cent from the top salaries in the EMP table:

SQL> SELECT eno,ename,sal FROM emp ORDER BY SAL DESC

FETCH FIRST 10 PERCENT ROWS ONLY;

* The following example offsets the first 5 rows and will display the next 5 rows from the table:

SQL> SELECT eno,ename,sal FROM emp ORDER BY SAL DESC

OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;

* All these limits can be very well used within the PL/SQL block too.

BEGIN

SELECT sal BULK COLLECT INTO sal\_v FROM EMP

FETCH FIRST 100 ROWS ONLY;

END;

### Miscellaneous SQL\*Plus enhancements

Implicit Results on SQL\*Plus: SQL\*Plus in 12c returns results from an implicit cursor of a PL/SQL block without actually binding it to a RefCursor. The new dbms\_sql.return\_result procedure will return and formats the results of SELECT statement query specified within PL/SQL block. The following code descries the usage:

SQL> CREATE PROCEDURE mp1 as res1 sys\_refcursor;

BEGIN

open res1 for SELECT eno,ename,sal FROM emp;

dbms\_sql.return\_result(res1);

END;

SQL> execute mp1;

When the procedure is executed, it return the formatted rows on the SQL\*Plus.

Display invisible columns: setting the following on the SQL\*Plus prompt:

SQL> SET COLINVISIBLE ON|OFF

The above setting is only valid for DESCRIBE command. It has not effect on the SELECT statement results on the invisible columns.

### In Line PL/SQL Functions in SQL - WITH clause improvements

In 12c, you can have faster running PL/SQL function/procedure in SQL, that are defined and declared within the WITH clause of SQL statements. The following examples demonstrate how to define and declare a procedure or function within the WITH clause:

WITH

PROCEDURE|FUNCTION test1 (…)

BEGIN

<logic>

END;

SELECT <referece\_your\_function|procedure\_here> FROM table\_name;

/

Although you can’t use the WITH clause directly in the PL/SQL unit, it can be referred through a dynamic SQL within that PL/SQL unit.

### Extended data types

In 12c, the data type VARCHAR2, NAVARCHAR2, and RAW size will support up to 32,767 bytes in contrast to 4,000 and 2,000 in the earlier releases. The extended character size will reduce the use of going for LOB data types, whenever possible. In order to enable the extended character size, you will have to set the MAX\_STRING\_SIZE initialization database parameter to EXTENDED.

Note: Once modified, you can’t change the settings back to STANDARD

### Generated as identity / Sequence Replacement

In old version of oracle database if you want to create automatic generated number you have to create sequence and use attribute nextval.

But with oracle database 12c this concept is changed new features add when you create table called generated as identity.

SQL> create table test (test\_id number generated as identity , test\_name varchar2(20));

SQL> desc test ;

Name Null? Type

----------------------------------------- -------- --------------

TEST\_ID NOT NULL NUMBER

TEST\_NAME VARCHAR2(20)

SQL> insert into test values (1,'osama');

insert into test values (1,'osama')

\*

ERROR at line 1:

ORA-32795: cannot insert into a generated always identity column

The TEST\_ID Column will be inserting automatically no need to use in insert command.

SQL> insert into test (TEST\_NAME) values ( 'Jennifer' );

1 row created.

SQL> select \* from test ;

TEST\_ID TEST\_NAME

---------- --------------------

1 Jennifer

### Materialized Views: Real-Time Materialized Views:

Materialized views can be used for query rewrite even if they are not fully synchronized with the base tables and are considered stale. Using materialized view logs for delta computation together with the stale materialized view, the database can compute the query and return correct results in real time.

For materialized views that can be used for query rewrite all of the time, with the accurate result being computed in real time, the result is optimized and fast query processing for best performance. This alleviates the stringent requirement of always having to have fresh materialized views for the best performance.

### Materialized Views: Statement-Level Refresh:

In addition to ON COMMIT and ON DEMAND refresh, the materialized join views can be refreshed when a DML operation takes place, without the need to commit such a transaction. This is predominantly relevant for star schema deployments.

The new ON STATEMENT refresh capability provides more flexibility to the application developers to take advantage of the materialized view rewrite, especially for complex transactions involving multiple DML statements. It offers built-in refresh capabilities that can replace customer-written trigger-based solutions, simplifying an application while offering higher performance.

