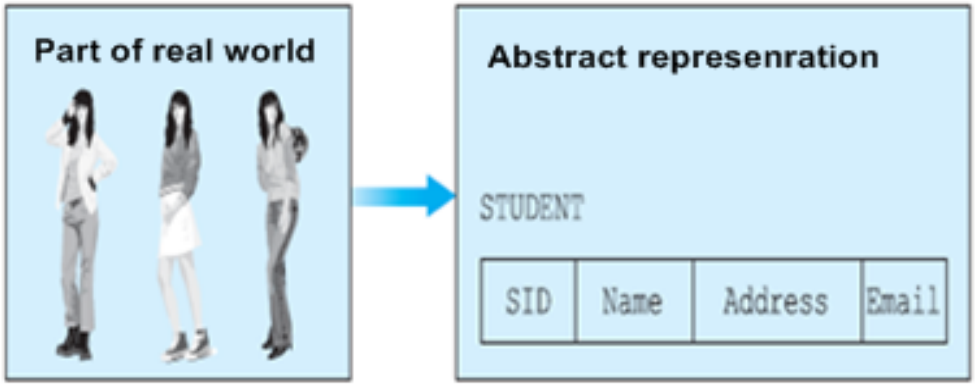
Chapter 2
Database System Concepts and Architectures

Table of Contents

2.1 Data Models, Schemas, and Instances

- **Data model (DB model)**
  - A collection of conceptual tools for describing data, data relationships, data semantics, and data constraints (so that it can provide some level of data abstraction)
  - Most data models also include a set of basic operations for specifying retrievals and updates on the database

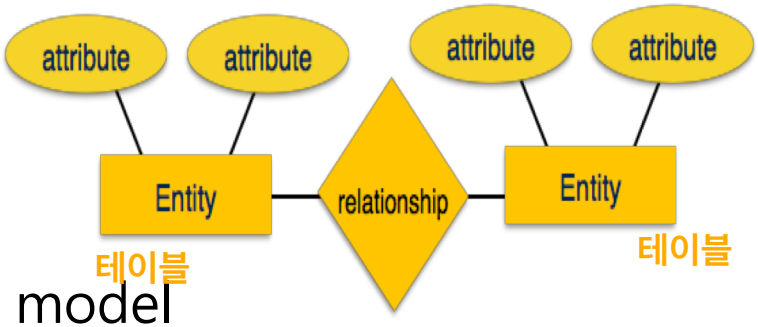


  - 반면, 데이터 모델링이란 실세계의 일부분을 DBMS가 지원하는 데이터 모델의 형태로 나타내는 과정
- **A data model generally consists of three parts**
  - Structure of the data, Operations on the data, Constraints on the data
- **Case of a relational model**
  - Structure of the data: table
  - Operations on the data: insert, delete, update, or retrieve any kind of data
  - Constraints on the data: key constraint, referential integrity constraints, etc

이러한 제약조건은 테이블을 생성할 때 함

2.1.1 Categories of Data Models

- **High-level** (or **conceptual) data models**
  - Provide concepts that are close to the way many users perceive data
  - Independent of DBMS
  - Entity-Relationship data model, Object data model



테이블

테이블

관계는 테이블이 될 수도 있고
안될수도 있음 (몇대몇 관계에 따라)

- **Representational (**or **Implementation) data models**
  - Provide concepts that may be easily understood by end users, but that are not too far from the way data is organized in computer storage
  - Dependent on DBMS
  - Hierarchical data model, Network data model, Relational data model
- **Low-level** (or **physical) data models**
  - Describe the details of how data is stored on storage by representing information such as record formats, record orderings, and access paths
  - Dependent on hardware
  - ISAM(Indexed Sequential Method), VSAM(Virtual Storage Access Method)

IBM이 개발한 인덱스 지원 시스템

정적인덱스 - 인덱스가 바뀔 때 구조가 안바뀜

동적인덱스 - 구조가 바뀜 (비플러스 트리)

2.1.2 Schemas, Instances, and Database State

- **Database schema (or database scheme)**
  - Description of a DB, which is specified during DB design and is not expected to change frequently 구조가 잘 안바뀜
  - Most data models have certain conventions for displaying schemas as diagrams
    - Right figure shows a schema diagram for the DB shown in left figure
    - The diagram displays the structure of each record type but not the actual instances of records

