Views, Triggers & Stored Procedures

CO226 : Database Systems
Lab - 8

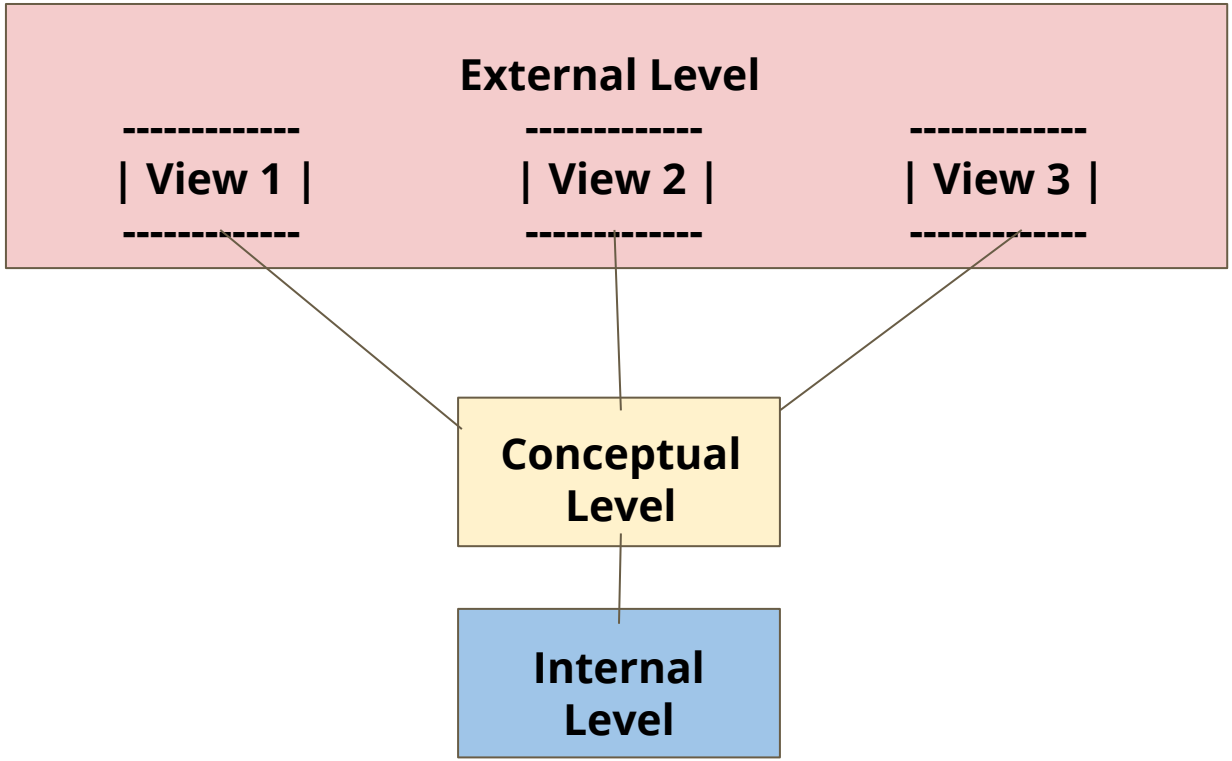
Kanchana Jayasinghe
kc43224jayasinghe@eng.pdn.ac.lk

MySQL Views

Outline

- Database Architecture
- What are User Views ?
- Characteristics of Views
- Creating a View
- Displaying Results of a View
- Removing a View
- Benefits of User Views

Database Architecture



Database Architecture cont.

- Conceptual Level
  - The conceptual level is a logical representation of the entire database content.
  - The conceptual level consists with base tables.
  - Base tables are real tables which contain physical records.

Database Architecture cont.

Department

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857

Base Tables in Conceptual Level

Employee

Emp_No	Emp_Name	Designation	DOB	Dept
179	Silva	Manager	12-05-74	SAL
857	Perera	Accountant	01-04-67	FIN
342	Dias	Programmer	25-09-74	SAL

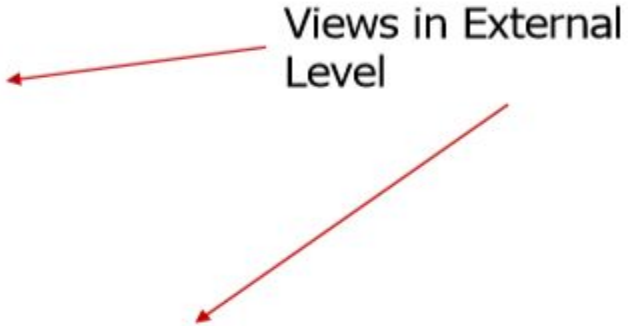
Database Architecture cont.

