



Databases – Introduction (Chapter 1)

Jaeyong Choi
Dept. of AI-Software, Gachon University





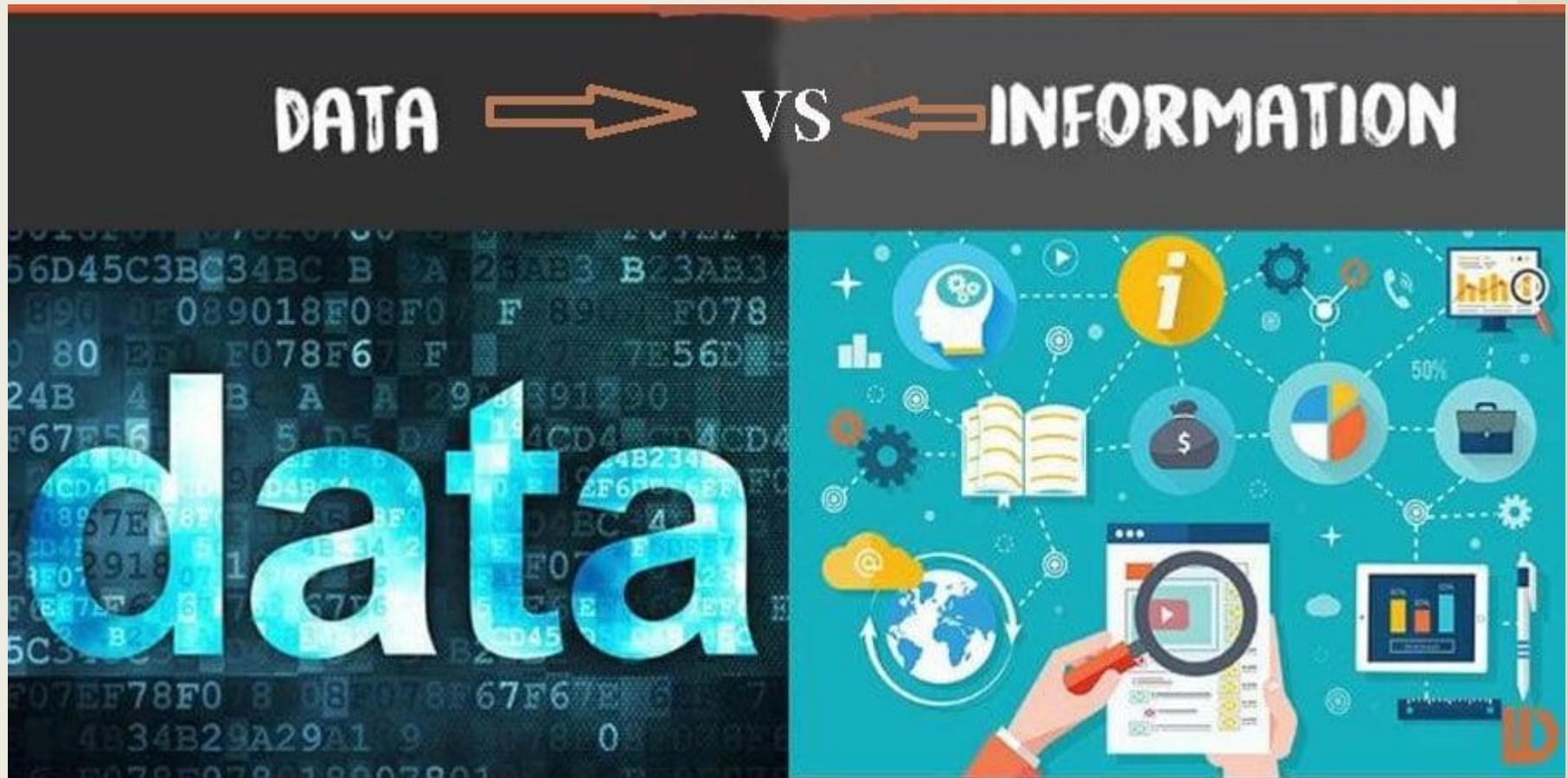
Outline



- ❑ Need for Databases
- ❑ Data Models
- ❑ Relational Databases
- ❑ Database Design
- ❑ Storage Manager
- ❑ Query Processing
- ❑ Transaction Manager



Data vs. Information





Database



❑ What is a database ?

- ❑ Collection of inter-related data
- ❑ *Fact* and *Value* that can be recorded
- ❑ Having implicit meaning
- ❑ Designed, built, and populated with data for a specific purpose

❑ Databases can be very large

- ❑ Defining structure for storage of information
- ❑ Manipulation methods for information

❑ Databases touch all aspects of our lives



Database Management System (DBMS)

❑ Computerized system

- ❑ Create and maintain a database
- ❑ General-purpose software system

❑ DBMS contains information about a particular organization

- ❑ Collection of inter-related data → Database
- ❑ Set of programs to access the data
- ❑ Providing a way to store and retrieve databases
- ❑ An environment that is *convenient* and *efficient* to use
- ❑ *Protecting* and *maintaining* over a long period of time



