



# Chapter 1: Introduction

**Database System Concepts, 6<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

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



# Chapter 1: Introduction

- Purpose of Database Systems
- View of Data
- Database Languages
- Relational Databases
- Database Design
- Data Storage and Querying
- Transaction Management
- Database Users and Administrators
- Database Architecture
- Overall Structure
- History of Database Systems



# Database Management System (DBMS)

- What is a DBMS?
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: all 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
- Databases touch all aspects of our lives



# University Database Example

- 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



# Purpose of Database Systems

- In the early days, database applications were built directly on top of file systems
- Drawbacks of using file systems to store data:



# Purpose of Database Systems

- In the early days, database applications were built directly on top of file systems
- Drawbacks of using file systems to store data:
  - Difficulty in accessing data
    - ▶ Need to write a new program to carry out each new task
  - Data redundancy and inconsistency
    - ▶ Multiple file formats, duplication of information in different files
  - Integrity problems
    - ▶ Integrity constraints (e.g.  $\text{account balance} > 0$ ) become “buried” in program code rather than being stated explicitly
    - ▶ Hard to add new constraints or change existing ones



# Purpose of Database Systems (Cont.)

- Drawbacks of using file 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
      - Example: Two people reading a balance and updating it 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





# Purpose of Database Systems (Cont.)

## ■ Drawbacks of using file 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

read a  
read b  
write a-50  
write b+50

Example: Two people reading a balance and updating it at the same time

problems

to provide user access to some, but not all, data

systems offer solutions to all the above problems



# Purpose of Database Systems (Cont.)

## ■ Drawbacks of using file 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

read a  
read b  
write a-50  
write b+50

Example: Two people reading a balance and updating it at the same time

problems

to provide user access to some, but not all, data

systems offer solutions to all the above problems



# Purpose of Database Systems (Cont.)

## ■ Drawbacks of using file 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

```
read a
read b
write a-50
write b+50
```

Example: Two people reading  
at the same time  
no problems  
to provide user access to  
systems offer solutions to all

```
a == 50
read a
write a-50
read a
write a-50
```



# Levels of Abstraction

- **Physical level:** describes **how** data is stored.
- **Logical level:** describes **what** 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.



# Levels of Abstraction

- **Physical level:** describes **how** data is stored.
- **Logical level:** describes **what** data stored in database, and the relationships among the data.

