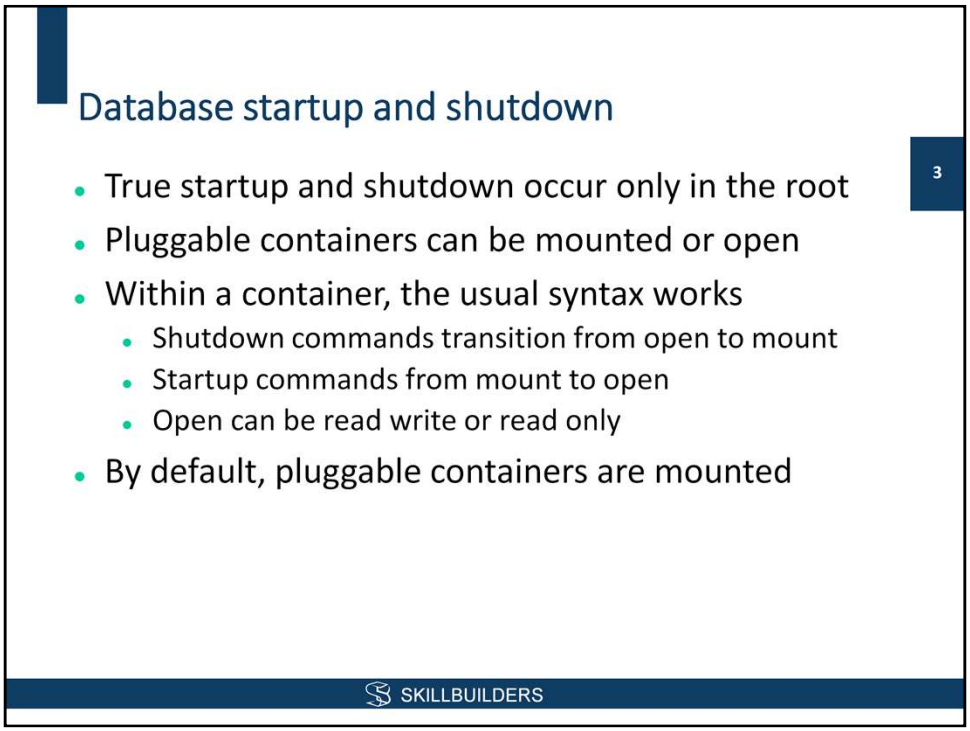
Lesson 4
Multitenant Administration

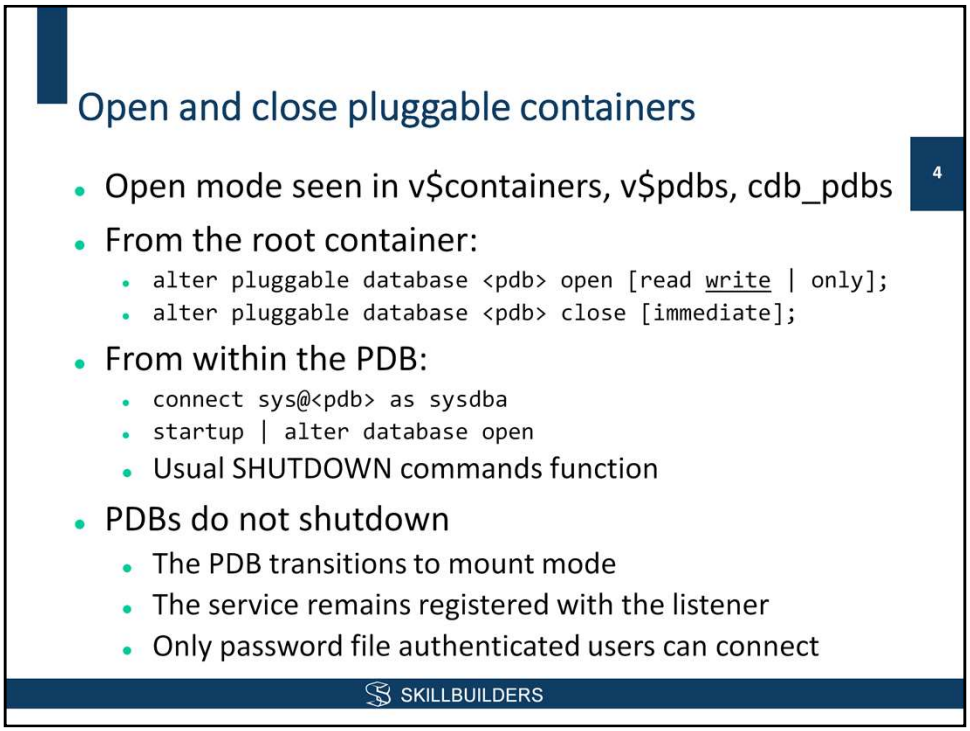
SKILLBUILDERS

Objectives

- Startup and shutdown a CDB
- Open and close pluggable containers
- Tablespace management
- Create local and common objects
- Understand use of roles and privileges
- Services in a multitenant environment
- Set instance parameters

