



SOLIX[®]
Technologies



DBA Handbook for **ORACLE**[®]

Chapter 1: Introduction	5
Chapter 2: Oracle Database Architecture	6
2.1 The Database	7
2.2 The Instance	7
2.3 Database Components	8
2.4 Memory Structures	20
2.5 Oracle Processes for an Instance	23
Chapter 3: Administration Activities	26
3.1 Installing the Oracle Database Software	27
3.2 Creating Databases	29
3.3 Database Operation	30
3.4 Space Management	34
3.5 User Management	37
3.6 Oracle Network Management	38
Chapter 4: Managing Schema Objects	44
4.1 Tables	45
4.2 Clusters	48
4.3 Views	49
4.4 Indexes	50
4.5 Synonyms	53
4.6 Triggers	54
4.7 Database Links	55
Chapter 5: Database Security and User Management	56
5.1 Roles	57
5.2 Privilege	59
5.3 Grant	60
5.4 Revoke	61
5.5 Password Management	62
5.6 Oracle Auditing	64
Chapter 6: Database Tuning	68
6.1 Database Tuning Strategy	69
6.2 SQL Tuning	70
6.3 Memory Tuning	72
6.4 I/O Tuning	73
6.5 Sort Tuning	73
6.6 The Optimizer	74
6.7 SQL Explain Plan	75

Chapter 7: Backup and Recovery	78
7.1 Backup and Recovery Mechanisms	79
7.2 Export / Import	81
7.3 SQL*	83
7.4 Standby Database	85
Chapter 8: Data Dictionary and Built-In Packages	90
8.1 Useful Admin Tables	91
8.2 Useful V\$ VIEWS :	92
8.3 Packages	93
Chapter 9: New Features in 9i and 10g	100
9.1 Version 9i. Enhancements	101
9.2 Improvements in Oracle Version 10g	104
Chapter 10: Database Monitoring	106
10.1 General Monitoring Scripts	107
10.2 Monitoring Memory Usage	107
10.3 Monitoring Disk I/O	108
10.4 Monitoring System Resources	110
10.5 Monitoring for Database Security	111
10.6 Monitoring Database Schema Objects	112
10.7 SQL Monitoring	114
10.8 Useful Unix Commands	115
Chapter 11: Archiving and Its Value to DBA	120
11.1 Introduction	121
11.2 What is driving the data growth?	121
11.3 Solution	121
11.4 Benefits	124

Preface:

Backed by a decade of expertise in Enterprise Applications Management, Solix Technologies, Inc. is a leading innovator in providing data management solutions to meet the demanding, high-availability, and high-performance requirements of enterprise applications. Solix extends its experience in Oracle Database Administration through this 'DBA Handbook'. This book intends to serve as a definitive handbook for the installation, administration, and maintenance of Oracle Database. It is focused on the administrative responsibilities and techniques for database administrators using Oracle Database.

Oracle Technology:

The Oracle Relational Database Management System (RDBMS) is the most popular relational database management system in use today. Organizations ranging from government agencies to financial institutions have made use of the Oracle RDBMS to maintain and process their data.

Why this book?

Oracle is a complex data processing environment encompassing hundreds of software components and commands with more than 45 volumes of comprehensive documentation. Several personnel performing a number of specific design and administrative roles usually share administration of Oracle RDBMS. As organizational needs and the number of users grow rapidly over time, so does the complexity of the Oracle system. This book attempts to present the user with easily accessible and concise information. It addresses many technical challenges with the help of several illustrative examples. It is assumed that the user has a basic understanding and familiarity of the Oracle Database architecture. Several important concepts have been elaborated wherever deemed appropriate.

Who should read this book?

This book is primarily targeted towards three categories of users:

- Database administrators who manage and maintain production Oracle databases.
- Oracle developers seeking to enhance their own database administration skills.
- Junior database administrators trying to get to know the tricks of the trade.

Which Oracle Release does it apply to?

This book applies to Oracle 8i, Oracle 9i, and Oracle 10g. Some of the features that are specific to Oracle 9i and Oracle 10g have been discussed in a separate chapter, "New Features in release 9i and 10g."

Resources:

Reach us at:

Solix Technologies, Inc.
685, West Maude Ave
Sunnyvale, CA 94085
Tel: +1-888-GO-SOLIX
Fax: +1-408-737-1607

Disclaimer:

This handbook contains references to brands and products of several companies that are not owned by Solix Technologies, Inc. As such, Solix, Inc. does not make any representations or evaluations in this regard. All scripts and queries are guidelines and have to be further customized according to your specific needs. Use them at your own risk.

Introduction

Contents

- The Introduction

CHAPTER 1: INTRODUCTION

A database administrator in any company is responsible for establishing policies and procedures pertaining to the management, security, maintenance and use of database management systems. Besides planning, implementing and maintaining databases for a company, they also play a key role in training users, programmers and test engineers on database use and procedures. A Database administrator's role requires knowledge in the following area:

- Different computer platforms and operating systems existing within an enterprise.
- Database component and their interaction with each other.
- Varied business rules defining the database system.

Swelling data growth issues and ever-changing business environment have left DBA's with new challenges, different from their day-to-day tasks. It is imperative that a DBA performs effectively and hence needs to constantly update his skills and knowledge base. Moreover, A DBA has to foresee the need to deploy data management solutions to ensure that the application environment continuously provides the required level of performance. Some of these solutions have been discussed towards the end of the book.

In a nutshell, this handbook is a complete guide to help improve a DBA's performance in an Oracle Database environment.

- Understand the business requirement for developing a database system.
- Plan resources (time and costs) needed to create database system.
- Install Oracle Software.
- Manage the database storage structures, schema objects, such as tables and indexes.
- Create and review troubleshooting procedures and plan for crisis management.
- Set up computer security procedures.
- Monitor database growth.
- Perform proactive monitoring and plan preventive maintenance.
- Identify ways to measure and improve system performance.
- Implement and maintain the primary and secondary storage devices for production data and data backups respectively.
- Upgrade Oracle Database and software to new release levels.

Database administrators spend a significant amount of time in maintenance and management of the database. Installation and configuration of the database only forms a small part of their day-to-day activities.

Oracle Database Architecture

Contents

- The Database
- The Instance
- Database Components
- Tablespace
- The Control Files
- The Datafiles
- Initialization Parameters Files (init.ora)
- Server Parameter Files
- The Redo Log Files
- Trace and Alert Log Files
- Memory Structures
- System Global Area
- Program Global Areas
- Oracle Processes for An Instance
- Background Processes

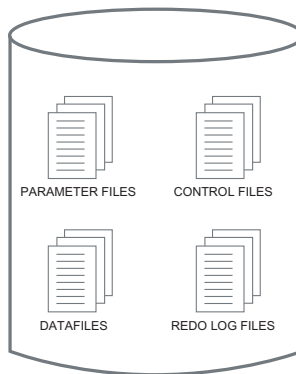
CHAPTER 2: ORACLE DATABASE ARCHITECTURE

Overview

The Oracle Database is a repository of storable, updateable, and retrievable data. The database itself is a collection of physical and logical structures consisting of system, user, control, and datafiles. The files are collectively known as the Oracle Database system. The Oracle Database server manages these files.

2.1 The Database

The databases refer to the physical storage of data. Oracle Database System is based on the relational database management model. The data is stored in two-dimensional tables composed of columns and rows. The columns of an individual table define the relational tables. Each of these columns defines a particular type of data and is called an attribute. The data pertaining to an individual set of records is stored in the rows. Individual tables can be related to each other. These tables may even contain object-oriented structures such as abstract data types and methods. All the data is stored in files and can be retrieved by using database structures that provide the logical mapping of the data onto the files. Different data types are stored individually in this manner.



ORACLE DATABASE
Figure 1.Oracle Database

2.2 The Instance

Instances refer to a particular set of software programs executed by the server. This software provides access to the data stored in the database. The set of processes that facilitate the storing, updating and retrieving the data along with the allocated memory on the database server form the Instance of the database. Instance acts as the interface that allows the users to communicate with the server. The data flows between a user and the database only if the instance is operational.

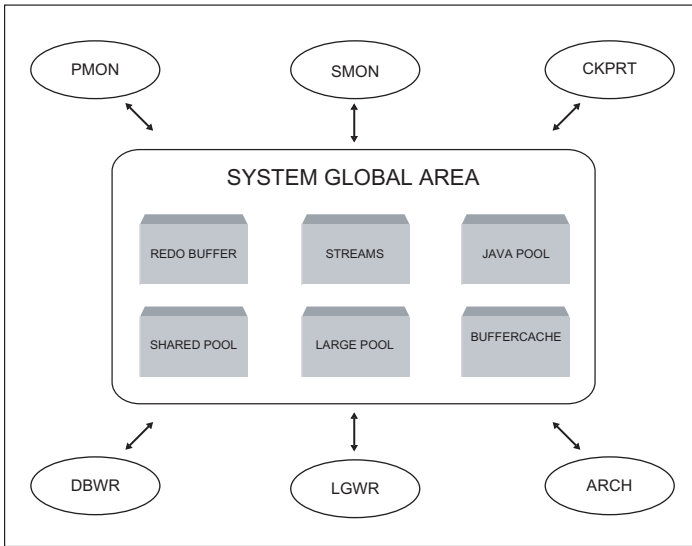


Figure 1.Oracle Instance.

Related Views

Views	Description
v\$database	Description of database parameter and status
v\$active_instances	Description of all active instances
v\$dbfile	Description of database files
v\$dblink	Description of database links definitions and status
v\$instance	Description of status of all instances
v\$rollstat	Description of rollback information
v\$undostat	Descriptions of undo information
v\$datafile	Description of datafiles information

2.3 Database Components

2.3.1 Tablespace

Tablespace is a logical structure belonging to a database system. Each database has to have at least one tablespace called the system tablespace. Each tablespace is made up of a collection of datafiles and can belong to only one database.

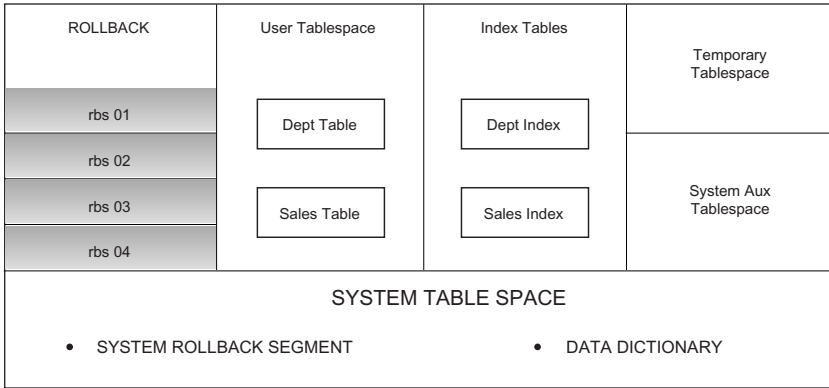


Figure 3. *Tablespaces.*

To enable manual system tablespace management:

To create and manage the SYSTEM tablespace manually set the EXTENT MANAGEMENT to LOCAL in the CREATE DATABASE statement. By default, the database defines and manages the extent sizes and creates a dictionary-managed SYSTEM tablespace.

Use the DBMS_SPACE_ADMIN package to migrate the dictionary-managed SYSTEM tablespace.

SYSAUX Tablespace (Specific to 10g)

The SYSAUX tablespace is defined at the point of creation of the database. It serves as a default tablespace to many Oracle features and packages that were previously assigned a separate tablespace. The size of SYSAUX tablespace depends on the size of each individual component. The typical size of the SYSAUX tablespace is about 250 MB at the time of creation of the database.

Default Permanent Tablespace

This tablespace is assigned to any non-system user for which an exclusive permanent tablespace is not allocated.

Oracle Managed Files (OMF) simplify file administration

- OMF are created and deleted by the Oracle server as directed by SQL commands
- OMF are established by setting two parameters:
 - DB_CREATE_FILE_DEST:
 - DB_CREATE_ONLINE_LOG_DEST_N:

2.3.2 The Control Files

The control files contain information about the contents and the state of the database. The location of important files such as datafiles, redo log files, and administrative information such as the database name, date and time of creation, current state, and list of backups performed, are stored here. If the administrator has enabled Oracle managed files, the control file is created as the Oracle managed control files. It is always good to backup the control file.

Related Views

Views	Description
v\$database	Description of the control file
v\$control files	Displays the list of control file names
v\$control file record section	Description of control file record
v\$parameter	Displays the names of the control files that are stored in the initialization parameter file

New Control File

A new control file is created in case:

- All copies of the control files pertaining to a particular database are permanently damaged.
- Some of the permanent parameters of the database such as the name of the database, etc. have to be changed. (The RESETLOGS clause must be specified in case the database has to be renamed.)

To create a new control file:

1. Collect all the information related to datafiles and the redo files by running the following queries.

```
· SQL>ALTER DATABASE BACKUP CONTROL FILE TO TRACE
```

2. Shut down the database.
3. Make a backup of all the datafiles that belong to the database by copying the files to a different location.
4. Start a new instance without mounting the database.

```
· SQL>STARTUP NOMOUNT
```

5. Create a new control file as shown using CREATE CONTROLFILE.

```

SQL> CREATE CONTROLFILE

SET DATABASE slx_db01
LOGFILE GROUP 1
    ('/mnt/sdc1/1159/slx_db01/redo01_01.log',
    '/mnt/sdc1/1159//slx_db01/redo01_02.log'),
GROUP 2
    ('/mnt/sdc1/1159/slx_db01/redo02_01.log',
    '/mnt/sdc1/1159/slx_db01/redo02_02.log'),
GROUP 3
    ('/mnt/sdc1/1159/slx_db01/redo03_01.log',
    '/mnt/sdc1/1159/slx_db01/redo03_02.log')
RESETLOGS

DATAFILE '/mnt/sdc1/1159//slx_db01/system01.dbf'
    SIZE 3M,
    '/mnt/sdc1/1159/slx_db01/rbs01.dbs'
    SIZE 5M,
    '/mnt/sdc1/1159/slx_db01/users01.dbs'
    SIZE 5M,
    '/mnt/sdc1/1159/slx_db01/temp01.dbs'
    SIZE 5M

MAXLOGFILES 50
MAXLOGMEMBERS 3
MAXLOGHISTORY 400
MAXDATAFILES 200
MAXINSTANCES 6
ARCHIVELOG;

```

Important Control File Parameters

MAXLOGFILES	The maximum number of redo log files that the database can have
MAXLOGMEMBERS	The maximum number of members that belong to each redo log file group
MAXLOGHISTORY	The maximum number of history files that each control file can contain is specified by this parameter. The history files facilitate the automatic recovery of the database
MAXDATAFILES	The number of datafiles that the control file can keep track of is specified by this parameter. In case more datafiles are added to the database, the control file will account for it automatically
MAXINSTANCES	The number of instances that the control file can track is specified by this parameter. This parameter is applicable to the RAC architecture

To enable multiple copies of control file the init.ora file should include the following:

```
CONTROL_FILES = (
/mnt/sdc1/1159/slx_db01/slx_db01ctl1.ctl,
    /mnt/sdc1/1159/slx_db01/slx_db01ctl2.ctl,
    /mnt/sdc1/1159/slx_db01/slx_db01ctl3.ctl).
```

To create additional copies of the control file:

1. Shutdown the database.
2. Copy the existing control file to a new location.
3. Restart the database.

To list all control files:

```
SQL>SELECT NAME FROM v$controlfile;
```

The CONTROL_FILES initialization parameter has to be edited so that it points to the new control file:

- ☞ *If the database has to be renamed, edit the DB_NAME in the initialization parameter file and restart the database by specifying the USING BACKUP CONTROL FILE clause.*
- ☞ *Once the database is opened using the new control file, check the alert file for any inconsistencies between the data dictionary and the control files.*
- ☞ *In case any datafiles present in the data dictionary are not included in the control file, filename MISSINGnnn appears in the control file and is interpreted by Oracle as files needing recovery or being offline.*
- ☞ *In case the user does not include a filename or adds a new filename while creating the control file, Oracle issues error such as ORA-01173, ORA-01176, ORA-01177, ORA-01215, or ORA-01216, while mounting the database.*

To backup a control file:

The control file can be backed up as a binary file.

```
ALTER DATABASE BACKUP CONTROLFILE TO
'(/mnt/sdc1/1159/slx_db01/backup/control.bkp');
```

To backup a control file as a text file:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

To recover a control file:

- Shut down the instance.
- Copy new files from backup files.
- Restart the instance.

To drop a control file:

- Edit init.ora, remove the entry corresponding to the control file.
- Shut down the database.
- Start up the database.

To relocate the control file:

1. Shut down the database.
2. Copy the control file to the new location.
3. Edit the `CONTROL_FILES` parameter in the initialization parameter file to point it towards a new location.
4. Restart the database.

2.3.3 The Datafiles

The datafiles hold data pertaining to the tables, the data dictionaries, and the rollback segments. Each Oracle Database has more than one physical datafile associated with it. A single datafile can be associated with only one database. The datafiles can be set to dynamically increase in size, if required. A collection of datafiles belonging to one logical unit is called tablespace.

Related Views

Views	Description
<code>dba_data_files</code>	Description of the database data file

To add a datafile:

```
ALTER TABLESPACE slx_tbs_01
  ADD DATAFILE 'slx_df04.dbf'
  SIZE 100K
  AUTOEXTEND ON
  NEXT 10K
  MAXSIZE 100K;
```

To resize a datafile:

```
ALTER DATABASE
  DATAFILE
  '/mnt/sdc1/1159/slx_db01/slx_dbf01.dbf'
  RESIZE 20M;
```

To recover a standby datafile:

The following statement recovers the standby datafiles including all the relevant archived logs and current standby database control file.

```
ALTER DATABASE
  RECOVER STANDBY DATAFILE
  '/mnt/sdc1/1159/slx_db01/slx_stbs01.dbf'
  UNTIL CONTROLFILE;.
```

2.3.4 Initialization Parameters Files (init.ora)

The characteristics of the Oracle Database can be defined by setting initialization parameters. The init.ora file stores these parameters and is referred to by the Oracle system before mounting the database.

- ☞ *There is always a sample initialization parameter file provided by Oracle's Database Configuration Assistant.*
- ☞ *This file can be used directly or can be customized. In case a parameter is not defined in the initialization file, Oracle proceeds with the default values.*
- ☞ *Use the ALTER SYSTEM statement after the database has been created to alter the initialization parameter.*

Sample init.ora file

```

CONTROL_FILES =
(/mnt/sdc1/1159/slx_db01/slx_db01ctl1.001.dbf,
/mnt/sdc1/1159/slx_db01/slx_db01ctl1.002.dbf,
(/mnt/sdc1/1159/slx_db01/slx_db01ctl1.003.dbf,)
DB_NAME = slx_db01DB_
DOMAIN = slx.com
LOG_ARCHIVE_DEST_1 =
    "LOCATION(/mnt/sdc1/1159/slx_db01/arch"
LOG_ARCHIVE_DEST_STATE_1 = enable
DB_BLOCK_SIZE = 8192
PGA_AGGREGATE_TARGET = 2500M
PROCESSES = 900
SESSIONS = 1200
OPEN_CURSORS = 1024
UNDO_MANAGEMENT = AUTO
SHARED_SERVERS = 2
REMOTE_LISTENER = tnsslx_lcg03
UNDO_TABLESPACE = slx_und01
COMPATIBLE = 10.1.0.0.0
NLS_LANGUAGE = AMERICAN
NLS_TERRITORY = AMERICA
DB_RECOVERY_FILE_DEST_SIZE = 60G

```

Important Initialization parameters include:

- Global Database Name
- Flash Recovery Area
- Control Files
- Database Block Sizes
- System Global Area
- Maximum Number of Processes
- Undo Space Management
- Compatibility of Initialization Parameter
- License Parameter

Global Database Name

The global database name is a concatenation of the individual database name and the domain name.

- ☞ *DB_NAME cannot contain more than eight characters and must be a text string.*
- ☞ *The DB_NAME is recorded in the datafiles, control files, and the redo files during the database creation.*
- ☞ *The database will not start if the database name in the control file does not match with the DB_NAME parameter.*

Flash Recovery Area

The database stores all the files related to backup and recovery in the flash recovery area. This area is separate from the database area where Oracle stores datafiles, redo logs, and the control files. This feature is specific to Oracle 10g.

The two main parameters associated with the definition of the flash recovery area are:

- *DB_RECOVERY_FILE_DEST specifies the destination of the files.*
- *DB_RECOVERY_FILE_DEST_SIZE specifies the maximum bytes that the flash recovery area can use.*

Control Files

The database can be associated with more than one control file. This is specified using the CONTROL_FILES initialization parameter. As the CREATE DATABASE statement is executed the control files specified by the CONTROL_FILES parameter are created.

Database Block Size

The standard block size for the entire database can be specified using the DB_BLOCK_SIZE.

- ☞ *Once the block size is set, it is used by the SYSTEM tablespace. If the parameter is not set, the default is obtained based on the operating system. This parameter cannot be changed after the creation of the database.*
- ☞ *A larger data block results in more efficient performance of Oracle in a data warehouse. On the other hand smaller block size would be more appropriate for OLTP sessions.*

System Global Area

System global area is made up of different pools of memory. These pools of memory are used to meet the memory allocation requests.

Maximum Number of Processes

The maximum number of processes that can be concurrently connected to the database is determined by the PROCESSES parameter. The lowest value assigned to this parameter is one. The number of processes mainly depends on the feature being used.

Undo Space Management

The undo space is used to store the undo changes that were made to the database before they are committed. These records are called undo records. To start the automatic undo management mode, set the UNDO_MANAGEMENT parameter in the initialization parameter to AUTO.

