

# Module 1: Introduction Database Concepts

Prof. Pallavi Saindane  
Department of Computer Engineering  
Vivekanand Education Society's Institute of Technology

# Topics Covered:

## **Introduction Database Concepts:**

- Introduction, Characteristics of databases
- File system v/s Database system
- Data Abstraction and Data Independence
- DBMS system architecture
- Database Administrator

# Some Important Definitions

- **What is Data?**

- Raw, unrelated, unorganized, isolated facts of an entity which has no meaning.
- Anything which can be given as input to data processing systems (recordable) -numbers, letters, set of character , images graphics etc.

1. Rs. 10,000/-, Rs 20,000/- Rs. 30,000/- etc.
2. CS102, CS 503 etc.
3. abc\_xyz , abc1234

**What is your understanding about these numbers / Set of characters???**

# Some Important Definitions

- What is Information?
  - Its is a processed data , meaningful data, relevant
  - **Information = Data + Meaning**

eg.

**Salary of Employee:** Rs. 10,000/-, Rs 20,000/- Rs. 30000/- etc.

**Course Codes:** CS102, CS 503 etc.

**User\_id:** abc\_xyz, **Password:** abc1234

# Some Important Definitions

- **What is Database**

- Collection of **related / similar Data**
- Data can be Structured , unstructured, Semistructured
- For Structured Data we do use Databases

**Eg. University Database, Hospital Database, IRCTC, Airline Database etc.**

# Some Important Definitions

- **What is Database Management system?**

- The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.
- DBMS = Database + Operations
- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use
- Eg. SQL Server, Oracle, MySql etc.

# University Database Example

## STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

## COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

## GRADE\_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

## PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Database manipulation involves querying and updating. Examples of queries are as follows:

- Retrieve the transcript—a list of all courses name and grades—of ‘Smith’
- List the prerequisites of the ‘Database’ course

Examples of updates include the following:

- Change the class of ‘Smith’ to sophomore
- Create a new section for the ‘Database’ course for this semester

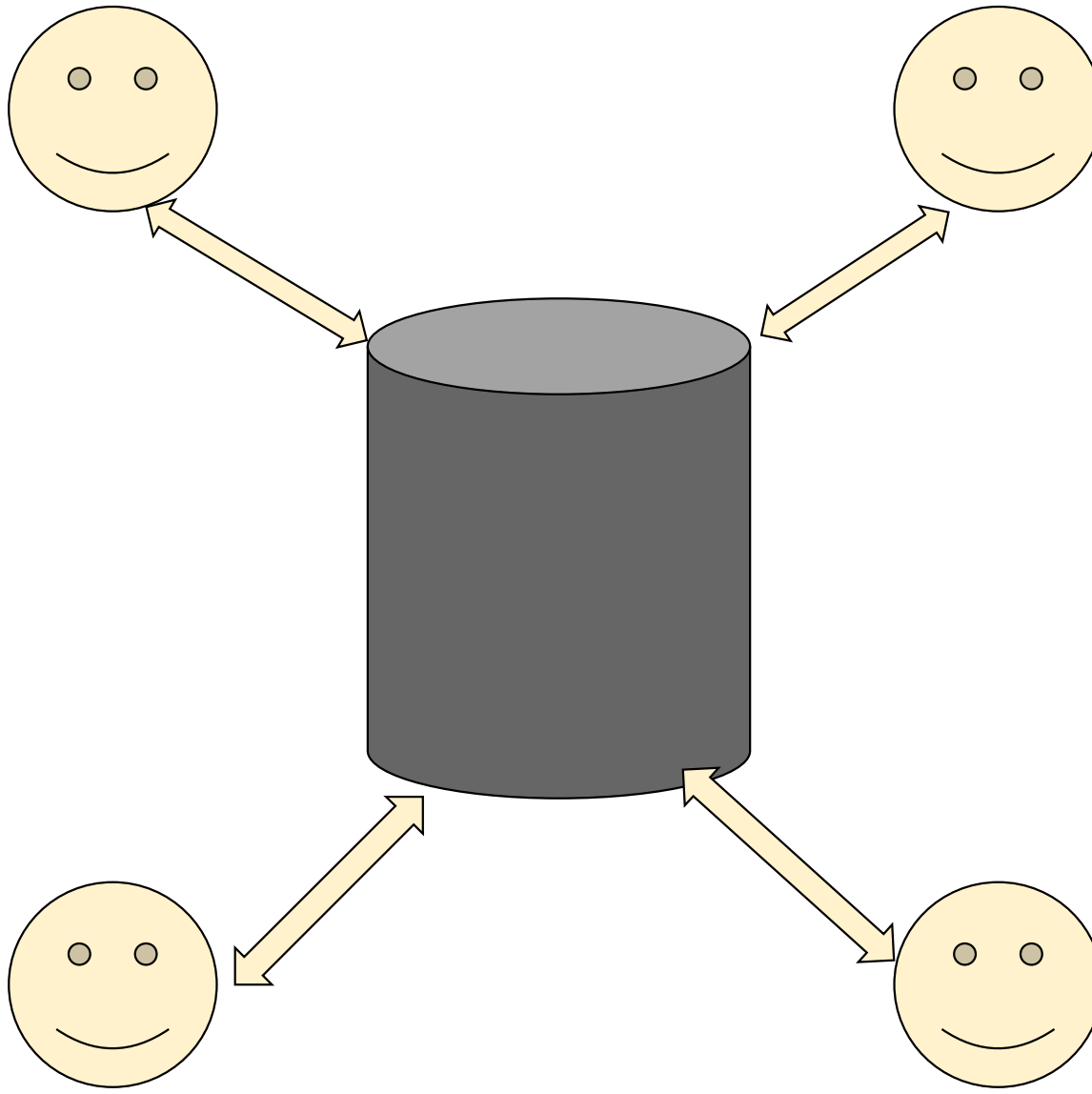
# Database Management System (DBMS)

