Database Management Systems

DBMS IT214

(3-0-3-4.5)

Rachit Chhaya

rachit_chhaya@daiict.ac.in

FB3 , Room no. 3109

Overview

- **Course Handout**
  - Resources
  - Evaluation Scheme
- **Course Plan**
  - Course Overview

Resources

- **Text Book**
  - Silberschatz, Korth & Sudarshan, *Database System Concepts,* 7th Edition, McGraw-Hill, 2019

- **Additional Books**
  - Ramakrishnan & Gehrke, *Database Management Systems*, McGraw-Hill, 2003
  - Elmasri & Navathe, *Fundamentals of Database Systems*, Pearson Education 1999
  - Date C.J., *An Introduction to Database Systems*, Addison-Wesley, 2001

- **Lecture folder**

Operational Details

- **Class**
  - (Tue, Thu)10:00 am, Fri 11:00 am @ CEP 110
- **Lab Work**
  - 3 hours/ week
  - Each Lab will be evaluated. Student must attend and complete each lab in order to even pass the course
- **Exams**
  - MidSem Exam/s
  - EndSem Exam

Evaluation Scheme

- Labs & Assignments 30%
- MidSem Exam/s 30%
- EndSem Exam 40%

Database and DBMS

- **Database System DBS**
  - Collection of interrelated data, describing activities of one or more related organizations.
  - **Example:** DAIICT database contains information about students, faculty, courses, classrooms, exams
  - Set of programs to access those data
- Database Management System DBMS is a software designed to assist in maintaining and utilizing large collections of data
- Alternative is to store the data in files and use application specific codes

Data Management

- Defining structures for data storage
- Providing mechanism for accessing this information

Why Study Databases?

- Shift from computation to information
- Volume, velocity, and variety of data
- Digital libraries, biological databases

Databases Everywhere!!!

- Database Applications
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions

Databases touch all aspects of our lives

DBMS Functionalities

- Define a database : in terms of data types, structures and constraints
- Construct or Load the Database on a secondary storage medium
- Manipulating the database : querying, generating reports, insertions, deletions and modifications to its content
- Concurrent Processing and Sharing by a set of users and programs – yet, keeping all data valid and consistent
- Crash Recovery

The DBMS Marketplace

- Relational DBMS companies – Oracle, Sybase – are among the largest software companies in the world.
- IBM offers its relational DB2 system. With IMS, a non relational (hierarchical) system, IBM is by some accounts the largest DBMS vendor in the world.
- Microsoft offers SQL-Server, plus Microsoft Access for the cheap DBMS on the desktop
- Relational companies also challenged by object-oriented DB companies.
- But countered with object-relational systems, which retain the relational core while allowing type extension as in OO systems.

Studying DBMS

(1) Modeling and design of databases

Data, Structuring the data

(2) Programming

queries and DB operations like update

SQL (DDL, DML) Data definition language, Data Manipulation Language

(3) DBMS implementation

IT214 = (1) + (2), while (3) is to be covered partly

Course Plan

ER model stands for an Entity-Relationship model.

- **Database Overview**

  Basic Definitions, Data Storage, Data Models, Queries, Query Optimization, Transaction Management, Distributed Databases
- **What Data?** Requirements Collection and Analysis
- **Structuring Data (Data models)**
  - E-R Model, Conceptual Design using E-R Model
  - Relational Model
    - introduction, database architecture, integrity constraints, database design
- **Query Languages:** Relational Algebra, SQL
- **Conceptual Database Design & Tuning**
  - relational data model
  - FD, Normal Forms, Decomposition, Normalization, Schema Refinement

