CS3400: Database Management Systems (DBMS)

P.Krishna Reddy
pkreddy@iiit.ac.in

Outline

- **Data, Information and DBMS**
- Historical perspective
- Early Database systems (with drawbacks)
- Abstract views of data
- Data models
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS Structure
- Course plan

Data and information

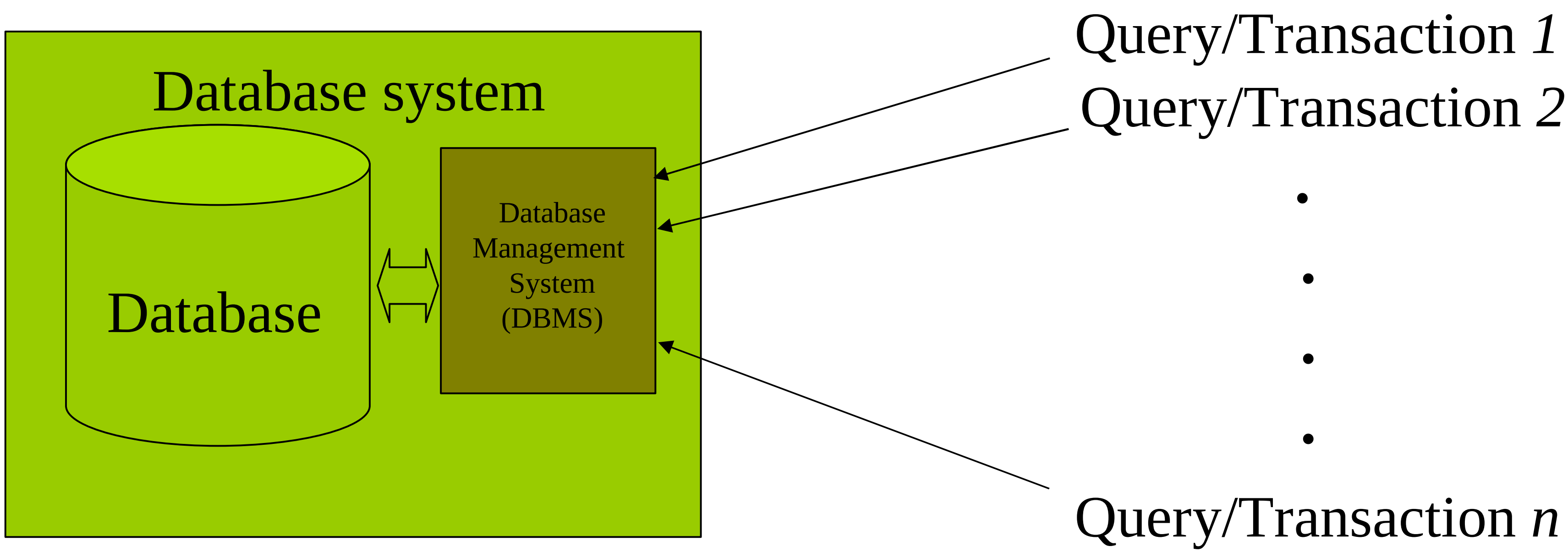
- Data
  - Symbols
  - Collection of facts
  - It simply exists and has no significance beyond its existence.
  - It does not have any meaning of itself
- Information:
  - Data that are processed to be useful, **provides answers to who, what, and when questions.**
  - Processed data
  - Data that has been given meaning by relational connection
    - Ex: If temperature drops 15 degrees and then it started raining.

Database Management System (DBMS)

- Database:
  - the collection of data
  - information about a particular enterprise
- A DBMS is a collection of interrelated data and Set of programs to access the data
- Goal:
  - DBMS provides an environment that is both *convenient* and *efficient* to use.
- Management of data:
  - Defining structures for storing information
  - Providing mechanisms for the manipulation of information
  - Safety of information
  - Parallel access

Database

- Data is a collection of facts
- A database is a collection of related data which is both integrated and shared.
  - **Integrated:** Unification of several distinct files
  - **Shared**: Database can be shared between users
- Users access the database through queries/transactions.



Database Applications

- **Banking:**
  - Customer information, accounts, loans and banking transactions
- **Airlines:**
  - reservations, schedules
  - Airlines were among the first to use databases in a geographically distributed manner
- **Universities:**
  - student information, registration, grades
- **Credit card transactions:**
  - for purchases on credit cards and generation of monthly statements

Database Applications

- **Telecommunications:**
  - Keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about communication networks
- **Finance:**
  - For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds
- **Sales:**
  - customers, products, purchases
- **Manufacturing:**
  - production, inventory, orders, supply chain
- **Human resources:**
  - employee records, salaries, tax deductions

Growth of database use

- Databases touch all aspects of our lives
- In the early days, very few people interacted directly with databases
- Now, users can directly access databases
  - Automatic teller machines, Phone interfaces, Internet revolution
- Every action is being recorded and maintained as a database
- Accessing databases is essential part of everyone's life
- Importance:
  - Oracle is one of the largest software companies
  - Databases systems are an important part of the product line of Microsoft and IBM.
  - Led to data mining/data warehousing
  - About 70% of Industry jobs

Outline

- Data, Information and DBMS
- **Historical perspective**
- Early Database systems (with drawbacks)
- Abstract views of data
- Data models
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS Structure
- Course plan

Historical Perspective

- Early 1960s
  - Integrated data store, first general-purpose DBMS designed by Charles Bachman at GE
  - Formed basis for **network data model**
  - Bachman received Turing Award in 1973 for his work in database area

Historical Perspective

- Late 1960s
  - IBM developed Information Management System (IMS), used even today in many major installations
  - IMS formed the basis for **hierarchical data model**
  - American Airlines and IBM jointly developed SABRE for making airline reservations
  - SABRE is used today to populate Web-based travel services such as Travelocity

Historical Perspective

- 1970
  - Edgar Codd, at IBM’s San Jose Research Laboratory, proposed **relational data model**.
  - It sparked the rapid development of several DBMSs based on relational model, along with a rich body of theoretical results that placed the field on a firm foundation.
  - Codd won 1981 Turing Award.
  - Database systems matured as an academic discipline
  - The benefits of DBMS were widely recognized, and the use of DBMSs for managing corporate data became standard practice.

