



# Chapter 1: Introduction

**Database System Concepts, 7<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Outline

- Database-System Applications
- Purpose of Database Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

上層

底層

(注意專有名詞)



# Database-System Applications

- 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
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing (製造業): management of production, inventory (倉儲), orders, supply chain (供應鏈).
- 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 (簡訊), and data usage, generating monthly bills, maintaining balances on prepaid calling cards (預付卡)
- Web-based services
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of various places of interest (POI, 有興趣造訪的地點) 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 in duplication of information in different files
  - 例子: `account (name, address, telephone, account-number, balance)`  
`loan (name, address, telephone, loan-number, amount)`
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
  - 例子: 輸出台北的顧客，再輸出台中的顧客
- Data isolation
  - Multiple files and formats
- Integrity problems
  - **Integrity constraints (完整性限制、整體性限制)** (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 (不可分割性) of updates (see page 29)
  - Failures may leave database in an inconsistent state with partial updates carried out
    - Example: Transfer of funds (say \$1000) from one account (say \$5000) to another (say \$2000) 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**



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



# 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 (資料模式)
    - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - Data abstraction (資料抽象化)
    - Hide the complexity of data structures from users through several levels of data abstraction, to simplify users' interactions with the system.



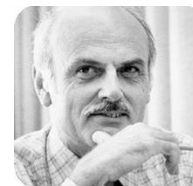
# Data Models

- 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, JSON)
- 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



**Ted Codd**  
Turing Award 1981

Columns

Rows

<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



# 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



# Levels of Abstraction

- **Physical level:** describes *how* a record (e.g., instructor) is stored.
- **Logical level:** describes *what* data stored in database, and the relationships among the data.

- Example (in PASCAL)

**type** *instructor* = **record**

*ID* : string;  
*name* : string;  
*dept\_name* : string;  
*salary* : integer;

**end;**

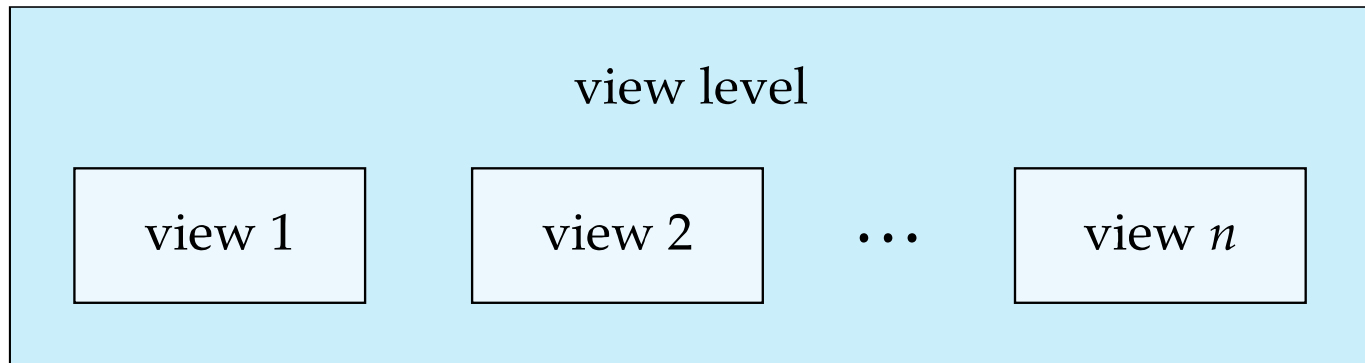
- **View (視界、檢視、視觀) level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.



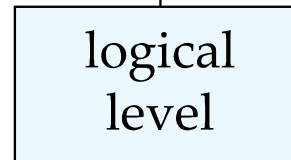
# View of Data

An architecture for a database system

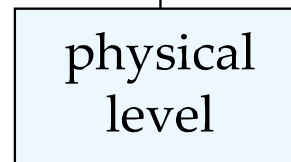
**Naïve  
user**



**Application  
programmer**



**DBA**





# Instances (實例) and Schemas (綱要)

- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database ← 一般 *schema* 是指這種
  - 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
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language



# Data Manipulation Language (Cont.)

- 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 the names of all instructors in Comp. Sci. dept

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

- SQL is **NOT** a Turing (杜林 or 圖靈) machine equivalent language
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- To be able to compute complex functions, SQL is usually embedded in some higher-level language



# Database Access from Application Program

- Complex computations and actions are 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.
- 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 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, including 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 for 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
  - File manager
  - Buffer manager
  - Transaction manager