## Database Applications:

- Banking: 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



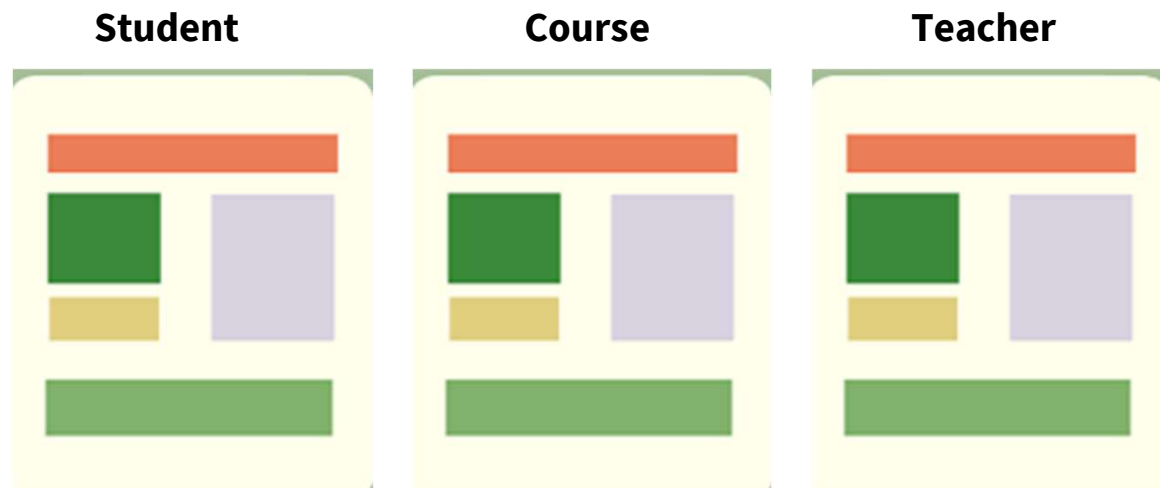
# Drawbacks of using file systems to store data



# Drawbacks of using file systems to store data

- **Difficulty in accessing data**

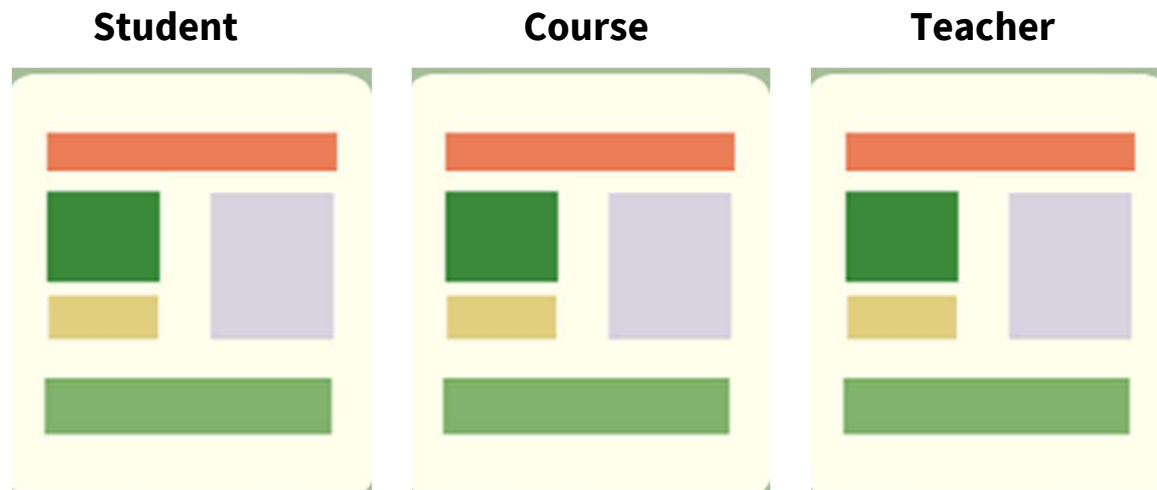
- Accessing multiple files simultaneously is difficult in File Systems
- Reaching to the appropriate point of access is a non-trivial task
- In DBMS with Simple Query we can achieve it.



**Find a details of student with roll no 10 , the course he has taken and a faculty who teaches him that course**

# Drawbacks of using file systems to store data

- **Data redundancy and inconsistency : duplication of information in different files**
  - Databases use **constraints / Normalization** for avoiding redundancy and thus aiming towards consistency



**Student : Student with roll No. 10 has taken a course 'Database management System'**

**Course: Name of the course 'Database management systems' is changed to 'Database and Info Systems' but the change is not reflected in Student file**

# Drawbacks of using file systems to store data

- **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.
- In DBMS Atomicity can be achieved through **'Transactions'**

- **Concurrent access by multiple users**

- Concurrent access needed for performance
- Uncontrolled concurrent accesses can lead to inconsistencies
  - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- In DBMS we do have **Concurrency Control mechanism** to ensure consistency of data

# Drawbacks of using file systems to store data

- **Data isolation** : Multiple files and formats. Difficult to achieve consistency in isolated mode.
  - In databases allows isolation so that multiple transactions can occur at the same time without adversely affecting the execution of each.
  - transactions should operate on the database in an isolated manner. The database should either perform A's entire transaction before executing B's or vice-versa. This exclusivity prevents B's transaction from reading intermediate data produced as a side effect of part of A's transaction that will not eventually be committed to the database.

## Drawbacks of using file systems to store data (Cont.)

- Security problems ( Role Based Security)
  - Hard to provide user **access to some, but not all data**
  - In DBMS we can provide a role based access of data.
- 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.
  - In DBMS we can apply constraints easily.

**Database systems offer solutions to all the above problems**

# Characteristics of DBMS

The main characteristics/ advantages of the database approach versus the file-processing approach are the following:

- **Data Independence:** Application programs should not, ideally, be exposed to details of data representation and storage,
- **Efficient Data Access:** A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently.
- **Data Integrity and Security:** DBMS can enforce integrity constraints. Also, it can enforce access controls that govern what data is visible to different classes of users.
- **Data Administration**
- **Concurrent Access and Crash Recovery:** A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
- **Reduced Application Development Time:** DBMS in conjunction with the high-level interface to the data, facilitates quick application development.

# Schema And Instance





# Schemas

- A **database schema** is the skeleton structure that represents the logical view of the entire **database**
- Analogous to **data types** in programming languages

## Back Customer Schema :

Name	Customer ID	Account#	Aadhaar ID	Mobile #
------	-------------	----------	------------	----------

## Account Schema:

Account#	Account Type	Interest rate	Min Bal	Balance
----------	--------------	---------------	---------	---------

# Instances

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable in programming language
  - Ex. Customer Instance and Account Instance

## Instance of Customer

Name	Customer ID	Account#	Aadhaar ID	Mobile #
Pavan Lal	6728	91734	12345863212	9823671905
Lata Sharma	5419	8766	84620258159	8457139253

## Instance of Account

Account#	Account Type	Interest rate	Min Bal	Balance
91734	Savings	4%	5000	12456
8766	Current	0%	0	453987

# **Data Abstraction and Data Independence**



# Database Management System - 3 Tier Architecture

**View  
Level**

## Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

**End  
User**



**Application  
Layer**

## Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.



