UCS310
Database Management System

Database System Concepts and Architecture

Lecture-03

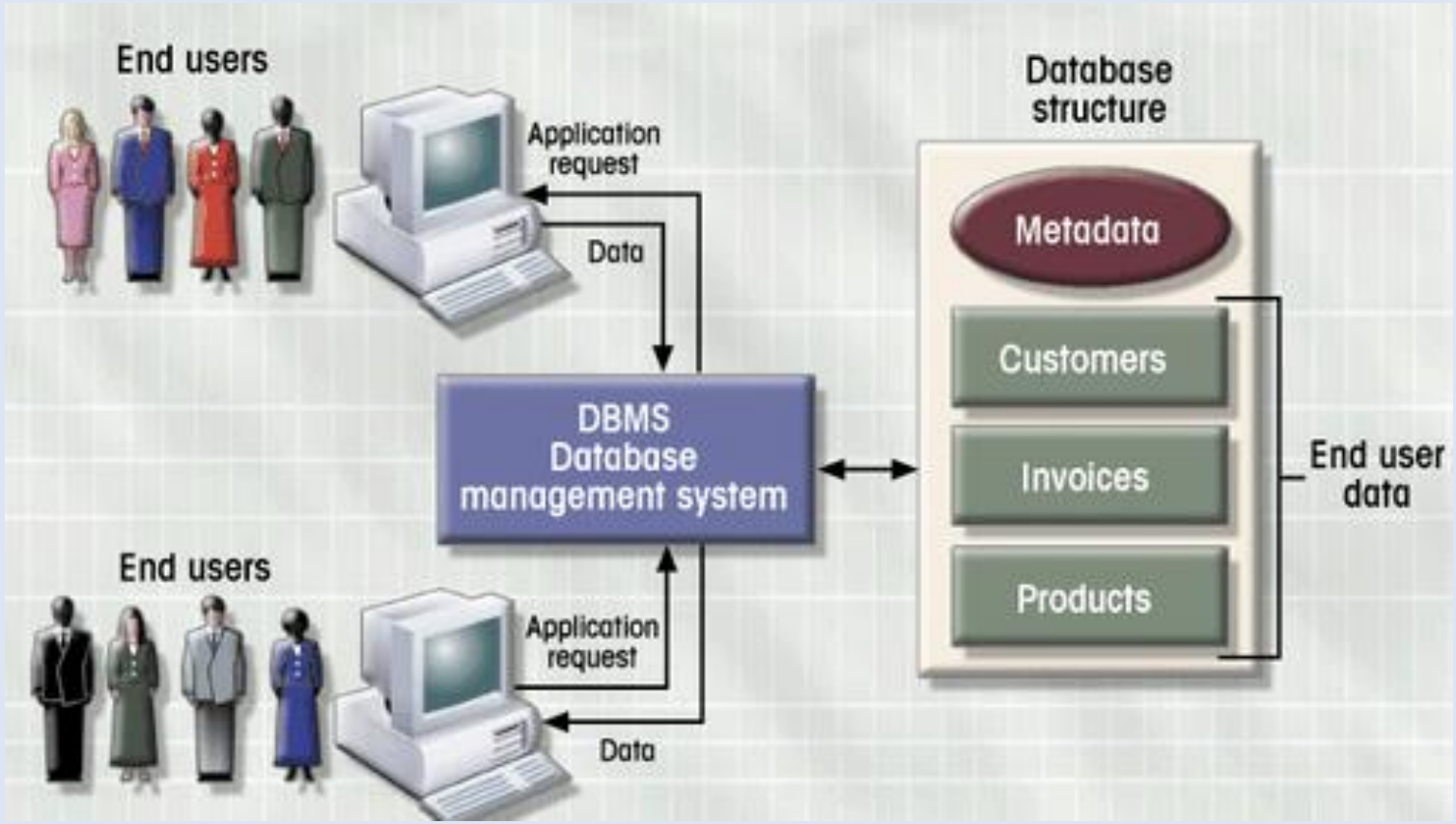
Date: 19 Jan 2023

Dr. Sumit Sharma
Assistant Professor
Computer Science and Engineering Department
Thapar Institute of Engineering and Technology, Patiala

Recap

- Basic notion & terminology of DBMS
- Role of data models & languages
- To understand the historical perspective
- Approaches to database design
  - Views of Data
  - Instances and Schemas
  - Data Abstraction
  - Data Models
  - Database Language (DML, DDL)

