Database Administration

- Outline / Learning Objectives
  - Overview
  - Duties/Tasks of a DBA
  - Table spaces
  - Data dictionary
  - Privileges
  - Roles (Groups)

Overview

- *Database administration* is a specialized area within a large information systems department that is separate from the application development area.
- *Application development* includes systems analysis, systems design, programming, and systems testing.
- Database administration is concerned with administrative tasks that must be accomplished in order for application developers and information system users to access an organization's databases.
- ***A database administrator (DBA) is neither superior to nor inferior to an application developer.***

Overview

- DBAs and application developers must be both *technically competent* and capable of working closely with other professionals in a support type of relationship.
- One of the primary roles of a DBA is to provide the *support* needed by an application developer so that the application developer can accomplish the task of building and maintaining information systems.

Duties/Tasks of a DBA

- *Install* relational database management system software and upgrades.
- *Design and create* a database including the allocation of system disk storage for current and future database storage requirements.
- *Start up and shut down* a database.
- Create *user accounts* and monitor user activities.
- Grant *database privileges* to control data security and data access.

Duties/Tasks of a DBA

- *Backup and recover* a database in the event of system failure.
- Tune a database to *optimize database performance*.
- Manage database *network connectivity*.
- Migrate a database to a *new version* of the DBMS software.

Tablespaces and Files

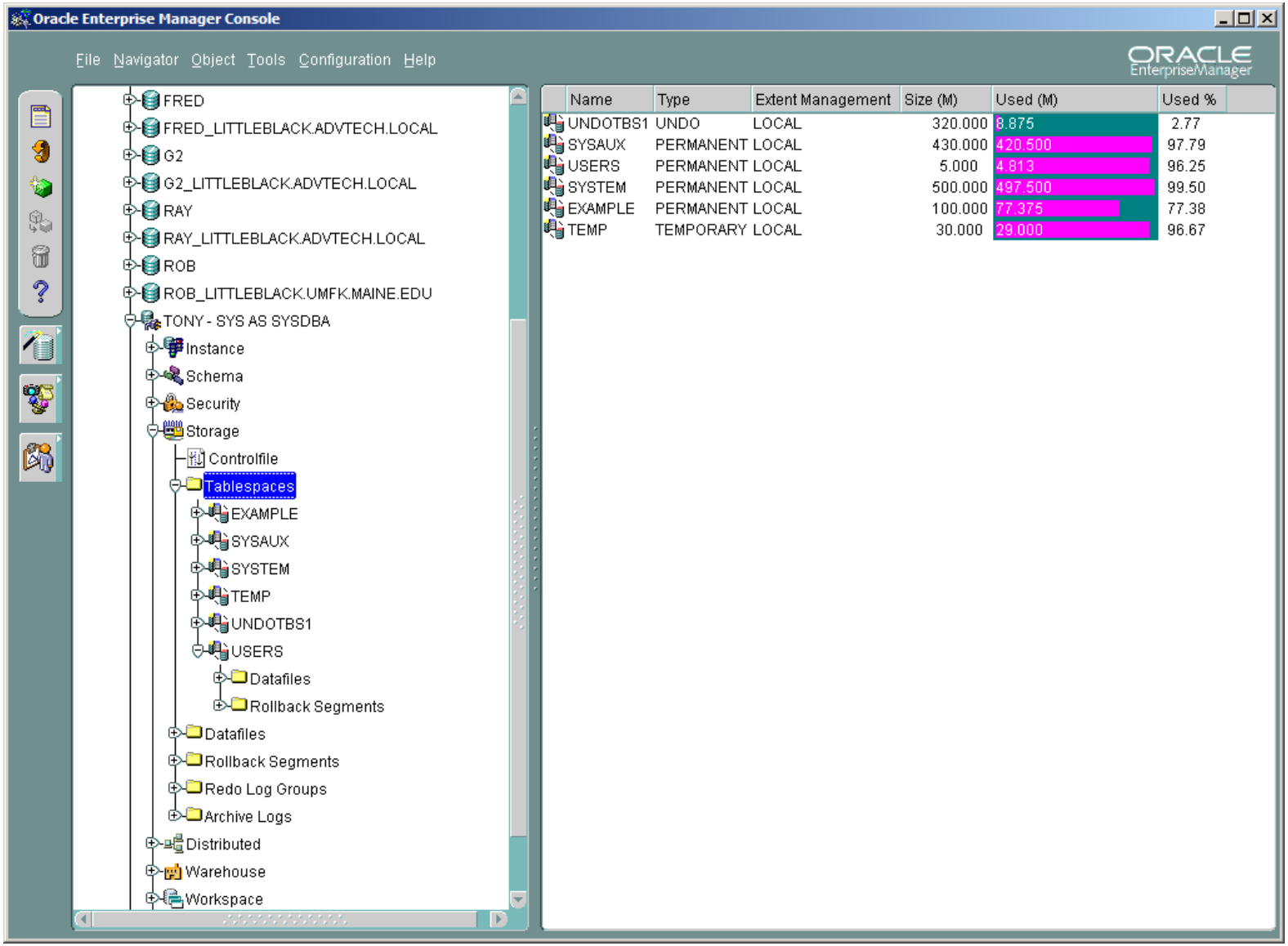
- An Oracle database is normally divided into logical components referred to as *table spaces*.
- Most table spaces are created by a DBA when a database is initially created.
- As the term implies, a tablespace is used to store tables.
- But, table spaces are actually used to store all types of database objects.
- These objects include indexes, sequences, procedures, views, and other database objects.

