# Oracle Database Document training

Quá trình họp tập đào tạo để quản trị Oracle Database sẽ phải trải qua 04 giai đoạn như sau:

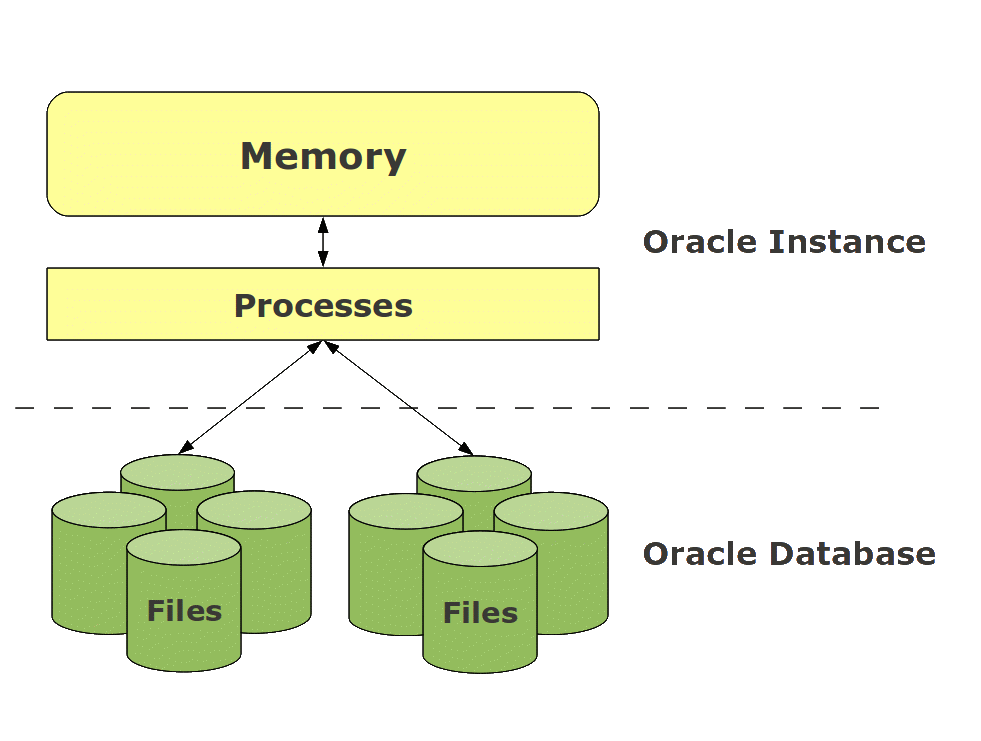
1. SQL: Nắm vững ngôn ngữ SQL;
2. Oracle Administration Workshop I.rar: Là nội dung cơ bản về kiến trúc, các thành phần Oracle; Cài đặt Single và các thao tác Quản trị database cơ bản;
3. Oracle Administration Workshop II.rar: Là nội dung liên quan đến các Task quản trị nâng cao: như backup, restore, performance, moving data ….
4. Oracle RAC Administration 11g.rar: Nội dung nâng cao liên quan đến quản trị Oracle Cluster RAC;

* Can bộ sẽ đọc phần bên dưới của tài liệu này sau đó đọc lần lượt và thực hành các bài trong các tài liệu 2,3,4; Còn về SQL thì mọi người tự học

# Oracle Basics 1 – Oracle Database vs Oracle Instance

While beginning to work with Oracle, one of the most common confusions is between Oracle Database and Oracle Instance. The term Oracle Database is mainly used to designate the whole Oracle RDBMS architecture. However, taking a closer look at Oracle RDBMS Server, we can see that it is composed of two entities, the Database and the Instance. What is really the difference between them? Let’s find out.

Starting to work with Oracle is also facing quite a lot of new concepts. Most of the people beginning with Oracle have the same questions. So did I some years ago :roll:. Therefore I decided to write some short articles, aiming to answer some basic questions in a short and simple way.

For this first part, let’s come back to our Oracle Database and Instance. As a picture is worth a thousand words, take a look at the diagram below.  
  
As we can see above, the Oracle RDBMS architecture spreads across 2 places: files and memory (including processes). That’s where we find the separation between the Oracle Instance and the Oracle Database.

The Oracle Instance includes the memory part and the processes of the RDBMS Server, while the Oracle Database includes all the physical files belonging to the server  (more precisely on the storage system).

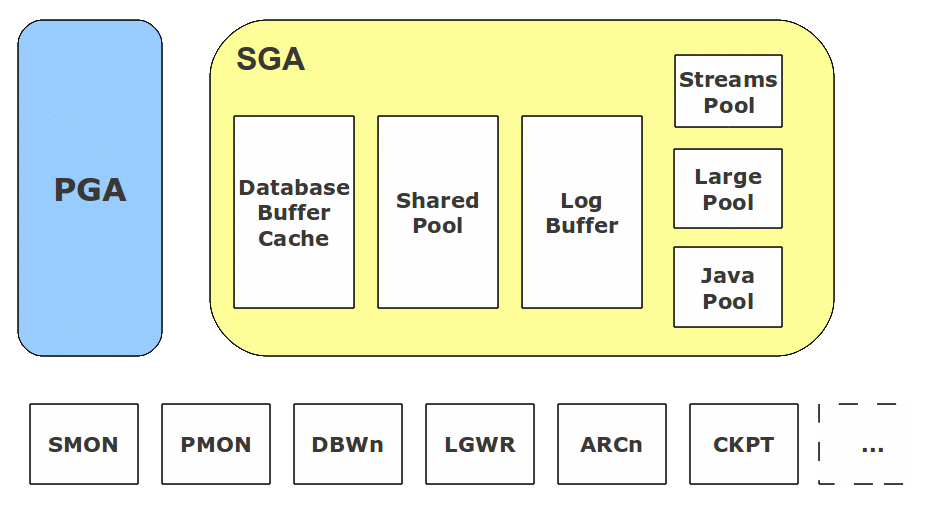
**What exactely is the Instance?**

Now that we have seen the difference between the Instance and the Database, let’s have a look at what exactly the Oracle Instance is.