DBMS Manages Interaction



Components of the DBMS Environment

- Hardware
- Software
- Data
- Users
- Procedures

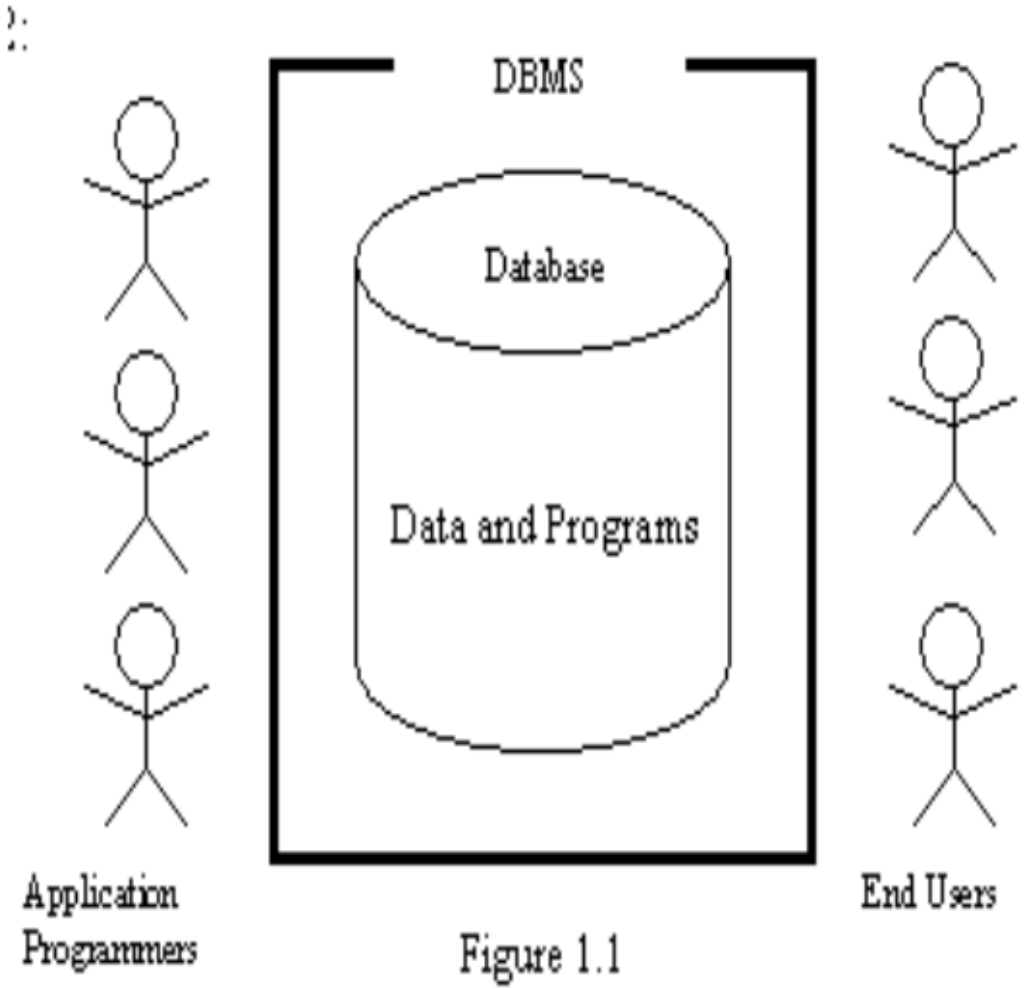


Figure 1.1

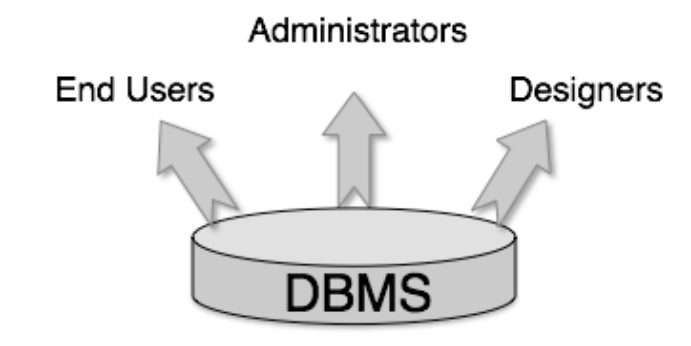
Hardware:

Actual computer system used for keeping and accessing the database

Software:

- Actual DBMS.
- Between the physical databases (i.e. the data as actually stored) and the system users
- A layer of software, usually called the Database Management System or DBMS

Users of DBMS



Administrators:

- Administrators maintain the DBMS
- They are responsible
  - Set up and Installation
  - Storage and configuration
  - Backup and Recovery
  - Look after its usage and by whom it should be used
  - Create access profiles for users and apply limitations to maintain isolation and force security

Users of DBMS

Database Designers:

- Group of people who actually work on the designing part of the database
- Keep a close watch on what data should be kept and in what format
- Identify and design the whole set of entities, relations, constraints, and views

Users of DBMS

End Users:

- Who actually takes the benefits of having a DBMS

- The End user can be of the following types:
  - **Online users** can directly access the database through an online terminal or user interface. They know the presence of the database and can manipulate the database using SQL language
  - **Naïve Users** who do not have deep knowledge about the database. These users use the database through menu-oriented application programs
  - **Programmer**: The user of the database who develops the application programs for online users and Naïve users

Procedures

Instructions and rules that govern the design and use of the database

The users of the system and the staff that manages the database require documented procedures on how to use or run the system

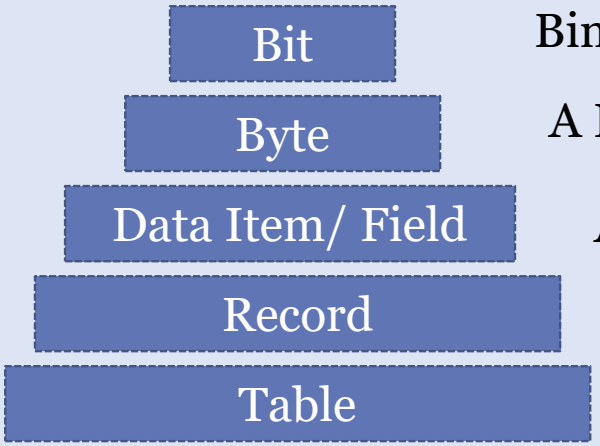
These may consist of instructions on how to:

- Log on to the DBMS
- Use a particular DBMS facility or application program
- Start and stop the DBMS
- Make backup copies of the database
- Handle hardware or software failures

Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine
- 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.

Arrangement of Data in Database



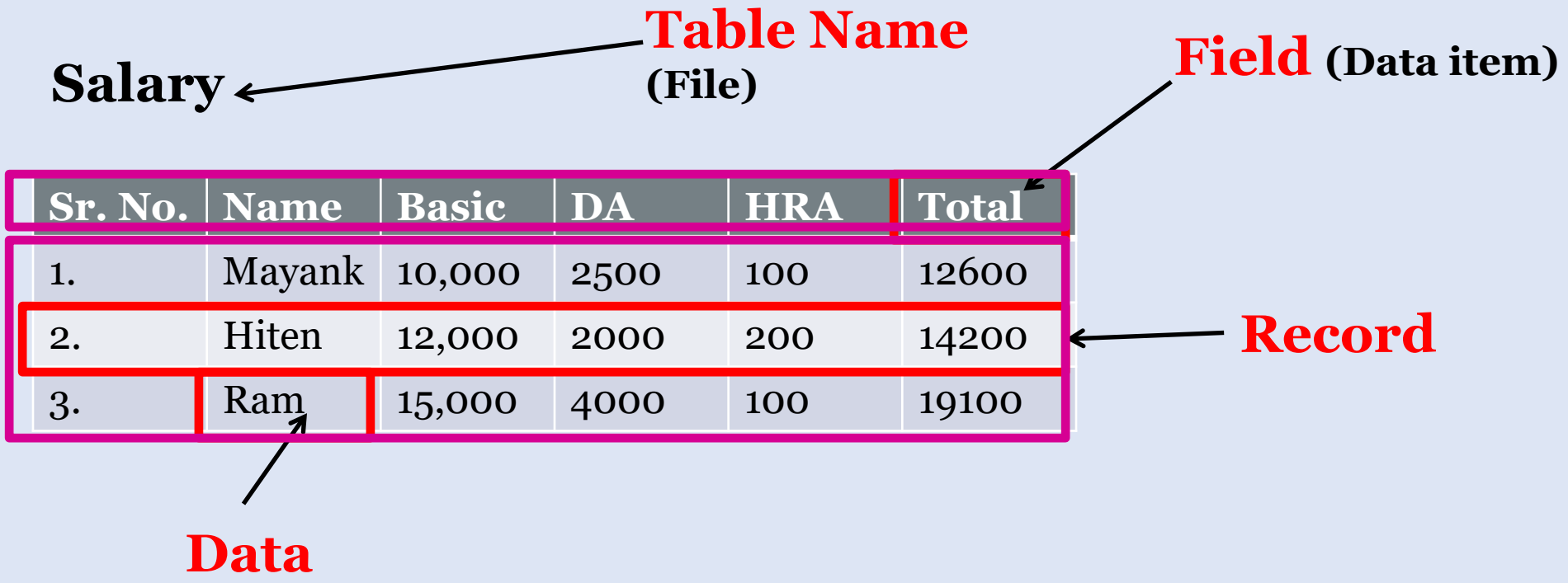
Binary digit 0 or 1

A Logical collection of bits

A Logical collection of bytes

Collection of relevant fields

Collection of Similar Records



Sr. No.	Name	Basic	DA	HRA	Total
1.	Mayank	10,000	2500	100	12600
2.	Hiten	12,000	2000	200	14200
3.	Ram	15,000	4000	100	19100

Advantages of DBMS

- **Controlling Redundancy**

- **Integrity can be enforced**

Integrity of data means that data in database is always accurate, such that incorrect information cannot be stored in database.