Tablespaces and Files

- A typical Oracle database will have most of the tablespaces listed here.
  - SYSTEM
  - USER
  - DATA
  - INDEXES
  - TEMP
  - UNDO
  - IDS
  - SPECIAL_APPS

Tablespaces and Files

- Tablespaces are *logical*, not *physical* objects.
- Tablespaces are actually stored in *data files*.
- Data files are the corresponding physical objects.
- A very large tablespace may require more than one data file in order to store all of its objects.
- A data file can only store data for a single tablespace.
- Tablespaces are created with the CREATE TABLESPACE command.


Oracle Enterprise Manager Console
File Navigator Object Tools Configuration Help
ORACLE EnterpriseManager
FRED
FRED_LITTLEBLACK.ADVTECH.LOCAL
G2
G2_LITTLEBLACK.ADVTECH.LOCAL
RAY
RAY_LITTLEBLACK.ADVTECH.LOCAL
ROB
ROB_LITTLEBLACK.UMFK.MAINE.EDU
TONY - SYS AS SYSDBA
Instance
Schema
Security
Storage
Controlfile
Tablespaces
EXAMPLE
SYSAUX
SYSTEM
TEMP
UNDOTBS1
USERS
Datafiles
Rollback Segments
Datafiles
Rollback Segments
Redo Log Groups
Archive Logs
Distributed
Warehouse
Workspace
Name Type Extent Management Size (M) Used (M) Used %
UNDOTBS1 UNDO LOCAL 320.000 8.875 2.77
SYSAUX PERMANENT LOCAL 430.000 420.500 97.79
USERS PERMANENT LOCAL 5.000 4.813 96.25
SYSTEM PERMANENT LOCAL 500.000 497.500 99.50
EXAMPLE PERMANENT LOCAL 100.000 77.375 77.38
TEMP TEMPORARY LOCAL 30.000 29.000 96.67

Data Dictionary

- The Oracle *data dictionary* consists of *read-only, base tables* that store information about the database.
- The data dictionary is stored in the *system* tablespace.
- Information stored in the data dictionary is termed *metadata*; that is, data about data.
- The read-only, base tables that comprise the data dictionary are rarely ever accessed by system developers or system users.
- Usually only the various Oracle processes will access these tables.

Data Dictionary

- In order to make it easier to access information and manage a database, the tables of the data dictionary are organized into various *user-accessible views*.
- These views are also part of the data dictionary.

Data Dictionary

- Some of the information stored in the data dictionary includes:
  - all schema object definitions – the definitions of the tables, indexes, sequences, views, and other database objects.
  - the amount of space allocated for each object, and the amount of space currently used by each object.
  - the names of the Oracle user accounts, and the privileges and roles granted to each account.
  - information needed to enforce integrity constraints.
  - other database information.