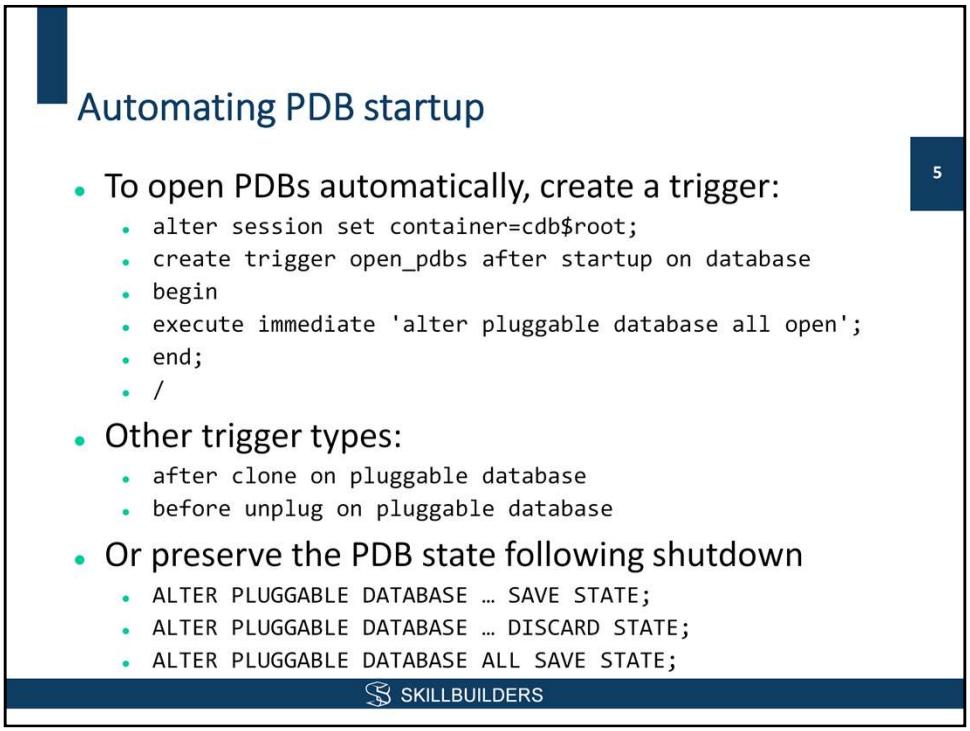


As there is only one database and one instance and one controlfile, it is clear that startup of the instance and mounting the controlfile can be done only from the root container. The pluggable containers can then be opened and closed, as long as the root itself is open.



Closing a PDB is similar to taking its tablspaces offline. The datafiles are checkpointed, and V$DATAFILE_HEADER reports that they are not fuzzy. They do however remain online.

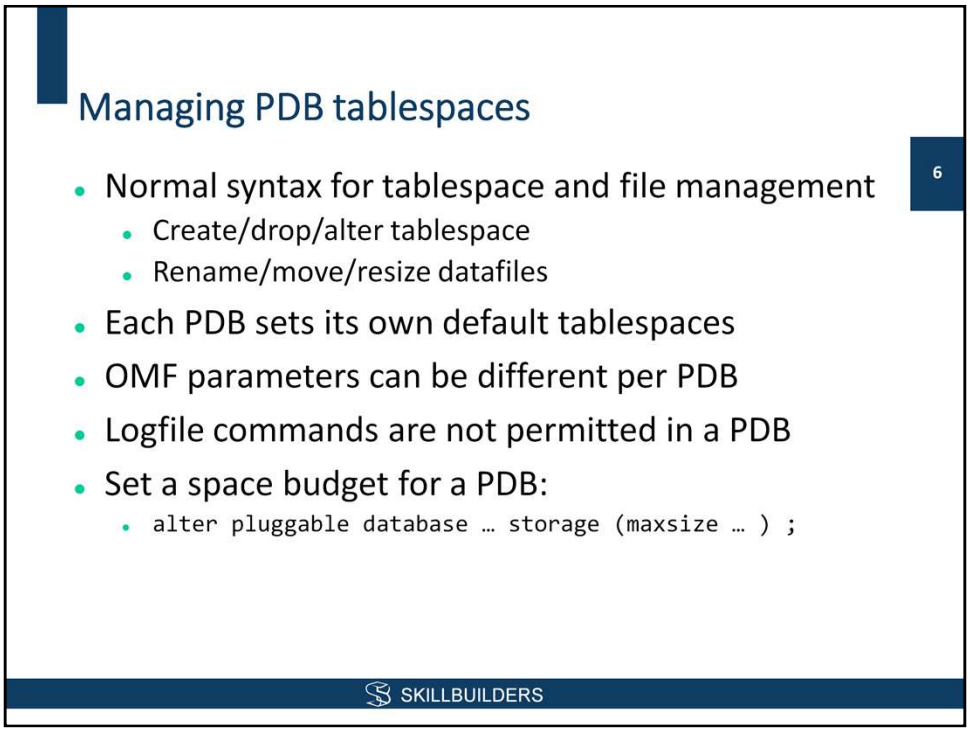
If a non-CDB is mounted, none of its services will be registered with the database listener. In a CDB, the default service for each PDB is always registered as long as the root container is open; there is no need to register PDBs statically in the listener.ora file. This permits a SYSDBA connection over SQL*Net which can be used to open the container, without the need to log on to the root container.