- **Inconsistency can be avoided**

When the same data is duplicated and changes are made at one site, which is not propagated to the other site, it gives rise to inconsistency and the two entries regarding the same data will not agree.

Advantages of DBMS

- Data can be shared
- Providing Backup and Recovery
- Standards can be enforced
- Restricting unauthorized access
- Solving enterprise requirement than individual requirement

Disadvantages of DBMS

- Complexity
- Size
- Performance
- Higher impact of a failure
- Cost of DBMS
- Additional Hardware costs
- Cost of Conversion

Database Design

The process of designing the general structure of the database:

- **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

- Why the database systems are more capable than the file-based systems
  - Key components that make this happen are part of the **database engine**.

Database Engine

- Why the database systems are more capable than the file-based systems
  - Key components that make this happen are part of the **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: to ensure system consistency even at failure
  - File manager: allocation of space on disk and data structure for disk storage
  - Buffer manager: fetching data

Storage Manager

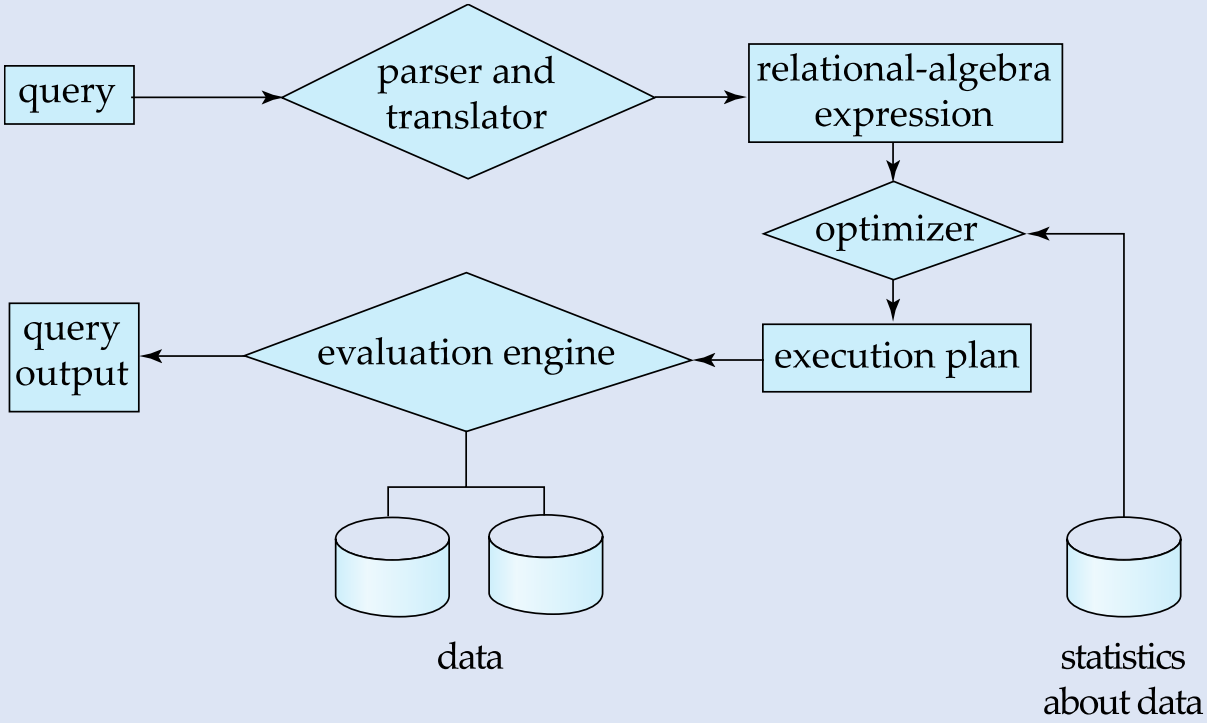
- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide fast access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