## Additions/Enhancements in Automatic Storage Management (ASM)

### Flex ASM

In a typical Grid Infrastructure installation, each node will have its own ASM instance running and act the as the storage container for the databases running on the node. There is a single point-of-failure threat with this setup. For instance, if the ASM instance on the node suffers or fails all the databases and instances running on the node will be impacted. To avoid ASM instance single-point-failure, Oracle 12c provides a Flex ASM feature. The Flex ASM is a different concept and architecture all together. Only a fewer number of ASM Instances need to run on a group of servers in the cluster. When an ASM instance fails on a node, Oracle Clusterware automatically starts surviving (replacement) ASM instance on a different node to maintain availability. In addition, this setup also provides ASM instance load balancing capabilities for the instances running on the node. Another advantage of Flex ASM is that it can be configured on a separate node.

When you choose Flex Cluster option as part of the cluster installation, Flex ASM configuration will be automatically selected as it is required by the Flex Cluster. You can also have traditional cluster over Flex ASM. When you decide to use Flex ASM, you must ensure the required networks are available. You can choose the Flex ASM storage option as part of Cluster installation, or use ASMCA to enable Flex ASM in a standard cluster environment.

The following command shows the current ASM mode:

$ ./asmcmd showclustermode

$ ./srvctl config asm

Or connect to the ASM instances and query the INSTANCE\_TYPE parameter. If the output value is ASMPROX, then, the Flex ASM is configured.

### Increased ASM storage limits

The ASM storage hard limits on maximum ASM disk groups and disk size has been drastically increased. In 12cR1, ASM support 511 ASM disk groups against 63 ASM disk groups in 11gR2. Also, an ASM disk can be now 32PB size against 20PB in 11gR2.

### Tuning ASM rebalance operations

The new EXPLAIN WORK FOR statement in 12c measures the amount of work required for a given ASM rebalance operation and inputs the result in V$ASM\_ESTIMATE dynamic view. Using the dynamic view, you can adjust the POWER LIMIT clause to improve the rebalancing operation work. For example, if you want to measure the amount of work required for adding a new ASM disk, before actually running the manual rebalance operation, you can use the following:

SQL> EXPLAIN WORK FOR ALTER DISKGROUP DG\_DATA ADD DISK data\_005;

SQL> SELECT est\_work FROM V$ASM\_ESTIMATE;

SQL> EXPLAIN WORK SET STATEMENT\_ID='ADD\_DISK' FOR ALTER DISKGROUP DG\_DATA AD DISK data\_005;

SQL> SELECT est\_work FROM V$ASM\_ESTIMATE WHERE STATEMENT\_ID = 'ADD\_DISK’;

You can adjust the POWER limit based on the output you get from the dynamic view to improve the rebalancing operations.

### ASM Disk Scrubbing

The new ASM Disk Scrubbing operation on a ASM diskgroup with normal or high redundancy level, verifies the logical data corruption on all ASM disks of that ASM diskgroup, and repairs the logical corruption automatically, if detected, using the ASM mirror disks. The disk scrubbing can be performed at disk group, specified disk or on a file and the impact is very minimal. The following examples demonstrate the disk scrubbing scenario:

SQL> ALTER DISKGROUP dg\_data SCRUB POWER LOW:HIGH:AUTO:MAX;

SQL> ALTER DISKGROUP dg\_data SCRUB FILE '+DG\_DATA/MYDB/DATAFILE/filename.xxxx.xxxx'

REPAIR POWER AUTO;

### Active Session History (ASH) for ASM

The V$ACTIVE\_SESSION\_HISOTRY dynamic view now provides the active session sampling on ASM instance too. However, the use of diagnostic pack is subject to the license.

## Additions/Enhancements in Grid Infrastructure

### Flex Clusters

Oracle 12c support two types of cluster configuration at the time of Clusterware installation: Traditional Standard Cluster and Flex cluster. In a traditional standard cluster, all nodes in a cluster are tightly integrated to each other and interact through a private network and can access the storage directly. On the other hand, the Flex Cluster introduced two types of nodes arranged in Hub and Leaf nodes architecture. The nodes arranged in Hub nodes category are similar to the traditional standard cluster, i.e. they are interconnected to each other through a private network and have the directly storage read/write access. The Leaf nodes are different from the Hub nodes. They don’t need to have direct access to the underlying storage; rather they access the storage/data through Hub nodes.

