Chapter-8

The database schema:

Table:

A table, which consists of a table name & rows columns of data, is the basic logical storage unit in the oracle database.

Cluster:

A cluster is a set of tables physically stored together as one table that shares a common column. If data in two or more tables is frequently retrieved together based on data in the common column using a clustered table can be quite efficient. Table can be accessed separately even though they are part of clustered table. Because of the structure of the cluster related a data requires much less I/O overhead if accessed simultaneously.

Index:

An index is a structure created to help retrieve data more quickly and efficiently. An index is declared on a column or set of columns.

View:

A view is a window into one or more tables. A view does not store any data. A view can be queried update and deleted as a table without restriction. Views are typically used to simply the user's perception of data access by providing limited information from one table, or a set of information from several transparently.

Sequence:

The oracle sequence generator is used to automatically generate a unique sequence of numbers in cache.

Stored procedure:

A stored procedure is a predefined SQK query that is stored that is stored in the data dictionary. Stored procedures are designed to allow more efficient queries. Using stored procedures, you can reduce the amount of information that must be passed to the RDBMS and thus reduce network traffic and improve performance.

Database trigger:

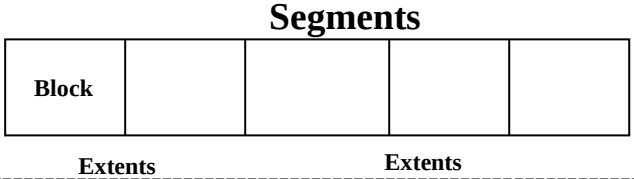
A database trigger is a procedure that is run automatically when an event occurs. This procedure, which is defined, by the administrator or developer triggers or is run whenever this event occurs. This procedure could be an inset a deletion or even a selection of data from a table.

Segment, Extents, data block

Data block: a block is the smallest unit of storage in an oracle DB. The DB block contain header information concerning the block it self as well as the data.

Extents: Extents consist of data blocks.

Segments: A segment is a set of extents used to store a particular type of data.

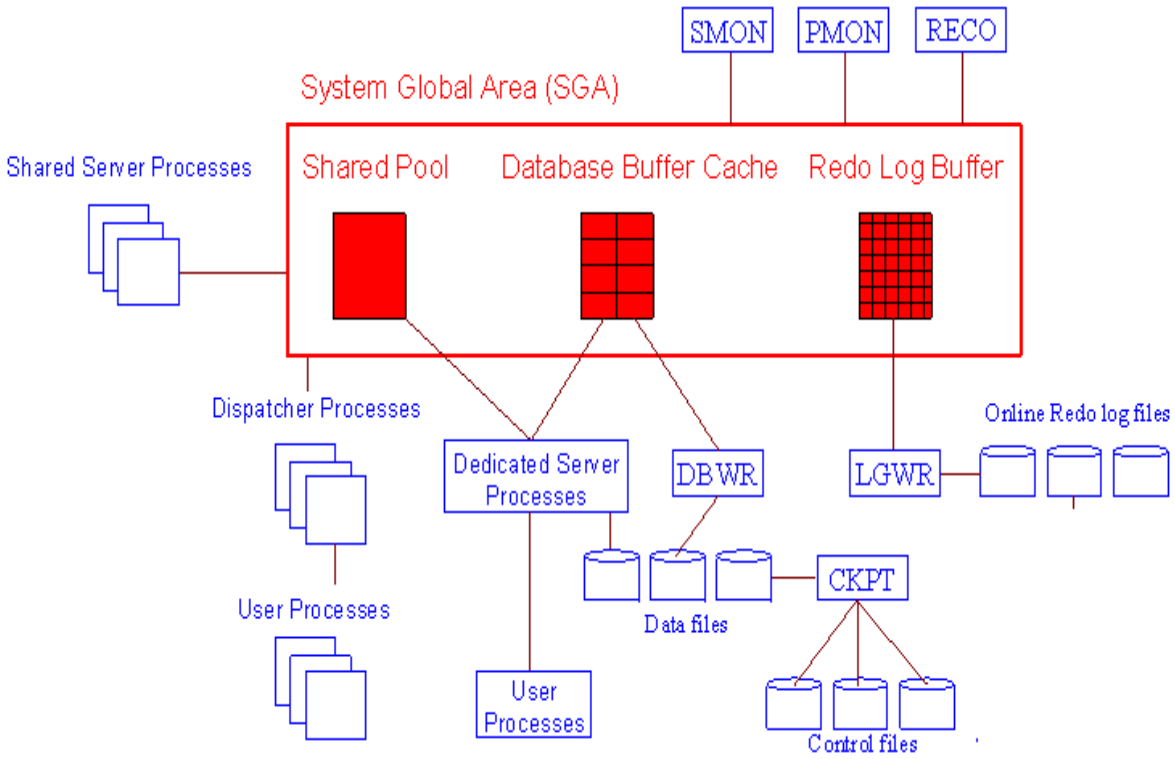


Chapter-8

Oracle Instance:

When a database is started on a database server, Oracle allocates a memory area called the System Global Area (SGA) and starts one or more Oracle processes. This combination of the SGA and the Oracle processes is called an ***Oracle instance*.**

The oracle instance consists of the oracle processes and shared memory necessary to access information in the database. The instance is made up of the user processes, the oracle background processes and the shared memory used by these processes. It handles all the database activities such as transaction processing, database recovery etc.



Each oracle processes use a common shared memory area for processing data concurrently. This memory block is called system global area. Each component of instance is described below.

Oracle Memory Structure:

Oracle uses shared memory for several purposes including caching of data and indexes as well as storing shared program code. This shared memory is broken into various pieces or memory structures. The basic memory structures associated with oracle are the system global area (SGA) and the program global area (PGA).

The System Global Area (SGA):

Chapter-8

The SGA is a shared memory region that oracle uses to store data and control information for one oracle instance. This is the primary memory component of instance. The SGA is allocated when the oracle instance starts and reallocated when the oracle instance shuts down. Each oracle instance that starts has its own SGA. The information in the SGA consists of the following elements each of which has fixed size and is create instance startup. The SGA is shared which means that multiple processes can access and modify the data contained in it.

It includes following things.

- Shared pool
- Database buffer cache
- Redo log buffer

The database buffer cache:

The buffer cache is composed of memory blocks. All data manipulated y Oracle server is first loaded into the buffer cache before being used. All data updates are performed in the buffer blocks

The data movement (swapping and loading) between buffer and disk or other parts of RAM is by least recently Used (LRU) algorithm. The LRU list keeps track of what data blocks are accessed and how often.

Buffer blocks that have been modified are called dirty and are placed on the dirty list. The dirty list keeps track of all data modifications made to the cache data that have not been flushed to disk. When Oracle receives a request to change data, the data change is made to the blocks in the buffer cache and written to the redo log, and then the block is put on the dirty list. Subsequent access to this data reads the new value from the changed data in the buffer cache. Dirty data from the dirty list are written to the disk database under deferred update policy.

The redo log buffer:

The *redo log buffer* is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in *redo entries*. Redo entries contain the information necessary to reconstruct, or redo, changes made to the database by INSERT, UPDATE, DELETE, CREATE, ALTER, or DROP operations. Redo entries are used for database recovery, in the event of system failure. The background process LGWR writes the redo log buffer to the active online redo log file (or group of files) on disk.

The Share pools:

It contains three major parts.

- Library cache
- Dictionary cache

- Server control structure

This is the area of the SGA that stores shared memory structure such as shared SQL areas in the library cache and internal information in the data dictionary. The shared pool is important because insufficient amount of memory allocated to the shared pool can cause performance degradation.

The Library Cache:

The library cache is used to store shared SQL. Here the parse tree and the execution plan for every unique SQL statement area cached. If multiple applications issue the same SQL statement the shared SQL area can be accessed by each to reduce the amount of memory needed and to reduce the processing time used for parsing and execution planning.

The data Dictionary cache:(Row cache)

The data dictionary contains a set of tables and views those oracle users as a reference to the database. Oracle stores information here about the logical and physical structure of the database. The data dictionary contain information such as the following:

- User information such as user privileges
- Integrity constraint defined for tables in the database
- Names and data types of all columns in database tables
- Information on space allocated and used for schema objects
- Stores available free space

The program global area (PGA)

The PGA is a memory area that contains data and control information for the oracle single process (server or background processes). The size and content of the PGA depends on the oracle server option you have installed. Sometimes it is also called process global area. This area consists of the following component:

Stack space:

This is the memory that holds the session's variables, arrays and so on.

Session information:

If you are not running the multithreaded server the session information is stored in the PGA. If you are running the multithreaded server the session information is stored in the SGA.