Historical Perspective

- 1980s
  - Relational data model consolidated its position as dominant DBMS paradigm, and database systems continued to gain widespread use
  - SQL query language, developed as part of **IBM's System R project**, is now the standard query language
  - SQL was standardized in late 1980s, and current standard **SQL:1999** was adopted by ANSI and ISO
    - ANSI stands for American National Standards Institute

Historical Perspective

- **Late 1980s till 1990s**
  - Considerable research into more powerful query language and richer data model, with emphasis on supporting complex analysis of data from all parts of an enterprise
  - Several vendors, e.g., **IBM's DB2, Oracle 8, Informix UDS**, extended their systems with the ability to store new data types such as images and text, and to ask more complex queries
  - **Data warehouses** have been developed by many vendors to consolidate data from several databases, and for carrying out specialized analysis

Outline

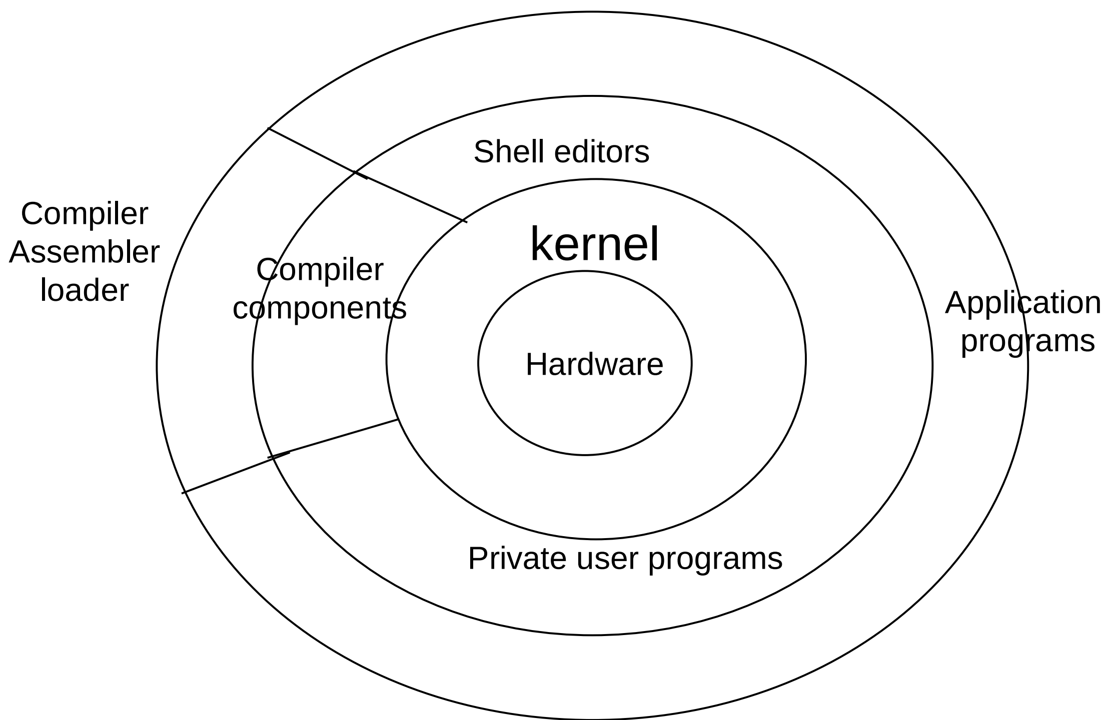
- Data, Information and DBMS
- Historical perspective
- **Early Database systems (with drawbacks)**
- Abstract views of data
- Data models
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS Structure
- Course plan

Early database systems

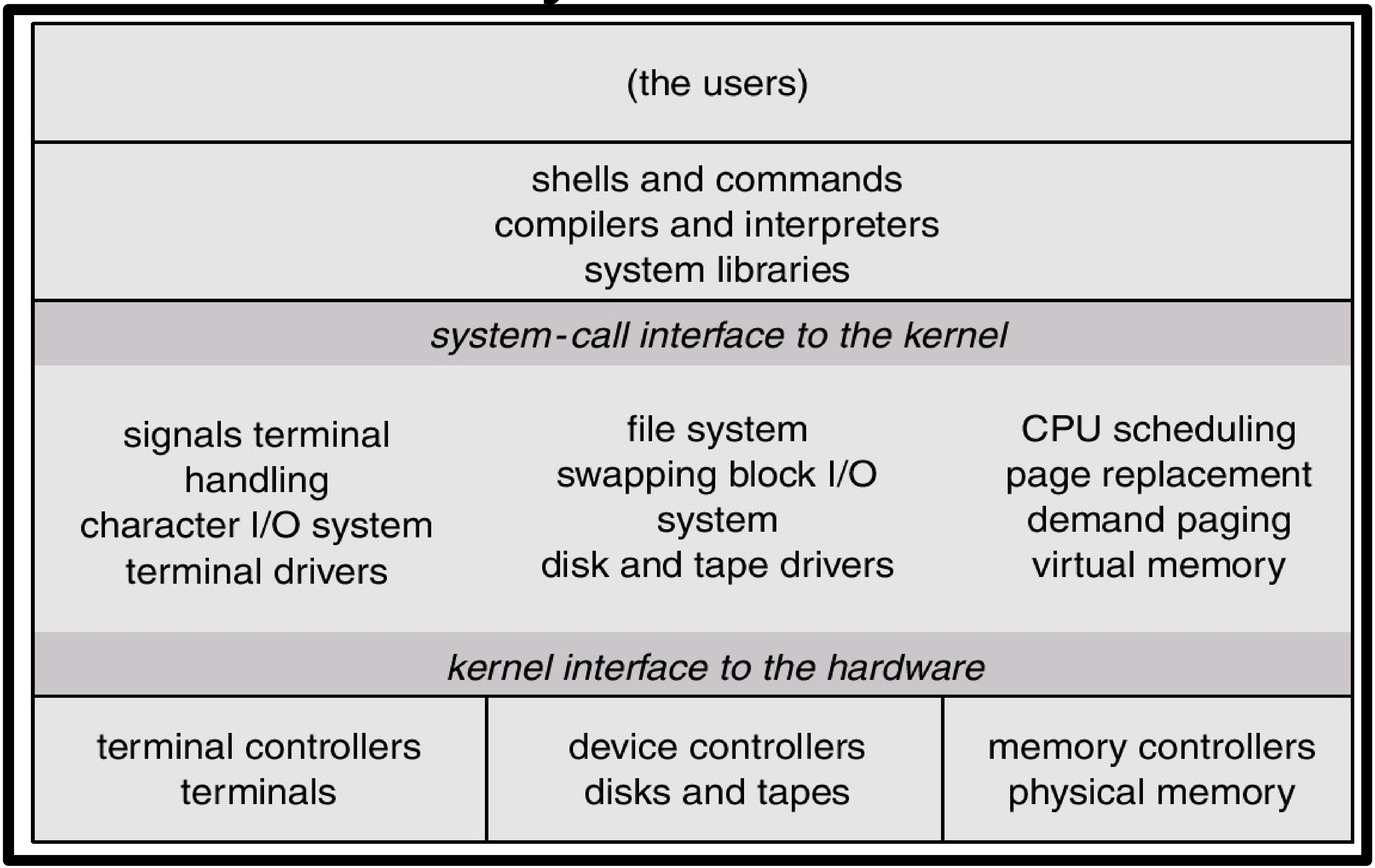
- Database applications were built on top of file systems
- Database information is stored in operating system files.
- To allow manipulation, the system has number of application program that manipulates the files
  - A program to debit or credit an account, to add an account, to find the balance of an account, to generate monthly statements
- System programmers wrote applications to meet the needs of a bank.
- New application programs are added to the system as the need arises.
- As time goes by the system requires more files and more application programs.
- The file processing system is supported by by conventional OS

UNIX System Structure

- UNIX kernel
  - Consists of everything below the system-call interface and above the physical hardware
  - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.