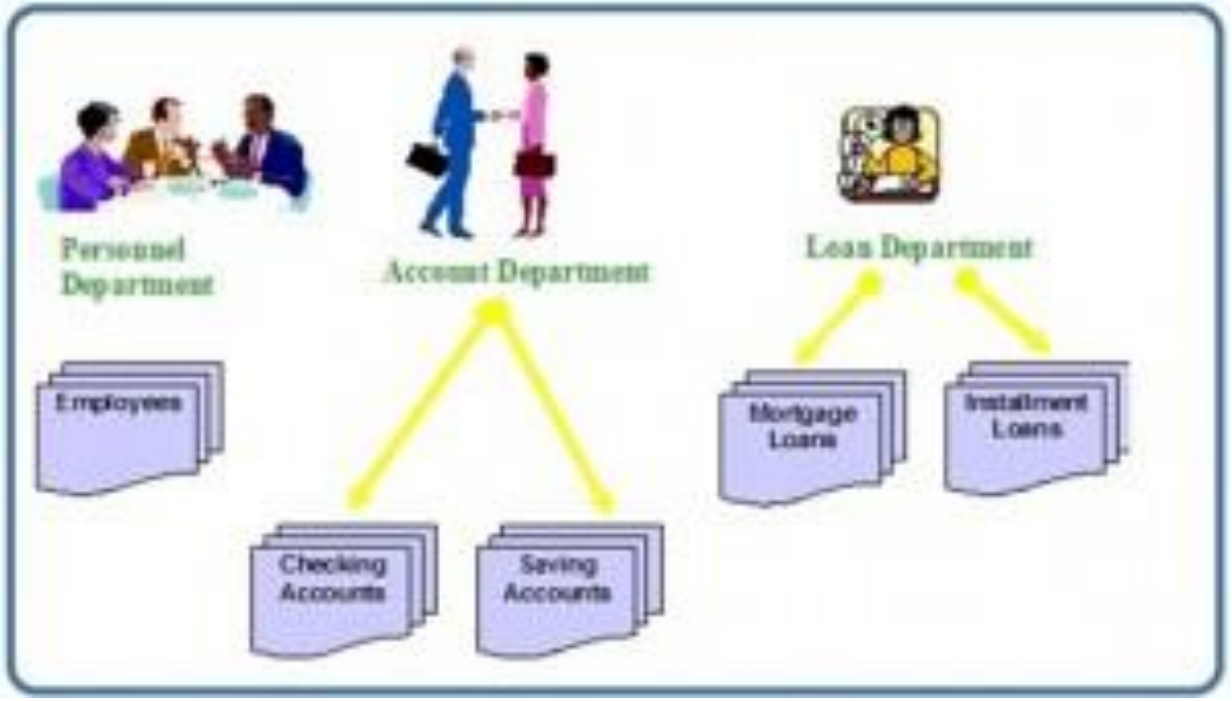
Course Plan

- **Data Storage and Indexing**
  - Storage/organization of data on disks, tapes
  - Indexing
- **Query Processing and Optimization**
  - Query Cost, Evaluation Plans, Materialized Views
- **Parallel and Distributed Databases**
  - Architecture, Storage, Query Processing
- **Transaction Management**
  - ACID Properties, Concurrency Control, Crash Recovery
- **Issues in Modern Databases**
- **Database Administration and DBM Tools**

  Oracle, postgresql/ postgres

Purpose and Nature of Database Systems

File System Example



Early Days Database Applications

- **Data redundancy and inconsistency**
  - data is stored in multiple file formats resulting in 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
- **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

Early Days Database Applications

- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- **Concurrent access by multiple users**
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- **Security problems**
  - Hard to provide user access to some, but not all, data

Database Systems

- **Database systems offer solutions to these problems**
  - Data redundancy & inconsistency
  - Data access
  - Data isolation
  - Integrity problems
  - Atomicity of updates
  - Concurrent accesses by multiple users
  - Security issues

What is a DBMS?

- A very large integrated collection of data
- Models real world entities
- Database Management System DBMS is software designed to assist in maintaining and utilizing large collections of data

Advantages of a DBMS

- Program-Data Independence
  - Insulation between programs and data: Allows changing data storage structures and operations without having to change the DBMS access programs.
- Efficient Data Access
  - DBMS uses a variety of techniques to store & retrieve data efficiently
- Data Integrity & Security
  - Before inserting salary of an employee, the DBMS can check that the dept. budget is not exceeded
  - Enforces access controls that govern what data is visible to different classes of users

