let's learn basic Oracle database architecture daily .

Day 1
Summary: in this tutorial, you will learn about the Oracle Database architecture and its components.

Oracle Database is an object-relational database management system developed and marketed by Oracle Corporation. Oracle Database is commonly referred to as Oracle RDBMS or simply Oracle.

Oracle.

Database and Instance
An Oracle Database consists of a database and at least one instance.

An instance, or database instance, is the combination of memory and processes that are a part of a running installation and a database is a set of files that store data.

Sometimes, a database instance is referred to as an entire running database. However, it is important to understand the distinctions between the two.

First, you can start a database instance without having it accessing any database files. This is how you create a database, starting an instance first and creating the database from within the instance.

Second, an instance can access only one database at a time. When you start an instance, the next step is to mount that instance to a database. And an instance can mount only one database at a single point in time.

Third, multiple database instances can access the same database. In a clustering environment, many instances on several servers can access a central database to enable high availability and scalability.

Finally, a database can exist without an instance. However, it would be unusable because it is just a set of files.

day 2

Oracle architecture

Connecting to the Database Instance
Connections and sessions are closely related to user processes but are very different in meaning.

A connection is a communication pathway between a user process and an Oracle Database instance. A communication pathway is established using available interprocess communication

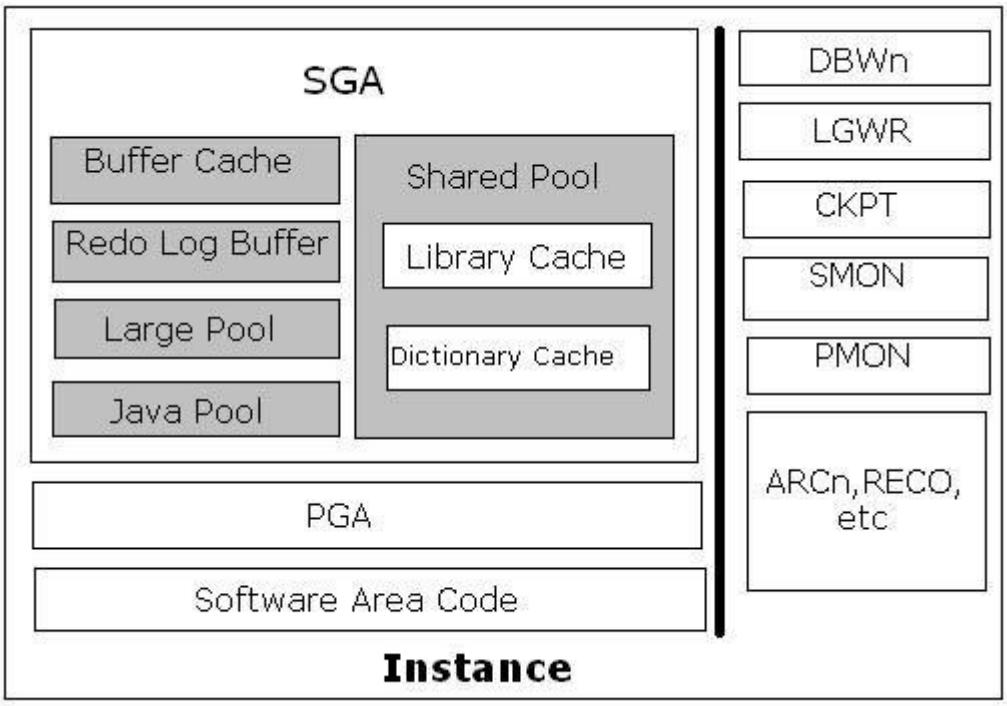
mechanisms (on a computer that runs both the user process and Oracle Database) or network software (when different computers run the database application and Oracle Database and communicate through a network).

A session represents the state of a current user login to the database instance.
For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time a user connects until the user disconnects or exits the database application. Multiple sessions can be created and exist concurrently for a single Oracle database user using the same username.
For example, a user with the username/password of HR/HR can connect to the same Oracle Database instance several times.

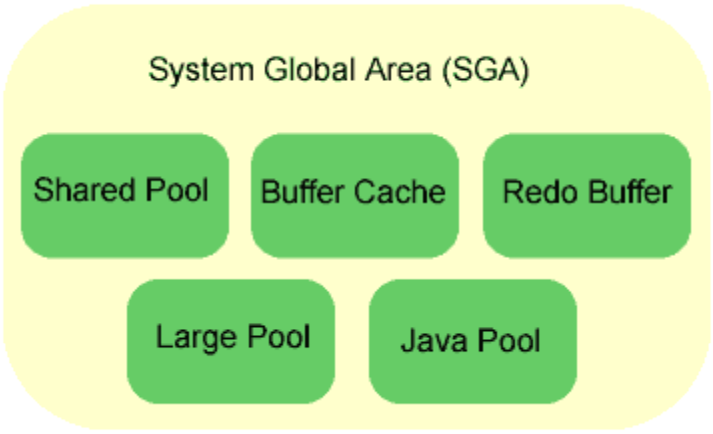
Day 3

Oracle Architecture #oracle #architecture

Instance
Database files themselves are useless without the memory structures and processes to interact with the database. Oracle defines the term instance as the memory structure and the background processes used to access data from a database. The memory structures and background processes continue an instance. The memory structure itself consists of System Global Area (SGA), Program Global Area (PGA). In the other hand, the mandatory background processes are Database Writer (DBWr), Log Writer (LGWR), Checkpoint (CKPT), System Monitor (SMON), and Process Monitor (PMON). And another optional background processes are Archiver (ARCn), Recoverer (RECO), etc.



Day 4
Let us see about SGA which is Part of Oracle Architecture ,ORACLE Memory Structures.
There are five memory structures that make up the System Global Area (SGA). The SGA will store many internal data structures that all processes need access to, cache data from disk, cache redo data before writing to disk, hold parsed SQL plans and so on.



Day 5
Let us see about SGA each component which is Part of Oracle Architecture ,ORACLE Memory Structures.

Let us go through particular component called shared pool from sga

#Shared Pool
What are the components of Oracle shared pool?

The main components of the shared pool are
Session Memory
Library Cache
Dictionary Cache

Session Memory
memory allocated for session variables, such as logon information, and other information required by a database session.

Library Cache
The library cache is not one cache, but many. It contains the pseudo code for PL/SQL program units. It contains parse trees and execution plans for shareable SQL statements.

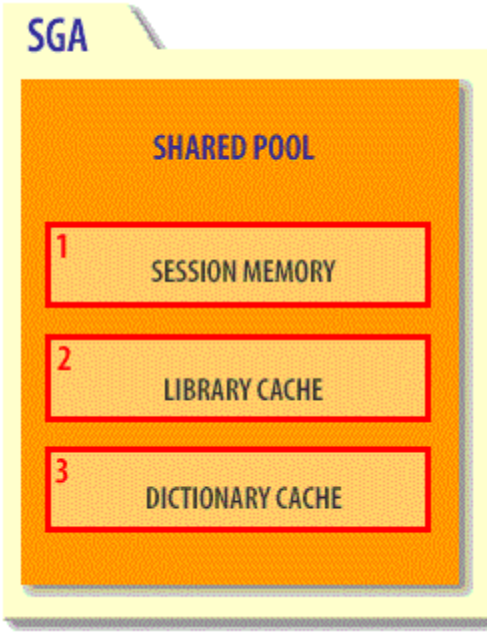
Dictionary Cache
Dictionary Cache is the memory areas designated to hold dictionary data. It is also known as the row cache because it holds data as rows instead of buffers (which hold entire blocks of data). When an Oracle database is first started, no data exists in the data dictionary cache.

How hash value is generated in Oracle?
When we execute any sql statement in Oracle, a hash value is being assigned to that sql statement and stored into the library cache. So, that later, if another user request the same query, then Oracle find the hash value and execute the same execution plan.

What Is Syntactic and Semantic Checking?
Rules of syntax specify how language elements are sequenced to form valid statements. Thus, syntactic checking verifies that keywords, object names, operators, delimiters, and so on are placed correctly in your SQL statement.

Rules of semantics specify how valid external references are made. Thus, semantic checking verifies that references to database objects and host variables are valid and that host-variable datatypes are correct.

How to check shared pool free memory?
SELECT * FROM v$sgastat WHERE name = 'free memory';



Day 6

Database Buffer Cache

Let us see why and what is Oracle Buffer Cache is for in Oracle Architecture.

Why read into the buffer cache?

The buffer cache is the in-memory area of the SGA where incoming Oracle data blocks are kept. On standard Unix databases, the data is read from disk into the Unix buffer where it is then transferred into the Oracle buffer. The size of the buffer cache can have a huge impact on Oracle system performance.

Here is 3 status of buffers.

*Free

*Pinned

*Dirty

what is Free block?

This event occurs mainly when a server process is trying to read a new buffer into the buffer cache but too many buffers are either pinned or dirty and thus unavailable for reuse. The session posts to DBWR then waits for DBWR to create free buffers by writing out dirty buffers to disk.

what is pinned block?

A buffer pin is like a latch (cache buffer chains), and the buffer is pinned count defines a estimate of the efficiency of rare re-visits to the same data buffer block. In plain English, a session may "pin" a handle to a data block to re-visit the data block at a later time without having to perform a second logical read.

Oracle does not always pin a buffer block, and these types of non-latched reads are recorded as "consistent gets - examination" and "buffer is not pinned count" in the AWR report. See my notes on consistent gets.

Oracle has several hidden parameters that define the buffer pin operations:

_buffer_handles_cached: This defines the total number of concurrent pins allowed and the default is 5.

_db_handles: This is he total number of "pins" that can be reserved by any database session.

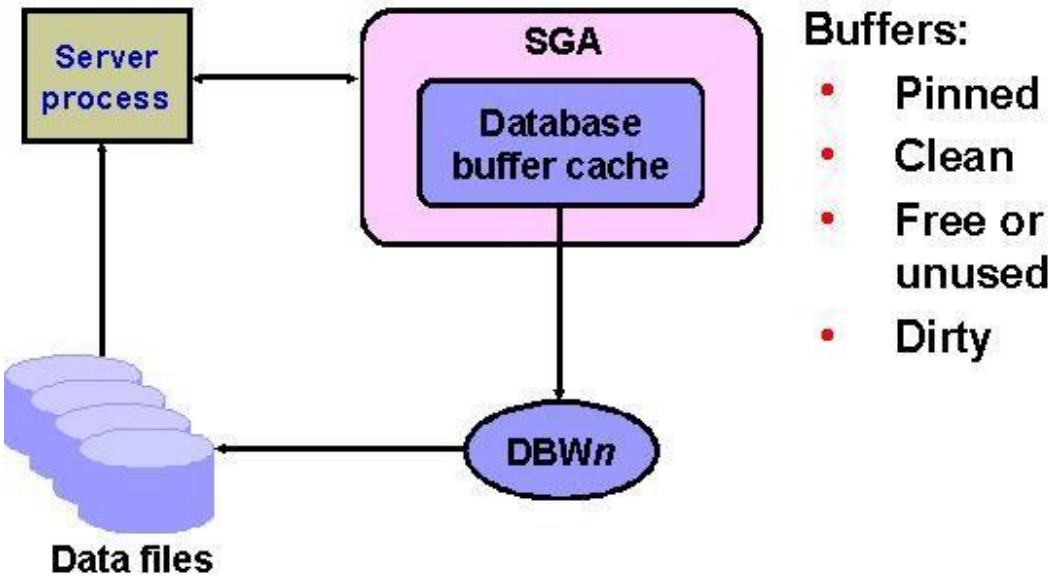
What Is a Dirty Block?

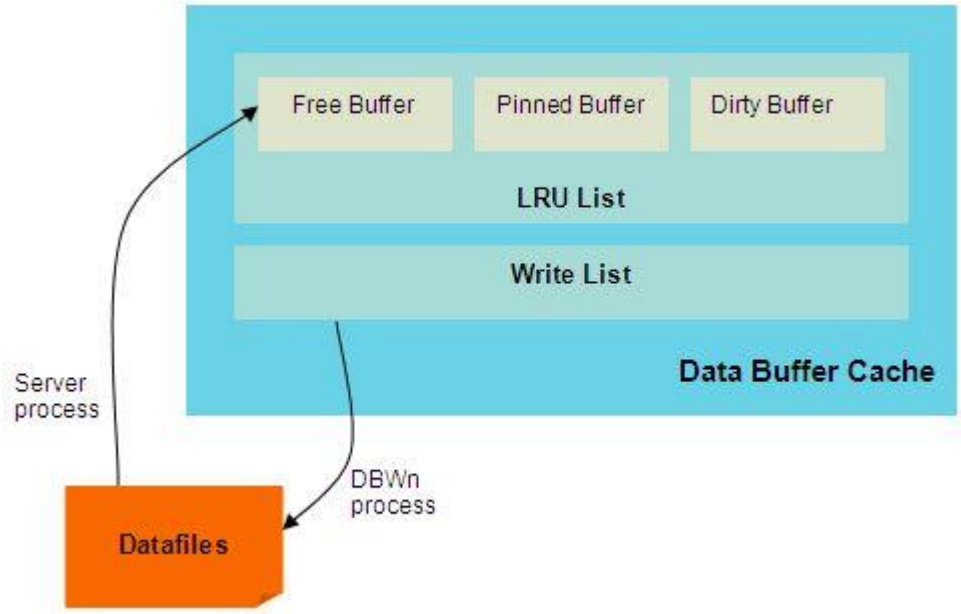
Whenever a server process changes or modifies a data block, it becomes a dirty block. Once a server process makes changes to the data block, the user may commit transactions, or transactions may not be committed for quite some time. In either case, the dirty block is not immediately written back to disk.

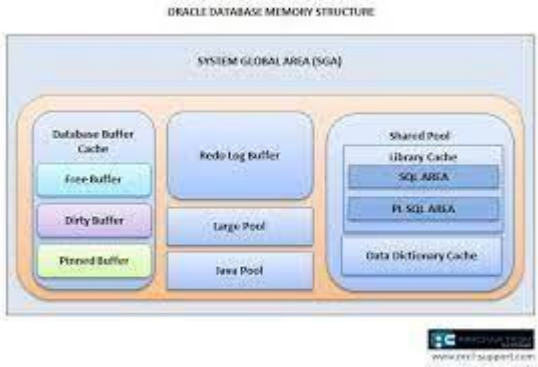
Writing dirty blocks to disk takes place under the following two conditions:

* When a server process cannot find a clean, reusable buffer after scanning a threshold number of buffers, then the database writer process writes the dirty blocks to disk.

* When the checkpoint takes place, the database writer process writes the dirty blocks to disk.







Day 7

#ORACLE #Architectre #Redologbuffercache

What is Redo log buffer cache?
The redo log buffer is the part of the System Global Area (SGA) that holds information about changes made to the database. Each of these changes generates a 'redo entry'. Redo entries are needed to reconstruct these changes during the recovery process. Redo log buffers can have the commit and non commit data for roll forward and roll backward .

How Oracle Allocates Space in the Redo Log Buffer?
As each user session makes changes to the database, these changes are first recorded in the redo log buffer. The following steps approximate the

process:
The session will acquire the redo allocation latch.
The session will allocate the memory it needs from the redo log buffer for the copy of the redo.
The redo allocation latch is released and a redo copy latch is acquired, if available. If there is not a redo copy latch available, then the redo allocation latch is held until the end of the operation.
The redo is copied to the redo log buffer.
The redo copy latch is released.

The redo allocation latch is acquired to allocate memory space in the log buffer. The number of redo allocation latches is determined by database parameter LOG_PARALLELISM (this parameter has been deprecated in Oracle). The redo allocation latch allocates space in the log buffer cache for each transaction entry. If transactions are small, or if there is only one CPU on the server, then the redo allocation latch also copies the transaction data into the log buffer cache. If a log switch is needed to get free space, this latch and the redo copy latch are released.
What is the difference between redo log buffer and redo log file?
So for the recovery purpose redo log files will have exact committed data or recoverable data. redo log buffers can have the commit and non commit data for roll forward and roll backward .

Day 8

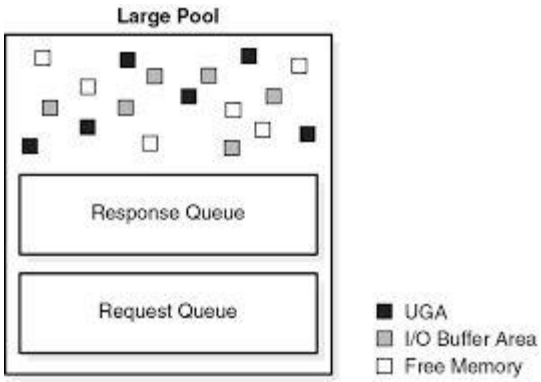
As we already done with sga components in Oracle architecture
like #Sharedpool #Dbbc(Database Buffer cache) #Rlbc(Redo log buffer cache). We are moving to rest of components like #Largepool and #Javapool

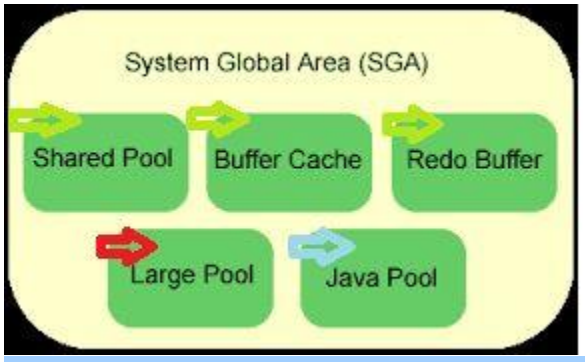
Today we are about see Large Pool.

The large pool is an optional memory area. It provides an area of memory from which large allocations can be made. Oracle's backup and restore utilities (eg: Rman backup and restore) typically allocate buffers that are hundreds of kilobytes in size. These will be allocated in the large pool if one is present.

How to check large pool size in Oracle?
By issuing a summation select against the v$sgastat view, a DBA can quickly determine the size of the large pool area currently being used. SELECT name, SUM(bytes) FROM V$SGASTAT WHERE pool='LARGE POOL' GROUP BY ROLLUP(name);

How to set Large_pool_size in Oracle?
ALTER SYSTEM SET large_pool_size=100M scope=both; ALTER SYSTEM SET shared_pool_reserved_size=512M scope=spfile; 3. If it not set then increase the SGA_TARGET or MEMORY_TARGET parameter as you set which will allocate more space.





Completed
In progress
Yet to start

Day 9
#OracleArchitecture

#Javapool
What does the Java pool do inside the Oracle SGA, and how di I sze the Java pool properly?

Answer: The java pool is a RAM region within the Oracle SGA and the java pool RAM is used to provide:

Parsing of Java code and scripts
Installation tasks related to Java applications with Oracle 11g
Java stored procedure code parsing
The JAVA Pool holds the JAVA execution code in a similar manner to the PL/SQL cache in the shared pool. The JAVA pool is used by many internal routines, such as import and export, and should be sized at approximately 60 megabytes if no other JAVA will be utilized in the user applications.

Tuning and sizing the Java pool
Like all Oracle pools, the Java Pool uses a least-recently-used algorithm to ensure that the most "popular" Java stays pinned in RAM. If the Java pool is undersized, Java performance may degrade because of the need to re-claim RAM space which has been aged out of the Java Pool.

With the Oracle JVM (Java Virtual Machine, you can guarantee that Java stays in RAM by placing Java in an Oracle stored procedure and pinning the package in the SGA with

dbms_shared_pool.keep. See Pinning Packages Objects in the Shared Pool.

In order to tune the best size for the Java pool within Oracle, either the AMM feature can be deployed or manual tuning can be conducted via the v$javapool and v$java_pool_advice dynamic performance views as shown in the following example listings:

```
SQL> desc v$javapool

 Name                  Null?  Type
 ----------------------------------------- CATEGORY                  VARCHAR2(16)
 MEMUSED                   NUMBER

SQL> select * from v$javapool;

CATEGORY      MEMUSED
--------------- ----------
         8755260
```

The dynamic performance view v$java_pool_advice provides details for tuning the Java pool for different pool sizes. These sizes range from 10% of the current Java pool size all the way to up to 200% of the Java pool size.

```
SQL> desc v$java_pool_advice

 Name                  Null?  Type
 ----------------------------------------- -------- ---------------
-   JAVA_POOL_SIZE_FOR_ESTIMATE           NUMBER
 JAVA_POOL_SIZE_FACTOR             NUMBER
 ESTD_LC_SIZE                  NUMBER
 ESTD_LC_MEMORY_OBJECTS            NUMBER
 ESTD_LC_TIME_SAVED                NUMBER
 ESTD_LC_TIME_SAVED_FACTOR           NUMBER
 ESTD_LC_LOAD_TIME             NUMBER
 ESTD_LC_LOAD_TIME_FACTOR            NUMBER
 ESTD_LC_MEMORY_OBJECT_HITS          NUMBER
```

Day 10
#ORACLE BACKGROUND PROCESS

Hi All
Last week we saw what are in memory structure and this week let us see about background process.

Key Background Processes To Be Known In ORACLE.

A multiprocessor Oracle database system uses background processes. Background processes are

the processes running behind the scene and are meant to perform certain maintenance activities or to deal with abnormal conditions arising in the instance. Each background process is meant for a specific purpose and its role is well defined.

Background processes consolidate functions that would otherwise be handled by multiple database programs running for each user process. Background processes asynchronously perform I/O and monitor other Oracle database processes to provide increased parallelism for better performance and reliability.

A background process is defined as any process that is listed in V$PROCESS and has a non-null value in the pname column.

Not all background processes are mandatory for an instance. Some are mandatory and some are optional. Mandatory background processes are DBWn, LGWR, CKPT, SMON, PMON, and RECO. All other processes are optional, will be invoked if that particular feature is activated.

Oracle background processes are visible as separate operating system processes in Unix/Linux. In Windows, these run as separate threads within the same service. Any issues related to background processes should be monitored and analysed from the trace files generated and the alert log.
Background processes are started automatically when the instance is started.

Rest of about each and every background process let us see by upcoming days.

Day 11
Let us see about one of the background process in oracle database today.

#ORACLEARCHITECTURE

#ORACLEBACKGROUNDPROCESS

#SMON

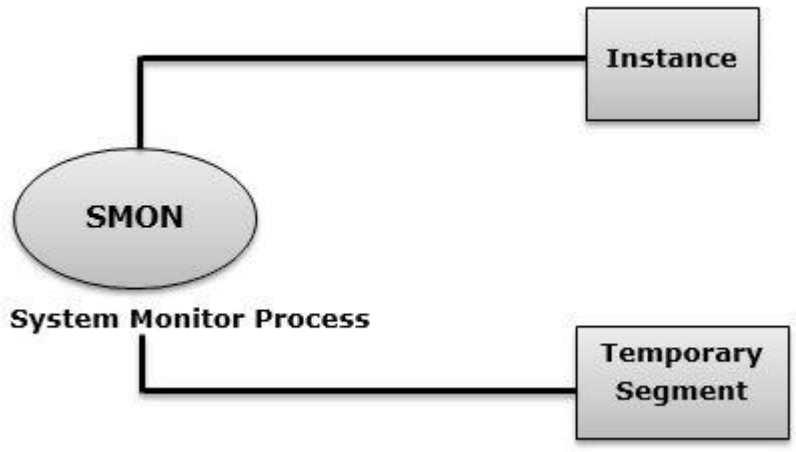
What is smon in detail?
System Monitor Process (SMON)
The system monitor process (SMON) performs recovery, if necessary, at instance startup. SMON is also responsible for cleaning up temporary segments that are no longer in use and for coalescing contiguous free extents within dictionary managed tablespaces. 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. SMON checks regularly to see whether it is needed. Other processes can call SMON if they detect a need for it. With Real Application Clusters, the SMON process of one instance can perform instance recovery for a failed CPU or instance.

The system monitor process (SMON) is in charge of a variety of system-level cleanup duties. The duties assigned to SMON include:
Performing instance recovery, if necessary, at instance startup. In an Oracle RAC database, the SMON process of one database instance can perform instance recovery for a failed instance.
Recovering terminated transactions that were skipped during instance recovery because of file-read or tablespace offline errors. SMON recovers the transactions when the tablespace or file is brought back online.
Cleaning up unused temporary segments. For example, Oracle Database allocates extents when creating an index. If the operation fails, then SMON cleans up the temporary space.
Coalescing contiguous free extents within dictionary-managed tablespaces.
SMON checks regularly to see whether it is needed. Other processes can call SMON if they detect a need for it.

