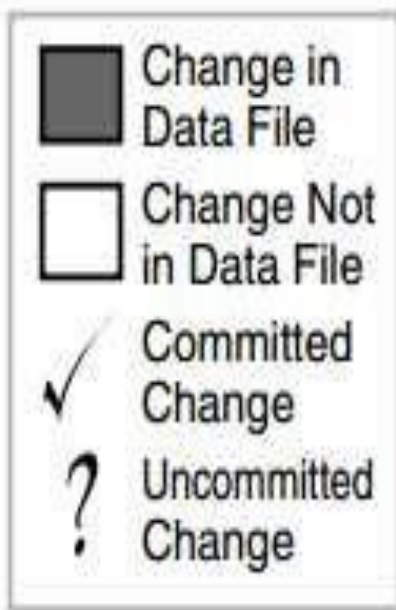




Introduction to Database Administration

TOPIC 160

Overview of Parameter Files with commands
Overview of Parameter Files with commands

Overview of Parameter Files with commands

Introduction
Initialization Parameters
Functional Group of Initialization Parameters
Purpose of Variable Parameters
Basic Initialization Parameters
Advance Initialization Parameters
Parameters during Installation
Parameters Files spfile and pfile
Overview of Parameter Files with commands

Introduction

To start a database instance, Oracle Database must read either a server parameter file, which is recommended, or a text initialization parameter file, which is a legacy implementation. These files contain a list of configuration parameters.

Overview of Parameter Files with commands

Initialization Parameters

Initialization parameters are configuration parameters that affect the basic operation of an instance. The instance reads initialization parameters from a file at startup.

Oracle Database provides many initialization parameters to optimize its operation in diverse environments. Only a few of these parameters must be explicitly set because the default values are adequate in most cases.

Overview of Parameter Files with commands

Functional Group of Initialization Parameters

Most initialization parameters belong to one of the following functional groups:

Parameters that name entities such as files or directories

Parameters that set limits for a process, database resource, or the database itself

Parameters that affect capacity, such as the size of the SGA (these parameters are called variable parameters)

Overview of Parameter Files with commands

Purpose of Variable Parameters

Variable parameters are of particular interest to database administrators because they can use these parameters to improve database performance.

Overview of Parameter Files with commands

Basic Initialization Parameters

In most cases, you must set and tune only the approximately 30 basic parameters to obtain reasonable performance. The basic parameters set characteristics such as the database name, locations of the control files, database block size, and undo tablespace

Overview of Parameter Files with commands

Advance Initialization Parameters

In rare situations, modification to the advanced parameters may be required for optimal performance.

The advanced parameters enable expert DBAs to adapt the behavior of the Oracle Database to meet unique requirements

Overview of Parameter Files with commands

Parameters during Installation

Oracle Database provides values in the starter initialization parameter file provided with your database software, or as created for you by the Database Configuration Assistant.

```
db_name=sample
control_files=/disk1/oradata/sample_cf.dbf
db_block_size=8192
open_cursors=52
undo_management=auto
shared_pool_size=280M
pga_aggregate_target=29M
.
```

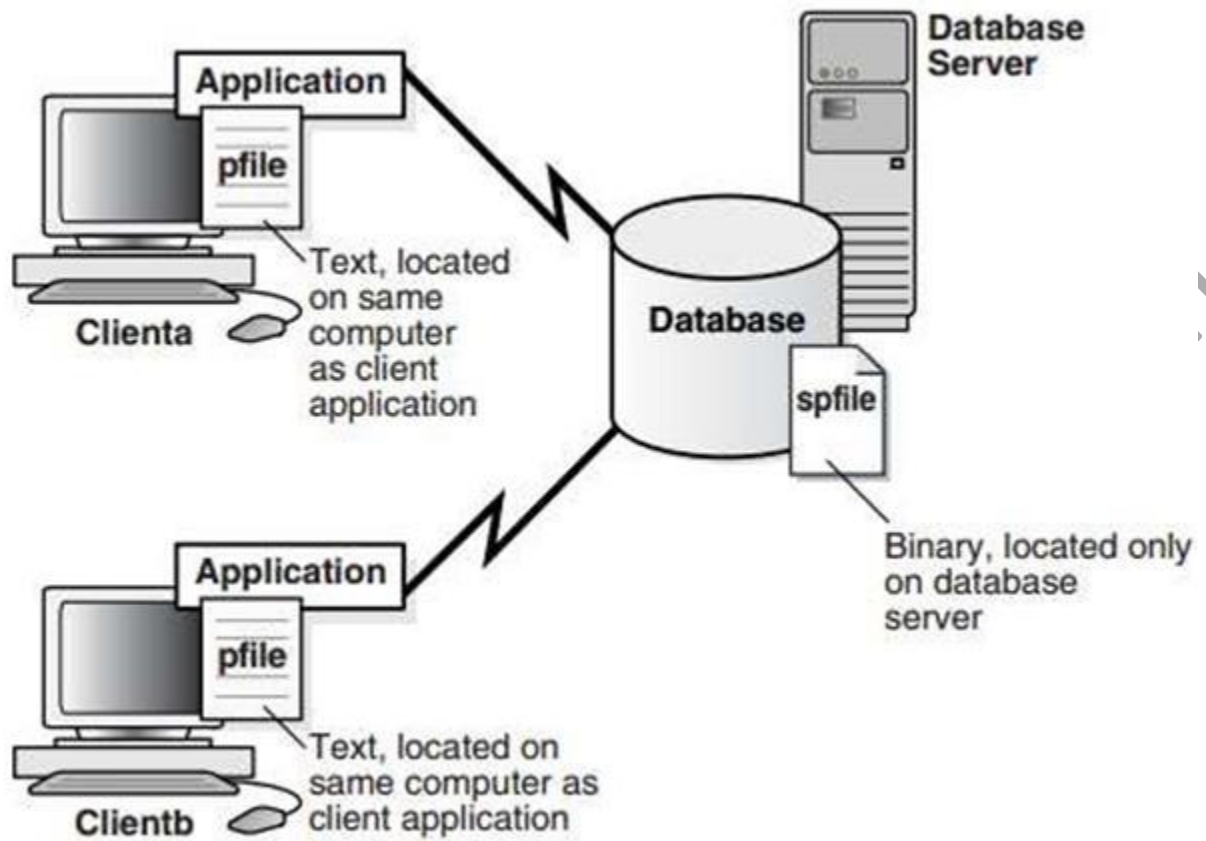
Overview of Parameter Files with commands

Parameters during Installation

create spfile from pfile; create pfile from spfile; show parameter processes; show parameter spfile;

Overview of Parameter Files with commands

Parameters Files spfile and pfile



Introduction to Database Administration

TOPIC 161

Initializing Parameter Files
Initializing Parameter Files

Initializing Parameter Files

Server's Parameters Characteristics

Text Parameters File

Text Parameters Characteristics

Use of **alter system** command

Static and Dynamic Parameters

Initializing Parameter Files

Server's Parameters Characteristics

Only one server parameter file (spfile) exists for a database. This file must reside on the database host.

The server parameter file is written to and read by only by Oracle Database, not by client applications.

The server parameter file is binary and cannot be modified by a text editor.

Initializing Parameter Files

Server's Parameters Characteristics

Initialization parameters stored in the server parameter file are persistent. Any changes made to the parameters while a database instance is running can persist across instance shutdown and startup.

Initializing Parameter Files

Text Parameters File

A text initialization parameter file (**pfile**) is a text file that contains a list of initialization parameters. It is readable and writable to DBA and its changes are reflected into **spfile** on Server.

The text initialization parameter file contains a series of key=value pairs, one per line

Initializing Parameter Files

Text Parameters Characteristics

When starting up or shutting down a database, the text initialization parameter file must reside on the same host as the client application that connects to the database.

A text initialization parameter file is text-based, not binary.

Oracle Database can read but not write to the text initialization parameter file. To change the parameter values you must manually alter the file with a text editor.

Initializing Parameter Files

Text Parameters Characteristics

Changes to initialization parameter values by ALTER SYSTEM are only in effect for the current instance. You must manually update the text initialization parameter file and restart the instance for the changes to be known.

Initializing Parameter Files

```
global_txn_processes      integer      1
job_queue_processes       integer      1000
log_archive_max_processes integer      4
processes                 integer      150
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup open
ORACLE instance started.

Total System Global Area 1071333376 bytes
Fixed Size                 1375792 bytes
Variable Size             696254928 bytes
Database Buffers          369098752 bytes
Redo Buffers               4603904 bytes
Database mounted.
Database opened.
SQL> show parameter processes;

NAME                                TYPE        VALUE
-----
aq_tm_processes                     integer      0
db_writer_processes                 integer      1
gcs_server_processes                integer      0
global_txn_processes                integer      1
job_queue_processes                 integer      1000
log_archive_max_processes            integer      4
processes                           integer      200
SQL>
```

Use of Alter System command

```
Alter system set processes=200
scope=spfile;
```

```
create pfile='E:\app\Haider\product\11.2.0\dbhome_1\database\pfileORCL.ora' from SPFILE;
```

This change can be seen by opening spfile
Initializing Parameter Files

Characteristic	Static	Dynamic
Requires modification of the parameter file (text or server)	Yes	No
Requires database instance restart before setting takes affect	Yes	No
Described as "Modifiable" in <i>Oracle Database Reference</i> initialization parameter entry	No	Yes
Modifiable only for the database or instance	Yes	No

Static and Dynamic Parameters

Static parameters include DB_BLOCK_SIZE, DB_NAME, and COMPATIBLE. Dynamic parameters are grouped into session-level parameters, which affect only the current user session, and system-level parameters, which affect the database and all sessions.

MEMORY_TARGET=system-level & NLS_DATE_FORMAT=session level parameters

Introduction to Database Administration

TOPIC 162

Diagnostic Instance Files

Diagnostic Instance Files

Diagnostic Instance Files

Overview of Diagnostic Files

Goals of fault Diagnosability Infrastructure

Automatic Diagnostic Repository

Problems & Incidence

DDL & DEBUG Log files - Example

ADR Directory Structure

Diagnostic Instance Files

Overview of Diagnostic Files

Oracle Database includes a fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving database problems. Problems include critical errors such as code bugs, metadata corruption, and customer data corruption.

Diagnostic Instance Files

Goals of fault Diagnosability Infrastructure

Detecting problems proactively

Limiting damage and interruptions after a problem is detected

Reducing problem diagnostic and resolution time

Simplifying customer interaction with Oracle Support

Diagnostic Instance Files

Automatic Diagnostic Repository

Automatic Diagnostic Repository (**ADR**) is a file-based repository that stores database diagnostic data such as trace files, the alert log, and Health Monitor reports. Key characteristics of ADR include:

Unified directory structure

Consistent diagnostic data formats

Unified tool set

Diagnostic Instance Files

Problems & Incidence

ADR proactively tracks problems, which are critical errors in the database. Critical errors manifest as internal errors, such as ORA-600, or other severe errors. Each problem has a problem key, which is a text string that describes the problem.

Diagnostic Instance Files

Problems & Incidence

When a problem occurs multiple times, ADR creates a time-stamped incident for each occurrence. An incident is uniquely identified by a numeric incident ID. When an incident occurs, ADR sends an incident alert to Enterprise Manager. Diagnosis and resolution of a critical error usually starts with an incident alert.

ADR informs you that a critical error is ongoing without overloading the system with diagnostic data

Diagnostic Instance Files

DDL & DEBUG Log Files - Example

DDL Log: In the alert log file, DDL statements can be recorded when using parameter `ENABLE_DDL_LOGGING` to `TRUE` with `NOMOUNT` startup of database (in Oracle 11g version).

Following commands can be recorded with `TRUE` option:

Create Table Create Index Create User Drop Table

Diagnostic Instance Files

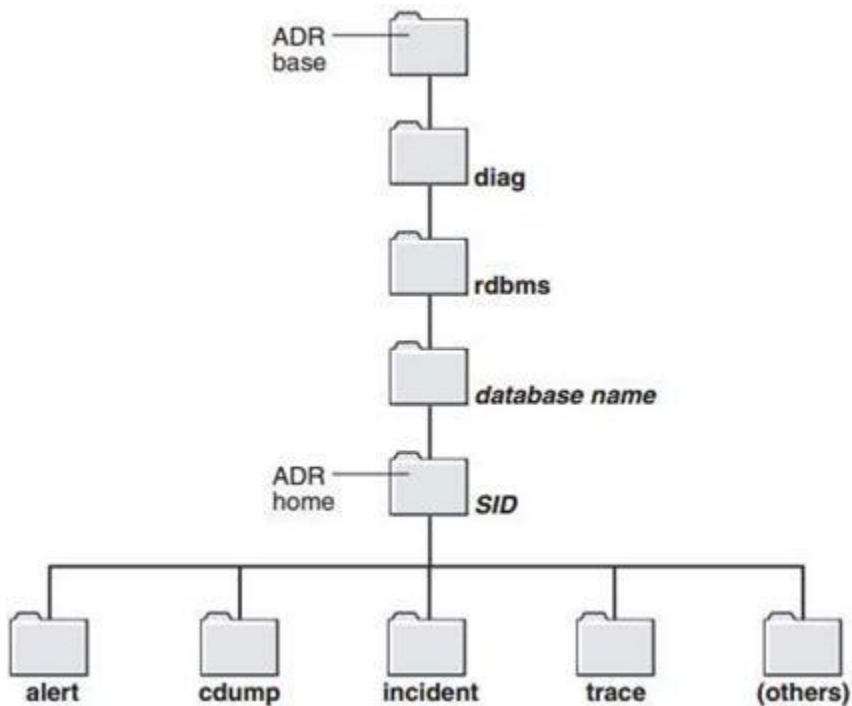
ADR Directory Structure

The ADR base is the ADR root directory. The ADR base can contain multiple ADR homes, where each ADR home is the root directory for all diagnostic data—traces, dumps, the alert log, and so on—for an instance of an Oracle product or component.

```
SQL> SELECT substr(NAME,1,30) Diag_name, substr(VALUE, 1, 50) diag_val
2  FROM V$DIAG_INFO;

DIAG_NAME                                DIAG_VAL
-----
Diag Enabled                             TRUE
ADR Base                                 e:\app\haider\diag\rdbms\orcl\orcl
ADR Home                                 e:\app\haider\diag\rdbms\orcl\orcl
Diag Trace                               e:\app\haider\diag\rdbms\orcl\orcl\trace
Diag Alert                               e:\app\haider\diag\rdbms\orcl\orcl\alert
Diag Incident                            e:\app\haider\diag\rdbms\orcl\orcl\incident
Diag Cdump                               e:\app\haider\diag\rdbms\orcl\orcl\cdump
Health Monitor                           e:\app\haider\diag\rdbms\orcl\orcl\hm
Default Trace File                       e:\app\haider\diag\rdbms\orcl\orcl\trace\orcl_ora_
Active Problem Count                     0
Active Incident Count                    0

11 rows selected.
```



Introduction to Database Administration

TOPIC 163

Commands & Log Alerts

Commands & Log Alerts

Commands & Log Alerts

Overview of Alert File

Alert Log File

Alert Log File – Contents

Alert Log File - Maintenance

Alert Log File - Location

Commands & Log Alerts

Overview of Alert File

Each database has an alert log, which is an XML and text file containing a chronological log of database messages and errors. The alert log contents include the following:

Deadlock occurs

Any error written by Oracle Server process

Any unexpected messages and corruptions happened

Any DBA DDL operations are managed
Commands & Log Alerts

Alert Log File

The Alert log file (ALERT.LOG) is a chronological log of messages and errors written out by an Oracle Database

This file should constantly be monitored to detect unexpected messages and corruptions
Commands & Log Alerts

Alert Log File - Contents

Non default initialization parameter
used at startup or
internal errors (Ora-600)
Block corruption error (Ora-1578)
DDL and STARTUP, SHUTDOWN,
ARCHIVE LOG, and RECOVER
Commands & Log Alerts

Alert Log File - Contents

The alert log file contains the following information that can be used to aid in tuning the database

Time to perform archiving
Instance recovery start and complete times
Deadlock and timeout errors
Incomplete checkpoints
Checkpoint start and end times
Commands & Log Alerts

Alert Log File - Maintenance

As Alert file grows bigger,
DBA can periodically backup the alert log file and delete the current file

When database attempts to write to the
alert file again, it will create a new file
Commands & Log Alerts

Alert Log File - Location

Each database has an Alert_SID.log file Resides on Server with the database Default location
Oracle- Base/diag/rdbms/DB_Name/SID/trace

End
Introduction to Database Administration

TOPIC 164

Trace Files Commands
Trace Files Commands

Trace Files Commands

Overview of Trace Files
Types of Trace Files
Contents of Trace Files
Locations of Trace Files
Trace Files Commands

Overview of Trace Files

A trace file is an administrative file that contain diagnostic data used to investigate problems. Also, trace files can provide guidance for tuning applications or an instance.
Trace Files Commands

Types of Trace Files

Each server and background process can periodically write to an associated trace file. The files information on the process environment, status, activities, and errors.