While working with an Oracle RDBMS Server, the instance will provide the interface between the user and the data he wants to manipulate. To do so, the Instance provides processes for client communication as well as for data access. However, these processes, also called background processes, would not be enough to provide an efficient service and the RDBMS functionalities such as the ACID (Atomicity, Consistency, Isolation, and Durability) principle. Therefore, the instance is also composed of several memory buffers and caches.

Basically an instance can be divided in 3 part:

* System Global Area (SGA)
* Program Global Area (PGA)
* Background processes



**SGA structure**

The SGA is a shared memory structure allocated at instance startup and released on shutdown. At a minimum, the SGA will contain the 3 following data structures:

* Database Buffer Cache

The database buffer cache is Oracle’s working area. When manipulating data (update, insert…) Oracle usually does not access and modify the data on disk. The data blocks are first copied in the database buffer cache and can then be accessed and/or modified in it.

* Redo Log Buffer

The log buffer is a short term staging area for the change vectors before they are written in the Redo Logs (see database part). A change vector is a modification applied to data. All Redo information are written to disk when a session commits its changes (and under some other conditions). The DML (Data Manipulation Language) statements (insert, update, delete) of all sessions are kept in the Redo Log Buffer to avoid I/O contention and sessions to wait for accessing the Redo Log files (concept of “piggy backing”).

* Shared Pool

The shared pool is the most complicated SGA structure. It is divided in several substructures, which are internally managed by the Oracle Server. Basically, the shared pool’s structures contain information like recently executed SQL or recently used objects structures.

Beside the 3 mandatory pools, the SGA can also contain some optional structures:

* Stream Pool

The Stream Pool is reserved for the Oracle Stream functions. More information about Oracle Streams can be found [here](https://download.oracle.com/docs/cd/B19306_01/server.102/b14220/integrat.htm#sthref3334).

* Java Pool

The Java Pool is only required if the application using the database is going to use Java stored procedures within the database.

* Large Pool

The Large Pool is an optional area, which will automatically be used by several processes which would otherwise take memory from the shared pool. The Large Pool will be used for large operations like shared server processes or RMAN operations.

**PGA structure**

Unlike the SGA which is available for all processes, the PGA is a piece of private memory dedicated to one session. Note that each background process has its own PGA too.  
The PGA is allocated at session creation and released once the session is killed.

**Background processes**

In addition to the memory structure, the instance has also several processes with dedicated usage. Oracle has a long list of background processes, but the main ones are the following:

* SMON

SMON is mainly in charge of mounting and opening the database. It mounts the database by locating and validating the control files and opens the database by doing so with the data files and redo logs. In addition, SMON is also in charge of several housekeeping tasks like liberating free space in the data files. If the instance crashed during the last utilization, the SMON is charged to recover the database (rollforward and rollback process).

* PMON

A user session is composed of a user process, on client side, and a server process on server side. PMON is in charge of monitoring all server processes. If it detects an unexpectedly terminated session, it will take care of killing the server process, liberating the PGA memory and rolling the pending transaction back.

* DBWn

Database Writer process manages the write access to the datafiles. It will write down the data blocks from the database buffer cache to disk. Depending on the load of the database, several DBWn processes can run in parallel.

* LGWR

As DBWn, which is in charge of the I/O management with the datafiles, the LGWR manages them for the online redo logs. It streams the content of the log buffer to the online redo logs.

* ARCn

The Archiver process copies the redo logs to a specified storage after each log switch as archived logfile. An instance can have up to 10 archiver processes. The archiver processes exists only if the database is in archive log mode.

* CKPT

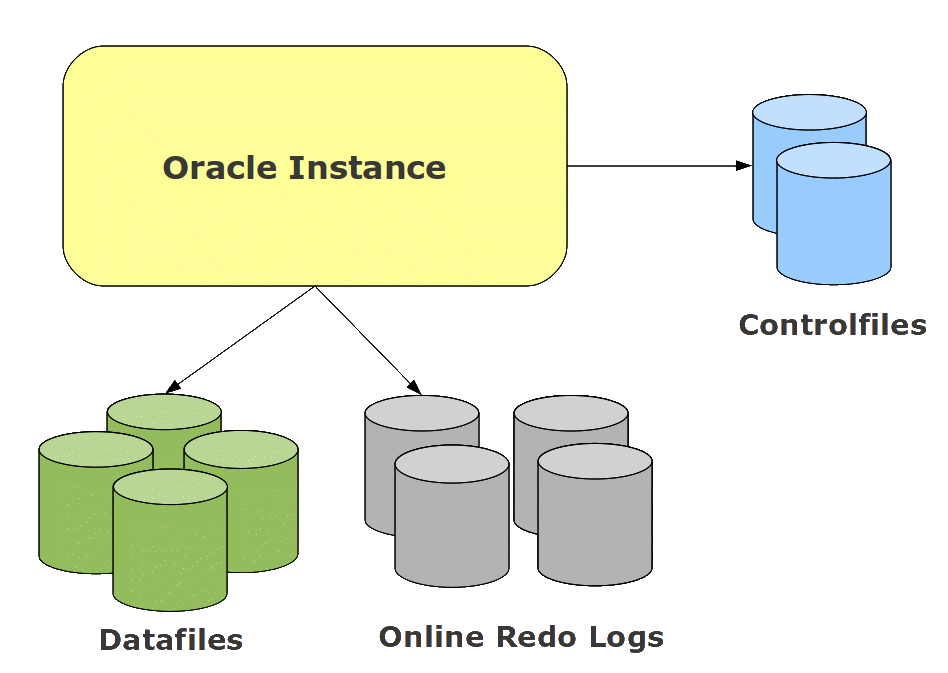
On a regular basis, the modified blocks are written on the datafiles by DBWn. These events are called checkpoints. The checkpoint process is responsible of instructing the DBWn at checkpoint and update the controlfiles and datafile headers with the most recent checkpoint.