You can configure Hub nodes up to 64, and Leaf nodes can be many. In an Oracle Flex Cluster, you can have Hub nodes without having Leaf nodes configured, but no Leaf nodes exist without Hub nodes. You can configure multiple Leaf nodes to a single Hub node. In Oracle Flex Cluster, only Hub nodes will have direct access to the OCR/Voting disks. When you plan large scale Cluster environments, this would be a great feature to use. This sort of setup greatly reduces interconnect traffic, provides room to scale up the cluster to the traditional standard cluster.

There are two ways to deploy the Flex Cluster:

* While configuring a brand new cluster
* Upgrade a standard cluster mode to Flex Cluster

If you are configuring a brand new cluster, you need to choose the type of cluster configuration during step 3, select Configure a Flex Cluster option and you will have to categorize the Hub and Leaf nodes on Step 6. Against each node, select the Role, Hub or Leaf, and optionally Virtual Hostname too.

The following steps are required to convert a standard cluster mode to Flex Cluster mode:

* Get the current status of the cluster using the following command:

$ ./crsctl get cluster mode status

* Run the following command as the root user:

$ ./crsctl set cluster mode flex

$ ./crsctl stop crs

$ ./crsctl start crs –wait

* Change the node role as per your design

$ ./crsctl get node role config

$ ./crsctl set node role hub|leaf

$ ./crsctl stop crs

$ ./crsctl start crs -wait

Note the following:

* You can’t revert back from Flex to Standard cluster mode
* Cluster node mode change requires cluster stack stop/start
* Ensure GNS is configured with a fixed VIP

### OCR backup in ASM disk group

With 12c, OCR can be now be backed-up in ASM disk group. This simplifies the access to the OCR backup files across all nodes. In case of OCR restore, you don’t need to worry about which node the OCR latest backup is on. One can simply identify the latest backup stored in the ASM from any node and can perform the restore easily.

The following example demonstrates how to set the ASM disk group as OCR backup location:

$ ./ocrconfig -backuploc +DG\_OCR

### IPv6 support

With Oracle 12c, Oracle now supports IPv4 and IPv6 network protocol configuration on the same network. You can now configure public network (Public/VIP) either on IPv4, IPv6 or combination protocol configuration. However, ensure you use the same set of IP protocol configuration across all nodes in a cluster.

## Additions/Enhancements in RAC (database)

### What-If command evaluation

Using the new What-if command evaluation (-eval) option with srvctl, one can now determine the impact of running the command. This new addition to the srvctl command, will let you simulate the command without it actually being executed or making any changes to the current system. This is particularly useful in a situation when you want to make a change to an existing system and you’re not sure of the outcome. Therefore, the command will provide the effect of making the change. The –eval option also can be used with crsctl command.

For example, if you want to know what will happen if you stop a particular database, you can use the following example:

$ ./srvctl stop database –d MYDB –eval

$ ./crsctl eval modify resource <resource\_name> -attr “value”

### Miscellaneous srvctl improvements

There are a few new additions to the srvctl command. The following demonstrates the new addition to stop/start database/instance resources on the cluster:

srvctl start database|instance –startoption NOMOUNT|MOUNT|OPEN

srvctl stop database|instance –stopoption NOMOUNT|MOUNT|OPEN

### Server Weight-Based Node Eviction :

Server weight-based node eviction acts as a tie-breaker mechanism in situations where Oracle Clusterware needs to evict a particular node or a group of nodes from a cluster, in which all nodes represent an equal choice for eviction. In such cases, the server weight-based node eviction mechanism helps to identify the node or the group of nodes to be evicted based on additional information about the load on those servers. Two principle mechanisms, a system inherent automatic mechanism and a user input-based mechanism exist to provide respective guidance.

### Load-Aware Resource Placement :

Load-aware resource placement prevents overloading a server with more applications than the server is capable of running. The metrics used to determine whether an application can be started on a given server, either as part of the startup or as a result of a failover, are based on the anticipated resource consumption of the application as well as the capacity of the server in terms of CPU and memory.

# RMAN new feauture

## SYSBACKUP Privilege

Prior to 12c, users needed SYSDBA privilege to backup the database. The new SYSBACKUP privilege allows the user the permissions to perform only backup operations.

The SYSBACKUP privilege allows the DBA to perform RMAN backup commands without additional privileges. Using this new role in 12c, you can segregate Administration and Backup operations.

With RMAN you have same authentication options that are available with SQL\*Plus, which are Operating system authentication and password file authentication.