- External Level
  - The external model represents how data is presented to users.
  - It is made up of views.
  - Views are virtual tables; they do not exist in physical storage, but appear to a user as if they did.

Database Architecture cont.

Department_View

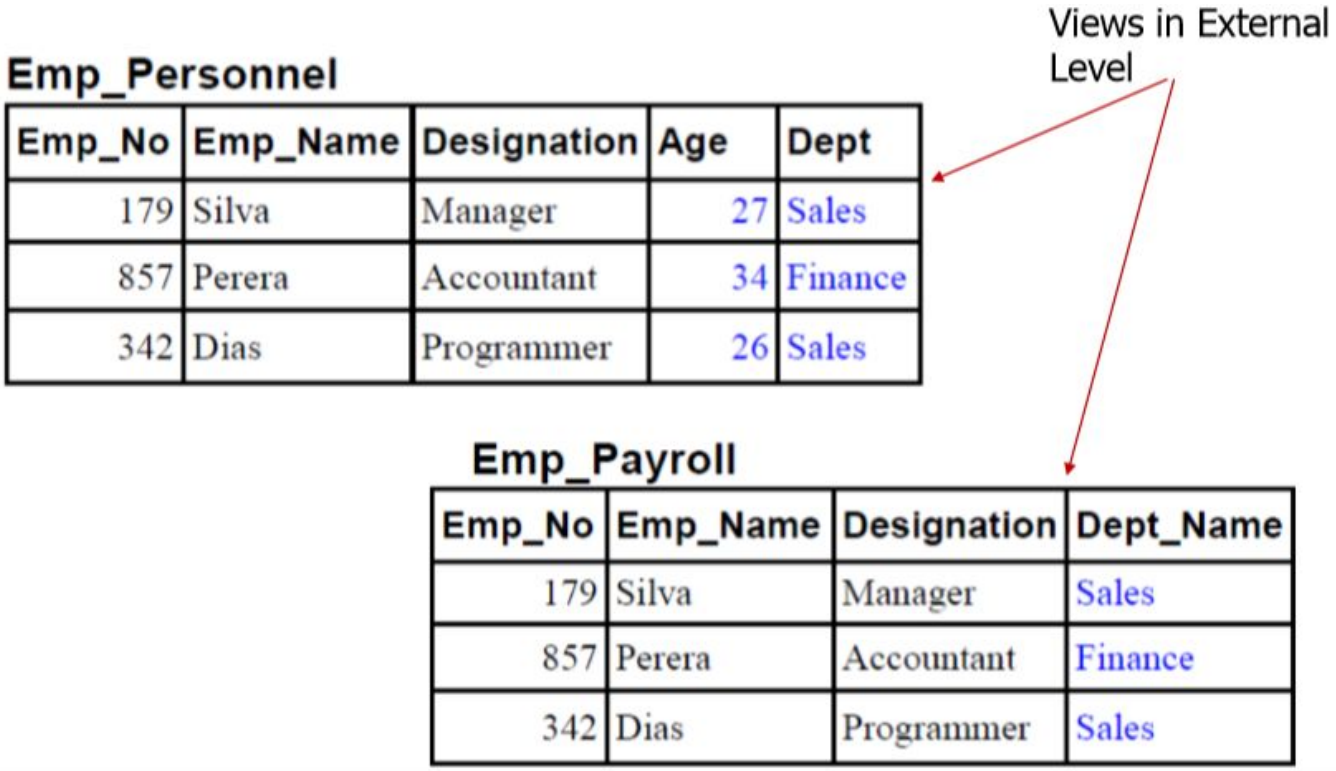
Dept_Code	Dep_Name	Manager
SAL	Sales	Silva
FIN	Finance	Perera



