IT615 – Data Base Management System

Dr. Manish Khare

Introduction

Database Systems

➢ DBMS contains information about a particular enterprise

- Collection of interrelated data

- Set of programs to access the data

- An environment that is both *convenient* and *efficient* to use

➢ Database systems are used to manage collections of data that are:

- Highly valuable

- Relatively large

- Accessed by multiple users and applications, often at the same time.

➢ A modern database system is a complex software system whose task is to manage a large, complex collection of data.

➢ Databases touch all aspects of our lives

Database Applications Examples

- ➢ Enterprise Information
  - Sales: customers, products, purchases information
  - 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

  ▪ Social-media

  ▪ Online retailers: order tracking, customized recommendations

  ▪ Online advertisements

➢ Document databases

➢ Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

University Database Example

➢ In this text we will be using a university database to illustrate all the concepts

➢ Data consists of information about:

- Students

- Instructors

- Classes

➢ Application program examples:

- Add new students, instructors, and courses

- Register students for courses, and generate class rosters

- Assign grades to students, compute grade point averages (GPA) and generate transcripts

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 induplication of information in different files

➢ 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

➤ A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.

➤ A major purpose of a database system is to provide users with an abstract view of the data.

- Data models
  - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
- Data abstraction
  - Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

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)
- ➢ Semi-structured data model (XML)
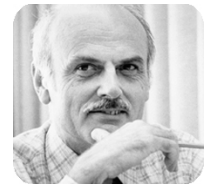- ➢ Other older models:
  - Network model
  - Hierarchical model

Relational Model

➢ All the data is stored in various tables.

➢ Example of tabular data in the relational model

Columns

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

Rows

(a) The *instructor* table

Ted Codd
Turing Award 1981

A Sample Relational Database

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

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

(b) The *department* table

View of Data

An architecture for a database system
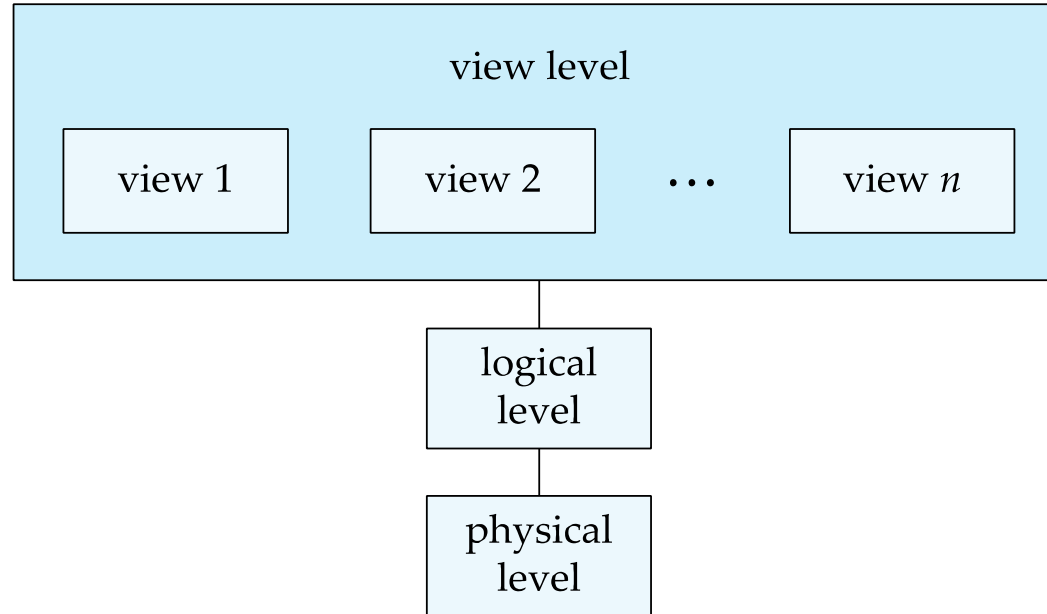
Instances and Schemas

- Similar to types and variables in programming languages

- **Logical Schema** – the overall logical structure of the database

  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them

    - Analogous to type information of a variable in a program

- **Physical schema** – the overall physical structure of the database

- **Instance** – the actual content of the database at a particular point in time

  - Analogous to the value of a variable

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 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

  ▪ DML also known as query language

➢ 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.'

