# CHAPTER 9

1. **GRANT statement**

A user automatically has all object privileges for schema objects contained in the user’s schema. A user can grant any object privilege on any schema object.

* Introduction

Use the GRANT statement to give privileges to a specific user or role, or to all users, to perform actions on database objects. You can also use the GRANT statement to grant a role to a user, to PUBLIC, or to another role.

The following types of privileges can be granted:

* + Delete data from a specific table.
  + Insert data into a specific table.
  + Create a foreign key reference to the named table or to a subset of columns from a table.
  + Select data from a table, view, or a subset of columns in a table.
  + Create a trigger on a table.
  + Update data in a table or in a subset of columns in a table.
  + Run a specified function or procedure.
  + Use a sequence generator or a user-defined type.
* Syntax:

\* Syntax for tables

**GRANT** [privilege-type](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljgrant.html#rrefsqljgrant__grantprivtype) **ON [TABLE] {** [table-Name](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftablename.html#rreftablename) **|** [view-Name](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefviewname.html#rrefviewname) **} TO** [grantees](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljgrant.html#rrefsqljgrant__grantgrantees)

\* Syntax for routines

**GRANT EXECUTE ON { FUNCTION | PROCEDURE }** [routine-designator](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljgrant.html#rrefsqljgrant__grantroutinename) **TO** [grantees](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljgrant.html#rrefsqljgrant__grantgrantees)

* Example

**GRANT** **UPDATE**, **TRIGGER** **ON** **TABLE** t **TO** example

**GRANT** EXECUTE **ON** **PROCEDURE** p **TO** example

1. **Revoke**

* Introduction

Use the REVOKE statement to remove privileges from a specific user or role, or from all users, to perform actions on database objects. You can also use the REVOKE statement to revoke a role from a user, from PUBLIC, or from another role

The following types of privileges can be revoked:

* + Delete data from a specific table.
  + Insert data into a specific table.
  + Create a foreign key reference to the named table or to a subset of columns from a table.
  + Select data from a table, view, or a subset of columns in a table.
  + Create a trigger on a table.
  + Update data in a table or in a subset of columns in a table.
  + Run a specified routine (function or procedure).
  + Use a sequence generator or a user-defined type.
* Syntax:

\* Syntax for tables

**REVOKE** [privilege-type](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljrevoke.html#rrefsqljrevoke__revokeprivtype) **ON [ TABLE ] {** [table-Name](https://docs.oracle.com/javadb/10.8.3.0/ref/rreftablename.html#rreftablename) **|** [view-Name](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefviewname.html#rrefviewname) **} FROM** [grantees](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljrevoke.html#rrefsqljrevoke__revokegrantees)

\* Syntax for routines

**REVOKE EXECUTE ON { FUNCTION | PROCEDURE }** [routine-designator](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljrevoke.html#rrefsqljrevoke__revokeroutinename) **FROM** [grantees](https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljrevoke.html#rrefsqljrevoke__revokegrantees) **RESTRICT**

The RESTRICT clause specifies that the EXECUTE privilege cannot be revoked if the specified routine is used in a view, trigger, or constraint, and the privilege is being revoked from the owner of the view, trigger, or constraint.

* Example:

**REVOKE** **UPDATE**, **TRIGGER** **ON** **TABLE** t **FROM** example

**REVOKE** EXECUTE **ON** **PROCEDURE** p **FROM** example RESTRICT

1. **Job**

Job is used to run action at a fixed time

* Create job

begin  
dbms\_scheduler.create\_job (  
   job\_name           =>  'run\_load\_sales',  
   job\_type           =>  'STORED\_PROCEDURE',  
   job\_action         =>  'name of stored procedure',  
   start\_date         =>  '01-MAR-2010 03:00:00 AM',  
   repeat\_interval    =>  'FREQ=DAILY',  
   enabled            =>  TRUE);  
END;