**GET LIST OF ALL  
SALES MADE  
LAST YEAR**



**ADD ALL SALES  
TOGETHER**

**Physical  
Layer**

## Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

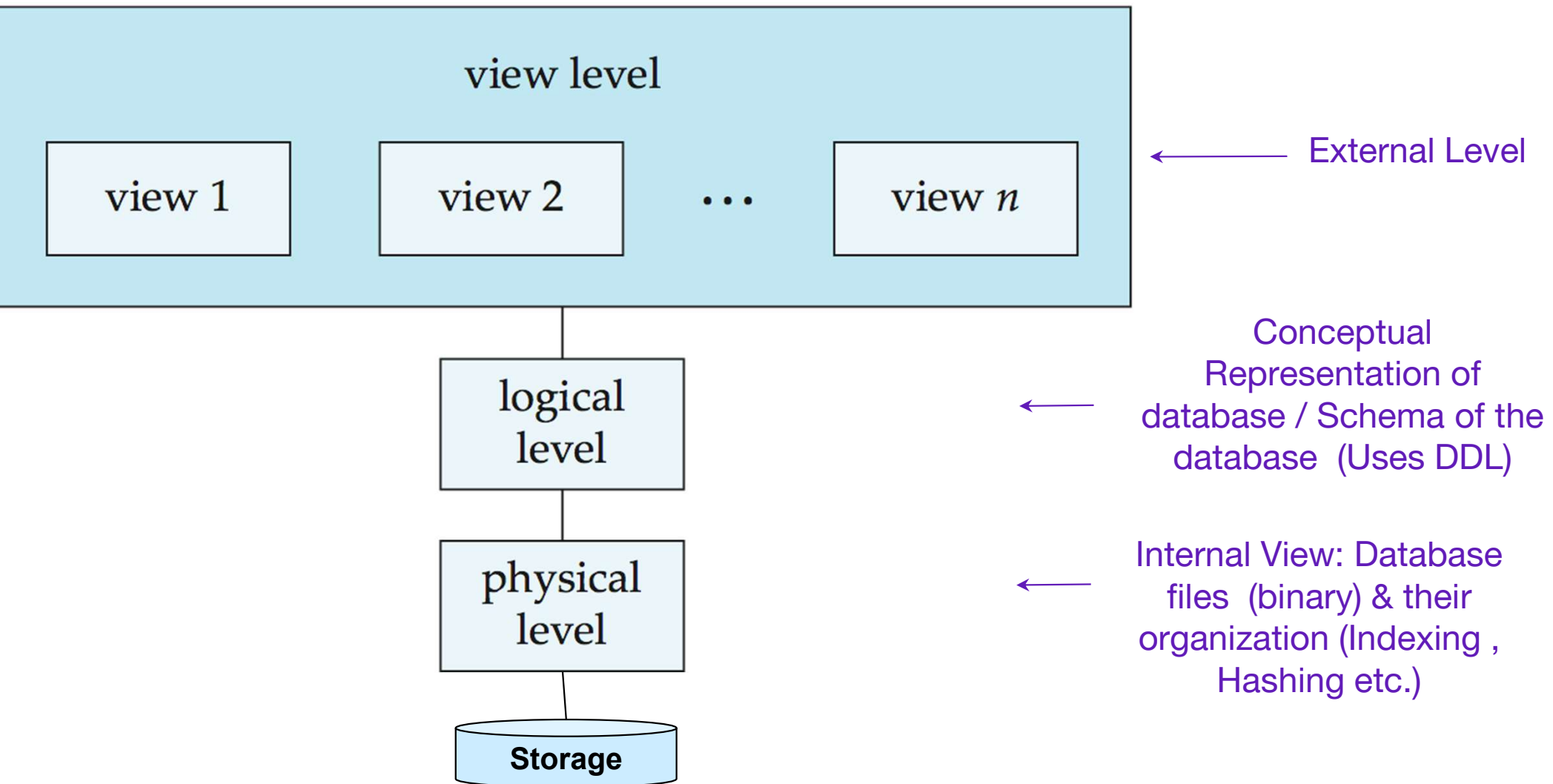


**Database**



**Storage**

# Levels of abstraction in DBMS



# Levels of Abstraction in three schema architecture

- **View level:**
  - Here application program tries to view the data
  - Application program deals with details which are required and others are usually hidden from view.
    - application programs hide details of data types.
    - Views can also hide information (such as an employee's salary) for security purposes.
    - It provides high level of abstraction by allowing customized and authorized view
- **Logical level:** describes data stored in database, and the relationships among the data in terms of data model. (Record is collection of multiple fields of different types)

```
type instructor = record  
    ID : string;  
    name : string;  
    dept_name : string;  
    salary : integer;  
  
end;
```

