What is a DBMS?

- A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data

- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*

Database System Applications

- Banking: For customer and accounts information
- Airlines: For reservations and schedule information
- Universities: For student information, course registrations
- Credit card transactions
- Sales: For customer, product, and purchase information
- Manufacturing: For production, inventory, orders

▪Human resources: For employees information, salaries, tax deductions

Data base system vs File system

▪Consider part of a savings-bank enterprise that keeps information about all customers and savings accounts

▪One way to keep the information on a computer is to store it in operating system files

▪The system has a number of application programs that manipulate the files, including

- *A program to debit or credit an account*
- *A program to add a new account*
- *A program to find the balance of an account*
- *A program to generate monthly statements*

- New application programs are added to the system as the need arises

- This typical file-processing system is supported by a conventional operating system

- The system stores permanent records in *various files*, and it needs different *application programs* to extract records from, and add records to, the appropriate files

- Before database management systems came along, organizations usually stored information in such systems

- file-processing system has a number of major disadvantages:

- Data redundancy and inconsistency
  -Multiple file formats, duplication of information in different files
  -Example: Savings account and Current account

- Difficulty in accessing data
  -Need to write a new program to carry out each new task
  -Example: when new query is asked

- Data isolation
  -Multiple files and formats

- Integrity problems
  -Values in the database must satisfy constraints
  -Hard to add new constraints or change existing ones

- Atomicity of updates
  - -Failures may leave database in an inconsistent state with partial updates carried out
  - -Example: Transfer of funds from one account to another should either complete or not happen at all (atomic)

- Concurrent access by multiple users
  - -Concurrent access needed for performance
  - -Uncontrolled concurrent accesses can lead to inconsistencies
    - -Example: Two people reading a balance and updating account at the same time

- Security problems
  - -Hard to provide user access to some, but not all, data

- Database systems offer solutions to all the above problems

View of Data

- A major purpose of a database system is to provide users with an abstract view of the data. That is the system hides certain details of how the data are stored and maintained

Data Abstraction

- For the system to be usable, it must retrieve data efficiently

- The need for efficiency has led designers to use complex data structures to represent data in the database

- Many database system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users interactions with the system

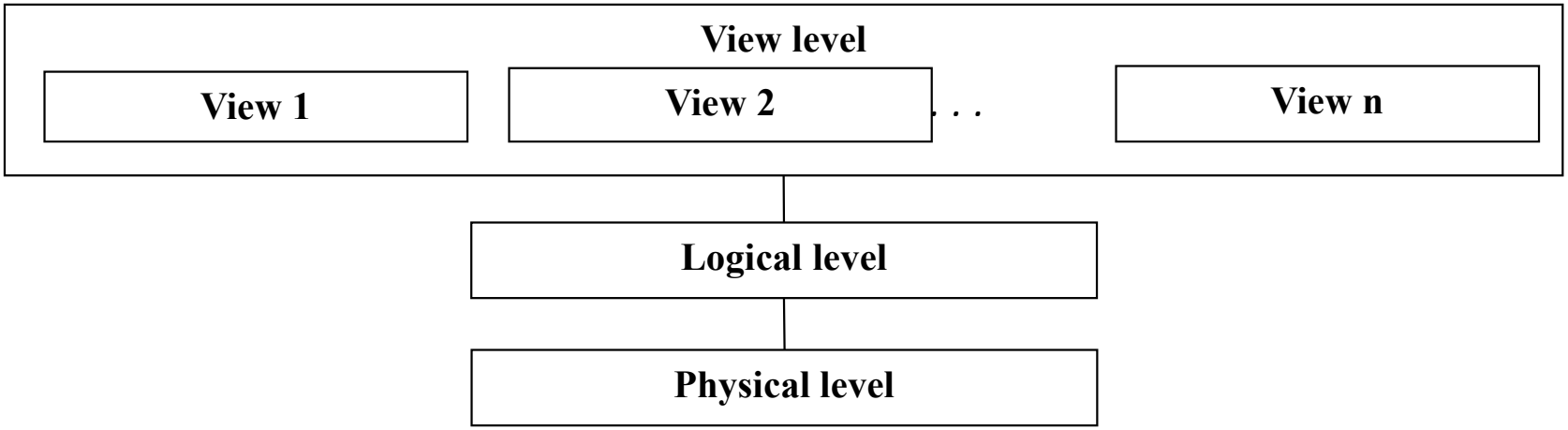


Figure 1.1 The three levels of data abstraction

Physical level: The lowest level of abstraction describes **how** the data are actually stored

