brainbuzz.com™
Cramsession

Cramsession™ for Oracle 8i Architecture and Administration

This study guide will help you to prepare you for the Oracle 1Z0-023 Exam. Exam topics include creating Oracle tablespaces, tables, indexes, roles, privileges and the effects of granting and revoking them as well as security, rollbacks and national language support.

Check for the newest version of this Cramsession
http://cramsession.brainbuzz.com/checkversion.asp?V=2451957&FN=oracle/8iAdmin.pdf

Rate this Cramsession
http://cramsession.brainbuzz.com/cramreviews/reviewCram.asp?cert=Oracle+8i+Admin

Feedback Forum for this Cramsession/Exam
http://boards.brainbuzz.com/boards/vbt.asp?b=807

More Cramsession Resources:

Search for Related Jobs
http://jobs.brainbuzz.com/JobSearch.asp?R=&CSRE

CramChallenge - practice questions
http://www.cramsession.com/signup/default.asp#day

IT Resources & Tech Library
http://itresources.brainbuzz.com

Certification & IT Newsletters
http://www.cramsession.com/signup/

SkillDrill - skills assessment
http://skilldrill.brainbuzz.com

Discounts, Freebies & Product Info
http://www.cramsession.com/signup/prodinfo.asp

Contents:

Oracle Architectural Components

An Oracle server contains an Oracle instance and an Oracle database. The Oracle instance is a set of memory structures and a group of background processes. The Oracle database is a set of files. ORACLE_SID is an O/S environment variable that is set to define the Oracle instance name. An Oracle instance can only reference one Oracle database at a time.

Memory Structures

SGA: System Global Area, sometimes call the Shared Global Area, is a group of shared memory structures.

- **The Shared Pool** - Comprised of the Library Cache (also know as the Shared SQL Area) and the Row Cache (also know as the Dictionary Cache or Data Dictionary Cache). Its size is specified by SHARED_POOL_SIZE variable in the parameter file.

  Library Cache

  - Contains the following for the most recently used SQL statement: Statement text, parsed code (parse tree), and execution plan.

  Row Cache

  - Also known as the data dictionary cache or dictionary cache. Contains the most recently used database definitions such as database files, tables, indexes, columns, users and privileges. The server process looks here during the parse phase to resolve object names and validate privileges.

- **Database Buffer Cache** - Contains an amount of database buffers as specified by the DB_BLOCK_BUFFERS initialization parameter in the size of DB_BLOCK_SIZE. Also contains the dirty buffer write queue, which holds dirty block buffers (i.e., already written to) until they can be written to disk. Least recently used(LRU) algorithm is used to decide which buffers to move out of the buffer cache to make room for new buffers to be read in.
- **Redo Log Buffer** - The size of the redo log buffer is specified by the LOG_BUFFER initialization parameter. Circular buffer contains information about changes made to the database. Records are written from the redo log buffer to the online redo log files by the LGWR process.
- **Java Pool** - This memory structure is used to store Java code.

- **Large Pool** - This memory structure is used to store large memory structures that are not specific to SQL statement processing. Data blocks that are copied during backup and restore operations can use this large pool to help reduce the I/O and fragmentation of the database buffer cache.

More About The Large Pool

Optional area within the SGA that provides large amounts of session memory for MTS and Oracle XA interface, I/O server processes, and backup and restore operations.

MTS improves performance due to more SQL stored in cache and less shrinking and growing of the SQL cache.

Large pool is able to fulfill the large memory requests of backup and restore operations and I/O server processes.

- Large pool does not have an LRU list.
- Specify the size of the large pool by setting the LARGE_POOL_SIZE init parameter.
- Accepts values followed by the suffix K or M.
- Default value is 0 (zero) unless parallel execution or DBWR_IO_SLAVES are configured.
- Min value is 600K. Max value is O/S specific.

When PARALLEL_AUTOMATIC_TUNING is set to TRUE parallel execution allocates buffers from the large pool, otherwise it allocates buffers from the SHARED_POOL.

If LARGE_POOL_SIZE is not set and PARALLEL_AUTOMATIC_TUNING is set to TRUE, Oracle will compute a value based upon PARALLEL_MAX_SERVERS, PARALLEL_THREADS_PER_CPU, PARALLEL_SERVER_INSTANCES, MTS_DISPATCHERS and DBWR_IO_SLAVES.

The POOL column in the V$SGASTAT view identifies which pool an object in memory resides in.

PGA (Program Global Area) - Used by a single server or background process. Is allocated and de-allocated when a user process is started and terminated. In a dedicated server configuration the PGA has and does the following:

- Area for sorting rows before returning them to a user process.
- Stages of cursor states for current session.

- Stores real variables instead of bind variables for executing SQL statements.
- The stack space is memory allocated to hold session variables and arrays.
- The user session data, privileges and performance info for the current user's session.
- The PGA is writeable and non-shared.

User session info is stored in the Shared Pool when Oracle is running in MTS (Multithreaded Server) mode. Otherwise user info stored in the PGA in dedicated server environment.

Background Processes

There are five mandatory processes.

PMON (Process Monitor)

- Cleans up abnormally terminated user sessions
- Rolls back any uncommitted transactions
- Releases locks held by a terminated process
- Frees SGA resources allocated to the failed process
- Restarts failed shared server and dispatcher processes

SMON (System Monitor)

- Performs automatic instance recovery

  Rolls forward transactions (txns) in the online redo log that have not been recorded on disk

  Open database and make unlocked data available

  Rollback uncommitted txns
- Reclaims (de-allocates) space used by temporary segments no longer in use
- Merges contiguous areas of free space in the datafiles. Known as defragging or coalescing.

DBWR (Database Writer)

- Writes all changed (i.e. dirty) buffers to datafiles.

- Uses a LRU (least recently used) algorithm to keep the most recently used blocks in memory.
- Defers writes for I/O optimization.
- Will write dirty blocks when one of the following occurs:
  1. The dirty list reaches a threshold value
  2. A process scans a specified number of buffers in the LRU list without finding a free buffer
  3. A time-out occurs
  4. A checkpoint occurs

Note that it is the server process that records changes to rollback segments and data blocks in the buffer cache!

LGWR (Log Writer) - Writes information from the redo log buffer to the redo log files.

- There is only one redo log writer per instance.
- A commit confirmation is not issued until the transaction has been recorded in the redo log file.
- Commits performed by other users before LGWR flushes the buffer on behalf of a user's commit are piggybacked to achieve an average of less than one I/O per commit.
- During very long transactions, the redo log buffer pool can become more than one-third full before LGWR writes it to the redo log file.

Will write redo log buffer entries to the redo log files when:

1. A commit occurs.
2. The redo log buffer pool becomes one-third full.
3. The redo log buffer has more has more than 1MB of recorded changes in it.
4. Before the DBWR completes cleaning the buffer blocks as during a checkpoint.

CKPT (Checkpoint Process) improves the performance of databases with many database files.

- Takes over LGWR's tasks of updating files at a checkpoint.
- Headers of datafiles and control files are updated at the end of a checkpoint.

- More frequent checkpoints will reduce the time required for instance recovery, but potentially at the cost of reduced system performance.

Other Oracle Background Processes

ARCH (Archiver Process) copies online redo log files to a designated storage device when LGWR switches to a new group.

- Copies redo log files to tape or disk for media failure recovery.
- Operates only when a log switch occurs.
- Is only needed when the database is running in ARCHIVELOG mode.

RECO (Recover Process) resolves failures involving distributed transactions.

LCKn (Lock Process) performs inter-instance locking in a parallel server system.

Pnnn (Parallel Query Process) performs: parallel query, parallel index creation, parallel data loading, and parallel 'CREATE TABLE AS SELECT...'.

SNPn (Snapshot Process) automatically refreshes snapshots (read only replicated tables). It is also responsible for the server job queues and replication queues.

S000...S999 (Server Processes)

- Parses and executes SQL statements.
- Reads data blocks from disk into the database buffer cache of the SGA.
- Returns the results of SQL statements to the user process.
- When a server process needs to read a data block from disk into the buffer cache it will:
  1. Search the LRU list.
  2. Look for a free buffer.
  3. Move dirty buffers to the dirty buffer list.

D001...D999 (Dispatcher Process)

- Used with MTS.
- SQL*Net Listener determines if each user process can use a shared server process.
- SQL*Net Listener gives the user process the address of a dispatcher process.
- The dispatcher process connects a user process to a shared server process.

- When the user process requests it, the listener will create a dedicated server process and connect the user process to it.

Oracle Database Files

Data Files

- Store the data dictionary
- User objects
- Before image data of current transactions

Redo Logs

- Record all changes made to the database and use them for data recovery.
- Written in a circular fashion.
- There must be at least two redo log groups.
- Each redo log group should contain the same number of members.

Control Files

- A binary file that describes the structure of the database.
- Required to mount, open, and access the database.
- Synchronization info needed for recovery is stored in the control file
- Recommended configuration is a min of two control files on separate disks.

Parameter File

- Used to size the SGA and locate the control files at instance start up.
- Contains all the database initialization parameters.

Password File

- Stores passwords for users with admin privileges.
- Used only with database authentication, not OS level authentication.

Archived redo log files

- Offline versions of the redo log files.
- Used for database recovery from media failure.

Alert File

- Logs all internal (ORA-600), block corruption (ORA-1578), and deadlock (ORA-60) errors.
- Logs all DDL and Server Manager commands such as STARTUP, SHUTDOWN, ARCHIVE LOG and RECOVER.
- Logs the values of all non-default initialization parameters at the time of database and instance startup.
- Located in destination specified by BACKGROUND_DUMP_DEST.
- Should be checked at least daily.

Trace Files

- Contains internal errors detected by server or background processes.
- If dumped by a background process the file is located in a destination specified by initialization parameter BACKGROUND_DUMP_DEST.
- If dumped by a server process the file is located in the destination specified by the initialization parameter USER_DUMP_DEST.
- Are created if SQL_TRACE parameter is TRUE.
- Are created if SQL_TRACE is enabled for a session.

Connecting to an Oracle Database

Users can be connected to an Oracle server in various configurations: a **direct host** based connection such as UNIX telnet session; a **two-tiered client-server** type connection such as pc-connection to a UNIX or NT server using SQLNet; or in a **3-tier configuration** where the client machine is connected to an application server which in turn is connected to the Oracle database server.

A **user process** is spawned on a client machine in a client server configuration, when a tool such as SQLPlus or a user-developed application is started on a client. User process includes the User Program Interface(UPI).

A **server process** is spawned that communicates with the Oracle Server on the host machine when a tool or application runs on the same machine as the Oracle

server(host based connection). The server process uses an area of shared memory described as the Program Global Area(PGA). Server process uses the Oracle Program Interface(OPI) to communicate with the Oracle server.

Query Processing

Parse

- SQL statement passed from user process to the server process.
- Check for existing copy of SQL statement in the shared pool.
- Check SQL syntax.
- Verify table and column definitions.
- Obtain parse locks.
- Validates user access to schema objects.
- Derive optimal execution plan.
- Load SQL and execution plan into shared SQL area.

Execute

- Parsed code is executed.
- Applies parse tree to data buffers.
- Performs physical reads.
- Performs constraint checking.
- Changes data if needed.
- For SELECT queries the server process gets ready to fetch the data.

Fetch

- For SELECT only.
- Returns dataset into BIND variables.

DML Processing

Parse and execute only, no fetch.

- Server process acquires data and rollback blocks into the buffer cache.

- Place exclusive row locks on rows that are about to change.
- Store record in the redo log buffer for before and after image.
- Save rollback data into a rollback segment block buffer.
- Applies changes to the database block buffer.

Commit processing

Fast Commit Mechanism - guarantees committed changes can be recovered (redo log files)

SCN - System Change Number. An SCN is assigned to each transaction. The SCN is unique within the entire database. Used as an internal time-stamp to synchronize data and provide read consistency. Is independent of the O/S date and time.

Commit Processing Steps

1. The server puts the commit record and SCN into the redo log buffer.
2. LGWR does contiguous write of all redo log buffer data up to and including the commit record to the redo log file.
3. User is notified of commit completion.
4. Server process indicates to Oracle that the transaction is complete and that locks can be released.

Advantages of fast commit

1. Sequential writes are faster than block writes to the data files.
2. Less data is required to write committed info to log files than writing to data files. Data files require whole data base blocks to be written.
3. Piggybacks log records from multiple transactions into a single log file write.
4. Usually only a single synchronous write is required per transaction, can be less.
5. Transaction size does not affect the length of the commit operation processing.

New Database Limits

Maximum database size: 512 petabytes (petabyte is 2 to the 50th power of bytes)

Maximum number of tablespaces: ~2 billion

Maximum number of datafiles per tablespace: 1022

Maximum number of partitions per table or index: 64,000

Maximum number of columns per table: 1000

Maximum columns per index: 32

New maximum size for CHAR: 2000 bytes

New maximum size for VARCHAR2: 4000 bytes

NCHAR and NVARCHAR2 are new multibyte national language support character equivalents of CHAR and VARCHAR2 and store the same amount of data respectively.

Size of Oracle 8 extended ROWID: 10 bytes

Size of Oracle 7 restricted ROWID: 6 bytes

Size of BLOB (binary large object type): 4GB

Size of CLOB (character large object type): 4GB

Size of NCLOB (NLS character large object type): 4GB

Getting Started with the Oracle Server

Features and Operations of the Universal Installer

- Java based engine
- Automatically resolves dependencies and uses complex logic handling
- Can install from a URL
- Implicit de-installation, the undo of install actions
- Supports multiple Oracle homes
- Can detect language of the O/S and will display test accordingly
- Can perform silent installation using response files

  On NT: setup.exe -responsefile <filename> -silent -nowelcome

  On UNIX: runInstaller -responsefile <filename> -silent -nowelcome

- Uses Oracle Optimal Flexible Architecture (OFA)

Operating System and Password File Authentication

The two automatically created DBA userid's are SYS and SYSTEM. Both are granted the DBA role.

