



Database Management Systems

Course Code: CSE309

2022

Chapter 1: Introduction

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



Course Teacher

Dr. Mrinal Kanti Baowaly

Associate Professor

Department of Computer Science
and Engineering, Bangabandhu
Sheikh Mujibur Rahman Science
and Technology University,
Bangladesh.

Email: baowaly@gmail.com





Introduction to Database Management System

Data

Data are the collection of facts, values, numbers, text, sound, picture or video and which carry a certain meaning in a context.

Database

A database is a large collection of interrelated data.

A database is an organized collection of information.



Database Applications Examples

- Enterprise Information
 - Sales: customers, products, purchases
 - Accounting: payments, receipts, assets
 - Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
 - customer information, accounts, loans, and banking transactions.
 - Credit card transactions
 - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data)
- Universities: registration, grades



Database Applications Examples (Cont.)

- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services
 - Online retailers: order tracking, customized recommendations
 - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.



Purpose of Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- Data redundancy and inconsistency: data is stored in multiple file formats resulting the same information may be duplicated in different files

The various copies of the same data may no longer agree

- Difficulty in accessing data
 - Need to write a new program to carry out each new task
- Data isolation
 - 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



Purpose of Database Systems (Cont.)

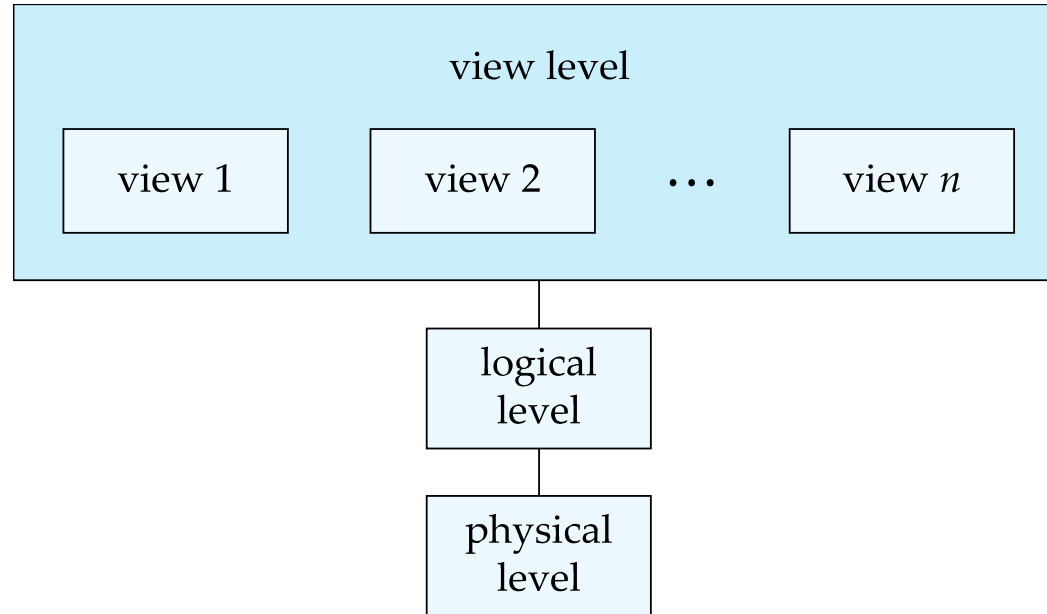
- Atomicity of updates
 - Failures may leave database in an inconsistent state with partial updates carried out
 - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
 - Concurrent access needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies
 - Ex: 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



View of Data

An architecture for a database system





Instances and Schemas

- Similar to types and variables in programming languages
- **Schema**: the overall design of the database
 - Analogous to the variable declarations (along with associated type definitions) in a program
- **Instance**: the collection of information stored in the database at a particular moment
 - Analogous to the value of a variable in a program
- **Physical schema** – the physical schema describes the database design at the physical level,
- **Logical Schema** – the logical schema describes the database design at the logical level.