Private SQL area:

This is an area in the PGA where information, such as binding variables and runtime buffers is kept.

Introduction to processes:

The processes in oracle are described as in two major parts.

- User processes execute the application or Oracle tool code.

- Oracle processes execute the Oracle server code. They include server processes and background processes

User Process:

User, or client processes are the user's connections to the RDBMS system. The user process manipulated the user's input and communicates with the oracle server process through the oracle program interface. The user process is also used to display the information requested by the user and if necessary can process this information into a more useful form. It maintains connection and session of user. A *connection* is a communication pathway between a user process and an Oracle instance. A *session* is a specific connection of a user to an Oracle instance via a user process. User process then gives up it to oracle process.

Oracle Process:

Oracle processes perform functions for users. Oracle processes can be split into two groups: server process and background process.

Server processes (Shadow processes):

Server processes also known as shadow processes, communicate with the user and interact with oracle to carry out the user's request. For example if the user process requests a piece of data not already in the SGA. The shadow process is responsible for reading the data blocks from the data files into the SGA. There can be a one to one correlate between user processes and shadow processes although one shadow process can connect to multiple user processes doing so reduces the utilization of system resources.

Oracle creates *server processes* to handle the requests of user processes connected to the instance. In some situations when the application and Oracle operate on the same machine, it is possible to combine the user process and corresponding server process into a single process to reduce system overhead. However, when the application and Oracle operate on different machines, a user process always communicates with Oracle via a separate server process.

Background Processes

To maximize performance and accommodate many users, a multi process Oracle system uses some additional Oracle processes called background processes. An oracle instance may have many background processes; not all are always present. The background processes in an Oracle instance include the following.

- ***DBWn (Database Writer):***

The database writer process writes the contents of buffers to data files. The DBWn processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one database writer process (DBW0) is adequate for most systems, you can configure additional processes (DBW1

through DBW9) to improve write performance if your system modifies data heavily.

When a buffer in the database buffer cache is modified, it is marked "dirty". The primary job of the DBW*n* process is to keep the buffer cache "clean" by writing dirty buffers to disk. As user processes dirtied buffers, the number of free buffers diminishes. If the number of free buffers drops too low, user processes that must read blocks from disk into the cache are not able to find free buffers. DBW*n* manages the buffer cache so that user processes can always find free buffers.

The DBW*n* process writes the least recently used (LRU) buffers to disk. By writing the least recently used dirty buffers to disk, DBW*n* improves the performance of finding free buffers while keeping recently used buffers resident in memory. For example, blocks that are part of frequently accessed small tables or indexes are kept in the cache so that they do not need to be read in again from disk. The LRU algorithm keeps more frequently accessed blocks in the buffer cache so that when a buffer is written to disk, it is unlikely to contain data that may be useful soon

- ***LGWR (Log Writer):***

The log writer process 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. LGWR performs this write when

- 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 DBW*n* process writes modified buffers to disk, if necessary

- ***CKPT (checkpoint)***

The CKPT process is responsible for signaling the DBWR process to perform a checkpoint and to update all the data files and control files headers for the database to indicate the most recent checkpoint to the data files by the DBWR. The CKPT process is optional. If the CKPT process is not present, the LGWR assumes these responsibilities.

- ***PMON (Process Monitor):***

PMON is responsible for keeping track of database processes and cleaning up of a process prematurely dies and freeing resources that the user process. PMON is also responsible for restarting any dispatcher processes that might have failed.

- ***SMON (System Monitor):***

SMON performs instance recovery at instance startup this includes

cleaning temporary segments and recovering transactions that have died because of a system crash. The SMON also defrayments the database by coalescing free extents within the database.

- ***RECO (Recovery):***

  RECO is used to clean transactions that were pending in a distributed database. RECO is responsible for committing or rolling back the local portion of the disputed transaction. The *recoverer process (RECO)* is a background process used with the distributed database configuration that automatically resolves failures involving distributed transactions

- ***ARCH (Archiver):***

  ARCH is responsible for copying the online redo log file to archival storage when they become full. ARCH is active only when, the RDBMS is operated in archive log mode. When a system is not operated in archive log mode. It might not be possible to recover after a system failure. It is possible to run in no archive log mode under certain circumstances. But typically should operator in archive log mode.

- ***LCKn (Parallel Sever Lock):***

  Up to 10 LKC processes are used for inter instance locking when the oracle parallel serve option is used.