Advantages of a DBMS

- Data Administration
  - When several users share data , centralizing the administration offers significant improvement
- Concurrent Access & Crash Recovery
  - DBMS schedules concurrent access to the data in such a manner that users think of the data as being accessed by only one user at a time
  - DBMS protects users from the ill-effects of system failures
- Reduced Application Development Time
  - Many important tasks are handled by the DBMS

Data Models

Structuring the Data

Data Models

- A data model is collection of concepts for describing data
- A schema is a description of a particular collection of data, using the given data model
- A relational data model is the most widely used data model today
- Network Model, Hierarchical Model, Object-Oriented Model, Relational Model, Knowledge Bases , XML, RDF, OWL, Graph Databases

Data Models

- Relational Model (E.F.Codd 1970)
  - Relation is a table with rows and columns
  - Every relation has a schema that describes the columns or fields

- Relational model is good for:
  - Large amounts of data, simple operations
  - Navigate among small number of relations
  - Difficult Applications for relational model
    - VLSI Design (CAD in general)
    - CASE
    - Graphical Data

Data Models

Where number of relations is large, relationships are complex

- Object Data Model

Object Data Model

- Complex Objects – Nested Structure (pointers or references)
- Encapsulation, set of Methods/Access functions
- Object Identity
- Inheritance – Defining new classes like old classes

Relational Model

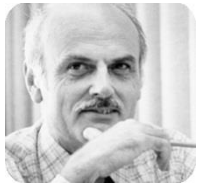
- All the data is stored in various tables.
- Example of tabular data in the relational model

Columns

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Rows

(a) The *instructor* table



Ted Codd
Turing Award 1981

A Sample Relational Database

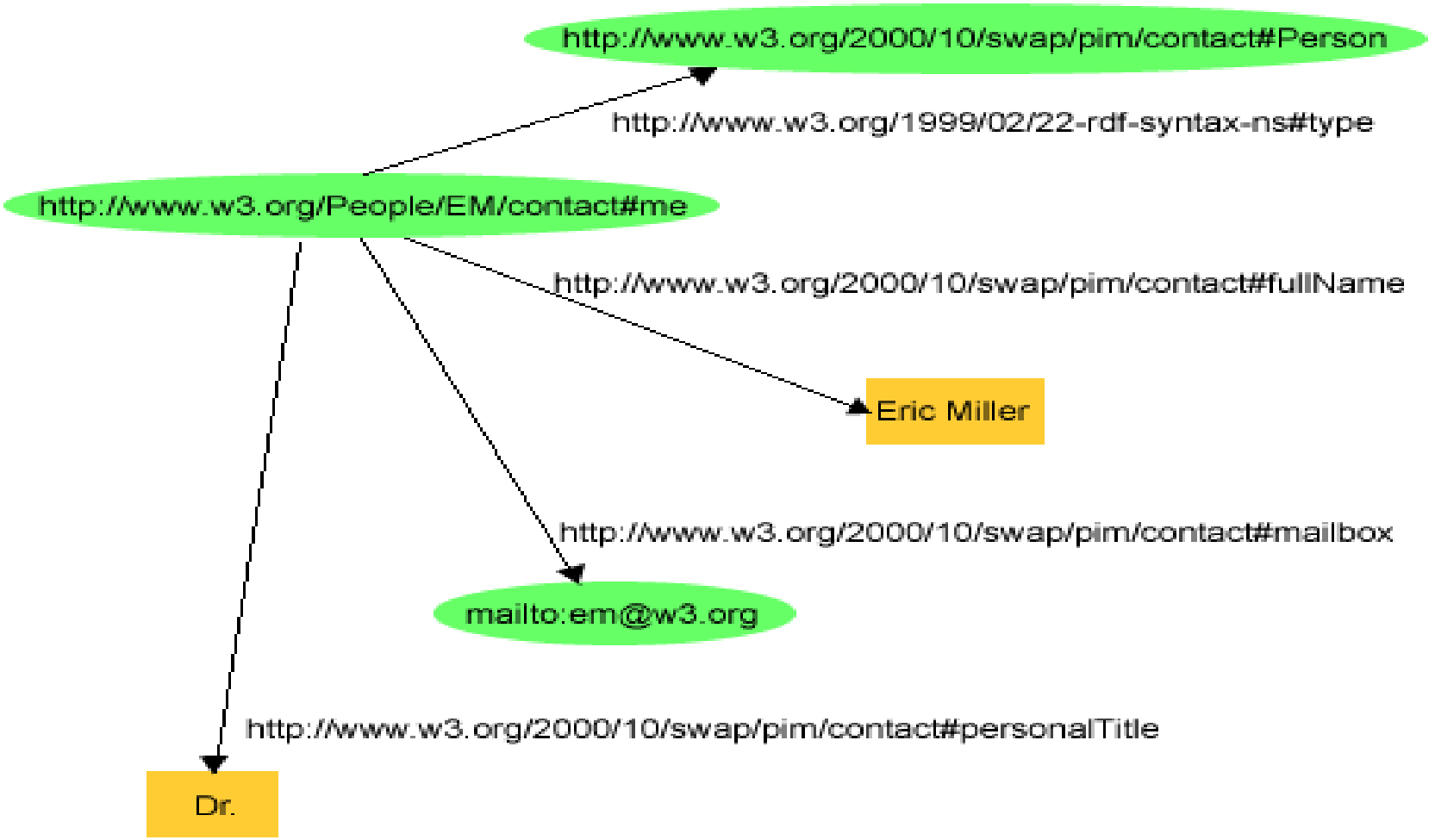
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

