Oracle Database 12cR1 - New feauture

# Architecture new feauture

## 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.

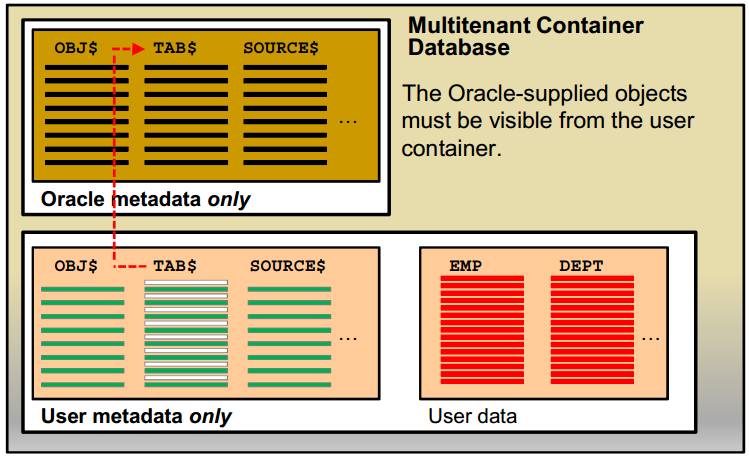
The multitenant architecture maintains:

* 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.

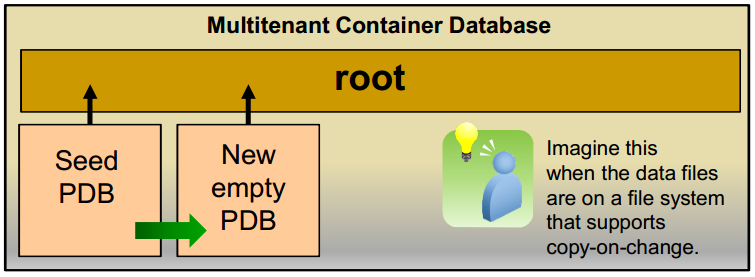
Separating the metadata is the first step, the second is allowing the application or users inside the “user” container to access the Oracle-supplied objects.

**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**



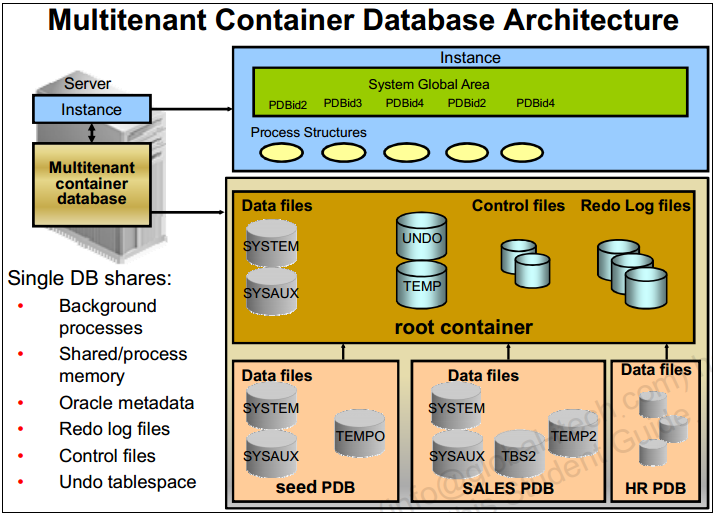
There are four methods to provision pluggable databases:

* 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.
* Th 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

### Assigning Default Temporary Tablespaces

The default temporary tablespace for the root is set at the root container level. There may be multiple temporary tablespaces, but only one can be the default.

A default temporary tablespace (or tablespace group) can be set for each PDB. A PDB may have multiple temporary tablespaces, but only one default per PDB.

When you create a user you can specify a temporary tablespace to be used by that user. If a temporary tablespace is not specified, the default tablespace for the PDB is used.The amount of space a PDB can use in the shared temporary tablespace can be limited:

ALTER PLUGGABLE DATABASE STORAGE (MAX\_SHARED\_TEMP\_SIZE 500M);

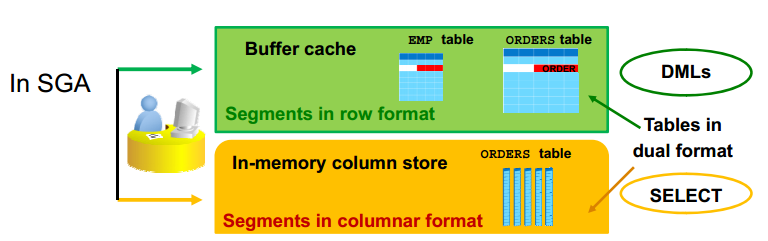
In this example, if the value used by sessions that are connected to the PDB is greater than 500M, then no additional storage in the shared temporary tablespace will be available for sessions connected to the PDB until the amount of storage used by them becomes smaller than 500M.

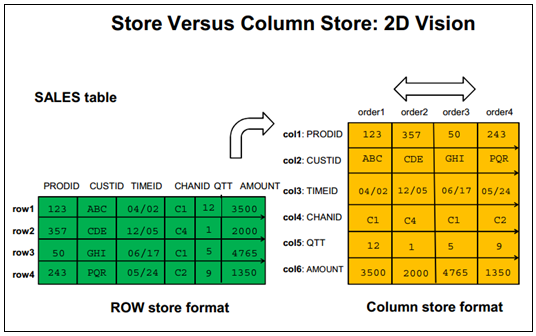
When you unplug a PDB from a CDB, its temporary tablespaces are also unplugged.

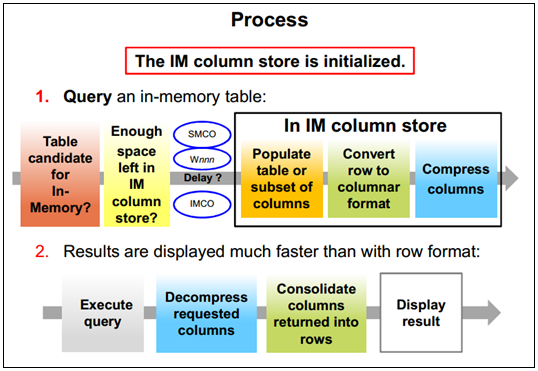
## In-Memory Column Store

### 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.

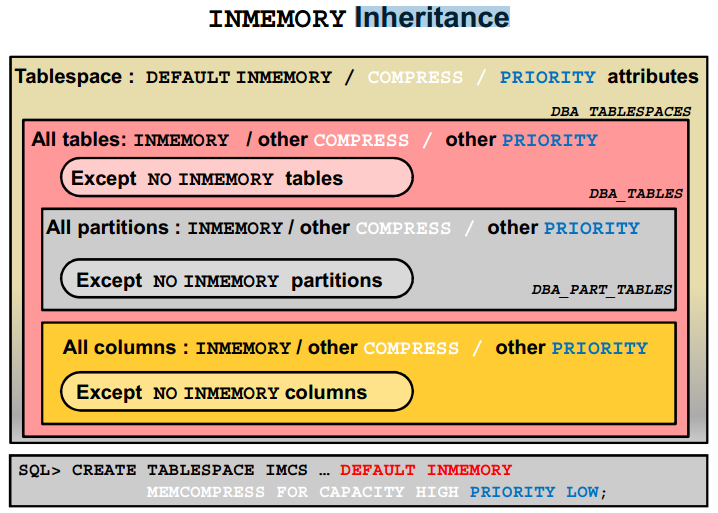
COMPATIBLE = 12.1.0.0.0

* 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:

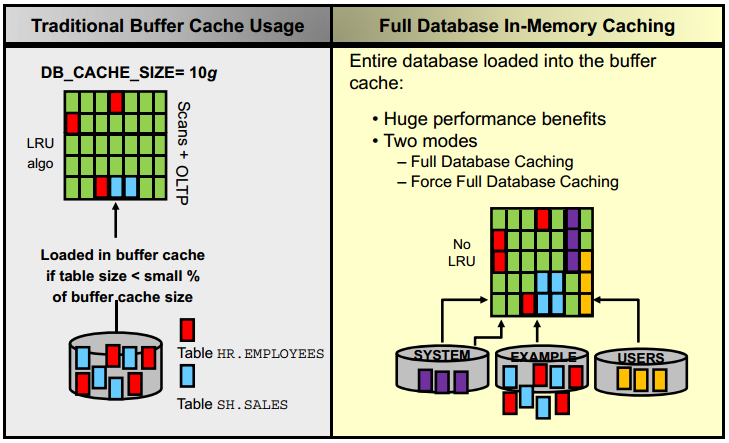
* Rac
* In-Memory PX
* Data Pump.

## Full Database In-Memory Caching

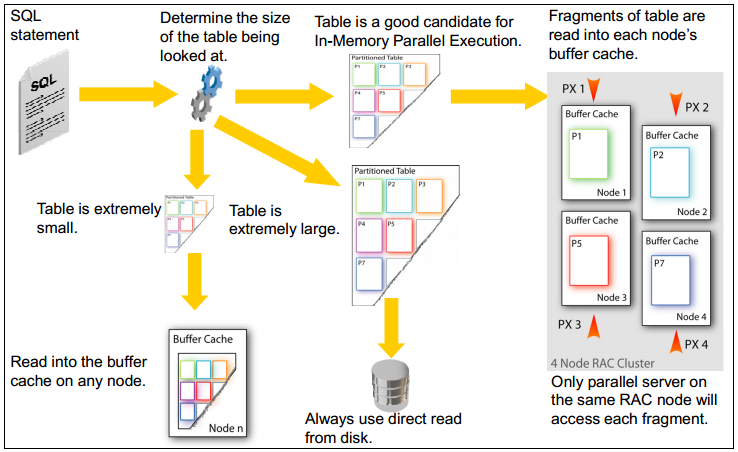
### Concept

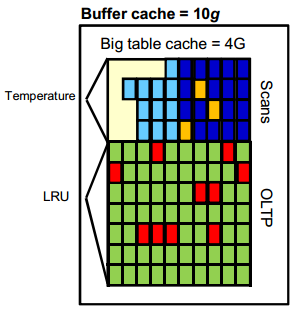
The Full Database In-memory Caching feature enables an entire database to be cached in memory when the database size (sum of all data files, SYSTEM tablespace LOB CACHE files , LOB CACHE files minus SYSAUX, TEMP) is smaller than the buffer cache size. Caching and running a database from memory leads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





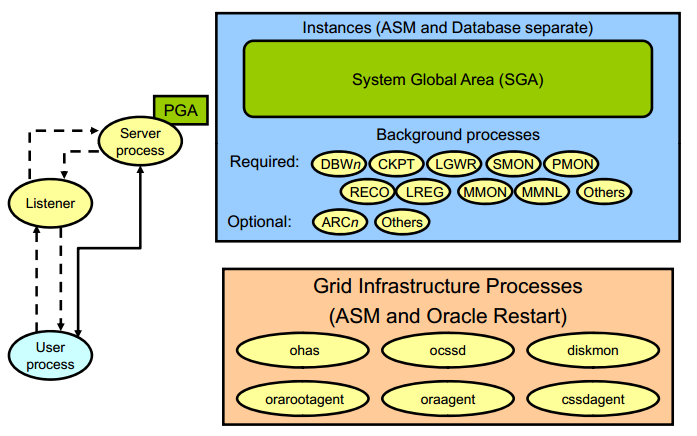
Automatic Big Table Caching enhances the in-memory parallel query capabilities of the Oracle Database in both single instance and Oracle RAC environments. An optional section of the buffer cache, called the big table cache, is used to store data for parallel table scans.

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.

## Kiến trúc Processes

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

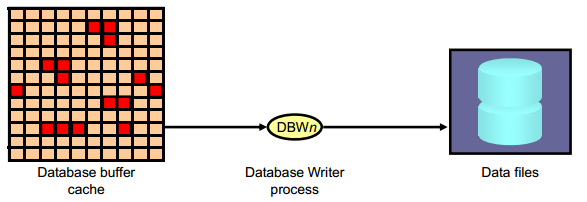
* + User processes that run the application or Oracle tool code
  + Oracle Database processes that run the Oracle Database server code (including server processes and background processes)
    - Server processes created on behalf of each user’s application can perform one or more of the following:
      * 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



Background process (các tiến trình nền) thực hiện các chức năng thay cho lời gọi tiến trình xử lý tương ứng. Nó điều khiển vào ra, cung cấp các cơ chế xử lý song song nâng cao hiệu quả và độ tin cậy. Tùy theo từng cấu hình mà Oracle instance có các Background process như:

* + 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:

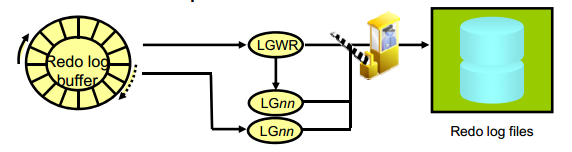
* + When a server process cannot find a clean reusable buffer after scanning a threshold

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.

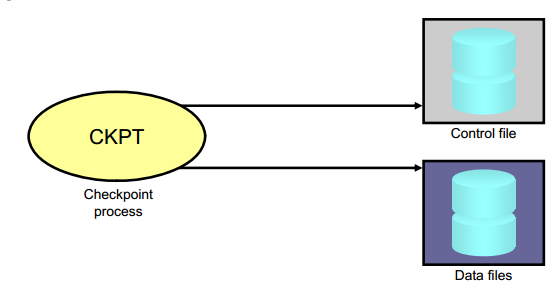
In all cases, DBWn performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system

### Log Writer Process (LGWR)



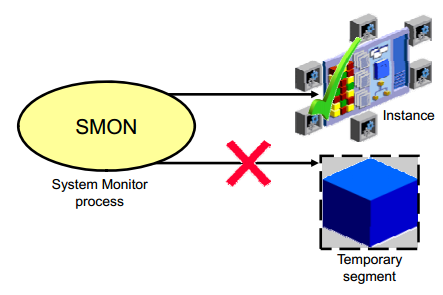
* + The Log Writer process (LGWR) is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.
  + LGWR starts and coordinates multiple helper processes 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)