Crash recovery
Remember that Oracle buffers data in the SGA, and does not necessarily write out changes to the datafiles immediately after they occur. If the instance crashes, any changes buffered in memory are lost. Whenever you start an Oracle instance, SMON checks to see if it has previously crashed. If it has, then SMON reads the redo log files, and applies any needed changes to the datafiles.

Cleaning up temporary spaces
Temporary segments are often used for sorting. When a sort finishes, SMON takes care of deallocating any segments used for that sort.

Coalescing free space
Coalescing free space refers to the practice of taking two adjacent areas of free space with a datafile and converting them into one larger area of free space. This procedure is repeated over and over in order to minimize fragmentation in the files.



Day 12
Let us see about one of the background process in oracle database today.

#ORACLEARCHITECTURE

#ORACLEBACKGROUNDPROCESS

#PMON

PMON (Process MONitor) is an Oracle background process created when you start a database instance. The PMON process will free up resources if a user process fails (eg. release database locks).

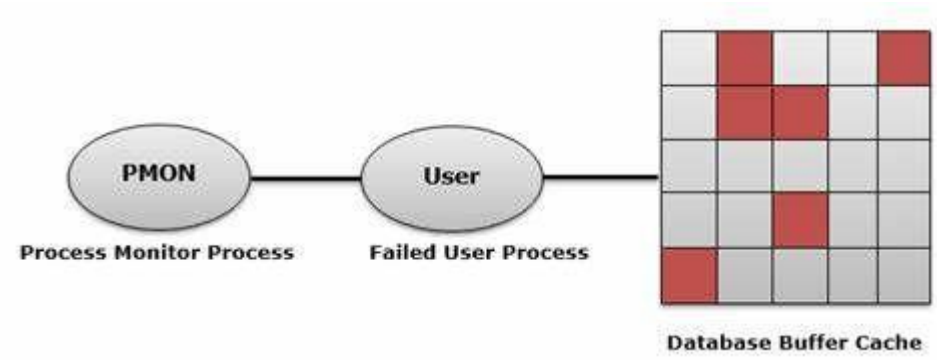
PMON normally wakes up every 3 seconds to perform its housekeeping activities. PMON must always be running for an instance. If not, the instance will terminate.

To speed-up housekeeping, one may also wake-up PMON (process 2 below) manually:

SQL> oradebug setmypid
SQL> oradebug wakeup 2
In Oracle releases prior to Oracle 12c, PMON also registered database serviced with the listener. This is now handled by the new LREG progress.

Check process
The following Unix/Linux command is used to check if the PMON process is running:

$ ps -ef | grep pmon
oracle 31144   1 0 11:10 ?   00:00:00 ora_pmon_orcl



Refresher on Oracle Architecture

Day 13 - **How the Log Writer Process (LGWR) Keeps Your Oracle Database Running Smoothly**

The log writer process (LGWR) is responsible for redo log buffer management–writing the redo log buffer 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.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer

to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy.

LGWR writes one contiguous portion of the buffer to disk. LGWR writes:

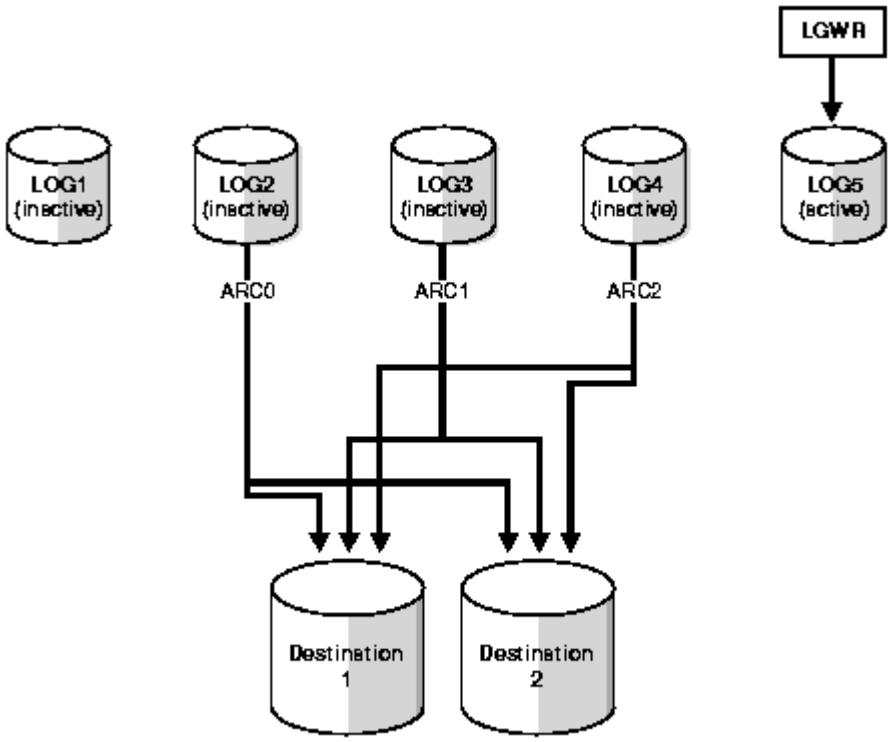
A commit record when a user process commits a transaction
Redo log buffers
*Every three seconds
*When the redo log buffer is one-third full
*When a DBWn process writes modified buffers to disk, if necessary

Note: Before DBWn can write a modified buffer, all redo records associated with the changes to the buffer must be written to disk (the write-ahead protocol). If DBWn finds that some redo records have not been written, it signals LGWR to write the redo records to disk and waits for LGWR to complete writing the redo log buffer before it can write out the data buffers.

LGWR writes synchronously to the active mirrored group of online redo log files. If one of the files in the group is damaged or unavailable, LGWR continues writing to other files in the group and logs an error in the LGWR trace file and in the system alert file. If all files in a group are damaged, or the group is unavailable because it has not been archived, LGWR cannot continue to function.

We discussed about one of the background process in oracle database today. Lets keep understanding more regularly.

#ORACLEARCHITECTURE #ORACLEBACKGROUNDPROCESS #LGWRPart 1 #jobseekers #oracledatabase #oracledba #databaseadministration #databaseadministration



Day 14 - **How the Log Writer Process (LGWR) Keeps Your Oracle Database Running Smoothly**

#ORACLEARCHITECTURE

#ORACLEBACKGROUNDPROCESS

#LGWR PART2
When a user issues a COMMIT statement, LGWR puts a commit record in the redo log buffer and writes it to disk immediately, along with the transaction's redo entries. The corresponding changes to data blocks are deferred until it is more efficient to write them. This is called a fast commit mechanism. The atomic write of the redo entry containing the transaction's commit record is the single event that determines the transaction has committed. Oracle returns a success code to the committing transaction, although the data buffers have not yet been written to disk.

Note: Sometimes, if more buffer space is needed, LGWR writes redo log entries before a transaction is committed. These entries become permanent only if the transaction is later committed.

When a user commits a transaction, the transaction is assigned a system change number (SCN), which Oracle records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can be synchronized in Oracle9i Real Application Clusters and distributed databases.

In times of high activity, LGWR can write to the online redo log file using group commits. For example, assume that a user commits a transaction. LGWR must write the transaction's redo entries to disk, and as this happens, other users issue COMMIT statements. However, LGWR cannot write to the online redo log file to commit these transactions until it has completed its previous write operation. After the first transaction's entries are written to the online redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than do transaction entries handled individually. Therefore, Oracle minimizes disk I/O and maximizes performance of LGWR. If requests to commit continue at a high rate, then every write (by LGWR) from the redo log buffer can contain multiple commit records.

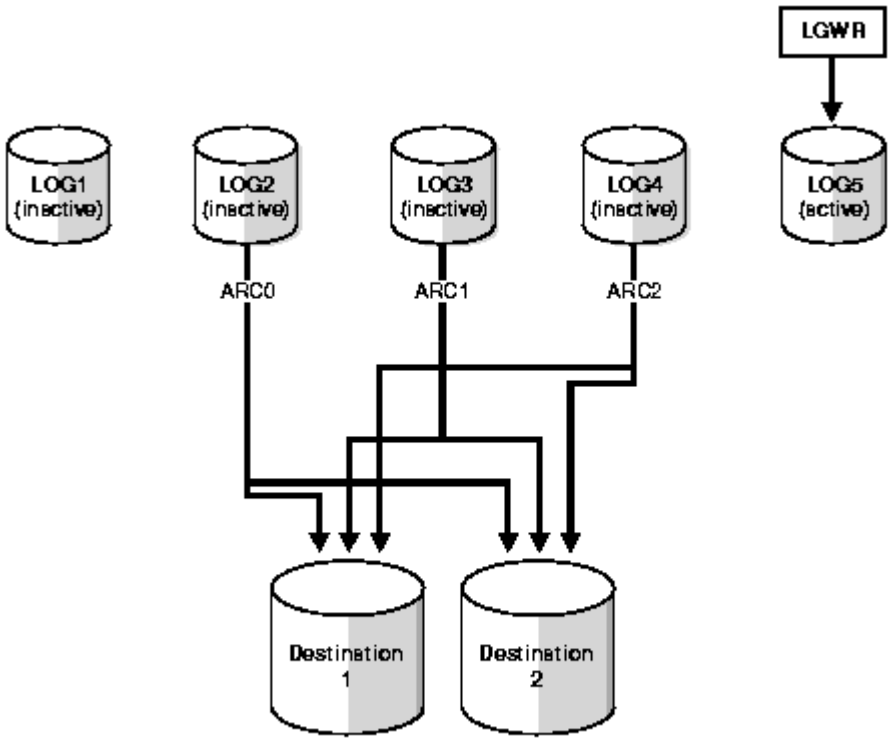
Day 15
Let us see about one of the background process in oracle database today.

#ORACLEARCHITECTURE

#ORACLEBACKGROUNDPROCESS

#DBWR

The Database Writer process, also known as DBWR, has a critical task of writing data to physical database files in Oracle. Whenever data is altered within the Oracle database, it is temporarily held in a memory buffer and marked as "dirty". The role of DBWR is to move this data from the buffer to the physical disk files, keeping the buffer cache clean and ensuring an adequate supply of free buffers. DBWR utilizes a least recently used (LRU) algorithm to keep frequently accessed buffers in memory, thereby reducing the number of I/O operations required while writing the less frequently accessed buffers to disk. The buffers are written to disk when a checkpoint is signaled, when the dirty list reaches a certain threshold length, or when there are no available free or reusable buffers. In addition to the required DBWR process, extra processes can be created to boost performance.



Day 16

#ORACLEARCHITECTURE
#ORACLEBACKGROUNDPROCESS
#Checkpoint

In this discussion, we're exploring the concept of checkpoints in an Oracle database. A checkpoint is a mechanism that defines the System Change Number (SCN) in the redo thread of the database, which is recorded in the control file and each data file header. This is crucial for recovery purposes. The CKPT process is responsible for checkpoint operation, and when a checkpoint occurs, Oracle must update the header of all data files to record the checkpoint details. The checkpoint information includes the checkpoint position, SCN, location in the online redo log to begin recovery, and more.

Checkpoints reduce the time required for recovery in the event of an instance or media failure. They ensure that dirty buffers in the buffer cache are regularly written to disk and that all committed data is written to disk during a consistent shutdown.

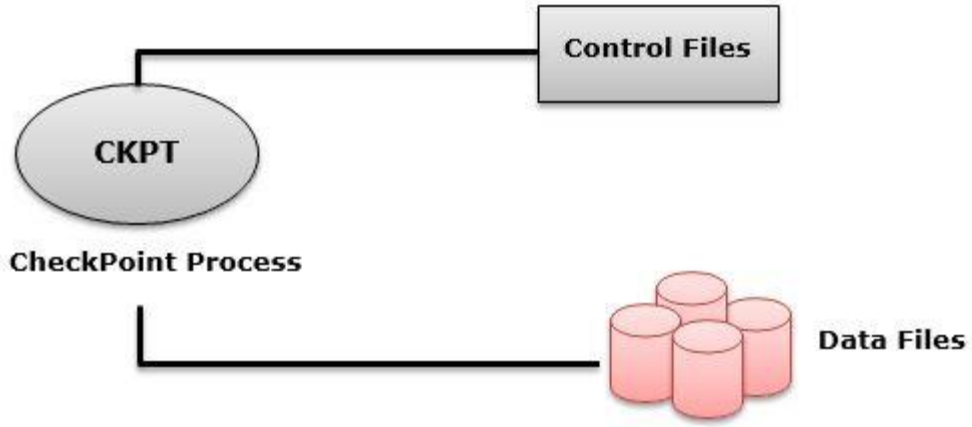
Checkpoints occur at each switch of logfiles, when the delay for LOG_CHECKPOINT_TIMEOUT is reached, when the size of bytes corresponding to (LOG_CHECKPOINT_INTERBAL * size of IO OS block) is written on the current redo log file, and when the ALTER SYSTEM SWITCH LOGFILE command is issued.

Thread checkpoints are the most commonly referred to type of checkpoint. They guarantee that all checkpoint-SCN details of all online data files have been written to disk. The database writes to disk all buffers modified by redo in a specific thread before the Thread Checkpoint SCN. The

set of thread checkpoints on all instances in a database is a database checkpoint. Thread checkpoints occur in situations such as consistent database shutdown, ALTER SYSTEM CHECKPOINT statement, online redo log switch, and ALTER DATABASE BEGIN BACKUP statement.

Tablespace and data file checkpoints occur when the database writes to disk all buffers modified by redo before a specific target. A tablespace checkpoint is a set of data file checkpoints, one for each data file in the tablespace. These checkpoints occur in various situations, such as making a tablespace read-only or taking it offline normal, shrinking a data file, or executing ALTER TABLESPACE BEGIN BACKUP.

Incremental checkpoints are a type of thread checkpoint intended to avoid writing large numbers of blocks at online redo log switches. DBWn checks at least every three seconds to determine if it has work to do. When DBWn writes dirty buffers, it advances the checkpoint position, causing CKPT to write the checkpoint position to the control file, but not to the data file headers. Other types of checkpoints include instance and media recovery checkpoints and checkpoints when schema objects are dropped or truncated.



Day 17

Today let us see about both archiver and Reco background Process together from oracle.

#oraclearchitecture

#oraclebackgoundprocess

#ARCn
Archiver processes copy the online redo log files to archival storage when the log files are full or a log switch occurs. The database must be in archive log mode to run archive processes.

#RECO
The recoverer background process (RECO)

RECO is a background process for distributed transactions. The RECO process manager two-

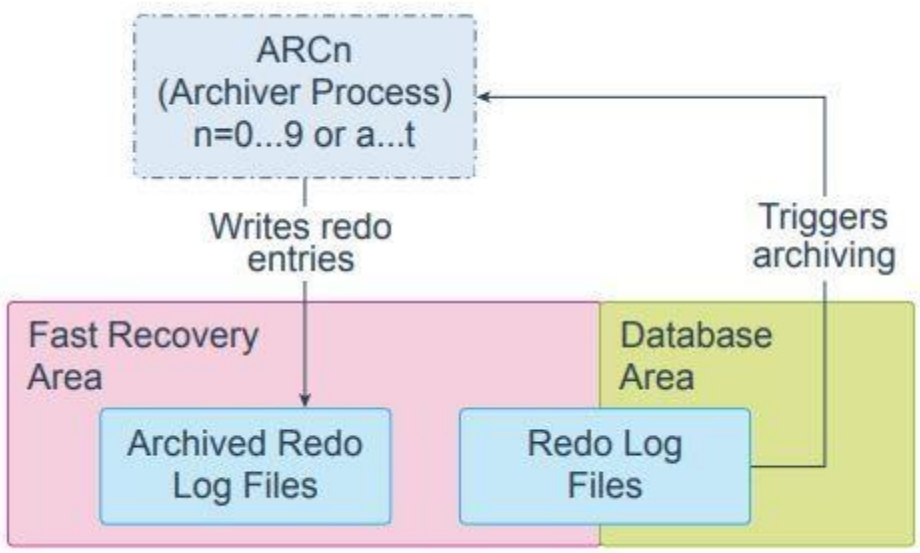
phase commits to track and resolve in-doubt transactions.

In UNIX, the RECO process awakens whenever a single transaction spans two geographical locations. For example, the following transaction requires a two-phase commit:

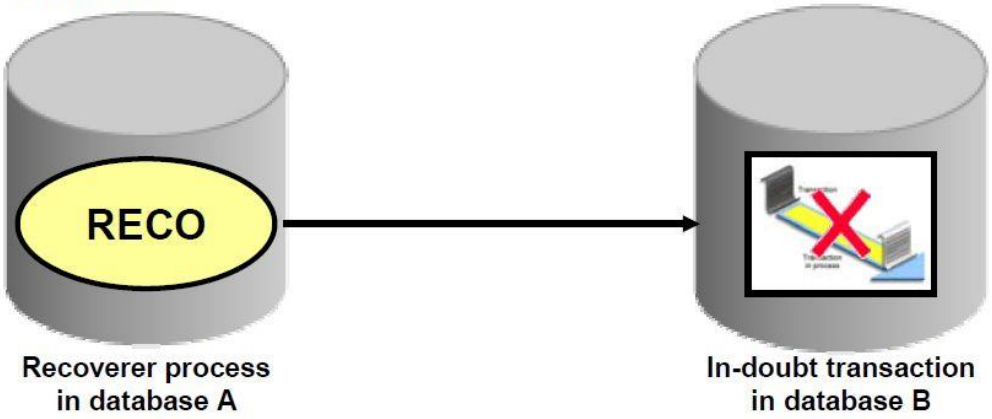
```
Update  -- Local update transaction

emp set sal = sal * 1.1
where
dept = 30;

update  -- Remote update transaction

emp@new_york
set
sal = sal * 1.1
where
dept = 40;
```

In this case, the RECO process will ensure that both the remote update and the local update complete or rollback as a single unit.

You will see the RECO process running whenever the distributed option of Oracle is installed, but the RECO process will only show activity when remote updates are being executed as part of a two-phase commit.



- Removes any rows that correspond to in-doubt transactions



Day 18

#Oraclememorystructure
#Oraclearchitecture

Over the past three weeks, we have learned about instance and background processes in the **Oracle database architecture**. Now, let's take a look at the **physical memory structure** in an Oracle database.

Let's look at Oracle **tablespaces** and how they are used to store data in an Oracle database logically.

Oracle divides a database into one or more tablespaces, which are logical storage units. Each tablespace consists of one or more data files, which physically store data objects such as tables and indexes on disk.

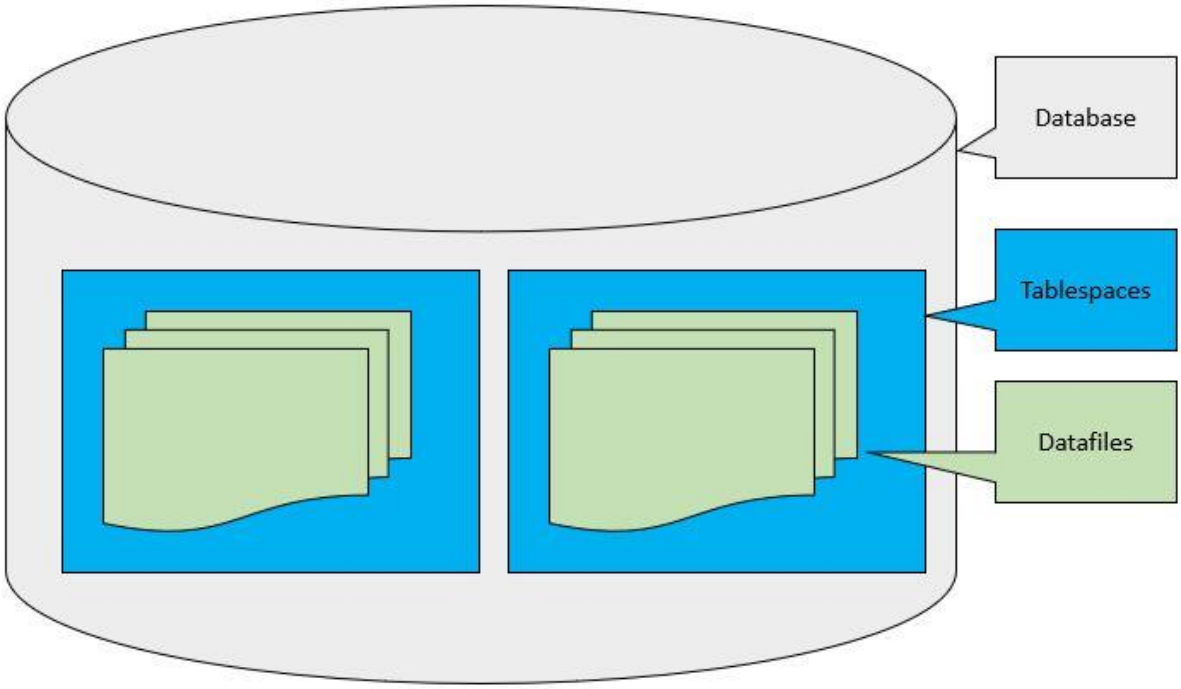
Essentially, Oracle stores data logically in tablespaces and physically in data files that are associated with the corresponding tablespaces.

Using tablespaces, you can control the storage size allocated for the database data, grant specific space quotas to database users, and take tablespaces online or offline to control data availability. You can also allocate data storage across devices, perform partial database backup or recovery, and more.

Oracle comes with several default tablespaces, including **SYSTEM**, **SYSAUX**, **USERS**, **UNDOTBS**1, **and TEMP**. These tablespaces store system-generated objects, data dictionary tables, ad-hoc user data, undo data, and temporary data, respectively.

Tablespaces can be online or offline, and the SYSTEM tablespace must always be online because it contains the data dictionary that must be available to Oracle. You can take a tablespace offline for maintenance, but attempting to access data in an offline tablespace will result in an error.

Oracle also allows **read-only tablespaces**, which enable the avoidance of backup and recovery for large, static parts of a database. Objects such as tables and indexes can be removed from read-only tablespaces, but new objects cannot be created or altered within them. To create a read-only tablespace, use the ALTER TABLESPACE command with the READ ONLY option.



Day 19

#Oraclememorystructure
#Oraclearchitecture
#Redologfile

What is the purpose of redo log files in the database?

Redo log files are a crucial component of Oracle Database's architecture. They are used to record all changes made to the database to facilitate recovery in case of a system failure.

When changes are made to the database, such as inserting, updating, or deleting records, these changes are first written to the redo log buffer in memory. The redo log buffer is periodically written to the redo log files on the disk. This process is called a log switch.

Redo log files are physical files that are stored on a disk, separate from the data files. The redo log files are created when the database is created, and they must be multiplexed, meaning that multiple copies of each redo log file must be created and stored on different physical disks.

In the event of a system failure, the redo log files are used to roll forward any changes that were made to the database before the failure occurred. This ensures that the database is brought back to a consistent state, with all changes up to the point of failure applied.

In summary, redo log files are a critical component of Oracle Database's architecture that are used to record all changes made to the database to facilitate recovery in case of a system failure.

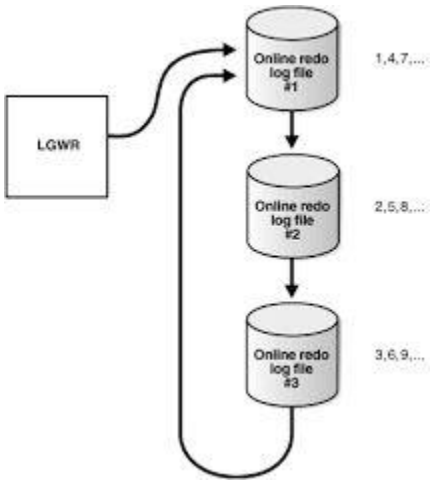
How do I read a redo log in Oracle?

LogMiner retrieves information from those redo log files and returns it through the **V$LOGMNR_CONTENTS** view. You can then use SQL to query the **V$LOGMNR_CONTENTS** view, as you would any other view. Each select operation that you perform against the **V$LOGMNR_CONTENTS** view causes the redo log files to be read sequentially.