The SQL trace facility also creates trace files, which provide performance information on individual SQL statements.
Trace Files Commands

Types of Trace Files

To enable tracing for a client identifier, service, module, action, session, instance, or database, you must execute the appropriate procedures in the DBMS_MONITOR package or use Oracle Enterprise Manager.

A dump is a special type of trace file. Whereas a trace tends to be continuous output of diagnostic data, a dump is typically a one-time output of diagnostic data in response to an event (such as an incident).

Trace Files Commands

Types of Trace Files

When an incident occurs, the database writes one or more dumps to the incident directory created for the incident. Incident dumps also contain the incident number in the file name

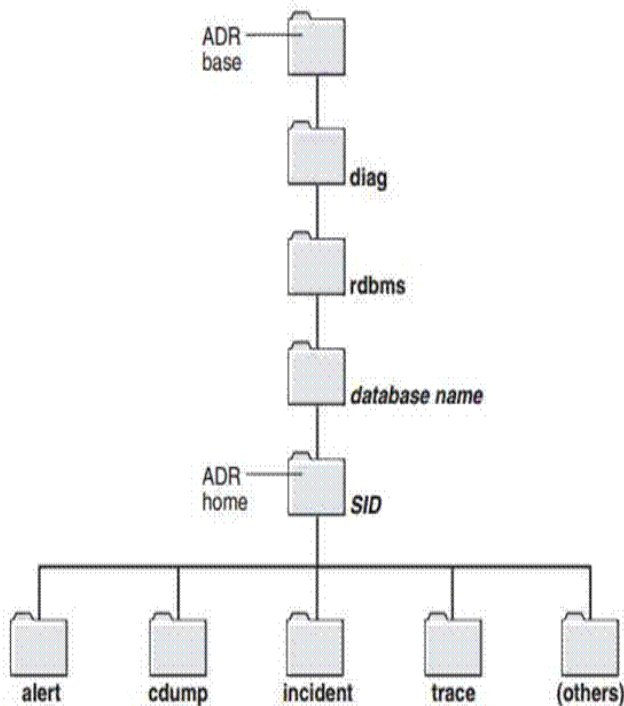
Trace Files Commands

Contents of Trace Files

Typically, database background process trace file names contain the
Oracle SID,
background process name, and
operating system process number

An example of a trace file for the RECO process is **mytest_reco_10355.trc**

Trace Files Commands



This PC > Stuff (E:) > app > Haider > diag > rdbms > orcl > orcl > trace

Name	Date modified	Type	Size
orcl_ora_10068.trc	09/10/2020 8:12 AM	TRC File	32 KB
orcl_reco_11996.trm	09/10/2020 8:11 AM	TRM File	1 KB
orcl_ora_10068.trm	09/10/2020 8:08 AM	TRM File	1 KB
orcl_mmon_17652.trm	09/10/2020 7:56 AM	TRM File	1 KB
orcl_ora_16802.trc	09/10/2020 7:47 AM	TRC File	2 KB

Locations of Trace Files

ADR stores trace files in the trace sub directory, as shown in Figure Trace file names are platform dependent and use the extension .trc.

Introduction to Database Administration

TOPIC 165

Managing users & Security

Managing users & Security

Managing users & Security

Types of Security

Threats to Databases

Security Domain

Schema Objects & User

User & Database

Creating Oracle User

Pre-defined Accounts

Managing users & Security

Types of Security

Following are the database security types and issues

Legal & Ethical Issues

Policy Issues in Enterprises

Systems' related Issues

Needs of Enterprises

Managing users & Security

Threats to Databases

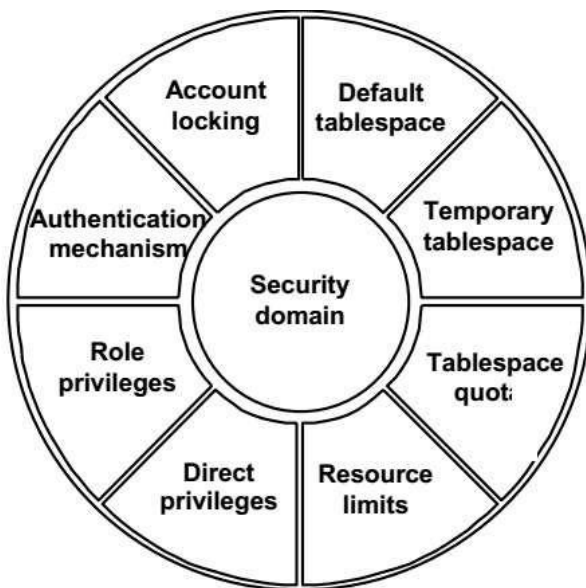
Following are the database related threats and issues

Loss of integrity

Loss of availability

Loss of Confidentiality

Managing users & Security



Security Domain

The database administrator defines the names of the users who are allowed to access a database. A security domain defines the settings that apply to the user.

Authentication mechanisms are

Data Dictionary

Operating System

Network

Managing users & Security

Schema Objects & User

A Schema is a collection of objects

A user is associated (owner) of schema

Creating schema means creating a user
as an owner of schema

Managing users & Security

User & Database

Accessing Database (connection with
the database)

Specifying database unique user
account of ID

Specifying authentication (roles &
privileges) to user account

Allocating space, quota, buffer etc. are
additional features in security.

Managing users & Security

Creating Oracle User

A unique user name or account

Authentications (roles/ rights)

A default tablespace

A temporary tablespace

A user profile

An account status (lock/unlock)

Managing users & Security

Pre-defined Accounts

SYS account (sys as sysdba)

is granted the DBA role

has all privileges with ADMIN
OPTION

is used for startup, shutdown etc.

owns the data dictionary

SYSTEM account is granted the DBA role

Normally not used for routine operations

Introduction to Database Administration

TOPIC 166

Creating Users with Commands

Creating Users with Commands

Creating Users with Commands

Creating Oracle User

Syntax for Creating Oracle User

Data Dictionary View DBA_USERS

Creating Users with Commands

Creating Oracle User

Checklist's clauses

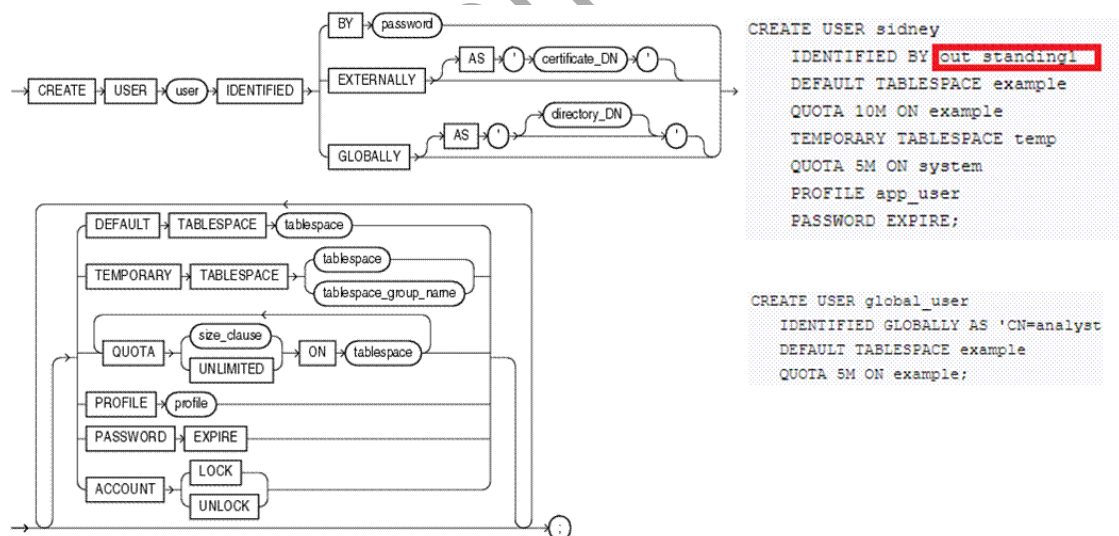
Tablespaces

Quota

Roles/ privileges

Creating Users with Commands

Syntax for Creating Oracle User



docs.oracle.com

Creating Users with Commands

Create User - sys@DBA01_STC-SUN01.US.ORACLE.COM

General Role System Privileges Object Privileges Consumer Group Quota Proxy Users

Name: AARON

Profile: DEFAULT

Authentication: Password

Enter Password: *****

Confirm Password: *****

☐ Expire Password Now

Tablespaces:

Default: USERS

Temporary: DEFAULTTEMPORARY

Status

☐ Locked ☒ Unlocked

Create Cancel Show SQL Help

Creating Oracle User using Oracle-EM

docs.oracle.com

Creating Users with Commands

Data Dictionary View DBA_USERS

```
SQL> select username, account_status, created  
2 from dba_users;
```

USERNAME	ACCOUNT_STATUS	CREATED
NOVUS	OPEN	18-DEC-17
NOVUSDB	OPEN	16-OCT-17
PROD	OPEN	19-JUL-17
NOVUSDB2	OPEN	13-DEC-17
NEONDEV	EXPIRED	23-JAN-17
SCOTT	OPEN	02-APR-10
NEONDB	OPEN	31-MAY-17
TEST	OPEN	14-FEB-17
MDDATA	EXPIRED	02-APR-10
ORACLE OCM	EXPIRED & LOCKED	02-APR-10

Introduction to Database Administration

TOPIC 167

Users information using Data Dictionary Views

Users information using Data Dictionary Views

Users information using Data Dictionary Views

Searching Security Dictionary Views

Single SQL to Display All Views

Listing All Tablespace Quotas

Listing All Profiles & Assigned Limit

Users information using Data Dictionary Views

Searching Security Dictionary Views

Searching Data Dictionary Views closest to security users' information. Search is managed by following SQL LIKE operator: '%QUOTA%', '%PASSWORD%' '%PROFILE%', '%COST%' '%SESS%', '%PROXY%' and *_USERS

Users information using Data Dictionary Views

Single SQL to Display All Views

```
SQL> SELECT *
2 FROM DICTIONARY
3 where table_name like '%QUOTA%'
4 or table_name like '%PASSWORD%'
5 or table_name like '%PROFILE%'
6 or table_name like '%COST%'
7 or table_name like '%SESS%'
8 or table_name like '%PROXY%';
```

TABLE_NAME	COMMENTS
DBA_HIST_SESS_TIME_STATS	CPU and I/O time for interesting (STREAMS) sessions
DBA_LOGMNR_SESSION	
DBA_PROFILES	Display all profiles and their limits
DBA_SQL_PROFILES	set of sql profiles
DBA_TS_QUOTAS	Tablespace quotas for all users
USER_AUDIT_SESSION	All audit trail records concerning CONNECT and DISCONNECT

Users information using Data Dictionary Views

Listing All Tablespace Quotas

```
SELECT * FROM DBA_TS_QUOTAS;
```

TABLESPACE	USERNAME	BYTES	MAX_BYTES	BLOCKS	MAX_BLOCKS
-----	-----	-----	-----	-----	-----
USERS	JFEE	0	512000	0	250
USERS	DCRANNEY	0	-1	0	-1

Users information using Data Dictionary Views

Listing All Profiles & Assigned Limit

```
SELECT * FROM DBA_PROFILES
ORDER BY PROFILE;
```

PROFILE	RESOURCE_NAME	RESOURCE	LIMIT
-----	-----	-----	-----
CLERK	COMPOSITE_LIMIT	KERNEL	DEFAULT
CLERK	FAILED_LOGIN_ATTEMPTS	PASSWORD	DEFAULT
CLERK	PASSWORD_LIFE_TIME	PASSWORD	DEFAULT
CLERK	PASSWORD_REUSE_TIME	PASSWORD	DEFAULT
CLERK	PASSWORD_REUSE_MAX	PASSWORD	DEFAULT
CLERK	PASSWORD_VERIFY_FUNCTION	PASSWORD	DEFAULT
CLERK	PASSWORD_LOCK_TIME	PASSWORD	DEFAULT
CLERK	PASSWORD_GRACE_TIME	PASSWORD	DEFAULT
CLERK	PRIVATE_SGA	KERNEL	DEFAULT
CLERK	CONNECT_TIME	KERNEL	600
CLERK	IDLE_TIME	KERNEL	30

Introduction to Database Administration

TOPIC 168

Managing Privileges

Managing Privileges

Managing Privileges

Types of Oracle User Privileges

System Privileges

System Privileges with ADMIN OPTION

Object Privileges Syntax

Object Privileges - Examples

Objects Privileges Options

Object Privileges using Oracle EM

Managing Privileges

Types of Oracle User Privileges

System: Enables users to perform particular actions in the database

Object: Enables users to access and manipulate a specific object

Managing Privileges

System Privileges

More than 100 distinct system privileges

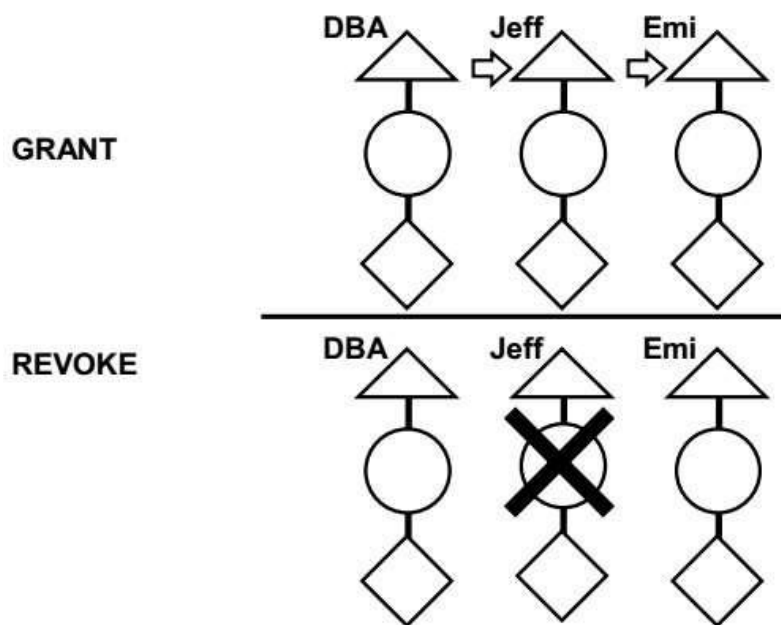
ANY keyword in privileges signifies that users have the privilege in any schema

GRANT command adds a privilege to a user or a group of users

REVOKE command deletes the privileges

Managing Privileges

System Privileges with ADMIN OPTION



Scenario

1. The DBA grants the CREATE TABLE system privilege to Jeff with the ADMIN OPTION.
2. Jeff creates a table.
3. Jeff grants the CREATE TABLE system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the CREATE TABLE system privilege from Jeff.

The result

Jeff's table still exists, but no new tables can be created.

Emi's table still exists and she still has the CREATE TABLE system privilege.

