Database Security

Views and Privileges

Views

CITY UNIVERSITY LONDON
EST 1894

- In some cases, it is not desirable for all users to see the entire logical model (that is, all the actual relations stored in the database.)
- Consider a person who needs to know instructor's name and department, but not the salary. This person should see a relation described, in SQL, by

```
SELECT ID, name, dept_name
FROM instructor
```

- A view provides a mechanism to hide certain data from the view of certain users.
- Any relation that is not of the conceptual model but is made visible to a user as a "virtual relation" is called a view.

What Is a View?

CITY UNIVERSITY LONDON
EST 1894

EMPLOYEES table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALA
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_FRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNO_D	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	25
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	105
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	110
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	86
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	70
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

20 rows selected.

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

Advantages of Views

CITY UNIVERSITY LONDON
EST 1894

- Views restrict access to the data because the view can display selected columns from the table.
- Views can be used to make simple queries to retrieve the results of complicated queries. For example, views can be used to query information from multiple tables without the user knowing how to write a join statement.
- Views provide groups of users access to data according to their particular criteria.
- Views provide data independence for ad hoc users and application programs. One view can be used to retrieve data from several tables.

Uses for SQL Views

CITY UNIVERSITY LONDON

- Security: hide columns and rows
- Display results of computations
- Hide complicated SQL syntax
- Provide a level of isolation between actual data and the user's view of data
- Assign different processing permissions to different users on same table(s)

Views

CITY UNIVERSITY LONDON
EST 1894

- A view is a relation defined in terms of stored tables (called base tables) and other views.
- Two kinds:
  - **Virtual** is not stored in the database; just a query for constructing the relation.
  - **Materialized** is actually constructed and stored.
- In databases, usually views are virtual by default. Materialized views are more complicated to create.

View definition in MySQL

CITY UNIVERSITY LONDON

```
CREATE [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = user]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain functions	No	Yes
Contain groups of data	No	Yes
DML writing operations through a view	Yes	Not always

Creating a View

CITY UNIVERSITY LONDON

Create the EMPVU80 view, which contains details of employees in department 80:

```
CREATE VIEW empvu80
AS SELECT  employee_id, last_name, salary
    FROM    employees
    WHERE   department_id = 80;
```

To access the data:

```
SELECT * FROM empvu80;
```

Creating a View

CITY UNIVERSITY LONDON

Create a view by using column aliases in the subquery:

```
CREATE VIEW salvu50
 AS SELECT  employee_id AS ID_NUMBER, last_name AS NAME,
            salary*12 AS ANN_SALARY
    FROM    employees
    WHERE   department_id = 50;
```

Creating a View

CITY UNIVERSITY LONDON

Another way to state aliases.

```
CREATE VIEW salvu50 (Id, Name, Salary)
 AS SELECT  employee_id, last_name,
            salary*12
    FROM    employees
    WHERE   department_id = 50;
```

Retrieve the data

CITY UNIVERSITY LONDON
EST 1894

```
SELECT * FROM salvu50;
```

```
+----+-------+--------+
| Id | Name  | Salary |
+----+-------+--------+
|  1 | Smith | 480000 |
|  2 | Child | 720000 |
+----+-------+--------+
```

Modifying a View

CITY UNIVERSITY LONDON
EST 1894

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT  employee_id, CONCAT(first_name, ' ', last_name),
    salary, department_id
   FROM    employees
   WHERE   department_id = 80;
```

Creating a Complex View

CITY UNIVERSITY LONDON

Create a complex view that contains group functions to display values from two tables:

```
CREATE VIEW dept_sum_vu  (name, minsal, maxsal, avgsal)
AS SELECT  d.department_name, MIN(e.salary),
           MAX(e.salary),AVG(e.salary)
   FROM    employees e, departments d
   ON      e.department_id = d.department_id
   GROUP_BY  d.department_name;
```

DML Writing Operations on a View

CITY UNIVERSITY LONDON

- For a view to be updatable, there must be a one-to-one relationship between the rows in the view and the rows in the underlying table.
- You can usually perform DML writing operations on simple views.
- You cannot update if the view contains the following:
  - Aggregate functions
  - A `GROUP BY` clause
  - The `DISTINCT` keyword
  - Subquery in the select list

DML Writing Operations on a View

CITY UNIVERSITY LONDON

You cannot add data through a view if the view includes:

- Aggregate functions
- A `GROUP BY` clause
- The `DISTINCT` keyword
- Subquery in the select list
- `NOT NULL` columns in the base tables that are not selected by the view

https://dev.mysql.com/doc/refman/8.4/en/view-updatability.html

Using CHECK OPTION Clause

CITY UNIVERSITY LONDON

You can ensure that DML writing operations performed on the view stay in the domain of the view by using the **WITH CHECK OPTION** clause:

```
CREATE OR REPLACE VIEW empvu20
AS SELECT  *
   FROM     employees
   WHERE    department_id = 20
   WITH CHECK OPTION;
```

Any attempt to change the department number for any row in the view fails because it violates the **WITH CHECK OPTION** constraint.