Physical Data Independence

- **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
- A data model provides a way to describe the design of a database at the physical, logical, and view levels
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
 - Network model
 - Hierarchical model



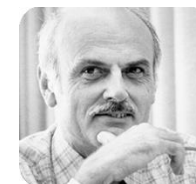
Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model

Columns

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

Rows



Ted Codd
Turing Award 1981

(a) The *instructor* table



A Sample Relational Database

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

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



Database Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)



Data Definition Language (DDL)

- Specification notation for defining the database schema

Example: **create table** *instructor* (
 ID **char**(5),
 name **varchar**(20),
 dept_name **varchar**(20),
 salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
 - Database schema
 - Integrity constraints
 - Primary key (ID uniquely identifies instructors)
 - Authorization
 - Who can access what



Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model. The types of access are:
 - Retrieval of information stored in the database
 - Insertion of new information into the database
 - Deletion of information from the database
 - Modification of information stored in the database
- There are basically two types of data-manipulation language
 - **Procedural DML** -- require a user to specify what data are needed and how to get those data.
 - **Declarative DML** -- require a user to specify what data are needed without specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as **non-procedural DMLs**
- The portion of a DML that involves information retrieval is called a **query language**.



SQL Query Language

- SQL query language is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

```
select name  
from instructor  
where dept_name = 'Comp. Sci.'
```



Database Access from Application Program

- Application programs generally access databases through one of
 - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database and retrieve the results
 - Language extensions to allow embedded SQL
- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine i.e. some computations are not possible using SQL
- 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.



Database Design

Database design is the process of designing the general structure of the database:

- **Requirements Analysis** – The outcome of this phase is a specification of user requirements of an enterprise. The database designer needs to interact extensively with domain experts and users to carry out this task.
- **Conceptual Design** – The designer chooses a data model (e.g. ER Model), and translates the user requirements into a conceptual schema of the database.
- **Logical Design** – Deciding on the database schema. The designer maps the high-level conceptual schema onto the implementation data model of the database system and results a system-specific database schema (using e.g. Normalization).
- **Physical Design** – Deciding on the physical layout of the database. These features include the form of file organization and the internal storage structures.



Functional Components of Database System

- 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
 - File manager
 - Buffer manager



Storage Manager (Cont.)

- 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.



