COSC 304
Introduction to Database Systems

Views and Security

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

Views

A ***view*** is a named query that is defined in the database.

- To the user, a view appears just like any other table and can be present in any SQL query where a table is present.

A view may either be:

- *virtual* - produced by a SQL query on demand.
- *materialized* - the view is stored as a derived table that is updated when the tables that it refers to are updated.

Creating Views

Views are created using the **CREATE VIEW** command:

```
CREATE VIEW viewName [(col1,col2,...,colN)]
AS selectStatement [WITH CHECK OPTION]
```

Notes:

- Select statement can be any SQL query and is called the ***defining query*** of the view.
- It is possible to rename the columns when defining the view, otherwise they default to the names produced by the query.
- `WITH CHECK OPTION` is used to insure that if updates are performed through the view, the updated rows must satisfy the `WHERE` clause of the query.

Views Example

Create a view that has only the employees of department 'D2':

```
CREATE VIEW empD2
AS SELECT * FROM emp WHERE dno = 'D2';
```

Create a view that only shows the employee number, title, and name:

```
CREATE VIEW staff (Number,Name,Title)
AS SELECT eno,ename,title FROM emp;
```

- The first example is a *horizontal view* because it only contains a subset of the rows.
- The second example is a *vertical view* because it only contains a subset of the columns.

Removing Views

Views are removed using the **DROP VIEW** command:

```
DROP VIEW viewName [RESTRICT|CASCADE]
```

Notes:

- RESTRICT will not delete a view if other views are dependent on it.
- CASCADE deletes the view and all dependent views.

View Resolution

When a query uses a view, the process of ***view resolution*** is performed to translate the query into a query over only the base relations.

```
CREATE VIEW staff (Number,Name,Job)
AS SELECT eno,ename,title FROM emp WHERE dno = 'D2';

SELECT Number, Name FROM staff
WHERE job = 'EE' ORDER BY Number;
```

Step #1: Replace column names in `SELECT` clause with names in defining query.

```
SELECT eno, ename FROM staff
WHERE job = 'EE' ORDER BY Number;
```

View Resolution (2)

Step #2: View names in FROM clause replaced by FROM clause in defining query.

```
SELECT eno, ename FROM emp
WHERE job = 'EE' ORDER BY Number;
```

Step #3: WHERE clause of user and defining query are combined. Replace column view names with names in defining query.

```
SELECT eno, ename FROM emp
WHERE title = 'EE' AND dno = 'D2' ORDER BY Number;
```

View Resolution (3)

Step #4: `GROUP BY` and `HAVING` clause copied to user query from defining query. (no change in example)

```
SELECT eno, ename FROM emp
WHERE title = 'EE' AND dno = 'D2' ORDER BY Number;
```

Step #5: Rename fields in `ORDER BY` clause.

```
SELECT eno, ename FROM emp
WHERE title = 'EE' AND dno = 'D2' ORDER BY eno;
```

View Limitations

1) If a view contains an aggregate function, it can only be used in `SELECT` and `ORDER BY` clauses of queries.

2) A view whose defining query has a `GROUP BY` cannot be joined with another table or view.

View Updatability

When a *base table is updated*, all views defined over that base table are automatically updated. When an *update is performed on the view*, it is possible to update the underlying table.

A view is only *updatable* if:

- does not use `DISTINCT`
- every element in `SELECT` is a column name (no derived fields)
- contains all fields of base relation that are non-NULL
- `FROM` clause contains only one table
- `WHERE` does not contain any nested selects
- no `GROUP BY` or `HAVING` in defining query

In practice, a view is only updatable if it uses only a single table, and any update through the view must not violate any of the integrity constraints of the table.

View WITH CHECK OPTION

WITH CHECK OPTION is used for updatable views. It is used to prevent updates through a view that would then remove rows from the view.

```
CREATE VIEW staff (Number,Name,Job,DeptNum)
AS SELECT eno,ename,title,'D2' FROM emp
                    WHERE DeptNum = 'D2';

UPDATE staff SET DeptNum = 'D3' WHERE Number='E3';
```

This update would remove the employee `E3` from the view because changing his department to '`D3`' no longer matches the view which only contains employees in department '`D2`'. To prevent this, can specify the `WITH CHECK OPTION`.

```
CREATE VIEW staff (Number,Name,Job,DeptNum)
AS SELECT eno,ename,title FROM emp WHERE DeptNum = 'D2'
WITH CHECK OPTION;
```

View Updatable Question

Question: Select one view definition that would be updatable.