**What exactely is the Database?**

The Oracle Database is the storage part of the RDBMS server. One differentiates between the physical and the logical storage.

The database is composed of 3 file types:

* controlfile
* online redo log
* datafile



**Controlfiles**

The controlfile is small but vital! It points to all other database elements like datafiles and online redo logs. It also contains iinformation to maintain the database integrity (System Change Number – SCN, timestamps,…) and all backup and recovery information, like latest archivelogs and database backups. For database security, the controlfile must be multiplexed. It can handle up to 8 copies.

**Online Redo Logs**

The online redo logs store all the change vectors that run on the database in a chronological chain. This information is necessary to reconstruct (=redo) all modifications done. It will be used in case of datafile or complete database recovery. An Oracle database runs at least with 2 online redo logs, but as for the controlfile, it is worth to multiplex them. The online redo logs are arranged in groups, of which the LGWR writes on circular basis. dbi services advices to use at least 3 groups with 2 members each. The point where the LGWR move from one group to another is called Log Switch. If the database is in archivelog mode, the current redo log is archived by ARCn processes at each log switch.

**Datafiles**

The datafiles are the repository of the data. An Oracle Database requires at least 2 datafiles, which have to be created at database creation. This is when the link between the logical storage and the physical storage is created. Datafiles are physical structures visible by the system administrators. Logically, the user data is stored in tablespaces, organized in segments, typically tables, indexes, and (binary or character) large objects. Datafiles can be renamed, resized, added, moved, or dropped at any time, but remember that some operations requires downtime.

I hope that this first introduction was clear enough. Have fun with Oracle!

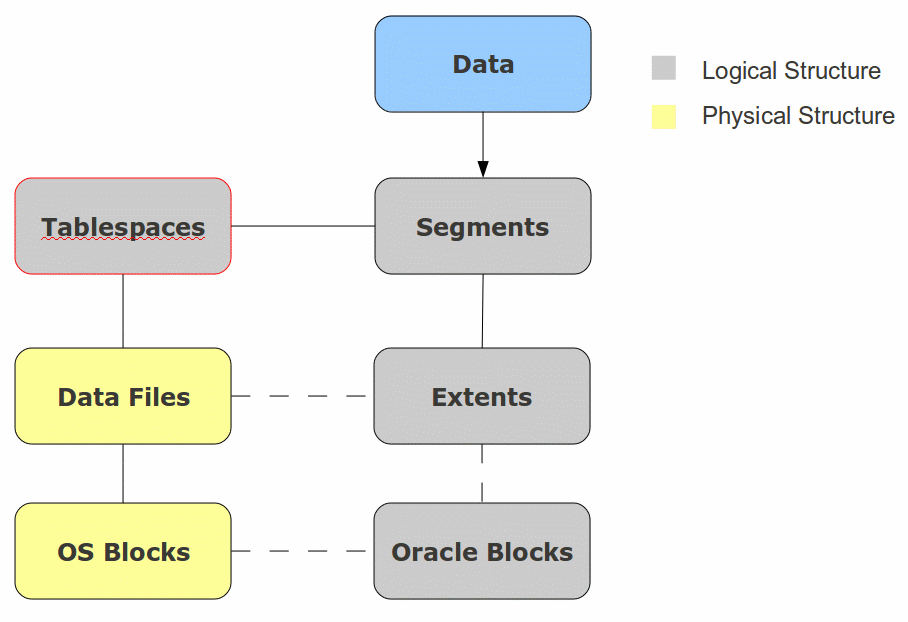
# Oracle Basics 2 – Data Storage

What is finally the primary objective of a database? Storing Data   
The question is, how Oracle makes it. For many beginners, a confusing topic is to get the picture between physical storage and logical one. So how does it work?

The storage in an Oracle database can be seen from 2 points of view:

* Database users
* System Administrators

On one hand you have users who see tables, indexes and so on. This could be summarized has objects or, so called in Oracle, Segments. On the other hand you have system adminitrators (Unix or Windows, doesn’t matter…), who see files and blocks.  
As DBA you are the missing link between these 2 points of view   
So, how do we go down from an Object, Segment, to the files and OS blocks? Take a look down there.



As we already said, data are stored in Segments (Tables, Indexes…). Segments are stored in Tablespaces and these are the bridge between logical storage and physical storage.

On one side, tablespaces are physical collection of data files, which are made of operating system blocks. On the other side, tablespaces are logical collection of Segments (Tables, indexes…). You could almost compare them to folders where you organize your data in.

Segments are composed of Extents, which are pieces of data. As Segments will most probably grow over time, Extents are added to follow the Segments’ size increase. Extents are made of Oracle Blocks.

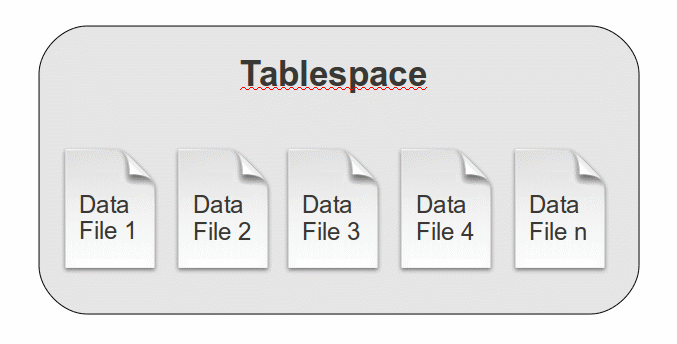
Finally we reach the lowest level, where Oracle Blocks are groups of Operating System Blocks. The Oracle block size is defined on database level but can also be customized per tablespace. The Oracle block size must be greater or equal and a multiple of the OS blocks. This is basis while speaking about Oracle performances. Many best practices can be found on Internet about it.