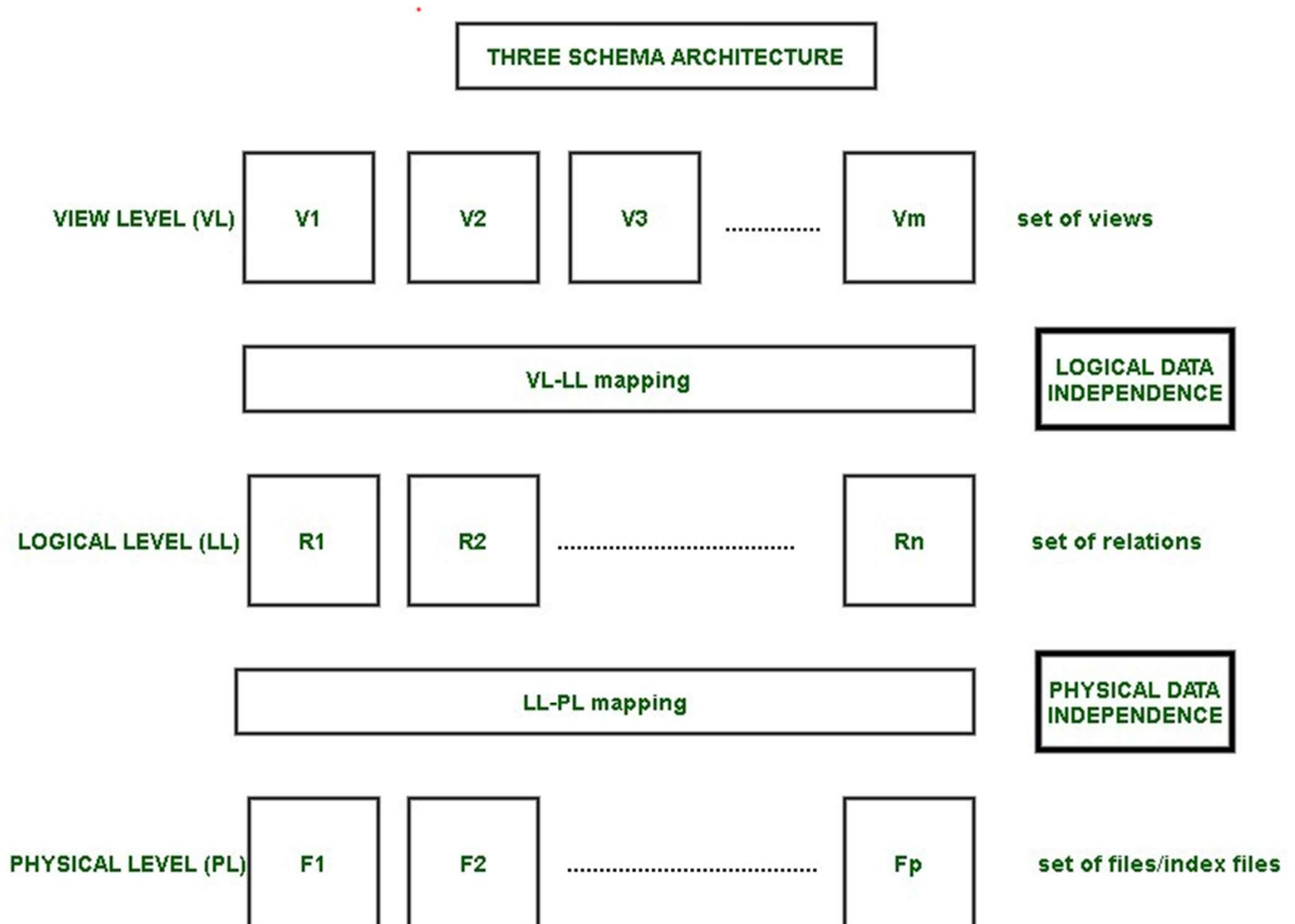
```
instructor ( ID:string, name:string,  
             dept_name:string, salary:integer)
```

- **Physical level:** describes how a record/relations (e.g., instructor) are actually stored on secondary devices such as disks and tapes.

# Important to note about Schema and Instance

- **External schema** : describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be "n" number of external views for each database
  - **Logical Schema** – the overall logical structure of the database (It is a way , in which a certain data needs to be organized)
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
  - **Physical / Internal schema**– the overall physical structure of the database (eg. What are different database files, how they are indexed etc.)
- 
- **Instances keep on changing But Schema remains unchanged as changing schema at logical level may change view , hence change in schema is very rare**
    - When database is designed
    - When it is upgraded

# Data Independence





# Data Independence

- 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.
1. **Logical Data Independence** : It is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item).
    - For example, the external schema for generating Transcripts should not be affected by changing the GRADE\_REPORT file (or record type). Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before.
    - In case if data item is deleted from conceptual schema then in that case external schema and an application program that use the data have to be changed because the data will no longer be available.

**STUDENT**

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

**COURSE**

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

**SECTION**

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

**PREREQUISITE**

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

**GRADE\_REPORT**

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

**GRADE\_REPORT**

Student_number	Student_name	Section_identifier	Course_number	Grade
17	Smith	112	MATH2410	B
17	Smith	119	CS1310	C
8	Brown	85	MATH2410	A
8	Brown	92	CS1310	A
8	Brown	102	CS3320	B
8	Brown	135	CS3380	A

**TRANSCRIPT**

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

# Data Independence

**2. Physical Data Independence** – the ability to modify the physical schema i.e. physical file structure of a database without disturbing the existing users and processes.

- eg. Storage device on which database file has been stored is changed & lets say user fires a query “select \* from table1”. The above query only states the table’s name which contains data and not the location where that table is stored.

Note: Physical data independence exists in most databases and file environments details. On the other hand, **logical data independence is harder to achieve** because it allows structural and constraint changes without affecting application programs—a much stricter requirement.

Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the **mapping** between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

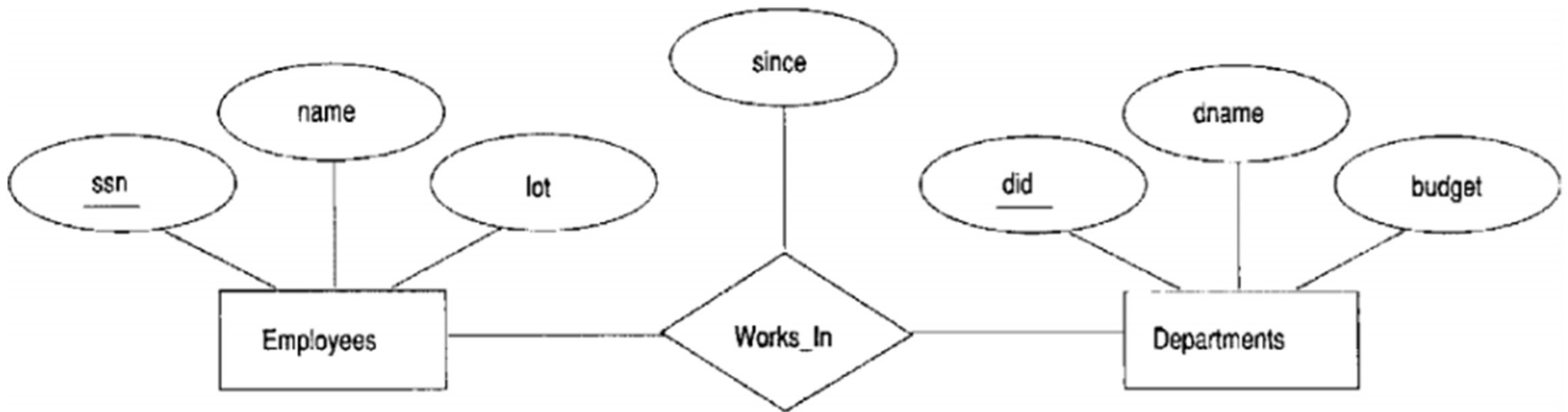
# DATA MODELS

# Data Models

- **Data Model** : A collection of tools for describing Data, Data relationships, Meaning of a data - Data semantics, Data constraints
- **Some Types of Data Models**
  - Entity-Relationship data model (mainly for database design)
  - Relational model
  - Object-relational data models
  - Semi structured data model (XML) – data exchange
  - Network model
  - Hierarchical model