UNIX System Structure



Drawbacks of file systems to manage data

- **Data redundancy and inconsistency**
  - Different programmers create the files and application programs over a longer period.
    - Multiple file formats, Programs in multiple languages
    - Same information can be duplicated in several places.
      - The address and telephone number of a customer may appear in saving account record and checking account records.
  - The redundancy leads to high storage cost and access cost..
  - The redundancy may lead to data inconsistency.
    - Various types of data may not agree.
    - Changing address may be reflected in saving account record but not else where.

Drawbacks: Cont.

- **Difficulty in accessing data**
  - Ex: find customers within a particular postal-code area.
  - Since designers could not anticipate this requirement, a new program has to be written to carry out each new task.
  - Several days later, the bank officer may want the list of customers who have an account balance of Rs. 10000/- more.
  - **Conventional file-processing environments do allow needed data to be retrieved in a efficient and convenient manner.**
- **Data isolation —**
  - Data is in multiple files and formats
  - Writing application programs is difficult.

Drawbacks: Cont.

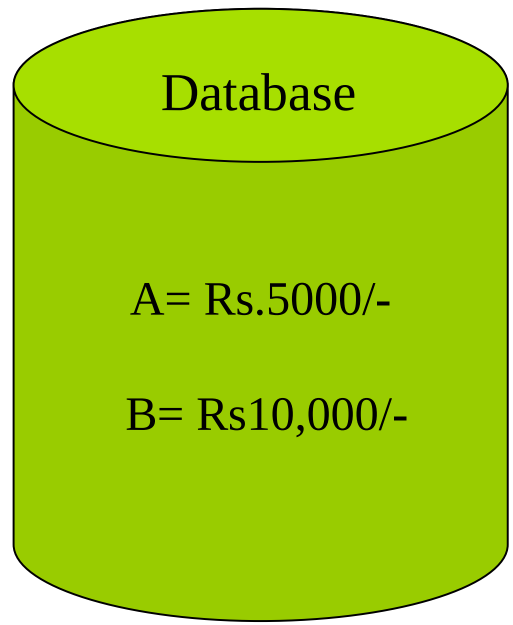
- **Integrity problems**
  - Data values stored in the database must satisfy certain types of integrity constraints.
    - (e.g. account balance > Rs.1000) become part of program code
  - Hard to add new constraints or change existing ones
  - The problem is compounded when constraints involve several data items from different files.

Drawbacks: Cont.

- Atomicity problems
  - Computer system may subject to failure
  - If a failure occurs, the data be restored to the consistent state that existed prior to the failure.
    - Failures may leave database in an inconsistent state with partial updates carried out
  - **E.g.** Transfer $1000/- from account A to account B.

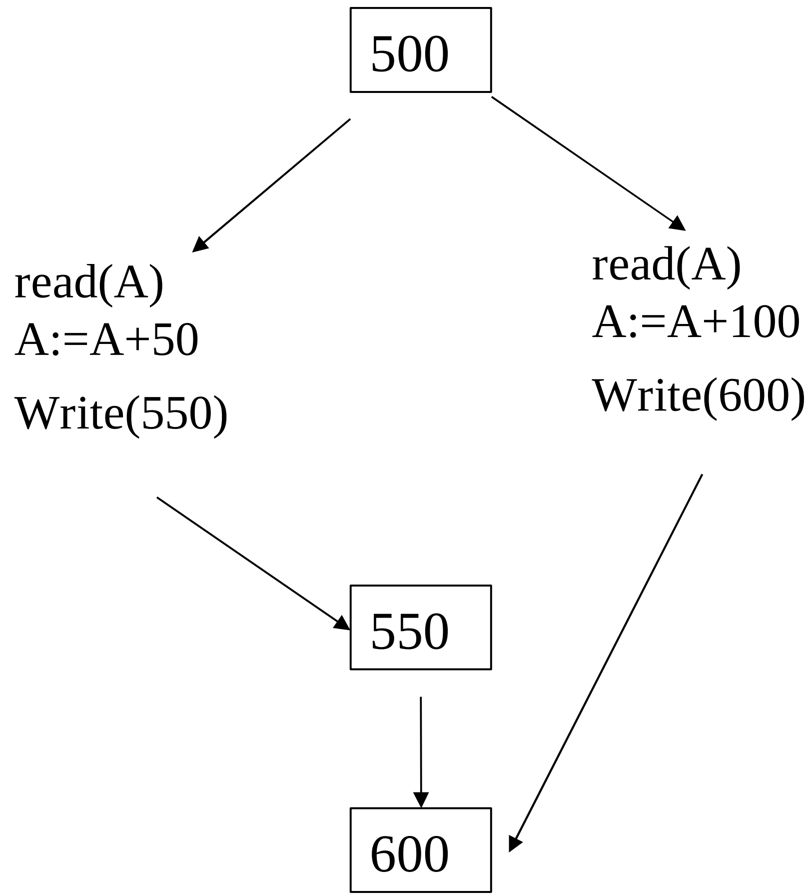
Transfer:

```
{
    Read A;
    Read (B)
    A=(A-1000);
    Read(B);
    B=(B+1000);
    Write (A)
    Write(B);
    commit;
}
```



Drawbacks cont. Concurrent access anomalies

- When transactions execute concurrently, some of them access same data.
- Uncontrolled interleaving of sub-operations of transactions can lead to many problems such as lost update problem and dirty read problem.
  - **Lost Update:**
    - Program P_1: deposit Rs.50 to account A
    - Program P_2: deposit Rs.100 to account A
  - **Dirty read**: If P_2 reads 550 and P_1 aborts then inconsistency results



Drawbacks Cont.

- **Security problems**
  - Not every user of the database system should be able to access all the data.
    - E.g. in a banking system, payroll personal need to see information about bank employees.
    - They do not need access to information about customer accounts.
    - Enforcing such constraints in file processing systems is difficult.

Purpose of database systems

- These difficulties prompted the development of database systems.
- **Database systems offer solutions to all the above problems**
  - Data redundancy and inconsistency
  - Difficulty in accessing data
  - Integrity
  - Data Isolation
  - Concurrency access
  - Security
  - Reduce application development time
  - Provide many more advantages

Outline

- Data, Information and DBMS
- Historical perspective
- Early Database systems (with drawbacks)
- **Abstract views of data**
- Data models
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS Structure
- Course plan

Abstract View of Data

- A database system aims to provide users with an ***abstract view*** of data by hiding certain details of how data is stored and manipulated.
- The starting point for the design of a database
  - An abstract description of the information requirements of the organization.

Abstract view

- **An Abstract View**
  - For example, in the estate agent example we may build an abstract view which contains the following:
    - Staff, Property, Owner, and Renter (maybe others, too).
    - describing properties or qualities of each entity (e.g. Staff have names, addresses, and salaries).
- Since a database is a shared resource, we may also be concerned to provide different users with *different views* of the data held in the database.