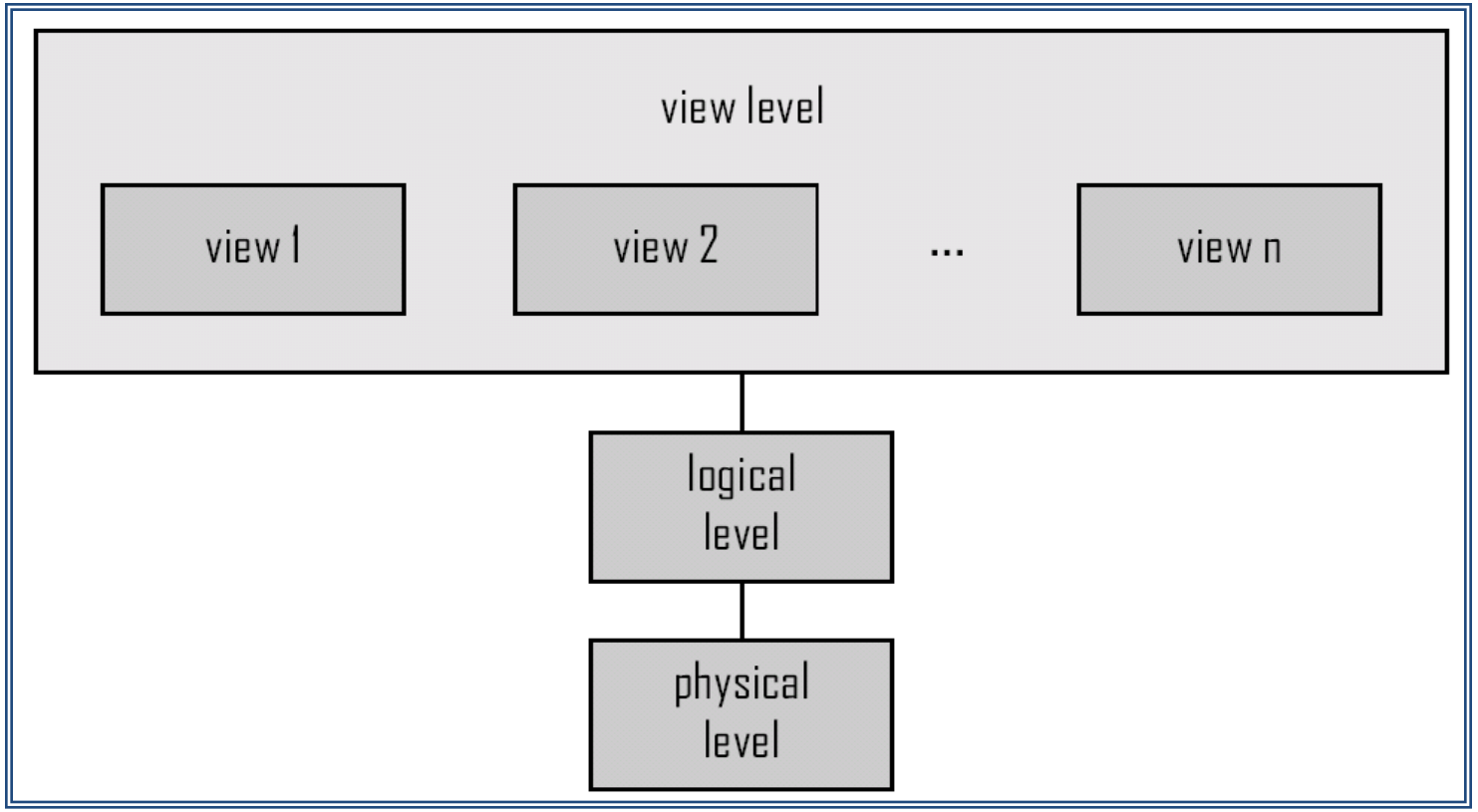
(b) The *department* table

RDF Model


http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

Levels of Abstraction

- Databases provide users with an abstract view of data



3 Levels of Abstraction

- Physical or Internal Level
  - Lowest level of abstraction describes how data are actually stored
  - Describes complex low-level data structures in detail
- Logical or Conceptual Level
  - Describes what data are stored in the DB & what relationships exist among those data
  - Describes the entire DB in terms of relatively simpler structures
- View or External Level
  - Highest level of abstraction which describes only a part of the DB
  - User's view of the DB. This level describes part of the DB that is relevant to each user

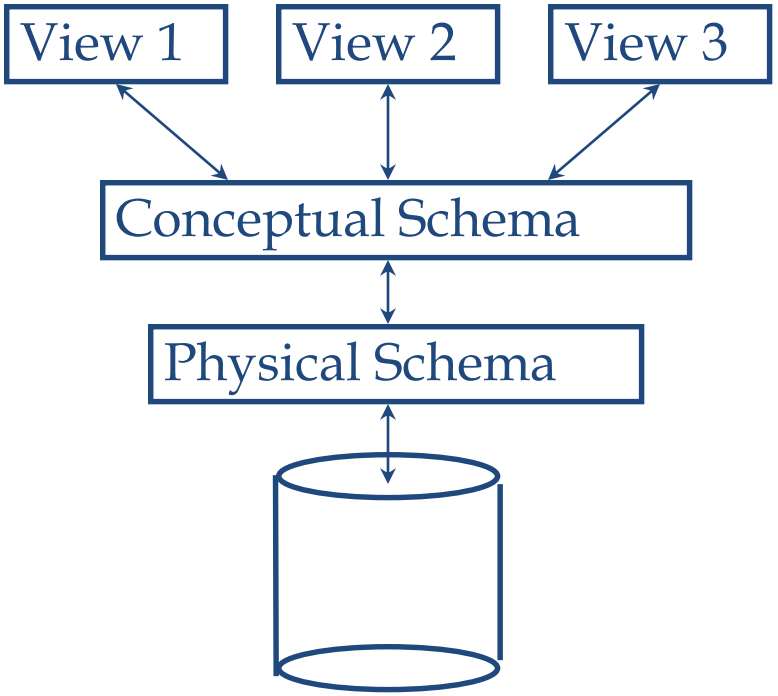
Data Independence

- Applications insulated from how data is structured and stored
- Logical data independence: Protection from changes in logical structure of data
- Physical data independence: Protection from changes in physical structure of data

Levels of Abstraction

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

- Views describe how users see the data
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used



** Schemas are defined using DDL; data is modified/queried using DML*

Instances and Schemas

- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema** – the overall physical structure of the database, indexes, space allocation, data compression, encryption, access paths

  DBMS: Data access optimized and storage minimized

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Example : University Database

- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files
  - Index on first column of Students
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