Database System

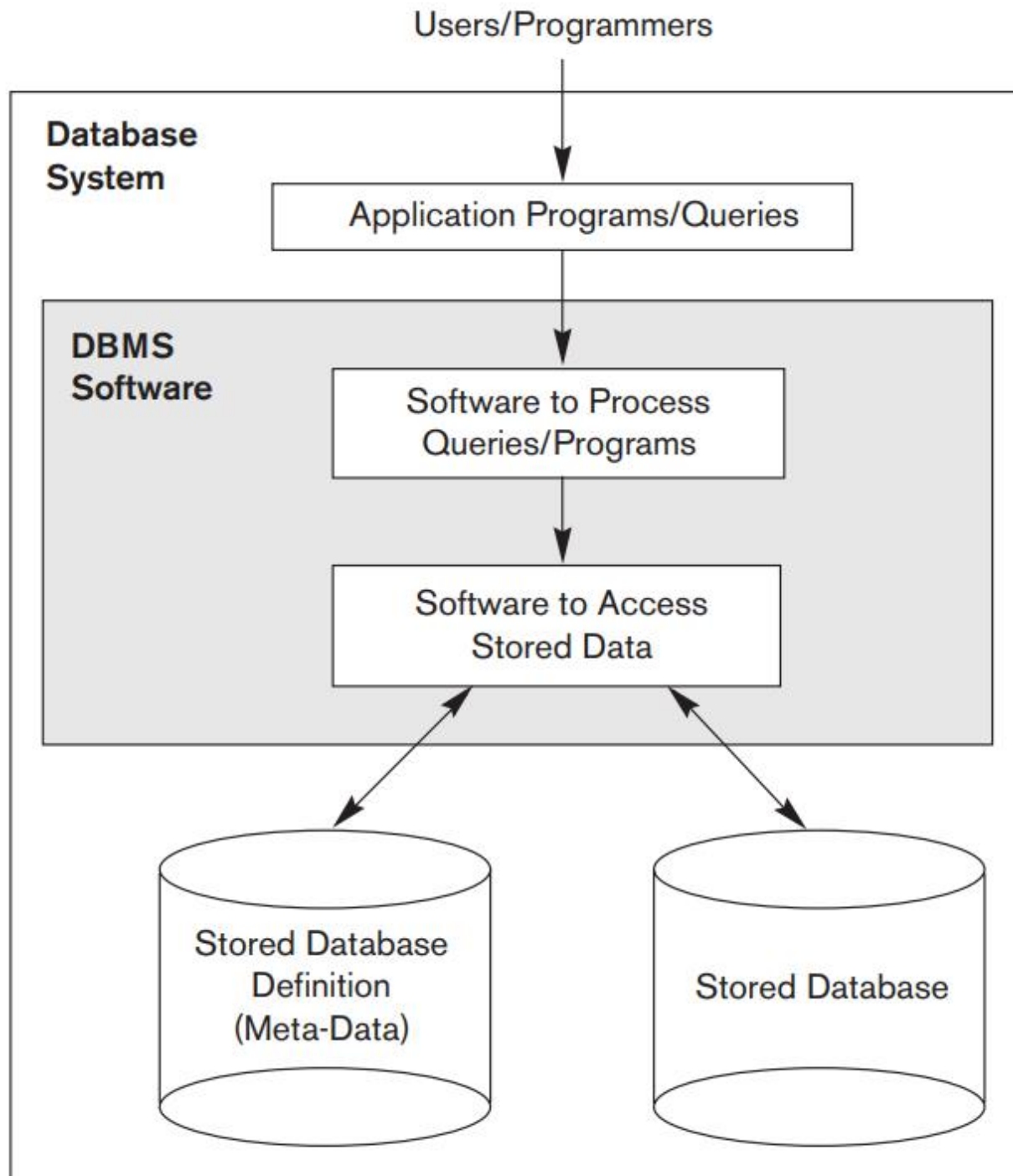


❑ What is Database System ?

- ❑ Developed in 1960s for the computerized management of commercial data
- ❑ Database + DBMS

❑ Database systems are used to manage data :

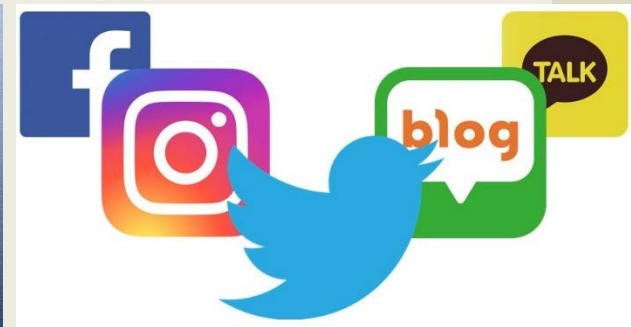
- ❑ Highly valuable
- ❑ Relatively large
- ❑ Accessed by multiple users and application (same time)





❑ Database applications:

- ❑ Banking: accounts, transactions
- ❑ Airlines: reservations, schedules
- ❑ Universities: students, courses, registration, grades
- ❑ Sales: customers, products, purchases
- ❑ Manufacturing: production, inventory, orders
- ❑ Human resources: employee records, salaries
- ❑ Social-media: connections btw users, posts, like info



Two modes of Database to support

❑ Online Transaction Processing

- ▣ Performing small updates in large database

❑ Data Analysis

- ▣ Processing of data to draw conclusion





Real World Example (1/2)

Meta : Extorting personal data Issue



Facebook 앱을 계속 사용하려면 업데이트를 검토하고 동의하세요

다음은 Meta에서 서비스를 제공하기 위해 회원님의 정보를 이용하는 몇 가지 주된 방법입니다. Meta가 서비스를 계속 제공하려면 회원님이 각 항목을 검토하고 동의하셔야 합니다.

지금 업데이트에 동의해야 하는 것은 아니나, 2022년 7월 26일 이후에는 업데이트에 동의해야 계정을 사용할 수 있습니다.

개인정보의 수집 및 이용[필수]



Meta Platforms, Inc.가 서비스 제공 및 맞춤화, 분석, 안전 및 보안, 맞춤형 광고 표시를 위한 개인정보 수집 및 이용에 동의합니다. [더 알아보기](#)



Instagram 앱을 계속 사용하려면 업데이트를 검토하고 동의하세요

다음은 Meta에서 서비스를 제공하기 위해 회원님의 정보를 이용하는 몇 가지 주된 방법입니다. Meta가 서비스를 계속 제공하려면 회원님이 각 항목을 검토하고 동의하셔야 합니다.

지금 업데이트에 동의해야 하는 것은 아니나, 2022년 7월 26일 이후에는 업데이트에 동의해야 계정을 사용할 수 있습니다.