# Entity-Relationship Model

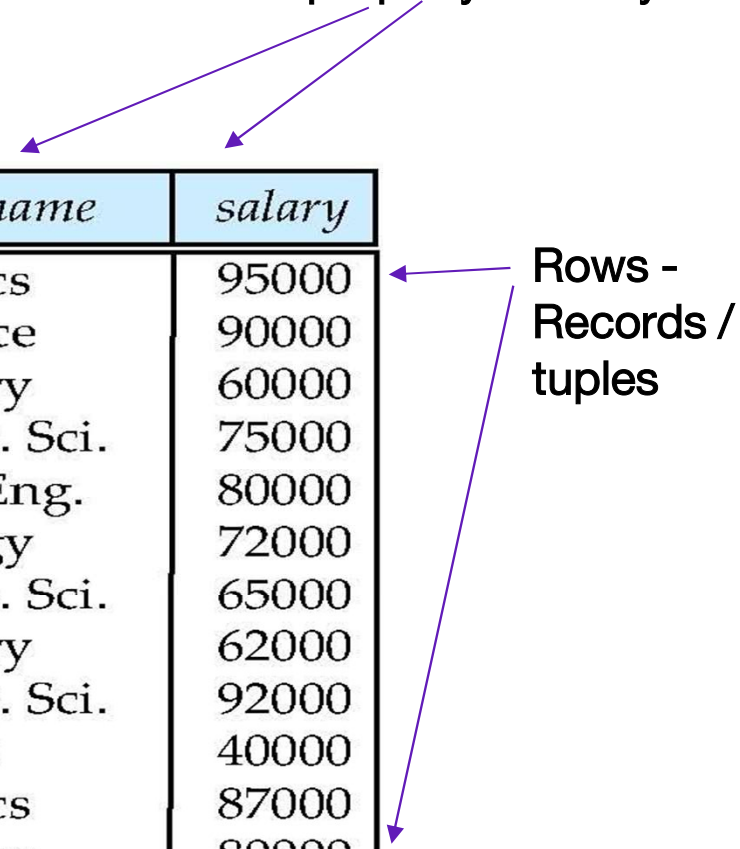
- The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects.
- The entity-relationship model is widely used in database design,



# Relational Model

- All the data is stored in various tables. Tables are called as **relations**.
- Columns represents fields / **attributes** and rows represents **record**
- Example of tabular data in the relational model

Columns - Fields / attributes / characteristics or property of entity



<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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 -  
Records /  
tuples

(a) The *instructor* table

# Object-Relational Data Models

- Relational model: flat, “**atomic**” values
- Object Relational Data Models
  - Extend the relational data model by including **object orientation and constructs** to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as **nested relations**.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.



# XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

```
<?xml version="1.0" standalone="yes"?>
<BankAccount>
  <Number>1234</Number>
  <Type>Checking</Type>
  <OpenDate>11/04/1974</OpenDate>
  <Balance>25382.20</Balance>
  <AccountHolder>
    <LastName>Singh</LastName>
    <FirstName>Darshan</FirstName>
  </AccountHolder>
</BankAccount>
```

# Database Languages



# Database Languages

- Now we have schema and instance also idea about data model. We need language constructs for defining schema and instance
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

# SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language(called as Limited or Restricted language)
- 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 (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

# DDL and DML

- **DDL (Data Definition Language)** : It is language or part of language which is used to define and manipulate schema of database.
- **DML (Data Manipulation Language)** : It is language used to access and manipulate data
- **DCL (Data Control Language )** : Who has access of the data
- **TCL (Transaction Control Language)** : For transaction management and concurrency control