The SYS userid default password is change_on_install and SYS is the owner of the database data dictionary tables and views.

The SYSTEM userid default password is manager and SYSTEM is the owner of additional internal tables and views used by Oracle tools.

Set up DBA Operating System Authentication

On **UNIX** Platform

- On UNIX user must be a member of the UNIX dba group.
- Set REMOTE_LOGIN_PASSWORDFILE parameter to NONE. In version 8.1.x or later the default for this parameter is EXCLUSIVE. Before 8.1.x is was NONE.
- CONNECT / AS (SYSDBA or SYSOPER)

On **NT** Platform

- Create a local users group called ORA_<SID>_DBA for a specific instance or RA_DBA for all sids.
- Add NT userid to this new group
- Add following line to sqlnet.ora file if not already there:

  SQLNET.AUITHENTICATION_SERVICES=(NTS)
- Set REMOTE_LOGIN_PASSWORDFILE parameter to NONE.
- CONNECT / AS (SYSDBA or SYSOPER)

Setup Password Authentication

- Create the password file using ORAPWD command. Usage is:

  orapwd file=<filename> password=<password> entires=<n>

  where <filename> on UNIX = orapw<sid>

  where <filename> on NT = pwd<sid>

- Set REMOTE_LOGIN_PASSWORDFILE parameter to

  EXCLUSIVE - only one instance can use the password file and the file contains other userids besides SYS and SYSTEM.

  SHARED - more than one instance can use the password file. SYS and INTERNAL are the only users known to the password file.
- V$PWFILE_USERS tells what users are in the password file.

Oracle Enterprise Manager and its Main Components

Is an enterprise wide unified management framework for the Oracle environment.

Utilizes intelligent agents that reside on the various managed nodes to provide for "lights out" database management.

Common services include jobs, events, discovery and security services.

Specialized applications include:

- DBA Management Pack
- Advanced Management Pack. Includes Tuning Pack, Diagnostics Pack and Change Management Pack.
- Application Management Pack. Includes support for Oracle Applications Concurrent Manager, WebForm Servers and Workflow subsystems.

OEM utilizes a 3-tier architecture

The first tier is the Java based UI console that can be run or installed from a web browser.

The second tier is the Oracle Management Server, which provides centralized intelligence and distributed control between clients and managed nodes, processing and administering all system tasks. The common services on the second tier are:

- Discovery Service
- Job Scheduling Service
- Event Service
- Security Service

The third tier uses the OEM repository and is composed of databases, nodes and other managed services.

Managing an Oracle Instance

Parameter File

Referred to as the init<SID>.ora file. If sid=orcl then file name = initorcl.ora . The file is read at time of system startup. Changes to the parameter file will take effect only after the database has been stopped and restarted.

Default Location in UNIX = $ORACLE_HOME/dbs. Default location in NT = %ORACLE_HOME%\database.

Using OEM you can view and modify init parms.

OEM stores "stored configurations" in its repository. In previous OEM versions stored configurations were kept in the NT registry. Must be connected via the OMS to use a stored configuration.

For the test, be sure and know the uses and rules for setting the various parameters that live in the init<sid>.ora file. You should know and be very familiar with the following parameters: BACKGROUND_DUMP_DEST, COMPATIBLE, CONTROL_FILES, DB_BLOCK_BUFFERS , DB_NAME, SHARED_POOL_SIZE, USER_DUMP_DEST, IFILE, LOG_BUFFER, MAX_DUMP_FILE_SIZE, PROCESSES, SQL_TRACE and TIMED_STATISTICS.

Database Startup Stages

NOMOUNT - The instance is started and the parameter file is read. SGA is created, background processes are started and trace and alter files are opened. Use for database creation.

MOUNT - Control file opened for this instance. Used for DBA maintenance activities including full database recovery, renaming datafiles, enabling or disabling redo logging. The DBA can alter the control file in this mode. Control file is opened and info read but datafile and redo log files are NOT verified in this step. Use for renaming data files, enabling and disabling archive log mode and for doing full database recovery.

OPEN - All files opened as described by the control file for this instance. Used for database access for all users. Datafiles and redo log files are verified and opened.

Instance Recovery

SMON will automatically perform instance recovery when opening up the database if the database crashed or was shut down using abort.

1. Rolls forward txns recorded in online redo log that have not been recorded in the data files.

2. Opens database before all roll forward txns have been applied.
3. SMON rolls back any uncommitted txns.

Database Shutdown Stages

CLOSE - Write buffer cache and redo buffer changes, close online redo log and data files. Control file remains open.

DISMOUNT - Control files are closed.

SHUTDOWN - Trace and Alert files closed, SGA de-allocated, background processes are stopped.

Additional Database Startup "Modes"

EXCLUSIVE - Only the current instance can access the database.

PARALLEL - Multiple instances can access the database (parallel server configuration)

RESTRICT - Restricted access for DBA. User must have RESTRICTED SESSION privilege to access the database.

RECOVER - For complete and incomplete recovery at instance start time.

FORCE - Forces the database to open. Similar to SHUTDOWN ABORT. Should always issue SHUTDOWN NORMAL and STARTUP NORMAL afterwards.

Database Shutdown Options

NORMAL - No new connections, waits for all users to disconnect, closes and dismounts database before instance shutdown, no instance recovery required upon next db startup.

IMMEDIATE - Rolls back all current uncommitted txns and disconnects users, closes and dismounts database before instance shutdown, no instance recovery required upon next db startup.

TRANSACTIONAL - No new connections, users disconnected when their current txn is completed, and, when all current txns are complete, shutdown immediate is executed.

ABORT - All txns are terminated, no txns are rolled back, instance terminated without closing the data files, next startup will require instance recovery.

Getting and Setting Parameter Values

Dynamic performance views are updated and managed by the oracle server, they contain data useful for tuning such as disk and memory structure/usage data and they start with the public synonym of V$. In the NOMOUNT stage V$ views read from memory can be accessed. Reading V$ views that read data from the control file require that the database be MOUNTED. The V$FIXED_TABLE view will display all dynamic performance views.

Some V$ views that contain data about the SGA and can be read in the NOMOUNT stage are: V$PARAMETER, V$SGA, V$OPTION, V$PROCESS, V$SESSION, V$VERSION, V$INSTANCE.

Some V$ views that contain data from the CONTROL FILE and can be read in the MOUNT stage are: V$THREAD, V$CONTROLFILE, V$DATABASE, V$DATAFILE, V$DATAFILE_HEADER, V$LOGFILE.

Once a database has been started use the following methods to see current parameter setting values:

1. SHOW PARAMETER - Displays all parameters and associated values in alpha order.
2. SHOW PARAMETER <value> - Displays all parameters containing <value> in their name.
3. Query the V$PARAMETER view using SQLPlus.
4. Use Instance Manager in OEM

Some parameters can be set and changed "on the fly" using the ALTER SESSION, ALTER SYSTEM or ALTER SYSTEM DEFERRED commands. The V$PARAMETER and V$SYSTEM_PARAMETER views have columns that will indicate which values can be altered using these commands. The column names are: ISSES_MODIFIABLE, ISSYS_MODIFIABLE and ISMODIFIED.

Know the difference between these three views/commands/fields and the scope and timing of the changes they affect.

Managing Sessions

The V$SESSION view contains sid and serial# needed to issue the ALTER SYSTEM KILL SESSION 'sid,serial#' command. The kill command will rollback the user's current txn, release any table or row locks the session is holding and free all resources being reserved by the client session.

Trace and Alert Files

Alert file contains a "log" of messages and errors. Alert file written while the Oracle instance is running. It records all db startups and shutdowns. Oracle will create an Alter file upon db startup if one does not exist. The location is defined by the BACKGROUND_DUMP_DEST parameter. Monitor the Alter file to detect internal and data block corruption errors, monitor database startup and shutdown operations and view the non-default values of other initialization parameters.

Trace files contain "dumps" created by background processes due to errors that occur during database processing. These trace file locations are also defined by the BACKGROUND_DUMP_DEST parameter.

Trace files can also contain session process information dumped at the request of the user; i.e., SQL_TRACE parameter. The USER_DUMP_DEST parameter dictates the location of these "user requested" trace files.

MAX_DUMP_FILE_SIZE limits the size of the user requested traces files. This value is specified in O/S blocks.

Creating a Database

Be sure you have actually gone through the process of creating at least five different Oracle databases, using varying logfile configurations and parameter settings!

A fully privileged and authenticated userid account must exist on the O/S. There must be enough memory and sufficient disk space to start the instance and create the physical database files.

Create a minimum of two control files located on separate physical disks.

Multiplex (place on separate physical disks) groups of redo log files.

Datafile locations should be based upon minimizing fragmentation(separate into different tablesapces based upon data life span; i.e., temp vs. perm), disk contention and separating objects (balance I/O by placing tables and indexes on separate disks).

Know the Oracle software and database file locations based upon the OPTIMAL FLEXIBLE ARCHITECTURE.

Be familiar with using the Oracle Database Configuration Assistant to create a database and the difference between all the choices it offers you.

Manual Database Creation

1. Determine unique value to use for instance and database name and what the database character set will be.

2. Establish required O/S system variables.
3. Create the parameter file - set DB_NAME, CONTROL_FILES and DB_BLOCK_SIZE at a minimum in this file. If the database name in the control file (create database <dbname>) does not = database name in parameter file (DB_NAME) instance will start but database will not MOUNT.
4. Create a password file.
5. Start the instance (NOMOUNT) - connect as SYSDBA and start up nomount. Use the PFILE clause in the startup command to specify a parameter file that is not in the default location.
6. Run the database creation script - KNOW ALL RULES ASSOCIATED WITH THE CREATE DATABASE COMMAND OPTIONS!!! - If REUSE is specified the file must exist, otherwise use the SIZE clause for files that don't already exist.

MAXLOGMEMBERS, MAXLOGFILES, MAXDATAFILES, MAXLOGHISTORY, and MAXINSTANCE actually allocate space in the control file. If these values are changed the control file must be recreated. When a create database command fails, shutdown the instance, manually delete all the files the unsuccessful create database command created, fix the errors and start again.

7. Run script to create data dictionary and other post creation tasks.

After database creation through step 6 above, the following exist:

1. System tablespace datafile(s)
2. Redo log and control files
3. SYS and SYSTEM users and their default passwords.
4. System rollback segment
5. Internal tables/no dictionary views.

Creating Data Dictionary Views and Standard Packages

The SYSTEM tablespace contains the data dictionary in two parts...base tables and views. **Base tables** are normalized and are created with the database by the **sql.bsq** script. The **views** simplify the base table info, are accessed via public synonyms and are created by the **catalog.sql** script.

The three view categories are USER_, ALL_, and DBA_. ALL_ and DBA_ usually contain an OWNER field. A user with select any table privilege can query the DBA_ tables. Be familiar with the difference between these three table types.

The DICTIONARY and DICT_COLUMNS views contain information about the data dictionary views and what columns they contain.

Stored Program Units

Can be called from SQL or PL/SQL. Types include PL/SQL program units, Java program units and C program units.

Benefits include:

- Reduce compile times
- Can call Java and C programs from SQL or PL/SQL
- Stored in shared pool to reduce disk I/O
- Security enforced by allowing access to data only through procedures and functions
- Single copy of program unit shared by multiple users during execution

A Stored PL/SQL Package is usually stored as two separate parts in the database

1. The specification, which is the interface
2. The body, which is the implementation of the specification.

Call a PL/SQL program unit within SQL*Plus by specifying the package name and the procedure using dot notation and parameters within parentheses:

EXECUTE package_name.procedure_name(parms...)

Oracle supplied packages include:

DBMS_SESSION, DBMS_UTILITY, DBMS_SPACE, DBMS_ROWID, DBMS_SHARED_POOL, and DBMS_LOB. Be familiar with the packages and some of the procedures they contain.

Query the DBA_OBJECTS view to get the status of a stored object. Use the DESC command to get the specification for all procedures and functions within a package.

sql.bsq - Creates the data dictionary base tables

catalog.sql - Creates the data dictionary views

catproc.sql - Sets up the ability to use PL/SQL and creates PLSQL packages to extend RDBMS functionality including views used for LOBS, tablespace, point-in-time-recovery and advanced queuing.

utl*.sql scripts - Create additional tables and views.

dbms*.sql - Creates package specifications and usually run during the execution of the catproc.sql script.

prvt*.plb - Provide the wrapped package code.

Database Event Triggers

These are new in Oracle8i. The database events that can fire a trigger are:

1. Instance startup or shutdown
2. Specific error msg or any error msg
3. A user logs in or out
4. A CREATE, ALTER or DROP statement gets executed on any schema

Maintaining the Control File

The control file is used to start up a database and to keep it operating successfully. Each control file is associated with a single database. The control file needs to be available for writing when the database is open because Oracle server updates it when the database is being used.

The control file contains:

- database name and identifying information
- data and redo log file locations and names
- tablespace names
- database creation and time stamps
- current log sequence number
- information about checkpoint
- log history
- Recovery Manager backup information

The control file is read when a database is mounted. If a control file specified by the CONTROL_FILES parameter can not be found the database will not be mounted.

The control file has two sections: 1) reusable 2) non reusable. Reusable section is written to in a circular manner similar to that for redo-logs.

Must recreate the control file to increase its size based upon new settings for parameters that affect its size.

Names and locations of all control files for a given instance can be found in V$PARAMETER or V$CONTROLFILE views.

For info on specific sections and values of parameters that affect the size of the control file query the V$CONTROLFILE_RECORD_SECTION view.

