<A>APPENDIX B

<B>DBA SQL COMMANDS

This appendix lists all of the commands used in this book. Some general guidelines for the commands are:

-- All commands that are standard SQL end with a semi-colon “;” in Oracle

-- For all formatting commands from SQLPlus, such as SET, COLUMN, BREAK, etc. the semi-colon is optional.

-- I have attempted t capitalize all commands, functions, etc. that are a part of standard Oracle SQL, SQLPLUS or PL/SQL in all scripts.

-- I have tried to standardize the commands as wire diagrams because I find the new Oracle document standard of using brackets, curley brackets, parenthesis and such abomidable. Except for short, generalized commands.

-- The ultimate reference is of course the documentation, where I know they goofed, I have tried to tell you, where I goofed, you will have to tell me.

<B>DBA SQL Commands:

<C>Generalized CREATE COMMAND:

```
CREATE object_type object_name
create options,
STORAGE ( storage parameters).
```

STORAGE Clause for Tables, Clusters, Indexes and Default Storage for Tablespaces:
The STORAGE clause specifies how an object uses the space that is allocated to it. Let's look at the format of the STORAGE clause.

```
>----STORAGE (-------------------------------------------------------------------------------------)----
.,
|-- INITIAL integer ---------------------|
|                    |- K --|           |
|                    |- M -|            |
|-- NEXT integer ------------------------|
|                    |- K --|           |
|                    |- M -|              |
 |-- MINEXTENTS integer ---------------|
|-- MAXEXTENTS -----------------------|
|                  |----integer-------|  |
|                   |-UNLIMITED-|
  |--PCTINCREASE integer (0-100) ----|
  |-- OPTIMAL integer --------------------|
 |                   |- K --|              |
 |                    |- M -|            |
  |-- FREELISTS integer ------------------|
 |-- FREELIST GROUPS integer-------|
```

Where:

INITIAL - This is the size in bytes of the initial extent of the object. The default is 10240 bytes. The minimum is 4096. The maximum is 4095 megabytes. All values are rounded to the nearest Oracle Block size.

NEXT - This is the size for the next extent after the INITIAL is used. The default is 10240 bytes, the minimum is 2048, the maximum is 4095 megabytes. This is the value that will be used for each new extent if PCTINCREASE is set to 0.

MINEXTENTS - This is the number of initial extents for the object. Generally, except for rollback segments, it is set to 1. If a large amount of space is required, and there is not enough contiguous space for the table, setting a smaller extent size and specifying several extents may solve the problem.

MAXEXTENTS - This is the largest number of extents allowed the object. This defaults to the max allowed for your blocksize for ORACLE7 and ORACLE8. In addition for ORACLE8 if UNLIMITED is set, there is no upper limit.

PCTINCREASE- This parameter tells Oracle how much to grow each extent after the INITIAL and NEXT extents are used. A specification of 50 will grow each extent after NEXT by 50%, *for each subsequent extent.* This means that for a table created with one initial and a next extent, any further extents will increase in size by 50% over their predecessor. Under ORACLE7 and ORACLE8 this parameter is only applied against the size of the previous extent.

OPTIMAL - This is used only for rollback segments and specifies the value to which a rollback segment will shrink back to after extending.

FREELIST GROUPS - This parameters specifies the number of freelist groups to maintain for a table or index.

FREELISTS - For objects other than tablespaces, specifies the number of freelists for each of the free list groups for the table, index or cluster. The minimum value is 1 and the maximum is block size dependent.

<C>CREATE DATABASE Command:

```
>>--CREATE DATABASE --------------------------------------------------------------->
                           |-- database_name--| |--CONTROLFILE REUSE--|
                v--, ----------------|
>- DATAFILE -- file specs ------------------------------------------------------->
                           |-AUTOEXTEND ---OFF-------------------|
                                      |--ON -------------------|
                                             |--NEXT x K|M--MAXSIZE  x K|M --------|
                                                                          |UNLIMITED-|
                           v----- , -------------------|
>---LOGFILE ------------------- file specifications---------------------->
             |--GROUP n --|
>-------------------------------------------------------------------------------------------------------------><
    |--MAXLOGFILES integer ----------------|  |---- EXCLUSIVE ----|
    |--MAXDATAFILES integer---------------|
    |--MAXINSTANCES integer -----------------|
    |--ARCHIVELOG or NOARCHIVELOG --|
    |--MAXLOGMEMBERS n -------------------|
    |--MAXLOGHISTORY n ---------------------|
    |--CHARACTER SET charset ----------------|
```

|--NATIONAL CHARACTER SET charset-|

Where:

database name **-** is the name of the database, maximum of eight characters long.

file specifications for data files are of the format: 'filename' SIZE integer K or M REUSE

K is for kilobytes, M is for Megabytes

REUSE specifies that if the file already exists, reuse it.

New with later versions of ORACLE7 is the AUTOEXTEND option which is used to allow your datafiles to automatically extend as needed. Be very careful with this command as it can use up a great deal of disk space rather rapidly if a mistake is made during table builds or inserts.

file specifications for log files depend on the operating system.

The MAXLOGFILES, MAXDATAFILES and MAXINSTANCES set hard limits for the

database, these should be set to the maximum you ever expect.
 MAXLOGMEMBERS and MAXLOGHISTORY are hard limits.

For ORACLE7, CHARACTER SET determines the character set that data will stored in, this value is operating and system dependent.

If you need archive logging, set ARCHIVELOG, if you never will need it, set NOARCHIVELOG.

Databases are created in EXCLUSIVE mode. Databases are either EXCLUSIVE or PARALLEL. A database must be altered into PARALLEL mode after creation.

The CHARACTER_SET is for normal data, the NATIONALCHARACTER SET specifies the national character set used to store data in columns specifically defined as NCHAR, NCLOB, or NVARCHAR2. You cannot change the national character set after creating the database. If not specified, the national character set defaults to the database character set.

<C>ALTER DATABASE Command:

```
>>--ALTER DATABASE database name ---------------------------------------------------------->

>--MOUNT-------------------------------------------------------------------------------------><
  |          |--STANDBY DATABASE--|  |--EXCLUSIVE---|                           |
  |                                  |--PARALLEL----|                           |
  |                                                                             |
  |--OPEN-------------------------------------------------------------------------|
  |          |------RESETLOGS--------|                                          |
  |          |------NORESETLOGS----|                                            |
  |                                                                             |
  |--ACTIVATE STANDBY DATABASE---------------------------------------------------|
  |                                                                             |
  |--CONVERT----------------------------------------------------------------------|
  |                              |--------------- ,------------------------------ |          |
  |                              V             v----- ,------|          |          |
  |-ADD LOGFILE THREAD y ---- GROUP n (----filespec----)---SIZE x -------------------- |
  |                                                                             |
 |                              |-------------------------,-------------------------|                |
 |                              Vv---------,--------|                                                |
 |-ADD LOGFILE MEMBER --'filename'---------------TO -- GROUP integer--------------------|
 |                                          |--REUSE--|      |-----('file list')------|              |
 |                       v------------, ------------|                                                |
 |-- DROP LOGFILE ---- GROUP integer -----------------------------------------------------------|
 |                       |  v---------, ------------|  |                                             |
 |                       |--- ('file list')-------------|                                            |
 |                                                                                                   |
 |                                                v----- , -----|                                    |
 |-CLEAR---- ------------------ LOGFILE GROUP-- integer-----------------------------------------|
 |          | UNARCHIVED |                        | v---- , -----| |UNRECOVERABLE DATAFILE| |
 |                                               | ('file list') | |                                 |
 |                                                                                                   |
 |                                    v----- , -----|                                                |
 |-DROP LOGFILE MEMBER----'filename'-------------------------------------------------------------
-|
 |                                                                                                   |
 |                                v----- , -----|         v----- , ------|                           |
 |-RENAME FILE------------'filespec'------TO-----'filespec'---------------------------------------------
|
 |                                                                                                   |
 |- -ARCHIVELOG--- ---------------------------------------------------------------------------------
-|
 | | -NOARCHIVELOG- |                                                                                |
 |                                                                                                   |
 |-RECOVER (recover clause)--------------------------------------------------------------------------------
-|
 |                                                                                                    |
 |-BACKUP CONTROLFILE- -TO -’filename’- ------- ------- --------------------------------- -------
-|
 |                                   |              | -REUSE- |                 |                  |
 |                                   | -TO TRACE------ ---------------------------|                  |
 |                                                   |-RESETLOGS-------|                     |
 |                                                   | -NORESETLOGS- |                     |
 |                                                                                                   |
 |                             v----- , ------|                                                     |
 |-CREATE DATAFILE----------’filename’------------------------ -------------------------------------
-|
 |                          |       v----- , ------|   |                                            |
 |                          | --AS-----filespec---- |                                            |
 |                                                                                                  |
 |-CREATE STANDBY CONTROL FILE AS--’filename’-- ------------------------------------------
|
 |                                                          | -REUSE- |                             |
 |              v----- , -----|                                                                    |
 |-DATAFILE --'filename'-- RESIZE integer- --- -----------------------------------------------------|
 |                         |               | -K-  |                                          | |
 |                         |               | -M- |                                           | |
 |                         | AUTOEXTEND OFF ---------------------------------------------------| |
 |                         |                 | ON --------  -----------------------------------------| |
 |                         |                        | NEXT x --- MAXSIZE - UNLIMITED----| |
 |                         |                               | K |             | integer - --------| |
 |                               |                                      | M |                         | K |    | |
 |                               |                                                                | M |   | |
 |                               |-END BACKUP------------------------------------------- --------------| |
 |                                                                                                     |
 |-ENABLE--- -------------THREAD integer-------------------------------------------------------------
|
 |            | -PUBLIC- |                                                                             |
 |                                                                                                     |
 |-DISABLE THREAD integer-----------------------------------------------------------------------------|
 |                                                                                                     |
 |-RESET COMPATIBILITY--------------------------------------------------------------------------------|
 |                                                                                                     |
 |-SET- -DBLOW  = 'text'- -----------------------------------------------------------------------------|
 |       |-DBHIGH = 'text'-----|                                                                       |
 |       | -DBMAC--- ----------|                                                                       |
 |                    |-ON-----|                                                                       |
 |                    | -OFF-- |                                                                       |
 |                                                                                                     |
 |-RENAME GLOBAL_NAME TO database-- --------- ------------------------------------------------|
                                            | -.domain- |
```

Where

database name - is the maximum of 8 character database name. If it is not specified, the value in the INIT.ORA file will be used.

filespec - is a file specification in the format of:

'filename' SIZE integer K or M REUSE

with:

filename a OS specific full path name

K or M specifies integer as Kilobytes or megabytes.

REUSE specifies to reuse existing file if it exists.

If SIZE isn't specified 500K will be used.

REUSE is optional.

filename - is a full path file name.

MOUNT - Database is available for some DBA functions, but not normal functions. Either exclusive which is default or PARALLEL.

STANDBY DATABASE - With 7.3 and greater operates against a hot-standby database (see Backup chapter)

OPEN - Database is mounted and opened for general use. Either with RESET LOGS (default) or NORESET LOGS (see Backup chapter)

ACTIVATE STANDBY DATABASE - See Backup chapter

ADD LOGFILE THREAD - Adds a thread or redo to a

PARALLEL instance.

ADD LOGFILE MEMBER -- Adds a logfile member to an existing group

CLEAR - reinitializes an online redo log and optionally not archive the redo log. CLEAR LOGFILE is similar to adding and dropping a redo log except that the command may be issued even if there are only two logs for the thread and also may be issued for the current redo log of a closed thread.

CLEAR LOGFILE cannot be used to clear a log needed for media recovery. If it is necessary to clear a log containing redo after the database checkpoint, then incomplete media recovery will be necessary. The current redo log of an open thread can never be cleared. The current log of a closed thread can be cleared by switching logs in the closed thread.

If the CLEAR LOGFILE command is interrupted by a system or instance failure, then the database may hang.
If so, the command must be reissued once the database
is restarted. If the failure occurred because of I/O errors accessing one member of a log group, then that member can be dropped and other members added.

UNARCHIVED must be speciied if you want to reuse a redo log that was not archived. Note that specifying

UNARCHIVED will make backups unusable if the redo log is needed for recovery.

UNRECOVERABLE DATAFILE must be specified if the tablespace has a datafile offline and the unarchived log must be cleared to bring the tablespace online. If so, then the datafile and entire tablespace must be dropped once the CLEAR LOGFILE command completes.

DROP LOGFILE - Drops an existing log group

DROP LOGFILE MEMBER -- Drops and existing log member

RENAME - renames the specified database file

ARCHIVELOG -- NOARCHIVELOG - Turns archive logging on or off.

RECOVER -- Puts database into recovery mode. The form of recovery is specified in the recovery clause. See the chapter on Backup.

BACKUP CONTROLFILE - This can be used in two ways. First, to make a recoverable backup copy of the control file ("TO 'filename'") and second, to make a script to rebuild the control file (" TO TRACE").

CREATE DATAFILE -- creates a new datafile in place of an old one. You can use this option to recreate a datafile that was lost with no backup. The 'filename' must identify a file that was once a part of the database. The filespec specifies the name and size of the new datafile. If you omit the AS clause, ORACLE creates the new file with

the same name and size as the file specified by
'filename'.

CREATE STANDBY CONTROLFILE -- Creates a control file for use with the standby database.

DATAFILE -- Allows you to perform manipulations against the datafiles in the instance such as resizing, turing autoextend on or off and setting backup status.

ENABLE and DISABLE threads - allows the enabling and disabling of redo log threads (only used for parallel databases).

RESET COMPATIBILITY - marks the database to be reset to an earlier version of Oracle7 when the database is next restarted. This will render archived redo logs unusable for recovery.

Note: This option will not work unless you have successfully disabled Oracle7 features that affect backward compatibility.

SET DBLOW|DBHIGH|DBMAC -- Used with Secure Oracle.

RENAME GLOBAL_NAME TO -- changes the global name of the database. A rename will automatically flush the shared pool. It doesn’t change data concerning your global name in remote instances, connect strings or db links.

<C>Startup Database From SQLDBA or SVRMGR:

```
STARTUP  [RESTRICTED] [FORCE] [PFILE=filename]
         [EXCLUSIVE or PARALLEL]
         [MOUNT or OPEN] dbname
         [NOMOUNT]
         [RECOVER]
```

This command starts the instance, opens the database named dbname using the parameter file specified by the filename following the PFILE= clause. This starts up the database in the default, EXCLUSIVE mode.

b. STARTUP RESTRICT OPEN dbname PFILE=filename

This command starts the instance, opens the database named dbname using the parameter file specified by the filename following the PFILE= clause. This starts up the database in the restricted only mode (only users with RESTRICTED SESSION privilege can log in).

c. STARTUP NOMOUNT

This command starts the instance, but leaves the database dismounted and closed. Cannot be used with EXCLUSIVE, MOUNT or OPEN.

d. STARTUP MOUNT

This command starts the instance and mounts the database, but leaves it closed.

e. STARTUP OPEN dbname PARALLEL

This command starts the instance, opens the database and puts the database in PARALLEL mode for multi-instance use in pre-ORACLE8 versions. In ORACLE8 simply setting the initialization parameter PARALLEL_SERVER to TRUE starts the instance in parallel server (shared) mode. PARALLEL is obsolete in ORACLE8. Cannot be used with EXCLUSIVE or NOMOUNT or if the INIT.ORA parameter SINGLE_PROCESS is set to TRUE. The SHARED parameter is also obsolete in ORACLE8.

f. STARTUP OPEN dbname EXCLUSIVE

This command is functionally identical to "a" above. Cannot be specified if PARALLEL or NOMOUNT is also specified in pre-ORACLE8 versions. EXCLUSIVE is obsolete in ORACLE8.

If PARALLEL_SERVE is FALSE the database defaults to EXCLUSIVE.

g. The FORCE parameter can be used with any of the above options to force a shutdown and restart of the database into that mode. This is not normally done and is only used for debugging and testing.

h. The RECOVER option can be used to immediately start recovery of the database on startup if desired.

<C>Database Shutdown From SQLDBA or SVRMGR:

```
SHUTDOWN
            [NORMAL]
            [IMMEDIATE]
            [ABORT]
```

a. No option means SHUTDOWN NORMAL - the database waits for all users to disconnect, prohibits new connects, then closes and dismounts the database, then shuts down the instance.

b. SHUTDOWN IMMEDIATE - cancels current calls like a system interrupt,

and closes and dismounts the database, then shuts down the instance. PMON gracefully shuts down the user processes. No instance recovery is required on startup.

c. SHUTDOWN ABORT - This doesn't wait for anything. It shuts the database down now. Instance recovery will probably be required on startup. You should escalate to this by trying the other shutdowns first.

<C>CREATE ROLLBACK SEGMENT Command:

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback name
      TABLESPACE tablespace name
      STORAGE storage clause;
```

Where:

Rollback name is the name for the rollback segment, this name must be unique.

Tablespace name is the name of the tablespace where the segment is to be created.

Storage clause is a storage clause that specifies the required storage parameters for the rollback segment. It is strongly suggested the following guidelines be used:

INITIAL = NEXT

MINEXTENTS = 2 (Default on CREATE ROLLBACK)

MAXEXTENTS = a calculated maximum based on size of the rollback segment tablespace, size of rollback segments extents and number of rollback segments.

OPTIMAL - This parameter reflects the size that the system will restore the rollback segment to after it has been extended by a large transaction.

When a rollback segment is created it is not online. To be used, it must be brought online using the ALTER ROLLBACK SEGMENT name ONLINE; command, or, the database must be shutdown, the INIT.ORA parameter ROLLBACK_SEGMENTS modified, and the database restarted. In any case, the INIT.ORA file parameter should be altered if the rollback segment is to be used permanently or else it will not be acquired when the database is shutdown and restarted.

<C>ALTER ROLLBACK SEGMENT Command:

```
>---ALTER ROLLBACK SEGMENT rollback_segment------------------------------------------->>
                                            |--ONLINE------------------------|
                                            |--OFFLINE-----------------------|
                                            |-STORAGE storage_clause----|
                                            |-SHRINK -------------------------|
                                                      |--TO integer------|
                                                                |-K-|
                                                                |-M-|
```

Where:

ONLINE brings the rollback segment online

OFFLINE takes the rollbacks segment offline (after any transactions it has are completed)

STORAGE storage clause cannot contain new values for INITIAL, MINEXTENTS or PCTINCREASE (which is not allowed for rollback segments)

<C>DROP ROLLBACK SEGMENT Command:

```
>----------DROP------------------ ROLLBACK SEGMENT rollback name----------------->>
             |-PUBLIC--|
```

A rollback segment must not be in use or online or it cannot be dropped. Once dropped it must be removed from the INIT.ORA - ROLLBACK SEGMENT clause or the database cannot be re-started.

<C>ALTER SYSTEM command format

```
>----ALTER SYSTEM------------------------------------------------------------------------------------
>>
  |------------------------------------------------------------ RESTRICTED SESSION--------------|
  |                |---ENABLE ---|                                                          |
  |                |-- DISABLE---|                                                          |
  |-- FLUSH SHARED_POOL-----------------------------------------------------------------------|
  |-------------------------------------------------------------------------------------------|
  |    |---- CHECKPOINT---------------------------------------------------------------|        |
  |    |                          |--GLOBAL-- |                              |        |
  |    |                          |--LOCAL-----|                             |        |
  |    |                                                                     |        |
  |    |--- CHECK DATAFILES ---------------------------------------------------|       |
  |                               |--GLOBAL-- |                                      |
  |                               |--LOCAL-----|                                     |
  |                                                                                  |
  |----- SET ---------------------------------------------------------------------------|
  |              |----RESOURCE_LIMIT = ---------------------------|                     |
  |              |                          |--TRUE-- |          |                     |
  |              |                          |--FALSE--|          |                     |
  |              |---- GLOBAL_NAMES = ----------------------------|                     |
  |              |                          |--TRUE-- |          |                     |
  |              |                          |--FALSE--|          |                     |
  |                                                                                   |
  |----- SCAN_INSTANCES = integer ----------------------------------------------------|
  |--- CACHE_INSTANCES = integer -----------------------------------------------------|
  |--- MTS_SERVERS = integer ---------------------------------------------------------|
  |--- MTS_DISPATCHERS = 'protocol, integer'------------------------------------------|
  |--- LICENSE_MAX_SESSIONS = integer ------------------------------------------------|
  |--- LICENSE_SESSIONS_WARNING = integer --------------------------------------------|
  |--- LICENSE_MAX_USERS = integer ---------------------------------------------------|
  |--- REMOTE_DEPENDENCIES_MODE = ----------------------------------------------------|
  |                                         |--TIMESTAMP--|                           |
  |                                   |--SIGNATURE--|                  |
  |--- SWITCH LOGFILE -----------------------------------------------------------------------------|
  |--------------------------------------- DISTRIBUTED RECOVERY ------------------------------|
  |    |--ENABLE--|                                                                             |
  |    |--DISABLE-|                                                                             |
  |---- ARCHIVE LOG archive_log_clause ---------------------------------------------------------|
  |--- KILL SESSION 'integer1, integer2'-------------------------------------------------------------|
```

Where:

RESOURCE_LIMIT - This either enables (TRUE) or disables (FALSE) the use of resource limits.

GLOBAL_NAMES - This either enables (TRUE) or disables (FALSE) the use of global names in database links.

MTS_SERVERS - The n specifies the number of shared server process to enable, up to the value of the MAX_SERVERS parameter.

MTS_DISPATCHERS - The protocol specifies the network protocol for the dispatcher(s), the n specifies the number of dispatchers for the specified protocols up to the value of MAX_DISPATCHERS (as a sum of all dispatchers under all protocols).

SWITCH LOGFILE - This switches the active log file groups.

CHECKPOINT - This performs either a GLOBAL (all open instances on the database) or LOCAL (current instance) checkpoint.

CHECK DATAFILES - This verifies access to data files. If GLOBAL is specified, all data files in all instances accessing the database are verified accessible. If LOCAL is specified only the current instance's data files are verified.

ENABLE RESTRICTED SESSION - This only allows users with RESTRICTED SESSION privilege to log in to the database.

DISABLE RESTRICTED SESSION - This allows any user to log on to the instance.

ENABLE RESTRICTED RECOVERY - This enables distributed recovery.

DISABLE RESTRICTED RECOVERY - This disables distributed recovery.

ARCHIVE LOG - Manually archives redo log files or enables or disables automatic archiving depending on the clause specified.

ARCHIVE LOG clauses:

```
THREAD n
 [SEQ n] [TO 'location']
 [CHANGE n] [TO 'location']
 [CURRENT] [TO 'location']
 [GROUP n] [TO 'location']
 [LOGFILE 'filename'] [TO 'location']
 [NEXT] [TO 'location']
 [ALL] [TO 'location']
 [START] [TO 'location']
```

```
                    [STOP]
```

<C>A Detailed look at ARCHIVE LOG clauses

For ORACLE7 the command is removed from SQLDBA and SVRMGR (except for the pull down display) and is placed under the ALTER SYSTEM command (shown above). The new command has additional clauses to handle the more complex archive log scheme under ORACLE7 and ORACLE8. The new syntax handles the threads and groups associated with the new archive logs. The new syntax follows.

<C>ALTER SYSTEM ARCHIVE LOG clause;

```
    ARCHIVE LOG clauses:

        THREAD n
         [SEQ n] [TO 'location']
         [CHANGE n] [TO 'location']
         [CURRENT] [TO 'location']
         [GROUP n] [TO 'location']
         [LOGFILE 'filename'] [TO 'location']
         [NEXT] [TO 'location']
         [ALL] [TO 'location']
         [START] [TO 'location']
         [STOP]

       Where:
```

THREAD - This specifies the specific redo log thread to effect. If this isn't specified then the current instance redo log thread is effected.

SEQ - This archives the redo log group that corresponds to the integer specified by the integer given as the argument.

CHANGE - This corresponds to the SCN (System Change Number) for the transaction you want to archive. It will force archival of the log containing the transaction with the SCN that matches the integer given as the argument to the CHANGE argument.

GROUP - This manually archives the redo logs in the specified group. If both THREAD and GROUP are specified, the group must belong to the specified thread.

CURRENT - This causes all non-archived redo log members of the current group to be archived.

LOGFILE - This manually archives the group that contains the file specified by 'filespec'. If thread is specified the file must be in a group contained in the thread specified.

NEXT - This forces manual archival of the next online redo log that requires it. If no thread is specified Oracle archives the earliest available unarchived redo log.

ALL - This archives all online archive logs that are part of the current thread haven't been archived. If no thread is specified then all unarchived logs from all threads are archived.

START - Starts automatic archiving of redo log file groups. This only applies to thread assigned to the current instance.

TO - This specifies the location to archive the logs to. This must be a full path specification.

STOP - This disables automatic archiving of redo file log groups. This applies to your current instance.

<C>CREATE TABLESPACE Command:

```
>--CREATE TABLESPACE tablespace --------------------------------------------------------------→>
                 v--------------------------------- , -----------------------------------------------------|
 >-DATAFILE- filespec -----------------------------------------------------------------------------→
                             |-AUTOEXTEND-----------------------------------------------------------|
                                        |ON---NEXT integer -- MAXSIZE --integer -------|
                                                      |k |                |k |
                                                      |m|                 |m|
                                        |OFF |                     |UNLIMITED|

>---------------------------------------------------------------------------------------------------------→
   |--LOGGING------|
   |--NOLOGGING--|
>-- MINIMUM EXTENT integer ---------------------------------------------------------------------→
```

```
                                  | K |
                                   | M |
>-------------------------------------------------------------------------------------------------------→
   |--DEFAULT STORAGE storage_clause --|
>-------------------------------------------------------------------------------------------------------→
  | ONLINE   |
  | OFFLINE |

>-------------------------------------------------------------------------------------------------------→
  | PERMANENT |
  | TEMPORARY |
```

Keywords and Parameters

tablespace	is the name of the tablespace to be created.
DATAFILE	specifies the data file or files to comprise the tablespace.
MINIMUM EXTENT integer --	controls free space fragmentation in the tablespace by ensuring that every used and/or free extent size in a tablespace is at least as large as,and is a multiple of, integer.
AUTOEXTEND	enables or disables the automatic extension of datafile.
OFF	disable autoextend if it is turned on. NEXT and MAXSIZE are

set to zero. Values for NEXT and MAXSIZE must be respecified in further ALTER TABLESPACE AUTOEXTEND commands.

ON	enable autoextend.
NEXT	disk space to allocate to the datafile when more extents are required.
MAXSIZE	maximum disk space allowed for allocation to the datafile.
UNLIMITED	set no limit on allocating disk space to the datafile.

LOGGING

NOLOGGING specifies the default logging attributes of all tables, index, and partitions within the tablespace. LOGGING is the default.

If NOLOGGING is specified, no undo and redo logs are generated for operations that support the NOLOGGING option on the tables, index, and partitions within the tablespace.

The tablespace-level logging attribute can be overridden by logging specifications at the table, index, and partition levels.

DEFAULT specifies the default storage parameters for all objects created in the tablespace.

ONLINE makes the tablespace available immediately after creation to users who have been granted access to the tablespace.

OFFLINE makes the tablespace unavailable immediately after creation.

If you omit both the ONLINE and OFFLINE options, Oracle creates the tablespace online by default. The data dictionary view DBA_TABLESPACES indicates whether each tablespace is online or offline.

PERMANENT specifies that the tablespace will be used to hold permanent objects. This is the default.

TEMPORARY specifies that the tablespace will only be used to hold temporary objects. For example, segments used by implicit sorts to handle ORDER BY clauses.

<C>ALTER TABLESPACE Command:

```
>--- ALTER TABLESPACE tablespace ------------------------------------------------------→>
  |                                              |-- LOGGING ------|                        |
  |                                              |-- NOLOGGING --|                       |
  |                        v----------------------- , ------------------------------------------|   |
  |- ADD DATAFILE -- 'filespec' ---------------------------------------------------------------|
                  |-AUTOEXTEND------------------------------------------------------------|
                              |ON---NEXT integer -- MAXSIZE --integer -------|
                                               |k |                    |k |
                                               |m|                     |m|
                              |OFF |                         |UNLIMITED|

>-----------------------------------------------------------------------------------------------------------→
   |--LOGGING------|
   |--NOLOGGING--|
>-- MINIMUM EXTENT integer ------------------------------------------------------------------→
                               | K |
                                | M |
>-------------------------------------------------------------------------------------------------------→
  |--DEFAULT STORAGE storage_clause --|
>-------------------------------------------------------------------------------------------------------→
 | ONLINE   |
 | OFFLINE |

>-------------------------------------------------------------------------------------------------------→
 | PERMANENT |
 | TEMPORARY |
```

Keywords and Parameters

tablespace is the name of the tablespace to be created.

DATAFILE specifies the data file or files to comprise the tablespace.

MINIMUM EXTENT integer -- controls free space fragmentation in the tablespace by ensuring that every used and/or free extent size in a tablespace is at least as large as,and is a multiple of, integer.

AUTOEXTEND enables or disables the automatic extension of datafile.

OFF disable autoextend if it is turned on. NEXT and MAXSIZE are set to zero. Values for NEXT and MAXSIZE must be respecified in further ALTER TABLESPACE AUTOEXTEND commands.

ON enable autoextend.

NEXT disk space to allocate to the datafile when more extents are required.

MAXSIZE maximum disk space allowed for allocation to the datafile.

UNLIMITED set no limit on allocating disk space to the datafile.

LOGGING

NOLOGGING specifies the default logging attributes of all tables, index, and partitions within the tablespace. LOGGING is the default.

If NOLOGGING is specified, no undo and redo logs are generated for

operations that support the NOLOGGING option on the tables,
index, and partitions within the tablespace.

The tablespace-level logging attribute can be overridden by logging specifications at the table, index, and partition levels.

DEFAULT specifies the default storage parameters for all objects created in the tablespace.

ONLINE makes the tablespace available immediately after creation to users who have been granted access to the tablespace.

OFFLINE makes the tablespace unavailable immediately after creation.

If you omit both the ONLINE and OFFLINE options, Oracle creates the tablespace online by default. The data dictionary view
DBA_TABLESPACES indicates whether each tablespace is online or offline.

PERMANENT specifies that the tablespace will be used to hold permanent objects. This
is the default.

TEMPORARY specifies that the tablespace will only be used to hold temporary objects.
For example, segments used by implicit sorts to handle ORDER BY clauses.

<C>DROP TABLESPACE Command:

```
>------DROP TABLESPACE tablespace name -------------------------------------------->>
                                      |--INCLUDING CONTENTS--|
```

The INCLUDING CONTENTS clause is optional, but if it isn't included, the tablespace must be empty of tables or other objects. If it is included, the tablespace will be dropped regardless of its contents unless it contains an online rollback segment. The SYSTEM tablespace cannot be dropped.

NOTE: This doesn't remove the physical data files from the system, you must use operating system level commands to remove the physical files.

<C>Coalesce Of Contiguous Free Space in Early ORACLE7 Releases:

```
alter session set events 'immediate trace name coalesce level ts#';
```

Where ts# is the tablespace number as specifed in the ts$ table.

CREATE TABLE Command:

<C>Relational Table CREATE Command Definition :

```
>---CREATE TABLE ---------------- table ----------------------------------------------------------->>
                 |--schema.--|         |  V------------ , ----------------------------------------|    |
                                       |(--column datatype -------------------------------------)-|
                                                          |--DEFAULT expr-|
                                                          |--WITH ROWID--|
   V----------------------------------------------------------------------------------------------------|
>---SCOPE IS ---------------- scope_table_name--------------------------------------------------------->>
                 |--schema.--|                        |---column_constraint----|
>--- table_constraint ------------------------------------------------------------------------------->>
   |--REF (ref_column_name) WITH ROWID --|
>--SCOPE FOR (ref_column_name) IS  -------------- scope_table_name--------------------------->>
                                              |-schema.-|

>---ORGANIZATION------------------------------------------------------------------------------->>
                              |---HEAP ---| (default value)
                              |--INDEX---| (for index only tables)

>---PCTTHRESHOLD  n ---------------------------------------------------------------------------->>
                              |-INCLUDING -column_name--| |OVERFLOW -------------------------|
                                                                     ______|-phys_attri-| |      |_
                                                                    | TABLESPACE tablespace|
>----------- phys_attrib ---------------------------------------------------------------------------->>
>--TABLESPACE tablespace ------------------------------------------------------------------------>>
>--LOB --lob_item -----STORE AS------------------------------------------------------------------------>>
        ^----,--------|                |                 | |                                              |
                                       |-lob_segname-| |--TABLESPACE tablespace -------------------|
                                                        | STORAGE storage_clause--------------------|
                                                        | CHUNK integer----------------------------------|
                                                        | PCTVERSION integer ------------------------|
                                                        |-- CACHE-----------------------------------------|
                                                        |  |-- NOCACHE LOGGING ---------|          |
                                                        |  |-- NOCACHE NOLOGGING -----|          |
                                                   | INDEX --lob_index_name--------------------|
                                                   |TABLESPACE tablespace--------------------|
                                                            |  STORAGE storage_clause    |
                                                            |  INITRANS integer              |
                                                            |  MAXTRANS integer            |
>--- NESTED TABLE nested_item STORE AS storage_table-----------------------------------------
>
>--- LOGGING --------------------------------------------------------------------------------------------
->
    |- NOLOGGING--|
>--- CLUSTER --cluster ----(---column---)--------------------------------------------------------------
>
                              ^----,------|
>--- PARALLEL parallel_clause ------------------------------------------------------------------------>
  V-----------------------------------------------------------------------------------------------------|
>-PARTITION BY RANGE (column_list) ------------------------------------------------------------
>
                                             |PARTITION--------VALUES LESS THAN (val_list)|
                                             |                   |name|                                     |
                                             |                                                              |
                                             |-----------------------phys_attrib--------------------------|
                                             | ----------------------TABLESPACE tablespace  -------|
                                             |----LOGGING---------------------------------------------|
                                                  | NOLOGGING |


          V-----------------------------------|
>---------- ENABLE enable_clause-------------------------------------------------------------------->>
```

```
            | DISABLE disable_clause |

>------- AS subquery ------------------------------------------------------------------------------------>>
>--------CACHE ------------------------------------------------------------------------------------------><
          | NOCACHE |
```

<C>The Keywords and Parameters for CREATE TABLE Commands

The CREATE TABLE commands have the following parameter definitions in ORACLE8:

schema -- This is the schema to contain the table. If you omit schema, Oracle creates the table in your own schema.

table -- This is the name of the table (or object table) to be created. A partitioned table cannot be a clustered table or an object table.

OF object_type -- This explicitly creates an object table of type object_type. The columns of an object table correspond to the top-level attributes of type object_type. Each row will contain an object instance and each instance will be assigned a unique, system-generated object identifier (OID) when a row is inserted. Object tables cannot be partitioned. Of course, relational tables cannot be object tables.

Column -- This specifies the name of a column of the table. A table can have up to 1000 columns in ORACLE8. You may only omit column definitions when using the AS subquery clause.

Attribute -- specifies the qualified column name of an item in an object.

Datatype -- This is the datatype of a column. You can omit the datatype only if the statement also designates the column as part of a foreign key in a referential integrity constraint. Oracle automatically assigns the column the datatype of the corresponding column of the referenced key of the referential integrity constraint.

Object types, REF object_type, VARRAYs, and nested tables are valid datatypes.

DEFAULT -- specifies a value to be assigned to the column if a subsequent INSERT statement omits a value for the column. The datatype of the expression must match the datatype of the column. The column must also be enough to hold this expression. A DEFAULT expression cannot contain references to other columns, the pseudocolumn CURRVAL, NEXTVAL, LEVEL, and ROWNUM, or date constants that are not fully specified.

WITH ROWID -- stores the ROWID and the REF value in column or attribute. Storing a REF value with a ROWID can improve the performance of dereferencing operations, but will also use more space. Default storage of REF values is without ROWIDs.

SCOPE IS scope_table_name -- This restricts the scope of the column REF values to scope_table_name. The REF values for the column must come from REF values obtained from the object table specified in the clause. You can only specify one scope table per REF column.

The scope_table_name is the name of the object table in which object instances (of the same type as the REF column) are stored. The values in the REF column point to objects in the scope table.

You must have SELECT privileges on the table or SELECT ANY TABLE system privileges.

SCOPE FOR (ref_column_name) IS scope_table_name -- This restricts the scope of the REF values in ref_column_name to scope_table_name. The REF values for the column must come from REF values obtained from the object table specified in the clause.

The ref_column_name is the name of a REF column in an object table or an embedded REF attribute within an object column of a relational table. The values in the REF column point to objects in the scope table.

REF (ref_column_name) -- This is a reference to a row in an object table. You can specify either a REF column name of an object or relational table or an embedded REF attribute within an object column as ref_column_name.

OIDINDEX -- specifies an index on the hidden object identifier column and/or the storage specification for the index. Either index or storage_specification must be specified.

Index -- is the name of the index on the hidden object identifier column. If not specified, a name is generated by the system.

column_constraint -- defines an integrity constraint as part of the column definition.

table_constraint -- defines an integrity constraint as part of the table definition.

ORGANIZATION INDEX -- specifies that table is created as an index-only table. In an index-only table, the data rows are held in an index defined on the primary key for the table.

ORGANIZATION HEAP -- specifies that the data rows of table are stored in no particular order. This is the default.

PCTTHRESHOLD -- This specifies the percentage of space reserved in the index block for index-only table row. Any portion of the row that exceeds the specified threshold is stored in the area. If OVERFLOW is not specified, then rows exceeding the THRESHOLD limit are rejected. PCTTHRESHOLD must be a value from 0 to 50.

INCLUDING column_name -- This specifies a column at which to divide an index-only table row into index and overflow portions. All columns which follow column_name are stored in the overflow data segment. A column_name is either the name of the last primary key column or any non-primary key column.

PCTFREE -- This specifies the percentage of space in each of the table's, object table's OIDINDEX, or partition's data blocks reserved for future updates to the table's rows. The value of PCTFREE must be a value from 0 to 99. A value of 0 allows the entire block to be filled by inserts of new rows. The default value is 10. This value reserves 10% of each block for updates to existing rows and allows inserts of new rows to fill a maximum of 90% of each block.

PCTFREE has the same function in the PARTITION description clause and in the commands that create and alter clusters, indexes, snapshots, and snapshot logs. The combination of PCTFREE and PCTUSED determines whether inserted rows will go into existing data blocks or into new blocks.

For non-partitioned tables, the value specified for PCTFREE is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for PCTFREE is the default physical attribute of the segments associated with the table. The default value of PCTFREE applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify PCTFREE in the PARTITION description clause.

PCTUSED -- This specifies the minimum percentage of used space that Oracle for each data block of the table, object table OIDINDEX, or index-only table overflow data segment. A block becomes a candidate for row insertion when its used space falls below PCTUSED. PCTUSED is specified as a positive integer from 1 to 99 and defaults to 40.

PCTUSED has the same function in the PARTITION description clause and in the commands that create and alter clusters, snapshots, and snapshot logs.

For non-partitioned tables, the value specified for PCTUSED is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for PCTUSED is the default physical attribute of the segments associated with the table partitions. The default value of PCTUSED applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify PCTUSED in the PARTITION description clause.

PCTUSED is not a valid table storage characteristic if creating an index-only table (ORGANIZATION INDEX).

The sum of PCTFREE and PCTUSED must be less than 100. You can use PCTFREE and PCTUSED together use space within a table more efficiently.

INITRANS -- This specifies the initial number of transaction entries allocated within each data block allocated to the table, object table OIDINDEX, partition, LOB index segment, or overflow data segment. This value can range from 1 to 255 and defaults to 1. In general, you should not change the INITRANS value from its default.

Each transaction that updates a block requires a transaction entry in the block. The size of a transaction entry depends on your operating system.

This parameter ensures that a minimum number of concurrent transactions can update the block and helps avoid the overhead of dynamically allocating a transaction entry.

The INITRANS parameter serves the same purpose in the PARTITION description clause and in clusters, indexes, snapshots, and snapshot logs as in tables. The minimum and default INITRANS value for a cluster or index is 2, rather than 1.

For non-partitioned tables, the value specified for INITRANS is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for INITRANS is the default physical attribute of the segments associated with the table partitions. The default value of INITRANS applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify INITRANS in the PARTITION description clause.

MAXTRANS -- This specifies the maximum number of concurrent transactions that can update a data block allocated to the table, object table OIDINDEX, partition, LOB index segment, or index-only overflow data segment. This limit does not apply to queries. This value can range from 1 to 255 and the default is a function of the data block size. You should not change the MAXTRANS value from its default.

If the number concurrent transactions updating a block exceeds the INITRANS value, Oracle dynamically allocates transaction entries in the block until either the MAXTRANS value is exceeded or the block has no more free space. The MAXTRANS parameter serves the same purpose in the PARTITION description clause, clusters, snapshots, and snapshot logs as in tables.

For non-partitioned tables, the value specified for MAXTRANS is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for MAXTRANS is default physical attribute of the segments associated with the table partitions. The default value of MAXTRANS applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify MAXTRANS in the PARTITION description clause.

TABLESPACE -- This specifies the tablespace in which Oracle creates the table, object table OIDINDEX, partition, LOB storage, LOB index segment, or index-only table overflow data segment. If you omit this option, then Oracle creates the table, partition, LOB storage, LOB index segment, or partition in the default tablespace of the owner of the schema containing the table.

For non-partitioned tables, the value specified for TABLESPACE is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for TABLESPACE is the default physical attribute of the segments associated with the table partitions. The default value of TABLESPACE applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify TABLESPACE in the PARTITION description clause.

STORAGE -- This specifies the storage characteristics for the table, object table

OIDINDEX, partition, LOB storage, LOB index segment, or index-only table overflow data segment. This clause has performance ramifications for large tables. Storage should be allocated to minimize dynamic allocation of additional space.

For non-partitioned tables, the value specified for STORAGE is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for STORAGE is the default physical attribute of the segments associated with the table partitions. The default value of STORAGE applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify STORAGE in the PARTITION description clause.

OVERFLOW -- specifies that index-only table data rows exceeding the specified threshold are placed in the data segment listed in this clause.

LOGGING -- This specifies that the creation of the table, (and any indices required because of constraints), partition, or LOB storage characteristics will be logged in the redo log file. LOGGING also specifies that subsequent operations against the table, partition, or LOB storage are logged in the redo file. This is the default.

If the database is run in ARCHIVELOG mode, media recovery from a backup will recreate the table (and any indices required because of constraints). You cannot specify LOGGING when using NOARCHIVELOG mode.

For non-partitioned tables, the value specified for LOGGING is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for LOGGING is the default physical attribute of the segments associated with the table partitions. The default value of LOGGING applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify LOGGING in the PARTITION description clause.

Note: In future versions of Oracle, the LOGGING keyword will replace the RECOVERABLE option. RECOVERABLE is still available as a valid keyword in Oracle when creating non-partitioned tables, however, it is not recommended. You must specify LOGGING if creating a partitioned table.

RECOVERABLE -- See LOGGING above. RECOVERABLE is not a valid keyword for creating partitioned tables or LOB storage characteristics.

NOLOGGING -- This specifies that the creation of the table (and any indexes required because of constraints), partition, or LOB storage characteristics will not be logged in the redo log file. NOLOGGING also specifies that subsequent operations against the table or LOB storage is not logged in the redo file. As a result, media recovery will not recreate the table (and any indices required because of constraints).

For non-partitioned tables, the value specified for NOLOGGING is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for NOLOGGING is the default physical attribute of the segments associated with the table partitions. The default value of NOLOGGING applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify NOLOGGING in the PARTITION description clause. Using this keyword makes table creation faster than using the LOGGING option because redo log entries are not written.

NOLOGGING is not a valid keyword for creating index-only tables.

Note: In future versions of Oracle, the NOLOGGING keyword will replace the UNRECOVERABLE option. UNRECOVERABLE is still available as a valid keyword in Oracle when creating non-partitioned tables, however, it is not recommended. You must specify NOLOGGING if creating a partitioned table.

UNRECOVERABLE --See NOLOGGING above. This keyword can only be specified with the AS subquery clause. UNRECOVERABLE is not a valid keyword for creating partitioned or index-only tables.

LOB -- This specifies the LOB storage characteristics.

lob_item -- This is the LOB column name or LOB object attribute for which you are explicitly defining tablespace and storage characteristics that are different from those of the table.

STORE AS lob_segname -- This specifies the name of the LOB data segment. You cannot use lob_segname if more than one lob_item is specified.

CHUNK integer -- This is the unit of LOB value allocation and manipulation. Oracle allocates each unit of LOB storage as CHUNK integer. You can also use K or M to specify this size in kilobytes or megabytes. The default value of integer is 1K and the maximum is 32K. For efficiency, use a multiple of the Oracle block size.

PCTVERSION integer -- This is the maximum percentage of overall LOB storage space used for creating new versions of the LOB. The default value is 10, meaning that older versions of the LOB data are not overwritten until 10% of the overall LOB storage space is used.

INDEX lob_index_name -- This the name of the LOB index segment. You cannot use lob_index_name if more than one lob_item is specified.

NESTED TABLE nested_item STORE AS storage_table -- This specifies storage_table as the name of the storage table in which the rows of all nested_item values reside. You must include this clause when creating a table with columns or column attributes whose type is a nested table.

The nested_item is the name of a column or a column-qualified attribute whose type is a nested table.

The storage_table is the name of the storage table. The storage table is created in the same schema and the same tablespace as the parent table.

CLUSTER -- This specifies that the table is to be part of the cluster. The columns listed in this clause are the table columns that correspond to the cluster's columns. Generally, the cluster columns of a table are the column or columns that comprise its primary key or a portion of its primary key.

Specify one column from the table for each column in the cluster key. The columns are matched by position, not by name. Since a clustered table uses the cluster's space allocation, do not use the PCTFREE, PCTUSED, INITRANS, or MAXTRANS parameters, the TABLESPACE option, or the STORAGE clause with the CLUSTER option.

PARALLEL parallel_clause -- This specifies the degree of parallelism for creating the table and the default degree of parallelism for queries on the table once created.

This is not a valid option when creating index-only tables.

PARTITION BY RANGE -- This specifies that the table is partitioned on ranges of values from column_list.

column_list -- This is an ordered list of columns used to determine into which partition a row belongs. You cannot specify more than 16 columns in column_list. The column_list cannot contain the ROWID pseudocolumn or any columns of datatype ROWID or LONG.

PARTITION partition_name -- This specifies the physical partition clause. If partition_name is omitted, Oracle generates a name with the form SYS_Pn for the partition. The partition_name must conform to the rules for naming schema objects and their parts.

VALUES LESS THAN -- This specifies the non-inclusive upper bound for the current partition.

value_list -- This is an ordered list of literal values corresponding to column_list in the PARTITION BY RANGE clause. You can substitute the keyword MAXVALUE for any literal in value_list. Specifying a value other than MAXVALUE for the highest partition bound imposes an implicit integrity constraint on the table.

MAXVALUE --This specifies a maximum value that will always sort higher than any other value, including NULL.

ENABLE -- This enables an integrity constraint.

DISABLE -- This disables an integrity constraint.

Constraints specified in the ENABLE and DISABLE clauses of a CREATE TABLE statement must be defined in the statement. You can also enable and disable constraints with the ENABLE and DISABLE keywords of the CONSTRAINT clause. If you define a constraint but do not explicitly enable or disable it, Oracle enables it by default.

You cannot use the ENABLE and DISABLE clauses in a CREATE TABLE statement to enable and disable triggers.

AS subquery --This inserts the rows returned by the subquery into the table upon its creation.

The number of columns in the table must equal the number of expressions in the subquery. The column definitions can only specify column names, default values, and integrity constraints, not datatypes. Oracle derives datatypes and lengths from the subquery. Oracle also follows the following rules for integrity constraints:

Oracle also automatically defines any NOT NULL constraints on columns in the new table that existed on the corresponding columns of the selected table if the subquery selects the column rather than an expression containing the column.

A CREATE TABLE statement cannot contain both AS clause and a referential integrity constraint definition

If a CREATE TABLE statement contains both the AS clause and a CONSTRAINT clause or an ENABLE clause with the EXCEPTIONS option, Oracle ignores the EXCEPTIONS option. If any rows violate the constraint, Oracle does not create the table and returns an error message.

If all expressions in the subquery are columns, rather than expressions, you can omit the columns from the table definition entirely. In this case, the names of the columns of table are the same as the columns in the subquery.

CACHE -- This specifies that the data will be accessed frequently, therefore the blocks retrieved for this table are placed at the most recently used end of the LRU list in the buffer cache when a full table scan is performed. This option is useful for small lookup tables.

CACHE as a parameter in the LOB storage clause specifies that Oracle preallocates and retains LOB data values in memory for faster access.

CACHE is the default for index-only tables.

NOCACHE -- This specifies that the data will not be accessed frequently, therefore the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed. For LOBs, the LOB value is not placed in the buffer cache.

This is the default behavior except when creating index-only tables. This is not a valid keyword when creating index-only tables.

NOCACHE as a parameter in the LOB storage clause specifies that LOB values are not pre-allocated in memory. This is the LOB storage default.

<C>ORACLE7 CREATE TABLE Command Format

In contrast to the complex ORACLE8 CREATE TABLE commands listed above, here is the ORACLE7 (7.3.2) CREATE TABLE command:

```
>--------------CREATE TABLE-------------- table_name ----------------------------------------→>
```

```
                                    |--schema.-|
   V--------------------,---------------------------------------------------------|
>---( table-column  column-format -----------------------------------------------)------------------→>
                                              |--column constraint --|

>----------------------------------------------------------------------------------------------------→>
  |---PCTFREE integer(0-100)---|
  |---PCTUSED integer(0-100)---|
  |---INITRANS integer-----------|
  |---MAXTRANS integer--------|

>--TABLESPACE tablespace name-------------------------------------------------------→>
>--STORAGE storage_clause------------------------------------------------------------→>
>--CLUSTER cluster_name (column, column ....)--------------------------------------→>
>---ENABLE  clause-------------------------------------------------------------------------→>
  |--DISABLE clause--|

>--AS query---------------------------------------------------------------------------------→>
>----------------------------------------------------------------------------------------------→>
     |--RECOVERABLE (default)--|
     |– UNRECOVERABLE---------|

>---------------------------------------------------------------------------------------------→>
    |--PARALLEL parallel_clause----|
    |-- NOPARALLEL (default)------|
>-------------------------------------------------------------------------------------------→>
     |--CACHE ------------------|
     |--NOCACHE (default)---|
```

Where:

Table-name - this is formatted either user.name or just the name, defaulting to the current user. This must be a unique name by user. This unique name applies to tables, views, synonyms, clusters and indexes. For uniqueness, the user portion counts.

Column-name - This is a unique name by table. The column name can be up to 30 characters long and cannot contain a quotation, slash or character other than A-Z, 0-9,_,$ and #. A column name must not duplicate an Oracle reserved word (see the SQL Language Manual for a list of these words). To use mixed case, include the mixed case portion in quotation marks. For example:

```
EMPNAME
DOG
AULT.EXPENSE
"SELECT" (Even though select is a reserved
          word, if it is enclosed in quotes it is
           okay to use.)
"NIP AND TUCK" (Even spaces are allowed
                with quotes)
```

Names should be meaningful, not a bunch of symbols like A, B, C, etc. The Oracle CASE products always pluralize the names of tables, if you will be using CASE you might want to follow this convention.

Column-format - This is one of the allowed SQL data types. They are listed in the SQL language manual. A brief list is:

CHAR(size)	Character type data, max size 255 Under ORACLE7 this is replaced by VARCHAR2. Under ORACLE7

CHAR

	will be right side padded to specified length.
VARCHAR2	Variable length character up to 2000.
DATE	Date format, from 1/1/4712 BC to 12/31/4712 AD. Standard Oracle format is (10-APR-93)
LONG	Character, up to 65,535 long. Only one LONG per table. 2 gig under ORACLE7.
RAW(size)	Raw binary data, max of 255 size in version 6, 2000 under ORACLE7.
LONG RAW	Raw binary data in hexadecimal format, 65,535 long. 2 gig under ORACLE7.
ROWID	Internal data type, not user definable, used to uniquely identify table rows.
NUMBER(p,s)	Numeric data with p being precision and s being scale. Defaults to 38 p, null

s.

DECIMAL(p,s)	Same as numeric.
INTEGER	Defaults to NUMBER(38), no scale.
SMALLINT	Same as INTEGER
FLOAT	Same as NUMBER(38)
FLOAT(b)	NUMBER with precision of 1 to 126.
REAL	NUMBER(63)
DOUBLE PRECISION	Same as NUMBER(38)

No scale specification means floating point.

Column-Constraint - This is used to specify constraints. Constraints are limits placed either on a table or column. Oracle Version 6 supports the format of constraints and stores constraint definitions but does not enforce them. It is a statement of the format:

CONSTRAINT name constraint type

Constraints also may be of the form:

NULL CONSTRAINT constraint_type

NOT NULL CONSTRAINT constraint_type

PRIMARY KEY CONSTRAINT constraint_type

UNIQUE CONSTRAINT constraint_type

CHECK condition CONSTRAINT constraint_type

REFERENCES table name (column name) CONSTRAINT constraint_type

DEFAULT default_value_clause

In the above formats the "CONSTRAINT constraint_type" is optional.

Tables may also have the additional constraints:

```
FOREIGN KEY (column, column)
          REFERENCES table name (column, column)
          CONSTRAINT constraint_type
```

The foreign key constraint IS enforced. However, no index is automatically generated. It is suggested that indexes be maintained on foreign keys or else excessive full table scans may result.

PCTFREE - This parameter tells Oracle how much space to leave in each Oracle block for future updates. This defaults to 10. If a table will have a large number of updates, a larger value is needed, if the table will be static, a small value can be used. This is very table and table usage specific. Improper specification can result in chaining or improper space usage and performance degradation.

PCTUSED - This parameter tells Oracle the minimum level of space in a block to maintain. PCTUSED defaults to 40. A block becomes a candidate for updates if its storage falls below PCTUSED.
The sum of PCTFREE and PCTUSED may not exceed 100.
A high PCTUSED value results in more efficient space utilization but higher overhead as Oracle must work harder to

maintain the free block list.

INITRANS	This option specifies the initial number of transactions that are allocated within each block.
MAXTRANS	This option species the maximum number of transactions that can update a block concurrently.
TABLESPACE	This specifies the tablespace, if other than the users default.
STORAGE	This specifies the storage options for the table.
CLUSTER	This specifies the table is to be part of the specified cluster through the specified columns. PCTFREE, PCTUSED and TABLESPACE options will produce errors when used with the CLUSTER clause.
ENABLE --or-- DISABLE	This enables or disables constraints for an ORACLE7 database.

RECOVERABLE – or – UNRECOVERABLE - this determines if the operation that creates the table should generate redo/rollback data or simply fail in the event of a problem.

PARALLEL parallel_clause – or – NOPARALLEL - This sets the default parallelism of the table (for a parallel query situation sets how may processes will be used.)

CACHE – or – NOCACHE - this tells Oracle whether to cache the table in the SGA or not

<C>ANALYZE TABLE Command:

```
>-ANALYZE TABLE-------------- table -----------------------------------------------------------------
>>
                    |-schema.--|      |-COMPUTE-----------------------------STATISTICS--|
                                      |ESTIMATE STATISTICS SAMPLE n ----------------|
                                      |                                  |--ROWS ---|
                                      |                                  |PERCENT-|
                                      |-VALIDATE STRUCTURE -----------------------------|
                                      |                             |--CASCADE--|     |
                                      |-DELETE STATISTICS ---------------------------------|
```

Where:

COMPUTE STATISTICS - Calculates statistics based on all rows.
ESTIMATE STATISTICS - Calculates an estimate of the statistics based on “n” ROWS or PERCENT of rows in table.
VALIDATE STRUCTURE - Validates that the table and its indexes are consistent and not corrupted.
CASCADE - For clusters validates all tables in cluster.
DELETE STATISTICS - removes current statistics

The results for statistics appear in the DBA_TABLES view.

ALTER TABLE Command:

<C>ALTER TABLE command for ORACLE8

```
>--ALTER TABLE -------------- table--------------------------------------------------------------------
>>
                  |-schema.-|
>--ADD ------------------------------------------------------------------------------------------------
>>
              |--column datatype --------------------------------------------------------------------|
              |                     |--DEFAULT expr-----------------------------|              |
              |                     |--WITH ROWID-------------------------------|              |
              |                     |-SCOPE IS [schema.] scope_table_name--|              |
              |                     |--column_constraint --------------------------|              |
              |-- table_constraint ---------------------------------------------------------------------|
              |-- REF (ref_column_name) WITH ROWID ----------------------------------------|
>---SCOPE FOR (ref_column_name) IS [schema.]scope_table_name ..------------------------------
>>
               V--------------------------------,-------------------------------------------|
>---MODIFY (----column-----------------------------------------------------------------------)-------------
>>
                          |--datatype--------------|
                          |--DEFAULT----------|
                          |--column_constraint--|
 >---physical_attributes_clause---------------------------------------------------------------------------
>>
 >---LOB_storage_clause ----------------------------------------------------------------------------------
>>
   ^----------------,-------------|
  >>-----MODIFY_LOB_storage_clause ---------------------------------------------------------------------
>>
```

```
         ^-------------,-----------------------------|
>>----NESTED_TABLE_storage_clause------------------------------------------------------------------>>
       ^--------------,------------------------------|
>>----DROP drop_clause ------------------------------------------------------------------------------
>>
>>---ALLOCATE EXTENT ----------------------------------------------------------------------------
>>
                                 |-SIZE integer ------------------|
                                 |                    |-K-|       |
                                 |                    |-M-|       |

                                 | DATAFILE 'filename'--------|
                                 | INSTANCE integer-----------|

>-----DEALLOCATE UNUSED------------------------------------------------------------------>>
                                      |--KEEP integer [K | M] ]----|
>----ENABLE ------------------------------------------------------------------------------------->>
                 |--enable_clause-----|
                 |-- TABLE LOCK--|
 >--DISABLE ------------------------------------------------------------------------------------->>
                 |--disable_clause-----|
                 |-- TABLE LOCK---|
 >--PARALLEL parallel_clause------------------------------------------------------------------->>
                                       |--NOCACHE-- |
                                       |-- CACHE-------|
>---RENAME TO new_table_name---------------------------------------------------------------->>
>---OVERFLOW ------------------------------------------------------------------------------------>>
```

```
                    |--physical_attributes_clause----------------------------|
                    |--- INCLUDING column_name-----------------------|
                    | ALLOCATE EXTENT--------------------------------|
                    |                         |-SIZE integer K | M---|     |
                    |                         | DATAFILE filename |     |
                    |                         | INSTANCE integer---|     |
                    |-DEALLOCATE UNUSED ----------------------------|
                                                 |-KEEP integer K | M-|
>-----ADD OVERFLOW ----------------------------------------------------------------------->>
                         |--physical_attributes_clause--------------------------|
                         |--- PCTTHRESHOLD integer -----------------------|
                         |--- INCLUDING column_name-----------------------|
                         |--- TABLESPACE tablespace ------------------------|
>---MODIFY PARTITION partition_name --------------------------------------------->>
                                           |-physical_attributes_clause-|
                                           |-----LOGGING--------------- |
                                           |---- NOLOGGING-----------|
>---MOVE PARTITION partition_name------------------------------------------------->>
                                           |--physical_attributes_clause ----|
                                           |------LOGGING ------------------|
                                           |----- NOLOGGING---------------|
                                           | TABLESPACE tablespace------|
                                           | PARALLEL parallel_clause----|
>---ADD PARTITION ---------------------------------------------------------------------------->>
                        |--new_partition_name----------------------------------------------------|
                        |-VALUES LESS THAN (value_list) ----------------------------------|
                        |                                 |physical_attributes_clause| |
                         |-----LOGGING-----------------------------------------------------------|
                         |  |-- NOLOGGING--|                                                    |
                         |---- TABLESPACE tablespace -----------------------------------------|
>------DROP PARTITION partition_name------------------------------------------------------------>>
>---TRUNCATE PARTITION partition_name ------------------------------------------------------->>
                                          |--DROP STORAGE --|
                                          |-- REUSE STORAGE-|
>---SPLIT PARTITION partition_name AT (value_list)------------------------------------------------
>>
                                                |--INTO ( PARTITION split_partition_1-|
                                                |--physical_attributes_clause --------------|
                                                |----LOGGING ---------------------------------|
                                                |  |--NOLOGGING---|                           |
                                                |  TABLESPACE tablespace --------------|
    |---- , PARTITION------------------------------------------------------------------------------------------
-|
                         |--split_partition2----------------|
                         |--physical_attributes_clause---|
                         |---LOGGING ---------------------|
                         | |--NOLOGGING---|               |
                         |-- TABLESPACE tablespace--|
                         |-- PARALLEL parallel_clause-|
>----EXCHANGE PARTITION partition_name WITH TABLE non_partitioned_table_name --
>>
>-----------------------------------------------------------------------------------------------------------------------
>>
                | -------------------------- INDEXES---|
                    |    |--INCLUDING--|                    |
                    |    |--EXCLUDING-|                     |
                    | ---------------------VALIDATION--|
                      |--WITH---------|
                      |-- WITHOUT--|

>---MODIFY PARTITION UNUSABLE LOCAL INDEXES--------------------------------------
>>
>---MODIFY PARTITION REBUILD UNUSABLE LOCAL INDEXES -------------------------
><
```

The physical_attributes_clause contains zero, one or more of the following:

```
  PCTFREE integer
  PCTUSED integer
 INITRANS integer
 MAXTRANS integer
 STORAGE storage_clause
```

The LOB_storage_clause has the following format:

```
                    V---,---------|
>---  LOB (-----lob_item-----) STORE AS---lob_segname---------------------------------------------
>>
                                                    |-(--------------------------------------)-|
                                                    |    |-TABLESPACE tablespace-|   |
                                                    |    |-STORAGE storage_clause-|   |
                                                    |- CHUNK integer --------------------|
                                                    |- PCTVERSION integer ------------|
                                                    |- CACHE -----------------------------|
                                                     |- NOCACHE LOGGING ----------|
                                                     |- NOCACHE NOLOGGING------|
                                                    | INDEX [lob_index_name]---------|
                                                    |--(-------------------------------------)-|
                                                    |     |-TABLESPACE name-------|   |
                                                    |     |-STORAGE storage_clause-|  |
                                                    |     |--INITRANS integer---------|   |
                                                    |     |--MAXTRANS integer -----|   |
```

The MODIFY_LOB_storage_clause has the following format:

```
>------ MODIFY LOB (lob_item)-------------------------------------------------------------------->>
                         |-- (--------------------------------------------------)--|
                         |         |-- STORAGE storage_clause -----------|       |
                         |         |-- PCTVERSION integer ----------------|       |
                         |         |--CACHE ----------------------------------|        |
                         |         |--NOCACHE LOGGING ---------------|        |
                         |         |--NOCACHE NOLOGGING -----------|       |
                         |         |--ALLOCATE EXTENT ----------------|       |
                         |         |                                  |-SIZE n-----|         |
                         |         |                                            |-K-|         |
                         |         |                                            |-M-|         |
                         |        |--DATAFILE 'filename' ------------------|        |
                         |        |--INSTANCE integer ---------------------|        |
                         |-- DEALLOCATE UNUSED----------------------------|
                         |                        |--KEEP integer -----|     |
                         |                                      |K |        |
                         |                                      |M|         |
                           | INDEX [lob_index_name]---------------------------------|
                         |--(---------------------------------------------------------)-|
                         |    |-TABLESPACE name-------------------------------|   |
                         |    |-STORAGE storage_clause------------------------|   |
                         |    |--INITRANS integer---------------------------------|  |
                         |    |--MAXTRANS integer -----------------------------|  |
                         |    |-ALLOCATE EXTENT-----------------------------|  |
                         |                        |(SIZE integer ----------| |
                         |                        |DATAFILE 'filename'| |
                         |                        | INSTANCE integer) |  |
                         |    |DEALLOCATE UNUSED-------------------------|  |
                                                    |--KEEP integer-------|
                                                               |K |
                                                               |M |
```

The NESTED_TABLE_storage_clause has the simple format:

NESTED TABLE nested_item STORE AS storage_table

<C>The keywords and parameters for the ORACLE8 commands and clauses

schema **--** This is the schema containing the table. If you omit schema, Oracle assumes the table is in your own schema.

table -- This is the name of the table to be altered. You can alter the definition of an index-only table.

ADD -- This adds a column or integrity constraint.

MODIFY -- This modifies the definition of an existing column. If you omit any of the optional parts of the column definition (datatype, default value, or column constraint), these parts remain unchanged.

column -- This is the name of the column to be added or modified.

datatype -- This specifies a datatype for a new column or a new datatype for an existing column.

You can only omit the datatype if the statement also designates the column as part of the foreign key of a referential integrity constraint. Oracle automatically assigns the column the same datatype as the corresponding column of the referenced key of the referential integrity constraint.

DEFAULT -- This specifies a default value for a new column or a new default for an existing column. Oracle assigns this value to the column if a subsequent INSERT statement omits a value for the column. The datatype of the default value must match the datatype specified for the column. The column must also be long enough to hold the default value. A DEFAULT expression cannot contain references to other columns, the pseudocolumns CURRVAL, NEXTVAL, LEVEL, and ROWNUM, or date constants that are not fully specified.

column_constraint -- This adds or removes a NOT NULL constraint to or from an existing column.

table_constraint -- This adds an integrity constraint to the table.

PCTFREE

PCTUSED

INITRANS

MAXTRANS -- These changes the value of specified parameters for the table, partition, the or overflow data segment, or the default characteristics of a partitioned table. See the PCTFREE, PCTUSED, INITRANS, and MAXTRANS parameters of the CREATE TABLE command.

STORAGE -- This changes the storage characteristics of the table, partition, or overflow data segment.

PCTTHRESHOLD -- This specifies the percentage of space reserved in the index block for an index-only table row. Any portion of the row that exceeds the specified threshold is stored in the overflow area. If OVERFLOW is not specified, then rows exceeding the THRESHOLD limit are rejected. PCTTHRESHOLD must be a value from 0 to 50.

The sub clauses for the PCTTRESHOLD clause are:

INCLUDING column_name -- This specifies a column at which to divide an index-only table row into index and overflow portions. All columns which follow column_name are stored in the overflow data segment. A column_name is either the name of the last primary key column or any non-primary key column.

LOB -- This specifies the LOB storage characteristics.

lob_item -- This is the LOB column name or LOB object attribute for which you are explicitly defining tablespace and storage characteristics that are different from those of the table.

The sub clauses for the LOB clause are:

STORE AS

lob_segname -- This specifies the name of the LOB data segment. You cannot use lob_segname if more than one lob_item is specified.

CHUNK integer -- This is the unit of LOB value allocation and manipulation. Oracle allocates each unit of LOB storage as CHUNK integer. You can also use K or M to specify this size in kilobytes or megabytes. The default value of integer is 1K and the maximum is 32K. For efficiency, use a multiple of the Oracle block size.

PCTVERSION integer -- This is the maximum percentage of overall LOB storage space used for creating new versions of the LOB. The default value is 10, meaning that older versions of the LOB data are not overwritten until 10% of the overall LOB storage space is used.

INDEX lob_index_name -- This is the name of the LOB index segment. You cannot use lob_index_name if more than one lob_item is specified.

MODIFY LOB (column) -- This modifies the physical attributes of the LOB storage column. You can only specify one LOB column for each MODIFY LOB clause.

NESTED TABLE nested_item STORE AS storage_table -- This specifies storage_table as the name of the storage table in which the rows of all nested_item values reside. You must include this clause when modifying a table with columns or column attributes whose type is a nested table.

The nested_item is the name of a column or a column-qualified attribute whose type is a nested table.

The storage_table is the name of the storage table. The storage table is modified in the same schema and the same tablespace as the parent table.

DROP -- This drops an integrity constraint.

ALLOCATE EXTENT -- This explicitly allocates a new extent for the table, the overflow data segment, the LOB data segment, or the LOB index.

The sub clauses for the ALLOCATE EXTENT clause are:

SIZE -- This specifies the size of the extent in bytes. You can use K or M to specify the extent size in kilobytes or megabytes. If you omit this parameter, Oracle determines the size based on the values of the table's overflow data segment's, or LOB index's STORAGE parameters.

DATAFILE -- This specifies one of the data files in the table's, overflow data segment's, LOB data's tablespace, or LOB index's tablespace to contain the new extent. If you omit this parameter, Oracle chooses the data file.

INSTANCE -- This makes the new extent available to the freelist group associated with the specified instance. If the instance number exceeds the maximum number of freelist groups, the former is divided by the latter, and the remainder is used to identify the freelist group to be used. An instance is identified by the value of its initialization parameter INSTANCE_NUMBER. If you omit this parameter, the space is allocated to the table, but is not drawn from any particular freelist group. Rather the master freelist is used, and space is allocated as needed. Only use this parameter if you are using Oracle with the Parallel Server option in parallel mode.

Explicitly allocating an extent with this clause does affect the size for the next extent to be allocated as specified by the NEXT and PCTINCREASE storage parameters.

DEALLOCATE UNUSED -- This explicitly deallocates unused space at the end of the table, overflow data segment, LOB data segment, or LOB index and makes the space available for other segments. You can free only unused space above the high-water mark. If KEEP is omitted, all unused space is freed.

The DEALLOCATE UNUSED sub clause has the definition:

KEEP -- This specifies the number of bytes above the high-water mark that the table, overflow data segment, LOB data segment, or LOB index will have after deallocation. If the number of remaining extents are less than MINEXTENTS, then MINEXTENTS is set to the current number of extents. If the initial extent becomes smaller than INITIAL, then INITIAL is set to the value of the current initial extent.

OVERFLOW -- This specifies the overflow data segment physical storage attributes to be modified for the index-only table. Parameters specified in this clause are only applicable to the overflow data segment. See the CREATE TABLE command.

ADD OVERFLOW -- This adds an overflow data segment to the specified index-only table.

ENABLE enable_clause -- This enables a single integrity constraint or all triggers associated with the table.

ENABLE TABLE LOCK -- This enables DML and DDL locks on a table in a parallel server environment.

DISABLE disable_clause -- This disables a single integrity constraint or all triggers associated with the tables.

Integrity constraints specified in DISABLED clauses must be defined in the ALTER TABLE statements or in a previously issued statement. You can also enable and disable integrity constraints with the ENABLE and DISABLE keywords of the CONSTRAINT clause. If you define an integrity constraint but do not explicitly enable or disable it, Oracle enables it by default.

DISABLE TABLE LOCK -- This disables DML and DDL locks on a table to improve performance in a parallel server environment .

PARALLEL parallel_clause -- This specifies the degree of parallelism for the table. PARALLEL is not a valid option for index-only tables.

CACHE -- This specifies that the data is accessed frequently, therefore the blocks retrieved for this table are placed at the most recently used end of the LRU list in the buffer cache when a full table scan is performed. This option is useful for small lookup tables.

CACHE is not a valid option for index-only tables.

NOCACHE -- This specifies that the data is not accessed frequently, therefore the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed. For LOBs, the LOB value is not placed in the buffer cache. This is the default behavior.

NOCACHE is not a valid option for index-only tables.

LOGGING -- LOGGING specifies that subsequent operations against the table, partition, or LOB storage that can execute without logging are logged in the redo file.

If the table is partitioned, only the default table attributes are affected.

If the database is run in ARCHIVELOG mode, media recovery from a backup will recreate the table (and any indexes required because of constraints).

The logging attribute of the base table is independent that of its indexes.

NOLOGGING -- specifies that altering the table (and any indexes required because of constraints), partition, or LOB storage characteristics will not be logged in the redo log file. NOLOGGING also specifies that subsequent operations against the table or LOB storage are not logged in the redo file. As a result, media recovery will not recreate the table (and any indexes required because of constraints).

When running in NOARCHIVELOG mode, all operations that can execute without logging will not generate log entries even if LOGGING is specified for the table.

Using this keyword makes table modifications faster than using the LOGGING option because redo log entries are not written.

NOLOGGING is not a valid keyword for altering index-only tables.

RENAME TO new_table_name -- This renames table to new_table_name.

MODIFY PARTITION partition_name -- This modifies the real physical attributes of a table partition. You can specify any of the following as new physical attributes for the partition:

logging attribute
PCTFREE
PCTUSED
INITRANS
MAXTRANS
STORAGE

RENAME PARTITION partition_name TO new_partition_name -- This renames table partition

partition_name to new_partition_name.

MOVE PARTITION partition_name -- moves table partition partition_name to another segment. You can move partition data to another tablespace, re-cluster data to reduce fragmentation, or change a create-time physical attribute.

ADD PARTITION new_partition_name -- adds a new partition new_partition_name to the "high" end of a partitioned table. You can specify any of the following as new physical attributes for the partition:

logging attribute
PCTFREE
PCTUSED
INITRANS
MAXTRANS
STORAGE

VALUES LESS THAN (value_list) -- This specifies the upper bound for the new partition. The value_list is a comma-separated, ordered list of literal values corresponding to column_list. The value_list must collate greater than the partition bound for the highest existing partition in the table.

DROP PARTITION partition_name -- This removes partition partition_name, and the data in that partition, from a partitioned table.

TRUNCATE PARTITION partition_name -- This removes all rows from a partition in a table.

TRUNCATE PARTITION clauses:

DROP STORAGE -- This specifies that space from the deleted rows be deallocated and made available for use by other schema objects in the tablespace.

REUSE STORAGE -- This specifies that space from the deleted rows remains allocated to the partition. The space is subsequently only available for inserts and updates to the same partition.

SPLIT PARTITION partition_name_old -- This creates two new partitions, each with a new segment and new physical attributes, and new initial extents. The segment associated with the old partition is discarded.

SPLIT PARTITION clauses:

AT (value_list) -- This specifies the new non-inclusive upper bound for split_partition_1. The value_list must compare less than the pre-split partition bound for partition_name_old and greater than the partition bound for the next lowest partition (if there is one).

INTO -- This describes the two partitions resulting from the split.

PARTITION split_partition_1,
PARTITION split_partition_2

specifies the names and physical attributes of the two partitions resulting from the split.

EXCHANGE PARTITION partition_name -- This converts partition partition_name into a non-partitioned table, and a non-partitioned table into a partition of a partitioned table by exchanging their data (and index) segments.

EXCHANGE PARTITION clauses:

WITH TABLE table -- This specifies the table with which the partition will be exchanged.

INCLUDING INDEXES -- This specifies that the local index partitions be exchanged with the corresponding regular indexes.

EXCLUDING INDEXES -- This specifies that all the local index partitions corresponding to the partition and all the regular indexes on the exchanged table are marked as unusable.

WITH VALIDATION -- This specifies that any rows in the exchanged table that do not collate properly return an error.

WITHOUT VALIDATION -- This specifies that the proper collation of rows in the exchanged table are not checked.

UNUSABLE LOCAL INDEXES -- This marks all the local index partitions associated with partition_name as unusable.

REBUILD UNUSABLE LOCAL INDEXES -- This rebuilds the unusable local index partitions associated with the specified partition_name.

<C>ORACLE7 Alter Table Command Format:

```
>----ALTER TABLE --------------tablename--------------------------------------------------------------
>>
                    |-schema.-|          |-ADD ------------------------------------------------|
                                         |          |--column name---|                         |
                                         |          |--table constraint-|                      |
                                         |          ^-----------,------------|                 |
                                         |--MODIFY---column name-------------------------|
                                         |                            |--data type--------|   |
                                         |                            |--constraint ------|   |
                                         |--DROP CONSTRAINT constraint----------------|
                                         |--PCTFREE integer-----------------------------------|
                                         |--PCTUSED integer-----------------------------------|
                                         |--INITRANS integer----------------------------------|
                                         |--MAXTRANS integer--------------------------------|
                                         |--STORAGE storage_clause------------------------|
                                         |--ALLOCATE EXTENT----------------------------|
                                         |--DEALLOCATE UNUSED-----------------------|
                                         |--ENABLE constraint--------------------------------|
                                         |                         |-TABLE LOCK-|        |
                                         |--DISABLE constraint-------------------------------|
                                         |                         |-TABLE LOCK -|       |
                                         |--PARALLEL –-------------------------------------|
                                         | |-NOPARALLEL-|                                 |
                                         |--CACHE –-----------------------------------------|
                        |-NOCACHE-|
```

<C>Table Statistics and Validity - The ANALYZE Command

<C>The ORACLE8 syntax for ANALYZE

```
>----ANALYZE------------------------------------------------------------------------------------------->>
             |----------------INDEX-----------------------------------------|
             |----------------TABLE-----------------------------------------|
             |----------------CLUSTER-------------------------------------|
             |----------------index-----PARTITION (partition_name)--|
             |  |-schema.-|                                       |
             |----------------table------PARTITION (partition_name)--|
>---- COMPUTE STATISTICS------------------------------------------------------------------------->>
                        |--FOR for_clause--|
>-----ESTIMATE STATISTICS-------------------------------------------------------------------->>
                        |-FOR for_clause-| |-SAMPLE integer ---------------------|
                                                   |-ROWS------|
                                                   |-PERCENT-|
>---DELETE STATISTICS----------------------------------------------------------------------------->>
>---VALIDATE ------------------------------------------------------------------------------------------>>
              |--REF UPDATE---------------------------|
              |--STRUCTURE---------------------------|
                          |-CASCADE-|
>---LIST CHAINED ROWS---------------------------------------------------------------------------->>
                       |--INTO -------------- table--|
                                          |-schema.-|
```

The format for the FOR clause (used for HISTOGRAM option):

```
FOR TABLE
FOR ALL --------------------- COLUMNS --------------------------
              |-INDEXED-|                  |-SIZE integer--|
FOR COLUMNS ------------------- column | attribute--------------------
                    |-SIZE integer-|                    |-SIZE integer-|
                                    ^-----------------------------------------|
FOR ALL ------------------ INDEXES
             |-LOCAL-|
```

<C> The keywords and parameters for the ORACLE8 ANALYZE command

attribute -- This specifies the qualified column name of an item in an object.

INDEX -- This identifies an index to be analyzed (if no FOR clause is used). If you omit schema, Oracle assumes the index is in your own schema.

TABLE -- identifies a table to be analyzed. If you omit schema, Oracle assumes the table is in your own schema. When you collect statistics for a table, Oracle also automatically collects the statistics for each of the table's indexes, provided that no FOR clauses are used.

PARTITION (partition_name) -- This specifies that statistics will be gathered for (partition_name). You cannot use this option when analyzing clusters.

CLUSTER -- This identifies a cluster to be analyzed. If you omit schema, Oracle assumes the cluster is in your own schema. When you collect statistics for a cluster, Oracle also automatically collects the statistics for all the cluster's tables and all their indexes, including the cluster index.

VALIDATE REF UPDATE -- This validates the REFs in the specified table, checks the ROWID portion in each REF, compares it with the true ROWID, and corrects, if necessary. You can only use this option when analyzing a table.

COMPUTE STATISTICS -- computes exact statistics about the analyzed object and stores them in a data dictionary

ESTIMATE STATISTICS -- This estimates statistics about the analyzed object and stores them in the data dictionary

The clauses for ESTIMATE STATISTICS are:

SAMPLE -- specifies the amount of data from the analyzed object Oracle samples to estimate statistics. If you omit this parameter, Oracle samples 1064 rows. If you specify more than half of the data, Oracle reads all the data and computes the statistics.

ROWS -- This causes Oracle to sample integer rows of the table or cluster or integer entries from the index. The integer must be at least 1.

PERCENT-- causes Oracle to sample integer percent of the rows from the table or cluster or integer percent of the index entries. The integer can range from 1 to 99.

The following HISTOGRAM clauses only apply to the ANALYZE TABLE version of this command:

FOR TABLE -- collect table statistics for the table.

FOR ALL COLUMNS -- collect column statistics for all columns and scalar attributes.

FOR ALL INDEXED COLUMNS -- collect column statistics for all indexed columns in the table.

FOR COLUMNS -- collect column statistics for the specified columns and scalar object attributes.

FOR ALL INDEXES -- all indexes associated with the table will be analyzed.

FOR ALL LOCAL INDEXES -- specifies that all local index partitions are analyzed. You must specify the keyword LOCAL if the PARTITION (partition_name) clause and the index option are specified.

SIZE -- This specifies the maximum number of partitions in the histogram. The default value is 75, minimum value is 1, and maximum value is 254.

DELETE STATISTICS -- delete any statistics about the analyzed object that are currently stored in the data dictionary

VALIDATE STRUCTURE -- This validates the structure of the analyzed object. If you use this option when analyzing a cluster, Oracle automatically validates the structure of the cluster's tables.

INTO -- This specifies a table into which Oracle lists the rowids of the partitions whose rows do not collate correctly. If you omit schema, Oracle assumes the list is in your own schema. If you omit this clause all together, Oracle assumes that the table is named INVALID_ROWS. According to Oracle documentation the SQL script used to create this table is UTLVALID.SQL, however as of 7.3.2 and 8.0.2 there is no trace of this file to be found.

CASCADE -- This validates the structure of the indexes associated with the table or cluster. If you use this option when validating a table, Oracle also validates the table's indexes. If you use this option when validating a cluster, Oracle also validates all the clustered tables' indexes, including the cluster index.

LIST CHAINED ROWS -- This identifies migrated and chained rows of the analyzed table or cluster. You cannot use this option when analyzing an index.

INTO -- This specifies a table into which Oracle lists the migrated and chained rows. If you omit schema, Oracle assumes the list table is in own schema. If you omit this clause altogether, Oracle assumes that the table is named CHAINED_ROWS. The script used to create this table is UTLCHAIN.SQL. The list table must be on your local database.

The structure under ORACLE8 for the table is:

```
CREATE TABLE chained_rows (
owner_name        varchar2(30),
table_name        varchar2(30),
cluster_name      varchar2(30),
partition_name    varchar2(30),
head_rowid        rowid,
analyze_timestamp date);
```

The structure under ORACLE7 for the table is:

```
CREATE TABLE chained_rows (
owner_name        varchar2(30),
table_name        varchar2(30),
cluster_name      varchar2(30),
head_rowid        rowid,
timestamp         date);
```

To analyze index-only tables, you must create a separate chained rows table for each index-only table created to accommodate the primary key storage of index-only tables. Use the SQL scripts DBMSIOTC.SQL and PRVTIOTC.PLB to define the DBMS_IOT package with the BUILD_CHAIN_ROWS_TABLE(owner VARCHAR2, iot_name VARCAHR2, chainrow_table_name VARCHAR2) procedure and then execute this procedure to create an IOT_CHAINED_ROWS table for an index-only table. The IOT_CHAINED_ROWS table has the structure:

```
CREATE TABLE iot_chained_rows (
owner_name              varchar2(30),
table_name              varchar2(30),
cluster_name            varchar2(30),
partition_name          varchar2(30),
head_rowid              rowid,
timestamp               date,
test1                   varchar2(6) <<--- This is the primary key
);                                   from the table being analyzed
```

<C>DELETE TABLE Command:

```
>-------DELETE --------------------------table name---------------------------------------------->>
             |-FROM-| |-schema.-|          |-@dblink-|
                                     |-PARTITION (partition_name)----------------|
                                                                    |-@dblink-|
>----------------------(subquery) --------------------------------------------------------------------->>
    |-THE-|                         |-alias-|
>--WHERE condition ----------------------------------------------------------------------------------->>
```

Where the clauses have the following definitions:

schema - This is the schema or owner of the table, view or partition being deleted from. If it is left off the users default schema is used.

table or view - This is the name of the table or view to be deleted from.

dblink - If the table, view or partition is in a remote database, this is the dblink to that database.

PARTITION(partition name) - This deletes from a specified (partition_name) sub-partition of a partitioned table.

THE -- This is used to flatten nested tables, the subquery following the THE clause tells Oracle how the flattening should occur.

subquery -- Used to tell Oracle how to delete from the table or nested table. If deletion is from a nested table the THE clause must be included.

alias -- Used when a correlated sub-query is used to denote table hierarchy in the query/delete commands.

WHERE condition - The condition each deleted row must meet or fail.

NOTE: You can use hints in a DELETE statement to optimize delete subquery processing.

The table name can include an alias, if the WHERE clause is left out, all of the rows in the table are deleted.

To delete specific rows from a nested table, the THE clause is specified (I like to think ‘FROM THE SET’ when I see THE):