* To connect to RMAN using Operating system Authentication Authentication with the SYSBACKUP Privilege use:

$ rman target ' "/ as sysbackup" '

* Authentication with the SYSDBA Privilege use:

$ rman target ' "/ as sysdba" '

* You can also implicitly connect using below command

$ rman target /

* To Connect to RMAN using Password file Authentication Authentication with the SYSBACKUP Privilege use:

$ rman target1 ‘ “bkpadm@DB1 as sysbackup” ‘

**Where bkpadm is the user and should have SYSDBA privilege.**

* Authentication with the SYSDBA Privilege

$ rman target ‘ “sysadm@DB1 as sysdba” ‘

* You can implicitly connect using below command. Where sysadm is the user and should have SYSDBA privilege.

$ rman target sysadm@DB1

Note that SYSBACKUP does not include data access privilege, such as SELECT ANY TABLE. When you don’t specify the role explicitly then the default used is AS SYSDBA.

## SQL Interface Improvements

In Oracle 12c, you can run SQL commands in RMAN without preceding the command with the SQL keyword.  You also no longer need to enclose the SQL command in quotes.

* The RMAN DESCRIBE provides the same functionality of SQL\*Plus DESCRIBE:

RMAN> desc dba\_profiles;

Name                              Null?           Type

---------------------------- --------         ----------------------

PROFILE                           NOT NULL     VARCHAR2(128)

RESOURCE\_NAME                 NOT NULL     VARCHAR2(32)

RESOURCE\_TYPE                                      VARCHAR2(8)

LIMIT                                              VARCHAR2(128)

COMMON                                             VARCHAR2(3)

* You can run SQL statements from RMAN command prompt:

RMAN> select sysdate from dual;

SYSDATE

---------

10-SEP-14

* You can run DDL/DML Commands from RMAN Command prompt, but note that in order to insert you need to use:

RMAN> create table ora\_table(col1 number, col2 varchar2(20));

Statement processed

RMAN> insert into ora\_table values (1,'Test');

Statement processed

RMAN> update ora\_table set col1=2;

Statement processed

RMAN> drop table ora\_table;

Statement processed

* The user can SHUTDOWN/STARTUP the database and also can use ALTER commands:

RMAN> shutdown immediate

database closed

database dismounted

Oracle instance shut down

RMAN> startup mount

connected to target database (not started)

Oracle instance started

database mounted

Total System Global Area    1610612736 bytes

Fixed Size                 2924928 bytes

Variable Size               520097408 bytes

Database Buffers         1073741824 bytes

Redo Buffers                13848576 bytes

RMAN> alter database open;

Statement processed

## Support for multitenant container and pluggable databases

The multitenant container database (CDB) and pluggable databases (PDB) are introduced in Oracle 12c, and RMAN provides full support for backup and recovery. Using RMAN you can back up an entire container database or individual pluggable databases and also can perform point-in-time recovery.

The multitenant architecture manages many databases as one and retains the isolation, resource control of each database. This will help to manage both infrastructure and human resources effectively.



Backing up a container database is similar to backing up a non-container database. When you back up a container database, RMAN backs up the root, pluggable databases in the container, and archive logs.  When you need to restore you can choose the whole container, one or more pluggable databases or the root only.

### Backup the CDB, PDB, and root

You should have SYSBACKUP or SYSDBA privilege to backup any of the databases.

* You can backup the **Container Database (CDB)** as same as non-container database using below command:

RMAN> BACKUP DATABASE plus ARCHIVELOG;

* You can backup the **Pluggable Database (PDB)** using below command:

RMAN> BACKUP PLUGGABLE DATABASE PDB1, PDB2;

* Or connect to pluggable Database in RMAN :

% rman target sys@PDB1  
RMAN> BACKUP DATABASE;

* You can backup the **root** using below command:

RMAN> BACKUP DATABASE ROOT;

### Complete recovery of CDB, PDB and root

You should have SYSBACKUP or SYSDBA privilege to restore any of the databases.  
**Restoring Container Database (CDB)** is similar to non-container database.  
You can restore the whole CDB using below script:

RMAN> RUN {

STARTUP MOUNT;

RESTORE DATABASE;

RECOVER DATABASE;

ALTER DATABASE OPEN;

}

Note that restoring CDB database will restore all the pluggable databases.  
You can **restore only ROOT**Database using below script:

RMAN> RUN {

STARTUP MOUNT;

RESTORE DATABASE ROOT;

RECOVER DATABASE ROOT;

ALTER DATABASE OPEN;

}

You can **restore Pluggable Databases** in two ways. Either you can restore from root container and connect directly to PDB to restore.