**Let’s take a small example**:

1 Database with 8KB Oracle Blocks on File Systems of 4KB blocks.  
==> 1 Oracle Block = 2 OS Blocks  
Now we can have a look a little bit deeper in all these elements

**Tablespaces and Data Files**

As seen, Tablespaces are logical containers composed of one or more datafiles.

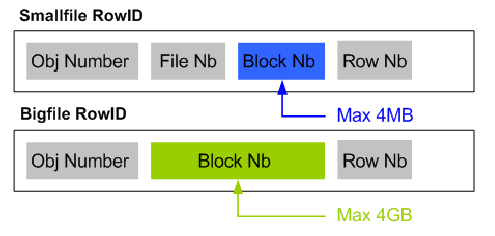


Oracle provides 2 types of tablespaces:

* Smallfile tablespaces – most commonly used
* Bigfile tablespaces

The Smallfile tablespaces are composed of several datafiles, while Bigfiles are composed of a single data file. One of the reasons why Oracle introduced big file tablespaces was the limitation on the number of data file composing a database (usually 64k). In Bigfile tablespaces, the data files max size limit is much higher than for small file tablespaces.

How does it work? You may have already heard about ROWID, which is a unique ID for each row in a database. In fact the ROWID is more than just an ID, it is also the pointer to the row location. Between Smallfiles and Bigfiles tablespaces, the ROWID is slightly different.



As by definition Bigfile tablespaces are made of a single file, there is no File Nb needed within the ROWID. Therefore the Block Nb is coded on 4GB instead of 4kB.

If we take our example of a 8kB database again, we have the following:  
Smallfile max size: 4kB blocks X 8kB (size of each block) = 32 GB per file  
Bigfile max size: 4GB blocks X 8kB (size of each block) = 32 TB per file

The tablespace management can be done either in the Oracle Data Dictionary or, since 10g, be Locally Managed. Nowadays almost all tablespaces are configured as Locally Managed TableSpace (LMTS). This means that the space management is done in a bitmap stored in the Segment Header of the tablespace. The LMTS has the advantage to reduce – but not avoid – fragmentation and improve DDL performances.

While creating data files, 2 solutions are possible. You can either create them with fix size which will be allocated at file creation, or you can create them in autoextend mode. Autoextend data files are created with an initial size and will then grow up automatically until the defined max size. If no max size is given at data file creation, then the limit defined by the Oracle Block Size (see above about ROWID and max data file size) will be applied.

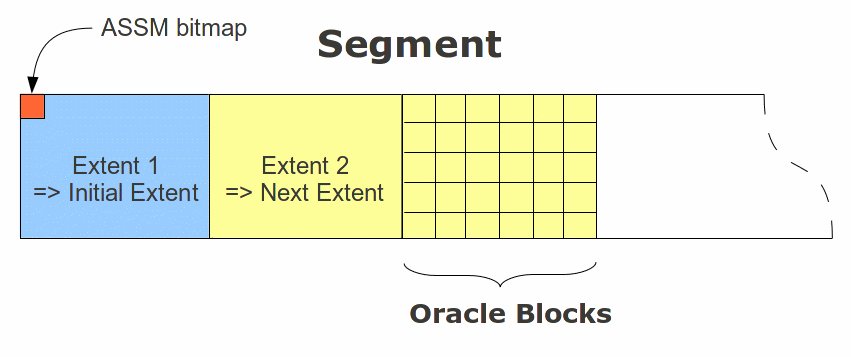
Here are the 2 main views in oracle to get information on tablespaces and data files: dba\_tablespaces, dba\_data\_files

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | SQL> desc dba\_tablespaces;  Name                       Null?         Type  -------------------------- ------------- ------------------------------------  TABLESPACE\_NAME            NOT NULL      VARCHAR2(30)  BLOCK\_SIZE                 NOT NULL      NUMBER  INITIAL\_EXTENT                           NUMBER  NEXT\_EXTENT                              NUMBER  MIN\_EXTENTS                NOT NULL      NUMBER  MAX\_EXTENTS                              NUMBER  MAX\_SIZE                                 NUMBER  PCT\_INCREASE                             NUMBER  MIN\_EXTLEN                               NUMBER  STATUS                                   VARCHAR2(9)  CONTENTS                                 VARCHAR2(9)  LOGGING                                  VARCHAR2(9)  FORCE\_LOGGING                            VARCHAR2(3)  EXTENT\_MANAGEMENT                        VARCHAR2(10)  ALLOCATION\_TYPE                          VARCHAR2(9)  PLUGGED\_IN                               VARCHAR2(3)  SEGMENT\_SPACE\_MANAGEMENT                 VARCHAR2(6)  DEF\_TAB\_COMPRESSION                      VARCHAR2(8)  RETENTION                                VARCHAR2(11)  BIGFILE                                  VARCHAR2(3)  PREDICATE\_EVALUATION                     VARCHAR2(7)  ENCRYPTED                                VARCHAR2(3)  COMPRESS\_FOR                             VARCHAR2(12)  SQL> desc dba\_data\_files;   Name                       Null?    Type   -------------------------- -------- -------------------------------   FILE\_NAME                           VARCHAR2(513)   FILE\_ID                             NUMBER   TABLESPACE\_NAME                     VARCHAR2(30)   BYTES                               NUMBER   BLOCKS                              NUMBER   STATUS                              VARCHAR2(9)   RELATIVE\_FNO                        NUMBER   AUTOEXTENSIBLE                      VARCHAR2(3)   MAXBYTES                            NUMBER   MAXBLOCKS                           NUMBER   INCREMENT\_BY                        NUMBER   USER\_BYTES                          NUMBER   USER\_BLOCKS                         NUMBER   ONLINE\_STATUS                       VARCHAR2(7) |

**Segments and Extents**

Segment is the generic name used in Oracle databases to represent objects like tables, indexes or partitions. These are stored in Data Files in pieces called Extents.