```
        DELETE THE (SELECT addresses
                FROM clientsv8 c
                WHERE c.customer_name = ‘Joes Bar and Grill, Inc.’) AS a
        WHERE a.addrtype=1;
```

Deleting from a single partition is accomplished by use of the PARTITION clause:

```
    DELETE FROM trains PARTITION (roundhouse1)
    WHERE service_date < to_date(‘01-Jan-1956 00:00:00, ‘DD-Mon-YYYY hh24:mi:ss’);
```

<C>TRUNCATE TABLE Command:

```
>----TRUNCATE TABLE ---------------table_name-------------------------------------------------->>
                     |-schema.-|              |-PRESERVE-- SNAPSHOT LOG--|
                                             | |-PURGE----|                   |
                                             |-DROP ------STORAGE---------------|
                                              |-REUSE-|
```

The DROP/REUSE STORAGE option allows you to shrink the table back to its high water mark or leave the table at its current size. Both DROP and REUSE qualifiers also apply to whatever index space is regained.

For tables: PRESERVE or PURGE SNAPSHOT options allow control over a tables snapshot logs as well.

The truncate command is faster than the delete command because it is a DDL command and generates no rollback data. If using truncate on a clustered table, the data must be removed from the entire cluster, not just the one table. Any referential integrity constraints on a table must be disabled before it can be truncated. Like a table DROP, a truncation is not recoverable. If a table is truncated you cannot rollback if you made a mistake. Use TRUNCATE carefully.

<C>DROP TABLE Command:

```
>----DROP TABLE---------------- table name---------------------------------------------------------->>
                |-schema.-|                |-CASCADE CONSTRAINTS-|
```

Oracle will drop the table regardless of its contents. The only time a drop will fail is when a table's primary key is referenced by another tables foreign key via a restraint clause.

<C>TYPES:

<C>The TYPE creation command is:

```
                                                    V--------------,------------------------|
>----CREATE TYPE -----------------type_name--(-------------------------------------------)--;--->>
                  |-schema.-|                    |-attribute, datatype-|
```

<C>Command to create an INCOMPLETE TYPE:

```
>--CREATE ----------------------TYPE ----------------type_name--------------------;----------->>
             |--OR REPLACE-|         |-schema.-|              |-AS OBJECT-|
```

An incomplete type would be used in the situation where a type referenced a second type which referenced the first type (a circular reference such as emp-supervisor). This allows the incomplete type to be referenced before it is completed. However, before a table can be constructed from an incomplete type it must be completed.

<C>The command to create a VARRAY type would be:

```
>CREATE ---------------------TYPE ------------type_name AS -VARRAY(n)----------  OF dtype;-
>>
            |-OR REPLACE-|     |-schema.-|                 | VARYING ARRAY(n)|
```

A VARRAY should be used when the number of items to be stored in the type is:

1. Known and fixed
2. Small (this is a relative term, remember the data is stuffed into a RAW and stored in line with the rest of the types data.)

A VARRAY cannot be used in a partitioned table. In early release (i.e. up to and including 8.0.3) varrays take up an inordinate amount of space. I suggest using nested tables as the overall storage requirements are lower and the limitations are the same.

<C>The command to create a NESTED TABLE TYPE would be:

```
>---  CREATE----------------------TYPE-------------- type_name AS TABLE OF dtype; -------->>
```

```
                   |-OR REPLACE-|          |-schema.-|
```

A NESTED TABLE should be used when:

1. The number of items is large or unknown.
2. The storage of the items needs to be managed.

A NESTED TABLE is stored in a STORE TABLE which must be specified in the CREATE TABLE command for each NESTED TABLE type used. The NESTED TABLE type cannot be used in partition tables. In some early documentation releases it incorrectly states that Oracle itself specifies the store table name, this is incorrect. Nested tables cannot contain VARRAYs or other nested tables.

<C>Object Types

If you will be using the TYPE to build an object table that will be REFed by a second table, it must be constructed as an OBJECT type and thus include an Object ID (OID). Nested tables and VARRAYs are limited in the types of TYPEs they can store, a second OBJECT table is not. IN cases where the ERD shows a series of one-to-many type relationships OBJECT tables will have to be used to show this relationship structure under the object -oriented paradigm in ORACLE8.

<C>The command to create an OBJECT TYPE would be:

```
>---  CREATE ------------------------TYPE ---------------type_name AS OBJECT------->>
                   |-OR REPLACE-|          |-schema.-|
>-- (------------------------------------------------------------------------------------------);----->>
        |--attribute_name  dtype----------------------------------------------------------|
        |  ^---------,---------------------|                                                   |
        |------------- MEMBER function_specification--------------------------------|
        |  |--MAP---|                                                                        |
        | |-ORDER-|                                                                          |
        |^-------------------------------------------------------|                           |
        |--- MEMBER procedure_specification----------------------------------------|
        |^------------------------,-------------------------------|                          |
        |--PRAGMA RESTRICT_REFERENCES (method_name, constraints)--|
         ^--------------------------,------------------------------------------------------|
```

The constraints for the PRAGMA line are:

RNDS -- Reads no database state

WNDS -- Writes no database state

RNPS --- Reads no package state

WNPS --- Writes no package state

Note: These can be specified in any order, but no duplicates are allowed.

NOTE: Object types cannot be used in partition tables.

The possible dtype specifications are:

REF schema.object_type_name

schema.type_name

VARCHAR2(size)

NUMBER (precision, scale)

DATE

RAW(size)

CHAR(size)

CHARACTER(size)

CHAR(size)

CHARACTER VARYING(size)

CHAR VARYING(size)

VARCHAR(size)

The following data types are provided for compatibility, but are treated internally the same as NUMBER:

NUMERIC(precision, scale)

DECIMAL(precision, scale)

DEC(precision, scale)

INTEGER

INT

SMALLINT

FLOAT(size)

DOUBLE PRECISION

REAL

The following are Large Object Data types:

BLOB

CLOB

BFILE

<C>Keywords And Parameters For The TYPE Commands

OR REPLACE -- This recreates the type if it already exists. You can use this option to change the definition of an existing type without first dropping it. Users previously granted privileges on the recreated object type can use and reference the object type without be granted privileges again.

schema -- This is the schema to contain the type. If this is omitted, Oracle creates the type in the users default schema.

type_name -- This is the name of an object type, a nested table type, or a VARRAY type.

AS OBJECT -- This creates the type as a user-defined object type. The variables that form the data structure are called attributes. The member procedures and functions that define the object's behavior are called methods.

Note: AS OBJECT is new for Oracle, version 8.0.2 and is now required when creating an object type.

AS TABLE -- This creates a named nested table of type datatype. When datatype is an object type, the nested table type describes a table whose columns match the name and attributes of the object type. When datatype is a scalar type, then the nested table type describes a table with a single, scalar type column called "column_value".

AS VARRAY(limit) -- This creates the type as an ordered set of elements, each of which has the same datatype. You must specify a name and a maximum limit of zero or more. The array limit must be an integer literal. Only variable length arrays are supported. Oracle does not support anonymous VARRAYs. The type name for the objects contained in the VARRAY must be one of the following:

-- a scalar datatype.

-- a REF

-- an object type, including an object with VARRAY attributes

The type name for the objects contained in the VARRAY cannot be any of the following:

-- an object type with a nested table attribute

-- a VARRAY type

-- a TABLE type

- A LOB data type

The (limit) value is an unsigned integer in the range of 1 to 2^31 (2,147,483,647). Of course, since the VARRAY is stored in line with the other data in a table as a RAW value, specifications of extremely large limits would be foolish since it would cause chaining and performance degradation. For a large number of elements, use a NESTED TABLE instead.

OF dtype -- This is the name of any Oracle built-in datatype or library type. ROWID, LONG, and LONG RAW are not valid datatypes.

REF object_type_name -- This associates an instance of a source type with an instance of the target object. A REF logically identifies and locates the target object. The target object must have an object identifier.

type_name -- This is the name of user-defined object type, nested table type, or a VARRAY type.

attribute_name -- This is an object attribute name. Attributes are data items with a name and a type specifier that form the structure of the object.

MEMBER -- This specifies a function or procedure subprogram method associated with the object

type which is referenced as an attribute. You must specify a corresponding method body in the object type body for each procedure or function specification.

procedure_specification -- This is the specification of a procedure subprogram.

function_specification -- This is the specification of a function subprogram.

MAP MEMBER function_specification -- This specifies a member function (map method) that returns the relative position of a given instance in the ordering of all instances of the object. A map method is called implicitly and induces an ordering of object instances by mapping them to values of a predefined scalar type. PL/SQL uses the ordering to evaluate Boolean expressions and to perform comparisons.

A scalar value is always manipulated as a single unit. Scalars are mapped directly to the underlying hardware. An integer, for example, occupies 4 or 8 contiguous bytes of storage, in memory or on disk.

An object specification can contain only one map method, which must be a function. The result type must be a predefined SQL scalar type, and the map function can have no arguments other than the implicit SELF argument.

You can define either MAP method or ORDER method in a type specification, but not both.

If a MAP or an ORDER method is not specified, only comparisons for equality or inequality can be performed and thus the object instances cannot be ordered. No comparison method needs to be specified to determine the equality of two object types.

ORDER MEMBER function_specification -- This specifies a member function (ORDER method)

that takes an instance of an object as an explicit argument and the implicit SELF argument and returns either a negative, zero, or positive integer. The negative, positive, or zero indicates that the implicit SELF argument is less than, equal to, or greater than the explicit argument.

When instances of the same object type definition are compared in an ORDER BY clause, the order method function_specification is invoked.

An object specification can contain only one ORDER method, which must be a function having the return type INTEGER.

You can either declare a MAP method or an ORDER method, but not both. If you declare either method, you can compare object instances in SQL.

If you do not declare either method, you can only compare object instances for equality or inequality. Note that instances of the same type definition are equal only if each pair of their corresponding attributes is equal. No comparison method needs to be specified to determine the equality of two object types.

PRAGMA RESTRICT_REFERENCES -- This is a compiler directive that denies member functions read/write access to database tables, packaged variables, or both, and thereby helps to avoid side effects. The arguments for this directive are:

method_name This is the name of the MEMBER function or procedure to which the pragma is being applied.

WNDS -- This specifies function writes no database state
(does not modify database tables).

WNPS -- This specifies function writes no package state
(does not modify packaged variables).

RNDS -- This specifies function reads no database state
(does not query database tables).

RNPS -- This specifies function reads no package state
(does not reference packages variables).

<C>Creation of Object Tables:

```
>---CREATE TABLE ---------------- table ---------------- OF ------------------object_type--------->>
                         |--schema--|                          |--schema.--|

>(------------------------------------------------column attribute -----------------------------------)→>
                                                                  |--DEFAULT expr-|
                                                                  |--WITH ROWID--|
  V----------------------------------------------------------------------------------------------|
>---SCOPE IS ---------------- scope_table_name------------------------------------------------------>>
```

```
                 |--schema.--|                      |---column_constraint----|

>--- table_constraint ------------------------------------------------------------------------------->>

                        |--REF (ref_column_name) WITH ROWID --|

>--SCOPE FOR (ref_column_name) IS  -------------- scope_table_name--------------------------->>

                                          |-schema.-|

                                          |  V------------ , ----------------------------------------|

>----------------------------------------------------column attribute -------------------------------------→>

                                                         |--DEFAULT expr-|

                                                         |--WITH ROWID--|


>----------- OIDINDEX ------------------------------------------------------------------------------→

                              |-index-|    |-phys_attrib--|   |-TABLESPACE tablespace---|

>----------- phys_attrib ------------------------------------------------------------------------------->>

>--TABLESPACE tablespace -------------------------------------------------------------------------->>

>--LOB --lob_item -----STORE AS------------------------------------------------------------------->>

         ^----,--------|                  |                  | |                                           |

                                    |-lob_segname-| |--TABLESPACE tablespace ----------------|

                                                  | STORAGE storage_clause-------------------|

                                                  | CHUNK integer--------------------------------|

                                                  | PCTVERSION integer ------------------------|

                                                  |-- CACHE-----------------------------------------|

                                                  |  |-- NOCACHE LOGGING ---------|          |

                                                  |  |-- NOCACHE NOLOGGING -----|          |

                                                  | INDEX --lob_index_name---------------------|

                                                  |TABLESPACE tablespace---------------------|

                                                           |  STORAGE storage_clause    |
                                                                  |  INITRANS integer              |
                                                                  |  MAXTRANS integer           |
>--- NESTED TABLE nested_item STORE AS storage_table----------------------------------------->
>--- LOGGING ------------------------------------------------------------------------------------------>
    |- NOLOGGING--|
>--- CLUSTER --cluster ----(---column---)------------------------------------------------------------>
                                        ^----,------|
>--- PARALLEL parallel_clause -------------------------------------------------------------------->
       V-----------------------------------|
>---------- ENABLE enable_clause----------------------------------------------------------------->>
            | DISABLE disable_clause |
>------- AS subquery ------------------------------------------------------------------------------->>
>--------CACHE ------------------------------------------------------------------------------------->>
          | NOCACHE |
```

phys_attrib ::=

```
 PCTFREE integer
 PCTUSED integer
 INITRANS integer
 MAXTRANS integer
 STORAGE (storage_clause )
```

<C>The Keywords and Parameters for CREATE TABLE Commands

The CREATE TABLE commands have the following parameter definitions in ORACLE8:

schema -- This is the schema to contain the table. If you omit schema, Oracle creates the table in your own schema.

table -- This is the name of the table (or object table) to be created. A partitioned table cannot be a clustered table or an object table.

OF object_type -- This explicitly creates an object table of type object_type. The columns of an object table correspond to the top-level attributes of type object_type. Each row will contain an object instance and each instance will be assigned a unique, system-generated object identifier (OID) when a row is inserted. If you omit schema, Oracle creates the object table in your own schema. Object tables cannot be partitioned.

Column -- This specifies the name of a column of the table. A table can have up to 1000 columns. You may only omit column definitions when using the AS subquery clause.

Attribute -- specifies the qualified column name of an item in an object.

Datatype -- This is the datatype of a column. You can omit the datatype only if the statement also designates the column as part of a foreign key in a referential integrity constraint. Oracle automatically assigns the column the datatype of the corresponding column of the referenced key of the referential integrity constraint.

Object types, REF object_type, VARRAYs, and nested tables are valid datatypes.

DEFAULT -- specifies a value to be assigned to the column if a subsequent INSERT statement omits a value for the column. The datatype of the expression must match the datatype of the column. The column must also be enough to hold this expression. For the syntax of expr, see "Expr" on page 3-144. A DEFAULT expression cannot contain references to other columns, the pseudocolumns CURRVAL, NEXTVAL, LEVEL, and ROWNUM, or date constants that are not fully specified.

WITH ROWID -- stores the ROWID and the REF value in column or attribute. Storing a REF value with a ROWID can improve the performance of dereferencing operations, but will also use more space. Default storage of REF values is without ROWIDs.

SCOPE IS scope_table_name -- This restricts the scope of the column REF values to scope_table_name. The REF values for the column must come from REF values obtained from the object table specified in the clause. You can only specify one scope table per REF column.

The scope_table_name is the name of the object table in which object instances (of the same type as the REF column) are stored. The values in the REF column point to objects in the scope table.

You must have SELECT privileges on the table or SELECT ANY TABLE system privileges.

SCOPE FOR (ref_column_name) IS scope_table_name -- This restricts the scope of the

REF values in ref_column_name to scope_table_name. The REF values for the column must come from REF values obtained from the object table specified in the clause.

The ref_column_name is the name of a REF column in an object table or an embedded REF attribute within an object column of a relational table. The values in the REF column point to objects in the scope table.

REF (ref_column_name) -- This is a reference to a row in an object table. You can specify either a REF column name of an object or relational table or an embedded REF attribute within an object column as ref_column_name.

OIDINDEX -- specifies an index on the hidden object identifier column and/or the storage specification for the index. Either index or storage_specification must be specified.

Index -- is the name of the index on the hidden object identifier column. If not specified, a name is generated by the system.

column_constraint -- defines an integrity constraint as part of the column definition.

table_constraint -- defines an integrity constraint as part of the table definition.

ORGANIZATION INDEX -- specifies that table is created as an index-only table. In an index-only table, the data rows are held in an index defined on the primary key for the table.

ORGANIZATION HEAP -- specifies that the data rows of table are stored in no particular order. This is the default.

PCTTHRESHOLD -- This specifies the percentage of space reserved in the index block for index-only table row. Any portion of the row that exceeds the specified threshold is stored in the area. If OVERFLOW is not specified, then rows exceeding the THRESHOLD limit are rejected. PCTTHRESHOLD must be a value from 0 to 50.

INCLUDING column_name -- This specifies a column at which to divide an index-only table row into index and overflow portions. All columns which follow column_name are stored in the overflow data segment. A column_name is either the name of the last primary key column or any non-primary key column.

PCTFREE -- This specifies the percentage of space in each of the table's, object table's OIDINDEX, or partition's data blocks reserved for future updates to the table's rows. The value of PCTFREE must be a value from 0 to 99. A value of 0 allows the entire block to be filled by inserts of new rows. The default value is 10. This value reserves 10% of each block for updates to existing rows and allows inserts of new rows to fill a maximum of 90% of each block.

PCTFREE has the same function in the PARTITION description clause and in the commands that create and alter clusters, indexes, snapshots, and snapshot logs. The combination of PCTFREE and PCTUSED determines whether inserted rows will go into existing data blocks or into new blocks.

For non-partitioned tables, the value specified for PCTFREE is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for PCTFREE is the default physical attribute of the segments associated with the table. The default value of PCTFREE applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify PCTFREE in the PARTITION description clause.

PCTUSED -- This specifies the minimum percentage of used space that Oracle for each data block of the table, object table OIDINDEX, or index-only table overflow data segment. A block becomes a candidate for row insertion when its used space falls below PCTUSED. PCTUSED is specified as a positive integer from 1 to 99 and defaults to 40.

PCTUSED has the same function in the PARTITION description clause and in the commands that create and alter clusters, snapshots, and snapshot logs.

For non-partitioned tables, the value specified for PCTUSED is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for PCTUSED is the default physical attribute of the segments associated with the table partitions. The default value of PCTUSED applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify PCTUSED in the PARTITION description clause.

PCTUSED is not a valid table storage characteristic if creating an index-only table (ORGANIZATION INDEX).

The sum of PCTFREE and PCTUSED must be less than 100. You can use PCTFREE and PCTUSED together use space within a table more efficiently.

INITRANS -- This specifies the initial number of transaction entries allocated within each data block allocated to the table, object table OIDINDEX, partition, LOB index segment, or overflow data segment. This value can range from 1 to 255 and defaults to 1. In general, you should not change the INITRANS value from its default.

Each transaction that updates a block requires a transaction entry in the block. The size of a transaction entry depends on your operating system.

This parameter ensures that a minimum number of concurrent transactions can update the block and helps avoid the overhead of dynamically allocating a transaction entry.

The INITRANS parameter serves the same purpose in the PARTITION description clause and in clusters, indexes, snapshots, and snapshot logs as in tables. The minimum and default INITRANS value for a cluster or index is 2, rather than 1.

For non-partitioned tables, the value specified for INITRANS is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for INITRANS is the default physical attribute of the segments associated with the table partitions. The default value of INITRANS applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify INITRANS in the PARTITION description clause.

MAXTRANS -- This specifies the maximum number of concurrent transactions that can update a data block allocated to the table, object table OIDINDEX, partition, LOB index segment, or index-only overflow data segment. This limit does not apply to queries. This value can range from 1 to 255 and the default is a function of the data block size. You should not change the MAXTRANS value from its default.

If the number concurrent transactions updating a block exceeds the INITRANS value, Oracle dynamically allocates transaction entries in the block until either the MAXTRANS value is exceeded or the block has no more free space. The MAXTRANS parameter serves the same purpose in the PARTITION description clause, clusters, snapshots, and snapshot logs as in tables.

For non-partitioned tables, the value specified for MAXTRANS is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for MAXTRANS is default physical attribute of the segments associated with the table partitions. The default value of MAXTRANS applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify MAXTRANS in the PARTITION description clause.

TABLESPACE --	This specifies the tablespace in which Oracle creates the table, object table OIDINDEX, partition, LOB storage, LOB index segment, or index-only table overflow data segment. If you omit this option, then Oracle creates the table, partition, LOB storage, LOB index segment, or partition in the default tablespace of the owner of the schema containing the table.
	For non-partitioned tables, the value specified for TABLESPACE is the actual physical attribute of the segment associated with the table.
	For partitioned tables, the value specified for TABLESPACE is the default physical attribute of the segments associated with the table partitions. The default value of TABLESPACE applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify TABLESPACE in the PARTITION description clause.
STORAGE --	This specifies the storage characteristics for the table, object table OIDINDEX, partition, LOB storage, LOB index segment, or index-only table overflow data segment. This clause has performance ramifications for large tables. Storage should be allocated to minimize dynamic allocation of additional space.
	For non-partitioned tables, the value specified for STORAGE is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for STORAGE is the default physical attribute of the segments associated with the table partitions. The default value of STORAGE applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify STORAGE in the PARTITION description clause.

OVERFLOW -- specifies that index-only table data rows exceeding the specified threshold are placed in the data segment listed in this clause.

LOGGING -- This specifies that the creation of the table, (and any indices required because of constraints), partition, or LOB storage characteristics will be logged in the redo log file. LOGGING also specifies that subsequent operations against the table, partition, or LOB storage are logged in the redo file. This is the default.

If the database is run in ARCHIVELOG mode, media recovery from a backup will recreate the table (and any indices required because of constraints). You cannot specify LOGGING when using NOARCHIVELOG mode.

For non-partitioned tables, the value specified for LOGGING is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for LOGGING is the default physical attribute of the segments associated with the table partitions. The default value of LOGGING applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify LOGGING in the PARTITION description clause.

Note: In future versions of Oracle, the LOGGING keyword will replace the RECOVERABLE option. RECOVERABLE is still available as a valid keyword in Oracle when creating non-partitioned tables, however, it is not recommended. You must specify LOGGING if creating a partitioned table.

RECOVERABLE --	See LOGGING above. RECOVERABLE is not a valid keyword for creating partitioned tables or LOB storage characteristics.
NOLOGGING --	This specifies that the creation of the table (and any indexes required because of constraints), partition, or LOB storage characteristics will not be logged in the redo log file. NOLOGGING also specifies that subsequent operations against the table or LOB storage is not logged in the redo file. As a result, media recovery will not recreate the table (and any indices required because of constraints).

For non-partitioned tables, the value specified for NOLOGGING is the actual physical attribute of the segment associated with the table.

For partitioned tables, the value specified for NOLOGGING is the default physical attribute of the segments associated with the table partitions. The default value of NOLOGGING applies to all partitions specified in the CREATE statement (and on subsequent ALTER TABLE ADD PARTITION statements) unless you specify NOLOGGING in the PARTITION description clause. Using this keyword makes table creation faster than using the LOGGING option because redo log entries are not written.

NOLOGGING is not a valid keyword for creating index-only tables.

Note: In future versions of Oracle, the NOLOGGING keyword will replace the UNRECOVERABLE option. UNRECOVERABLE is still available as a valid keyword in Oracle when creating non-partitioned tables, however, it is not recommended. You must specify NOLOGGING if creating a partitioned table.

UNRECOVERABLE --See NOLOGGING above. This keyword can only be specified with the AS subquery clause. UNRECOVERABLE is not a valid keyword for creating partitioned or index-only tables.

LOB -- This specifies the LOB storage characteristics.

lob_item -- This is the LOB column name or LOB object attribute for which you are explicitly defining tablespace and storage characteristics that are different from those of the table.

STORE AS lob_segname -- This specifies the name of the LOB data segment. You cannot use lob_segname if more than one lob_item is specified.

CHUNK integer -- This is the unit of LOB value allocation and manipulation. Oracle allocates each unit of LOB storage as CHUNK integer. You can also use K or M to specify this size in kilobytes or megabytes. The default value of integer is 1K and the maximum is 32K. For efficiency, use a multiple of the Oracle block size.

PCTVERSION integer -- This is the maximum percentage of overall LOB storage space used for creating new versions of the LOB. The default value is 10, meaning that older versions of the LOB data are not overwritten until 10% of the overall LOB storage space is used.

INDEX lob_index_name -- This the name of the LOB index segment. You cannot use lob_index_name if more than one lob_item is specified.

NESTED TABLE nested_item STORE AS storage_table -- This specifies storage_table as the name of the storage table in which the rows of all nested_item values reside. You must include this clause when creating a table with columns or column attributes whose type is a nested table.

The nested_item is the name of a column or a column-qualified attribute whose type is a nested table.

The storage_table is the name of the storage table. The storage table is created in the same schema and the same tablespace as the parent table.

CLUSTER -- This specifies that the table is to be part of the cluster. The columns listed in this clause are the table columns that correspond to the cluster's columns. Generally, the cluster columns of a table are the column or columns that comprise its primary key or a portion of its primary key.

Specify one column from the table for each column in the cluster key. The columns are matched by position, not by name. Since a clustered table uses the cluster's space allocation, do not use the PCTFREE, PCTUSED, INITRANS, or MAXTRANS parameters, the TABLESPACE option, or the STORAGE clause with the CLUSTER option.

PARALLEL parallel_clause -- This specifies the degree of parallelism for creating the table and the default degree of parallelism for queries on the table once created.

This is not a valid option when creating index-only tables.

PARTITION BY RANGE -- This specifies that the table is partitioned on ranges of values from column_list.

column_list -- This is an ordered list of columns used to determine into which partition a row belongs. You cannot specify more than 16 columns in column_list. The column_list cannot contain the ROWID pseudocolumn or any columns of datatype ROWID or LONG.

PARTITION partition_name -- This specifies the physical partition clause. If partition_name is omitted, Oracle generates a name with the form SYS_Pn for the partition. The partition_name must conform to the rules for naming schema objects and their parts.

VALUES LESS THAN -- This specifies the non-inclusive upper bound for the current partition.

value_list -- This is an ordered list of literal values corresponding to column_list in the PARTITION BY RANGE clause. You can substitute the keyword MAXVALUE for any literal in value_list. Specifying a value other than MAXVALUE for the highest partition bound imposes an implicit integrity constraint on the table.

MAXVALUE --This specifies a maximum value that will always sort higher than any other value, including NULL.

ENABLE -- This enables an integrity constraint.

DISABLE -- This disables an integrity constraint.

Constraints specified in the ENABLE and DISABLE clauses of a CREATE TABLE statement must be defined in the statement. You can also enable and disable constraints with the ENABLE and DISABLE keywords of the CONSTRAINT clause. If you define a constraint but do not explicitly enable or disable it, Oracle enables it by default.

You cannot use the ENABLE and DISABLE clauses in a CREATE TABLE statement to enable and disable triggers.

AS subquery --This inserts the rows returned by the subquery into the table upon its creation.

The number of columns in the table must equal the number of expressions in the subquery. The column definitions can only specify column names, default values, and integrity constraints, not datatypes. Oracle derives datatypes and lengths from the subquery. Oracle also follows the following rules for integrity constraints:

Oracle also automatically defines any NOT NULL constraints on columns in the new table that existed on the corresponding columns of the selected table if the subquery selects the column rather than an expression containing the column.

A CREATE TABLE statement cannot contain both AS clause and a referential integrity constraint definition

If a CREATE TABLE statement contains both the AS clause and a CONSTRAINT clause or an ENABLE clause with the EXCEPTIONS option, Oracle ignores the EXCEPTIONS option. If any rows violate the constraint, Oracle does not create the table and returns an error message.

If all expressions in the subquery are columns, rather than expressions, you can omit the columns from the table definition entirely. In this case, the names of the columns of table are the same as the columns in the subquery.

CACHE -- This specifies that the data will be accessed frequently, therefore the blocks retrieved for this table are placed at the most recently used end of the LRU list in the buffer cache when a full table scan is performed. This option is useful for small lookup tables.

CACHE as a parameter in the LOB storage clause specifies that Oracle preallocates and retains LOB data values in memory for faster access.

CACHE is the default for index-only tables.

NOCACHE -- This specifies that the data will not be accessed frequently, therefore the blocks retrieved for this table are placed at the least recently used end of the LRU list in the buffer cache when a full table scan is performed. For LOBs, the LOB value is not placed in the buffer cache.

This is the default behavior except when creating index-only tables. This is not a valid keyword when creating index-only tables.

NOCACHE as a parameter in the LOB storage clause specifies that LOB values are not pre-allocated in memory. This is the LOB storage default.

<C>CREATE INDEX Command:

<C>ORACLE8 CREATE INDEX Command Syntax.