How many types of redo log files are there?
two types
Redo log files occur in two types: online redo logs ("ORL" or "redo logs" for short) and archived redo logs ("archive logs")

The **LOG_BUFFER_SIZE** and **LOG_BUFFERS** parameters control the redo log buffers. The **LOG_BUFFER_SIZE** should be set to reduce the number of writes required per redo log but not be so large that it results in an excessive IO wait time. Some studies have shown that a size bigger than one megabyte rarely results in performance gains. Generally, I size the LOG_BUFFER_SIZE such that it is equal to or results in an even divisor of the redo log size.

Monitor redo logs using the alert log, **V$LOGHIST**, **V$LOGFILE**, **V$RECOVERY_LOG**, **and V$LOG** DPTs.



Day 20

#CONTROLFILE
#ORACLEARCHITECTURE

Today Let's see what is called Controlfile in the Oracle database and what is inside the control file is available.

In Oracle, a control file is a binary file that is used to manage and coordinate the physical

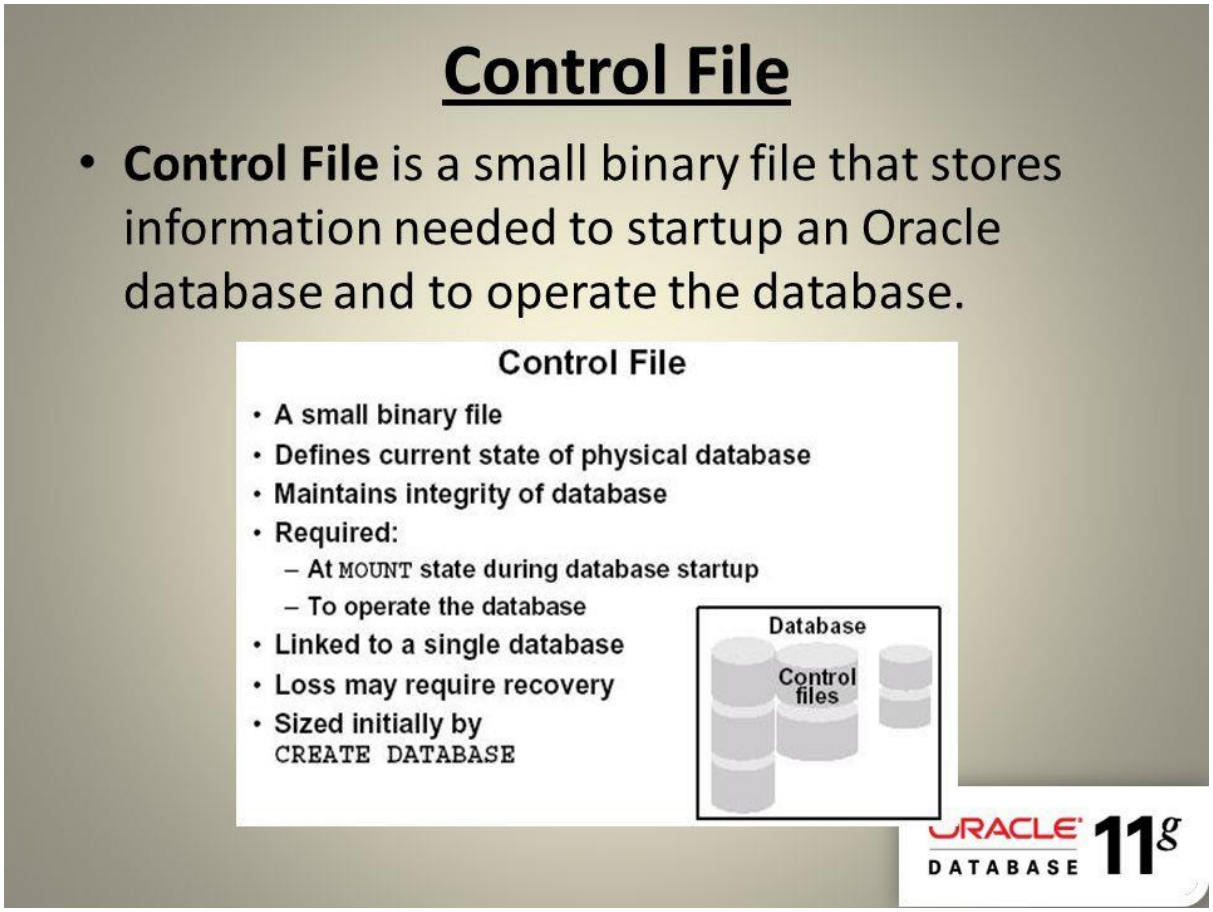
structures of a database. It contains critical information about the database, such as the database name, the physical structure of the database, and the current state of the database.

The control file is created when a database is created and is typically stored in the Oracle database's data directory. The control file is read by the Oracle instance during database startup to determine the database's configuration and to ensure data consistency.

The information contained in a control file includes:

==>Database name and a unique identifier
==>Time stamp of database creation
==>Tablespace information, including the data files and redo log files associated with each tablespace
==>Current state of the database (mount, open, or closed)
==>Information about archived logs and backups
==>Current sequence numbers of redo logs
==>Checkpoint information

The control file is critical to the operation of an Oracle database and should be backed up regularly to ensure data integrity and availability in the event of a system failure.



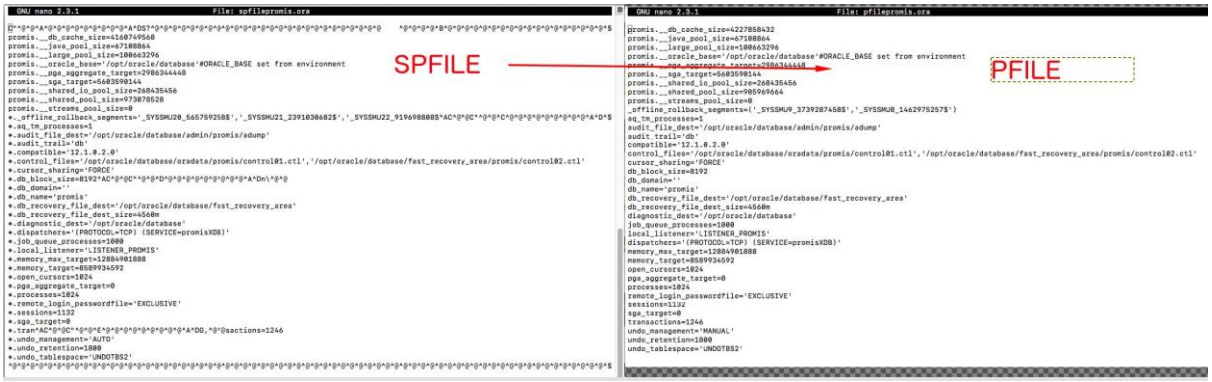
Day21

Let us see today what is pfile and spfile in oracle.

#ORACLEARCHITECTURE
#ORACLEPFILE
#ORACLESPFILE

Connecting to an Oracle database requires a valid Oracle Net Service Name that identifies the database. While there is no mandatory parameter in the Oracle PFILE specifically required for establishing a connection to the database, some important initialization parameters must be set correctly in the PFILE. These include DB_NAME, CONTROL_FILES, LOG_ARCHIVE_DEST_n, and SHARED_POOL_SIZE. Once these parameters are set in the PFILE, the Oracle instance can be started, and the database can be accessed through the Oracle Net Service Name.

To start an Oracle database, the "sga_target" parameter is mandatory. The SGA is a shared memory area used by the Oracle instance to store data and control information. It caches data blocks, stores shared SQL areas and other important information required by the Oracle database. The size of the SGA depends on the database size and expected workload. The "startup" command initializes the Oracle instance by reading initialization parameters from a parameter file (PFILE or SPFILE). If the "sga_target" parameter is not set in the parameter file, the startup command will fail.



Day 22
#ORACLEPHYSICALFILE
#ORACLEARCHITECTURE
#ORACLEPASSWORDFILE

what is a password file in the oracle database?
In an Oracle database, the password file is a file that stores a list of usernames and passwords for users who are allowed to connect to the database with administrative privileges, such as the SYS and SYSTEM accounts.

The password file is created using the orapwd utility and is stored in a location specified by the

parameter "REMOTE_LOGIN_PASSWORDFILE" in the database initialization file. By default, the password file is named "orapwSID" where SID is the system identifier of the database.

The password file provides an additional layer of security for the database as it restricts access to administrative privileges only to authorized users. When the password file is used, users must provide a valid username and password to connect to the database with administrative privileges.

It is recommended to use a password file in production environments to secure the database from unauthorized access.

How to create a password file in **oracle** Linux?

To create a password file in Oracle Linux, you can use the orapwd utility provided by Oracle. Here are the steps to create a password file:

Open a terminal window and log in to the Oracle Linux server as the Oracle user.

Navigate to the $**ORACLE_HOME/bin** directory. This is the directory where the orapwd utility is located.

Run the following command to create a password file named "orapwSID" in the $ORACLE_HOME/dbs directory, where SID is the system identifier of your Oracle database:

orapwd file=$**ORACLE_HOME/dbs/orapwSID password**=<**password**> **entries**=<**number of entries**> **force**=**y**

Replace <password> with the password you want to use for the SYS and SYSTEM accounts, and <number of entries> with the number of entries you want to allow in the password file. The force parameter is optional and can be used to overwrite an existing password file.

After running the command, you will be prompted to confirm the password. Enter the password again and press Enter.

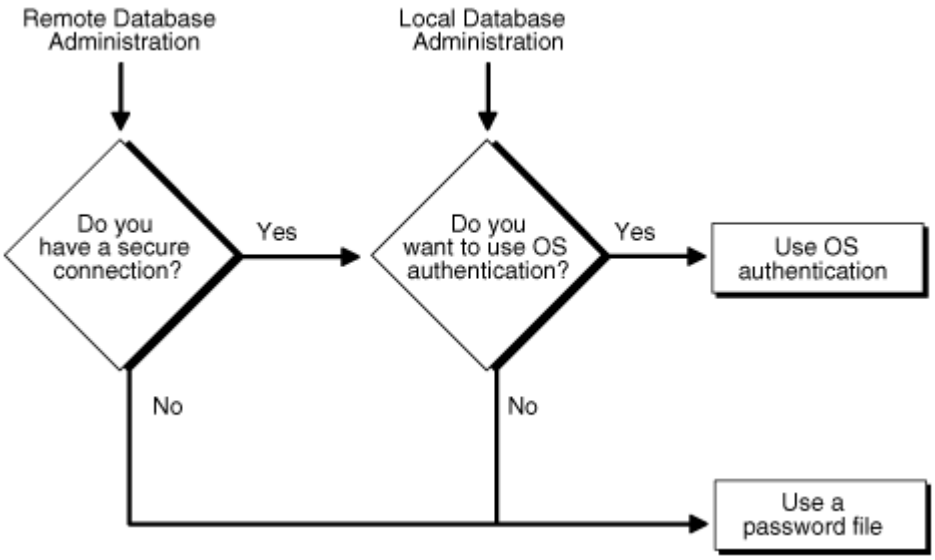
The password file will be created in the $**ORACLE_HOME/dbs** directory.

Finally, set the **REMOTE_LOGIN_PASSWORDFILE** parameter in the database initialization file (initSID.ora) to point to the password file:

REMOTE_LOGIN_PASSWORDFILE = **exclusive**
REMOTE_OS_AUTHENT = **FALSE**

This will ensure that only users listed in the password file are able to connect to the database with administrative privileges.

That's it! You have successfully created a password file in Oracle Linux.



Day 23

Today let us see about temp file in oracle.
#ORACLEARHITECTURE
#ORACLETEMPLFILE
In Oracle, a temporary tablespace is used to store temporary data such as sort results, global temporary tables, and temporary LOBs.

Oracle creates temporary files in the temporary tablespace to store this data.

A temporary file is a file that is created by Oracle to be used as part of the temporary tablespace.

It is used to hold temporary data that is created by Oracle during the execution of SQL statements. Temporary files are created automatically when you create a temporary tablespace, but you can also manually create them if needed.

To create a temporary file in Oracle, you can use the ALTER TABLESPACE command with the ADD TEMPFILE clause. For example, to add a new temporary file to the temp tablespace with a size of 1 GB and a file name of /u01/app/oracle/oradata/temp01.dbf, you can use the following command:
ALTER TABLESPACE temp ADD TEMPFILE '/u01/app/oracle/oradata/temp01.dbf' SIZE 1G;

To shrink a temporary file in Oracle, you can use the ALTER DATABASE command with the DATAFILE clause and the RESIZE clause. For example, to shrink the temp01.dbf file in the temp tablespace to 500 MB, you can use the following command:

ALTER DATABASE DATAFILE '/u01/app/oracle/oradata/temp01.dbf' RESIZE 500M;

To add a new temporary file to an existing temporary tablespace in Oracle, you can use the ALTER TABLESPACE command with the ADD TEMPFILE clause. For example, to add a new temporary file to the temp tablespace with a size of 1 GB and a file name of /u01/app/oracle/oradata/temp02.dbf, you can use the following

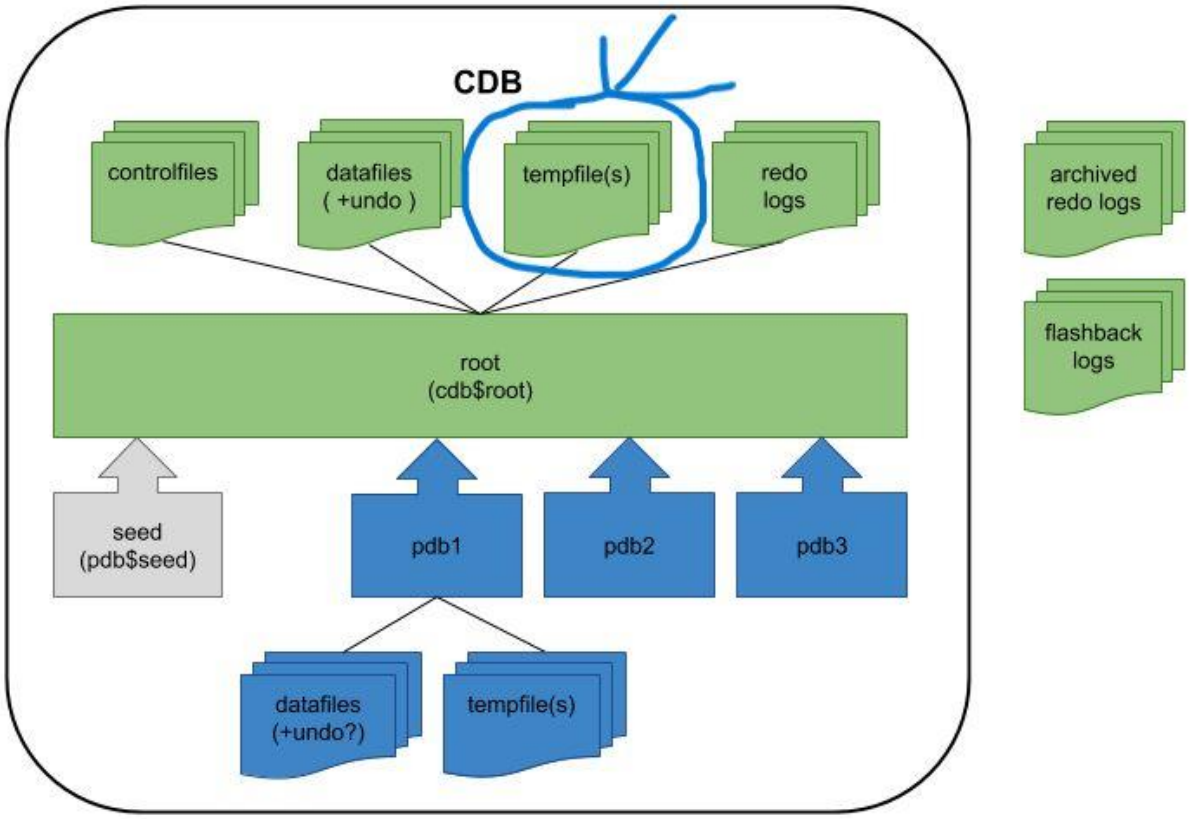
command:
ALTER TABLESPACE temp ADD TEMPFILE '/u01/app/oracle/oradata/temp02.dbf' SIZE 1G;

Note that adding a new temporary file to an existing temporary tablespace can improve performance by allowing Oracle to distribute temporary data across multiple files.

To view the temporary files in an Oracle database, you can query the DBA_TEMP_FILES view. This view contains information about all the temporary files associated with the temporary tablespaces in the database. You need to have the SELECT_CATALOG_ROLE system privilege to query this view.
To query the DBA_TEMP_FILES view, you can use the following SQL statement:

SELECT TABLESPACE_NAME, FILE_NAME, BYTES/1024/1024 SIZE_MB, STATUS FROM DBA_TEMP_FILES;

This SQL statement retrieves the name of the tablespace.
You can also use the USER_TEMP_FILES view or ALL_TEMP_FILES view to retrieve information about the temporary files associated with the current user or all users, respectively. The structure and columns of these views are the same as the DBA_TEMP_FILES view, but they only show information for the current user or all users, respectively.
Note that the DBA_TEMP_FILES view and the other views that provide information about temporary files are only available to users with the appropriate privileges.



Day 24

#ORACLEARCHITECTURE
#UNDODATAFILE

In Oracle, the Undo tablespace is used to manage transaction undo data. When a transaction modifies data in a table, the original data is copied to the Undo tablespace before being overwritten. If the transaction is rolled back or aborted, the undo data is used to restore the original data to the table.

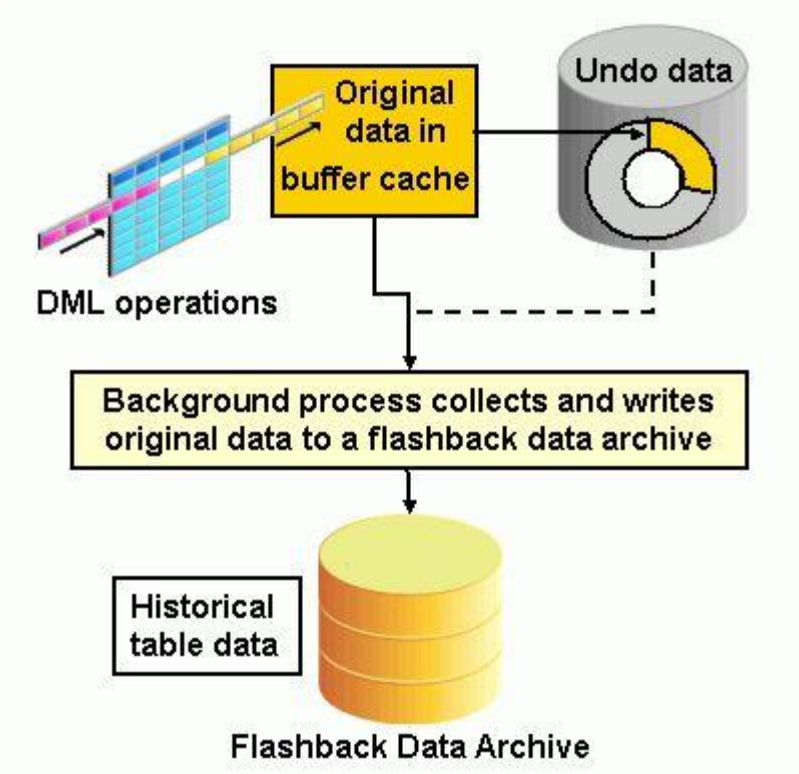
An Undo datafile is a file that contains the undo data for a particular instance of an Oracle database. It stores the information needed to undo changes made to the database during transactions. Each undo datafile can be associated with one or more datafiles, which contain the actual data of the database.

The contents of an undo datafile are typically made up of a series of undo segments. These undo segments are allocated to transactions as needed and are used to store the undo data for those transactions. Each undo segment consists of a number of undo blocks, which store the actual undo data.
For example, to create a new undo tablespace called "UNDOTBS2" with a single undo datafile of 1GB,

CREATE UNDO TABLESPACE UNDOTBS2 DATAFILE '/path/to/undo_datafile2.dbf' SIZE 1G;

To recover an undo data file, you can use the same recovery procedures as for any other data file. If the data file is damaged or lost, you can restore it from a backup and then apply any necessary redo logs to bring it up to date. If the data file cannot be recovered, you can create a new undo tablespace and begin using it instead. It's important to note that undo data is not typically backed up and restored like regular data, as it is not intended to be permanent and is regenerated as needed during normal database operations.

To resize an undo data file, you can use the ALTER DATABASE DATAFILE command. For example, to increase the size of an undo datafile to 500MB, you could use the following command:
ALTER DATABASE DATAFILE '/path/to/undo_datafile.dbf' RESIZE 500M;
To create a new undo data file, you can use the CREATE UNDO TABLESPACE command.

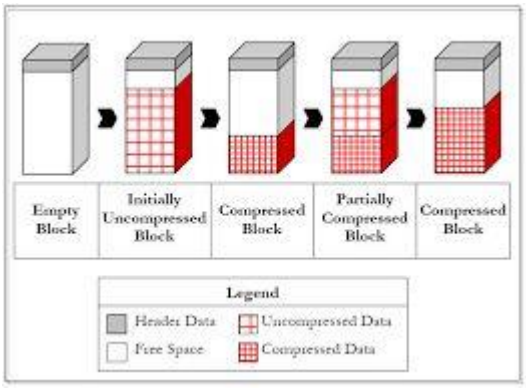


Day 25

Last week saw about oracle's physical structure from now on let see about Oracle's logical architecture.

#ORACLEARCHITECTURE
#ORACLELOGICALSTRUCTURE
#blocks

What is Block in oracle?
In Oracle, a "block" typically refers to a unit of physical storage used by the database to store data. A block is a fixed-size unit of storage that is the smallest unit of data that Oracle reads from or writes to disk.
Blocks are used to store data for tables, indexes, clusters, and other database objects. Each block contains a header that describes the contents of the block, followed by the actual data. The size of a block is typically 2KB or 4KB, although larger block sizes may be used for certain types of data.
When data is read from or written to the database, Oracle retrieves or updates entire blocks at a time. This is why optimizing block size and layout is an important consideration for database performance tuning.
Activate to view larger image,



Day 26

Last week saw about oracle's physical structure from now on let see about Oracle's logical architecture.

#ORACLEARCHITECTURE
#ORACLELOGICALSTRUCTURE
#Extents

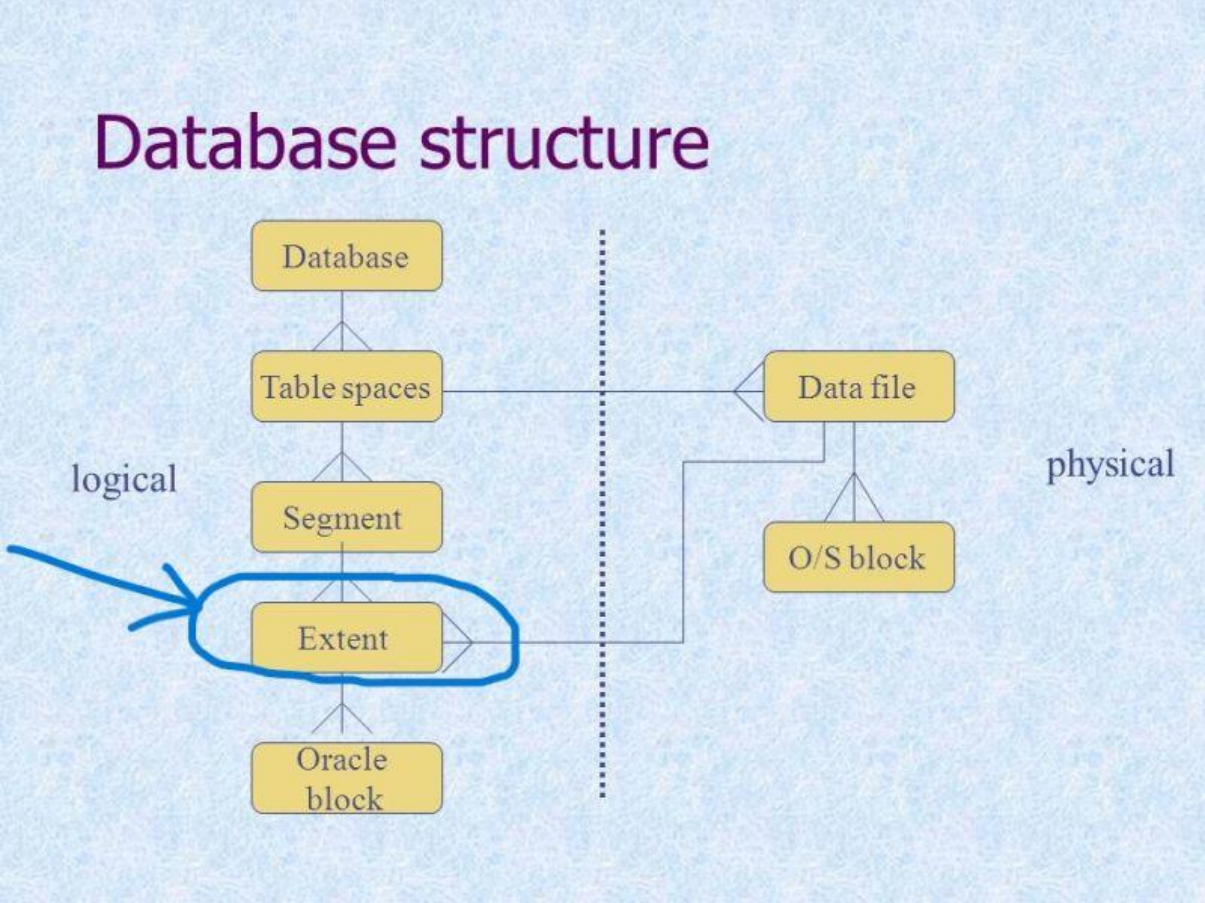
What is extent in Oracle?