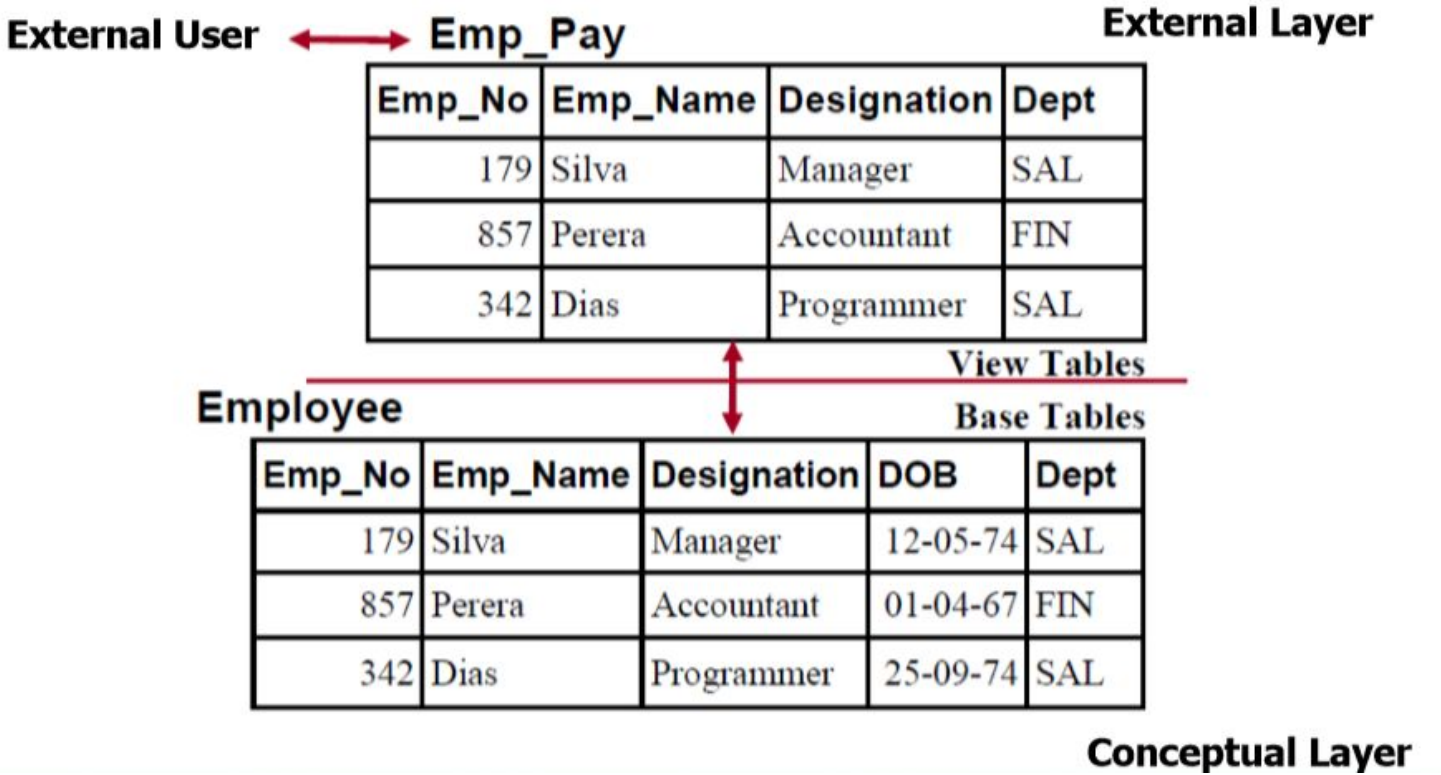
Employee_View

Emp_No	Emp_Name	Designation	Age	Dept
179	Silva	Manager	27	SAL
857	Perera	Accountant	34	FIN
342	Dias	Programmer	26	SAL