Types of Views – USER, ALL, and DBA

- There are many different types of data dictionary views.
- In order to assist system users in using the data dictionary, the views are divided into three different categories: USER, ALL, and DBA.
  - USER – The *user* prefix is added to all views that display information about objects that belong to an individual user. We term this the user's schema of the database.
  - ALL – The *all* prefix is added to all views that display information about all objects in the global database schema. This expands on an individual user's perspective of the database.
  - DBA – The *dba* prefix is added to all views that fall within the database administrator's schema of the database.

User Accounts and Privileges

- Individual accounts, including those that belong to DBAs must be created.
- When a database is initially created, a DBA will logon to the database as the user SYS or SYSTEM, and create a personal DBA account.
- The DBA account will be granted all of the privileges needed to administer the database.

Creating, Altering, and Dropping User Accounts

- The DBA creates individual accounts for each system user.
- A simple form of the CREATE USER command is shown here:

```
CREATE USER bock IDENTIFIED BY secret_password;
```

- A DBA or other system administrator must have the CREATE USER system privilege in order to create a user account.
- Even though *bock* now has an account, *bock* still cannot connect to the database until the DBA grants *bock* the CREATE SESSION system privilege.

Creating, Altering, and Dropping User Accounts

- The following SQL Example gives a more complete form of the CREATE USER command.
- This CREATE USER command creates a user account and also allocates space for the storage of objects in various tablespaces.
- This command will only execute if your database has the tablespaces named here.

```
CREATE USER bock IDENTIFIED BY secret_password
    DEFAULT TABLESPACE users
    TEMPORARY TABLESPACE temp
    QUOTA 10M ON users
    QUOTA UNLIMITED ON temp
    QUOTA 5M ON data
    PASSWORD EXPIRE;
```

Altering User Accounts

- Suppose that we suspect *bock* is doing something with the database that is unauthorized! We can stop *bock* from creating additional objects by altering the *bock* account.
- The commands shown in the following SQL Example will alter *bock's* quota on the *users* and *data* tablespaces.
- Existing objects *bock* has created will not be modified, but he will be unable to create new objects.

```
ALTER USER bock QUOTA 0 ON users;
ALTER USER bock QUOTA 0 ON data;
```

- You can also use the ALTER USER command to allocate additional space and to restore quota allocations.

Dropping User Accounts

- The DROP USER command will delete a user account.
- It is necessary to use the CASCADE option if the user has created any objects.

```
DROP USER bock;
DROP USER bock CASCADE;
```

- If you fail to specify CASCADE, then the DROP USER command will fail if the user has created objects.
- We need to EXERCISE CAUTION while dropping user accounts!
- What if *bock* is a system developer who has created some critical objects such as tables that our organization uses to store inventory and payroll information?

Dropping User Accounts

- If we drop *bock* with a CASCADE, the applications will fail because the tables and indexes that *bock* created will be destroyed.
- A better alternative is to lock *bock* out of his account because his employment has terminated.
- We can do this by revoking his privilege to connect to the system.

Granting and Revoking Privileges

- The GRANT command in the following SQL Example will grant user *bock* the CREATE SESSION privilege need to logon to the SQL*PLUS or any of the Oracle's tools.

```
GRANT PRIVILEGE create session TO bock;
```

- There are many different privileges.
- They are divided into two categories: *system privileges* and *object privileges*.
- System privileges allow a system user to perform specific types of operations such as creating, dropping, and altering objects.
- Object privileges allow a system user to perform a specific operation on a specific object such as a view, table, or index.

System Privileges

- System privileges are also termed system-wide privileges, and include privileges that focus on managing objects that you own, and privileges that focus on managing objects that any system user may own.
- Here are example system privileges:

  CREATE SESSION; ALTER SESSION

  CREATE TABLE

  CREATE ANY TABLE; ALTER ANY TABLE

  CREATE ANY INDEX

  UNLIMITED TABLESPACE

System Privileges

- Basically, if you can create an object, you can also drop the object.
- The CREATE TABLE privilege also includes the privilege to create indexes for a table and to subsequently drop those indexes as necessary.
- Some example GRANT commands are shown here.
- Note that the privileges being granted are separated by a comma as are the user account names:

```
GRANT create session TO bock, bordoloi, user001;
GRANT create table, unlimited tablespace TO bock;
GRANT create table, create any table TO bock, bordoloi;
```