```
                                                                      V-------,-----------|
>-CREATE -----------------INDEX--------------index-ON-----------------------------------------------
>>
          |-BITMAP---|         |-schema.-|           |                  v--------,-------------|
          |-UNIQUE---|                               |-------------table(-column----------------)-|
                                                    | |-schema.-|                 |-ASC-----|    |
                                                    |                             |-DESC---|    |
                                                     |-CLUSTER--------------cluster-----------|
                                                                |-schema.-|
>-----------------------------------------------------------------------------------------------------
>>
   |-physical_attributes_clause-|
>------------------------------------------------------------------------------------------------------------
>>
    |-LOGGING-----|
    |-NOLOGGING-|
>-----------------------------------------------------------------------------------------------------------
>>
   |-TABLESPACE tablespace -|
   |-DEFAULT--------------------|
>-----------------------------------------------------------------------------------------------------------
>>
  |-NOSORT---|
  |-REVERSE--|
>-----------------------------------------------------------------------------------------------------------
>>
                                                          V----------------,---------------------------|
  |-GLOBAL PARTITION BY RANGE (column_list)----------------------------------------------|
                                                          |-PARTITION [partition_name]--------|
                                                          ___________________________|  |___
                                                          |                                              |
                                                          |-VALUES LESS THAN (value_list)--|
                                                        |- physical_attributes_clause-----------|
                                                         |------------------------------------------|
                                                          |-LOGGING -------|
                                                          |-NOLOGGING--- |
              V-------------------------,-------------------------------------------------------------|
>--LOCAL--------------------------------------------------------------------------------------------------
>>
```

```
                |-PARTITION [partition_name]-----------------------------------------------------------|
                                                 |--physical_attributes_clause---------------------|
                                                  |------------------------------------------------------|
                                                    |-LOGGING----- |
                                                    |-NOLOGGING -|
>---PARALLEL parallel_clause --------------------------------------------------------------------->>
```

The physical_attributes_clause contains zero, one or more of the following:

```
  PCTFREE integer
  PCTUSED integer
  INITRANS integer
  MAXTRANS integer
  STORAGE storage_clause
```

The keywords and parameters for the ORACLE8 CREATE INDEX clauses are:

UNIQUE -- This specifies that the value of the column (or group of columns) in the table to be indexed must be unique.

If the index is local non-prefixed (see LOCAL clause below), then the index columns must contain the partitioning columns.

BITMAP -- This specifies that index is to be created as a bitmap, rather than as a B-tree. You cannot use this keyword when creating a global partitioned index.

schema -- This is the schema to contain the index. If you omit schema, Oracle creates the index in your own schema.

index -- This is the name of the index to be created. An index can contain several partitions.

You cannot range partition a cluster index or an index defined on a clustered table.

table -- This is the name of the table for which the index is to be created. If you do not qualify table with schema, Oracle assumes the table is contained in your own schema.

- If the index is LOCAL, then table must be partitioned.
- You cannot create an index on an index-only table.
- You can create an index on a nested table storage table.

column -- This is the name of a column in the table. An index can have up to 16 columns. A datatype of LONG or LONG RAW cannot be used for indexing.

You can create an index on a scalar object (non-VARRAY) attribute column or on the system-defined NESTED_TABLE_ID column of the nested table storage table. If an object attribute column is specified, the column name must be qualified with the table name. If a nested table column attribute is specified, then it must be qualified with the outermost table name, the containing column name, and all intermediate attribute names leading to the nested table column attribute.

ASC and DESC -- These are allowed for DB2 syntax compatibility, although indexes are always created in ascending order. Indexes on character data are created in ascending order of the character values in the database character set.

CLUSTER schema.cluster -- This specifies the cluster for which a cluster index is to be created. If you do not qualify cluster with schema, Oracle assumes the cluster is contained in your current schema. You cannot create a cluster index for a hash cluster.

INITRANS and MAXTRANS -- These establish values for these parameters for the index. See the INITRANS and MAXTRANS parameters of the CREATE TABLE command.

TABLESPACE tablespace -- This is the name of the tablespace to hold the index or index partition. If this option is omitted, Oracle creates the index in the default tablespace of the owner of the schema containing the index. This causes immediate contention if the table, and its index, are both contained in the default tablespace of the schema owner.

For a partitioned index, this is the tablespace name.

For a LOCAL index, you can specify the keyword DEFAULT in place of a tablespace name. New partitions added to the LOCAL index will be created in the same tablespace(s) as the corresponding partition(s) of the underlying table.

STORAGE -- This establishes the storage characteristics for the index.

PCTFREE -- This is the percentage of space to leave free for updates and insertions within each of the index's data blocks.

NOSORT -- This indicates to Oracle that the rows are stored in the database in ascending order and therefore Oracle does not have to sort the rows when creating the index. You cannot specify REVERSE with this option. If the rows are not in ascending order when this

clause

is used, an ORA-01409 error is returned and no index is created.

REVERSE -- This stores the bytes of the index block in reverse order, excluding the ROWID. You cannot specify NOSORT with this option.

LOGGING -- This specifies that the creation of the index will be logged in the undo and redo log and redo and log data will be recorded on activity in this index.

If index is non-partitioned, this is the logging attribute of the index.

If index is partitioned, this is the default logging attribute of the index partitions created. If index is LOCAL, this value is used as the default attribute for index partitions created when new partitions are added to the base table of the index.

If the [NO]LOGGING clause is omitted, the logging attribute of the index defaults to the logging attribute of the tablespace in which it resides

If the database is run in NOARCHIVELOG mode, index creation is not logged in the undo

and redo log file, even if LOGGING is specified. Media recovery from backup will not recreate the index.

If the database is run in ARCHIVELOG mode, media recovery from a backup **will** recreate the index.

NOLOGGING -- This specifies that the creation of the index **will not** be logged in the undo and redo log file. As a result, media recovery will not recreate the index.

If index is non-partitioned, this is the logging attribute of the index.

If index is partitioned, this is the default logging attribute of the index partitions created. If index is LOCAL, this value is used as the default attribute for index partitions created when new partitions are added to the base table of the index.

If the [NO]LOGGING clause is omitted, the logging attribute of the index defaults to the logging attribute of the tablespace in which it resides.

Using this keyword makes index creation faster than using the LOGGING option because undo and redo log entries are not written

GLOBAL-- This specifies that the partitioning of the index is user-defined and is not equi-partitioned with the underlying table. By default, non-partitioned indexes are global indexes.

PARTITION BY RANGE specifies that the global index is partitioned on the ranges of values from the columns specified in column_list. You cannot specify this clause for a LOCAL index.

(column_list) -- This is the name of the column(s) of a table on which the index is partitioned. The column_list must specify a left prefix of the index column list.

You cannot specify more than 16 columns in column_list and the columns cannot contain the ROWID pseudocolumn or a column of type ROWID.

LOCAL -- This specifies that the index is range partitioned on the same columns, with the same number of partitions, and the same partition bounds as the underlying partitioned table. Oracle automatically maintains LOCAL index partitioning as the underlying table is repartitioned.

PARTITION partition_name -- This names the individual partitions. The number of PARTITION clauses determines the number of partitions. If the index is local, the number of index partitions must be equal to the number, and will correspond to the order of the table partitions.

The partition_name is the name of the physical index partition. If partition_name is omitted Oracle punishes you by generating a name with the form SYS_Pn where n is

some

arbitrary value from a SYS maintained sequence. NAME YOUR PARTITIONS!

For LOCAL indexes, if partition_name is omitted Oracle generates a name that is consistent with the corresponding table partition. If the name conflicts with an existing index partition name, the form SYS_Pn is used.

VALUES LESS THAN (value_list) -- This specifies the (non-inclusive) upper bound for the current partition in a global index. This means that is the value specified is 10 then everything less that 10, *but not including 10*, will be stored in this partition. The value_list is a comma-separated, ordered list of literal values corresponding to column_list in the PARTITION BY RANGE clause. Always specify MAXVALUE as the value_list of the last partition.

You cannot specify this clause for a local index.

PARALLEL -- This specifies the degree of parallelism for creating the index.

<C>ORACLE7 CREATE INDEX Command

```
>-CREATE ------------------ INDEX ------------- Index  ON ------------- table (column------------)-
>>
               |-UNIQUE---|            |-schema.-|            |-schema.-|                |-ASC---| |
               |-BITMAP---|                                  |                          |-DESC-|  |
                                                             |                 ^--------,----------| |
                                                             |-CLUSTER----------- cluster name-|
                                                                       |-schema.-|
>------------------------------------------------------------------------------------------------------------
>>
  |-INITRANS n----|
  |-MAXTRANS n-|
>------------------------------------------------------------------------------------------------------------
>>
   |--TABLESPACE tablespace name--|
  |--STORAGE storage clause----------|
  |--PCTFREE n -------------------------|
  |--NOSORT------------------------------|
  |--RECOVERABLE –-----------------|
     |– UNRECOVERABLE---|
>------------------------------------------------------------------------------------------------------------
>>
```

|-PARALLEL clause-|

Where:

Index name - is a user unique index name

Table name - is the name of the table to be indexed, the table must exist.

Column - is the name of the column to include in the index, maximum of 16.

The Order of a concatenated key is important. Only queries that access columns in this order will use the index. For example, table EXAMPLE has 16 columns. The first three are used as the concatenated index.
Only queries that contain columns 1,2,3 --or-- 1,2 --or-- 1 will use the index.

Tablespace name - is the name of the tablespace in which to store the index.

Storage clause - is a standard storage clause

NOSORT - tells oracle to not sort the index, since the table is already loaded into Oracle in ascending Order.

RECOVERABLE – or – UNRECOVERABLE - This is used to tell Oracle whether to generate redo/rollback information. For large index creates, it is suggested that the unrecoverable option be used to speed index creation.

<C>ANALYZE INDEX Command:

```
>-ANALYZE INDEX-------------- index ------------------------------------------------------------------
>>
                        |-schema.--|        |-COMPUTE-----------------------------STATISTICS--|
                                            |ESTIMATE STATISTICS SAMPLE n ----------------|
                                            |                                        |--ROWS ---|
                                            |                                        |PERCENT-|
                                            |-VALIDATE STRUCTURE -----------------------------|
                                            |                              |--CASCADE--|
                                            |-DELETE STATISTICS ---------------------------------|
```

Where:

COMPUTE STATISTICS - Calculates statistics based on all rows.
ESTIMATE STATISTICS - Calculates an estimate of the statistics based on “n” ROWS or PERCENT of rows in table.
VALIDATE STRUCTURE - Validates that the table and its indexes are consistent and not corrupted.
CASCADE - For clusters validates all tables in cluster.
DELETE STATISTICS - Removes current statistics.

The results appear in the INDEX_STATS view. One thing to remember is that unlike the DBA_TABLES view only one row at a time is saved in the INDEX_STATS view, the row for the last index analyzed.

<C>ALTER INDEX Commands:

<C>ALTER INDEX Command Syntax for ORACLE8