In Oracle, "extent" refers to a contiguous set of data blocks allocated to a table or index. When a table or index is created, Oracle allocates an initial set of extents to hold the data, which are referred to as "initial extents."

As data is inserted into the table or index and the initial extents become full, additional extents are allocated to accommodate the new data. These additional extents are referred to as "next extents." The size of each extent is determined by the tablespace in which the table or index is created.

Oracle also allows for "uniform extents," which are extents that are all the same size. This can help to reduce fragmentation and improve performance, as it ensures that data is stored in contiguous blocks.

In summary, an extent is a unit of space allocation in Oracle that is used to hold the data for a table or index. It consists of a contiguous set of data blocks, and additional extents are allocated as needed to accommodate new data.



Day 27

Last week saw about oracle's physical structure from now on let see about Oracle's logical architecture.

#ORACLEARCHITECTURE
#ORACLELOGICALSTRUCTURE
#Segment

In Oracle, a segment is a logical storage structure that represents a specific type of data within a tablespace. There are several types of segments in Oracle, including:

Data segments: These segments contain actual data for tables and indexes.

Index segments: These segments contain index data that is used to speed up access to table data.

Temporary segments: These segments are used to store temporary data that is created during the execution of SQL statements, such as sorting and joining operations.

LOB segments: These segments are used to store large objects, such as text, images, and audio.

Cluster segments: These segments are used to store data for cluster tables, which are tables that share the same storage structure.

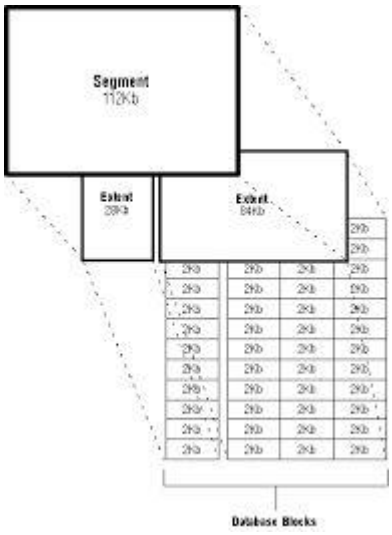
To view the segments in an Oracle database, you can use the DBA_SEGMENTS view, which provides information about the segments in each tablespace. You can also use the USER_SEGMENTS and ALL_SEGMENTS views to view information about the segments that are owned by the current user and all users, respectively.

To calculate the size of a table using segments in Oracle, you can use the following SQL statement:

```
SELECT SUM(bytes)/1024/1024 AS "Table Size (MB)"
FROM user_segments
WHERE segment_name = 'TABLE_NAME';
```

DBA_SEGMENTS describes the storage allocated for all segments in the database.

This statement will return the size of the table in megabytes, based on the size of the data segment associated with the table. Note that this calculation does not include any indexes or other associated segments, so the actual size of the table may be larger.



Day 28

Last week saw about Oracle's physical structure from now on let's see about Oracle's logical architecture.

#ORACLEARCHITECTURE
#ORACLELOGICALSTRUCTURE
#Tablespace

In Oracle, a tablespace is a logical storage unit used to store database objects, such as tables,

indexes, and other schema objects. It's a way to organize and manage the physical storage of data within an Oracle database.

A tablespace is created as a collection of one or more physical files, usually located on a disk. These files are used to store the database objects that belong to the tablespace. A tablespace can be created for a particular user or for a group of users who share a common set of data.

Each tablespace has a name and is identified by a unique number, which is called the tablespace ID. The database administrator can assign a default tablespace to a user so that any new objects that the user creates are automatically stored in that tablespace.

There are several types of tablespaces in Oracle, including system, temporary, undo, and user tablespaces. The system tablespace contains the data dictionary, which stores information about the database schema, while the temporary tablespace is used to store temporary data during sorting and other operations. The undo tablespace stores the information necessary to undo changes made to the database, while the user tablespace is used to store user data.

Tablespaces are an important aspect of Oracle database management, as they provide a way to manage disk space usage, optimize performance, and simplify backup and recovery operations.

Day 29

Last week saw about Oracle's physical structure from now on let's see about Oracle's logical architecture.

#ORACLEARCHITECTURE
#ORACLELOGICALSTRUCTURE
#Tablespace

Below contents is about how to add tablespace and drop tablespace in oracle Database.

To create a tablespace in Oracle, you can use the CREATE TABLESPACE statement.

Here's an example of the syntax:

CREATE TABLESPACE tablespace_name DATAFILE 'file_name' SIZE file_size AUTOEXTEND ON NEXT file_increment_size MAXSIZE maximum_file_size;

Let's break down this syntax:

tablespace_name is the name you want to give to the tablespace.
DATAFILE 'file_name' specifies the name of the physical file that will be used to store the data for the tablespace. You can specify multiple data files separated by commas if you want to spread the data across multiple files.

SIZE file_size specifies the initial size of the data file in bytes or another unit of measure like KB or MB. This value should be greater than or equal to the size of the objects you plan to store in the

tablespace.
AUTOEXTEND ON enables the data file to automatically grow in size as needed. If you don't include this option, you'll need to manually add more space to the data file when it becomes full.

NEXT file_increment_size specifies the amount by which the data file will grow when it needs more space. This value is also specified in bytes or another unit of measure like KB or MB.

MAXSIZE maximum_file_size specifies the maximum size to which the datafile can grow. This value is also specified in bytes or another unit of measure like KB or MB.

Here's an example of creating a tablespace called users with an initial size of 1 GB and auto-extension enabled:

```
CREATE TABLESPACE users
DATAFILE '/u01/app/oracle/oradata/dbname/users01.dbf'
SIZE 1G
AUTOEXTEND ON
NEXT 100M
MAXSIZE UNLIMITED;
```

To drop a tablespace in Oracle, you can use the DROP TABLESPACE statement with the following syntax:

```
DROP TABLESPACE tablespace_name [INCLUDING CONTENTS] [CASCADE CONSTRAINTS];
```

Where tablespace_name is the name of the tablespace that you want to drop.

If you include the INCLUDING CONTENTS clause, Oracle will drop all objects within the tablespace, including tables, indexes, and other objects.

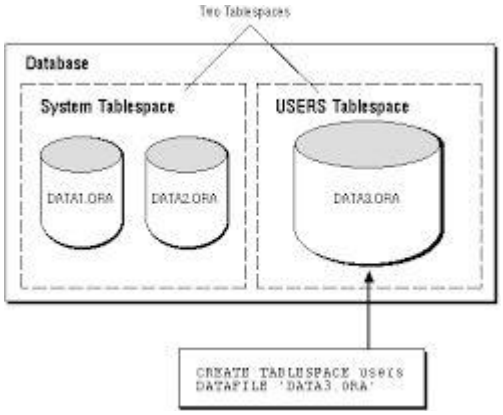
If you include the CASCADE CONSTRAINTS clause, Oracle will also drop any constraints associated with objects in the tablespace.

Note that you must have the DROP TABLESPACE system privilege or the DROP ANY TABLESPACE system privilege to drop a tablespace.

Here's an example of dropping a tablespace named example_tablespace including its contents and associated constraints:

```
DROP TABLESPACE example_tablespace INCLUDING CONTENTS CASCADE CONSTRAINTS;
```

Note that you'll need to have the appropriate privileges to create a tablespace, such as the CREATE TABLESPACE privilege or the DBA role.



Day 30

Let us see about Fragmentation in Oracle.

#ORACLEARCHITECTURE
#ORACLELOGICALSTRUCTURE
#Fragmentation

In Oracle, **fragmentation** refers to the state of data in a table or index where the data blocks are not allocated or utilized in an optimal manner. This means that the data is spread out across multiple blocks in a non-contiguous manner, which can lead to performance issues such as slow query response times.

There are several reasons why **fragmentation** can occur in Oracle:

Insert, update, and delete operations: As data is added, updated, and deleted from a table, the space allocated to it may not be released properly. This can result in the table having unused space, which can lead to fragmentation.

Tablespace management: If a tablespace is not managed properly, it can lead to fragmentation. For example, if a tablespace is not sized properly, it can cause **fragmentation** as data is added to the table.

Extent allocation: If extent allocation is not properly configured, it can result in **fragmentation**. For example, if the next extent size is too small, it can cause the table to be fragmented.

Indexes: Indexes can also become fragmented due to insert, update, and delete operations. This can lead to slow query response times as the database has to search through multiple index blocks.

To address **fragmentation** in Oracle, the following actions can be taken:

Rebuild indexes: Rebuilding indexes can help remove **fragmentation** and improve performance.

Coalesce free space: This operation merges free space in a table to create contiguous blocks, reducing **fragmentation**.

Reorganize tables: This involves moving data from fragmented blocks to contiguous blocks, which can help improve performance.

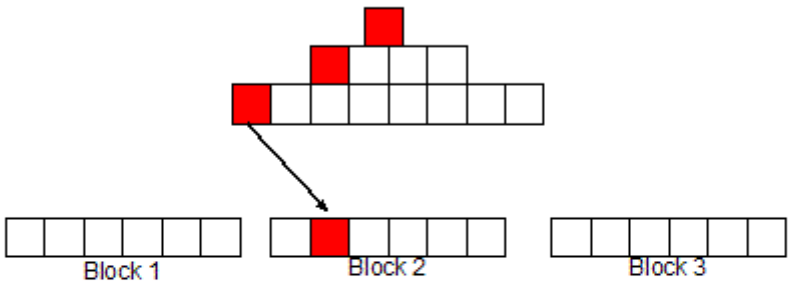
Monitor tablespace usage: Regular monitoring of tablespace usage can help identify potential **fragmentation** issues before they become a problem.

Overall, **fragmentation** in Oracle can occur due to a variety of factors, but it can be addressed through proper database management practices such as monitoring tablespace usage, rebuilding indexes, and coalescing free space.

Note :
Below you can see a query to find a fragmented table in Oracle database.

Sometimes even chain count is also calculated.

```
select
  table_name,round((blocks*8),2) "size (kb)" ,
  round((num_rows*avg_row_len/1024),2) "actual_data (kb)",
  (round((blocks*8),2) - round((num_rows*avg_row_len/1024),2)) "wasted_space (kb)"
from
  dba_tables
where
  (round((blocks*8),2) > round((num_rows*avg_row_len/1024),2))
order by 4 desc;
```



Select order_nbr, item_name from ordor natural join item;

Un-Clustered table rows

Clustering_factor ~= num_rows

Day 31

Today let us see about **Defragmentation** and in what all ways can be defragmented.

In Oracle Database, **defragmentation** refers to the process of reorganizing data to reduce fragmentation and improve database performance. Fragmentation occurs when data is spread out across different storage locations, such as data blocks or disk sectors, resulting in inefficient storage utilization and slower access times.

There are several ways to perform defragmentation in Oracle Database:

Table or Index Rebuild: This involves creating a new copy of a table or index and copying the data into the new structure. This process can eliminate fragmentation and reclaim wasted space.

Online Table Redefinition: Oracle provides the DBMS_REDEFINITION package, which allows you to redefine a table online. This process creates a new table version and gradually migrates the data from the old version to the new version while the table remains accessible to users.

Shrink Segments: Oracle allows you to shrink segments, such as tables, indexes, or partitions, to reclaim unused space. The shrink operation reorganizes the data within the segment and releases the unallocated space back to the tablespace.

Move Tables or Partitions: You can move tables or partitions from one tablespace to another using the ALTER TABLE...MOVE statement. This process can eliminate fragmentation and consolidate data in a more efficient manner.

Coalesce Fragments: The COALESCE command combines neighboring free extents within a segment into larger, contiguous extents. This process helps reduce fragmentation and improve storage utilization.

Reorganize Data Files: If data files have become fragmented at the operating system level, you can use operating system utilities to defragment the physical files. This can help improve disk I/O performance.

It's important to note that the choice of **defragmentation** method depends on the specific scenario, the type of fragmentation, and the available resources. It's recommended to analyze the fragmentation and consult the Oracle documentation or seek advice from a database administrator to determine the most appropriate **defragmentation** approach for your situation.

Day 32
#oracle
#Defragmentationmethods

We saw different methods of defragmentation in Oracle. Among these let us see **Table** alone

today and tomorrow let's see the index rebuild.

how to do **Table** defragmentation oracle with command?

In Oracle, there is no specific command to defragment a table directly. However, you can achieve defragmentation indirectly by performing an "Online Table Redefinition" using the DBMS_REDEFINITION package. This process involves creating an interim table with the desired structure, copying the data from the original table to the interim table, and then replacing the original table with the interim table. This effectively reorganizes the table and eliminates fragmentation. Here's a step-by-step guide:

Create an empty interim table with the desired structure:

```
CREATE TABLE interim_table (
 -- Define the columns as per your requirements
);
```

Start the redefinition process:

```
BEGIN
 DBMS_REDEFINITION.START_REDEF_TABLE(
  uname => 'SCHEMA_NAME',
  orig_table => 'ORIGINAL_TABLE',
  int_table => 'INTERIM_TABLE'
 );
END;
```

Replace 'SCHEMA_NAME' with the name of the schema containing the original table, 'ORIGINAL_TABLE' with the name of the table you want to defragment, and 'INTERIM_TABLE' with the name of the interim table you created.

Copy the data from the original table to the interim table:

```
DECLARE
 num_errors PLS_INTEGER;
BEGIN
 DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS(
  uname => 'SCHEMA_NAME',
  orig_table => 'ORIGINAL_TABLE',
  int_table => 'INTERIM_TABLE',
  copy_indexes => DBMS_REDEFINITION.CONS_ORIG_PARAMS,
  copy_triggers => TRUE,
  copy_constraints => TRUE,
  copy_privileges => TRUE,
  ignore_errors => FALSE,
  num_errors => num_errors
 );
```

```
 DBMS_OUTPUT.PUT_LINE('Number of errors: ' || num_errors);
END;
```

Synchronize the original and interim tables:

```
BEGIN
 DBMS_REDEFINITION.SYNC_INTERIM_TABLE(
  uname => 'SCHEMA_NAME',
  orig_table => 'ORIGINAL_TABLE',
  int_table => 'INTERIM_TABLE'
 );
END;
```

Complete the redefinition process:

```
BEGIN
 DBMS_REDEFINITION.FINISH_REDEF_TABLE(
  uname => 'SCHEMA_NAME',
  orig_table => 'ORIGINAL_TABLE',
  int_table => 'INTERIM_TABLE'
 );
END;
```

Day 33
#oracle
#Defragmentationmethods

Let us see one of the Defragmentation methods in Oracle.

how to **Index Rebuild** in oracle with command?

Creating an index:

CREATE INDEX index_name
ON table_name (column1, column2, ...);
Replace index_name with the desired name for your index and table_name with the name of the table on which you want to create the index. Specify the columns on which you want to create the index within the parentheses.

For example, to create an index named "idx_employees_last_name" on the "employees" table for the "last_name" column, you can use the following command:

```
CREATE INDEX idx_employees_last_name
ON employees (last_name);
```

It's important to ensure that you have the necessary privileges to create tables and indexes in the desired schema. Additionally, consider defining appropriate constraints, such as primary keys or foreign keys, to maintain data integrity within your tables.

To rebuild an index in Oracle using commands, you can use the ALTER INDEX...REBUILD statement. This statement allows you to rebuild an existing index, which can help improve index performance or reclaim fragmented space. Here's how you can rebuild an index:

ALTER INDEX index_name REBUILD;

Replace index_name with the name of the index you want to rebuild. For example, if you have an index named "idx_employees_last_name" on the "employees" table, you can use the following command to rebuild it:

ALTER INDEX idx_employees_last_name REBUILD;

Day 34

#oracle
#Defragmentationmethods

Let us see one of the **Defragmentation** methods in Oracle.

To shrink segments in Oracle, you can use the ALTER TABLE or ALTER INDEX statements with the **SHRINK** SPACE clause. Here are the commands to shrink segments:

Shrink a Table Segment:

ALTER TABLE table_name SHRINK SPACE;

Replace table_name with the name of the table whose segment you want to shrink.

Shrink an Index Segment:

ALTER INDEX index_name SHRINK SPACE;
Replace index_name with the name of the index whose segment you want to shrink.

Note: Shrink operations can be resource-intensive and may impact database performance. It's recommended to perform these operations during a maintenance window or when the system load is low.

You can also specify additional options with the SHRINK SPACE clause to control the behavior of

the shrink operation. Here are a few examples:

Shrink a Table Segment and Compact the High Water Mark:
ALTER TABLE table_name SHRINK SPACE COMPACT;

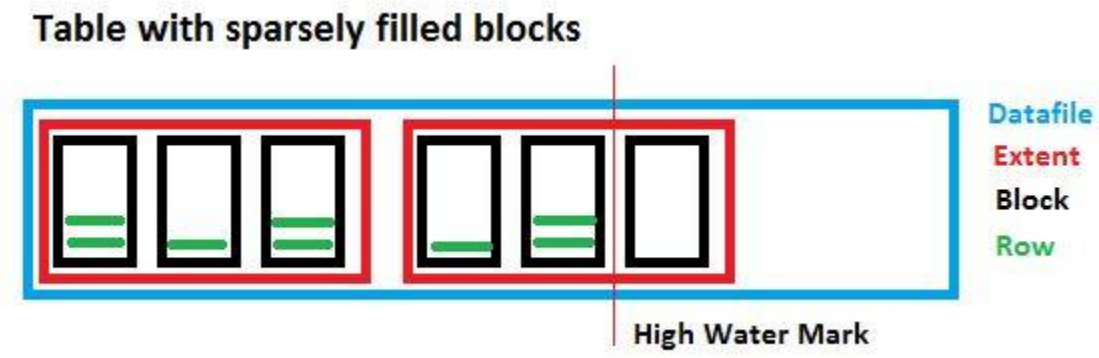
Shrink a Table Segment and Reclaim Unused Space:
ALTER TABLE table_name SHRINK SPACE CASCADE;

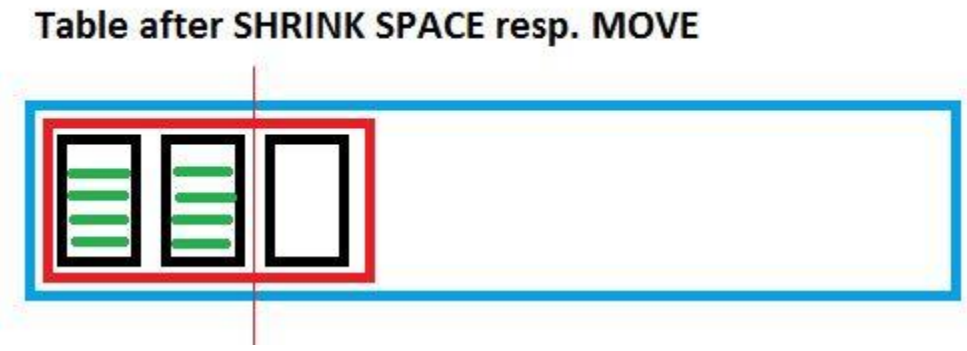
Shrink a Table Segment to a Specific Size:
ALTER TABLE table_name SHRINK SPACE COMPACT SIZE target_size;

Replace target_size with the desired size in bytes, kilobytes, megabytes, or gigabytes.

Shrink an Index Segment and Rebuild It:
ALTER INDEX index_name SHRINK SPACE CASCADE;

Remember to have the necessary privileges to perform these operations, and it's always recommended to test the impact of shrinking segments in a non-production environment before applying it to a production system.





Day 35

#oracle
#Defragmentationmethods

Let us see one of the Defragmentation methods in Oracle.

How to **move tables or partitions**?

To move tables or partitions in an Oracle database, you can use the ALTER TABLE statement with the MOVE clause. Here's how you can move tables or partitions in Oracle:

Connect to the Database: Start by connecting to your Oracle database using a SQL client like SQL*Plus or SQL Developer.

Identify the Table or Partition: Determine the name of the table or partition you want to move.

Decide on the Move Method: There are two methods for moving tables or partitions:

a. **Online Move**: This method allows you to move the table or partition while it remains accessible to users.
It uses an online redefinition process, which requires additional steps and preparation.

b. **Offline Move**: This method involves taking the table or partition offline during the move process.
It is typically faster but requires downtime for the duration of the move.

Move a Table or Partition Online:

a. **Prepare the Destination**: Create an empty table or partition in the desired location with the same structure as the source table or partition.

b. **Execute the Online Move**:

ALTER TABLE table_name MOVE ONLINE TABLESPACE new_tablespace;
or
ALTER TABLE table_name MOVE PARTITION partition_name ONLINE TABLESPACE new_tablespace;

Replace table_name with the name of the table, partition_name with the name of the partition and new_tablespace with the name of the new tablespace where the table or partition should be moved.

Move a Table or Partition Offline:

a. **Prepare the Destination**: Create an empty table or partition in the desired location with the same structure as the source table or partition.

b. **Execute the Offline Move**:

ALTER TABLE table_name MOVE TABLESPACE new_tablespace;
or
ALTER TABLE table_name MOVE PARTITION partition_name TABLESPACE new_tablespace;

Replace table_name with the name of the table, partition_name with the name of the partition and new_tablespace with the name of the new tablespace where the table or partition should be moved.

Verify the Move: After the move operation completes, verify that the table or partition is now in the desired location.
You can use the DBA_TABLES or DBA_TAB_PARTITIONS view to check the table or partition's tablespace.

Remember to consider the impact of moving tables or partitions, such as potential downtime or performance implications.
It is recommended to test the move operation in a non-production environment before performing it in a production database.

Day 36

#oracle
#Defragmentationmethods

Let us see one of the **Defragmentation** methods in Oracle.

How to do **coalesce fragments** in Oracle Database?

There are two ways to **coalesce fragments** in Oracle Database:

Using the ALTER TABLE statement with the **COALESCE** clause.
Using the DBMS_REINDEX package.
Using the ALTER TABLE statement

The following syntax can be used to **coalesce fragments** in a table using the ALTER TABLE statement:

ALTER TABLE table_name COALESCE;
Use code with caution.
For example, the following statement would coalesce fragments in the employees table:

ALTER TABLE employees **COALESCE**;
Use code with caution.
Using the DBMS_REINDEX package

The DBMS_REINDEX package can also be used to coalesce fragments in a table. The following syntax can be used:

```
BEGIN
 DBMS_REINDEX.COALESCE('table_name');
END;
/
```

Use code with caution.
For example, the following statement would coalesce fragments in the employees table:

```
BEGIN
 DBMS_REINDEX.COALESCE('employees');
END;
/
```

Which method should you use?

The ALTER TABLE statement is the simpler of the two methods. However, it can only be used to **coalesce fragments** in a single table. The DBMS_REINDEX package can be used to **coalesce fragments** in multiple tables at once.