- ***Dnnn (Dispatcher):***

  The *dispatcher processes* support multi-threaded configuration by allowing user processes to share a limited number of server processes. With the multi-threaded server, fewer shared server processes are required for the same number of users; therefore, the multi-threaded server can support a greater number of users, particularly in client/server environments where the client application and server operate on different machines

Export:

The Oracle export utility is designed to write oracle object definitions and data to an oracle specific binary file. This file is known as the export file. An export file is oracle specific and can be read only by the oracle import utility. The export utility can be used for several different purposes,

- To back up the database
- To move data between databases
- To rebuild a database
- To reorganize a database

The oracle export utility provides straightforward and simple service. Export writes object definitions and table data to an oracle format binary file. This information van be used to transfer data between database on different machines or to supplement the normal backup process.

The export file first contains the object data followed by the related object. The export file also used to reorganized data within the database. The normal

backup process copies an image of the data file. If recovery needed, it can only write back the same image. Because export organizes the data as it is written to the export file importing that data does not necessarily place the data in the exact same place disk, this provides a grant advantage because it can reduce fragmentation and row chaining.

NOTE : Under Windows NT, the export program is called EXP80. Under other operating systems it may be known simple as EXP.

Syntax : EXP80 username/password/ [options ..]

Example : C:\> Exp80

Import:

The oracle import utility has one function to load data that has been been exported into an oracle database. The import utility can read only exported data. If you want to load other data into an oracle database. You must use another utility such as SQL*Loader.

- For backup/recovery
- To move data between
- To rebuild a database
- To reorganize a database

As with the export utility, the import utility features a command dine utility for performing imports. The import utility's command line utility is called IMP80 under Windows NT. Under certain other operating systems, it is known simply as IMP.

IMP80 supports a variety of options, many very similar to the export options. Because of the importance of the export and import commands.

Syntax : IMP80 username/password [Option ..]

Example : C:\> IMP80

Now follow the instruction as on screen.

SQL * Loader:

SQL * Loader is another oracle utility that is used for loading data into an oracle database. Where the import utility is designed to accept data in a specific format. SQL * Loader is designed to be flexible and to accept data in a variety of formats. SQL*Loader accepts two input files types: [1] The actual input data file [2] Loader control file.

The control file is used to specify the format of the data file. The control file is also used to specify such things as the column data types field delimiters and various other data specific information.

Like export and import SQL *Loader can be invoked from within data manger or from the command line, regardless of which method you use, you are still required to specify a control file.

The Loader Control File:

The control file is used to specify information about the data to be loaded.

The format of the control file contains control information and can also contain the data itself.

The control file can contain multiple lines for each statement and is not case sensitive except for character within single or double quotes. The control file can also include comments that are indicated by double hyphens (--).

The basics of the control file involve control statements that tell SQL*Loader the following:

- What operation to perform
- Where to find the input data file
- Where to load the data
- The format o the data
- The data can be fixed length or delimited.

Data files:

Data files can be operating system files or in the case some operating systems, RAW device. Data files stores the information contained in the database. The information for a single table can span many data files or many tables can share a set of data files.

The direct path Loader:

The conventional loading technique uses SQL insert statements to load the data into the database. Each insert goes through all the logic and steps performed in a standard insert statement/ to improve performance another option is available the direct path loader. When you use the direct path loader data is inserted directly into the data files thus bypassing much of the logic involved in the conventional path load. The direct path loader is faster and more efficient but there are a few restrictions on it.[1] You cannot direct path load a clustered table [2] you cannot direct path load a table when active transactions are pending.

Redo Log files:

The redo log files are used to store redo information. Each time data is changed in the database, a log record is written describing the change. With this information the database can be recovered in the event of a system failure. The redo log file is necessary for proper recovery. If this file is lost due to sa disk failure, you will not be able to recover in the event of a system failure therefore you must protect the redo log file against this kind of failure.

How dies the redo log work?

When a commit operation is performed the redo information is written into the redo log buffer. The LGWR process writes the redo log files with the information in the redo log buffer. The commit operation is not completed until the redo log has been written. After that has occurred that transaction is irrevocable and will be recovered in the event of a system failure. The redo log is made up of two or more redo log files or log file groups. A log file group is a set of files

that oracle automatically mirrors. Typically the size of the redo file is based on the capability of the medium that will contain the archive log files.