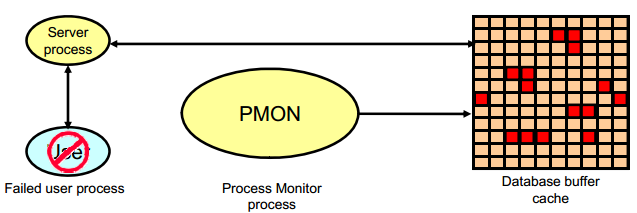
* + A checkpoint is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.
  + When a checkpoint occurs, Oracle Database must update the headers of 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)



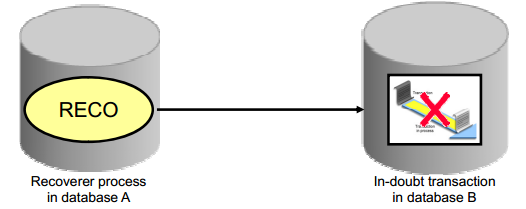
* + The System Monitor process (SMON) performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them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

### Process Monitor Process (PMON)



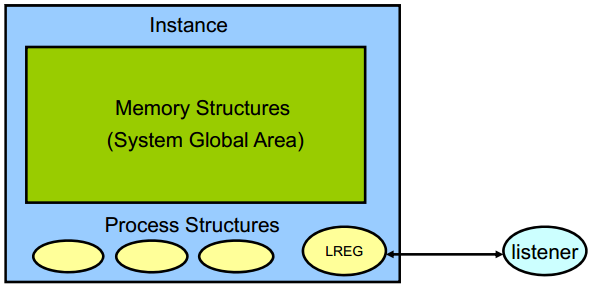
* + The Process Monitor process (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using
  + PMON periodically checks the status of dispatcher and server processes, and restarts any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

### Recoverer Process (RECO)



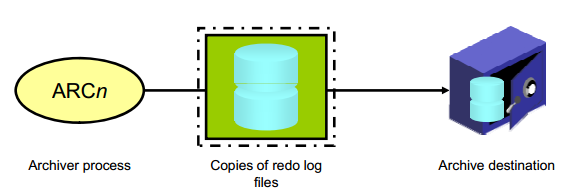
* + The Recoverer process (RECO) is a background process that is used with the distribute database configuration that automatically resolves failures involving distributed transactions
  + Automatically connects to other databases involved in indoubt distributed transactions
  + Automatically resolves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

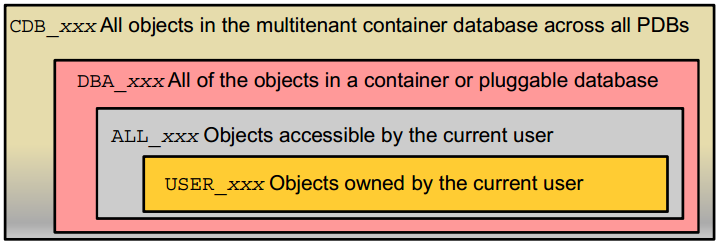
### Archiver Processes (ARC*n*)



* + 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

## Các view cơ bản của Oracle



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

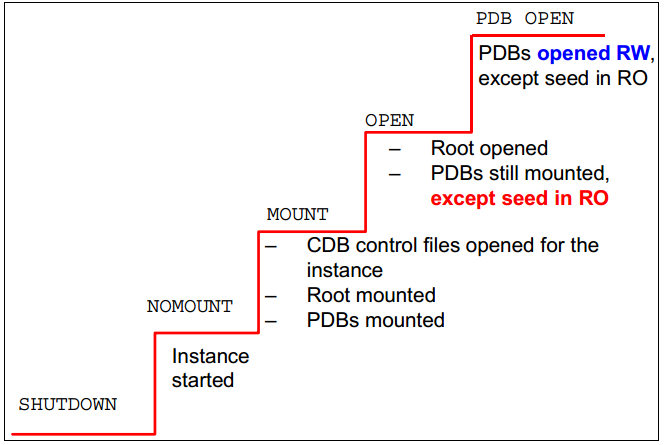
* CDB\_pdbs: All PDBS within CDB
* CDB\_tablespaces: All tablespaces within CDB
* CDB\_users: All users within CDB (common and local)

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

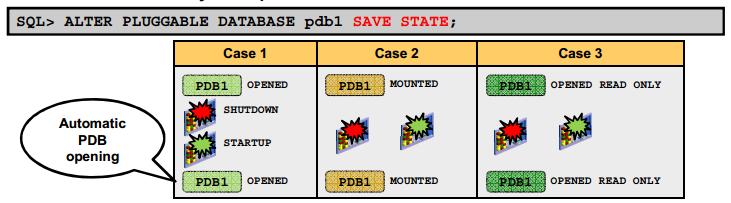
* All PDBs closed (no new specific message)
* CDB closed
* CDB dismounted
* Instance shut down

**Close PDB**

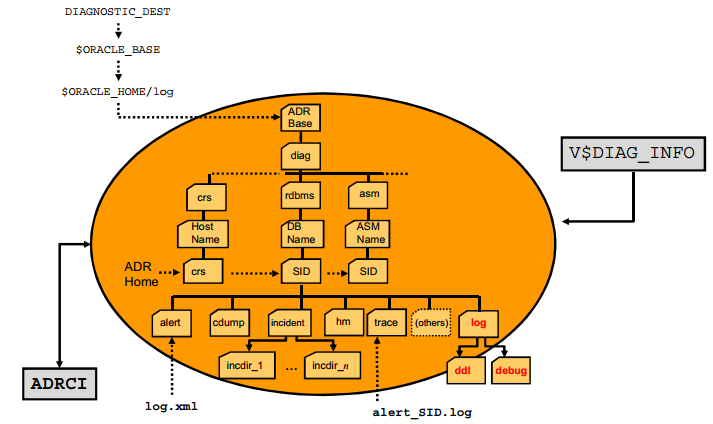
SQL> CONNECT sys@PDB1 AS SYSDBA

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;

## **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.

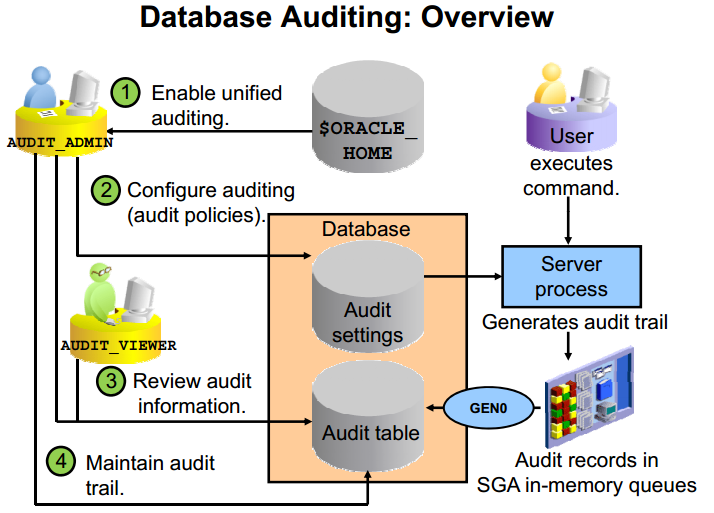
* **Mandatory auditing**: All Oracle databases audit certain actions regardless of other audit options or parameters. The reason for mandatory audit logs is that the database needs to record some database activities, such as connections by privileged users.
* **Standard database auditing**: Select the 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