CHECK OPTION settings

CITY UNIVERSITY LONDON

- The `LOCAL` keyword restricts the `CHECK OPTION` only to the view being defined.
- `CASCADED` causes the checks for underlying views to be evaluated as well.
- When neither keyword is given, the default is `CASCADED`.

MERGE and TEMPTABLE Algorithms

CITY UNIVERSITY LONDON

- It affects how MySQL processes the view. `ALGORITHM` takes three values: `MERGE`, `TEMPTABLE`, or `UNDEFINED`.
- For `MERGE`, the text of a statement that refers to the view and the view definition are merged such that parts of the view definition replace corresponding parts of the statement.
- For `TEMPTABLE`, the results from the view are retrieved into a temporary table, which then is used to execute the statement.
- If no `ALGORITHM` clause is present, the default algorithm is determined by the value of the `derived_merge` flag of the `optimizer_switch` system variable.

https://dev.mysql.com/doc/refman/8.4/en/view-algorithms.html

Removing a View

CITY UNIVERSITY LONDON

You can remove a view without losing data because a view is based on underlying tables in the database.

```
DROP VIEW empvu80;
```

Materialized Views

CITY UNIVERSITY LONDON
EST 1894

- Sometimes it is good to store a result of the previous view so that it does not have to be computed each time.
- Some databases provide materialized view.
- In others one can use TRIGGER statement to do so.
- **Problem:** each time a base table changes, the materialized view may change.
  - Cannot afford to recompute the view with each change.
- **Solution:** Periodic reconstruction of the materialized view, which is otherwise “out of date”.

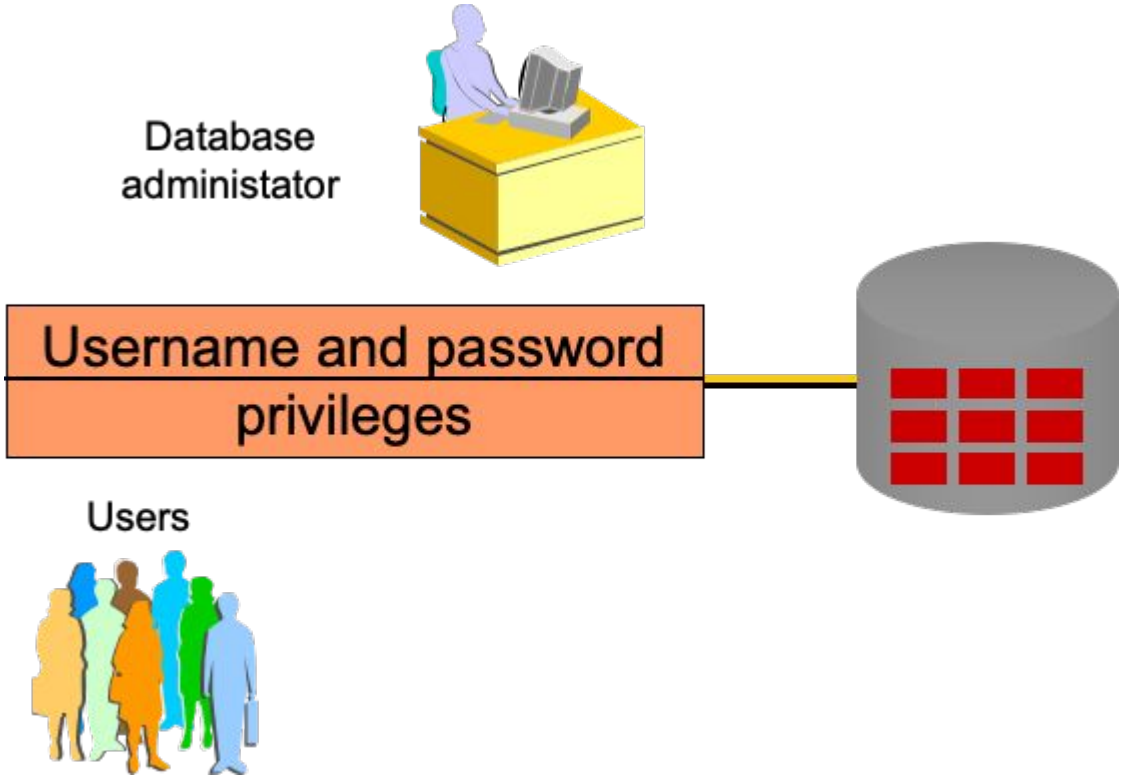
Example of materialized view usage

CITY UNIVERSITY LONDON

- A supermarket network stores every sale at every store in a database.
- Overnight, the sales for the day are used to update a data warehouse: which is the materialized views of the sales.
- The warehouse is used by analysts to predict trends and move goods to where they are selling best.

