Chapter 1: Introduction

Database System Concepts, 6th Ed.

Data, Database, and DBMS

- Data [Elmasri and Navathe. Fundamentals of Database Systems]
  - Known facts that can be recorded and that have implicit meaning

  e.g.) names, phone numbers, addresses, …

- Database
  - Collection of interrelated data

- Database management system (DBMS)
  - Database + set of programs to access those data
  - An environment that is both *convenient* and *efficient* to use

Database System Applications

- Databases touch all aspects of our lives

- 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

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Data are scattered in multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of Using File Systems to Store Data (Cont.)

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

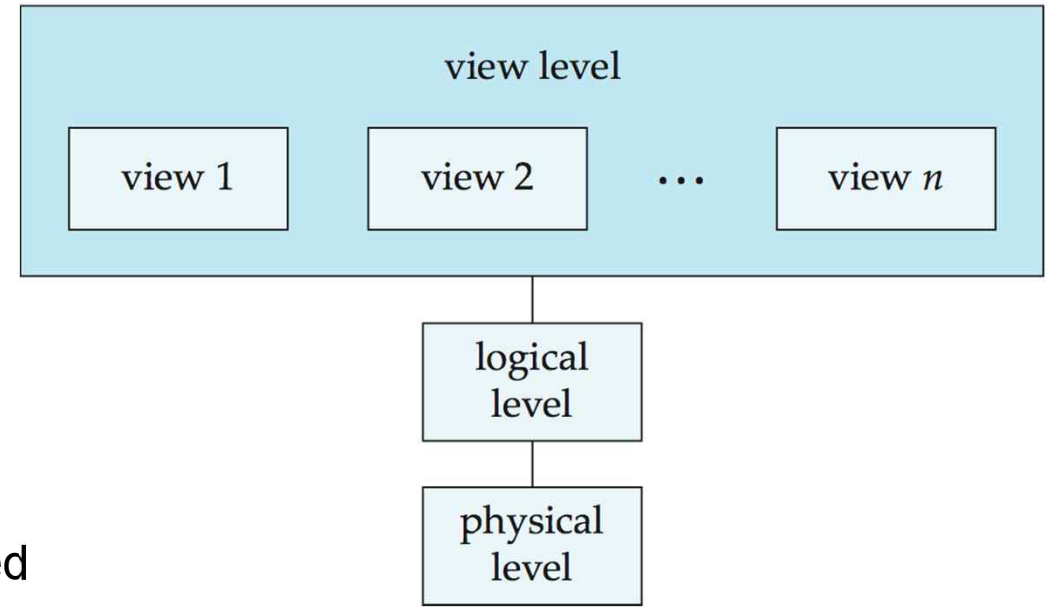
Database systems offer solutions to all the above problems

Levels of Abstraction

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

- **Logical level:** describes data stored in database, and the relationships among the data

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



- **Physical level:** describes how a record is stored

Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - Analogous to type information of a variable in a program
  - Example: The database consists of information about a set of customers and accounts and the relationship between them
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Physical Data Independence

- Schemas according to the level of abstraction
  - **Logical schema**: database design at the logical level
  - **Physical schema**: database design at the physical level

