

Database Management System

IT 261

Bachelor of Information Technology

Prepared by Dr. Khaled Eskaf

Lecture 1





Learning Objectives

This course is centered on the comprehensive study and application of relational database management systems (RDBMS), with a specific focus on analysis, design, normalization, and implementation techniques. It's tailored to provide a robust understanding of how relational databases function and how they can be effectively utilized and managed using MySQL, a popular open-source relational database management system.

GRADING

Quiz /week = 8 Marks = 8X10 weeks = 80%

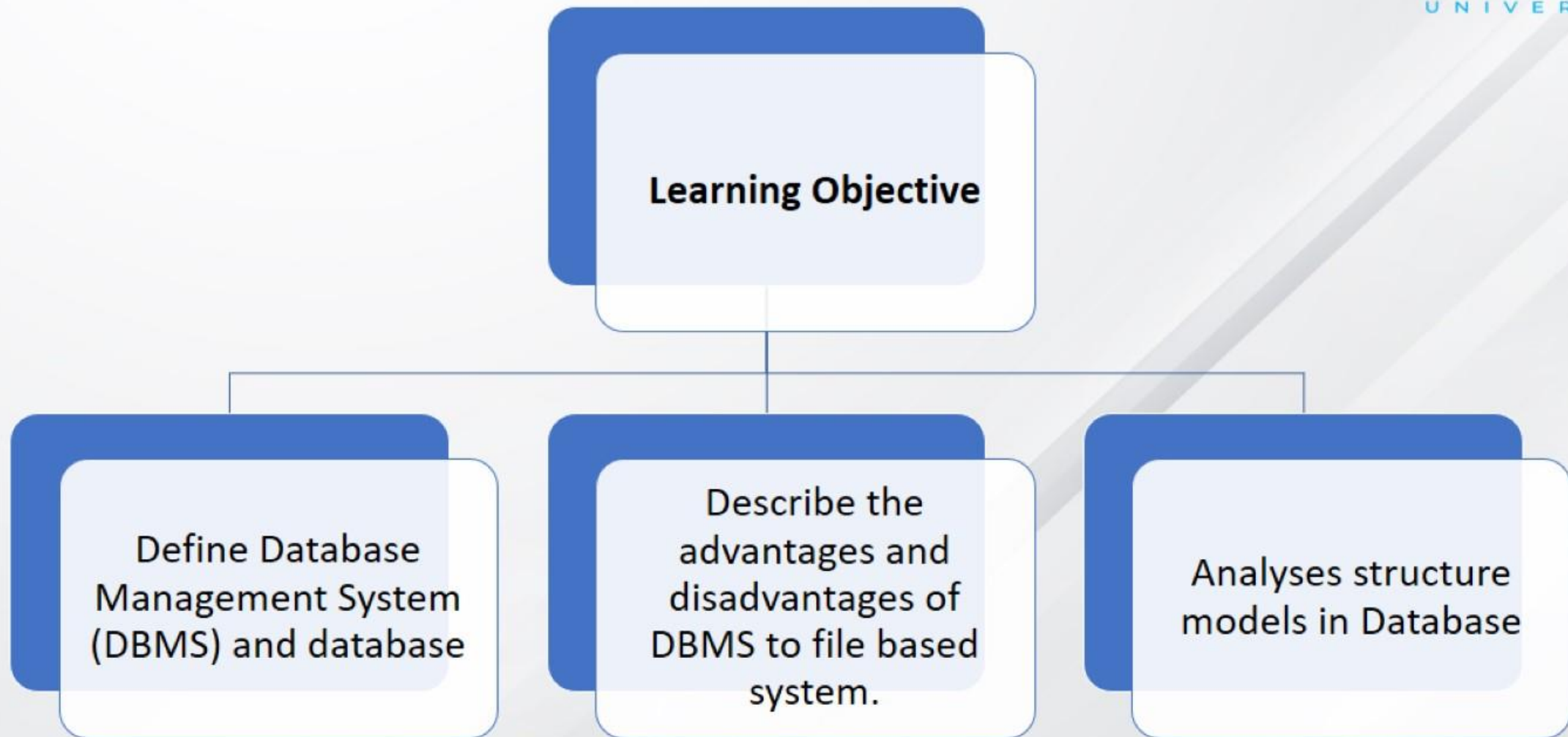
Final Exam: 20%

By the end of this course, you will not only understand the theoretical aspects of database management but will have also navigated the practical challenges and solutions in MySQL.