# DDL and DML

- **Data Definition Language** : Specification notation for defining the database schema
  - Example: 

```
create table instructor (  
    ID char(5) unique,  
    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 manipulating the data organized by the appropriate data model
  - DML also known as query language

# Database Design Approach



# Database Design

1. The process of designing the general structure of the database:

**1. Logical Design** – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.

a. Business decision –

- i. What are the schemas
- ii. What are constraints
- iii. What is authorization

a. Computer Science decision –

- i. What relation schemas should we have
- ii. how should the attributes be distributed among the various relation schemas
- iii. What should be type of attributes and so on

**1. Physical Design** – Deciding on the physical layout of the database



Try This -

Business requirement:

Design a database to store information of an instructor like id, name ,salary, the department to which instructor belongs, building in which department is housed and budget.

# Database Design (Cont.)

- Is there any problem with this relation?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

# Database Design (Cont.)

- Look at redundancy and thus potential for update anomaly

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

# Database Design (Cont.)

**Split it Into Two Tables as given Below:**

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Physics	Watson	70000
12121	Wu	90000	Finance	Finance	Painter	120000
32343	El Said	60000	History	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Music	Packard	80000
58583	Califieri	62000	History			
83821	Brandt	92000	Comp. Sci			
15151	Mozart	40000	Music			
33456	Gold	87000	Physics			
76543	Singh	80000	Finance			

There are several such issues ..

How to decide database design is good or needs refinement ?

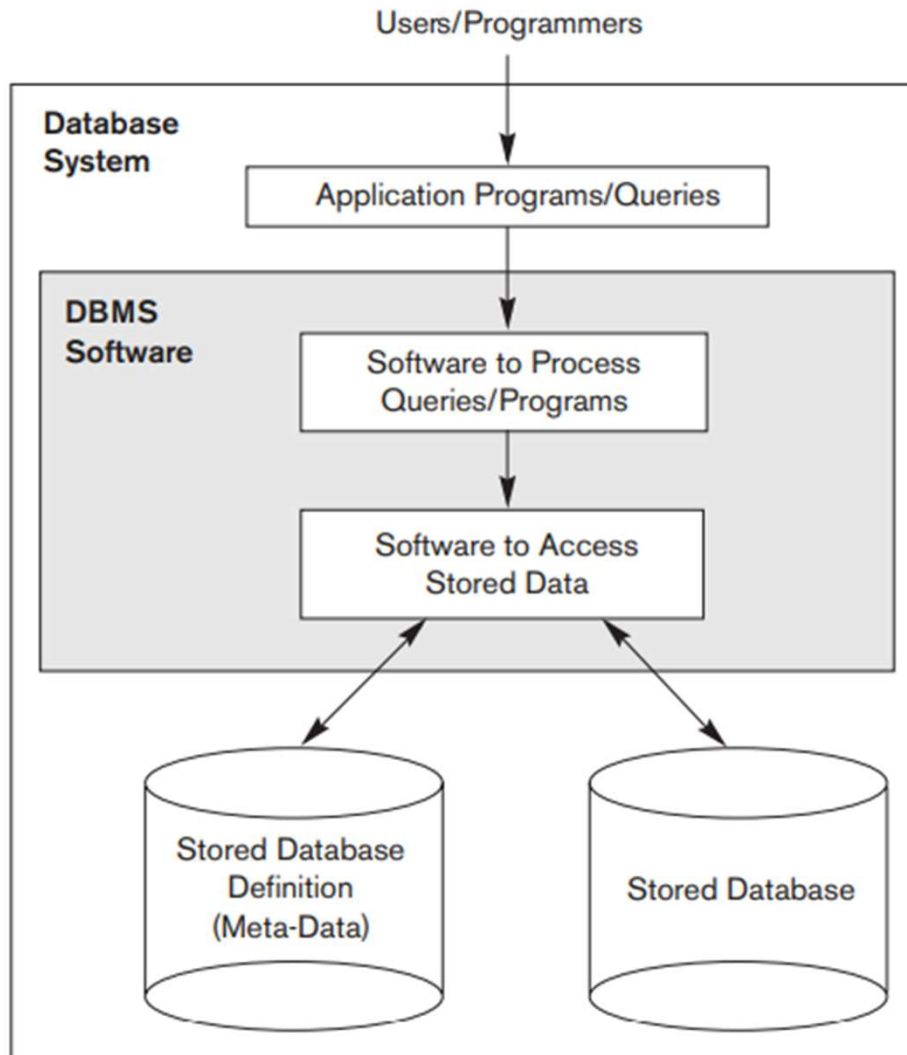
# Design Approach

- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two Steps of doing so:
  - **Entity Relationship Model**
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram*:
  - **Normalization Theory**
    - This normalization theory tries to capture that what are the **properties that must hold** in this database design, that must be satisfied on this database design, in terms of what is known as **database dependencies**.

# DBMS Architecture



# Simplified DBMS Environmet



# Database Engine

It's a Core of Database Management System

- **Storage manager**

- Components:

- Authorization and integrity manager
    - Transaction manager
    - File manager
    - Buffer manager,
    - Data Structures used : Data Files, Data Dictionaries, Indices

- **Query processor**

- Components

- DDL interpreter
    - DML compiler
    - Query evaluation engine



# Storage Management

- **Storage manager** is 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
- Issues:
  - Access to the storage
  - Organization of Files
  - Indexing and hashing
  - Eg. Binary search tree Vs. Auxiliary search tree using indexing

# Storage Manager -Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- 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.

# Query Processing (Cont.)

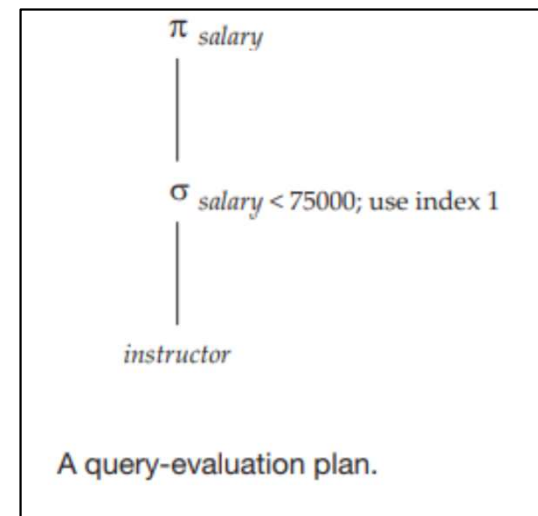
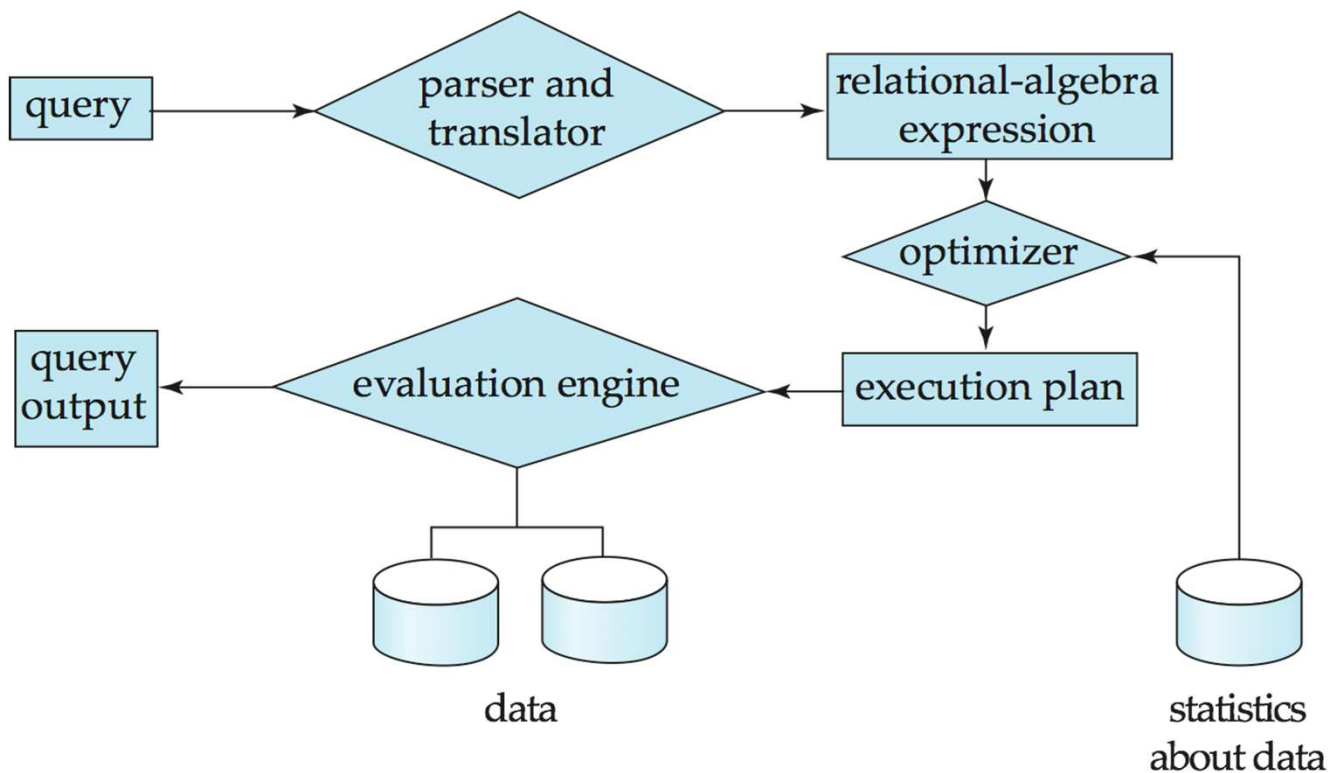
- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

# Query Processing

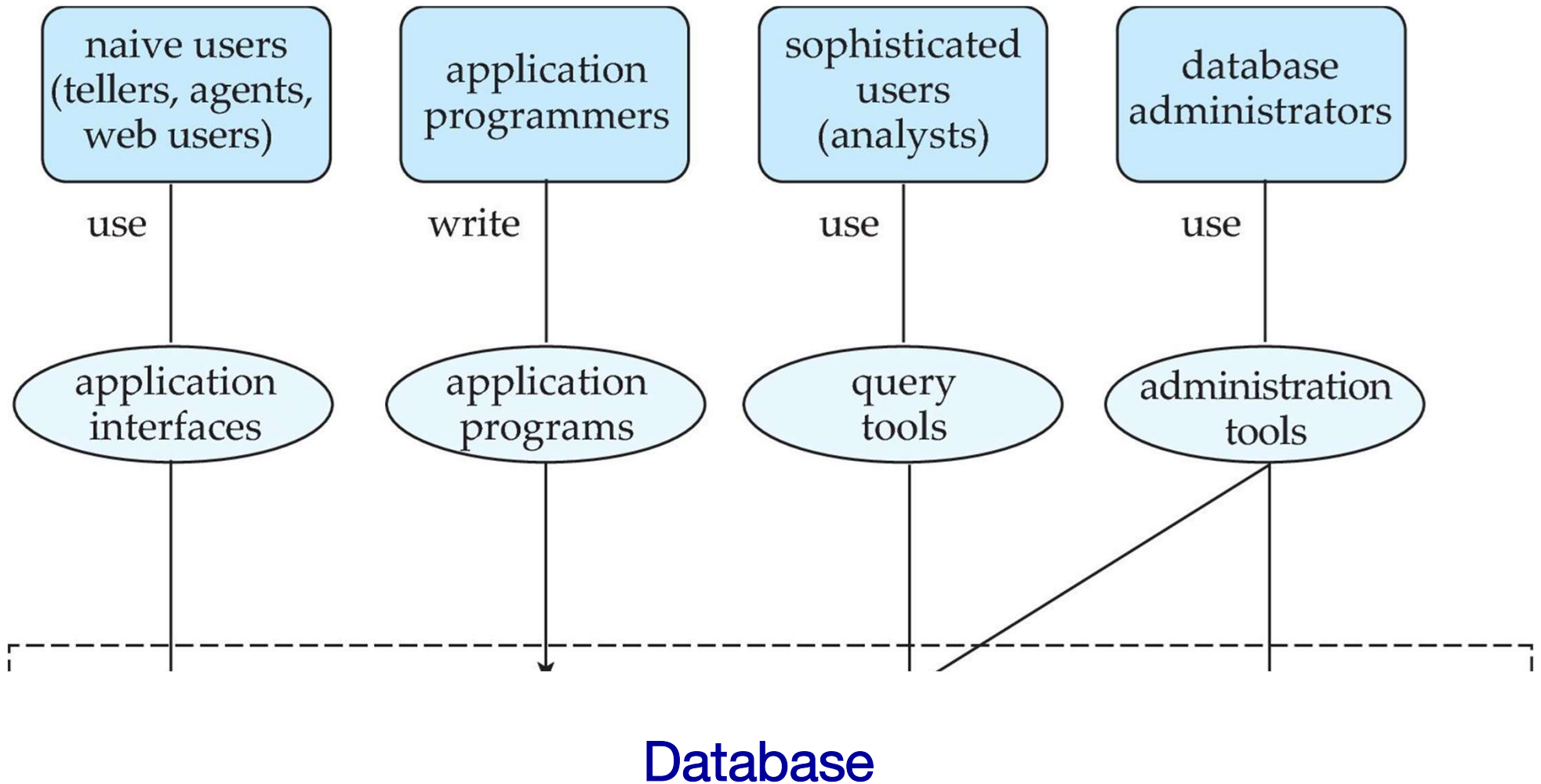
1. Parsing and translation
2. Optimization
3. Evaluation