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

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

**end;**

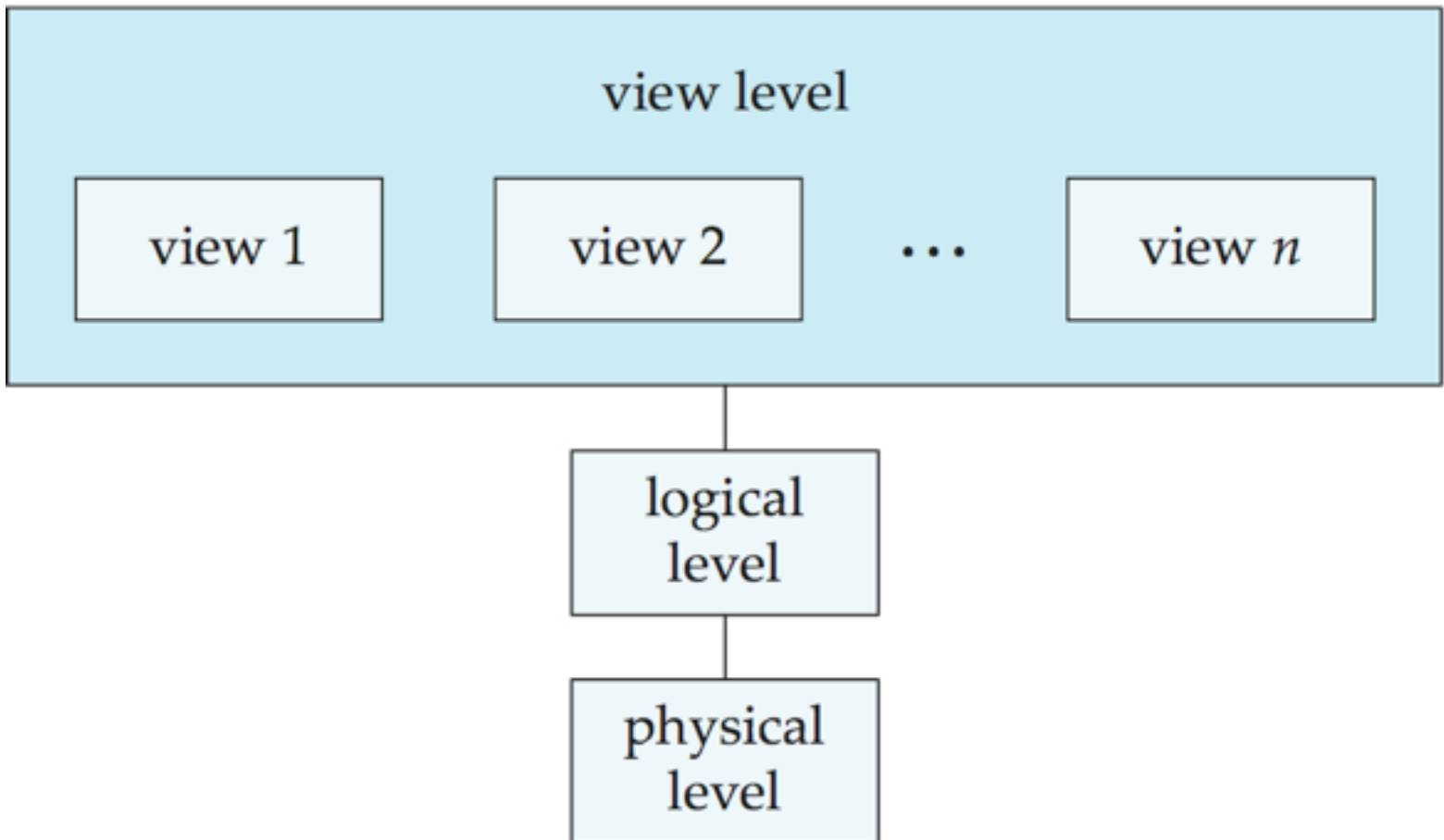
<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Kim	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.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

- **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
- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of **customers** and **accounts** and the **relationship** between them)
  - Analogous to type information of a **variable** in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema



# 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

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



# 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



# 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))

- **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
  - DML also known as query language
- Two classes of languages
  - **Procedural** – user specifies what data is required and **how** to get those data
  - **Declarative (nonprocedural)** – user specifies **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"
```

## ■ Chapters 3, 4 and 5



# 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 Design?

- Is there any problem with this design?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000



# Design Approaches

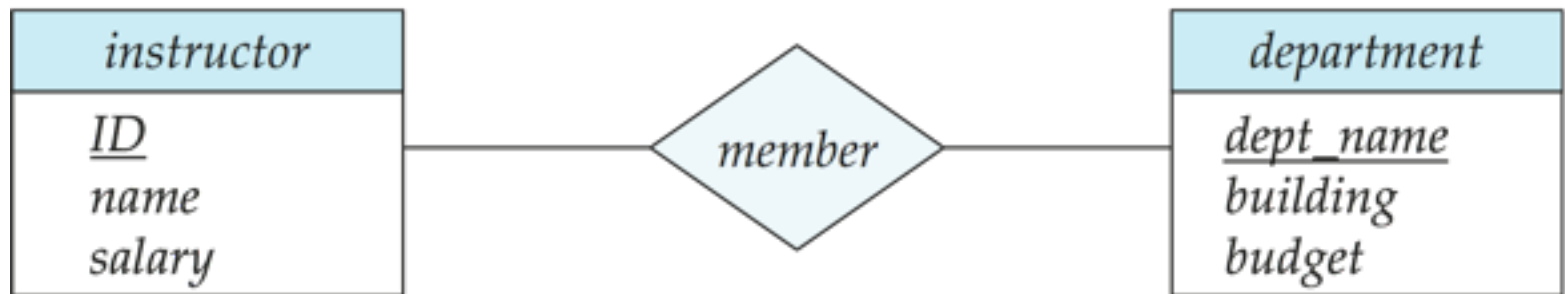
- Entity Relationship Model (Chapter 7)
  - Models an enterprise as a collection of *entities* and *relationships*
  - Represented diagrammatically by an *entity-relationship diagram*:
- Normalization Theory (Chapter 8)