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

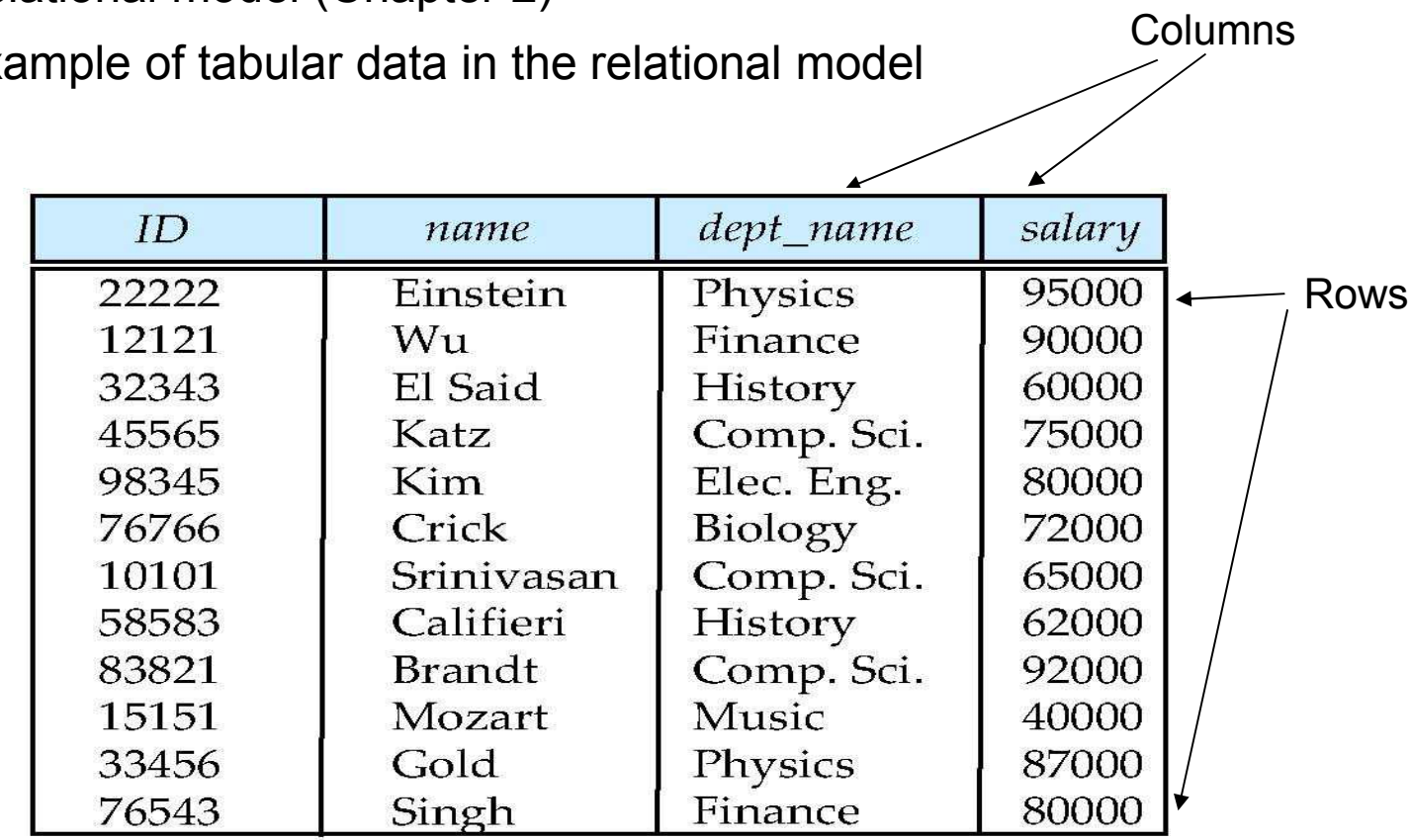
Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints

- **Relational model**
- **Entity-Relationship data model** (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