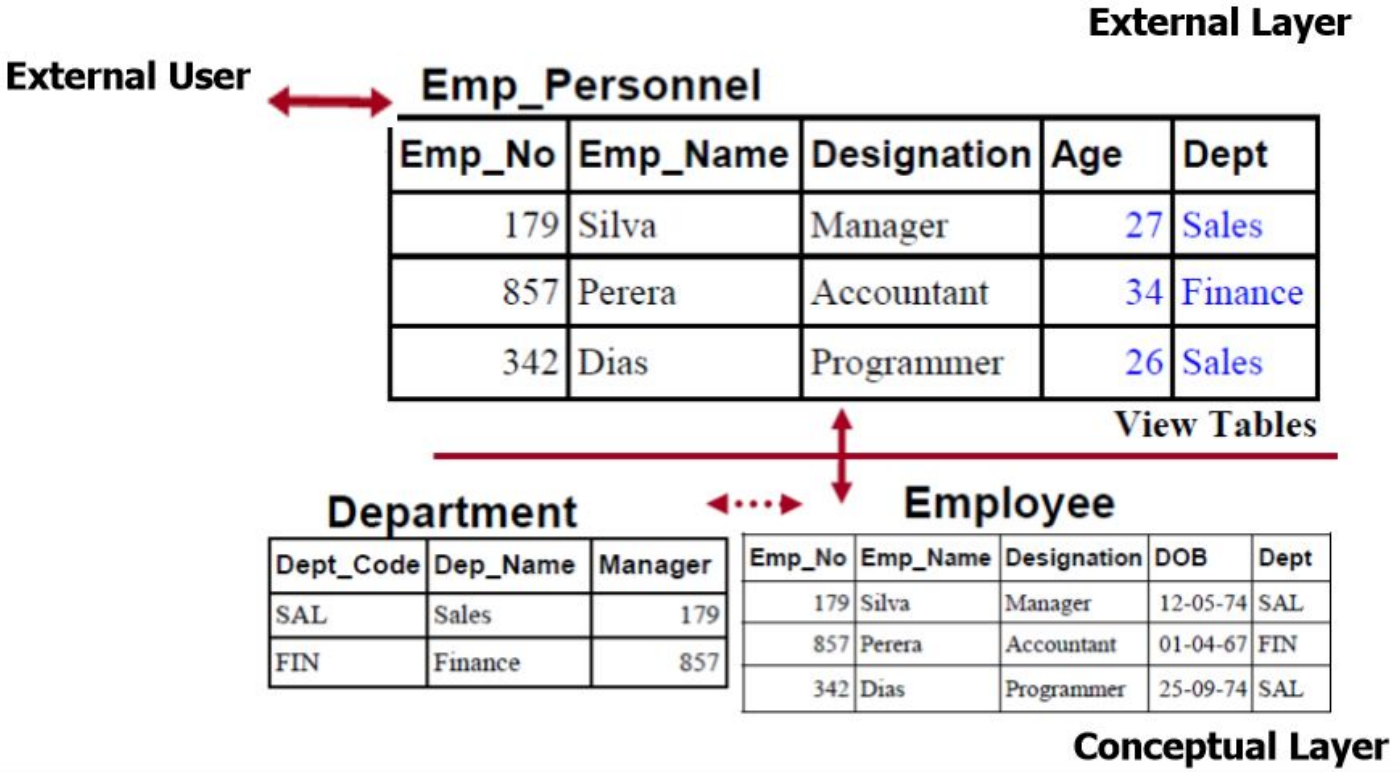
Database Architecture cont.



Database Architecture cont.



Database Architecture cont.



What are User Views ?

- User views
  - A view is a “Virtual Table”
  - Derived or virtual tables that are visible to users
  - Do not occupies any storage space

- Base Tables
  - Store actual rows of data
  - Occupies a particular amount of storage space

Characteristics of Views

- Behave as if it contains actual rows of data, but in fact it does not contain data
- Rows are derived from base table or tables from which the view is defined
- Being virtual tables the possible update operations that can be applied to views are limited
- However, it does not provide any limitations on querying a view

Creating a View

- Syntax:

  CREATE VIEW view_name (List of attribute names,...)
  AS query

- Column names/attributes specified must have the same number of columns derived from the query.
- Data definitions for each column are derived from the source table.
- Columns will assume corresponding column names in the source table.
- Names must be specified for calculated or identical columns.

Example 1

Works__On View

Fname	Lname	Pname	Hours

```
CREATE VIEW Works_On
AS
SELECT Fname, Lname, Pname, Hours
FROM Employee, Project, Works_On
WHERE Essn = Empid
        AND Pno = Pnumber ;
```

Example 2

Dept_Info

DeptName	No_of_Emps	Salary

```
CREATE VIEW Dept_Info (Dept_Name, No_Of_Emps, Salary)
AS
SELECT Dname, COUNT(*), SUM(Salary)
FROM Department, Employee
WHERE Dnumber = Dno
GROUP BY  Dname;
```

Displaying Results of a View

- Syntax:

  SELECT List of attributes
  FROM view_name
  WHERE condition;

- Same as we are retrieving data from a table.