Control Files:

Control file are used to keep information critical to the operation of the RDBMS. The control file resides on the operating system file system. These files are used in the startup of the instance to identify where the datafile and redo log files are in the system. The loss of control file can be devastating to the operation of the RDBMS. It is always a good idea to have multiple control files on different disk volumes so that a failure does not cause the loss of all the control files. You can add an additional control files after following these steps has created the database:

- Shut down the oracle instance
- Copy the control file to another location on another disk volume
- Edit the parameter file to include the new file name in the control_files parameter

Restart the oracle instance. The control file can also be created using CREATE CONTROLFILE command, and ALTER DATABASE database BACKUP CONTROLFILE command.

Initialization Parameters:

The initialization parameter file is used to establish specific database features each time an Oracle8 instance is started. By changing initialization parameter values, you can specify features such as:

- The amount of memory the database uses
- Whether to archive full online redo log files
- Which control files currently exist for the database?

Where is the Initialization Parameter File Located?

The initialization parameter file is located in INIT%ORACLE_SID%.ORA file located in the \ORAWIN95\DATABASE directory. The computer that starts the instance must have access to the appropriate initialization parameter file. The starter database in Personal Oracle8 uses the initialization parameter file located in \ORAWIN95\DATABASE.,

Initialization Parameter File: Definition

An initialization parameter file is an ASCII text file containing a list of parameters. Every database instance has a corresponding initialization parameter file and ORACLE_SID parameter. To allow initialization parameters to be unique to a particular database, each database normally has its own initialization parameter file.

The Sample Initialization Parameter File: Definition

The sample initialization parameter file is the initialization parameter file (INITORCL.ORA) used by the Starter Database in \ORAWIN95\DATABASE.

Use this file as a model for creating a new Personal Oracle7 database.

Initialization Parameters to Check When Creating a New Database

Check the initialization parameters described in this section carefully if you decide to create a new database; they cannot be modified after database creation.

DB_NAME: Specifies the name of the database to be created. The database name is a string of eight characters or less. You cannot change the name of a database once it has been created.

CHARACTER_SET: Specifies the database NLS character set to use. This parameter can be set only when you create the database.

CONTROL_FILES: Designates the names and locations of all control files to be created and maintained. By default, Personal Oracle8 installs a single control file for the Starter Database, CTL1ORCL.ORA, in \ORAWIN95\DATABASE.

If you create your database with the CREATE DATABASE command, Personal Oracle8 creates the control file, CTL1sid.ORA, where sid is the SID of that database. To reduce the risk of losing the control file due to disk drive failure, use at least two control files, each located on a separate storage device. The size of Personal Oracle8 control files varies according to the complexity of your database structure. The maximum size of a Personal Oracle8 control file is 2500 database blocks.

DB_WRITER_PROCESSES specifies the number of DBW*n* processes

SHARED_POOL_SIZE: The value of this parameter affects the size of the SGA or more appropriately the size of the shared pool within the SGA.

DB_CACHE_SIZE**:** The value of this parameter affects the size of the SGA: It sets the size of the default buffer pool.

LOG_BUFFER: The value of this parameter defines the size of the redo log buffer.

DB_BLOCK_SIZE:Determines the size of a database blocks.

DB_BLOCK_BUFFERS: The value of this parameter affects the size of the SGA, or more precisely, the size of the buffer cache.

TableSpace:

The oracle Tablespace is the lowest logical layer of the oracle data structure. The Tablespace consists of one or more datafiles these can be files on the operations system files system or row devices. The Tablespace is important in that it provides the finest granularity for laying out data across datafiles. After the Tablespace is created. Every oracle database must have at least one

Tablespace. When you create a database the system Tablespace is created by default. This is where the data dictionary is kept. Tablespace can hold any one of four types of segments: Data segments, index segments, rollback segments, temporary segments.

Creating Tablespace:

Creating a Tablespace consists of specifying one or more datafiles and storage parameters. The datafiles can be either file system files or raw devices. The storage parameters specify how the Tablespace is used.