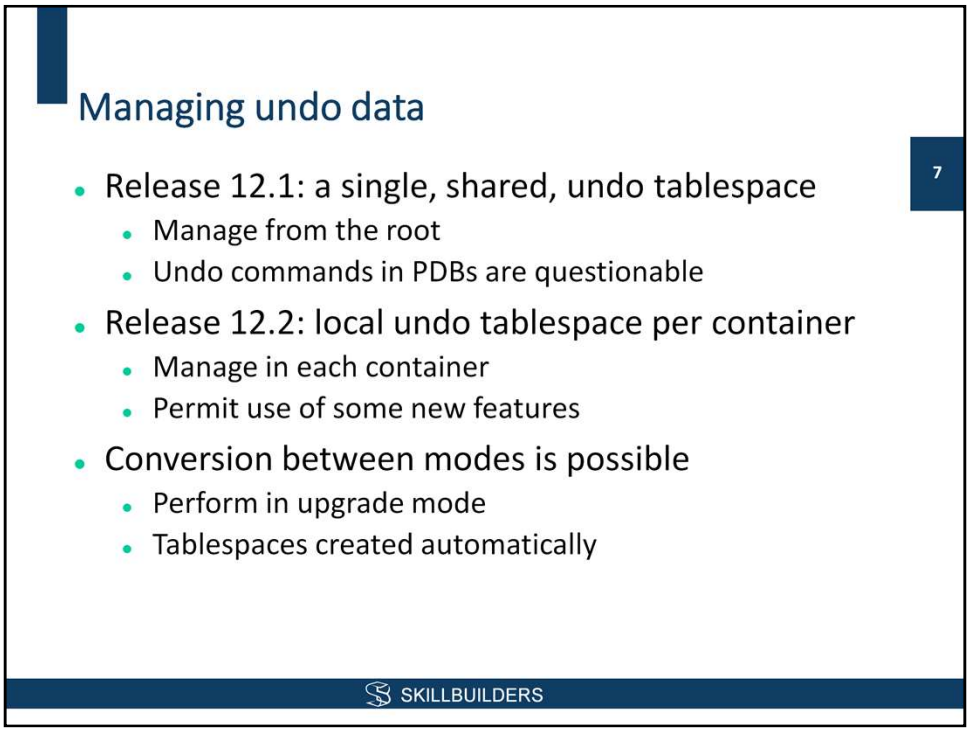
By default, pluggable containers will be mounted when a CDB is opened. The SAVE STATE command instructs the CDB to open or mount the container on startup depending on its current state. The saved state is visible in CDB_PDB_SAVED_STATES. The triggers give more precise control.

The AFTER CLONE and BEFORE UNPLUG triggers should be created in PDBs. They fire once and then self-delete. Usage cases could include configuring external references such as database links, or controlling the availability of the job system, or setting user passwords. A simple example:

```
create trigger post_clone after clone on pluggable database
  begin
  execute immediate 'alter system set job_queue_processes=0';
  end;
  /
```



One issue that may arise in a large environment is controlling the use of space by individual PDBs. It is likely that different PDBs will have different administrators, and that they should reside on different disc groups or filesystems. Setting a space budget will prevent one DBA from inadvertently taking up excessive storage.

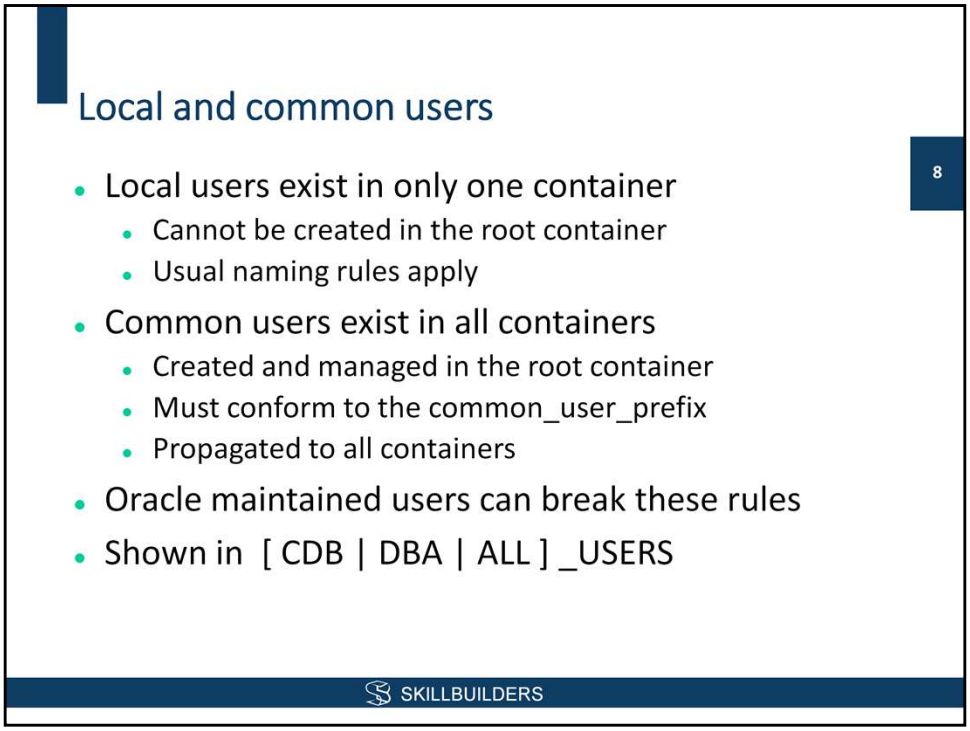


Operating in shared undo mode simplifies administration and in a single tenant environment may be the best option. In a multi-tenancy database, local undo mode offers some advantages. For example, it is impossible for excessive undo data generated in one container (perhaps by badly structured transactions) to cause problems in other containers. Flashback of one container also becomes possible without the need for creating an auxiliary database, as is hot cloning of a container.