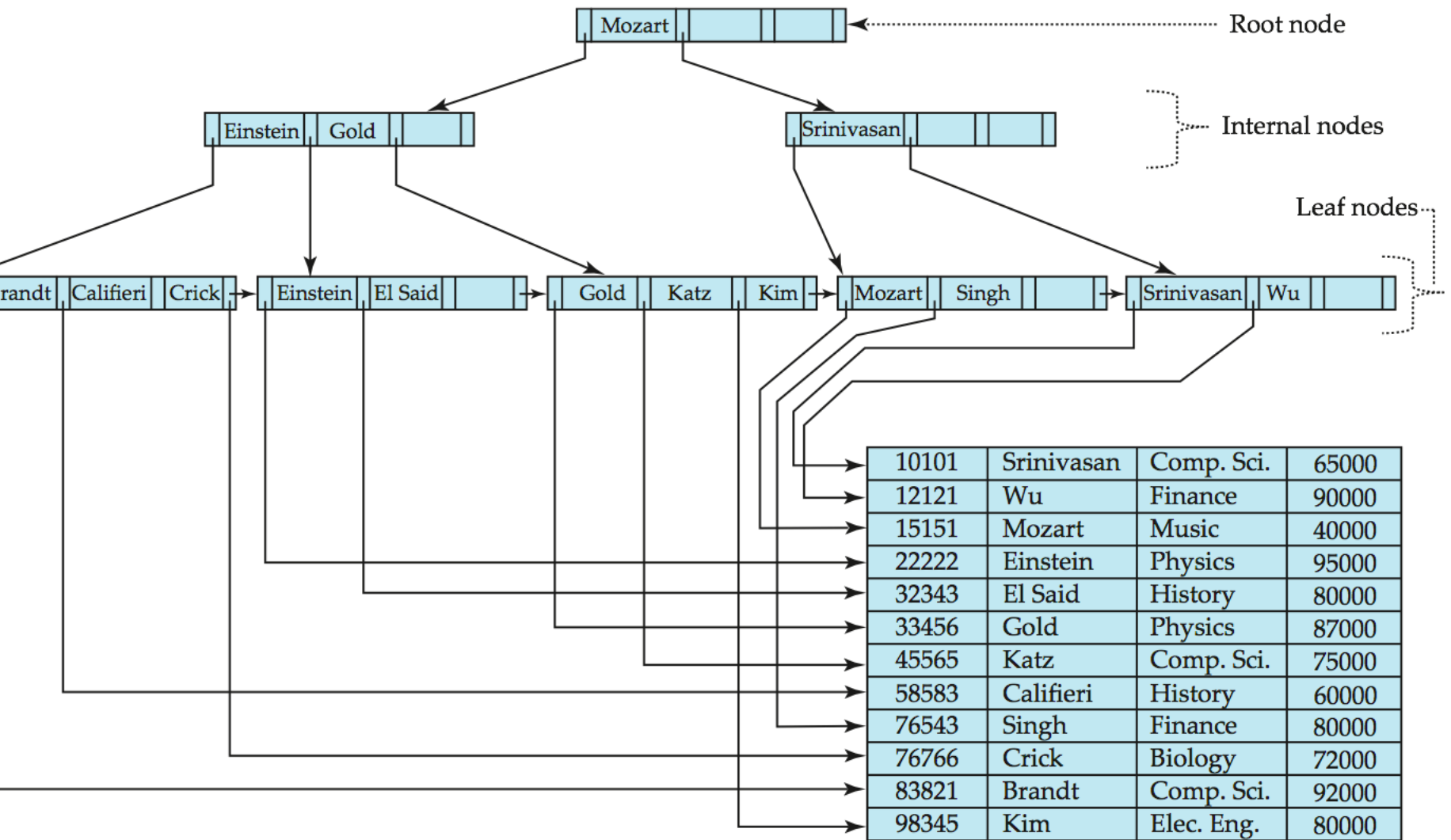


# Storage Manager (Cont.)

- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary (資料字典) -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices (index的複數, 索引) -- can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.