  - Formalize what designs are bad, and test for them





# 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*:





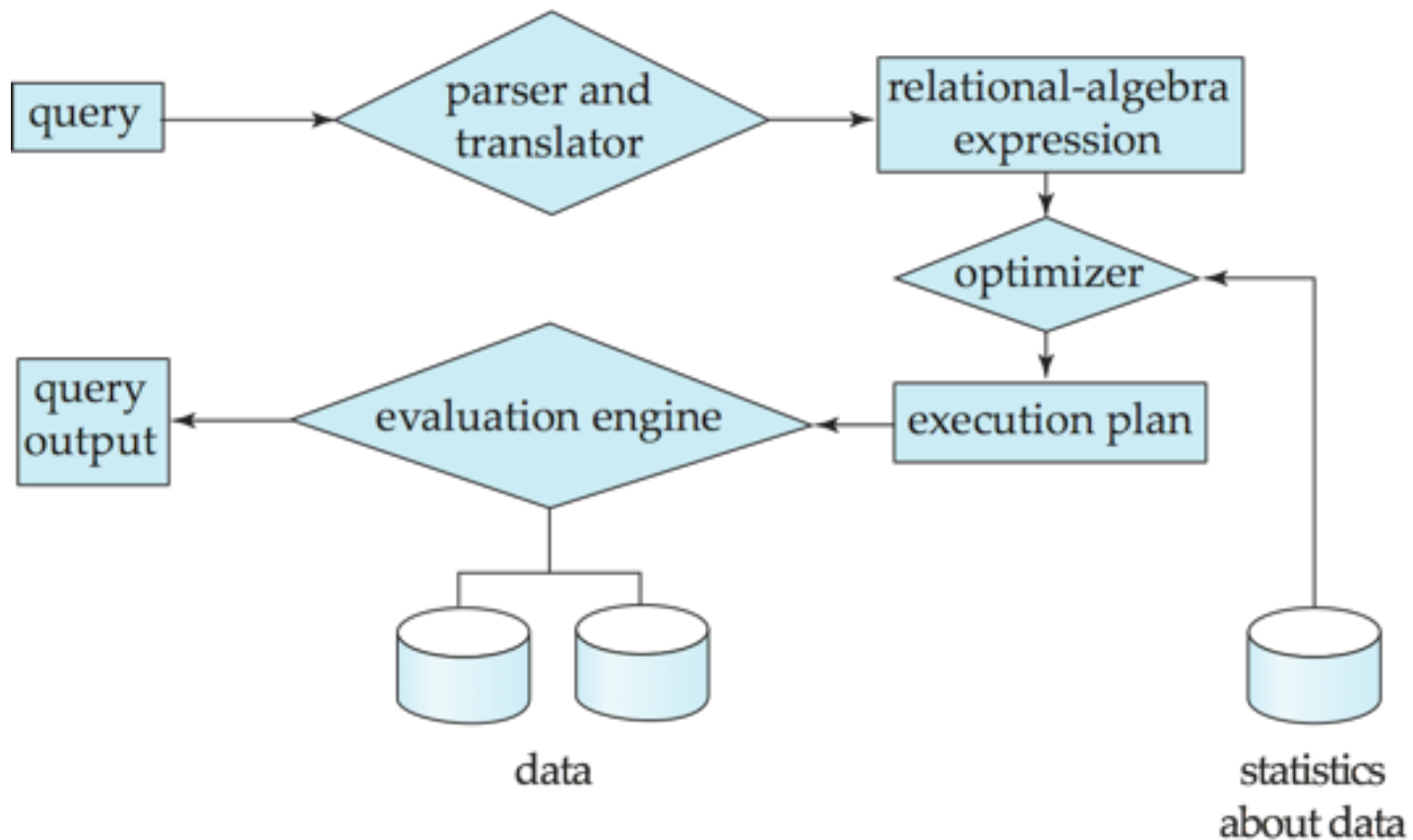
# Storage Management

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



# Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation





# Query Optimization

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions
- How to find the evaluation among many that minimizes the cost?



# Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **failure recovery 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.



# Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **failure recovery 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.

read a

read b

write a-50

write b+50



# Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **failure recovery 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.

read a

read b

write a-50

write b+50



# Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **failure recovery 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.