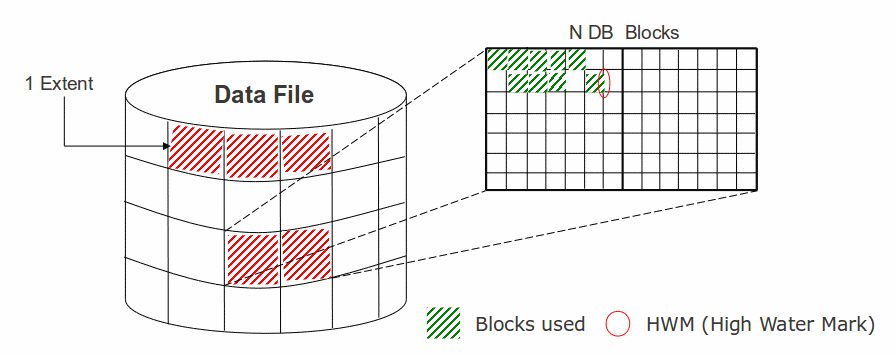
The Segments can be either in MANUAL mode or in AUTO mode – Automatic Segment Space Management (ASSM)  
In earlier Oracle version, the MANUAL mode managed free blocks and free space in a Free List stored the Data Dictionary, which overloaded System tablespace. Since Oracle 10g and ASSM the free blocks and the free space are managed in a bitmap in the Segment Header of each Tablespace.



The 2 main views to find segments and extents information are: dba\_segments and dba\_extents

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | SQL> desc dba\_segments  Name                       Null?           Type  -------------------------- --------------- -----------------------------------  OWNER                                      VARCHAR2(30)  SEGMENT\_NAME                               VARCHAR2(81)  PARTITION\_NAME                             VARCHAR2(30)  SEGMENT\_TYPE                               VARCHAR2(18)  SEGMENT\_SUBTYPE                            VARCHAR2(10)  TABLESPACE\_NAME                            VARCHAR2(30)  HEADER\_FILE                                NUMBER  HEADER\_BLOCK                               NUMBER  BYTES                                      NUMBER  BLOCKS                                     NUMBER  EXTENTS                                    NUMBER  INITIAL\_EXTENT                             NUMBER  NEXT\_EXTENT                                NUMBER  MIN\_EXTENTS                                NUMBER  MAX\_EXTENTS                                NUMBER  MAX\_SIZE                                   NUMBER  RETENTION                                  VARCHAR2(7)  MINRETENTION                               NUMBER  PCT\_INCREASE                               NUMBER  FREELISTS                                  NUMBER  FREELIST\_GROUPS                            NUMBER  RELATIVE\_FNO                               NUMBER  BUFFER\_POOL                                VARCHAR2(7)  FLASH\_CACHE                                VARCHAR2(7)  CELL\_FLASH\_CACHE                           VARCHAR2(7)  SQL> desc dba\_extents  Name                       Null?           Type  -------------------------- --------------- ------------------------------------  OWNER                                      VARCHAR2(30)  SEGMENT\_NAME                               VARCHAR2(81)  PARTITION\_NAME                             VARCHAR2(30)  SEGMENT\_TYPE                               VARCHAR2(18)  TABLESPACE\_NAME                            VARCHAR2(30)  EXTENT\_ID                                  NUMBER  FILE\_ID                                    NUMBER  BLOCK\_ID                                   NUMBER  BYTES                                      NUMBER  BLOCKS                                     NUMBER  RELATIVE\_FNO                               NUMBER |

Another important concept to understand, in case of Table segments, is the “High Watermark” (HWM). It defines the position of the last formated block for the segment. It means that in case of a Full table scan (FTS – i.e. select \* from table1;) Oracle will go through all the segment’s blocks up to the HWM position.



The key here is to keep in mind that deleting data, which means free blocks, does NOT move the HWM position. This can have a quite huge impact on requests performances.  
The only ways to move down the HWM is either to truncate the table, but this also means losing all data, or to make a table reorganization. However this quite beyond our current topic

So here we are with this introduction into Oracle data storage.  
Have fun with Oracle

**Oracle Basics** 3 **– Database startup and shutdown**

Startup and shutdown processes within oracle database servers are split in different steps (for startup) and modes (for shutdown) and even if these principles are quite trivial, some confusions between these steps and/or modes can easily be done. So I’m going to give a picture of both of these processes and their variations.

Before shutting any database server down, it needs first to be started, so let’s begin with the startup process…

Note that to avoid confusion, I use the term database server to designate the entity composed from both the Database and the Instance.

**Starting a database server: a 3 steps process**

Before understanding the process itself, the first question is “How to start a database server?” The answer is that you need to connect locally to it and to be a SYSDBA!  
So then the next question is “How to connect to something which is NOT running yet?” To do so you need 3 environment variables:

ORACLE\_HOME –> Gives the Oracle Binaries Path  
ORACLE\_SID  –> Gives the Instance SID to connect to  
PATH –> Must contain $ORACLE\_HOME/bin or %ORACLE\_HOME%bin (Windows)

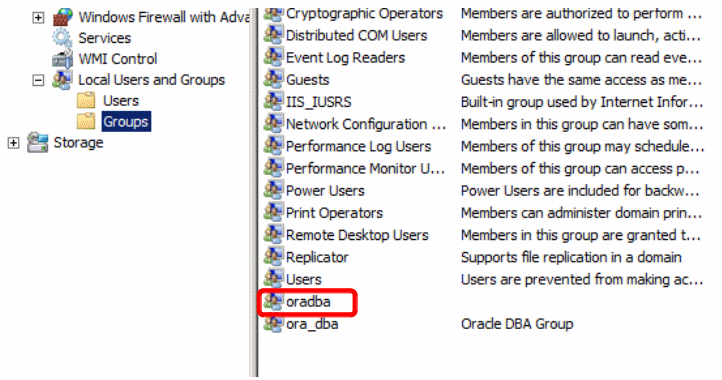
|  |  |
| --- | --- |
| 1  2  3  4 | oracle@vmoratest1:/home/oracle/ [TYRION] echo $ORACLE\_HOME  /u00/app/oracle/product/11.2.0/db\_1  oracle@vmoratest1:/home/oracle/ [TYRION] echo $ORACLE\_SID  TYRION |

Then you have to be on the server and be part of the DBA Operating System group. In UNIX world this corresponds to the group “dba”, while in Windows it is “oradba”.