개인정보의 수집 및 이용[필수]



Meta Platforms, Inc.가 서비스 제공 및 맞춤화, 분석, 안전 및 보안, 맞춤형 광고 표시를 위한 개인정보 수집 및

동의함

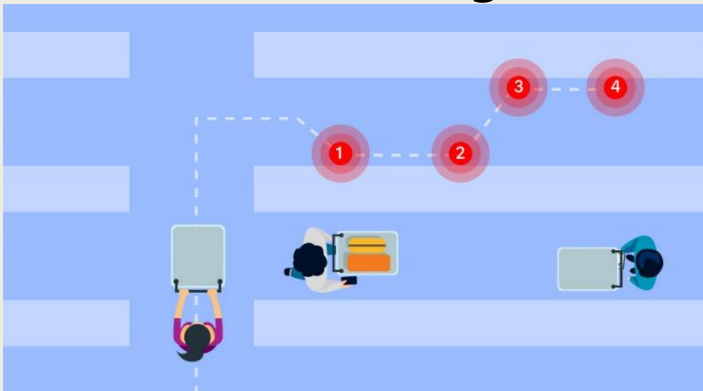
동의를 누름으로써 회원님은 계정을 계속 사용하기 위해 위 정보의 수집, 이용, 제공 및 방침과 약관에 동의함을 확인합니다. 동의하지 않는다면 [계정 옵션을 확인하세요](#).



Real World Example (2/2)

❑ Coupang : Rocket Delivery

❑ Warehouse Management System



❑ Random Stow





University Database



❑ 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, the applications were built directly on top of [File-Processing Systems](#)



Drawbacks of File Systems (1/2)

❑ Data redundancy and inconsistency

- ❑ Duplication of information in different files
- ❑ Multiple file formats

data redundancy (데이터 중복): 데이터가 여러 파일에 나타나는 것으로 이로 인해 자료의 분산으로 일관된 자료 처리가 어렵고 데이터 저장의 낭비 초래

❑ Difficulty in accessing data

- ❑ Need to write a new program to carry out each new task

❑ Data isolation

- ❑ Multiple files and formats

data isolation (데이터 고립): typically defined at database level as a property that defines how/when the changes made by one operation become visible to others

❑ Integrity problems

- ❑ Consistency 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

data integrity (데이터 무결성): 데이터의 정확성과 일관성을 유지하고 보증하는 것을 가리킴.



Drawbacks of File Systems (2/2)



Atomicity(원자성): 시스템의 어떤 상황 하에서도 한 트랜잭션에 대한 모든 연산들의 결과가 데이터 베이스에 모두 반영되든가 아니면 전혀 반응되지 않아야 함을 의미하는 성질.

❑ Atomicity of updates

- ❑ Failures may leave database in an inconsistent state with partial updates carried out

❑ Concurrent access by multiple users

- ❑ Concurrent access needed for performance
- ❑ Uncontrolled concurrent accesses can lead to inconsistencies

❑ Security problems

- ❑ Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems



Data Abstraction and Architecture

❑ *Data Abstraction*

- Hiding the complexity from users through several levels



❑ *3-Level Schema Architecture*

- Breaks the databases down into three different categories
- Used to separate the user applications and physical database
- Attributed to the ANSI/SPARC group



Schemas and Instances (1/2)




❑ *Schema*

- ❑ Overall design and constraints of the database

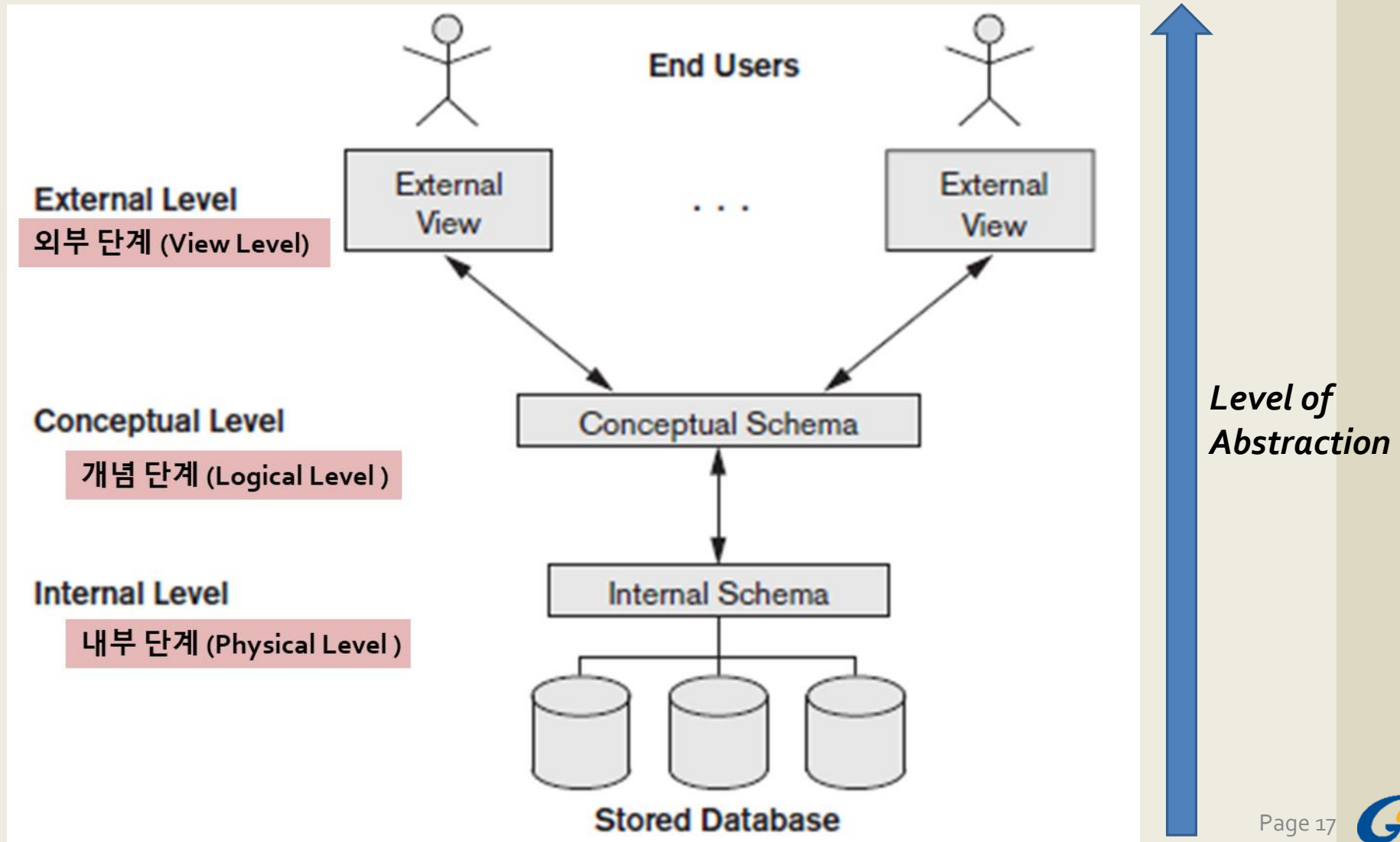
❑ *Instance*

- ❑ Actual content of the database at a particular time
- ❑ Analogous to the value of a variable

