ORACLE

SPAŢII TABEL

Datele gestionate de SGBDR Oracle sunt împărţite în mai multe unităţi logice de stocare numite spaţii-tabel (*tablespaces*). In felul acesta datele sunt grupate luând în considerare aria lor de aplicabilitate, de exemplu: *tablespace pentru datele compartimentului Resurse umane, tablespace pentru datele compartimentului Contabilitate, tablespace pentru datele compartimentului Aprovizionare.*

Din punct de vedere fizic, datele SGBDR Oracle sunt memorate în *fişierele de date* asociate spaţiilor-tabel. Cu alte cuvinte, datele din baza de date sunt stocate *logic* în spaţii-tabel şi *fizic* în fişiere de date.

Un spaţiu-tabel poate fi constituit din mai multe fişiere de date care pot fi distribuite pe mai multe discuri, deci mărimea unui spaţiu tabel nu este limitată de mărimea unui fişier sau spaţiu fizic de stocare. Pe de altă parte, un fişier de date este asociat doar unui spaţiu-tabel.Toate obiectele bazei de date (tabele, clustere, indecşi, etc.) trebuie să aibă specificat un spaţiu-tabel unde trebuie să fie create. Datele care alcătuiesc aceste obiecte sunt apoi stocate în fişierele de date alocate spaţiului-tabel respectiv.

Orice bază de date conţine un spaţiu-tabel numit SYSTEM. Ca parte a procesului de creare a bazei de date, Oracle creează automat acest spaţiu-tabel. Deşi o bază de date relativ mică poate încăpea în spaţiul-tabel `SYSTEM`, se recomandă crearea unor spaţii-tabel separate pentru datele aplicaţiei. Spaţiul-tabel `SYSTEM` este locaţia iniţială a obiectelor bazei de date. El conţine şi dicţionarul de date, adică informaţii despre obiectele bazei de date (tabele, vederi, indecşi, clustere, etc.).

Spaţiile-tabel sunt create şi administrate de către `DBA` (administratorul bazei de date). Fişierele de date vor fi create odată cu crearea sau modificarea (extinderea) unui spaţiu tabel, cu excepţia situaţiei când se specifică explicit refolosirea unui fişier de date existent (folosind opţiunea `REUSE` explicată mai jos).

În general, nu trebuie create spaţii tabel suplimentare fără un motiv întemeiat, fiind recomandată existenţa pe cât posibil a cât mai puţine spaţii tabel. De asemenea, în cadrul fiecărui spaţiu tabel este recomandată crearea a cât mai puţine fişiere de date, de dimensiuni cât mai mari. De exemplu, la crearea sau mărirea dimensiunii unui spaţiu tabel este recomandată crearea unui singur fişier de date de dimensiune mai mare decât a mai multor fişiere de dimensiuni mici.

Mărimea unui fişier de date este cea specificată la crearea acestuia şi nu reprezintă cantitatea datelor stocate în el. De exemplu, un fişier de date care va fi creat cu mărimea de 10 MB, va folosi toţi cei 10 MB indiferent dacă el conţine un milion de rânduri sau unul singur. Datorită acestui fapt, dimensiunea unui fişier de date trebuie planificată cu atenţie pentru ca, odată creat, un fişier de date nu mai poate fi micşorat. Începând cu versiunea 7.3 Oracle oferă posibilitatea ca fişierele de date să se extindă

în mod automat (folosind opţiunea AUTOEXTEND explicată mai jos) dacă spaţiul tabel corespunzător devine neîncăpător pentru datele ce trebuiesc stocate.

Spaţiile-tabel pot fi create folosind comanda SQL CREATE TABLESPACE, având sintaxa următoare:

```
CREATE TABLESPACE nume_spaţiu_tabel
DATAFILE specificaţie_fişier_de_date
[,specificaţie_fişier_de_date...]
[MINIMUM EXTENT întreg [K|M]]
[LOGGING | NOLOGGING]
[DEFAULT STORAGE parametrii_de_stocare ]
[ONLINE | OFFLINE]
[PERMANENT | TEMPORARY]
```

unde:

- DATAFILE specifică fişierul sau fişierele de date pe care le va cuprinde spaţiul tabel. Aceste fişiere vor fi create odată cu crearea spaţiului tabel. Sintaxa specificaţiei unui fişier de date este detaliată mai jos.
- MINIMUM EXTENT specifică mărimea minimă a extinderilor cuprinse în acest spaţiu tabel. Aceasta este în bytes (în mod implicit), Kbytes (dacă se foloseşte opţiunea K) sau Mbytes (dacă se foloseşte opţiunea M).
- LOGGING şi respectiv NOLOGGING[1] determină dacă la executarea anumitor comenzi SQL (încărcarea de date într-un tabel, crearea indecşilor, etc.) asupra obiectelor din spaţiul tabel se vor păstra sau nu informaţii despre aceste operaţii în fişierul jurnal pentru recuperare (redo log). Dacă nu se specifică nici o opţiune, valoarea implicită este LOGGING.
- DEFAULT STORAGE specifică valorile implicite (default) pentru parametrii de stocare ale tuturor obiectelor create în spaţiul tabel. Parametrii de stocare vor fi detaliaţi mai jos.
- ONLINE activează spaţiul tabel imediat după creare, făcându-l accesibil utilizatorilor care au permisiunea de a accesa spaţiul tabel. OFFLINE dezactivează spaţiul tabel imediat după creare, făcându-l inaccesibil. ONLINE este valoarea implicită.
- PERMANENT arată că spaţiul tabel va fi folosit pentru a stoca obiecte permanente. TEMPORARY arată că spaţiul tabel va fi folosit doar pentru a stoca obiecte temporare. PERMANENT este valoarea implicită.