➤ SQL is **NOT** a Turing machine equivalent language

➤ To be able to compute complex functions SQL is usually embedded in some higher-level 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

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.

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?
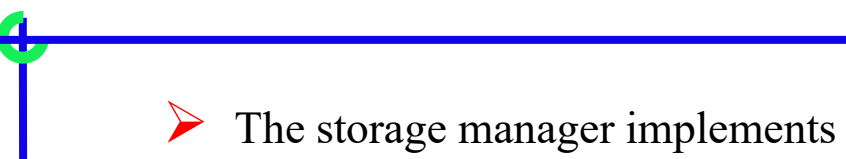
➢ Physical Design – Deciding on the physical layout of the database

Database Engine

➢ 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.
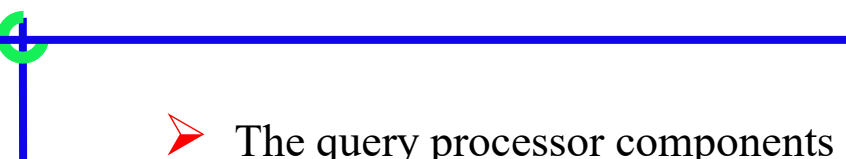
Storage Manager

➢ 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
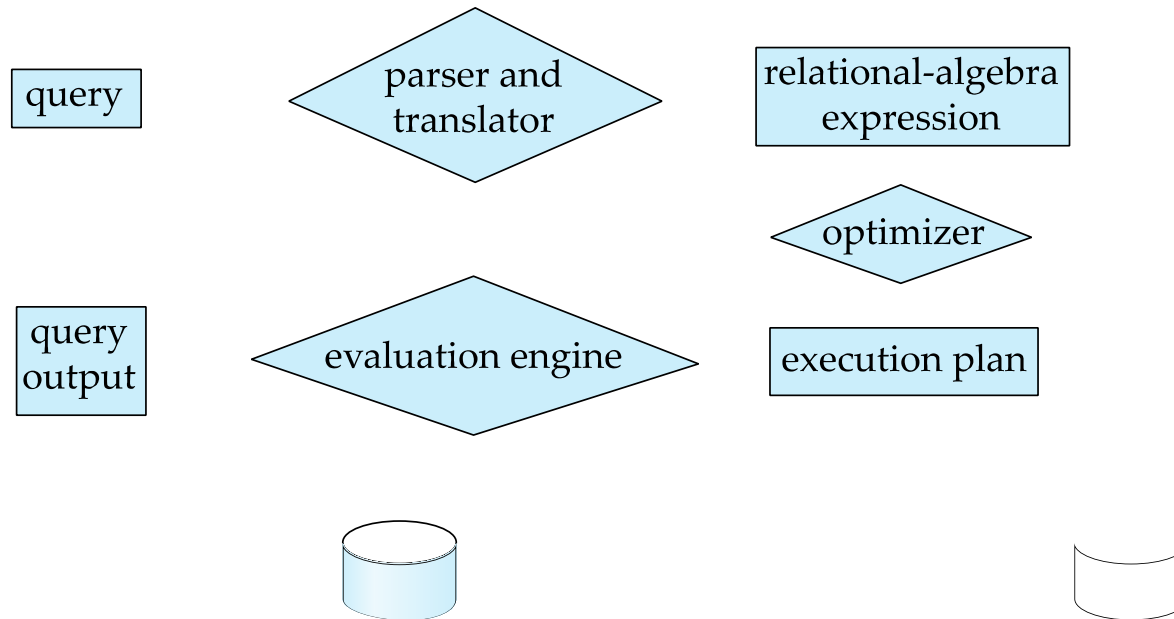- Transaction manager
- File manager
- Buffer manager

