Views & Stored Procedures

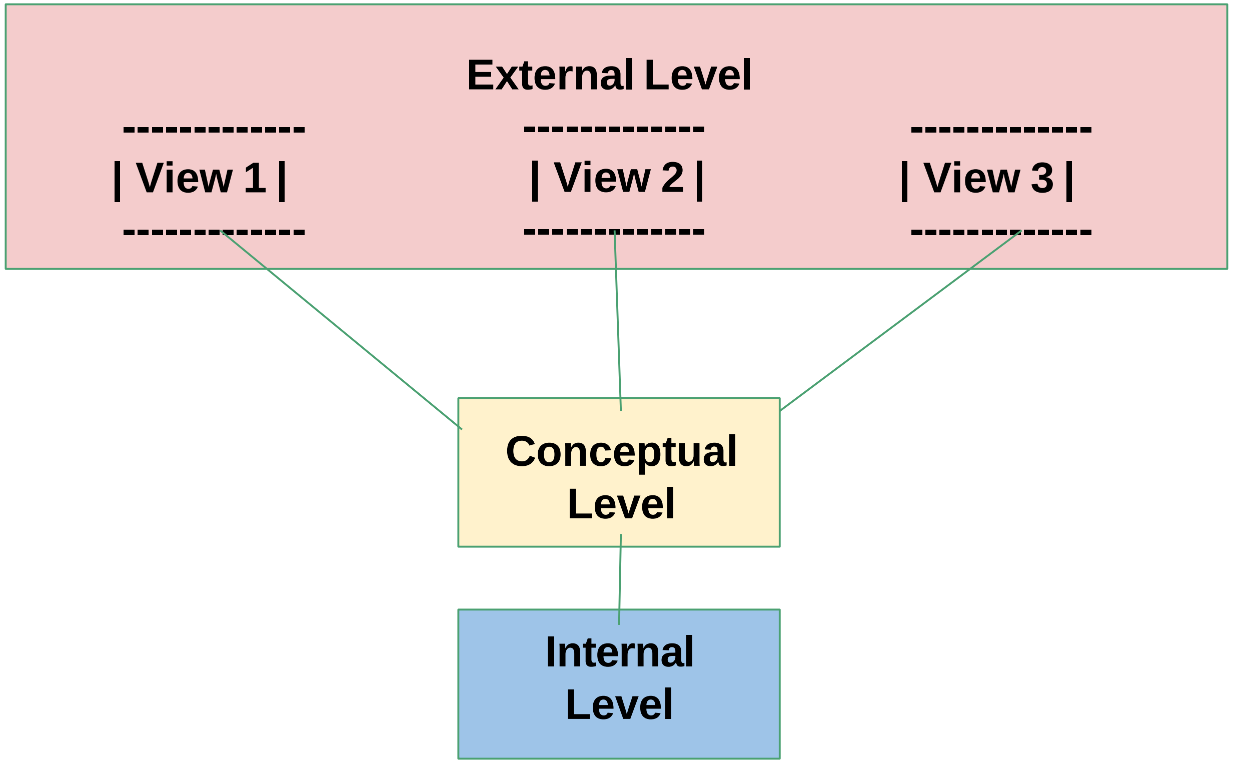
CO226 : Database Systems

Lab - 7

Kalani Udayanthi
kalaniu@eng.pdn.ac.lk

MySQL Views