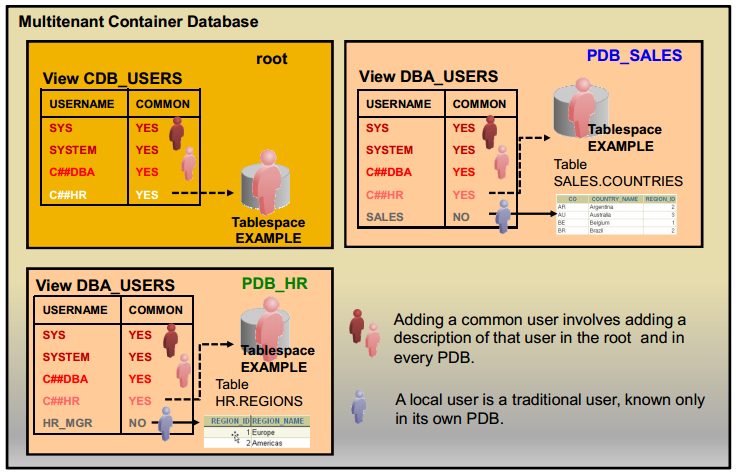
make -f ins\_rdbms.mk uniaud\_on ioracle ORACLE\_HOME=$ORACLE\_HOME

* + 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.

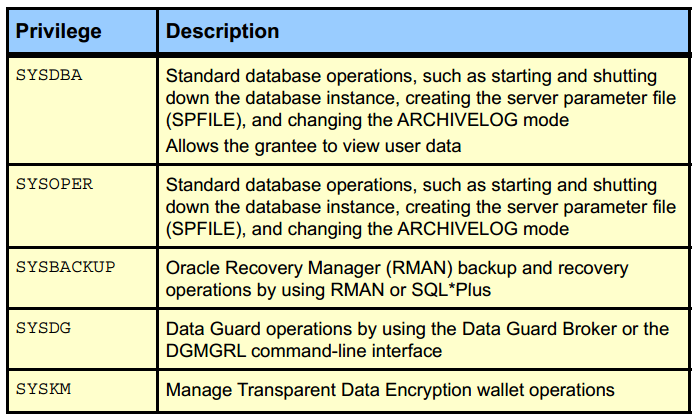
* + A local user is defined in the PDB’s own data dictionary—and so is not known outside of that PDB.
  + A local user can connect only to the PDB where it is defined.
  + A local user is specific to a particular PDB and owns a schema in this PDB.
  + According to the privileges granted, a user can work 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.

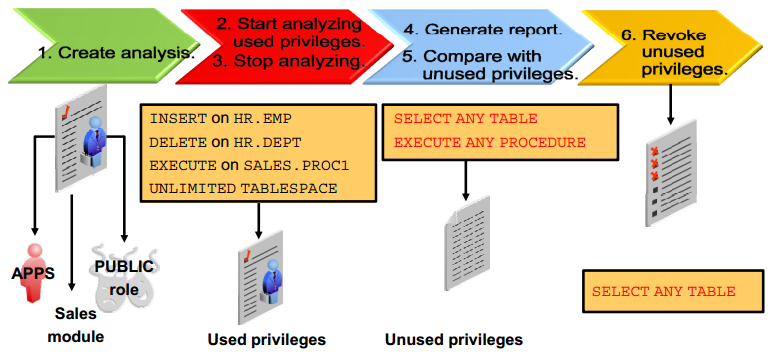
**Admin Account**

* + 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:

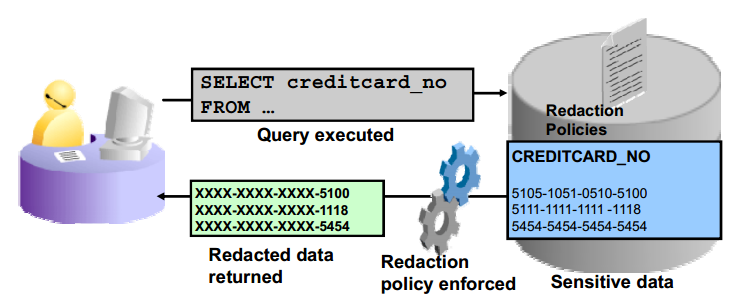
* + Make a role nondefault. The user must now explicitly enable the role before the role’s privileges can be exercised.
  + Have a role require additional authentication by using the IDENTIFIED clause to indicate that a user must be authorized by a specified method before the role is enabled with the 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**



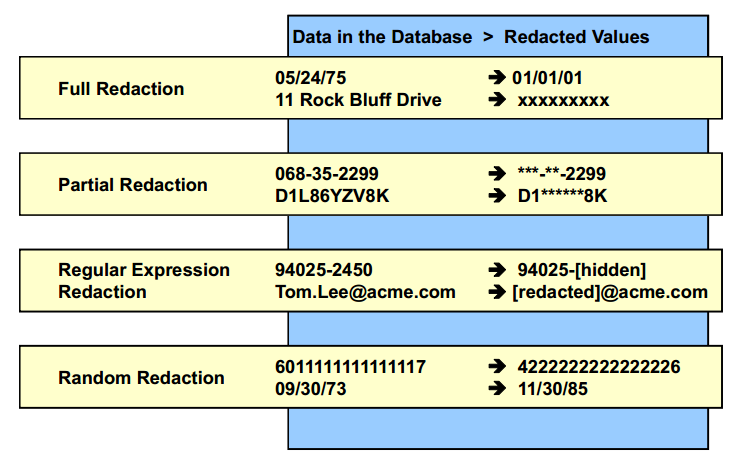
Currently, many Oracle Database customers prevent the display of sensitive data to end users by performing redacting in each application. Oracle Data Redaction moves this functionality from the application to the database. This approach has several benefits over redacting data in the application tier,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