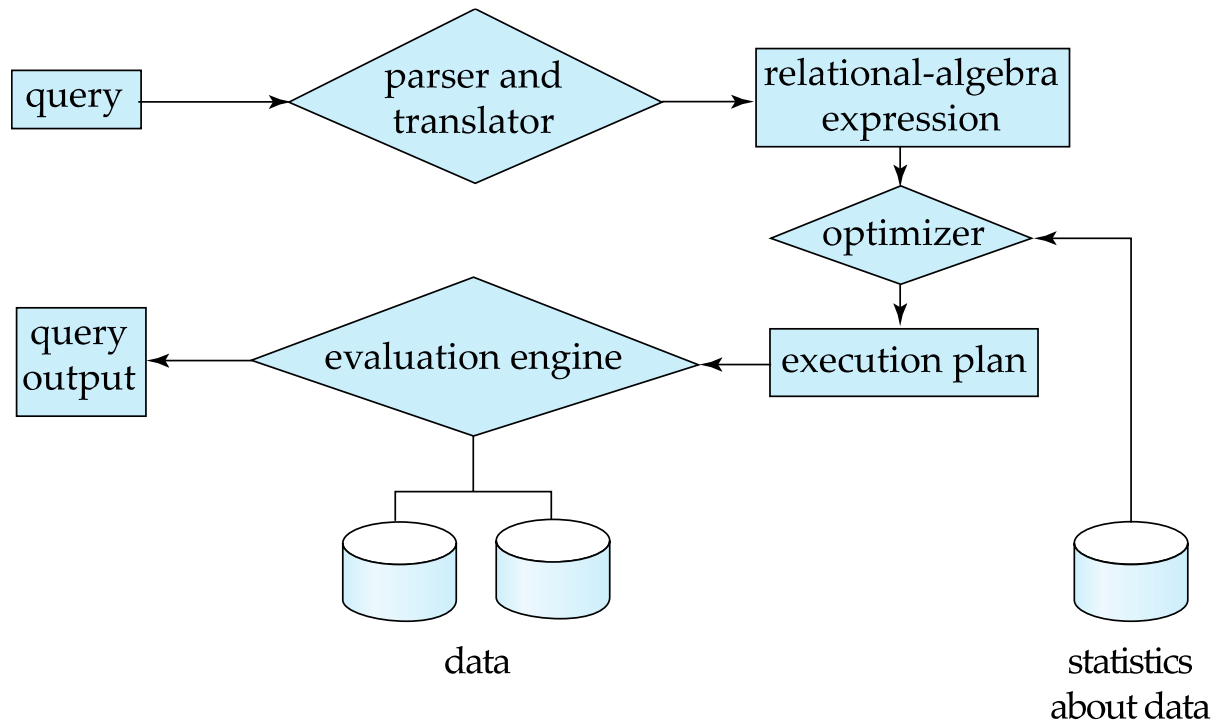
Query Processor

- 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.



Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



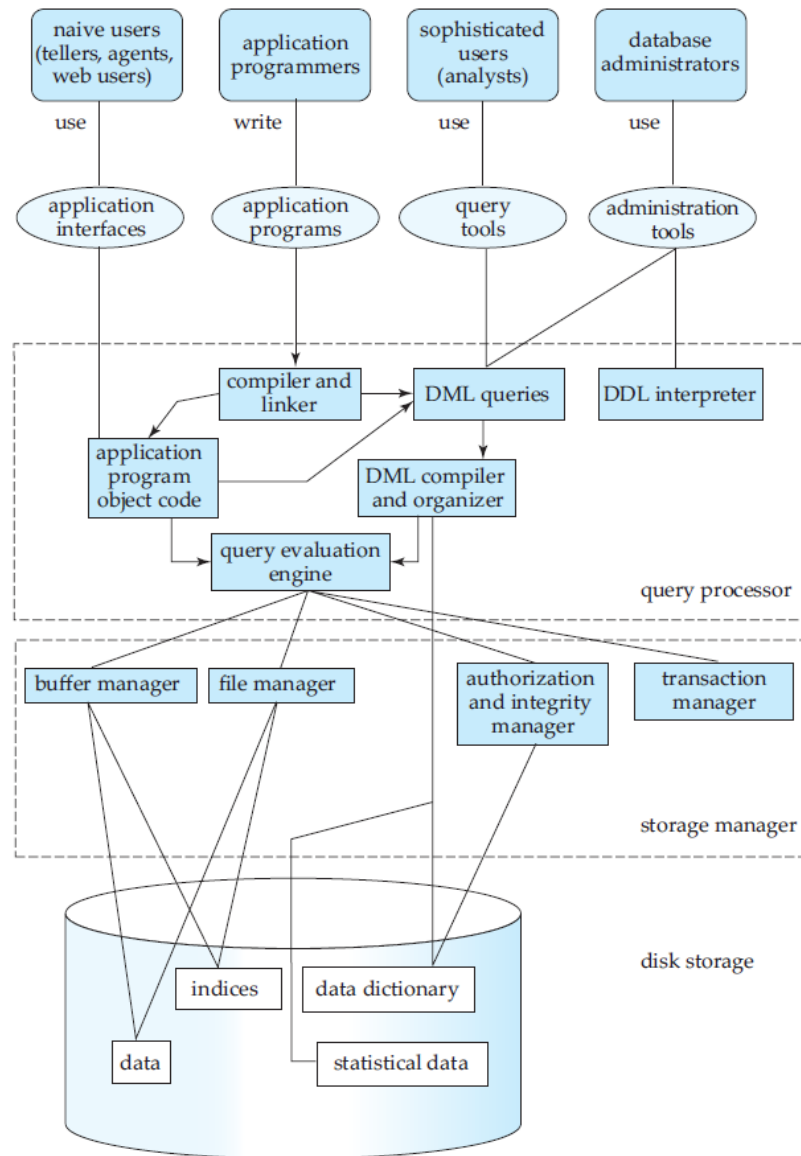


Transaction Management

- 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.