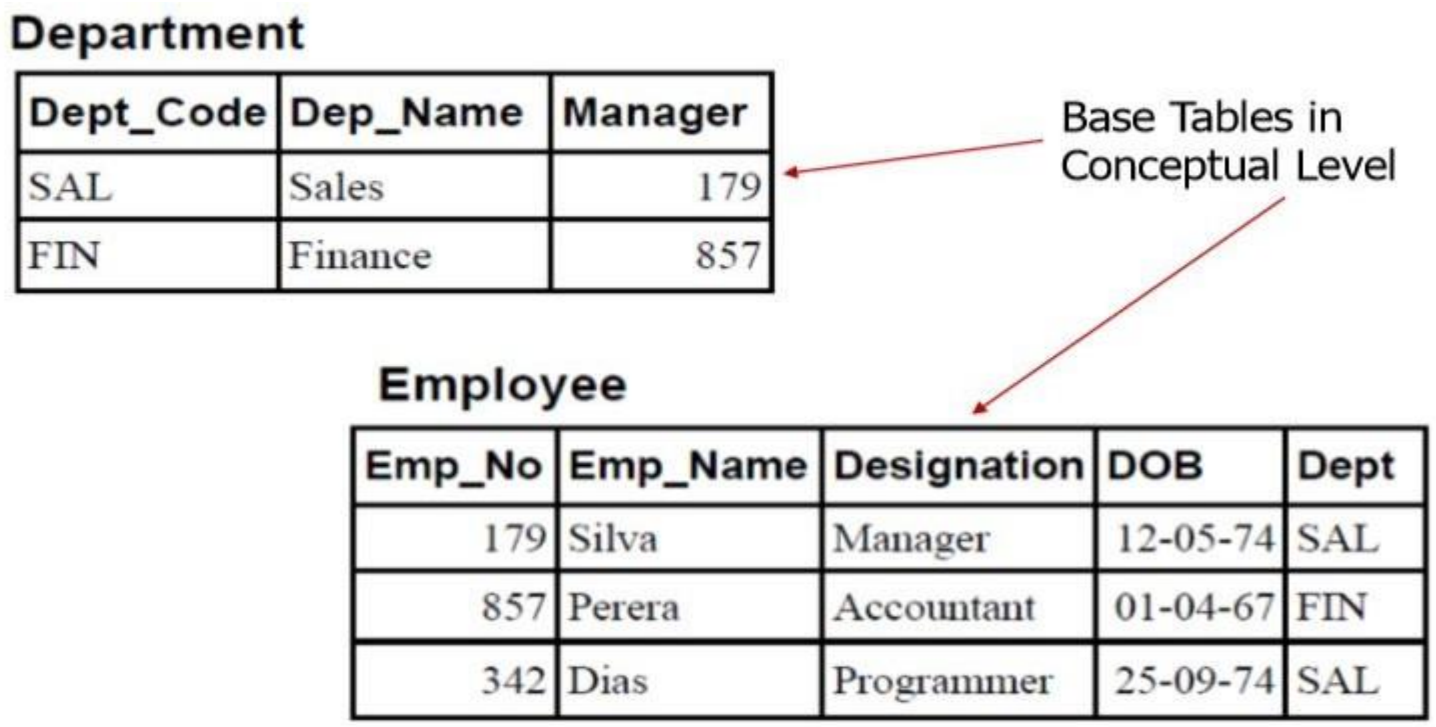
Database Architecture



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

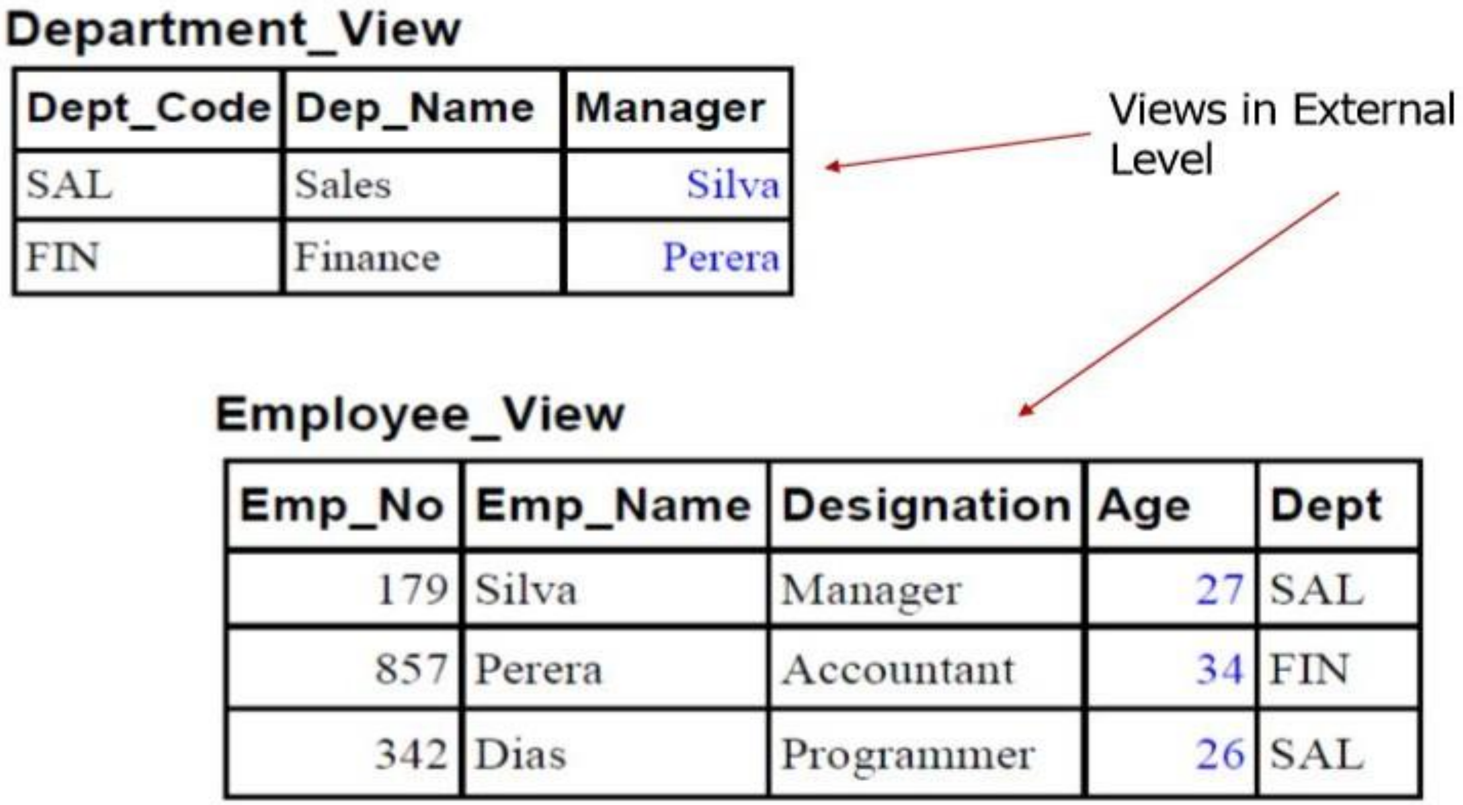
Employee