- Relational model (Chapter 2)
- Example of tabular data in the relational model

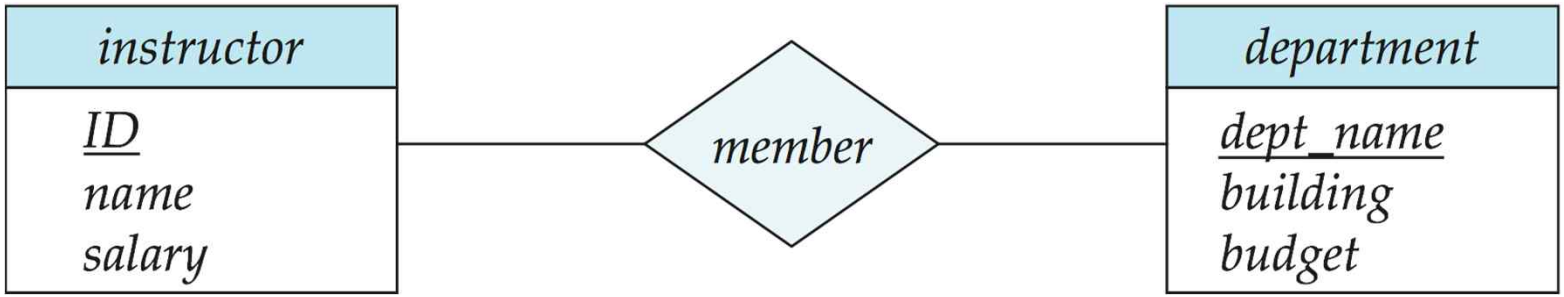


ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music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

(a) The *instructor* table

The Entity-Relationship Model

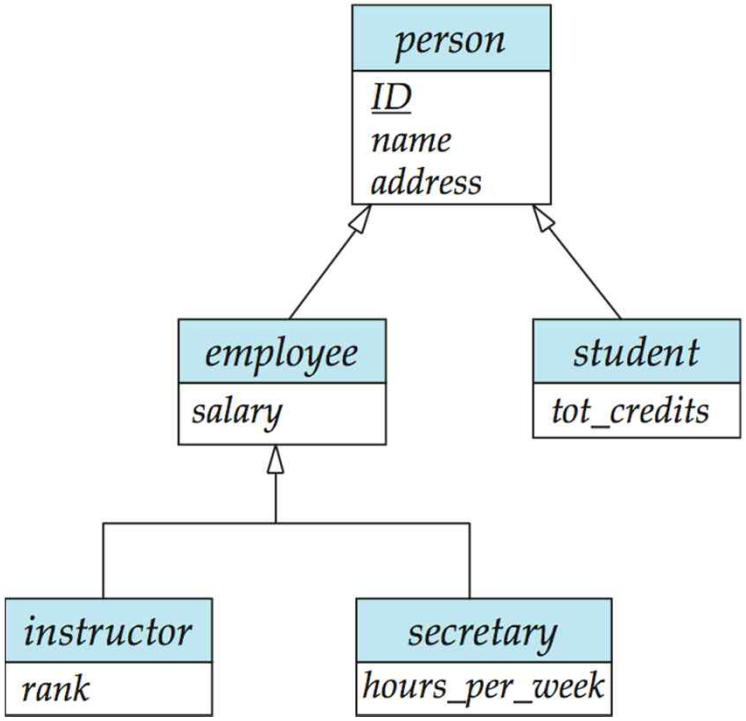
- Models an enterprise as a collection of *entities* and *relationships*
  - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram:*



Object-Based Data Model

- Supports complex types and inheritance

title	*author_array*	*publisher*	*keyword_set*
		(*name, branch*)	
Compilers	[Smith, Jones]	(McGraw-Hill, NewYork)	{parsing, analysis}
Networks	[Jones, Frick]	(Oxford, London)	{Internet, Web}



XML Data Model

- Supports extensible tags and nested structures

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
    </department>
    <course>
        <course_id> CS-101 </course_id>
        <title> Intro. to Computer Science </title>
        <dept_name> Comp. Sci </dept_name>
        <credits> 4 </credits>
    </course>
</university>
```