- The physical level describes complex low-level data structures in detail

Logical level: The next-higher level of abstraction describes **what** data are stored in the database, and what relationships exist among those data

- The logical level describes the entire database in terms of a small number of relatively simple structures

- Implementation of the simple structures at the logical level may involve complex physical-level structures

View level: The highest level of abstraction describes only part of the entire database

- The view level of abstraction exists to simplify users interaction with the system

Example:

```
struct customer
{
   customer_id: string;
   customer_name: string;
   customer_street: string;
   customer_city: string;
};
```

- A banking enterprise may have several such record types, including
  -Account, with fields account_number and account_balance
  -Employee, with fields employee_name and employee_salary

- At the physical level, a customer, account, or employee record can be described as a block of consecutive storage locations

- At the logical level, each such record is described by a structure definition, as in the previous code segment, and the interrelationship of these record types is defined as well

- At the view level, computer users see a set of application programs that hide details of the data types

- At the view level, several views of the database are defined, and database users see these views

- The views also provide a security mechanism to prevent users from accessing certain parts of the database

  For example, tellers in a bank see only that part of the database that has information on customer accounts; they cannot access information about salaries of employees

Instances and Schemas

- Databases change over time as information is inserted and deleted

- The collection of information stored in the database at a particular moment is called an instance of the database

- The overall design of the database is called the database schema

- The concept of database schemas and instances can be understood by analogy to a program written in a programming language

- A database schema corresponds to the variable declarations in a program, each variable has a particular value at a given instant

- The values of the variables in a program at a point in time correspond to an instance of a database schema

- Database systems have several schemas, partitioned according to the levels of abstraction

- The physical schema describes the database design at the physical level, while the logical schema describes the database design at the logical level

- A database may also have several schemas at the view level, sometimes called subschemas, that describe different views of the database

- Programmers develop applications by using the logical schema

- Physical schema is hidden beneath the logical schema, and can usually be changed easily without affecting application programs

- Application programs are said to exhibit *physical data independence if they do not depend on the physical schema*, and thus need not be rewritten if the physical schema changes

Data Models

- Underlying the structure of a database is the *data model*
- Data model is a collection of tools for describing data, data relationships, data semantics, and consistency constraints

Types of Data Models

- Relational Model
- The Entity-Relationship Model
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model
- Network model
- Hierarchical model

Relational Model

- The relational model uses a collection of tables to represent both data and the relationships among those data

Example

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

Figure 1.2 Customer table

- The relational model is an example of a record-based model

- Record-based models are so named because the database is structured in fixed-format records of several types

- Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes

- The columns of the table correspond to the attributes of the record type

- The relational data model is the most widely used data model

Entity-Relationship Model

- The entity-relationship (E-R) data model is a collection of *entities* and *relationships* among these entities
- An *entity* is a "thing" or "object" in the real world that is distinguishable from other objects
- For example, each person is an entity, and bank accounts can be considered as entities
- Entities are described in a database by a set of attributes
- For example, the attributes *account-number* and *balance* may describe one particular account in a bank, and they form attributes of the *account* entity set
- Attributes *customer-name, customer-street* and *customer-city* may describe a *customer* entity

- A *relationship* is an association among several entities
- For example, a *depositor* relationship associates a *customer* with each *account* that he has
- The set of all entities of the same type and the set of all relationships of the same type are termed an *entity set* and *relationship set*, respectively
- The overall logical structure (schema) of a database can be expressed graphically by an *E-R diagram,* which is built up from the following components:
  - *Rectangles*, which represent entity sets
  - *Ellipses*, which represent attributes
  - *Diamonds*, which represent relationships among entity sets
  - *Lines*, which link attributes to entity sets and entity sets to relationships