Use below script to restore from root. Using this approach you can able to restore and recover multiple PDB’s with a single command.

RMAN > RUN {

RESTORE PLUGGABLE DATABASE PDB1, PDB2;

RECOVER PLUGGABLE DATABASE PDB1, PDB2;

ALTER PLUGGABLE DATABASE PDB1, PDB2 OPEN;

}

Use below script to connect PDB, restore and recover the database. Using this approach you will be able to restore and recover **only** one PDB.

$ rman target=bkpadm@PDB1

RMAN> run{

RESTORE DATABASE;

RECOVER DATABASE;

}

The steps for performing a point-in-time recovery of the CDB or PDB are the same as a normal database. But note that when you perform Point-in-time recovery on the CDB, it will effect on all the PDBs as well.

When you perform point-in-time recovery on a PDB, it will affect that single database.  
The command to perform a point-in-time recovery is:

SET UNTIL TIME "TO\_DATE(’01-Jan-2014 01:00:00’,’DD-MON-YYYY HH24:MI:SS’)";

SET UNTIL SCN 1999945; # alternatively, specify SCN

SET UNTIL SEQUENCE 100; # alternatively, specify log seq

Below are the few examples to ALTER PLUGGABLE DATABASE.

* Use this command to open all PDBs in one command:

ALTER PLUGGABLE DATABASE ALL OPEN;

* Use this command to open all PDBs except PDB3:

ALTER PLUGGABLE DATABASE ALL EXCEPT PDB3 OPEN;

* Use this command to open PDB4,PDB5 in read only mode:

ALTER PLUGGABLE DATABASE PDB4, PDB5 OPEN READ ONLY;

* Use below command to shut down all PDBs in single command:

ALTER PLUGGABLE DATABASE ALL CLOSE IMMEDIATE;

### Backup of Archived redo logs

You can back up archive logs when they connect to root as a *common* user with SYSDBA or SYSBACKUP privilege, but you cannot back up or delete archive logs when you connect to PDB as a *local* user with SYSDBA or SYSBACKUP privilege.

You are only able to switch the archived logs when you connect to the root of a CDB, but you cannot switch archived redo logs when connected to a PDB.

If you have more than one archive log destination, when you use RMAN to backup the archive redo logs it backs up only one copy of the archived redo logs. RMAN does not include more than one copy because multiple destinations will have same log sequence number.

You can use any of the below commands to backup the archived redo logs  
The command below backs up the database and all the archived redo logs:

RMAN > BACKUP DATABASE PLUS ARCHIVELOG;

The command below only backs up one copy of the sequence number for all archived redo logs.

RMAN> BACKUP ARCHIVELOG ALL;

## DUPLICATE enhancements:

When you duplicate a database using RMAN DUPLICATE, the database is created and opened with RESETLOGS mode. With Oracle database 12c, you can specify that the database must not be opened with “NOOPEN” clause.

This NOOPEN clause useful under following situations:

* If you need to make changes to initialization parameters such as block change tracking, flashback database settings
* Opening the database conflict with other source database
* If you plan to create database for upgrade and want to open in upgrade mode

The command below creates duplicate database, but it will not open.

RMAN> DUPLICATE TARGET DATABASE TO DB1

FROM ACTIVE DATABASE NOOPEN;

## Multisection Backup Improvements

The multisection backup functionality was introduced in Oracle 11g to handle large data file backups. Using this functionality RMAN can break up a large file into sections during the backup and recovery, which can improve the performance of large datafiles backup. You can select the size using the SECTION SIZE keyword and each channel will create separate files within the backup set, and backup the database in parallel. This functionality supports only backup sets in 11g.

In Oracle 12c, the multisection backup supports incremental backups and image copies, including backup sets (introduced in 11g). This functionality can only be used for data files, you cannot use this to backup control files.

If the SECTION SIZE that you selected is larger than the actual file then RMAN does not use multisection backup. If you specify a small SECTION SIZE that produces more than 256 sections then RMAN increases the SECTION SIZE to a value that results 256 sections.

* The following example creates a multisection backup of the database using image copies.