Customer				
	ID Number	Name	Age	Address
	INT	CHAR(10)	INT	CHAR(20)



3-Level Schema Architecture





Schemas and Instances (2/2)

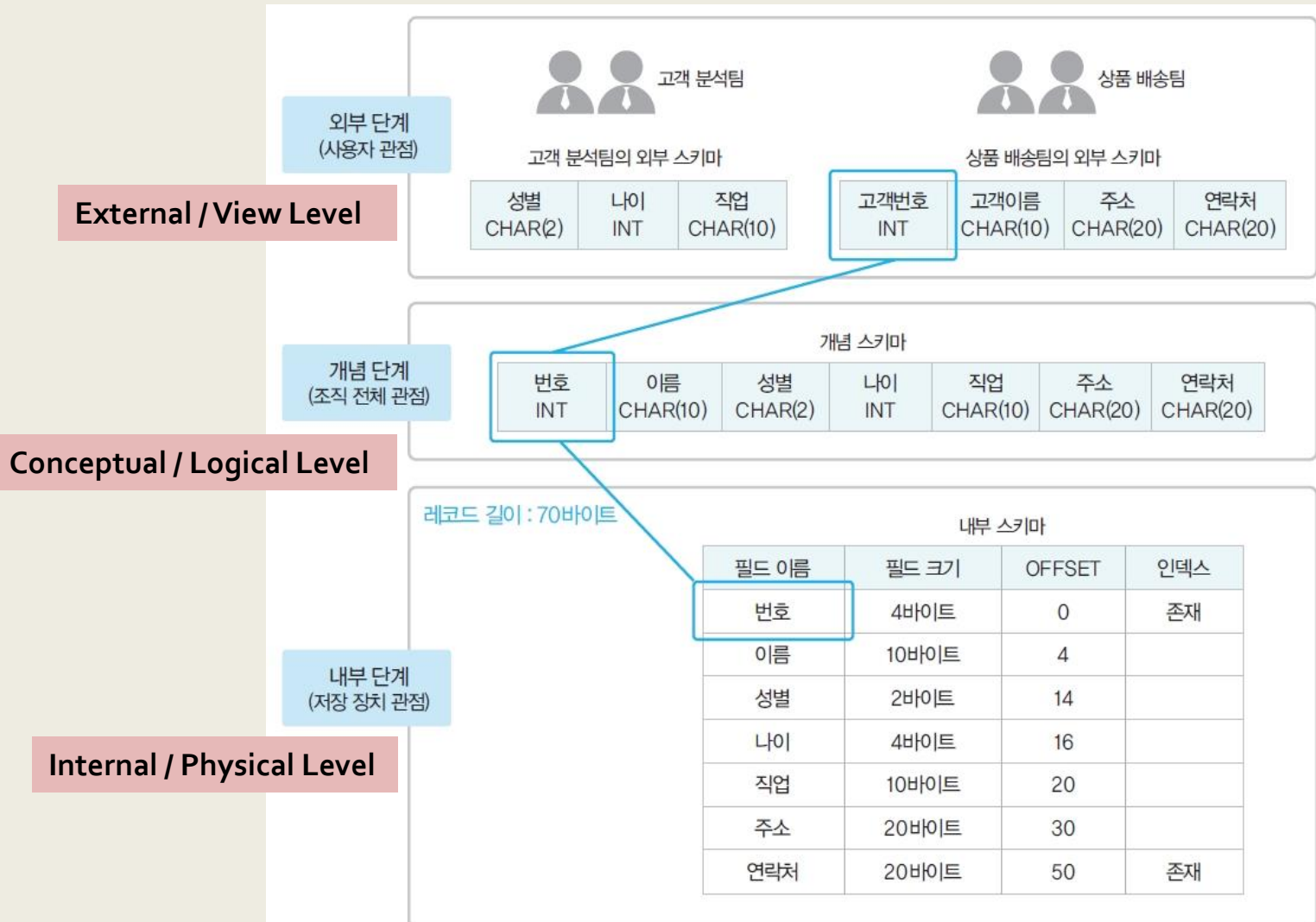


- ❑ ***Logical schema (Conceptual schema)***
 - ❑ Overall logical structure of the database
 - ❑ E.g., a database consists of information about customers, bank accounts and their relationship
 - ❑ Analogous to type information of a variable in a program

- ❑ ***Physical schema (Internal schema)***
 - ❑ Overall physical structure of the database

3-Level Schema Architecture Example

❑ E-commerce





Data Independence

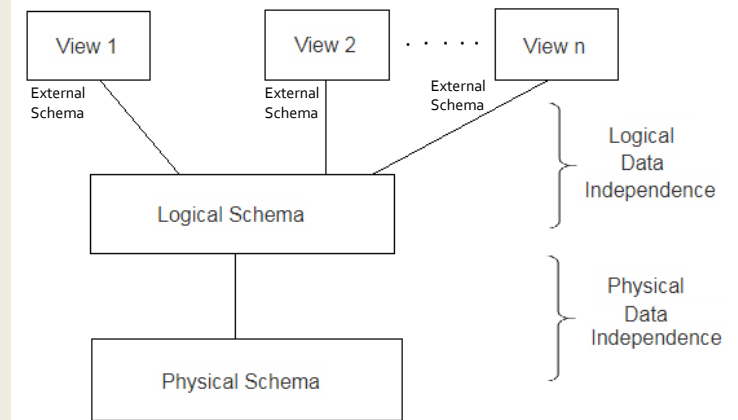


❑ *Logical data independence*

- Capacity to change *logical schema* without having to change *external schemas* or application programs

❑ *Physical data independence*

- 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 Models



- ❑ **Data model – a collection of tools for describing:**
 - ❑ Data, data relationships, data semantics, data constraints