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 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 Architecture (Centralized/Shared-Memory)
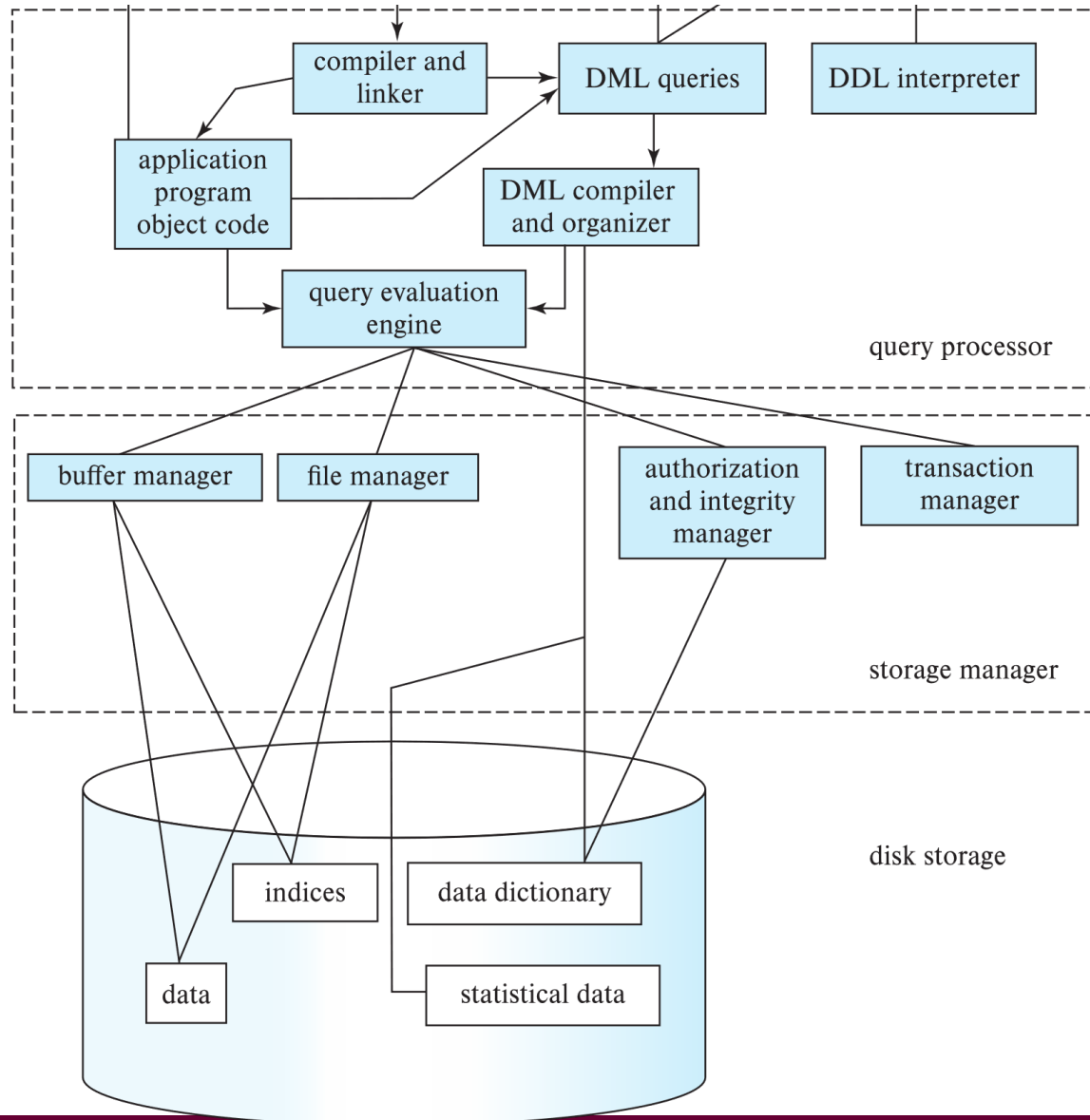
Database Applications