Syntax: Create Tablespace

Datafile file_speicification
[AUTOEXTEND OFF]
or [AUTOEXTEND ON [NEXT number K or M]
[NOLOGGING or LOGGING]
[, file_specification
[AUTOEXTEND OFF]
or [AUTOEXTEND ON [NEXT number K or M]
[MAXSIZE UNLIMITED or MAXZISE number K or M]
[NOLOGGING or LOGGING]]
[MINIMUM EXTENT number K or M]
[DEFAUL STORAGE storage_clause]
[ONLINE or OFFLINE]
[PERAMETER or TEMPORERY]

Modifying Tablespace:

It is often necessary to modify a Tablespace that has already been created. This can be accomplished in several ways, either using Enterprise Manager or Server Manager.

Syntax: Alter Tablespace *Tablespace*

[NOLOGGING or LOGGING]
ADD Datafile file_speicification
[AUTOEXTEND OFF]
or [AUTOEXTEND ON [NEXT number K or M]
[MAXSIZE UNLIMITED or MAXSIZE number K or M]
[AUTOEXTEND OFF]
or [AUTOEXTEND ON [NEXT number K or M]
[MAXSIZE UNLIMITED or MAXZISE number K or M]
[RENAME DATAFILE ‘filename’[,filename]…..
TO ‘filename’[, ‘filename’]….]
[COLALESCE]
[DEFAUL STORAGE storage_clause]
[MINIMUM EXTENT number K or M]
[ONLINE]

[OFFLINE NORMAL or OFFLINE TEMPORARY or OFFLINE IMMEDIATE]
[BEGIN BACKUP or END BACKUP]
[READ ONLY OR READ WRITE]
[PERAMENENT or TEMPORERY]

Tablespace

For management, security, and performance reasons, the database is logically divided into one or more tablespaces that each comprises one or more database files. A database file is always associated with only one tablespace.

A tablespace is a logical division of a database comprising one or more physical database files.

Every Oracle database has a tablespace named SYSTEM that has the very first file of the database allocated to it. The SYSTEM tablespace is the default location of all objects when a database is first created. The simplest database setup is one database file in the SYSTEM tablespace (simple, but not recommended).Typically, you create many tablespaces to partition the different parts of the database. For example, you might have one tablespace for tables, another to hold indexes, and so on, and each of these tablespaces would have one or more database files associated to them.

Tablespaces can be added, dropped, taken offline and online, and associated with additional database files. By adding another file to a tablespace, you increase the size of the tablespace and therefore the database itself.You cannot drop the SYSTEM tablespace; this would destroy the database because the system tables are there. You also cannot take the SYSTEM tablespace offline.

The create tablespace command allows one or more files to be assigned immediately to the tablespace. It also specifies a default space for any tables created without an explicit storage clause mentioned in the create table statement. This is the basic format for the create tablespace command:

CREATE TABLESPACE talbot datafile '/db01/oracle/GBT/talbot.dbf' size 1000k default storage (initial 1M next 1M minextents 1 maxextents 100 pctincrease 0) permanent;

Note: The permanent keyword in the create tablespace command tells oracle that you will be storing permanent objects (such as tables) in the tablespace. If the tablespace is only used for temporary segments, then you can specify the temporary keyword instead. The default value is permanent.

The initial file assigned to this tablespace is included in the command, as well as its size in the bytes, not blocks. The number of bytes is an integer and can be followed by a K (to multiply by 1024- about a thousand) or an M (to be multiply by 1048576- about a million). Default storage sets up the storage that a table

will get if storage is not specified in the create table statement. Here, the initial default extent is 1M bytes (not blocks) and the next (incremental) extent is 1M bytes.

Minextents allows you to set aside additional extents beyond the first at the time a table is created. These additional extents will not necessarily be contiguous (physically adjacent) with the initial extent, or with each other, but the space will at least be reserved.

Maxextents is the limit of additional extents allowed. You can specify maxextents unlimited, in which case there is no limit to the number of extents allowed for the table or index.

pctincrease is a growth factor for extents. When set to a non-zero value, each incremental extent will be the specified percentage larger than the one before it. This has the effect of reducing the number of extents, and noncontiguous space, used by a table that grows large. However, it causes the space allocated to the table to grow exponentially. If the data volume in the table grows at a constant rate, you should set pctincrease to 0.