Managing Privileges

Object Privileges Syntax

```
GRANT { object_privilege [(column_list)]  
      [, object_privilege [(column_list)] ]...  
      |ALL [PRIVILEGES]}  
ON   [schema.]object  
TO   {user|role|PUBLIC}[, {user|role|PUBLIC} ]  
      [WITH GRANT OPTION]
```

Managing Privileges

Object Privileges - Examples

grant select on EMP to utest;

grant select, update on DEPT to utest;

grant select on v_emp to utest;

--login with utest select *
from scott.dept;

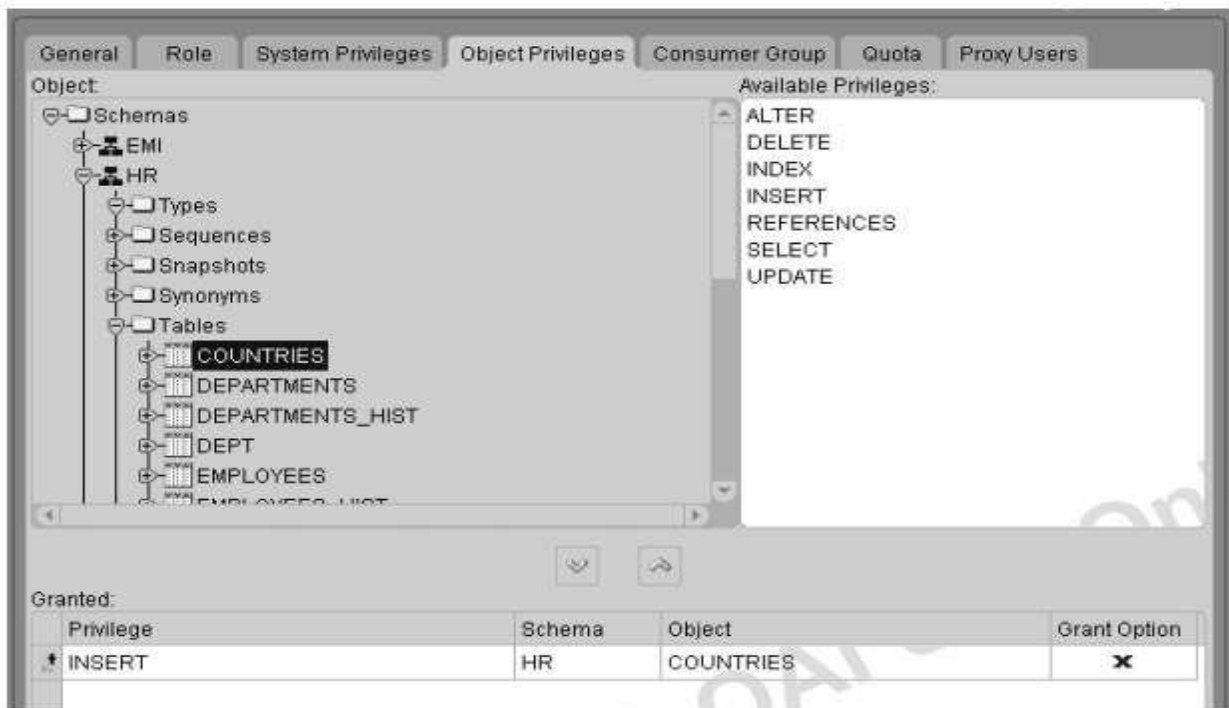
Managing Privileges

Objects Privileges Options

Object priv.	Table	View	Sequence	Procedure
ALTER	√	√	√	
DELETE	√	√		
EXECUTE				√
INDEX	√	√		
INSERT	√	√		
REFERENCES	√			
SELECT	√	√	√	
UPDATE	√	√		

--scott is the owner of EMP table and granting GRANT select on EMP to jef;
Managing Privileges

Object Privileges using Oracle EM



TOPIC 169

System Privileges Commands

System Privileges Commands

System Privileges Commands

System Privileges

System Privileges with ADMIN OPTION

System Privileges - Examples

System Privileges SYSOPER & SYSDBA

System Privileges using Oracle EM

System Privileges Commands

System Privileges

More than 100 distinct system privileges

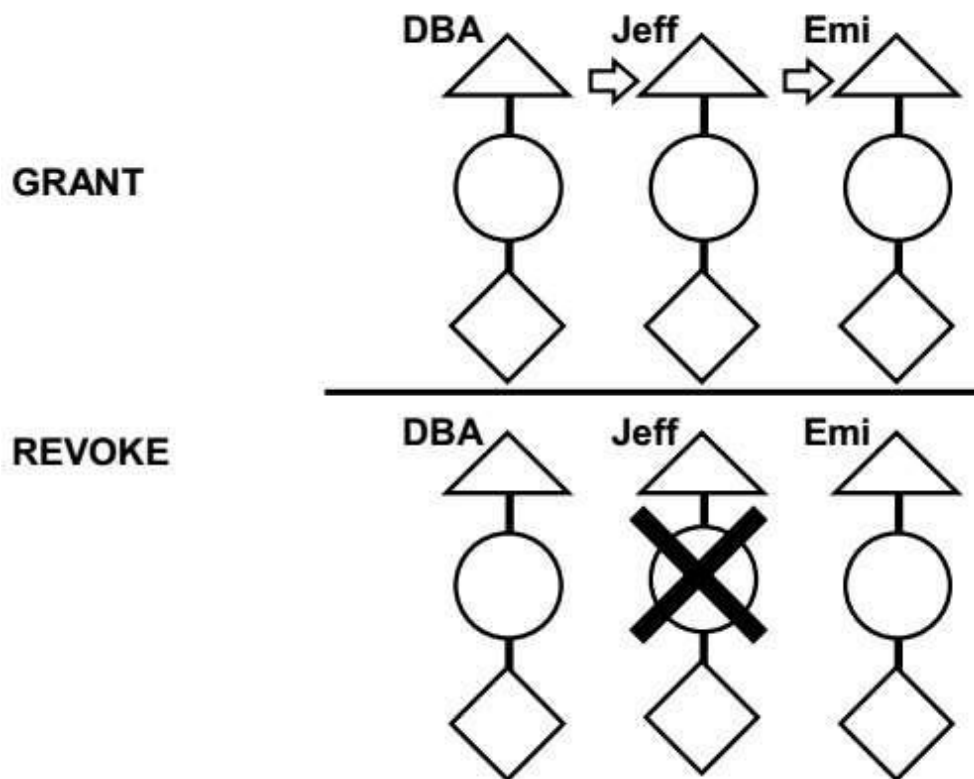
ANY keyword in privileges signifies that users have the privilege in any schema

GRANT command adds a privilege to a user or a group of users

REVOKE command deletes the privileges

System Privileges Commands

System Privileges with ADMIN OPTION



Scenario

1. The DBA grants the `CREATE TABLE` system privilege to Jeff with the `ADMIN OPTION`.
2. Jeff creates a table.
3. Jeff grants the `CREATE TABLE` system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the `CREATE TABLE` system privilege from Jeff.

The result

Jeff's table still exists, but no new tables can be created.

Emi's table still exists and she still has the `CREATE TABLE` system privilege.

System Privileges Commands

System Privileges - Examples

Grant create table to utes;

Grant create view, create procedure to utes; Grant create view to scott;

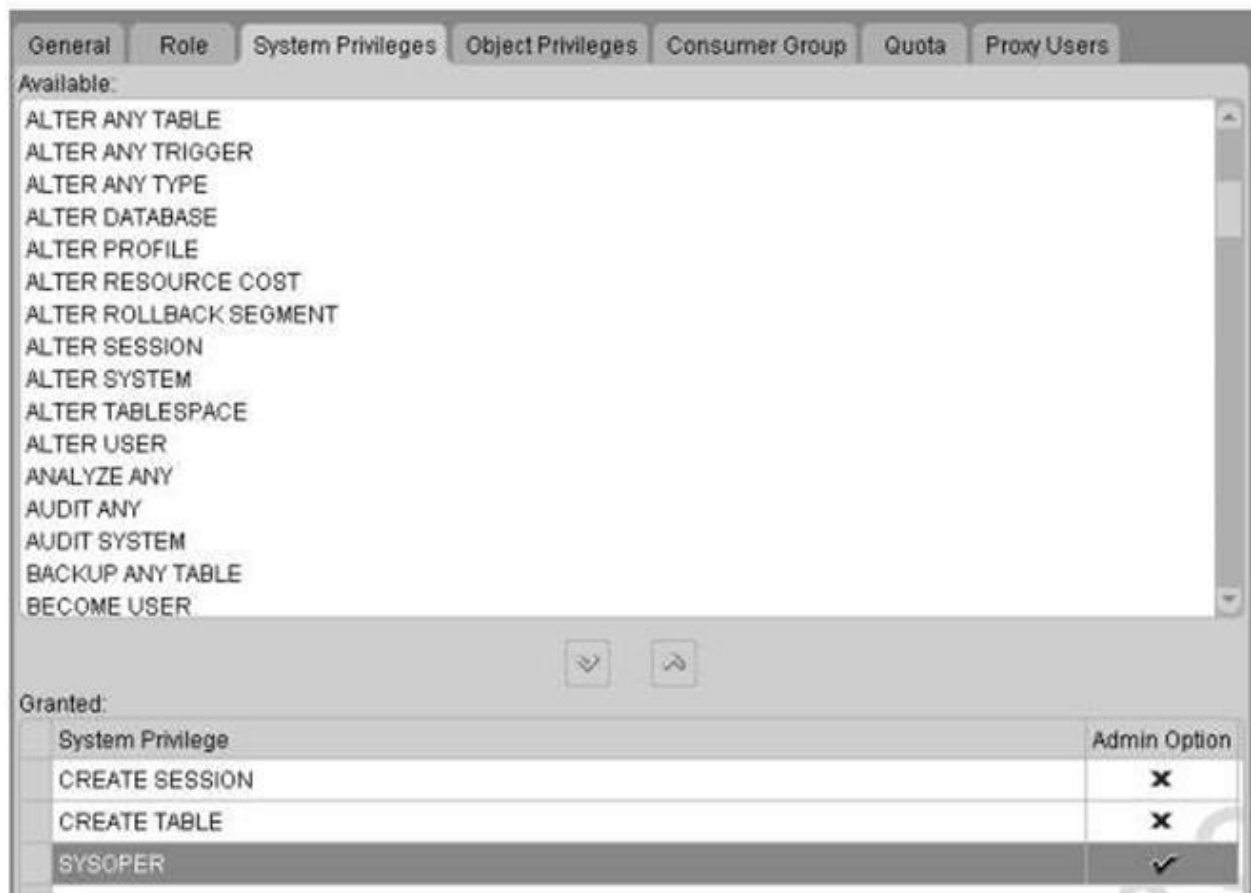
Revoke create table from utest;
System Privileges Commands

Category	Examples
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN MOUNT ALTER DATABASE BACKUP CONTROLFILE TO RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	SYSOPER PRIVILEGES WITH ADMIN OPTION CREATE DATABASE ALTER TABLESPACE BEGIN/END BACKUP RESTRICTED SESSION RECOVER DATABASE UNTIL

System Privileges SYSOPER & SYSDBA

--sys is giving grants to utest
GRANT create table, create table to utest;
System Privileges Commands

System Privileges using Oracle EM



Introduction to Database Administration

TOPIC 170

System Multiple Privileges Commands

System Multiple Privileges Commands

System Multiple Privileges Commands

What is User Privileges?

Giving Privileges to Users

Script Create Synonym

Synonym Privileges to Users

System and Object Privileges

System Multiple Privileges Commands

What is User Privileges?

A user privilege is the right to run a particular type of SQL statement, or the right to access an object that belongs to another user, run a PL/SQL package, and so on.

System Privileges

Object Privileges

Roles

System Multiple Privileges Commands

Giving Privileges to Users

-login sys / as sysdba

create user u1 identified by u1
temporary tablespace temp; conn u1/u1
--error ...

grant create session, create table to u1;
System Multiple Privileges Commands

Script Create Synonym

Run following SQL in owner's (scott) schema and run each command of script in granting user (say u1). Synonyms will be created in u1 schama.

```
SQL> select 'create synonym '||tname||' for scott.'||tname||';'
2  from tab;

'CREATESYNONYM' || TNAME || ' FORSCOTT.' || TNAME || ';'
```

```
create synonym BLOBS for scott.BLOBS;
create synonym BONUS for scott.BONUS;
create synonym CAMPUS for scott.CAMPUS;
create synonym CHAINED_ROWS for scott.CHAINED_ROWS;
create synonym CLIENT for scott.CLIENT;
create synonym CUSTOMER for scott.CUSTOMER;
```

System Multiple Privileges Commands

Synonym Privileges to Users

User u1 is using object table as scott.emp, for example

Select * from scott.EMP; --when scott is authorizing select on EMP to u1

User u1 can run the following command create synonym emp for **scott.emp**;

select * from tab; --can display EMP as synonym

select * from emp;

(Subject to all assigned privileges)

System Multiple Privileges Commands

System and Object Privileges

SYSTEM PRIVILEGES	OBJECT PRIVILEGES
In System privilege are more than 100 privileges	All DML are come into object privileges.
CREATE SESSION	SELECT
CREATE TABLE	INSERT
CREATE USER	UPDATE
CREATE VIEW	DELETE
CREATE SEQUENCE	REFERENCES
CREATE PROCEDURE	ALTER
CREATE ROLE etc.	INDEX
	EXECUTE

TOPIC 171

Revoke Privileges

Revoke Privileges

Revoke Privileges

Data Dictionary for Privileges

Revoke & Example

To See Existing Privileges

Revoke Privileges CASCADE

Revoke Privileges

Data Dictionary for Privileges

We can use string search PRIV to display possible data dictionary views. For Example,

select table_name from dict

where table_name like '%PRIV%';

Revoke Privileges

Data Dictionary for Privileges

We choose following suitable PRIV View:

desc USER_TAB_PRIVS is showing columns GRANTEE

OWNER TABLE_NAME GRANTOR PRIVILEGE GRANTABLE HIERARCHY

Revoke Privileges

Revoke & Example

The REVOKE statement is used to revoke object privileges.

You previously granted the object privilege to the user or role.

You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.

REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
Revoke Privileges

Revoke & Example

Assuming you are the original grantor, the following statement revokes the SELECT and INSERT privileges on the emp table from the users jfee and tsmith:

REVOKE SELECT, insert ON emp FROM jfee, tsmith;
Revoke Privileges

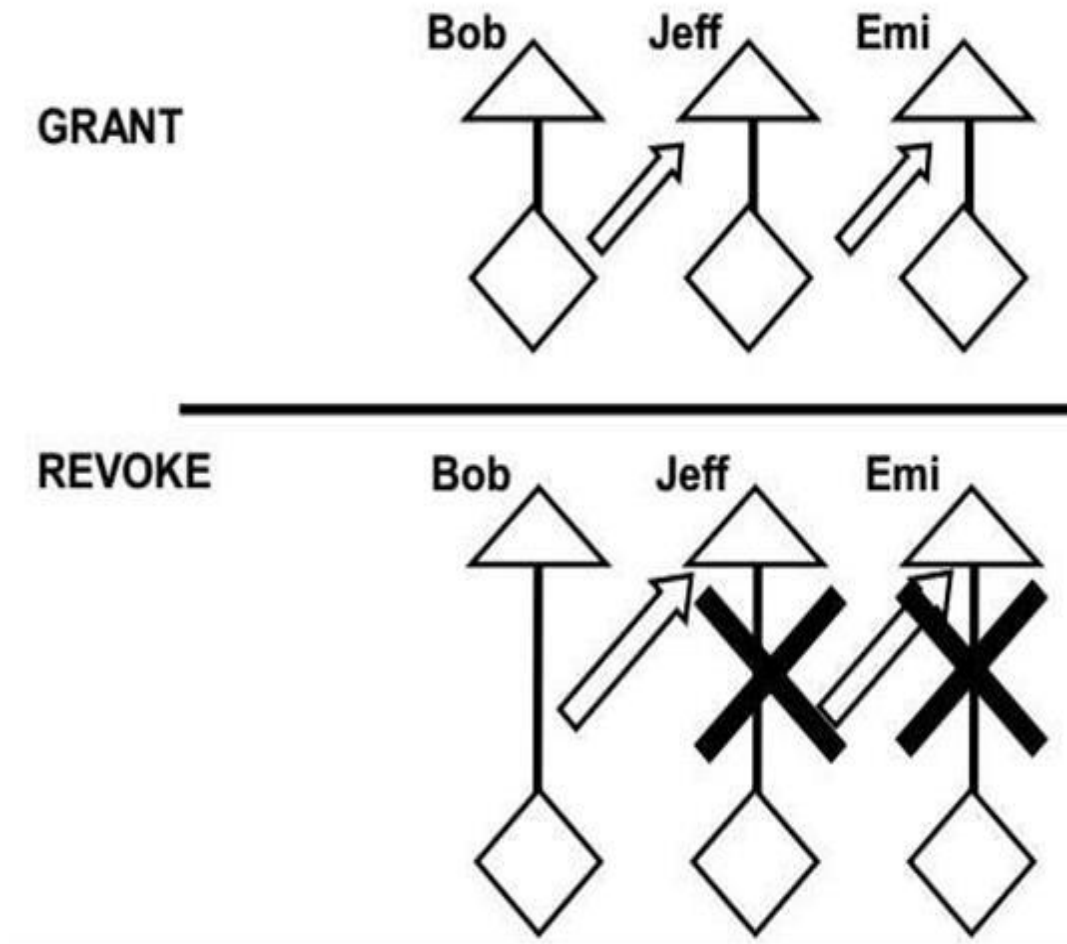
To See Existing Privileges

```
SQL> COLUMN OWNER FORMAT A20
SQL> COLUMN GRANTOR FORMAT A20
SQL> COLUMN GRANTEE FORMAT A20
SQL> COLUMN privilege FORMAT A20
SQL> select OWNER, GRANTOR, GRANTEE, privilege
  2  from USER_TAB_PRIVS
  3  where table_name='EMP';
```

OWNER	GRANTOR	GRANTEE	PRIVILEGE
SCOTT	SCOTT	U1	SELECT

GRANT select on EMP to u1;

Revoke Privileges



Revoke Privileges CASCADE

--Login with Bob

Grant select on EMP to Jeff with grant option;

--Login Jeff

Grant select on EMP to Emi;

--Login with Bob

Revoke

select on EMP from Jeff;

--This revoke is cascaded to Emi

Introduction to Database Administration

TOPIC 172

Managing Roles

Managing Roles

Managing Roles

Introduction

Data Dictionary for Privileges

Privileges & Roles

Roles & Users

Managing Roles

Introduction

Roles are named groups of related privileges or rights.