Example

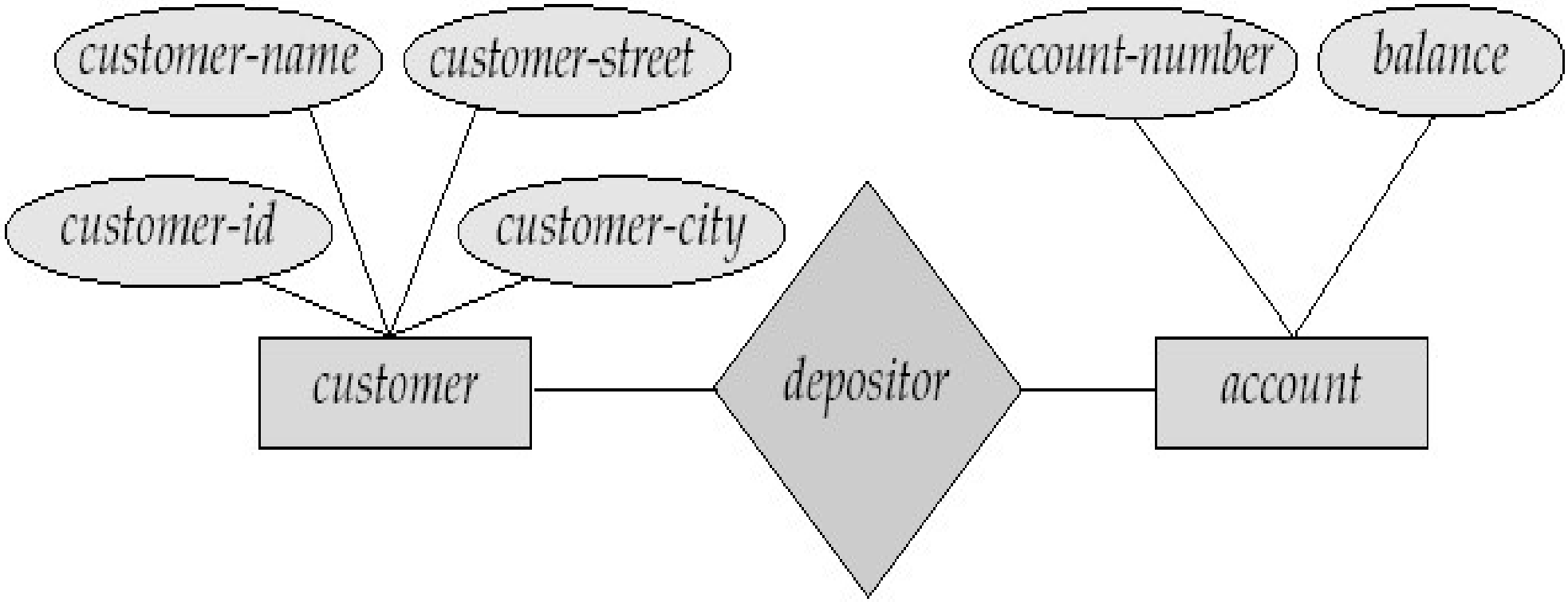


Figure 1.3

Object-based data models (Object-oriented and Object-relational)

- In object-oriented model the data is stored in the form of objects

- The object-relational data model combines features of the object-oriented data model and relational data model

Semistructured data model

- Semistructured data models permit the specification of data where individual data items of the same type may have different sets of attributes
- This is in contrast with the data models mentioned earlier, where every data item of a particular type must have the same set of attributes
- The extensible markup language (XML) is widely used to represent semistructured data