Compatible Initialization Parameter

Every version of Oracle uses a number of features that are based on the file system format. All the files relevant to these features can be used by different versions of Oracle by setting the appropriate COMPATIBLE initialization parameter.

The License Parameter

This parameter enforces named user licensing. The maximum number of users that can be created can be specified. New users cannot be created once the maximum number of users is reached.

```
LICENSE_MAX_USERS = 250
```

To alter the initialization parameter values:

Note: The initialization parameter can be edited using the ALTER SYSTEM statement.

- ☞ *In case the initialization parameter text file is used to alter the parameters, the new value takes effect the next time you start an instance of the database. However, you can change the value of some parameters for the duration of the current session.*
- ☞ *Altering the server parameter file would make the initialization file persistent. Use the SET clause with the ALTER SYSTEM statement to change the initialization parameters*
- ☞ *When SCOPE = SPFILE, the changes to the dynamic parameters are effective at the next STARTUP.*
- ☞ *When SCOPE = MEMORY, the new dynamic parameters are effective immediately but not persistent.*
- ☞ *When SCOPE = BOTH, the new dynamic parameters are effective immediately and remain persistent.*

To reset the initialization parameter:

The initialization parameters are restored to the default value by using the empty string. In case of Boolean parameters, the value has to be stated explicitly.

```
SQL> ALTER SYSTEM SET init_parameter = '';
```

To Create initialization text file from the server parameter files:

```
SQL> CREATE PFILE='/mnt/sdc1/1159/slx_init.ora'  
FROM SPFILE='/mnt/sdc1/1159/slx_spfile.ora';
```

- ☞ *Oracle creates a default name that is platform specific in case the name of the new parameter file is not specified.*

2.3.5 Server Parameter Files

The initialization parameters are stored in a binary server parameter file. This file is persistent over the database startup and shutdown. The changes that are made in the initialization file, even when the instance is running, are persistent during startup and shutdown.

The actual parameter file is created using the SPFILE statement. Once the STARTUP command is issued, the initialization parameters are read from the server parameter file.

To deploy Server Parameter File during the startup instead of the initialization parameter file:

- Place the file on the server machine.
- Create the server parameter file using CREATE SPFILE statement before the STARTUP command is issued.
- Log in as SYSDBA or SYSOPER.

```
SQL> CREATE SPFILE=
        '/mnt/sdc1/1159/slx_db01/spfileslx_db1.ora'
        FROM PFILE=
        '/mnt/sdc1/1159/slx_db01/admin/initslx_db01/scripts/init.ora';
```

- SHUTDOWN the database.

☞ A new server parameter file is created.

☞ In case no name is supplied then Oracle gives a default name `spfile$ORACLE_SID.ora`.

☞ If the server parameter file is created in any location other than the default location, the parameter file has to contain the following line.

```
SPFILE = '/mnt/sdc1/1159/slx_db01/$ORACLE_SID_ifile.ora'
```

☞ The new server parameter file will overwrite any existing file.

2.3.6 The Redo Log Files

Every database is associated with two or more redo log files collectively called redo log. Commit results in the redo log generation. The datafiles are updated asynchronously. The redo-log records all changes made to the data. If the database fails before the data in the datafiles is modified, the changes are lost. These changes can be obtained from the redo log files. Multiple copies of the log files are maintained in order to protect against failure of the redo log file.

To add a redo file:

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER
'/mnt/sdc1/1159/slx_db01/redo_log3.log' TO GROUP 3
```

To add a redo log file group:

```
SQL> ALTER DATABASE
ADD LOGFILE GROUP 3
('/mnt/sdc1/1159/slx_db01/redo_log3.log',
'/mnt/sdc1/1159/slx_db01/redo_log4.log') SIZE 50K;
```

To rename a redo log file member:

```
SQL> ALTER DATABASE RENAME FILE
'/mnt/sdc1/1159/slx_db01/redo_log4.log' TO
'/mnt/sdc1/1159/slx_db01/redo_log5.log';
```

To clear log file:

```
SQL> ALTER DATABASE
CLEAR LOGFILE '/mnt/sdc1/1159/slx_db01/redo_log4.log';
```

To drop a redo file:

```
SQL> ALTER DATABASE
DROP LOGFILE MEMBER
'/mnt/sdc1/1159/slx_db01/redo_log3.log/log3.log'
```

To drop the entire group:

```
SQL> ALTER DATABASE DROP LOGFILE group 4;
```

2.3.7 Alert Log Files

Oracle records all major events such as the startup status, shutdown status, addition of tablespaces, etc. corresponding to the database. This information is stored in the alert log files.

2.3.8 Auto Trace Files

The server dumps some information into the trace files in case there is an error while running the background processes. Most of this information can be used to troubleshoot any issue with the database. The Oracle support team uses this information to debug.

By default these log files are located in

\$ORACLE_HOME/\$ORACLE_SID/admin/bdump

2.3.9 User-generated Trace Files

The user to diagnose performance issues can track specific events. Making use of the ENABLE TRACE option before performing any specific action will do this

To return the path of the trace files:

```
SQL> SHOW PARAMETER user_dump_dest;
```

To enable SQL trace:

```
SQL> ALTER SESSION SET SQL_TRACE=TRUE;
```

2.4 Memory Structures

Oracle uses memory to store the program code, process connected sessions, and manage information about various executing processes, data blocks, and redo logs. The memory structure is classified as system global area and program global area. All the processes share the system global area. The program global area is private to each server process.

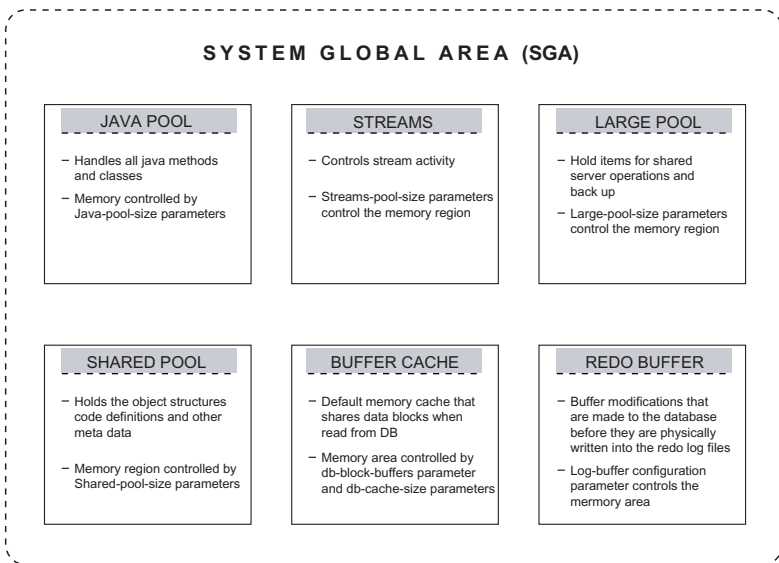


Figure 4. System Global Area

Related Views

Views	Description
v\$sga	SGA status
v\$sga_stat	SGA components status
v\$tablespace	Tablespace definition and configuration
v\$cache	Description of cache information
v\$librarycache	Status of the library cache
v\$db_object_cache	Status of cache used by each object

2.4.1 System Global Area

The SGA is a portion of memory containing the data and the control information that belongs to each Oracle Database Instance. The data present here is shared among all the users that are connected to the same instance.

Various SGA components allocate and de-allocate units of memory as and when there are requests for memory allocation from the processes. The units of memory are called granules. The size of these granules depends on the size of the entire SGA. Typically an SGA of 1GB has granules of 4 MB in size, and an SGA larger than 1 GB has granules 16 MB in size.

The size of the SGA can be limited using the `SGA_MAX_SIZE` parameter. In case it is not specified, Oracle assigns a specific value to `SGA_MAX_SIZE` based on the sum of the sizes of all the components.

Setting Maximum Memory Manually

```
SHARED_POOL_SIZE=128M
DB_CACHE_SIZE=896M
```

Database Buffer Cache

The database buffer cache holds copies of data blocks that are retrieved from the datafiles. The database buffer is logically divided into two portions: the Write list and the Least Recently Used (LRU) list.

To set the database buffer cache size manually:

```
DB_BLOCK_SIZE=4096
DB_CACHE_SIZE=1024M
```

To set non-standard block sizes for multiple block sizes:

- `DB_2K_CACHE_SIZE`
- `DB_4K_CACHE_SIZE`
- `DB_8K_CACHE_SIZE`
- `DB_16K_CACHE_SIZE`
- `DB_32K_CACHE_SIZE`

Redo Log Buffers

All changes that are made to the database including INSERT, DELETE, UPDATE, ALTER, CREATE, and DROP are saved in the redo log buffers. Redo entries from these buffers are used if the database crashes. The LGWR process writes the redo log buffers into the redo log files.

To View SGA details

```
SQL> SHOW SGA
Total System Global Area      560812908 bytes
Fixed Size                    455532 bytes
Variable Size                 385875968 bytes
Database Buffers              163840000 bytes
Redo Buffers                   10641408 bytes
```

Shared Pool

The shared pool contains the dictionary and the library cache, control structures, and parallel execution buffers. The size of the shared pool can be set by the `SHARED_POOL_SIZE` initialization parameter and is typically 8 MB or 64 MB.

SQL Pool Area

The SQL statements issued by the user are parsed and executed in this memory area. The memory for this portion is allocated dynamically depending on the complexity of the statement. Multiple users running the same program share the same memory area. The specific values pertaining to the session are stored in the private area of the user.

Java Pool

The java code specific to the session is stored in this portion of memory. The information about the library cache memory used for java can be viewed using the java pool advisor statistics. Setting `STATISTICS_LEVEL` to `TYPICAL` or higher enables the java pool advisor.

Large Pool

A portion of the memory can be programmed to serve as session memory for all the shared servers that perform transactions requiring them to interact with more than one database. All I/O, backup, and restore processes use this pool of memory.

Stream Pool

The SGA memory is used by the streams allocated from this pool, if the `STREAMS_POOL_SIZE` is set greater than zero. In case this parameter is set to zero, a portion of the shared pool is allocated to the streams.

Data Dictionary Cache

The Data dictionary is a collection of a set of data tables and views that holds the referential information about the database, its structures, and its users. Two locations hold the data dictionary: the data dictionary cache and the row cache. All user processes share these areas.

- ☞ *External controls are provided in order to dynamically increase or decrease the use of the physical memory.*
- ☞ *The SGA growth is based on the maximum value set for the parameter `SGA_MAX_SIZE`.*
- ☞ *Oracle managed policies specify the buffer cache and the SGA individual pools sizes.*
- ☞ *The memory can be locked on to the SGA by using the `LOCK_SGA` parameter.*
- ☞ *SGA starting address can be specified by setting `SHARED_MEMORY_ADDRESS` and `HI_SHARED_MEMORY_ADDRESS` parameters.*

2.4.2 Program Global Areas

The data and control information of a particular server process is assigned to a portion of the program global area. Other process and servers do not share this memory space. This memory space is accessed only by Oracle. The accumulation of the memory allocated to each server process belonging to an instance is called aggregated PGA.

The PGA consists of:

- Private SQL area
- Cursor area
- Session memory

Managing the PGA

- ☞ *The total size of memory dedicated towards the PGA can be set using the initialization parameter `PGA_AGGREGATE_TARGET`.*
- ☞ *The memory management for the PGA set to automatic by setting the parameter `WORKAREA_SIZE_POLICY` to `AUTO`.*

Related Views

Views	Description
<code>v\$sysstat</code>	Lists all the available statistics
<code>v\$sesstat</code>	Session statistics
<code>v\$pgastat</code>	PGA Memory status

To find the PGA memory utilization

```
SQL> SELECT pga_used_mem, pga_allocated_mem,
           pga_max_mem      FROM v$process;
```

2.5 Oracle Processes for An Instance

A process can be defined as a 'thread of control' that executes a series of instructions. Each process is associated with a private memory space in which it is executed. Oracle supports multiprocessing, enabling several users and applications to run different parts of the Oracle code without compromising the performance of other systems.

The database processes can be classified into dedicated server processes and shared server processes. The database allows all dedicated server processes to be processed by default. The shared server processes have to be set up by configuring initialization parameters.

Related Views

Views	Description
v\$session	Status and definition session attributes
v\$process	Status of all the process
v\$access	Description of object access information
v\$session_event	Status of all session events
v\$mystat	Status of current session
v\$session_connect_info	Session connection such as authorizations and network service banners
v\$session_cursor_cache	Session open cursor status

2.5.1 Background Processes

The database system always runs a set of background processes common to all the users and the databases. These background processes monitor other database processes and also perform the I/O operations. Some of the important background processes are listed below.

Database Writer (DBWn)

This process writes the data blocks from the buffer area into the actual datafiles. There can be a maximum of 20 writer processes at any time. The number of database writers can be specified using the DB_WRITE_PROCESSES. They are conventionally named DBW0 to DBW9 and DBW_a to DBW_j.

Log Writer (LGWR)

This process writes the redo log entries present in the redo log buffer. The redo log buffers present in the system global area are written into the redo log file. The redo log files are also written if the user process commits a record modification. Once the user commits a transaction, the transaction is assigned a particular number called the system change number. This information is also recorded in the redo log.

Checkpoint (CKPT)

The database writer DBW_n writes all the data present in the SGA buffers into the datafiles at specific intervals of time. The point where the DBW_n has to resume is called a checkpoint. The checkpoint process is responsible for all the updates to the data and control files.

System Monitor (SMON)

This process performs recovery in case the instance fails and is started again. The SMON frees up all the temporary segments that are not in use anymore, and performs recovery of any transactions that have not been executed due to system failure.

Process Monitor (PMON)

This process monitors all the user processes. In case of failure, it initializes recovery. It also frees up the cache and monitors the dispatcher process and server processes. It revives them in case of failure.

Archiver (ARCn)

This process copies the redo files to an archival storage. The archiver is present only when the database is placed in the ARCHIVELOG mode by enabling auto archiving. Each instance can have a maximum of 10 processes (ARC0 to ARC9) associated with it. Multiple archiver logs can be specified if a high workload is expected. Setting the LOG_ARCHIVE_MAX_PROCESSES can increase the number of archiver processes.

Recoverer (RECO)

All the distributed transactions pending due to system or network failure are recovered using the recoverer. This process also performs pending commit or rollback actions belonging to the local portion of distributed transaction.

Dispatcher (Dnnn)

In case the shared server configuration is used, this process manages the communication connections in a distributed environment.

Global Cache Service (LMS)

This process provides inter-instance control.

Job Queue Process

All batch processes are run using the job queue processes. The job queue process manages the processes that are scheduled at certain intervals. The coordinator process called the CJQ0 looks up the JOB\$ table for new jobs to be run. The process executes one job at a time in case the JOB_QUEUE_PROCESSES is set to 0.

Queue Monitor Process

The queue monitor is capable of monitoring message queues process. This process is optional. The user can configure up to 10 queue monitor processes.

Administration Activities

Contents

- Installing the Oracle Database Software
- Creating Databases
- Database Operation
- Starting the database
- Shutting Down the Database
- Quiescing a Database
- Suspend and Resume a Database
- Space Management
- Tablespace Management
- Rollback Segment
- User Management
- Oracle Network Management
- Oracle Networking Components
- Listener
- The Dispatcher

CHAPTER 3: ADMINISTRATION ACTIVITIES

3.1 Installing the Oracle Database Software

The Oracle software is typically installed using the Oracle Universal Installer (OUI). The OUI is a GUI-based tool that reviews Oracle software currently installed. The user has to select the fresh installation choice, and has the option of adding additional components using this tool.

Oracle performs an automated system requirement check. This requirement check includes a basic software and hardware assessment including minimum memory requirements, storage space, and file systems.

Linux and Unix systems

The installer invokes the Specific Inventory Directory page in case no previous installs were performed on the machine. An installation file directory, that the operating system has write permission to, must be specified. This area is not same as Oracle home. All the operating system variables pertaining to the Oracle Database are set automatically by the installer.

Oracle Universal installer can be invoked using the `oraINST.sh`. The Oracle Universal installer prompts for various parameter including `ORACLE_HOME`.

Installation modes

Oracle Enterprise Edition

The Enterprise Edition is the most thorough installation of the Oracle database product. This installation provides the largest number of features.

Oracle Standard Edition

This edition is a subset of the Enterprise Edition and is suitable for smaller implementations. It is also an economic option.

Custom Install

This mode is suitable for customization of the Enterprise Edition. This mode enables specific individual components to be selected or unselected at the point of installation.

Data Storage Mechanisms

Database files are constituted by administrator data, database meta data, and files pertaining to recovery of the database in case of a failure. The administrator can decide on the type of storage allocated to each of these files. Following options maybe configured

The File System

The operating system manages all database related files. The directory path where the Oracle stores the database related files must be specified. The actual creation and management of these files is carried out by Oracle automatically. This is the default option and is most widely used.

The Automatic Storage Management

The administrator defines a group of disks that can be used by Oracle to store and maintain the database related files. Using this option eases the database file management and increases performance. The Automatic Storage Management requires a separate instance to set up and manage the group of disks. The administrator is guided through the Automatic Storage Management launching.

Raw Devices

These storage devices are outside the control of the operating system. They are unformatted disk spaces. It is necessary to ensure that the operating system has not allocated this space to any other process.

Backup Strategy

The administrator can select backup and recovery options. The backup configuration can be automated or the administrator can plan a backup strategy later.

Schema Passwords

The database schema passwords for the SYS and the SYSTEM have to be specified. The SYS and SYSTEM are primary usernames that facilitate the administration of the database. The highest-level administrators retain these passwords.

Summary Page

The Oracle Universal Installer displays the complete summary of all components that have been selected for installation. Clicking the install button will start the installation process. The DBA runs the script, root.sh, in a new terminal window when prompted. The configuration tools page comes up towards the end of the installation. Starting the NET Service listener process can configure the Oracle Network.

The listener can be started from the command prompt.

```
$ lsnrctl START $ORACLE_SID;
```

Password Management

The password management page comes up after the installation is complete. Locked accounts cannot be accessed but can be unlocked by removing the check mark in the Lock Account column. This page contains important information about various web application port numbers.

3.2 Creating Databases

Oracle Databases can be created either using the Database Configuration Assistant or a CREATE DATABASE statement. Using the DBCA is easier. Creating the database manually requires careful planning. Additional actions such as creating new users, tablespaces and data dictionary tables have to be performed in case the database is created manually.

To create the database:

☞ Make sure that there is sufficient memory and disk space.

☞ Specify Oracle SID.

```
$ ORACLE_SID = slx_db01;
EXPORT ORACLE_SID;
```

☞ Create initialization parameter file by copying sample parameter file init.ora. Make sure that you customize the parameter to specific requirements since parameters such as BLOCK_SIZE cannot be changed once set.

☞ Connect as SYSDBA.

```
$ sqlplus /nolog
$ SQL> CONNECT AS SYSDBA
$ SQL> STARTUP NOMOUNT
```

☞ Run the script to create a database.

```
CREATE DATABASE slx_db01
USER SYS IDENTIFIED BY sol123
USER SYSTEM IDENTIFIED BY sol123
LOGFILE GROUP 1
('/mnt/sdc1/1159/slx_db01/redo01.log') SIZE 100M, GROUP 2
('/mnt/sdc1/1159/slx_db01/redo02.log') SIZE 100M, GROUP 3
('/mnt/sdc1/1159/slx_db01/redo03.log') SIZE 100M
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
MAXINSTANCES 1
CHARACTER SET US7ASCII
NATIONAL CHARACTER SET AL16
UTF16DATAFILE '/mnt/sdc1/1159/slx_db01/system01.dbf' SIZE 325M
REUSE
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE
'/mnt/sdc1/1159/slx_db01/sysaux01.dbf' SIZE 325M REUSE
DEFAULT TABLESPACE tbs_1
DEFAULT TEMPORARY TABLESPACE tempts1
TEMPFILE '/mnt/sdc1/1159/slx_db01/temp01.dbf'
SIZE 20M REUSE
UNDO TABLESPACE undotbs
DATAFILE '/mnt/sdc1/1159/slx_db01/undotbs01.dbf'
SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

☞ Run the catproc.sql and catalog.sql while connected as SYS. These files are located by default in \$ORACLE_HOME/rdbm/admin

☞ These actions result in the creation of the database with default datafiles, control files, redo log files, system tablespace, and data dictionary. The default users SYS and SYSTEM are also defined.