STUDENT

Name	Student_number	Class	Major

스키마

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

하나 : 릴레이션 스키마

인스턴스

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

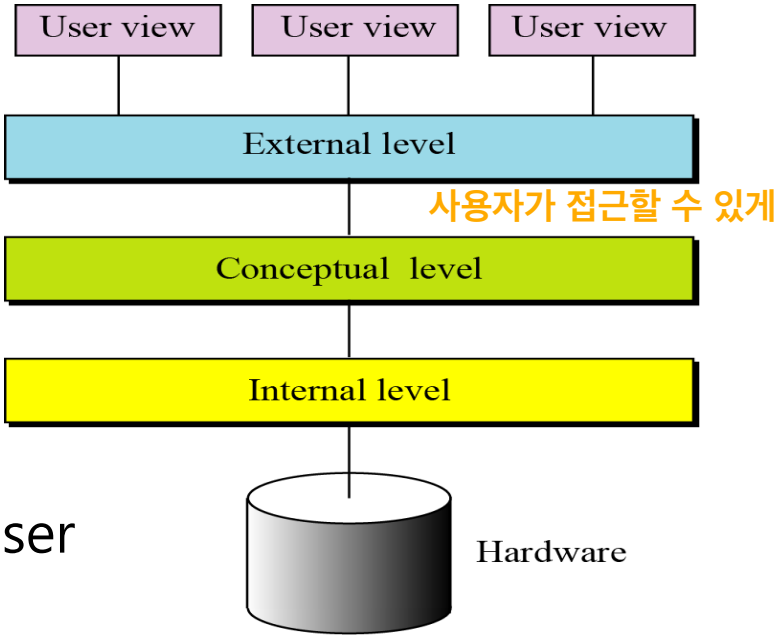
- **Database state (snapshot/occurrences/instances)**
  - Data in DB at a particular moment in time
  - DB state may change frequently as time goes on 골격은 잘 안바뀌고 데이터는 계속해 추가/변경됨
- **The database schema is sometimes called the *intension*, and a database state is called an *extension* of the schema**
- **Schema evolution** 스키마를 변경할 대는 alter table 같은 명령어 사용
  - Changes applied to a schema as the application requirements change

2.2 Three-Schema Architecture and Data Independence

표준화 기구

- **The goal of the ANSI/SPARK three-schema architecture is to separate the user applications from the physical DB**

1) **Internal level :** internal schema
   - Describes how the data are actually stored
   - Physical storage structure of the DB
2) **Conceptual level :** conceptual schema
   - Describes what data are stored in the DB
   - Structure of the whole DB
3) **External** or **view level :** external schema
   - Describes part of the DB that a particular user group is interested in