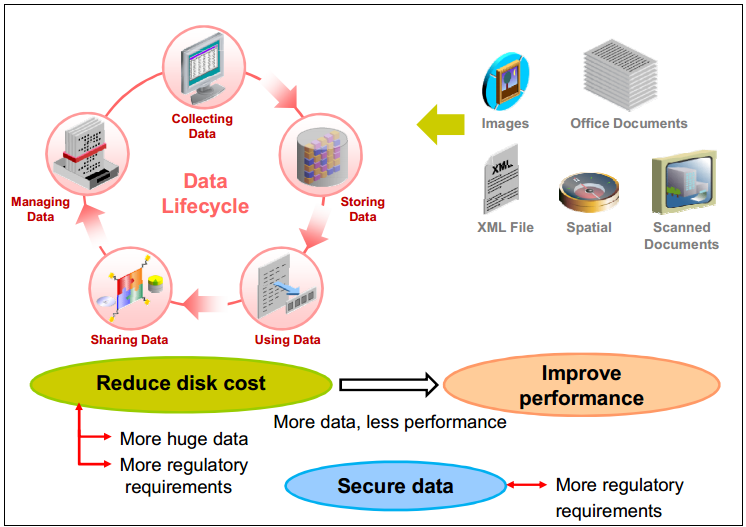
## **ADO and Storage**

### 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:

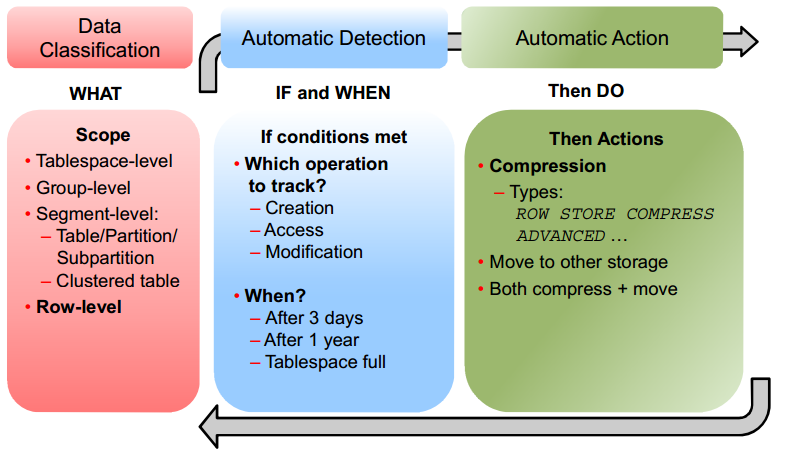
* + Data accesses at segment-level
  + Data modifications at block and segment levels

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.

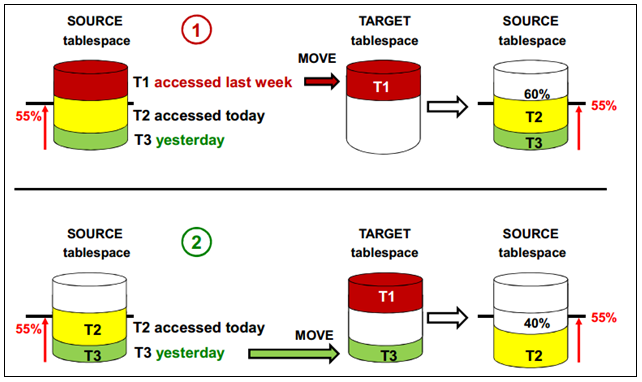
**Storage Tiering**

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

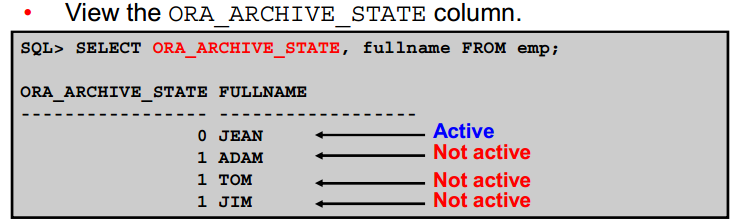
SQL> CREATE TABLE emp  
(EMPNO NUMBER(7), FULLNAME VARCHAR2(40),  
JOB VARCHAR2(9), MGR NUMBER(7))  
ROW ARCHIVAL;

* + 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')

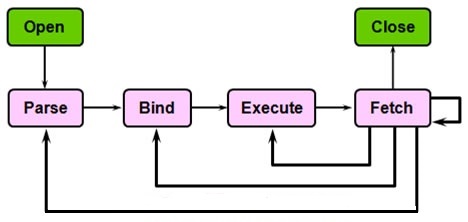
* + Disable tính năng valid:

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

## **Query SQL**

### **Kế hoạch thực thi 1 câu lệnh trong Oracle**

Oracle Server xử lý câu lệnh SQL theo trình tự chính sách thực thi các bước sau:



**Open**

* + Ngầm định khai báo và khởi tạo Cursor cho câu lệnh SQL

**Parse**

* + Syntatic: Kiểm tra cú pháp
  + Semantic : Kiểm tra đối tượng (object)
  + View merging: Rewrite lại câu lệnh dựa vào các based table thay vì sử dụng view
  + Statement Transformation : Rewirte lại sự biến đổi của câu lệnh để phân tích thành những câu đơn giản hơn.
  + Optmization : Tối ưu hóa câu lệnh
  + QEP Generation : Query Evulation Plan : đánh giá kế hoạch cho câu lệnh

**Bind**

* + Tìm và gán giá trị cho các [**bind-variable**](http://www.vietpace.com/kienthuc/VietPace_toiuu_caulenh_Oracle_SQL_Phan1.html#_Giải_nghĩa_từ) nếu có

**Execute**

* + Thực thi các bước mô tả trong “sơ đồ thực thi câu lệnh SQL”

**Fetch**

* + Chuyển kết quả về nơi gọi thực thi lệnh
  + Fetch ở đây có thể lặp lại nhiều lần do tham số limit của nó ( giới hạn xử lý mỗi lần)

**Close**

* + Ngầm định đóng Cursor cho câu lệnh

### Excution plan

EXPLAIN PLAN

SET statement\_id = 'ex\_plan2' FOR

SELECT last\_name FROM employees

WHERE last\_name LIKE 'Pe%';

SELECT PLAN\_TABLE\_OUTPUT

FROM TABLE(DBMS\_XPLAN.DISPLAY(NULL, 'ex\_plan2','BASIC'));

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

| Id | Operation | Name |

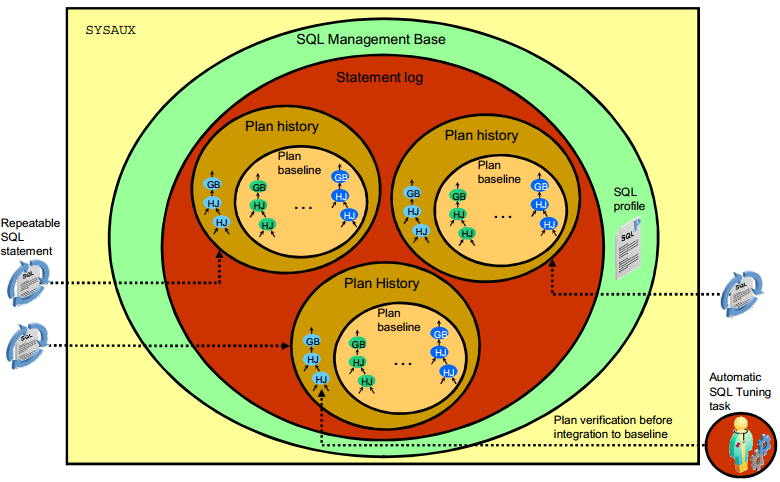
----------------------------------------

| 0 | SELECT STATEMENT | |

| 1 | INDEX RANGE SCAN| EMP\_NAME\_IX |

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

### SQL Plan Baseline



The statement log, plan history, and plan baselines are stored in the SQL Management Base (SMB), which also contains SQL profiles. The SMB is part of the database dictionary and is stored in the SYSAUX tablespace. The SMB has automatic space management (for example, periodic purging of unused plans). You can configure the SMB to change the plan retention policy and set space size limits.