3.3 Database Operation

3.3.1 To Start the Database

The database is made operational to the users by mounting and then opening it

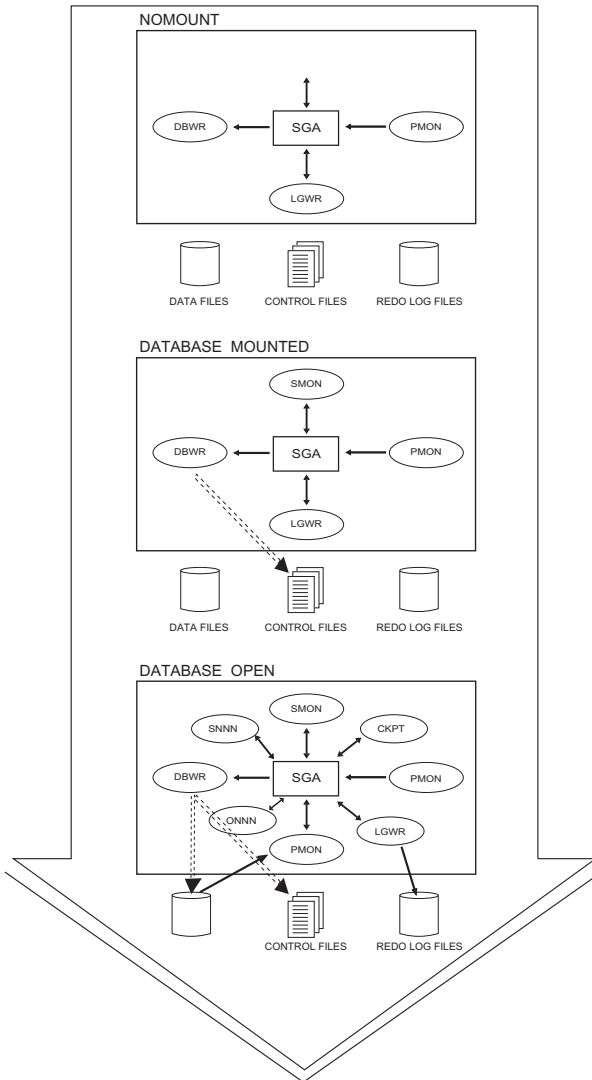


Figure 5. Starting Oracle Database.

Using SQL*Plus to start the database

1. The administrator is required to connect to the SYSDBA.

```
$ sqlplus /nolog
SQL> conn / as SYSDBA;
```


2. Start up an Oracle Database instance.

```
SQL> Startup
```

☞ Oracle identifies the parameter file by looking in the following order:

- spfile\$ORACLE_SID.ora
- spfile.ora
- init\$ORACLE_SID.ora

☞ The administrator can instruct Oracle to read a specific initialization parameters from the initialization parameter file by using the PFILE clause during the STARTUP command.

```
STARTUP PFILE =/mnt/sdc1/1159/slx_db01/init_slx_db01.ora;
```

Starting Modes

To start and mount the database:

The following statement starts the instance by mounting the database and opening it at the same time. This mode allows all valid users to connect to the database and perform operations.

```
SQL> STARTUP
```

To start the instance without mounting the database:

This is typically done while creating the database.

```
SQL> STARTUP NOMOUNT
```

To start an instance and mount a database without opening the database:

The database is mounted and not opened in case the administrators have to perform any maintenance operations such as enabling or disabling redo log archiving files or performing full database recovery.

```
SQL> STARTUP MOUNT
```

To start an instance and mount a database without opening the database:

Access to an instance can also be in restricted mode such that the databases are available only to administrators. The administrators may perform functions such as exporting or importing database data, loading data using the SQL*Loader, or performing migration and upgrade operations.

```
SQL> STARTUP RESTRICT
```

☞ While in the restricted mode, all users that have CREATE SESSION RESTRICTED privileges can connect to the database.

☞ The database administrators cannot use the Oracle Net Manager to connect to the database remotely.

To disable the restricted mode:

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

To enable restricted session while database is in normal mode:

```
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

Force Startup

The administrator can forcefully start another instance of the database.

```
SQL> STARTUP FORCE
```

- ☞ This mode is used in case the current instance could not be shutdown using the normal commands such as SHUTDOWN NORMAL, SHUTDOWN IMMEDIATE, or SHUTDOWN TRANSACTIONAL commands.
- ☞ This would cause the instance to forcefully shutdown in ABORT mode and restart it.

To start an instance to commence complete media recovery:

```
SQL> STARTUP OPEN RECOVER
```

- ☞ In case the administrator wants to perform complete recovery, the instance has to be started and the database has to be mounted.
- ☞ Recovery can be performed only in case a recovery is required. In case the administrator issues recover option without any requirement, Oracle issues an error message.

To open databases those are closed at the start of the session:

```
SQL> ALTER DATABASE OPEN;
```

Read Only Mode

Users can open the database and retrieve data without having the ability to change the contents of the database.

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

3.3.2 Shutting Down the Database

The database and the instance can be shut down only while being connected as SYSOPER or SYSDBA. There are different ways of shut down the database.

Normal Clause

The database can be shut down normally using the SHUTDOWN command in the SQL*Plus.

```
SQL> SHUTDOWN NORMAL
```

☞ *Once the statement is issued, Oracle stops accepting any new connections. The database waits for all the users that are connected to the database to disconnect.*

Shutdown Immediate

Shutdown Immediate is typically done when an automated and unattended backup is to be performed, or power interruptions are anticipated.

```
SQL> shutdown immediate;
```

Shutdown Transactional

The database starts the shutting down process. The database does not accept any new connections and all uncommitted transactions are rolled back. The database disconnects all users immediately. All users are disconnected after the transactions are completed. This way there is no loss of any data.

```
SQL> shutdown transactional;
```

Abort

Abort is typically done, if the database has to be shut down instantaneously, typically within about a minute. No new connections are allowed after aborting. Existing connections are ended forcefully.

```
SQL> SHUTDOWN ABORT
```

☞ *The SQL statement processing is stopped.*

☞ *Requires the instance recovery procedures next time when the database is started.*

3.3.3 Quiescing a Database

The database can be put in state that allows only DBA transactions, queries, and PL/SQL statements to be run on the system. Such a state is called the quiesce mode.

Some actions such as changing the schema of a database or simply adding a column to the database could potentially fail, if executed in normal state. However, these transactions can be run while the database is in a quiesced state. The Database Resource Manager feature has to be activated. The non-DBA sessions become inactive.

```
SQL> ALTER SYSTEM QUIESCE RESTRICTED;
```

☞ *All the attempts to change the current resources are queued until the database state is restored to normal.*

Undo Quiesce

```
SQL> ALTER SYSTEM UNQUIESCE;
```

3.3.4 Suspend and Resume a Database

The database can be placed in suspend mode. In suspend mode, the database stops all the I/O to datafiles and control files. All existing I/O operations are completed and new I/O operations are placed in a queue. This command is not specific to any single instance. All active instances will be placed in quiesce mode. The initialization of new instances is suspended.

To suspend database:

```
SQL> ALTER SYSTEM SUSPEND;
```

To resume system:

```
SQL> ALTER SYSTEM RESUME;
```

- ☞ *The SUSPEND and RESUME commands may be issued from separate instances.*
- ☞ *This feature is mainly used to split a mirror disk or a file in order to provide a backup.*

Force Logging Mode

The logging mode generates redo records in the database redo log. Some DDL statements such as CREATE TABLE can be run in the NOLOGGING clause to improve execution speed.

```
SQL> ALTER DATABASE NO FORCE LOGGING;
```

- ☞ *The status of FORCE LOGGING mode remains the same even after shutting down and restarting the database.*
- ☞ *Running the database in the forced logging mode results in complete recovery.*
- ☞ *System performance degrades drastically if forced logging mode with noarchive mode is enabled.*

3.4 Space Management

The data blocks are the lowest level in the granular structure that store data on the disk. Configuring the DB_BLOCK_SIZE initialization parameter can define the size of each block. The parameters, PCTFREE and PCTUSED, allow the user to decrease the amount of unused space in data blocks and the amount of row mitigation between the data blocks.

Related Views

Views	Description
dba_free_space	Size of the tablespaces
dba_extents	Extent attributes
dba_rollback_segs	Rollback segment attributes
dba_objects	Properties of all object
dba_undo_extents	Undo extent properties
dba_segments	All segment information
dba_tablespaces	Description of all tablespaces
dba_object_size	Sizes of various pl/sql objects

3.4.1 Tablespace Management

The tablespaces can be easily managed using the ASM (10g) system provided by Oracle. Two threshold levels (Warning and Critical) have to be defined to indicate that the tablespace is running out of storage space. By default, the warning level is set at 85% utilization and the critical level is set at 97% utilization.

The SET_THRESHOLD and GET_THRESHOLD procedures in the DBMS_SERVER_ALERT can be used to set and get threshold values respectively.

To view the current status of the tablespace:

```
SQL> SELECT property_value
FROM database_properties
WHERE property_name = 'DEFAULT_TBS_TYPE';
```

To create a tablespace:

```
SQL> CREATE TABLESPACE slx_tbs01
DATAFILE '/mnt/sdc1/1159/slx_db01/slx_ts01.dbf'
SIZE 100M;
```

To change tablespace modes to read/ write:

```
SQL> Alter tablespace slx_tbs01 READ WRITE;
```

To backup a tablespace online:

1. Set the datafile or tablespace in backup mode by issuing the ALTER TABLESPACE ...BEGIN BACKUP command to prevent change in sequence number in datafile header.

```
SQL> ALTER TABLESPACE users BEGIN BACKUP;
```

2. Use an OS backup utility to copy all datafiles in the table space to backup storage

Unix:

```
$cp /ORADATA/u03/users01.dbf /BACKUP/users01.dbf
```

3. After the datafiles of tablespace have been backed up, set them into normal mode by issuing the following command:

```
SQL> ALTER TABLESPACE users END BACKUP;
```

4. Archive the unarchived redo logs so that the redo required to recover the tablespace backup is archived as follows

```
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
```

To add a data file to a tablespace:

```
SQL> ALTER TABLESPACE slx_tbs01
      ADD DATAFILE 'slx_tbs02.dbf'
      SIZE 100K
      AUTOEXTEND ON
      NEXT 10K
      MAXSIZE 1000K;
```

☞ In case excess space is required, new 10-KB extents will be added up to a maximum of 1000 KB.

Logging Attribute

```
SQL> ALTER TABLESPACE Slx_tbs03 NOLOGGING;
```

To alter the extent allocation:

```
SQL> ALTER TABLESPACE slx_tbs01 MINIMUM EXTENT 128K;
```

To drop a tablespace:

```
SQL> DROP TABLESPACE slx_tbs01
      INCLUDING CONTENTS
      CASCADE CONSTRAINTS;
```

3.4.2 Rollback Segment

Rollback segment is a database object used to store data necessary to undo any action that is performed on the database. A table is maintained for every transaction recognized by the SMON process. In most cases the Automatic Undo mode is enabled to let Oracle manage the undo files.

Related Views

Views	Description
dba_segemnts	Storage allocated for all database segments
dba_rollback_segs	Description of rollback segments
user_segments	Storage allocated for all database segments
v\$rollstat	Contains extent and latch information
v\$rollname	Contains roll names

3.5 User Management

Each user must provide a valid user name and password in order to gain access to the database. To create the user, the following attributes must be specified:

- User name
- Authentication method
- Default tablespace
- Temporary tablespace
- Other tablespaces and quotas
- User profile

- ☞ *The default DBA user accounts provided by Oracle are SYS and SYSTEM.*
- ☞ *The User SYS is assigned the password, change_on_install and owns the Database Data Dictionary.*
- ☞ *The User SYSTEM is identified by the Password manager and owns the additional internal tables and views that are used by Oracle.*

To create a new user:

```
SQL> CREATE USER slx_user01
      IDENTIFIED BY this_is_the_password
      DEFAULT TABLESPACE slx_tbs01
      QUOTA 10M ON slx_tbs01
      TEMPORARY TABLESPACE temp
      QUOTA 5M ON system
```

To display current user account:

```
SQL> SHOW USER
```

To change a user password:

```
SQL> Alter USER slx_user01 IDENTIFIED BY new_password
```

To grant a session to a user:

```
SQL> GRANT CREATE SESSION TO slx_user01;
```

To alter tablespace allocation:

```
SQL> ALTER USER slx_user01 QUOTA 50M on slx_tbs01;
```

To restrict users from creating objects in the system tablespace:

```
SQL> ALTER USER slx_user01 QUOTA 0 ON system;
```

To allocate unlimited tablespace to a user:

```
SQL> ALTER USER slx_user01 QUOTA UNLIMITED  
On slx_tbs01;
```

To configure external database users:

```
SQL> CREATE USER slx_app_user01  
IDENTIFIED EXTERNALLY  
DEFAULT TABLESPACE slx_tbs01  
QUOTA 5M ON slx_tbs01  
PROFILE slx_apps_user
```

3.6 Oracle Network Management

3.6.1 Oracle Networking Components

The implementation of the Oracle Database is often in a distributed environment. It is critical to manage the connectivity, scalability, and security of the Oracle Database Network.

Some of the important components that help the administrator are:

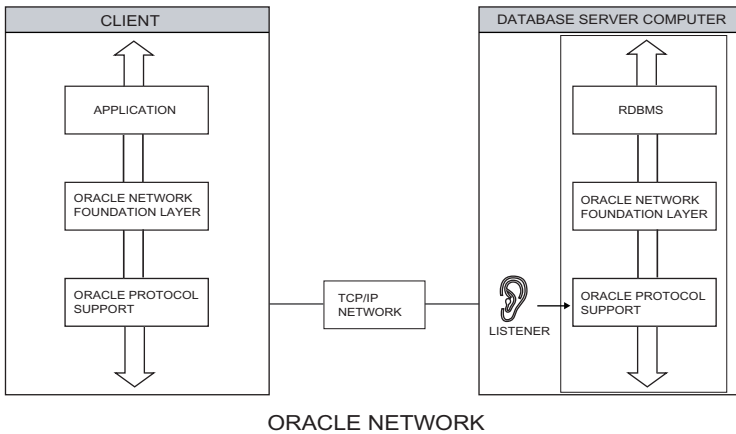
- Oracle Net
- Oracle Net Listener
- Oracle Connection Manager
- Networking Tools

Oracle Net

Oracle Net is an application layer software that resides on the client and the database server. It is responsible for establishing and maintaining the connection between the server and the client. The main software components that comprise Oracle Net are Oracle Net Foundation Layer and Oracle Protocol Support. The Oracle Net Foundation initiates and maintains the connection, whereas the Oracle Protocol Support helps communicate using the standardized communication protocols such as TCP/IP or TCP/IP with SSL.

Oracle Net Listener

The Oracle Net Listener is a process that runs on the database server. This process listens for new connections at the server side. The listener is configured with a protocol address. The clients have to use the same protocol address in order to send any requests. Once the connection is established the client and the server communicate directly with each other.



ORACLE NETWORK
Figure 6. Oracle Network.

Oracle Connection Manager

The Oracle Connection Manager funnels multiple sessions through a single transport layer protocol. This helps reduce demand on resources and enables the server to use fewer connections for incoming requests.

Networking Tools

Some of the Oracle Net services user interface tools include:

- Oracle Net Configuration Assistant - To configure Listeners
- Oracle Enterprise Manager - Manage Oracle Net Services
- Oracle Net Manager - Built in wizards and utilities to test connectivity, migrate data, create/ modify network components

3.6.2 Listener

The listener is a process that runs on the client and the server. This process listens for new communication requests and manages all the traffic on the Oracle Network. The listener can be configured with one or more protocol addresses. The configuration parameters pertaining to the listener are stored in the listener.ora file.

Sample listener.ora

```

LISTENER=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp) (HOST=slx_server01) (PORT=1521))
      (ADDRESS=(PROTOCOL=ipc) (KEY=extproc))))
SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (GLOBAL_DBNAME=slx_db01.slx.us.com)
      (ORACLE_HOME=/oracle10g)
      (SID_NAME=slx_db01))
    (SID_DESC=
      (SID_NAME=plsextproc)
      (ORACLE_HOME=/oracle10g)
      (PROGRAM=extproc)))

```

To start the listener:

```
$ lsnrctl START [listener name]
```

To stop the listener:

```
$ lsnrctl STOP [listener name]
```

To display the status of the listener:

```
$ lsnrctl STATUS [listener name]
```

Sample Status of the Listener

```

Connecting to (ADDRESS=(PROTOCOL=IPC) (KEY=EXTPROCVIS))
STATUS of the LISTENER
-----
Alias                slx_db01
Version              TNSLSNR for Linux: Version 9.2.0.3.0
- Production
Start Date           23-FEB-2005 16:48:57
Uptime               12 days 19 hr. 58 min. 57 sec
Trace Level          OFF
Security             OFF
SNMP                 OFF
Listener Parameter File
/mnt/slx_db01/network/admin/slx_db01/listener.ora
Listener Log File
/mnt/slx_db01/network/admin/slx_db01.log
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc) (KEY=EXTPROCVIS)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=slx_server01.slx.com)
(PORT=1522)))
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for
  this service...

```

To display detailed information about the listener:

```
$ lsnrctl service
```

```
LSNRCTL for Linux: Version 9.2.0.3.0 -
Production on 08-MAR-2005 12:59:10
Copyright (c) 1991, 2002, Oracle Corporation.
All rights reserved.

Connecting to (ADDRESS=(PROTOCOL=IPC) (KEY=EXTPROCVIS))
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for
  this service...
  Handler(s):
    "DEDICATED" established:0 refused:0
    LOCAL SERVER
...continued on next page
```

```
Service "slx_server01" has 1 instance(s).
  Instance "slx_db01", status UNKNOWN, has 1 handler(s) for
  this service...
  Handler(s):
    "DEDICATED" established:17358 refused:0
    LOCAL SERVER
```

To use the OEM to start / stop the listener:

- ☞ Click on *Net Services Administration* link.
- ☞ Select *Listeners* from the administer list.
- ☞ Select *start / stop* from the actions list.

3.6.3 The Dispatcher

Each shared server needs at least one dispatcher in order to work. If none of the dispatchers are configured, then Oracle defines a dispatcher for the TCP protocol.

Related Views

Views	Description
v\$dispatcher	Description of all the dispatcher
v\$dispatcher_rate	Description of dispatcher attributes
v\$queue	Description of queue status

Configuring A New Dispatcher

```
DISPATCHERS=" (ADDRESS= (PROTOCOL=TCP) (HOST=10.2.157.3) )
(DISPATCHERS=2) "
```

Optional Attributes.	Description
CONNECTIONS	Specifies the maximum number of network connection allowed for each dispatcher
SESSIONS	Specifies the maximum number of network sessions
TICKS	Specifies the max timeout values
LISTENER	Specifies an alias name that is used by the PMON to register dispatcher processes
MULTIPLEX	Enables the multiplexing feature
POOL	Enables pooling
SERVICE	Specifies service names the dispatcher uses to register with the listeners

To shut down a specific dispatcher:

```
SQL> ALTER SYSTEM SHUTDOWN IMMEDIATE 'slx_Dispatch02';
```

To shutdown all the dispatcher:

```
SQL> ALTER SYSTEM SET DISPATCHERS = '';
```


Managing Schema Objects

Contents

- Tables
- Clusters
- Views
- Indexes
- Synonyms
- Triggers
- Database Links

CHAPTER 4: MANAGING SCHEMA OBJECTS

4.1 Tables

Tables are basic units in which data can be stored as rows and columns in the Oracle database. Each table is identified by a name and a set of columns. Each column contains values of the same data type and holds a particular attribute. Separate rules called integrity constraints can be enforced on the data that can be stored in the column. A row can be defined as a collection of column information pertaining to a single record. Each table has to be assigned a specific table space. The user trying to create a table should have appropriate privileges.

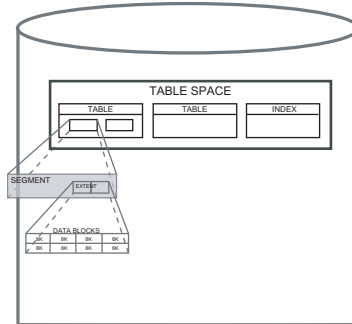


Figure 7. Oracle Data Storage Architecture

Related Views

Views	Description
dba_tables	Description of all relational tables in the database
dba_views	Description of all views in the database
dba_tab_comments	Comments on all tables and views in the database
dba_object_tables	Description of all object tables in the database
dba_types	Description of all types in the database
dba_method_params	Description of method parameters of all types in the database
dba_part_tables	Description of all partitioned tables
dba_part_lobs	Description of all LOBS
dba_part_col_statistics	Description of partitioned columns
dba_part_key_columns	Description of partitioned key columns

To create a table:

```
SQL> CREATE TABLE employee
(slx_emp_id NUMBER(10),
first_name VARCHAR2(30),
last_name VARCHAR2(30) not null,
phone_number VARCHAR2(15),
hire_date DATE not null,
slx_dept_no NUMBER(4)
);
```

Tablespace allocation

The table can be specified to use a particular tablespace by including this parameter towards the end of the CREATE table statement.

```
SQL> CREATE TABLE employee
(slx_emp_id    NUMBER(10),
first_name    VARCHAR2(30),
last_name     VARCHAR2(30) not null,
phone_number  VARCHAR2(15),
hire_date     DATE not null,
slx_dept_no   NUMBER(4)
)
TABLESPACE slx_tbs01
STORAGE (INITIAL    6155
        NEXT        6155
        MINEXTENTS  128 K
        MAXEXTENTS  128 K);
```

Temporary table

A temporary table can be created using the temporary clause. The data in these temporary tables is deleted at the end of the session.

```
SQL> CREATE GLOBAL TEMPORARY TABLE
      slx_routine
      ON COMMIT PRESERVE ROWS
      AS
      SELECT * FROM routine WHERE activity_date = SYSDATE;
```

Nested table

```
SQL> CREATE TABLE employee
(slx_emp_id    NUMBER(10) PRIMARY KEY,
last_name     VARCHAR2(30) not null,
phone_number  VARCHAR2(15),
hire_date     DATE not null,
slx_dept_no   NUMBER(4),
compensation_details textdoc_tab,
)
NESTED TABLE compensation_details
STORE AS
      compensation_nestedtable;

-- The table contains a nested column called the compensation_details.
```

To move table to new Tablespace

```
SQL> ALTER TABLE employee MOVE newTS
STORAGE (INITIAL 40K
        NEXT 80K
        MINEXTENTS 2
        PCTINCREASE 0);
```


To lock a table:

The LOCK TABLE statement is used to lock one or more tables.