```
>--ALTER INDEX ---------------index---------------------------------------------------------->>
                    |-schema.-|         |-REBUILD--------------------------------------|
                                                  |-PARALLEL parallel_clause--|
                                                  | |-NOPARALLEL--------------| |
                                                  |--LOGGING----------------------|
                                                  | |-NOLOGGING-|                 |
                                                  |--REVERSE----------------------|
                                                  | |-NOREVERSE-|                 |
                                                  |--physical_attributes_clause---|
                                                  |--TABLESPACE tablespace--|
>---- DEALLOCATE UNUSED-------------------------------------------------------------------->>
                                  |--KEEP integer------------|
                                                  |-K--|
                                                  |-M--|
>---ALLOCATE EXTENT------------------------------------------------------------------------>>
                             |--SIZE integer----------------------------------|
                            |               |-K-|                      |
                            |               |-M-|                      |
                            |---DATAFILE 'filename'----------------------|
                            |--INSTANCE integer--------------------------|
>--PARALLEL parallel_clause------------------------------------------------------------------------>>
  |--NOPARALLEL------------------|
>--physical_attributes_clause----------------------------------------------------------------------->>
>--LOGGING------------------------------------------------------------------------------------------>>
   |-- NOLOGGING--|
>---RENAME TO new_index_name -----------------------------------------------------------------
>>
>---MODIFY PARTITION partition_name-----------------------------------------------------------
>>
                                  |--physical_attributes_clause --|
                                   |--LOGGING--------------------|
                                   | |--NOLOGGING--|              |
                                   |--UNUSABLE-------------------|
>---RENAME PARTITION partition_name TO new_partition_name -------------------------------
>>
>--DROP PARTITION partition_name--------------------------------------------------------------------
>>
>-SPLIT PARTITION partition_name AT (value_list)-------------------------------------------------
>>
                                        |           V--------------------------------|
                                        |- INTO (-----------------------------------)--|
                                        |          |-PARTITION [split_n]--------|   |
                                        |          |- physical_attributes_clause--|   |
                                        |           |- TABLESPACE tablespace -|  |
                                        |           |---LOGGING ------------------|   |
                                        |              |-NOLOGGING---|
                                        |--PARALLEL parallel_clause--------------|
                                         |-NOPARALLEL ---------------|
>----REBUILD PARTITION partition_name ------------------------------------------------------------
>>
                                     |-physical_attributes_clause----------------|
                                      |-TABLESPACE tablespace --------------|
                                              |--PARALLEL parallel_clause------------|
                                              | | NOPARALLEL---------------|            |
                                              |---LOGGING--------------------------------|
                                                |-NOLOGGING---|
>----UNUSABLE------------------------------------------------------------------------------------------
><
```

The physical_attributes_clause of the ORACLE8 ALTER INDEX command consists of zero, one or more of the following:

 PCTFREE integer
 INITRANS integer
 MAXTRANS integer
 STORAGE storage_clause

Keywords and Parameters

schema -- This is the schema containing the index. If you omit schema, Oracle assumes the index is in your own schema.

index -- This is the name of the index to be altered.

The following operations can only be performed on partitioned global indexes:

drop partition
split partition

rename partition

rebuild partition

modify partition

partition_name -- This is the name of the index partition to be altered. It must be a partition in index.

PCTFREE

INITRANS

MAXTRANS -- These change the value of these parameters for the index or index partition See the PCTFREE, INITRANS and MAXTRANS parameters of the CREATE TABLE command for more information.

STORAGE This changes the storage parameters for the index.

ALLOCATE EXTENT -- This explicitly allocates a new extent for the index.

The sub-clauses for the ALLOCATE EXTENT clause are:

SIZE -- This specifies the size of the extent in bytes. You can use K or M to specify the extent size in kilobytes or megabytes. If you omit this parameter, Oracle determines the size based on the values of the index's STORAGE parameters.

DATAFILE -- specifies one of the data files in the index's tablespace to contain the new extent. If you omit this parameter, chooses the data file.

INSTANCE -- This makes the new extent available to the specified instance. An instance is identified by the value of its initialization parameter INSTANCE_NUMBER. If you omit this parameter, the extent is available to all instances. This parameter is valid only if you are using Oracle with the Parallel Server option in parallel mode.

Explicitly allocating an extent with this clause does affect the size for the next extent to be allocated as specified by the NEXT and PCTINCREASE storage parameters.

DEALLOCATE UNUSED -- This explicitly deallocates unused space at the end of the index and make the freed space available for other segments. Only unused space above the high-water mark can be freed. If KEEP is omitted, all unused space is freed.

The sub-clauses for the DEALLOCATE UNUSED clause are:

KEEP--specifies the number of bytes above the high-water mark that the index will have after deallocation. If the number of remaining extents are less than MINEXTENTS, then MINEXTENTS is set to the current number of extents. If the initial extent becomes smaller than INITIAL, then INITIAL is set to the value of the current initial extent.

REBUILD -- This clause is used to recreate an existing index.

The sub-clauses for the REBUILD clause are:

REVERSE -- This stores the bytes of the index block in reverse order, excluding the ROWID, when the index is rebuilt.

NOREVERSE -- stores the bytes of the index block without reversing the order when the index is rebuilt. Rebuilding a REVERSE index without the NOREVERSE keyword produces a rebuilt, reverse keyed index.

PARALLEL parallel_clause

NOPARALLEL -- This specifies that rebuilding the index, or some queries against the index or the index partition is performed either in serial or parallel execution.

LOGGING -- This specifies that the rebuilding of the index or index partition will be logged in the undo and redo log file. LOGGING further specifies that subsequent operations against the index or index partition will be logged in the undo and redo log file. This is the default.

If the database is run in ARCHIVELOG mode, media recovery from a backup will recreate the index. You cannot specify LOGGING when using NOARCHIVELOG mode.

Note: In future versions of Oracle, the LOGGING keyword will replace the RECOVERABLE option. RECOVERABLE is still available as a valid keyword in

Oracle

when altering or rebuilding non-partitioned indexes, however, it is not recommended. You must specify LOGGING if altering or rebuilding a partitioned index.

RECOVERABLE See LOGGING above. You cannot use RECOVERABLE for partitioned indexes or index partitions.

NOLOGGING -- This specifies that the rebuilding of the index or index partition will not be logged in the undo and redo log file. NOLOGGING further specifies that subsequent operations against the index or index partition that can execute without logging will not be logged in the redo log file. As a result, media recovery will not recreate the index.

When this option is used, index creation is faster than the LOGGING option because no redo log entries are written.

Note: In future versions of Oracle, the NOLOGGING keyword will replace the UNRECOVERABLE option. UNRECOVERABLE is still available as a valid keyword
in
Oracle when altering or rebuilding non-partitioned indexes, however, it is not recommended. You must specify NOLOGGING if altering or rebuilding a partitioned index.

UNRECOVERABLE -- See NOLOGGING above. You cannot use UNRECOVERABLE for partitioned indexes or index partitions.

TABLESPACE -- This specifies the tablespace where the rebuilt index will be stored. The default is the default tablespace of the user issuing the command.

RENAME -- This renames index to new_index_name. The new_index_name is a single identifier and does not include the schema name.

RENAME PARTITION -- This renames index partition_name to new_partition_name.

MODIFY PARTITION -- This modifies the real physical attributes, logging option, or storage characteristics of index partition partition_name.

UNUSABLE -- This marks the index or index partition(s) as unusable. An unusable index must be

rebuilt, or dropped and recreated before it can be used. While one partition is marked unusable, the other partitions of the index are still valid, and you can execute statements that require the index if the statements do not access the unusable partition. You can also split or rename the unusable partition before rebuilding it.

REBUILD PARTITION -- This rebuilds one partition of an index. You can also use this option to move an index partition to another tablespace or to change a create-time physical attribute.

DROP PARTITION -- This removes a partition and the data in it from a partitioned global index. Dropping a partition of a global index marks the index's next partition as unusable. You cannot drop the highest partition of a global index.

SPLIT PARTITION -- This splits a global partitioned index into two partitions, adding a new partition to the index.

Splitting a partition marked as unusable, results in two partitions, both marked as unusable. The partitions must be rebuilt before using them.

Splitting a usuable partition results in two partitions populated with data that are both marked as usable.

The sub-clauses for the PARTITION SPLIT clause are:

AT (value_list) -- This specifies the new non-inclusive upper bound for split_1. The value_list must compare less than the pre-split partition bound for partition_name and greater than the partition bound for the next lowest partition (if there is one).

INTO -- This describes the two partitions resulting from the split.

PARTITION split_1,

PARTITION split_2 -- These specify the names and physical attributes of the two partitions resulting from the split.

MERGE PARTITION -- While there is no explicit MERGE statement, you can merge a partition using either the DROP PARTITION or EXCHANGE PARTITION clauses.

The only way to merge partitions in a local index is to merge partitions in the underlying table.

If the index partition BULLDOGS is empty, you can merge global index partition BULLDOGS into the next highest partition, GOGSU, by issuing the following statement:

```
ALTER INDEX UGAS DROP PARTITION BULLDOGS;
```

If the index partition BULLDOGS contains data, issue the following statements:

```
ALTER INDEX UGAS DROP PARTITION BULLDOGS;
ALTER INDEX UGAS REBUILD PARTITION GOGSU;
```

While the first statement marks partition GOHSU unusable, the second makes it valid again.

<C>ORACLE7 ALTER INDEX Command

```
>--ALTER INDEX----------------- index------------------------------------------------------------------
>>
                 |-schema.-|         |-physical_attributes_clause--------------------------------|
                                     |                        V------------,--------------|   |
                                     |-ALLOCATE EXTENT (--------------------------------)-|
                                                              |--SIZE integer ------------|
                                                              |             |-K-|     |
                                                              |             |-M-|     |
                                                              |-DATAFILE 'file'-------|
                                                               |-INSTANCE integer----|

>-------DEALLOCATE UNUSED-------------------------------------------------------------------->>
                                   |--- KEEP integer ----------------|
                                                 |-K-|
                                                 |-M-|

>----REBUILD ------------------------------------------------------------------------------------><
                  |---PARALLEL integer ------------|
                  |  |-NOPARALLEL ----|          |
                  |---RECOVERABLE ---------------|
                  |  |-UNRECOVERABLE --|       |
                  |- TABLESPACE tablespace -----|
```

The parameters for the physical_attributes_clause are zero, one or more of the following:

PCTFREE integer

INITRANS integer

MAXTRANS integer

STORAGE storage_clause

Where the parameters are as described in the CREATE TABLE command with the exception of the REBUILD clause. The REBUILD clause allows rebuild of the specified index on-the-fly in 7.3.x and greater databases.

DROP INDEX Command:

```
>----- DROP INDEX -----------------index_name ----------------------------------------------------><
                   |- schema.-|
```

<C>CREATE SYNONYM Command:

```
>----CREATE ---------------- SYNONYM  synonym FOR ---------------object-----------------------
>>
             |-PUBLIC-|                              |-schema.-|      |-@database link-|
```

Where:

Synonym - is an allowed name (it cannot be the name of an existing object for this user. For purposes of uniqueness, the schema name is considered as a part of the name for an object.)

Schema.object - is an existing table, view, package, procedure, function or sequence name.

Database link - is an existing database link (covered later)

<C>DROP SYNONYM Command:

```
>----DROP ------------------ SYNONYM synonym------------------------->>
           |-PUBLIC-|
```

<C>CREATE SEQUENCE Command:

```
>------CREATE SEQUENCE ------------------sequence name ------------------------------------>>
                          |-schema.|              |-INCREMENT BY n---------|
                                                  |-START WITH n-------------|
                                                  |-MAXVALUE n -------------|
                                                  | |-NOMAXVALUE-|          |
                                                  |-MINVALUE n---------------|
                                                  | |-NOMINVALUE-|          |
                                                  |-CYCLE-----------------------|
                                                  | |-NOCYCLE (default)-|    |
                                                  |-CACHE n---------------------|
                                                  | |-NOCACHE-|             |
                                                  |-ORDER ----------------------|
                                                    |-NOORDER-|
```

Where:

Sequence name is the name you want the sequence to have. This

	may include the user name if created from an account with DBA privilege.
n	is an integer, positive or negative.
INCREMENT BY	tells the system how to increment the sequence. if it is positive the values are ascending, negative, descending.
START WITH	tells the system what integer to start with.
MINVALUE	tells the system how low the sequence can go. For ascending sequences it defaults to 1, for descending the default value is 10e27-1.
MAXVALUE	tells the system the highest value that will be allowed. for descending sequences the default is 1, for ascending sequences the default is 10e27-1.
CYCLE	This option causes the sequence to automatically recycle to minvalue when maxvaule is reached for ascending sequences, for descending sequences it will cause recycle from minvalue back to maxvalue.
CACHE	This option will cache the specified number of sequence values into the buffers in the SGA. This

speeds access , but all cached numbers are lost when the database is shutdown. Default value is 20, maximum value is maxvalue-minvalue.

ORDER	This option forces sequence numbers to be output in order of request. In cases where they are used for time stamping, this may be required. In most cases, the sequence numbers will be in order anyway and ORDER is not required.

<C>ALTER SEQUENCE Command:

```
>----ALTER SEQUENCE---------------- sequence name ---------------------------------------------
>>
                     |-schema. |                 |-INCREMENT BY n---------|
                                                 |-MAXVALUE n -------------|
                                                 | |-NOMAXVALUE-|          |
                                                 |-MINVALUE n---------------|
                                                 | |-NOMINVALUE-|          |
                                                 |-CYCLE-----------------------|
                                                 | |-NOCYCLE (default)-|      |
                                                 |-CACHE n---------------------|
                                                 | |-NOCACHE-|                |
                                                 |-ORDER ----------------------|
                                                   |-NOORDER-|
```

Only future sequence numbers are effected by this statement. To alter the START WITH clause, the sequence must be dropped and recreated. For ascending sequences, the MAXVALUE cannot be less than the current sequence value. For descending sequences, the MINVALUE cannot be greater than the current sequence value.

<C>DROP SEQUENCE Command:

```
>----DROP SEQUENCE----------------------- sequence name----------------------------------------->
                                   |-schema.-|
```

If triggers and procedures reference the sequence, these triggers and procedures will fail if the sequence is dropped.

<C>CREATE DATABASE LINK Command:

```
>----CREATE ---------------- DATABASE LINK link name----------------------------------------->>
                 |-PUBLIC-|
>----CONNECT TO user IDENTIFIED BY password--------------------------------------------->>
>-----USING 'connect string'----------------------------------------------------------------------------><
```

Where:

database link - Under version 6 this was a user specified name. Under ORACLE7 this is the GLOBAL db_name and the DB_DOMAIN. DBLINKS are schema independent.

PUBLIC - This is specified for database links that are to be used by all users. The user must have DBA privilege to specify a PUBLIC link.

CONNECT TO- This clause is used to force connection to a specific database user at the database being connected to. This allows the DBA to restrict access to confidential or sensitive information for one user instead of all users. If this clause isn't specified, the user's user name and password will be used in its place.

'connect string' - This is the protocol specific connection command.

For a version 2 SQLNET or NET8 connection the string would be:

sid|alias.domain

Where:

sid|alias - Either the actual SID for the database or the alias entered in the tnsnames.ora file for the platform or names server.

domain - This is the domain to which the instance belongs.

Connect strings are very system specific and DBAs should read all documentation provided by Oracle on their system's SQL*Net version before attempting to use them.

The database link would be used in the following manner:

```
SELECT * FROM emp@link;
```

The combination of table name and link can be placed into a single synonym for ease of use.

<C>DROP DATABASE LINK Command:

```
>-----DROP ------------------- DATABASE LINK dblink------------------>>
                 |-PUBLIC-|
```

<C>CREATE VIEW Command:

```
>--CREATE---------------------------------------VIEW --------------- view name (alias, alias,...)----
>>
                |--OR REPLACE--|                          |-schema.-|
                |--FORCE ----------|
                | |-NO FORCE-|     |
>--- OF ------------- type_name ----- WITH OBJECT OID --------------------------------------------
>>
           |-schema.-|                                          |-DEFAULT---------------------------|
                                                               |-( attribute, attribute, ...)------------|
 >--AS subquery---------------------------------------------------------------------------------------------
>>
>--WITH ----------------------------------------------------------------------------------------------->>
               |-CHECK OPTION-|   |-CONSTRAINT constraint-|
               |- READ ONLY-----|
```

Where:

view name - is the name for the view.

alias - is a valid column name, it isn't required to be the same as the column it is based on. If alias' aren't used, the names of the columns are used. If a column is modified by an expression, it must be aliased. If four columns are in the query, there must be four alias'.

subquery - This is any valid SELECT statement that doesn't include an ORDER BY or FOR UPDATE clause.

WITH CHECK.. - This clause specifies that inserts and updates through the view must be selectable from the view. This can be used in a view based on a view.

READ ONLY - This specifies that the view is READ ONLY and cannot be updated

CONSTRAINT- This specifies the name associated with the WITH CHECK constraint.

FORCE - This specifies that the view be created even if all permissions or objects it specifies as part of the view aren’t available. Before the view can be used the permissions or objects must be in the database and accessible.

NO FORCE (default) -- This means all objects and permissions must be in place before the view can be created.

OF object_type -- This explicitly creates an object view of type object_type. The columns in the object view are the same as the top level attributes of the specified object_type. Each row will have an assigned OID.

WITH OBJECT OID -- This specifies the attributes of the row that will be used as a key to uniquely identify each row of the object view, these should correspond to the primary key of the base table. If the base object has an OID already you can specify DEFAULT. This is only for ORACLE8.

a view can usually be used in the following commands:

COMMENT

DELETE

INSERT

LOCK TABLE

UPDATE

SELECT

<C>ALTER VIEW Command:

```
>--ALTER VIEW -------------view name COMPILE------------------------------------>>
                  |-schema. |
```

<C>DROP VIEW Command:

```
>---DROP VIEW ----------------view name----------------------------------------------->>
                   |-schema.-|
```

<C>CREATE TRIGGER Command:

```
>--- CREATE ----------------------- TRIGGER ----------------trigger----------------------------->>
              |-OR REPLACE-|             |-schema.-|          |- BEFORE ----------|
                                                             |-AFTER -------------|
                                                             |- INSTEAD OF ----|
>---DELETE-------- OF column -------------- table --------------------------------------------------->>
   |-INSERT---|    |-ON---------| |-schema.-| |-view--|
   |-UPDATE--|
>-- REFERENCING OLD--or--NEW AS old--or--new------------------------------------------>>
>--FOR EACH--ROW--------------------------------------------------------------------------->>
                    |-STATEMENT-|    |-WHEN (condition)--|
>---pl/sql block--------------------------------------------------------------------------------><
```

Where:

OR REPLACE replaces trigger if it exists, this can be used to change the definition of a trigger without dropping it first.

schema This takes the place of the owner

trigger This is the triggers name.

BEFORE This specifies the trigger is fired before the specified action.

AFTER This indicates the trigger is fired after the specified action.

INSTEAD OF This statement indicates that for this view the action be taken instead of the specified action.

DELETE / INSERT / UPDATE These are the specified actions, only one per trigger.

OF This limits the action response to the listed columns.

ON This specifies the table name of the table the trigger is for.

REFERENCING	This deals with correlation names. This allows specification of old and new values for a column.
FOR EACH ROW or STATEMENT	This forces the trigger to fire on each effected row, making it a row trigger. The trigger is fired for each row that is affected by the trigger and that meets the WHEN clause constraints. If STATEMENT is used, it makes it a statement level trigger.
WHEN	This specifies any constraints on the trigger. This is a trigger restriction clause. This contains a standard SQL clause.
pl/sql Block	This is the trigger body and is in the standard pl/sql format.

Database triggers are complex and if you do not save the creation script, it is very difficult to readily recall the exact command used in many cases.

<C>ALTER TRIGGER Command:

```
>--ALTER TRIGGER -----------------trigger------------------------------------------------------>>
                  |-.schema.-|         |-ENABLE-------------------------------|
                                       |--DISABLE-----------------------------|
                                       |--COMPILE----------------------------|
                                                          |-DEBUG-|
                    or:
>---ALTER TABLE ----------------table -------- ENABLE ----- ALL TRIGGERS---------->>
```

```
                     |-schema.-|              |-DISABLE-|
```

One limit on the usefulness of the ALTER TABLE in either disabling or enabling triggers is that is an all or nothing proposition. It is better to use the ALTER TRIGGER command unless you want all of the triggers on the table enabled or disabled at one time.

The DEBUG option instructs the PL/SQL compiler to generate and store the code for use by the PL/SQL debugger.

<C>DROP TRIGGER Command:

```
>--DROP TRIGGER ----------------trigger------------------>>
                     |-schema.-|
```

<C>CREATE FUNCTION or PROCEDURE Command:

```
>- CREATE --------------------- FUNCTION ------------- function --(argument, argument... )---->>
            |-OR REPLACE-|              |-schema.-|
>--- RETURN datatype -- IS ----------------------------------------------------------------------------><
                       |-AS-| |-- pl/sql subprogram body ----------------|
                             |--external_body ---------------------------|
```

An argument has the form:

```
argument_name ------------------------- argument_datatype
                |-argument_type-|
```

Where

argument_name -- This is the name given to this argument and can be any valid variable
name.

argument_type -- This is the type of argument: IN, OUT, or IN OUT and specifies how
the
argument is to be treated (strictly input, strictly output or both). This is optional
and will default to IN if not specified.

argument datatype -- This is the datatype of the argument and can be any valid scalar
datatype.

The external_body has the form:

```
|---EXTERNAL LIBRARY ----------------- library_name-----------------------------------------------
--|
|                                |-schema.-|                  |-NAME external_proc_name------------|  |
|                                                             |-LANGUAGE language_name----------|  |
|                                                             |-CALLING STANDARD--- C----------| |
|                                                                                        |-PASCAL-|  |
|--PARAMETERS (external_parameter_list) -------------------------------------------------------------
-|
                                                  |--WITH CONTEXT---|
```

The external_parameter_list has the form:

{{parameter_name [PROPERTY] | RETURN prop } [BY REF] [extern_datatype] | CONTEXT}
with the above repeated as many times as is needed.

prop has the values:

INDICATOR, LENGTH, MAXLEN, CHARSETID, or, CHARSETFORM

The command for creating a procedure is almost identical except there is no required RETURN clause. The CREATE PROCEDURE command can be used to create either internal standalone procedures or procedures that register calls to external procedures.

<C>CREATE PROCEDURE Command

```
>- CREATE -------------------- PROCEDURE ------------ procedure -(argument, argument... )----
>>
            |-OR REPLACE-|              |-schema.-|
>-- IS ----------------------------------------------------------------------------------------------------
><
  |-AS-| |-- pl/sql subprogram body ----------------|
        |--external_body ----------------------------|
```

An argument has the form:

```
argument_name ---------------------- argument_datatype
              |-argument_type-|
```

Where

argument_name -- This is the name given to this argument and can be any valid variable name.

argument_type -- This is the type of argument: IN, OUT, or IN OUT and specifies how the

argument is to be treated (strictly input, strictly output or both).

argument_datatype -- This is the datatype of the argument and can be any valid scalar

datatype.

The external_body has the form:

```
|---EXTERNAL LIBRARY ----------------- library_name------------------------------------------------
--|
|                          |-schema.-|             |-NAME external_proc_name------------|  |
|                                                  |-LANGUAGE language_name----------|  |
|                                                  |-CALLING STANDARD--- C----------| |
|                                                                       |-PASCAL-|  |
|--PARAMETERS (external_parameter_list) ------------------------------------------------------------
-|
                                          |--WITH CONTEXT---|
```

The external_parameter_list has the form:

{{parameter_name [PROPERTY] | RETURN prop } [BY REF] [extern_datatype] | CONTEXT}

with the above repeated as many times as is needed.

prop has the values:

INDICATOR, LENGTH, MAXLEN, CHARSETID, or, CHARSETFORM

For both procedures and functions the command arguments are listed below.

OR REPLACE - This optional statement specifies that if the procedure or function exists, replace it, it doesn't exist, create it.

schema - This is the schema to place the procedure or function into. If other than the user's default schema, the user must have CREATE ANY PROCEDURE system privilege.

procedure or function - This is the name of the procedure or function being created.

argument(s) - This is the argument to the procedure or function, may be more than one of these.

IN - This specifies that the argument must be specified when calling the procedure or function. For functions, an argument must always be provided.

OUT - This specifies the procedure passes a value for this argument back to the calling object. Not used with functions.

IN OUT - This specifies both the IN and OUT features are in effect for the procedure. This is not used with functions.

datatype- This is the data type of the argument. Precision, length and scale cannot be specified, they are derived from the calling object.

pl/sql body - This is a SQL, PL/SQL body of statements.

IS or AS - The documentation states that these are interchangable but one or the other must be specified, however, Oracle didn't tell this to some of their tools so if you get an error using one, try the other.

<C>ALTER FUNCTION or PROCEDURE Command:

```
>----ALTER-------------------------------------------------COMPILE-----------------------------------
>>
             |-FUNCTION--------------------function-|
             |                  |-schema.-|         |
             |- PROCEDURE-------------- procedure-|
                                |-schema.-|
```

Where:

COMPILE - This is the only possible argument and it forces a recompile of the object.

<C>DROP FUNCTION or PROCEDURE Command:

```
>-----DROP------------------------------------------------------------------------------------------->>
             |-FUNCTION--------------------function-|
             |                  |-schema.-|         |
             |- PROCEDURE-------------- procedure-|
                                |-schema.-|
```

This will invalidate any related functions or procedures and they will have to be recompiled before use.

<C>CREATE PACKAGE Command:

```
>---CREATE ------------------------------ PACKAGE --------------------- package--------------->>
                |--OR REPLACE --|                      |-schema.-|
>-----IS-- pl/sql package specification-------------------------------------------------------------><
    |-AS-|
```

Where:

OR REPLACE - This is used when the user wishes to create, or replace a package. If the package definition exists, it is replaced, if it doesn't exist, it is created.

schema - This is the schema the package is to be created in. If this is not specified the package is created in the user's default schema.

package- This is the name of the package to be created.

pl/sql package specification - This is the list of procedures, functions or variables that make up the package. All components listed are considered to be public in nature.

<C>CREATE PACKAGE BODY Command:

```
CREATE [OR REPLACE] PACKAGE BODY [schema.]package
IS--or--AS pl/sql package body;
```

Where:

OR REPLACE - When this is used if the package body exists it is replaced if it doesn't exist, it is created.

schema - This specifies the schema to create the package in. If this is not specified, the package body is create in the user's default schema.

pl/sql package body - This is the collection of all of the SQL and pl/sql text required to create all of the objects in the package.

<C>ALTER PACKAGE Command:

```
>---ALTER PACKAGE --------------package COMPILE--- PACKAGE------------------------->>
                  |-schema.-|                    |--BODY------|
```

<C>DROP PACKAGE Command:

```
>--DROP PACKAGE ---------------------------------- package----------------------------------><
                |-BODY-|      |-schema.-|
```

Exclusion of the keyword BODY results in the drop of both the definition and the body.

Inclusion of BODY drops just the package body leaving the definition intact.

When a package is dropped, all dependent objects are invalidated. If the package is not recreated before one of the dependent objects is accessed, ORACLE7 tries to recompile the package, this will return an error and cause failure of the command.

<C>CREATE SNAPSHOT LOG Command:

```
>---CREATE SNAPSHOT LOG ON ----------------table----------------------------------------->>
                                     |-schema.-|
>------------------------------------------------------------------------------------------------------->>
   |- WITH ------------------------------------------------------|
            |-PRIMARY KEY----------------|
            |-,ROWID--------------------------|
           |,filter column, filter column...--|
>------------------------------------------------------------------------------------------------------->>
        |-PCTFREE n ------------------|
        |-PCTUSED n-------------------|
        |-INITRANS n-------------------|
        |-MAXTRANS n----------------|
        |-TABLESPACE tablespace---|
        |-STORAGE ( storage clause)-|
```

Where:

schema	- This is the schema in which to store the log, if not specified, this will default to the user's own schema.

table	- This is the table name to create the snapshot log for.
WITH	- This specifies what is recorded about the source table in the log, PRIMARY KEY, ROWID or both. The columns listed as filter columns are used to tell Oracle that changes to these columns should be recorded.
PCTFREE PCTUSED INITRANS MAXTRANS	- These are the values for these creation parameters to use for the created log file.
TABLESPACE	- This specifies the tablespace in which to create the snapshot log. This will default to the user's default tablespace if not specified.
STORAGE	- This is a standard storage clause.

<C>ALTER SNAPSHOT LOG Command:

```
>---ALTER SNAPSHOT LOG ON ------------------table----------------------------------------->>
                                           |-schema.-|
>------------------------------------------------------------------------------------------------------>>
   |- ADD -----------------------------------------------------|
              |-PRIMARY KEY----------------|
              |-,ROWID--------------------------|
          |,filter column, filter column...--|
>---------------------------------------------------------------------------------------------------->>
        |-PCTFREE n ------------------|
        |-PCTUSED n-------------------|
        |-INITRANS n-------------------|
        |-MAXTRANS n----------------|
        |-STORAGE ( storage clause)-|
```

Where:

schema	- This is the schema in which to store the log, if not specified, this will default to the user's own schema.
table	- This is the table name to alter the snapshot log for.
ADD	- This adds additional logging options such as ROWID or additional filter columns to the snapshot log profile.
PCTFREE PCTUSED INITRANS MAXTRANS	- These are the values for these creation parameters to use for the created log file.
STORAGE	- This is a standard storage clause.

To change a snapshot log's location either an export/import or a drop/create of the snapshot log is required.

<C>DROP SNAPSHOT LOG Command:

```
>--DROP SNAPSHOT LOG ON------------------table--------------------------->>
                                |-schema.-|
```

<C>CREATE SNAPSHOT command.

```
>---CREATE SNAPSHOT --------------snapshot----------------------------------------------------->>
                          |-schema.-|
>--------------------------------------------------------------------------------------------------->>
   |-PCTFREE integer---------------------------------|
   |-PCTUSED integer--------------------------------|
   |-INITRANS integer--------------------------------|
   |-MAXTRANS integer------------------------------|
   |- TABLESPACE tablespace----------------------|
   |- STORAGE storage_clause----------------------|
   |- CLUSTER cluster (column [, column] ...)----|
  >--USING  ------------------------------------------------------------------------------------------>>
              |-INDEX ----------------------------------------------------------------|
              |           |- PCTFREE integer-----|                          |
              |           |- PCTUSED integer-----|                          |
              |           |-INITRANS integer ----|                          |
              |           |-MAXTRANS integer -|                             |
              |--- DEFAULT-- ROLLBACK SEGMENT--------------------------|
              |   |-MASTER--|                        |-rollback_segment-|  |
              |   |-LOCAL----|                                          |
>---REFRESH ---------------------------------------------------------------------------------------->>
               |-FAST-------------------------------|  |-START WITH date-| |-NEXT date-|
               |-COMPLETE---------------------|
                    |-FORCE----------------------------|
                    |-WITH -----------------------------|
                                |-PRIMARY KEY-|
                                |-ROWID------------|
>-------------------------- AS subquery------------------------------------------------------------------><
  |-FOR UPDATE-|
```

The keywords and parameters for the CREATE SNAPSHOT command are:

schema -- This is the schema to contain the snapshot. If you omit schema, Oracle creates the snapshot in your schema.

snapshot -- This is the name of the snapshot to be created.

Oracle chooses names for the table, views, and index used to maintain the snapshot by adding a prefix and suffix to the snapshot name. To limit these names to 30 bytes and allow them to contain the entire snapshot name, it is recommended that you limit your snapshot names to 19 bytes.

PCTFREE
PCTUSED
INITRANS
MAXTRANS These establish values for the specified parameters for the internal table Oracle uses to maintain the snapshot's data. For information on the PCTFREE, PCTUSED, INITRANS, and MAXTRANS parameters, see the CREATE TABLE command.

TABLESPACE -- This specifies the tablespace in which the snapshot is to be created. If you omit this option, Oracle creates the snapshot in the default tablespace of the owner of the snapshot's schema.

STORAGE -- This establishes storage characteristics for the table Oracle uses to maintain the snapshot's data.

CLUSTER -- This creates the snapshot as part of the specified cluster. Since a clustered snapshot uses the cluster's space allocation, do not use the PCTFREE, PCTUSED, INITRANS, MAXTRANS, TABLESPACE, or STORAGE parameters with the CLUSTER option.

USING INDEX --This specifies parameters for the index Oracle creates to maintain the snapshot. You can choose the values of the INITRANS, MAXTRANS, TABLESPACE, STORAGE, and PCTFREE parameters. For information on the PCTFREE, PCTUSED, INITRANS, and MAXTRANS parameters, see the CREATE TABLE command.

ROLLBACK SEGMENT -- This specifies the local snapshot and/or remote master rollback segments to be used during snapshot refresh.

rollback_segment -- This is the name of the rollback segment to be used.

DEFAULT -- This specifies that Oracle will choose which rollback segment to use.

MASTER -- This specifies the rollback segment to be used at the remote master for the individual snapshot .

LOCAL -- This specifies the rollback segment to be used for the local refresh group that contains the snapshot.

If you do not specify MASTER or LOCAL, Oracle uses LOCAL by default. If you do not specify rollback_segment, Oracle chooses the rollback segment to be used, automatically. If you specify DEFAULT, you cannot specify rollback_segment.

REFRESH -- This specifies how and when Oracle automatically refreshes the snapshot:

FAST -- This specifies a fast refresh, or a refresh using only the updated data stored in the snapshot log associated with the master table.

COMPLETE -- This specifies a complete refresh, or a refresh that re-executes the snapshot's query.

FORCE -- This specifies a fast refresh if one is possible or complete refresh if a fast refresh is not possible. Oracle decides whether a fast refresh is possible at refresh time.

If you omit the FAST, COMPLETE, and FORCE options, Oracle uses FORCE by default.

START WITH -- This specifies a date expression for the first automatic refresh time.

NEXT -- This specifies a date expression for calculating the interval between automatic refreshes.

Both the START WITH and NEXT values must evaluate to a time in the future. If you omit the START WITH value, Oracle determines the first automatic refresh time by evaluating the NEXT expression when you create the snapshot. If you specify a START WITH value but omit the NEXT value, Oracle refreshes the snapshot only once. If you omit both the START WITH and NEXT values, or if you omit the REFRESH clause entirely, Oracle does not automatically refresh the snapshot.

WITH PRIMARY KEY -- This specifies that primary key snapshots are to be created. Primary key snapshots allow snapshot master tables to be reorganized without impacting the snapshot's ability to continue to fast refresh.

Primary key snapshots can also be defined as simple snapshots with subqueries.

WITH ROWID -- This specifies that ROWID snapshots are to be created.

ROWID snapshots provide backward compatibility with Oracle7 Release 7.3 masters.

If you omit both WITH PRIMARY KEY and WITH ROWID, Oracle creates primary key snapshots by default.

FOR UPDATE -- This allows a simple snapshot to be updated. When used in conjunction with the

Replication Option, these updates will be propagated to the master.

AS subquery -- This specifies the snapshot query. When you create the snapshot, executes this query and places the results in the snapshot. The select list can contain up to 1000 expressions. The syntax of a snapshot query is described with the syntax description of subquery. The syntax of a snapshot query is subject to the same restrictions as a view query. For a list of these restrictions, see the CREATE VIEW command.

<C>ALTER SNAPSHOT Command:

```
>---ALTER SNAPSHOT ------------------snapshot------------------------------------------------------
>>
                              |-schema.-|
>---------------------------------------------------------------------------------------------------------------->>
   |-PCTFREE integer---------------------------------|
   |-PCTUSED integer---------------------------------|
   |-INITRANS integer--------------------------------|
   |-MAXTRANS integer-----------------------------|
   |- TABLESPACE tablespace----------------------|
   |- STORAGE storage_clause----------------------|
   |- CLUSTER cluster (column [, column] ...)----|
 >--USING  ----------------------------------------------------------------------------------------------->>
              |-INDEX -----------------------------------------------------------------|
              |           |- PCTFREE integer-----|                             |
              |           |- PCTUSED integer-----|                             |
              |           |-INITRANS integer ----|                             |
              |           |-MAXTRANS integer -|                                |
              |--- DEFAULT-- ROLLBACK SEGMENT--------------------------|
              |   |-MASTER--|                            |-rollback_segment-|  |
>---REFRESH ------------------------------------------------------------------------------------------->>
                |-FAST------------------------------|  |-START WITH date-| |-NEXT date-|
|-COMPLETE---------------------|
|-FORCE----------------------------|
|-WITH PRIMARY KEY--------|
```

Where:

schema	- This is the schema in which to store the log, if not specified, this will default to the user's own schema.
table	- This is the table name to create the snapshot log for.
PCTFREE PCTUSED INITRANS MAXTRANS	- These are the values for these creation parameters to use for the created log file.
STORAGE	- This is a standard storage clause.
REFRESH	- This specifies the refresh mode, either FAST, COMPLETE or FORCE. FAST uses a SNAPSHOT LOG, COMPLETE re-performs the subquery and is the only valid mode for a complex snapshot, FORCE causes the system to first try a FAST and if this is not possible, then a COMPLETE. FAST is the default mode.
START WITH	- This specifies the date for the first refresh.
NEXT	- This specifies either a date, or a time interval for the

next refresh of the snapshot.

Start with and next values are used to determine the refresh cycle for the snapshot. If just start with is specified only the initial refresh is done. If both are specified, the first is done on the start with date and the next is evaluated against the start with to determine future refreshes. If just the next value is specified it computes based on the date the snapshot is created. If neither is specified, the snapshot is not automatically refreshed.

<C>DROP SNAPSHOT Command:

```
>----DROP SNAPSHOT ------------------snapshot------------------------------------------------>>
                        |-schema.-|
```

When a snapshot is dropped, if it has a snapshot log associated with it, only the rows required for maintaining that snapshot are dropped. Dropping a master table upon which a snapshot is based doesn't drop the snapshot. However, any subsequent refreshes will fail.

<C>CREATE SCHEMA Command:

```
>--------------CREATE SCHEMA AUTHORIZATION schema ------------------------------------
>>
   V----------------------------------------|
>------------------------------------------------------------------------------------------------------>>
    |--CREATE TABLE command--|
    |--CREATE VIEW command----|
    |--GRANT command--------------|
```

Where:

schema	- This is the users schema, it must be their user name.
command	- This corresponds to the appropriate CREATE object command.

The individual create commands are not separated with a command terminator, the terminator is placed at the end of the entire create sequence of commands for the schema.

Schemas cannot be altered or dropped, only individual schema objects can be altered or dropped.

<C>CREATE LIBRARY commnad:

```
>--- CREATE ----------------------LIBRARY ------------- library name -------- IS ---- 'filename'--
><
               |-OR REPLACE-|           |-schema.-|                    |-AS-|
```

Where:

schema -- This is the schema in which the library is to reside. If not specified it defaults to the users default schema.

library name -- This is the name for the schema object and must comply with object naming standards.

filename -- This is the existing operating system shared library that is to correspond to the internal library name.

<C>ALTER LIBRARY Command:

```
>----- ALTER LIBRARY ----------------- lib_alias -----IS----- ‘filename’-----------------------------
>>
                              |-schema.-|             |-AS-|
```

Where:

lib_alias -- is the library name created with the CREATE LIBRARY commnad.

filename -- is the new file name for the shared library to be associated with this alias.

This command allows a library alias to be associated with a different source library. This does not invalidate dependencies. This command may not be available until later versions of ORACLE8. Although documented in certain documentation it was not available as of 8.0.2 for NT4.0.

<C>DROP LIBRARY Command:

```
>---DROP LIBRARY ----------------lib_alias------------------------------------------------------>>
                       |-schema.-|
```

The drop command only removes the library alias at the database level and does not effect the status of the operating system shared library.

<C>CREATE DIRECTORY command:

```
>>---- CREATE ---------------------DIRECTORY-------------directory----AS----'path name'---
>>
                    |-OR REPLACE-|
```

<C>DROP DIRECTORY Command:

```
>>---DROP DIRECTORY directory_alias---------------------------------------------->>
```

Once a directory is dropped all BFILEs in that directory location become inaccessible.

<C>CREATE CLUSTER command for both ORACLE7 and ORACLE8 is:

```
>--CREATE CLUSTER---------------cluster name----(column datatype, column datatype,....) ->>
                             |-schema.-|
>----------------------------------------------------------------------------------------------------------
>>
         |-PCTUSED n------------------------|
         |-PCTFREE n-------------------------|
         |-SIZE n -------------------------------|
         |          |-K-|                    |
         |          |-M-|                    |
         |-INITRANS n ------------------------|
         |-MAXTRANS n---------------------|
         |-TABLESPACE tablespace---------|
         |-STORAGE storage -----------------|
         |-INDEX –------------------------------|
            |-HASHKEYS n-----------------------|
                              |-HASH IS expr-|
>------PARALLEL-------------------------------------------------------------------------------------->>
      |-NOPARALLEL-|
>--CACHE –------------------------------------------------------------------------------------------>>
   |-NOCACHE-|
```

Where:

cluster name	is the name for the cluster, if the user has DBA privilege, a user name may be specified (user.cluster).
(column datatype, column datatype...)	is a list of columns and their data types called the cluster key. The names for the columns do not have to match the table column names, but the data types, lengths and precision's do have to match.
n	is an integer (not all of the n's are the same value, n is just used for convenience).
SIZE	This is the expected size of the average cluster. This is calculated by: 19 + (sum of column lengths) + (1 X num of columns) SIZE should rounded up to the nearest equal divisor of your block size. For example, if your block size is 2048 and the cluster length is 223, round up to 256.
storage	This storage clause will be used as the default for the

	tables in the cluster.
INDEX	This specifies to create an indexed cluster (default).
HASH IS	This specifies to create a HASH cluster. The specified column must be a zero precision number.
HASHKEYS	This creates a hash cluster and specifies the number (n) of keys. The value is rounded up to the nearest prime number.

The other parameters are the same as for the CREATE TABLE command.

<C>ALTER CLUSTER Command:

```
>--ALTER CLUSTER-------------- cluster name -------------------------------------------------------
>>
                    |-schema.-|           |-PCTUSED n --------------------------------------|
                                          |-PCTFREE n --------------------------------------|
                                          |-SIZE n ------------------------------------------------|
                                          |              |-K-|                                   |
                                          |              |-M-|                                   |
                                          |-INITRANS n -------------------------------------|
                                          |-MAXTRANS n ----------------------------------|
                                          |-STORAGE  storage -----------------------------|
>---ALLOCATE EXTENT--------------------------------------------------------------------------------
>>
                                    |--- SIZE integer ----------------------------------------------|
                                    |                      |-K-|                                  |
                                    |                      |-M-|                                  |
                                    |---DATAFILE 'filename'-----------------------------------|
                                    |---INSTANCE integer --------------------------------------|
>----DEALLOCATE UNUSED-------------------------------------------------------------------->>
                                         |-- KEEP integer ---------|
                                                                |-K-|
                                                                |-M-|
>----PARALLEL –------------------------------------------------------------------------------>>
   |– NOPARALLEL--|
```

The definitions for the above parameters are the same as for the CREATE TABLE, CREATE CLUSTER and storage clause definitions.

<C>ANALYZE CLUSTER Command:

```
>-ANALYZE CLUSTER------------ CLUSTER -----------------------------------------------------------
>>
                         |-schema.--|    |-COMPUTE-----------------------------STATISTICS--|
                                         |ESTIMATE STATISTICS SAMPLE n ----------------|
                                         |                                   |--ROWS ---|
                                         |                                   |PERCENT-|
                                         |-VALIDATE STRUCTURE -----------------------------|
                                         |                            |--CASCADE--|     |
                                         |-DELETE STATISTICS ----------------------------------|
```

Where:

COMPUTE STATISTICS - Calculates statistics based on all rows.
ESTIMATE STATISTICS - Calculates an estimate of the statistics based on "n" ROWS or PERCENT of rows in table.
VALIDATE STRUCTURE - Validates that the table and its indexes are consistent and not corrupted.
CASCADE - For clusters validates all tables in cluster.
DELETE STATISTICS - Removes current statistics.

<C>DROP CLUSTER Command:

```
>---DROP CLUSTER ----------------cluster name -----------------------------------------------------
>>
                   |-schema.-|          |-INCLUDING TABLES----------------------|
                                        |- CASCADE CONSTRAINTS--------------|
```

<C>Some Miscellaneous Thoughts:

In the scripts in this book you will see many more single commands than are covered in the above major commands. I show many examples of the use of SELECT, DELETE, INSERT and for the most part these are fairly self explanatory. You will also see SQLPLUS and PL/SQL commands, again, with a little thought these are pretty easy to manage. I suggest if you don't know SQL, rudimentary PL/SQL and SQLPlus that you crack the manuals or rev of the CD and get learing, this isn't a beginners guide to SQL, PL/SQL and SQLPlus. Afraid I can't do it all for you, some things you just have to do your self.

You should have the SQL, PL/SQL and SQLPlus references handy I also suggest downloading the Oracle Administrators and PL/SQL Developer demos from the http://www.revealnet.com/ website, they are good for several uses befoe they timeout and provide online refernces for all the commands you will see in this book and then some.