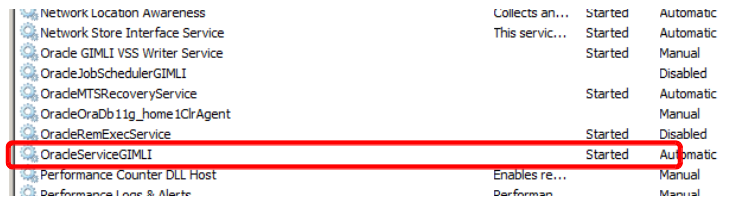
**Unix**:

|  |  |
| --- | --- |
| 1  2 | oracle@vmoratest1:/home/oracle/ [TYRION] id  uid=500(oracle) gid=501(dba) groups=501(dba),502(oinstall) |

**Windows**:



In Windows environment, you need also that the Instance Service is started.



Finally, start SQLPLUS and simply connect as internal: sqlplus / as sysdba    where the / means to connect using OS authentication (DBA group) on the Instance defined by the variable ORACLE\_SID.

|  |  |
| --- | --- |
| 1  2  3  4 | oracle@vmoratest1:/home/oracle/ [TYRION] sqlplus / as sysdba  SQL\*Plus: Release 11.2.0.1.0 Production on Wed May 18 09:02:07 2011  Copyright (c) 1982, 2009, Oracle.  All rights reserved.  Connected to an idle instance. |

Once connected, simply run the command: startup

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <SQL> startup  ORACLE instance started.  Total System Global Area  732352512 bytes  Fixed Size            1339036 bytes  Variable Size          440402276 bytes  Database Buffers      285212672 bytes  Redo Buffers            5398528 bytes  Database mounted.  Database opened.  SQL> select open\_mode from v$database;  OPEN\_MODE  --------------------  READ WRITE |

This is it! you database server is started   
So now let’s have a looking what is going done backstage.

Basically starting an Oracle database is divided in 3 steps:

* NOMOUNT
* MOUNT
* OPEN

For each of them there is a corresponding startup command:

* NOMOUNT => startup nomount
* MOUNT   => startup mount
* OPEN    => startup

As you can see it the startup command automatically takes the database in Open mode. However some operations, like restores and recovers require to decompose the startup process. In this case swithing up from one mode to the next one is done using an “alter database” command.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | SQL> startup nomount  ORACLE instance started.  Total System Global Area  732352512 bytes  Fixed Size            1339036 bytes  Variable Size          440402276 bytes  Database Buffers      285212672 bytes  Redo Buffers            5398528 bytes  SQL> alter database mount;  Database altered.  SQL> alter database open;  Database altered.  SQL> select open\_mode from v$database;  OPEN\_MODE  --------------------  READ WRITE |

Note that you can NOT switch a database one mode/step down. This requires a shutdown!

Here we need to open a small bracket for the databases running on Windows servers.  
As said above, on Windows the service OracleService must be running to connect to the database. However starting this service will fire up a full startup of the database, which means that a healthy database will be fully open once the service is started. On databases requiring restore/recover operations, you will basically also need to start the service and let him failing to open the database before you can go ahead.

In order to really understand the logic behind the startup process, you need to keep in mind the difference between the Oracle Instance and the Oracle Database. You can have a look to my [first article](https://blog.dbi-services.com/oracle-basics-1-oracle-database-vs-oracle-instance) on Oracle Basis to get a refresh on it.

**Step 1 – NOMOUNT**

In the NOMOUNT level, only the Oracle INSTANCE is started. To do so Oracle looks for the Instance spfile or pfile and starts all Processes and Memory Structures.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SQL> startup nomount  ORACLE instance started.  Total System Global Area  732352512 bytes  Fixed Size            1339036 bytes  Variable Size          440402276 bytes  Database Buffers      285212672 bytes  Redo Buffers            5398528 bytes |

Let’s have a look on the running processes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | oracle@vmoratest1:/home/oracle/ [TYRION] ps -ef |grep TYRION  oracle    3999     1  0 09:12 ?        00:00:00 ora\_pmon\_TYRION  oracle    4001     1  0 09:12 ?        00:00:00 ora\_vktm\_TYRION  oracle    4005     1  0 09:12 ?        00:00:00 ora\_gen0\_TYRION  oracle    4007     1  0 09:12 ?        00:00:00 ora\_diag\_TYRION  oracle    4009     1  0 09:12 ?        00:00:00 ora\_dbrm\_TYRION  oracle    4011     1  0 09:12 ?        00:00:00 ora\_psp0\_TYRION  oracle    4013     1  0 09:12 ?        00:00:00 ora\_dia0\_TYRION  oracle    4015     1  1 09:12 ?        00:00:00 ora\_mman\_TYRION  oracle    4017     1  0 09:12 ?        00:00:00 ora\_dbw0\_TYRION  oracle    4019     1  0 09:12 ?        00:00:00 ora\_lgwr\_TYRION  oracle    4021     1  0 09:12 ?        00:00:00 ora\_ckpt\_TYRION  oracle    4023     1  0 09:12 ?        00:00:00 ora\_smon\_TYRION  oracle    4025     1  0 09:12 ?        00:00:00 ora\_reco\_TYRION  oracle    4027     1  0 09:12 ?        00:00:00 ora\_mmon\_TYRION  oracle    4029     1  0 09:12 ?        00:00:00 ora\_mmnl\_TYRION  oracle    4031     1  0 09:12 ?        00:00:00 ora\_d000\_TYRION  oracle    4033     1  0 09:12 ?        00:00:00 ora\_s000\_TYRION  oracle    4106  3356  0 09:12 pts/1    00:00:00 grep TYRION |

Here we can recognize processes like SMON, LGWR, DBW or PMON

The question now is, how Oracle find the spfile?  
In fact Oracle goes automatically in the dbs (UNIX) or database (Windows) folder in ORACLE\_HOME and look for a spfile spfile.ora. If there is no spfile Oracle will the search for a pfile, init.ora, and finally take a default spfile if none are present. A good practice, OFA (Oracle Flexible Architecture), is to store the spfile in the admin directory of the database and to use a pfile to point to it.