```
SQL> LOCK TABLE employee
      IN EXCLUSIVE MODE
      NOWAIT;
```

- ☞ The lock table statement can be used to permit a user or deny the user access to a table. The lock does not prevent the users to query the table.
- ☞ This lock overrides the automatic locking and is active until either the transactions are committed or rolled back.

To add a column:

```
SQL> ALTER TABLE employee
      ADD (bonus NUMBER (7,2));
```

To rename a column name:

```
SQL> ALTER TABLE employee
      RENAME COLUMN emp_id TO eid;
```

To drop a column name:

```
SQL> ALTER TABLE employee DROP COLUMN salary;
```

To set unused columns:

```
SQL> ALTER TABLE employee SET UNUSED hiredate;
```

To remove unused columns:

```
SQL> ALTER TABLE employee DROP UNUSED COLUMNS
      CHECKPOINT 250;
```

- ☞ Remove the Unused Columns by specifying checkpoints to 250 rows at a time in order to avoid a potential exhaustion of undo space.

To drop a table:

```
SQL> DROP TABLE employee PURGE;
```

4.2 Clusters

A cluster is a schema object that contains data from a set of more than one table, each having one or more columns in common.

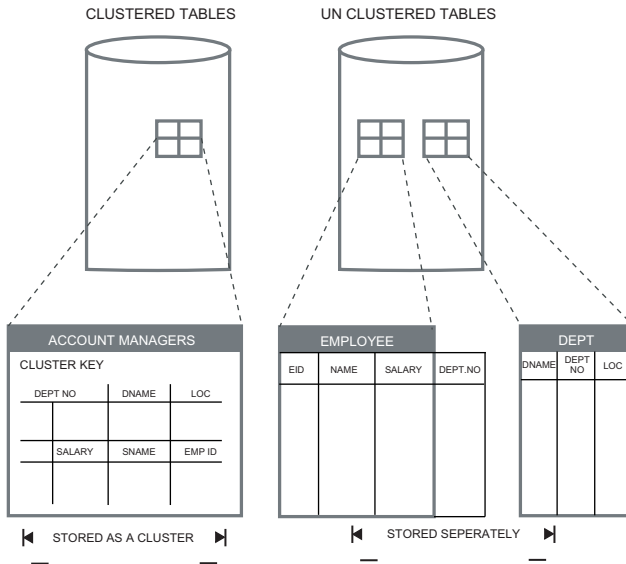


Figure 8. Clustered and Unclustered Tables

Related Views

Views	Description
user_clusters	Descriptions of user's own clusters
dba_clusters	Description of all clusters in the database
all_clusters	Description of clusters accessible to the user

To create clusters:

A user who wishes to create a cluster must have the CREATE CLUSTER system privilege.

```
SQL> CREATE CLUSTER account_managers
      (dept NUMBER(4))
      SIZE 512
      STORAGE (initial 100K next 50K);
```

To add tables to the cluster:

```
SQL> CREATE TABLE slx_dept01
      CLUSTER account_managers (dept_no) AS
      SELECT * FROM employee WHERE dept_no = 10;

CREATE TABLE slx_dept02
      CLUSTER account_managers (dept_no) AS
      SELECT * FROM employee WHERE dept_no = 20;
```

To alter a cluster:

The user must have ALTER ANY CLUSTER system privilege.

```
SQL> ALTER CLUSTER account_managers
      SIZE 1024 CACHE;
```

To drop clusters:

```
SQL> ALTER CLUSTER account_managers
      DEALLOCATE UNUSED KEEP 30 K;
      DROP CLUSTER;
```

4.3 Views

A view is a logical representation of data that belongs to one or more tables. A view does not store any data. A view derives its data from the data tables that are specified in its definition. The tables that view retrieves its data from are called base tables. Views can be further queried, updated, inserted into, and deleted from.

To create views:

```
SQL> CREATE VIEW personnel AS
      SELECT empno, last_name, dept_no
      FROM employee
      WHERE dept_no = 7;
```

To create views with joins:

```
SQL> CREATE VIEW work AS
      SELECT last_name, slx_emp_id,, dept_name
      FROM employee, department
      WHERE
      Employee.dept_id = department.dept_id;
```

To drop/replace a view:

```
SQL> CREATE OR REPLACE VIEW work AS
      SELECT slx_emp_id, last_name, dept_no
      FROM employee
      WHERE dept_no = 30
```

Join View

A join view has a sub query that contains at least one join. In order to modify the base table, at least one of the columns in the base table should have a unique index.

```
SQL> CREATE VIEW location_v AS
      SELECT
      department.dept_id, department.dept_name,
      location.loc_id, location.city
      FROM
      department, location
      WHERE
      department.loc_id = location.loc_id;
```

To find columns of the table those are updateable:

```
SQL> SELECT column_name, updatable
      FROM user_updatable_columns
      WHERE table_name = 'employee';
```

To insert values using the views:

Since all the columns in the view that are mapped to the dept table are marked updateable and the primary key of the table dept is retained in the view, it is possible to insert data into the tables through views.

```
SQL> INSERT INTO view_locations (dept_id, dept_name)
      VALUES (007, 'Sales');
```

To create a read only view:

```
SQL> CREATE VIEW Slx_customer(name, credit)
      AS SELECT cust_last_name, credit_limit
      FROM cust
      WITH READ ONLY;
```

To drop views:

```
SQL> DROP VIEW emp_dept;
```

☞ The user should have the *DROP ANY VIEW* system privilege to be able to drop a view in the database

To update view with a constraint:

```
SQL> CREATE VIEW manager AS
      SELECT emp_id, last_name, dept_no, job_id
      FROM emp
      WHERE job_id = 'Manager';
```

4.4 Indexes

In an RDBMS system, the actual location of the row is relevant only when the database has to retrieve a particular row. In order to find the relevant row, each row that belongs to a table is assigned a RowID. Indexes are simple database structures that make it easy to locate the row of the table being looked up. There are three types of indexes: the cluster indexes, table indexes, and the bitmap indexes. Indexes are optional structures that allow the SQL statements to execute more efficiently.

The most commonly used indexes are the B-tree indexes. It is more efficient to create the indexes after inserting or loading the data.

Related Views:

Views	Description
dba_indexes	Description of all indexes in the database
dba_ind_columns	Description of Columns comprising Indexes on all Tables and Clusters
dba_part_indexes	Description of all partitioned indexes

To create an index:

```
SQL> CREATE INDEX indexemp ON employee(last_name)
      TABLESPACE users
      STORAGE (INITIAL 20K
      NEXT 20k PCTFREE 0);
```

To create a unique index:

Unique indexes may be necessary to ensure that no two rows of a table have duplicate values.

```
SQL> CREATE UNIQUE INDEX indexdept
      ON department (dept_name)
      TABLESPACE indx;
```

To create an index online:

```
SQL> CREATE INDEX emp_name
      ON employee (mgr, emp1, emp2, emp3)
      ONLINE;
```

☞ The *ONLINE* clause allows the update of the tables at the time of creation of the indexes.

NOSORT Mode

If the table has been created using a fast parallel load where all rows are already sorted, the statement can be used to create an index quickly.

```
SQL> CREATE INDEX emp_name_ix
      ON employee
      NOSORT
      NOLOGGING;
```

Cluster Index

Since the cluster indexes are built on all the columns belonging to the cluster key, the columns need not be specified. In such cases, all the rows are indexed.

```
SQL> CREATE INDEX accountindex ON CLUSTER
      account_managers;
```

Function-Based Index

```
SQL> CREATE INDEX indexemp
      ON employee (UPPER (last_name));
```

Range Partitioned Global Index

```
SQL> CREATE INDEX salary_ix
      ON emp (salary)
      GLOBAL PARTITION BY RANGE (salary)
      (PARTITION p1 VALUES LESS THAN (25000),
       PARTITION p2 VALUES LESS THAN (50000),
       (MAXVALUE));
```

Hash-Partitioned Global Index

```
SQL> CREATE INDEX dept_id_ix
      ON employee (dept_id)
      GLOBAL PARTITION BY HASH (dept_id)
      PARTITIONS 3;
```

To alter the index:

The index should be present in the schema belonging to the user or the user should have the ALTER ANY INDEX system privilege in order to alter an existing index.

To rebuild the index:

```
SQL> ALTER INDEX emp_name_ix REBUILD;
```

Key Compression Index:

Key compressed indexes may be used to eliminate multiple occurrences of the key column prefix values. This compression breaks the index key into two values: the prefix and the suffix. All the prefix entries are grouped together resulting in significant space savings.

```
SQL> ALTER INDEX emp_name_ix REBUILD COMPRESS;
```

To store index blocks in reverse order:

```
SQL> ALTER INDEX emp_name_ix REBUILD REVERSE;
```

To rebuild an index in parallel:

```
SQL> ALTER INDEX emp_name_ix REBUILD PARALLEL;
```

Parallel Queries

```
SQL> ALTER INDEX emp_name_ix PARALLEL;
```

☞ Setting parallel attributes for index emp_name_ix enables the scans on the index to be paralleled.

To rename an index:

```
SQL> ALTER INDEX emp_name_ix RENAME TO
employee_name_ix ;
```

To mark an index as unusable:

```
SQL> ALTER INDEX emp_name_ix UNUSABLE;
```

To modify default index attributes:

```
SQL> ALTER INDEX emp_name_ix
MODIFY DEFAULT ATTRIBUTES INITRANS 5
STORAGE (NEXT 100K);
```

☞ The default attributes are changed to make use of 5 initial transaction entries and an incremental extent of 100K.

To drop an index:

```
SQL> DROP INDEX emp_name_ix;
```

4.5 Synonyms

Synonyms are used to make the location of the database object transparent to users of the distributed environment. Synonyms can refer to tables, types, views, sequences, procedures, functions, and packages. Synonyms are schema objects themselves and are stored in the data dictionary. A synonym can have the same name as the object to which it is referring.

Related Views

Views	Description
dba_synonyms	All synonyms in the database

To create a synonym:

```
SQL> CREATE SYNONYM employees
FOR hr.employee;
```

To create a public synonym:

```
SQL> CREATE PUBLIC SYNONYM employee
FOR hr.employee@us.slx.com;
```

To drop a synonym:

```
DROP PUBLIC SYNONYM employee;
```

4.6 Triggers

Triggers are stored PL/SQL blocks associated with a database or any of its components. It can also be an anonymous PL/SQL block or a call procedure. Once a trigger is created, it is enabled automatically by the database.

Related Views

Views	Description
dba_triggers	All triggers in the database
dba_internal_triggers	All internal triggers in the database
dba_trigger_cols	Column usage in all triggers

DML Trigger Example:

```
SQL> CREATE TRIGGER schema.slx_trigger
  BEFORE
  DELETE OR INSERT OR UPDATE
  ON schema.table_name pl/sql_block
```

This trigger is fired before executing a delete, insert, or update operation.

DDL Trigger Example:

```
SQL> CREATE TRIGGER slx_object AFTER CREATE
  ON SCHEMA
  pl/sql_block
```

These triggers can be used to audit the creation of a new data dictionary objects in the schema.

Calling procedures inside triggers

```
SQL> CREATE TRIGGER slx_emp.salary_check
  BEFORE INSERT OR UPDATE OF salary, emp_id ON
  Slx_hr.emp FOR EACH ROW
  WHEN (new.emp_id <> 'AD_VP')
  CALL salary_check(:new.emp_id, :new.salary,
  :new.last_name);
```

Database event Trigger

To log all the errors:

```
SQL> CREATE TRIGGER Slx_lg_err AFTER SERVERERROR
  ON DATABASE
  BEGIN
  IF (IS_SERVERERROR (1017)) THEN
    <perform special processing of logon error>
  ELSE
    <log the error number obtained>
  END IF;END;
```


To enable the triggers:

```
SQL> ALTER TRIGGER slx_lg_err ENABLE;
```

To disable the triggers:

```
SQL> ALTER TRIGGER slx_lg_err DISABLE;
```

To drop a trigger:

```
SQL> DROP TRIGGER slx_lg_err;
```

4.7 Database Links

Database links can be thought of as pointers that point to objects in a remote database.

There are three basic types of links:

1. Private Database Links
2. Public Database Links
3. Global Database Links

To create database links the user must have the following privileges:

- CREATE DATABASE LINK
- CREATE PUBLIC DATABASE LINK
- CREATE SESSION

Related Views

Views	Description
dba_db_links	All database links in the database

Fixed-User Database Link

```
SQL> CREATE DATABASE LINK local
CONNECT TO sales IDENTIFIED BY slx_user01
USING 'local';
```

☞ The user Scott logged on a remote database. Scott defines a fixed user database link called local to the sales schema present in the local database.

☞ If the database link is successfully created, Scott can query all the tables present in the schema sales on the local database by using the following statement.

```
SQL> SELECT * FROM sales@local;
```

To drop a Database Link:

```
SQL> DROP PUBLIC DATABASE LINK remote;
```

Database Security and User Management

Contents

- Roles
- Privilege
- Grant
- Revoke
- Password Management
- Account Locks
- Password Complexity Verification
- Password Aging and Expiration
- Password History
- Oracle Auditing

CHAPTER 5: DATABASE SECURITY AND USER MANAGEMENT

Overview

Every database should enforce a security policy to protect against accidental or malicious destruction of data or the database system. It is the responsibility of the administrator to create, maintain and enforce a security policy. The administrator creates users identified by a username and a password. The administrator while creating a new user specifies attributes such as user name, authentication mode, user profile, default tablespace, temporary tablespace and other tablespace.

Related Views

Views	Description
v\$enabledprivs	List of all enabled privileges
dba_users	Information about all users of the database
dba_policies	All row level security policies in the database

5.1 Roles

Role may be defined as a set of privileges or collection of other roles. Roles help manage the privileges for the database and the user group. Oracle provides some predefined roles. Roles are primarily used to control user access to the database.

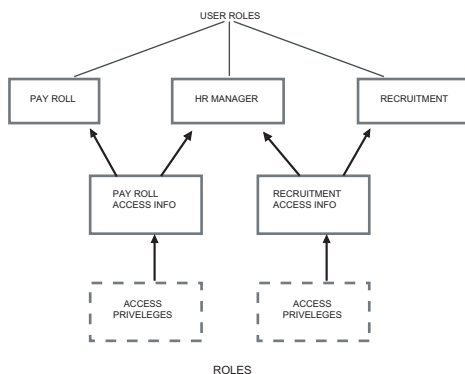


Figure 8. Roles

Roles can be granted to users by users who have the privilege to do so. During the creation of the database, the default SYS role is created with the privilege to grant roles to other users. Some predefined roles are:

- connect
- resource
- dba
- exp_full_database
- imp_full_database

List all the roles

```
SQL> SELECT * FROM dba_role_privs;
```

Create Slx_sales_manager role:

```
SQL> CREATE ROLE slx_sales_manager;
```

Password for a Role

Roles can be created using the CREATE ROLE statement. The user must have the CREATE ROLE privilege. The NOT IDENTIFIED clause indicates that the role is authorized by the database and no password is required to enable the role. The IDENTIFIED clause indicates that any user has to be authorized before the role is enabled. A password can also be set while creating the role.

```
SQL> CREATE ROLE Slx_sales_manager
IDENTIFIED BY bigsale;
```

External Role


The EXTERNALLY clause can be used to create an external role which requires the user to be authorized by an external service.

To create a role externally:

```
SQL> CREATE ROLE Slx_customer01
IDENTIFIED EXTERNALLY;
```

To create a global role:

```
SQL> CREATE ROLE slx_user IDENTIFIED GLOBALLY;
```

 The GLOBALLY clause allows the user to create a global role. The user has to be authorized to use the role by the enterprise directory service.

To set a role:

Roles for the current session can be enabled or disabled using the SET ROLE statement. The roles that are currently enabled can be viewed by querying the view SESSION_ROLES.

```
SQL> SET ROLE slx_Sales_manager IDENTIFIED BY S!$4g;
```

To set all roles:

```
SQL> SET ROLE ALL;
```

To restrict a role using set all:

```
SQL> SET ROLE ALL EXCEPT slx_sales_manager;
```

To disable all roles:

```
SQL> SET ROLE NONE;
```

To alter a role status:

```
SQL> ALTER ROLE slx_warehouse_user NOT IDENTIFIED;
```

To change a password:

```
SQL> ALTER ROLE slx_sales_manager  
IDENTIFIED BY S12#@B;
```

To change the status of a role to application role:

```
SQL> ALTER ROLE slx_sales_manager  
IDENTIFIED USING sales.admin;
```

To drop a role:

```
SQL> DROP ROLE slx_sales_manager;
```

☞ If a role is dropped, it is also revoked from all the users that have been granted this role.

☞ All the current user sessions do not get effected and new user sessions with the dropped roles are not allowed.

☞ The user who decides to drop the role should have ADMIN OPTION or DROP ANY ROLE privilege.

5.2 Privilege

A privilege is a right to perform a certain activity. Activities include connecting to a database, creating a table or selecting rows from tables that belong to other schemas. Privileges can be granted to users, roles or other applications. There are two types of privileges:

- System Privileges
- Schema Object Privileges

The schema object privilege is a right to perform actions on any schema objects such as clusters, indexes, database links and triggers.

Related Views

Views	Description
dba_role_privs	All Roles which exist in the database
dba_sys_privs	System privileges granted to users and roles

dba_tab_privs	All grants on objects in the database
dba_profiles	Display all profiles and their limits
dba_roles	All Roles which exist in the database

To grant a CREATE SESSION system privilege:

```
SQL> GRANT CREATE SESSION
      TO slx_sales;
```

To grant a system privilege to a role:

```
SQL> GRANT
      CREATE ANY MATERIALIZED VIEW,
      ALTER ANY MATERIALIZED VIEW,
      DROP ANY MATERIALIZED VIEW
      TO slx_sales_manager
      WITH ADMIN OPTION;
```

5.3 Grant

The grant statement is mostly used to grant system privileges to users and roles. Roles can also be granted to other roles or users. The user should have either the ADMIN OPTION or the GRANT ANY ROLE privilege to be able to grant a role. The user must have GRANT ANY OBJECT privilege to be able to grant any object privilege.

To grant a system privilege:

```
SQL> GRANT CREATE SESSION
      TO slx_soix_manager;
-- slx_sales_manager is a role.
```

To grant a privilege to a role:

```
SQL> GRANT
      CREATE ANY MATERIALIZED VIEW,
      ALTER ANY MATERIALIZED VIEW,
      DROP ANY MATERIALIZED VIEW
      TO slx_sales_manager
      WITH ADMIN OPTION;
```

To grant a role with the Admin option:

```
SQL> GRANT slx_sales_manager
      TO slx_user01
      WITH ADMIN OPTION;
```

☞ The ADMIN OPTION enables the user slx_user01 to grant or revoke the role to or from any users. The user Scott can also Drop the role.

To grant all:

```
SQL> GRANT ALL
      ON orders TO Slx_sales;
```

☞ Grant ALL grants DELETE, INSERT, SELECT, and UPDATE Privilege to a User.

To grant a role to another role:

```
SQL> GRANT slx_admin
      To slx_sales_manager;
```

To grant an object READ / WRITE privilege on a directory:

```
SQL> GRANT READ
      ON DIRECTORY slx_file_dir
      TO slx_Sales
      WITH GRANT OPTION;
```

To grant object all privileges on a table to a user:

```
SQL> GRANT ALL
      ON Slx_product_price
      TO Slx_sales
      WITH GRANT OPTION;
```

To grant object privileges on a view:

```
SQL> GRANT SELECT, UPDATE
      ON Slx_sales_view01
      TO PUBLIC;
```

5.4 Revoke

The revoke statement is used to revoke any system privileges or roles from users and roles. It is also used to revoke object privileges for a particular object from users and roles. The user must have the privilege with the ADMIN OPTION to be able to revoke any privilege. The user can revoke a role either by having the role with the ADMIN OPTION or by having the GRANT ANY ROLE system privilege.

To revoke a system privilege DROP ANY TABLE from a user:

```
SQL> REVOKE DROP ANY TABLE
      FROM slx_user01;
```

☞ The user can no longer drop any table from his/her schema.

To revoke a role:

```
SQL> REVOKE slx_sales_manager
      FROM slx_user01;
```

To revoke a system privilege from a role:

```
SQL> REVOKE CREATE TABLESPACE
      FROM slx_sales_manager;
```

To revoke a role from a role:

```
SQL> REVOKE slx_user
      FROM slx_Sales_manager;
```

To revoke a delete privilege:

```
SQL> REVOKE DELETE
      ON orders FROM Slx_sales_manager;
```

To revoke all object privileges:

```
SQL> REVOKE ALL
      ON orders FROM Slx_sales_manager;
```

To revoke an object privilege:

```
SQL> REVOKE UPDATE
      ON Slx_product_view FROM public;
```

5.5 Password Management

The security of the entire database system depends on the passwords. Passwords can be managed through user policies. Passwords are assigned to user profiles at the time of creation of the users. Some of the important aspects of password management are:

- Account Locks
- Password Complexity Verification
- Password Expiration
- Password History

5.5.1 Account Locks

The Oracle system automatically locks the account, if the user fails to provide the correct password at the time of log in. The number of failed attempts that the user is allowed can be specified while using the CREATE PROFILE statement. The amount of time that accounts may remain locked can also be specified in the CREATE PROFILE statement.