If you only need to **coalesce fragments** in a single table, then the ALTER TABLE statement is the best option. If you need to coalesce fragments in multiple tables, then the DBMS_REINDEX package is the best option.

When should you **coalesce fragments**?

You should **coalesce fragments** when you notice that your database is performing poorly. Fragmentation can cause performance problems because it makes it more difficult for the database to find the data it needs.

You can use the DBMS_STATS package to check the fragmentation level of your database. The DBMS_STATS package will return a value called fragmentation_ratio. A fragmentation_ratio of 100% indicates that the table is completely fragmented. A fragmentation_ratio of 0% indicates that the table is not fragmented.

If the fragmentation_ratio of a table is high, then you should coalesce the fragments in the table.

How often should you c**oalesce fragments**?

You should **coalesce fragments** on a regular basis. The frequency with which you need to coalesce fragments depends on the size of your database and the amount of activity on your database.

If you have a small database with low activity, then you may only need to coalesce fragments once a month. If you have a large database with high activity, then you may need to coalesce fragments once a week.

Conclusion

Coalescing fragments can improve the performance of your Oracle Database. You should **coalesce fragments** on a regular basis to ensure that your database is performing at its best.

Day 37

#oracle
#Defragmentationmethods

Let us see one of the **Defragmentation** methods in Oracle.

Reorganize Data Files
==========

How to **Reorganize Data Files** with commands in oracle database?

In Oracle Database, you can organize and manage data files using SQL statements and commands. Here are some common commands to help you organize data files in Oracle:

Renaming Data Files:
To rename a data file in Oracle, you need to perform the following steps:

a. Take the tablespace containing the data file offline:
ALTER TABLESPACE <tablespace_name> OFFLINE;

b. Rename the physical data file using the operating system commands.

c. Update the data file's metadata in the Oracle database:

ALTER DATABASE RENAME FILE '<old_file_path>' TO '<new_file_path>';

d. Bring the tablespace back online:
ALTER TABLESPACE <tablespace_name> ONLINE;

Moving Data Files:
To move a data file to a different location, you can use the following steps:

a. Take the tablespace containing the data file offline:
ALTER TABLESPACE <tablespace_name> OFFLINE;

b. Move the physical data file to the desired location using operating system commands.

c. Update the data file's metadata in the Oracle database:

ALTER DATABASE MOVE DATAFILE '<old_file_path>' TO '<new_file_path>';
d. Bring the tablespace back online:

ALTER TABLESPACE <tablespace_name> ONLINE;

Adding Data Files:
To add a new data file to a tablespace in Oracle, you can use the following command:

ALTER TABLESPACE <tablespace_name> ADD DATAFILE '<file_path>' SIZE <size> [AUTOEXTEND ON NEXT <next_size> MAXSIZE <max_size>];
Replace <tablespace_name> with the name of the tablespace where you want to add the data file, <file_path> with the path of the new data file, <size> with the initial size of the file (e.g., '100M' for 100 megabytes), and optionally specify the autoextend settings.

Dropping Data Files:
To drop a data file from a tablespace, you need to perform the following steps:

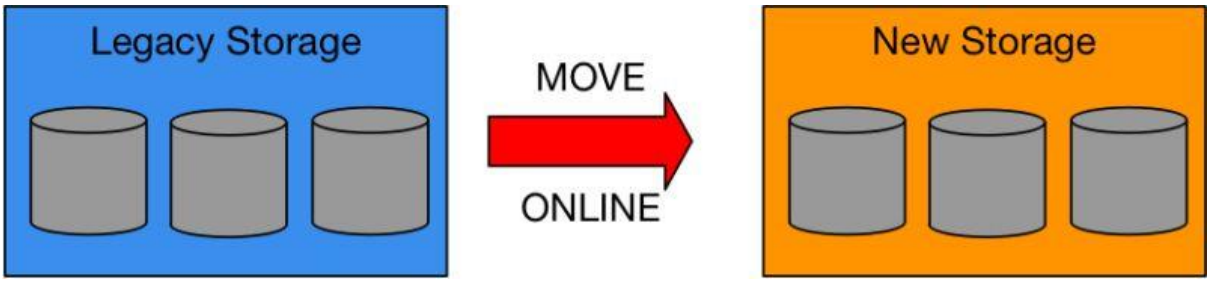
a. Take the tablespace containing the data file offline:

ALTER TABLESPACE <tablespace_name> OFFLINE;

b. Drop the data file from the tablespace:
ALTER TABLESPACE <tablespace_name> DROP DATAFILE '<file_path>';

c. Bring the tablespace back online:
ALTER TABLESPACE <tablespace_name> ONLINE;

Please note that performing any operations on data files in Oracle Database requires appropriate privileges. Make sure you have the necessary permissions or consult your database administrator if needed.



Day 38

Let us see Defragmentation's last method.

#Export import

Let's say this traditional backup

what are all benefits of Oracle export and import and why do we have to use this?

The Export/Import utility (exp and imp) in Oracle provides several benefits and use cases, including:

Data Migration: Export/Import can be used to migrate data from one Oracle 9i database to another. It allows you to export schema(s) or the entire database from a source database and import it into a target database, facilitating data transfer between environments.

Schema-Level Backup and Recovery: Export/Import can be used to create backups of specific schemas in Oracle 9i. By exporting a schema, you can create a binary export file that contains the schema objects and data, which can be later imported to restore the schema in case of data loss or corruption.

Platform Independence: The exported binary file is platform-independent, which means you can export data from one operating system and import it into an Oracle 9i database running on a different operating system. This feature is useful when migrating databases across different platforms.

Data Subset Extraction: Export/Import allows you to extract a subset of data from a database. You can specify criteria such as tables, rows, or columns during the export process to export only the required data, which can be useful for creating subsets of production data for testing or development purposes.

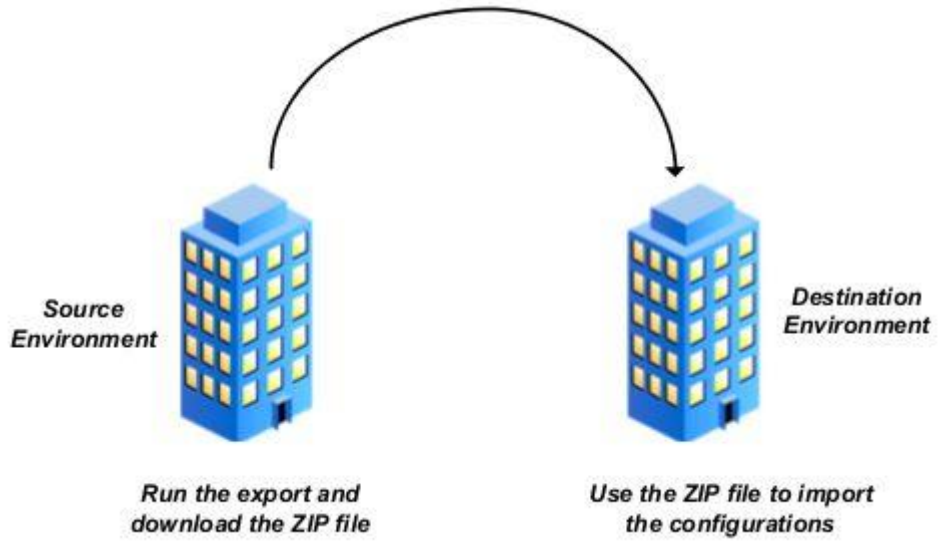
Performance Tuning: Export/Import can be used as a performance tuning tool. By exporting and importing a schema or specific objects, you can reorganize and restructure the data, which can potentially improve performance by eliminating fragmentation or optimizing storage.

Database Cloning: Export/Import is useful for creating clones of databases. By exporting a database or schema and importing it into a new database, you can quickly create a copy of the original database for testing, development, or reporting purposes.

Database Upgrades: Export/Import can be utilized during database upgrades. You can export a schema or the entire database from the older version of Oracle 9i and import it into a newer version of Oracle Database to migrate the data and schema objects during the upgrade process.

While the Export/Import utility provides these benefits, it's important to note that Oracle Data Pump, introduced in Oracle Database 10g, offers enhanced functionality and performance compared to the older Export/Import utility. It is recommended to consider using Data Pump or

other advanced tools for data migration, backup, and recovery operations in newer versions of Oracle Database.



#Day 39

#Export Parameter
This is a utility used to take a backup from the OS level after setting a particular sid (Sid is the database name)
Below is the command to see what are all the parameters
Command
exp help=y

Example: **EXP SCOTT/TIGER**
Or, you can control how Export runs by entering the EXP command followed by various arguments. To specify parameters, you use keywords:

Format: **EXP KEYWORD=value or KEYWORD=(value1,value2,...,valueN)**
Example: **EXP SCOTT/TIGER GRANTS=Y TABLES=(EMP,DEPT,MGR)**
or TABLES=(T1:P1,T1:P2), if T1 is partitioned table

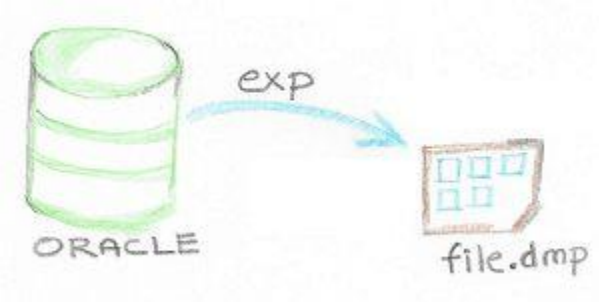
USERID must be the first parameter on the command line.

Keyword Description (Default)

USERID username/password
BUFFER size of data buffer
FILE output files (EXPDAT.DMP)
COMPRESS import into one extent (Y)
GRANTS export grants (Y)

INDEXES export indexes (Y)
DIRECT direct path (N)
LOG log file of screen output
ROWS export data rows (Y)
CONSISTENT cross-table consistency(N)
FULL export entire file (N)
OWNER list of owner usernames
TABLES list of table names
RECORDLENGTH length of IO record
INCTYPE incremental export type
RECORD track incr. export (Y)
TRIGGERS export triggers (Y)
STATISTICS analyze objects (ESTIMATE)
PARFILE parameter filename
CONSTRAINTS export constraints (Y)
OBJECT_CONSISTENT transaction set to read only during object export (N)
FEEDBACK display progress every x rows (0)
FILESIZE maximum size of each dump file
FLASHBACK_SCN SCN used to set session snapshot back to
FLASHBACK_TIME time used to get the SCN closest to the specified time
QUERY select clause used to export a subset of a table
RESUMABLE suspend when a space related error is encountered(N)
RESUMABLE_NAME text string used to identify resumable statement
RESUMABLE_TIMEOUT wait time for RESUMABLE
TTS_FULL_CHECK perform full or partial dependency check for TTS
VOLSIZE number of bytes to write to each tape volume
TABLESPACES list of tablespaces to export
TRANSPORT_TABLESPACE export transportable tablespace metadata (N)
TEMPLATE template name which invokes iAS mode export

Export terminated successfully without warnings.



#Day 40

#SCHEMAREFRESH
#EXPORTIMPORT

What is dumpfile in oracle export utility?
In Oracle, the export dump file generated by the Export utility (EXP) is in a proprietary binary format known as the Oracle Data Pump format. The file typically has the extension ".dmp".

what is logfile in oracle export utility?
In Oracle, the export log file generated during an export operation using the Export utility (EXP) is in a plain text format. The log file records the progress and details of the export process and is typically saved with the extension ".log".

What is the owner in oracle export utility?
In the context of the Oracle Export utility (EXP), the "owner" parameter refers to the specification of the database schema or user whose objects will be exported.

Here are the step-by-step instructions for refreshing a schema:

Export the schema using the EXP utility:
EXP USERNAME/PASSWORD@SID OWNER=SCHEMA_NAME FILE=EXPORT_DUMP.DMP LOG=EXPORT.LOG

Replace the username and password with the credentials for a user with the necessary privileges. SID represents the Oracle system identifier, and schema_name is the name of the schema you want to export. The file parameter specifies the name of the export dump file, and the log parameter indicates the name of the log file for recording the export process.

Drop the schema objects:
DROP USER schema_name CASCADE;
This command drops the schema and all its objects, including tables, indexes, views, etc. Use caution with the CASCADE option, as it deletes all dependent objects.

Recreate the schema:
CREATE USER schema_name IDENTIFIED BY password;
GRANT CONNECT, RESOURCE TO schema_name;

Replace schema_name with the name of the schema you want to recreate, and password with the desired password for the schema user. The GRANT statement provides necessary privileges to the schema.

Import the schema using the IMP utility:
imp username/password@SID fromuser=source_schema touser=schema_name file=export_dump.dmp log=import.log
Here, username and password represent the credentials of a user with the required import privileges.
SID is the Oracle system identifier, source_schema is the name of the exported schema, and schema_name is the name of the newly created schema.
The file parameter specifies the export dump file, and the log parameter indicates the name of the log file for the import process.

Verify the import logs:

Check the import log file (import.log in this example) for any errors or warnings encountered during the import process.
Review the log to ensure that the schema was successfully imported.

#Day 41

#FULLDBREFRESH
#EXPORTIMPORT

How to do full database refresh steps in Oracle with steps and commands using export and import?

Certainly! If you're performing a full database refresh in Oracle using the **Export/Import utilities**, **here are the steps involved**:

Prepare the Target Environment:

Ensure that the target environment meets the hardware and software requirements for Oracle.
Install Oracle software on the target system.
Create an Oracle instance on the target system.

Connect to the source database as a user with administrative privileges (e.g., SYSDBA).
Take a consistent backup of the source database using the Export utility.
Run the following command to export the entire database:

exp userid=<username>/<password> file=<backup_file>.dmp full=y

Here I didn't mention a log because it is not mandatory but in case the export fails during export backup we need a log for sure.

This command exports the entire database, including all schemas and data, into a binary dump file.
Transfer the Backup to the Target Environment:

Copy the backup file created in the previous step to the target environment. You can use tools like FTP or SCP to transfer the file.

Prepare the Target Database:

Start the Oracle instance on the target system.
Create a new database using the Oracle Database Configuration Assistant (DBCA) or manually create the necessary tablespaces and schema objects.
Configure the initialization parameters to match the source database (e.g., memory settings, file locations).

Import the Backup into the Target Database:

Connect to the target database as a user with administrative privileges (e.g., SYSDBA).

Run the following command to import the data into the target database:

imp userid=<username>/<password> file=<backup_file>.dmp full=y
This command imports the entire database from the backup file into the target database.

Perform Post-Restore Tasks:

Run any required scripts to recompile invalid objects in the target database.
Perform any necessary post-restore testing to ensure the integrity and functionality of the target database.
Please note that the above steps provide a high-level overview of the process. You may need to customize the commands and parameters based on your specific requirements and environment. It's important to refer to the Oracle documentation and consult the appropriate resources for detailed instructions on using the Export/Import utilities in Oracle.

#Day 42

Following up #Day 39 post

here below what are all parameters is in import backup.

This is a utility used to restore a backup from the OS level after setting a particular sid (Sid is the database name)
Below is the command to see what are all the parameters

imp help=y in Oracle might look like based on general knowledge of Oracle database utilities. Here's an example:

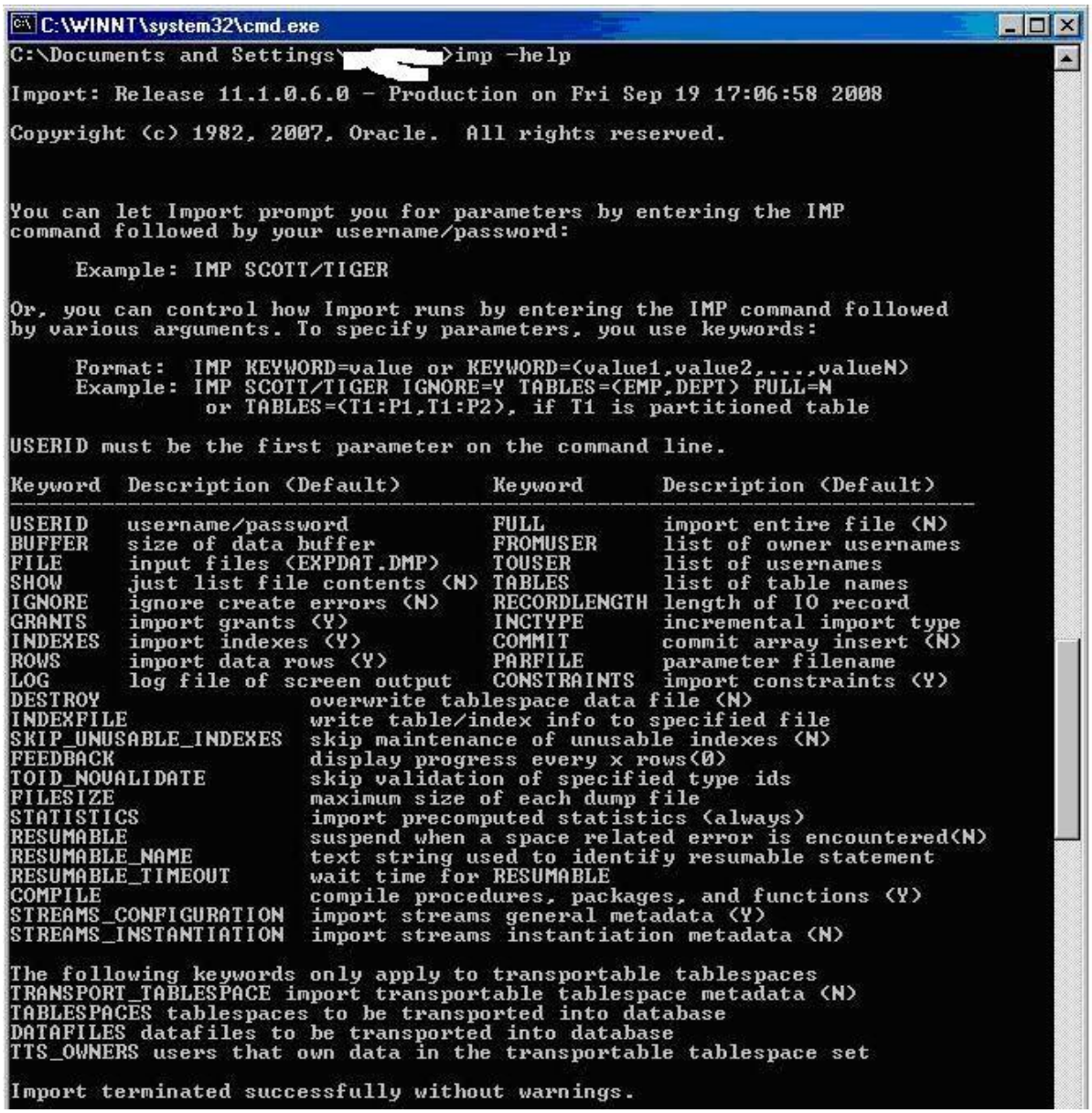
Import: Release- Production on Wed Jun 2 2023 10:00:00

You can specify the following parameters while running the IMP command:

USERID username/password
FROMUSER list of usernames separated by commas
TOUSER username(s) to import into
FILE input files (EXPDAT.DMP)
SHOW just list file contents (YES)
IGNORE ignore create errors (YES)
INDEXES import indexes (YES)
GRANTS import grants (YES)
ROWS import data rows (YES)
FULL import entire file (NO)

The actual output may vary slightly depending on the version and configuration of Oracle that

you are using. It's always recommended to consult the official Oracle documentation or resources specific to your Oracle for the most accurate and up-to-date information regarding the imp command and its parameters.



#Day 43
#oracle
#DATAPUMP

What is DATAPUMP in Oracle and its architecture?

Whenever using this utility by default it creates a Master table till completing the export dump and import dump.

Last week saw about traditional backup export and let us see about Datapump
In Oracle, Data Pump is a feature that provides high-speed data and metadata movement between databases.
It is a powerful utility for performing efficient data and metadata operations, such as export, import, and maintenance tasks.

Data Pump was introduced in Oracle Database 10g as a replacement for the older Export and Import utilities.
It offers several advantages over its predecessor, including better performance, enhanced functionality, and improved manageability.

Data Pump consists of two components:

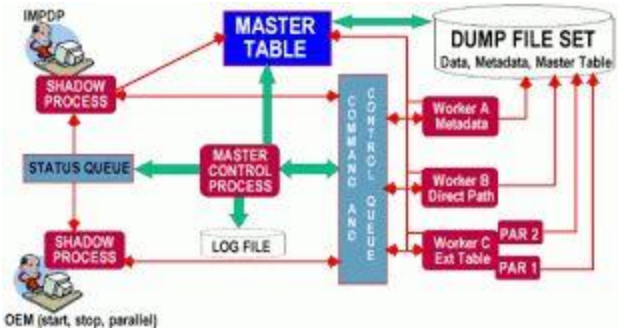
Data Pump Export: This component allows you to extract data and metadata from an Oracle database into a set of operating system files.
You can export a complete database, specific schemas, or individual database objects like tables, views, or procedures.
The exported data can be in a binary format for efficient transport or in a text format for easy readability.

Data Pump Import: This component enables you to import data and metadata from the Data Pump Export files back into an Oracle database.
It allows you to restore a database, load data into an existing database, or selectively import specific objects or schemas.
Data Pump Import also provides various options for controlling how the import process handles conflicts, transformations, and data filtering.

Data Pump offers significant advantages over the traditional Export and Import utilities.
It provides better performance through parallel processing, which allows data to be moved quickly between databases.
It also supports advanced features like compressing data during the export/import process, filtering data based on complex conditions, and transforming data as it is moved between databases.

Furthermore, Data Pump provides comprehensive logging and error handling capabilities, making it easier to monitor and troubleshoot the data movement operations.
It is widely used for various tasks, such as database backups, data migration, database cloning, and data synchronization between different database instances.

Overall, Data Pump is a vital tool in Oracle for efficiently managing and transferring data and metadata between databases, offering improved performance, flexibility, and control over the export and import operations.



#Day 44
#oracle
#DATAPUMP

Oracle Data Pump was first released in Oracle 10g (expdp and impdp).
Till Oracle 9i version export and import were used like exp and imp.

But below given parameters are from Oracle 19c version

SQL> !expdp help=y

Example: expdp scott/tiger DIRECTORY=dmpdir DUMPFILE=scott.dmp
Format: expdp KEYWORD=value or KEYWORD=(value1,value2,...,valueN)
Example: expdp scott/tiger DUMPFILE=scott.dmp DIRECTORY=dmpdir SCHEMAS=scott or TABLES=(T1:P1,T1:P2), if T1 is a partitioned table

ACCESS_METHOD
Instructs Export to use a specific method to unload data.
Valid keyword values are: [AUTOMATIC], DIRECT_PATH, and EXTERNAL_TABLE.

ATTACH
Attaches to an existing job.
For example, ATTACH=job_name.

CLUSTER
Enables the utilization of cluster resources and distribution of workers across Oracle RAC [YES].

COMPRESSION
Reduces the size of the dump file.
Valid keyword values are ALL, DATA_ONLY, [METADATA_ONLY], and NONE.

COMPRESSION_ALGORITHM
Specifies the compression algorithm to be used.
Valid keyword values are [BASIC], LOW, MEDIUM, and HIGH.

CONTENT
Specifies the data to be unloaded.
Valid keyword values are [ALL], DATA_ONLY, and METADATA_ONLY.

DATA_OPTIONS
Data layer option flags.
Valid keyword values are GROUP_PARTITION_TABLE_DATA, VERIFY_STREAM_FORMAT, and XML_CLOBS.

DIRECTORY
Specifies the directory object to be used for dump and log files.

DUMPFILE
Specifies a list of destination dump file names [expdat. dmp].
For example, DUMPFILE=scott1.dmp, scott2.dmp, dmpdir:scott3.dmp.

ENCRYPTION
Encrypts part or all of the dump file.
Valid keyword values are ALL, DATA_ONLY, ENCRYPTED_COLUMNS_ONLY, METADATA_ONLY, and NONE.

ENCRYPTION_ALGORITHM
Specifies the encryption algorithm to be used.
Valid keyword values are [AES128], AES192, and AES256.

ENCRYPTION_MODE
Specifies the method of generating the encryption key.
Valid keyword values are DUAL, PASSWORD, and [TRANSPARENT].