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processor

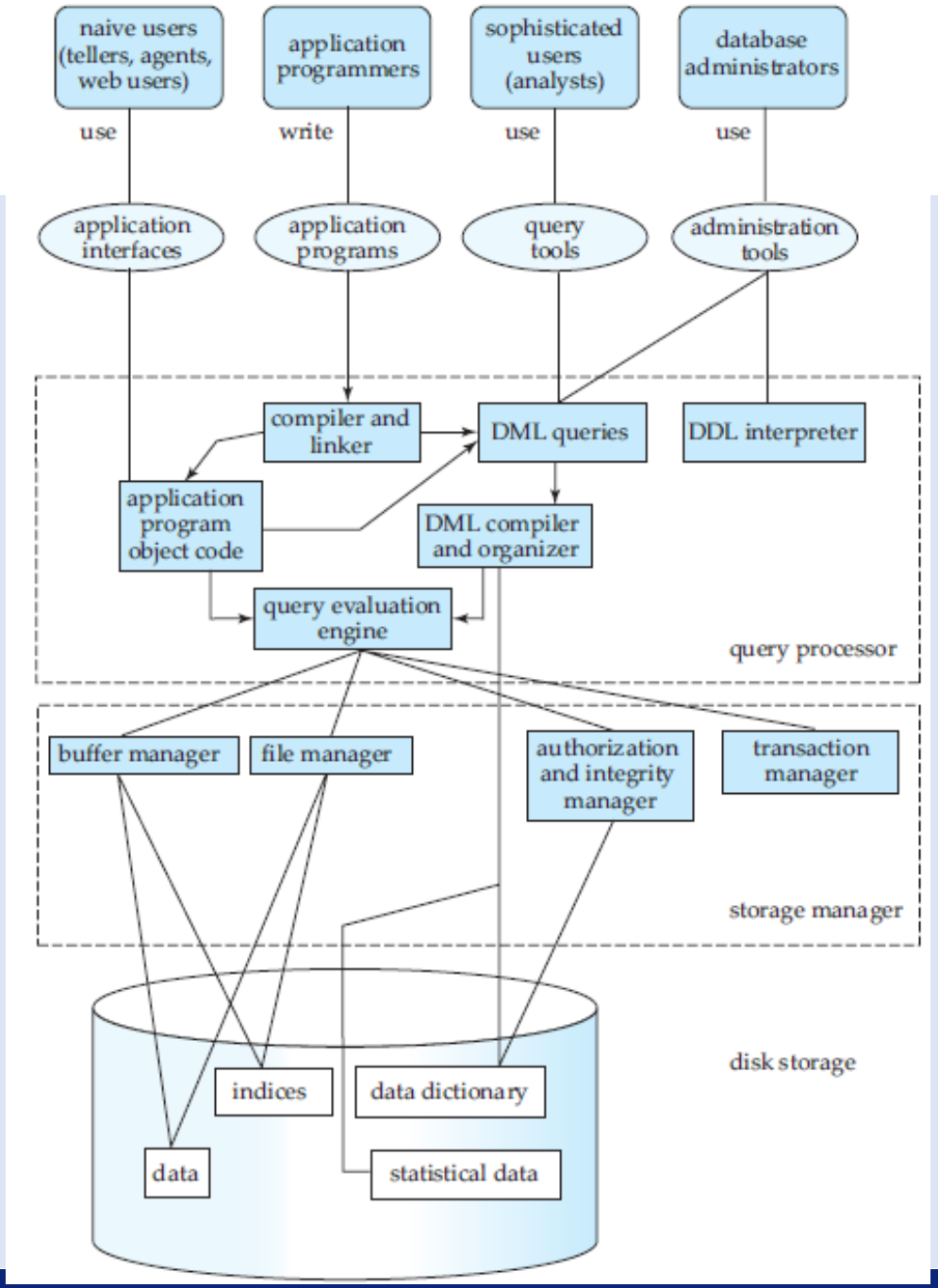
The query processor components include:

- **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

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

System Structure

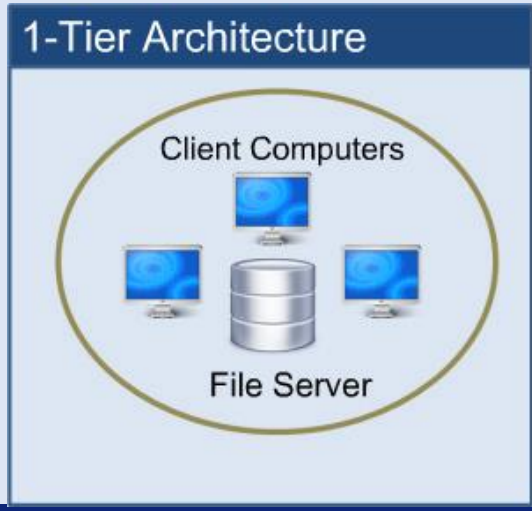


DBMS Architecture

- The design of a DBMS depends on its architecture
- It can be centralized or decentralized or hierarchical.
- The architecture of a DBMS can be seen as either single-tier or multi-tier.
- An **n-tier** architecture divides the whole system into related but independent **n-modules**, which can be independently modified, altered, changed, or replaced.