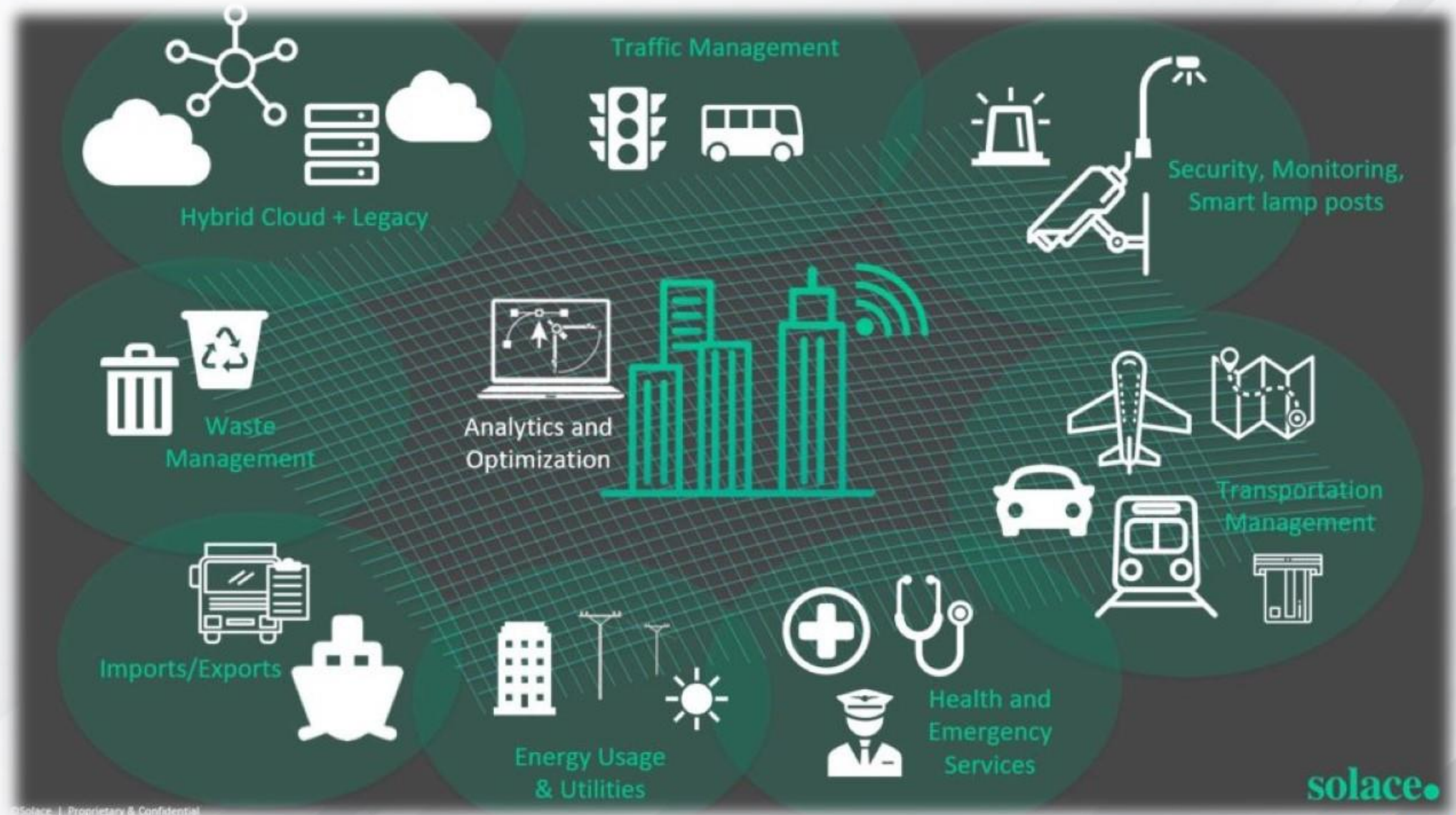


Course Outlines

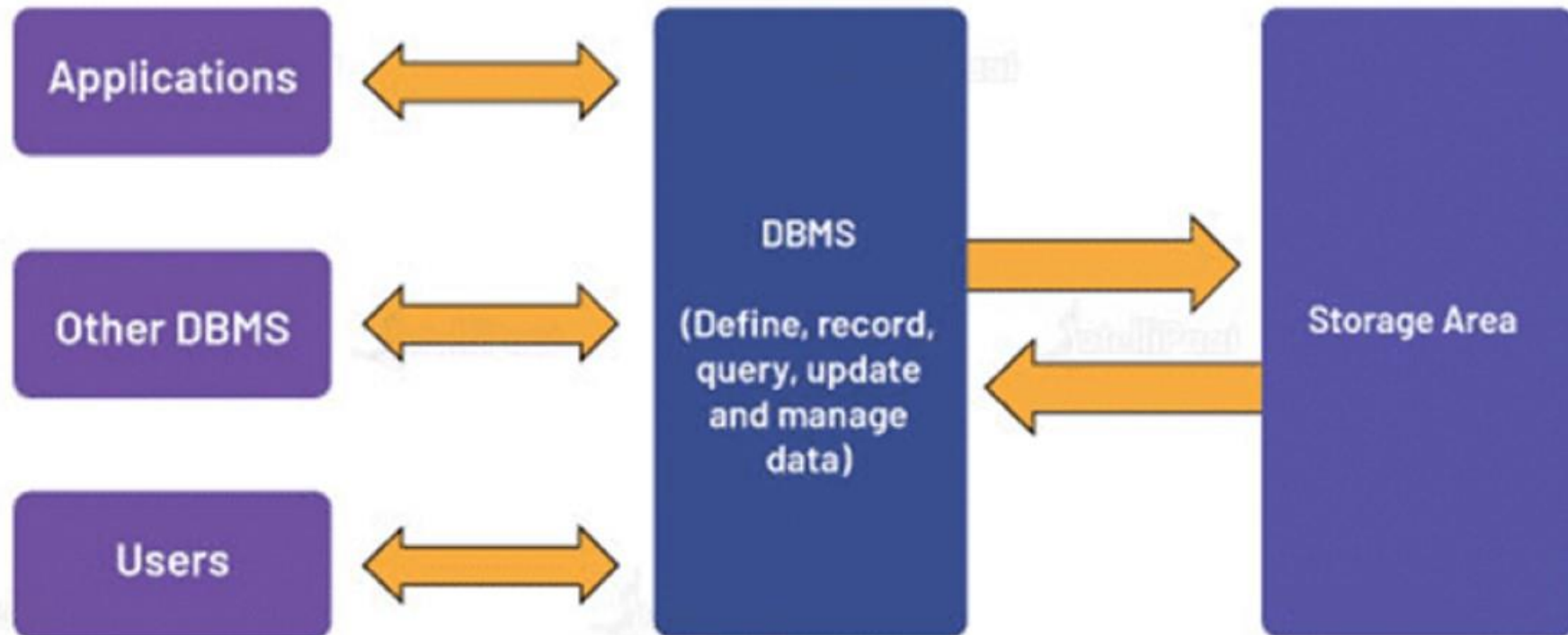
Week	Tonic	Obiective	MySQL. Practical Activities
1	Introduction to Database Concepts	Understand basic database concepts.	Install MySQL, explore the interface, create a basic database.
2	Relational Data Model	Learn about the relational data model.	Create relational tables in MySQL.
3	Relational Integrity Rules	Understand relational integrity constraints.	Implement primary and foreign key constraints in MySQL.
4	Relational Algebra and Basic SQL	Explore relational algebra and basic SQL commands.	Use basic SQL queries in MySQL.
5	Advanced SQL Queries	Delve into complex SQL operations.	Execute advanced SQL commands in MySQL.
6	Introduction to Database Normalization	Begin learning database normalization principles.	Apply the first normal form (1NF) in MySQL.
7	Advanced Database Normalization	Explore advanced normalization techniques.	Normalize a database in MySQL to 2NF and 3NF.
8	Data Modeling and ER Diagrams	Understand data modeling and ER diagrams.	Create an ER diagram and implement it in MySQL.
9	Database Design for Real-World Apps	Apply database design to real-world scenarios.	Develop a database schema based on a real-world example.
10	Project and Review	Apply all learned concepts in a comprehensive project.	Complete a project in MySQL incorporating all learned skills.