- ❑ **Data models**
 - ❑ Relational Model
 - ❑ Entity-Relationship Model (mainly for database design)
 - ❑ Object-Based Data Model: object-oriented and object-relational models
 - ❑ Semi-Structured Data Model: XML, JSON

- ❑ **Other older models**
 - ❑ Hierarchical model
 - ❑ Network model



Preview of Relational Model

□ Relational Model

- All the data is stored in various *tables* (or *relations*)
- Example of tabular data in the relational model

<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

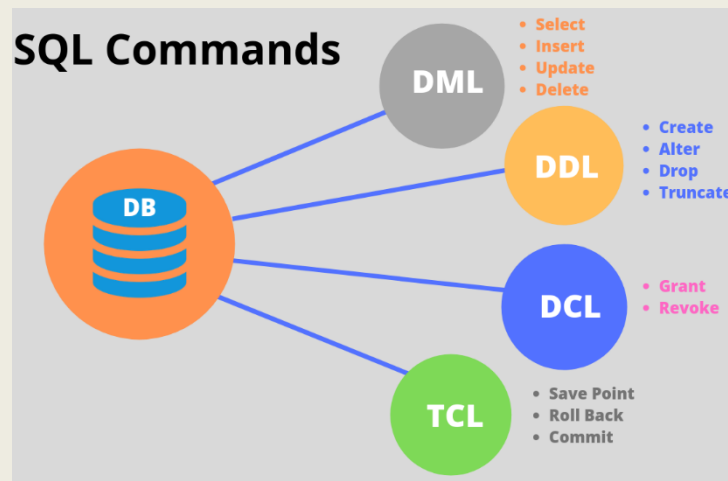
row

column

Structured Query Language (SQL)

SQL

- A special-purpose programming language designed for *defining* and *managing* data in a **DBMS**
- Most widely used commercial language
- Application programs generally access databases through:
 - Language extensions to allow embedded SQL *or*
 - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database





Data Definition Language (DDL)

□ DDL

- Specification notation for defining the database schema
- DDL compiler generates a set of table templates stored in a *data dictionary*

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

□ Data dictionary contains metadata

- Database schema
- Integrity constraints: primary key
- Authorization: who can access what data

Data Manipulation Language (DML)

□ DML

- Language for accessing and manipulating the data
 - DML also known as *query language*
- *Procedural* language
 - Specifies *what data* are needed with specifying how to get the data
- *Declarative (non-procedural)* language
 - Specifies *what data* is retrieved without specifying how to get

Data Manipulation Language (DML)

```
select instructor.ID, department.dept_name  
from instructor, department  
where instructor.dept_name = department.dept_name and  
       department.budget > 95000;
```

<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

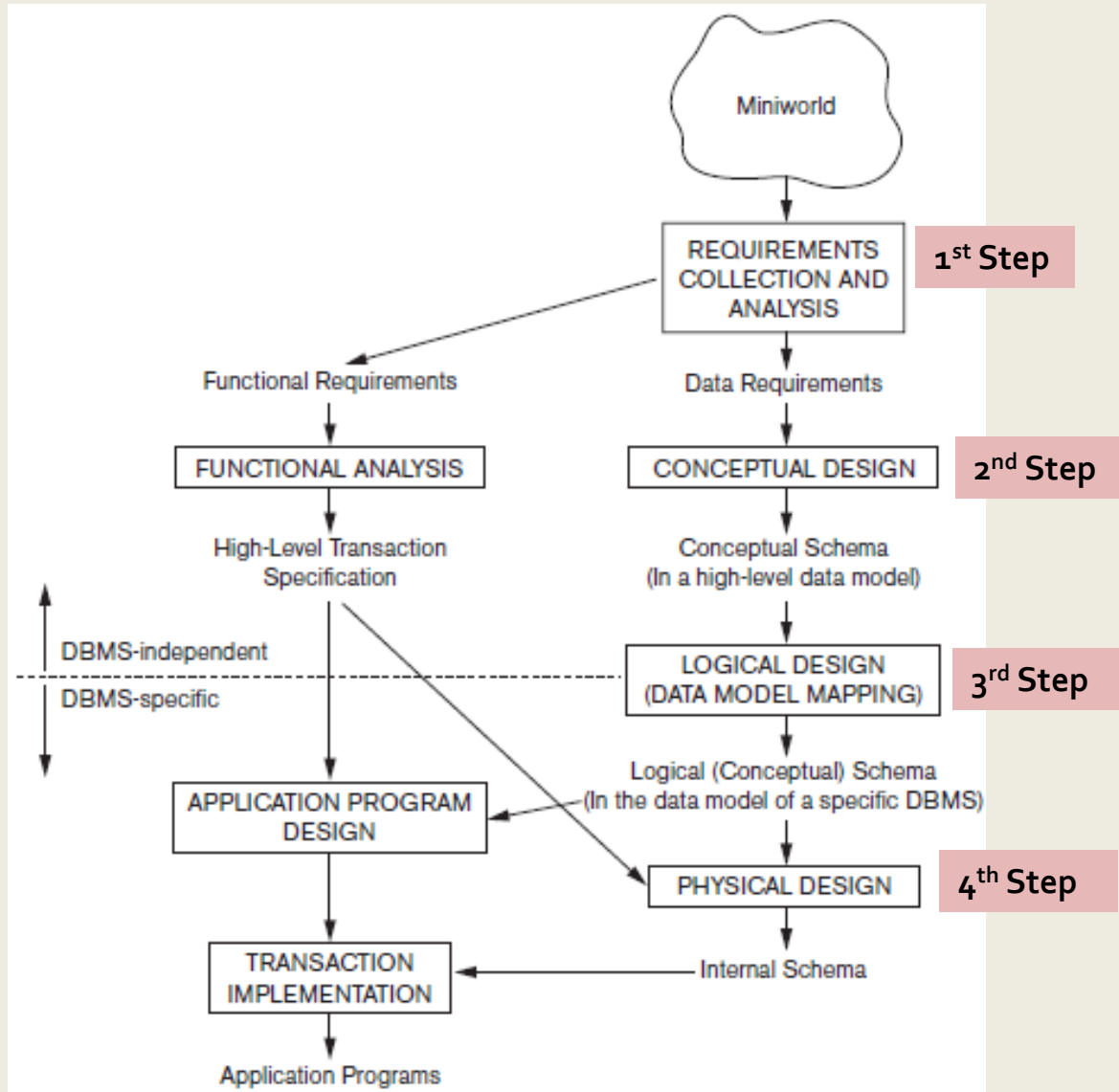
Data Manipulation Language (DML)

```
select instructor.ID, department.dept_name  
from instructor, department  
where instructor.dept_name= department.dept_name and  
       department.budget > 95000;
```