CONTROL_FILES parameter can specify up to eight files.

To add a control file:

1. Shut down the database
2. Use O/S to copy an existing control file to a new location under a new name.
3. Edit the parameter file CONTROL_FILES value to reflect the new file created in the previous step.
4. Start up the database.

Maintaining the Redo Log Files

Redo log files contain a record of all changes made to data blocks in the buffer cache. The redo logs are used for recovery in a situation such as instance failure.

Oracle server requires a minimum of two online redo log file groups to function properly.

Each member in a log group is assigned an identical log sequence number each time Oracle starts to write to a log group. Current log sequence number values are also stored in the control file and all data file headers in order to synchronize and insure data integrity.

MAXLOGFILES limit the number of online redo groups. It is dependent on the O/S.

MAXLOGMEMBERS limits the max number of members per group. It is dependent on the O/S.

The LOG_FILES parameter has been obsoleted in 8.1

Sequential writes to the current online redo log by the LGWR process occur when any of the following happen:

1. A commit occurs
2. Redo log buffer poll gets one-third full
3. There are more than 1MB of changed records in the redo log buffer
4. A LGWR timeout occurs - one every 3 seconds.

5. Before the DBWR process writes dirty blocks from the db buffer cache to the data files.

When a logswitch occurs a checkpoint starts.

During a checkpoint:

1. A number of dirty buffers in the db buffer cache that are written to data files by DBWR. FAST_START_IO_TARGET parm controls the number of buffers written at once.
2. CKPT updates the headers of all data files and it updates the control file to reflect the successful completion of the checkpoint.

A checkpoint occurs when:

1. A log switch occurs
2. Instance is shutdown normal, transactional, or immediate
3. As dictated by the LOG_CHECKPOINT_INTERVAL (specified as a number of O/S blocks. Checkpoint starts when the number of O/S blocks specified has been written by LGWR. If the value exceeds the redo log file size checkpoint will only occur at logswitch) and LOG_CHECKPOINT_TIMEOUT (specify as a number of seconds. If set to zero (0) disables timeout checkpoints) parameters. Or as specified by FAST_START_IO_TARGET parm.
4. Requested on the fly by the dba.

Set the LOG_CHECKPOINTS_TO_ALERT = TRUE to log all checkpoints to the alter file. Default value is FALSE.

To get redo log info:

1. ARCHIVE LOG LIST- server manager command.
2. V$DATABASE - name and log_mode.
3. V$INSTANCE - archiver. Can also use instance manager GUI.
4. V$THREAD - for group info such as current_group and log sequence number.
5. V$LOG - group#, members (count of), status, sequence# and bytes.

   Status values:

   UNUSED - The group has never been written to.

   CURRENT - The group is the current redo log group.

ACTIVE - The group is online and needed for instance recovery, but is not being written to.

CLEARING - Log file being recreated as empty after an ALTER DATABASE CLEAR LOGFILE command has been issued.

CLEARING_CURRENT - Current logfile is being cleared of a closed thread.

INACTIVE - The group is online, but is not needed for instance recovery

6. V$LOGFILE - group#, status, member number.

Status values:

INVALID - inaccessible

STALE - incomplete

DELETED - no longer being used

BLANK/NULL - file in use

Force a log switch - ALTER SYSTEM SWITCH LOGFILE;

Force a checkpoint - ALTER SYSTEM CHECKPOINT;

Three parameters help to control checkpoints:

LOG_CHECKPOINT_INTERVAL - Specified in OS blocks, not db blocks. Prior to 8.1 checkpoint started as soon as LGWR wrote the number of blocks specified by this parm. 8.1 and forward. The checkpoint position in the log cannot lag at the end of the log for more than the number blocks specified in this parm. Allows for a fixed number of redo-blocks that need to be read during instance recovery. Volume based.

LOG_CHECKPOINT_TIMEOUT - Prior to 8.1, specifies number of seconds transpired before another checkpoint occurs. In 8.1 and forward, the checkpoint position in the log cannot lag at the end of the log for more than the number of seconds specified in this parm. Allows for a fixed number of seconds of worth redo-blocks that need to be read during instance recovery. Time based.

FAST_START_IO_TARGET - 8.1 and Enterprise Edition only. The smaller the value the more frequent the checkpoints. Improves instance recovery time by forcing the DBWR to work harder.

To relocate a redo log file:

1. shut down database
2. perform O/S copy of redo log files to new location
3. startup mount
4. ALTER DATABASE RENAME FILE command
5. open database (alter database open)

To drop a redo log member:

ALTER DATABASE DROP LOGFILE MEMBER command. If in ARCHIVELOG mode, you cannot drop an online member until it has been archived. If group is active you must force a log switch before dropping one of its members. You cannot drop the last valid member of a group. Dropping a member does not remove its O/S file.

A number of redo log files depends upon your system usage. If you're seeing many waits on a log group in the alter log, you need to add another group. Place members of groups on different disks. Put online and archive log files on separate disks to reduce contention between LGWR and DBWR.

Checkpoints should occur more frequently than log switches.

If redo logs are big, increase checkpoints

- For checkpoints to occur as frequently as log switches:

set LOG_CHECKPOINT_INTERVAL > size of redo log file.

- For checkpoints to occur more frequently than log switches:

Deletes generate the more redo because the entire row needs to be stored. Inserts only need to store the ROWID in the redo logs.

set LOG_CHECKPOINT_INTERVAL = (((size of redo log file) / (db_block_size)) / desired frequency of checkpoint between log switch) * number of operating system blocks in one Oracle block.

The minimum size of an online redo log is 50k.

Can use the OEM Storage Manager GUI to relocate or rename or create redo log members.

When all members of the current or next group become inaccessible, the instance will shutdown.

Some Possible LGWR Errors include:

- A member of a group of two or more is not available
- No members from the next group are available
- No members from the current group are available

Use the Oracle LogMiner procedure to analyze redo-log files.

Runs in 8.1 or later and can analyze online and archived log files from oracle 8.0 and above to:

- Track changes to the database, specific table, or user level.
- Map data access patterns.
- Undo changes in the database.
- Perform tuning and capacity planning based upon archived data volume and activity.

Before Using LogMiner:

1. Set the UTL_FILE_DIR parm to a value that is allowed for PL/SQL file I/O and bounce the database so it will take effect.
2. Execute the DBMS_LOGMNR_D.BUILD procedure to create the dictionary file.

Setup V$LOGMNR_CONTENTS view:

1. create new list and specify first log file

EXECUTE DBMS_LOGMNR.ADD_LOGFILE ('file-name', DBMS_LOGMNR.NEW);

2. add additional files to the list

EXECUTE DBMS_LOGMNR.ADD_LOGFILE ('file-name', DBMS_LOGMNR.ADDFILE);

3. remove files from the list if desired

EXECUTE DBMS_LOGMNR.ADD_LOGFILE ('file-name', DBMS_LOGMNR.REMOVEFILE);

Start LogMiner Analyze Session

EXECUTE DBMS_LOGMNR.START_LOGMNR(<OPTION>);

View the V$LOGMNR_CONTENTS view to see results of LogMiner analysis. Only the session that performed the analysis can view the results.

Finish the Analysis

EXECUTE DBMS_LOGMNR.END_LOGMNR;

Other LogMiner Views

V$LOGMNR_DICTIONARY - dictionary file in use

V$LOGMNR_PARAMETERS - current LogMiner parm settings

V$LOGMNR_CONTENTS - contents of redo log files being analyzed

Managing Tablespaces and Data Files

This section is one of the bread and butter activities of an Oracle DBA. As such, you should be very familiar with all the related syntax and concepts. This section is not going to provide detailed syntax and descriptions of every command and option. You should have enough practice with these commands that they come natural to you.

Understand the syntax and all options associated with the CREATE TABLESPACE command. Some of the newer options are:

MINIMUM EXTENT - Every used extent size in the tablespace will be a multiple of this value.

LOGGING - By default, all objects within the tablespace will have redo written.

NOLOGGING - By default, all objects within the tablespace will NOT have redo written.

A new feature of tablespaces is that they can be defined as either **Dictionary Managed** or **Locally Managed**. Locally managed tablespaces perform their own extent management, as opposed to dictionary managed tablespaces where the extent mgmt is performed by the data-dictionary.

You can **not** change the extent management method of a tablespace after it has been created.

Locally managed tablespaces use bitmaps in each datafile to keep track of the status of their data blocks. There is a bitmap value for each block or group of blocks in the tablespace.

No rollback is generated as a result of extents being allocated or freed for reuse.

You **do not specify** NEXT, MINEXTENTS, MAXEXTENTS, PCTINCREASE or DEFAULT STORAGE for locally managed tablespaces.

Advantages and Rules for Locally Managed tablespace:

- Reduces space mgmt operations associated with maintaining rollback segments and updating the data dictionary.
- Takes away the need to coalesce free extents.
- Extent sizes can be managed by the system or all have the same size. Use the **UNIFORM** or **AUTOALLOCATE (system-managed)** options to specify. If using **AUTOALLOCATE**, you can **specify the initial extent size** and then **Oracle will determine the size of additional extents** (minimum and default additional extent size is 64K). When using UNIFORM, the default is 1M unless you specify a value. **All locally managed temporary tablespaces must have UNIFORM specified.**
- You can create the system tablespace EXTENT MANAGEMENT LOCAL as part of the CREATE DATABASE command. When **system tablespace is managed locally** you can still **create dictionary-managed tablespaces**. However, you **must create all rollback segments in locally managed tablespaces**.
- You can use EXTENT MANAGEMENT LOCAL as part of the CREATE TABLE command for permanent tablespaces, other than system.
- You can specify EXTENT MANAGEMENT LOCAL as part of the CREATE TEMPORARY TABLESPACE command.

Temporary Tablespaces

- Used during sort operations
- Won't contain any permanent records
- Usage of locally managed extents is recommended by Oracle
- It's recommended to use the CREATE TEMPORARY TABLESPACE command instead of the CREATE TABLESPACE command with the TEMPORARY option.

The temporary datafiles used by temporary tablespaces differ from regular data files in the following ways

- Tempfiles are always set to NOLOGGING mode
- They cannot be made read-only
- They cannot be renamed
- They cannot be created with the ALTER DATABASE command

- Not recovered during media recovery
- No info is generated by the BACKUP CONTROLFILE or CREATE CONTROLFILE commands

Know the storage clause inside and out. Some defaults

- Min size of INITIAL is 2*db_block_size. Default size of INITIAL = 5 db blocks.
- Min size of NEXT is one block. Default is 5 db blocks.
- The min and default for MINEXTENTS = 1.
- Min value of PCTINCREASE = 0 and the default is 50.
- Min value of MAXEXTENTS=1 and the default is a function of the db block size.

Know how to add a data file to a tablespace

ALTER TABLESPACE ...ADD DATA FILE

Know how to change the size of a data file manually

ALTER DATABASE...DATAFILE ...RESIZE...

Know how to use the ALTER TABLESPACE command to change the Default Storage clause.

Know how to take tablespaces offline.

- ALTER TABLESPACE <TS-NAME> OFFLINE;

  OFFLINE Options:

  NORMAL - performs a checkpoint for all datafiles in the tablespace.

  TEMPORARY - performs a checkpoint for just the online datafiles in the tablespace.

  IMMEDIATE - does not ensure that tablespace files are available and does not perform a checkpoint

  FOR RECOVER - tablespaces offline for tablespace point-in-time recovery.

- ALTER TABLESPACE <TS-NAME> ONLINE;

- These events are recorded in the control file.
- Cannot access any objects in a tablespace when it is offline.

Know how to make a tablespace Read-Only

- tablespaces must be online
- no active txns allowed prior to 8i
- 8i allows current txns to complete
- can not contain any active rollback segments
- can not be currently involved in any online backup

Know how to drop a tablespace

- if it contains data you must use the INCLUDING CONTENTS clause.
- after it's dropped, data is no longer in the database.
- you still need to remove the actual OS file, only the pointer is deleted
- read only tablespaces can be dropped.
- take offline before dropping to ensure there are no outstanding txns on any segments within the tablespace being dropped.

To move data files associated with a tablespace use either the ALTER TABLESPACE or ALTER DATABASE commands.

ALTER TABLESPACE <ts-name> RENAME DATAFILE from to;

Use for a non-system tablespace with no active rollbacks or temp segs.

Steps:

1. Place tablespace offline
2. Move or rename file using OS command
3. ALTER TABLESPACE RENAME DATAFILE command
4. Place tablespace online
5. Delete old file at OS level if needed

ALTER DATABASE ,DBNAME> RENAME FILE from to ;

Can use for any type of data file.

Steps:

1. shut down db
2. move or rename file using OS command
3. Mount db
4. ALTER DATABASE RENAME FILE command
5. Open db

Become familiar with the DBA_TABLESPACES, DBA_DATA_FILES,DBA_TEMP_FILES, V$DATAFILE, V$TABLESPACE and V$TEMPFILE views.

Storage Structure and Relationships

The logical structure of a database includes tablespaces, segments, extents, and data blocks. Do not store user data in the SYSTEM tablespace.

SYSTEM tablespace and a tablespace with an active rollback segment can NOT be taken offline.

Maxnumber of tablespaces per database is 64K. Number of tablespaces can not exceed the number of datafiles. Max number of datafiles per tablespace is 1023. MINIMUM EXTENT can only be specified for a tablespace, not the individual objects in a tablespace.

Types of segments

1. **Table Segment** - Unclustered or non-partitioned tables. All data in a table seg must exist in a single tablespace.

2. **Table Partition** - Data within a table can be stored in multiple partitions and each partition can exist in a different tablespace if desired. Each partition can have its own storage parameters. Helps to reduce I/O contention by distributing the table data. Requires the use of the Oracle8i Partitioning option.