Smart City Data Base Management



Database Management System



Definitions

Database:

- A very large, integrated collection of data..
- Models real-world enterprise.
- Entities (e.g., students, courses)
 - Relationships (Ex., Eva is taking Biology)

Roll No.	Name	Course	
CS08	Steive	Comp. Sci.	→ Entity 1
EE54	Jhoson	Electronics	→ Entity 2
B12	Eva	Biology	→ Entity 3
F32	Jhoson	Finance	→ Entity 4
M26	Erica	Maths	→ Entity 5

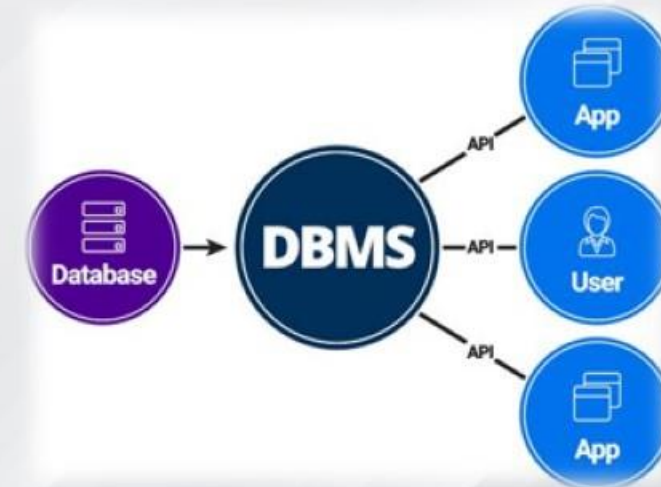
Student Table

Database Management System (DBMS)

a software package designed to store and manage databases.

• Examples of Database Applications:

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades



Purpose of Database Systems

❑ In the early days, database applications were built directly on top of file systems

❑ Drawbacks of using file systems to store data:

- Data redundancy and inconsistency
- Multiple file formats, duplication of information in different files
- Difficulty in accessing data
 - Need to write a new program to carry out each new task
- Data isolation — multiple files and formats



Drawbacks of using file systems (cont.)

❖ Integrity problems

- ❑ Integrity constraints (e.g. account balance > 0) become “buried” in program code rather than being stated explicitly
- ❑ Hard to add new constraints or change existing ones

❖ Atomicity of updates

- ❑ Failures may leave database in an inconsistent state with partial updates carried out
- ❑ E.g. transfer of funds from one account to another should either complete or not happen at all

Drawbacks of File System

- = Data redundancy
- = Require more primary memory
- = Processor has to do lot of work
- = More Computations



- Separation of the Data definition and the Program
- Abstraction into a simple model
- Data independence and efficient access.
- Reduced application development time – ad-hoc queries
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.
- Support for multiple different views



☐ Shift from computation to information:

- at the “low end”: scramble to webspace (a mess!)
- at the “high end”: scientific applications

☐ Datasets increasing in diversity and volume.

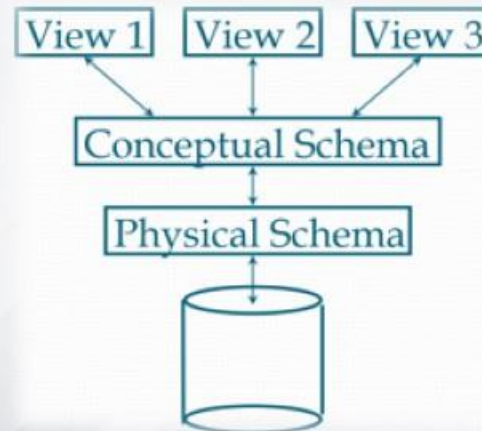
- Digital libraries, interactive video, Human Genome project, EOS project.
- ... need for DBMS exploding

☐ DBMS encompasses most of CS

- OS, languages, theory, “AI”, multimedia, logic

Many **views**, single **conceptual (logical) schema** and **physical schema**.