Specificaţia unui fişier de date, notată mai sus prin *specificaţie_fişier_de_date*are următoarea sintaxă:

[1] Opţiunea NOLOGGING este o noutate adusă de versiunea Oracle8 şi poate fi folosită pentru a mări performanţa instrucţiunilor SQL în cazul bazelor de date mari. De exemplu, încărcarea într-un tabel a milioane de înregistrări în modul implicit LOGGING ar duce la înscrierea unei mari cantităţi de informaţii în fişierele jurnal, consumând astfel spaţiu fizic pe disc şi încetinind vizibil procesul.

nume_fişier
SIZE întreg [K|M]]
REUSE]
AUTOEXTEND {OFF|ON[NEXT întreg [K|M]]
[MAXSIZE [UNLIMITED| întreg [K|M]]]}]

unde

- SIZE specifică mărimea fişierului în bytes (în mod implicit), Kbytes (dacă se foloseşte opţiunea K) sau Mbytes (dacă se foloseşte opţiunea M).
- REUSE determină refolosirea unui fişier de date deja existent. În absenţa opţiunii REUSE, Oracle va crea un nou fişier de date dacă nu există deja un fişier cu numele menţionat sau va produce un mesaj de eroare în caz contrar. În cazul folosirii opţiunii REUSE, Oracle va folosi fişierul dacă el există sau va crea unul nou în caz contrar.
- opţiunea AUTOEXTEND arată dacă fişierul poate creşte automat (ON) sau nu (OFF) în cazul în care spaţiul tabel are nevoie de mai mult spaţiu. Valoarea implicită este OFF. În cazul când opţiunea AUTOEXTEND este ON, se pot specifica atât mărimea creşterii fişierului, în bytes Kbytes (K) sau Mbytes (M), prin opţiunea NEXT, cât şi valoarea maximă a fişierului, care poate fi un număr întreg de bytes Kbytes (K) sau Mbytes (M) sau pate fi nelimitată (UNLIMITED).

De exemplu, comanda SQL

```
CREATE TABLESPACE ts_alfa
DATAFILE ‘/date/ts_alfa01.dbf’
SIZE 20M;
```

creează spaţiul-tabel ts_alfa cu un fişier de date ts_alfa01.dbf de mărime 20M. În cazul în care se doreşte ca spaţiul tabel să se autoextindă automat cu câte 5M până la dimensiunea de 50 M se foloseşte comanda

```
CREATE TABLESPACE ts_alfa
DATAFILE ‘/date/ts_alfa01.dbf’
SIZE 20M
AUTOEXTEND ON NEXT 5M MAXSIZE 50M;
```

Sintaxa completă

```
CREATE [TEMPORARY / UNDO] TABLESPACE  <tblspc_name>
DATAFILE / TEMPFILE
'<datafile01_name and Path where file to create>' SIZE <integer M>
[,'<datafile02_name and Path where file to create>' SIZE <integer M>
[,'<datafile0N_name and Path where file to create>' SIZE <integer M>
[,...]]]
BLOCKSIZE  <DB_BLOCK_SIZE parameter /2k/4k/8k/16k/32k >
```

```
 AUTOEXTEND { [OFF/ON (NEXT <integer K/M >  MAXSIZE<integer K/M >) /
UNLIMITED] }
 LOGGING/NOLOGGING (Logging default)
 ONLINE/OFFLINE (Online default)
 EXTENT MANAGEMENT { [DICTIONARY] /
          [LOCAL Default (AUTOALLOCATE / UNIFORM <integer K/M >)] }
PERMANENT  / TEMPORARY (Permanent default)
 MINIMUM EXTENT
DEFAULT STORAGE  {    [INITIAL <integer K/M >]
                       [NEXT <integer K/M >]
                       [PCTINCREASE <integer K/M >]
                       [MINEXTENTS <integer>]
                       [MAXEXTENTS <integer> / UNLIMITED]
                       [FREELISTS <integer>]
                       [FREELIST GROUPS <integer>]
                       [OPTIMAL <integer>/NULL]
                       [BUFFER_POOL < DEFAULT/KEEP/RECYCLE >] }
 CHUNK <integer K/M >
 NOCACHE;
```

BLOCKSIZE – By Default blocksize define in the parameter DB_BLOCK_SIZE. In Oracle9i, multiple blocksize that is different block size for different tablespaces, can be defined; all datafiles of a same tablespace have the same block size.
DEFAULT STORAGE :
INITIAL – Specifies the size of the object's first extent.3 k minmum for Locally and 2 k minimum Dictionary.
NEXT – Specifies the size of the object's sucessive extent.
PCTINCREASE – Specifies the ratio of the third or the preceding extent of the object. The default value for PCTINCREASE is 50 % and the minimum value is 0%.
MINEXTENTS – The total number of extent allocated to the segment at the time of creation
MAXEXTENTS – The maximum number of extent that can be allocated to the segment .
MININUM EXTENT – The size is specifies in this clause.The extent are multiple of the size specified in this clause .NEXT and INITIAL extent size specified should be multiple of minmum extent.
PERMANENT / TEMPORARY – Permannent is default, use to store the table,index etc,Temporary is for temporay segments(sorts in Sql) can not store table,index in temporary tablespace.
LOGGING / NOLOGGING – Logging is default,the DDL operation & direct insert load are recorded in the redo log file.
ONLINE / OFFLINE - Online is default,tablespace is available as soon as created

Exemple:

```
CREATE TABLESPACE ts_alfa DATAFILE 'd:\date\fd_alfa.dbf'
SIZE 20M ;
```

```
CREATE TEMPORARY TABLESPACE ts_temp TEMPFILE 'd:\date\
fd_temp.dbf' SIZE 20M ;
```