Database System Architecture





Database Architecture

- Centralized databases
 - One to a few cores, shared memory
- Client-server,
 - One server machine executes work on behalf of multiple client machines.
- Parallel databases
 - Many core shared memory
 - Shared disk
 - Shared nothing
- Distributed databases
 - Geographical distribution
 - Schema/data heterogeneity



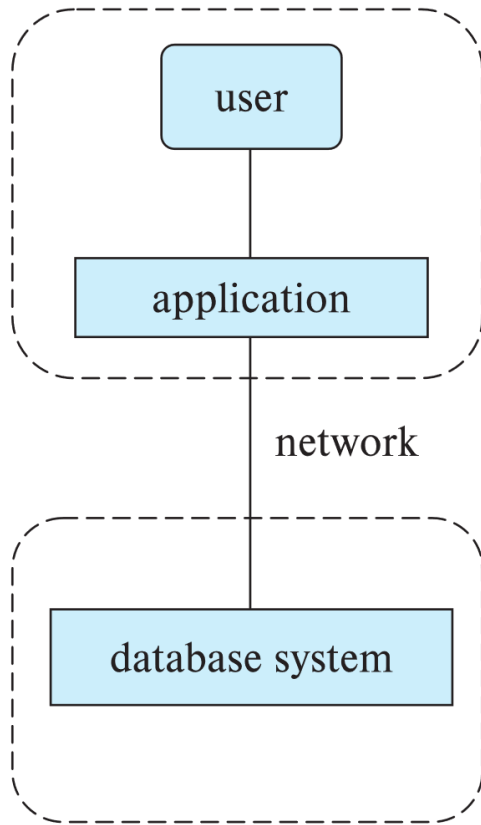
Database Applications

Database applications are usually partitioned into two or three parts

- Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine
- Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.
 - The client end communicates with an application server, usually through a forms interface.
 - The application server in turn communicates with a database system to access data.