The default values for storage are operating-system-specific. The minimum and maximum values for each of these options are available in the alphabetical reference under create table and "storage". These options may be changed with the alter tablespace command. The create table command for the LEDGER table looks like this:

Example:

CREATE TABLE Ledger (ActionDate Date, Action varchar2(8), Item varchar2(30), Quantity Number, QuantityType varchar2(10), Rate Number, Amount number(9,2), Person varchar(25)) tablespace TALBOT;

In this form, the Ledger table will inherit the default storage definitions of the TALBOT tablespace. To override these default, the storage clause is used in the create table command:

CREATE TABLE Ledger (ActionDate Date, Action varchar2(8), Item varchar2(30), Quantity Number, QuantityType varchar2(10), Rate Number, Amount number(9,2), Person varchar(25)) tablespace TALBOT
storage (initial 512K next 512K minextents 2 maxextents 50 pctincrease 0) ;

If you use temporary tables, you can create a tablespace dedicated to their storage needs. Use the create temporary tablespace command to support this special type of table.

Example:

To create new tablespace named Trans_tab

```
CREATE TABLESPACE Trans_tab
DATAFILE 'data1' SIZE 50M
DEFAULT STORAGE (
INITIAL 50K
```

NEXT 50K
MINEXTENTS 2
MAXEXTENTS 50
PCTINCREASE 0)
OFFLINE;
To add datafile of tablespace named Trans_tab
ALTER TABLESPACE Trans_tab ADD DATAFILE 'data2' SIZE 1M;
To Enlarge Datafile size
ALTER DATABASE DATAFILE 'data2' RESIZE 100M;

Oracle Blocks :

- ⇒ Oracle "formats" the database files into a number of Oracle blocks when they're first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.
- ⇒ These blocks are usually 1 KB (the default for PC-DOS systems), 2 KB (the default for most UNIX machines), 4 KB (the default for IBM mainframes), or larger. For a 50 MB database file, there would be 25,600 Oracle blocks assuming a block size of 2 KB (50 MB/2 KB).
- ⇒ The block size should be a multiple of the operating system block size. Regardless of the block size, not the entire block is available for holding data; Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow.
- ⇒ These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but you should do this only when the database is first created. When you first create a database, it uses some of the blocks within the first file, and the rest of the blocks are free. In the data dictionary, Oracle maintains a list of the free blocks for each data file in each tablespace.
- ⇒ Each Oracle block is numbered sequentially for each database file starting at the no. 1. Two blocks can have the same block address if they are in different database files.

Formatting Command of SQL PLUS:

The basic SQL PLUS commands are bellow:

REM (Remark):

Which stand for remark. SQL PLUS ignores anything's on a line that begins with their letters, thus allowing you to add comments, and explanations to any start file you create.

Example : Rem name : Sales Register

SET HEADSEP:

The punctuation that follows set headsep. SQL PLUS how you will indicate where you with to break a page title or a column heading that runs longer than

one line. When you first activate SQL PLUS the default headsep character is the broken vertical bar (|).

Example : set headsep !

T TITLE:

Ttitle puts a title at the top of each page of a report. OFF and ON suppress and restore the display of the text without changing its contents. Title by itself displays the current titles options and test or variable. SQL PLUS uses ttitle in the new form if the first word after ttitle is valid option. The valid options are:

COL[UM] n skips directly to position n from the left margin of the current line.

S[KIP] n prints n blank lines. If no n is specified, one blank line is printed. If n is 0, no blank lines are printed and the current position for printing becomes positions 1 of the current line.

LE[FT], CE[NTER] and R[IGHT] left-justify, center and right justify data on the current line. Any text or variables follwong these commands are justified as a group up to the end of the command or a left center right or column.

FORMAT string specifies the format model that will control the format of subsequent text or variables, and follows the same syntax as format in a COLUMN command, such as FORMAT A12 or FORMAT $999,990.99. Date value are printed according to the default format unless a variable has been loaded with a date reformatted by to_char.

Example : SQL > Ttitle center ‘This is SALES REPORT’

BTITLE:

Btitle puts a title at the bottom of each page of a report. OFF and ON suppress and restore the display of the text without changing its contents. Title by itself displays the current btitles options and test or variable. SQL PLUS uses btitle in the new form if the first word after btitle is valid option. The valid options are:

COL[UM] n skips directly to position n from the left margin of the current line.

S[KIP] n prints n blank lines. If no n is specified, one blank line is printed. If n is 0, no blank lines are printed and the current position for printing becomes positions 1 of the current line.

LE[FT], CE[NTER] and R[IGHT] left-justify, center and right justify data on the current line. Any text or variables follwong these commands are justified as a group up to the end of the command or a left center right or column.

