



Chapter 1: Introduction

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan

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



Database Systems

- 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.



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



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
- Universities: registration, grades



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
- Difficulty in accessing data
- Data isolation
 - Multiple files and formats
 - writing new application programs to retrieve the appropriate data is difficult
- 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



Purpose of Database Systems (Cont.)

- 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



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



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

<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

(a) The *instructor* table



Levels of Abstraction

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

type *instructor* = **record**

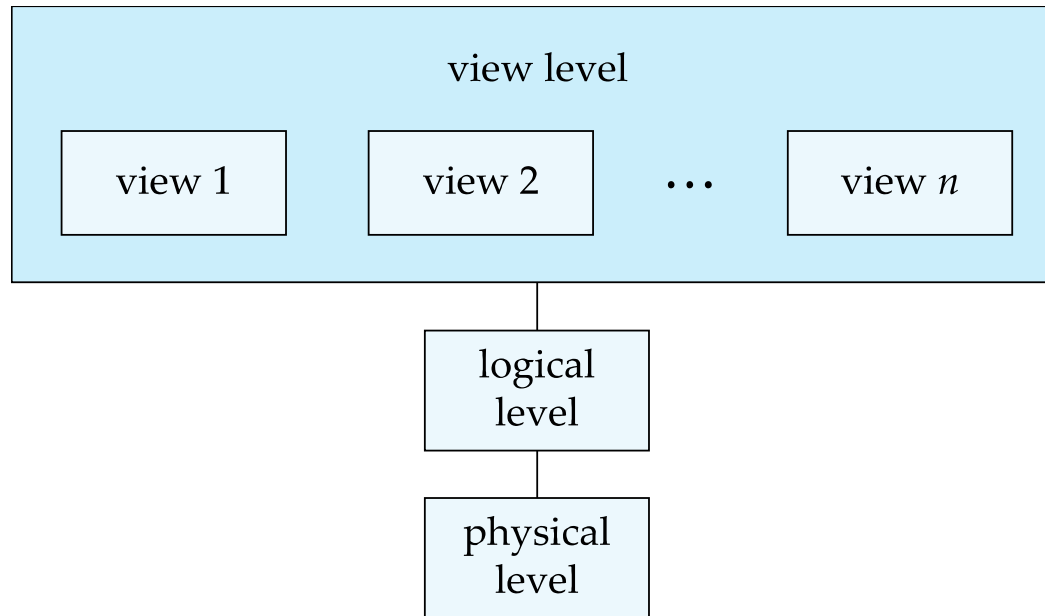
```
ID : string;  
name : string;  
dept_name : string;  
salary : integer;  
end;
```

- **View 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





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



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



SQL Query Language

- 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 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
 - tests for the satisfaction of integrity constraints and checks the authority of users to access data.
 - Transaction manager
 - ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicts.
 - File manager
 - manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
 - Buffer manager
 - It is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.



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.