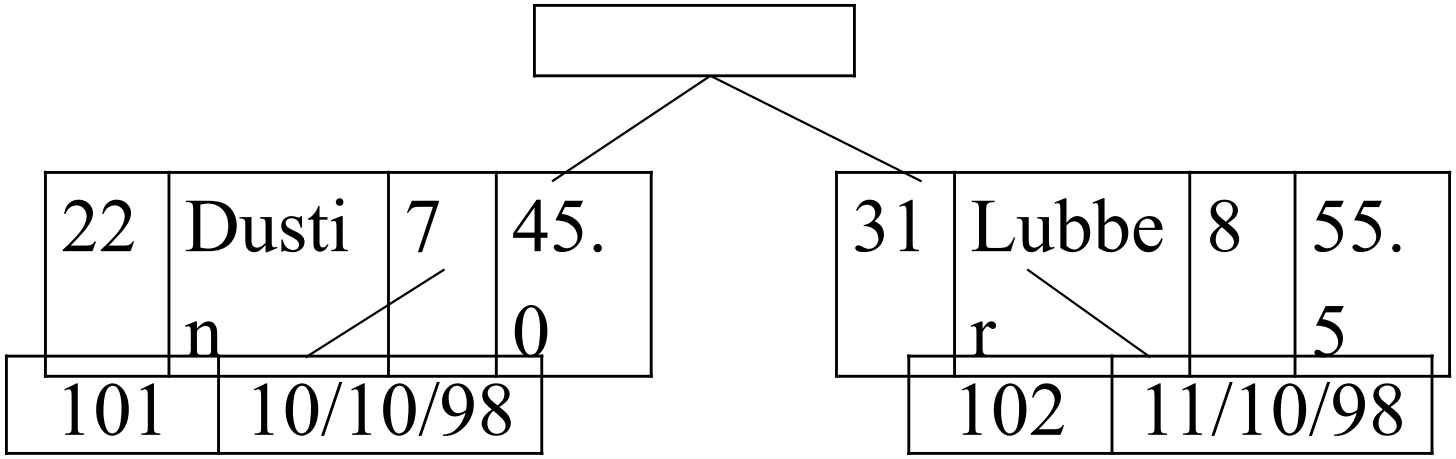
Network model

- Data in this model are represented by collection of records and relationships among data are connected by links

22	Dustin	7	45.0	101	10/10/98
31	Lubber	8	55.5	102	11/10/98
74	Horatio	9	35.0	103	9/8/98

Hierarchical model

- Hierarchical model is same as the network model
- In this model records are represented in the form of tree



Database Languages

- A database system provides a *data definition language* to specify the *database schema*, a *data manipulation language* to express *database queries and updates* and a *data control language* to *control a database*

- In practice these are not separate languages; instead they simply form parts of a single database language, such as the widely used SQL language

Data-Definition Language

- This supports the creation, deletion, and modification of definitions for tables and views

- The following statement in the SQL language defines the *account* table:

create table *account*
(*account-number* **char(10),**
Balance **number(5)**)

- Execution of the above DDL statement creates the *account* table

- In addition, it updates a special set of tables called the **data dictionary** or **data directory**

- A data dictionary contains **metadata—that is, data about data**

- The schema of a table is an example of metadata
- A database system consults the data dictionary before reading or modifying actual data

Data-Manipulation Language

- Data manipulation is
  - 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
- A **data-manipulation language (DML)** is a language that enables users to access or manipulate data
- There are basically two types:

Procedural DMLs require a user to specify *what* data are needed and *how* to get those data

Declarative DMLs (also referred to as **nonprocedural DMLs**) require a user to specify *what* data are needed *without* specifying how to get those data

- This query in the SQL language finds the name of the customer whose customer-id is 192-83-7465:

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

Data-Control Language

- This subset of SQL controls a database, including administrative privileges and saving data

Database Access from Application Programs

- **Application programs** are programs that are used to interact with the database
- Application programs are usually written in a *host* language, such as Cobol, C, C++, or Java
- Examples in a banking system are programs that generate payroll checks, debit accounts, credit accounts, or transfer funds between accounts
- To access the database, DML statements need to be executed from the host language
- There are two ways to do this:
  - By providing an application program interface (set of procedures) that can be used to send DML and DDL statements to the database, and retrieve the results

The Open Database Connectivity (ODBC) standard defined by Microsoft for use with the C language is a commonly

used application program interface standard. The Java Database Connectivity (JDBC) standard provides corresponding features to the Java language

❑By extending the host language syntax to embed DML calls within the host language program. Usually, a special character prefaces DML calls, and a preprocessor, called the DML **precompiler,** converts the DML statements to normal procedure calls in the host language

Database Users and Administrators

- A primary goal of a database system is to retrieve information from and store new information in the database
- People who work with a database can be categorized as database users or database administrators

Database Users and User Interfaces

- There are four different types of database-system users, differentiated by the way they expect to interact with the system
- Different types of user interfaces have been designed for the different types of users

1.Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously

(Naive users are users who do not have knowledge about the system)

-For example, a bank teller who needs to transfer $50 from account *A* to account *B* invokes a program called *transfer.* This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be