Controlling Access (Privileges)

Controlling User Access



Privileges

CITY UNIVERSITY LONDON

- Database security:
  - System security
  - Data security
- System privileges: Gain access to the database
- Object privileges: Manipulate the content of the database objects
- Schema: Collection of objects, such as tables, views, and sequences

Privileges

CITY UNIVERSITY LONDON

- The schema is owned by a database user and has the same name as that user.
- Users can also be given the privilege to grant additional privileges to other users or to roles, which are named groups of related privileges.
- The database administrator is a high-level user with the ability to grant users access to the database and its objects.

System Privileges

CITY UNIVERSITY LONDON

- Many (types of) privileges are available.
- The DBA has high-level system privileges:
  - Create new users
  - Remove users
  - Remove tables
  - Back up tables

Creating Users

CITY UNIVERSITY LONDON

- The DBA creates users by using the CREATE USER statement.

```
CREATE USER      user
IDENTIFIED BY  password;
```

- The user does not have any privileges at this point.
- The DBA can then grant a number of privileges to that user.
- These privileges determine what the user can do at the database level.

https://dev.mysql.com/doc/refman/8.4/en/create-user.html

User System Privileges

CITY UNIVERSITY LONDON

Once a user is created, the DBA can grant specific system privileges to a user.

```
GRANT privilege [, privilege...]
TO user [, user...];
```

An application developer may have the following system privileges:

- ALL
- CREATE TABLE
- CREATE VIEW
- CREATE ROUTINE

https://dev.mysql.com/doc/refman/8.4/en/grant.html

Granting System Privileges. Example.

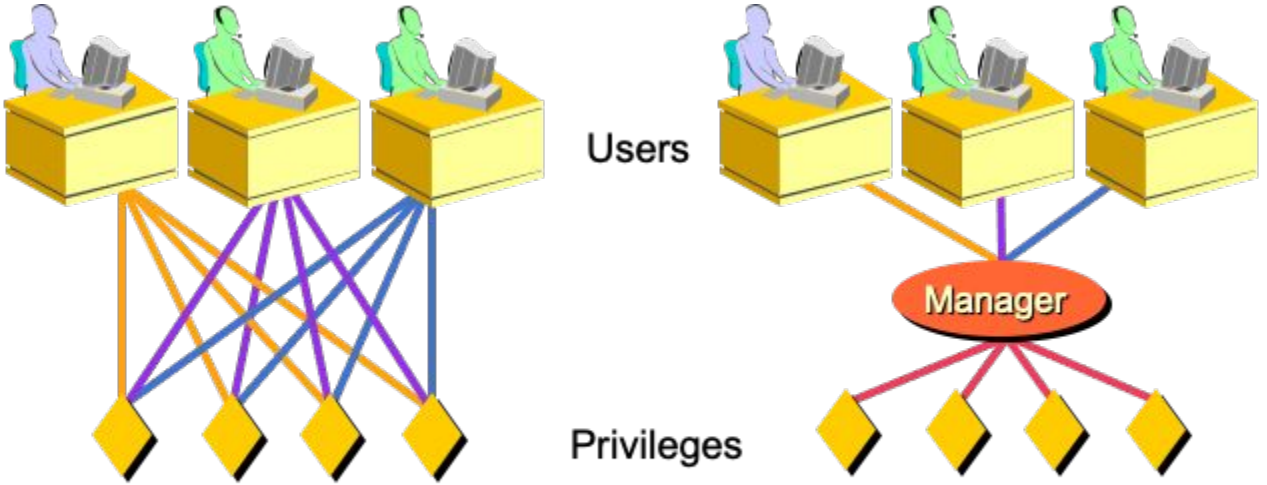
CITY UNIVERSITY LONDON

The DBA can grant a user specific system privileges.

```
GRANT   create table, create view
TO      scott;
```

What Is a Role?

CITY UNIVERSITY LONDON
EST 1894



Allocating privileges without a role

Allocating privileges through a role

Roles

CITY UNIVERSITY LONDON

- Are named groups of related privileges
- Can be granted to users
- Simplify the process of granting and revoking privileges
- Are created by a DBA

Privileges and a Role. Statements.

CITY UNIVERSITY LONDON

```
CREATE ROLE manager;

GRANT create table, create view
  TO manager;

GRANT manager TO BLAKE, CLARK;
```

Changing Your Password

CITY UNIVERSITY LONDON

The DBA creates your user account and initializes your password.