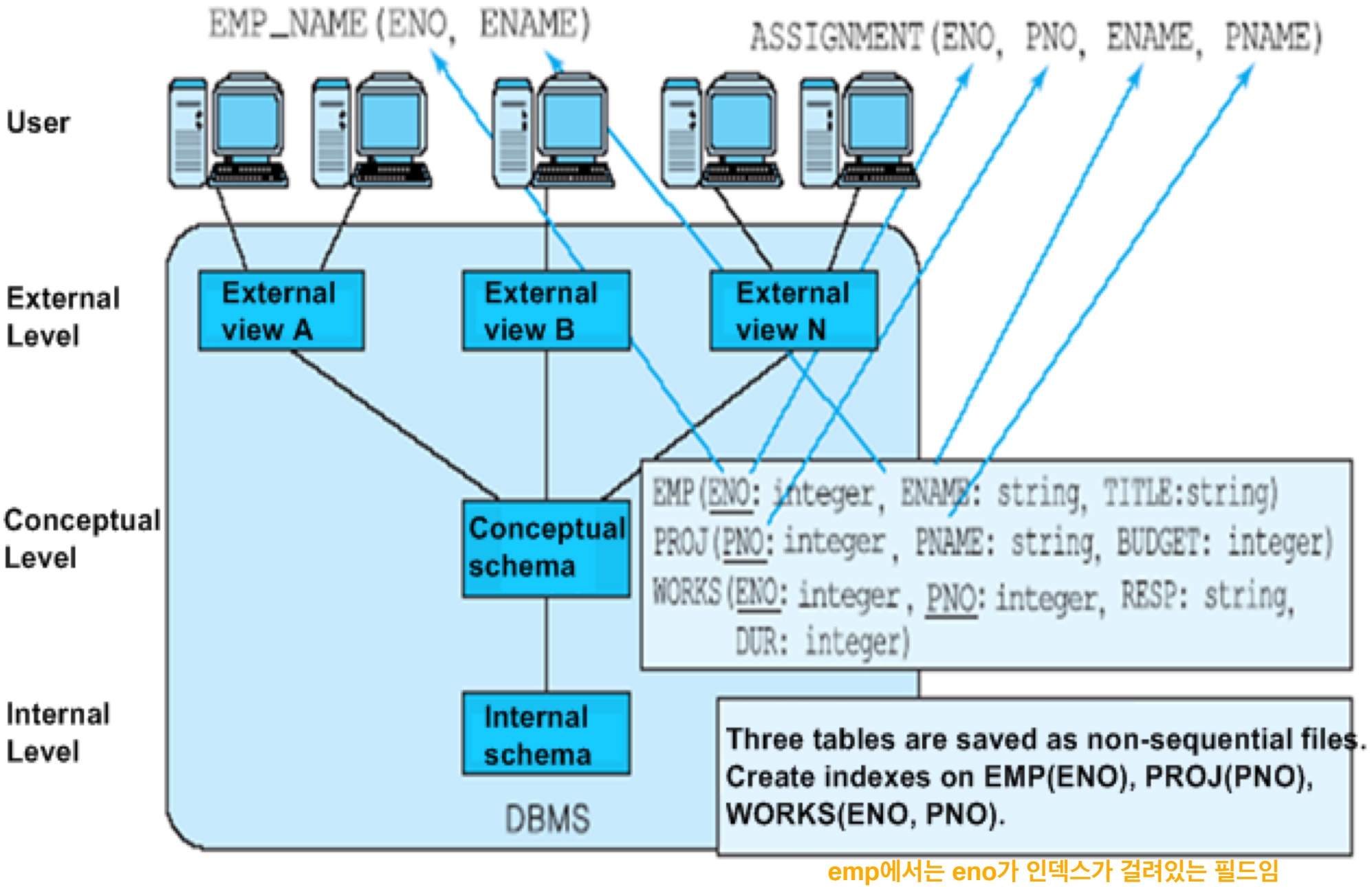


- **Mapping**

세개의 서로다른 레벨로 구분하기 위해서는
- 아래로 내려가면서 맵핑이 필요함

  - DBMS must support the processes of transforming request and results between levels

2.2 Three-Schema Architecture and Data Independence



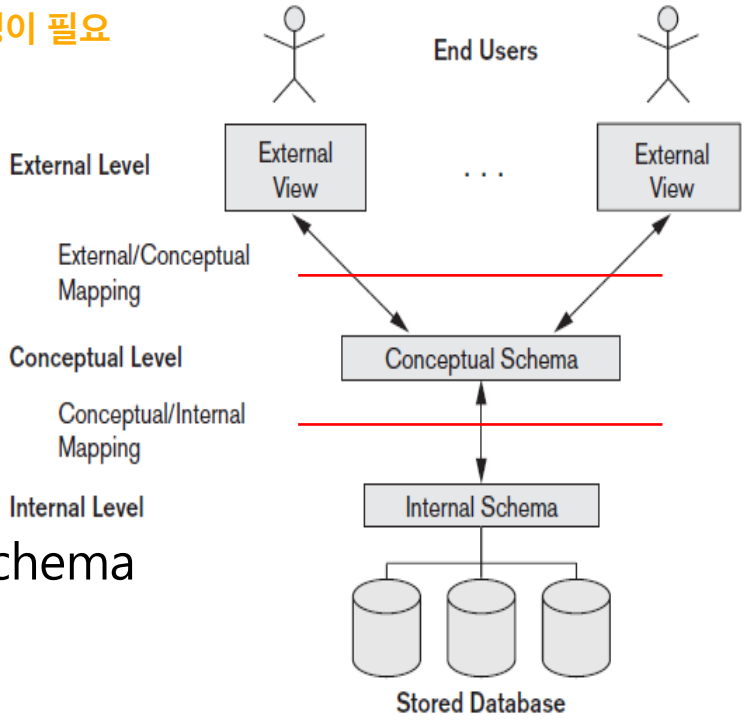
emp에서는 eno가 인덱스가 걸려있는 필드임

2.2 Three-Schema Architecture and Data Independence

- **Data independence** 데이터 독립성 -> 어느 한 레벨이 다른 레벨에 영향 x
  - Occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed 두 가지 맵핑이 필요

1) Logical data independence
   - Capacity to change the conceptual schema without having to change external schemas or application programs

2) Physical data independence
   - Capacity to change the internal schema without having to change the conceptual schema

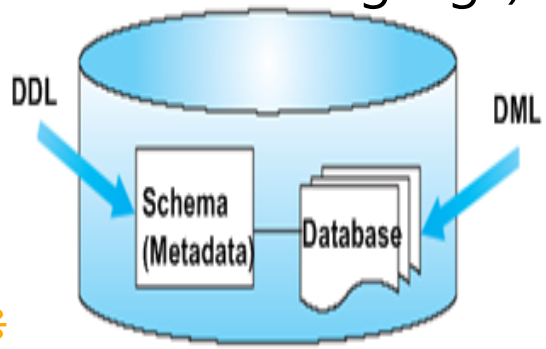


벤더 입장에서는 성능 문제 때문에 충실히 이러한 것을 따르지는 않음
-> 오버헤드가 커서

  - In a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels
  - Few DBMSs have fully implemented the three-schema independence because it creates overhead for mapping between levels