```
SQL> CREATE PROFILE slx_prof LIMIT
      FAILED_LOGIN_ATTEMPTS 5
      PASSWORD_LOCK_TIME 7;
SQL> ALTER USER slx_user01 PROFILE slx_prof;
```


☞ The statement specifies the maximum failed login attempts as 5 and in case the user fails to login after 5 attempts, the account is locked for 7 days. At the end of the seventh day the account will automatically be unlocked.

Locking Account

```
SQL> ALTER USER slx_USER01 ACCOUNT LOCK;
```

Unlocking Account Before the Password Lock Time

```
SQL> ALTER USER slx_USER01 ACCOUNT UNLOCK;
```

5.5.2 Password Complexity Verification

Oracle provides a sample password complexity verification routine (utlpwdmg.sql). This routine sets the default profile parameters and ensures that the following elementary conditions are met:

- The password must consist of at least four characters.
- The password must be different from the username. The password must have a minimum of one alpha, one number and one punctuation mark character.
- The password should not be a commonly used word such as database, Oracle account or database.
- The password has to be different from the previous passwords by at least 3 characters.

The users slx_user01 can change their passwords by using the ALTER USER statement

```
SQL> ALTER USER slx_user01 IDENTIFIED BY new_password;
```

Alter Password

The administrator with ALTER ANY USER privileges can alter any user's password.

```
SQL> ALTER USER any_user IDENTIFIED BY new_password;
```

5.5.3 Password Aging and Expiration

The administrator can specify the maximum lifetime for the passwords by using the PASSWORD_LIFE_TIME clause while creating the profile.

Setting Password Lifetime

```
PASSWORD_LIFE_TIME 10
```

☞ The password is valid only within the specified amount of time and expires thereafter.

The PASSWORD_GRACE_TIME

If the user logs in after the password expires, a warning will appear informing the user that the password will expire in the number of days as specified in the PASSWORD_GRACE_TIME clause.

```
PASSWORD_GRACE_TIME = 3.
```

5.5.4 Password History

Oracle maintains a history of all passwords used by the user. The PASSWORD_REUSE_TIME clause specifies the minimum number of days before a previously used password can be used again. If this parameter is set to UNLIMITED, the user cannot use a previously existing password. If no parameter is set, the user can reuse the password any time. The PASSWORD_REUSE_MAX clause specifies the number of times the password has to be changed for the user to be able to reuse the password. This parameter can be set to UNLIMITED. If both the above parameters are set to UNLIMITED, Oracle ignores both parameters.

5.6 Oracle Auditing

Oracle Auditing is the process of accounting and logging database activities. Auditing can also be triggered by the security policies enforced by a company. Important data is audited often to ensure data integrity against malicious data tampering. Some of the important types of auditing are:

- **Statement auditing** audits all the statements that perform select operations associated with the table:

```
SQL> Audit USER any_user IDENTIFIED BY new_password;
```

- **Privilege auditing** audits all the statements that issue a create table command:

```
SQL> Audit Create Table;
```

- **Schema Object Auditing** audits all the statements (such as Create, Drop, etc) associated with the table.

```
SQL> Audit table;
```

- **Focusing Audit statements**

```
SQL> AUDIT CREATE TABLE WHENEVER SUCCESSFUL;
--Audits only those create table statements that
are successful.
```

```
SQL> AUDIT CREATE TABLE WHENEVER NOT SUCCESSFUL;
--Audits only those create table statements that
are not successful.
```

· Fine Grained Auditing

Fine grained auditing enables auditing of actions performed on certain rows and stores the results in DBA_FGA_AUDIT_TRAIL. The DBMS_FGA utility can be used for fine grain auditing. These policies are stored in DBA_AUDIT_POLICIES.

To view current policies:

```
SQL> SELECT object_schema, object_name, policy_text,
policy_text
FROM dba_audit_policies
```

To add a policy:

```
EXECUTE DBMS.FGA.ADD_POLICY('Slx_Sales', 'Vendor',
'Policy01', product_id=1500');

SQL> ANALYSE TABLE vendor COMPUTE STATISTICS;

-- Select statements with product_id =1500 will
trigger auditing.
```

To view the SQL statements:

```
SQL> SELECT sessionid, timestamp#, sqltext
FROM sys.fga_log$;
DELETE FROM sys.fga_log$;
```

Using Audit Options**Important Views**

dba_audit_trail	All audit trail entries
dba_audit_session	All audit trail records concerning CONNECT and DISCONNECT

To enable auditing:

```
SQL> CONNECT / as sysdba
SQL> AUDIT SESSION;
```

To view common auditing information:

```
SQL> SELECT os_username, username, terminal, owner,
action, timestamp, obj_name, sessionid
FROM dba_audit_trail
ORDER By timestamp;
```

To view session audit information:

```
SQL> SELECT os_username, username, terminal, timestamp,
logoff_time
FROM dba_audit_session
ORDER BY timestamp;
```

To by-pass audit:

```
SQL> AUDIT TRUNCATE;
```

To audit roles:

```
SQL> AUDIT ROLE;-- Audits statements that include
create, drop, set, alter Roles.
```

To audit statements issued by a particular user:

```
SQL> AUDIT TABLE BY slx_user01;-- Audits statements
that include create, drop, set, alter Roles.
```

To audit directories:

```
SQL> AUDIT DIRECTORY;
-- Audits Actions performed on the directory.
```

To audit a file:

```
SQL> AUDIT WRITE ON DIRECTORY slx_file01;
-- Audits Actions performed on the file slx_file01 directory.
```

To audit all options on a column in a table:

```
SQL> AUDIT ALL ON slx_Sales.Sale_id;
-- Audits Actions performed on the column slx_id
belonging to the table slx_sales.
```

To specify default audit options for an object:

```
SQL> AUDIT ALTER, CREATE, INSERT, UPDATE, DELETE,
DROP ON DEFAULT;
```

To audit delete statements:

```
SQL> AUDIT DELETE ANY Table;
-- Audits statements that include delete Table Statement
```


Database Tuning

Contents

- Database Tuning Strategy
- SQL Tuning
- Tracing Sessions Using ORADEBUG
- Memory Tuning
- I/O Tuning
- Sort Tuning
- The Optimizer
- Optimizer Modes
- Sub-Queries/Views Optimizer Hints
- Access Optimizer Hints
- Joins Optimizer Hints
- Miscellaneous Optimizer Hints
- SQL Explain Plan

CHAPTER 6: DATABASE TUNING

6.1 Database Tuning Strategy

The following areas should be considered for tuning. The outlined steps need to be performed in order to avoid any adverse effects while tuning.

Creating an Effective Database Design

A poor database design results in poor system performance. Designers are expected to normalize the application to at least third normal form. De-normalizing selectively can result in improved performance of the database system. Processes such as data replication, data partition and tables aggregation have to be planned while designing a database.

Tuning the Application

Oracle system performance depends on the quality of the SQL code. Planning and scheduling batch jobs during off peak hours can result enhanced performance.

Tuning the Memory

Memory allocation is extremely critical for the application processes to run efficiently. Database buffers including shared pool, buffer cache and log buffers have to be planned appropriately. Buffer hit ratios should be monitored constantly. Frequently accessed objects have to be loaded into the memory.

Tuning Disk I/O Tuning

The database files have to be sized appropriately so that the disk throughput is maximized. Processes such as frequent disk sort, complete table scans, missing indexes and data fragmentation have to be monitored.

Avoiding Database Contention

Events that contribute to delay such as locks, waits, and latches have to be monitored carefully. Fixes should be implemented as soon as possible.

Operating System Tuning

Operating system parameters such as CPU utilization, I/O utilization, and memory utilization have to be closely monitored. These parameters provide important information about the system level processes.

Tuning Indicators

The following parameters can be treated as high-level tuning indicators to measure database performance:

- **Buffer Cache Hit Ratio**

$$\text{Hit Ratio} = 1 - \frac{(\text{physical reads} - \text{physical reads direct} - \text{physical reads direct}(\text{lob}))}{\text{session logical reads}}$$

DB_CACHE_SIZE has to be increased to have higher hit ratio.

· Library Cache Hit Ratio

v\$LIBRARYCACHE view: GETHITRATIO determines the percentage of calls that find a cursor to share(GETHITS/GETS). This ratio should be in the high 90s in OLTP environments. If not, you can improve the efficiency of your application code.

```
select namespace, gethitratio from v$librarycache;
```

The SHARED_POOL_SIZE has to be increased to have higher hit ratio.

Cost-based Optimization

The cost-based optimizer rewrites some of the queries related to collocated inline views. It also optimizes distributed queries based on the statistics of the referenced tables. The cost-based optimizer should have access to accurate and complete information on all the tables present in the database. If the optimizer finds any missing statistics, the optimizer reverts to rule-based optimization for the statement.

The cost-based optimizer changes the plan of execution whenever the statistics of an object change or any of the initialization parameters such as hash_join_enabled, sort_area_size, or db_file_multiblock_read_count are altered.

Related Views

Views	Description
v\$system_cursor_cache	Describes the hits and hits ratio
v\$global_transaction	Describes all the global transactions
v\$sgastat	Describes the SGA component stats
v\$sqlarea	Describes the SQL statement status
v\$sqltext	Describes the SQL statement location
v\$rollstat	Describes the rollback status
v\$statname	Describes the statistics information
v\$latch	Describes the latch information

6.2 SQL Tuning

SQL tuning increases database system performance. It involves recognizing high load, verifying execution plans and implementing corrective actions. The goal is to decrease response time and reduce resource utilization, thus enhancing availability. This can be achieved by methods such as:

- Reducing the Workload
- Balancing the Workload
- Parallelizing the Workload

The following features can be used to identify high load SQL statements:

- Automatic Database Diagnostic Monitor
- v\$sql View
- Automatic Workload Repository
- SQL Trace

Related Views

Views	Description
v\$sql	Describes the SQL attributes
v\$open_cursor	Describes the status of all open cursors
v\$sqlarea	Describes the SQL statement status
v\$sqltext	Describes the SQL statement status
v\$sql_cursor	Describes the status of the cursor hit and cursor-hit ratio
v\$sql_workarea	Describes the SQL work area attributes
v\$sql_shared_memory	Describes the SQL shared memory attributes

System Inefficiency

The view v\$sqlarea helps locate any specific SQL statement that may be responsible for system inefficiency.

Status of Most Common Resources

The status of most common resources can be found querying the following views:

- Buffer gets V\$SQLAREA.BUFFER_GETS (for high CPU using statements).
- Disk reads V\$SQLAREA.DISK_READS (for high I/O statements).
- Sorts V\$SQLAREA.SORTS (for many sorts).

System resources used by a single SQL query

The system resources used by a single SQL query can be determined by dividing the BUFFER_GETS of a single SQL statement by the total number of BUFFER_GETS during the period. The total number of BUFFER_GETS in the system can be found from V\$SYSSTAT table.

SQL text

```
SELECT sql_id, sql_text
FROM v$sqltext;
```

Describing Rows of the Table

Optimizer statistics can be used to evaluate index information and the number of rows of each table.

Automatic SQL Tuning Features

Some of the features used for SQL Tuning are:

- **Automatic Database Diagnostic Monitor (ADDM):** This feature is used to analyze all the SQL information for possible performance enhancement.
- **SQL Tuning Advisor:** SQL Tuning Advisor is used for its quick and efficient technique of optimizing SQL statements without modifying the functionality of the statements.

- **SQLAccess Advisor:** This tuning tool provides advice on materialized views and indexes.

6.2.1 Tracing Sessions Using ORADEBUG

To enable SQL Trace using oradebug:

1. Connect to sqlplus as sysdba.

```
$ sqlplus /nolog
SQL> CONNECT / as sysdba
```

2. Get the Oracle Process Identifier, O/S Process Identifier from v\$process.

```
SQL> SELECT pid, spid,username FROM v$process;
```

3. Using OS Process ID or Using Oracle Process ID.

```
SQL>ORADEBUB SETOSPID <spid#>
SQL>ORADEBUB SETORAPID <pid#>
```

4. Enable SQL Trace for the session (Includes Trace bind values and waits)

```
SQL> ORADEBUB EVENT 10046 TRACE NAME CONTEXT FOREVER, LEVEL 12
```

5. The output of the trace is located in USER_DUMP_DEST Directory and can be formatted using tkprof.

```
SQL> SELECT value FROM v$parameter
WHERE name='user_dump_dest';
```

Semi-Join

A semi-join between two tables returns rows from the first table where one or more matches are found in the second table. Semi-joins are written using EXISTS or IN.

Anti-Join

An anti-join between two tables returns rows from the first table where no matches are found in the second table. An anti-join is essentially the opposite of a semi-join. Anti-joins are written using the NOT EXISTS or NOT IN.

6.3 Memory Tuning

Tuning operations related to the memory have to be performed after tuning the application and before tuning the I/O. The SGA must be able to fit in the physical memory at all times.

To view the status of the SGA:

```
SQL> SELECT * FROM v$sga;
```

To preload the SGA into the memory:

```
Set the parameter PRE_PAGE_SGA = yes in the init.ora file.
```

Tuning Library Cache

The library cache and the data dictionary cache have to be tuned optimally. The library cache contains all the SQL and PL/SQL areas.

To view the performance of the library cache:

```
SQL> SELECT SUM(pins), SUM(reloads)
        FROM v$librarycache;
```

☞ The library cache can be tuned by increasing the `SHARED_POOL_SIZE` and the `OPEN_CURSORS` in the `inti.ora` file.

To ensure that the shared SQL area is never de-allocated when an associated cursor is open:

```
cursor_space_for_time = TRUE;
```

To set the number of cursors that can be cached for a user session:

```
Session_cached_cursors = 50;
```

6.4 I/O Tuning

I/O tuning should be attempted only after tuning memory. It is good practice not to place all Oracle files on the same disks as non-Oracle files. The main objective is to minimize the concurrent access to the disk.

Information regarding the Physical reads and Physical writes:

```
SQL> SELECT name, phyrds, phywrts
        FROM v$datafile, v$filestat
        WHERE v$datafile.file# = v$filestat.file#;
```

☞ Chained rows are also another cause for bottlenecks. Running the `utilchai.sql` provided by Oracle eliminates the chained rows.

6.5 Sort Tuning

Sorting is one of the most expensive operations. It is critical for the sorting operation to be completed in the memory and not let it extend to the physical disk.

To find number of sorts on memory and the number of sorts on disk:

```
SQL> SELECT name, value
       FROM v$sysstat
       WHERE UPPER (name) IN ('sorts (mem)', 'sorts (disk)');
```

6.6 The Optimizer

The optimizer determines the most efficient way to access data. It is used to process the SQL statements. The efficiency of retrieving data or performing an action can vary depending on the order in which the tables or indexes are accessed.

6.6.1 Optimizer Modes

FIRST_ROWS, ALL_ROWS	Forces CBO to base plan to choose first rows or all rows most efficiently
RULE	Forces rule if possible and also disables the query optimizer
ORDERED	Forces the access of tables in the order of appearance in the FROM clause
ORDERED_PREDICATES	Does not apply predicate evaluation on index keys

6.6.2 Sub-Queries/Views Optimizer Hints:

PUSH_SUBQ	Subqueries in a query block to be executed are evaluated at the earliest possible time. Normally subqueries are executed at the end or applied after outer join or remote join with a merge join
NO_MERGE (v)	Prevents a mergeable view to be merged into the parent query
PUSH_SUBQ	Causes all subqueries in a query block to be executed at the earliest possible time. Normally subqueries are executed at the end
MERGE (v)	Merges a view
PUSH_JOIN_PRED (v)	Pushes join predicates into view
NO_PUSH_JOIN_PRED (v)	Does not allow pushing of join predicates

6.6.3 Access Optimizer Hints

FULL (tab)	Use Full table scan on tab
CACHE (tab)	The retrieved blocks are placed in the most recently used end of the least recently used list

NOCACHE (tab)	The retrieved blocks are placed in the least recently used end of the least recently used list
ROWID (tab)	To access tab by ROWID directly
CLUSTER (tab)	Choose cluster scan explicitly for a specific table
HASH (tab)	Choose hash scan explicitly for a specific table
INDEX (tab [ind])	Choose index scan explicitly for a table
INDEX_ASC (tab [ind])	Choose index scan explicitly for a table and scan index in ascending order of their values
INDEX_FFS (tab [ind])	Index fast full scans - rather than FTS
INDEX_COMBINE (tab i1. i5)	Chooses bitmap indexes explicitly
INDEX_SS (tab [ind])	Chooses index skip scan specifically
AND_EQUAL (tab i1. i5)	Merge scans of 2 to 5 single column indexes
NO_EXPAND	Prevents from OR-expansion
DRIVING_SITE (table)	Forces query execution to be done at the site where a table resides

6.6.4 Miscellaneous Optimizer Hints

APPEND	Only valid for INSERT. SELECT. Allows INSERT to work like direct load or to perform parallel insert. NOAPPEND Does not use INSERT APPEND functionality.
REWRITE (v1 [, v2])	Forces the optimizer to rewrite the query with respect to the materialized view
NOREWRITE	Does not allow rewriting of the query

6.7 SQL Explain Plan

The Oracle optimizer chooses a sequence of operations required to execute any SQL statement containing the SELECT, INSERT, UPDATE and DELETE instructions. The EXPLAIN PLAN statement can be used to display the chosen order of the operations. The performance of the system can be analyzed by understanding the query.

The Output Table

The output table has to be created to view the reports generated by the Explain Plan statement. In the Unix environment this can be setup by running the UTLXPLAN.SQL script.

This script is located in the \$ORACLE_HOME/rdbms/admin directory.

To run Explain Plan statement:

```
EXPLAIN PLAN FOR
  SELECT table_name FROM all_tables;
```

To set statement ID:

```
EXPLAIN PLAN    SET STATEMENT_ID = 'St_01' FOR
  SELECT table_name FROM all_tables;
```

To specify a different storage location for the result:

```
EXPLAIN PLAN
  SELECT table_name FROM all_tables;
```

To view the report:

```
SELECT CARDINALITY "rows", operation, cost, options
  FROM plan_table;
```


Backup and Recovery

Contents

- Backup and Recovery Mechanisms
- Cold or Off-line Backups
- Hot or On-line Backups
- Export/Import
- Common Import/ Export Errors
- SQL*Loader
- Standby Database
- Oracle Data Guard
- Database Recovery using Redo Log Files

CHAPTER 7: BACKUP AND RECOVERY

Overview

The main purpose of Oracle Database backup is to physically back up the database files. These files include the control files, server parameter files, archived redo file and the datafiles. All these files are sufficient to recover the database files. At the physical level the backup mechanisms provide safe recovery against accidental deletions or failure of the disk drive.

Related Views

Views	Description
v\$backup	Describes the backup file status
v\$backup_device	Describes the backup device configurations
v\$backup_redolog	Describes the redo log files status
v\$archive	Describes the archive status
v\$backup_corruption	Describes the corrupted block status
v\$backup_piece	Describes the backup pieces status
v\$recovery_status	Describes the recovery status
v\$backup_datafile	Describes the backup related datafiles
v\$archive_dest	Describes archive destinations

7.1 Backup and Recovery Mechanisms

Export/Import

The logical definitions from the database are extracted to a file. These files are logical in nature and cannot ensure complete recovery of the database.

RMAN Backups

RMAN is a utility used to back up the entire database.

Restoring and Backing up

In order to restore a database, all backup files from a backup media such as a secondary storage device are copied to the primary disk. Damaged files can be replaced or the database itself can be moved to a new location.

For the database to be completely up to date, all the redo logs have to be applied to the database. The user can roll forward to a specific time or to a specific transaction that is recorded in the log files.

On-line database backups

In order to back up the database, each tablespace of the database must be switched to backup mode before the files can be copied to the secondary storage (tapes).

```
SQL> ALTER TABLESPACE slx_tbs01 BEGIN BACKUP;
      -- copy slx_tbs01File1 to /backupDir/
ALTER TABLESPACE slx_tbs01 END BACKUP;
```

- ☞ It is a good practice to back up each tablespace one by one instead of placing all the tablespace in the backup mode. This results in lesser overhead.
- ☞ At the end of the backup process, the control files have to be backed up.

To switch the log file:

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

To switch to update control file headers:

```
SQL> ALTER DATABASE
      BACKUP CONTROLFILE
TO '/mnt/sdc1/1159//slx_db01/backupDir/control.dbf';
```

- ☞ The backups cannot be run at the time of peak processing.
- ☞ Oracle writes the entire database blocks in the redo log files while in the backup mode. This results in huge archive files that can freeze the database.

ARCHIVELOG mode

The archive log mode provides higher availability and guarantees complete recoverability. It is necessary to put the database in ARCHIVELOG mode before starting backup of the database online.

To enable ARCHIVELOG mode:

```
1. CONNECT SYS AS sysdba
2. STARTUP MOUNT EXCLUSIVE;
3. ALTER DATABASE ARCHIVELOG;
4. ARCHIVE LOG START;
5. ALTER DATABASE OPEN;
```

7.1.1 Cold or Off-line Backups

Cold backing up is done while the database is shut down or offline. The database is unavailable to users. Following are the files that need to be backed up in order to provide a complete copy of the database:

- Datafiles
- Control files
- Init.ora
- Online redo files

```

☞ Make the Backup scripts to obtain all the files
to be backed up.