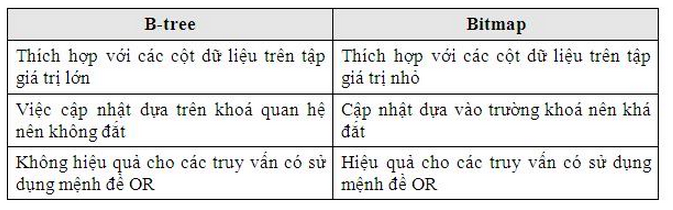
### Cơ chế index của Oracle.

Index là những cấu trúc tùy chọn liên quan đến bảng và cột.

* + Có thể tạo index trên một hoặc nhiều cột của một bảng để tăng tốc độ thực hiện câu lệnh SQL trên bảng đó.
  + Các index này sẽ giúp bạn xác định vị trí thông tin nhanh hơn. Index là cách thức chính của việc giảm đĩa I / O khi được sử dụng đúng cách.
  + Quyết định index đối với 1 bảng dựa vào yêu tố
    - Tạo Index nếu kết quả nhỏ hơn 15% tổng số row của 1 bảng lớn
    - Tăng hiệu năng cho phép join bảng ( bằng cách index cột dùng để join)
    - Không index cho bảng nhỏ
  + Không index tại cột kiểu LONG và LONGRAW
  + Chỉ có 1 index duy nhất được hoạt động trong 1 bảng

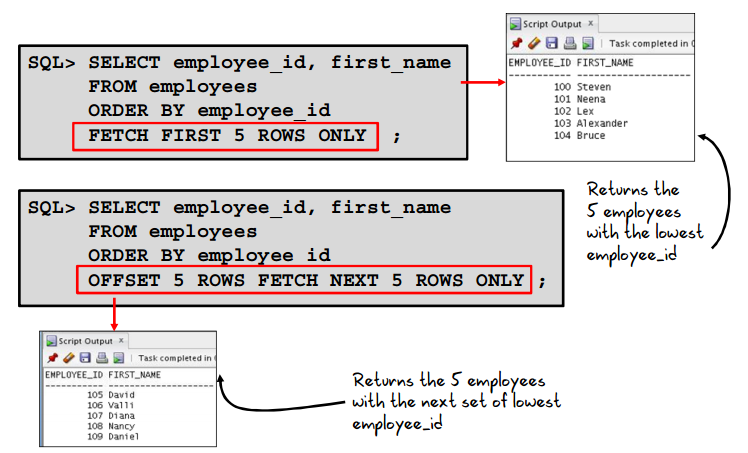
Các loại index chính:

* + B-Tree indexes
    - Btree Index được dùng để giúp truy vấn các câu truy vấn dạng Insert, Update, delete.
    - B-tree là loại index mặc định - nếu tạo ra một index mà không xác định bất cứ điều gì, thì đó là một index B-tree.
  + Bitmap indexes
    - Là 1 index được dùng để làm việc với những trường có dữ liệu rời rạc, với số lượng ít các giá trị khác nhau (tức mức độ lặp lại ở trong trường này thường là lớn)
    - Được sử dụng khI Mức độ dữ liệu trùng lặp lớn: trong oracle thì quyết định là: distinct val/ total val < 1% thì dùng bitmap index : Tức nếu số giá trị rời rạc của 1 cột trong bảng trên tổng số dòng của bảng mà nhỏ hơn 1% thì ta dùngBit map index.
    - Ví dụ trường giới tính: distinct val = 2 <số giá trị riêng biệt> và Total val = 1000 dòng. 2/1000 < 1% dùng bitmap.
    - Không hoặc là ít thao tác update hoặc insert lên bảng dữ liệu.
    - Bảng có rất nhiều cột



* + Bitmap join indexes
    - Bitmap join index cho tham gia giữa các bảng (2+). Tạo Bitmap join index được xác định trên một bảng duy nhất. Nó lưu trữ các kết quả của sự join.
  + Function-based indexes
    - Có thể tạo fuction-based indexes cho những truy vấn sử dụng các function đối với các trường:
      * Upper(abc)
      * 12 \*salary

### SQL Row-Limiting Clause



You specify the row limiting clause in the SQL SELECT statement by placing it after the ORDER BY clause. Note that an ORDER BY clause is not required.

* + OFFSET: Use this clause to specify the number of rows to skip before row limiting begins. The value for offset must be a number. If you specify a negative number, offset is treated as 0. If you specify NULL or a number greater than or equal to the number of rows that are returned by the query, 0 rows are returned.
  + ROW | ROWS: Use these keywords interchangeably. They are provided for semantic clarity.
  + FETCH: Use this clause to specify the number of rows or percentage of rows to return:
    - FIRST | NEXT: Use these keywords interchangeably. They are provided for semantic clarity.
    - row\_count | percent PERCENT: Use row\_count to specify the number of rows to return. Use percent PERCENT to specify the percentage of the total number of selected rows to return. The value for percent must be a number.

The first code example returns the five employees with the lowest employee\_id. The second code example returns the five employees with the next set of lowest employee\_id.

## Backup & Recover Database

### RMAN

* + No SQL prefix or quotes required
  + Provides SQL\*Plus DESCRIBE functionality

RMAN> ALTER TABLESPACE users  
ADD DATAFILE '/testdata/users02.dbf' SIZE 10M;  
RMAN> SELECT NAME, DBID, LOG\_MODE FROM V$DATABASE;  
RMAN> DESC tab1  
Name Null? Type  
--------------------------------- -------- ---------------  
TEST\_NAME VARCHAR2(128)

**Duplicating an Active Database**

RMAN> SET ENCRYPTION …;  
RMAN> DUPLICATE TARGET DATABASE TO orcl2  
FROM ACTIVE DATABASE  
[USING BACKUPSET]  
[SECTION SIZE …]  
[USING COMPRESSED BACKUPSET] …;

* + NOOPEN Option: to check before open
    - Moving the location of the database (for example, to ASM)
    - Upgrading a database (where the database must not be open with resetlogs, prior to running upgrade scripts)

**Transporting Data Across Platforms**

* + Using backupset instead of image copy
    - Conversion on the source host

RMAN> BACKUP TO PLATFORM 'Linux x86 64-bit' FORMAT '/bkp\_dir/transp.bck' DATABASE;

* + - Conversion at the destination host

RMAN> BACKUP FOR TRANSPORT FORMAT '/bkp\_dir/transp.bck' DATABASE;

### Backup

Back up CDB and applications independently:

* + ARCHIVELOG mode at CDB level
  + CDB and PDB backups
  + Hot backup at CDB and PDB levels

Recover CDB or PDBs at different levels:

* + Instance failure: CDB level
  + Complete media recovery:
    - CDB or PDB temp file
    - Control file/redo log file/root data file: CDB mounted
    - PDB data file
  + Incomplete media recovery: CDB mounted or PDB closed
  + Flashback database: CDB mounted

### Restore DB

Instance recovery: CDB level only

Automatic missing temp file recreation at CDB open

Complete media recovery after file loss or corruption

* + CDB: Same as for non-CDB
    - Redo log files, control files
    - Root data files
  + PDB data files
  + Tablespace: PDB or CDB data files

Incomplete media recovery after file loss or corruption

* + CDB: The whole CDB back in time
  + PDB: A whole PDB back in time
  + TSPITR for any tablespace except SYSTEM, UNDO, SYSAUX

Block recovery: No change

Flashback database: CDB mounted

## Manageability ( Using EM )

### Real-Time Database Operation Monitoring

**Concept**

Real-Time Database Operation Monitoring extends and generalizes Real-Time SQL Monitoring. Real-Time SQL Monitoring helps determine where a currently executing SQL statement is in its execution plan and where the statement is spending its time.