Each user can be granted roles and privileges for each role, giving them a tailored set of abilities.

Roles define what actions the user can take, and privileges specify the targets to which their roles apply.

You can view the permissions granted by each role, add and remove users, and assign roles and notification profiles to users.

Managing Roles

Data Dictionary for Privileges

We can use string search PRIV to display possible data dictionary views. For Example,

```
select table_name
```

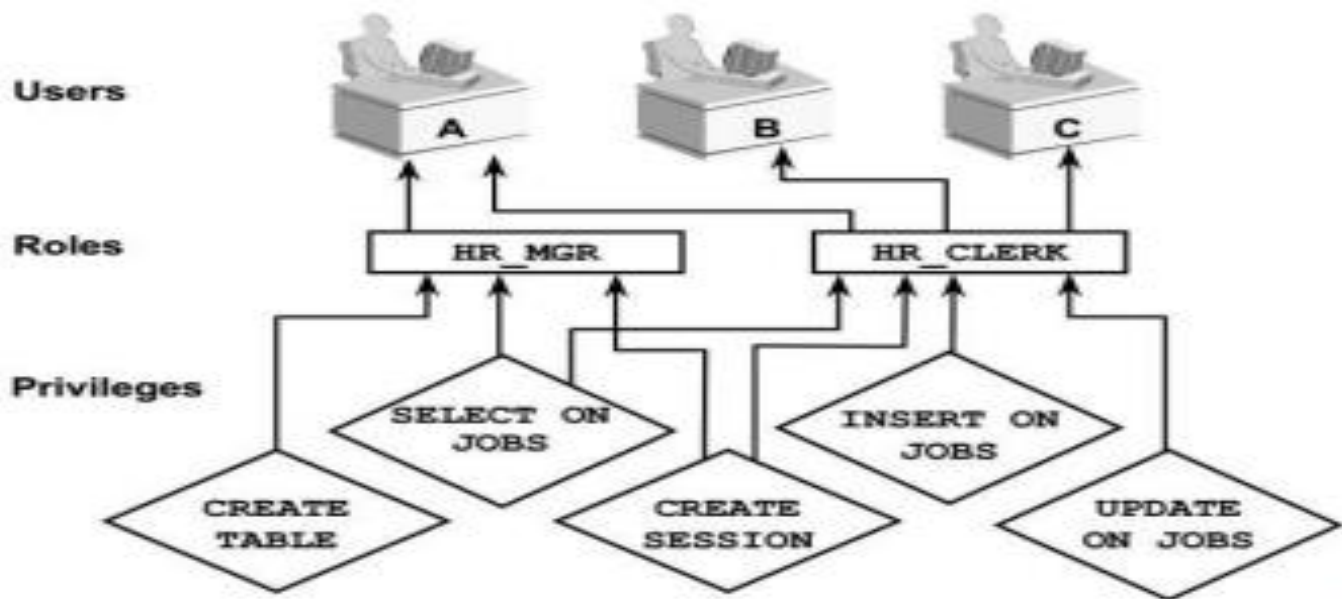
```
from dict
```

```
where table_name like '%PRIV%';
```

Privileges & Roles

Type	Description
System privilege	A system-defined privilege usually granted only by administrators. These privileges allow users to perform specific database operations.
Object privilege	A system-defined privilege that controls access to a specific object.
Role	A collection of privileges and other roles. Some system-defined roles exist, but most are created by administrators. Roles group together privileges and other roles, which facilitates the granting of multiple privileges and roles to users.

Roles & Users



Introduction to Database Administration

TOPIC 173

Benefits of Roles-1

Benefits of Roles-1

Benefits of Roles-1

Benefits of Roles

Standard and Default Roles

Roles of User SYS

Analyzing Roles & Privileges to Users

Benefits of Roles-1

Benefits of Roles

Easier privilege management

Dynamic privilege management

Selective availability of privileges

Can be granted through the operating system

Benefits of Roles-1

Benefits of Roles

Easier privilege management

Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role, and then grant that role to each user.

Dynamic privilege management

If the privileges associated with a role are modified, all the users who are granted the role acquire the modified privileges automatically and immediately.

Benefits of Roles-1

Benefits of Roles

Selective availability of privileges

Roles can be enabled or disabled to turn privileges on or off temporarily. Enabling a role can also be used to verify that a user has been granted that role.

Granting through the operating system

Operating system commands or utilities can be used to assign roles to users in the database.

Benefits of Roles-1

Standard and Default Roles

CONNECT is a role that Enterprise Manager automatically grants to a user when you create a user as shown in "Creating Users". This role has the CREATE SESSION privilege.

RESOURCE extends the privileges of a user beyond those granted by the CONNECT role. It includes CREATE PROCEDURE, CREATE TRIGGER, and other system privileges.

Benefits of Roles-1

Standard and Default Roles

DBA is the standard role that can be granted by an administrator to another administrator. It includes all system privileges and should only be granted to the most trusted and qualified of users. Assigning this role to a user enables the user to administer the database.

Benefits of Roles-1

Roles of User SYS

When you create an Oracle database, the user SYS is automatically created and granted the DBA role.

All base tables and views for the database data dictionary are stored in the schema SYS. These base tables and views are critical for the operation of Oracle Database.

Benefits of Roles-1

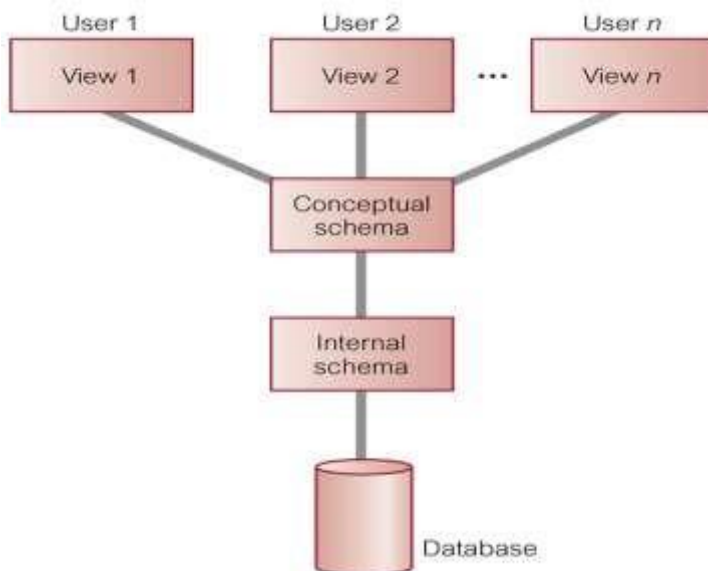
Roles of User SYS

To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by the database. They should never be modified by any user or database administrator.

Also, you should not create any tables in the schema of user SYS, although you can change the storage parameters of the data dictionary settings if necessary.

Benefits of Roles-1

Analyzing Roles & Privileges to Users



Assuming there are three departments in some Enterprise

Sales

Accounting

Human Development & Resource (HR)

Employees are working as an external users

in these departments

Analyzing how Roles and Privileges are need

to be decided

Users Ali(Sales), Jamil(HR) and Nadeem (Accounting)

Introduction to Database Administration

TOPIC 174

Benefits of Roles-2

Benefits of Roles-2

Benefits of Roles-2

Characteristics of Roles

Roles with Synonyms

Comparing Roles with Privileges

Benefits of Roles-2

Characteristics of Roles

Roles can be granted to and revoked from users with the same commands that are used to grant and revoke system privileges.

Roles can be granted to any user or role. However, a role cannot be granted to itself and cannot be granted circularly.

A role can consist of both system and object privileges.

Benefits of Roles-2

Characteristics of Roles

A role can be enabled or disabled for each user who is granted the role.

A role can be required a password to be enabled.

Each role name must be unique among existing usernames and role names.

Roles are not owned by anyone; and they are not in any schema.

Roles have their description stored in the data dictionary.

Benefits of Roles-2

Roles with Synonyms

Roles and privileges are assigned to base table or view. Any Synonym will be granted as per base table or view.

For example,

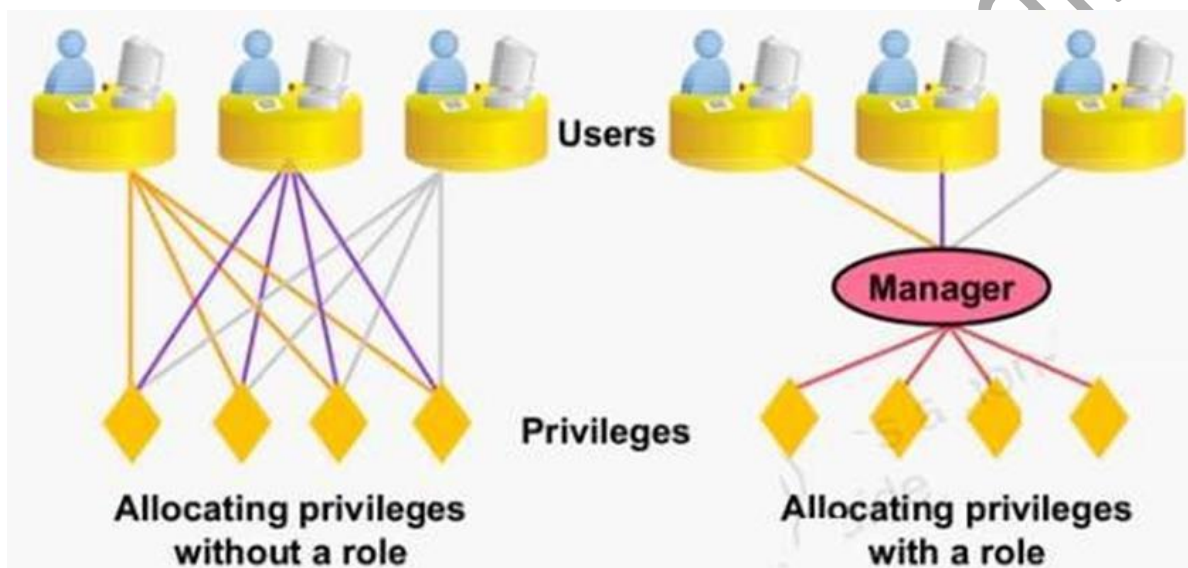
--Table EMP owner gives grant: grant select on emp to u1;

--run in u1's schema

create synonym emp for scott.emp; select * from emp;

Benefits of Roles-2

Comparing Roles with Privileges



Introduction to Database Administration

TOPIC 175

Creating Roles

Creating Roles

Creating Roles

SQL>Create Role

Syntax of SQL>Create Role

Roles with ADMIN option
Syntax of SQL-Create Role using EM
Creating Roles

SQL-Create Role

Use the CREATE ROLE statement to create roles. You must have the CREATE ROLE system privilege to create roles. When you create a role that is not identified externally or by password, the role is granted with the ADMIN option.

You need to decide set of privileges to put in one role and a user who will use those privileges by assigning just role.

Syntax of SQL-Create Role

Use the following command to create a role:

```
CREATE ROLE role [NOT IDENTIFIED | IDENTIFIED  
{BY password | EXTERNALLY | GLOBALLY | USING package}]
```

where:

- **role:** Is the name of the role
- **NOT IDENTIFIED:** Indicates that no verification is required when enabling the role
- **IDENTIFIED:** Indicates that verification is required when enabling the role
- **BY password:** Provides the password that the user must specify when enabling the role
- **USING package:** Creates an application role, which is a role that can be enabled only by applications using an authorized package
- **EXTERNALLY:** Indicates that a user must be authorized by an external service (such as the operating system or a third-party service) before enabling the role
- **GLOBALLY:** Indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the SET ROLE statement, or at login.

Roles with ADMIN option

Not identified:

Create role oe_clerk;

By password:

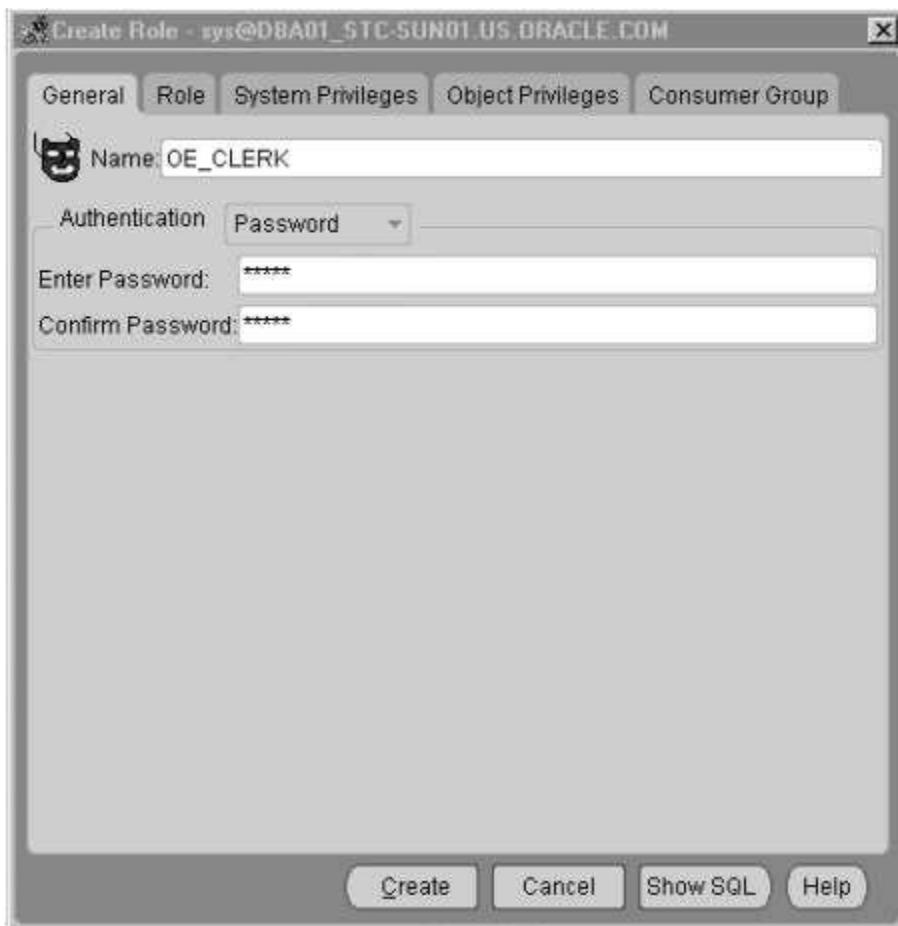
Create role hr_clerk IDENTIFIED BY bonus;

Identified externally:

Create role hr_manager identified Externally;

Creating Roles

Syntax of SQL-Create Role using EM



End

Introduction to Database Administration

TOPIC 176

Pre-defined Roles & Assigning Roles

Pre-defined Roles & Assigning Roles

Pre-defined Roles & Assigning Roles

Meaning of Pre-defined Roles & Assigning Roles

Other Special Roles

List of Pre-defined Roles & Assigning Roles

Assigning Roles - An Enterprise Example

Pre-defined Roles & Assigning Roles

Meaning of Pre-defined Roles & Assigning Roles

The roles listed are defined automatically for Oracle databases when you run database creation scripts.