▪As another example, consider a user who wishes to find his account balance over the Internet. Such a user may access a form, where he enters his account number

2.Application programmers are computer professionals who write application programs

-Application programmers can choose from many tools to develop user interfaces

-Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program

-*Fourth-generation languages,* include features to facilitate the generation of forms and the display of data on the screen

-Most major commercial database systems include a fourth-generation language

3.Sophisticated users interact with the system without writing programs

-They form their requests in a database query language. They submit each such query to a **query processor**, whose function is to break down DML statements into instructions that the storage manager understands

-Analysts who submit queries to explore data in the database fall in this category

-Online analytical processing (OLAP) tools simplify analysts tasks by letting them view summaries of data in different ways

For instance, an analyst can see total sales by region (for example, North, South, East, and West), or by product, or by a combination of region and product (that is, total sales of each product in each region)

-Another class of tools for analysts is **data mining tools**, which help them find certain kinds of patterns in data

4.Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework

-Among these applications are computer-aided design systems, knowledge-base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems

Database Administrator

- One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a **database administrator (DBA)**
- The functions of a DBA include:

-Schema definition. The DBA creates the original database schema by executing a set of data definition statements in the DDL

-Storage structure and access-method definition.

-Schema and physical-organization modification. The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance

-Granting of authorization for data access. By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system

-Routine maintenance. Examples of the database administrator's routine maintenance activities are:

-Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding

-Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required

-Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

Transaction Management

- Often, several operations on the database form a single logical unit of work
- An example is a funds transfer in which one account (say *A)* is debited and another account (say *B)* is credited

- Clearly, it is essential that either both the credit and debit occur, or that neither occur
- That is, the funds transfer must happen in its entirety or not at all. This all-or-none requirement is called **atomicity**
- In addition, it is essential that the execution of the funds transfer preserve the consistency of the database. That is, the value of the sum $A + B$ must be preserved
- This correctness requirement is called **consistency**
- Finally, after the successful execution of a funds transfer, the new values of accounts A and B must persist, despite the possibility of system failure. This persistence requirement is called **durability**
- A **transaction** is a collection of operations that performs a single logical function in a database application

- We require that transactions do not violate any database-consistency constraints
- That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates
- It is the programmer's responsibility to define properly the various transactions, so that each preserves the consistency of the database
- For example, the transaction to transfer funds from account *A* to account *B* could be defined to be composed of two separate programs: one that debits account *A,* and another that credits account *B*
- The execution of these two programs one after the other will indeed preserve consistency

- Ensuring the atomicity and durability properties is the responsibility of the database system itself—specifically, of the **transaction-management component**
- If we are to ensure the atomicity property, a failed transaction must have no effect on the state of the database. Thus, the database must be restored to the state in which it was before the transaction in question started executing
- The database system must therefore perform **failure recovery**, that is, detect system failures and restore the database to the state that existed prior to the occurrence of the failure
- Finally, when several transactions update the database concurrently, the consistency of data may no longer be preserved, even though each individual transaction is correct

- It is the responsibility of the **concurrency-control manager** to control the interaction among the concurrent transactions, to ensure the consistency of the database

Note: each transaction possess **ACID** properties

Database System Structure

- The functional components of a database system can be broadly divided into the *storage manager* and the *query processor* components
- The *storage manager* is important because databases typically require a large amount of storage space
- Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data
- A gigabyte is 1000 megabytes and a terabyte is 1000 gigabytes
- Since the main memory of computers cannot store this much information, so this information is stored on disks. Data are moved between disk storage and main memory as needed