3. **Cluster** - Rows are "clustered" together based upon key column value(s). A cluster can contain multiple tables and is a type of data segment. Objects in a cluster belong to the same segment and share the same storage values.

4. **Index Organized Table** - Data is actually stored in the index. All data is retrieved from index tree - no table lookup is needed.

5. **Index Partition** - Contain index partitions. A single index partition canNOT span multiple tablespaces. Helps to reduce I/O contention by distributing the index data. Requires the use of the Oracle8i Partitioning option.

6. **Rollback Segment** - Holds rollback data for undo changes, read-consistency and recovery.

7. **Temporary Segment** - Contain intermediate results data such as that used for sort operations.

8. **LOB Segment** - Large Objects such as pictures, documents or video clips. If the actual value placed in this column is large Oracle actually stores the data in separate LOB segments and the table segment will only contain a locator or pointer to the LOB segment.

9. **LOB Index** - Is automatically created when an LOB segment is created. Used to lookup LOB segment values.

10. **Nested Table** - A table that is actually stored in a column of a table. Also referred to as an inner table. Stored in its own segment separate from the "parent" table. Requires the Oracle8i Objects option.

11. **Bootstrap Segment** - Also called a cache segment. Created by sql.bsq at database creation time. Used to help initialize the data dictionary cache when db is opened. Cannot be queried or updated. No dba maintenance required. Resides in the SYSTEM tablespace and is owned by user SYS.

Storage Clause can be defined at tablespace or segment level. If defined at segment level, overrides tablespace level (except for MINIMUM EXTENT). If not defined at tablespace or segment level, Oracle uses default values. BE SURE AND KNOW THE DEFAULTS!!!! Any changes to storage parameters will only apply to new extents not yet allocated.

The data files, at the time of initial tablespace creation, associated with a tablespace contain a header block and one free extent that is made up of all the remaining free space. Lots of extent allocation and de-allocation can cause fragmentation in the tablespace data files. Extents require contiguous space when being allocated.

SMON coalesces free extents when the PCTINCREASE value is > 0. Use the DBA_FREE_SPACE_COALESCED view to see if any extents are available for coalescing. Use the ALTER TABLESPACE <tablespace name> COALESCE command to manually coalesce a tablespace.

Parts of a Database Block

Header - general block info such as block address and segment type. It is located at the top of the block and grows from the top down.

Free Space - Set of bytes in the middle of the block available for headers or row data.

Data Space - Stores table or index data. Has three components: 1) Row header = 3bytes. 2) Column data, of varying size. 3) Length bytes if cols < 250 =1, cols >= 250 = 3. Grows from the bottom up.

Space Mgmt Parameters. Used to control free space in a data segment:

INITRANS - Default 1 for data segment, 2 for indexed segment. Number of transaction entries allocated initially in the block header.

MAXTRANS - Default 255. Maximum number of transactions that can access the block concurrently.

PCTFREE - Default = 10% of spaces in each block reserved for future updates to a table's rows. When used on index segments, specifies an amount of space reserved for additional index entries. Grows from the top down.

PCTUSED - Default = 40% of used space for each data block. A block becomes a candidate for row insertion when it's used space falls below PCTUSED. Grows from the bottom up.

The following data dictionary views contain data about storage structures. Become very familiar with what data is in each:

DBA_TABLESPACES , DBA_DATA_FILES , DBA_FREE_SPACE, DBA_SEGMENTS, DBA_EXTENTS , DBA_SEGMENTS ,

Fragmentation Propensity by Object/Segment types

Data dictionary objects - None

Applications - Very Low

Data and Index - Low, but more frequently than the two above.

Rollback segments - High

Temporary Segments - high

Always try to isolate data based upon life span if possible.

Some other criteria noted by Oracle for organizing tablespace are

- Control space allocation by assigning usage limits to users
- Control availability by taking tablespaces offline and online
- Distribute data storage across disks to reduce I/O contention
- Perform partial backup and recovery operations
- Keep large volumes of static data in read-only tablespaces

Managing Rollback Segments

Rollback Segment Info:

Stores uncommitted and "pre-change" data and file and data block ids to support rollback, recovery and read consistency.

Contains a header that stores info about current txns using the rollback segment.

A txn can only use ONE RB segment.

Many concurrent txns can use the same RB seg.

At instance startup Oracle requires at least one rollback segment in the SYSTEM tablespace.

If a user is going to create tables in other tablespaces besides SYSTEM at least one more rollback segment is required.

Types of rollback segments:

System - Created in the system tablespace at time of database creation. Used for changes to objects in system tablespace only.

Private(default!!) - Exclusive to a single instance. Must be included in the parm ROLLBACK_SEGMENTS in the init<sid>.ora file or brought online manually to be recognized by the instance.

Public - Forms a pool of rollback segments that can be used by any instance. Useful with the Parallel Server option.

Deferred - Created when a rollback txn is executed against a tablespace that is offline. Created automatically by Oracle in the SYSTEM tablespace and no DBA maintenance is required.

Public and private rollback segments are the only ones that can be created by a DBA.

You need the **CREATE ROLLBACK SEGMENT** system privilege to create rollback segments.

Use the **SET TRANSACTION USE ROLLBACK SEGMENT <rb_seg>** command to assign a txn to a specific rb segment. Otherwise Oracle selects rb seg based upon the one that is available with the fewest number of txns in it.

RB segs contain extents and the extents are written and reused in a circular fashion. A "wrap" is when a single txn fills an extent and starts to write to the next extent in a rb segment. Multiple txns can write to a single rb segment extent, but only one txn can write to a rb segment block.

Rb segs extend themselves by adding additional extents when

1. The last extent in the "circle" is full and the first extent of the "circle" is not free.
2. An extent is full and the next extent in the circle is not available. The rb processing needs to always "wrap" to the next extent. It can not skip even if extents are free beyond the next one in the circle.

The **MAXEXTENTS** parameter controls the maximum number of extents that can be allocated for a rb segment.

The **OPTIMAL** parameter tells Oracle the size in bytes that an rb segment should shrink too upon completion of the txns that contributed to the growth of the rb seg. If specified the value must be equal to the min amount of space allocated by the **MINEXTENTS** parameter. Set optimal value based upon the amount of space needed by an average transaction.

RB Seg Creation Guidelines

- Set INITIAL and NEXT to the same value.
- Set OPTIMAL such that you reduce the amount of allocating and de-allocating occurring.
- Try not to set MAXEXTENTS to UNLIMITED. This avoids the use of excessive - amounts of disk space because of a program error.
- All rollback segments should be the same size. The PCTINCREASE storage parameter value is not allowed. It is always equal to zero (0) for rb segs.
- Requires a minimum of 2 extents of the same size.
- Always store them in their own tablesapce to avoid I/O contention! The TABLESPACE clause is optional in a CREATE ROLLBACK SEGMENT command.

If it is not included, the rollback segment will be created in the SYSTEM tablespace.

Rule of 4: (Total number of active transactions / 4) = the number of rollback segments to create for a database. If total < 8, round up to nearest multiple of 4. If over 50 use 50. Sorta dumb rule but you will may be quizzed on it.

The size of the rb segment depends upon the type of txns you have(insert, update, delete) and the actual data itself. Inserts require less rb data than deletes.

Oracle recommends a value of **20** for the **MINEXTENTS** parameter to reduce the need to allocate additional extents for a rb seg "on the fly".

Be familiar with the syntax to create, alter, shrink, and drop rollback segs and the use of the **MAX_ROLLBACK_SEGMENTS** parameter.

Know how to bring an rb seg online and offline. You should know all these things from actual experience! Note that an rb segment that is taken offline that still has active txns in it has its status changed to PENDING OFFLINE.

SCN (System Change Number): Each entry in a rollback segment is associated with an SCN. Allows rollback segment entries to be associated with a particular transaction.

Use the ALTER ROLLBACK SEGMEMENT command to:

- bring rb segs on or offline.
- change the OPTIMAL and/or MAXEXTENTS values.
- shrink a rb segs to a specific size

To change the INITIAL, NEXT or MINEXTENTS values for a rbseg you must drop and recreate it.

You must take a rbseg offline before you can use the DROP ROLLBACK SEGMENT command.

DBA_ROLLBACK_SEGS view contains rb segment identification, location, type, and status information.

V$ROLLNAME and **V$ROLLSTAT** dynamic views contain online rb segment statistics in use by an instance. Join on the usn column. The **OPTIMAL** value can only be seen in the **V$ROLLSTAT** view. If the current status of a rb seg is **PENDING OFFLINE** it shows as **ONLINE** in the **DBA_ROLLBACK_SEGS** view.

Use the **V$TRANSACTION** and **V$SESSION** views to see rb segment activity by current active txns. Join on saddr column.

For **OLTP** systems plan many smaller rb segs. Four txns per rbseg. Max ten txns per rb seg.

For **Batch** or **long running txns** plan fewer large rb segs. One txn per seg.

General Rollback Issues and Solutions:

ORA-01560- Insufficient space in tablespace - Extend (resize) datafiles, change to autoextend or add data files to tablespace.

ORA-01628 - Can't allocate additional extent. - Increase MAXEXTENTS value for rb seg or drop and recreate the rb seg with larger extents.

ORA-01555 Snapshot too old - Use higher MINEXTENTS, larger extent sizes, or higher OPTIMAL value. Can **not** avoid this error using MAXESTENTS.

Blocking txns can cause excessive extents to be created in a rb segment. Know how to detect these txns by querying the V$ROLLSTAT, V$SESSION and V$TRANSACTION views.

If you can not take a tablespace offline do the following:

1. Check DBA_ROLLBACK_SEGS to id rbsegs and then take them offline.
2. Check V$TRANSACTION to id txns using rb segs.
3. Check v$SESSION to obtain username and session info and terminate the sessions.
4. Take tablespace offline.

Related init parms:

TRANSACTIONS - Set to average number of active transactions at any given time.

TRANSACTIONS_PER_ROLLBACK_SEGMENT - Number of transactions to allocate to any given rollback segment.

The calculation (TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT) determines the number of rollback segments to be taken from the public pool.

wait ratio = (waits/gets) * 100. Evaluate V$ROLLNAME and V$ROLLSTAT. If wait ratio is > 1 this is not good.

Managing Tables

Regular Table - Default table type. Most common table. Rows stored in any order.

Partitioned Tables - Table has one or more partitions, each storing rows. Partitions can be range partitioned, hash partitioned or composite partitioned. Each partition of a table is an individual segment and can be stored in a separate tablespace. Useful for large tables. Has specific commands to manage the partitions.

Index-Organized Table - An **IOT** is a table that **stores its data in only an index**. Instead of having data and index storage structures, it contains only an index structure that contains both the index value and the data values. **Fast** key-based access for queries using exact match and range searches based upon the primary key in applications such as Information Retrieval, Spatial data and OLAP applications

IOT's can be reorganized with the ALTER TABLE statement MOVE clause.

Reduced storage requirements. Key columns are not duplicated in an index and a table.

CREATE TABLE command is used to create an IOT and an **ORGANIZATION INDEX** clause defines it as an IOT.

Example:

```
CREATE TABLE iot_table
( f1 char(20),
  f2 NUMBER,
  f3 NUMBER,
CONSTRAINT pk_iot_table PRIMARY KEY (f1, f2))
ORGANIZATION INDEX
TABLESPACE iot_tablespace
PCTTHRESHOLD 20
OVERFLOW TABLESPACE iot_overflow_tablespace ;
```

Must specify a primary key on an IOT using a column constraint (single column) or table constraint clause (multicolumn).

The ORGANIZATION INDEX qualifier defines the table as an IOT.

Optional row overflow specification

PCTTHRESHOLD – Amount of space, expressed as a percent, to save in each index block for an IOT row. Must be big enough to hold the primary key. Cannot specify on individual partitions of a partitioned IOT.

OVERFLOW TABLESPACE – tablespace location to store portions of rows that exceed the PCTTHRESHOLD value.

- Can **not use ROWID column datatype** on an IOT.

- **No unique constraints or LONG datatype columns**.
- If **partitioned,** can **not** contain LOB or varray type columns.
- If **non-partitioned can** contain **nested table column types**.
- **reorganize** with the **ALTER TABLE** command **MOVE** clause

Query the **DBA_TABLES view** and evaluate the **IOT_NAME and IOT_TYPE columns** to view information about IOT's. These fields are new to this table to support IOT's.

Clustered Table

- Comprised of a group of tables that share the same data blocks.
- Have a cluster key that identify the rows that should be stored together.
- The cluster key can be a composite key.
- The tables that are clustered must contain columns that correspond to the cluster key.
- Transparent to the application.
- Updating a cluster key column might cause a row to be physically moved.
- Cluster key is independent of a primary key.
- Use to improve performance for random access. Full table scans on clustered tables are usually slow.

Each row in a table has a row header and row data.

Row header contains number of cols in the row, chaining info and row lock status.

Row data, for **each column in a row**, the column length and data is stored. If column length <= 250 requires 1 byte to store column length. If column length > 250 requires 3 bytes to store column length.

Scalar Data Types

Fixed Length Character Types such as CHAR and NCHAR - Stored with padded blanks. Max value = 2000 bytes. Min value= 1 byte or 1 character depending upon the character set.

Variable Length Character types such as VARCHAR2 and NVARCHAR2 - Only use the number of bytes needed to store actual data. Max length is 4000 byes.

Numeric data - stored as variable length. Up to 38 significant digits. Require 1 byte for the exponent, 1 byte for every 2 digits in the mantissa, 1 byte for negative numbers if the number of significant digits is less than 38 bytes.

Date - fixed length. Requires 7 bytes. The date always includes the time.

Raw - Stores small binary data. There is no character set conversion between platforms or database using Oracle tools. 2000 bytes max.