read a

read b

write a-50

write b+50

a == 50

read a

write a-50

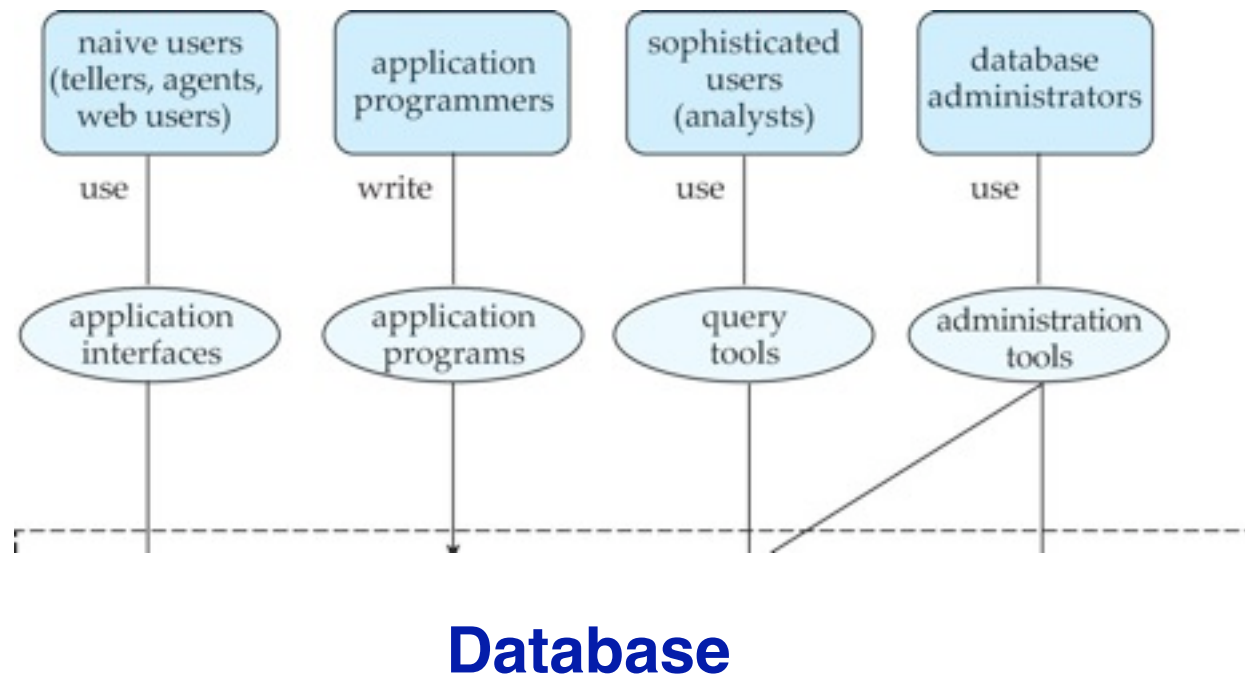
read a

write a-50





# Database Users and Administrators





# 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



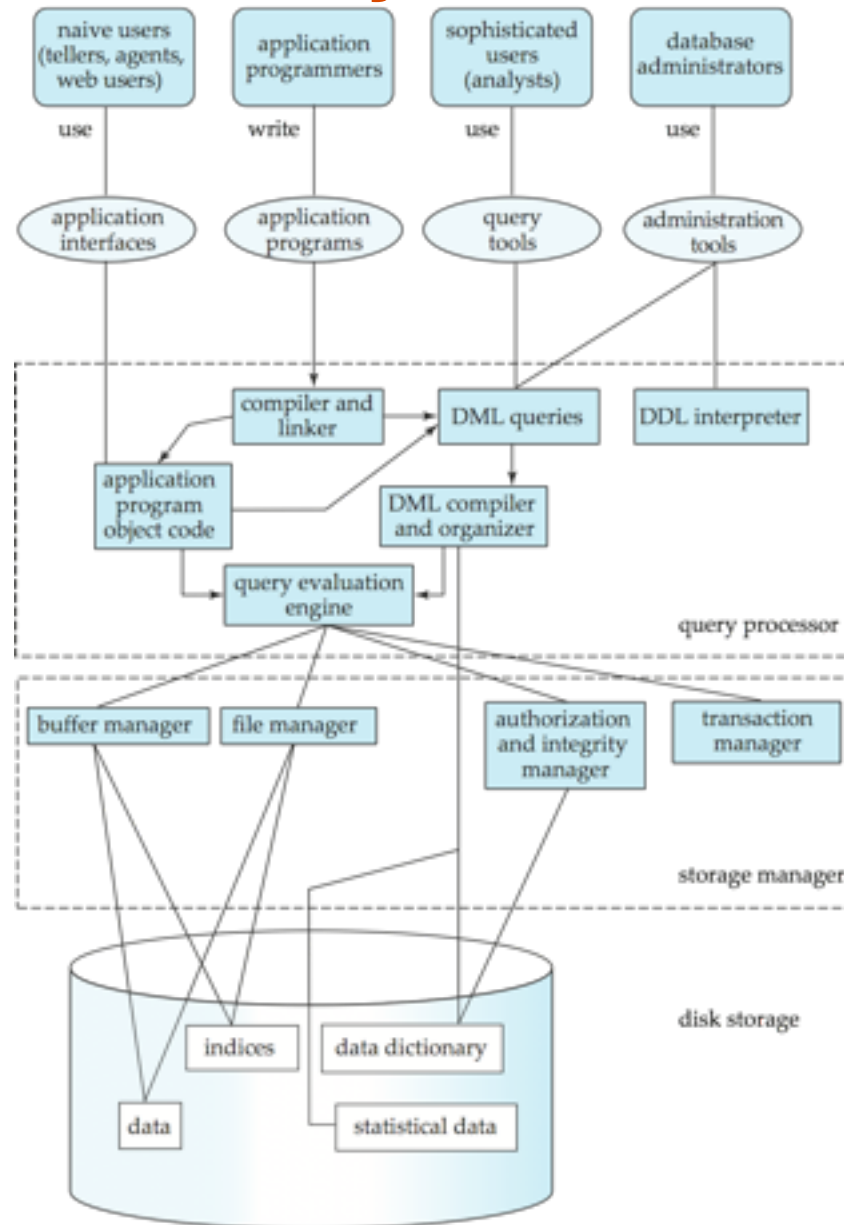
# Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed



# Database System Internals





# History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - ▶ Tapes provide only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allow 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 begins Ingres prototype
  - High-performance (for the era) transaction processing



# History (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, ..



# Chapter 1: Introduction

- Purpose of Database Systems
- View of Data
- Database Languages
- Relational Databases
- Database Design
- Data Storage and Querying
- Transaction Management
- Database Users and Administrators
- Database Architecture
- Overall Structure
- History of Database Systems



# End of Chapter 1