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

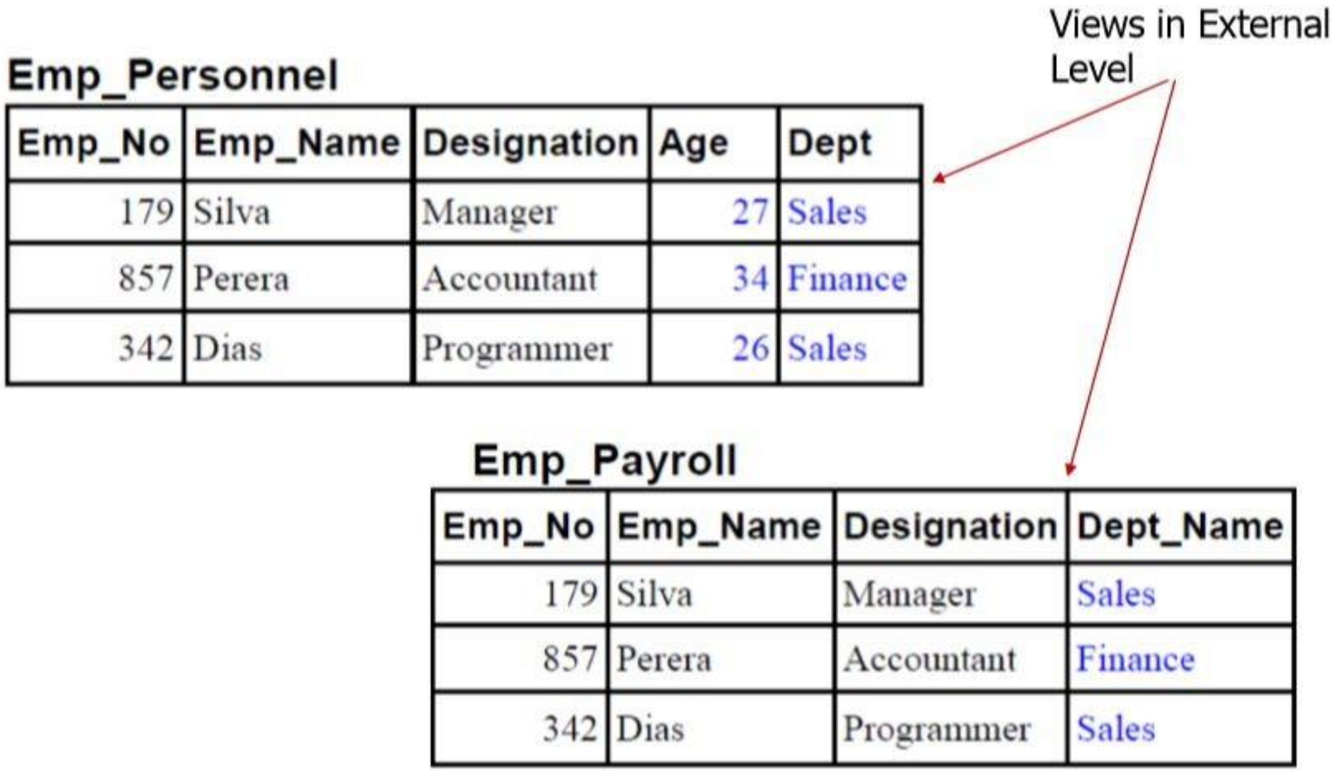
Dept_Code	Dep_Name	Manager
SAL	Sales	Silva
FIN	Finance	Perera