There are a large number of expressions that can be used to define the repeat interval for a job.  Here are a few examples:

|  |  |
| --- | --- |
| **repeat\_interval** | **Description** |
| freq=hourly | Run every hour |
| freq=daily; byhour=3 | Run at 3 am every day |
| freq=daily; byhour=8,20 | Run at 8 am and 8 pm every day |
| freq=monthly; bymonthday=1 | Run on the first day of every month |
| freq=monthly; bymonthday=-1 | Run on the last day of every month |
| freq=yearly; bymonth=sep; bymonthday=20; | Run yearly on September 20th |

Example:

**BEGIN**

dbms\_scheduler.create\_job('"JOB\_RPT\_ASSET\_DAILY"',

job\_type=>'STORED\_PROCEDURE',

job\_action=> 'sprpt\_asset\_daily',

number\_of\_arguments=>0,

start\_date=>**TO\_TIMESTAMP\_TZ**('22-DEC-2017 05.38.49.000000000 PM +07:00','DD-MON-RRRR HH.MI.SSXFF AM TZR','NLS\_DATE\_LANGUAGE=english'),

repeat\_interval=>'FREQ=DAILY; INTERVAL=1; BYDAY=MON,TUE,WED,THU,FRI; BYHOUR=19; BYMINUTE=50',

end\_date=>**NULL**,

job\_class=>'"DEFAULT\_JOB\_CLASS"',

enabled=>**FALSE**,

auto\_drop=>**FALSE**,

comments=>'Tu dong bao cao tai san tung khach hang cho ezmargin, ezreport lay bao cao'

);

dbms\_scheduler.enable('"JOB\_RPT\_ASSET\_DAILY"');

**COMMIT**;

**END**;

**BEGIN**

dbms\_scheduler.create\_job('"JOB\_RPT\_STOCK\_STATEMENT\_I"',

job\_type=>'PLSQL\_BLOCK',

job\_action=>'begin sptrpt\_stock\_statement\_i(1); end;',

number\_of\_arguments=>0,

start\_date=>**TO\_TIMESTAMP\_TZ**('29-NOV-2016 10.43.30.000000000 AM +07:00','DD-MON-RRRR HH.MI.SSXFF AM TZR','NLS\_DATE\_LANGUAGE=english'),

repeat\_interval=>

'FREQ=DAILY; INTERVAL=1; BYDAY=MON,TUE,WED,THU,FRI; BYHOUR=15,18; BYMINUTE=30',

end\_date=>**NULL**,

job\_class=>'"DEFAULT\_JOB\_CLASS"',

enabled=>**FALSE**,

auto\_drop=>**FALSE**,

comments=> 'Sao ke CK daily, chay vao 15h30, 18h30'

);

dbms\_scheduler.enable('"JOB\_RPT\_STOCK\_STATEMENT\_I"');

**COMMIT**;

**END**;

* Running a Job Manually

begin  
dbms\_scheduler.run\_job (job\_name => 'name');  
end;

* Stopping Running Jobs

dbms\_scheduler.stop\_job (job\_name => 'name');  
end;

* Disabling and Enabling Jobs

begin  
dbms\_scheduler.disable (job\_name => 'name',..);  
end;  
To re-enable a job which has been disabled, use the enable procedure.

begin  
dbms\_scheduler.enable (job\_name => 'name',..);  
end;

* Dropping Jobs

begin  
dbms\_scheduler.drop\_job ('name',..);  
end;

1. **Practices**
   1. Create tblLog table with following structures:



* + 1. Write stored procedure “sp\_tblLog\_insert” to insert system date to tblLog table
    2. Create job to run sp\_tblLog\_insert on 6 am on every day. Run job
  1. Grant another user access to your job\_grades table. Take back the privilege from the user.
  2. Grant another user query privilege on your table. Then, verify whether that user can utilize the privilege. There are three users: user01, user02, user03.

User01 contains: Employees

* + Grant user02 privilege to view records in Employees table. Include an option for this user to further grant this privilege to other users.
  + From user02, write to query Employees table of user01
  + Grant privilege to view records in Employees table to a third user
  + Take back the privilege from the user who performs step 2 (user02)
  1. Grant another user query and data manipulation privileges on your Departments table. Make sure that the user cannot pass on these privileges to other users.
  2. Take back the privileges on the Departments table granted to another user.