The ANSI-APARC architecture

- ***ANSI-SPARC*** architecture (1975).
  - American National Standards Institute (ANSI)
  - Standards Planning And Requirements Committee (SPARC)
- Although this never became a formal standard, it is useful to help to understand the functionality of a typical DBMS.
- The **ANSI-SPARC** model of a database identifies *three distinct levels* at which data items can be described.
- These levels form a *three-level architecture* comprising:
  - an view or *external* level,
  - a logical or *conceptual* level, and
  - physical or *internal* level.

View of Data

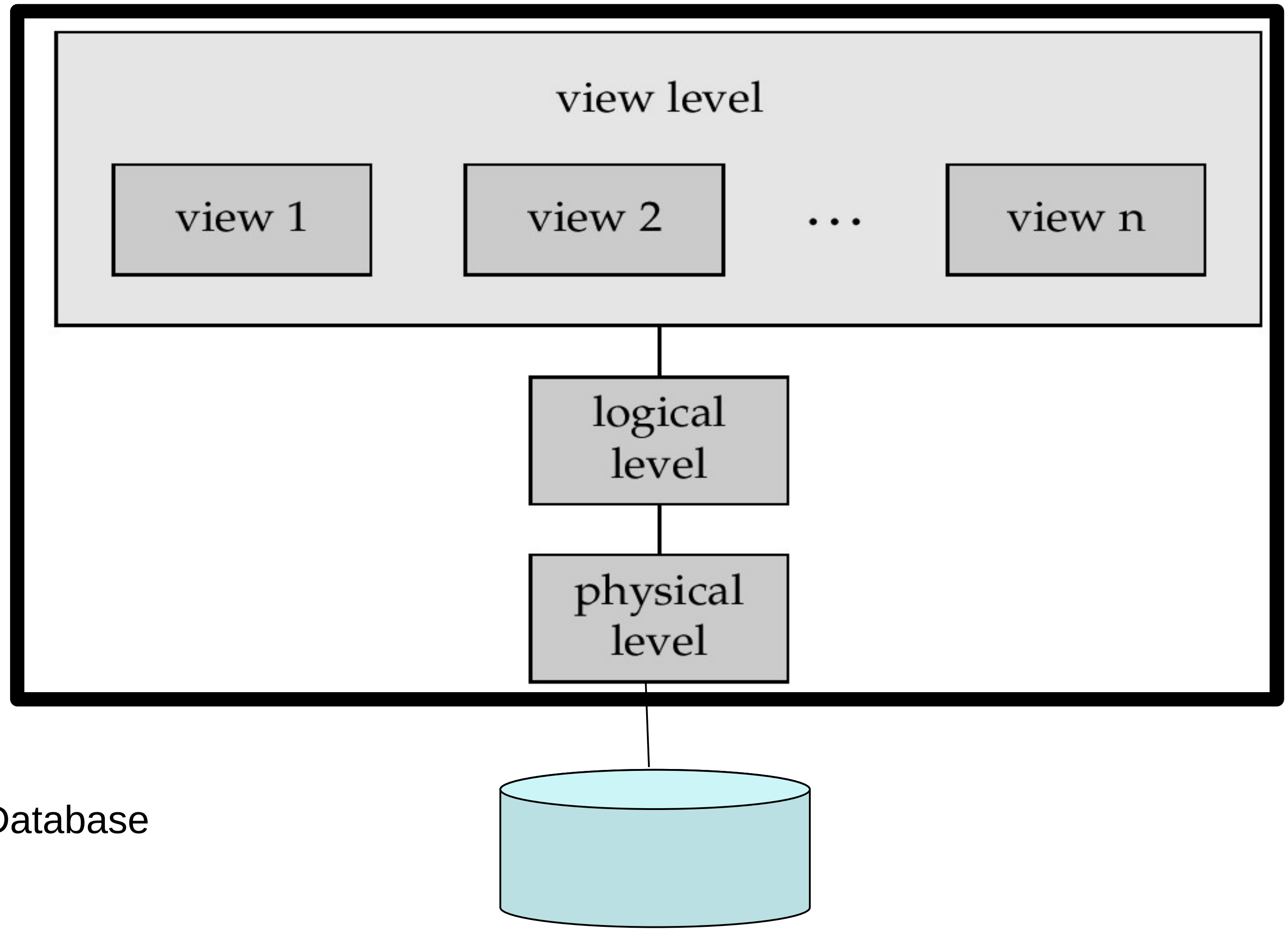
An architecture for a database system

User view of
the database

Community view
of the database

Physical representation
of the database

Database



The three-level architecture

- The objective of the three-level architecture is to separate the users‘ view(s) of the database from the way that it is physically represented.
- The Three-Level Architecture is desirable for the following reasons:
  - **It allows independent customized user views**.
    - Each user should be able to access the same data ,but have a different customized view of the data. These should be independent: changes to one view should not affect others.
  - **It hides the physical storage details for users**.
    - Users should not have to deal with physical database storage details. They should be allow work with the data itself, without concern for how it is physically stored.

The three-level architecture (Cont.)

- The database administrator should be able to change the database storage structures without affecting the users' views.
  - From time to time rationalisations or other 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.
- The database administrator should be able to change the conceptual or global structure of the database without affecting the users.
  - This should be possible while still maintaining the desired individual users' views

The External or view level

- **The external level represents the user's view of the database.**
  - It consists of a number of different views of the database, potentially one for each user.
- It describes the part of the database that is relevant to a particular user.
  - For example, large organizations may have finance and stock control departments.
  - Workers in finance will not usually view stock details as they are more concerned with the accounting side of things.
  - Thus, workers in each department will require a different user interface to the information stored in the database.
- Views may provide different representations of the same data.
  - Some users might view dates in the form (day/month/year) while others prefer (year/month/day).
- Some views might include derived or calculated data.
  - For example, a person's age might be calculated from their date of birth since storing their age would require it to be updated each year.

The Conceptual or logical level

- **The Conceptual level describes what data is stored in the database and the relationships among the data.**
- It is a complete view of the data requirements of the organization that is independent of any storage considerations.
- The conceptual level represents:
  - All entities their attributes and their relationships
  - The constraints on the data.
  - Security and integrity information.
- The conceptual level supports each external view, in that any data available to a user must be contained in, or derivable from, the conceptual level.
- The description of the conceptual level must not contain any storage dependent details.

The Internal or Physical level

- **The internal level covers the physical representation of the database on the computer (and may be specified in some programming language).**
- It describes how the data is stored in the database in terms of particular data structures and file organizations.
- The internal level is concerned with:
  - Allocating storage space for data and indexes.
  - Describing the forms that records will take when stored.
  - Record placement. Assembling records into files.
  - Data compression and encryption techniques.
- **The internal level interfaces with the OS to place data on the storage devices, build the indexes, retrieve the data, etc.**
- **Below the internal level is the physical level which is managed by the OS under the direction of the DBMS. It deals with the mechanics of physically storing data on a device such as a disk.**

Differences between the levels

- External level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.
- Conceptual level: describes data stored in database, and the relationships among the data.
- Internal level describes how a record (e.g., customer) is stored.