A)
```
CREATE VIEW v AS
SELECT DISTINCT salary FROM emp
```

B)
```
CREATE VIEW v AS
SELECT * FROM emp WHERE eno > 'E3'
```

C)
```
CREATE VIEW v AS
SELECT name FROM emp
```

D)
```
CREATE VIEW v AS
SELECT eno, salary/12 FROM emp
```

Advantages and Disadvantages of Views

Advantages:

- Data independence - allows base relations to change without affecting users.
- Security - views can be used to limit access to certain data to certain users.
- Easier querying - using views can often reduce the complexity of some queries.
- Convenience/Customization - users only see the parts of the database that they have interest and access to.

Disadvantages:

- Updatable views are not always supported and are restrictive.
- Performance - views add increased overhead: during query parsing and especially if they are materialized.

Views Practice Questions

Creates views that:

- 1) Show only employees that have title '`EE`'.
- 2) List only projects that have someone working on them.
- 3) List only the employees that manage another employee.
- 4) List the department number, name, and average employee salary in the department.
- 5) Show all employees but only lists their number, name, and title.
- 6) Show all employee and `worksOn` information for employee '`E1`'.

SQL Security

Security in SQL is based on ***authorization identifiers***, ***ownership***, and ***privileges***.

An authorization identifier (or user id) is associated with each user. Normally, a password is also associated with a authorization identifier. Every SQL statement performed by the DBMS is executed on behalf of some user.

The authorization identifier is used to determine which database objects the user has access to.

Whenever a user creates an object, the user is the owner of the object and initially is the only one with the ability to access it.

SQL Privileges

Privileges give users the right to perform operations on database objects. The set of privileges are:

- `SELECT` - the user can retrieve data from table
- `INSERT` - the user can insert data into table
- `UPDATE` - the user can modify data in the table
- `DELETE` - the user can delete data (rows) from the table
- `REFERENCES` - the ability to reference columns of a named table in integrity constraints
- `USAGE` - the ability to use domains, character sets, and translations (i.e. other database objects besides tables)

Notes:

- `INSERT` and `UPDATE` can be restricted to certain columns.
- When a user creates a table, they become the owner and have full privileges on the table.

SQL GRANT Command

The **GRANT** command is use to give privileges on database objects to users.

```
GRANT {privilegeList | ALL [PRIVILEGES]}
  ON ObjectName
  TO {AuthorizationIdList | PUBLIC}
  [WITH GRANT OPTION]
```

The privilege list is one or more of the following privileges:

```
SELECT [(columnName [,...]]
DELETE
INSERT [(columnName [,...]]
UPDATE [(columnName [,...]]
REFERENCES [(columnName [,...]]
USAGE
```

SQL GRANT Command (2)

The `ALL PRIVILEGES` keyword grants all privileges to the user except the ability to grant privileges to other users.

The `PUBLIC` keyword grants access to all users (present and future) of the database.

The `WITH GRANT OPTION` allows users to grant privileges to other users. A user can only grant privileges that they themselves hold.

GRANT Examples

Allow all users to query the Dept relation:

```
GRANT SELECT ON dept TO PUBLIC;
```

Only allow users Manager and Director to access and change Salary in Emp:

```
GRANT SELECT, UPDATE(salary) ON Emp TO Manager,Director;
```

Allow the Director full access to Proj and the ability to grant privileges to other users:

```
GRANT ALL PRIVILEGES ON Proj TO Director
    WITH GRANT OPTION;
```

Required Privileges Example

What privileges are required for these statements:

```
UPDATE Emp SET salary=salary*1.1 WHERE eno IN (
    SELECT eno FROM WorksOn WHERE hours > 30);
```

Answer:

```
SELECT on Emp

UPDATE(salary) on Emp

SELECT on WorksOn
```

Required Privileges Question

Question: What are the required privileges for this statement?

```
DELETE FROM dept WHERE dno NOT IN
    (SELECT dno FROM WorksOn);
```

A) DELETE, SELECT

B) DELETE on dept, DELETE on WorksOn

C) DELETE on dept, SELECT on WorksOn

D) DELETE on dept

E) DELETE on dept, SELECT on WorksOn, SELECT on dept

Required Privileges Question (2)

Question: What are the required privileges for this statement?

```
INSERT INTO WorksOn (eno,pno)
            VALUES ('E5','P5');
```

A) INSERT on WorksOn
B) INSERT
C) INSERT, SELECT
D) INSERT on WorksOn, UPDATE on WorksOn
E) none

GRANT Question

Question: **True or False:** A user `WITH GRANT OPTION` can grant a privilege that they do not hold themselves.