Views in External Level

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.


Views in External Level

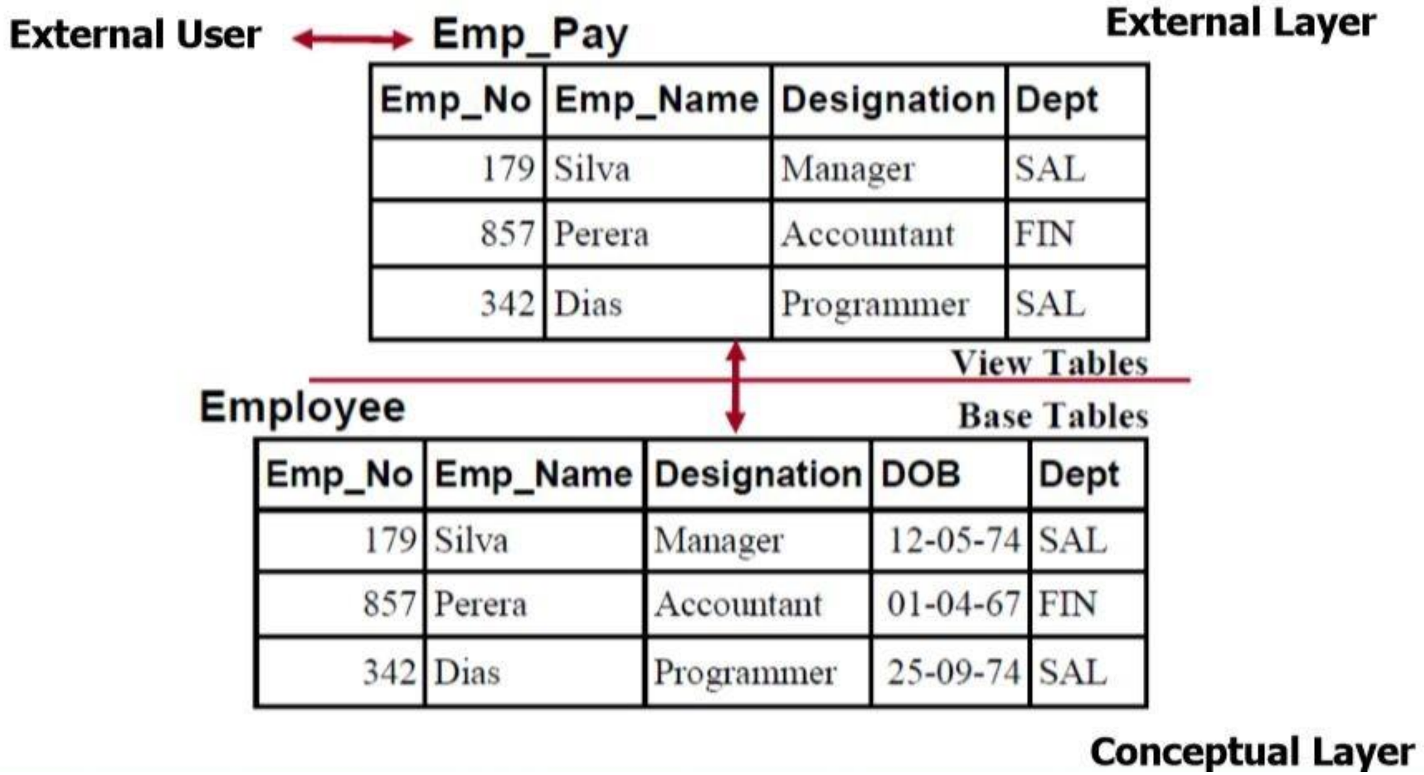
Emp_Personnel

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

Emp_Payroll

Emp_No	Emp_Name	Designation	Dept_Name
179	Silva	Manager	Sales
857	Perera	Accountant	Finance
342	Dias	Programmer	Sales

Database Architecture cont.