Serverul Oracle permite adaugarea ulterioara a fisierelor de date unui spatiu-tabel pentru a se putea creste totalul de spatiu de pe disc alocat spatiului-tabel. Acest lucru se realizeaza cu comanda:

```
ALTER TABLESPACE nume
ADD DATAFILE specificatii_fisier  [ clauza_autoextend ]
```

Exemplu:

```
ALTER TABLESPACE ts_alfa
     ADD DATAFILE 'd:\date\fd_alfa2.dbf' SIZE 100M
```

Comanda care realizeaza stergerea spațiilor tabel are urmatoarea sintaxa:

```
DROP TABLESPACE nume
 [ INCLUDING CONTENTS  [ AND DATAFILES ]
[ CASCADE CONSTRAINTS ]  ]
```

unde:

- INCLUDING CONTENTS va forta stergerea tuturor segmentelor spatiului-tabel;
- AND DATAFILES va sterge si fisierele de date asociate spatiului-tabel;
- CASCADE CONSTRAINTS va sterge restrictiile de integritate referentiala (restrictii care apar daca tabelele ce urmeaza a fi sterse sunt parintii unor tabele din alt spatiu-tabel).

Exemplu:

```
DROP TABLESPACE ts_alfa INCLUDING CONTENTS AND DATAFILES;
```

Nu se poate sterge spatiul-tabel SYSTEM sau daca are segmente active. Prin urmare se recomanda ca inainte de a se sterge un spatiu-tabel, acesta sa fie facut OFFLINE pentru a fi siguri ca nu exista tranzactii care sa acceseze segmentele spatiului-tabel.

Comenzi pentru obtinerea informatiilor despre tablespaces:

```
SELECT tablespace_name, file_name
FROM dba_data_files;
```

```
SELECT tablespace_name, file_name
FROM dba_temp_files;
```

```
SELECT tablespace_name, file_name
FROM sys.dba_temp_files;
```

```
SELECT tablespace_name, file_name
FROM sys.dba_data_files;
```

```
SELECT dd.tablespace_name tablespace_name, dd.file_name
file_name, dd.bytes/1024 TABLESPACE_KB,
SUM(fs.bytes)/1024 KBYTES_FREE, MAX(fs.bytes)/1024
NEXT_FREE
```

```
FROM sys.dba_free_space fs, sys.dba_data_files dd
WHERE dd.tablespace_name = fs.tablespace_name
AND dd.file_id = fs.file_id
GROUP BY dd.tablespace_name, dd.file_name, dd.bytes/1024
ORDER BY dd.tablespace_name, dd.file_name;
```

UTILIZATORI SI SECURITATE

Baza de date Oracle conţine propriul ei sistem de securitate care previne accesul neautorizat la baza de date. Sistemul de securitate al bazei de date Oracle este realizat prin intermediul *utilizatorilor(conturilor utilizator)* bazei de date. Serverul bazei de date solicită numele(*contul*) utilizatorului şi parola pentru fiecare accesare la baza de date; indiferent de utilitarul folosit pentru interfaţă, serverul bazei de date nu permite accesul la baza de date dacă nu este utilizat un nume şi o parolă corectă.

Un cont utilizator are asociate anumite drepturi(privilegii) cu privire la acţiunile sale faţă de serverul Oracle sau faţă de obiectele stocate în baza de date. Unui utilizator îi este asociată automat –la creare– o ***schemă*** *cu acelaşi numecu al utilizatorului.*

O **schemă** reprezintă o colecţie de obiecte formată din tabele, vederi, proceduri stocate etc deţinute şi accesibile utilizatorului. Un utilizator poate avea o singură schemă, care va avea acelaşi nume cu acesta. O *schemă* este creată pentru a administra o aplicaţie, având drepturi depline asupra tuturor obiectelor aplicaţiei. Acest lucru are anumite avantaje dintr-o serie de motive: separă în general administrarea obiectelor bazei de date de administrarea obiectelor aplicaţiei şi permite ca toate datele aplicaţiei să poată fi exportate sau mutate de către proprietarul acestora.

Între spaţiile tabel şi schema unei baze de date nu există nici o legătură; obiectele unei scheme se pot găsi în spaţii tabel diferite, după cum şi într-un spaţiu tabel se pot găsi obiecte din mai multe scheme.

Pentru a accesa un obiect din propria schema, un utilizator poate folosi doar numele acestuia. Pentru a accesa un obiect din schema altui utilizator, trebuie specificat atât numele obiectului cât şi schema din care face parte, folosind sintaxa:

```
schema.obiect
```

De exemplu, un utilizator, să-l numim ***U1***, va putea crea în schema proprie un tabel numit `persoana`, folosind comanda:

```
CREATE TABLE persoana...;
```

Dacă însă un alt utilizator doreşte crearea tabelului `persoana` în schema utilizatorului ***U1***, va trebui să folosească comanda:

```
CREATE TABLE U1.persoana ...;
```

Pentru interogarea tabelului, utilizatorul ***U1*** va folosi comanda:

```
SELECT * FROM persoana;
```

În schimb, pentru a interoga tabelul, orice alt utilizator va folosi comanda:

```
SELECT * FROM U1.persoana;
```

În plus, pentru a accesa un obiect din schema altui utilizator, un utilizator va trebui să dispună de privilegiile necesare respectivului tip de acces. De exemplu, pentru interogarea tabelului `persoana`, orice alt utilizator va trebui să posede privilegiul `SELECT` pe tabelul `persoana` sau privilegiul de sistem `SELECT ANY TABLE`.