LONG and LONG RAW - Remain for backwards compatibility. Not interchangeable with LOB data types. Max size 2GB. Only one per table. SELECT returns data. Data stored in-line. No object support. Sequential access to chunks.

CLOB - Large fixed width character data - Max size 4GB. Multiple columns per table. SELECT returns locator, data stored in-line or out-of-line(separate segments). Supports object types. Random access to chunks.

NCLOB - for large fixed width national character set data.

BLOB - unstructured data stored in the database. Binary large objects.

BFILE - unstructured data stored in an O/S file external to the database.

Since BFILEs are stored externally they do not require the LOB storage clause but instead need a directory object which tells Oracle where they are located in the file system. Directory objects are created with the CREATE OR REPLACE DIRECTORY statement and specify the operating system file path of the directory which contains the BFILE in question.

You need to make sure that Oracle has OS permissions to access a directory object. Oracle will let you create invalid directory objects (i.e., non-existent directories or directories for which it does not have permissions) and will not check their validity and will NOT create them for you, so check your typing carefully when creating them. You will only get an error down the line when Oracle tries to access the BFILE and cannot.

Creating directory objects which reference directories that contain the actual Oracle binaries, datafiles, control files, etcetera is a big security risk and is strongly discouraged by Oracle.

ROWID - pseudo column. Unique id for each row in the database. Fastest way to access a row. Stored in indexes. Requires 10 bytes of storage and is displayed using 18 characters. Comprised of DATA OBJECT NUMBER (32 bits), RELATIVE FILE NUMBER (10 bits), BLOCK NUMBER(22 bits), ROW NUMBER(16 bits). 80 bits = 10 bytes. Is displayed using base-64 encoding, uses "A-Z" , "a-z", "0-9", "+" , and "/" characters.

UROWID - Universal Rowid, new in 8i. Supports rowid's from foreign non-Oracle tables. Must set the COMPATIBLE parm to 8.1 or higher.

Restricted ROWID - Used prior to Oracle8. Requires 6 bytes internally. Does NOT contain the data object number

Collections Data Types

Objects that contains objects. Two types:

VARRYS - **Varying Arrays** - Store small lists. An ordered set. All elements are the same data type. Each element has a "positional" index (subscript) associated with it. The size of the array is equal to the number of elements it contains. Varray can "vary" in size, but max value must be specified at creation time.

NESTED TABLES - Define a table as a column within a table. Use to store large collection sets. Consists of unordered set of rows. All rows have same structure. The rows are stored separate from parent table and accessed via pointers in the parent table. There is no predetermined size of nested table.

LONG's vs. LOB's

There are four types of large object types, or LOBs: CLOB, NCLOB, BLOB and BFILE.

While CLOBs, NCLOBs and BLOBs are stored in tablespaces, BFILEs are pointers to binary files stored outside the actual database. Because of this BFILEs are read-only.

- Tables can contain only one LONG column and many LOB columns.
- A table containing a LONG cannot be partitioned, whereas with LOB's it can.
- Max size of a LONG is 2GB, max size of a LOB is 4GB.
- LONG's are read sequentially while LOB's can be randomly accessed.
- LONG's are stored in-line, which is within the column of the table. LOB's can be stored in-line or out-of-line. LOB's are made up of two parts, the locator and the actual LOB data. The Locator is stored in the column, but the LOB data can be stored in a separate tablespace. BFILES are pointers to external files.
- LONG's can not be defined as attributes in a user-defined object, while LOB's can.
- LOB bind variables can be defined.

Create LOB storage

There is an additional storage clause that defines out-of-line storage characteristics for LOB's.

Example:

```
CREATE TABLE docs
(doc_id         varchar2(50),
 doc            CLOB,
CONSTRAINT pk_docs PRIMARY KEY (doc_id))
STORAGE (INITIAL 32K NEXT 32K PCTINCREASE 0)
TABLESPACE doc1
LOB (doc) STORE AS
  (TABLESPACE doc2
    STORAGE (INITIAL 1M NEXT 1M PCTINCREASE 0)
                CHUNK 16K PCTVERSION 10 NOCACHE LOGGING) ;
```

Note that the LOB has its own tablespace defined for out-of-line storage.

CHUNK: number of dB blocks in a LOB "manipulation" chunk. Max value is 32K. Must be less than or equal to the NEXT value. It is a multiple of the dB block size and is the minimum unit when accessing LOB's. CHUNKs must be composed of contiguous blocks. On the other hand, CHUNKs need not be stored next to each other.

PCTVERSION: % of how much LOB storage needs to be changed before empty chunk space is reused. Older versions of LOB's will not be overwritten until at least this much of the storage has been consumed. 10 is the default.

(NO)CACHE (NO)LOGGING: specifies whether or not to load objects into the dB buffer cache and whether to turn on or off redo logging.

Oracle does not maintain data integrity for external files, BFILES.

Know the create table syntax!!!

Temporary Tables

- Hold session private data for the life of a transaction or session
- Create using the CREATE GLOBAL TEMPORARY TABLE <table-name>; command.
- No dml locks acquired or placed on temp table data.
- ON COMMIT DELETE ROWS means that rows live for the life on the transaction.
- ON COMMIT PRESDERVER ROWS means that rows live for the life of the session.
- Can create indexes, views and triggers on temp tables

- Can export and import the definitions of temp tables, but no temp table data is ever exported or imported. The definition is visible to all sessions.

PCTFREE:

- After PCTFREE is met the block is considered full and is not available for inserts of new rows. PCTFREE can also be specified for indexes.
- Free space is filled by inserting new rows, growth of existing rows, and by growth of data block header.
- A block is removed from the free list when free spaces drops down below the PCTFREE threshold.
- When specified for indexes, PCTFREE reserves space for both inserts and updates.
- Free space is calculated as a percent of DB_BLOCK_SIZE minus the header.
- Set a higher value when expecting columns to be updated from a null value or expecting columns size to increase from updates.

PCTUSED:

- When the percentage of a block being used falls below PCTUSED, either through row deletion or updates reducing column storage, the block is again available for insertion of new rows.
- Oracle adds the block to a free list when it finds that a block has dropped below its PCTUSED quota.

Setting a lower PCTFREE:

- Allows inserts to fill blocks more completely.
- May require fewer blocks to store data.
- Can increase processing costs if Oracle Server must frequently re-org blocks.
- Can cause row migration.
- Use if rows will not be updated much.

Setting a higher PCTFREE:

- Reserves more room for future updates.
- May require more blocks to store data.

- Lowers processing costs because of less block re-org.
- Reduces the need to chain rows.

Setting a lower PCTUSED:

- Reduces processing costs because blocks are not often free.
- Increases unused space.
- Use with high insert and delete activity.

Setting a higher PCTUSED:

- Increases processing costs because blocks may often become free.
- Improves space usage.
- Can reduce row migration by setting PCTFREE higher. Reduce row chaining by setting a bigger db_block_size or splitting tables into multiple smaller tables with less columns.

KNOW THE DIFFERENCE BETWEEN ROW MIGRATION AND ROW CHAINING.

Know the CREATE TABLE...AS SELECT command and all its options.

Know the syntax to manually add an extent to a table.

Can use the **ALTER TABLE <table-name> MOVE TABLESPACE <tablespace-name>;** command to **reorganize a non-partitioned table**. Good for

- moving a table to another tablespace
- reorganizing a table to eliminate row migration

After you move a table you must rebuild its indexes!!!

Table High Water Mark - indicates the last block ever used for the table. The high water mark is moved as data is inserted into the table and is not reset when rows are deleted. It's stored in the table segment head. A full table scans all the way to the high water mark.

You can use the **DBMS_SPACE package** to find the high water mark instead of analyzing a table. Faster and will not affect cost based optimization behavior. DBMS_SPACE is created by dbmsutil.sql and prvtutil.plb are run by catproc.sql.

ALTER TABLE ... DEALLOCATE UNUSED does not lower the high water mark. Releases space from unused extents. The KEEP clause indicates how much space to

KEEP above the high-water mark. If not specified, all unused space above the high-water mark is released. Use KEEP 0 to release all the space BELOW the high water mark.

TRUNCATE TABLE - deletes all rows and has these other results: no rollback data is generated (is a DDL command), associated indexes are truncated, delete triggers do not fire, DROP clause(default) causes all extents to be allocated, high water mark is reset to first block in the table and NEXT_EXTENT is set to the size of the lowest numbered extent that was de-allocated.

New 8i ability to drop a column from a table

ALTER TABLE <table-name> DROP COLUMN <col-name> CASCADE CONSTRAINTS CHECKPOINT nnn ;

Can specify a checkpoint value to reduce the amount of rollback needed.

Use ALTER TABLE <table-name> DROP COLUMNS CONTNIUE; command to resume an interrupted drop operation.

Restrictions:

- cannot drop a column from an object type table
- cannot drop columns from nested tables
- cannot drop all cols in a table
- cannot drop a partition key column
- cannot drop a col from a table owned by SYS
- cannot drop a parent key column
- cannot drop a primary key column from an index-organized table

Can mark a column as unused and then drop it a later time.

- is very fast to mark as unused.
- is ignored by queries as if it were not there.
- not displayed in the DESC command.
- column with same name can be added to the table.
- use to drop 2 columns from a single table and avoid 2 passes through table.
- Query the DBA_UNUSED_COL_TABS view to find unused columns.

ANALYZE TABLE command - collects table statistics and store them in the data dictionary for use by cost based optimizer. Also use to delete stats in the data dictionary, validate structure of a table, migrate ids or chain rows.

DB_BLOCK_CHECKSUM = TRUE will cause a checksum value to be written in the header of every data block when the block is written to disk.

The ESTIMATE STATISTICS uses a sample of rows (1064 by default)

COMPUTE STATISTICS bases values upon full table scan.

DBA_OBJECTS, DBA_SEGMENTS are views for getting table info. **DBA_TABLES** has the block usage and chaining info in it.

Updated after compute statistics run. Blocks column shows high water mark. **DBA_EXTENTS** for number, location and size of table extents.

DBMS_ROWID package - used to convert between the two ROWID formats and also to display their various components. This package is created by the dbmsutil.sql script, which is called from the catproc.sql script.

1. ROWID_CREATE - create a rowid from separate parts
2. ROWID_OBJCET - get object id
3. ROWID_RELATIVE_FNO - get relative file number
4. ROWID_BLOCK_NUMBER - get block number
5. ROWID_ROW_ID - get row number
6. ROWID_TO_ABSOLUTE_FNO - get absolute file number
7. ROWID_TO_EXTENTED - convert rowid from restricted to extended(oracle8)
8. ROWID_TO_RESTRICTED - convert from extended to restricted

Managing Indexes

Index Classifications

single column or concatenated (composite/multi) columns.

Unique or non-unique: Max number of columns allowed in a composite index = 32. Combined size of all index columns should not exceed about 1/3 the size of the data block size.

Function Based: Pre-computes the result of a function and stores its value in an index. Can be both B-tree or bitmapped index type. Good to use for queries that have functions in their WHERE clause. Can be B-Tree or Bitmap.

Expression can not contain aggregate functions, nested table, REF or LOB columns or, if an object-type contains a nested table, REF or LOB.

Must use cost-based optimizer and collect statistics for a functional index to be used.

Requires QUERY REWRITE or GLOBAL QUERY REWRITE privileges to create.

To enable function based indexes your must issue

ALTER SESSION SET QUERY_REWRITE_ENABLE = TRUE

And

ALTER SESSION SET QUERY_REWRITE_INTEGRITY = TRUSTED

Partitioned/Non-Partitioned: Use for large tables. Store index entries in several segments. Spread over multiple tablespaces. Reduces index lookup contention. Increases manageability.

B-Tree: Regular index. Can be normal or reverse order.

B-tree structure - root is top level, branch blocks below root, and leaf nodes, which contain the info to point to the rows in the table.

Leaf structure – contains entry header that stores number of cols and locking info and key column length-value pairs, which contain the size of a column and the value for the column. The ROWID of a row contains the key values. Key values are repeated if multiple rows have the same key value. There is no index entry for rows will all null key field values. Restricted ROWID format is used to reference rows of a table because all rows belong to the same segment.

Effect of DML on an index - Inserts add an index entry. Deletes result in a logical deletion. All entries in an index block must be deleted before the space is available for new index records. Updates result in a logical delete and an insert.

REVERSE KEY INDEX - Store the key value in reverse byte order. Use for inserting records on an ascending/sequential key. Helps to distribute the index values for faster query access. **Insertions** will be **better distributed** across all **leafs keys** in an index instead of clustered together, when index values are sequential.

Useful in queries using equality operators. Not good for ranges and searches. Helps to **improve performance** in oracle parallel server **(OPS)** when index changes are centered on a **small set of leaf blocks**.

Use when inserts are in ascending values and deletes are occurring to the lower values.

May help an **OLTP in OPS** in some cases.

BITMAP INDEX - Stores a bitmap instead of a key value. Uses B-tree structure and does contain the rowid. Good for indexes with a high number of rows and low

cardinality (few possible values such as Gender = M or F) of the key fields values. Good in queries that often use a combination of WHERE condition with the OR operator. Also good when a key value is static (not changed very often) or read-only.

The BITMAP index leaf node contains - entry header, key values (length value pairs), start ROWID, end ROWID, bitmap segment that id's if the row contains the key value. Updates to bitmap index are EXPENSIVE. More suited to DSS type systems and not OLTP type systems.

ALTER TABLE statement **no longer invalidates a BI**.

New ALTER TABLE statement RECORDS_PER_BLOCK clause that restricts the quantity of rows that can be stored in each dB block.