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.
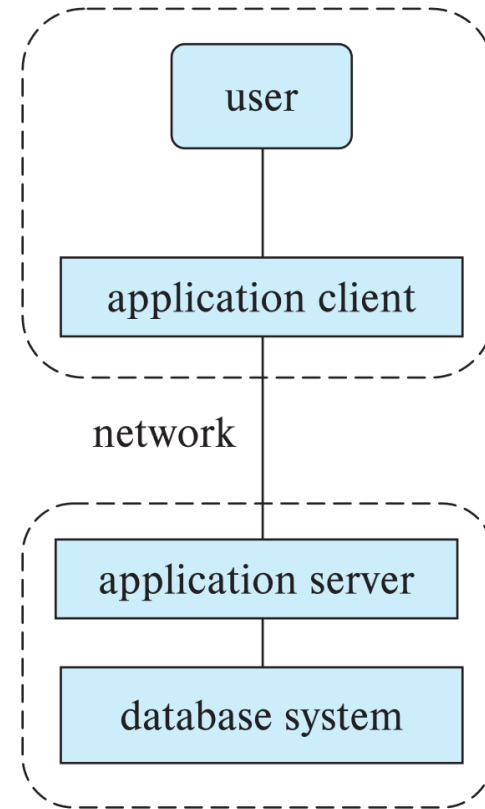
Two-tier and three-tier architectures
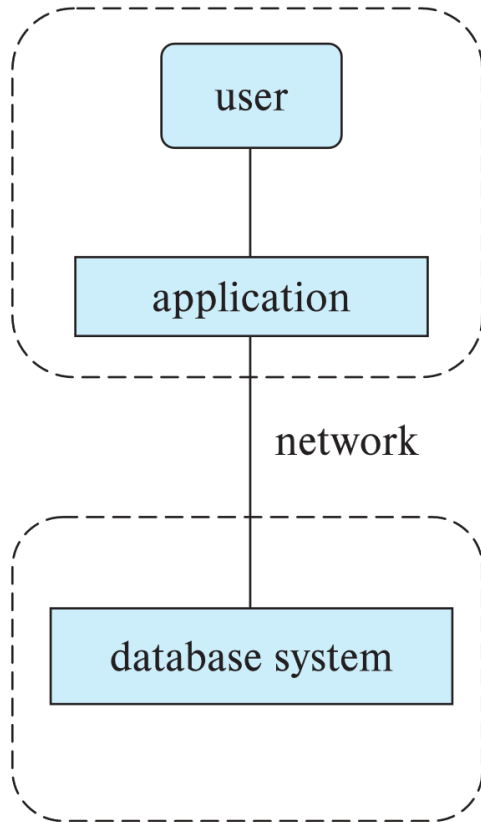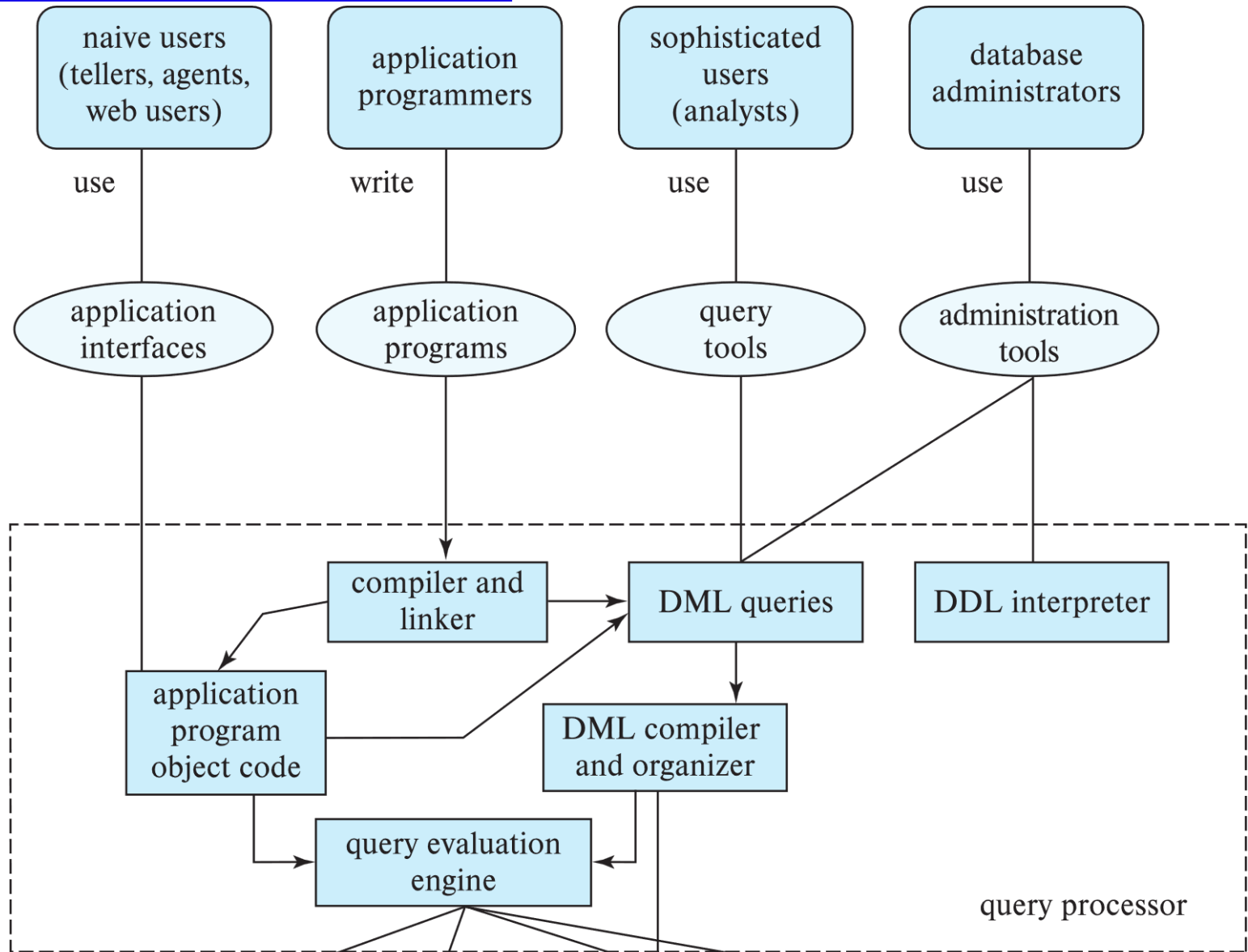