În general, trebuie să existe măcar trei tipuri de acces la o aplicaţie: administratorul bazei de date, dezvoltatorul şi utilizatorul:

- *Administratorul bazei de date* va trebui să administreze structurile şi obiectele bazei de date (cum ar fi fişierele, spaţiile tabel şi tabelele). Administratorul bazei de date este de fapt proprietarul aplicaţiei şi al bazei de date. El trebuie să deţină o varietate de privilegii de sistem.
- *Dezvoltatorul* va trebui să poată efectua atât operaţii de interogare, cât şi de manipulare (DML) şi definire (DDL) a datelor. Pentru aplicaţiile de dimensiuni reduse, proprietarul aplicaţiei este de obicei utilizat şi pentru dezvoltare. În general însă, pentru dezvoltare se foloseşte o altă schemă decât cea care deţine obiectele aplicaţiei, deci un dezvoltator trebuie să deţină în principal privilegii la nivel de obiect.
- *Utilizatorul*, pe de altă parte, va putea efectua doar operaţii de interogare şi manipulare a datelor, fără a avea însă permisiunea de a efectua operaţii de definire a datelor.

Schema care deţine obiectele aplicaţiei va acorda privilegiile necesare fiecărui tip de acces la aplicaţie.

In procesul de creare a bazei de date Oracle, sunt creaţi doi utilizatori **sys** şi **system** cu drepturi de administrare a bazei de date (deţin rolul DBA) .

Pentru a crea un nou utilizator şi implicit o nouă schemă în baza de date, trebuie să ne conectăm folosind un cont de administrator (*system*, *sys* sau un alt cont utilizator ce deţine rolul *DBA*).

```
create user student identified by student
```

Clauza `identified by` stabileşte parola asociata contului utilizator.

Sintaxa:

```
CREATE USER nume_utilizator
IDENTIFIED {BY parola | EXTERNALLY}
[DEFAULT TABLESPACE nume_spațiu_tabel]
[TEMPORARY TABLESPACE nume_spațiu_tabel]
```

```
[QUOTA {întreg [K|M]|UNLIMITED}on nume_spaţiu_tabel ]...
[PROFILE nume_profil ]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK|UNLOCK}]
```

unde

- IDENTIFIED indică modul în care Oracle permite accesul utilizatorului: prin parolă, când Oracle menţine în mod intern o parolă pentru utilizator, pe care acesta o va utiliza pentru accesarea bazei de date; EXTERNALLY, când verificarea accesului utilizatorului este făcută de către sistemul de operare - în acest caz utilizatorul trebuie să fie identic cu cel definit în sistemul de operare.
- DEFAULT TABLESPACE indică spaţiul tabel folosit în mod implicit de obiectele create de către utilizator. Dacă această clauză este omisă, spaţiul tabel implicit este SYSTEM.
- TEMPORARY TABLESPACE indică spaţiul tabel pentru segmentele temporare ale utilizatorului. Dacă această clauză este omisă, spaţiul tabel implicit pentru segmentele temporare este SYSTEM.
- QUOTA permite utilizatorului să aloce spaţiu pe un spaţiu tabel; la folosirea acestei clauze poate fi specificat spaţiul maxim alocat (în bytes, Kbytes (K) sau Mbytes (M)) sau poate fi folosită opţiunea UNLIMITED pentru a permite utilizatorului să aloce oricât de mult spaţiu pe spaţiul tabel.
- PROFILE atribuie utilizatorului un profil. Un profil este un set de limitări pentru resursele bazei de date (de exemplu, numărul maxim de sesiuni concurente, timpul maxim al unei sesiuni, etc.). Dacă profilul este atribuit unui utilizator, atunci utilizatorul nu poate depăşi limitele specificate de profil. Un profil poate fi creat folosind comanda SQL CREATE PROFILE. Dacă această clauză este omisă, Oracle atribuie în mod implicit utilizatorului profilul DEFAULT.
- PASSWORD EXPIRE forţează ca parola iniţială să fie schimbată de către utilizator în momentul în care acesta se va conecta pentru prima dată la baza de date.
- ACCOUNT LOCK|UNLOCK blochează sau deblochează contul permiţând sau nu accesul utilizatorului la baza de date. Dacă se foloseşte opţiunea de blocare (LOCK), contul utilizatorului este creat însă utilizatorul nu se va putea conecta până când nu se deblochează în mod explicit contul. Această opţiune se foloseşte, de exemplu, în cazul în care se preferă crearea unui cont pentru un angajat nou dar nu i se permite accesul până când nu este pregătit să-l folosească. Opţiunea implicită este UNLOCK.

Pentru a putea crea un alt utilizator, trebuie ca utilizatorul ce realizează acest lucru să deţină privilegiul de sistem CREATE USER.

Exemple:

```
CREATE USER u_programator
IDENTIFIED BY progr
```

```
DEFAULT TABLESPACE ts_alfa
QUOTA UNLIMITED ON ts_alfa
TEMPORARY TABLESPACE ts_temp
QUOTA UNLIMITED ON system;
```

```
create user u_extern identified by u_extern
```

Caracteristicile unui utilizator pot fi schimbate folosind comanda SQL ALTER USER. Folosind această comandă se poate schimba, de exemplu, parola unui utilizator:

```
ALTER USER u_programator
IDENTIFIED BY u_programator;
```

```
ALTER USER u_extern
    IDENTIFIED BY abcd
    DEFAULT TABLESPACE ts_alfa
QUOTA UNLIMITED ON ts_alfa;
```