```
<university>
    <department>
        <dept_name> Comp. Sci. </dept_name>
        <building> Taylor </building>
        <budget> 100000 </budget>
        <course>
            <course_id> CS-101 </course_id>
            <title> Intro. to Computer Science </title>
            <dept_name> Comp. Sci </dept_name>
            <credits> 4 </credits>
        </course>
    </department>
</university>
```

Data Definition Language (DDL)

- Language for defining the database schema

- 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)
    - Referential integrity (**references** constraint in SQL)
      - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation
  - Authorization

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - Retrieval, insertion, deletion, modification
- Query language: part of DML that requests data retrieval
  - Commonly used as a synonym for DML

- Two classes of languages
  - **Procedural** – user specifies what data is required and how to get those data
  - **Declarative (nonprocedural)** – user specifies only what data is required, without specifying how to get those data
- SQL is the most widely used query language

SQL

- **SQL**: widely used non-procedural language
  - Example: Find the name of the instructor with ID 22222

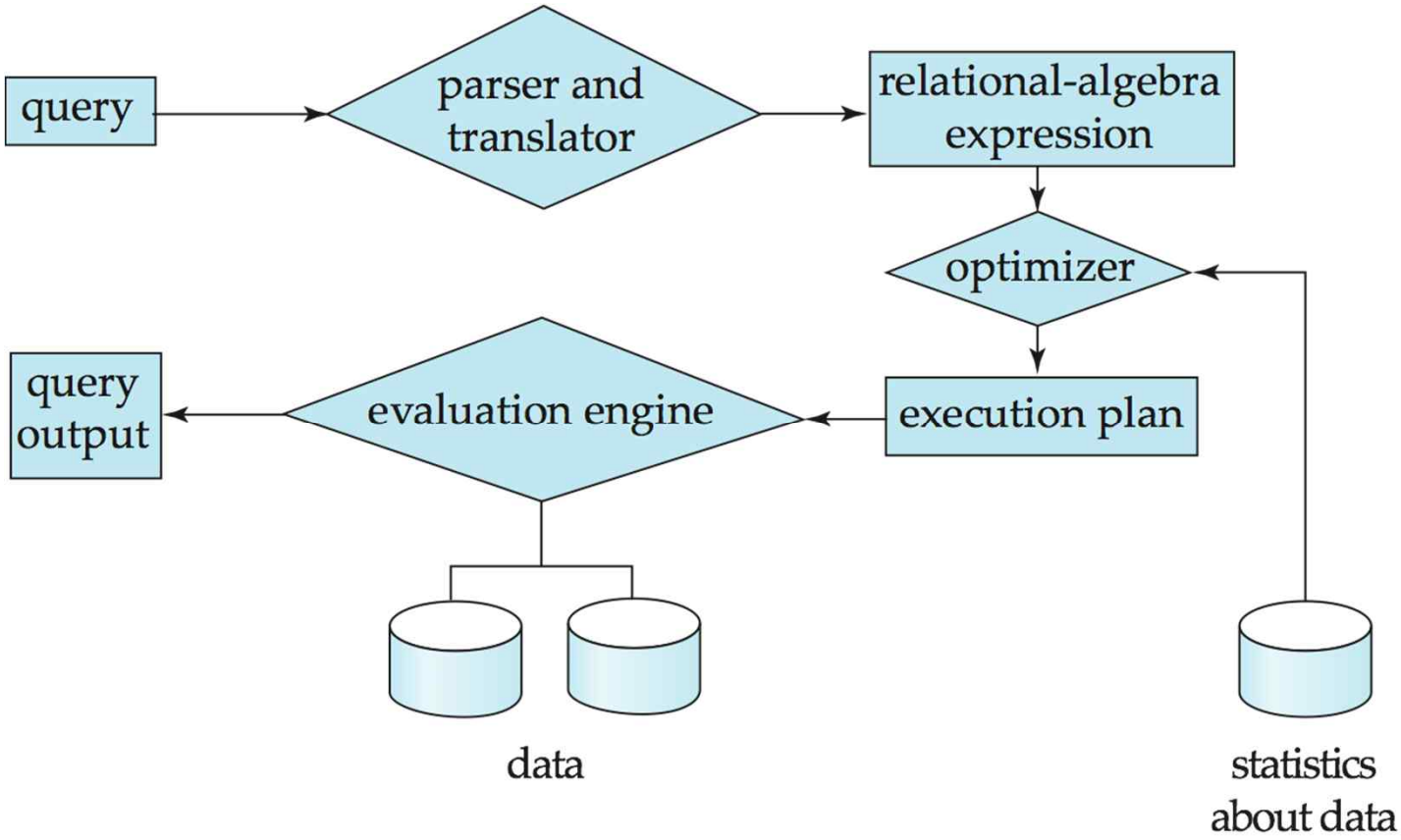
    select *name*
    from *instructor*
    where *instructor.ID* = '22222'
  - Example: Find the ID and building of instructors in the Physics dept.

    select *instructor.ID*, *department.building*
    from *instructor*, *department*
    where *instructor.dept name* = "physics"

- 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
- Chapters 3, 4 and 5

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Storage Management

- DBMS must effectively and efficiently manage storage (disk) space

- **Storage manager**
  - 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
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Transaction Management

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

- **Transaction** – 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

Database Users

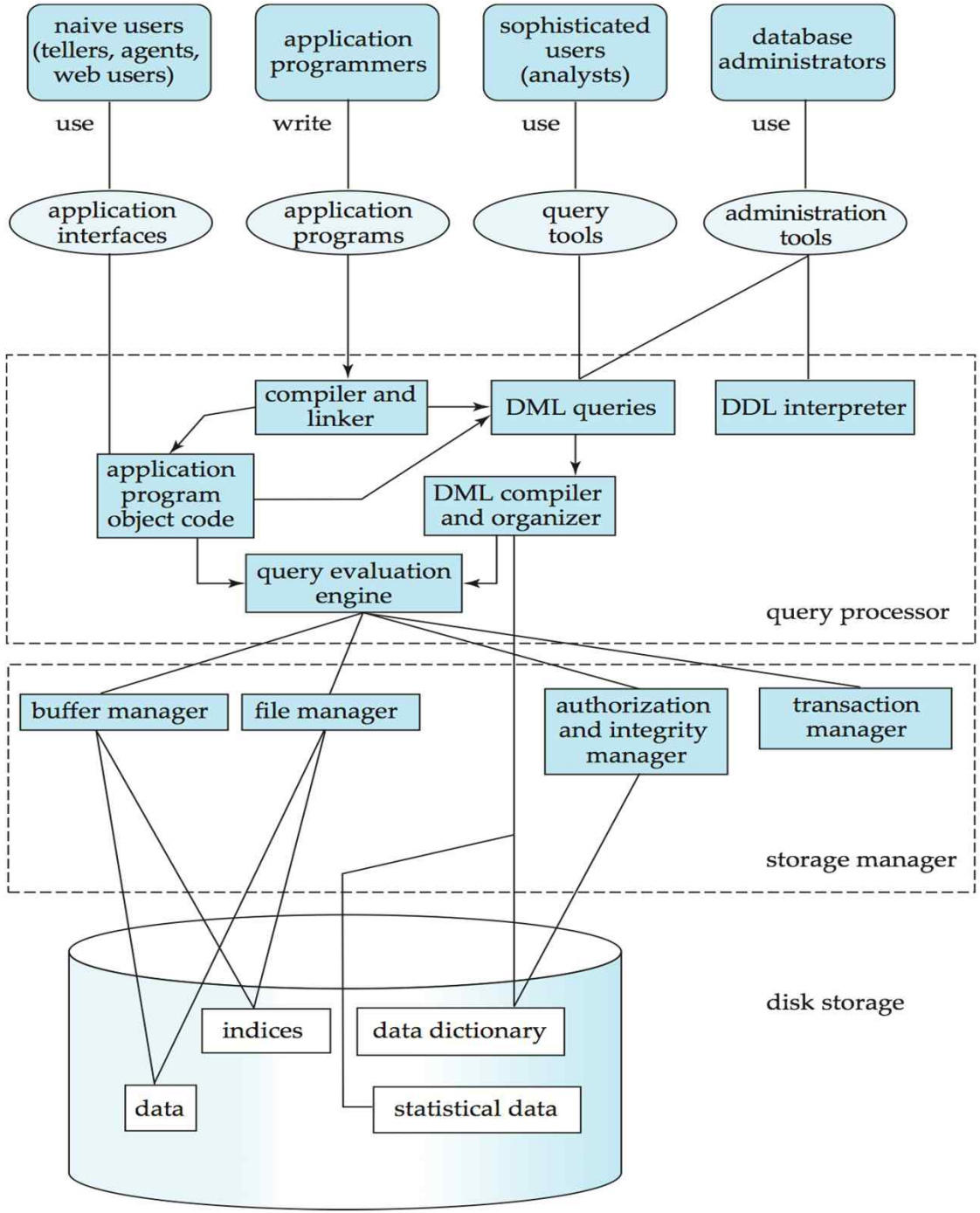