RMAN> BACKUP AS COPY SECTION SIZE 1024M DATABASE;

RMAN> BACKUP INCREMENTAL LEVEL 1 SECTION SIZE 1024M DATABASE;

* The following example creates multisection incremental level1 backup

RMAN> BACKUP INCREMENTAL LEVEL 1 SECTION SIZE 1024M DATABASE;

To improve the backup performance, use unused block compression and block change tracking in conjunction with multisection incremental backups.

## Restoring and Recovering Files over Network

Using RMAN you can restore and recover a database, datafile, controlfile, tablespace or spfile over the network from a physical standby database. To restore the database over the network, use the RESTORE… FROM SERVICE command and use the RECOVER…FROM SERVICE command to recover the database over the network. The FROM SERVICE clause specifies the service name of the physical standby.

You can also use multisection, compression and encryption to improve backup and restore performance.

* Use SECTION SIZE with RESTORE command to perform multisection restore
* Use SET ENCRYPTION clause before the RESTORE command to specify the encryption
* Use USING COMPRESSED BACKUPSET clause to compress backup sets

This feature is useful to synchronize primary and standby database. Here are the few scenarios

* Roll-forward a physical standby database to sync with the primary database
* Restore the primary database using physical standby database.
* Restore physical standby database using the primary database.

In the following example restoring data file over the network from physical standby to primary database:

* Connected to primary database implicitly

RMAN> CONNECT TARGET /

* Backup sets encrypted using AES128 encryption algorithm

RMAN> SET ENCRYPTION ALGORITHM 'AES128';

* Restoring the datafile on the primary using datafile on physical database with service “standby\_db”

RMAN> RESTORE DATAFILE '/db1/oradata/users.dbf'

FROM SERVICE standby\_db SECTION SIZE 1024M;

## Storage Snapshot Optimization

This new feature enables you to take a storage snapshot of your database using third-party technologies without keeping the database in BACKUP mode. When you need to recover, you can use point in time of the snapshot. You can roll forward by using the database archive logs, and use this snapshot feature to recover part or all of the database.

The best practice is to use different storage location to keep snapshot than the one where the database currently running.

In order to backup your Oracle database using storage snapshot optimization, the third-party snapshot technologies must meet the following requirements:

* The snapshot preserves the write order for each file.
* The database is crash consistent during the snapshot.
* The snapshot technology stores the time at which the snapshot is completed.

If third-party snapshot technology vendor cannot guarantee compliance with above requirements then you must keep the database in BACKUP mode to take the snapshot.

Follow these steps to keep the database in BACKUP mode and take snapshot

* SQL> ALTER DATABASE BEGIN BACKUP
* Take Snapshot using third-part technologies
* SQL> ALTER DATABASE END BACKUP

Use RECOVER…SNAPSHOT TIME command to recover the database in one step from RMAN or SQL\*Plus. You can recover the database to a point-in-time or current time after the snapshot was taken. When performing point-in-time make sure that the recovery time cannot be earlier than the snapshot time.

To recover database completely use below command

RECOVER DATABASE;

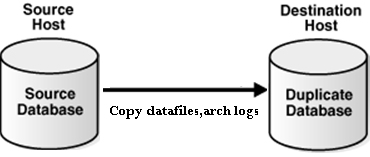
To recover database using particular snapshot use below command

RECOVER DATABASE UNTIL TIME ‘10/10/2014 10:00:00’ SNAPSHOT TIME ‘10/10/2014 09:00:00’

If there are any structural changes during the snapshot then the snapshots are unusable. Do not perform the operations such as ONLINE, OFFLINE, DROP, RENAME, ADD, READONLY and SHRINK on data files and table spaces.

## Active Database Duplication Improvements

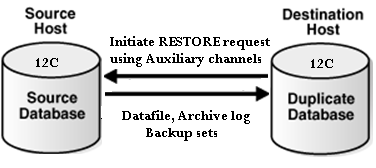
Active Database duplication was introduced in Oracle Database 11g. Using this feature you can create clone or standby database by copying the data files and archive logs using the TARGET (Source) database channels over the network to clone AUXILIARY database. As you are using the TARGET database channels you will see processing load on the TARGET instance. In this method no backups of the TARGET database are required.



In Oracle 11g, Performance typically gated by network bandwidth

In Oracle 12c, you can perform Active Database duplication using the backup sets. You can allocate sufficient AUXILIARY channels to connect TARGET database and retrieve the backup’s sets over the network, this reduced the load on the TARGET (source) database. You can use unused block compression to reduce the size of the backups transported over the network.