Sintaxa comenzii **ALTER USER este**

```
ALTER USER
{ user
{ IDENTIFIED
{ BY password [ REPLACE old_password ]
      | EXTERNALLY [ AS 'certificate_DN' ]
      | GLOBALLY [ AS '[directory_DN]' ]
      }
    | DEFAULT TABLESPACE tablespace
    | TEMPORARY TABLESPACE
{ tablespace | tablespace_group_name }
    | QUOTA{ size_clause
           | UNLIMITED
           } ON tablespace
      [ QUOTA{ size_clause
             | UNLIMITED
             } ON tablespace
      ]...
    | PROFILE profile
    | DEFAULT ROLE{ role [, role ]...
                  | ALL [ EXCEPT
                          role [, role ]... ]
                  | NONE
                  }
    | PASSWORD EXPIRE
    | ACCOUNT{ LOCK | UNLOCK }
    }
[  {IDENTIFIED
{ BY password [ REPLACE old_password ]
           | EXTERNALLY [ AS 'certificate_DN' ]
           | GLOBALLY [ AS '[directory_DN]' ]
           }
         | DEFAULT TABLESPACE tablespace
         | TEMPORARY TABLESPACE
{ tablespace | tablespace_group_name }
```

```
        | QUOTA{ size_clause
               | UNLIMITED
               } ON tablespace
          [ QUOTA{ size_clause
                 | UNLIMITED
                 } ON tablespace
          ]...
        | PROFILE profile
        | DEFAULT ROLE{ role [, role ]...
                      | ALL [ EXCEPT
                              role [, role ]... ]
                      | NONE
                      }
        | PASSWORD EXPIRE
        | ACCOUNT{ LOCK | UNLOCK }
        }
    ]...
| user [, user ]...
  proxy_clause ;
```

Distrugerea unui utilizator se poate face folosind comanda SQL DROP USER. Dacă există obiecte în schema utilizatorului, atunci acestea trebuie distruse înainte de distrugerea utilizatorului. Aceasta se poate face automat folosind opţiunea CASCADE a acestei comenzi:

```
DROP USER u_programator CASCADE;
```

GRANT si REVOKE

ORACLE utilizeaza un sistem de securitate descentralizat: utilizatorii sunt responsabili pentru acordarea drepturilor de acces la obiectele pe care le detin, celorlalti utilizatori.

Comenzile GRANT si REVOKE sunt utilizate pentru definirea nivelurilor de acces la obiectele bazei de date.

Accesul unui utilizator la baza de date este administrat printr-un număr de drepturi numite privilegii de sistem, sau privilegii la nivelul bazei de date, care permit utilizatorului să efectueze operaţii precum conectarea la baza de date şi crearea de obiecte. Odată ce utilizatorul a creat obiecte ale bazei de date, el este apoi responsabil de a acorda drepturi altor utilizatori pentru obiectele care sunt proprietatea lui. Aceste drepturi sunt numite privilegii la nivel de obiect.

Rolurile sunt utilizate pentru a simplifica administrarea privilegiilor. Astfel, în loc de a acorda un anumit privilegiu direct unui utilizator, privilegiile sunt acordate unui rol, iar un rol este acordat la rândul lui unui utilizator. Cu alte cuvinte, rolurile reprezintă grupuri de privilegii.

Un rol poate fi creat folosind comanda SQL CREATE ROLE, având sintaxa următoare:

```
CREATE ROLE rol
[ NOT IDENTIFIED | IDENTIFIED {BY parola | EXTERNALLY}]
```

unde

- `NOT IDENTIFIED` arată că orice utilizator căruia îi va fi acordat rolul creat nu va trebui identificat în momentul când activează rolul
- `IDENTIFIED` arată că orice utilizator căruia îi va fi acordat rolul creat va trebui verificat. Această verificare poate fi făcută intern de către Oracle prin parolă sau extern de către sistemul de operare (EXTERNALLY). În acest ultim caz, în funcţie de sistemul de operare, utilizatorul va trebui să specifice o parolă sistemului de operare la activarea rolului.
- Dacă ambele opţiuni `NOT IDENTIFIED` şi `IDENTIFIED` sunt omise, opţiunea implicită este `NOT IDENTIFIED`.

```
CREATE ROLE rol_secretara;
CREATE ROLE rol_manager IDENTIFIED BY man;
```

Un utilizator poate crea un rol numai dacă deţine privilegiul de sistem `CREATE ROLE`.

Schimbarea modului de identificare a unui rol poate fi făcută folosind comanda SQL `ALTER ROLE`:

```
ALTER ROLE rol_manager NOT IDENTIFIED;
```

Distrugerea unui rol se poate face folosind comanda SQL `DROP ROLE`:

```
DROP ROLE rol_manager;
```

Privilegii de sistem

Acţiunile pe care un utilizator le poate efectua asupra bazei de date sunt administrate prin privilegiile de sistem acordate acestuia. În Oracle există peste 80 de privilegii de sistem, denumirea lor fiind inspirată de acţiunile pe care le permit. Ele variază de la permisiunea de conectare la baza de date (create session) la dreptul de a crea un tabel (create any table) sau index (create any index) sau de a distruge un tabel (DROP ANY TABLE) sau index (drop any index) din schema oricărui utilizator.

Exista un numar mare de privilegii sistem.
(http://www.psoug.org/reference/system_privs.html)

Privilegiile de sistem pot fi clasificate in:
Privilegii pentru operatii care afecteaza intreg sistemul, ca de exemplu
CREATE SESSION (permisiunea de a se conecta la baza de date), CREATE TABLESPACE

Privilegii care afecteaza obiectele din schema proprie, de ex. CREATE TABLE

Privilegii care afecteaza obiectele din orice schema, de ex. CREATE ANY TABLE

In numele lor, particula ANY arata ca userul are acel privilegiu in orice schema.

Pentru a adauga privilegii la un user se foloseste cererea GRANT.
Pentru a inlatura privilegii de la un user se foloseste REVOKE.