ENCRYPTION_PASSWORD
Specifies the password key for creating encrypted data within a dump file.

ENCRYPTION_PWD_PROMPT
Specifies whether to prompt for the encryption password [NO].
Terminal echo will be suppressed while reading standard input.

ESTIMATE
Calculates job estimates.
Valid keyword values are: [BLOCKS] and STATISTICS.

ESTIMATE_ONLY
Calculates job estimates without performing the export [NO].

EXCLUDE
Excludes specific object types.
For example, EXCLUDE=SCHEMA:"='HR'".

FILESIZE
Specifies the size of each dump file in bytes

#Day 45
#ORACLE
#Datapump

How to take a schema-level backup and restore using Data Pump in Oracle, you can follow these steps:

Exporting (**Taking the Backup**):
Open a command prompt or terminal and navigate to the directory where Oracle Data Pump Export (expdp) is located.
Connect to the Oracle database using the appropriate credentials:

expdp username/password@sid
Specify the export parameters, including SCHEMAS: The name of the schema you want to back up.

DIRECTORY: The Oracle directory object pointing to the location where the export dump file will be created.

DUMPFILE: The name of the export dump file.

LOGFILE: The name of the log file to record the export process.
Execute the export command:

expdp username/password@sid SCHEMAS=schema_name DIRECTORY=directory_object DUMPFILE=dumpfile_name.dmp LOGFILE=logfile_name.log

Wait for the export process to complete. The dump file and log file will be created in the specified directory.

After taking export from the source to move taken export backup from the source server to the target server follow the below steps before importing into the target server from the Linux os level

Example
========
scp export.dmp(**filename of export dump** in source linux server) username2(**target server oracle username**)@ipaddress(**target IP address**):directory2/filename2(**target folder or directory**)

after executing the above scp command asks you target server password. By entering the target password starts transferring the export backup to the target or destination location given above.

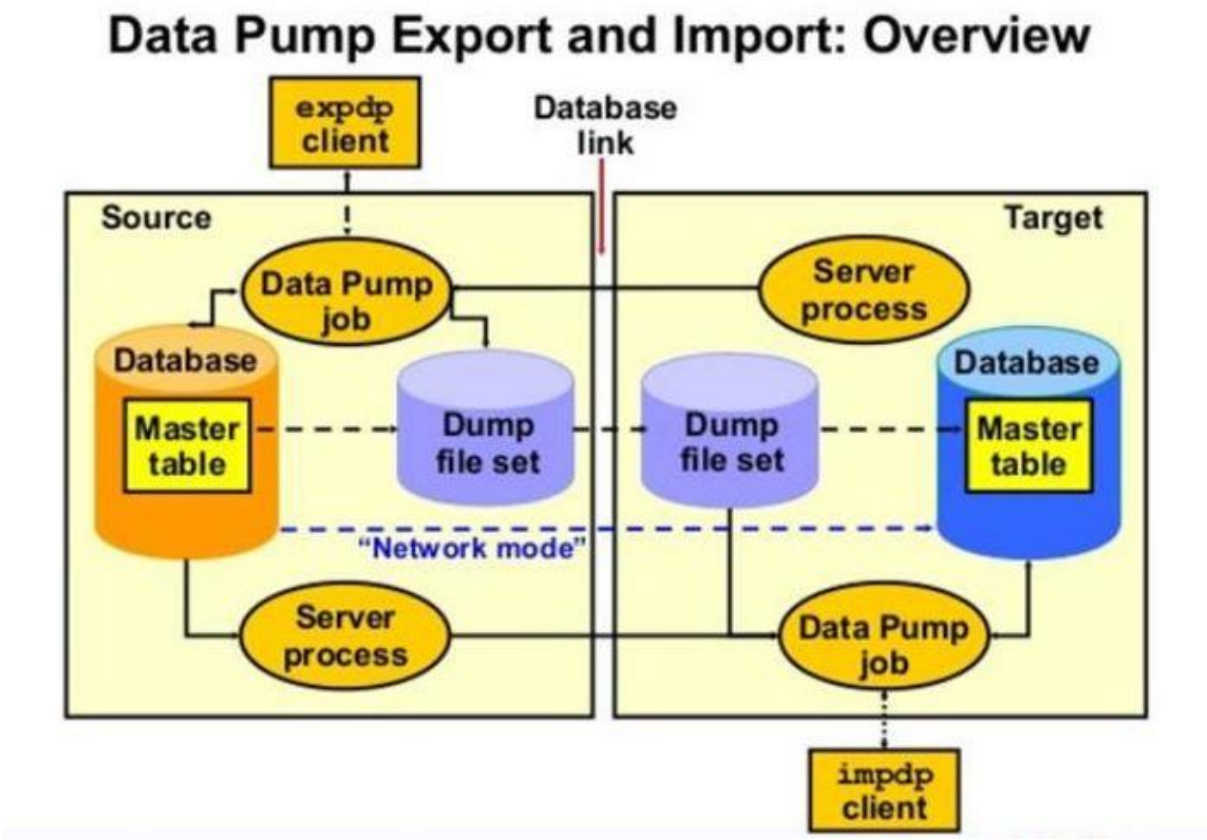
in reality it will be like this as below which I done from my personal device from local server
==========
[oracle@ol7 ~]$ scp test.ora oracle@192.168.106.143:/home/oracle/test.ora
oracle@192.168.106.143's password:
test.ora

Importing (Restoring the Backup):
Open a command prompt or terminal and navigate to the directory where Oracle Data Pump Import (impdp) is located.

impdp username/password@sid
Specify the import parameters, including **REMAP_SCHEMA**: The mapping of the source schema name to the target schema name (optional).

impdp username/password@sid REMAP_SCHEMA=source_schema:target_schema DIRECTORY=directory_object DUMPFILE=dumpfile_name.dmp LOGFILE=logfile_name.log

Wait for the import process to complete. The schema will be restored to the target database.



#Day 46
#ORACLE
#Datapumpfull=y

Day 45 we saw schema export and import using datapump utility.

Due to space limitations on LinkedIn, I am unable to share the complete information about how

to take a **full database backup using Expdp**. However, I have shared a partial dump of today's **full database export**. We can continue with the import process and discuss the remaining steps on a later occasion.

To take a **full database export** and import it into Oracle 12c, you need to follow several steps. Below is a detailed guide that includes prerequisites and post-requirements for migrating the database from a source server to a target server.

Prerequisites:

Ensure that you have the necessary privileges and permissions on both the source and target servers to perform the export and import operations.
Verify that sufficient disk space is available on the target server to accommodate the database export file.

Install Oracle Database 12c on the target server and configure the necessary network settings to establish connectivity between the source and target databases.

Export Steps:

Connect to the source database using SQL*Plus or any other Oracle client tool.

Run the following command to initiate the export:

expdp username/password@source_service_name FULL=Y DIRECTORY=directory_name DUMPFILE=dumpfile_name.dmp LOGFILE=log_file_name.log

Replace username/password with the credentials of a user with sufficient privileges to perform the export.
Replace source_service_name with the service name or SID of the source database.

Replace directory_name with the name of an Oracle directory object that points to a location where the export file will be created.
Replace dumpfile_name.dmp with the desired name for the export file.

Replace log_file_name.log with the desired name for the log file.
Wait for the export process to complete. This may take a significant amount of time, depending on the database size.

#Day 47
#ORACLE
#Datapumpimport full=y

Following up #Day46 post

Import Steps:

Copy the export file (dumpfile_name.dmp) from the source server to a location accessible by the target server.

Connect to the target database using SQL*Plus or another Oracle client tool.

Create a directory object in the target database that points to the location where the export file is located. Run the following command:

CREATE DIRECTORY directory_name AS '/path/to/directory';
Replace directory_name with the desired name for the directory object.
Replace /path/to/directory with the actual path to the directory where the export file is located.
Run the following command to initiate the import:

impdp username/password@target_service_name FULL=Y DIRECTORY=directory_name DUMPFILE=dumpfile_name.dmp LOGFILE=log_file_name.log

Replace username/password with the credentials of a user with sufficient privileges to perform the import.
Replace target_service_name with the service name or SID of the target database.

Replace directory_name with the name of the directory object created in step 3.

Replace dumpfile_name.dmp with the name of the export file.
Replace log_file_name.log with the desired name for the log file.
Wait for the import process to complete. This can take a considerable amount of time, depending on the size of the database.

Post-Requirements:

Verify the success of the import by checking the import log file (log_file_name.log) for any errors or warnings.

Perform thorough testing of the imported database to ensure that all data and functionality have been successfully migrated.

Update any necessary configurations or settings on the target database to match the source database, such as network configurations, user permissions, and application-specific configurations.

Note: It's recommended to consult the Oracle documentation or seek assistance from an Oracle database administrator (DBA) for specific details related to your environment and requirements.

#Day 48
#ORACLE
#Datapumpimportdumphelp=y

Here Let us see what are all import datapump parameters
As LinkedIn have some restrictions to avoid excess content I have mentioned half of import dump parameters however let us see the rest of the remaining parameters of import dump the next day

Below parameters taken from 19c version oracle database.

SQL> !impdp help=y

Import: Release 19.0.0.0.0 - Production on Tue Jun 13 11:57:27 2023
Version 19.14.0.0.0

The Data Pump Import utility provides a mechanism for transferring data objects
between Oracle databases. The utility is invoked with the following command:

Example: impdp scott/tiger DIRECTORY=dmpdir DUMPFILE=scott.dmp

You can control how Import runs by entering the 'impdp' command followed
by various parameters. To specify parameters, you use keywords:

Format: impdp KEYWORD=value or KEYWORD=(value1,value2,...,valueN)
Example: impdp scott/tiger DIRECTORY=dmpdir DUMPFILE=scott.dmp

USERID must be the first parameter on the command line.

The available keywords and their descriptions follow. Default values are listed within square brackets.

ABORT_STEP
Stop the job after it is initialized or at the indicated object.
Valid values are -1 or N where N is zero or greater.
N corresponds to the object's process order number in the master table.

ACCESS_METHOD
Instructs Import to use a particular method to load data.
Valid keyword values are: [AUTOMATIC], CONVENTIONAL, DIRECT_PATH,
EXTERNAL_TABLE, and INSERT_AS_SELECT.

ATTACH
Attach to an existing job.
For example, ATTACH=job_name.

CLUSTER
Utilize cluster resources and distribute workers across the Oracle RAC [YES].

CONTENT
Specifies data to load.

Valid keywords are: [ALL], DATA_ONLY and METADATA_ONLY.

DATA_OPTIONS
Data layer option flags.
Valid keywords are: DISABLE_APPEND_HINT, ENABLE_NETWORK_COMPRESSION, REJECT_ROWS_WITH_REPL_CHAR, SKIP_CONSTRAINT_ERRORS, CONTINUE_LOAD_ON_FORMAT_ERROR, TRUST_EXISTING_TABLE_PARTITIONS and VALIDATE_TABLE_DATA.

DIRECTORY
Directory object to be used for dump, log and SQL files.

DUMPFILE
List of dump files to import from [expdat.dmp].
For example, DUMPFILE=scott1.dmp, scott2.dmp, dmpdir:scott3.dmp.

ENCRYPTION_PASSWORD
Password key for accessing encrypted data within a dump file.
Not valid for network import jobs.

ENCRYPTION_PWD_PROMPT
Specifies whether to prompt for the encryption password [NO].
Terminal echo is suppressed while standard input is read.

ESTIMATE
Calculate network job estimates.
Valid keywords are: [BLOCKS] and STATISTICS.

EXCLUDE
Exclude specific object types.
For example, EXCLUDE=SCHEMA:"='HR'".

FLASHBACK_SCN
SCN used to reset session snapshot.

FLASHBACK_TIME
Time used to find the closest corresponding SCN value.

FULL
Import everything from source [YES].

#Day 49
#ORACLE
#Datapumpperformanceparameter

Datapump performance parameter

Oracle Data Pump is a powerful tool used for data movement and data migration in Oracle

databases. It provides high-speed bulk data and metadata movement between databases, both within a single Oracle database instance and across different instances.

When using Oracle Data Pump, you can configure and optimize various performance parameters to improve the efficiency of data pump operations. Here are some important performance parameters used in
Oracle Data Pump:

PARALLEL: This parameter controls the degree of parallelism for Data Pump export or import operations. It specifies the number of worker processes that can execute in parallel. Increasing the parallelism can speed up the overall operation, especially when dealing with large datasets.

DUMPFILE: This parameter specifies the name and location of the dump file(s) used by Data Pump. You can specify multiple dump files to take advantage of parallelism and distribute the workload across multiple files and disks.

LOGFILE:
Optional but required in case of any export or import failure and information of backup
LOGFILE: This parameter specifies the name and location of the log file generated by Data Pump. The log file contains detailed information about the export or import operation, including any errors or warnings encountered.

CONTENT: This parameter determines the type of data to be exported or imported. It can be set to "ALL" to include all objects, "DATA_ONLY" to exclude metadata and export only data, or "METADATA_ONLY" to exclude data and export only metadata.

BUFFER: This parameter controls the size of the buffer used by Data Pump for data movement. A larger buffer size can improve performance by reducing I/O operations, but it may consume more memory.

ESTIMATE: This parameter allows you to estimate the resource requirements for an export or import operation without actually performing the operation. It provides useful information for planning purposes.

COMPRESSION: This parameter enables compression for Data Pump export or import operations. Compression reduces the size of the dump file, resulting in faster data transfer and reduced storage requirements.

TRANSFORM: This parameter enables you to transform data during the export or import operation. For example, you can specify transformations to transform data between character sets, reorganize tables, or filter data based on specific criteria.

These are some of the performance parameters used in Oracle Data Pump. By carefully configuring these parameters based on your specific requirements and environment, you can optimize the performance of Data Pump operations.

#Day50

#ORACLE

#DIFFERENCEBETWEENEXPIMPEXPDPIMPDP

DIFFERENCE BETWEEN EXPORT IMPORT AND DATAPUMP IN ORACLE

In Oracle databases, Data Pump and the traditional Export/Import utilities are two different mechanisms used for moving data and metadata between databases. Here are the main differences between them:

Functionality: Data Pump is a more advanced and feature-rich utility compared to the traditional Export/Import. It provides enhanced capabilities for high-speed data and metadata movement, while also allowing for more granular control over the export and import operations.

Data Pump Formats: Data Pump supports two file formats: the binary dump file (.dmp) and the new Oracle Database Cloud file (.expdp). The binary dump file is the default format and contains both data and metadata. The Cloud file format is an XML representation of the metadata only. In contrast, the Export/Import utilities use a binary dump file format.

Performance: Data Pump is generally faster than the traditional Export/Import utilities. It achieves better performance through parallelism, where multiple worker processes can perform data extraction, transformation, and loading concurrently. Data Pump also provides advanced compression options to optimize the data transfer process.

Network Mode: Data Pump operates in network mode, meaning it can perform export and import operations directly over a network link between the source and target databases. This feature enables efficient data transfer and eliminates the need for intermediate storage. The Export/Import utilities, on the other hand, require a file-based approach, where the dump files are created on disk and then transferred manually.

Metadata Filtering: Data Pump allows for selective export and import operations by providing various filtering options. You can specify which tables, schemas, or objects to include or exclude from the export/import operation based on different criteria. The traditional Export/Import utilities also offer similar filtering capabilities but are more limited compared to Data Pump.

Compatibility: Data Pump is available from Oracle Database 10g onwards and is the recommended method for data and metadata movement in newer versions. The traditional Export/Import utilities are still supported for backward compatibility and are suitable for simpler scenarios or when working with older versions of Oracle.

Overall, Data Pump offers better performance, more advanced features, and greater flexibility for data and metadata movement in Oracle databases. However, the traditional Export/Import utilities still serve a purpose, particularly for simpler or backward compatibility scenarios.

#Day 51
#ORACLE
#RMAN
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP

RECOVERYMANAGER
==========

RMAN in ORACLE
Previously we saw about traditional backup export-import & Server Based Utility Datapump this week let's see about server-managed backup utility in Oracle.

what is rman in oracle?

RMAN, which stands for Recovery Manager, is a utility Oracle provides for backup and recovery operations in Oracle Database. It is specifically designed to manage Oracle databases' backup, restoration, and recovery.

RMAN operates at the Oracle database block level, allowing it to efficiently manage backup and recovery operations. It can create full or incremental backups of data files, control files, and archived redo logs. RMAN also provides features such as backup compression, encryption, and parallelism to improve performance and optimize storage utilization.

The main advantages of using RMAN for backup and recovery in Oracle are:

Integration with the Oracle database: **RMAN** is tightly integrated with Oracle Database, making it the recommended tool for managing backups and recoveries. It leverages the database's underlying architecture to perform efficient and consistent backups.

Simplified administration: **RMAN** simplifies the backup and recovery process by automating many tasks. It provides a command-line interface and a graphical user interface (Oracle Enterprise Manager) for managing backup and recovery operations.

Incremental backups: **RMAN** allows you to perform incremental backups, which only capture the changes made since the last backup. This reduces the backup size and duration, making it more efficient.

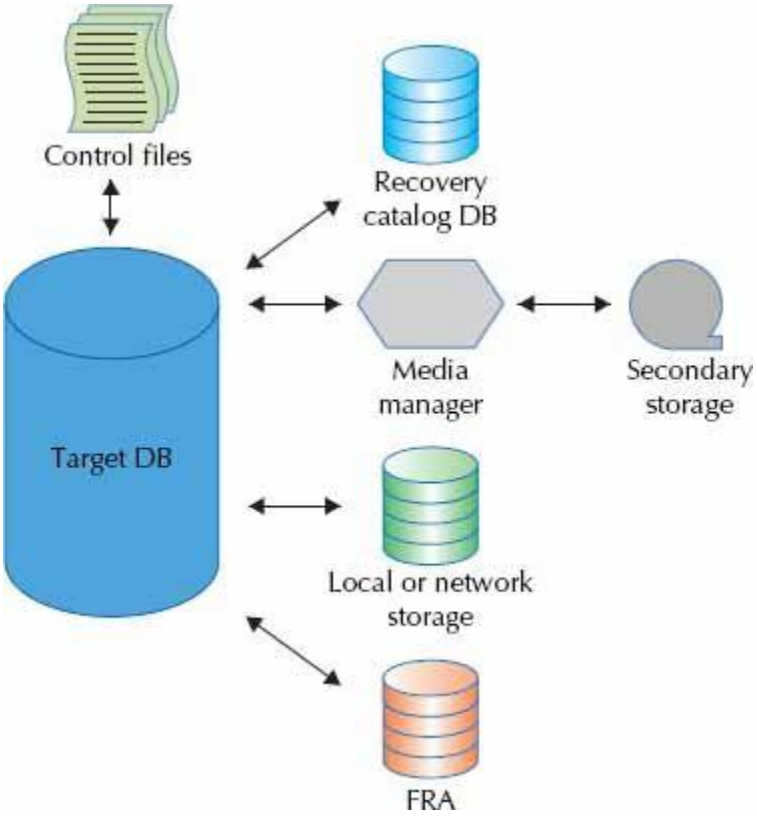
Block-level recovery: **RMAN** can recover individual data blocks rather than requiring the entire data file to be restored. This enables faster recovery and minimizes downtime.

Media management integration: **RMAN** integrates with various media management software and tape libraries, enabling you to store backups on different devices and manage them efficiently.

Disaster recovery capabilities: **RMAN** provides features for disaster recovery scenarios, such as restoring backups to a different server or recovering a database to a specific point in time.

Overall, **RMAN** is a powerful and flexible tool for managing backup and recovery operations in

Oracle Database, offering data protection and minimizing downtime in case of failures or disasters.



#Day 52
#ORACLE
#RMAN
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP

Today let us see when and which version of Oracle Rman was introduced and what types of backup can be taken in Rman.

From which Oracle version was Rman introduced in Oracle?

RMAN (Recovery Manager) was introduced in Oracle Database **version** 8.0. RMAN is a powerful tool Oracle provides for managing backup and recovery operations. It allows users to perform efficient backup and recovery tasks, including backup and restoration of data files, control files, and archived redo logs. RMAN simplifies backup and recovery processes and provides features such as incremental backups, block-level corruption detection, and parallelized operations.

What types of backup we can take in Oracle Rman?

In Oracle RMAN (Recovery Manager), there are several types of backups you can take. Here are the commonly used backup types in RMAN:

Full Backup:

A full backup captures the entire database, including all data files, control files, and archived redo logs.
It provides a complete copy of the database at the time of the backup.
Full backups serve as a baseline for other types of backups and can be used for restoring the entire database.

Incremental Backup:

Incremental backups capture only the changes made since the last backup, whether it's a full backup or a previous incremental backup.
Incremental backups can be taken at different levels, such as level 0, level 1, or higher levels, depending on the granularity required.
They provide an efficient way to back up and restore databases by reducing backup size and improving recovery time.

Archive Log Backup:

Archive log backups precisely capture the archived redo logs generated by the database.
These backups are essential for point-in-time recovery and can be used in conjunction with full or incremental backups to restore the database to a specific time.
Archive log backups ensure the availability of redo data necessary for recovery.

Control File Backup:

Control file backups capture the control files, which contain metadata about the physical structure of the database.
These backups are crucial for database recovery and can be used to restore the control file in case of a failure.
It is recommended to regularly back up control files to ensure recoverability.

Image Copy Backup:

Image copy backups create an exact copy of data files, control files, or archived redo logs at the physical level.
Image copies provide an alternative to traditional RMAN backups and can be used for faster restore and recovery operations.
They are typically used for specific recovery scenarios or as an additional backup option.

#Day 53
#ORACLE
#RMAN
#RECOVERYMANAGER

RMAN
=====
Today let us see how to connect rman prompt in Oracle database from Linux.

Before connecting to rman prompt what are all the prerequisites.

1.**Check whether Sid has set it or not**.?

Ans: To check please execute the below command in Linux operating system where the Oracle database is running at.

Command
=======
[oracle@ol7 ~]$ ps -ef|grep pmon
oracle 4470   1 0 10:36 ?   00:00:00 ora_pmon_test
oracle 5441 5387 0 10:44 pts/0 00:00:00 grep --color=auto pmon

2.**Can Rman backup be taken when the database is not running**?

Ans:No, you cannot take an RMAN (Recovery Manager) backup when the database is not running in Oracle. RMAN is an Oracle utility that operates within the context of a running database instance.

To perform an RMAN backup, the Oracle database must be in a mounted or open state. RMAN interacts with the Oracle database instance to perform backup and recovery operations, leveraging the database's underlying infrastructure.

If the database is not running, you won't be able to use RMAN to initiate a backup. It is necessary to start the database instance and have it in a mounted or open state before executing RMAN commands to back up the database.

However, there are alternative methods to perform backups when the database is not running, such as taking an offline backup using operating system-level tools or employing database export utilities like Data Pump. These methods have different mechanisms and considerations compared to RMAN backups.

3.**How to check whether the Database running or not**?
Ans:To check whether a database is running or not in Oracle, you can use the SQL*Plus tool or any other Oracle database client.

Open a terminal or command prompt.

Start SQL*Plus by entering the command sqlplus followed by the appropriate connection string for your database.

For example:

sqlplus :**username/password@hostname**:**port/service_name**
username is Oracle username and its password

Hostname is called the IP address of the server
the port is called port no which mentioned tnsnames
the service name is called listener

Replace username, password, hostname, port, and service_name with the appropriate values for your database configuration.

The rest of the content is exceeding so let me share those in next post .

#Day 54

Continuity of day 53 **Post**

#ORACLE
#RMAN
#RECOVERYMANAGER

Set ORACLE SID to Connect Database

Example:
[oracle@ol7 ~]$ export ORACLE_SID=test
[oracle@ol7 ~]$ sqlplus /as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production
Version 19.14.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Use SQL*Plus to execute SQL, PL/SQL and SQL*Plus statements.

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Jul 10 11:23:28 2023
Version 19.14.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Enter user-name: /as sysdba
Connected to an idle instance.

SQL>

Once connected, you can execute a simple query to check the database status.

For example:
Below command was executed in my local system.

SELECT INSTANCE_NAME, STATUS FROM V$INSTANCE;

SQL> SELECT INSTANCE_NAME, STATUS FROM V$INSTANCE;

INSTANCE_NAME STATUS
---------- ------------
test OPEN

This query retrieves information about the instance name and status of the database. If the database is running, the status should be "**OPEN**". If the status is "**MOUNTED**" or "**CLOSED**", the database is not currently open and accessible.