Users are differentiated by the way they expect to interact with the system

- **Application programmers** – interact with system through DML calls
- **Sophisticated users** – form requests in a database query language
- **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework
- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - Examples, people accessing database over the web, bank tellers, clerical staff

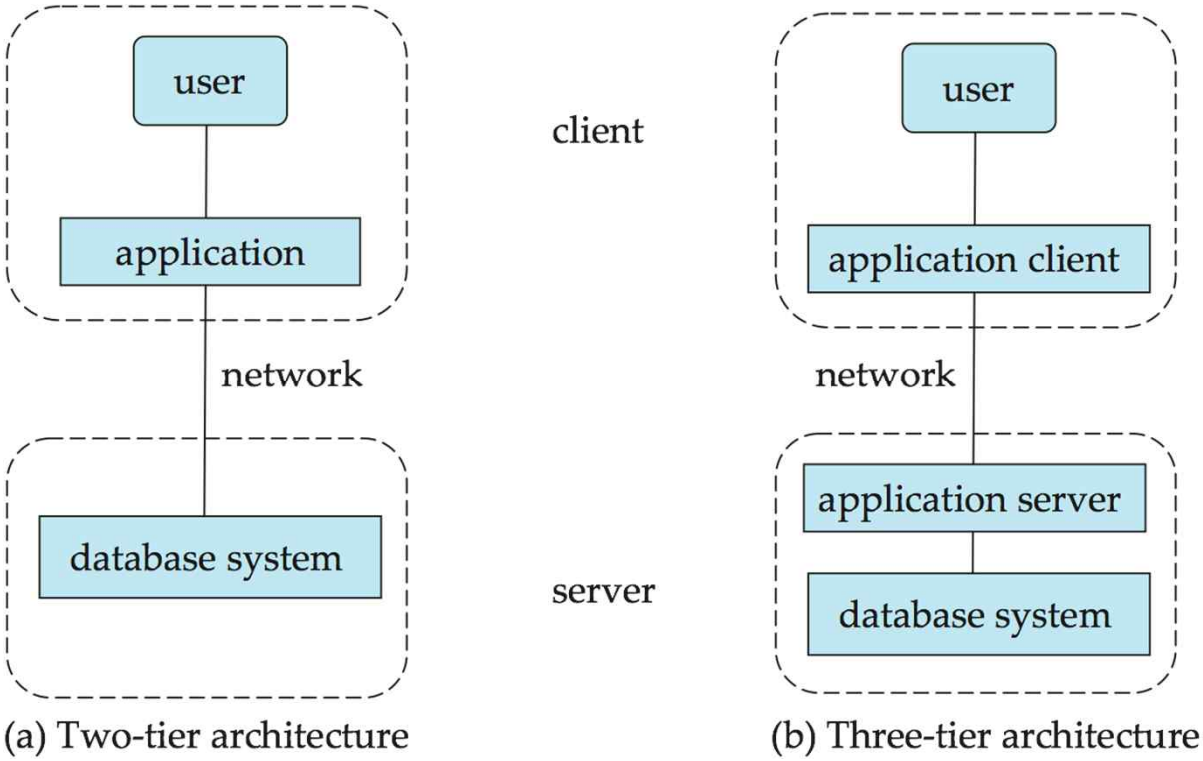
Database Administrator

- 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
  - 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

Overall Database System Structure



Application Architecture



(a) Two-tier architecture (b) Three-tier architecture

- **Two-tier architecture**: application programs communicate with DBMS using API standards (like ODBC/JDBC)
- **Three-tier architecture**: "middleware" (like WAS – Web application server) is used for accessing data

History of Database Systems

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

History of Database Systems (Cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

End of Chapter 1