Exemple de privilegii:
CREATE session, CREATE table, CREATE view, CREATE procedure, CREATE synonym, CREATE sequence,
ALTER table, ALTER view, ALTER procedure, ALTER synonym, ALTER sequence
DROP table, DROP view, DROP procedure, DROP synonym,

Create Any Procedure
Create Any Sequence
Create Any SQL Profile
Create Any Synonym
Create Any Table
Create Any Trigger
Create Any Type
Create Any View

Observatii:
CREATE TABLE include si privilegiul CREATE INDEX.

Privilegiile CREATE TABLE, CREATE PROCEDURE si CREATE CLUSTER includ si dreptul de a sterge aceste obiecte.
Privilegiul UNLIMITED TABLESPACE nu poate fi asignat unui rol ci doar userilor particulari.
Pentru trunchierea unei tabele este necesar privilegiul DROP ANY TABLE

Privilegii la nivel de obiect

Securitatea obiectelor unei baze de date este administrată ca un număr de privilegii la nivel de obiect, care determină ce acces au utilizatorii la obiectele bazei de date. Privilegiile la nivel de obiect existente în Oracle sunt urmatoarele

Privilegiul de obiect	**Descrierea permisiunii**
SELECT	Selectarea rândurilor dintr-un tabel, vedere sau instantaneu şi extragerea numerelor dintr-un generator de secvenţe
INSERT	Inserarea înregistrărilor într-un tabel sau vedere
UPDATE	Actualizarea înregistrărilor dintr-un tabel sau vedere
DELETE	Ştergerea înregistrărilor dintr-un tabel sau vedere
ALTER	Modificarea structurii şi parametrilor unui tabel sau a unei secvenţe
REFERENCES	Referirea unui tabel utilizând chei străine

EXECUTE	Executarea unei proceduri funcţii, pachet sau proceduri externe şi accesarea obiectelor declarate în specificaţia pachetului: în plus, pentru opţiunea obiect incepand cu Oracle8, acest privilegiu se poate acorda şi asupra unui tip de date creat de utilizator, în acest caz acest tip de date putând fi folosit la crearea unor tabele, la definirea unor coloane din tabele şi la declararea unor variabile sau parametrii
INDEX	Crearea indecşilor tabelului

Pentru a efectua o acţiune asupra unui obiect (interogare, actualizare, distrugere, etc.), un utilizator trebuie să se găsească în unul din următoarele cazuri:
- să fie proprietarul acelui obiect;
- să aibă acordat privilegiul la nivelul obiectului respectiv de efectua acea acţiune ;
- să aibă acordat privilegiul de sistem care să îi permită acest lucru.

De exemplu, pentru a putea interoga (SELECT) un tabel, un utilizator trebuie să fie proprietarul acelui tabel, să posede privilegiul SELECT pe acel tabel sau să posede privilegiul de sistem SELECT ANY TABLE.

Acordarea rolurilor şi privilegiilor

Privilegiile de sistem, privilegiile la nivel de obiect sau rolurile pot fi acordate unui utilizator sau unui rol folosind comanda GRANT.
Bineînţeles, pentru a putea acorda un privilegiu sau un rol, utilizatorul respectiv trebuie să aibă acest drept. De exemplu, pentru a acorda un privilegiu de sistem sau un rol, utilizatorul trebuie să aibă privilegiul sau rolul respectiv acordat cu opţiunea ADMIN OPTION sau să aibă privilegiul GRANT ANY PRIVILEGE, respectiv GRANT ANY ROLE.

Acordarea unui privilegiu de sistem sau a unui rol

Pentru acordarea de privilegii de sistem sau roluri sintaxa comenzii GRANT este următoarea:

```
GRANT {privilegiu_de_sistem| rol} [,{privilegiu_de_sistem
| rol}]...
TO {utilizator |rol|PUBLIC} [,{utilizator|rol|PUBLIC}]...
[WITH ADMIN OPTION]
```

unde

- PUBLIC este folosit pentru a acorda privilegiile de sistem sau rolurile specificate tuturor rolurilor şi utilizatorilor existenţi.
- WITH ADMIN OPTION permite utilizatorului (rolului) căruia îi este acordat rolul (privilegiul) să acorde la rândul lui rolul (privilegiul) altui utilizator (rol). De asemenea, dacă unui utilizator sau rol îi este acordat un rol folosind WITH ADMIN

OPTION, atunci utilizatorul sau rolul respectiv poate modifica (ALTER ROLE) sau distruge (DROP ROLE) rolul acordat.

Un rol nu se poate acorda lui însuşi în mod direct sau printr-un set circular de atribuiri. Dacă se încearcă acest lucru, Oracle va genera un mesaj de eroare.

În continuare vom prezenta câteva exemple de acordare a privilegiilor de sistem sau a rolurilor.

Acordarea privilegiilor de sistem CREATE CLUSTER şi CREATE TABLE pentru rolul manager:

```
GRANT CREATE CLUSTER, CREATE TABLE TO rol_manager;
```

Acordarea privilegiului de sistem CREATE SESSION pentru utilizatorul manager, permiţând acestuia conectarea la baza de date:

```
GRANT CREATE SESSION TO rol_manager;
```

Acordarea rolului manager pentru utilizatorul**u_programator**:

```
GRANT rol_manager TO u_programator;
```

Acordarea unui privilegiu la nivel de obiect

Pentru acordarea de privilegii la nivel de obiect sintaxa comenzii GRANT este uşor diferită deoarece este nevoie să se identifice un obiect specific:

```
GRANT
{privilegiu_de_obiect | ALL}[(coloana [,coloana]...)]
[,{privilegiu_de_obiect | ALL}[(coloana [,coloana]...)]]
...
ON obiect
TO {utilizator | rol | PUBLIC}
[,{utilizator| rol |PUBLIC}]...
[WITH GRANT OPTION]
```

unde:

- ALL este folosit pentru a acorda toate privilegiile pentru obiectul respectiv
- coloana reprezintă coloana pentru care este acordat privilegiul. Coloanele pot fi specificate numai când sunt acordate privilegiile INSERT, REFERENCES, UPDATE. Dacă nu este listată nici o coloană, atunci privilegiul se acordă pe toate coloanele tabelei sau vederii în cauză.
- WITHGRANTOPTION permite celui care îi sunt acordate privilegiile de a le acorda la rândul lui altui utilizator sau rol.