Now connect Rman prompt from sql prompt by executing below command called Rman target.

SQL> !rman target/

Recovery Manager: Release 19.0.0.0.0 - Production on Mon Jul 10 10:07:47 2023
Version 19.14.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

connected to target database: TEST (DBID=2427001681)

So this is how basic steps are to be followed before connecting to Recovery Manager prompt window from Oracle database or Linux Os.

#Day 55
#ORACLE
#RMAN
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP

Continuity of Day 52 As we discussed on that day types of backup let us see how to take full backup in rman today.

How to take a **Full Backup** in rman oracle what are all steps to be followed?

To take a full backup using RMAN (Recovery Manager) in Oracle, you need to follow these steps:

Connect to RMAN: Open a command prompt or terminal and connect to RMAN using the following command:

rman target /
This command connects to the local database using the operating system authentication.

Set the Backup Destination: Set the location where you want to store the backup files. You can use the CONFIGURE command in RMAN to set the backup destination. For example, to set the backup destination to '/backup', use the following command:

CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '**/backup**/%**U**';
Start the Backup: Issue the BACKUP command to start the full backup. Specify the DATABASE option to back up the entire database. For example:

BACKUP DATABASE;

This command will initiate the backup of all database files, including data files, control files, and archived redo logs.

Below is the output from my local system.

Monitor the Backup Progress: RMAN will display the progress of the backup operation, including the files being backed up and the completion percentage.

Verify the Backup: After the backup completes, it is recommended to verify the backup to ensure its integrity. You can use the VALIDATE command to perform a backup validation. For example:

VALIDATE DATABASE;
This command checks the backup files against the source database for any physical corruptions.

Close RMAN: Once the backup and validation are complete, you can exit RMAN by typing:

EXIT;

Note: It's important to have a proper backup strategy, including regular full backups, incremental backups, and archiving of redo logs. Additionally, consider storing your backups in a separate location from the database server to ensure data safety in case of a disaster.

Remember to consult the Oracle documentation and follow best practices for backup and recovery operations specific to your environment.

Here I am attaching my local system database rman backup output.
Executing
Example :backup database;

And if u notice that backup has been taken including spfile and controlfile then which means after connecting to rman dba would have been configured auto backup on. That's the reason why each and every time u take a backup automatically controlfile backup also is initiated .I let u know about configuring multiple options in next post.

```
Starting backup at 11-JUL-23
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=62 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002 name=/u01/test/oradata/sysaux01.dbf
input datafile file number=00001 name=/u01/test/oradata/system01.dbf
input datafile file number=00003 name=/u01/test/oradata/undotbs01.dbf
input datafile file number=00004 name=/u01/test/oradata/users01.dbf
input datafile file number=00005 name=/u01/test/oradata/userstest.dbf
channel ORA_DISK_1: starting piece 1 at 11-JUL-23
channel ORA_DISK_1: finished piece 1 at 11-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/0e210oho_14_1_1 tag=TAG20230711T171328 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 11-JUL-23

Starting Control File and SPFILE Autobackup at 11-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/c-2427001681-20230711-00 comment=NONE
Finished Control File and SPFILE Autobackup at 11-JUL-23
```

#Day 56
#ORACLE
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP

Incremental Backup in ORACLE Rman.

Continuity of **Day** 52 among types of backup let us see how to take **incremental backup**.

How to take **Incremental Backup** in rman oracle with steps?

To take an incremental backup using RMAN (Recovery Manager) in Oracle, you can follow these steps:

Connect to the RMAN utility: Open a command prompt or terminal and connect to the RMAN utility by executing the following command:

rman target /
This will connect you to the default target database.

Check the current backup status (optional): You can check the current backup status and details by running the following RMAN command:

RMAN> **list backup**;
This step is optional but can provide helpful information before proceeding with the incremental backup.

Set the backup type to incremental: Specify the backup type as total by

Executing the following RMAN command:

RMAN> backup incremental level <level_number> database;
Replace <level_number> with the desired incremental level (e.g., 0 for a full backup, 1 for the first incremental backup, 2 for the second incremental backup, and so on).

Configure the backup destination: Set the backup destination to a specific location by executing the following RMAN command:

RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '<backup_location>';
Replace <backup_location> with the path where you want to store the backup files.

Start the incremental backup: Begin the incremental backup process by running the following RMAN command:

RMAN> backup incremental level <level_number> database;
Again, replace <level_number> with the desired incremental level.

Monitor the backup progress (optional): You can monitor the progress of the backup by executing the following RMAN command:

RMAN> show progress;
This step is optional but can provide real-time information about the backup process.

Verify the backup (optional): After the backup completes, you can verify its integrity by running the following RMAN command:

RMAN> restore database validate;
This step is optional but recommended to ensure the backup files are valid and can be used for recovery if needed.

Exit RMAN: Once the backup process is complete, you can exit the RMAN utility by executing the following command:

RMAN> exit;
This will return you to the command prompt or terminal.

Remember to regularly schedule incremental backups to ensure a consistent and up-to-date backup strategy for your Oracle database.

Here I have taken level 1 incremental backup from local system.Results I have attached it as an image.

```
RMAN> backup incremental level 1 database;

Starting backup at 13-JUL-23
using channel ORA_DISK_1
no parent backup or copy of datafile 2 found
no parent backup or copy of datafile 1 found
no parent backup or copy of datafile 3 found
no parent backup or copy of datafile 4 found
no parent backup or copy of datafile 5 found
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002 name=/u01/test/oradata/sysaux01.dbf
input datafile file number=00001 name=/u01/test/oradata/system01.dbf
input datafile file number=00003 name=/u01/test/oradata/undotbs01.dbf
input datafile file number=00004 name=/u01/test/oradata/users01.dbf
input datafile file number=00005 name=/u01/test/oradata/userstest.dbf
channel ORA_DISK_1: starting piece 1 at 13-JUL-23
channel ORA_DISK_1: finished piece 1 at 13-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/0g2157nm_16_1_1 tag=TAG20230713T095710 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 13-JUL-23

Starting Control File and SPFILE Autobackup at 13-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/c-2427001681-20230713-00 comment=NONE
Finished Control File and SPFILE Autobackup at 13-JUL-23
```

#Day 57
#ORACLE
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP.

How to take archive log backup in rman Oracle steps and tricks in Oracle 19**c** ?

Today let us see Archivelog backup among types of backups in Oracle

List of backup in rman

-->Full Backup
-->Incremental Backup
-->**Archive Log Backup**
-->Control File Backup
-->Image Copy Backup

RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF <number_of_days>;

Set the destination for the backup:

RMAN> CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '<backup_destination_path>/arch_%U';

Start the backup:

RMAN> BACKUP AS BACKUPSET ARCHIVELOG ALL;

This command will back up all the available archive logs.

You can also specify additional options, such as:

Specific range of archive logs:

RMAN> BACKUP AS BACKUPSET ARCHIVELOG FROM SEQUENCE <start_sequence_number> UNTIL SEQUENCE <end_sequence_number>;

RMAN> BACKUP AS BACKUPSET ARCHIVELOG ALL THREAD <thread_number>;

If you want to compress the backup, you can use the following command:

RMAN> BACKUP AS COMPRESSED BACKUPSET ARCHIVELOG ALL;

After the backup completes, you can cross-check the backup using the following command:

RMAN> CROSSCHECK BACKUP;

To view the details of the backup, use the following command:

RMAN> REPORT OBSOLETE;

This command will show you any obsolete backups that can be deleted.

Finally,

exit RMAN:

RMAN> EXIT;

These steps outline the basic process of taking an archive log backup using RMAN in

Oracle 19c.
Make sure to adjust the commands according to your specific environment and requirements.

As LinkedIn has some limited content allowed here I have attached a backup of archive log output as an attachment.

```
[oracle@ol7 ~]$ export ORACLE_SID=test
[oracle@ol7 ~]$ sqlplus

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Jul 17 12:44:01 2023
Version 19.14.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Enter user-name: /as sysdba
Connected to an idle instance.

SQL> startup;
ORACLE instance started.

Total System Global Area 1610610768 bytes
Fixed Size                  8910928 bytes
Variable Size            1581252608 bytes
Database Buffers           16777216 bytes
Redo Buffers                3670016 bytes
Database mounted.
Database opened.
SQL> !rman target/

Recovery Manager: Release 19.0.0.0.0 - Production on Mon Jul 17 12:44:28 2023
Version 19.14.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: TEST (DBID=2427001681)

RMAN> BACKUP AS BACKUPSET ARCHIVELOG ALL;

Starting backup at 17-JUL-23
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=89 device type=DISK
```

```
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=1 RECID=7 STAMP=1131384445
input archived log thread=1 sequence=2 RECID=8 STAMP=1131385822
input archived log thread=1 sequence=3 RECID=9 STAMP=1131385845
input archived log thread=1 sequence=4 RECID=10 STAMP=1131386000
input archived log thread=1 sequence=5 RECID=11 STAMP=1131386006
input archived log thread=1 sequence=6 RECID=12 STAMP=1131388021
input archived log thread=1 sequence=7 RECID=13 STAMP=1132220895
input archived log thread=1 sequence=8 RECID=14 STAMP=1132220908
input archived log thread=1 sequence=9 RECID=15 STAMP=1135002052
input archived log thread=1 sequence=10 RECID=16 STAMP=1135850788
input archived log thread=1 sequence=11 RECID=17 STAMP=1135852448
input archived log thread=1 sequence=12 RECID=18 STAMP=1135886440
input archived log thread=1 sequence=13 RECID=19 STAMP=1135965654
input archived log thread=1 sequence=14 RECID=20 STAMP=1136041254
input archived log thread=1 sequence=15 RECID=21 STAMP=1136631723
input archived log thread=1 sequence=16 RECID=22 STAMP=1137438586
input archived log thread=1 sequence=17 RECID=23 STAMP=1137439458
input archived log thread=1 sequence=18 RECID=24 STAMP=1137439500
input archived log thread=1 sequence=19 RECID=25 STAMP=1137603608
input archived log thread=1 sequence=20 RECID=26 STAMP=1137639615
input archived log thread=1 sequence=21 RECID=27 STAMP=1137700861
input archived log thread=1 sequence=22 RECID=28 STAMP=1137744010
input archived log thread=1 sequence=23 RECID=29 STAMP=1137783663
input archived log thread=1 sequence=24 RECID=30 STAMP=1137823123
input archived log thread=1 sequence=25 RECID=31 STAMP=1137862811
input archived log thread=1 sequence=26 RECID=32 STAMP=1137958937
input archived log thread=1 sequence=27 RECID=33 STAMP=1138024856
input archived log thread=1 sequence=28 RECID=34 STAMP=1138057238
input archived log thread=1 sequence=29 RECID=35 STAMP=1138093237
input archived log thread=1 sequence=30 RECID=36 STAMP=1138132845
input archived log thread=1 sequence=31 RECID=37 STAMP=1138172446
input archived log thread=1 sequence=32 RECID=38 STAMP=1138212017
input archived log thread=1 sequence=33 RECID=39 STAMP=1138241993
input archived log thread=1 sequence=34 RECID=40 STAMP=1138291234
input archived log thread=1 sequence=35 RECID=41 STAMP=1138318064
input archived log thread=1 sequence=36 RECID=42 STAMP=1138785565
```

```
input archived log thread=1 sequence=36 RECID=42 STAMP=1138785565
input archived log thread=1 sequence=37 RECID=43 STAMP=1138787261
input archived log thread=1 sequence=38 RECID=44 STAMP=1138817036
input archived log thread=1 sequence=39 RECID=45 STAMP=1138917658
input archived log thread=1 sequence=40 RECID=46 STAMP=1138928458
input archived log thread=1 sequence=41 RECID=47 STAMP=1139399837
input archived log thread=1 sequence=42 RECID=48 STAMP=1141470523
input archived log thread=1 sequence=43 RECID=49 STAMP=1141470589
input archived log thread=1 sequence=44 RECID=50 STAMP=1141495205
input archived log thread=1 sequence=45 RECID=51 STAMP=1141527664
input archived log thread=1 sequence=46 RECID=52 STAMP=1141554403
input archived log thread=1 sequence=47 RECID=53 STAMP=1141923760
input archived log thread=1 sequence=48 RECID=54 STAMP=1142013323
input archived log thread=1 sequence=49 RECID=55 STAMP=1142100054
input archived log thread=1 sequence=50 RECID=56 STAMP=1142197765
input archived log thread=1 sequence=51 RECID=57 STAMP=1142226018
input archived log thread=1 sequence=52 RECID=58 STAMP=1142426655
input archived log thread=1 sequence=53 RECID=59 STAMP=1142426691
channel ORA_DISK_1: starting piece 1 at 17-JUL-23
channel ORA_DISK_1: finished piece 1 at 17-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/0i21g324_18_1_1 tag=TAG20230717T124452 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=4 RECID=4 STAMP=1131384132
input archived log thread=1 sequence=5 RECID=5 STAMP=1131384133
input archived log thread=1 sequence=6 RECID=6 STAMP=1131384133
channel ORA_DISK_1: starting piece 1 at 17-JUL-23
channel ORA_DISK_1: finished piece 1 at 17-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/0j21g32j_19_1_1 tag=TAG20230717T124452 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 17-JUL-23

Starting Control File and SPFILE Autobackup at 17-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/c-2427001681-20230717-00 comment=NONE
Finished Control File and SPFILE Autobackup at 17-JUL-23

RMAN>
```

#Day 58
#ORACLE
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP.

How to take **Control file** backup in rman Oracle steps and tricks in Oracle ?

Today let us see Control file backup among types of backups in Oracle

List of backup in rman

-->Full Backup
-->Incremental Backup
-->Archive Log Backup
-->**Control File Backup**
-->Image Copy Backup

How to take Control file backup **in rman Oracle steps and tricks in Oracle** ?

To take a control file backup in Oracle, **you can use the SQL*Plus utility or the Oracle Recovery Manager** (**RMAN**). **Here's how you can do it using both methods**:

Using SQL*Plus:

Step 1: **Connect to the Oracle database using SQL*Plus as a privileged user** (**e.g.**, **SYSDBA or SYSOPER**).

sqlplus / as sysdba

Step 2: **Run the following SQL command to create a control file backup**:

ALTER DATABASE BACKUP CONTROLFILE TO '/path/to/backup/controlfile.bkp';

Replace /path/to/backup/ with the desired location where you want to store the control file backup.

Step 3: **Optionally**, **you can also create a binary backup of the control file by adding the BINARY keyword to the command**:

ALTER DATABASE BACKUP CONTROLFILE TO '/path/to/backup/controlfile.bkp' BINARY;

Using Oracle Recovery Manager (**RMAN**):
Step 1: **Connect to RMAN as a privileged user** (**e.g.**, **SYSDBA or SYSOPER**).

rman target /

Step 2: **Run the following RMAN command to take a control file backup**:

BACKUP CURRENT CONTROLFILE TO '/path/to/backup/controlfile.bkp';

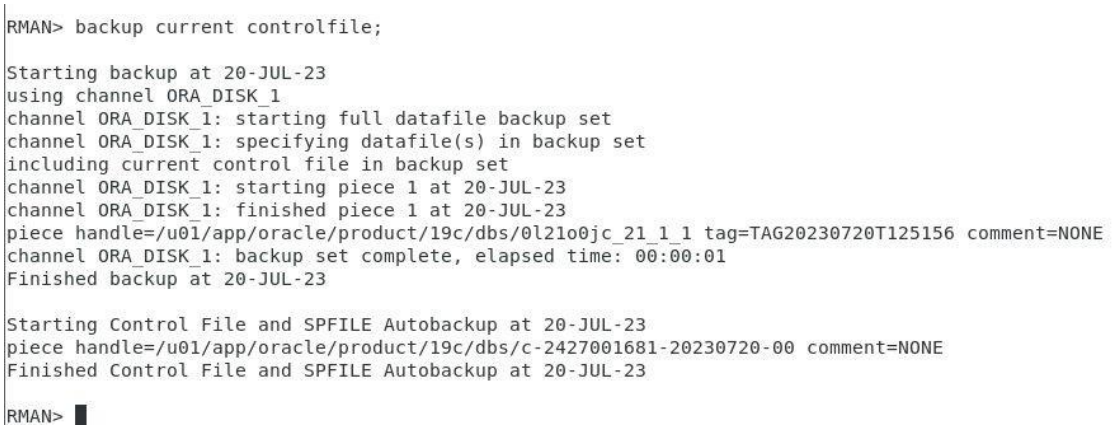
Replace /path/to/backup/ with the desired location where you want to store the control file backup.

Step 3: **Optionally**, **you can also create a binary backup of the control file by adding the AS BINARY clause to the command**:

BACKUP AS BINARY CURRENT CONTROLFILE TO '/path/to/backup/controlfile.bkp';

Remember that it's essential to regularly back up your control files, as they are critical components of your Oracle database. In case of a database failure, having a recent control file backup will aid in the recovery process. Also, store the backup files in a secure location to prevent any accidental loss.

Here I have attached controlfile backup taken in rman from my local system .
Activate to view larger image,



#Day 59
#ORACLE
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP.

Today let us see **Image Copy Backup** among types of backups in Oracle

List of backup in rman
-->Full Backup
-->Incremental Backup
-->Archive Log Backup
-->Control File Backup
-->**Image Copy Backup**

how to take Image Copy Backup in rman oracle tips and tricks ?

Taking an Image Copy Backup in Oracle using RMAN (Recovery Manager) is a useful method to create a physical-level backup of the database files. An Image Copy Backup is a byte-for-byte copy of the data files, control files, and archived redo logs. It can be helpful for certain scenarios such as cloning databases, offloading backups to a different location, or performing backups while the database is online.

Here are the steps to take an Image Copy Backup using RMAN in Oracle:

Connect to RMAN:

Open a terminal or command prompt and connect to RMAN using the following command:

rman target /

This will connect to the default target database using operating system authentication.

Set the Backup Destination:
Before taking the backup, set the destination for the image copy. You can specify a disk or tape location where the image copy will be stored.

CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO COPY;

CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT
'/path/to/backup/destination/%U';

Replace /path/to/backup/destination/ with the actual directory path where you want to store the image copy backup.

Start the Backup:

Now, **initiate the image copy backup using the RMAN BACKUP command**:
BACKUP AS COPY DATABASE;

This command will take an image copy backup of the entire database, including data files, control files, and archived redo logs.

Alternatively, if you want to take an image copy backup of specific data files or tablespaces, you can use the BACKUP AS COPY command with the DATAFILE or TABLESPACE options:

BACKUP AS COPY DATAFILE '/path/to/datafile.dbf';
or
BACKUP AS COPY TABLESPACE users, example;

Validate the Backup (**Optional**):
You can also validate the image copy backup after it is completed to ensure its

integrity. Validation compares the files in the image copy with the source files to confirm that they match.

VALIDATE COPY OF DATABASE;
Exit RMAN:

Once the backup process is completed, **you can exit RMAN by typing**:
EXIT;

Remember to regularly test your backups to ensure they are recoverable in case of any data loss or disaster scenarios. Additionally, consider storing backups on separate media to enhance data protection.

Here I have attached image copy of database backup taken in rman from my local system .

```
RMAN> BACKUP AS COPY DATABASE;

Starting backup at 21-JUL-23
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
input datafile file number=00002 name=/u01/test/oradata/sysaux01.dbf
output file name=/u01/app/oracle/product/19c/dbs/data_D-TEST_I-2427001681_TS-SYSAUX_FNO-2_0n21q916 tag=TAG20230721T092806 RECID=1 STAMP=1142760491
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00001 name=/u01/test/oradata/system01.dbf
output file name=/u01/app/oracle/product/19c/dbs/data_D-TEST_I-2427001681_TS-SYSTEM_FNO-1_0o21q91d tag=TAG20230721T092806 RECID=2 STAMP=1142760496
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003 name=/u01/test/oradata/undotbs01.dbf
output file name=/u01/app/oracle/product/19c/dbs/data_D-TEST_I-2427001681_TS-UNDOTBS1_FNO-3_0p21q91g tag=TAG20230721T092806 RECID=3 STAMP=1142760499
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00004 name=/u01/test/oradata/users01.dbf
output file name=/u01/app/oracle/product/19c/dbs/data_D-TEST_I-2427001681_TS-USERS_FNO-4_0q21q91j tag=TAG20230721T092806 RECID=4 STAMP=1142760500
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00005 name=/u01/test/oradata/userstest.dbf
output file name=/u01/app/oracle/product/19c/dbs/data_D-TEST_I-2427001681_TS-USERSTEST_FNO-5_0r21q91l tag=TAG20230721T092806 RECID=5 STAMP=1142760501
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 21-JUL-23

Starting Control File and SPFILE Autobackup at 21-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/c-2427001681-20230721-00 comment=NONE
Finished Control File and SPFILE Autobackup at 21-JUL-23
```

#Day 60
#ORACLE
#RECOVERYMANAGER
#SERVERMANAGEDBACKUP

So last week we saw about types of backups taken in rman but apart from that there are a few more backups that are taken for backup and recovery called tablespace backup and datafile backup in rman that let us discuss later while we see about recovery in rman.

Today let us see about different types of backup Configurations in rman.
Actually, **this configuration is done in the environment as per the business requirements**.

what retention policy to redundancy in rman oracle in detail?

In Oracle Recovery Manager (RMAN), the retention policy to redundancy is a concept used to manage backup sets and control the number of redundant backups retained in the database. It

ensures that you maintain a specific number of backups regardless of their age. This approach allows you to have multiple backups available for recovery purposes while avoiding an excessive number of redundant backups that consume storage space unnecessarily.

The retention policy to redundancy can be configured using the RETENTION POLICY TO REDUNDANCY clause when executing the RMAN BACKUP command or setting the retention policy for the database.

The syntax for setting the retention policy to redundancy is as follows:

```
BACKUP [ AS COPY | AS COMPRESSED BACKUPSET | ... ] DATABASE
[ PLUS ARCHIVELOG | ARCHIVELOG ALL | ... ]
[ WITH TAG 'tag_name' ]
RETENTION POLICY TO REDUNDANCY [ n ];
```

Here, n represents the number of backups you want to retain as redundant backups. RMAN will automatically manage the backup sets to comply with this redundancy setting.

For example, set the retention policy to redundancy as RETENTION POLICY TO REDUNDANCY 2. RMAN will ensure that there are always two redundant backups available for recovery purposes, even if older backups are expired based on other retention policies.

RMAN uses the following rules to maintain the specified redundancy level:

If the number of redundant backups is less than the specified redundancy level (n), RMAN will create additional backups to reach the required number.

If the number of redundant backups is greater than or equal to the specified redundancy level, RMAN will not create new backups. However, it will not delete any existing backups either, as they are considered redundant.

If you perform a DELETE command in RMAN, RMAN will obey the retention policy to redundancy and avoid deleting backups that are still needed to meet the specified redundancy level.

If you set the retention policy to redundancy without specifying a value for n, RMAN will automatically choose a default value for redundancy based on your backup configuration and other retention policies.

Below picture attached is executed in my system rest let us see in the next post.

```
RMAN> show all;

RMAN configuration parameters for database with db_unique_name TEST are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/u01/app/oracle/product/19c/dbs/snapcf_test.f'; # default

RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 1;

new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1;
new RMAN configuration parameters are successfully stored
```

#Day 61
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

What is CONFIGURE BACKUP OPTIMIZATION ON and OFF; # default in rman oracle in detail ?

The CONFIGURE BACKUP OPTIMIZATION command in Oracle RMAN (Recovery Manager) is used to enable or disable the backup optimization feature. Backup optimization is a mechanism in RMAN that improves the efficiency of backups by skipping data files that have not been changed since the last backup. This feature is enabled by default.

When the BACKUP OPTIMIZATION is set to ON, RMAN will use a technique called "incremental backup" to skip unchanged data blocks during subsequent backups.

Here's how it works in more detail:

First Backup: When you perform a full backup, RMAN copies all the data blocks from the data files to the backup set.

Incremental Backup: After the first full backup, when you perform subsequent backups (e.g., daily backups), RMAN uses the information stored in the control file to identify data blocks that have changed since the last backup. Only the changed blocks are included in the incremental backup, making the backup process faster and using less storage space.