<i>ID</i>	<i>dept_name</i>
12121	Finance
45565	Comp. Sci.
10101	Comp. Sci.
83821	Comp. Sci.
76543	Finance



Database Design (1/2)





Database Design (2/2)



❑ Logical design

- ❑ Deciding on the database schema; 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 them?
- ❑ Entity-Relationship model (Chapter 6)
 - ❑ Models an organization as a collection of entities and their relationships
 - ❑ Represented diagrammatically by an entity-relationship diagram
- ❑ Normalization (Chapter 7)
 - ❑ Formalize what designs are bad, and test for them

❑ Physical design

- ❑ Deciding on the physical layout of the database



Entity-Relationship Model

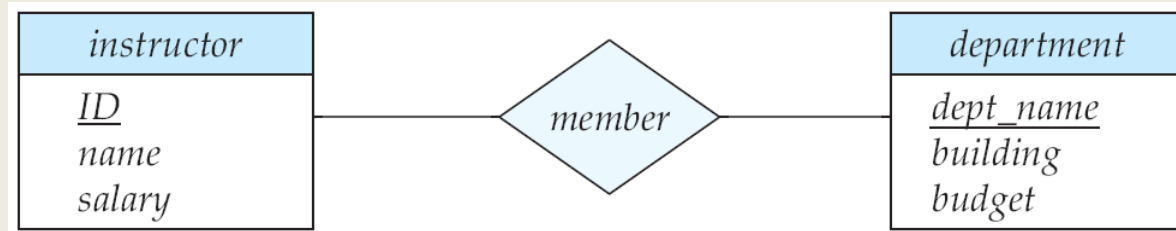


Entity

- A “thing” or “object” in the organization that is distinguishable from others
- Described by a set of attributes

Relationship

- An association among several entities





Object-Relational Data Model



❑ OR data model

- A Object oriented DB model + a Relational DB model
 - Supports objects, classes, inheritance etc
 - Supports for data types, tabular structures etc.
- **Extends the relational data model** by including object orientation and constructs to deal with added data types
- **Allows attributes of tuples to have complex types**, including non-atomic values such as nested relations
- **Preserves relational features**, in particular the declarative access to data, while extending modeling power
- **Provides upward compatibility** with existing relational languages

Extensible Markup Language (XML)

□ XML

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language, not a database language
- The ability to **specify new tags** and **create nested tag structures** made XML a great way to exchange data
- XML has become the **basis** for all new generation **data interchange formats**
- A wide variety of tools is available for parsing, browsing, and querying XML documents/data



Storage Management



❑ *Storage manager*

- ❑ A program module that provides the interface between the low-level data and the application programs & queries
- ❑ Responsible for:
 - ❑ Efficient storing, retrieving, and updating of data
 - ❑ Interaction with the OS file manager

❑ **Issues:**

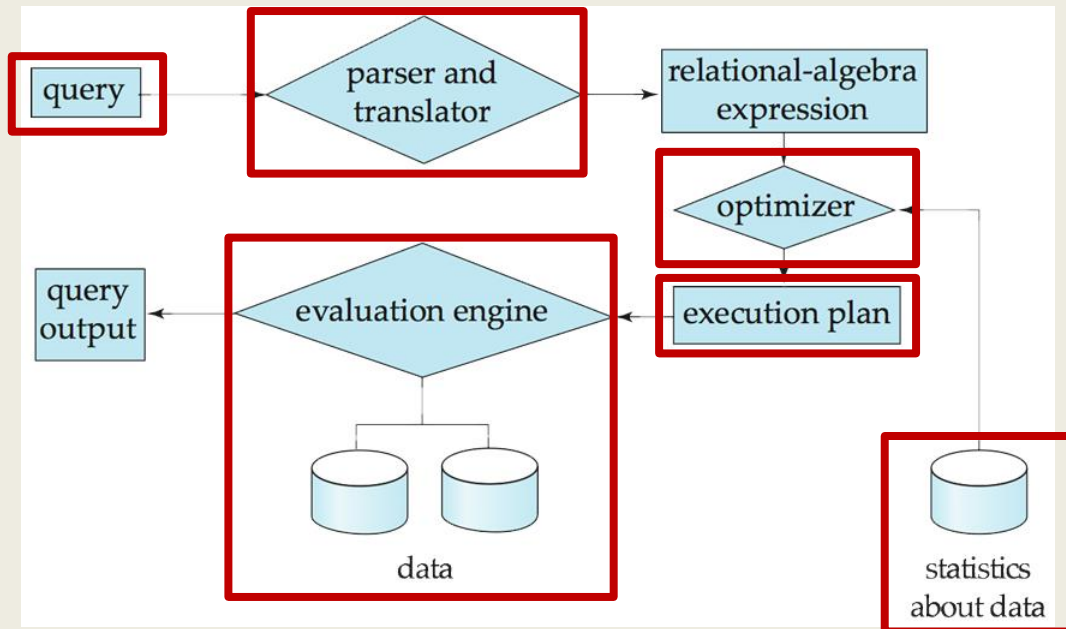
- ❑ Storage access
- ❑ File organization
- ❑ Indexing and hashing