A) true

B) false

GRANT Question (2)

Question: **True or False:** A users may be granted the same privilege on the same object from multiple users.

A) true

B) false

SQL REVOKE Command

The REVOKE command is used to remove privileges on database objects from users.

```
REVOKE [GRANT OPTION FOR] {privilegeList | ALL [PRIVILEGES]}
  ON ObjectName
  FROM {AuthorizationIdList | PUBLIC} {RESTRICT|CASCADE}
```

Notes:

- ALL PRIVILEGES removes all privileges on object.
- GRANT OPTION FOR removes the ability for users to pass privileges on to other users (not the privileges themselves).
- RESTRICT - REVOKE fails if privilege has been passed to other users.
- CASCADE - removes any privileges and objects created using the revoked privileges including those passed on to others.

SQL REVOKE Command (2)

Privileges are granted to an object to a user *from* a specific user. If a user then revokes their granting of privileges, that only applies to that user.

Example:

- User A grants all privileges to user B on table T.
- User B grants all privileges to user C on table T.
- User E grants `SELECT` privilege to user C on table T.
- User C grants all privileges to user D on table T.
- User A revokes all privileges on table T from B (using `cascade`).
- This causes all privileges to be removed for user C as well, except the `SELECT` privilege which was granted by user E.
- User D now has only the `SELECT` privilege as well.

REVOKE Examples

Allow no users to query the Dept relation:

```
REVOKE SELECT ON dept FROM PUBLIC;
```

Remove all privileges from user Joe on Emp table:

```
REVOKE ALL PRIVILEGES ON Emp FROM Joe;
```

REVOKE Question

Question: User *A* executes:

```
GRANT SELECT, UPDATE ON T TO B WITH GRANT OPTION;
```

then User B executes:

```
GRANT SELECT, UPDATE ON T TO C;
```

then User A executes:

```
REVOKE GRANT OPTION FOR SELECT, UPDATE ON T FROM B;
```

True or False: User *C* loses `SELECT` on *T*.

A) true

B) false

SQL Security and Views

Views are used to provide security and access restriction to certain database objects.

Example: Consider the `Emp` relation. We want the user `Staff` to have only query access but not be able to see users `birthdate` and `salary`. How do we accomplish this?

Step #1: Create a view on the `Emp` relation.

```
CREATE VIEW EmpView AS
SELECT eno, ename, title, supereno, dno FROM emp;
```

SQL Security and Views (2)

Step #2: Provide `SELECT` privilege to staff.

```
GRANT SELECT ON EmpView TO Staff;
```

Step #3: `REVOKE` privileges on base relation (`Emp`)

```
REVOKE SELECT ON Emp From Staff;
```

Privileges on Views Question

Question: Table *T* is part of view definition query for view *V*. If user *A* has `SELECT` on *V*, but not on *T*, can user *A* run a query on *V* and see results?

A) yes

B) no

Security Practice Questions

Use `GRANT` and `REVOKE` to provide the desired access for each of these cases. You may also need views. Assume that you are the DBA who has full privileges (owner) of all objects currently in the database.

1) Allow all users access to the `Dept` relation.

2) Allow the user `Accounting` to read and update the `salary` of `Emp`.

3) Allow user `Davis` to have full control of the employees in Dept `D2` including the ability to grant privileges to others.

4) Allow user `Smith` to only see their employee record (`eno='E3'`) and `WorksOn` information.

5) `Davis` is replaced as head of Dept. `D2`, so only leave him query access to employees in `D2`.

Conclusion

Views are used to define virtual relations over base relations. This may be useful to:

- reduce query complexity
- improve performance (especially if view is materialized)
- provide different user views of the database
- allow security to be enforced on subsets of the schema

SQL security is enforced using `GRANT`/`REVOKE`.

- Views are useful for security as users can be given privileges to access a view but not the entire relation.

Objectives

Views:

- define views using `CREATE VIEW` from high-level description
- explain and illustrate the process of view resolution
- explain when a view is updateable and argue why updating views is not possible in certain cases
- explain what the `WITH CHECK OPTION` does for views
- list advantages and disadvantages of views

Security:

- be able to use `GRANT`/`REVOKE` syntax
- list the types of privileges and know when to use them
- given a SQL command, explain what privileges are required for it to execute
- explain how views are useful for security