Differences between the levels Example

- External Level

External View 1

Staff No.	Frame	L	Name	Age	Salary

External View 2

Staff No.	L.Name	Branch No.

- Conceptual Level

Staff No.	Fname	L	Name	Age	DOB

Salary.	Branch No.

- Internal Level

```
Struct Staff {
        int Staff No.;
        Int Branch No.;
        Char Fname[15];
        Char Lname[15];
        Struct  date Date of birth;
        Float Salary;
        Struct staff  * next;
}
```

Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the overall design of the database
  – e.g., 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
- **Instance** – the actual content of the database at a particular point in time
  – Analogous to the value of a variable

Three Schemas

- There are three different types of schema corresponding to the three levels in the ANSI-SPARC architecture.
- The external schemas describe the different external views of the data.
  - There may be many external schemas for a given database.
- The conceptual schema describes all the data items and relationships between them, together with integrity constraints (later).
  - There is only one conceptual schema per database.
- At the lowest level, the internal schema contains definitions of the stored records, the methods of representation, the data fields, and indexes.
  - There is only one internal schema per database.

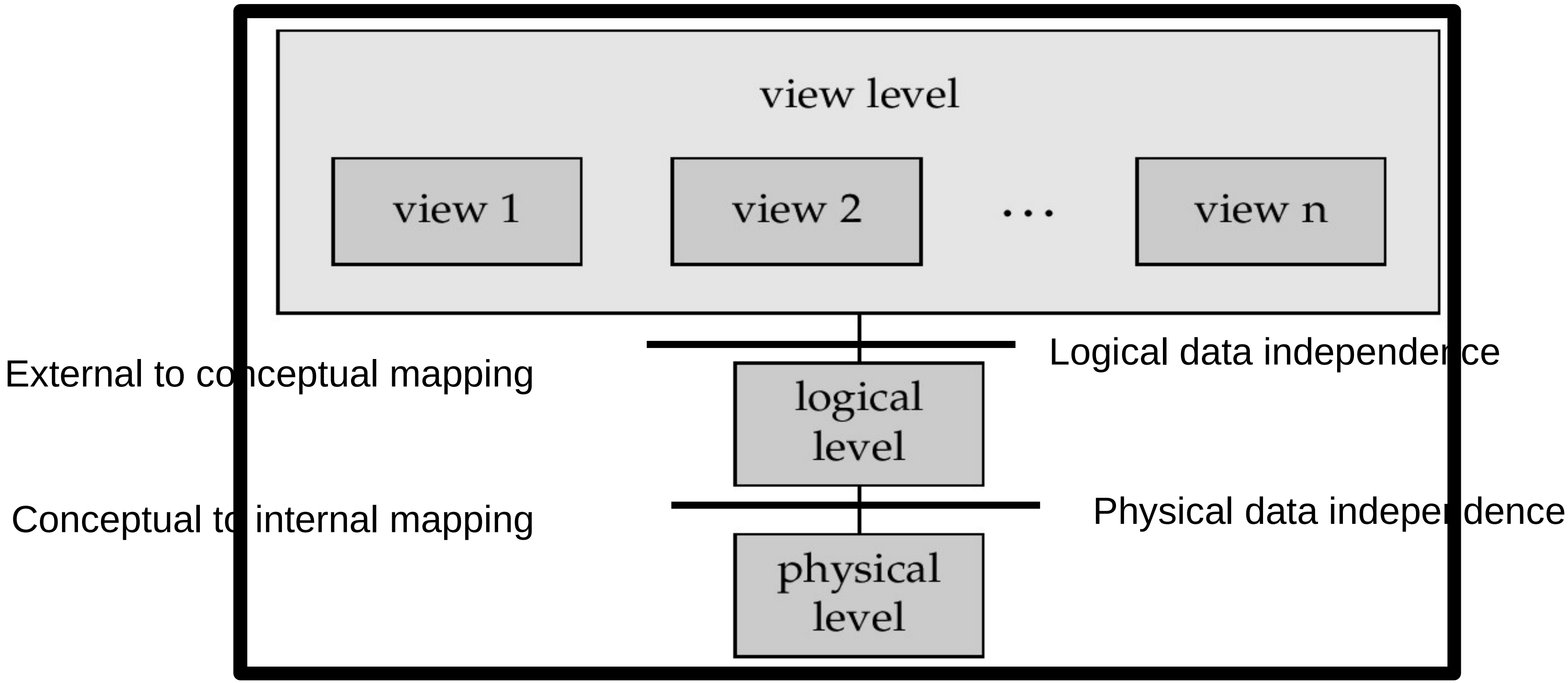
Mapping between schemas

- The DBMS is responsible for mapping between the three types of schema (i.e. how they actually correspond with each other).
- It must also check the schemas for consistency.
  - Each external schema must be derivable from the conceptual schema.
  - Each external schema is related to the conceptual schema by the external/conceptual mapping
- **External/conceptual mapping** enables the DBMS to map data in the user's view onto the relevant part of the conceptual schema.
- **A conceptual/internal mapping** relates the conceptual schema to the internal schema.
- This enables the DBMS to find the actual record or combination of records in physical storage that constitute a logical record in the conceptual schema.

Data independence

- A major objective of the ANSI-SPARC architecture is to provide data independence meaning that upper levels are isolated from changes to lower levels.
- There are two kinds of data independence:
- **Logical data independence** refers to the immunity of external schemas to changes in the conceptual schema.
  - Changes to the conceptual schema (adding/removing entities, attributes, or relationships) should be possible without having to change existing external schemas or rewrite application programs.
- **Physical data independence** refers to the immunity of the conceptual schema to changes in the internal schema.
  - Changes to the internal schema (using different storage structures or file organizations) should be possible without having to change the conceptual or external schemas.

An architecture for a database system



Outline

- Data, Information and DBMS
- Historical perspective
- Early Database systems (with drawbacks)
- Abstract views of data
- **Data models**
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS structure
- Course plan

Data Models

- A collection of conceptual tools for describing
  - data
  - data relationships
  - data semantics
  - data constraints
- Entity-Relationship model
- Relational model
- Other models:
  - object-oriented model
  - semi-structured data models
  - Older models: network model and hierarchical model