(a) Two-tier architecture

(b) Three-tier architecture

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

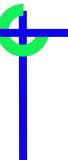# History of Database Systems (Cont.)

- ➢ 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
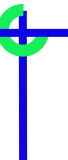   - Multi-core main-memory databases

Summary

➢ A database-management system (DBMS) consists of a collection of interrelated data and a collection of programs to access those data. The data describe one particular enterprise.

➢ The primary goal of a DBMS is to provide an environment that is both convenient and efficient for people to use in retrieving and storing information.

➢ Database systems are ubiquitous today, and most people interact, either directly or indirectly, with databases many times every day.

➢ Database systems are designed to store large bodies of information. The management of data involves both the definition of structures for the storage of information and the provision of mechanisms for the manipulation of information. In addition, the database system must provide for the safety of the information stored in the face of system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

➤ 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.

➤ Underlying the structure of a database is the data model: a collection of conceptual tools for describing data, data relationships, data semantics, and data constraints.

➤ The relational data model is the most widely deployed model for storing data in databases. Other data models are the object-oriented model, the object-relational model, and semi-structured data models.

➤ A data-manipulation language (DML) is a language that enables users to access or manipulate data. Nonprocedural DMLs, which require a user to specify only what data are needed, without specifying exactly how to get those data, are widely used today.

➢ A data-definition language (DDL) is a language for specifying the database schema and other properties of the data.

➢ Database design mainly involves the design of the database schema. The entity relationship (E-R) data model is a widely used model for database design. It provides a convenient graphical representation to view data, relationships, and constraints.

➢ A database system has several subsystems.

■ The storage manager subsystem provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

■ The query processor subsystem compiles and executes DDL and DML statements.

➢ Transaction management ensures that the database remains in a consistent (correct) state despite system failures. The transaction manager ensures that concurrent transaction executions proceed without conflicts.

➢ The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, or parallel, involving multiple machines. Distributed databases span multiple geographically separated machines.

➢ Database applications are typically broken up into a front-end part that runs at client machines and a part that runs at the backend. In two-tier architectures, the front end directly communicates with a database running at the back end. In three tier architectures, the back end part is itself broken up into an application server and a database server.

➢ 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.

➢ Data-analysis techniques attempt to automatically discover rules and patterns from data. The field of data mining combines knowledge-discovery techniques invented by artificial intelligence researchers and statistical analysts with efficient implementation techniques that enable them to be used on extremely large databases.