Removing a View

- Syntax:

  DROP VIEW view_name ;

- Eg:

  DROP VIEW Dept_Info;

- Removes only the definition of the view table.
- Data that it used to retrieve is not affected.

Benefits of User Views

1. Security
   - Protect data from unauthorized access.
   - Each user is given permission to access the database via only a small set of views that contain specific data the user is authorized to see.
2. Query Simplicity
   - Turning multiple table queries to single table queries against views, by drawing data from several tables.
   - It provides flexible and powerful data access capabilities.

Benefits of User Views cont.

3. Natural Interface
    - Personalized view of database structure can be created that make sense for the user.
    - Also it is possible to restructure the way in which tables are seen.
    - So that different users see it from different perspectives, thus allowing more natural views of the same enterprise

Benefits of User Views cont.

4. Insulation from Change
    - Data independence - maintain independence among different user views and between each user view and the physical constructs.
    - A view can present a consistent image of the database structure, even if the underlying source tables are restructured.

MySQL Triggers

Outline

- What is a Trigger?
- Types of Triggers
- Create Trigger in MySQL
- Displaying a Trigger
- Drop a Trigger
- Advantages of Triggers
- Disadvantages of Triggers

What is a Trigger ?

- A trigger is a stored program invoked automatically in response to an event such as **insert**, **update**, or **delete** that occurs in the associated table.
- For example,

  you can define a trigger that is invoked automatically before a new row is inserted into a table.

- MySQL supports triggers that are invoked in response to the **INSERT**, **UPDATE** or **DELETE** event.

Types of Triggers

The SQL standard defines two types of triggers: row-level triggers and statement-level triggers.

- A **row-level trigger** is activated for each row that is inserted, updated, or deleted.

  For example, if a table has 100 rows inserted, updated, or deleted, the trigger is automatically invoked 100 times for the 100 rows affected.

- A **statement-level trigger** is executed once for each transaction regardless of how many rows are inserted, updated, or deleted.

* * **MySQL supports only row-level triggers**. It doesn't support statement-level triggers.

Create Trigger in MySQL

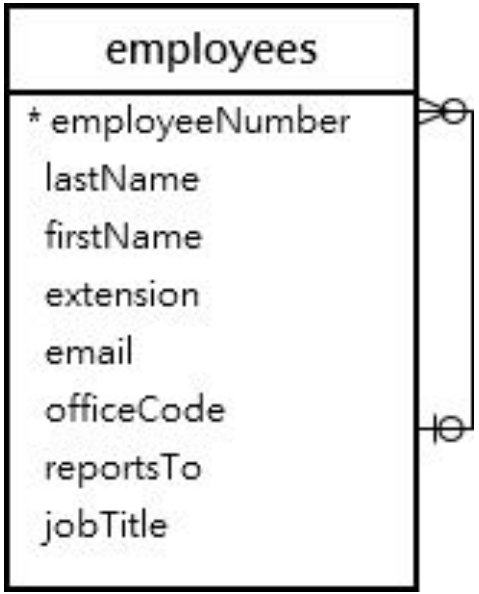
- The CREATE TRIGGER statement creates a new trigger.

  Syntax :

```
mysql > Delimiter//
mysql > CREATE TRIGGER trigger_name
        {BEFORE | AFTER} {INSERT | UPDATE| DELETE }
        ON table_name FOR EACH ROW
        Trigger_body;
mysql > Delimiter;
```

Example

1. Create a trigger in MySQL to log the changes of the **employees** table.



- First, create a new table named **employees_audit** to keep the changes to the **employees** table:

```
CREATE TABLE employees_audit (
    id INT AUTO_INCREMENT PRIMARY KEY,
    employeeNumber INT NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    changedat DATETIME DEFAULT NULL,
    action VARCHAR(50) DEFAULT NULL
    );
```