$ cat listfiles.sql
    set echo off
    set pause off
    set feed off
    set head off
    spool bck_cold.sh

SELECT 'cp' ||name || 'BACKUP' FROM v$datafile;
SELECT 'cp' ||member || 'BACKUP' FROM v$logfile;
SELECT 'cp' ||name || 'BACKUP' FROM v$controlfile;

spool off

☞ SQL> connect sys AS sysdba.
☞ SQL> STARTUP
☞ SQL>@listfiles.sql
☞ SQL>SHUTDOWN IMMEDIATE
☞ SQL> EXIT
☞ sh bck_cold.sh

```

7.1.2 Hot or On-line Backups

The database is put in ARCHIVELOG mode and the tablespaces are set to backup mode. The database is still available for read/write operations. All files including data files, control files and server parameter files have to be backed up. The database is not required to be in ARCHIVELOG mode while performing Oracle exports.

7.2 Export/Import

Oracle provides two utilities; export (exp) and import (imp), to perform logical database backup and recovery. These utilities move Oracle data from one machine to another using a binary file format. The binary file format can be used only between Oracle systems. The binary format is mostly used to back up small databases. The Oracle Datapump has replaced this utility in Oracle version 10g. Data can be reorganized or database corruption can be detected using these utilities. They allow seamless transfer of tablespaces between databases.

Two executables called imp and exp can be found in the \$ORACLE_HOME/bin directory. These executables can be run from the command line parameters or by using the parameter files.

The data pump has replaced the imp and exp in Oracle 10g.

```

EXP Slx_user01/slx123
    File = slx_hr.dmp
    log=slx_hr.log tables=slx_hr
    rows=yes indexes=no

EXP Slx_user01/slx123
    file=hr.dmp tables=(emp_id,dept)

imp Slx_user01/ slx123
    file=hr.dmp full=yes

imp Slx_user01/ slx123
    file=hr.dmp from user=Slx_user01
    TO user= slx_user01

tables=dept

EXP userid= Slx_user01/ slx123@slx_dbs01

parfile=export.txt.

The export.txt contains:
    BUFFER=100000
    FILE=hr.dmp
    FULL=n
    OWNER=Slx_user01
    GRANTS=y
    COMPRESS=Y

```

7.2.1 Common Import/ Export Errors:

ORA-00001: Unique constraint violated.

Duplicate rows are being imported. By setting IGNORE=NO all tables that already exist can be skipped. The IMP utility is designed to display error messages in case duplicated objects are being created.

ORA-01555: Snapshot too old.

Users have to be notified to stop work or the parameter CONSISTENT has to be set to NO.

ORA-01562: Failed to extend rollback segment.

Bigger rollback segments have to be created or the parameter COMMIT has to be set to 'Y' while importing.

IMP-00015: Statement failed ... object already exists.

Set the parameters IGNORE to Y. Using this option may result in duplicate rows being created.

7.3 SQL*Loader

The SQL*Loader moves data from external sources into an Oracle Database. SQL*Loader can support various data formats, selective loading and multi-table loading.

```
SQLLDR slx_user01/slx123 CONTROL=loader.ctl
```

Sample control file

The sample control file (loader.ctl) loads the data file containing delimited data.

```
LOAD DATA
  INFILE '/mnt/sdc1/data/data01.csv'
  INTO TABLE employee
  FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
  (emp_id, emp_name, dept_id, sal)

-- The data01.csv file may look like this: 01212,
  "Slx_emp02", 5012, 88000 01213,
  "Slx_emp02", 5012, 42000
```

The control file with data included

```
INFILE '/mnt/sdc1/1159/slx_db01/s/data/data01.csv'
  INTO TABLE employee
  FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
  (emp_id, emp_name, dept_id, sal)

-- The data01.csv file may look like this: 01212,
  "Slx_emp02", 5012, 88000 01213,
  "Slx_emp02", 5012, 42000
```

To load variable length delimited data records:

```
LOAD DATA
  INFILE * I
  INTO TABLE load_delimited_data
  FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
  TRAILING NULLCOLS
  (data1,
   data2)
BEGINDATA
  01212, "Slx_emp02", 5012, 88000
  01213, 5456, 42000
```

To load variable and fixed length data records:

```
LOAD DATA
  INFILE *
  INTO TABLE load_positional_data
  (data1 POSITION(1:5), data2 POSITION(6:11)
  data3 POSITION(12:15) data4 POSITION(16:20) )
BEGINDATA
  01212Slx_emp02501288000
  01213Slx_emp02501242000
```

To modify data as it is loaded into the database:

```

LOAD DATA
  INFILE *
  INTO TABLE modified_data
  (rec_no "seq.nextval", region
  CONSTANT'31',time_loaded "to_char(SYSDATE, 'HH24:MI')",
  data1 POSITION(1:5) ":data1/100",
  data2 POSITION(6:11) "upper(:data2)",
  data3 POSITION(16:22))

BEGINDATA
  01212Slx_emp02501288000
  01213Slx_emp02501242000

```

To load data into multiple tables at once:

```

LOAD DATA
  INFILE * REPLACE
  INTO TABLE hr
  WHEN empno != ' '
  ( empno POSITION(1:4) INTEGER EXTERNAL,
  ename POSITION(6:15) CHAR,
  deptno POSITION(17:18) CHAR,
  mgr POSITION(20:23) INTEGER EXTERNAL
  )
  INTO TABLE dept
  WHEN dept_no != ' '
  ( dept_no POSITION(25:27) INTEGER EXTERNAL,
  empno POSITION(1:4) INTEGER EXTERNAL
  )

```

To skip certain columns while loading data:

```

LOAD DATA
  TRUNCATE INTO TABLE T1
  FIELDS TERMINATED BY ','
  ( field1,
  field2 FILLER,
  field3
  )

```

Loading multi-line records

One logical record can be created from multiple physical records using **CONCATENATE** clause or the **CONTINUEIF** clause.

CONCATENATE: Combines the same number of physical records together to form one logical record.

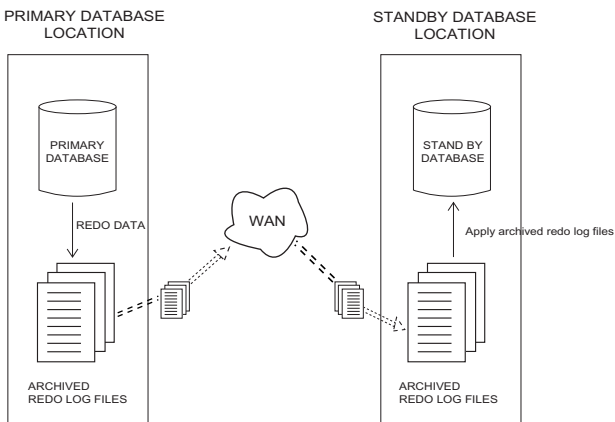
CONTINUEIF: Used to indicate that a multiple record set should be treated as one

7.4 Standby Database

A duplicate copy of the production database is always maintained as a standby database and is used in the event of failure or data corruption of the production data. In order to facilitate complete recovery all the datafiles, control files, and redo log files of the primary and secondary databases are backed up onto secondary storage, typically in a remote location. The standby database can also be used to balance load in conjunction with the primary database by serving the reporting requests while in read only mode. The standby databases are maintained using the Oracle Data Guard. The Oracle Data Guard was introduced in Oracle versions 9i and higher.

7.4.1 Oracle Data Guard

Oracle Data Guard can maintain up to nine standby databases. In case the primary database fails, one of the standby databases can take over as the primary database. The Oracle Data Guard automatically maintains all copies of the standby database by transmitting all the redo data from the redo log files of the primary database to each of the standby databases, and then applying the redo to the standby database. The standby database can either be physical or logical in nature.



DISASTER RECOVERY

Figure 11. *Data Guard Concept.*

The Physical Standby Database

The physical database maintains a block-by-block, physically identical copy of the primary database. The synchronization is made possible by applying the redo data from the primary database.

The Logical Standby Database

The logical standby database consists of the logical information consistent with the primary database but with possible differences in the physical organization of the database. The logical database is synchronized to the primary database by applying the redo log files of the primary database. The logical database is also used to serve the reporting requests from users and helps in the upgrade of the database.

Using the command line interface to create the physical standby database

The primary database and the replica of the primary database chosen to be the standby database have to be created at different locations. The `DG_BROKER_START` parameter must be set to `TRUE` in the initialization parameter file.

To enable the standby database:

```
%DGMRL
DGMRL> CONNECT sys/ password.
-- Connect to primary database on the local station.

DGMRL> CONNECT sys/password @primary_db01.kr.slx.com
--Connect to the stand by database.

DGMRL> CREATE CONFIGURATION 'Broker01' AS
        PRIMARY DATABASE IS 'Primary_DB01'
        CONNECT IDENTIFIER IS Primary_DB01.kr.slx.com;
-- Primary_DB01 is the DB_UNIQUE_NAME

DGMRL> ADD DATABASE 'Standby_DB01' AS
        CONNECT IDENTIFIER IS Standby_DB01.kr.slx.com
        MAINTAINED AS PHYSICAL;
-- Standby_DB01 is the DB_UNIQUE_NAME

DGMRL> SHOW CONFIGURATION;
--Verify the status of the Configuration.

DGMRL> EDIT DATABASE 'Standby_DB01'
SET PROPERTY 'LogArchiveFormat'='log_%t_%s_%r_%d.arc';
-- Log archive format standard.

DGMRL> EDIT DATABASE 'Standby_DB01'
SET PROPERTY 'StandbyArchiveLocation'='/disk01/archive/';
-- Sets the Archive location

DGMRL> SHOW DATABASE VERBOSE 'Standby_DB01';
-- Verify database status.

DGMRL> ENABLE CONFIGURATION;
--Enables the configuration.

DGMRL> SHOW CONFIGURATION;
--Verify the configuration.

DGMRL> ENABLE DATABASE 'Standby_DB01';
-- Enable the database.

DGMRL> SHOW DATABASE 'Standby_DB01';
-- Verify the configuration.
```

To set the transport mode log of the standby database to SYNC

```
DGMRL> EDIT DATABASE 'Standby_DB01'
SET PROPERTY 'LogXptMode'='SYNC';
```


To configure the protection mode to MAXPROTECTION:

```
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE
AS MAXPROTECTION;
```

To issue the SWITCHOVER command:

```
DGMGRL> SWITCHOVER TO "Standby_DB01";
```

To issue the FAILOVER command:

```
DGMGRL> CONNECT sys/knl_test7@Standby_DB01.kr.slx.com

DGMGRL> FAILOVER TO "Standby_DB01";
-- This makes the standby database the primary database.
```

To disable the configuration:

```
DGMGRL> DISABLE CONFIGURATION;
```

To disable the standby database:

```
DGMGRL> DISABLE DATABASE 'Standby_DB01';
```

7.4.2 Database Recovery Using Redo Log Files

Oracle Database generates all the names of redo log files

```
SQL> ALTER DATABASE
RECOVER AUTOMATIC DATABASE;
```

To specify the applicable Oracle redo log file:

```
SQL> ALTER DATABASE
RECOVER LOGFILE '/u02/oracle/dbs/log3.log';
```

To recover a standby database:

This recovery includes datafile, archived logs and the current control file.

```
SQL> ALTER DATABASE
RECOVER STANDBY DATAFILE
'/mnt/sdc1/1159/slx_db01/slx_tbs_2.f'
UNTIL CONTROLFILE;
```

To perform a time based recovery of the database:

```
SQL> ALTER DATABASE
RECOVER AUTOMATIC UNTIL TIME '2005-01-2:13:00:00';
```

To recover a database in the managed standby mode:

```
SQL> ALTER DATABASE
RECOVER MANAGED STANDBY DATABASE;
```

Recover databases in the managed standby mode:

```
SQL> ALTER DATABASE  
RECOVER MANAGED STANDBY DATABASE TIMEOUT 60;
```

- ☞ The recovery process waits up to 60 minutes for the next archive log.
- ☞ In case the subsequent log arrives before the timeout (60 minutes), the recovery continues indefinitely till it is manually terminated.

To terminate the managed recovery operation:

```
SQL> ALTER DATABASE  
RECOVER MANAGED STANDBY DATABASE CANCEL IMMEDIATE;
```


Data Dictionary and Built-In Packages

Contents

- Useful Admin Tables
- Useful v\$ Views
- Packages
- DBMS_SQL
- DBMS_JOB
- DBMS_SPACE
- The DBMS_OUTPUT

8.1 Useful Admin Tables

TABLE NAME	Description
dba_tables	Description of all relational tables in the database
dba_tab_columns	Description of all the columns of user's tables, views and clusters
dba_indexes	Description for all indexes in the database
dba_ind_columns	Description of all the columns comprising Indexes on all tables and clusters
dba_constraints	Information about accessible columns in constraint definitions
dba_segments	Description of the storage allocated for all database segments
dba_objects	Description all objects in the database
dba_data_files	Information about database datafiles
dba_free_space	Description of the free extents in all tablespaces
dba_tablespaces	Description of all tablespaces
dba_users	Information about all users
dba_ts_quotas	Description of the tablespace quotas to all users
dba_roles	Description of all roles that exist in the database
dba_role_privs	Description of the roles granted to users and privileges
dba_sys_privs	Description of the system privileges granted to users and roles.
dba_tab_privs	Description of all grants on objects in the database
dba_rollback	Description of all grants on objects in the database
dba_locks	Description of all locks on sessions
dba_triggers	Description all triggers in the database
dba_db_links	Description of all database links in the database

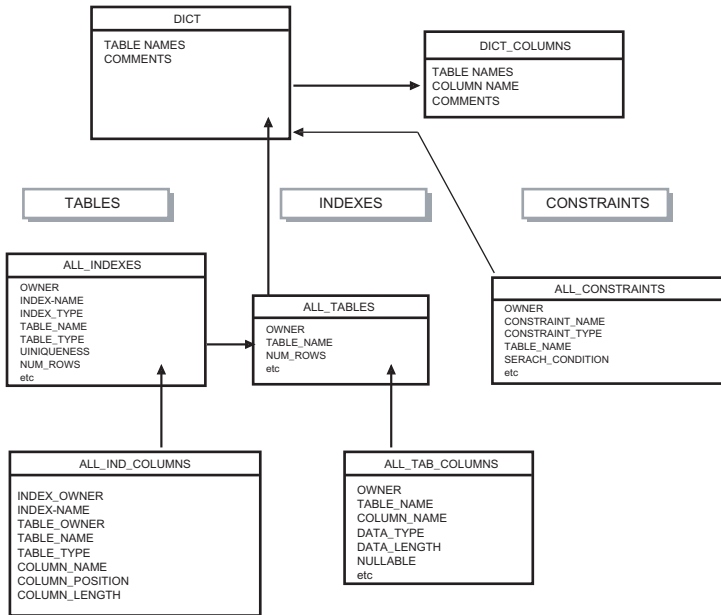


Figure 12. Data Dictionary View.

8.2 Useful V\$ VIEWS :

Views	Description
v\$session	Information about all the sessions
v\$database	Information about the database
v\$version	Information about the version of the database that has been installed
v\$controlfile	Location and status of the control files
v\$logfile	Location and status of the log files
v\$lock	Description of all the locks
v\$rollstat	Statistics of the rollback segments
v\$sort_segment	Information related to sorting
v\$lock_activity	Description of lock activity period
v\$latch_misses	Description of latch misses location
v\$locked_object	List of all locked objects
dba_summaries	Description of the summaries accessible to DBA
dba_summary_aggregates	Description of the aggregates of the summaries accessible to DBA
dba_summary_detail_tables	Description of the detail tables accessible to DBA

8.3 Packages

8.3.1 DBMS_SQL

Stored procedures can be written and used dynamically during runtime. Using the dynamic SQL statements gives more flexibility to the user. Procedures can be created without knowing which tables they will operate on until runtime. The DBMS_SQL package allows the user to place the procedure created by the user into PL/SQL blocks.

DBMS_SQL package has been replaced by the EXECUTE IMMEDIATE statement. This statement displays much more user friendly error messages. The statement in quotes is parsed and executed immediately. If the statement is of DML in nature, commit is not performed implicitly.

Using 'EXECUTE IMMEDIATE'

```
declare
  l_count  number;
begin
  execute immediate 'select count(*) from emp'
    into l_count;
  dbms_output.put_line(l_count);
end;
```

8.3.2 DBMS_JOB

This package manages and schedules all jobs in the job queue. The DBMS_SCHEDULER supersedes the DBMS_JOB package. The user needs no specific privileges to use this package. This package does not allow jobs to be deleted or altered. Only the owner of the job can alter the job status.

To submit a job to the queue:

```
DBMS_JOB.SUBMIT (JOB          OUT      BINARY_INTEGER,
                 WHAT        IN       VARCHAR2, NEXT_DATE IN DATE
                 DEFAULTSYSDATE,
                 INTERVAL IN   VARCHAR2 DEFAULT 'NULL',
                 NO_PARSE IN   BOOLEAN DEFAULT FALSE,
                 INSTANCE IN   BINARY_INTEGER DEFAULT
                 ANY_INSTANCE,
                 FORCE        IN   BOOLEAN DEFAULT FALSE);
```

To assign an instance to execute a job:

```
DBMS_JOB.INSTANCE (JOB IN BINARY_INTEGER,
                  INSTANCE IN BINARY_INTEGER, FORCE
                  IN BOOLEAN DEFAULT FALSE);
```

☞ The INSTANCE has a default value of NULL indicating that job affinity will not change.

To run the DBMS_JOB package:

```
DBMS_JOB.RUN (JOB      IN BINARY_INTEGER,
              FORCE IN BOOLEAN DEFAULT FALSE);
```

To change the parameter of the DBMS_JOB:

```
DBMS_JOB.CHANGE (job      IN BINARY_INTEGER,
                 what      IN VARCHAR2,
                 next_date IN DATE,
                 interval  IN VARCHAR2,
                 instance  IN BINARY_INTEGER DEFAULT NULL,
                 force     IN BOOLEAN DEFAULT FALSE);
```

To schedule the running of DBMS_JOB package:

```
DBMS_JOB.INTERVAL (job IN BINARY_INTEGER,
                  interval IN VARCHAR2);
```

To remove the job from the job queue:

```
DBMS_JOB.REMOVE (job IN BINARY_INTEGER);
```

To submit a new job to the DBMS_JOB package:

```
DBMS_JOB.SUBMIT (
  job      OUT BINARY_INTEGER,
  what     IN  VARCHAR2,
  next_date IN DATE DEFAULT sysdate,
  interval IN VARCHAR2 DEFAULT 'null',
  no_parse IN BOOLEAN DEFAULT FALSE,
  instance IN BINARY_INTEGER DEFAULT any_instance,
  force    IN  BOOLEAN DEFAULT FALSE);
```

To submit a new job to the job queue:

The job calls the DBMS_DDL.ANALYZE_OBJECT procedure to generate optimizer statistics for the DQUON.ACCOUNTS table. The statistics are based on a sample of half of the rows of the ACCOUNTS table. The job is run every 24 hours.

```
VARIABLE job_number;
BEGIN
  DBMS_JOB.SUBMIT(:jobno,
  'dbms_ddl.analyze_object(''TABLE'',
  ''Sales'', ''VENDORS'',
  ''Products'', NULL, 50);'
  SYSDATE, 'SYSDATE + 1');
  COMMIT;
END;
/

Statement processed.
print jobno
JOBNO
-----
19028
```


8.3.3 DBMS_SPACE

This package is a collection of procedures used to analyze the space requirements. This package can be run only with SYS privileges.

CREATE_INDEX_COST Procedure

This procedure estimates the cost incurred while creating an index on a table.

```
DBMS_SPACE.CREATE_INDEX_COST (
    ddl                IN   VARCHAR2,
    used_bytes         OUT  NUMBER,
    alloc_bytes        OUT  NUMBER,
    plan_table         IN   VARCHAR2 DEFAULT NULL);
```

CREATE_TABLE_COST Procedures

This procedure finds the size of the table, if the attributes are specified.

```
DBMS_SPACE.CREATE_TABLE_COST (
    tablespace_name   IN   VARCHAR2,
    avg_row_size      IN   NUMBER,
    row_count          IN   NUMBER,
    pct_free           IN   NUMBER,
    used_bytes         OUT  NUMBER,
    alloc_bytes        OUT  NUMBER);
```

FREE_BLOCKS Procedure

```
DBMS_SPACE.FREE_BLOCKS (
    segment_owner     IN   VARCHAR2,
    segment_name      IN   VARCHAR2,
    segment_type      IN   VARCHAR2,
    freelist_group_id IN   NUMBER,
    free_blks         OUT  NUMBER,
    scan_limit        IN   NUMBER DEFAULT NULL,
    partition_name    IN   VARCHAR2 DEFAULT NULL);
```

OBJECT_DEPENDENT_SEGMENTS

```
DBMS_SPACE.OBJECT_DEPENDENT_SEGMENTS (
    objowner   IN   VARCHAR2,
    objname    IN   VARCHAR2,
    partname   IN   VARCHAR2,
    objtype    IN   NUMBER)
RETURN dependent_segments_table PIPELINED;
```

OBJECT_GROWTH_TREND

This procedure describes the space used by an object at a specific point in time.

```
DBMS_SPACE.OBJECT_GROWTH_TREND (
  object_owner  IN    VARCHAR2,
  object_name   IN    VARCHAR2,
  object_type   IN    VARCHAR2,
  partition_name IN    VARCHAR2 DEFAULT NULL,
  start_time    IN    TIMESTAMP DEFAULT NULL,
  end_time      IN    IMESTAMP  DEFAULT NULL,
  interval      IN    DSINTERVAL_UNCONSTRAINED
                DEFAULT NULL,
  skip_interpolated IN  VARCHAR2 DEFAULT 'FALSE',
  timeout_seconds IN  NUMBER  DEFAULT NULL,
  single_datapoint_flag IN VARCHAR2 DEFAULT 'TRUE')
RETURN object_growth_trend_table PIPELINED;
```

SPACE_USAGE Procedure

This procedure finds the free blocks in an auto space managed segment.