Here is a small example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | oracle@vmoratest1:/home/oracle/ [TYRION] cd $ORACLE\_HOME/dbs  oracle@vmoratest1:/u00/app/oracle/product/11.2.0/db\_1/dbs/ [TYRION] ls  hc\_DBUA0.dat  hc\_TYRION.dat  init.ora  initTYRION.ora  lkULTHAN\_SITE1  orapwTYRION  peshm\_DBUA0\_0  peshm\_ULTHAN\_SITE1\_0  oracle@vmoratest1:/u00/app/oracle/product/11.2.0/db\_1/dbs/ [TYRION] cat initTYRION.ora  SPFILE='/u00/app/oracle/admin/TYRION/pfile/spfileTYRION.ora'  oracle@vmoratest1:/u00/app/oracle/product/11.2.0/db\_1/dbs/ [TYRION] cd /u00/app/oracle/admin/TYRION/pfile/  oracle@vmoratest1:/u00/app/oracle/admin/TYRION/pfile/ [TYRION] ls  spfileTYRION.ora |

On a instance in NOMOUNT mode, it is only possible to access and interact to the spfile information

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | SQL> show parameter name  NAME                             TYPE       VALUE  -------------------------------- ---------- ------------------------------  db\_file\_name\_convert             string  db\_name                          string     ULTHAN  db\_unique\_name                   string     ULTHAN\_SITE1  global\_names                     boolean    FALSE  instance\_name                    string     TYRION  lock\_name\_space                  string  log\_file\_name\_convert            string  service\_names                    string     TYRION.it.dhu-domain.com  SQL> show parameter spfile  NAME                       TYPE       VALUE  ------------------------- ----------- ------------------------------  spfile                     string     /u00/app/oracle/admin/TYRION/pfile/spfileTYRION.ora |

and to access some instance views

|  |  |
| --- | --- |
| 1  2  3  4 | SQL> select INSTANCE\_NAME,HOST\_NAME,STATUS from v$instance;  INSTANCE\_NAME      HOST\_NAME                         STATUS  ----------------- --------------------------------- ------------  TYRION             vmoratest1.it.dhu-domain.com      STARTED |

The startup\_mode column permits to identify the current mode of the database server. Here STARTED means in NOMOUNT.

**Step 2 – MOUNT**

Once taking the database server from NOMOUNT to MOUNT, Oracle access the control files, as declared in the spfile, and mount the database.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SQL> show parameter control\_files  NAME                TYPE          VALUE  ------------------- ------------- ----------------------------------------------------  control\_files       string        /u01/oradata/TYRION/control01TYRION.dbf, /u02/oradata                                    /TYRION, /control02TYRION.dbf,/u03/oradata/TYRION                                    /control03TYRION.dbf  SQL> alter database mount;  Database altered. |

Check the new status

|  |  |
| --- | --- |
| 1  2  3  4  5 | SQL> select INSTANCE\_NAME,HOST\_NAME,STATUS from v$instance;    INSTANCE\_NAME      HOST\_NAME                         STATUS  ----------------- --------------------------------- ------------  TYRION             vmoratest1.it.dhu-domain.com      MOUNTED |

Let’s check the control files usage:

|  |  |
| --- | --- |
| 1  2 | [root@vmoratest1 TYRION]# fuser -a ./\*  ./control01TYRION.dbf:  4017  4019  4021 |

At this point Oracle knows information like the files composing the databases and their location, the backup history, the SCN status aso… SYSDBA users are still the own one, who can access the database, but now they have access to several information stored in v$views (thanks to the control file).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | SQL> select open\_mode from v$database;  OPEN\_MODE  --------------------  MOUNTED  SQL> select name from v$datafile;  NAME  --------------------------------------------------------------------------------  /u01/oradata/TYRION/system01TYRION.dbf  /u01/oradata/TYRION/sysaux01TYRION.dbf  /u01/oradata/TYRION/undotbs01TYRION.dbf  /u01/oradata/TYRION/users01TYRION.dbf  SQL> select member from v$logfile;  MEMBER  --------------------------------------------------------------------------------  /u01/oradata/TYRION/redog1m1TYRION.dbf  /u02/oradata/TYRION/redog1m2TYRION.dbf  /u01/oradata/TYRION/redog2m1TYRION.dbf  /u02/oradata/TYRION/redog2m2TYRION.dbf  /u01/oradata/TYRION/redog3m1TYRION.dbf  /u02/oradata/TYRION/redog3m2TYRION.dbf  6 rows selected. |

**Step 3 – OPEN**

The last step is to open the database, which means accessing the data files and redo logs. Once the database is OPEN, all users can log on again and go ahead with their normal activities.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SQL> alter database open;  Database altered.  SQL> select status,open\_mode from v$instance,v$database;  STATUS         OPEN\_MODE  -------------- --------------------  OPEN           READ WRITE |

**Some variations**

For some maintenance or administration activities, it may be necessary to open the database but to avoid any user to access it. The best solution is then to open the database in RESTRICT mode. This means that only SYSDBA users will be allowed to log in.