System Privileges

- Once a DBA has created a user account, a system privilege may be granted to the user account with the WITH ADMIN OPTION.
- In fact, this option may be used in any GRANT command that grants a system privilege.
- The WITH ADMIN OPTION enables the grantee (the account receiving the privilege) to, in turn, grant the privilege to other user accounts.
- Following SQL Example grants CREATE SESSION and CREATE TABLE to *bock*, and also gives *bock* the authority to grant these two privileges to other system user accounts.

```
GRANT create session, create table TO bock WITH ADMIN
OPTION;
```

Object Privileges

- Object privileges work on a specific object, so the form of the GRANT command is a bit different. The general form is:

GRANT privilege1, privilege2, . . .
ON object_name TO user1, user2, . . . | public;

- For example, if *bock* has created a table named *employee* and wants to give the user named *bordoloi* the privilege to select rows from the table, the GRANT command in the following example enables *bordoloi* to select from the table.

GRANT select ON bock.employee TO bordoloi;

- Suppose that *bock* also wants *bordoloi* to be able to insert rows into the *employee* table so as to *relieve* bock of some of the workload associated with loading new data.

- The revised GRANT command in the following SQL Example includes the INSERT privilege.

GRANT select, insert ON bock.employee TO bordoloi;

Revoking Privileges

- System privileges may be revoked with the REVOKE command.
- You can only revoke a privilege that was specifically granted previously with a GRANT command.
- There are no cascading effects of revoking privileges.
- This means that if DBA1 grants system privileges WITH ADMIN OPTION to DBA2, and then DBA1 subsequently changes jobs and is no longer authorized DBA privileges, all privileges granted to DBA1 can be revoked without worrying about privileges that DBA1 might have granted to DBA2 or to any system user account, for that matter.
- Following SQL Example shows the revocation of the SELECT ANY TABLE privilege from the user account *bordoloi*.

```
REVOKE select any table FROM bordoloi;
```

Revoking Privileges

- Object privileges are revoked in a similar fashion, except that the object for which privileges are revoked must be named in the REVOKE command.
- The keyword ALL can be used to revoke all privileges for an object from a user account.

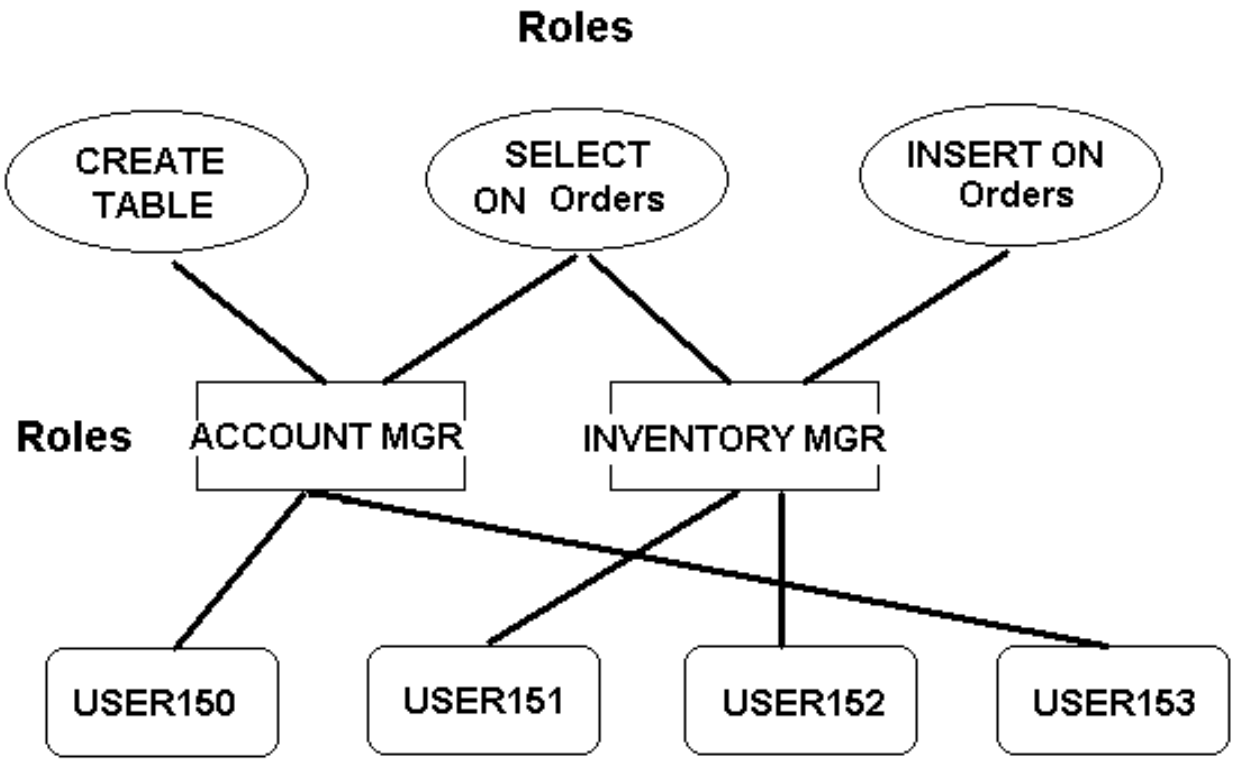
```
REVOKE select ON bock.employee FROM bordoloi;
REVOKE ALL ON bock.employee FROM bordoloi;
```