1-tier architecture

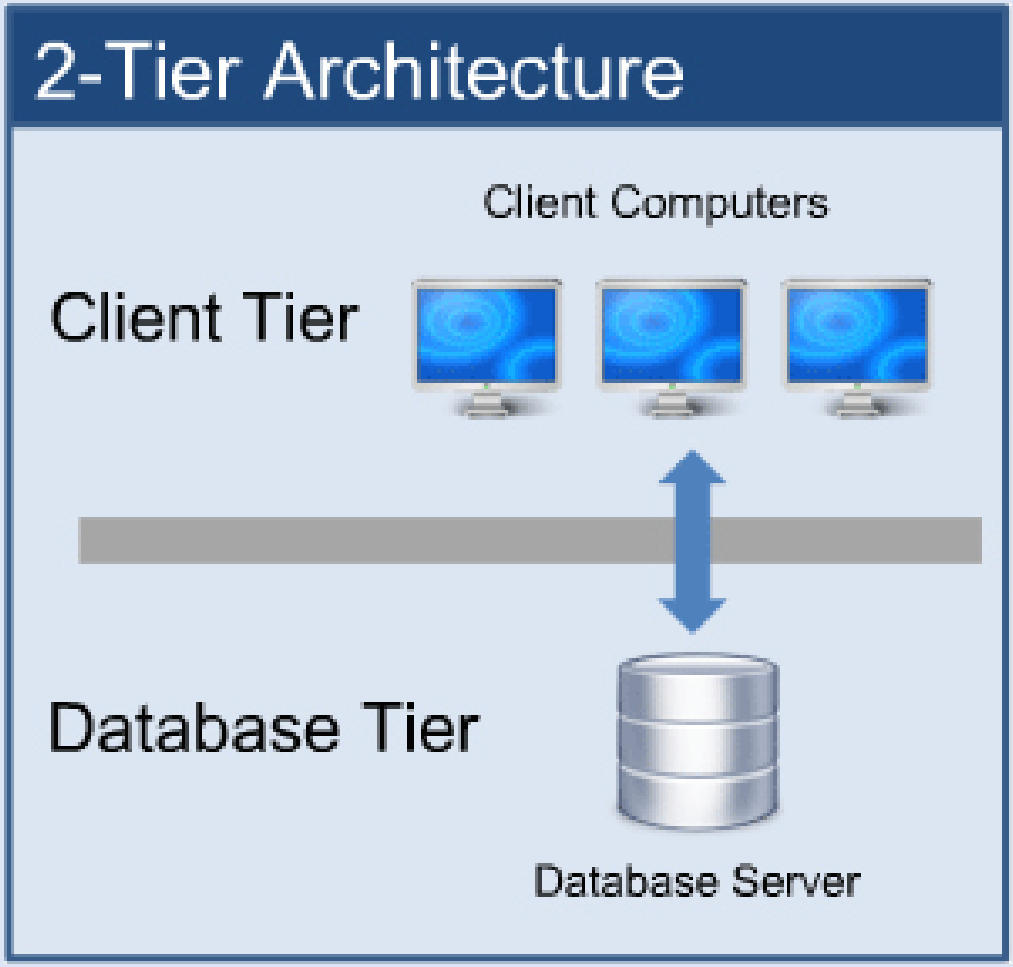
- In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it.
- Any changes done here will directly be done on the DBMS itself.
- Database designers of the database normally prefer to use single-tier architecture.
- 1-tier architecture is simple and cheap, but usually unsecured and data can easily be lost if you are not careful.



