Institute of Technology Tralee

Institiúid Teicneolaíochta Trá Lí

Database Concepts

Lab 08 – Controlling Access to Data

In this lab you will learn how to:

- Create a database View
- Drop a database View
- Grant and Revoke user privileges

Before you start:

Ensure that you have the tables from the Stock ordering database used last week:

The tables are:

Stock
Customers
Orders
OrderItems

Creating and Dropping Views

A view is a representation of one or more tables (in SQL).

Views are used to:

- Hide the complexity of relationships between tables
- Provide security for sensitive data in tables

When a view is defined, an SQL statement is associated with the view name. Whenever the view is accessed, the SQL statement will be executed. Views can provide a ***limited*** view of a table.

Syntax:

```
CREATE VIEW view_name AS
SELECT * | (col list)
FROM table1, table2,….
[WHERE condition]
[GROUP BY col_name [HAVING condition]]
[ORDER BY col1,col2,…];
```

Consider the view

```
CREATE VIEW BigOrders AS
SELECT OrdID, OrDate, OrdValue
FROM Orders
WHERE OrdValue > 500;
```

The view created has a limited number of columns (OrdID, OrdDate, OrdValue) and a limited set of data rows (WHERE OrdValue > 500) from the table Orders.

Once a view is created, it can be queried with a SELECT statement as if it were a table.

```
SELECT *
FROM BigOrders;
```

Views can be dropped in a similar way to tables. The **DROP VIEW** command provides this facility. In the following example, the view ***BigORders*** is dropped.

```
DROP VIEW BigOrders;
```

Views can be created to join several tables together. Consider the following example which creates a view that joins the tables Orders and Stock

```
CREATE VIEW Order_Details AS
SELECT O.OrdID,OrdDate, I.StockID, Description
FROM Orders O, OrderItems I, Stock S
WHERE O.ordID = I.OrdID AND
        I.StockID = S.StockID;
```

We might then query the view as follows:

```
SELECT *
FROM Order_Details;
```

OR

```
SELECT OrdDate, Description
FROM Order_Details
ORDER BY Description, OrdDate;
```

A view can be used as part of other queries or as the basis for developing applications.

A view can be created that contains an aggregate function.

Consider a view that returns the total number sold for each stock item:

```
CREATE VIEW Total_Sold AS
SELECT StockId, SUM(Qty) as Tot_Sold
FROM OrderItems
GROUP BY StockId;
```

In general, views are read only.

To see which views are defined in a schema, query the system table **USER_VIEWS.**

In SQL*Plus / SQL Developer:

```
SELECT View_Name
FROM User_Views;
```

In SQL Developer, you can simply expand the ‘Views’ tab in the Connections window to see all views that exist.

Grant and Revoke Statements

The GRANT and REVOKE statements allow a user to control access to database resources (objects):

- Tables
- Views
- Sequences
- Procedures

The **GRANT** command grants authorisation for a ***subject*** (another user or user group) to perform some ***action*** (SELECT, INSERT, UPDATE, DELETE, ALTER) on an ***object*** (Table, view, stored procedure).

The general syntax for the GRANT statement is:

GRANT <action 1>, <action 2>,….
ON Tablename
TO Subject;

For example, if the user TOM wishes to allow the user BOB to view the rows in ***his*** Stock table, TOM would execute the following GRANT statement:

GRANT SELECT
ON Stock
TO BOB;

Following this, BOB may now issue SQL SELECT statements on the table ***TOM.Stock***.

For example:

```
SELECT *
FROM TOM.Stock;
```

Your T-Number is your user name. This means that the table Orders in your tablespace is actually identified by:

t00036647. Orders

However, the owner's username is taken as the ***default owner*** and does not need to be specified.

Exercise 1:

Work in pairs for the next part of this lab.

Give your T-Number to your partner.
You will now give your partner permission to see ***your*** table lecturers.

```
GRANT SELECT
ON Orders
TO tnnnnnnnn;
```

Use SQL SELECT to look at your partners table Orders.

- What happens if you try to update this table?
- Can you see changes that your partner makes to his/her own table?
- Ask your partner to execute a COMMIT statement and take another look!

The **REVOKE** command reverses the authorisation by removing privileges from a user.

The syntax for the REVOKE statement is:

REVOKE <action>
ON <object>
FROM <subject>;

For example:

REVOKE SELECT
ON Stock
FROM BOB;

Current authorisations can be viewed by selecting from the USER_TAB_PRIVS view:

```
SELECT *
FROM USER_TAB_PRIVS;
```