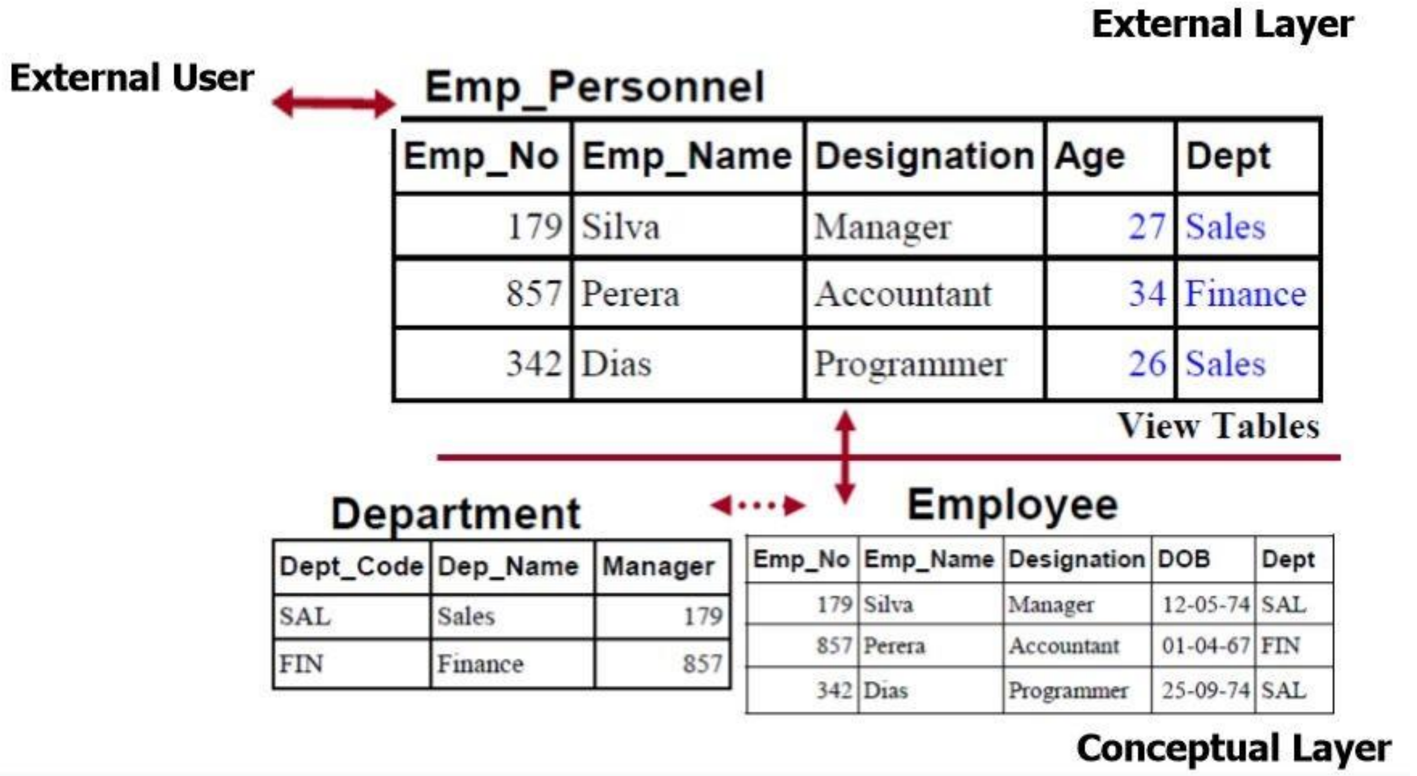
Emp_Pay

Emp_No	Emp_Name	Designation	Dept
179	Silva	Manager	SAL
857	Perera	Accountant	FIN
342	Dias	Programmer	SAL

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.



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

Dept_Code	Dep_Name	Manager
SAL	Sales	179
FIN	Finance	857

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

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

Why do we need Views in SQL Server?