2-tier architecture

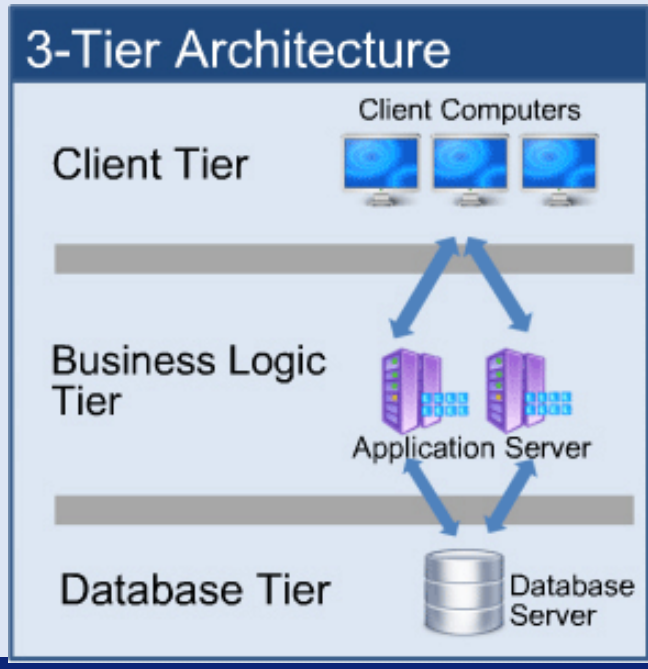
- This architecture is also called **Client-Server architecture** because of the two components:
  - Client that runs the application and
  - Server that handles the database back-end
- If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed
- Programmers use 2-tier architecture where they access the DBMS by means of an application.
- Here the application tier is entirely independent of the database in terms of operation, design, and programming.

2-tier architecture



3-tier architecture

- A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database.
- It is the most widely used architecture to design a DBMS.



- **Database (Data) Tier:**
  - At this tier, the database resides along with its query processing languages.
  - We also have the relations that define the data and their constraints at this level.
  - The database tier is not aware of any other user beyond the application tier.

- **Application (Middle) Tier/Business Logic Tier:**
  - At this tier reside the application server and the programs that access the database.
  - For a user, this application tier presents an abstracted view of the database.
  - The application layer sits in the middle and acts as a mediator between the end-user and the database.

- **User (Presentation) Tier/ Client Tire:**
  - End-users operate on this tier and they know nothing about any existence of the database beyond this layer.
  - At this layer, multiple views of the database can be provided by the application.
  - All views are generated by applications that reside in the application tier.

ID	*name*	*salary*	*dept_name*	*building*	*budget*
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

Instructor

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) Instructor

(b) Department

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

Data Model

- Data models define how the logical structure of a database is modeled
- Data models define how data is connected to each other and how they are processed and stored inside the system
- Two types of data models (mainly covered in this course):
  - **Entity –Relationship Model**
  - **Relational Model**

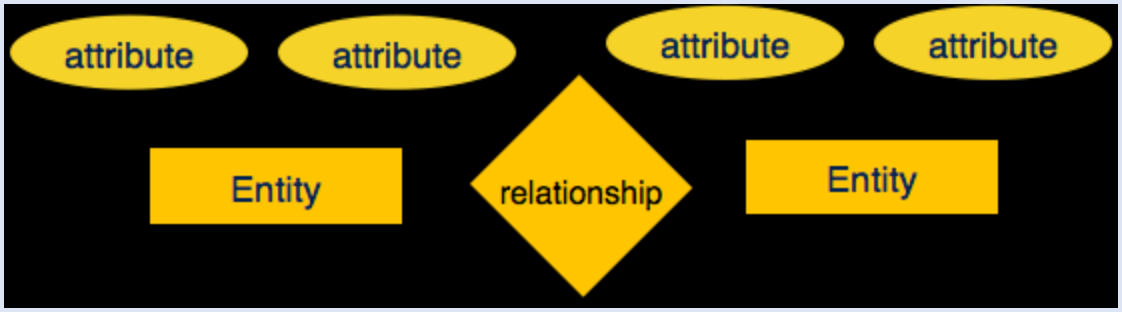
Both provide a way to describe the design of a database at the logical level