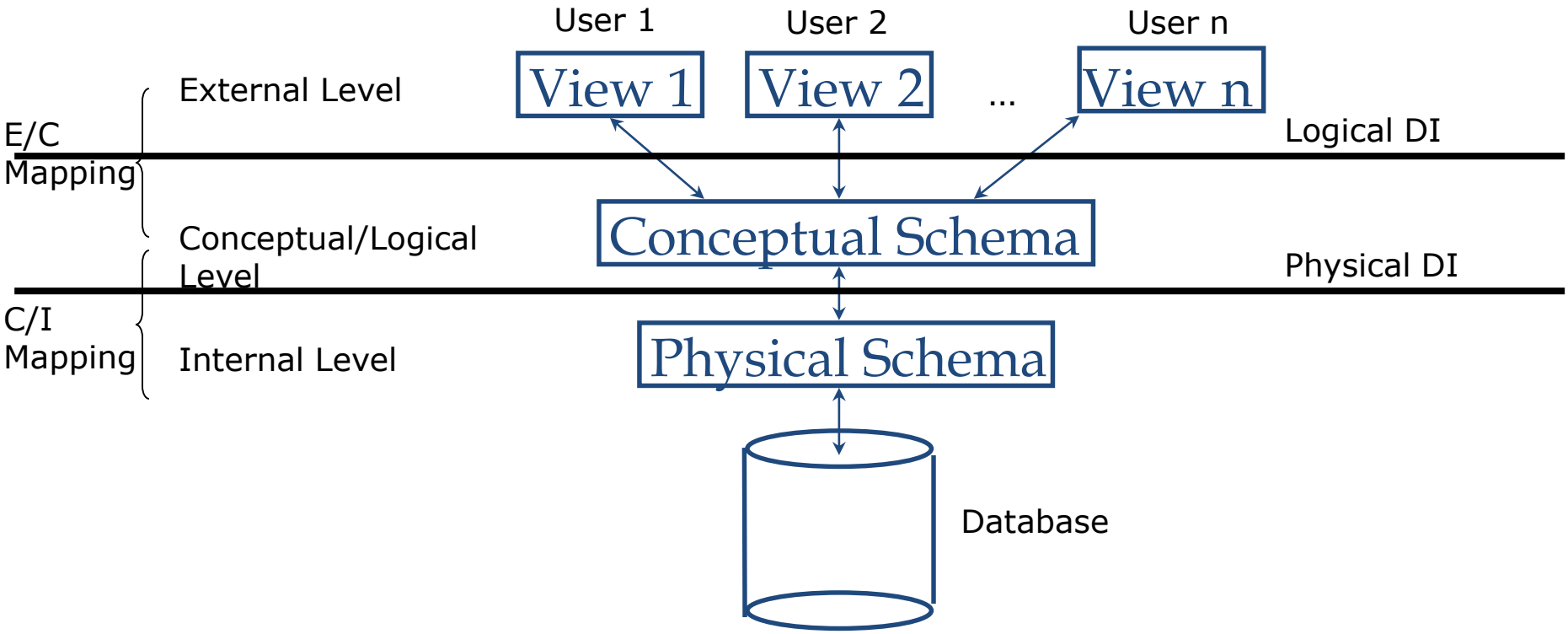
Physical Data Independence

- Applications insulated from how data is structured and stored
- Logical data independence: Protection from changes in logical structure of data
- Physical data independence: Protection from changes in physical structure of data

  the ability to modify the physical schema without changing 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.

ANSI/SPARC 3-Tier Architecture



ANSI-SPARC Architecture

- Separates users' view from the way in which the data is arranged physically or logically
- It hides the physical storage details from users: Users should not have to deal with physical database storage details. They should be allowed to work with the data itself, without concern for how it is physically stored
- The database administrator should be able to change the database storage structures without affecting the users' views : From time to time changes to the structure of an organisation's data will be required.
- The internal structure of the database should be unaffected by changes to the physical aspects of the storage : For example, a changeover to a new disk.

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20)**,**
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as query language
- There are basically two types of data-manipulation language
  - **Procedural DML** -- require a user to specify what data are needed and how to get those data.
  - **Declarative DML** -- require a user to specify what data are needed without specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a **query** language.