CONTENT, RESOURCE, and DBA roles are provided for backward compatibility to earlier versions of the Oracle server.

The EXP_FULL_DATABASE and IMP_FULL_DATABASE roles are provided for convenience in using the Import and Export utilities.

Pre-defined Roles & Assigning Roles

Meaning of Pre-defined Roles & Assigning Roles

The roles DELETE_CATALOG_ROLE,

EXECUTE_CATALOG_ROLE, and

SELECT_CATALOG_ROLE are provided for accessing data dictionary views and packages.

These roles can be granted to users who do not have the **DBA role** but who requires an access to the views and tables in the data dictionary.

Pre-defined Roles & Assigning Roles

Other Special Roles

The Oracle server also creates other roles that authorize you to administer the database. On many operating systems, these roles are called OSOPER and OSDBA. Their names may be different on your operating system.

Other roles are defined by SQL scripts provided with the database. For example,

AQ_ADMINISTRATOR_ROLE provides privileges to administer advanced queuing.

AQ_USER_ROLE is obsolete but is kept mainly for release 8.0 compatibility.

SELECT * FROM DBA_SYS_PRIVS;
Pre-defined Roles & Assigning Roles

Role Name	Description
CONNECT, RESOURCE, DBA	These roles are provided for backward compatibility
EXP_FULL_DATABASE	Privileges to export the database
IMP_FULL_DATABASE	Privileges to import the database
DELETE_CATALOG_ROLE	DELETE privileges on data dictionary tables
EXECUTE_CATALOG_ROLE	EXECUTE privilege on data dictionary packages
SELECT_CATALOG_ROLE	SELECT privilege on data dictionary tables

List of Pre-defined Roles & Assigning Roles
Pre-defined Roles & Assigning Roles

Assigning Roles – An Enterprise Example

Mumtaz (Manager)
Usma
Admin Dept



Adnan (Manager)
Abbasi
HR Dept



Fida (Manager)
Ali
Accounting Dept



Empno	Name	DOB	CNIC	Address	City	TelNo	Deptno	EmpCard#
-------	------	-----	------	---------	------	-------	--------	----------

Empno	Name	DOB	CNIC	Address	City	TelNo	Salary	Bonus	Deptno
-------	------	-----	------	---------	------	-------	--------	-------	--------

Empno	Name	DOB	CNIC	Address	City	TelNo	BloodGroup	Salary	Bonus	Deptno	EmpCard#
-------	------	-----	------	---------	------	-------	------------	--------	-------	--------	----------

Introduction to Database Administration

TOPIC 177

Modifying & Align Roles

Modifying & Align Roles

Modifying & Align Roles

Alter Role

Alter Role - Sample SQLs

Alter Role - Syntax

Alter Role - EM

Modifying & Align Roles

Alter Role

ALTER ROLE modifies the authentication method.

Modifying roles requires the ADMIN option or ALTER ANY ROLE privilege

Modifying & Align Roles

Alter Role – Sample SQLs

```
ALTER ROLE oe_clerk  
IDENTIFIED BY order;
```

```
ALTER ROLE hr_clerk  
IDENTIFIED EXTERNALLY;
```

```
ALTER ROLE hr_manager NOT IDENTIFIED;  
Modifying & Align Roles
```

Alter Role – Syntax

```
ALTER ROLE role {NOT IDENTIFIED | IDENTIFIED  
  {BY password | USING package| EXTERNALLY | GLOBALLY }};
```

where:

- **role:** Is the name of the role
- **NOT IDENTIFIED:** Indicates that no verification is required when enabling the role
- **IDENTIFIED:** Indicates that verification is required when enabling the role
- **BY password:** Provides the password used when enabling the role
- **EXTERNALLY:** Indicates that a user must be authorized by an external service (such as the operating system or a third-party service) before enabling the role
- **GLOBALLY:** Indicates that a user must be authorized to use the role by the enterprise directory service before the role is enabled with the SET ROLE statement, or at login

Modifying & Align Roles



Alter Role – EM

Introduction to Database Administration

TOPIC 178

Default Roles

Default Roles

Default Roles

Default Roles

Assigning & Disable Roles - Sample SQLs

Establishing Default Roles

Default Roles

Assigning & Disable Roles – Sample SQLs

```
GRANT oe_clerk TO scott;
```

```
GRANT hr_clerk TO hr_manager;
```

```
GRANT hr_manager TO scott WITH ADMIN OPTION;
```

Disable role during session of user u1 SQL> SET ROLE NONE;

Establishing Default Roles

A user can be assigned many roles.

A user can be assigned a default role.

Limit the number of default roles for a user.

```
ALTER USER scott  
  DEFAULT ROLE hr_clerk, oe_clerk;
```

```
ALTER USER scott DEFAULT ROLE ALL;
```

```
ALTER USER scott DEFAULT ROLE ALL EXCEPT  
  hr_clerk;
```

```
ALTER USER scott DEFAULT ROLE NONE;
```

Default Roles

Establishing Default Roles

A user can be assigned many roles. A default role is a subset of these roles that is automatically enabled when the user logs on. By default, all the roles assigned to a user are enabled at logon without the need of a password. Limit the default roles for a user with the *ALTER USER* command.

Default Roles

Establishing Default Roles

The DEFAULT ROLE clause applies only to roles that have been granted directly to the user with a GRANT statement. The DEFAULT ROLE clause cannot be used to enable the following:

Roles not granted to the user.

Roles granted through other roles.

Roles managed by an external service (such as the operating system).

Default Roles

Establishing Default Roles

Use the following syntax to assign default roles to a user:

```
ALTER USER user DEAFULT ROLE  
{role [,role]...| ALL [EXCEPT role [,role]...] |  
NONE}
```

Where:

User: is the name of the user who is granted the roles

Role: is the role to be made the default
role for the user

Introduction to Database Administration

TOPIC 179

Application Roles

Application Roles

Application Roles

Application Roles

Application Roles - SQL

Application Roles

Application Roles

Application roles can be enabled only by authorized PL/SQL packages

The USING package clause creates an application role

```
CREATE ROLE admin_role  
IDENTIFIED USING hr.employee;
```

Application Roles

Application Roles

The USING package clause in the CREATE ROLE statement creates an application role. An application role can be enabled only by applications by using an authorized PL/SQL package. Application developers do not need to secure a role by embedding passwords inside applications. Instead, they can create an application role and specify which PL/SQL package is authorized to enable the role.

Application Roles

Application Roles - SQL

```
CREATE ROLE admin_role IDENTIFIED USING hr.employee;
```

In this example, admin_role is an application role and the role can be enabled only by modules that are defined inside the hr.employee PL/SQL package.

End

Introduction to Database Administration

TOPIC 180

Enable & Disable Role

Enable & Disable Role

Enable & Disable Role

Restrictions on Enable & Disable Role

Specifying Roles to be Enabled

Restrictions on Enable & Disable Role - Sample

Restrictions on Enable & Disable Role - Syntax

Enable & Disable Role

Enable & Disable Role

Disable a role to temporarily revoke the role from a user.

Enable a role to grant it temporarily.

The SET ROLE command enables and disables roles

Default roles are enabled for a user at login.

A password may be required to enable a role.

Enable & Disable Role

Enable & Disable Roles

Enable and disable roles to temporarily activate and deactivate the privileges associated with the roles.

To enable a role, a role must first be granted to the user.

Enable & Disable Role

Enable & Disable Roles

When a role is enabled, the user can use the privileges granted to that role. If a role is disabled, the user cannot use the privileges associated with that role unless those privileges are granted directly to a user or to another role enabled for that user. Roles are enabled for a session. At the next session, the user's active roles revert to default roles.

Enable & Disable Role

Specifying Roles to be Enabled

The SET command and the DBMS_SESSION.SET_ROLE procedure enable all of the roles included in the command and disable all other roles. Roles can be enabled from any tool or program that allows PL/SQL commands; however, a role cannot be enabled in a stored procedure.

You can use the ALTER USER-DEFAULT ROLE command to indicate which roles will be enabled for a user at login. All other roles are disabled.

Enable & Disable Role

Specifying Roles to be Enabled

A password may be required to enable a role. The password must be included in the SET ROLE command to enable the role.

Default roles assigned to a user do not require a password; they are enabled at login, the same as a role without a password.

Enable & Disable Role

Restrictions on Enable & Disable Role

A role cannot be enabled from a stored procedure, because this action may change the security domain (set of privileges) that allowed the procedure to be called in the first place.

So, in PL/SQL, Role can be enabled and disabled in anonymous blocks and application procedures (for example, client Oracle Forms procedures), but not in stored procedures.

Enable & Disable Role

Restrictions on Enable & Disable Role

IF a stored procedure contains the command SET ROLE, the following error is generated at run time:

ORA-06565: cannot execute SET ROLE from within stored procedure

Enable & Disable Role

Restrictions on Enable & Disable Role - Sample

```
SET ROLE hr_clerk;
```

```
SET ROLE oe_clerk IDENTIFIED BY order;
```

```
SET ROLE ALL EXCEPT oe_clerk;
```

Enable & Disable Role

Restrictions on Enable & Disable Role - Syntax

```
SET ROLE {role [ IDENTIFIED BY password ]  
        [, role [ IDENTIFIED BY password ]]  
        | ALL [ EXCEPT role [, role ] ]  
        | NONE }
```

Introduction to Database Administration

TOPIC 181

Removing Roles

Removing Roles

Removing Roles

Revoke Roles

Revoke Roles from Users

Revoke Roles using OEM

Removing Roles

Removing Roles

Revoke Roles

Disable a role to temporarily revoke the role from a user.

Enable a role to grant it temporarily.

The SET ROLE command enables and disables roles

Default roles are enabled for a user at login.

A password may be required to enable a role.

Removing Roles

Revoke Roles from Users

To revoke a role from a user, use the SQL statement REVOKE. Any user with the admin option for a role can revoke the role from any other database user or role. Also, users with the GRANT ANY ROLE privilege can revoke any role.

Removing Roles

Revoke Roles from Users

```
REVOKE role [, role ]  
FROM {user|role|PUBLIC}  
[, {user|role|PUBLIC} ]
```

where

role: Is the role to be revoked or the role from which roles are revoked.

user: is the user from which the system privileges or roles are revoked.

PUBLIC: Revokes the privilege or role from all users

Removing Roles

Revoke Roles using OEM

Using Oracle Enterprise Manager to Revoke a Role from a User

From the OEM Console:

Navigate to Security>Users.

Highlight the user for whom a role is to be revoked.

Navigate to Roles Granted.

Select the role to be revoked.

Select Revoke from the right-mouse menu.

Select Yes to confirm revocation.

Removing Roles

Removing Roles

To remove a role from the database, use the following syntax:

SQL> DROP ROLE role;

When you drop a role, the Oracle server revokes it from all users and roles to whom it has been granted and removes it from the database.

In order to drop the role, you must have been granted the role with ADMIN OPTION or have the DROP ANY ROLE system privilege.

Removing Roles

Removing Roles

- **Dropping a role:**
 - Removes it from all users and roles it was granted
 - Removes it from the database
- **Requires the ADMIN OPTION or DROP ANY ROLE privilege**
- **To drop a role:**

```
DROP ROLE hr_manager;
```

Introduction to Database Administration

TOPIC 182

Guidelines for Creating Roles

Guidelines for Creating Roles

Guidelines for Creating Roles

Guidelines

Guidelines for an Example

Guidelines for Assigning Roles

Guidelines for using Password and Default Roles

Roles with Password

Guidelines for Creating Roles

Guidelines

Because a role includes the privileges that are necessary to perform a task, the role name is usually an application task or a job title.

Our example uses both application tasks and job titles for role names.

Guidelines for Creating Roles

Guidelines for an Example

Use the following steps to create, assign, and grant users roles:

Create a role for each application task. The name of the application role corresponds to a task in the application, such as PAYROLL.

Assign the privileges necessary to perform the task to the application role.

Guidelines for Creating Roles

Guidelines for an Example

Create a role for each type of user. The name of the user role corresponds to a job title, such as PAY_CLERK.

Grant application roles to user's roles.

Grant user's roles to users.

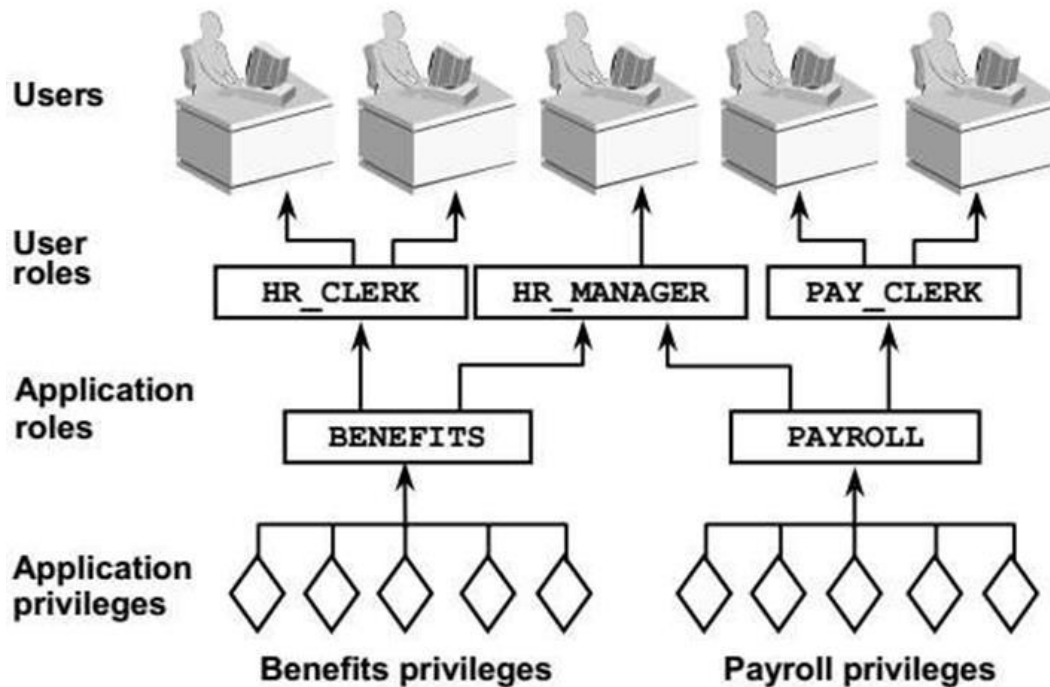
Guidelines for Creating Roles

Guidelines for an Example

If a modification to the application requires that the new privileges are needed to perform the payroll task, then the DBA only needs to assign the new privileges to the PAYROLL application role. All of the users that are currently performing this task will receive the new privileges.

Guidelines for Creating Roles

Guidelines for Assigning Roles



Guidelines for Creating Roles

Guidelines for using Password and Default Roles

Passwords provide an additional level of security when enabling a role.

For example, the application might require a user to enter a password when enabling the **PAY_CLERK** role, because this role can be used to issue checks.