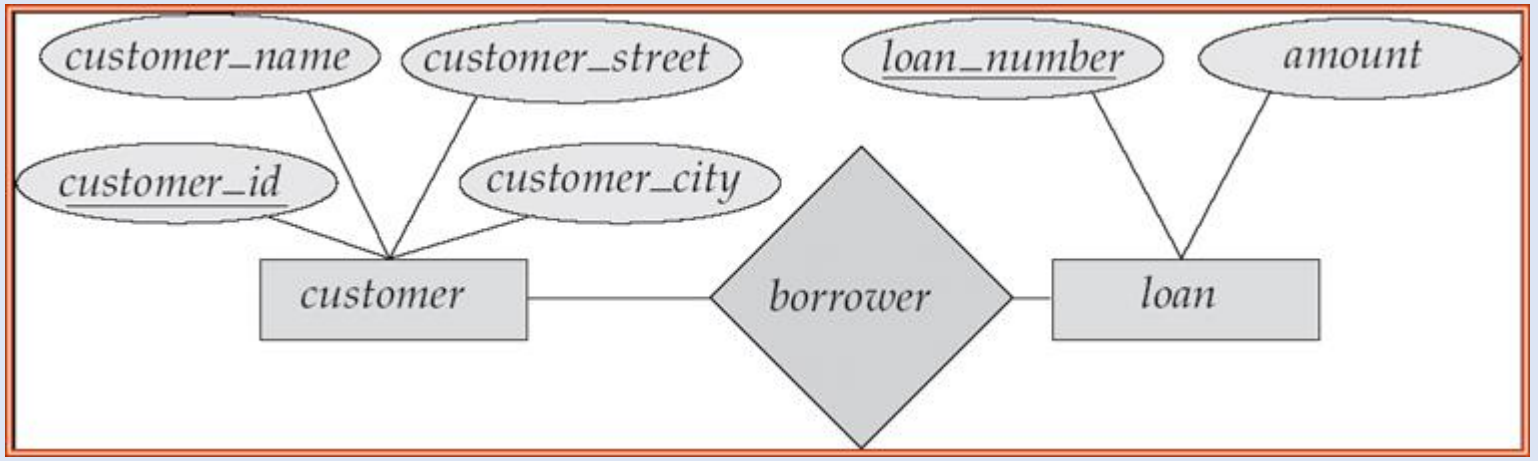
Entity-Relationship(ER) Model

- Entity-Relationship (ER) Model is based on perceptions of real-world entities and relationships among them
- While formulating real-world scenarios into the database model, the ER Model creates
  - entity set,
  - relationship set,
  - general attributes and
  - constraints.

- ER Model is best used for the conceptual design of a database.
- ER Model is based on –
  - Entities and their attributes.
  - Relationships among entities

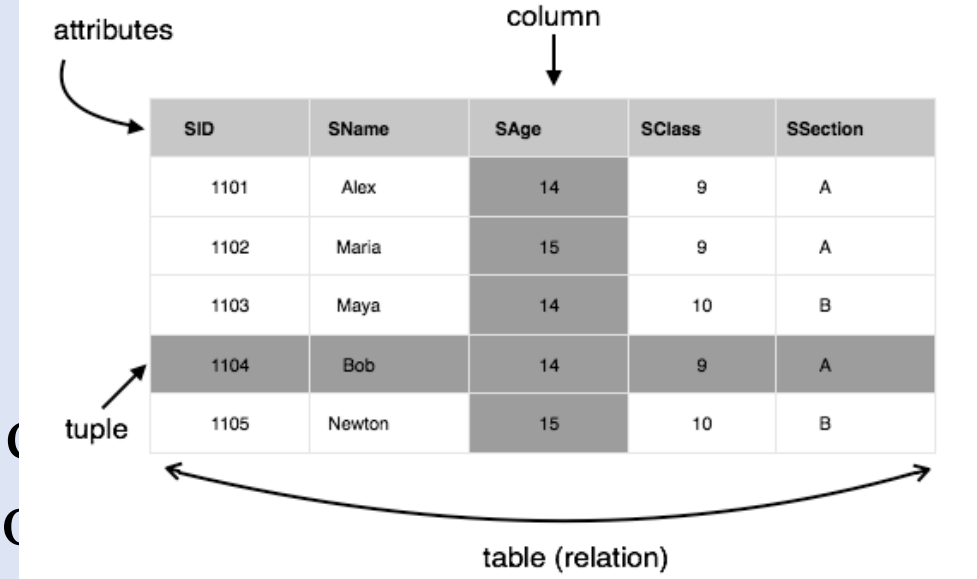
Entity-Relationship(ER) Model



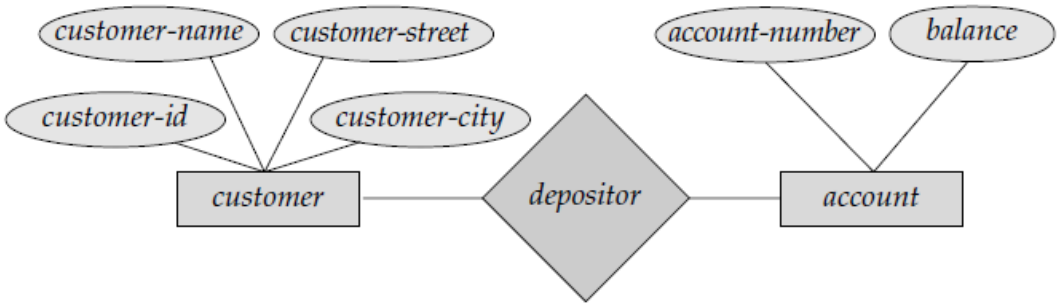


Relationship Model

- Relational model uses a collection of tables to represent both data and relationships among those data
- Each table has multiple columns and each column has a unique name



- The database design is c
translated to the Relatic



customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table