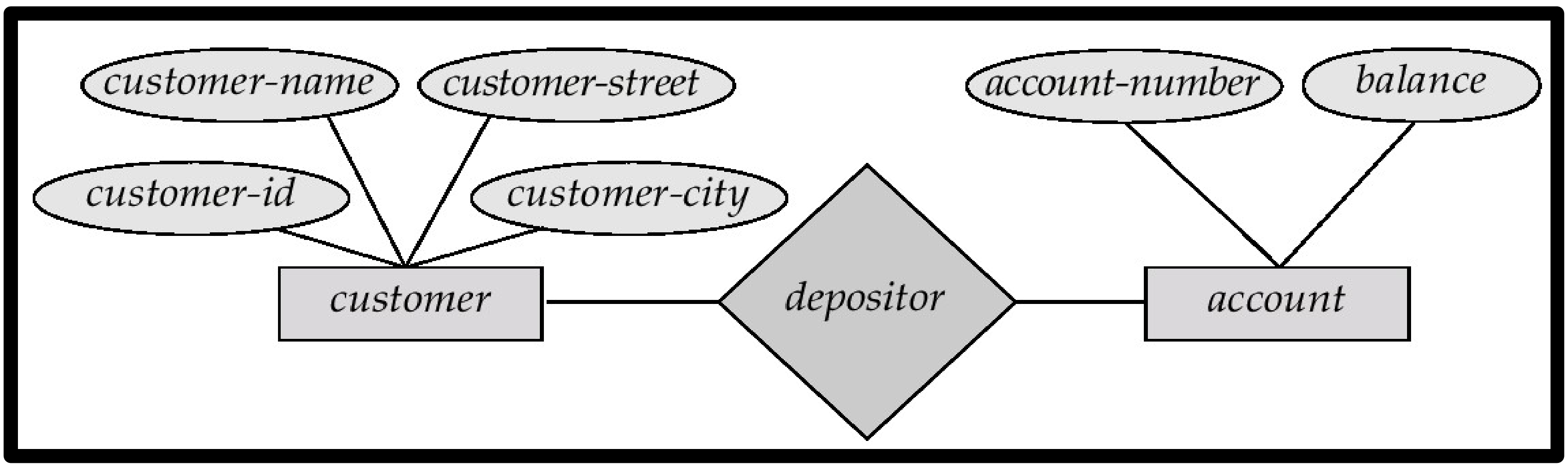
Entity Relationship Model

- It is based on a perception of a real world that consists of collection of objects called entities, and relationship among these objects
- E-R model of real world
  - Entities (objects)
    - E.g. customers, accounts, bank branch
  - Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set *depositor* associates customers with accounts
- Widely used for database design
  - Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing

Entity-Relationship Model (Cont.)

Example of schema in the entity-relationship model

- **Rectangles** represent entity sets
- **Ellipses** represent attributes
- **Diamonds** represent relationships among entity sets.
- **Lines** link attributes to entity sets and entity sets to relationships.



Relational Model

- The relational model uses a collection of tables to represent both data and the relationships among those data.
  - Each table has multiple columns and each column has unique name
- Example of tabular data in the relational model

Attributes

Customer-id	*customer-name*	*customer-street*	*customer-city*	*account-number*
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

A Sample Relational Database

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

Other data models

- Object oriented data model can be considered as an extension of E-R model with notions of encapsulation, methods, and object identity.
- Object-relational model combines features of the object-oriented data model and relational data model.
- Semi-structured data models permit the specification of data where individual data items of the same type may have different sets of attributes.
  - XML (eXtensible Markup Language) is a semi-structured data model which is widely used to represent XML data.
- Old data models: the network data model and the hierarchical data model.
  - Difficulty to model the data
  - These models were tied closely with implementation.

Database languages

- **Data-definition language (DDL):** to specify database schema
- **Data manipulation language (DML):** to express database queries and updates

Data Definition Language (DDL)

- DDL is used to specify database schema by a set of definitions.
- It is a specification notation for defining the database schema
  - E.g.

    create table *account* (
    account-number **char**(10),
    balance **integer**)
- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
  - database schema
  - Data *storage and definition* language
    - language in which the storage structure and access methods used by the database system are specified
    - Usually an extension of the data definition language
- DDL provides facilities to define consistency constraints.
  - Eg. The balance should not fall below Rs. 100/-
- DBS systems checks these constraints whenever a database systems is updated.

Data Manipulation

- Data manipulation is about
  - The retrieval of information stored in the database
  - The insertion of new information into the database.
  - The deletion of information from the database
  - The modification of information stored in the database.
  - Specification notation for defining the database schema

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural (Declarative DMLs)– user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language

SQL

- A query is a statement requesting the retrieval of information
- SQL: widely used non-procedural language
  - E.g. find the name of the customer with customer-id 192-83-7465

    select *customer.customer-name*
    from *customer*
    where *customer.customer-id* = '192-83-7465'

  - E.g. find the balances of all accounts held by the customer with customer-id 192-83-7465

    select *account.balance*
    from *depositor*, *account*
    where *depositor.customer-id* = '192-83-7465' **and**
    depositor.account-number = *account.account-number*

Database access from application programs

- Application programs are programs that are used to interact with the database which are written in a host language, such as C , C++, Cobol or Java.
  - Example: generate payroll checks, debit accounts, credit accounts, or transfer funds between accounts.
- To access the database, DML statements are executed from the host 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
  - **ODBC: open database connectivity standard defined by Microsoft**
  - **JDBC : Java database connectivity**

Outline

- Data, Information and DBMS
- Historical perspective
- Early Database systems (with drawbacks)
- Abstract views of data
- Data models
- **Database users**
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS structure
- Course plan

Database Users

- Users are differentiated by the way they expect to interact with the system
- **Application programmers** – interact with system through DML calls
  - Develop interfaces
- **Sophisticated users** – form requests in a database query language
  - Submit the queries to explore data
  - Queries are submitted to query processor
  - OLAP : online analytic processing and data mining
- Specialized users – write specialized database applications that do not fit into the traditional data processing framework
  - Expert systems, CAD/CAM
- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - E.g. people accessing database over the web, bank tellers, clerical staff

Database Administrator

- A person who coordinates all the activities of the database system;
- the database administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
  - Schema definition
    - Executes a set of DDL statements.
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints
  - Acting as liaison with users
  - Monitoring performance and responding to changes in requirements
    - Backup, disk space, monitoring (tuning) the performance

Outline

- Data, Information and DBMS
- Historical perspective
- Early Database systems (with drawbacks)
- Abstract views of data
- Data models
- Database users
- **Components of DBMS**
  - **Transaction management**
  - **Storage management**
  - **Query processor**
- DBMS structure
- Course plan

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.
- ACID: atomicity, consistency, isolation and durability
  - Atomicity: all or nothing
  - Consistency: correct execution
  - Isolation: correctness against concurrent access
  - Durability: persistence of database, failure recovery

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 file manager
  - efficient storing, retrieving and updating of data
- The storage manager components include
  - Authorization and integrity manager: which tests for the satisfaction of integrity constarints and checks the authority of usrs to access data
  - Transaction manager: which ensures that the database remains in a consistent state despite systems failures and concurrent access.
  - File manager: which manages allocation of disk space
  - Buffer manager: which is responsible for fetching data from disk storeage to main memory

Data structures implemented by storage manager

- **Data files**: which stores the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices** which provides fast access to data items

Query processor components

- **DDL interpreter,** which interprets DDL statements and records the definitions in the data dictionary
- **DML compiler**: Translates DML statements in a query langauge consisting of low-level instructions (evaluation plan).
- **Query evaluation engine**: which executes low-level instructions generated by the DML compiler.