Pentru o mai bună înţelegere a acestei comenzi, vom prezenta în continuare câteva exemple:

Acordarea privilegiului SELECT (dreptul de interogare) asupra tabelului Studenti pentru utilizatorul u_programator şi rolul rol_manager:

```
GRANT SELECT ON Studenti
TO u_programator, rol_manager WITH GRANT OPTION;
```

Acordarea privilegiului UPDATE (dreptul de actualizare) asupra coloanelor nume şi prenume ale tabelului studenti pentru utilizatorul u_programator:

```
GRANT UPDATE(nume, prenume) ON studenti TO u_programator;
```

Acordarea privilegiului INSERT (dreptul de inserare a înregistrărilor) şi a privilegiului DELETE (dreptul de ştergere al înregistrărilor) asupra tabelului studenti pentru utilizatorul u_extern:

```
GRANT INSERT, DELETE ON studenti TO u_extern;
```

Acordarea tuturor privilegiilor (SELECT, INSERT, UPDATE, DELETE, ALTER, REFERENCES, INDEX) asupra tabelului studenti pentru utilizatorul:**u_extern**

```
GRANT ALL ON studenti TO u_extern
```

Acordarea privilegiului SELECT (dreptul de interogare) asupra tabelului Studenti pentru toţi utilizatorii şi pentru toate rolurile existente:

```
GRANT SELECT ON studenti TO PUBLIC;
```

Trebuie menţionat că utilizatorul care este proprietarul schemei din care face parte obiectul are automat toate privilegiile asupra obiectului cu opţiunea WITH GRANT OPTION.

Revocarea rolurilor şi privilegiilor

Pentru a revoca un privilegiu sau un rol unui rol sau unui utilizator se poate folosi comanda SQL REVOKE. Comanda REVOKE trebuie folosită cu foarte mare atenţie deoarece poate anula un privilegiu propriu. Ca şi în cazul comenzii GRANT, şi comanda REVOKE are o sintaxă pentru revocarea privilegiilor de sistem sau a rolurilor şi altă sintaxă pentru revocarea privilegiilor la nivel de obiect.

Revocarea unui privilegiu de sistem sau a unui rol

Pentru revocarea de privilegii de sistem sau roluri, sintaxa comenzii REVOKE este următoarea:

```
REVOKE {privilegiu_de_sistem | rol} [,{privilegiu_de_sistem |
rol}]...
FROM {utilizator|rol|PUBLIC} [,{utilizator|rol|PUBLIC}]...
```

Dacă se anulează un rol care conţine alte roluri, întregul set de privilegii asociat cu fiecare rol va fi anulat. Dacă oricare dintre rolurile şi privilegiile conţinute de rolul anulat a fost acordat şi în mod direct utilizatorului, atunci acesta va continua să aibă privilegiile corespunzătoare.

Vom prezenta în continuare câteva exemple.

Revocarea privilegiului CREATE CLUSTER pentru rolul rol_costel:

```
REVOKE CREATE CLUSTER FROM rol_manager;
```

Revocarea rolului **rol_manager** pentru utilizatorul u_programator:

```
REVOKE rol_manager FROM u_programator;
```

Revocarea unui privilegiu la nivel de obiect

Pentru a revoca un privilegiu la nivel de obiect, sintaxa comenzii REVOKE este următoarea:

```
REVOKE {privilegiu_de_obiect | ALL} [(coloana [, coloana]...)]
      [,{privilegiu_de_obiect | ALL} [(coloana [,coloana]...)]]...
ON obiect
FROM {utilizator|rol|PUBLIC}, [{utilizator|rol|PUBLIC}]...
[CASCADE CONSTRAINTS]
[FORCE]
```

unde:

- CASCADE CONSTRAINTS indică faptul că se va şterge orice restricţie de integritate referenţială definită asupra tabelului de către utilizatorul respectiv. Această opţiune trebuie să fie folosită dacă privilegiul revocat este REFERENCES sau ALL PRIVILEGES şi dacă cel căruia i se revocă privilegiul a definit o restricţie de integritate referenţială asupra tabelului.
- FORCE forţează revocarea privilegiului EXECUTE asupra unui tip de date definit de către utilizator care are tabele dependente, aceste tabele devenind în acest caz invalide. În absenţa acestei opţiuni, dacă există tabele dependente, revocarea privilegiului EXECUTE asupra tipului de date va eşua.

Pentru a revoca toate privilegiile utilizatorului u_extern deţinute pe tabelul studenti, se va folosi următoarea comandă:

```
REVOKE ALL ON studenti FROM u_extern;
```

Pentru a revoca numai privilegiul SELECT deţinut de toţi utilizatorii şi rolurile existente pentru tabelul persoana, se va folosi următoarea comandă:

```
REVOKE SELECT ON studenti FROM PUBLIC;
```

Activarea şi dezactivarea rolurilor unui utilizator

La conectarea unui utilizator, Oracle va activa toate rolurile implicite (default) ale utilizatorului. Rolurile implicite ale utilizatorului sunt toate rolurile acordate acestuia, cu excepţia situaţiei în care rolurile implicite se definesc folosind clauza DEFAULT ROLE a comenzii SQL ALTER USER. Folosind comanda SQL ALTER USER cu clauza DEFAULT ROLE se pot identifica exact care din rolurile acordate utilizatorului sunt active şi care sunt inactive atunci când utilizatorul se conectează la baza de date. Orice modificare a listei de roluri implicite va avea efect începând cu următoarea conectare a utilizatorului la baza de date.