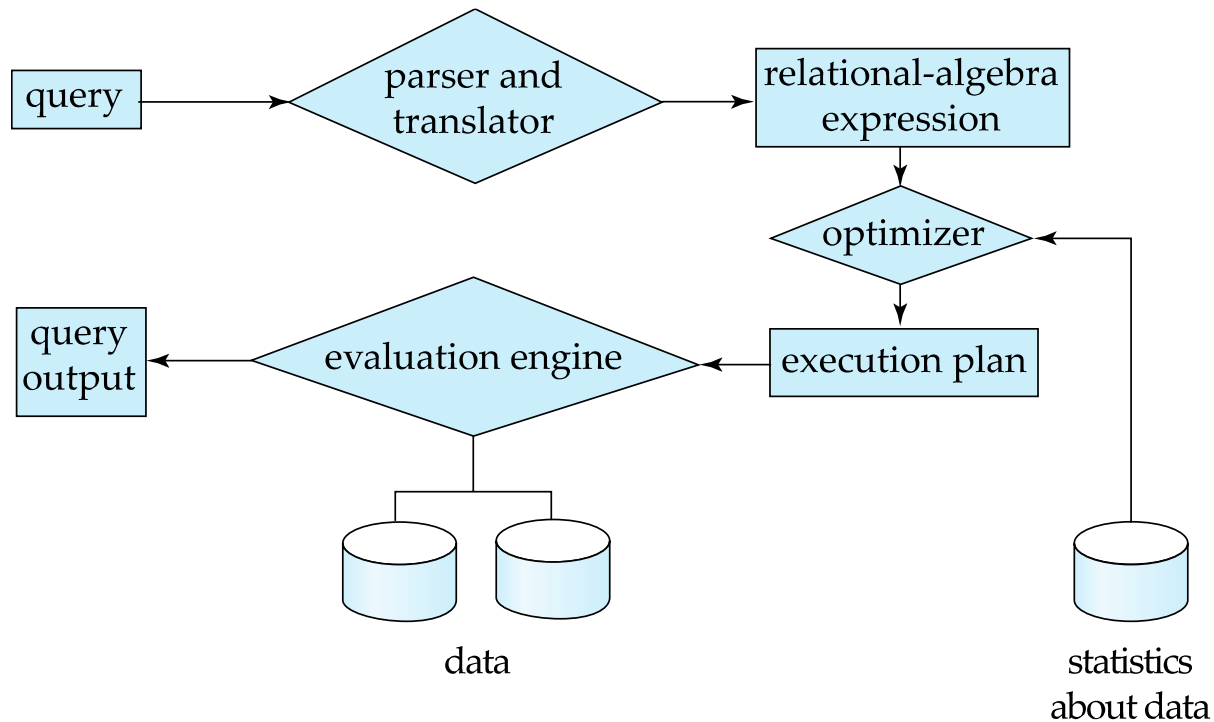
Query Processor

- 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.



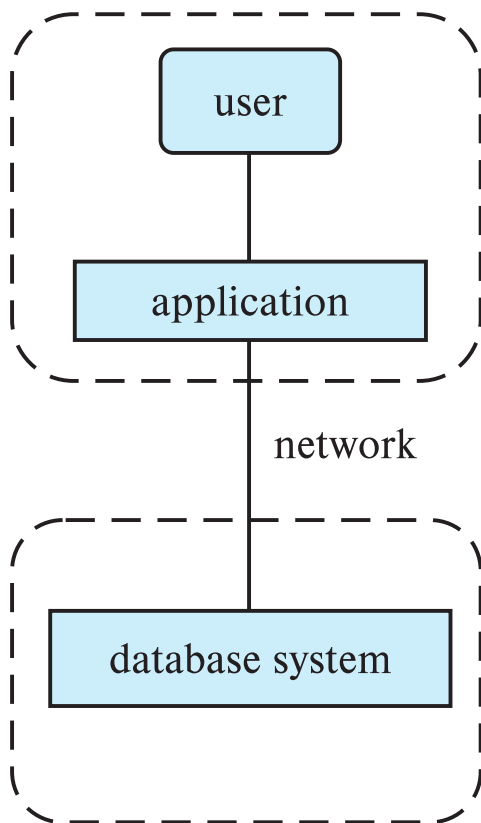
Database Applications

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.



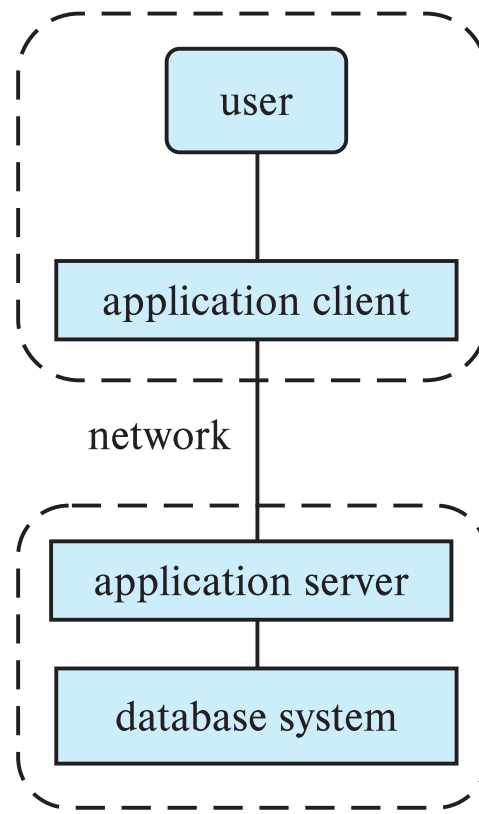
Two-tier and three-tier architectures



(a) Two-tier architecture

client

server



(b) Three-tier architecture



Database 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