To convert between modes:

```
shutdown immediate;
startup upgrade;
alter database local undo ON | OFF ;
shutdown immediate;
startup;
```

An undo tablespace is created in the seed container, copied to all other containers when they open.



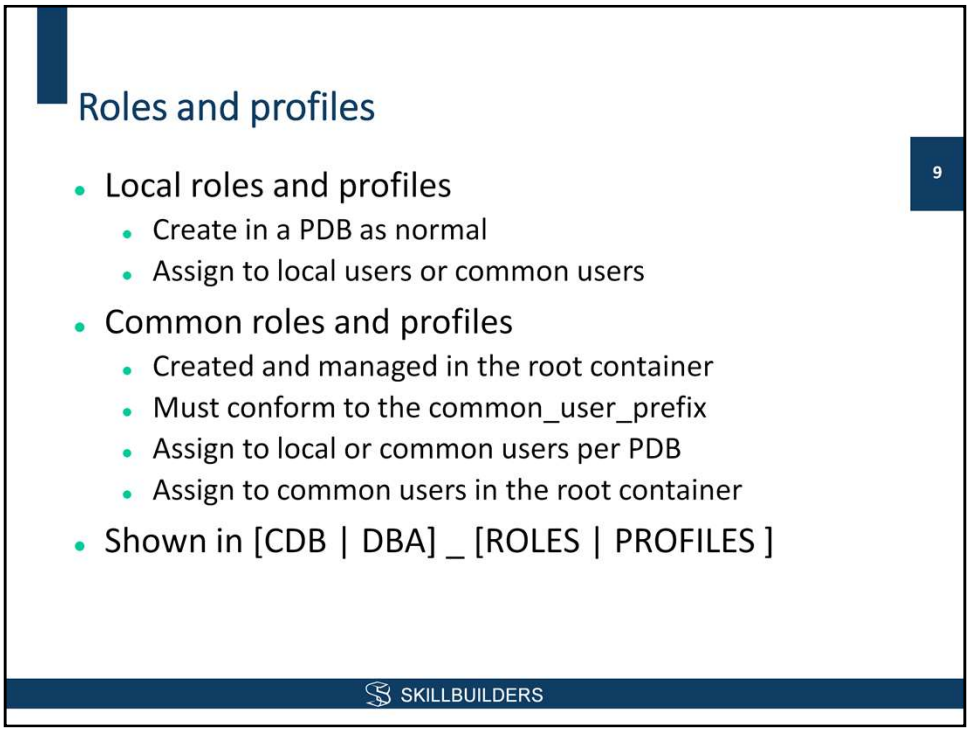
Within a pluggable container, user and schema management is exactly as in a non-CDB. These users are "local" users defined within the PDB's data dictionary. The username must be unique within the container, but can exist independently in multiple pluggable containers.

Users created in the root container are "common" users. Common users are automatically propagated into all PDBs that are open read write, and into closed or read only PDBs when they next open read write.

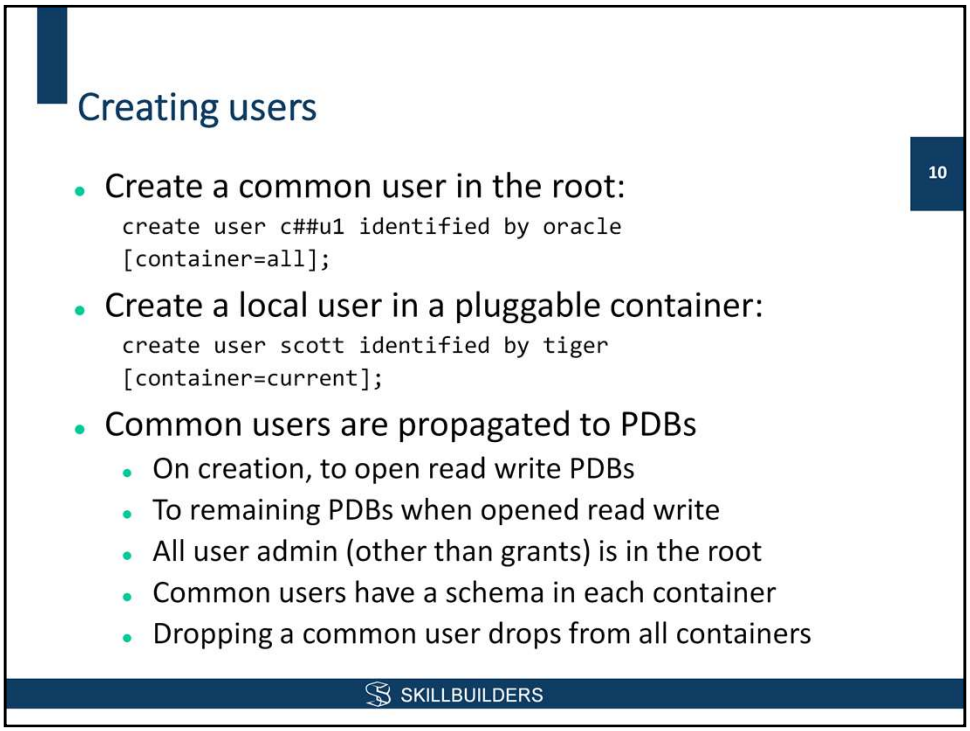
The common_user_prefix parameter specifies a prefix that cannot be used for local users and must be used for common users. The default is C##; changing this is likely to cause confusion. Oracle supplied common users (such as SYSTEM, or CTXSYS) do not conform to the prefix.