```
DBMS_SPACE.SPACE_USAGE (
  segment_owner  IN  VARCHAR2,
  segment_name   IN  VARCHAR2,
  segment_type   IN  VARCHAR2,
  unformatted_blocks OUT NUMBER,
  unformatted_bytes OUT NUMBER,
  fs1_blocks     OUT NUMBER,
  fs1_bytes      OUT NUMBER,
  fs2_blocks     OUT NUMBER,
  fs2_bytes      OUT NUMBER,
  fs3_blocks     OUT NUMBER,
  fs3_bytes      OUT NUMBER,
  fs4_blocks     OUT NUMBER,
  fs4_bytes      OUT NUMBER,
  full_blocks    OUT NUMBER,
  full_bytes     OUT NUMBER,
  partition_name IN  VARCHAR2 DEFAULT NULL);
```

8.3.4 The DBMS_OUTPUT

This package enables the user to send messages from stored procedures, packages and triggers. The PUT_LINE and GET_LINE procedures place the information in the buffer. This information can be displayed using the GET_LINE procedure.

To display the line information:

```
DBMS_OUTPUT.PUT_LINE ('This is where I am:'||: new.col||'
  is the new value');
```

To get back the line information:

```
BEGIN
    DBMS_OUTPUT.GET_LINE (: buffer: status);
END;
```

Sample Usage

```
CREATE FUNCTION dept_salary (dnum NUMBER)
RETURN NUMBER IS
CURSOR emp_cursor IS
SELECT sal, com FROM employee WHERE deptno = dnum;
total_wages NUMBER(11, 2) := 0;
counter NUMBER(10) := 1;
BEGIN
    FOR emp_record IN emp_cursor LOOP
        emp_record.comm := NVL(emp_record.comm, 0);
        total_wages := total_wages + emp_record.sal
            + emp_record.comm;
        DBMS_OUTPUT.PUT_LINE('Loop number = ' || counter ||
            '; Wages = ' || TO_CHAR(total_wages));
        counter := counter + 1; /* Increment debug counter */
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Total wages = ' ||
        TO_CHAR(total_wages));
    RETURN total_wages;
```

```
END dept_salary;
```

EMPLOYEE table contains the following rows:

EMPNO	SAL	COM	DEPT
1002	2500	500	05
1003	1000		07
1005	5000		01
1007	1200	250	07

To view sample output:

```
SET SERVEROUTPUT ON
VARIABLE salary NUMBER;
EXECUTE :salary := dept_salary(20);
```

Sample Output

```
Loop number = 1; Wages = 2000
Loop number = 2; Wages = 3250
Total wages = 3250
```

The errors associated with this package are:

```
ORU-10027: Buffer overflow.
ORU-10028: Line length overflow.
```

DBMS_OUTPUT Package Subprograms

Subprogram	Description
DISABLE PROCEDURE	Disables message output
ENABLE PROCEDURE	Enables message output
GET_LINE PROCEDURE	Retrieves one line from buffer
GET_LINES PROCEDURE	Retrieves an array of lines from buffer
PUT PROCEDURES	Places a line in the buffer
PUT_LINE PROCEDURES	Places partial line in the buffer
NEW_LINE PROCEDURE	Terminates a line created with PUT

New Features in 9i and 10g

Contents

- Version 9i Enhancements
- Improvements in Oracle Version 10g

CHAPTER 9: NEW FEATURES IN 9i AND 10g

9.1 Version 9i. Enhancements

9.1.1 Manageability

Database complexity increases with every version of Oracle. Important features have been added to version 9i to make the database more manageable. Notable improvements have been done to the Oracle Enterprise Manager. These improvements are outlined below.

Datafiles

- Datafiles can be automatically managed by Oracle.
- The datafiles can be created, named, resized and deleted by Oracle.

SGA

- Most of the components of the SGA can be dynamically sized.
- SGA_MAX_SIZE - Sets the largest allowable value.
- DB_BLOCK_BUFFER is replaced by DB_CACHE_SIZE.
- DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE can be altered without shutting down the database.
- SGA and buffer size always remain static.
- The view v\$sql_plan and v\$sql_workarea can be joined to view memory usage of every SQL statement.

Dataguard

- Dataguard is not an add-on anymore and can be managed by the OEM.
- The standby database can be synchronized to the production database by applying the redo log files to the standby database on a real-time basis. This ensures zero data loss. However, this results in considerable performance penalties.

LOB

- Long columns can be easily converted to CLOBs.
- LOB columns can be manipulated just like the long columns.
- Character function can be applied to CLOBs just like varchar2 columns.

Index Monitoring

- The status of the Index can be viewed using the view v\$object. And v\$object_usage.

Block Sizes

- The standard block size is set using the DB_BLOCK_SIZE. This is used by the SYSTEM and the temporary tablespaces.
- Block sizes for application tablespace can be set to be different from the standard block size.

DBMS_REDEFINITION

- Actions such as renaming columns, adding indexes and moving table spaces can be done with the users being online. This is made possible by the DBMS_REDEFINITION package.
- Users can query and write to the database while the redefinition is taking place.
- Database default temporary tablespace can be specified for each database.

Rollback segments

- Rollback segments can be automatically managed.

Timestamp

- This new data type can include time zone information to automatically adjust for day light savings time.

String Lengths

- String lengths can be specified in terms of the number of characters.
- NLS_LENGTH_SEMANTIC can be used to toggle string length setting from characters or bytes.

9.1.2 Performance

The most important improvement with respect to performance is the introduction of the Real Application Clusters (RAC). This feature was earlier called the Oracle Parallel Server.

RAC

- The database can be implemented on multiple servers. This enhances the processing power of the database system. This system is called the real application clusters.
- RACs are a collection of software and hardware that form a single robust computing environment.
- All active instances share a single database and can concurrently execute their respective transactions. Each transaction is coordinated in order to provide data integrity and concurrency.
- A single point of failure is avoided.

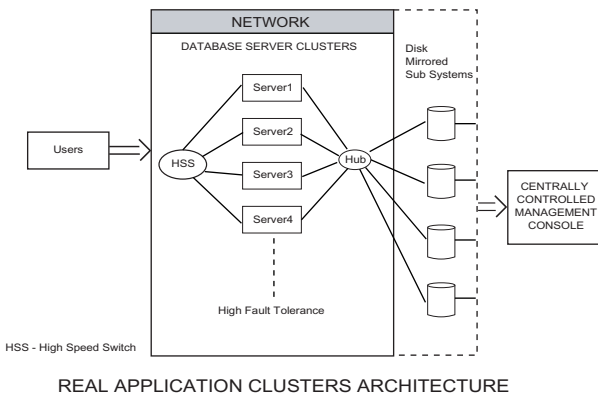


Figure 13. Real Application Clusters.

Cost-based Optimizer

- The optimizer considers the values of the bind variables to determine the selection of the predicates.
- MERGE can be used to update or insert rows.
- The optimizer determines the selective predicates in the where clause.
- For most SQL queries a join to the target table is performed.
- The CASE function of the ANSI SQL standard is also specified.

SQL & PL/SQL

- Oracle provides extensive support for ANSI standards.
- Oracle supports composite partitioning on list of values as well as list of ranges.
- Stored procedures can be natively compiled by setting `plsql_compiler_flags` to `NATIVE`.
- Oracle supports pipelining where the row output from a function can be directly passed into the next operator. This feature enhances the parallel processing ability.
- The SQL engine parser is used when evaluating the SQL statements in PL/SQL.

Flashback

- The image of the data at a previously set time point can be retrieved even after completely processing any updates performed beyond that time point.
- The data is reconstructed using the undo information.
- Flashback to a specific SCN is also possible.

Cache Hit Ratio

- Oracle is capable of estimating the cache hit ratios for various sizes of buffer caches.
- The view `v$db_cache_advice` can be queried for buffer pool sizes which replaces `v$current_bucket`.
- The physical I/O estimates can be obtained for different cache sizes.

9.1.3 Security Improvement**Three-Tier Security**

- Credential proxy for X.509 certificates enabled.
- Secure connection pooling and integration with LDAP enabled.
- Supports PKCS 12 certificates.

Fine Grain Auditing

- Enables audit of access to confidential tables.
- Complete audit trail of SELECT statements can be implemented using the `DBMS_FGA` package.
- Easy to implement standard audit requirements

Single Sign-on

- Single sign-on for Oracle server is enabled.

Encryption of Stored Data

- Supports fine grain access control.
- Supports most advanced encryption algorithms.

9.2 Improvements in Oracle Version 10g

Automatic Database Diagnostic Monitor (ADDM)

ADDM is a self-diagnostic monitor that identifies problems and devices methods of solving them. This monitor automatically checks the state of the database and its workload at regular intervals by obtaining the snapshot of the entire database. ADDM proactively examines the system for performance problems due to bottlenecks in CPU performance, memory, network connections and lock contention. The ADDM retains the snapshots in the SYSAUX tablespace and purges them after a specified time. ADDM can also be run manually from the Enterprise Manager web page. Some of the parameters measured by the ADDM are:

- Issues related to the CPU load
- Issues related to the memory usage
- Issues related to the I/O usage
- Resource intensive PL/SQL and Java
- Issues related to the RAC
- Issues related to different Application
- Issues related to the database configuration
- Issues related to object contention

Automatic SQL Tuning

Oracle provides an interface that performs within Automatic SQL Tuning called the SQL access advisor. This feature analyses all the possible ways of running the SQL statement and presents the user with the most efficient means of executing the SQL queries. It also identifies and manages the materialized views per workload. The input to the advisor may be acquired from the ADDM, cursor cache or SQL tuning sets.

Automatic Workload Repository

The Automatic Workload Repository is an evolution of the stats package available in the earlier versions. This package provides performance-monitoring tools and also acts as a source of information to the other packages such as the ADDM and the SQL tuning advisor. It maintains important information such as the active session history, time model statistics and various system and session statistics.

Automatic Storage Management (ASM)

The ASM provides the users with a single storage management interface across all the platforms. It provides most optimal I/O balancing. ASM also performs parallel load balancing to enhance efficiency. It is also capable of automatically detecting any newly added storage devices. It maintains multiple copies of data to provide the user with redundancy. An ASM instance can be created using the DBCA. The most important component of the ASM is the physical disk groups. These disk groups contain ASM files. ASM provides several templates that can be used to define the redundancy.

Oracle Data Pump

Data Pump is a replacement for the EXP and the IMP packages. It facilitates the loading and unloading of data and meta data. This package enables the logical backing up of the database, schemas, and individual components such as tables. The interface is called the DBMS_DATAPUMP. It is server based and thus results in high-speed data movement.

Scheduler

The Scheduler not only is able to run regular PL/SQL code blocks within the database, but it can also run any OS executables. This package is versatile and easy to use. The Scheduler can run jobs even when the database is not running. The Scheduler replaces the DBMS_JOB package.

Database Monitoring

Contents

- General Monitoring Scripts
- Monitoring Memory Usage
- Monitoring Disk I/O
- Monitoring System Resources
- Monitoring for Database Security
- Monitoring Database Schema Objects
- SQL Monitoring
- Useful Unix Commands

CHAPTER 10: DATABASE MONITORING

10.1 General Monitoring Scripts

To view Oracle version information:

```
SQL>SELECT banner FROM v$version;
```

To locate all the databases:

```
SQL>SELECT name, created, log_mode, open_mode
FROM v$database;
```

To determine Instance ID (SID):

```
SQL>SELECT enabled, Open_time, status, instance
FROM v$thread;
```

To determine Global Database Name:

```
SQL>SELECT name,value$
FROM sys.props$
WHERE name = 'GLOBAL_DB_NAME';
Or
SQL>SELECT * FROM global_name;
```

10.2 Monitoring Memory Usage

To View SGA Information:

```
SQL> SELECT name, pool, bytes
FROM v$sgastat
WHERE
name IN ('free memory', 'parameters',
'memory in use', 'db_block_buffers',
'log_buffer', 'dictionary cache',
'sql area', 'library cache')
/
```

To view Sort Area Information:

```
SELECT
name, value, class, statistic#
"Stats#"
FROM
v$sysstat
WHERE
name LIKE 'sorts%'
/
```

To determine status of the library cache:

```
PROMPT 'Library Cache (Shared SQLAREA) tune
shared_pool_size'
PROMPT 'Ratio should be below 1% (1.00)'
SELECT SUM(pins), SUM(reloads),SUM(gets),
SUM(reloads)/SUM(pins)*100
FROM v$librarycache
/
```

To determine status of the Data Dictionary Cache:

```
PROMPT 'Data Dictionary Cache - tune
shared_pool_size'
PROMPT 'Ratio should be less than 10 to 15%
(10.00) '
SELECT
SUM(gets), SUM(getmisses),SUM(scans),
SUM(scanmisses),SUM(scancompletes),
SUM(getmisses)/SUM(gets)*100
FROM
v$rowcache
/
```

10.3 Monitoring Disk I/O

To determine status of the Buffer Cache Hit Ratio:

```
PROMPT 'Buffer Cache Hit Ratio '
SELECT
SUM(decode(name, 'consistent gets',value, 0))
"Consis Gets",n SUM(decode(name, 'db block
gets',value, 0)) "DB Blk Gets",
SUM(DECODE(name, 'physical reads',value, 0))
"Phys Reads",
(SUM(DECODE(name, 'consistent gets',value, 0))
+ SUM(DECODE(name, 'db block gets',value, 0))
- SUM(DECODE(name, 'physical reads',value, 0)))
/ (SUM(DECODE(name, 'consistent gets',value, 0))
+ SUM(DECODE(name, 'db block gets',value, 0))
* 100
"Hit Ratio"
FROM v$sysstat WHERE
name IN('db block gets','consistentgets',
'physical reads')
/
```

To identify users that have a certain hit ratio:

```

PROMPT The following list of users has a hit
ratio less than 80%.
COL user_session format a25
SELECT se.username||'('|| se.sid||')'
user_session,
      SUM(DECODE(name, 'consistent
gets',value, 0)) "Consis Gets",
      SUM(DECODE(name, 'db block
gets',value, 0)) "DB Blk Gets",
      SUM(DECODE(name, 'physical
reads',value, 0)) "Phys Reads",
      (SUM(DECODE(name, 'consistent
gets',value, 0)) +
      SUM(DECODE(name, 'db block
gets',value, 0)) -
      SUM(DECODE(name, 'physical
reads',value, 0)))
      DECODE( (SUM(decode(name, 'consistent
gets',value, 0)) +
      SUM(DECODE(name, 'db block
gets',value, 0))) ,0,1,
      (SUM(DECODE(name, 'consistent
gets',value, 0)) +
      SUM(DECODE(name, 'db block
gets',value, 0)) ) ) * 100 "Hit Ratio"
FROM v$sesstat ss, v$statname sn, v$session se
WHERE   ss.sid = se.sid
        AND sn.statistic# = ss.statistic#
        AND value != 0
        AND sn.name in ('db block gets',
        'consistent gets', 'physical reads')
GROUP BY se.username, se.sid HAVING
      (SUM(DECODE(name, 'consistent
gets',value, 0)) +
      SUM(DECODE(name, 'db block
gets',value, 0)) -
      SUM(DECODE(name, 'physical
reads',value, 0)))
      DECODE( (SUM(DECODE(name, 'consistent
gets',value, 0)) +
      SUM(DECODE(name, 'db block
gets',value, 0))) ,0,1,
      (SUM(DECODE(name, 'consistent
gets',value, 0)) +
      SUM(DECODE(name, 'db block
gets',value, 0)) ) ) * 100 < 80

```

10.4 Monitoring System Resources

To list wait sessions:

```
SELECT event, count(*)
FROM v$session_wait
GROUP BY event/
```

To display the length of the write request queue:

```
PROMPT ' 'column "Write Request Length"
FORMAT 999,999.99
SELECT
      SUM( DECODE (name, 'summed dirty queue
length', value))/SUM( DECODE (name,
      'write requests', value))
FROM v$sysstat
WHERE name IN('summed dirty queue length' ,
'write requests')AND value > 0
/
```

To determine list of latches held:

```
SELECT lat.name , pro.username
FROM v$process pro, v$latchholder lh,
      v$latch lat
WHERE lh.pid = pro.pid and lh.laddr =
      lat.addr
/
```

To determine status of the latches:

```
SELECT name, gets, misses,sleeps,
      immediate_gets , immediate_misses,
      waiters_woken, waits_holding_latch,
      spin_gets
FROM v$latch
WHERE name LIKE 'redo%'
/
```

To determine status of the redo allocation latch:

```
SELECT (misses/DECODE(gets, 0, 1, gets))*100
      Ratio1, (immediate_misses/
      DECODE (immediate_misses+
      immediate_gets, 0, 1,
      immediate_misses + immediate_gets
      )*100) obratio
FROM v$latch lat
WHERE lat.name = 'redo allocation'
/
```


To determine status of the redo copy latch:

```
PROMPT Redo copy latch
PROMPT For contention of redo latch, increase
the value
PROMPT of log_simultaneous_copies in init.ora.
PROMPT all ratios should be <= 1%
SELECT
(misses/decode(gets,0,1,gets))*100 Ratio1,
(immediate_misses/decode(immediate_misses+
immediate_gets,0,1,
immediate_misses + immediate_gets )*100)

obratio
FROM v$latch lat
WHERE lat.name = 'redo copy'
```

10.5 Monitoring for Database Security**To list all users:**

To list the users currently connected to the system and the command they are currently running and the number of seconds since the 'last call' was issued.

```
SELECT sess.osuser osu, pro.username prou,
       sess.username sessu, sess.status
       status, sess.sid sid, sess.serial# ser,
       lpad(pro.spid,7) pid, sess.last_call_et
Call
FROM v$process pro, v$session sess,
       v$sqlarea sqla
WHERE
pro.addr=sess.paddr
AND sess.username IS NOT NULL AND
sess.sql_address=sqla.address(+) AND
sess.sql_hash_value=sqla.hash_value(+)
ORDER BY osu, call
/
```

To view user information:

```
SELECT username, osuser, status, schemaname,
       machine, terminal, state,
       TO_CHAR(logon_time, 'DD/MM/YYYY
       HH24:MI')
FROM v$session
/
```

To view detailed user information:

```

SELECT SUBSTR (sess.username, 1,12) user,
       sess.sid||','||sess.serial# sersid,
       sess.terminal,
       sess.status,
       sort.tablespace
       DECODE (dfile.file_name,
              NULL,tdfile.file_name)
       sort.contents
       sort.extents
       sort.blocks * par.value/1024 kb
FROM v$parameter par, v$session sess,
     dba_data_files dfile, dba_temp_files
     tdfile, v$sort_usage sort
WHERE sess.saddr = sort.session_addr AND
      (sort.segfile# = dfile.file_id OR
       sort.segfile# = tdfile.file_id) AND
      sort.contents = 'TEMPORARY' AND
      par.name = 'db_block_size'
ORDER BY sess.sid, tdfile.file_name
/

```

10.6 Monitoring Database Schema Objects**To determine status of the tablespaces:**

```

SELECT freesp.tablespace_name,
       sum(freesp.tots) Tot_Size,
       sum(freesp.sumbytes) Total_free,
       sum(freesp.sumbytes)*100/sum(
       freesp.tots) Percent_Free,
       sum(freesp.largest) Maximum_Free,
       sum(freesp.chunks) Free_Chunks
FROM
  (SELECT tablespace_name,0 tots,
         sum(bytes) sumbytes, max(bytes)
         largest,count(* ) chunks
  FROM dba_free_space freesp
  GROUP BY tablespace_name
  UNION
  SELECT tablespace_name,
         sum(bytes)sumbytes, 0, 0, 0
  FROM dba_data_files
  GROUP BY tablespace_name
  )freesp
GROUP BY freesp.tablespace_name;
SELECT name,type, value val
FROM v$parameter
WHERE name='db_name';
/

```

	<pre> SELECT FREESP.tablespace_name, FREESP.Maxbytes/1024 MaxIndex, NEXTSP.owner, NEXTSP.segment_type, NEXTSP.segment_name, next_extent/1024 NextIndex, FROM (SELECT owner, segment_type, segment_name, tablespace_name, next_extent FROM sys.DBA_SEGMENTS) NEXTSP, (SELECT tablespace_name,max(bytes) Maxbytes FROM sys.DBA_FREE_SPACE GROUP BY tablespace_name) FREESP WHERE FREESP.tablespace_name = NEXTSP.tablespace_name AND NEXTSP.next_extent * 2 >= FREESP.Maxbytes ORDER BY owner, segment_type, tablespace_name, segment_name; </pre>
--	--

To view rollback segment information:

	<pre> SELECT wait.class, (SUM(wait.count)/sum(stat.value)) * 100 FROM v\$waitstat wait , v\$sysstat stat WHERE wait.class in ('undo header', 'undo block', 'system undo header', 'system undo block') AND stat.name IN ('consistent gets', 'db block gets') GROUP BY wait.class / </pre>
--	---

To view rollback segment shrinkage information:

	<pre> SELECT extends, extents,shrinks,hwm,size, gets, name,waits FROM v\$rollstat, v\$rollname WHERE v\$rollstat.usn = v\$rollname.usn </pre>
--	--

To determine status of the redo log space requests:

	<pre> SELECT name, value, class, statistic# FROM v\$sysstat WHERE name = 'redo log space requests' </pre>
--	---

To display all roll back segments:

```

SELECT segment_name, owner,
       tablespace_name, segment_id,
       file_id, block_id, initial_extent/1024,
       next_extent/1024, min_extents,
       max_extents
FROM dba_rollback_segs;

```

10.7 SQL Monitoring**To determine the SQL information for a given SID:**

```

SELECT sql_text
       FROM v$sqltext
       WHERE address =
(SELECT sql_address FROM
 v$sqlsession where sid=102)
AND hash_value = (SELECT sql_hash_value
 FROM v$sqlsession where sid=102)
ORDER BY piece;