You can change your password by using the ALTER USER statement.

```
ALTER USER scott
  IDENTIFIED BY lion;
```

Object Privileges

CITY UNIVERSITY LONDON
EST 1894

Object Privilege	Table	View	Routine
ALTER	X		
DELETE	X	X	
EXECUTE			X
INSERT	X	X	
REFERENCES	X		
SELECT	X	X	
UPDATE	X	X	

Object Privileges

CITY UNIVERSITY LONDON

Object privileges vary from object to object.

An owner has all the privileges on the object.

An owner can give specific privileges on that owner's object.

```
GRANT   priv_type [(columns)]
ON      object
TO      {user|role}
[WITH GRANT OPTION];
```

Granting Object Privileges

CITY UNIVERSITY LONDON

Grant query privileges on the EMP table.

```
GRANT select ON emp TO sue, rich;
```

Grant privileges to update specific columns to users and roles

```
GRANT update (dname, loc) ON dept TO scott, manager;
```

Using WITH GRANT OPTION

CITY UNIVERSITY LONDON

Give a user authority to pass along the privileges.

```
GRANT select, insert
ON    dept
TO    scott
WITH GRANT OPTION;
```

How to Revoke Object Privileges

CITY UNIVERSITY LONDON
EST 1894

You use the REVOKE statement to revoke privileges granted to other users.

```
REVOKE [IF EXISTS]
    priv_type [(column_list)]  [, priv_type [(column_list)]]
    ON object
    FROM {user|role} [, {user|role}]
```

https://dev.mysql.com/doc/refman/8.4/en/revoke.html

By SQL99 standard there should also be REVOKE CASCADE.

In this case, privileges granted to others through the WITH GRANT OPTION will also be revoked.

Revoking Object Privileges

CITY UNIVERSITY LONDON

As user Alice, revoke the SELECT and INSERT privileges given to user Scott on the DEPT table.

```
REVOKE select, insert
ON     dept
FROM   scott;
```

Summary

CITY UNIVERSITY LONDON
EST 1894

Statement	Action
`CREATE USER`	Allows the DBA to create a user
`GRANT`	Allows the user to give other users privileges to access the user's objects
`CREATE ROLE`	Allows the DBA to create a collection of privileges
`ALTER USER`	Allows users to change their password
`REVOKE`	Removes privileges on an object from users

Example

Suppose that the DBA creates four accounts

`A1, A2, A3, A4`

and wants only A1 to be able to create base relations. Then the DBA must issue the following GRANT command in SQL

```
GRANT CREATE TABLE TO A1;
```

Example

CITY UNIVERSITY LONDON
EST 1894

- Suppose that A1 creates the two base relations EMPLOYEE and DEPARTMENT
  - A1 is then owner of these two relations and hence all the relation privileges on each of them.
- Suppose that A1 wants to grant A2 the privilege to insert and delete tuples in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

```
GRANT INSERT, DELETE ON
EMPLOYEE, DEPARTMENT TO A2;
```

Example

- Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.
- A1 can issue the command:

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT
             TO A3 WITH GRANT OPTION;
```

- A3 can grant the SELECT privilege on the EMPLOYEE relation to A4 by issuing:

```
GRANT SELECT ON EMPLOYEE TO A4;
```

- Note that A4 can’t propagate the SELECT privilege because GRANT OPTION was not given to A4.

Example

Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

It will not revoke privileges of A4, but if there was CASCADE possible, then it will.

Example

- Suppose that A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege.
- The limitation is to retrieve only the NAME, BDATE, and ADDRESS attributes and only for the tuples with DNO=5.
- A1 then create the view:

```
CREATE VIEW A3EMPLOYEE AS
     SELECT NAME, BDATE, ADDRESS
     FROM EMPLOYEE
     WHERE DNO = 5;
```

- After the view is created, A1 can grant SELECT on the view A3EMPLOYEE to A3 as follows:

```
GRANT SELECT ON A3EMPLOYEE TO A3
     WITH GRANT OPTION;
```

Example

- Finally, suppose that A1 wants to allow A4 to update only the SALARY attribute of EMPLOYEE.
- A1 can issue:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;
```

Summary

Views

CITY UNIVERSITY LONDON

- A virtual table which is the result of an SQL query
- The SQL query is executed every time the view is invoked
- Once a view has been created it can be used as if it is a base table

Example view

CITY UNIVERSITY LONDON

Grounds south of the river Tyne

```
CREATE VIEW mccGrounds_south
AS
    (SELECT *
    FROM mccGround
    WHERE g_town in ('Durham', 'Sunderland'));

SELECT * FROM mccGrounds_south;
```