2.3 DB Language and Interfaces

- **Data Definition Language (DDL)**
  - Specifies conceptual schema
  - Specifies the internal schema (**SDL**: Storage Definition Language) 저장소와 연관된 함수
  - Specifies the external schema or user views (**VDL**: View Definition Language)



- **Data Manipulation Language (DML)**
  - Allows retrieval, insertion, deletion, update of data

  1) Low-level or procedural DML 이전 계층, 네트워크 구조에서 사용
     - Must be embedded in a general-purpose programming language
     - Record-at-a-time DMLs
       - Hierarchical model's DL/1: GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT 트리 링크를 따라서 네비게이션을 따라 하는 것

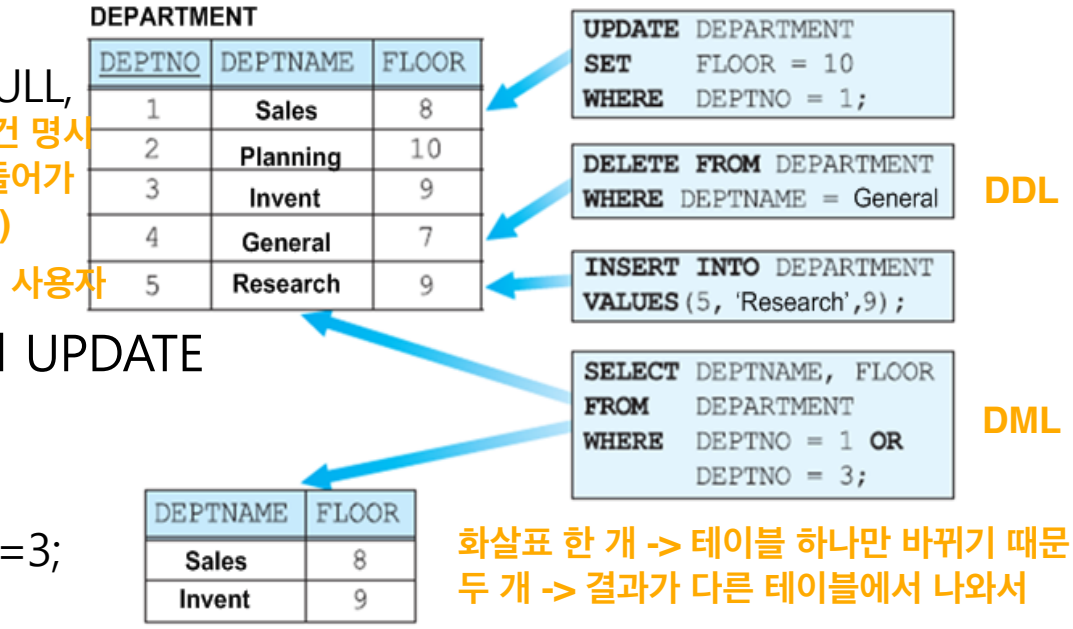
  2) High-level or nonprocedural DML SQL
     - Can be used on its own to specify complex DB operations concisely
     - Declarative language where it specifies which data to retrieve rather than how to retrieve it how 대신 what을 - 선언적인 질의!
     - Set-at-a-time or set-oriented DMLs

- **Host language, data sublanguage, query language** c/c++... 독단적으로 이용

2.3 DB Language and Interfaces

- **Structured Query Language (SQL)**
  - SQL is the standard for commercial relational DBMSs
  - Represents a combination of DDL and DML, as well as statements for constraint specification, and other features
    - DDL: CREATE/ALTER/DROP TABLE, CREATE/DROP INDEX, etc
      - **CREATE TABLE** DEPARTMENT
        (DEPTNO INTEGER NOT NULL,
        DEPTNAME CHAR(10),
        FLOOR INTEGER);

제약조건 명시 (널이 들어가면 안됨)



    - DML: SELECT, INSERT, DELETE, and UPDATE
      - **SELECT** DEPTNAME, FLOOR
        FROM DEPARTMENT
        WHERE DEPTNO=1 **OR** DEPTNO=3;
    - DCL: BEGIN TRANSACTION, COMMIT, ROLLBACK, GRANT, REVOKE, etc

권한 허가

  - It is a *declarative* rather than *procedural* language, which means that users declare what they want without having to write a step-by-step procedure

앞의 로우레벨

2.3 DB Language and Interfaces

sql 질의를 모르는 사람도 이용할 수 있는 인터페이스 제공

- **Menu-based interfaces for Web clients or browsing**
  - Present the user with lists of options (menus) that lead the user through the formulation of a request
- **Forms-based interfaces**
  - Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries

Menu/form interface provided by MySQL 피피티

2.3 DB Language and Interfaces

- **Graphical user interfaces**
  - User can specify a query by manipulating the diagram where a GUI displays a schema to the user in diagrammatic form
- **Natural language interfaces**
  - Accept requests written in English or some other language and attempt to understand them
- **Speech input and output** 음성인식 - 이건 좀 어려움(잡음, 여러 사람 말할 때 - 사람과 다름 / (영상 - 조명))
  - Limited use of speech as an input query and speech as an answer to a question or result of a request is becoming commonplace
- **Interfaces for parametric users**
  - Parametric users, such as bank teller, often have a small set of operations that they must perform repeatedly
- **Interfaces for the DBA**
  - Most DB system contain privileged commands that can be used only by the DBA staff
  - These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a DB

2.4 The Database System Environment

- **DBMS component modules**
  - DDL compiler
  - Interactive query interface
    - Query compiler
    - Query optimizer
  - Precompiler
  - DML compiler
  - Runtime database processor
  - Concurrency control system
  - Backup and recovery system
  - Buffer management
  - Stored data manager
  - System catalog

dbms마다 버퍼를
직접 관리할 수 있고
운영체제 버퍼를 사용할
수도 있음

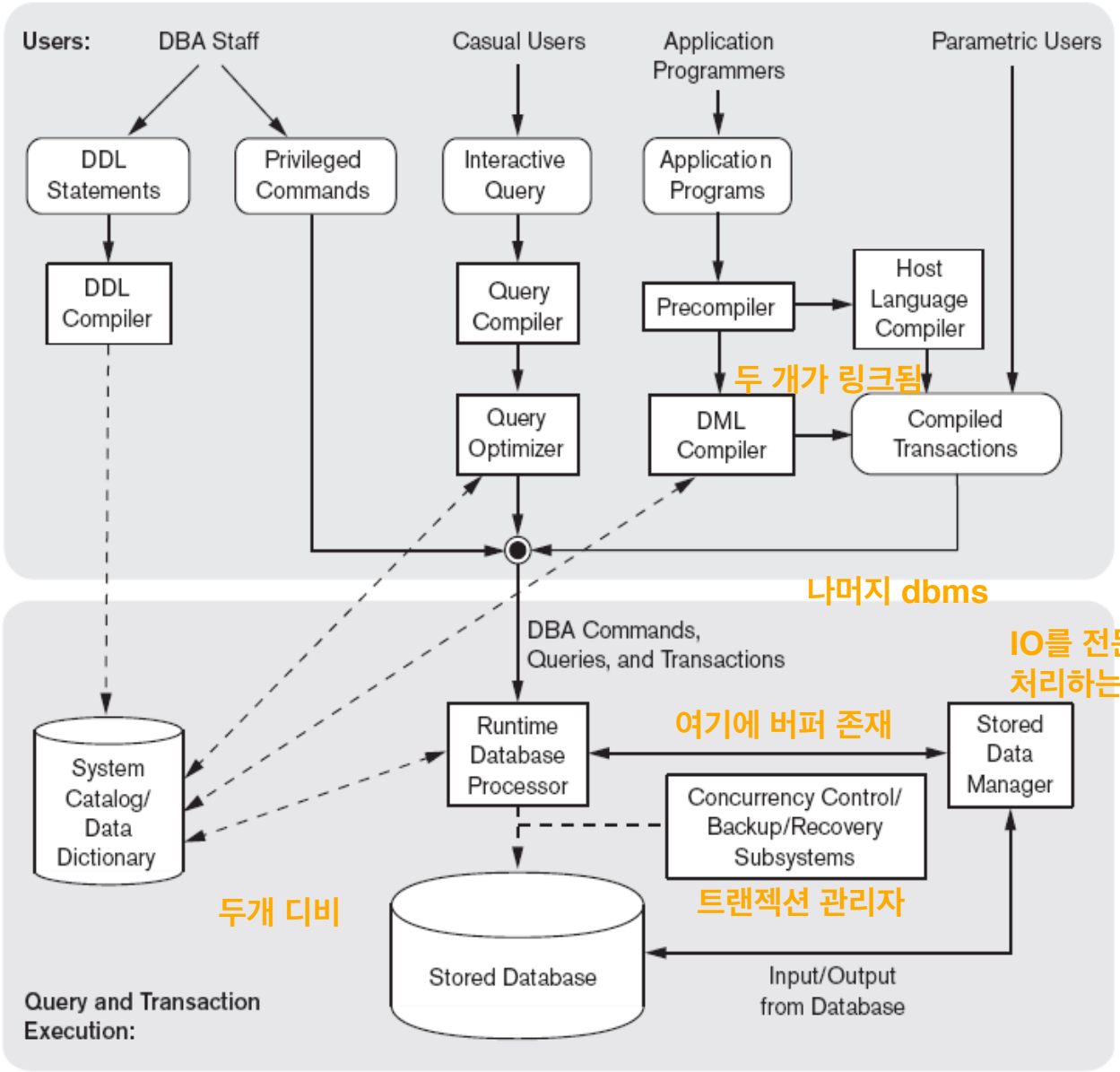
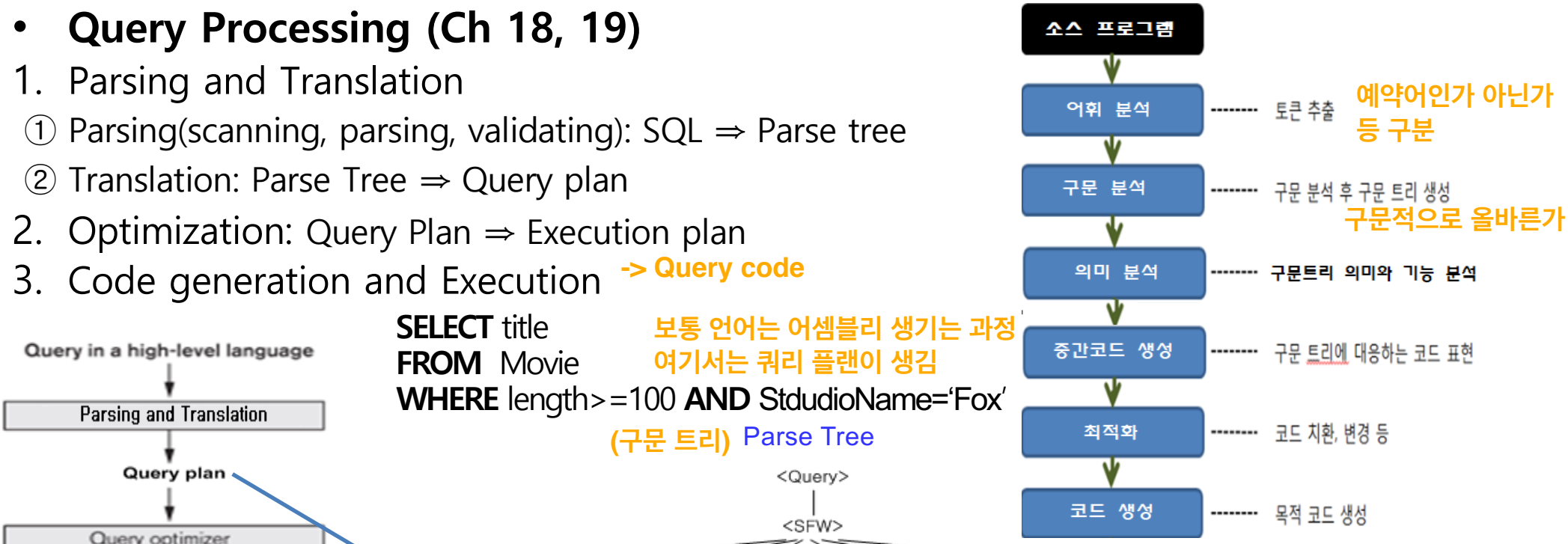
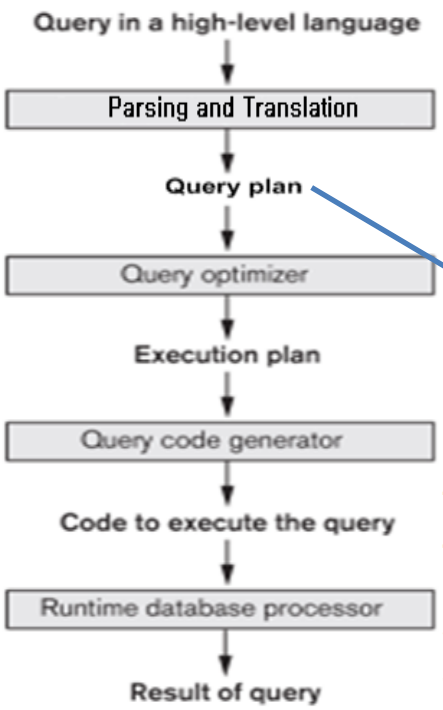


Figure 2.3
Component modules of a DBMS and their interactions.