> **MINIMIZE** RECORDS_PER_BLOCK – Calculates the max number of records in any of the current dB blocks and sets that as a max that can be inserted into any new blocks. Improves query performance.
>
> - Can not specify if bitmap index already exists on the specified table.
> - Can not specify for an Index-organized table or nested table
> - Can not specify on an empty table.
>
> **NOMINIMIZE** RECORDS_PER_BLOCK – disables minimize. DEFAULT

Be sure you know the CREATE INDEX syntax and all the associated clauses.

Index Creation Guidelines

Indexes speed up query and slow down DML. Try to keep count of indexes low on tables that are changing frequently.

Store indexes in their own tablespaces, not tablespaces dedicated to rb segs, temporary segs, or data segs. Use a few standard extent sizes that are multiples of 5*db_dblock_size to help reduce index fragmentation. Try and use the nologging option for creating indexes on large tables to speed up index creation processing. Set initrans higher for indexes than tables. Use higher PCTFREE value for indexes that will contain values that are not created in sequential order.

Know the syntax for CREATE INDEX....REVERSE option. Note that the NOSORT option is not available for reverse key indexes.

Know the syntax to create a bitmap index. Note that a bitmap index cannot be unique. CREATE_BITMAP_AREA_SIZE parameter specifies the amount of space used for storing bitmap segments in memory. Default = 8MB. Larger value may increase index creation time. If cardinality is very low specify this value in K bytes. The higher the cardinality the more memory needed for best performance.

Know how to **ALTER INDEX** to **change storage parameters and block utilization parameters**. Know the ALTER INDEX...ALLOCATE EXTENT command.

ALTER INDEX ... DEALLOCATE will release unused index space **above** the high-water mark. Truncating a table results in truncation of any associated indexes. When an index is truncated all index space is de-allocated.

Rebuilding indexes

ALTER INDEX <index-name> REBUILD TABLESPACE <tablespace-name>;

Use to move an index to a different tablespace, remove deleted entries and improve space utilization, change a reverse key index to a normal b-tree index and vice versa. CANNOT USE TO CHANGE A BITMAP-INDEX TO NORMAL B-TREE OR VICE VERSA.

Rebuild can fix index fragmentation and improve performance, when an index has had many deletions.

During the index rebuild, the new index uses existing index as a data source. No sorting is required when building an index from an existing index. Better performance. The old index is deleted after the new index is created. Need space for both old and new index during the rebuild. Rebuilt index does not contain any deleted entries. Queries can use existing index while new one is being rebuilt/created.

Index rebuild has 3 phases:

1. prepare phase – short locks placed on the table
2. build phase – journal table populated
3. merge phase – unlocked journal rows deleted

Online Index Rebuild

ALTER INDEX <index_name> REBUILD ONLINE;

Can perform online index builds on partitioned, non-partitioned, and index-organized tables with b-tree indexes, using either of the following syntax:

During ONLINE index rebuild a DML SS-locks exist that **disallow any DDL tasks**.

Can perform DML. However, not recommended especially if a large percent of rows are involved.

Can not perform any parallel DML during an online index build.

Usually, **twice the amount of current index storage is required** for an online rebuild.

Indexes that can not be rebuilt on-line include cluster type, bitmap type and secondary indexes created on IOT's.

Analyze an index to check index block for corruption and to populate the INDEX_STATS view. syntax: ANALYZE INDEX ...VALIDATE STRUCTURE.

Reorg index if high number of deleted rows. If ratio of DEL_LF_ROWS/LF_ROWS > 30% reorg.

Drop and recreate indexes before bulk loads. Drop infrequently used indexes and create them when needed. Drop and recreate invalid or corrupt indexes.

DBA_INDEXES, DBA_IND_COLUMNS and DBA_OBJECTS views contain various index info. Know them well.

Maintaining Data Integrity

Data integrity is maintained by application code, database triggers, or declarative integrity constraints.

Database triggers are normally created to support complex business rules that cannot be defined as integrity constraints. DB triggers can be enabled or disabled.

Integrity constraints are the best method for enforcing business rules because they - are faster, easy to declare and modify, are centralized, are flexible, and are fully documented in the data dictionary.

There are five types of constraints. NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, and CHECK.

NOT NULL and CHECK constraints do directly require DBA attention.

Integrity constraints have four states:

1. **Disabled Novalidate** - Constraint is stored in data dictionary, but is not checked. Constraint is not checked for current data in a table or new data being added.
2. **Disabled Validate** - No changes are allowed on the constrained column. Index on the constraint is dropped. Allows the loading of a unique constraint from a non-partitioned table to a partitioned table via the EXCHANGE PARTITION option of the ALTER TABLE command.

When the **unique key is the same as the partition key**, DISABLE VALIDATE saves "overhead" and has no bad effects on the load. If the **unique key is not the same as the partition key**, table scans are done during the exchange. This results in increased processing time required and can offset the benefit of loading without an index.

3. **Enabled Novalidate** (enforced) - New data must meet integrity constraint. Existing data in table does not have to meet integrity constraint. Usually an intermediate stage to ensure all new data meets integrity check before being added to a table.
4. **Enabled validate** - All new and existing data must meet integrity checks.

Nondeferred or Immediate Constraints - enforced at the end of DML statement. Txn rolled back if constraint violated.

Deferred Constraints - Check only when a txn commits. Entire txn rolled back in violation at commit time. Useful for entering parent and child foreign keys values at the same time. Deferred constraints must be defined that way when they are created. Can be defined and changed on the fly to:

1. **INITIALLY IMMEDIATE** - by default should function as an immediate constraint unless set explicitly otherwise.
2. **INITIALLY DEFERRED** - by default constraint should be enforced at end of txn.

An application can use the ALTER SESSION or SET CONSTRAINTS commands to set constraints on the fly to IMMEDIATE or DEFERRED.

Implementing unique and non-unique index

- If constraint is disabled no index is required.
- If constraint is enabled and leading columns match an index, the index is used to enforce the constraint.
- If constraint is enabled and no index found for leading columns, an index with the same name as the constraint is created. If key is deferrable a non-unique index is created. If key is nondeferable a unique index is created.

Foreign Key Considerations:

If dropping a parent table use cascade constraints clause.

If truncating the parent tables disable or drop the foreign key constraints.

To drop a tablesapce that contains a parent table use the cascade constraints clause.

To avoid locks of a child table while doing DML on a parent table, create indexes on foreign key columns. Also has other advantages.

To perform DML on child table be sure the tablespace containing the parent key index is online.

Constraints can be defined in the CREATE TABLE command using **in-line** (as part of the column definition) or **out-of-line** (constraint clause as part of create table, but after the columns have been defined) **syntax**. Be very familiar with both types of syntax.

Must use **out-of-line** constraints when two or more columns are named in a single constraint or when a table is altered to add any constraint other than NOT NULL.

Can also define constraint in the ALTER TABLE command. Be sure you are familiar with all the various types of syntax and clauses!!!

Constraint Definition Guidelines

- Place primary and unique constraints indexes in separate tablespaces.
- Use non-unique indexes if bulk loads are frequent.
- If using self-referencing constraints either define the constraint after the initial load or define as deferrable.

Constraint Restrictions:

1. A NOT DEFERRABLE constraint can not be set to deferred using the SET CONSTRAINT command.
2. Can not specify DEFERRABLE or NOT DEFERRABLE using the ALTER TABLE...MODIFY CONSTRAINT command.
3. Must drop and re-create a constraint to change its deferability status.

Know the syntax and rules for enabling and disabling constraints.

Using Exceptions table and Identifying Row Violations

- Run **utlexcpt.sql** to create the **EXCEPTIONS table**.
- Enable constraint using the ALTER TABLE command and include the exceptions clause.

- Id invalid rows by querying the exceptions table.
- Fix the invalid rows in the table.
- Truncate exceptions table and re-enable the constraint.

Use **DBA_CONSTRAINTS** and **DBA_CONS_COLUMNS** views to see constraint info.

New or important columns in the DBA_CONSTRAINTS view:

DEFFERED	SEARCH_CONDITION
DEFFERABLE	R_OWNER
VALIDATED	R_CONSTRAINT_NAME
RELY	GENERATED
CONSTRAINT_TYPE	LAST_CHANGE

Loading Data

Direct-Load Insert - Used to copy data between tables living in the same database. By-passes the buffer cache and writes directly to the data files. Invoke using the APPEND hint. Available only with the INSERT INTO SELECT command. Can be used on partitioned and non-partitioned tables. Maintains indexes and all enabled constraints. Use LOGGING and NOLOGGING clause to control redo generation. All data is inserted above the high water mark of a table.

Parallel Direct-Load Insert - Allows for multiple concurrent parallel insert sessions to insert data into the same table. Use a PARALLEL hint in the INSERT statement or create or alter the table to specify the PARALLEL clause. Must enable parallel DML at the session level before txn starts...ALTER SESSION ENABLE PARALLEL DML.

Sequence of operations

1. Each session inserts its data into a temp segment.
2. The last extent in each temp segment is trimmed at the end of the session so that unused space is released.
3. At the end of all sessions, all the temp segs are combined into a single seg.
4. The new single temp seg is added to the table segment.

Restrictions:

1. Indexes are not maintained during the load. Drop indexes before load and recreate afterwards.

2. Referential integrity, check constraints and triggers need to be manually disabled and re-enabled.
3. Rows can only be appended. To replace the table, truncate the table before starting the load.

SQL-Loader - Load data from external files into the database. Has many features, be sure you are familiar with them.

Associated SQL-Loader files:

Control file - Contains input format, output tables and optional load logic.

Data files - External data files to be loaded.

Parameter file - Optional file that can be used to store input parameter values.

Bad file - Contains records rejected during load by SQL-Loader or failing database validation.

Log file - Contains a record of the load activities such as which files were read and the number of records loaded/rejected.

Discard file - Contains records that were not loaded because they failed optional "selection logic" included in the SQL-loader control file.

Two loading methods

Conventional Path Load

- Builds array of insert rows and uses the SQL INSERT command. Records are parsed and validated based upon field specifications. Can be used to load into clustered and non-clustered tables. Redo activity determined by logging attribute defined on the table being loaded.
- Uses commit to save changes. Enforces all constraints. INSERT triggers are fired. Can load into clustered tables. Table is not locked to changes by other users.

Direct Path Load

- Build blocks of data in memory and save the blocks directly into the extents allocated for the table being loaded. Bypasses the buffer cache and access SGA only for extent mgmt and adjustment of high water mark. No redo unless database in archive log mode. **Catldr.sql** script generates views required for direct path loads.
- Uses data-saves to save data. Only primary key, unique, and not null constraints enforced.
- No INSERT triggers fire.

- No loading into clustered tables.
- Table is locked to changes by other users.

Know the SQL-Loader command line keywords and in which cases the order of them matters.

Be familiar with trouble-shooting SQL*Loader errors.

*8i SQL*Loader Enhancements*

Can load objects, collections and LOB's.

Removal of the 64K physical record limit.

New FILLER keyword for filler fields.

New VARCHARC, VARRAW, LONG VARRAW and VARRAWC datatypes that function similar to VARCHAR.

Can specify a customer record separator.

Can AND together DEFAULTIF and NULLIF predicates.

Can specify a field delimiter that is longer than one character.

Reorganizing Data

Export and Import - Provide the DBA ability to move data between oracle db's (dev to prod, dev to test, etc...) (different oracle platforms or versions), within db's, between tablespaces, db users, and to reorganize data to reduce fragmentation and increase db performance. Provides for a logical backup.

Catexp.sql script creates views used by export/import utility.

Export modes

TABLE - Able to specify selected tables for export. All related indexes, triggers, constraints and grants can be exported for the specified tables. All users can export their own objects. Privileged users can export others selected tables.

USER - Export all objects owned for a given user. If a privileged user, the operation includes triggers and indexes created by others on the users tables. Unprivileged users can only export objects they own.

FULL - Exports all objects in database except objects owned by sys. Must be privileged user. A privileged user is one that has been granted the **EXP_FULL_DATABASE** roll.

Two types of Export methods:

Conventional Path - default method. Uses SQL SELECT statements to extract data. Data is passed to the export file from disk to buffer cache to evaluation buffer to export client process, which then performs the write. Exports data in the character set specified for the session.

Direct Path - Faster than conventional method. Data goes from disk to buffer cache to export process. No evaluation buffer is used and no block reorganization is performed to bring row pieces together. Exports data in the character set specified for the database. If using direct path method and session character set does not match the database character set an error will occur and the export will abort.

Be sure you know all the Export and Import command line parameters.

Import Order

1. Type definitions
2. Table definitions
3. Table data
4. Indexes on the table
5. Integrity constraints, views, procedures and triggers
6. Bitmap, domain and functional indexes

Guidelines

- Use a parameter file for commonly used command line options
- Don't use CONSISTENT=Y on a large volume of data or a database with lots of update activity on tables being exported.
- If there is a large amount of deleted rows do not use compress=y because it will result in the initial extent being allocated too large during importing.
- Allocate a large buffer
- Use direct path export if importing into 7.3.3 or greater.

NLS considerations

If the import session or import database character set differ from the export file's character set conversion is required and Oracle will perform it during the import.

The target character set must be a superset of the source character set to be sure all characters are converted. Oracle will replace unconverted values with the default "unknown" value of the target character set.

8i Export and Import Utility Enhancements

Can use a query in EXPORT to unload tables.

Can specify multiple dump files and work around the previous 2GB export file limit.

Can export tables with objects and LOB's, even in direct mode.

Can export and import pre-calculated optimizer stats and not needed to re-compute stats during import for selected exports and tables.

More control over the validation of type object identifiers during import.

Transportable Tablespaces

Use to copy or move tablespaces **between Oracle databases that have the same block size, character set, and are on the same hardware platform.**

Can be used to move data from an OLTP system to a DSS or data warehouse system.

Can be used to "publish" data on CDROM to be distributed and loaded into other Oracle databases.