You can use Real-Time Database Operation Monitoring for performance monitoring of active SQL statements, PL/SQL procedures, and functions. You can also see the breakdown of time and resources used for recently completed statements.

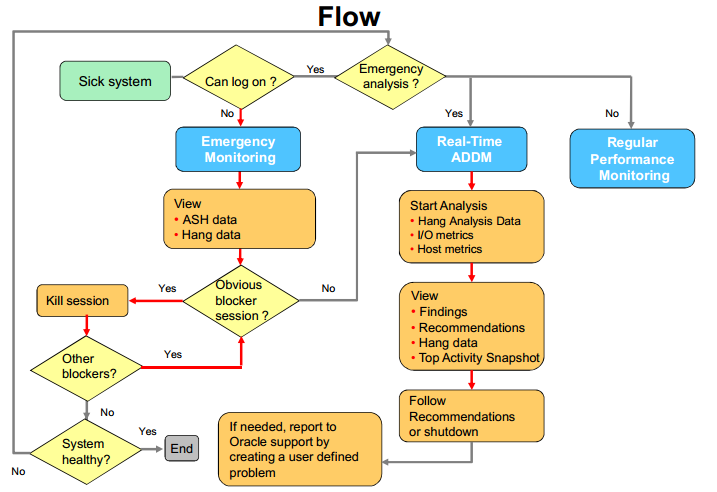
**Use Cases**

In which situations would a DBA use Real-Time Database Operation Monitoring?

* + A large retailer has a DSS with hundreds of batch jobs for Extraction, Transformation, and Loading (ETL). You want to monitor these jobs in real time. As an Enterprise Manager user, you experience a very long wait on a page load. You want to easily identify what is running on behalf of the page load. At times a PL/SQL job that is required to finish in 24 hours is taking up to 80 hours to run. You need to identify and tune the expensive statements in the job. You want to be alerted if the job is expected to run too long, so that it can be killed, fixed, and restarted before it consumes too many resources.
  + After an upgrade a batch job went from 4.5 hours to 8 hours. You want to capture the performance statistics of the job steps, before and after the upgrade, then compare the two sets and identify the changes, for example, SQL execution plan, resource consumption, degree of parallelism.

SQL> VAR dbop\_eid NUMBER;  
SQL> EXEC :dbop\_eid := DBMS\_SQL\_MONITOR.BEGIN\_OPERATION  
('ORA.MV.refresh', FORCED\_TRACKING => 'Y')  
SQL> SELECT ...  
SQL> SELECT ...  
SQL> EXEC DBMS\_SQL\_MONITOR.END\_OPERATION  
('ORA.MV.refresh', :dbop\_eid)

### Real-Time ADDM



**Emergency Monitoring**

Running Emergency Monitoring should be the final resort before bouncing the database.

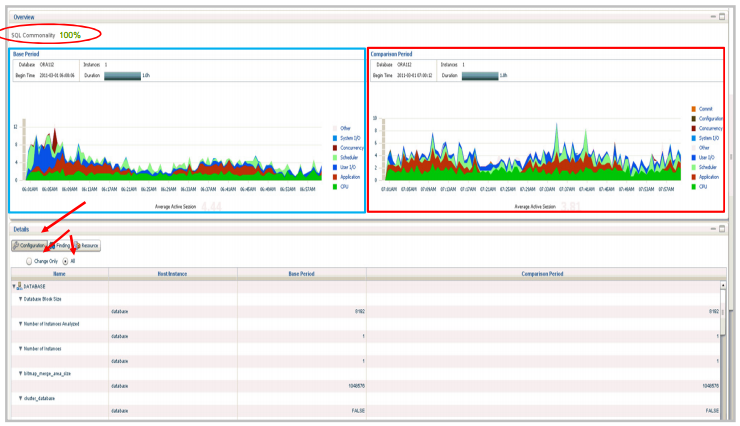
* + Switching to Emergency Monitoring allows you to:
    - Connect to the instance in diagnostic mode
    - View the Emergency Performance page with data collected refreshed in real time
    - View ASH data and Hang Analysis table of top blocking and blocked sessions and deadlocks, refreshed in real time
  + Viewing Hang Analysis data helps you in:
    - Killing the root blocker responsible for hangs or deadlocks
    - Shutting down and starting up the instance

**Real-Time ADDM**

* + Switch to Real-Time ADDM before bouncing the instance.
    - Starts collecting performance data from all database instances
    - Analyzes recent data for systems paralyzed because of severe contention on local or global resources
    - Provides holistic analysis for systems experiencing unusually high (though not severe) database activity
    - Detects findings for the recent activity (past 10 minutes)
    - Offers actionable recommendations
  + Use the recommendations to solve.
  + Return back to regular Performance Monitoring.

Note: Can be invoked for a RAC environment

### Compare Period ADDM



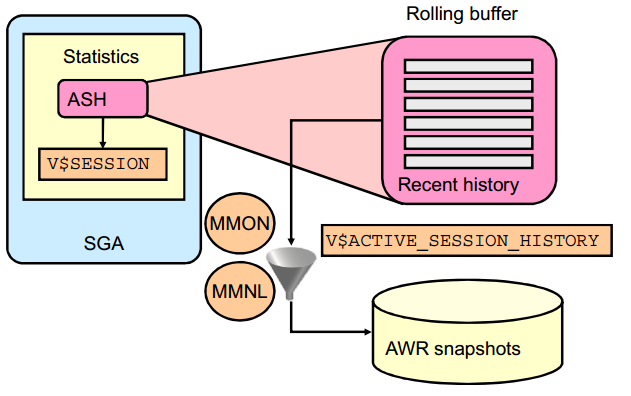
Compare 2 AWR to have a view about the system

The Compare Period ADDM, as opposed to the former method, performs a cause-to-effect analysis.

* + It first identifies the system changes that may have caused the performance change. For example, it detects a configuration change in DB version, or a workload change with SQL changes. These causes can cause performance difference.
  + Then it identifies the effects of these particular changes. For that purpose, it runs an ADDM analysis for the base period and one for the compare period and then measures the differences between both periods.
  + Finally, it maps the effects to the causes with rule sets. For example, an SGA\_TARGET decrease can cause an I/O increase.

Compare Period ADDM provides more than what you had before with former methods, the identification of changes, and an intelligent cause-to-effect analysis.

### Active Session History (ASH)



One of the problems with AWR data is that, by its very nature, analysis of what is currently happening in a system requires detailed information about the activity in the last 5 to 10 minutes. Because the AWR takes snapshots of the system every 60 minutes, the last snapshot could be almost an hour old. Therefore, the AWR does not contain enough information to perform current analysis.