```
select salary
from instructor
where salary < 75000;
```

- $\sigma_{\text{salary} < 75000} (\Pi_{\text{salary}} (\text{instructor}))$
- $\Pi_{\text{salary}} (\sigma_{\text{salary} < 75000} (\text{instructor}))$



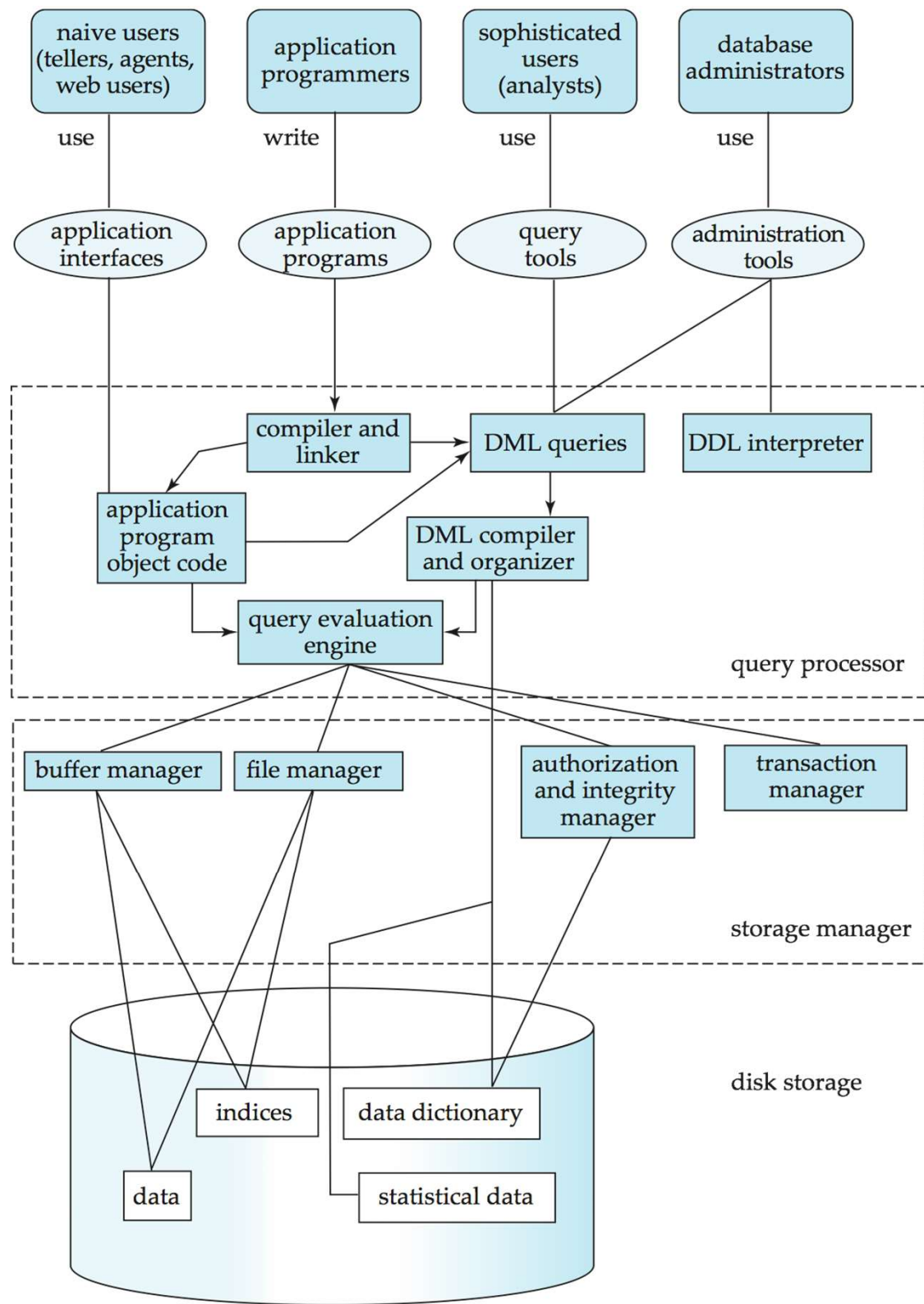
# Database Users and Administrators



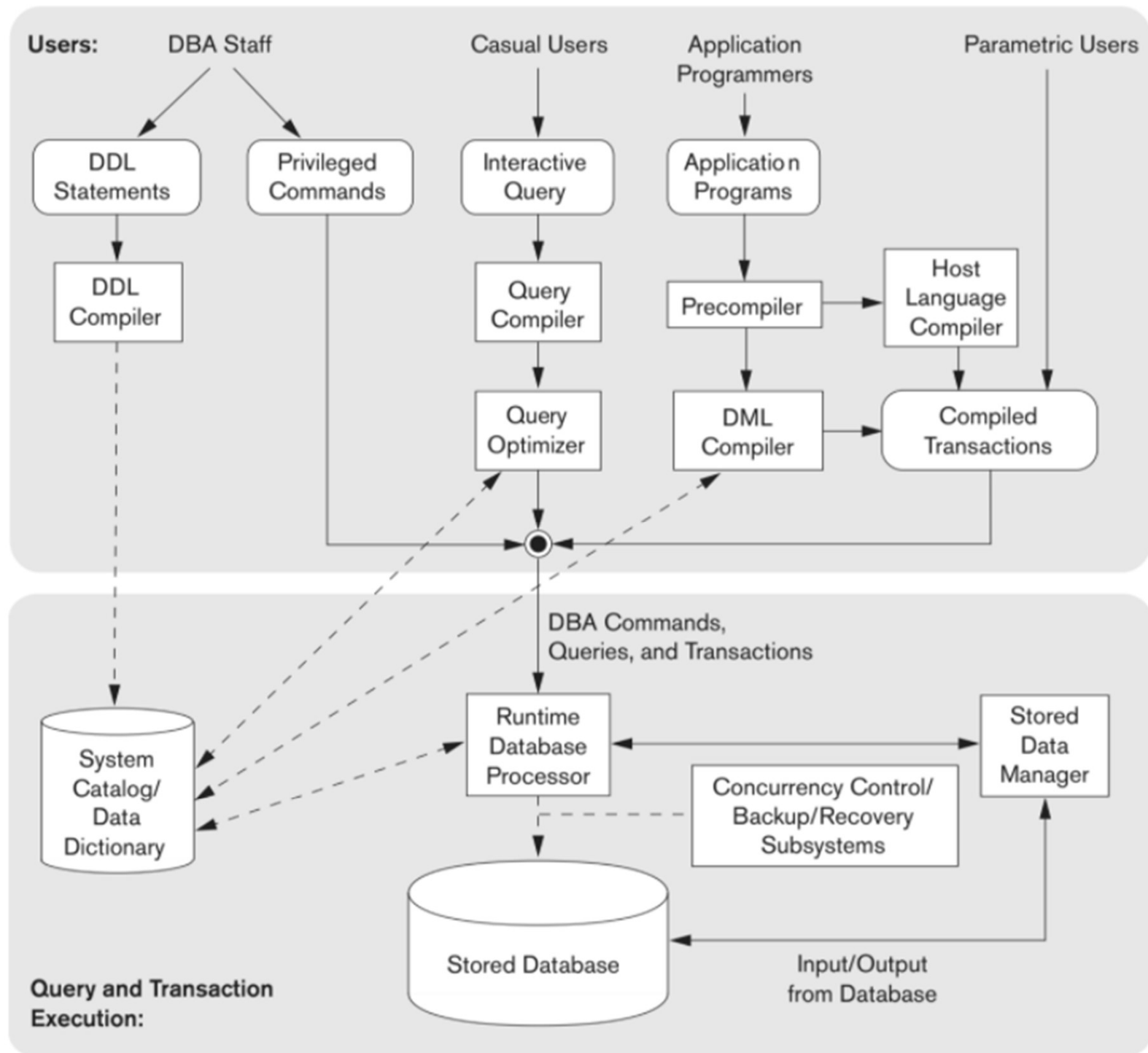
# **Roles and Responsibilities of DBA**

- Design of the Conceptual and Physical Schemas
- Security and Authorization
- Data Availability and Recovery from Failures
- Database Tuning

# Database System Internals



# Database System Internals





**End of Unit I**