SQL Query Language

- SQL query language is nonprocedural. A query takes as input several tables (or only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

```
select name
from instructor
where dept_name = 'Comp. Sci.‘
```

- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface API (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Access from Application Program

- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a **host language**, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- **Application programs** -- are programs that are used to interact with the database in this fashion.

Database Design

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be divided into
  - The storage manager
  - The query processor component
  - The transaction management component

Storage Manager

- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager

Storage Manager

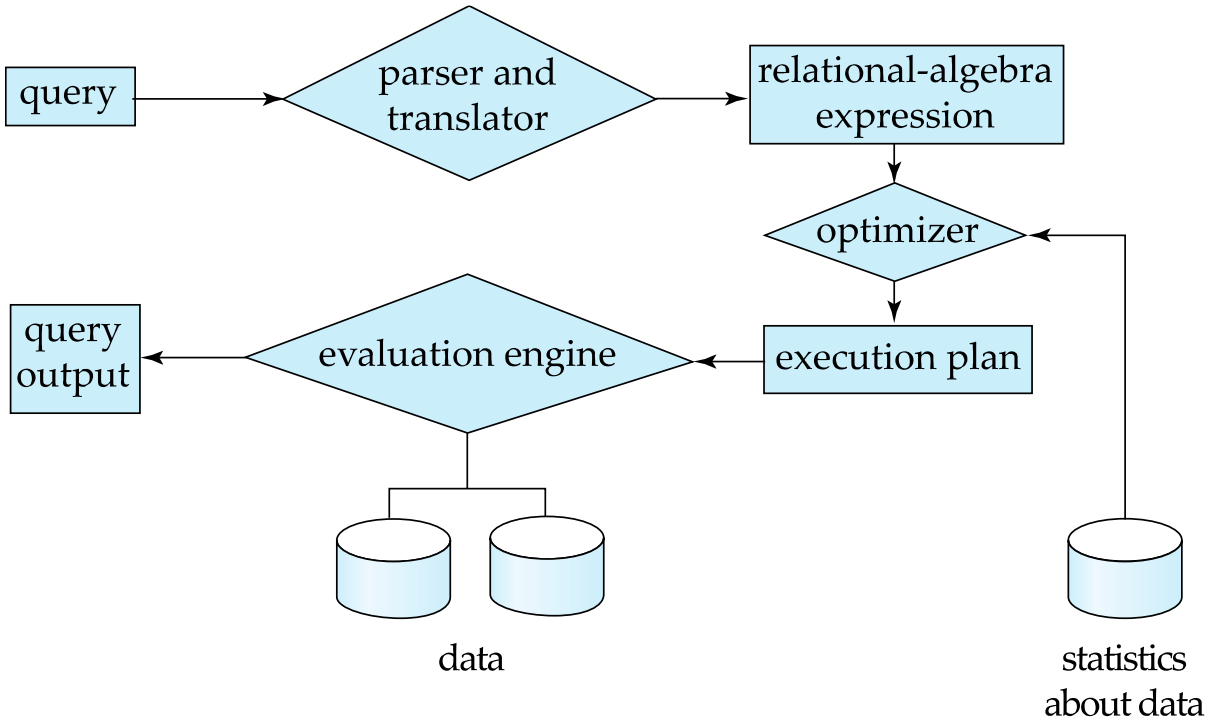
- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.

Query Processor

- **DDL interpreter** -- interprets DDL statements and records the definitions in the data dictionary.
- **DML compiler** -- translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
  - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
- **Query evaluation engine** -- executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Concurrency Control

- Concurrent execution of user program is essential for good DBMS performance
  - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu working on several user programs concurrently
- Interleaving actions of different user programs can lead to inconsistencies
- DBMS ensures such problems don’t arise: users can pretend they are using a single-user system

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of Database Systems

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon,
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map reduce
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

Summary

- DBMS is used to maintain, query large volume of data
- Benefits of DBMS include recovery from system crashes, concurrent access, quick application development, data integrity and security
- Levels of abstraction give data independence
- A DBMS typically has a layered architecture