Chapter 1: Introduction

Revision by Gun-Woo Kim

Dept. of Computer Science and Engineering

Hanyang University

Database System Concepts, 6th Ed.

Contents

1.1 Database Management System (DBMS)

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 Applications:

- Banking: 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 can be very large.

Databases touch all aspects of our lives

Database Example - University

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

In the early days, database applications were built directly on top of file systems

1.2 Purpose of Database Systems

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation — multiple files and formats

이전에는 각각의 프로그램에서 쓰는 파일 규격이 달랐기때문

- 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
- Atomicity of updates

현재의 DBMS에서 정전등으로 꺼지고 백업안되있는데 다른 서버에는 이미 업데이트가 되는 경우같은

  - Failures may leave database in an inconsistent state with partial updates carried out
- Concurrent access by multiple users
- Security problems

Database systems offer solutions to all the above problems

1.3 View of Data

Physical level: describes how a record (e.g., customer) is stored. 컴파일러에 해당

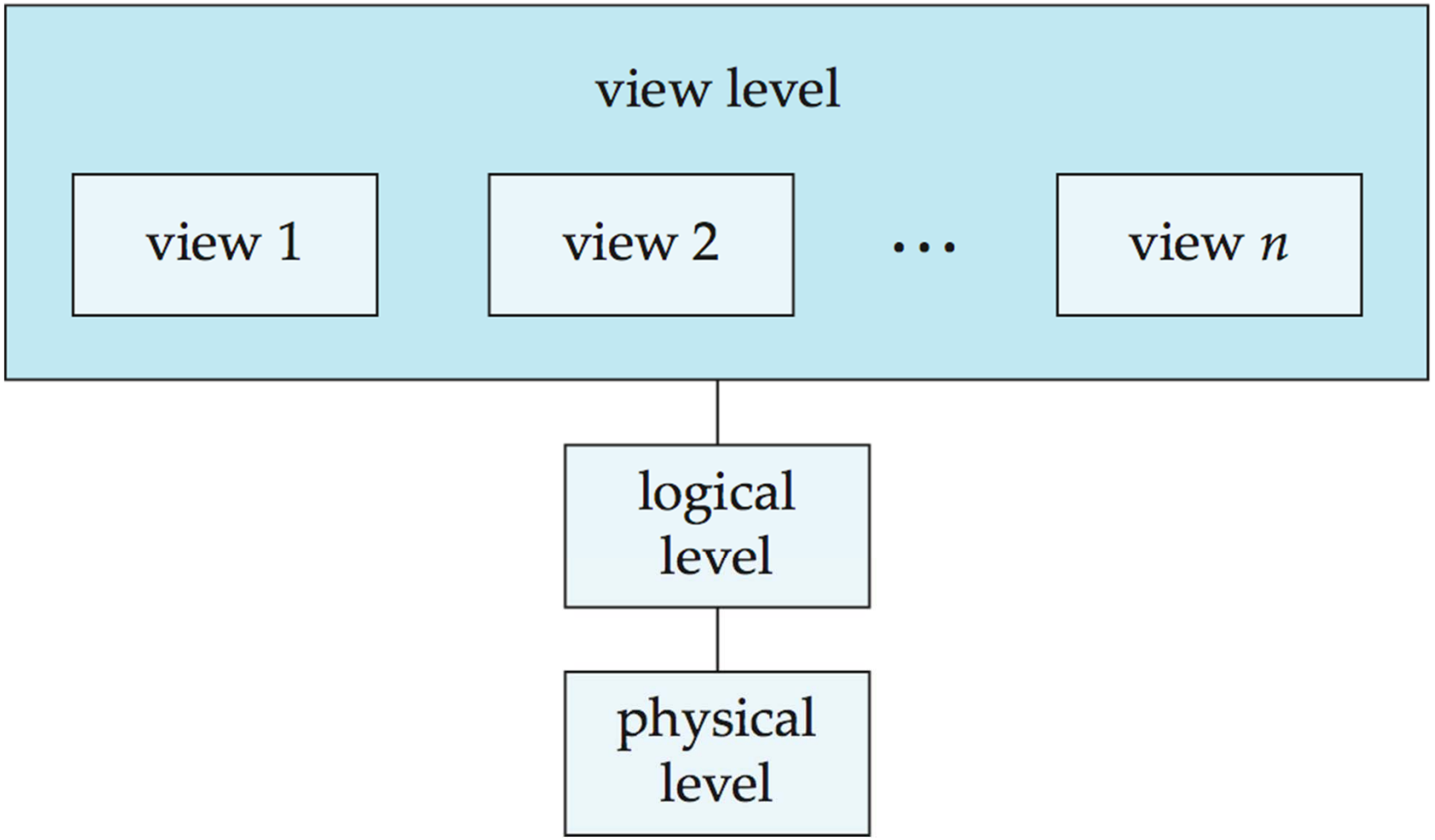
Logical level: describes data stored in database, and the relationships among the data.

```
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.

Levels of Abstraction

An architecture for a database system



Instances and Schemas

Similar to types and variables in programming languages