The Active Session History contains the history of recent session activity. Because recording session activity is expensive, ASH samples V$SESSION every second and records the events that the sessions are waiting for. Inactive sessions are not sampled. The sampling facility is very efficient because it directly accesses the internal database structures.

ASH is designed as a rolling buffer in memory; earlier information is overwritten when needed. The ASH statistics are available through the V$ACTIVE\_SESSION\_HISTORY view. This view contains one row for each active session per sample.

Flushing all ASH data to disk is unacceptable because of its volume. The approach is to filter the data while flushing it to disk. This is done automatically by MMON every 60 minutes and by Manageability Monitor Light (MMNL) whenever the buffer is full.

The ASH memory comes from the SGA and is fixed for the lifetime of the instance. It represents 2 MB of memory per CPU. However, the size of ASH cannot exceed 5% of the shared pool size.

## Performance Enhancements

### Extended Data Type

MAX\_STRING\_SIZE = { STANDARD | EXTENDED }

STANDARD: 4000 bytes or character

EXTENDED: 32767 bytes or character

Change to Extended data type

* Shut down the database instance.
* Restart the database in UPGRADE mode.
* Change the setting of MAX\_STRING\_SIZE to EXTENDED;
* Run the *$ORACLE\_HOME/rdbms/admin/utl32k.sql* script as SYSDBA.
* Restart the database instance.

\*\*\*Note: You cannot change the value from EXTENDED to STANDARD.

**SQL Row-Limiting Clause**

# Oracle Database Concept

## Oracle SCN

SCN (System Change Number): đây là một số gia tăng duy nhất trong cơ sở dữ liệu (như đồng hồ thời gian của bạn). Số SCN được tăng lên mỗi 3 giây. Con số này là rất hữu ích trong khi phục hồi cơ sở dữ liệu.

* + SCN xác định một phiên bản đã commit của cơ sở dữ liệu tại một thời điểm. Oracle gán mọi transaction đã commit một SCN.
  + SCN là một dấu thời gian nội bộ cho một commited version của cơ sở dữ liệu . Các máy chủ cơ sở dữ liệu Oracle sử dụng SCN clock để đảm bảo tính nhất quán SCN transaction. Ví dụ, khi một người sử dụng commit một transaction, cơ sở dữ liệu ghi lại một SCN này và commit ghi trên redo log.
  + SCNs là quan trọng đối với transaction bởi vì nó có chức năng như một dấu thời gian đồng bộ commit transaction , ngay cả khi transaction không thành công. Nếu một transaction thay đổi dữ liệu sai hoặc không phù hợp, một quản trị viên có thể sử dụng SCN này để phối hợp thay đổi trên cơ sở dữ liệu. SCN cho transaction commit cũng có thể được sử dụng để xác định các transaction sau đó.
  + Nếu SCN không đồng nhất giữa các thành phần CSDL sẽ nảy sinh vấn đề. Database sẽ lấy SCN của control file và apply cho tất cả các thành phần khác
  + Control file lưu 1 thông tin rất quan trọng là SCN. Con số này được phát sinh và tăng liên tục theo thời gian. Oracle dựa vào con số này đề đồng bộ tất cả các file trong database như: datafile, control file, redo log file v.v...

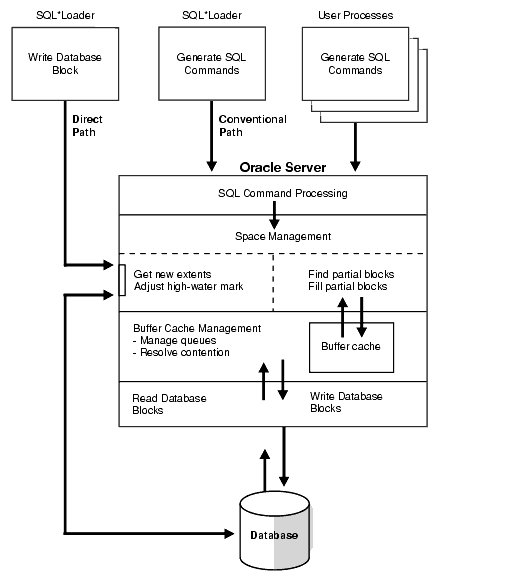
## Oracle Service

An automatic workload management facility, called database services. Database services (services) are logical abstractions for managing workloads in Oracle Database. Services divide workloads into mutually disjoint groupings. Each service represents a workload with common attributes, service-level thresholds, and priorities.

A single service can represent an application, multiple applications or a subset of a single application. For example, the Oracle E-Business suite defines a service for each responsibility, such as general ledger, accounts receivable, order entry, and so on. A single service can be associated with one or more instances of an Oracle RAC database, and a single instance can support multiple services.

## Direct path và conventional path

SQL loader cung cấp 2 phương thức để cập nhật dữ liệu

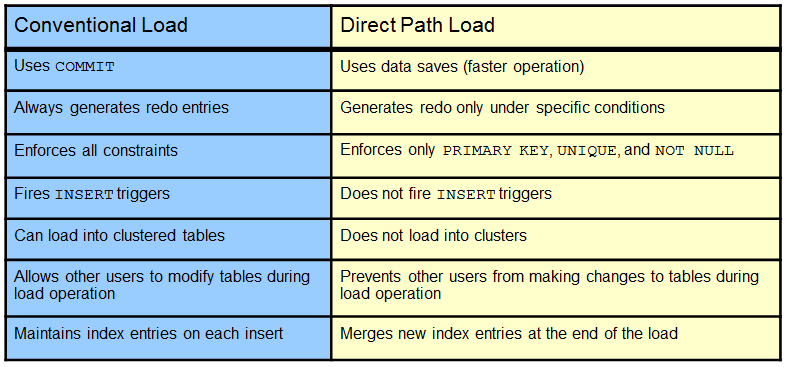


* + Conventional path thực thi câu lệnh insert tới các bảng trong 1 Oracle DB:

INSERT INTO TABLE T PARTITION (P) VALUES ...

* + Direct path ghi trực tiếp lên các datablock trên datafile

LOAD INTO TABLE T PARTITION (P) VALUES ...



## AMM vs ASMM

* + **AMM in Oracle 11g:** The 11g release uses AMM and manages all of the SGA AND PGA via the memory\_targetparameter.
  + **ASMM in Oracle10g:** Oracle ASMM was with Oracle 10g and uses two parameters  sga\_max\_size for the SGA and pga\_aggregate\_targetfor the PGA.
* **Use ASMM better**

## Cluster table

* + Ý tưởng: ghép các table thường xuyên được sử dụng để truy vấn cùng nhau thành 1 cluster
  + Sử dụng cột dùng để join các bảng thành cluster key
  + Đọc thêm dữ trong Tài liệu của hãng

## Quản lý Oracle Resources

* + Ý tưởng chính : cấp quota cho các resource consumer group
  + Đọc thêm trong tài liệu

## Database Link

Sending data or messages between sites requires network configuration on both sites.

You must configure the following:

* + Network connectivity (for example, tnsnames.ora)
  + Database links

SQL> CREATE [PUBLIC] DATABASE LINK <link\_name>

CONNECT TO <user\_name>

IDENTIFIED BY <password>

USING '<service\_name>';

## Data pump

## SQL loader