Level 0 **and Level** 1 **Incremental Backups**: RMAN supports two types of incremental backups:

Level 0 and Level 1. A Level 0 incremental backup is the same as a full backup, while a Level 1 backup contains the incremental changes since the last Level 0 backup.

By default, RMAN will use backup optimization and perform Level 1 incremental backups, which means it will only back up changed blocks since the last Level 0 or Level 1 backup. This default setting is generally beneficial as it optimizes the backup process.

However, in some cases, you may want to turn off backup optimization. For example, if you encounter issues with the control file or the backup catalog, turning off backup optimization can help ensure that all data blocks are backed up completely without relying on the incremental approach. To disable backup optimization,

You can use the following command in RMAN:

CONFIGURE BACKUP OPTIMIZATION OFF;

Remember that disabling backup optimization may increase the size and time required for backups since RMAN will no longer skip unchanged blocks during subsequent backups.

It's essential to understand your specific backup requirements and environment before modifying any RMAN configurations to ensure the best balance between backup performance and data protection.

Below I have attached it after configuring optimization on what is happening in rman with an example in my local system . And note what I highlighted in yellow.

```
RMAN> backup incremental level 1 database;

Starting backup at 26-JUL-23
using channel ORA_DISK_1
no parent backup or copy of datafile 2 found
no parent backup or copy of datafile 1 found
no parent backup or copy of datafile 3 found
no parent backup or copy of datafile 4 found
no parent backup or copy of datafile 5 found
channel ORA_DISK_1: starting incremental level 0 datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002 name=/u01/test/oradata/sysaux01.dbf
input datafile file number=00001 name=/u01/test/oradata/system01.dbf
input datafile file number=00003 name=/u01/test/oradata/undotbs01.dbf
input datafile file number=00004 name=/u01/test/oradata/users01.dbf
input datafile file number=00005 name=/u01/test/oradata/userstest.dbf
channel ORA_DISK_1: starting piece 1 at 26-JUL-23
channel ORA_DISK_1: finished piece 1 at 26-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/1a227i1v_42_1_1 tag=TAG20230726T102151 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 26-JUL-23

Starting Control File and SPFILE Autobackup at 26-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/c-2427001681-20230726-04 comment=NONE
Finished Control File and SPFILE Autobackup at 26-JUL-23
```

```
piece handle=/u01/app/oracle/product/19c/dbs/0u227gq5_30_1_1 tag=TAG20230726T100011 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 26-JUL-23

Starting backup at 26-JUL-23
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00002 name=/u01/test/oradata/sysaux01.dbf
input datafile file number=00001 name=/u01/test/oradata/system01.dbf
input datafile file number=00003 name=/u01/test/oradata/undotbs01.dbf
input datafile file number=00004 name=/u01/test/oradata/users01.dbf
input datafile file number=00005 name=/u01/test/oradata/userstest.dbf
channel ORA_DISK_1: starting piece 1 at 26-JUL-23
channel ORA_DISK_1: finished piece 1 at 26-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/0v227gq8_31_1_1 tag=TAG20230726T100039 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:25
Finished backup at 26-JUL-23

Starting backup at 26-JUL-23
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=58 RECID=64 STAMP=1143194465
channel ORA_DISK_1: starting piece 1 at 26-JUL-23
channel ORA_DISK_1: finished piece 1 at 26-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/10227gr2_32_1_1 tag=TAG20230726T100105 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 26-JUL-23

Starting Control File and SPFILE Autobackup at 26-JUL-23
piece handle=/u01/app/oracle/product/19c/dbs/c-2427001681-20230726-00 comment=NONE
Finished Control File and SPFILE Autobackup at 26-JUL-23
```

```
RMAN> backup database plus archivelog;
```

```
RMAN> configure backup optimization on;

new RMAN configuration parameters:
CONFIGURE BACKUP OPTIMIZATION ON;
new RMAN configuration parameters are successfully stored
```

#Day 62
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

Continuity of day 61 post configure **optimization** on

in yesterday's post, I highlighted in yellow that after configuring optimization .

what happens if I take a backup for the first time so there are no changes?

In today's post let's see what happens in case if we take the same backup of incremental database what does it do and what shows while taking backup in rman.

Note: Attached output please check.

So Here u can see the same level of incremental backup initiated a second time with no changes. So it's getting skipped. So that's how when the optimization on enabled. Activate to view larger image,

#Day 63
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

What is CONFIGURE DEFAULT DEVICE TYPE TO disk; in rman with example in detail in oracle?

In Oracle RMAN (Recovery Manager), the CONFIGURE DEFAULT DEVICE TYPE TO disk command sets the default device type to disk. This means that RMAN will use the

disk as the default destination for backups and restores unless specified otherwise in the RMAN commands.

When you configure RMAN with CONFIGURE DEFAULT DEVICE TYPE TO disk, any RMAN backup or restore operation that doesn't explicitly specify a different device type will use disk storage as the default. This can simplify your RMAN commands since you won't need to specify the device type for every backup or restore operation if you primarily use disk storage.

Here's how to use the **CONFIGURE DEFAULT DEVICE TYPE TO disk** command with examples:

Setting the Default Device Type to Disk:

RMAN> CONFIGURE DEFAULT DEVICE TYPE TO disk;
After running this command, any subsequent RMAN backups and restores will use disk storage as the default.

Performing a Backup with Default Device Type (Disk):

RMAN> BACKUP DATABASE;
In this example, the BACKUP DATABASE command will back up the entire database to disk because the default device type is set to disk. You don't need to mention the device type explicitly.

Restoring from a Backup with Default Device Type (Disk):

RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE;

In these examples, the RESTORE DATABASE and RECOVER DATABASE commands will restore and recover the database using the backups stored on disk because the default device type is set to disk. You don't need to specify the device type for these operations.

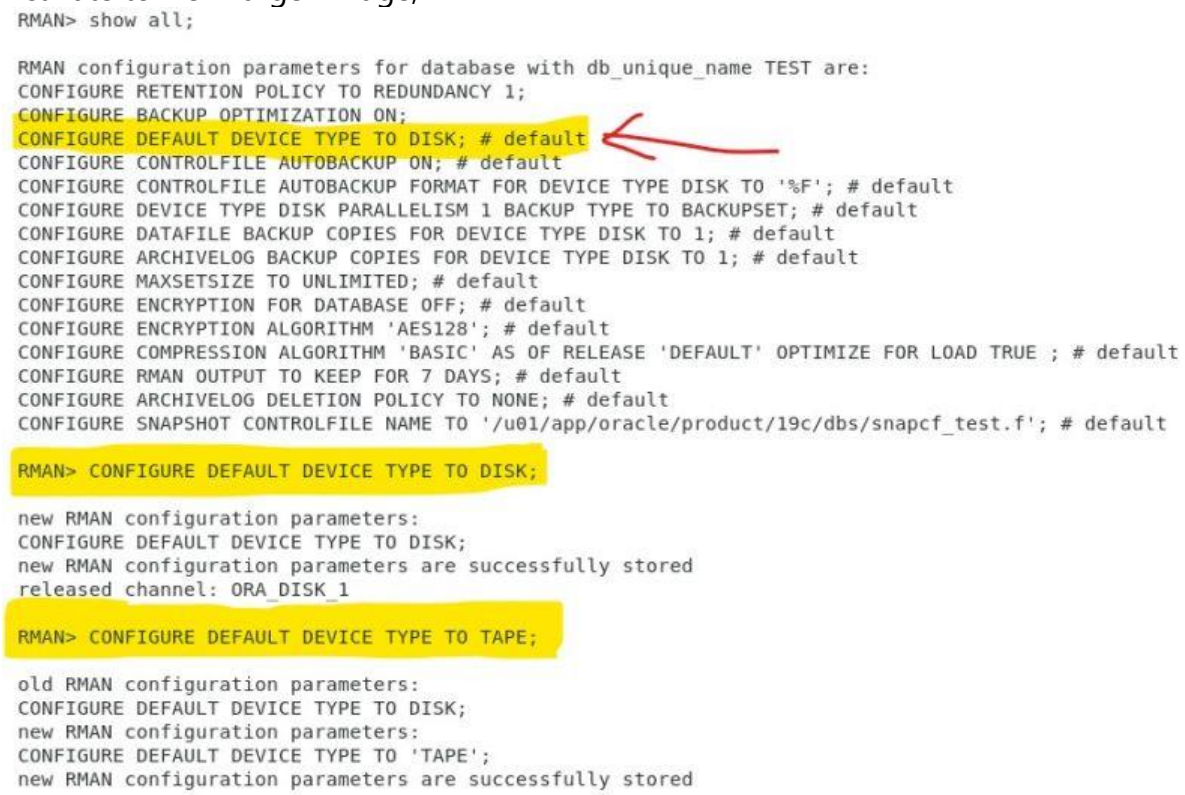
Overriding the Default Device Type for a Specific Backup:

RMAN> BACKUP DATABASE DEVICE TYPE sbt;
In this example, the BACKUP DATABASE command specifies DEVICE TYPE sbt, which means it will use the tape device type (sbt - System Backup to Tape) for this specific backup, even though the default device type is disk.

RMAN offers various options and configurations to meet specific backup and

recovery requirements. The CONFIGURE DEFAULT DEVICE TYPE TO disk command is just one aspect of RMAN's flexibility and customization capabilities, allowing you to streamline disk-based backups and restores.

Note the attachment about the configuration of disk and tape storage in rman. Activate to view larger image,



#Day 64
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

What is **CONFIGURE CONTROLFILE AUTOBACKUP ON**; # default in rman detail oracle ?

In Oracle Database, the RMAN (Recovery Manager) utility is used for database backup and recovery operations. When using RMAN to back up a database, one critical aspect is the control

file auto backup feature.

Control file auto backup is a feature that automatically backs up the control file after certain operations in RMAN. The control file is a critical component of the Oracle database as it maintains the metadata about the database structure and its changes. Having a recent backup of the control file is essential for recovery purposes in case of database failure.

The **CONFIGURE CONTROLFILE AUTOBACKUP ON**; command enables control file auto backup in RMAN. When this command is executed, RMAN will automatically create a backup of the control file whenever certain RMAN operations are performed, such as:

Before and after a BACKUP or COPY command that creates a data file or control file backup.
Before and after a BACKUP or COPY command that creates a backup of the current control file.
After a SWITCH command is issued to change the current redo log file.
By default, in recent versions of Oracle (including Oracle 11g and above), control file auto backup is usually enabled by default in the RMAN configuration. When you run RMAN without explicitly configuring this option, the control file auto backup feature is already active.

Control file auto backups are stored in the Fast Recovery Area (if configured) or a user-defined location, ensuring that a recent backup of the control file is always available for recovery purposes.

To see the current configuration of control file auto backup in RMAN, you can use the following command:

SHOW CONTROLFILE AUTOBACKUP;

This command will display whether the control file auto backup is enabled or disabled in your RMAN configuration.

Below have executed from my local system .

RMAN> show controlfile autobackup;

RMAN configuration parameters for database with db_unique_name TEST are:
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;

new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored.

#Day 65
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

In Oracle Database, the **CONFIGURE CONTROLFILE AUTOBACKUP FORMAT** command is used in Oracle Recovery Manager (RMAN) to specify the format for naming and storing control file autobackup files. Control file autobackups are automatically generated backups of the database control file, which are crucial for database recovery operations.

Let's break down the command you've provided:

CONFIGURE: This keyword is used to configure various settings within RMAN.

CONTROLFILE AUTOBACKUP FORMAT: This part of the command specifies that you want to configure the format for naming control file autobackup files.

FOR DEVICE TYPE DISK: This specifies that the configuration is intended for the device type "DISK," which means the autobackup files will be stored on disk storage.

TO '%F': This is the format specification for naming the autobackup files. The %F is a placeholder that RMAN replaces with a timestamp and other information to generate a unique filename for each autobackup. The resulting filename will include the date and time of the autobackup, making it easy to identify and manage backup files.

In summary, this RMAN command is configuring the format for naming control file autobackup files that are stored on disk. The use of %F in the format string ensures that RMAN generates unique filenames with timestamps for each autobackup, making it convenient for backup management and recovery scenarios.

Here's an example of how the %F format specifier works:

Suppose you execute the backup command with the specified configuration and the backup takes place on August 4, 2023, at 15:30:00. The resulting **control file autobackup filename could look like this**:

c-3487264808-20230804-01
In this example, c indicates a control file autobackup, followed by a system-generated unique identifier (3487264808), the date (20230804 for August 4, 2023), and a sequence number (01 for the first autobackup on that date). The %F format specifier is responsible for generating this filename based on the current timestamp and other relevant information.

Another Example:
Here is an example of how to use the CONFIGURE CONTROLFILE AUTOBACKUP FORMAT command:

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '+dgroup1/%F';

This command configures the control file autobackups to be written to the +dgroup1 disk group and to be named with the format +dgroup1/c-IIIIIIIIII-YYYYMMDD-QQ.

#Day 66
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

In Oracle RMAN (Recovery Manager), you can configure the device type, disk parallelism, and backup type using RMAN commands and configuration settings in your RMAN scripts or on the command line. Here's how you can configure these settings and an example:

Device Type Configuration:
The DEVICE TYPE setting allows you to specify the type of device you want to use for your backups. In this example, you want to configure it to use the disk as the device type.

CONFIGURE DEVICE TYPE DISK PARALLELISM 1;

This command sets the device type to disk and specifies a parallelism of 1, which means that RMAN will use one channel for backup operations.
Backup Type Configuration:
The BACKUP TYPE setting allows you to specify the default backup type. In this case, you want to set it to BACKUPSET, which is the default.

CONFIGURE DEFAULT DEVICE TYPE TO DISK; **CONFIGURE BACKUP TYPE TO BACKUPSET**;

These commands set the default device type to disk and the default backup type to backupset.

Now, let's put it all together in a simple example:

-- **Connect to the RMAN utility rman target** / -- Configure device type and backup type
CONFIGURE DEVICE TYPE DISK PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
CONFIGURE BACKUP TYPE TO BACKUPSET;

-- Perform a backup using the configured settings **BACKUP DATABASE**;

In this example:
We connect to the RMAN utility using rman target / to target the current database.

We configure the device type to disk with a parallelism of 1.

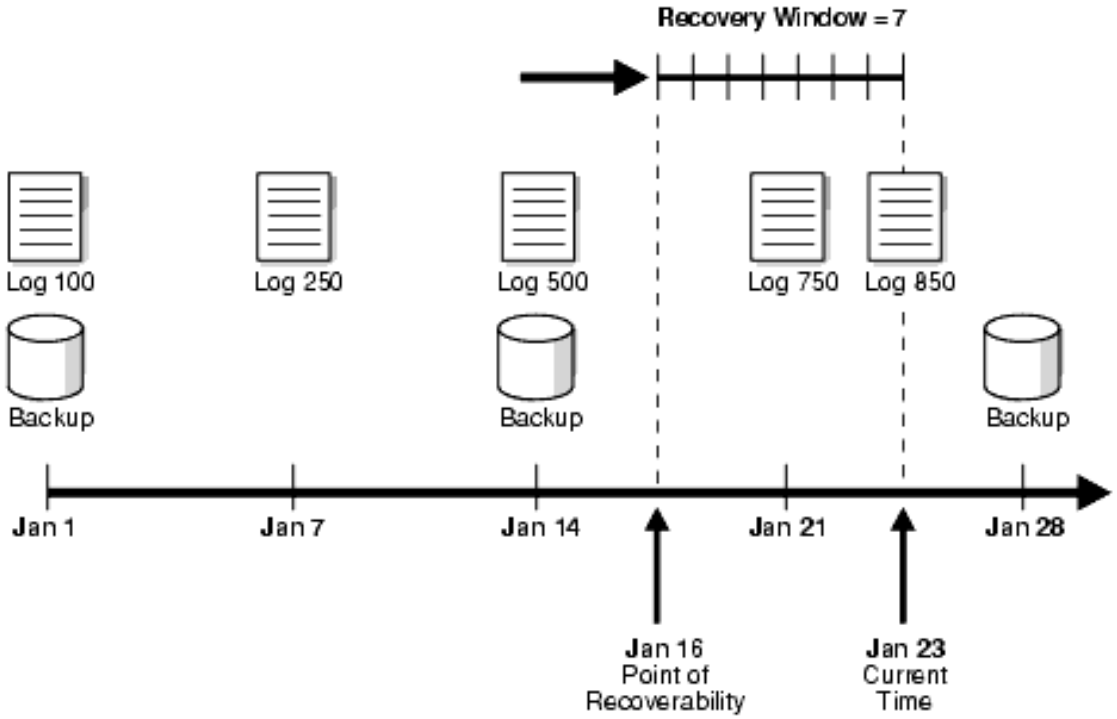
We set the default device type to disk and the default backup type to backupset.

Finally, we perform a database backup using the configured settings by running the BACKUP DATABASE command.

This backup will use one channel (parallelism 1) and create backup sets as specified in the configuration.

You can customize these settings further to suit your specific backup requirements, such as specifying backup destinations, retention policies, and backup piece size, among other options.

Activate to view larger image,



#Day 67
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

The RMAN (Recovery Manager) command you provided,
"**CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO** 1;" is used in Oracle Database to configure the number of backup copies that should be created for datafiles when performing backups to disk using RMAN. Let me break down the command for you in more detail:

CONFIGURE: This keyword is used to configure various settings and parameters within RMAN.

DATAFILE BACKUP COPIES: This part of the command specifies that you want to configure the number of backup copies for datafiles. In other words, it determines how many backup copies of each datafile should be created during a backup operation.

FOR DEVICE TYPE DISK: This part specifies that the configuration is specific to backups on disk. It means that the setting you are defining is for backup sets stored on disk devices. You can also have different configurations for other device types like tape.

TO 1: This sets the number of backup copies to 1. In other words, it instructs RMAN to create only one backup copy of each datafile during a backup operation.

By default, RMAN creates one backup copy when you perform a backup to disk. So, the command you provided is setting the configuration to the default value.
You can change the number of backup copies to a value greater than 1 if you want to create multiple copies of each datafile during backup. For example, you can use:

CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 2;

This would configure RMAN to create two backup copies of each datafile during backup operations to disk. Having multiple backup copies can be useful for redundancy and disaster recovery purposes.

#Day 68
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see about different types of backup Configurations in rman.

Actually, this configuration is done in the environment as per the business requirements.

The command **CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO** 1; in Oracle RMAN (Recovery Manager) is used to configure the number of backup copies to be created when performing archive log backups to disk. In the provided command, it sets the number of backup copies to 1, which is the default value.

Here's what happens in detail when you execute this command:

Configuration Setting: The command sets a configuration parameter within RMAN. This parameter affects how RMAN performs archive log backups to disk for subsequent operations.

ARCHIVELOG BACKUP: Archive log backups are a crucial part of Oracle database maintenance. These backups capture archived redo logs, which contain changes made to the database, and are essential for recovery scenarios.

COPIES: The COPIES parameter specifies how many identical copies of the archive log backup should be created. In this case, it sets the number of copies to 1, which means that RMAN will create one backup copy.

FOR DEVICE TYPE DISK: This part of the command specifies that the configuration applies to archive log backups written to disk. Oracle RMAN allows you to configure different settings for different backup devices, such as disk or tape.

TO 1: This part of the command specifies the number of copies to be created, which is 1 in this case. If you wanted to create more copies, you could replace 1 with the desired number.

Here are some key points to consider:

Creating multiple backup copies can provide redundancy and additional security. If one copy becomes unusable, you still have other copies for recovery purposes.

Increasing the number of backup copies can increase storage requirements since each copy consumes disk space.

The default value for the number of backup copies is 1, so if you don't specify this configuration, RMAN will create a single backup copy by default.

You can change this configuration later if needed by issuing a similar CONFIGURE command with a different number of copies.

Overall, the command you provided is used to configure the number of backup copies for archive log backups to disk, and in your case, it sets it to the default value of 1.

#Day 69
#ORACLE
#RECOVERYMANAGER(RMAN)
#RECOVERYMANAGERCONFIGURATION
#SERVERMANAGEDBACKUP

Today let us see particularly what about CONFIGURE MAXSETSIZE TO UNLIMITED.

In Oracle Recovery Manager (RMAN), the CONFIGURE MAXSETSIZE TO UNLIMITED command is used to specify the maximum size of backup sets that can be created during backup operations. Let's break down its usage in more detail and why you might use it:

CONFIGURE Command: The CONFIGURE command in RMAN is used to set various configuration

parameters that control the behavior of RMAN operations. These parameters can be set at different levels, including the global level (for all backups) or at the specific backup level (for a particular backup operation).

MAXSETSIZE: This parameter specifies the maximum size for each backup set created during a backup operation. A backup set is a collection of one or more backup pieces, and these sets are used to group related data together for efficient storage and management.

TO UNLIMITED: When you set MAXSETSIZE to "UNLIMITED," you are essentially telling RMAN that there should be no maximum size limit imposed on backup sets. In other words, RMAN will create backup sets without any size restrictions.

Now, let's discuss why you might want to use CONFIGURE MAXSETSIZE TO UNLIMITED:

Large Databases: If you have a large Oracle database with a significant amount of data to back up, imposing a strict size limit on backup sets can be impractical. Using UNLIMITED allows RMAN to create backup sets as large as needed to efficiently back up your data.

Avoiding Manual Intervention: Without setting MAXSETSIZE to "UNLIMITED," you would need to carefully manage and monitor backup set sizes to ensure they don't exceed the specified limit. This can be time-consuming and may require manual intervention during backup operations.

Optimizing Performance: Allowing RMAN to create larger backup sets can lead to better backup performance, as it reduces the overhead of managing multiple smaller sets. This can result in faster backup and restore times.

Simplified Backup Configuration: Setting MAXSETSIZE to "UNLIMITED" simplifies your RMAN backup configuration. You don't need to specify a size limit for each backup operation, making your backup scripts less complex.

However, it's essential to use caution when setting **MAXSETSIZE to** "**UNLIMITED**." While it can be beneficial for large databases, it can also lead to excessively large backup sets that may be challenging to manage and restore efficiently. It's essential to strike a balance between performance and manageability based on your specific database and backup requirements.

In summary, **CONFIGURE MAXSETSIZE TO UNLIMITED** is used to instruct RMAN not to impose a maximum size limit on backup sets during backup operations, which can be useful for simplifying backup configurations and optimizing performance for large Oracle databases.

#Day 70
#RMAN

Today let us see about different types of backup Configurations in rman.
Actually, this configuration is done in the environment as per the business requirements.

CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
In Oracle Database, the CONFIGURE ENCRYPTION FOR DATABASE OFF command is used with Oracle Recovery Manager (RMAN) to disable encryption for database backups. This command is typically used when you want to ensure that your database backups are not encrypted. I'll explain this command in detail:

Oracle Recovery Manager (**RMAN**): RMAN is a utility provided by Oracle to manage and perform backup and recovery operations on Oracle databases. It allows you to create backups of your database for disaster recovery purposes and to ensure data integrity.

ENCRYPTION FOR DATABASE: This part of the command specifies that you are configuring encryption settings for the entire database. When encryption is enabled, RMAN will encrypt the data in backup sets and image copies created by RMAN.

OFF: This keyword indicates that you are turning off encryption for database backups. When you run CONFIGURE ENCRYPTION FOR DATABASE OFF, you are disabling encryption for future RMAN backup operations.

Here's a step-by-step explanation of what happens when you execute this command:

Configuration Setting: When you run CONFIGURE ENCRYPTION FOR DATABASE OFF, you are changing the RMAN configuration setting for encryption. This means that any subsequent backup operations initiated by RMAN will not use encryption for the database backups.

Impact on Backups: Backups taken after you execute this command will not be encrypted. This can be useful if you have previously configured RMAN to use encryption for backups but want to disable it for some reason, such as performance considerations or because you have other security measures in place.

Existing Backups: It's important to note that this command does not affect existing backups. If you have previously created encrypted backups, those backups will remain encrypted unless you re-create them without encryption.

Security Considerations: Disabling encryption for database backups may have security implications, especially if your organization has specific compliance requirements or security policies that mandate encryption for backup data. Make sure to consider the security needs of your environment before using this command.

In summary, CONFIGURE ENCRYPTION FOR DATABASE OFF is an RMAN command that disables encryption for future database backup operations. It's a configuration setting that affects the way RMAN creates backups, and it doesn't affect existing backups. Always consider your organization's security and compliance requirements when configuring encryption for database backups in Oracle.