Schema – the logical structure of the database

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

Ability to modify the physical schema without changing the logical schema

크게 다루지는 않는데, 물리적 **schema**가 바뀌어도논리는 안봐뀜

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

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

1.4 Database Language

Generally, database systems provide two database languages

- Data Definition Language (DDL)
  - Specify the database schema
- Data Manipulation Language (DML)
  - Express database queries and updates

DDL and DML are not two separate language

- They form parts of a single database language, such as the widely used SQL language

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)
- Referential integrity (**references** constraint in SQL)
  - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation

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

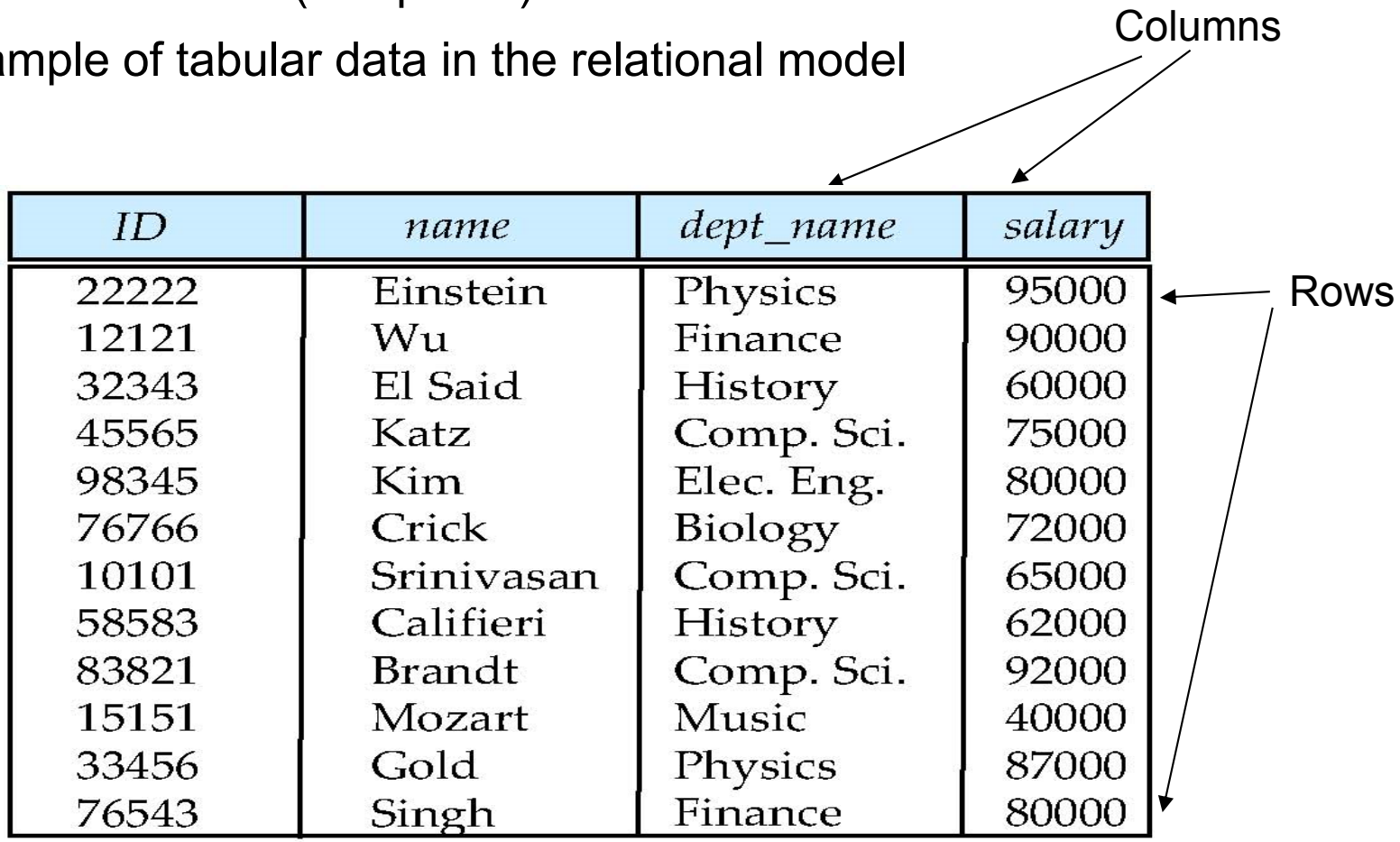
- **Procedural** – user specifies what data is required and how to get those data
- **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data 특정한것만 가져옴

SQL is the most widely used query language

1.5 Relational Model

Relational model (Chapter 2)

Example of tabular data in the relational model



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

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

SQL

SQL: widely used non-procedural language

SQL DML

- Example: Find the name of the instructor with ID 22222