While performing the Active database duplication you can also encrypt backups and multisection backups.



As you see in below duplication example, it is using AUXILIARY channel.

oracle@gc12c ~]$ rman target sys/password@TESTDB auxiliary sys/password@DUPDB

Recovery Manager: Release 12.1.0.1.0 - Production on Wed Oct 15 10:26:58 2014

Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.

connected to target database: TESTDB (DBID=488428308732)

connected to auxiliary database: DUPDB (not mounted)

RMAN> run {

2> duplicate target database to DUPDB from active database

2> using backupset

4> db\_file\_name\_convert('TESTDB','DUPDB');

5> }

Starting Duplicate Db at 15-OCT-2014 10:27:11

using target database control file instead of recovery catalog

allocated channel: ORA\_AUX\_DISK\_1

channel ORA\_AUX\_DISK\_1: SID=20 device type=DISK

current log archived ……………….

Note that if the numbers of AUXILIARY channels allocated are less than target channels, or no auxiliary channels are allocated, then RMAN uses image copies to perform active database duplication. In order to RMAN use backup sets you need to establish connection to target database using a net service name and one of the following conditions must be satisfied.

* The number of AUXILIARY channels should be equal or greater than TARGET channels
* The DUPLICATION …FROM ACTIVE DATABASE should contain either USING BACKUPSET, USING COMPRESSED BACKUPSET or SECTION SIZE clause

## Cross-Platform Backup and Restore Improvements

Cross-platform transportable tablespace and database were introduced in Oracle 10g. In this method the user must use EXPORT or DATAPUMP to export metadata. The tablespace should be in read-only mode during the process and only image copies are used. This method is more suitable for smaller or low transaction rate applications.

In Oracle 12c, you can transport data across platforms using either full or incremental backups, using image copies or backup sets and added platform support for incremental backup. To perform cross-platform backups using backup sets, the destination database must be Oracle 12c or later. These features simplify the platform migration and minimize read-only down time on the source database using incremental based approach. By using this improvement users can reduce downtime by 8 times versus traditional migration approaches.

If the user performs the conversion at source, CONVERT TABLESPACE command should be used. If the user performs conversion at destination, CONVERT DATAFILE command should be used.

### Cross platform transportable tablespace

You can use RMAN to transport tablespace across platforms with the same endian or different endian formats. If transporting tablespace between different endian formats then the user must use CONVERT command to perform conversion. If transporting tablespace between same endian formats then the CONVERT command isn’t needed.

Transportable tablespace is useful under following situations:

* When migrating tablespace across the platforms
* When moving data from large data warehouse to data marts on small servers
* Sharing read-only tablespace across heterogeneous cluster where hosts have same endian format

User can use below query to get the platform name of the connected database

SQL> SELECT PLATFORM\_NAME, ENDIAN\_FORMAT FROM V$TRANSPORTABLE\_PLATFORM

WHERE PLATFORM\_ID = (SELECT PLATFORM\_ID FROM V$DATABASE);

Before you convert the tablespace make sure you run DBMS\_TTS.TRANSPORT\_SET\_CHECK procedure to check if tablespace is self-contained or not.

For ex: – EXECUTE DBMS\_TTS.TRANSPORT\_SET\_CHECK(‘USER\_TBS’, TRUE);

After executing these commands, you can check for violations in TRANSPORT\_SET\_VIOLATIONS view. User must resolve before proceeding with conversion.

### Cross-platform transportable database

You can use RMAN to transport entire database across platforms with same endian format only. When transporting database the user doesn’t need to perform EXPORT/IMPORT as SYSTEM tablespace is part of the database that being copied. You can use CONVERT DATABASE command to convert and automatically transport the database to its destination. You can convert the data files either on source or destination platforms.

When you run the CONVERT DATABASE on the destination platforms, it does not convert the format of the files, rather it generates the two scripts to perform conversion manually.

* CONVERT SCRIPT – This script used to convert data files copies in batch mode
* TRANSPORT SCRIPT – This script contains SQL Statements to create new databases on the destination platform

Transportable Database is useful under following situations:

* When Evaluation migration path for new platforms
* When you need database on less expensive servers that use different platform
* Distribute data from one source system to multiple targets with different platforms

Before converting the database make sure you run DBMS\_TDB.CHECK\_DB function. This function checks for incompatible endian formats, active transactions or incorrect compatibility settings between source and destination.