- Views describe how users see the data.
 - Conceptual schema defines logical structure. Sometime we separate between conceptual level and logical level
 - Physical schema describes the files and indexes used
-
- *Schemas are defined using DDL (Data Definition Language)*
 - *data is modified/queried using DML (Data Manipulation Language)*



Levels of Abstraction

Physical level: describes how a record (e.g., customer) is stored.

Logical level: describes data stored in database, and the relationships among the data.

```
type customer = record  
    customer_id : string;  
    customer_name : string;  
    customer_street : string;  
    customer_city : string;
```

```
end;
```

View level: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes

Instances and Schemas:

- **Schema** – the logical structure of the database
 - Example: The database consists of information about a set of customers and accounts and the relationship between them)
 - Analogous to type information of a variable in a program
 - **Physical schema**: database design at the physical level
 - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
 - Analogous to the value of a variable

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
 - Applications depend on the logical schema
 - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

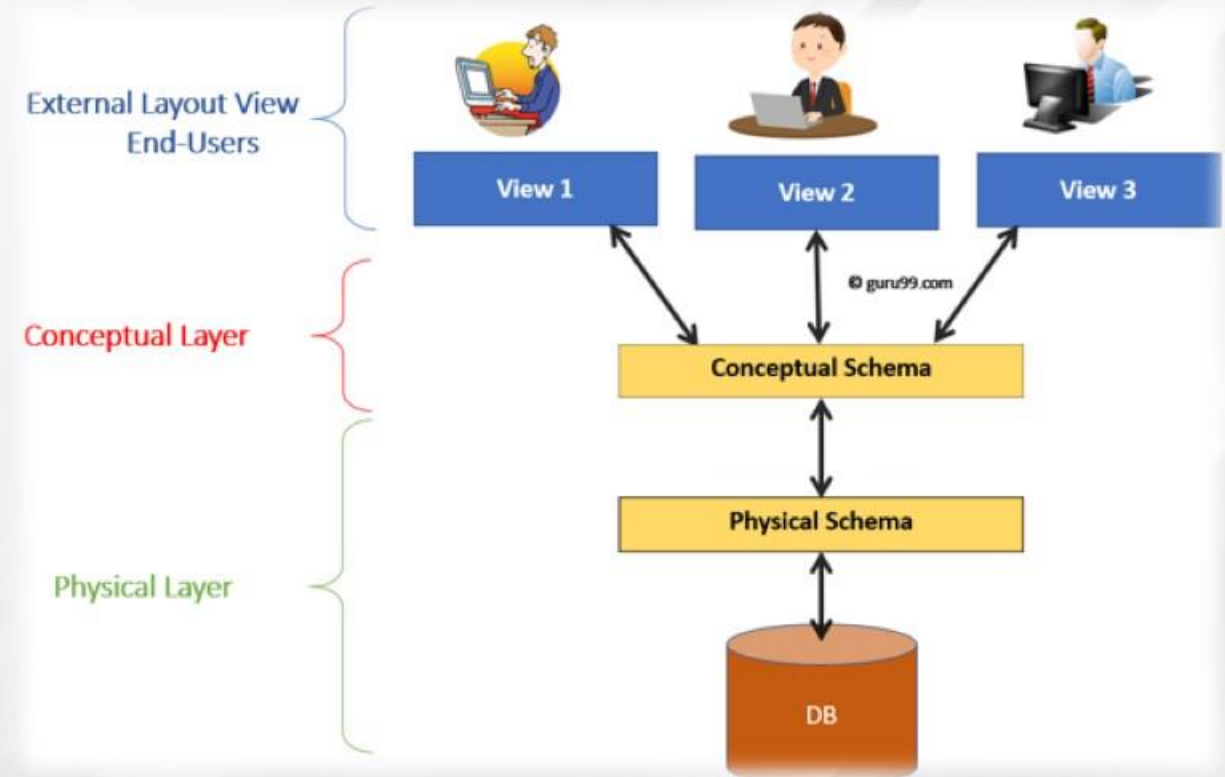
Schemas

- ☆ The **overall design** of the database.
- ☆ Database schema = Variable declaration.
- ☆ **Type** definitions.
- ☆ **No frequent changes** or not at all.
- ☆ Example:

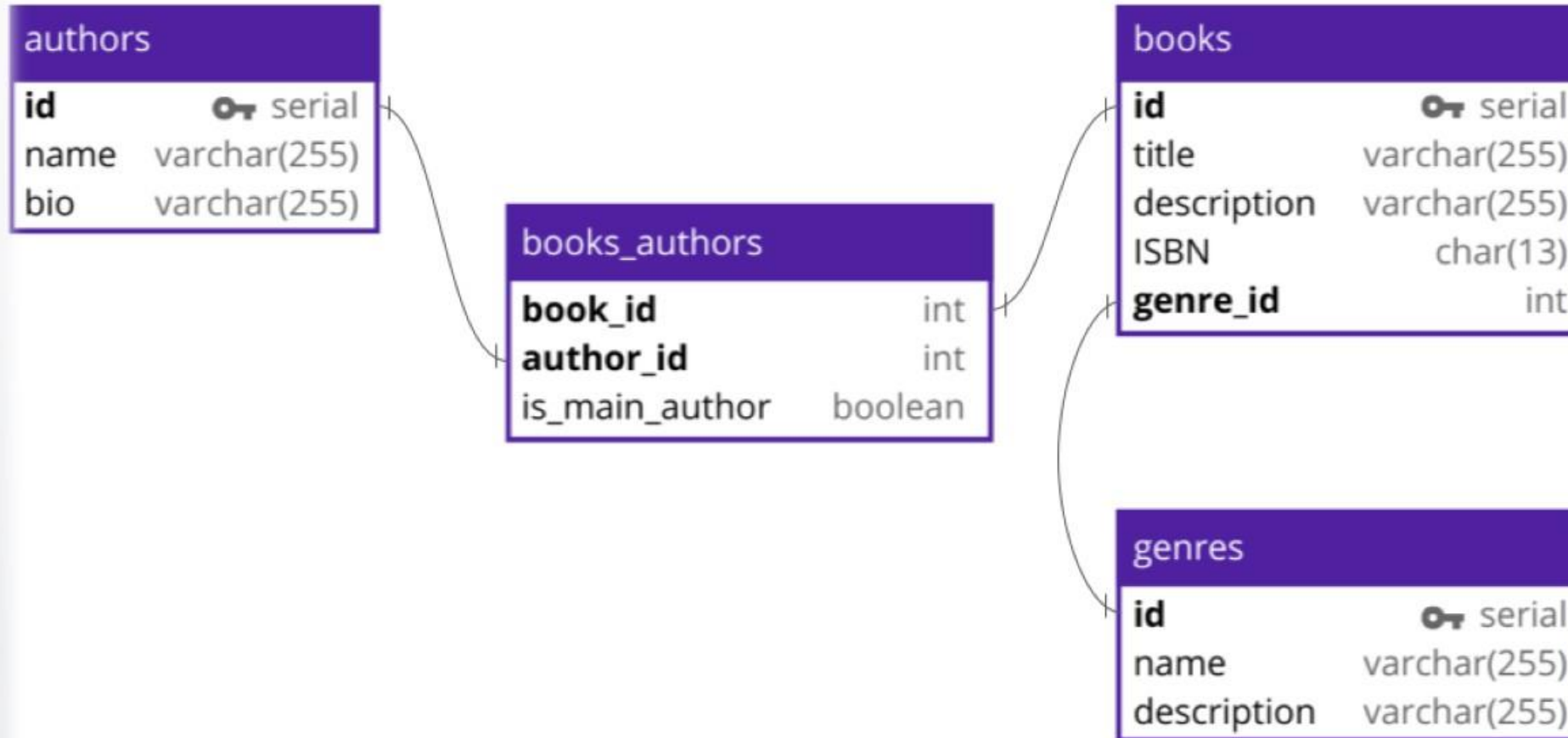
```
type Student = record
    Rollno : numeric (5);
    Name   : char (25);
    Class  : char (10);
end;
```

Instances

- ☆ The collection of information stored at a **particular moment**.
- ☆ **Frequent** changes.
- ☆ Database instant = Value of the variable.



Example of Database Schema.







Thank You