All administration of common users (such as password changes) is performed in the root container, though privileges can be granted in pluggable containers.

See the %_USERS views to determine whether a user is common or local, and Oracle maintained or user created.

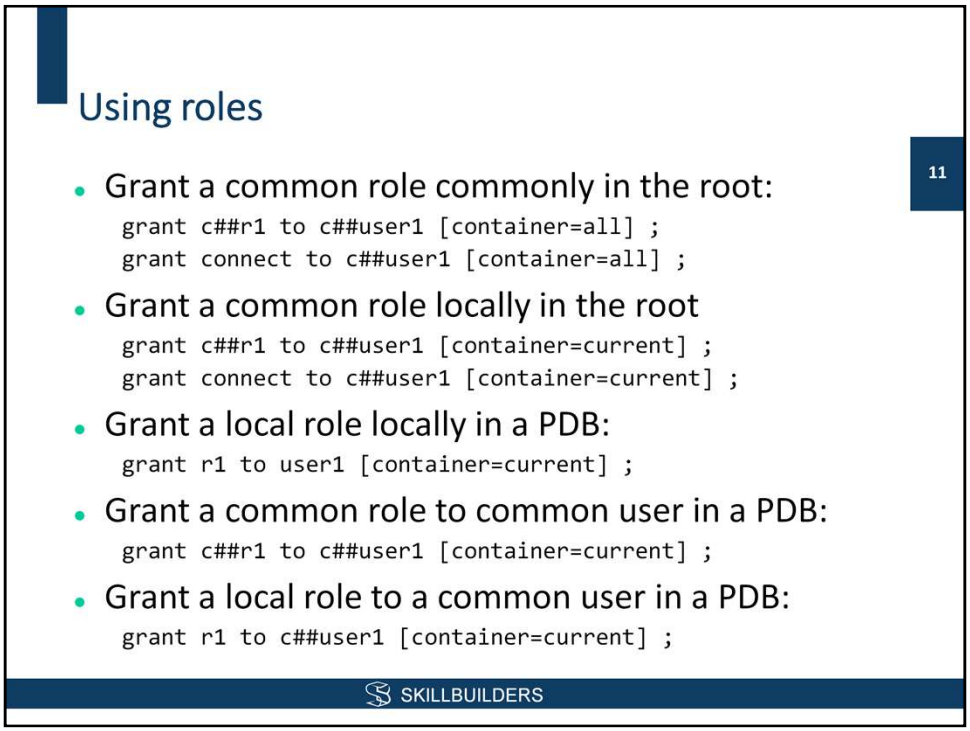


Along with users, roles and profiles are objects that may be common or local. Local roles and profiles are visible only within the container where they were created and are managed in the usual manner. Common roles and profiles are created in the root (named with the prefix) and propagated to all containers.