In Oracle 12c, Oracle 10.2.x and 11g database backups can be restored and recovered cross-platform to Oracle 12c. Note that cross-platform transportable database is not the same thing as transportable tablespace

## Recovering Tables and Table Partitions using RMAN Backups

RMAN enables you to recover tables and table partitions at a point-in-time without affecting the other objects in the database. Use RECOVER TABLE command to recover tables or table partitions from an RMAN backup.

This feature is useful in the following scenarios:

* When the object has Logical corruption or dropped
* When there is no sufficient undo to perform Flashback table
* When DDL operation modified the structure and you want to recover the data (Flashback cannot rewind the structural changes)
* If you need to recover a small number of tables to a point-in-time

Before you prepare to recover the tables and table partitions, make sure you verify the pre-requisites and determine the point-in-time. RMAN enables you to specify the point-in-time either using SCN, Time or sequence number. In order to perform table/ table Partition recovery these conditions must be met:

* Database must be in ARCHIVELOG mode and read-write mode
* At least one full backup is available along with archived logs
* Enough disk space is available on the database server for auxiliary instance
* If present, any dependent objects to include in recovery
* COMPATIBLE parameter must be set to 11.1.0 or higher to recover table partition

RMAN enables recovery of selected tables without affecting remaining database objects. During the recovery process RMAN creates an auxiliary database, which is used to recover the tables or table partitions to a specified point-in-time. User need to specify the auxiliary database location using AUXILIARY DESTINATION clause in the RECOVERY command or SET NEWNAME command. Please find the steps performed by RMAN during the recovery process:

* Determine the backup which has the tables or table partitions that needs to recover to specified point-in-time
* Create auxiliary database and recovery the tables or table partitions until specified point-in-time
* Take a data dump export with recovered tables or table partitions
* Import the dump into target database
* Rename the recovered tables or table partitions in the target database

Please find an example to recovery TBL1 table.

RECOVER TABLE TESTUSER.TBL1

UNTIL SCN 384840289

AUXILIARY DESTINATION '/tmp/TESTDB/recover'

DATAPUMP DESTINATION '/tmp/TESTDB/dumpfiles'

DUMP FILE 'testdump.dat';

If source table exists then user can specify NOTABLEIMPORT or REMAP TABLE. Also user can use UNTIL TIME or UNTIL SEQUENCE clause to specify point-in-time recovery.

Please find an example to recovery TBL1 table as TBL1\_REC to the state that it was 2 days before the current date.

RECOVER TABLE TESTUSER.TBL1

UNTIL TIME 'SYSDATE-2'

AUXILIARY DESTINATION '/tmp/TESTDB/recover'

REMAP TABLE 'TESTUSER'.'TBL1':'TBL1\_REC';

There are some limitations recovering tables and table partitions:

* We cannot recover table and table partitions belonging to SYS schema
* We cannot recover table or table partitions from SYSAUX,SYSTEM tablespace Tables
* We cannot recover tables with named NOT NULL constraint using REMAP option.
* We cannot recover Table/Table partitions ON STANDBY database
* We cannot recovery table partitions if version is prior Oracle Database 11g R1

Note that there are other methods available to recover tables to a point-in-time such as Oracle Flashback and Tablespace Point-in-Time Recovery.

## Unified auditing and RMAN

In Unified auditing you can consolidate all audit records into single audit trail. User can view the audit trail by querying UNIFIED\_AUDIT\_TRAIL data dictionary view for single-instance and GV$UNIFIED\_AUDIT\_TRAIL for Oracle RAC. In order to query UNIFIED\_AUDIT\_TRAIL view user must have AUDIT\_ADMIN OR audit viewer ROLE.

The UNIFIED\_AUDIT\_TRAIL view has fields begin with RMAN\_ and these fields automatically record RMAN related events.

Please find the Oracle recovery Manager specific columns in UNIFIED\_AUDIT\_TRAIL data dictionary view.

* RMAN\_SESSION\_RECID – It contains the RMAN session identifier
* RMAN\_SESSION\_STAMP – It contains the timestamp for the RMAN session.
* RMAN\_OPERATION – It contains the operation executed by RMAN job
* RMAN\_OBJECT\_TYPE – It contains the type of object involved in a RMAN session.
* RMAN\_DEVICE\_TYPE – it contains device type associated with RMAN session. This can be SBT or DISK.