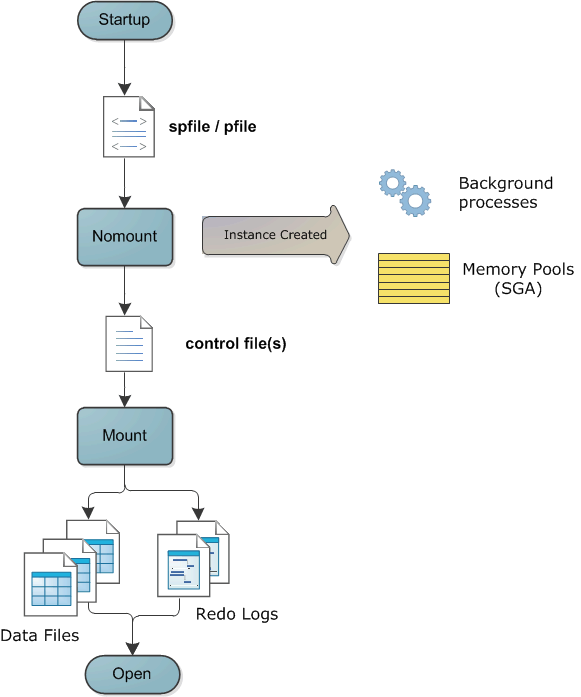
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SQL> startup restrict  ORACLE instance started.  Total System Global Area  732352512 bytes  Fixed Size            1339036 bytes  Variable Size          440402276 bytes  Database Buffers      285212672 bytes  Redo Buffers            5398528 bytes  Database mounted.  Database opened. |

A other possibility is to open the database in READ ONLY instead of READ WRITE.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SQL> alter database open read only;  Database altered.  SQL> select open\_mode from v$database;  OPEN\_MODE  --------------------  READ ONLY |

**Startup process summary**

This drawing summarize the Oracle database startup process.



**Shutting down a database**

Sometimes it is necessary to shutdown a database server, like for example activating the archivelog or flashback mode, changing a static parameter or simply patching it. In this case, the DBA has 4 different solutions:

1. shutdown normal (default mode with shutdown command)   => shutdown
2. shutdown transactional    => shutdown transactional
3. shutdown immediate    => shutdown immediate
4. shutdown abort   => shutdown abort

These modes are going from the safer to the most risky one, but also from the slower to the faster one.

**Shutdown Normal**

This is the default. No new user connections will be allowed, but all current connections continue normaly. Once all users have logged off, the database will finally be allowed to shutdown.

**Shutdown Transactional**

No new user connections are permitted and existing sessions that are not involved in active transactions will be terminated. However sessions currently involved in a transaction are allowed to complete the transaction and will then be terminated. Once all sessions are terminated, the database will shutdown.

**Shutdown Immediate**

No new sessions are permitted, all currently connected sessions are  
terminated an any active transactions are rolled back. Then the database will go down.

**Shutdown Abort**

As far as Oracle is concerned, this is the equivalent of a power failure. The instance terminates immediately (instance “crash”). Nothing is written to disk, no file handles are closed and there is no transactions are terminated, even not in a orderly way. A shutdown abort will not damage the database, but some operations like backups are not advisable after an abort.

**Shutdown process description**

Basically in a “clean” shutdown process (Normal, Transactionnal or Immediate), the process will be the reverse of startup. During an orderly shutdown, the database is first be closed, then dismounted, and finally the instance is stopped.

During the close phase:

* all sessions are terminated
* PMON roll back any incomplete transactions.
* A checkpoint is issued, which forces the DBWn process to write all updated data from the db buffer cache to the datafiles
* LGWR flushes any change vectors still in memory to the logfiles
* The file headers are updated, and the file handles closed.

At this point the database is in a consistent state: all datafiles and logfiles are synchronized.  
During the dismount phase the control files are closed  
Then the instance is stopped by deallocating the SGA memory and terminating the background processes.  
In case of the Abort mode, it leaves the database in an inconsistent state:

* Committed transactions have been lost, because they were only in memory and DBWn had not yet written them to the datafiles
* Uncommitted transactions in the datafiles may not yet have been rolled back.

After a shutdown abort, the SMON process will have to perform an instance recovery at next startup.  
To finish, I will open a last bracket about Windows environment

I said that in Windows there is a service OracleService to start/stop the database. Basically you can stop your database just by stopping the corresponding service. In this case Oracle runs a SHUTDOWN IMMEDIATE by default. The type of shutdown the service runs can be customized, since Oracle 11g, using the parameter -SHUTMODE normal|immediate|abort of oradim.

However there is a small issue with this principle   
Unfortunately in case of a server shutdown/reboot, the service will be stopped before the database shutdown is  performed!

**Alert.log output**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | Fatal NI connect error 12638, connecting to:   (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=beq)))  VERSION INFORMATION:   TNS for 64-bit Windows: Version 11.2.0.2.0 - Production   Oracle Bequeath NT Protocol Adapter for 64-bit Windows: Version 11.2.0.2.0 - Production   Time: 25-MAY-2011 17:40:47   Tracing not turned on.   Tns error struct:   ns main err code: 12638  TNS-12638: Credential retrieval failed   ns secondary err code: 0   nt main err code: 0   nt secondary err code: 0   nt OS err code: 0  This means that the database is not shutdown but crashed! Therefore a instance recovery will be performed at next startup:    Completed: alter database mount exclusive  alter database open  Beginning crash recovery of 1 threads  Started redo scan  Completed redo scan  ...  ... |

A solution is to use the registry entry ORA\_SID\_SHUTDOWN\_TIMEOUT (HKEY\_LOCAL\_MACHINE –> Software –> Oracle –> Key\_) to theoricaly define a timeout while the service waits for the database shutdown confirmation before stopping. If the timeout is reached then it performs a shutdown abort.



And here comes now the Bug 1638610!

The parameter ORA\_SID\_SHUTDOWN\_TIMEOUT isn’t taken in account if the setting SQLNET.AUTHENTICATION\_SERVICES is set in the sqlnet.ora. This bug applies from Oracle 8i to 10g and is still true in Oracle 11gR2. The bug state is set by Oracle as “not feasible to fix”…

I hope that this small article, gave you a good understanding about Oracle startup and shutdown processes