```
select name
from   instructor
where instructor.ID = '22222'
```

SQL DDL

- Example: Define the *department* table.

```
create table department
        (dept_name      char (20),
         building       char (15),
         budget         numeric (12,2));
```

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

1.6 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 (cont.)

Is there any problem with this design?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
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

중복된 데이터가 많으면 비효율적

Design Approaches

Normalization Theory (Chapter 8)

Formalize what designs are bad, and test for them

Entity Relationship Model (Chapter 7)

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:*

The Entity-Relationship Model

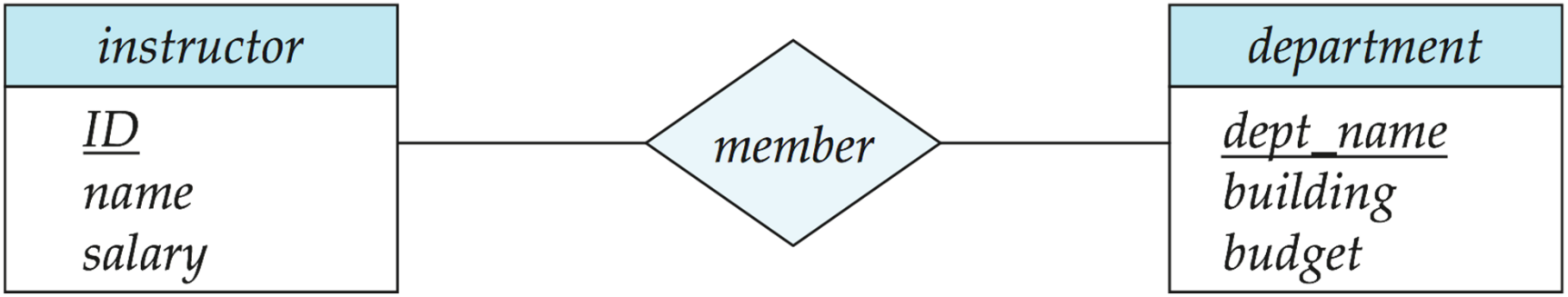
Models an enterprise as a collection of *entities* and *relationships*

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:*



What happened to dept_name of instructor and student?

Object-Relational Data Models

Relational model: flat, “atomic” values

Object Relational Data Models

- Extend the relational data model by including object orientation and constructs to deal with added data types.
- Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
- Provide upward compatibility with existing relational languages.