- Next, create a **BEFORE UPDATE** trigger that is invoked before a change is made to the **employees** table.

```
mysql> Delimiter //
mysql> CREATE TRIGGER before_employee_update
            BEFORE UPDATE ON employees
            FOR EACH ROW
        INSERT INTO employees_audit
        SET action = 'update',
            employeeNumber = OLD.employeeNumber,
            lastname = OLD.lastname,
            changedat = NOW();
mysql> Delimiter;
```

- Then, show all triggers in the current database by using the SHOW TRIGGERS statement:

  Syntax :

  SHOW TRIGGERS;

	Trigger	Event	Table	Statement	Timing
▶	before_employee_update	UPDATE	employees	INSERT INTO employees_audit SET action = 'update', employeeNumber = OLD.employeeNumber, lastname = OLD.lastname, changedat = NOW()	BEFORE

- After that, update a row in the **employees** table:

```
UPDATE employees
SET
    lastName = 'Petter'
WHERE
    employeeNumber = 1056;
```

- Finally, query the **employees_audit** table to check if the trigger was fired by the **UPDATE** statement:

```
SELECT * FROM employees_audit;
```

- The following shows the output of the query:

	id	employeeNumber	lastname	changedat	action
▶	1	1056	Patterson	2020-05-15 15:38:30	update

Displaying a Trigger

- To display all the Triggers in the current database we use **SHOW TRIGGERS** syntax.

  Syntax :

  SHOW TRIGGERS;

Drop a Trigger

- To drop a Trigger in MySQL we use **DROP TRIGGER** syntax.

  Syntax :

  DROP TRIGGER [trigger_name];

- Ex :

  To drop the before_employee_update trigger in previous example:

  DROP TRIGGER before_employee_update;

Advantages of Triggers

- Triggers provide another way to check the integrity of data.
- Triggers handle errors from the database layer.
- Triggers give an alternative way to **run scheduled tasks**. By using triggers, you don't have to wait for the **scheduled events** to run because the triggers are invoked automatically *before* or *after* a change is made to the data in a table.
- Triggers can be useful for auditing the data changes in tables.

Disadvantages of Triggers

- Triggers can only provide extended validations, not all validations. For simple validations, you can use the **NOT NULL**, **UNIQUE**, **CHECK** and **FOREIGN KEY** constraints.
- Triggers can be difficult to troubleshoot because they execute automatically in the database, which may not invisible to the client applications.
- Triggers may increase the overhead of the MySQL Server.

Stored Procedures

Outline

- Introduction to Stored Procedure
- Create Procedure
- CALL Statement
- Advantages of Stored Procedures
- Disadvantages of Stored Procedures

Introduction to Stored Procedure

- The following **SELECT** statement returns all rows in the table customers from the database:

```
SELECT
        customerName,
        city,
        state,
        postalCode,
        Country
FROM
        Customers
ORDER BY customerName;
```

- When you use MySQL Workbench or mysql shell to issue the query to MySQL Server, MySQL processes the query and returns the result set.
- If you want to save this query on the database server for execution later, one way to do it is to use a stored procedure.

Create Procedure

- The following **CREATE PROCEDURE** statement creates a new stored procedure that wraps the query.

  Syntax :

```
mysql> DELIMITER //

mysql> CREATE PROCEDURE procedureName()
       BEGIN
              Query you want to execute;
       END //

mysql> DELIMITER ;
```

Example

- By definition, a stored procedure is a segment of declarative SQL statements stored inside the MySQL Server. In this example, we have just created a stored procedure with the name GetCustomers().

```
mysql>  DELIMITER //

mysql>  CREATE PROCEDURE GetCustomers()
            BEGIN
                SELECT
                    customerName,
                    city,
                    state,
                    postalCode,
                    country
                FROM
                    customers
                ORDER BY customerName;
            END//
 mysql>  DELIMITER ;
```

CALL Statement

- Once you save the stored procedure, you can invoke it by using the **CALL** statement:

  Syntax:

  CALL procedureName();

- And the statement returns the same result as the query.

  Ex:

  CALL GetCustomers();

Advantages of Stored Procedures

- Reduce network traffic
- Centralize business logic in the database
- Make database more secure

Disadvantages of Stored Procedures

- Resource usages
- Troubleshooting
- Maintenances

Summary

- MySQL Views
- MySQL Triggers
- Stored Procedures