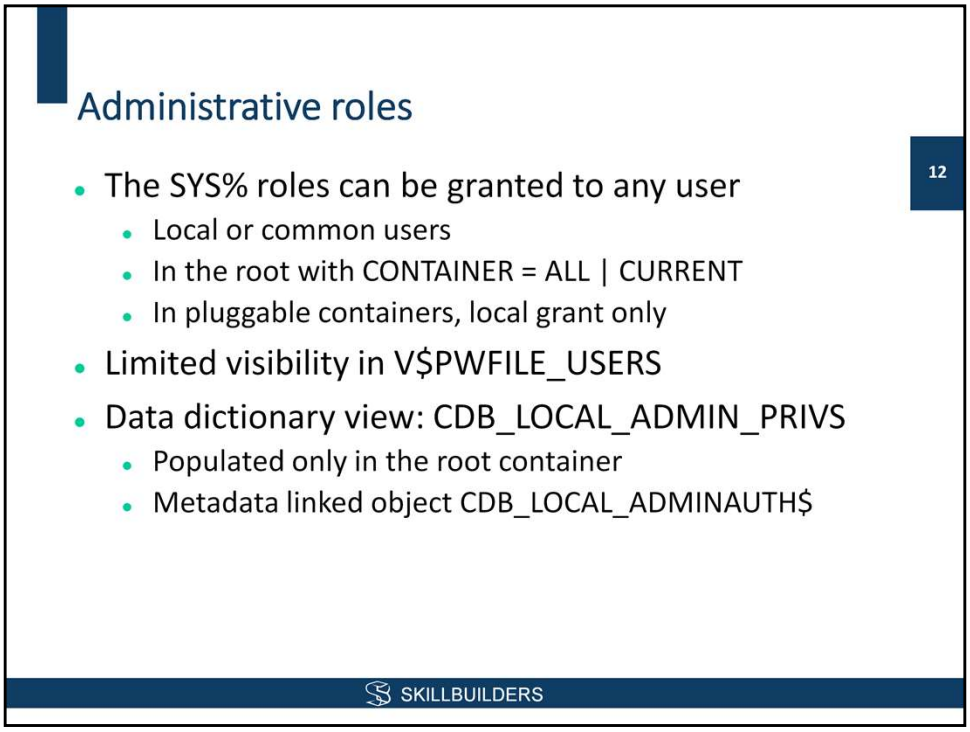


The CREATE USER command has a CONTAINER= clause. When in the root container, the default (and only permitted) value is ALL; when in a pluggable container, the default (and only permitted) value is CURRENT.

Even though a user may be common, his schema is not. The schema is managed individually in each container, and objects are private to that container. To create objects, the common user must be granted appropriate privileges and tablespace quotas in whatever container the objects will exist.

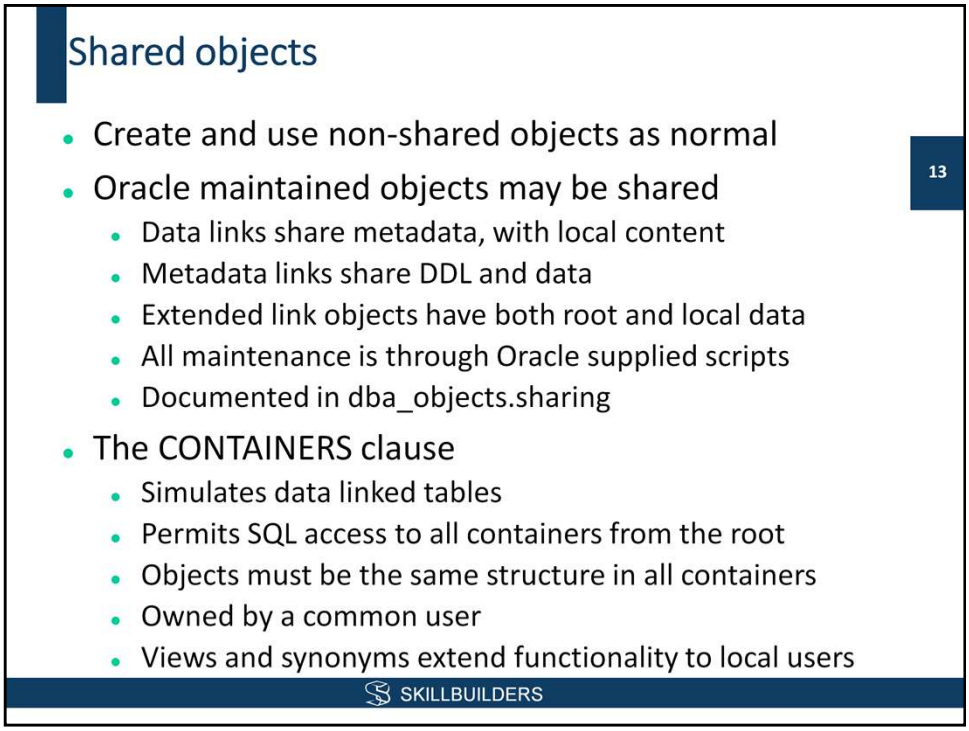


Within a PDB, roles (common or local) can be granted to common or local users. The scope of the grant is that container, and container=current is default, and the only permitted syntax. In the root container (where all users are common) grants can be local (the default) or global according to the container= clause.



The administrative privileges can be granted locally in any container or globally in the root. The password file itself is updated only when the grant is in the root. Grants within a PDB are written to the data dictionary.

When connected with one of these privileges, most privileged commands will run as normal syntactically though the effect when run in a pluggable container may be different from when run in the root container.