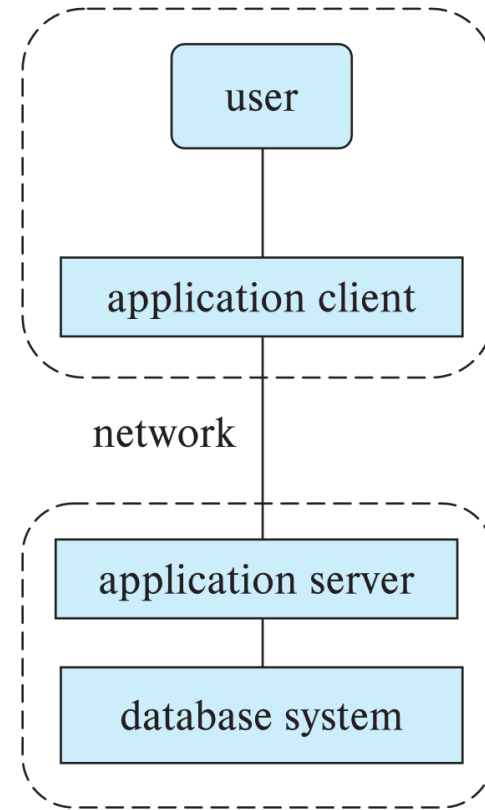
Two-tier and three-tier architectures



(a) Two-tier architecture

client

server



(b) Three-tier architecture



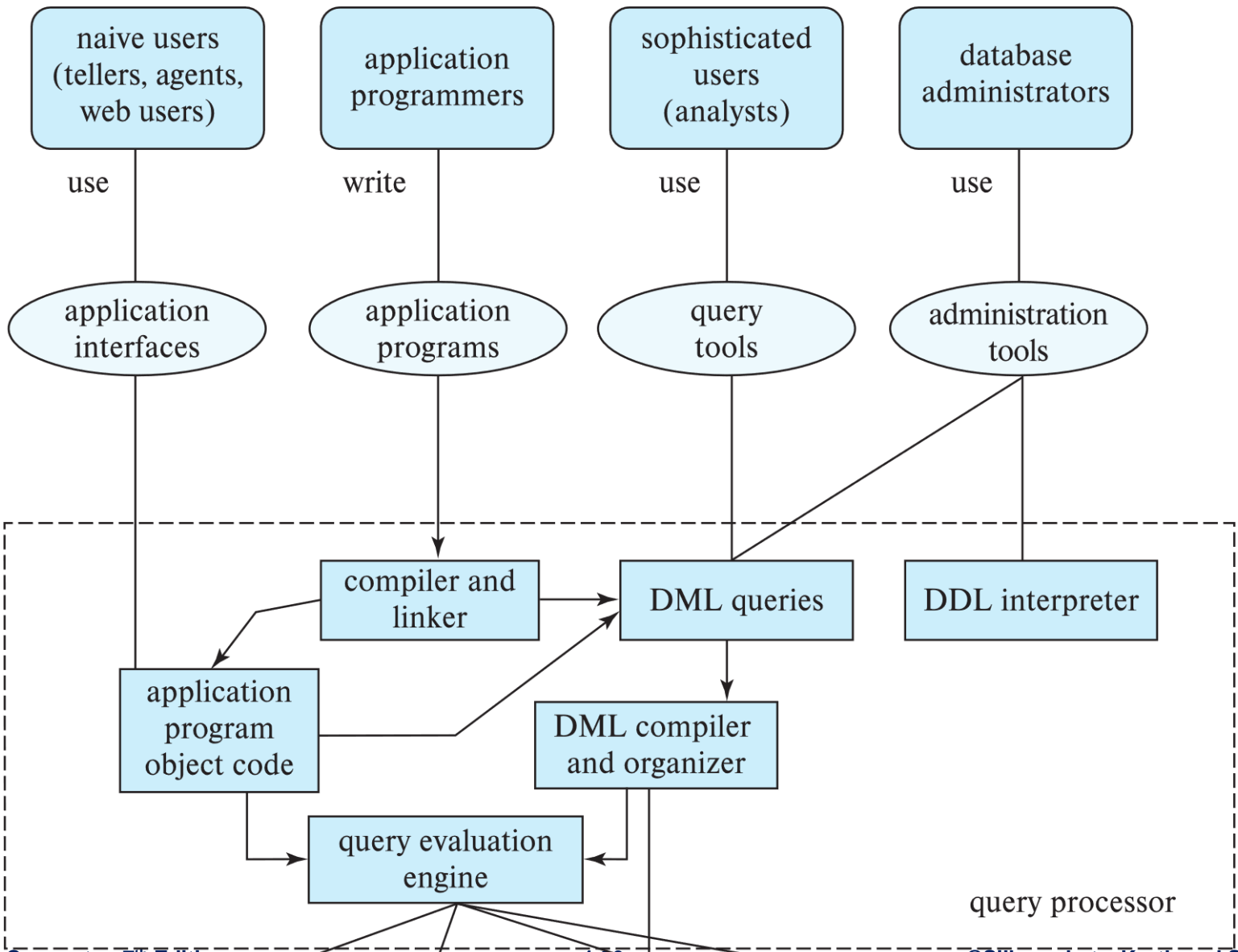
Database Users

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

- **Naive users** – invoke one of the permanent application programs that have been written previously
 - Use application interfaces
 - Examples, people accessing database over the web, bank tellers, clerical staff
- **Application programmers** – interact with system through DML calls
 - Write application programs
- **Sophisticated users** – form requests in a database query language
 - Use Query tools
 - Example, analysts who submit queries to explore data
- **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework



Database Users





Database Administrator

A person who has central control over the system is called a **database administrator (DBA)**. Functions of a DBA include:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database



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
 - Ted Codd defines the relational data model
 - Would win the ACM Turing Award for this work
 - IBM Research begins System R prototype
 - UC Berkeley (Michael Stonebraker) begins Ingres prototype
 - Oracle releases first commercial relational database
 - 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
 - Wisconsin, IBM, Teradata
 - Object-oriented database systems
- 1990s:
 - Large decision support and data-mining applications
 - Large multi-terabyte data warehouses
 - Emergence of Web commerce



History of Database Systems (Cont.)

- 2000s
 - Big data storage systems
 - Google BigTable, Yahoo PNuts, Amazon,
 - “NoSQL” systems.
 - Big data analysis: beyond SQL
 - Map reduce and friends
- 2010s
 - SQL reloaded
 - SQL front end to Map Reduce systems
 - Massively parallel database systems
 - Multi-core main-memory databases



End of Chapter 1