Query Processing

❑ *Query processor*

- ❑ Parsing and translation
- ❑ Optimization
- ❑ Evaluation



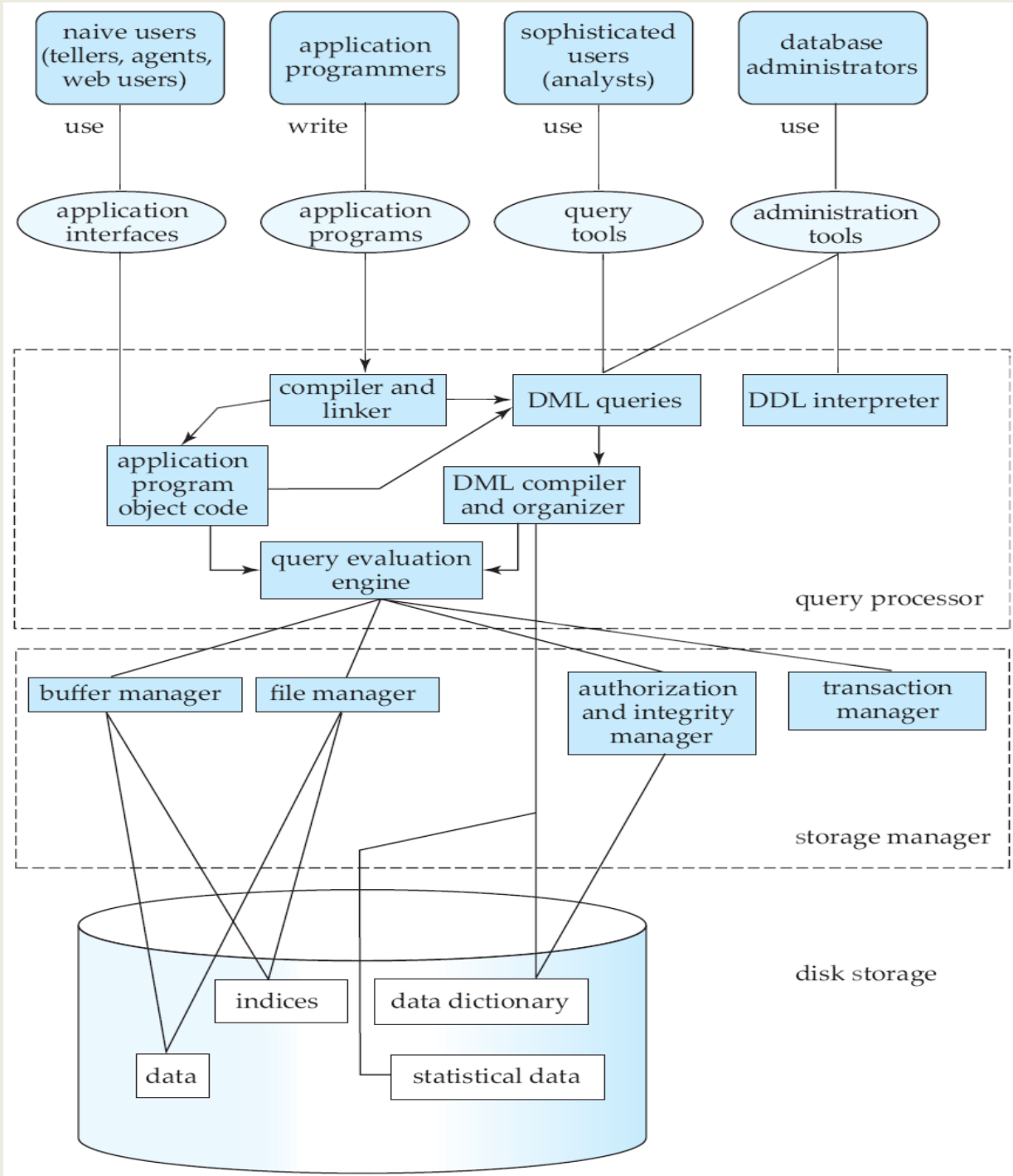


Transaction Management



- ❑ **Consider:**
 - ❑ 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 manager***
 - ❑ 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 System Internals



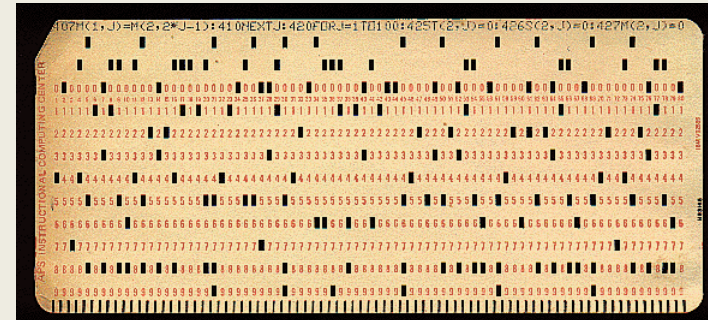


History of Database Systems



❑ 1950s and early 1960s:

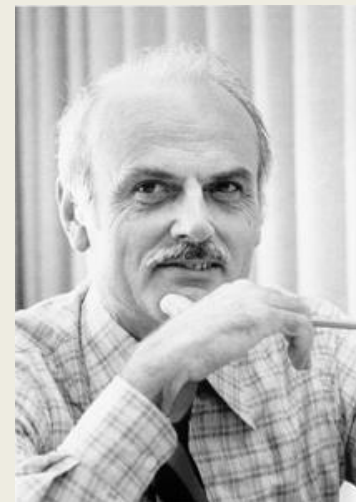
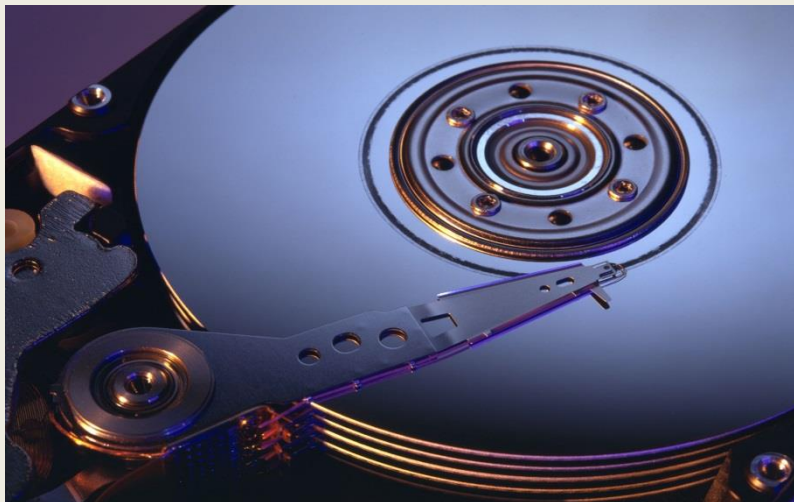
- ❑ Data processing using magnetic tapes for storage
 - ❑ Tapes provided only sequential access
- ❑ Punched cards for input