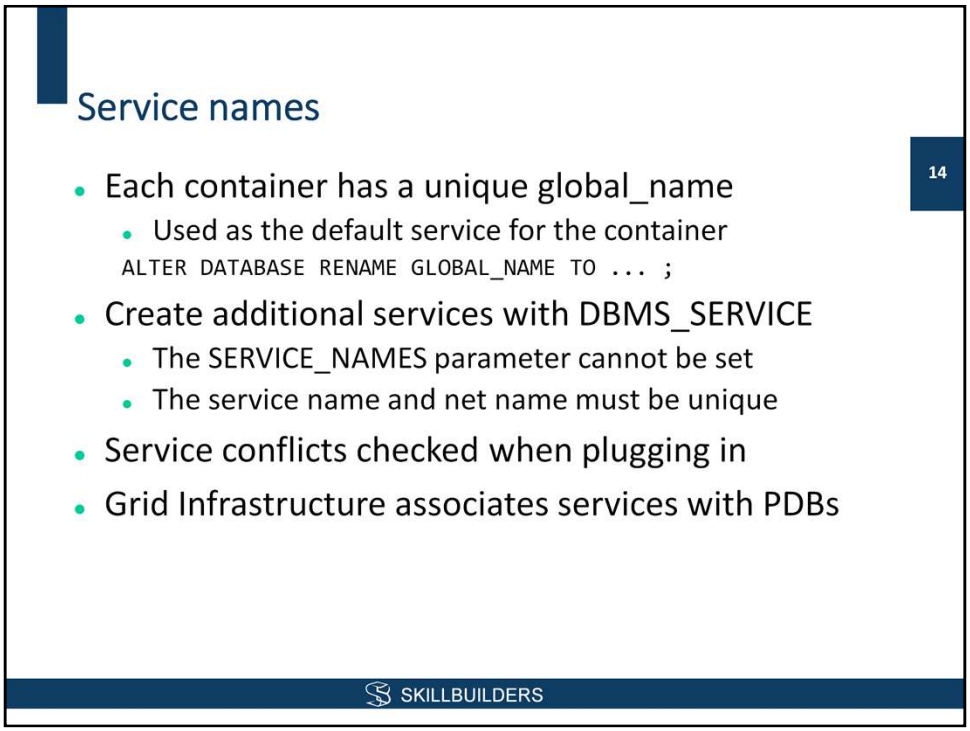


Extended metadata links have a shared root segment plus local segments.

If the same structured table, view, or synonym exists in a common user's schema in all containers, from the root SELECT and DML can be executed against it with the CONTAINERS clause. The object will have a CON_ID column implicitly appended which identifies from which container any row comes:

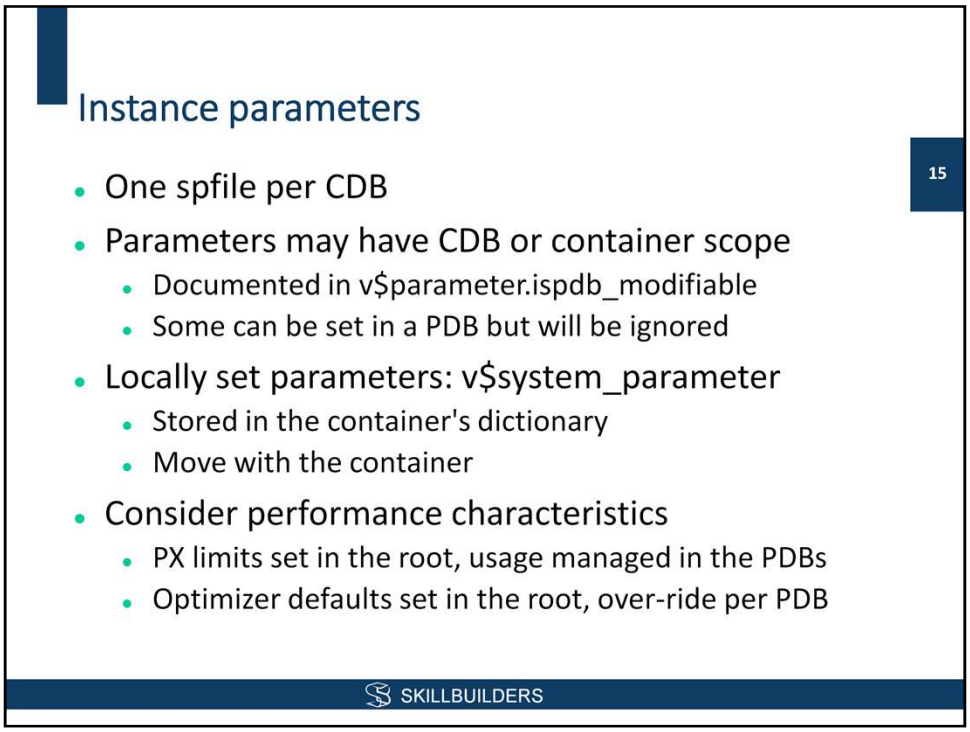
```
cdba> sho con_name
CON_NAME
------------------------------
CDB$ROOT
cdba> desc c##10.t1
 Name                                      Null?    Type
 ----------------------------------------- -------- --------
 C1                                                 VARCHAR2(10)

cdba> select * from containers(c##10.t1) where con_id in (1,4);
C1             CON_ID
---------- ----------
root                1
jw2                 4
```

```
cdba>
```