Guidelines for Creating Roles

Guidelines for using Password and Default Roles

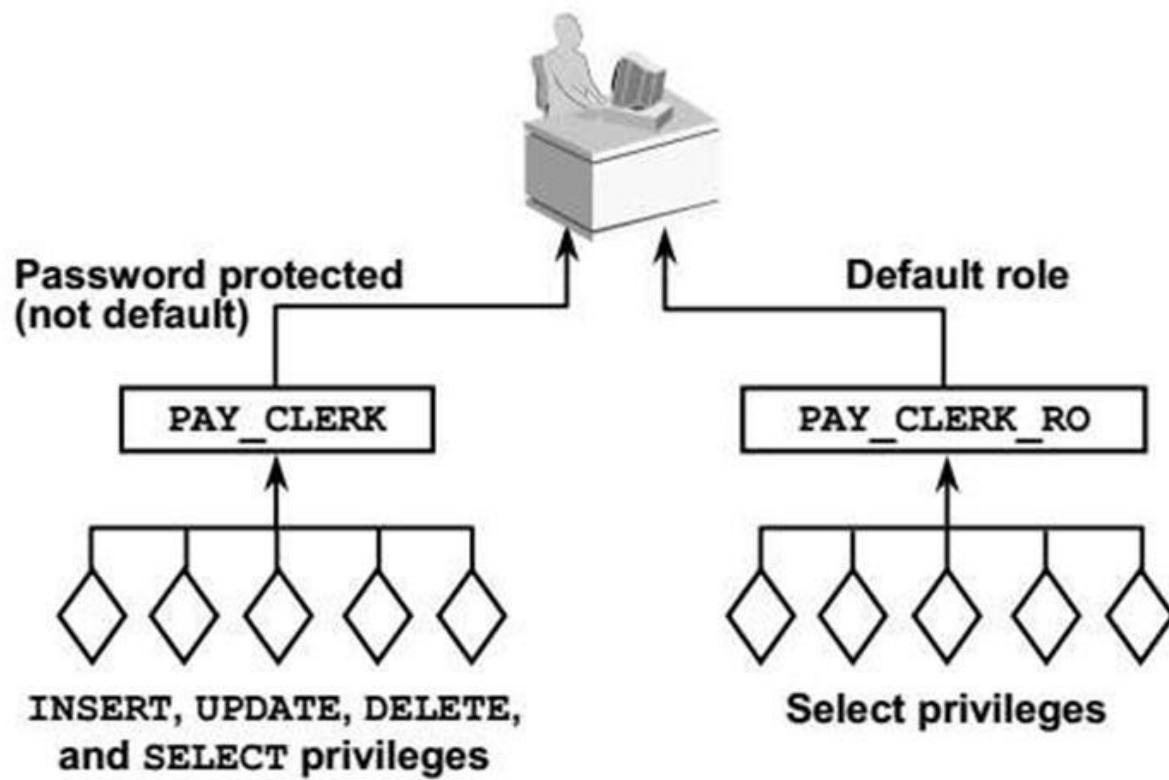
Passwords allow a role to be enabled only through an application. This technique is shown in the example in the slide.

The DBA has granted the user two roles, **PAY_CLERK** and **PAY_CLERK_RO**.

The **PAY_CLERK** has been granted all of the privileges that are necessary to perform the payroll clerk function.

Guidelines for Creating Roles

Roles with Password



Introduction to Database Administration

TOPIC 183

Using Roles Data Dictionary Views
Using Roles Data Dictionary Views

Using Roles Data Dictionary Views

Data Dictionary Views of Roles

Roles' Views

Privileges and Roles - An Example

Using Roles Data Dictionary Views

Data Dictionary Views of Roles

Information about roles can be obtained by querying the following SQL

```
select table_name, comments from DICT
```

```
where table_name like '%ROLE%';
```

Using Roles Data Dictionary Views

Roles' Views

DBA_ROLES: All roles that exist in the database

DBA_ROLE_PRIVS: Roles granted to users and roles

ROLE_ROL_PRIVS: Roles that are granted to roles

DBA_SYS_PRIVS: System privileges granted to users and roles

ROLE_SYS_PRIVS: System privileges granted to roles

ROLE_TAB_PRIVS: Object privileges granted to roles

SESSION_ROLES: Roles that the user currently has enabled

Using Roles Data Dictionary Views

Privileges and Roles – An Example

User is **u1**

Privileges are

select, update for table **EMP** Assigning Role EMP_SU Assigning EMP_SU to user **u1**

Using Roles Data Dictionary Views

Privileges and Roles – An Example

To user has which roles

```
SQL> select Grantee, granted_role
  2   from DBA_ROLE_PRIVS
  3   where rownum<=10;
```

GRANTEE	GRANTED_ROLE
U1	EMP_SU
DBA	OLAP_DBA
DBA	XDBADMIN
DBA	JAVA_ADMIN
DBA	JAVA_DEPLOY
DBA	OLAP_XS_ADMIN
DBA	WM_ADMIN_ROLE
DBA	SCHEDULER_ADMIN
DBA	XDB_SET_INVOKER
DBA	EXP_FULL_DATABASE

Introduction to Database Administration

TOPIC 184

Introduction to Data Migration & Data Loading

Introduction to Data Migration & Data Loading

Introduction to Data Migration & Data Loading

Data Migration

Data Migration - Source to Target Schema

Data Migration Choices

Data Migration & Staging Schema

Introduction to Data Migration & Data Loading

Data Migration

Data migration is the process of transferring data from one storage system or computing environment to another.

Non integrated/ distributed data can be integrated through data migration

Introduction to Data Migration & Data Loading

Data Migration

There are many reasons your enterprise might need to undertake a data migration project. For example, you might be replacing servers or storage devices or consolidating or decommissioning data center.

Data migration is also an essential step in the overall process of migrating on-premises IT infrastructure to a cloud computing environment.

Introduction to Data Migration & Data Loading

Data Migration

Whether you are moving to a public cloud, private cloud, hybrid cloud, or multi-cloud environment, you will need to find a secure, cost-effective, and efficient method of migrating your data to its new storage location.

Introduction to Data Migration & Data Loading

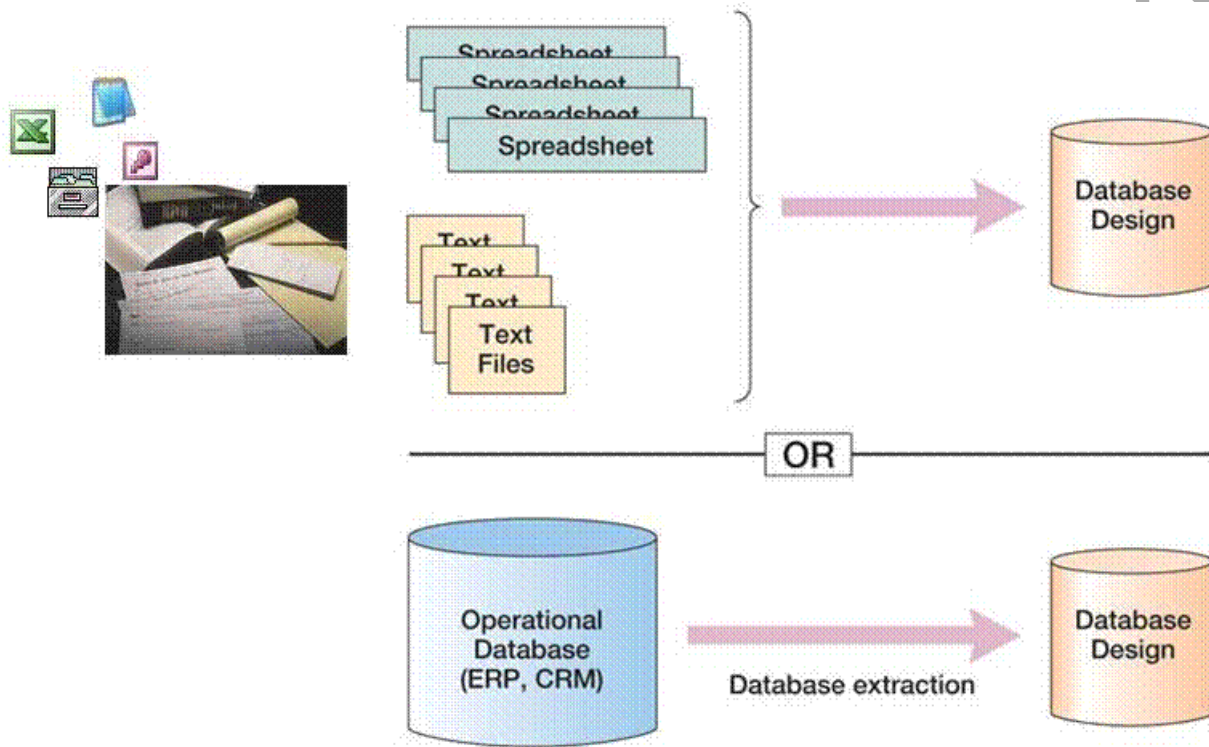
Data Migration – Source to Target Schema

Typically, the first step in the database migration process involves converting the source database's schema (if necessary) so that it's compatible with the target database. A database's

schema is like a blueprint for how it is organized, controlling its logical architecture and structure. If the target database management system uses a data definition language (DDL) that is not compatible with the source's, the schema will need to be converted.

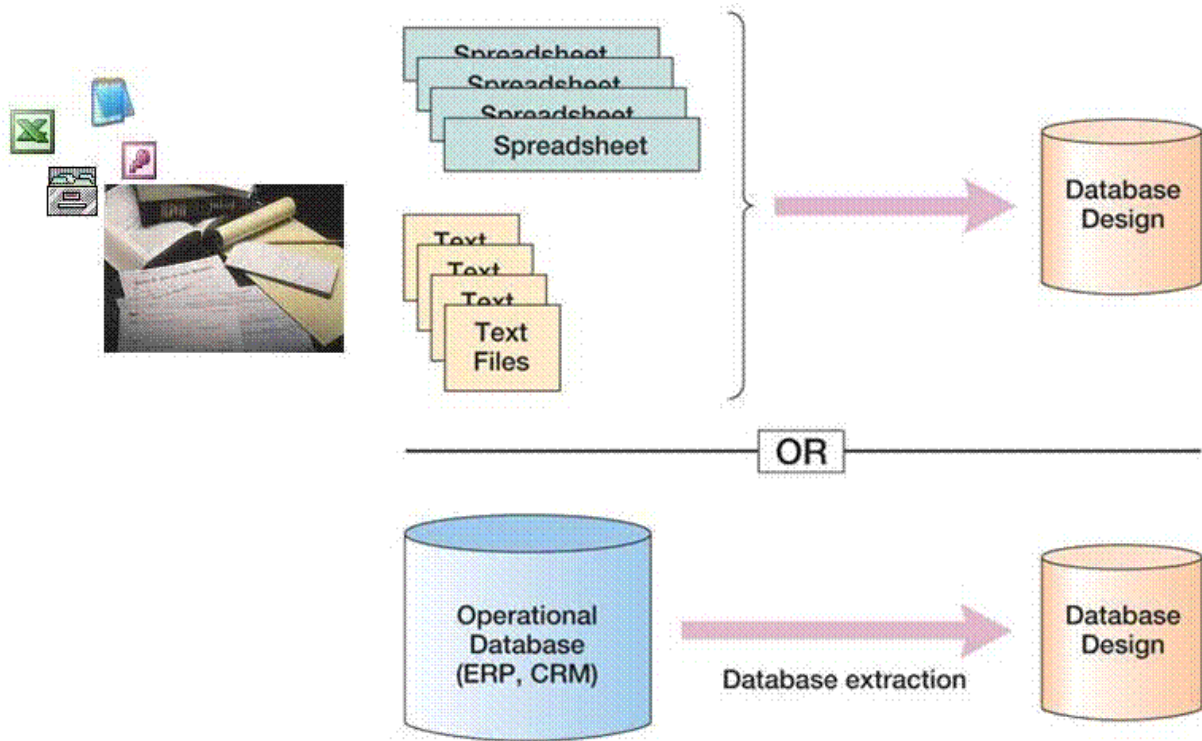
Introduction to Data Migration & Data Loading

Data Migration Choices



Introduction to Data Migration & Data Loading

Data Migration Ways

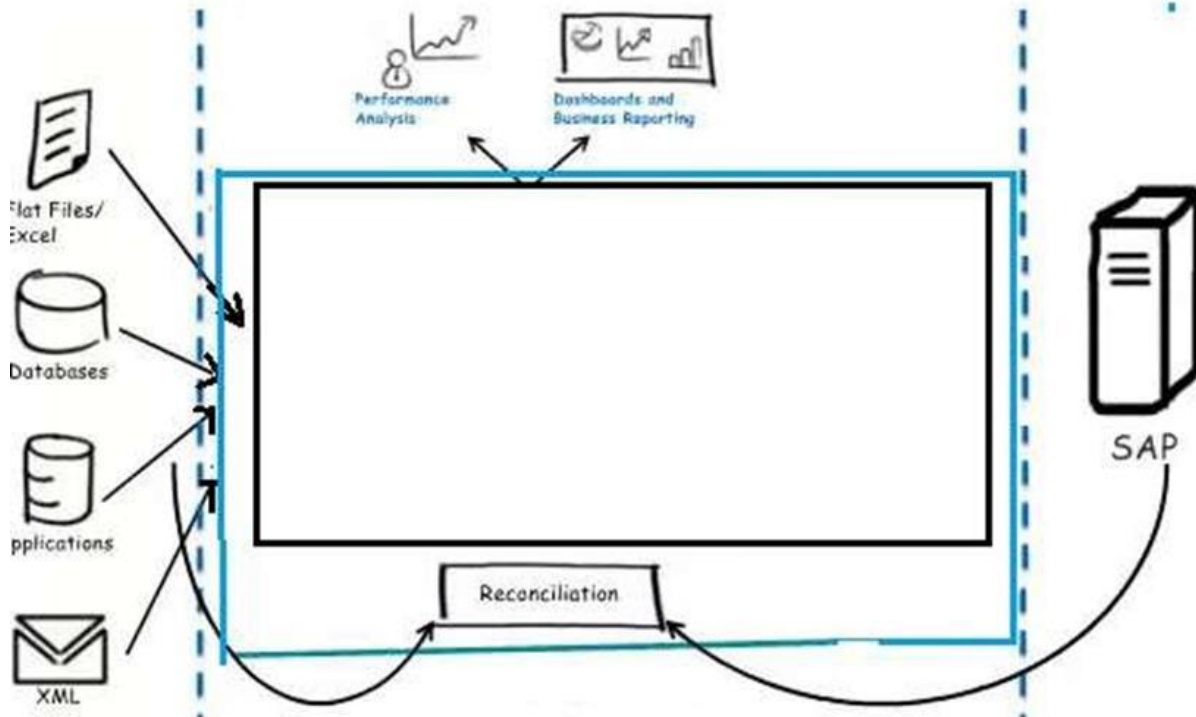


Introduction to Data Migration & Data Loading

Data Migration Ways



Introduction to Data Migration & Data Loading



Data Migration & Staging Schema

Introduction to Database Administration

TOPIC 185

Data Loading Methods

Data Loading Methods

Data Loading Methods

Introduction

SQL*Loader Method

Direct Load Method

Oracle Export and Oracle Import

Data Loading Methods

Introduction

Several methods are available for loading data into tables in an Oracle database. Of the methods available, Direct Load insert and SQL*Loader are discussed here.

Data Loading Methods

SQL*Loader Method

SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that places little limitation on the format of the data in the data file.