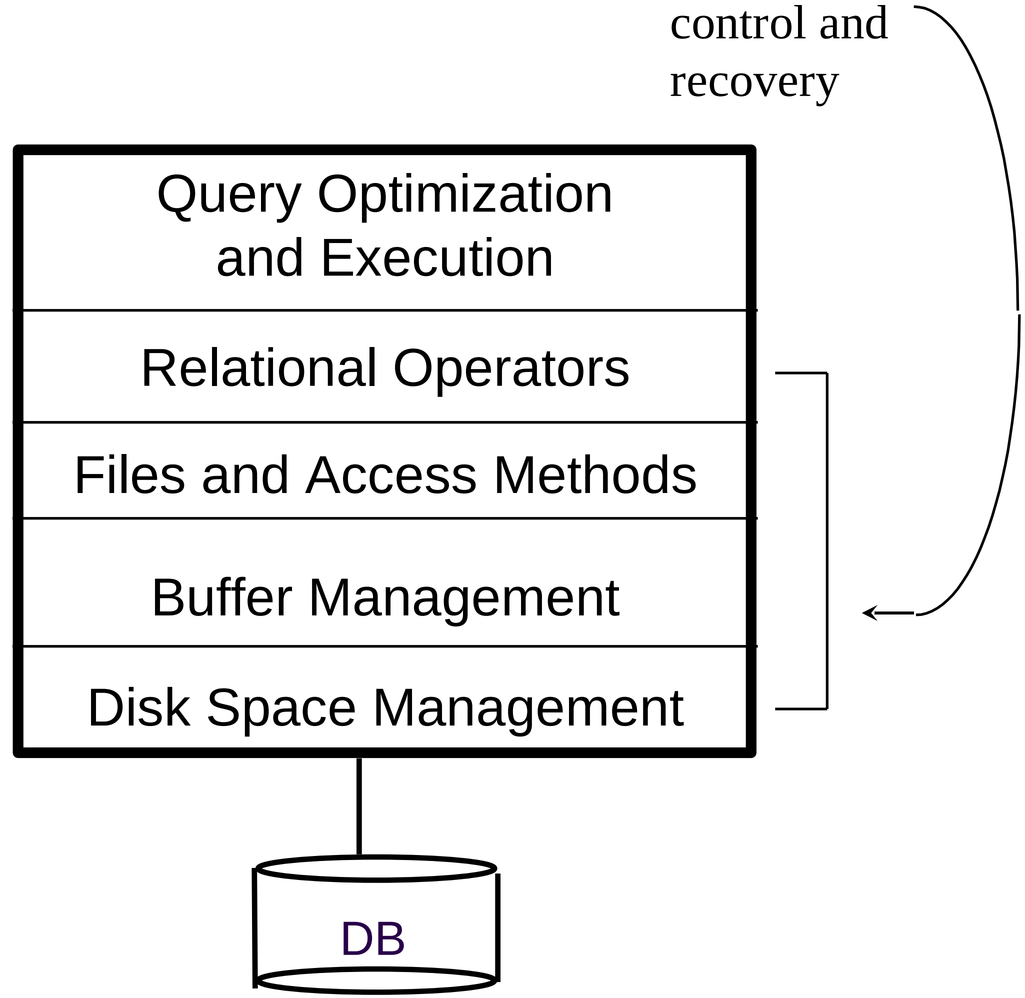
Outline

- Data, Information and DBMS
- Historical perspective
- Early Database systems (with drawbacks)
- Abstract views of data
- Data models
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- **DBMS Structure**
- Course plan

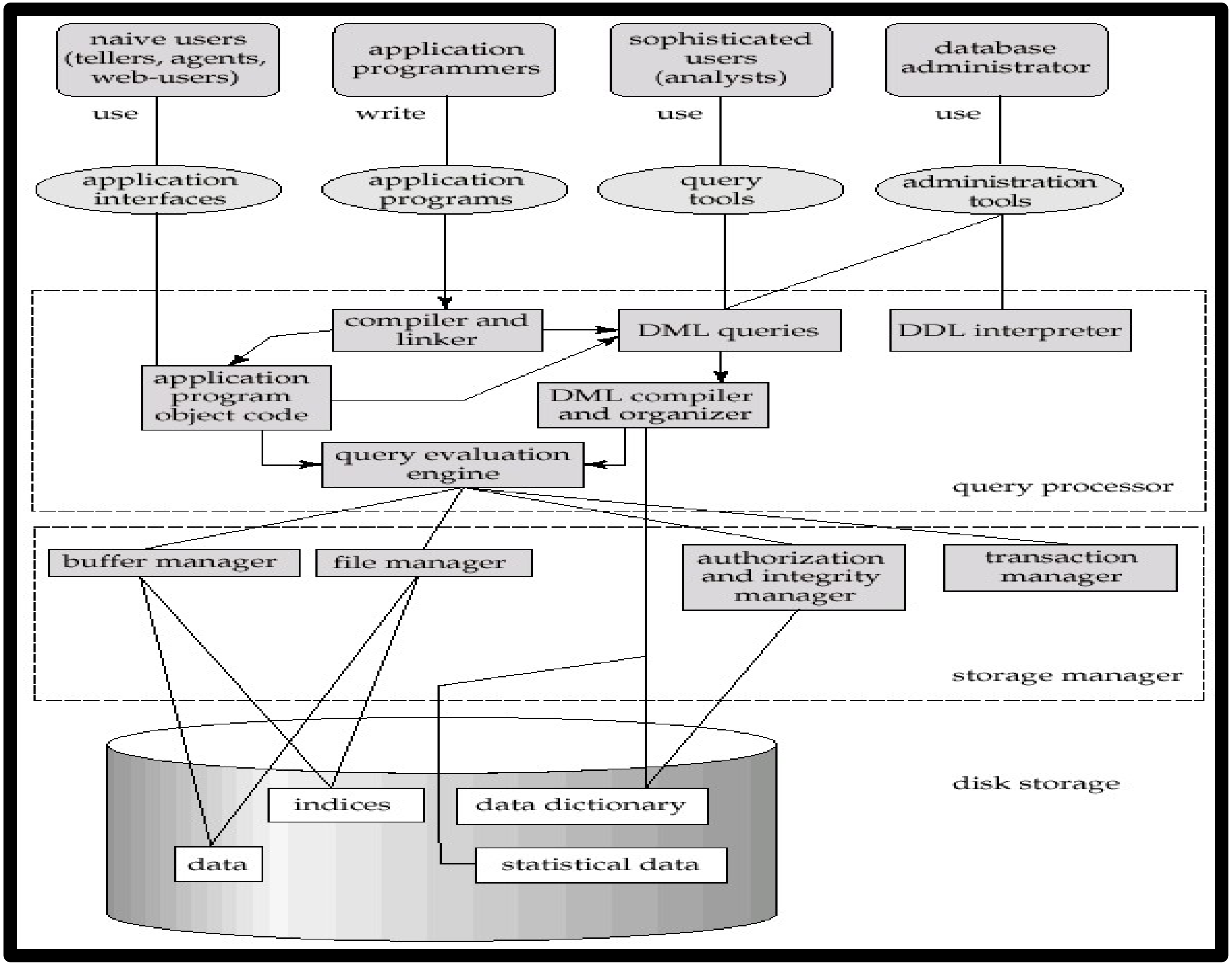
Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.