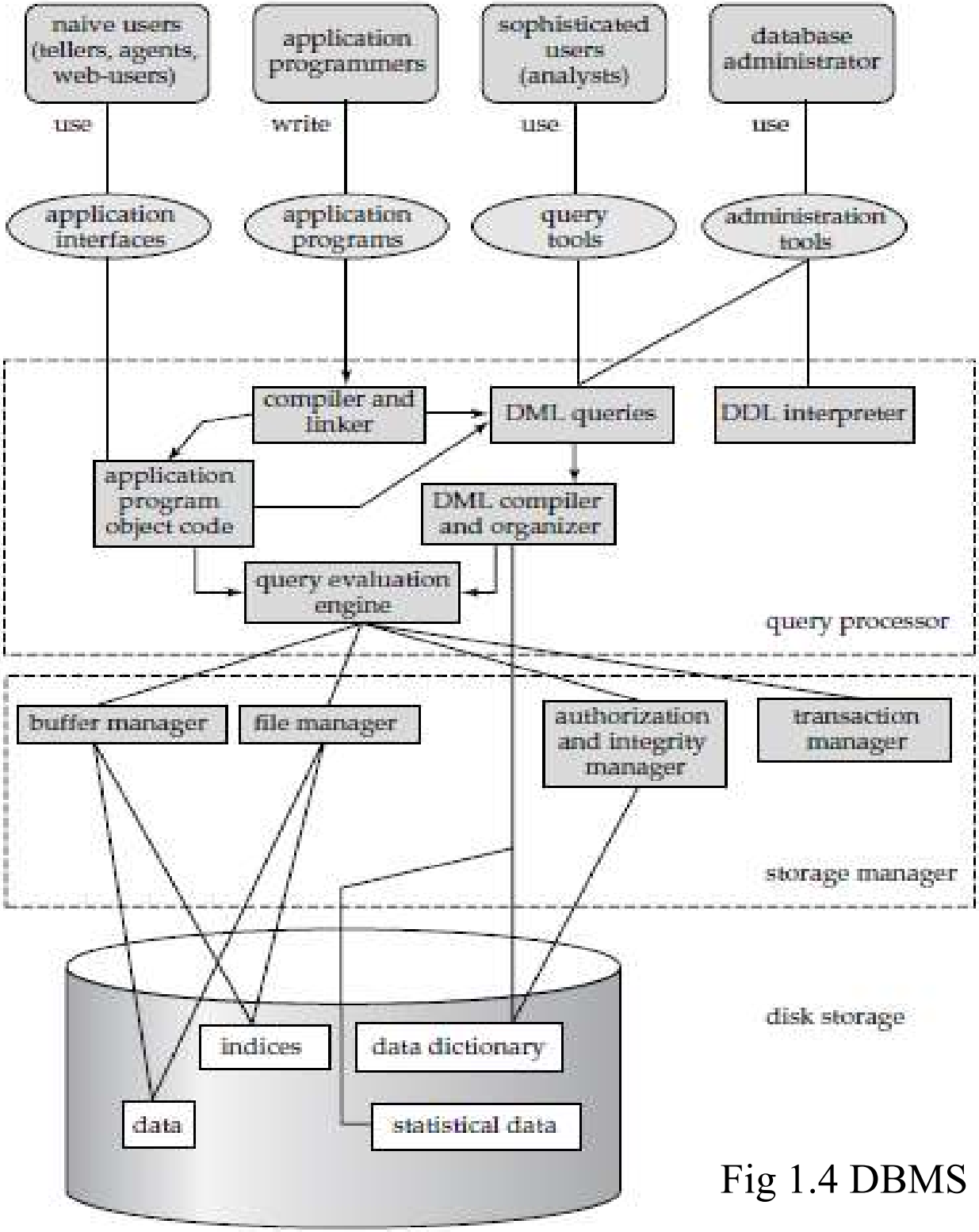


Fig 1.4 DBMS structure

- The *query processor* is important because it helps the database system simplify and facilitate access to data
- The job of the database system is to translate queries at the logical level, into an efficient sequence of operations at the physical level

Storage Manager

- A *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 for the interaction with the file manager
- The raw data are stored on the disk

- The storage manager is responsible for storing, retrieving, and updating data in the database
- The storage manager components include:

-Authorization and integrity manager, which tests for the satisfaction of integrity constraints and checks the authority of users to access data

-Transaction manager, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting

-File manager, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk

-Buffer manager, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory

- The storage manager implements several data structures as part of the physical system implementation:

-Data files, which store the database itself

-Data dictionary, which stores metadata about the structure of the database, in particular the schema of the database

-Indices, which provide fast access to data items that hold particular values

The Query Processor

- The query processor components include

-DDL interpreter, which interprets DDL statements and records the definitions in the data dictionary

-**DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands

--A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives

-Query evaluation engine, which executes low-level instructions generated by the DML compiler