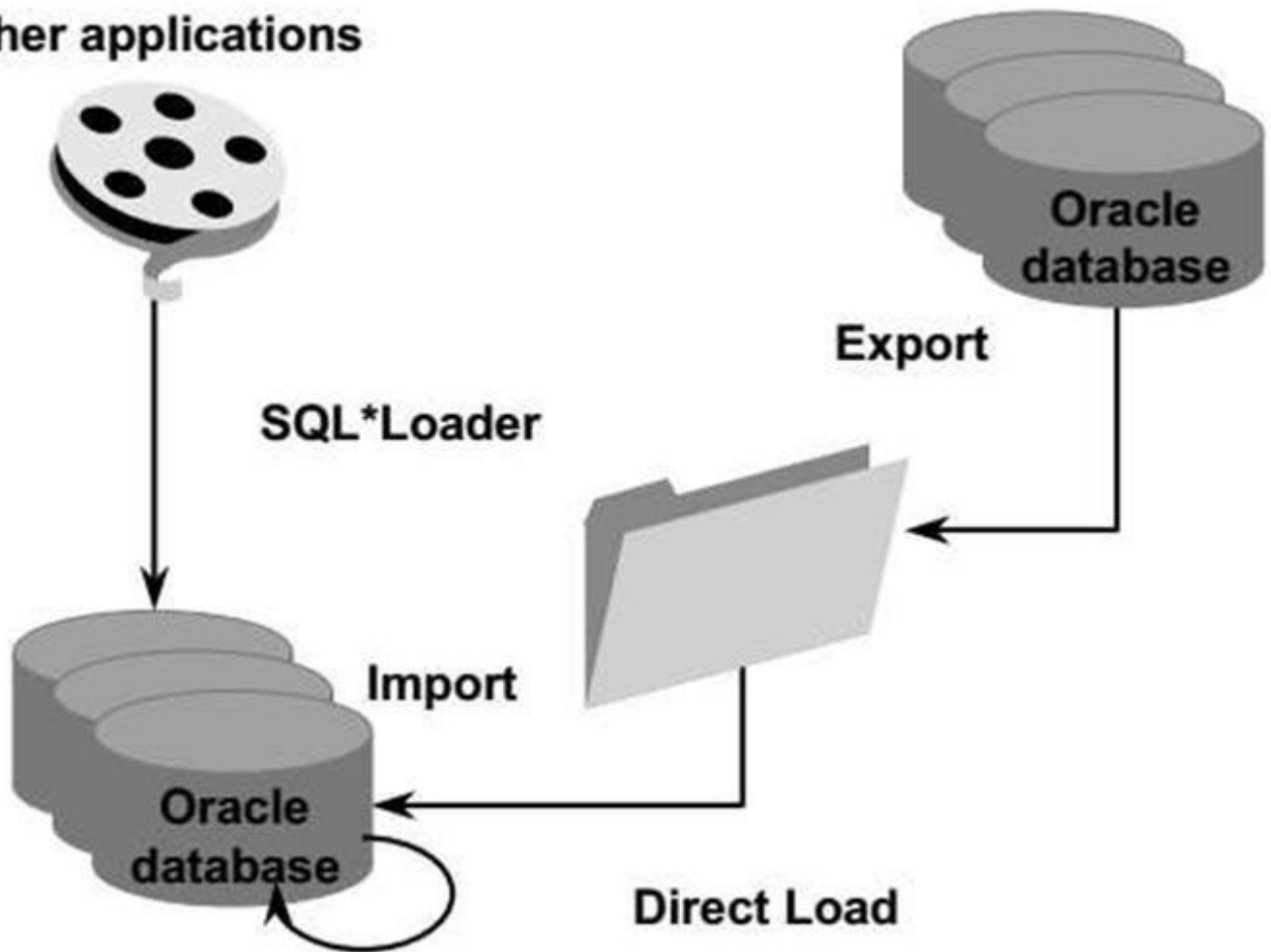
Data Loading Methods

Direct Load Method

Direct Load insert can be used to copy data from one table to another table within the same database. It speeds up the insert operation, bypassing the database buffer cache and writing data directly into the data files.

Data Loading Methods

Other applications



Oracle Export and Oracle Import
Introduction to Database Administration

TOPIC 186

Direct & Parallel Load

Direct & Parallel Load

Direct & Parallel Load

Direct Load

Serial Direct Load - Example

Parallel Direct Load

Parallel Direct Load - Example

Direct & Parallel Load

Direct Load

Direct Load insert can be performed in the following ways:

Normal (Serially), or in parallel

Into partition tables, non-partitioned

tables, or single partitions of a table

With or without logging of redo data

Direct & Parallel Load

Direct Load

Direct Load insert (serial or parallel) can only support the INSERT ...

SELECT syntax of an INSERT statement, not the INSERT Values syntax. The parallelism for INSERT

SELECT is determined from either parallel

hints or parallel table definition.

Oracle provides syntax extensions that extend the scope of the INSERT SELECT statement, so that you can insert rows into multiple tables as part of a single DML statement.

Direct & Parallel Load

Direct Load

A Direct Load insert can be invoked by using the APPEND hint, as shown in following command:

```
INSERT /*+APPEND */ INTO [ schema. ]  
table [ [NO] LOGGING ]  
sub query;
```

Where

schema: Is the owner of the table

table: Is the name of the table

sub-query: Is the sub-query used to select the columns and rows for insert

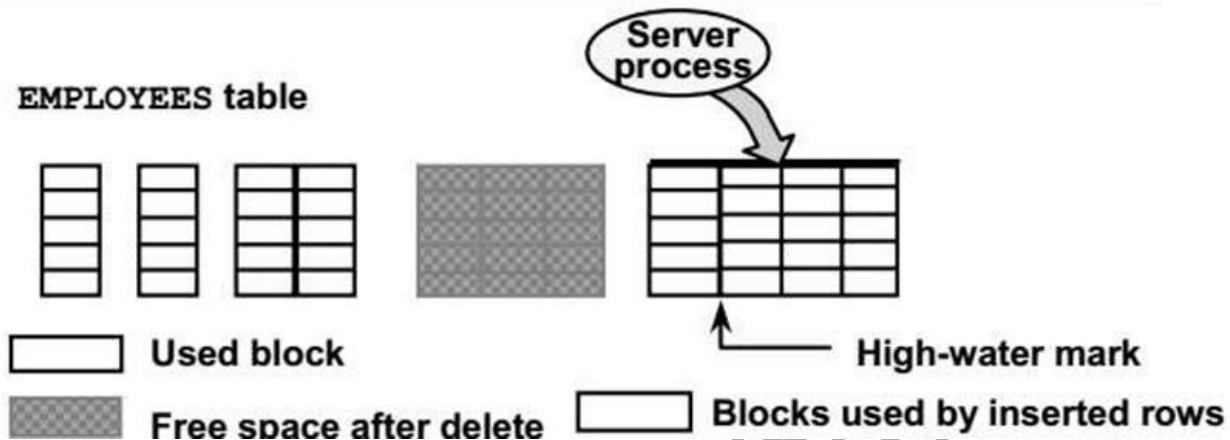
Direct & Parallel Load

Serial Direct Load - Example

```

INSERT /*+ APPEND */ INTO emp
NOLOGGING
SELECT * FROM t_employees;
COMMIT;

```



Direct & Parallel Load

Parallel Direct Load

Direct Load inserts can be made in parallel by one of the following methods:

Using a PARALLEL hint in the INSERT statement, as in the above example

Creating the table or altering it to specify the PARALLEL clause

When parallel Direct Load inserts are made, the Oracle server uses several processes, known as parallel query slaves, to insert data into the table.

Direct & Parallel Load

Parallel Direct Load

Temporary segments are allocated to store the data inserted by each slave process.

When the transaction commits, the extents in these individual segments become a part of the table in which records are inserted.

The ALTER SESSION ENABLE PARALLEL DML command must be executed at the beginning of a transaction.

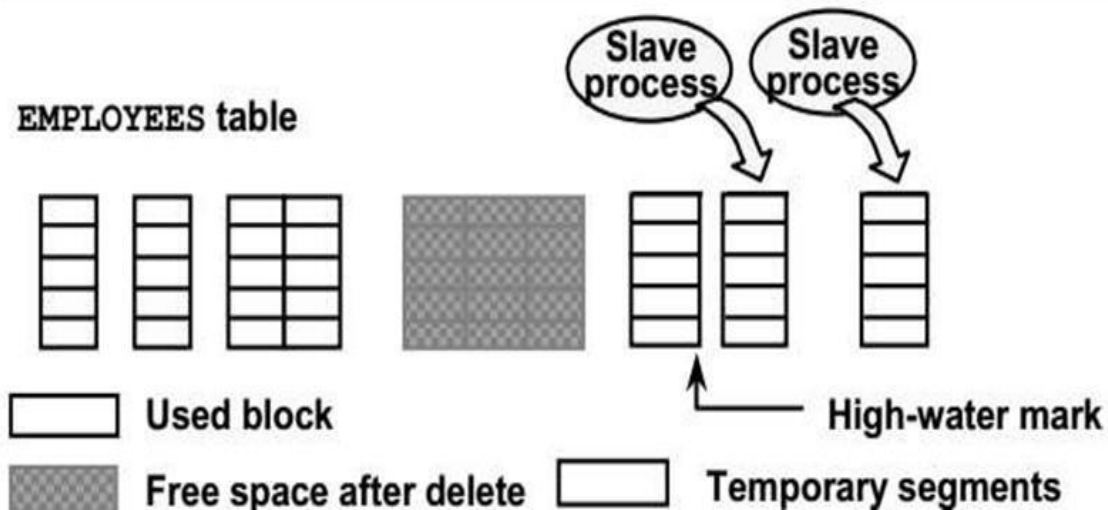
An object that is modified using parallel

Direct Load insert cannot be queried or modified again within the same transaction.

Direct & Parallel Load

Parallel Direct Load - Example

```
ALTER SESSION ENABLE PARALLEL DML;  
INSERT /*+PARALLEL(hr.employees,2) */  
INTO hr.employees NOLOGGING  
SELECT * FROM hr.old_employees;
```



Introduction to Database Administration

TOPIC 187

SQL*Loader
SQL*Loader

SQL*Loader

Introduction
Files used by SQL*Loader

Using SQL*Loader

SQL*Loader

Introduction

SQL*Loader loads data from external files into tables in an Oracle database.

SQL*Loader has the following features:

SQL*Loader can use one or more input files.

Several input records can be combined into one logical record for loading.

Input fields can be of fixed or variable lengths.

Input data can be in any format: character, binary, packed decimal, date, and zoned decimal.

SQL*Loader

Introduction

Data can be loaded from different types of media such as disk, tape, or named pipes.

Data can be loaded into several tables in one run.

Options are available to replace or to append to existing data in the tables.

SQL functions can be applied on the input data before the row is stored in the database.

SQL*Loader

Introduction

Column values can be auto generated based on rules. For example, a sequential key value can be generated and stored in a column.

Data can be loaded directly into the table, bypassing the database buffer cache.

SQL*Loader

Files used by SQL*Loader

SQL*Loader uses the following files:

Loader control file: Specifies the input format, output tables, and optional conditions that can be used to load only part of the records found in the input data files

Input data files: Contain the data in the format defined in the control file

Parameter file: Is an optional file that can be used to define the command line parameters for the load

SQL*Loader

Files used by SQL*Loader

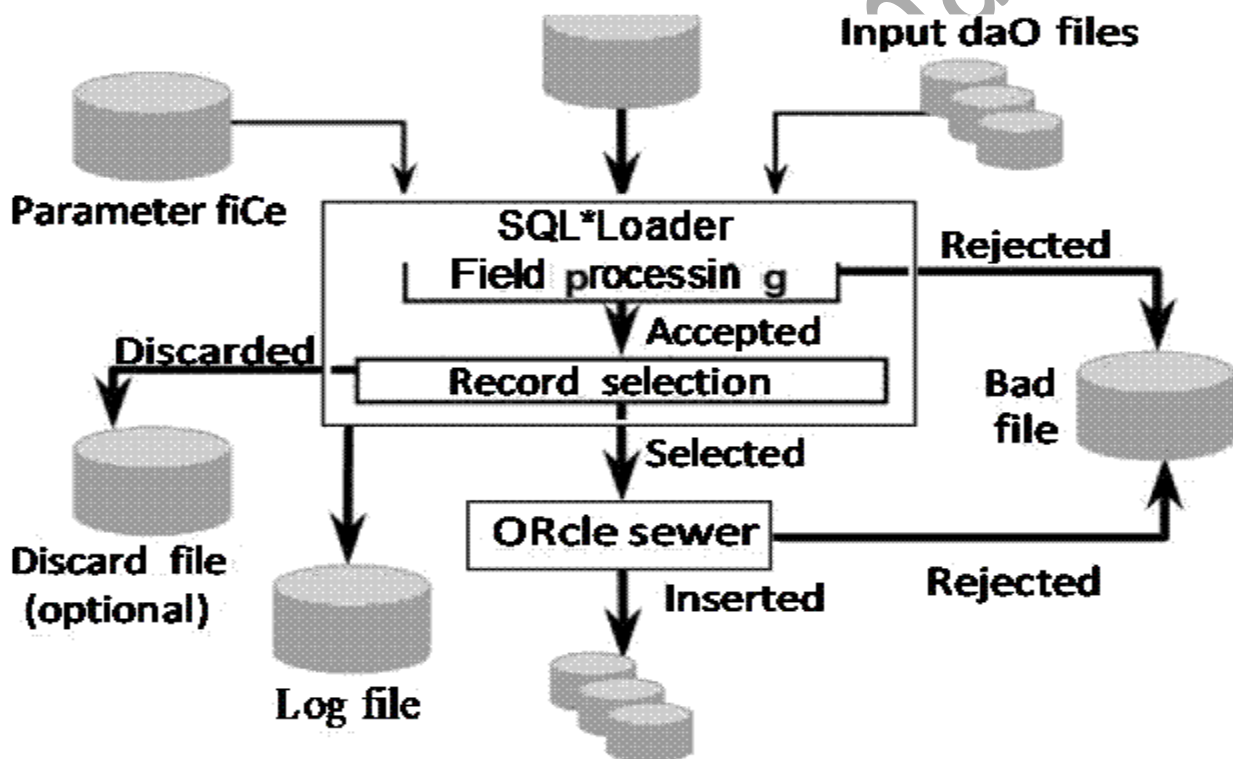
Log file: Is created by SQL*Loader and contains a record of the load

Bad file: Is used by the utility to write the records that are rejected during the load.

Discard file: Is a file that can be created, if necessary, to store all records that did not satisfy the selection criteria

Files used by SQL*Loader

Loader control file



Database data files
SQL*Loader

Using SQL*Loader

When you invoke SQL*Loader, you can specify parameters that establish session characteristics. Parameters can be entered in any order, optionally separated by commas.

You can specify values for parameters, or in some cases, you can accept the default without entering a value.

SQL*Loader

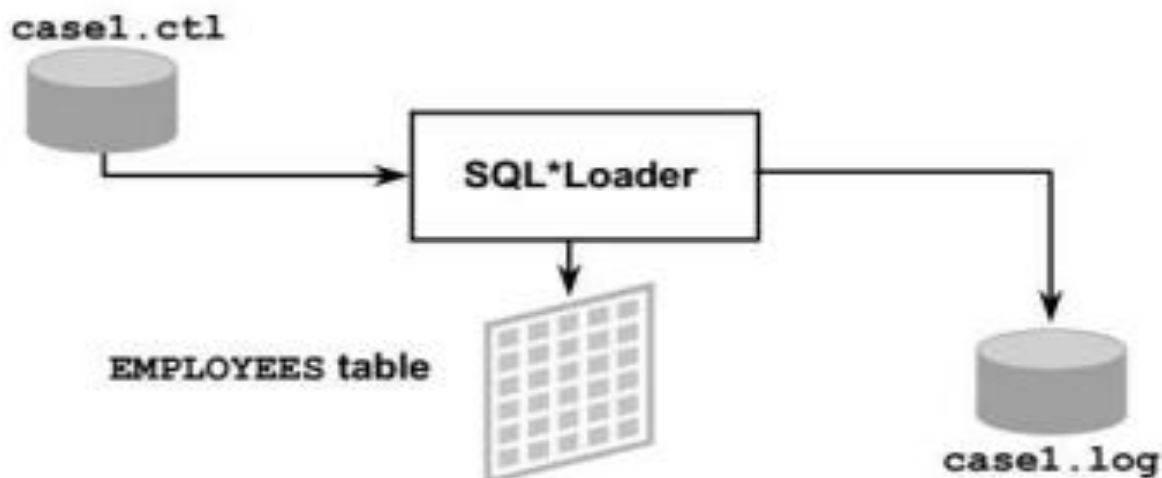
Using SQL*Loader

If you invoke SQL*Loader without specifying any parameters, SQL*Loader displays a Help screen that lists the available parameters and their default values.

SQL*Loader

Using SQL*Loader

```
$sqlldr hr/hr \  
> control=case1.ctl \  
> log=case1.log direct=Y
```



Introduction to Database Administration

TOPIC 188

SQL*Loader Control Files
SQL*Loader Control Files

SQL*Loader Control Files

Introduction

Three Sections of Control Files
Sample Control File
Sample Control File Explanation
SQL*Loader Control Files

Introduction

The loader control file tells SQL*Loader:
Where to find the load data
The data format
Configuration details:
Memory management
Record rejection
Interrupted load handling details
How to manipulate the data
SQL*Loader Control Files

Introduction

The SQL*Loader control file is a text file that contains DDL instructions. DDL is used to control the following aspects of a SQL*Loader session:
Where SQL*Loader finds the data to load
How SQL*Loader expects that data to be formatted
How SQL*Loader configures (memory management, rejecting records, interrupted load handling, and so on) as it loads the data
SQL*Loader Control Files

Introduction

How SQL*Loader manipulates the data being loaded
SQL*Loader Control Files

Three Sections of Control Files

The first section contains session-wide information, for example:
Global options such as bind size, rows, records to skip, and so on
INFILE clauses to specify where the input data is located
How data is to be loaded
SQL*Loader Control Files

Three Sections of Control Files

The second section consists of one or more INTO TABLE blocks. Each of these blocks contain information about the table into which the data is to be loaded, such as the table name and the columns of the table.