FORMAT string specifies the format model that will control the format of subsequent text or variables, and follows the same syntax as format in a

COLUMN command, such as FORMAT A12 or FORMAT $999,990.99. Date value are printed according to the default format unless a variable has been loaded with a date reformatted by to_char.

Example : SQL > btitle center 'SALES REPORT.sql'

REPHEADER:

Repheader puts a header on the first page of a report. OFF and ON suppress and resoter the display of the test without changing its contents. Repheader by itself displays the current repheader option and text or variable.

COL[UM] n skips directly to position n from the left margin of the current line.

S[KIP] n prints n blank lines. If no n is specified, one blank line is printed. If n is 0, no blank lines are printed and the current position for printing becomes positions 1 of the current line.

LE[FT], CE[NTER] and R[IGHT] left-justify, center and right justify data on the current line. Any text or variables follwong these commands are justified as a group up to the end of the command or a left center right or column.

FORMAT string specifies the format model that will control the format of subsequent text or variables, and follows the same syntax as format in a COLUMN command, such as FORMAT A12 or FORMAT $999,990.99.

Example : SQL > repheader center 'This is SALES REPORT'

REPFOOTER:

Repheader puts a footer on the last page of a report. OFF and ON suppress and resoter the display of the test without changing its contents. Repfooter by itself displays the current repfooter option and text or variable.

COL[UM] n skips directly to position n from the left margin of the current line.

S[KIP] n prints n blank lines. If no n is specified, one blank line is printed. If n is 0, no blank lines are printed and the current position for printing becomes positions 1 of the current line.

LE[FT], CE[NTER] and R[IGHT] left-justify, center and right justify data on the current line. Any text or variables follwong these commands are justified as a group up to the end of the command or a left center right or column.

FORMAT string specifies the format model that will control the format of subsequent text or variables, and follows the same syntax as format in a COLUMN command, such as FORMAT A12 or FORMAT $999,990.99.

Example : SQL > repfooter center 'This is SALES REPORT'

SET HEADING:

The set heading on/off command is used to turn off/on the column titles that normally would appear select command.

BREAK ON:

A break occurs when SQL PLUS detects a specified change such as the end of a page or a change in the value of an expression. A break will cause SQL PLUS to perform some action you have specified in the BREAK command, such as SKIP and to print some result from COMPUTE command, such as averages or totals for a column. Only one break command may be in effect at a time. We used to turn on/off the break command.

Example : SQL> Break on JOB Skip 2
Select *from Employee
Order by JOB;

COLUMN:

Column allows you to change the heading and format of any column in a select statement.

Example : SQL > Column Emp_nm Heading ‘Employee

Re labels the column and gives it a new heading. This heading like the report title, breaks onto two lines because it has the headsep character r(|) embedded in it.

COMPUTE SUM:

The totals calculated for each section on the report were produced by the compute sum command. This command always work in conjunction with the **break on** command.

Example: SQL > Break on JOB skip 2
Compute sum of SAL on JOB

You can use a break on command without a compute sum command, such as for organizing your report into sections where no totals are needed.

SET LINESIZE:

The command set linesize governs the maximum number of characters that will appear on a single line. If you put more columns of information in SQL query than will fit into the linesize you have allotted. SQLPLUS also uses linesize to determine where to center the ttitle and where to place the date and page number.

Example : Set Linesize 18

SET PAGESIZE:

The set pagesize command sets the total number of lines SQLPLUS will place on each page, including the ttietle, btitle, column headings and any blank line it prints. Set pagesize is coordinate with set newpage.

Example : SQL> set pagesize 66

SET NEWPAGE:

A batter name for set newpage might have “been set blank line” because what is really does is print blank line before the top line of each page in your report.
Example : SQL >Set newpage 9

SAVE:

If the changes you wish to make to your SQL statement are extensive or you simply wish to work in your own editor save the SQL you have created so far in interactive mode, by writing the SQL to a file SQL to a file, like this:

Example : SQL > save emp.sql

EDITING:

If the changes you wish to make in your sql file then give Ed (Edit) command. This command is use from SQL prompt. In SQL were editing with notepad from window operating system.

Example : SQL > Ed emp.sql

START :

One back in SQLPLUS test your editing work by executing the file you have just edited:

Example: SQL> Start emp.sql

SHOW:

Looking at command settings that follow the set command requires using the word SHOW:

Example : SQL > Show linesize

Linesize 79

****