- **Query Processing (Ch 18, 19)**

1. Parsing and Translation
   ① Parsing(scanning, parsing, validating): SQL ⇒ Parse tree
   ② Translation: Parse Tree ⇒ Query plan
2. Optimization: Query Plan ⇒ Execution plan
3. Code generation and Execution -> Query code

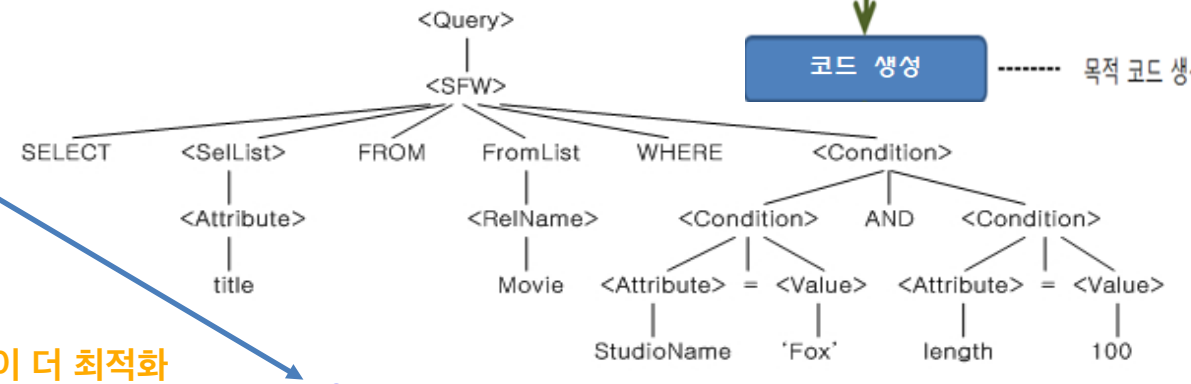




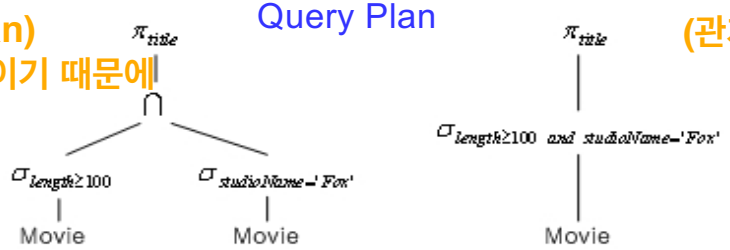
SELECT title
FROM Movie
WHERE length>=100 **AND** StdudioName='Fox'

보통 언어는 어셈블리 생기는 과정
여기서는 쿼리 플랜이 생김

(구문 트리) Parse Tree



Query Plan



두 개가 같음
오른쪽의 것이 더 최적화 (execution plan)
sql은 상위 레벨이기 때문에 최적화가 필요

(관계형대수) - sql이전의 언어(어셈블리?)

- **Runtime DB processor**
  - Executing the query code, whether in compiled or interpreted mode, to produce the query result
  - It interacts with most of the other component of the DBMS such as catalog, stored data manager, and CC/Recovery modules

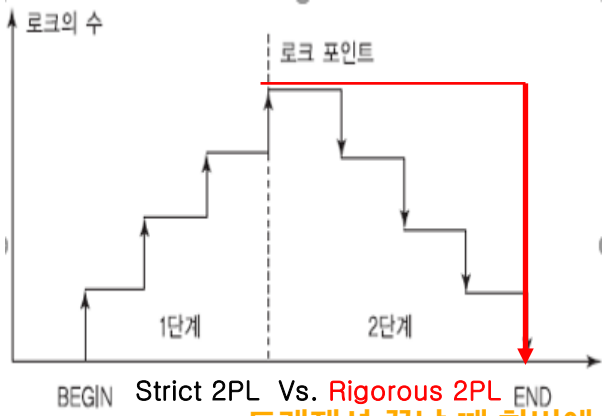
서로 다른 트랜잭션이 데이터에 접근할 때 문제가 발생 할 수 있음

- **Concurrency control module (Ch 20, 21)** 시리얼한 결과를 보장해줌
  - It ensures that TRs are performed concurrently without violating the data integrity of the respective databases
  - It ensures serializability of TRs by several mechanisms such as lock and timestamp
- **갱신 손실(lost update): 수행 중인 트랜잭션이 갱신한 내용을 다른 트랜잭션이 덮어 씀으로써 갱신이 무효가 되는 것**
  - 트랜잭션 T1은 X에서 Y로 100000을 이체하고, 트랜잭션 T2는 X의 값에 50000을 더한다. 두 트랜잭션이 수행되기 전의 X와 Y의 초기값이 각각 300000과 600000 이라면 T1의 수행을 먼저 완료하고 T2의 수행을 완료하던지, 또는 T2의 수행을 먼저 완료하고 T1의 수행을 완료하던지, X의 최종값은 250000, Y의 최종값은 700000이 되어야 한다.