The third section is optional and, if present, contains input data.

SQL*Loader Control Files

Sample Control File

```
1  -- This is a sample control file
2  LOAD DATA
3  INFILE 'SAMPLE.DAT'
4  BADFILE 'sample.bad'
5  DISCARDFILE 'sample.dsc'
6  APPEND
7  INTO TABLE emp
8  WHEN (57) = '.'
9  TRAILING NULLCOLS
10 (hiredate SYSDATE,
    deptno POSITION(1:2) INTEGER EXTERNAL(3)
    NULLIF deptno=BLANKS,
    job POSITION(7:14) CHAR TERMINATED BY WHITESPACE
    NULLIF job=BLANKS "UPPER(:job)",
    mgr POSITION(28:31) INTEGER EXTERNAL
    TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
    comm POSITION(34:41) CHAR
```

SQL*Loader Control Files

Sample Control File Explanation

This is how comments are entered in a control file. Comments can appear anywhere in the command section of the file, but they should not appear within the data.

The LOAD DATA statement tells SQL*Loader that this is the beginning of a new data load. If you were continuing a load that had been interrupted in progress, you would use the CONTINUE LOAD DATA statement.

The INFILE keyword specifies the name of a data file containing data that you want to load.
SQL*Loader Control Files

Control File Syntax Consideration

The syntax is free-format Syntax is not case sensitive

Comments extend from the two hyphen (--) that mark the beginning of the comment to the end of the line

The CONSTANT keyword is reserved

SQL*Loader Control Files

Control File Syntax Consideration

The syntax is free format. (Statements can extend over multiple lines.)

It is case insensitive; however, strings enclosed in single or double quotation marks are taken literally, including case.

In control file syntax, comments extend from the two hyphens (—) that mark the beginning of the comment to the end of the line. The optional third section of the control file is interpreted as data rather than as control file syntax; consequently, comments in this section are not supported. The CONSTANT keyword has special meaning to SQL*Loader and is therefore reserved. To avoid potential conflicts, do not use the word CONSTANT as a name for any tables or columns.

Introduction to Database Administration

TOPIC 189

Input & Data Files

Input & Data Files

Input & Data Files

Introduction

Fixed-Record Format

Variable-Record Format

Stream-Record Format Input & Data Files

Introduction

SQL*Loader reads data from one or more files specified in the control file.
From SQL*Loader's perspective, the data in the data file is organized as records.
A data file can be in one of three formats:

Fixed-record format
Variable-record format
Stream-record format
Input & Data Files

Fixed-Record Format

Records in a file are of same byte Format is specified as

INFILE <data file> "fix n"

Following is a control file contents for two
columns

load data

infile 'example.dat' "fix 10"

into table example

fields terminated by ' ' (col1, col2)

Input & Data Files

Fixed-Record Format

example.dat: 0001, abcd

0002, fghi

0003, klmn

Input & Data Files

Variable-Record Format

A file is in variable format when the length of each record in a character field is included at the beginning of each record in the data file.

For example, specifying data file as INFILE "datafile_name" "var n"

n represents the number of bytes in the record length field
If n is not specified, SQL*Loader assumes a length of 6.
Input & Data Files

Variable-Record Format

In example.dat, record length fields/ columns are 3 bytes long

```
load data
infile 'example.dat' "var 3"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5), col2 char(7))
example.dat:
009hello,cd,
010world,im,
012my,name is,
```

Input & Data Files

Stream-Record Format

A file is in stream-record format when records are not specified by size; instead SQL*Loader forms records by scanning for the record terminator. It is flexible but it has performance issues. For example, specifying data file as INFILE <datafile_name> ["str terminator_string"]

Input & Data Files

Stream-Record Format

Non printable characters will be

\n new line

\t horizontal tab

\f form feed

\v vertical tab
\r carriage return
Input & Data Files

Stream-Record Format

In example.dat, data loading with stream-record format

```
load data
infile 'example.dat' "str '|' \n"
into table example
fields terminated by ',' optionally enclosed by '"'
(col1 char(5),
col2 char(7))
example.dat:
hello,world,|
james,bond,|
```

Introduction to Database Administration

TOPIC 190

Logical Records & Loading Methods

Logical Records & Loading Methods

Logical Records & Loading Methods

Logical Records

Using CONCATENATE to Assemble Logical Records

Using CONTINUEIF to Assemble Logical Records

Conventional Path Load

Direct Path Load

Loading Methods

Logical Records & Loading Methods

Logical Records

SQL*Loader can be instructed to follow one of the following two logical record- forming strategies:

Combine a fixed number of physical records to form each logical record

Combine physical records into logical records while a certain condition is true

Logical Records & Loading Methods

Logical Records

SQL*Loader organizes the input data into physical records, according to the specified record format. By default, a physical record is a logical record. However, for added flexibility, SQL*Loader can be instructed to combine a number of physical records into a logical record. SQL*Loader can do

Combine fixed number of physical records to form each logical record

Combine physical records into logical records while a certain condition is true

Logical Records & Loading Methods

Using CONCATENATE to Assemble Logical Records

CONCATENATE is used when SQL*Loader should always add the same number of physical records to form one logical record. For example,

CONCATENATE integer

Means number of physical records to combine

Logical Records & Loading Methods

Using CONTINUEIF to Assemble Logical Records

CONTINUEIF must be used if the number of physical records to be continued varies. The CONTINUEIF keyword is followed by a condition that is evaluated for each physical record as it is read.

For example, two records might be combined if there was a pound sign (#) in character position 80 of the first record.

Logical Records & Loading Methods

Conventional Path Load

Conventional path Load builds an array of rows to be inserted and uses the SQL INSERT statement to load the data. During conventional path loads, input records are parsed based on field specifications, and an array of records is built and inserted into the table specified in the control file.

Records that do not conform to the field specifications are rejected and those records that do not satisfy the selection criteria are discarded.

Logical Records & Loading Methods

Conventional Path Load

Conventional path loads can be used to load data into both the **clustered and unclustered tables**. Redo generation is controlled by the logging attribute for the table that is being loaded.

Logical Records & Loading Methods

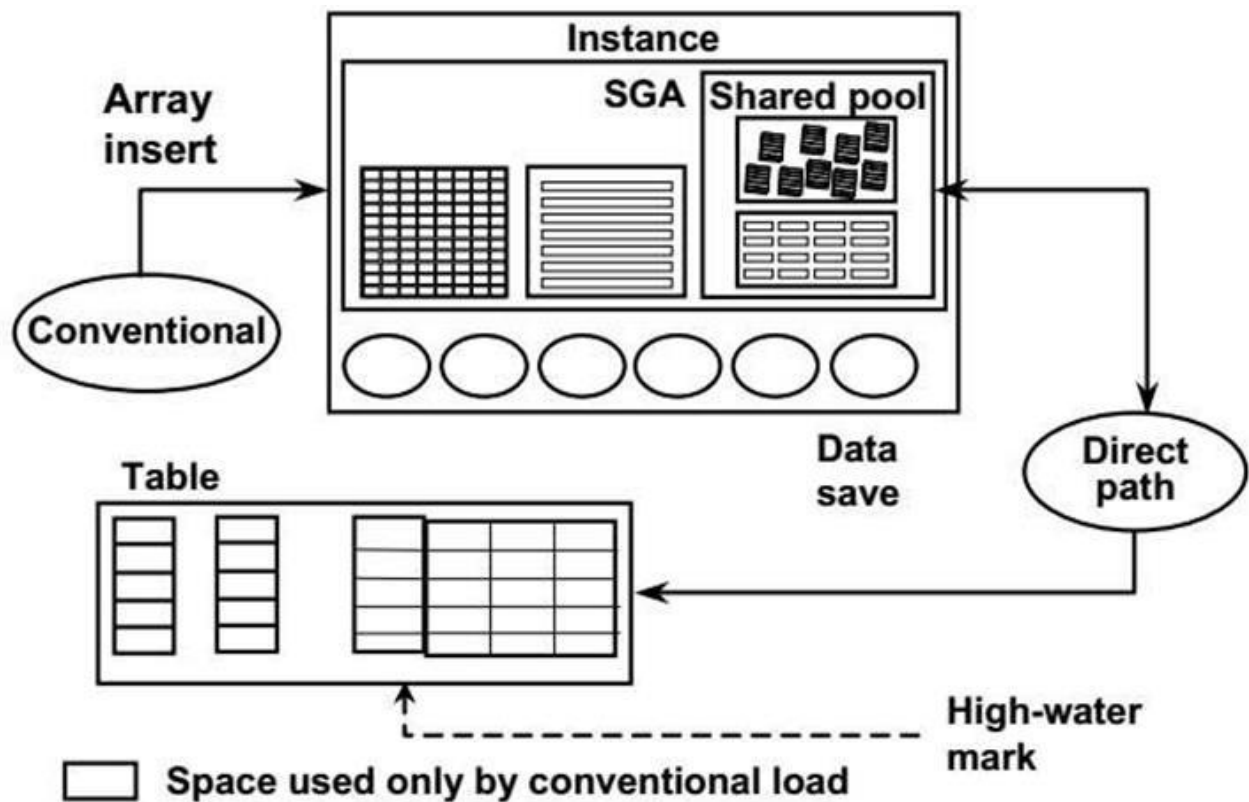
Direct Path Load

A Direct path load builds blocks of data in memory and saves these blocks directly into the extents allocated for the table being loaded. Online redo log file entries are not generated unless the database is in ARCHIVELOG mode.

Direct load path is generally faster than conventional path load.

Logical Records & Loading Methods

Loading Methods



Introduction to Database Administration

TOPIC 191

Direct & Conventional Path Loads

Direct & Conventional Path Loads

Direct & Conventional Path Loads

Method of Saving Data

Comparison of Conventional and Direct Path Loads

Logging Changes

Enforcing Constraints

Parallel Direct Paths

Enforcing Constraints

Direct & Conventional Path Loads

Method of Saving Data

Conventional path loads use SQL processing and database COMMITs for saving data. The insertion of an array of records is followed by a commit operation. Each data load might involve several transactions.

Direct & Conventional Path Loads

Method of Saving Data

Direct path loads use data saves to write blocks of data to Oracle data files. The following features differentiate a data save from a COMMIT:

During a data save, only full database blocks are written to the database. The blocks are written after the high-water mark of the table.

After a data save, the high-water mark is moved. Internal resources are not released after a data save. A data save does not end the transaction.

Indexes are not updated at each data save.

Direct & Conventional Path Loads

Conventional Load	Direct Path Load
Uses COMMITs to make changes permanent	Uses data saves
Redo entries always generated	Generates redo only under specific conditions
Enforces all constraints	Enforces only primary key, unique, and NOT NULL
INSERT triggers fire	INSERT triggers do not fire
Can load into clustered tables	Cannot load into clustered tables
Other users can make changes to tables	Other users cannot make changes to tables

Comparison of Conventional and Direct Path Loads

Direct & Conventional Path Loads

Logging Changes

Conventional path loading generates redo entries just as any DML statement. When using a Direct path load, redo entries are not generated if:

The database is in NOARCHIVELOG mode The database is in ARCHIVELOG mode, but logging is disabled. Logging can be disabled by setting the NOLOGGING attribute for the table or by using the UNRECOVERABLE clause in the control file

Direct & Conventional Path Loads

Enforcing Constraints

During Direct path loads, the constraints are handled as follows:

NOT NULL constraints are checked when arrays are built.

FK and CHECK constraints are disabled

PK and unique constraints are checked during and at the end of the run and may be disabled

Direct & Conventional Path Loads

Parallel Direct Paths

Multiple SQL*Loader sessions improve the performance of a Direct path load. Three models of concurrency can be used to minimize the time required for data loading:

Parallel conventional path loads

Intersegment concurrency with Direct path load method

Intra segment concurrency with Direct path load method

Direct & Conventional Path Loads

Parallel Direct Paths

Concurrent conventional path

When trigger or integrity constraints pose a problem

Intersegment concurrency with Direct path load method

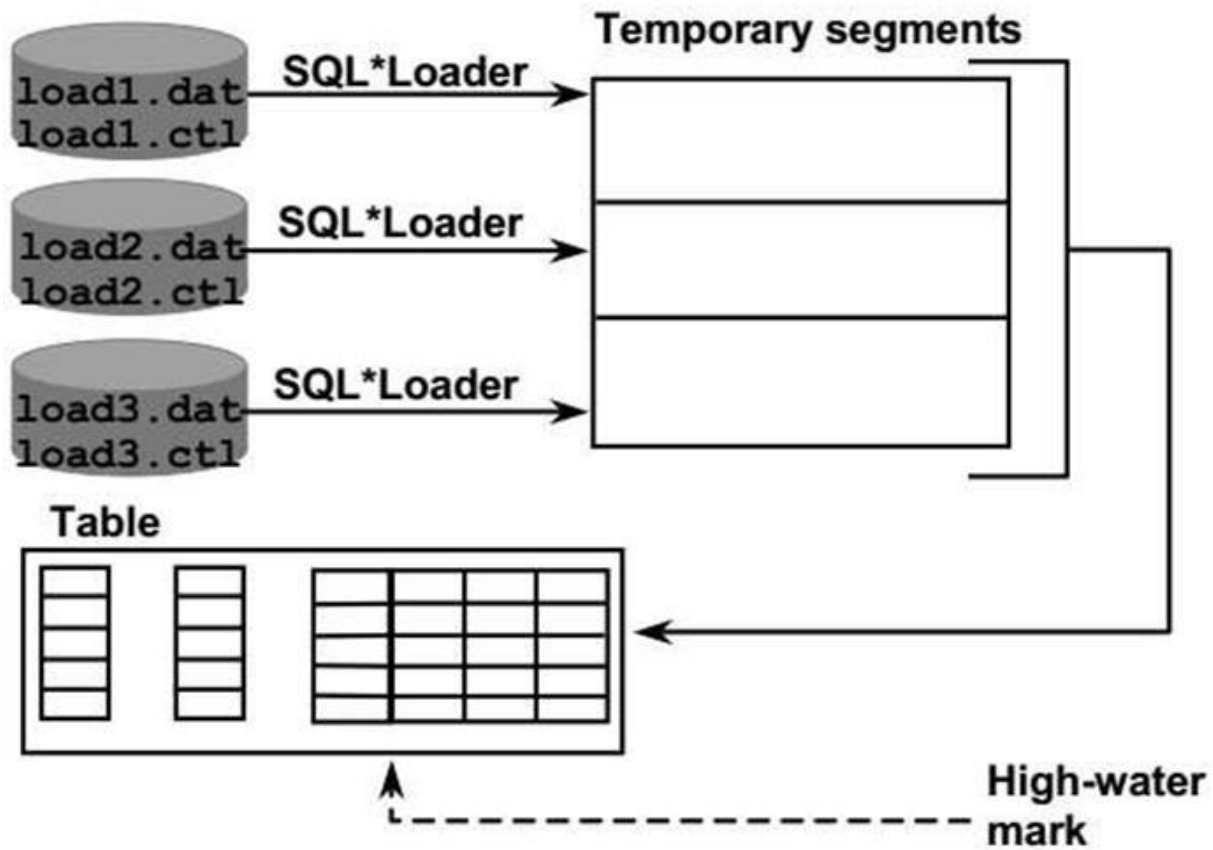
Loading of different objects such as tables and partitions of same table.

Intra segment concurrency with Direct path load method

Multiple Direct path load session concurrently into same table or partition

Direct & Conventional Path Loads

Enforcing Constraints



Introduction to Database Administration

TOPIC 192

Data Conversion

Data Conversion

Data Conversion

Introduction

SQL*Loader Rejects

Discarded or Rejected Records

Data Conversion

Introduction

During a conventional path load, data fields in the data file are converted into columns in the database in two steps:

The field specifications in the control file are used to interpret the format of the data file and convert it to a SQL INSERT statement using that data.

The Oracle database server accepts the data and executes the INSERT statement to store the data in the database

Data Conversion

SQL*Loader Rejects

When input format is invalid

When delimited field exceeds its maximum length

All bad records are saved in bad file

Oracle determines row is valid or not.

If row or record is rejected, SQL*Loader puts it in the bad file

Why row can be rejected?

(unique or not null constraints)

Data Conversion

Discarded or Rejected Records