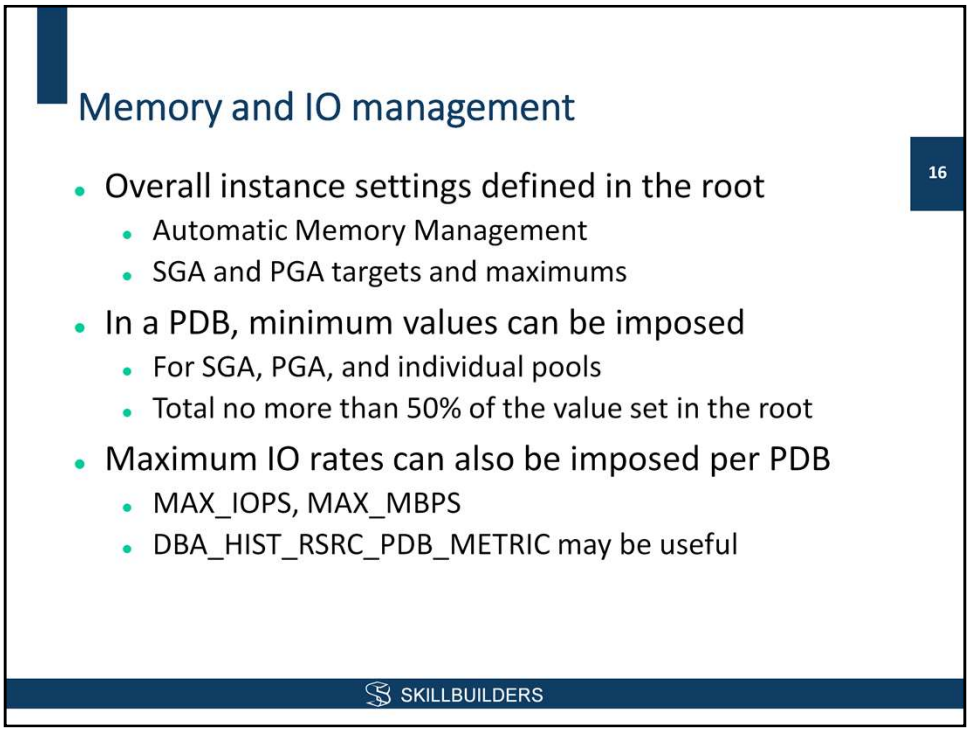


Use of services is essential in a multitenant environment, and slightly different from a non-CDB in that the SERVICE_NAMES parameter cannot be set. This should not be an issue, as more sophisticated techniques (the DBMS_SERVICE package and Grid Infrastructure) have been available for many years to manage services.

DBMS_SERVICE.CREATE_SERVICE will define the service in the data dictionary of the PDB where it is run. The srvctl add service command includes the switch –pdb to specify which PDB it is a part of.



Usually it is obvious whether a parameter is PDB modifiable, though there are some anomalies. As a general rule, anything to do with the shared structures will not be modifiable; anything that has a purely local scope should be.



The Resource Manager is multitenant aware, and will be discussed later.

From release 12.2 it is possible to limit the impact of activity in one pluggable container on another by setting parameters to control memory usage. As in non-CDB, the parameters for individual structures (such as DB_CACHE_SIZE) determine minimums, and SGA_TARGET and PGA_AGGREGATE_TARGET specify maximums. SGA_MIN_SIZE specifies a minimum. If set, these parameters will control how much space in the buffer cache is occupied by blocks belonging to the datafiles of each container and how much library cache space is occupied by code running within it. In effect, the LRU algorithms are modified to give each container virtual caches with the instance's SGA.

The v$cache or v$bh views can be aggregated to see how much cache memory is actually being occupied by objects from a particular container at any given moment.

Multitenant admin: summary

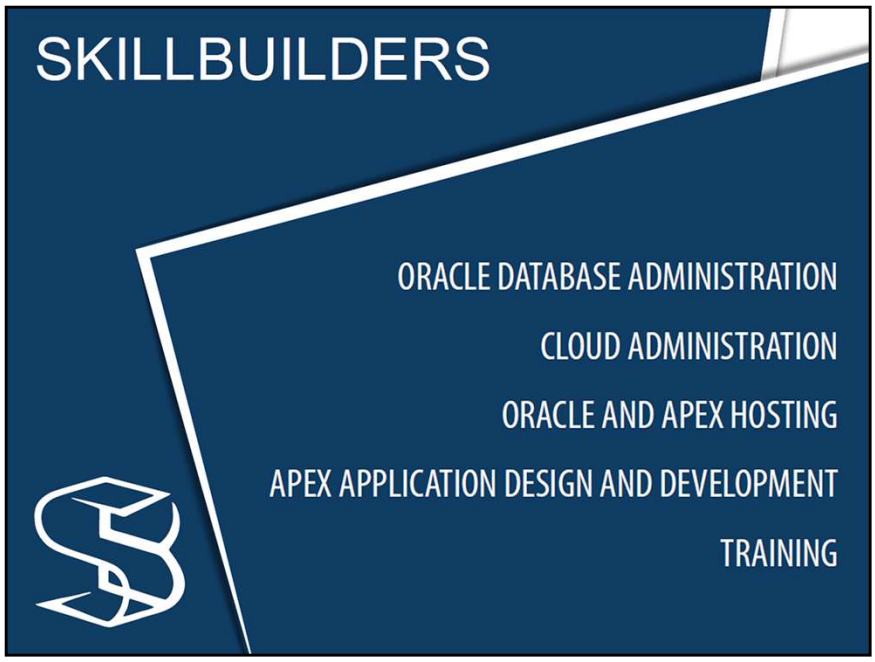
- Within a PDB, most admin is unchanged
- Startup/shutdown is always CDB level
- Individual PDBs can be opened and closed
- Users and roles may be local or common
- Services manage connecting sessions to PDBs

Exercise

- Open and close PDBs
- Create tablespaces
- Create local and common users and roles
- Start services within PDBs

18

SKILLBUILDERS


SKILLBUILDERS
ORACLE DATABASE ADMINISTRATION
CLOUD ADMINISTRATION
ORACLE AND APEX HOSTING
APEX APPLICATION DESIGN AND DEVELOPMENT
TRAINING