Roles

- The concept of a *role* is a simple one, and its purpose is to make it easier for a DBA to manage privileges.
- A role is like a container of a group of privileges for a specific type of system user, such as an inventory manager.
- Each time we hire an inventory manager, we would assign the manager a new system user account and authorize that account all of the privileges contained in the role called *inventory_mgr*.
- Further, we can simplify the management of privileges because we can allocate a role to another role!

Roles

- The following figure depicts privileges being allocated to roles, and the roles being allocated to system users.



Roles

- From studying the figure, it should be obvious that if you add a new system user who works as an Account Manager, then you can allocate almost all of the privileges this user will need by simply allocating the role named *account_mgr* to the system user.
- Further, if it is determined that all account managers need an additional privilege, such as DELETE ON ORDERS, that privilege can be granted to the *account_mgr* role, and all of the system users who are granted the role named *account_mgr* will inherit the new privilege! This considerably simplifies user account management.

Roles

- A role name must be unique within the database.
- A role can be allocated both system and object privileges.
- Roles are not owned by anyone so they do not appear in any user account schema.
- You allocate privileges to roles the same way that you allocate them to a system user account; however, you must first create the role by using the CREATE ROLE command.
- The following example commands create the role named *inventory_mgr* and grant several privileges to the role.
- Next the role is allocated to the system user account for *bordoloi*.
- The system user *bock* is granted the *inventory_mgr* role with the privilege to grant the role to other system users through the WITH ADMIN OPTION.

Roles

```
CREATE ROLE inventory_mgr;
GRANT select ON bock.employee TO inventory_mgr;
GRANT select, unlimited tablespace TO inventory_mgr;
GRANT inventory_mgr TO bordoloi;
GRANT inventory_mgr TO bock WITH ADMIN OPTION;
```

Roles

- When a role is dropped, Oracle revokes the role and all privileges granted through the role from all system users and from other roles.
- If you wish to alter a role, the best approach is to create a new role with the desired privileges, grant that role to the system users/roles that possess the role to be dropped; then, you can drop the first role.
- If you were granted a role with the ADMIN OPTION, then you can drop the role.
- You may also drop a role if you have the DROP ANY ROLE system privilege.
- The DROP ROLE command is very simple.
- Following SQL Example shows the command used to drop the *account_mgr* role.

```
DROP ROLE account_mgr;
```

Summary (Learning Outcomes and things to remember)

- DBA and DB developers: level of competence
- Table spaces (SYSTEM, USER, …) and data dictionary
- DB objects: tables, table spaces, views, indexes, procedures, users, …
- CREATE USER <user_name> …
- GRANT PRIVILEGE
  - GRANT PRIVILEGE create session TO <user>;
  - GRANT PRIVILEGE create table TO <user>;
  - Object privileges:
    - GRANT select ON <table> TO <user>;
- Roles
  - CREATE ROLE manager;
  - GRANT select on <table> TO manager;