❑ Late 1960s and 1970s:

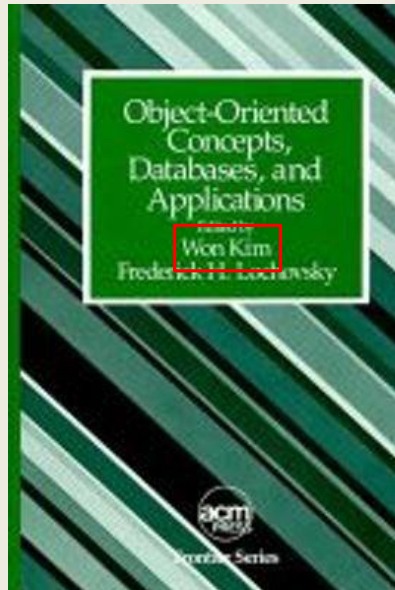
- ❑ Hard disks allowed direct access to data
- ❑ Network and hierarchical data models in widespread use
- ❑ Edgar F. Codd defines the relational data model
 - ❑ Win the ACM Turing Award for this work
 - ❑ IBM Research begins System R prototype





❑ 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



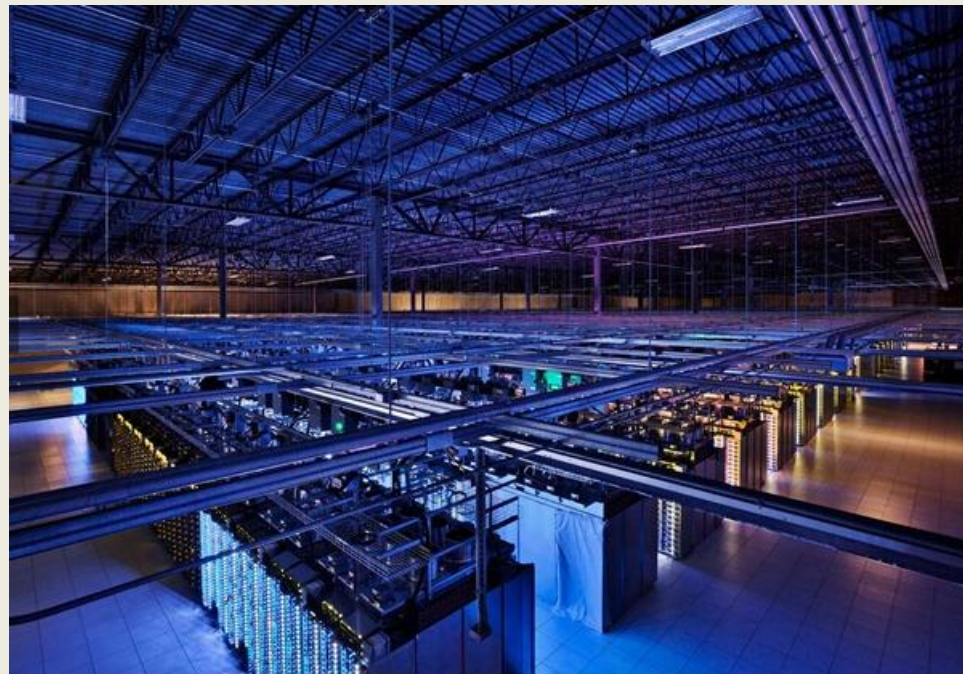
❑ 2000s:

- ❑ XML and JSON standards
- ❑ Automated database administration
- ❑ Giant data storage systems
 - ❑ Google BigTable, Apache HBase, Amazon DynamoDB, etc.



❑ 구글 데이터베이스 센터

- 하루에 200억 페이지 이상을 인덱스
- 30억 개가 넘는 검색어를 실시간으로 처리
- 4억 3천명 정도의 지메일(Gmail) 서비스를 제공





❑ 2000-:

❑ NoSQL

- ❑ Carlo Strozzi came with NoSQL term in 1998.
- ❑ *non-relational* database systems used for storing and retrieving data
- ❑ In today's world, we should not store all the data in table format only which has not predefined fixed schemas(fix no of columns). Like User-generated data, GEO location data, IoT generated data, social graphs are examples of real-world data which has been increasing exponentially.

NoSQL DATABASE TYPES

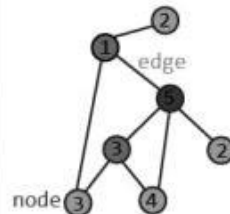
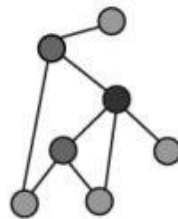
Document



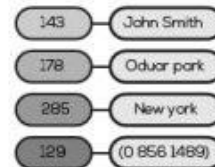
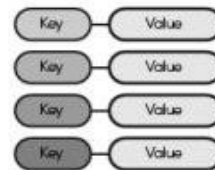
```
{
  "user": {
    "id": "143",
    "name": "improgrammer",
    "city": "New York"
  }
}
```



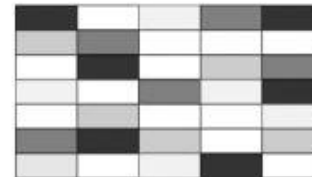
Graph



Key-Value



Wide-Column



1	Fruit	A Foo	B Baz	
2	City	E DC	D PLA	G FLD
3	State	A NZ	C CL	





❑ **2010-:**

❑ Cloud Services

- ❑ Various clients in multiple, widely distributed server farms
- ❑ Data are delivered to users via web-based services

❑ Software as a service

- ❑ Not only stores the data, but also runs/maintains the application software

❑ Issues

- ❑ Responsibility for security and data ownership



Assignment #1



- ❑ Do Practice Exercises (p.32):
 - ❑ 1.4

- ❑ Do Exercises (p.32):
 - ❑ 1.7, 1.8, 1.10, 1.11

- ❑ Due: Before the next lecture (9/7, Wed.), 9:59AM