Faster than export and import. Transporting a tablespaces requires only a copy of the datafiles and integrating the tablespace metadata into the data dictionary.

Can also transport indexes so they don't need to be rebuilt.

The tablespace must be fully "self-contained". This means that none of the following conditions may be true:

- There is an index inside of the tablespace set that is based upon a table that in not contained in the tablespace set.
- A partitioned table does not have all of its partitions located within the tablespace set. In other words, some of the tables partitions live in a different tablespace set.
- There is an LOB column in one of the tables within the tablespace set that points to LOB's outside of the tablespace set.

Perform the following to create a self-contained transportable set:

- ALTER TABLESPACE tablespace_name READ ONLY; for all tablespaces in the tablespace set.

- Use the EXPORT command with TRANSPORT_TABLESPACE=Y and TABLESPACES= ts1, ts2...ts*n*. Where ts1, ts2 are the tablespace names in the self-contained set. Use the TRIGGERS=Y/N, CONSTRAINTS=Y/N, GRANTS=Y/N, and FILE=filename parameters as well. Note that the FILE value specifies the name of the output file that will contain the structural information. Note that Y is the default setting for the Y/N value parameters specified.
- Copy the source tablespace data files to the destination machine.
- Transfer the export file created on the source system to the destination system.
- Use the import facility to update the data dictionary on the target system.
- Alter tablespace to turn-off read-only, if writing is desired.

When transporting **from an OLTP system to a data-warehouse environment** specify the **GRANTS = Y, TRIGGERS = Y, CONSTRAINTS = Y and TRANSPORT TABLESPACE = Y** options in the export command to be sure the objects and all associated table information is brought over. **However, if Oracle test asks you which two parameters you are most likely NOT to need, that is set to N, answer TRIGGERS and CONSTRAINTS.**

The **TRIGGERS export parameter** is **new to Oracle 8i** and is there to control the transport of TRIGGERS related to the transportable tablespace set.

Specify the **DATAFILES import** parameter to **name the datafiles** that **belong to the transferred tablespace being imported.**

If a BFILE column is part of a tablespace that is moved, you must manually copy the referenced files from the source to the destination computer.

Bitmap indexes and tables with VARRAY's or nested tables can not be transported.

Use the **DBMS_TTS.TRANSPORT_SET_CHECK** to verify a transportable set is self-contained. Populates the **TRANSPORT_SET_VIOLATIONS** view with any exceptions. This package is created by the **dbmsplts.sql** script, which is run by the **catproc.sql** script.

Can also use the **DBMS_TTS.ISSELFCONTAINED** function to validate a self contained transportable set.

Managing Password Security and Resources

Profiles are named sets of password and resource limits. Assigned by the CREATE/ALTER USER command. Can be enabled or disabled. Can be related to the default profile. Can limit resources at the session or "call" level.

A profile can limit the following

- CPU time
- I/O operations
- Idle time
- Connect time
- Memory space used
- Number of concurrent sessions
- Password aging and expiration
- Password history
- Password creation complexity verification
- Account locking

The **DEFAULT profile** is created by Oracle at the time of database creation. If not explicitly assigned a profile, all users will be assigned the default profile.

Some uses of Profiles

- Control use of resource intensive operations
- Force idle session logoffs
- Group users with similar resource usage
- Manage resources in complex and large multi-user systems
- Be able to control password usage

Some characteristics of Profile

- The current session is not affected by profile assignments
- Can only assign a profile to a user NOT A ROLE OR OTHER PROFILES
- The DEFAULT profile is created by Oracle at the time of database creation. If not explicitly assigned a profile, all users will be assigned the default profile

Password Mgmt Features

Account Locking - Can lock a user out of the system after a limited number of failed login attempts.

Aging and expiration of passwords - Can assign a lifetime to a password and force it to expire and be changed.

History of used passwords - Prevent reuse of previous passwords.

Verification of password complexity - Can control the values used for creating a password.

Enable password management by creating a profile that specifies password related limits and then assign the profile to user(s) via the CREATE USER or ALTER USER commands.

If a user **is** logged into the system, changes to passwords, expiration dates, and locks do not take effect until they logout.

Password Related Limit Settings that can be used in a profile

FAILED_LOGIN_ATTEMPTS - number of unsuccessful login attempts before locking out the user.

PASSWORD_LOCK_TIME - Number of days a lock stays in place after a password expires. To set to partial days use the following:

1/24 for 1 hour

10/1400 for 10 minutes

5/1440 for 5 minutes

PASSWORD_LIFE_TIME - Number of days a password stays effective before expiring.

PASSWORD_GRACE_TIME - Number of days "grace period" for creating a new password after first login after password has expired.

PASSWORD_REUSE_TIME - Number of days before a password can be reused.

PASSWORD_REUSE_MAX - Max number of times that a password can be reused.

PASSWORD_VERIFY_FUNCTION - PL/SQL function used to verify the complexity of a password before allowing it to be used as a valid password.

Function must be created in the SYS schema and follow these specific guidelines:

function_name (userid_parameter IN VARCHAR2 (30), password_parameter IN VARCHAR2(30), old_password_parameter IN VARHCAR2(30)) RETURN BOOLEAN

Oracle provides a password complexity function that is created by the utlpwdmg.sql script that must be run as the SYS user.

Oracle default password verify function characteristics:

- Min length is 4 characters
- Password not equal to username
- Must have at least 1 special character, 1 numeric value, and 1 character value
- Must differ from previous password by at least 3 characters

Know the ALTER PROFILE and DROP PROFILE CASCADE commands.

To control usage of resources by using profiles

1. Create the profile (CREATE PROFILE command)
2. Assign profiles to users (CREATE/ALTER USER commands)
3. Enable resource limits (ALTER SYSTEM command or RESOURCE_LIMIT init parm)

The **RESOURCE_COST** view contains the weights assigned to various resources.

Session level resource settings

CPU_PER_SESSION - hundredths of seconds

SESSIONS_PER_USER - concurrent sessions per username

CONNECT_TIME - minutes of elapsed connect time

IDLE_TIME - minutes of inactive time

LOGICAL_READS_PER_SESSION -logical and physical data block reads

PRIVATE_SGA -bytes in the SGA. Only for MTS

Call level resource settings

CPU_PER_CALL - hundredths of seconds

LOGICAL_READS_PER_CALL - number of data blocks read per call

To enable resource limits set **RESOURCE_LIMIT=TRUE** or **ALTER SYSTEM SET RESOURCE_LIMIT=TRUE**.

ALTER SYSTEM command

Settings made with this command stay in effect until altered again or the database is shutdown.

Use this method to enable enforcement when db can not be shutdown.

The CREATE PROFILE system privilege is needed to create a profile. Oracle creates a default profile when the database is created.

Use ALTER PROFILE command to change system limitations. Changes to profiles do not affect the current session. The ALTER PROFILE system privilege is required to alter a profile.

'DROP PROFILE CASCADE' must be used if the profile has been assigned to a user.

Default profile:

- Assigned as default to user if profile is not explicitly assigned.
- All unspecified limits for any profile have the default profile value.
- Initially, all defaults are unlimited.
- Can be altered so that no user can have unlimited use of resources by default.

Use ALTER USER command to assign a profile to an existing user.

Composite limit: Weighted sum of CPU_PER_SESSION, CONNECT_TIME, PRIVATE_SGA, and LOGICAL_READS_PER_SESSION. Use to limit resources for a session.

Use the **ALTER RESOURCE COST** command to specify weights for each session resource limit.

Data dictionary views related to profiles:

DBA_USERS - expiration and locking dates and account status.

USER_RESOURCE_LIMITS

DBA_PROFILES - display password profile information.

RESOURCE_COST - display weights assigned to each resource.

Managing Users

Users can be authenticated by the database, the operating system or the network.

SYS and **SYSTEM** are reserved user names in Oracle.

A **schema** is a collection of database objects such as tables, indexes, etc. that are associated or owned by a particular user.

The following can be associated to a user in Oracle

- Tablespace Quotas
- A Default Tablespace
- A Temporary Tablespace
- Account Locking
- Resource Limits
- Direct Privileges
- Role Privileges

Be familiar with the CREATE, ALTER and DROP USER commands.

A **CREATE USER** statement can have multiple **QUOTA** clauses, each for different tablespaces. A quota on a tablespace is required to create database objects in it. Inserts, updates and deletes are controlled by privileges, not quotas.

Only the owner of an object needs a usage quota, and even then only when the object is created or grows.

Users accessing objects owned by others do not need usage quotas.

It may be required to modify a users quota under the following circumstances

- the users tables start to grow a lot
- an application requires additional tables or indexes
- if objects are reorg'ed and moved to different tablespaces

Using the **IDENTIFIED EXTERNALLY** option on the CREATE USER command means that a user is authenticated by the O/S.

Set the **OS_AUTHENT_PREFIX** for O/S authenticated users. Default value is OPS$. Setting this to null, by placing two double quotes next to each other, will force the user to be authenticated by the O/S.

Setting **OS_AUTHENT_PREFIX=OPS$** and creating a user with:

CREATE USER ops$user IDENTIFIED BY password...

allows the user to login locally to the Oracle server without a password or login in remotely and use a password.

Users that are correctly set up with O/S authentication will be able to login to SQLPlus as follows: SQLPLUS /

Setting **REMOTE_OS_AUTHENT=TRUE** allows a user to be authenticated by a remote operating system.

Be sure SYSTEM is not the default or temporary tablespace of any user!!! If the DEFAULT TABLESPACE or TEMPORARY TABLESPACE clauses are not used, SYSTEM is the default for both of these.

The **ALTER USER system privilege** is required to issue the 'ALTER USER' SQL command. This includes assigning a profile or default role to an existing user.

If you set a quota of zero on an existing tablespace, the objects owned by users remain in the revoked tablespace. However, the objects cannot be allocated any new space.

The 'ALTER USER' command can change user profile and roles.

Must have **ALTER SYSTEM privilege** to kill an active user's session.

The DBA can lock a user account when changing a user password by explicitly locking the user account.

ALTER USER bill IDENTIFIED BY billw PASSWORD EXPIRE - will expire a user's password

ALTER USER bill IDENTIFIED BY billw ACCOUNT LOCK - will lock a user's account

ALTER USER bill IDENTIFIED BY billw ACCOUNT UNLOCK; - unlocks a locked users account.

The three commands above are enforced the next time the user attempts to login.

A user can not be dropped when they are connected.

DBA_USERS view contains basic user info such as user status and the default and temporary tablespaces for every user.

The **DBA_TS_QUOTA** view contains tablespace quota information. USER_TS_QUOTAS contain tablespace info that can be accessed by an individual user.

Managing Privileges

System Privileges - Allow particular actions or class of actions by users in the database. Include creating, dropping, altering tables views, rollback segs, and procedures.

Object Privileges - Allows users to perform a particular action on a specific object (table, view, sequence, stored procedure, function, or package).

SYSTEM PRIVILEGES

- Currently about 126 exist.
- The ANY keyword gives the user the privilege in every schema.
- The GRANT command adds a privilege to a user or a group of users.
- REVOKE x FROM PUBLIC revokes the system privilege or role from all users.
- Privileges can be classified as those enabling system-wide operations(CREATE SESSION , CREATE TABLESPACE), those enabling management of a user's own schema objects(CREATE TABLE), and those enabling management in any schema (CREATE ANY TABLE).
- Includes the ANALYZE and AUDIT capabilities.
- Not specific to a named schema, object or structure. Specific to a particular operation or class of operations.
- SELECT ANY TABLE gives the user the right to query any table in the database.
- Can be granted with the **WITH ADMIN OPTION**. This allows the person receiving the privilege or role to grant it to another user or role. The grantee can also alter or drop the role if granted WITH ADMIN OPTION. The GRANT ANY ROLE system privilege allows a user to grant any role in the database.
- A role is a named group of privileges that are granted to users and other roles.
- System privileges **SYSDBA and SYSOPER cannot**: 1) be granted WITH ADMIN OPTION and 2) be granted to roles.

To enable password file authentication

- Create the password file if it does not exist (use ORAPWD)
- Be sure REMOTE_LOGIN_PASSWORD_FILE=EXCLUSIVE
- Grant the SYSOPER or SYSDBA privileges to the users you want to add to the password file.
- Can query the V$PWFILE_USERS view to see userids that have been added to the password file.

The **DBA_SYS_PRIVS** view displays all system privileges granted to roles and users at the database level.

The **SESSION_PRIVS** view displays all system privileges granted to roles and users at the session level.

O7_DICTIONARY_ACCESSIBILITY parameter - When set to TRUE (default) allows SELECT ANY TABLE access to SYS schema and EXECUTE ANY PROCEDURE access to procedures in the SYS schema. If set to false, these privileges will not be able to access SYS schema objects.

Be sure you are very familiar with revoking system privileges.

Object Privileges

When an object privilege is granted it allows the grantee to perform an operation on an object. The following is a summary of object privileges and the object they can be granted on.

- SELECT FROM object (table, view, sequence)
- UPDATE object (table or view)
- INSERT INTO object (table or view)
- ALTER object (table or seq)
- CREATE TRIGGER ON object (tables only)
- DELETE FROM object (table or view)
- TRUNCATE object (tables only)
- EXECUTE OBJECT (procedure or function)
- CREATE INDEX ON object (tables only)
- REFERENCES - CREATE or ALTER TABLE defining a foreign key on a table only.

Can be granted with the WITH GRANT OPTION. Allow grantee to grant to other users and roles.

WITH GRANT OPTION can not be granted to a role.

Object privileges granted WITH GRANT OPTION are revoked when the grantor's privilege is revoked.