1.7 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 to the following tasks:

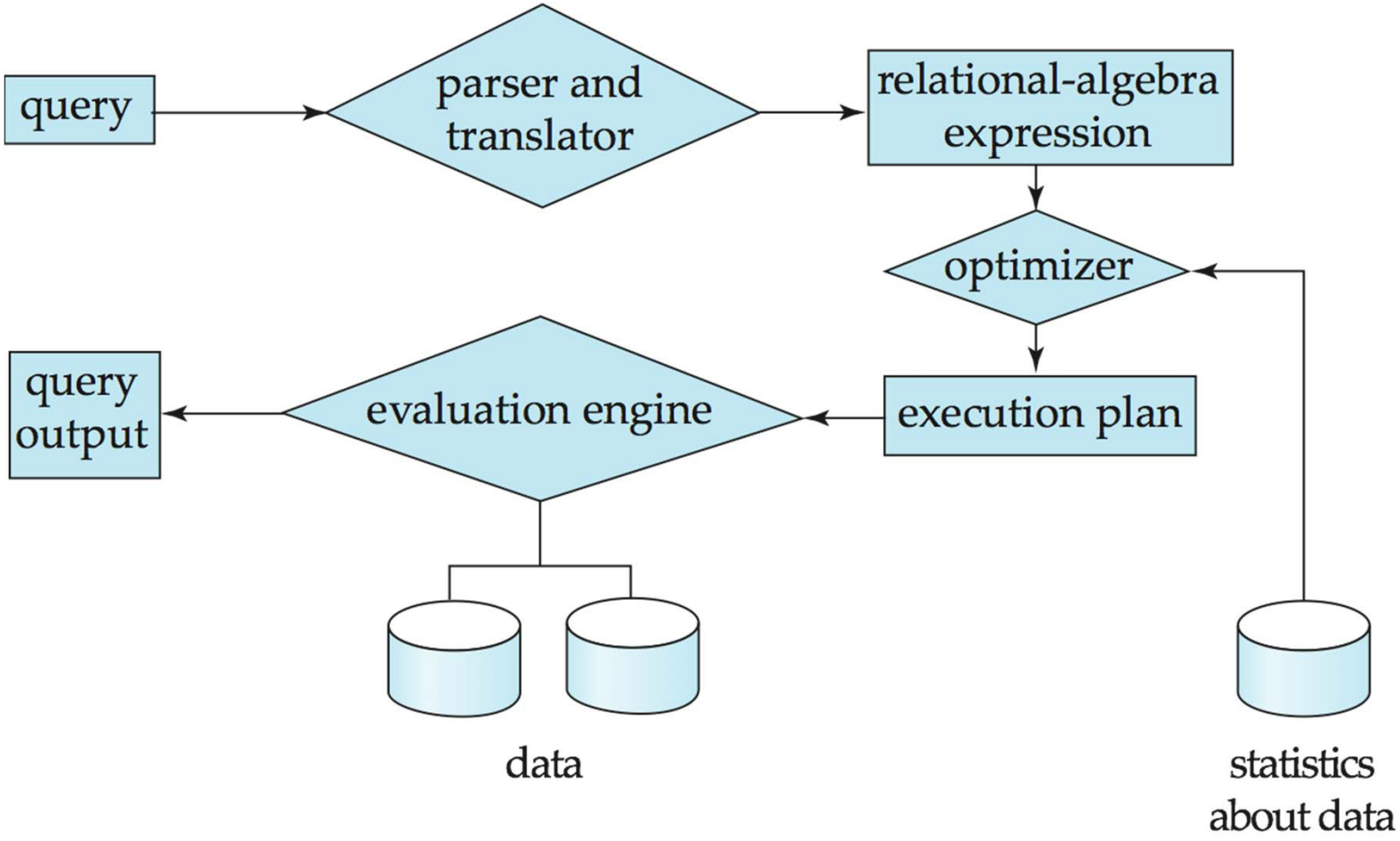
- Interaction with the file manager
- Efficient storing, retrieving and updating of data

Issues:

- Storage access
- File organization
- Indexing and hashing

1.8 Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

- 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

1.9 Transaction Management

What if the system fails?

What if more than one user is concurrently updating the same data?

Transaction

A collection of operations that performs a single logical function in a database application

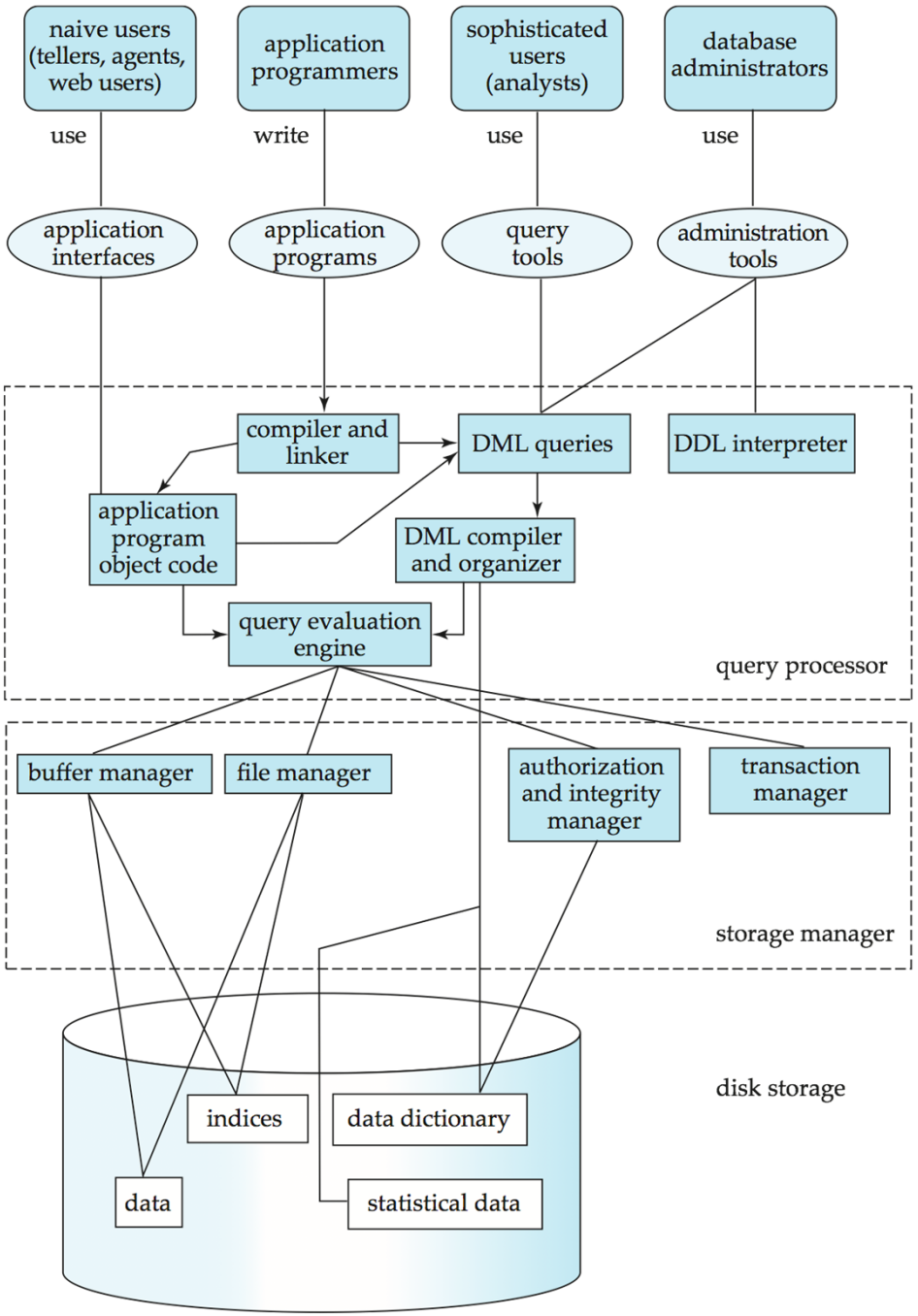