În cadrul unei sesiuni, rolurile atribuite unui utilizator pot fi activate sau dezactivate folosind comanda SQL SET ROLE, având sintaxa următoare:

```
SET ROLE {rol [IDENTIFIED BY parola] [, rol[IDENTIFIED BY parola]]...
| ALL[EXCEPT rol [,rol]...]
| NONE}
```

Aşa cum se observă din sintaxa prezentată mai sus, există trei opţiuni alternative care au următoarea semnificaţie:

- Lista rolurilor specifică rolurile care sunt activate pentru sesiunea curentă. Toate rolurile care nu sunt listate vor fi dezactivate pentru sesiunea curentă (vezi primele două exemple de mai jos). În cazul când rolul are o parolă, pentru activarea rolului este necesară specificarea acesteia.
- ALL activează pentru sesiunea curentă toate rolurile acordate utilizatorului, mai puţin cele enumerate în clauza EXCEPT, dacă ea există. Această opţiune nu poate fi folosită pentru activarea rolurilor cu parola.
- NONE dezactivează toate rolurile pentru sesiunea curentă

În continuare vom prezenta câteva exemple de folosire a acestei comenzi:

Activarea rolului rol1:

```
SET ROLE rol1;
```

Activarea rolului rol2 identificat prin parola rolul2:

```
SET ROLE rol2 IDENTIFIED BY rolul2;
```

Activarea tuturor rolurilor pentru sesiunea curentă:

```
SET ROLE ALL;
```

Activarea tuturor rolurilor în afară de rolul rol1:

```
SET ROLE ALL EXCEPT rol1;
```

Dezactivarea tuturor rolurilor pentru sesiunea curentă:

```
SET ROLL NONE;
```

Pentru a determina rolurile active din sesiunea curentă se poate examina vederea `SESSION_ROLES` din dicţionarul de date.Numărul maxim de roluri care pot fi active la un anumit moment este specificat de către parametrul de iniţializare `MAX_ENABLED_ROLES`.

Rolurile **CONNECT**, **RESOURCE** şi **DBA**

CONNECT, RESOURCE şi DBA sunt roluri predefinite. Atunci când cream un utilizator cu Oracle Enterprise Manager, rolul Connect este acordat în mod automat acelui utilizator.

Rolul CONNECT acordă utilizatorilor următoarele privilegii de sistem:

* ALTER SESSION
* CREATE CLUSTER
* CREATE DATABASE LINK
* CREATE SEQUENCE
* CREATE SESSION
* CREATE SYNONYM
* CREATE TABLE
* CREATE VIEW

Rolul **RESOURCE** acordă utilizatorilor următoarele privilegii de sistem:

* CREATE CLUSTER
* CREATE PROCEDURE
* CREATE SEQUENCE
* CREATE TABLE
* CREATE TRIGGER
* CREATE TYPE

Rolul **DBA** include toate privilegiile de sistem (95 granturi separat).

INFORMATII DESPRE OBIECTELE BAZEI DE DATE

```
create user student identified by student
```

Privilegii de sistem

```
grant select any table to student
```

```
grant create any table to student
```

- permite utiliz student sa creeze tabele in orice schema

```
describe t1
```
 – informatii despre structura tabelului t1

```
GRANT select ON tabel TO PUBLIC
```
 Se acorda privilegiul select pentru tabelul tabel tuturor rolurilor si utilizatorilor existenti

```
GRANT select ON tabel TO utilizator
```
Se acorda privilegiul select pentru tabelul tabel utilizatoruluiutilizator

```
GRANT select ON tabel TO rol
```

```
GRANT  all privileges ON tabel to utilizator/rol
```

sau

```
GRANT  all ON tabel to utilizator/rol
```

Se acorda toate privilegiile pentru tabelul *tabel*utilizatoruluui*utilizator* sau rolului *rol*

```
REVOKE select ON tabel FROM utilizator
```

INFORMATII DESPRE OBIECTELE BAZEI DE DATE
select * from all_users –lista tuturor utilizatorilor
select * from user_catalog --informatii despre tabelele si vederile utilizatorului curent
select * from user_objects --informatii despre toate obiectele utilizatorului curent
select * from user_all_tables --lista tabelelor disponibile utilizatorului
select * from user_tables --lista tabelelor disponibile utilizatorului
select * from user_constraints --informatii despre constrangeri
select * from user_indexes --informatii despre indecsi
select * from user_Triggers --informatii despre triggere

select * from user_tab_privs– informatii despre privilegiile utilizatorului

describe t1 – informatii despre structura tabelului

-- SCHEMA RESURSE_UMANE
drop table Angajati;
drop table Compartimente;

create table Compartimente
(CodCompartiment char(10) primary key,

```
 DenCompartiment varchar2(30) not null
 );

create  table Angajati
 (CodAngajat char(10) primary key,
  Nume varchar(30) not null,
  CodCompartiment char(10)  references tCompartimente(CodCompartiment)
  );

Grant select on Angajati to ResUM01;
Grant select on Compartimente to ResUM01;

grant selecton Angajati to statistica;
grant references on Angajati to statistica;


-- schema STATISTICA

create table Colaboratori(codAng primary key references
RESURSEUMANE.ANGAJATI(CODANGAJAT),
functia varchar(30));


select A.*,b.nume from COLABORATORI A inner join
RESURSEUMANE.ANGAJATI B on A.codang=B.codangajat
```