Related data dictionary views:

DBA_TAB_PRIVS - all privileges on objects

DBA_COL_PRIVS - all privileges on columns

DBA_SYS_PRIVS - system privileges granted to users and roles

USER_TAB_PRIVS - user is owner, grantor, or grantee

USER_TAB_PRIVS_MADE - user is the owner

USER_TAB_PRIVS_RECD - user is the grantee

USER_COL_PRIVS - user is owner, grantor, or grantee

USER_COL_PRIVS_MADE - user is the owner

USER_COL_PRIVS_RECD - user is the grantee

Grantors can revoke privileges from only those to whom they have granted privileges to.

You must use the CASCADE CONSTRAINTS option to revoke the REFERENCES privilege.

General tricky stuff

SELECT is an object privilege, but SELECT ANY TABLE is a system privilege.

Be sure you know the difference between all system and object privileges very well and all the rules for cascading when revoking privileges.

You must have the DROP ANY TABLE privilege to truncate a table.

Auditing Guidelines

- Defining the purpose of auditing
- Identify what to audit
- Manage the audit trail

Auditing Categories

Privileged operations - such as startup, shut down, and connects to the db by users with SYSOPER or SYSDBA privileges.

Database auditing - record selected user actions within the database. Column values are not captured as part of "database" auditing.

Application level auditing - Need to use application level auditing to capture this info. This is also called **value-based auditing**. An example of this would be a trigger that stores txn data and other selected info about the transaction before any insert, update, or delete occurs.

All auditing information is stored in the audit trail and is only written if the DBA has set the **AUDIT_TRAIL** parameter.

Set the **AUDIT_TRAIL** parameter in init<sid>.ora to write the audit trail. Possible values are:

DB - enables auditing - audit records written to the database SYS.AUD$ view

OS - enables auditing - audit records written to the operating system audit trail if permitted by O/S. UNIX O/S write file location is defined by the AUDIT_FILE_DEST parameter. Default AUDIT_FILE_DEST = $ORACLE_HOME/rdbms/audit. NT O/S writes records to the event viewer.

NONE - disables auditing is the default value.

Instance startup, shutdown, and connections to the database with administrator privilege is ALWAYS audited, no matter what the AUDIT_TRAIL parameter value is.

Use the AUDIT command to specify auditing options.

Auditing is not carried out if you are connected as INTERNAL or as the user SYS.

Audit records are written during the execute phase of statement execution.

Oracle database auditing consists of **statement**, **privilege**, and **object level** auditing.

Some AUDIT command options

USER - audit only these users.

BY SESSION - writes one record per db object per session to audit file.

BY ACCESS - write a record to the audit each time a statement is submitted.

WHENEVER - audit successful and unsuccessful SQL statements.

If you omit the WHENEVER clause, both successful and unsuccessful attempts will be audited.

Statement and privilege auditing options, specified by the audit command, apply only to subsequent sessions, not the current one.

When auditing unsuccessful actions, no audit record is generated for invalid statements such as syntax errors.

Object audit options take effect in the current sessions once issued.

If auditing outside your own schema, the AUDIT ANY privilege is required.

The AUDIT SYSTEM privilege is required to set DEFAULT audit options.

Related audit options data dictionary views

ALL_DEF_AUDIT_OPTS - default audit options

DBA_STMT_AUDIT_OPTS - system auditing options

DBA_PRIV_AUDIT_OPTS - system privilege auditing options

DBA_OBJ_AUDIT_OPTS - schema object auditing options

Disable with the NOAUDIT command.

The audit trail is located in the SYS.AUD$ data dictionary table.

Related audit results data dictionary views:

DBA_AUDIT_TRAIL - all audit trail entries

DBA_AUDIT_EXISTS - all records for EXISTS/NOT EXISTS

DBA_AUDIT_OBJECT - all records concerning schema objects

DBA_AUDIT_SESSION - all connect and disconnect entries

DBA_AUDIT_STATEMENT - all records concerning statement auditing

Protect the audit trail by auditing deletes from the aud$ table and granting DELETE_CATALOG_ROLE role to only the DBA.

Run cataudit.sql and catnoaudit.sql to create and remove the data dictionary tables and views for auditing.

Move the aud$ table to its own tablespace and truncate it frequently to get the size of the db from growing too big.

Managing Roles

Roles provide a way to easily group and manage privileges. You create a role and then assign privileges to it. You then can grant that role to specific users.

General Role Info

Rolls are granted and revoked to and from users using the same commands as for privileges.

- Can be granted to any user or role except itself.
- Can contain system and object privileges.
- Can be enabled and disabled for all users granted the role.
- Can be used in conjunction with a password to enable it.
- Name must be unique among existing user and role names.
- Not owned by anyone and not part of any schema.
- Descriptions are stored in the data dictionary.

Benefits of roles

- Simplify privilege management.
- Can be modified and effect all users assigned the role dynamically.
- Can be enabled and disabled on the fly.
- Granted via the O/S.
- No cascading revokes for object privileges.
- Improves performance and reduces the number of grants stored in the database.

CREATE ROLE privilege is required to create a role.

Role creation options

NOT IDENTIFIED - users granted the role need not be verified by oracle to enable it.

IDENTIFIED - users granted the role need to be verified by oracle to enable it.

BY password - user must specify the password to enable the role

EXTERNALLY - Oracle verifies access to role via OS system utility.

Predefined roles

CONNECT and RESOURCE - Provided for backwards compatibility

DBA - All system privileges WITH ADMIN OPTION

EXP_FULL_DATABASE - Export the entire database

IMP_FULL_DATABASE - Import the entire database

DELETE_CATALOG_ROLE - Delete privileges on the data dictionary tables.

EXECUTE_CATALOG_ROLE - Execute privileges on the data dictionary tables.

SELECT_CATALOG_ROLE - Select privileges on the data dictionary tables.

Oracle provides a couple of dba admin roles. They are **OSOPER** and **OSDBA**. These names may vary by OS.

The Oracle **dbmsaqad.sql** script creates a couple of advanced queuing roles. They are **AQ_ADMINISTRATOR_ROLE** and **AQ_USER_ROLE**.

OSOPER/SYSOPER roles can

STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP CONTROLFILE, ALTER TABLESPACE BEGIN/END BACKUP, ARCHIVE LOG AND RECOVER. Also contains the RESTRICTED SESSION privilege.

OSDBA/SYSDBA roles can

All system privileges WITH ADMIN OPTION and the OSOPER role. Permits CREATE DATABASE and time-based recovery. OSOPER/SYSOPER does not.

CREATING ROLES

First, create the role. Second, grant privileges to the role. Finally, grant the role to the user.

Use the **ALTER ROLE** command to change the authorization or authentication method needed to enable the role.

The **GRANT <role-name> TO...** command uses the same syntax as granting a system privilege to a user.

The WITH ADMIN OPTION allows

- The grantee to grant the role to another user or role.
- The grantee can revoke the role from other users.
- The grantee can alter or drop the role.

The creator of a role is automatically assigned the ADMIN OPTION.

If a user is granted a role WITHOUT the ADMIN OPTION, the GRANT ANY ROLE system privilege is required to grant and revoke the role to other users.

Establish a user's **default role** with the **ALTER USER** command. If the **ALL option** is used with the **ALTER USER … DEFAULT ROLE** command, all the roles granted to the user will be the default roles.

The **SET ROLE** command enables roles. Any roles not specified are disabled for the session.

SET ROLE ALL

- Enables and disables all roles granted to a user except those listed in the EXCEPT clause.
- Roles listed in the EXCEPT clause must be granted directly to you, not through another role.
- Can not use this option to enable roles with passwords that have been granted directly to you.
- Works when all roles involved either has no password or is authorized externally (by the OS).
- Can not execute this command from within a stored procedure.

The **REVOKE** command revokes a role from a user, a role, or from PUBLIC. The PUBLIC clause revokes the role from ALL the database users.

Use the **DROP ROLE** command to remove a role from the database. Must have been created the role WITH ADMIN option or have DROP ANY ROLE privilege.

RULE Creation Guidelines

- Create a role for each type of application task and name the role accordingly.
- Assign the application specific privileges to the application specific role.
- Create roles based upon each kind of different application user.
- Grant the application "task" roles, not individual privileges, to the application "user" roles.
- Grant the user and application roles to the user

Related data dictionary views:

ROLE_SYS_PRIVS - system privs granted to roles

ROLE_TAB_PRIVS - object privs granted to roles

ROLE_ROLE_PRIVS - roles granted to other roles

SESSION_ROLES - roles user has currently enabled

USER_ROLE_PRIVS - roles granted to the user

DBA_SYS_PRIVS - description of sys-privs granted to users and roles

DBA_ROLES - all roles that exist in the db and if they require a password or not

DBA_ROLE_PRIVS - roles granted to users and roles

Fine-Grained Access Control

Provides a method to associate security policies with table and views. **Define once on the database** server and then it is applicable to all applications that access the table or view. Database server automatically enforces the policies.

Implement the security policy with package functions and then associate with the table or view.

A table or view may have multiple policies defined on it. The policy enforcement is cumulative. "AND"logic is used to evaluate the multiple access control conditions that have been defined.

Using National Language Support

Oracle 8i supports about 45 languages, 60 territories, 60 linguistic sort sequences and multiple encoded character sets.

NLS features include support for languages, territories, character sets, linguistic sorting, localized messaged, multiple date, time, numeric, and monetary formats.

Encoding schemes supported

- **Single Byte Character Sets**

7-bit schemes support up to 128 characters. An example is ASCII 7-bit American (US7ASCII).

8-bit schemes support up to 256 characters. An example is ISO 8859-1 West European (WE8IS08859P1).

- **Varying-Width Multibyte Character Sets**

Each character is represented by one or more bytes. commonly used for Asian language support. Examplesinclude Japanese Extended UNIX Code (JEUC) and Chinese GB2312-80 (CGB23120-80).

- **Unicode Character Set**

Worldwide "superset" encoding standard that represents all characters for computer use including technical symbols. Unicode 2.0 can represent 38,885 characters.

Database Character Sets

- Defined at database creation.
- Can not be changed without re-creating the database.
- Store data types of CHAR, VARCHAR2, CLOB, and LONG.
- Can store varying-width character sets.

National Character Sets

- Defined at database creation.
- Cannot be changed without re-creating the database.
- Store data types of NCHAR, NVARCHAR2, NCLOB.
- Can store fixed-width and varying-width character sets.

Try to choose a database character set and a national character set that are closely related.

String operation might be faster using fixed-width character, but variable-width character sets use space better.

Can specify the NLS parameter: 1) on the server 2) as client environment variable 3) using ALTER SESSION command.

NLS_LANGUAGE parameter specifies

- language used for all messages
- day and month names
- symbols for A.D , B.C , A.M, and P.M.
- default sorting (collation sequence)

NLS_LANGUAGE parameter determines default values for the following parameters:

- NLS_DATE_LANGUAGE
- NLS_SORT

NLS_TERRITORY parameter specifies

- day and week numbering
- default date format
- decimal character
- group separator
- default ISO and local currency symbols

NLS_TERRITORY parameter determines default values for the following parameters:

- NLS_CURRENCY
- NLS_ISO_CURRENCY
- NLS_DATE_FORMAT
- NLS_NUMERIC_CHARACTERS

New init parm **NLS_DUAL_CURRENCY** supports the Euro currency.

NLS_LANG = <language>_<territory>.<characterset> can be set as an environment variable for each client session to override default NLS behavior.

NLS_CALENDAR can be set to specify which calendar to use.

The following NLS init parms can only be set in the client environment

- NLS_CREDIT
- NLS_DEBIT
- NLS_DISPLAY
- NLS_LANG
- NLS_LIST_SEPARATOR
- NLS_MONETARY

- NLS_NCHAR

Must set the **ORA_NLS33** environment variable in order to create a database in a character set other than US7ASCII.

On **UNIX** set to $ORACLE_HOME/ocommon/nls/admin/data

On **NT** this is set in the registry at install time in the HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE folder.

Can use the **ALTER SESSION** command to change individual NLS parms for a session. Can also use the **DBMS_SESSION.SET_NLS** package.procedure to change NLS values for a session.

Know how to change the sort sequence by setting the NLS_SORT on the fly with the **alter session** command.

Some SQL functions allow NLS parameters to be specified and can override environment and/or current NLS session settings.

The TO_DATE , TO_NUMBER, TO_CHAR, NLS_UPPER, NLS_LOWER, NLS_INITCAP, and NLSSORT functions accept specific NLS parameters. Know which NLS parameters can be used with each of these functions!!! For example The NLS_SORT parameter can be used with the NLS_UPPER, NLS_LOWER, NLS_INITCAP, and NLSSORT functions.

Number Format Masks

D decimal separator

G thousands group separator

L local currency symbol

C local ISO currency symbol

Date Format Masks

RM,rm	Roman month number
IW	ISO week number
IYYY IYY IY I	ISO year

Use the **NLS_COMP** parameter to indicate that comparison operations should use linguistic ordering. DEFAULT value is binary. Makes it so you don't have to use the NLSSORT function within the where clause.

Data is converted from **NLS_LANG** to database character set **during an Oracle IMPORT**.

SQL Loader - **Conventional load method** converts data to the value of current sessions NLS_LANG. **Direct load method** - data converted into database character set.

Related views

NLS_DATABASE_PARAMETERS	- Contains database and national character set values.
NLS_INSTANCE_PARAMETERS	- Only contains NLS values set in the init<sid>.ora file.
NLS_SESSION_PARAMETERS	- Contains NLS session values.
V$NLS_VALID_VALUES	- Contains valid NLS parameter settings
V$NLS_PARAMETERS	- Contains current values of NLS parameters

New columns found in various views

CHARACTER_SET_NAME	- character set name
CHAR_CS	- database character set
NCHAR_CS	- national character set

Special thanks to

Rick Sands

for contributing this Cramsession.