# Example of Index and Data File





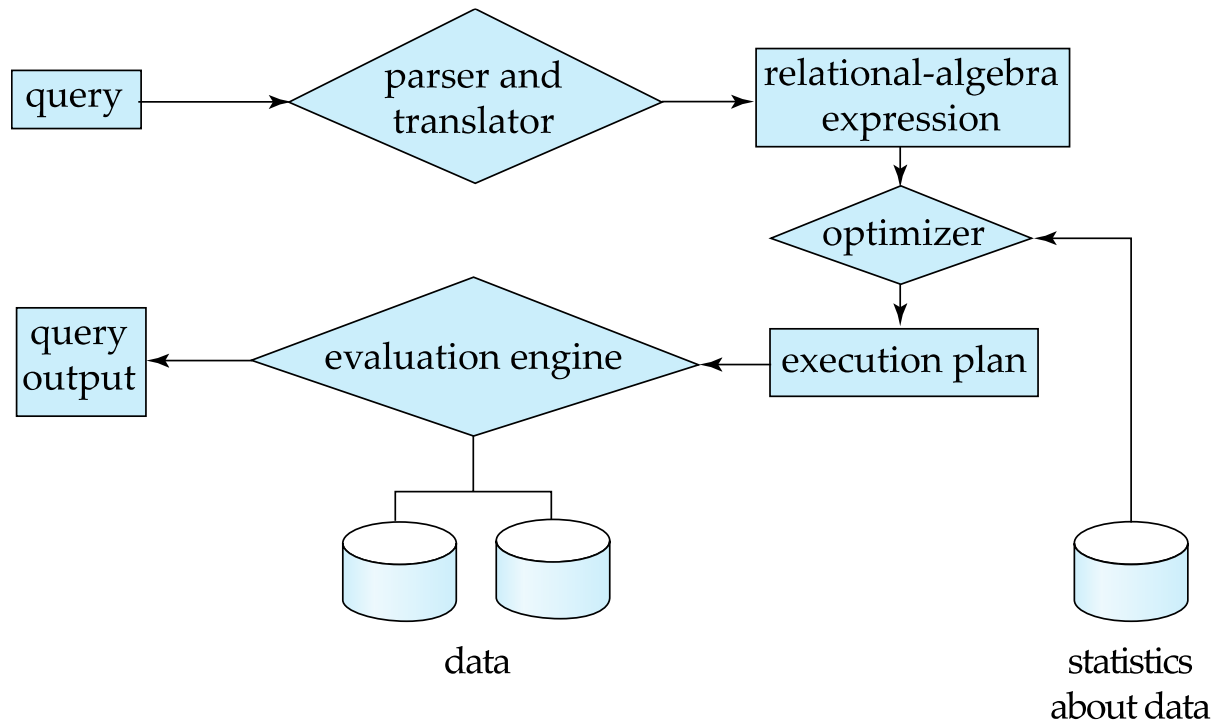
# Query Processor

- 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 (執行計畫) 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 the various alternatives.
  - Query evaluation engine -- executes low-level instructions generated by the DML compiler.



# Query Processing

1. Parsing and translation
2. Optimization (最佳化, 優化)
3. Evaluation (這邊是execution的意思, 也就是執行)





# Transaction Management

- A **transaction (交易, 異動)** is a collection of operations that performs a single logical function in a database application (see page 7)
- **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.