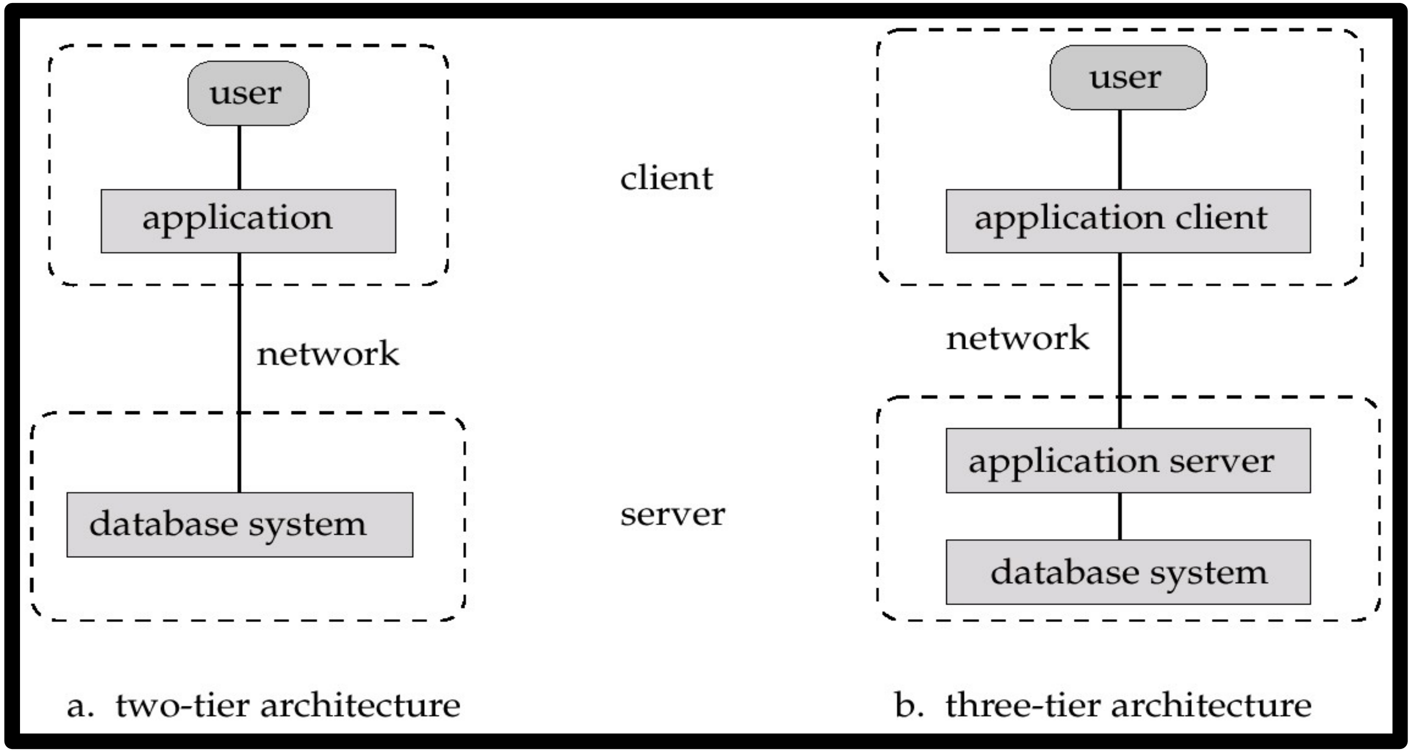
These layers must consider concurrency control and recovery



Overall System Structure



Application Architectures



Two-tier architecture: E.g. client programs using ODBC/JDBC to communicate with a database

Three-tier architecture: E.g. web-based applications, and applications built using “middleware”

Summary

- DBMS used to maintain, query large datasets.
- Benefits 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.
- DBAs hold responsible jobs and are well-paid!
- DBMS R&D is one of the broadest, most exciting areas in CS.

Outline

- Data, Information and DBMS
- Historical perspective
- Early Database systems (with drawbacks)
- Abstract views of data
- Data models
- Database users
- Components of DBMS
  - Transaction management
  - Storage management
  - Query processor
- DBMS Structure
- **Course plan**

About DBMS Course

- TITLE: CS3400: Database Management System [3-0-3-4]
- CREDITS : 4
- TYPE-WHEN : Elective (B.Tech 2nd year /MTech 1st year)
- PRE-REQUISITE : Operating Systems, data structures, high-level programming language such as C/C++ or Java.
- Timings:
  - **825 AM to 955 AM** (MON and WED)
  - Door will be closed at 830AM!

Objectives

- The objective of this course is to introduce the fundamentals concepts of database management systems. The concepts include database design, database languages, and database-system implementation.

Course topics

- Introduction [3 hrs]
- Entity-Relationship model [3 hrs]
- Normal forms [3 hrs]
- Relational model [3 hours)
- Relational algebra and calculus [6hrs]
- SQL [6 hrs]
- Database application development[1.5 hr]
- Internet applications[1.5 hr]
- Overview of Storage and indexing [3 hrs]
- Overview of query evaluation [3hrs]
- Overview of transaction Management [3hrs]
- Concurrency control [3hrs]
- Crash recovery[3hrs]
- Security and authorization [1.5hrs]
- Additional topics: Parallel and distributed databases, object-database systems, deductive databases, data warehousing, data mining, Information retrieval and XML, Spatial data management [3 hours]

References

- Books
  - Main books:
    - Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, Third edition, Mc Graw Hill, 2003.
  - Other books
    - Abraham Silberschatz, Henry F.Korth, S.Sudarshan, Database system concepts, fifth edition, Mc Graw Hill, 2006.
    - Elmasri & Navathe, Fundamentals of Database Systems, Addison Wesley, 2000.
- Research Papers
  - About six basic research papers will be covered.
- Tutorials
  - Mandatory Tutorials: ER Data model, relational algebra, tuple relational calculus, SQL, normalization, file organization, indices, join algorithms, query optimization, serializability theory.

LAB WORK

- Mandatory Project:
  - Three-phase project in groups of 2 students. Phase I: database and application requirements analysis and ER Model development, Phase II: translation to relational data model, normalization, and creation and population of database, Phase III: application development accessing the database.
- The lab is very intensive. Please do not ask for the extension of deadline. Each experiment will be evaluated.

Reading/Practicing Assignments

- Problems will be given
- You have to solve on your own

GRADING

- MidSem1: 15 %;
- MidSemII: 15 %;
- EndSem: 30%;
- Project/Lab: 25 %
- Quizes: 15%

OUTCOME

- After completing the course, the students will understand
  - how to design a database, database-based application.
  - use a DBMS
  - the fundamental concepts of several relational database management systems such as ORACLE, DB2, My SQL, Microsoft SQL server and so on
  - the solutions/options to interesting problems which have been encountered by the designers of preceding DBMSs.
  - the critical role of database system in designing several information system-based software systems or applications.