- To protect the data. You can grant permissions to a view without allowing users to query the original tables.
- A view is a logical table but what it stores internally is a select statement that is used for creating the view.
- Simply we can say that view will act as an interface between the data provider (Table) and the User.
- A view is created based on a table any changes that are performed on the table reflect into the view any changes performed on the view reflected in the table also.

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

Differences between a table and a view?

When compared with a table we have the following differences between a table and view.

1. The table is physical and the view is logical
2. A table is an independent object whereas view is a dependent object that is a view depends on a table or tables from which it is loading the data.
3. When a new table is created from an existing table the new and old tables are independent themselves that is the changes of one table will not be reflected into the other table whereas if a view is created based on a table any changes that are performed on the table reflects into the view and any changes performed on the view reflected in the table also.

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.

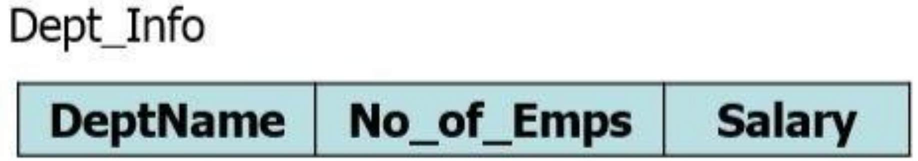
Example 1

Works__On View

Fname	Lname	Pname	Hours

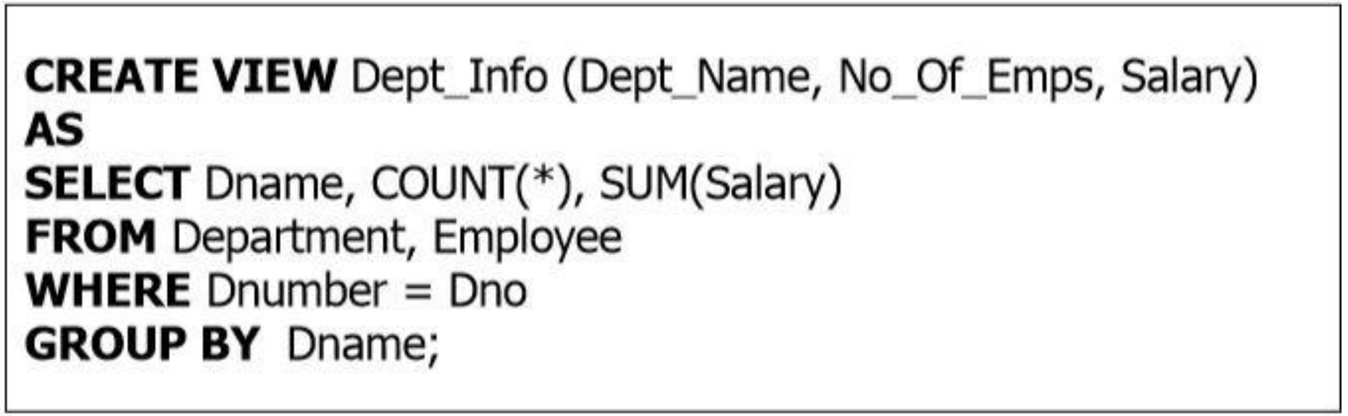
```
CREATE VIEW Works_On
AS
SELECT Fname, Lname, Pname, Hours
FROM Employee, Project, Works_On
WHERE Essn = Empid
          AND Pno = Pnumber ;
```

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.

Frequently asked questions on Views...

1. Can we drop a table that has dependent views on it?

   Yes, we can drop a table even if any dependent views are associated with it, but the views that are associated with it will not be dropped. They will still execute in the database only with the status as inactive object and all those views become active and start functioning provided the table is recreated.

2. Can we create a view based on other views?

   Yes, we can create a view based on other views. Usually, we create views based on tables, but it is also possible to create views based on views.

3. Can we update the views?

   Yes, views can be updated. However, updating a view that is based on multiple tables, may not update the underlying tables correctly. To correctly update a view that is based on multiple tables we can make use “Instead OF triggers” in SQL Server.

Limitations of Views

- We cannot pass parameters to a view.
- Rules and Defaults cannot be associated with views.
- The ORDER BY clause is invalid in views unless TOP or FOR XML is also specified.
- Views cannot be based on temporary tables.

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.

What is a Stored Procedure?

- A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
- So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
- You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Create a Stored 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
- Stored Procedures