SELECT dbi.table_owner||'.'||dbi.table_name,
       dbi.owner||'.'||dbi.index_name ind_name,
       dbi.distinct_keys, dbi.uniqueness,
       dicol.column_position||' '||
       dicol.column_name col_name
FROM dba_ind_columns dicol, dba_indexes dbi
WHERE DECODE('&&owner',
             NULL, 'x', dbi.table_owner) =
       NVL(upper('&&owner'), 'x')
AND DECODE('&&table_name',
           NULL, 'x', dbi.table_name)
LIKE NVL(upper('&&table_name'), 'x')
AND dbi.index_name = dicol.index_name
AND dbi.table_name = dicol.table_name
AND dbi.owner = dicol.index_owner;

```

To show active transaction in progress:

```

SELECT sid, serial#, sess.status, username,
       terminal, osuser, trans.start_time,
       roll.name,
       (trans.used_ublk*'&&x')/1024,
       trans.used_ublk ,
       DECODE(trans.space, 'YES', 'SPACE TX',
       DECODE(trans.recursive, 'YES',
       RECURSIVE TX',
       DECODE(trans.noundo, 'YES', 'NO UNDO
       TX', trans.status)
       )) status
FROM sys.v_$transaction trans,
       sys.v_$rollname roll, sys.v_$session
WHERE trans.xidusn = roll.usn
       AND trans.ses_addr = sess.saddr;

```

10.8 Useful Unix Commands

10.8.1 General Commands

Command	Description	Sample usage
<i>pwd</i>	Displays current directory	root> pwd
<i>ls</i>	Lists all the files in the current directory	root> ls
<i>ls -al</i>	Lists all hidden files with details	root> ls -al
<i>cd</i>	Changes directory	root> cd /u01/app/oracle
<i>touch</i>	Creates new empty file	root> touch slx_01.log
<i>rm</i>	Deletes files	root> rm slx_01.log
<i>mv</i>	Moves / renames files	root> mv [from] [to]
<i>cp</i>	Copies files/ directories	root> cp [from] [to]
<i>mkdir</i>	Creates new directory	root> mkdir slx_usr01
<i>find</i>	Locates specific files	root> find / -n slx_sql01.sql root>find / -print grep -i slx_sql01.sql
<i>which</i>	Finds PATH settings of the executable	root> rmdir slx_usr01

10.8.2 User Management

Command	Description	Sample usage
<i>umask</i>	To set default File Permissions for current User	root> umask 022
<i>chmod</i>	To change the File Permissions after the creation of the file	root> chmod 777 .log root> chmod o+rwx *.log root> chmod g+r *.log root> chmod -Rx *.log
<i>chown</i>	To reset the file ownership	root> chown -R oinstall.dba *
<i>useradd</i>	To add user	root> useradd -G [primary group] -g [secondary group] -d [default directory] -m [creates the default directory] -s [the default shell]
<i>usermod</i>	To modify user setting	root> usermod -s /bin /csh slx_usr01

<i>userdel</i>	To drop existing user	root> userdel -r slx_usr01
<i>passwd</i>	To set the passwrod	root> passwd m slx_usr01
<i>who</i>	To list current Oracle users	root> who grep -i ora

10.8.3 Performance Characteristics

Command	Description	Sample usage
<i>ps</i>	To list current Oracle processes	root> ps -ef grep -i ora
<i>kill</i>	To kill process based on process id	root> kill -9 12345
<i>uname</i>	To get the user information	root> uname -a
<i>hostname</i>	To get host information	root> hostname
<i>cat alert_lin1.log</i>	To return Oracle error lines	root> cat alert_LIN1.log grep -i ORA-
<i>rm</i>	To remove back up logs	find /backup/logs/ -name daily_backup* -mtime +21 -exec rm -f {} \;
<i>gzip</i>	To zip files	gzip myfile
<i>gunzip</i>	To uncompress	gunzip myfile.gz
<i>compress</i>	To Compress Files	compress myfile
<i>uncompress</i>	To uncompress	uncompress myfile

10.8.4 CPU Performance

vmstat

To display system Statistics(for 5 seconds apart for 2 times):

```
root>$ vmstat 5 2
```

procs			memory		page						disk			faults				cpu	
r	b	w	Swap	Free	re	mf	Pi	po	fr	De	sr	s0	s1	in	sy	cs	us	sy	id
0	0	0	2217200	30804	2	55	178	139	222	1124	251	14	3	257	449	261	2	3	88
0	0	0	2217200	29808	8	2	143	5	319	0	19	2	17	263	251	198	12	14	91

procs.

This parameter gives information about the number of processes.

R: rocesses in the run queue.

B: processes blocked for resources.

W: Processes those are swapped.

Memory

Reports the usage of the memory.

- Swap: currently available swap space.
- Free: size of free space.

Page

Reports the paging activities.

- Re: pages that were reclaimed.
- Mf: minor faults.
- Pi: KB paged in.
- Po: KB paged out.
- Fr: free space.
- De: short term memory short fall.
- Sr: pages scanned.

Disk

Number of disk operations for the disk s0 and s1.

Faults.

The number of traps and interrupts.

- Si: system calls.
- Cs : context switches.

CPU

The use of the CPU time.

- Us: user time.
- Si: system time.
- Cs: idle time

```
root>sar -u 5 2
```

	CPU	%user	%nice	%system	%idle
02:50:51 PM	all	26.53	0.00	2.95	70.53
02:50:56 PM	all	26.13	0.00	2.32	71.55
Average:	all	26.33	0.00	2.63	71.05

user: CPU utilization that occurred during the execution at the user level.

nice: CPU utilization that occurred during the execution at the user level with nice priority.

system: CPU utilization that occurred during the execution at the system level.

idle: Time that the CPU was idle

mpstat

Reports preprocessor statistics 5 Seconds apart for 2 times.

```
mpstat 5 2
```

	CPU	%user	%nice	%system	%idle	intr/s
02:51:56 PM	all	27.57	0.00	2.56	69.87	246.52
02:52:06 PM	all	26.32	0.05	2.59	71.04	233.27
Average:	all	26.94	0.03	2.57	70.46	239.86

ps

To display the top 10 CPU users.

```
02:51:56 PM CPU %user %nice %system %idle intr/s
02:52:01 PM all 27.57 0.00 2.56 69.87 246.52
02:52:06 PM all 26.32 0.05 2.59 71.04 233.27
Average:    all 26.94 0.03 2.57 70.46 239.86
```

```
$ ps -e -o pcpu -o pid -o user -o args | sort -k 1 | tail -21r
```

```
%CPU PID User Command
```

```
0.4 9510 oraprod ora_qmn0_PROD
```

```
0.6 17207 oracle /mnt/sdc1/ora10gAS/opmn/bin/opmn -d
```

```
0.6 17644 oracle /mnt/sdc1/ora10gAS/bin/emagent
```

```
1.4 20523 oracle /mnt/sdc1/ora10gAS/Apache/.. /Apache/Apache -U
185008166
```

Cron

To run jobs at regular intervals:

- *Login as root*
- *\$ crontab -l > newjob*
- *Edit newjob file.*
- *\$ crontab new job*

10.8.5 Useful File Locations

Path	Contents
/etc/passwd	User settings
/etc/group	Group settings for users
/etc/hosts	Hostname lookup information
/etc/system	Kernel parameters

Archiving and Its Value to DBA

Contents

- Introduction
- What is driving the data growth?
- Solution
- Benefits

CHAPTER 11: ARCHIVING AND ITS VALUE TO DBA

11.1 Introduction

Data growth is one of the biggest challenges faced by IT organizations, today. As transaction data grows, applications slow down, storage infrastructure costs increase, back up and recovery takes ages, system stability slips, upgrades take time and ultimately users, DBAs and developers get frustrated. Unmanaged data growth is leading to disruption in DBA tasks and activities and ultimately a decline in DBA performance. Database administrators are constantly monitoring data growth and fine-tuning to maintain performance rather than investing time in strategic planning and management.

11.2 The Problem

What is driving the data growth?

- High volume online transaction processing
 - o ERP/ CRM or SCM application environments grow larger and more complex as they retain more historical business transactions for longer periods
- Compliance to regulatory requirements
 - o With regulatory mandates like Sarbanes-Oxley Act becoming a focus, organizations are compelled to retain historic data in an accessible and intelligent form for periods as long as 30 years
 - o Other record retention requirements include HIPPA, 21 CFR 11, IRS and SEC Rule 17a-4
- Need for Data Multiplication
 - o Multiple copies of the production environment needed for development, testing, staging and training purposes
 - o Disaster recovery and business continuity
 - o Routine back up and recovery
- Application Upgrades
 - o Upgrades can lead to increase in significant amount of data due to enhanced data models

11.3 Solution

Solix Technologies develops a variety of archiving and data management tools enabling DBAs to keep their critical databases clean, stable and operating at peak performance. Solix ARCHIVEjinni™, a state-of-the-art data management solution automatically detects, analyzes, monitors, subsets and migrates inactive data to cost effective storage tiers. Migration and management of data from creation to deletion in accordance with defined data retention policies and business rules provides visibility and control.

ARCHIVEjinni™ ensures referential integrity of data even after it has been purged from the production system. Overall there is no risk for a DBA, since the De-Archive feature within the product restores archive data back to live data whenever needed.

Administrators need not be reluctant to remove old data from the production database fearing critical records may be permanently deleted or referential integrity of the data may be comprised. . With ARCHIVEjinni™ DBAs are freed from having to synchronize the production and archived databases in case of any new upgrade or patch implementation. ARCHIVEjinni™ automatically detects disparities and performs an automatic update in the archive instance for any change in the production schema.

Timely access to information when needed can make a significant difference for enterprises challenged with document control and compliance regulations. ARCHIVEjinni™ facilitates this need by storing inactive data into an online archive database and provides an interoperable access layer that allows for transparent access to both live and archive data in a single logged in session.

ARCHIVEjinni's Configurator tool provides DBAs with a unique capability to design custom archive and purge rules (different from the standard purge rules) with a simple drag and drop function. The Configurator validates new archive rules based on a pre-seeded knowledge base (parent-child relationships). DBAs utilizing ARCHIVEjinni™ further benefit by creating smaller subset databases for testing, development, patching or training purposes. This feature helps reduce infrastructure costs and application performance concerns of a database administrator.

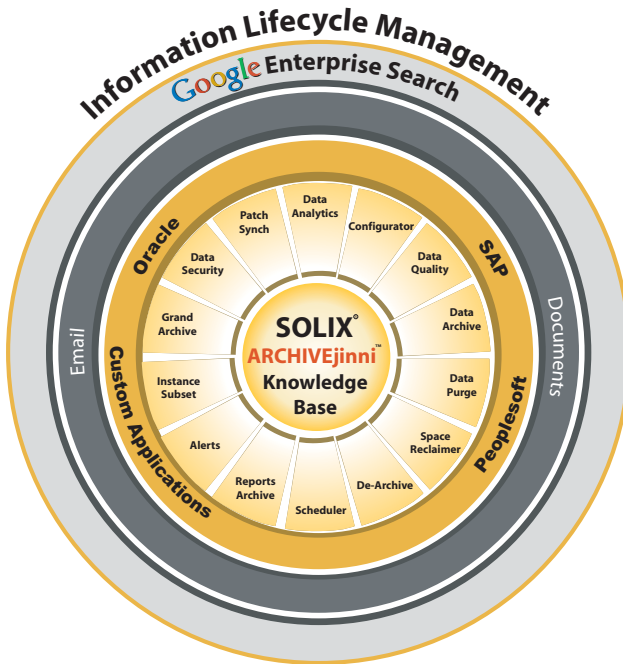
Simply stated, ARCHIVEjinni™ is the most comprehensive solution enabling DBAs to resolve application performance and storage cost issues arising from data growth.

ARCHIVEjinni™ Features

- 1) **Data Analytics** tool provides rich analytics and reporting on static data growth in different application modules. Initial assessment will provide information on installed modules, data per module, static data growth, archive eligible data and parameters.
- 2) **Configurator** allows the user to design new configurations for archiving and purging data, which are different from standard purge routines created by the application. This feature offers the flexibility to design new configurations and generate a dynamic code with a simple drag and drop capability.
- 3) **Data Quality** helps in correcting, standardizing and validating data..
- 4) **Data Archive** allows for archiving of data both labeled by the enterprise application such as Oracle purge routines and those requisitioned by the customer.

- 5) **Data Purge** archived data can be purged from the production instance with the data purge feature.
- 6) **Space Reclaimer** reclaims the space made available as a result of the archive and purge function.
- 7) **De-Archive** restores transaction or batch transactions from the archive instance to the production instance.
- 8) **Scheduler** offers scheduled archive and purge capability.
- 9) **Reports Archive** provides multiple users real time access and sharing of reports generated and archived.
- 10) **Alerts** provide intelligence on data growth, scheduled reporting and other associated notifications linked to the regulatory policies.
- 11) **Dash Board** provides a view of all database instances. Monitors and analyzes application performance in both graphical and tabular form. Supports aggregation, drill-downs and alerts.
- 12) **Data Auditor** Archives workflow data for audit trail.
- 13) **Data Security** feature encrypts sensitive information in both production and non-production database for protection against inappropriate visibility.
- 14) **Patch Sync** detects disparities between production and archive data structures and performs an appropriate automatic update in the archive instance for every new patch in the production schema.
- 15) **Instance Subset** allows users to create smaller subsets of a production instance for test, development, patching, training and outsourcing purposes. This contributes to reduced infrastructure and maintenance costs.

ARCHIVEjinni feature overview



11.4 Benefits

- Increased application performance and availability
- Improved regulatory compliance (SOX, HIPPA) and data protection
- Reduced storage, maintenance and operating costs
- Optimized backup, recovery, cloning and upgrades
- Intelligent reporting capability

ARCHIVING STRATEGIES

ARCHIVEjinni™ offers different cost effective strategies to store and access archived data. Three alternative methodologies have been discussed below:

Methodology 1: Production and Archived Data in the Production Database

In this approach, both archive and production data is stored in a different schema within the same production database instance. The advantage of this approach is that both the archive and production database can be simultaneously accessed through an interoperable layer. Since both the archive and production data is stored in the same instance, improvement in application performance is insignificant.

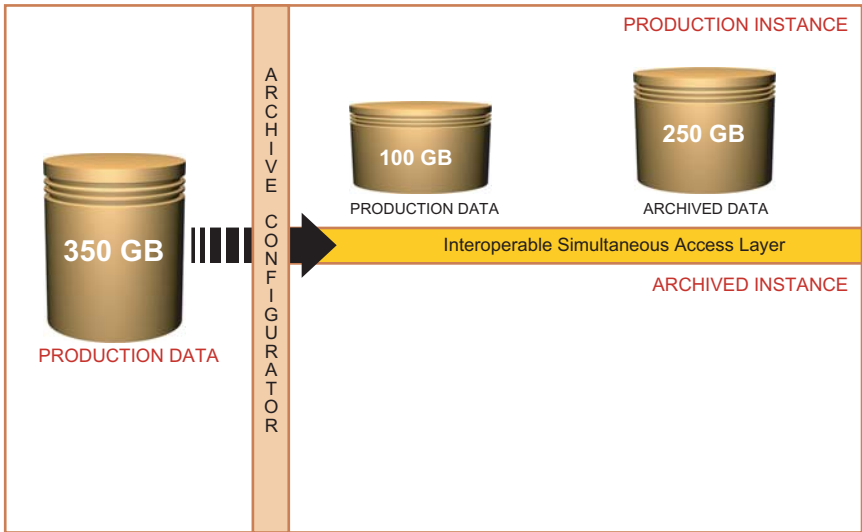


Figure 14. SOLIX ARCHIVEjinni

Methodology 2: Production and Archived Data in their Respective Instances

In this approach production data is stored in the production instance and archive data is stored in the archived instance. This approach helps improve application performance, but limits the application accessibility, as a database link has to be set up to enable simultaneous access of data.

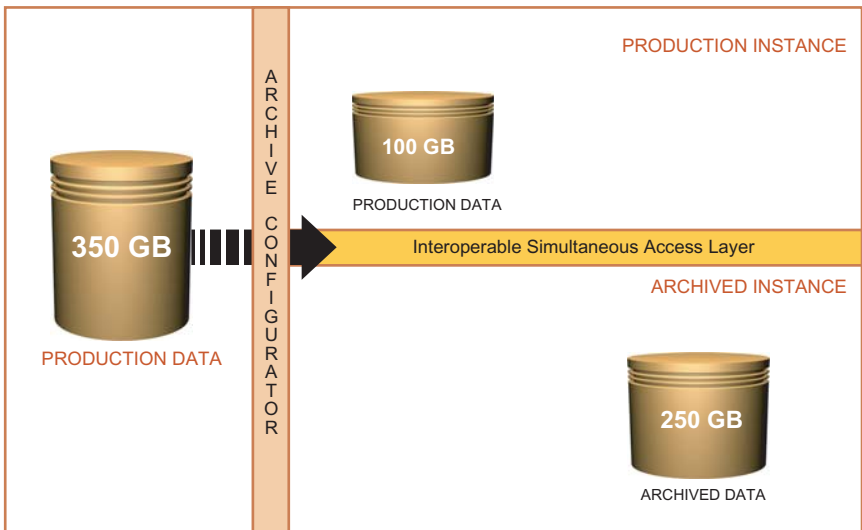


Figure 15. SOLIX ARCHIVEjinni

Methodology 3: Production Data in the Production Instance and Production Copy and Archived Data in Archived Instance

This approach overcomes the challenges faced in the above two methodologies. Customers experience immense improvement in application performance as both the production and the archived data are stored in two distinct instances. Moreover, since in the archived instance both the archived data and the production data copy are made available, the interoperable layer can simultaneously access both the archived and production data. The flip side of this alternative is that every time there is a change in the production data it needs to be replicated in the production schema within the archived instance.

Once the users do not need access to the historical archived data, the inactive data in the archived instance is moved to backup media for GrandArchive (long term archiving).

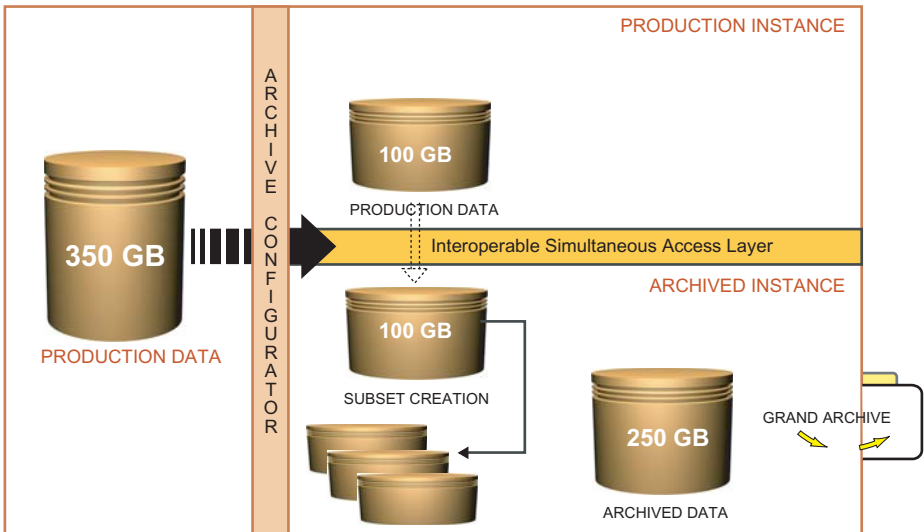
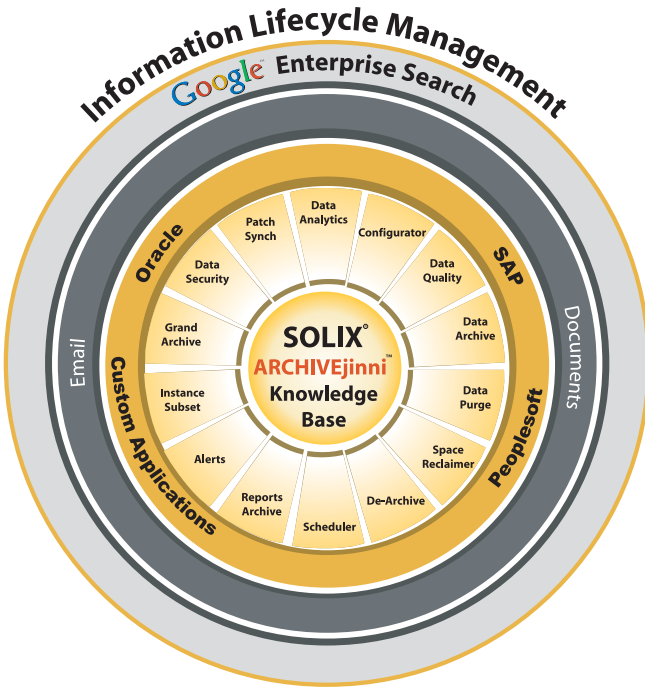


Figure 16. SOLIX ARCHIVEjinni

DBA Handbook for Oracle

Abstract

Solix Technologies, Inc. is a leading provider of solutions for enterprise database and application management. Solix products are best known for simplifying and improving data administration. Solix solutions are designed to automate data archiving, maintaining the online availability of data while optimizing application performance and storage utilization.



Copyrighted Material

Solix Technologies, Solix product names and the Solix logo are trademarks or registered trademarks of Solix Technologies, Inc. All other product names mentioned herein are the trademarks and property of their respective owners.

© 2005 Solix Technologies Inc. All rights reserved.