-> 可用log

- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

-> 可用lock

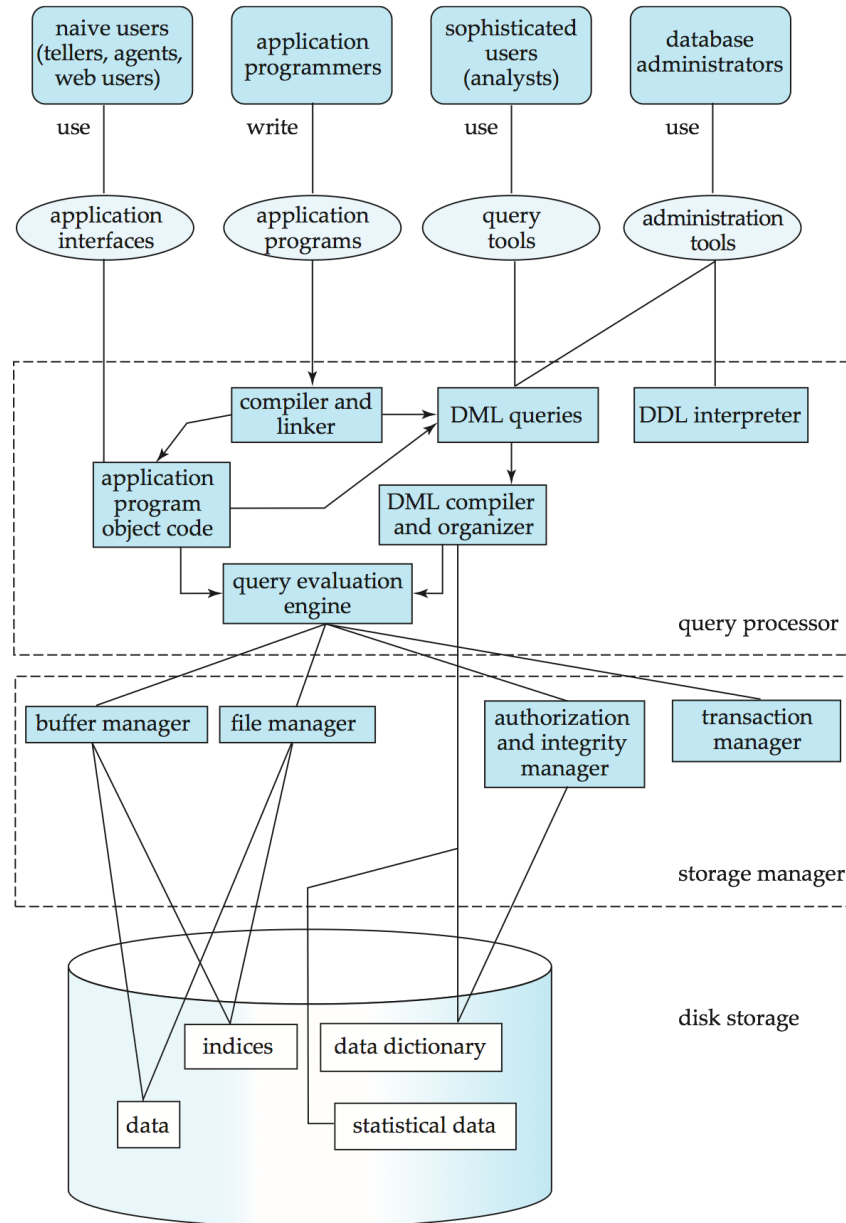


# Database Architecture

- Centralized databases (see the next page)
  - 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



# System Structure





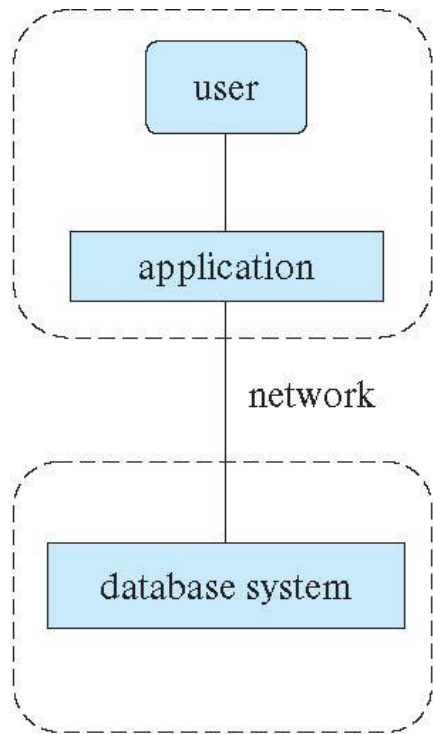
# Database Applications

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 form (表單) interface.
  - The application server in turn communicates with a database system to access data.

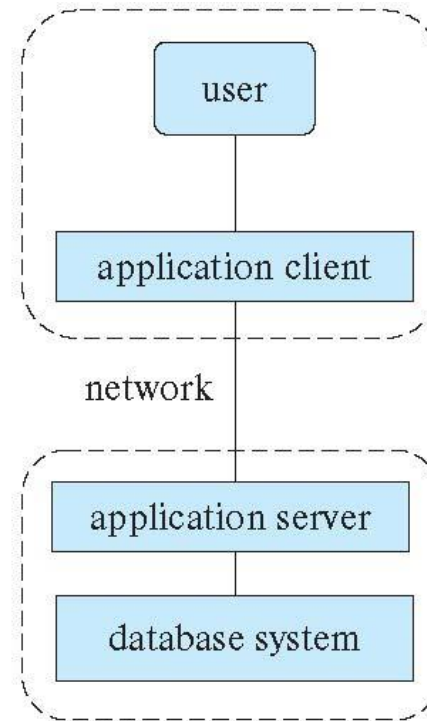


# Two-tier and three-tier architectures



(a) Two-tier architecture

client



server

(b) Three-tier architecture



# Database Users

There are four different types of database-system users

- Naïve (天真的) users -- unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- Application programmers -- are computer professionals who write application programs.
- Sophisticated (有經驗的) users -- interact with the system without writing programs
  - using a database query language or by
  - using tools such as data analysis software.
- Specialized users -- write specialized database applications that do not fit into the traditional data-processing framework. For example, CAD, graphic data, audio, video.



# Database Administrator

A person who has central control over the system is called a **database administrator (DBA)**, whose functions are:

- 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 and ensuring that performance is not degraded by very expensive tasks submitted by some users



# 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
    - win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle (甲骨文) 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