Transaction-management component ensures that

The database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

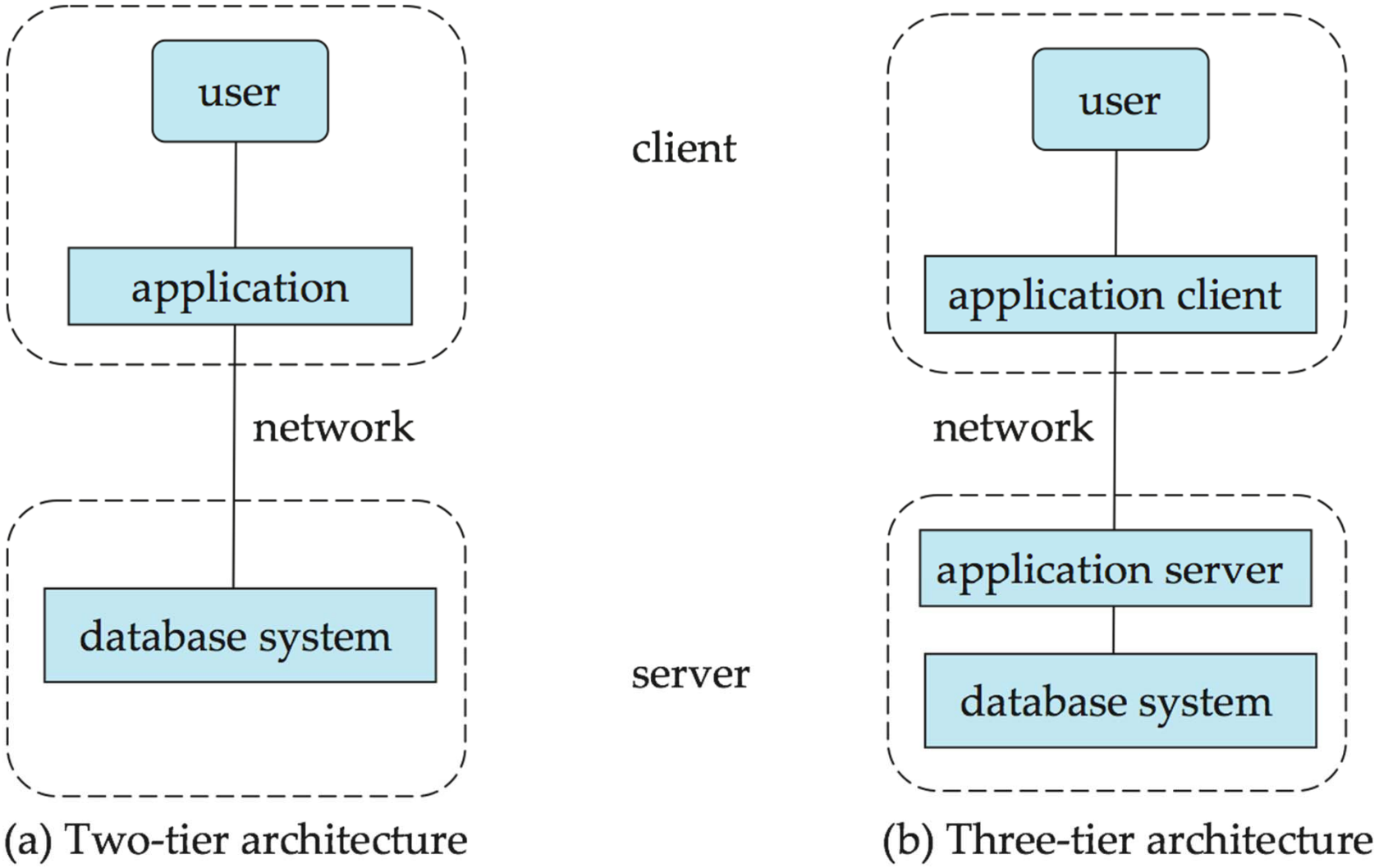
Concurrency-control manager

A manager that controls the interaction among the concurrent transactions, to ensure the consistency of the database.

1.10 Database Architecture



Database Architecture (cont.)



(a) Two-tier architecture

(b) Three-tier architecture

사용자가 직접적으로 데이터베이스에 엑세스

다수의 사용자가 데이터 베이스에 액세스

큰다른점은 application server의 유무

1.11 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 begins Ingres prototype
- 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
- 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, ..