T1	T2	X와 Y의 값
read_item(X); X=X-100000;		X=300000 Y=600000
	read_item(X); X=X+50000;	X=300000 Y=600000
write_item(X); read_item(Y);		X=200000 Y=600000
	write_item(X);	X=350000 Y=600000
Y=Y+100000; write_item(Y);		X=350000 Y=700000

〈표 9.2〉 로크 양립성 행렬

		현재 걸려 있는 로크		
		공유 로크	독점 로크	로크가 걸려 있지 않음
요청 중인 로크	공유 로크	허용	대기	허용
	독점 로크	대기	대기	허용



- **Locking**
  - 트랜잭션에서 갱신을 목적으로 데이터 항목을 접근할 때는 독점 lock을 요청함
  - 트랜잭션에서 읽을 목적으로 데이터 항목을 접근할 때는 공유 lock을 요청함
  - 트랜잭션이 데이터 항목에 대한 접근을 끝낸 후에 lock을 해제함

트랜잭션 끝날 때 한번에 풀어줌

첫 번째가 락을 빨리 빨리 풀어줘서 좋은데 시스템이 이를 모면 두 번째를 사용

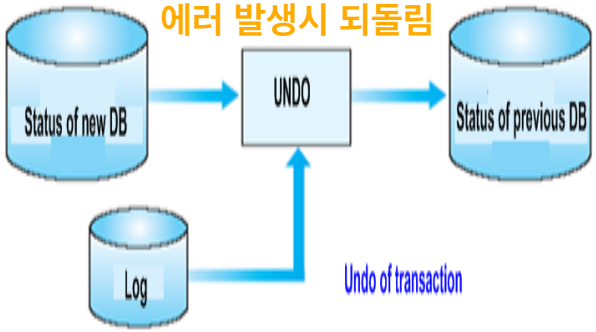
- **2-phase locking protocol (2PL)** 이러한 단순 락이 문제 발생 할 수 있어서 이를 이용
  - ① Growing phase: 새로운 lock을 요청할 수 있지만 보유하고 있던 lock은 해제할 수 없음
  - ② Shrinking phase: 보유하고 있던 lock을 해제할 수 있지만 새로운 lock을 요청할 수 없음

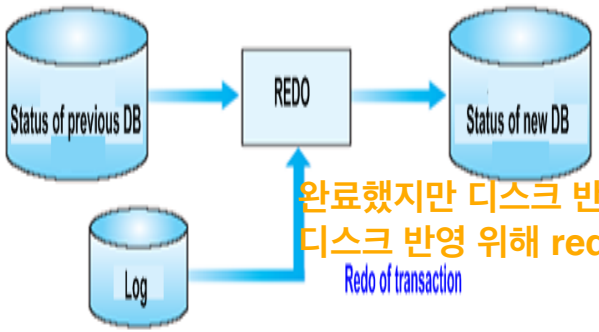
- **Recovery module (Ch 22)** (redo / undo) All or Nothing

체크 포인트를 지정해 완전 옛날꺼 필요 x

  - It ensures TR's atomicity and durability despite failures
  - DBMS carries out logging or back up for database recovery
    - Actions taken during normal TR processing to ensure enough information exists in the log to recover from failures
    - Actions such as undo/redo taken after a failure to recover the DB contents to a state that ensures atomicity and durability

에러 발생시 되돌림





완료했지만 디스크 반영x
디스크 반영 위해 redo

〈표 9.4〉 재수행할 트랜잭션과 취소할 트랜잭션

마지막으로 기록된 로그 순서 번호	작업	결과
I = 0	아무 작업도 필요 없음	T1과 T2가 수행을 시작하기 전
1 ≤ I ≤ 4	- T1을 취소함: 1부터 I까지 T1이 생성한 로그 레코드의 이전값을 사용하여 데이터베이스 항목의 값을 되돌림	T1이 수행을 하지 않은 것과 같음
5 ≤ I ≤ 9	- T1을 재수행: 1부터 4까지 T1이 생성한 로그 레코드의 새값을 사용하여 데이터베이스 항목의 값을 기록함 - T2를 취소함: 5부터 I까지 T2가 생성한 로그 레코드의 이전값을 사용하여 데이터베이스 항목의 값을 되돌림	- T1의 수행이 완료됨 - T2는 수행을 하지 않은 것과 같음
10	T1과 T2를 재수행	T1과 T2의 수행이 완료됨

커밋이 안나옴 - 언두 실행

〈표 9.3〉 로그 레코드의 예

```
T1:  read_item(A);        T2:  read_item(A);
     A = A + 30;               A = A + 10;
     read_item(B);             write_item(A)
     B = B + 100;              read_item(D);
     write_item(B);            D = D - 10;
     read_item(C);             read_item(E);
     C = 2 * C;                read_item(B);
     write_item(C);            E = E + B;
     A = A + B + C;            write_item(E)
     write_item(A);            D = D + E;
                               write_item(D)
```

로그 순서 번호	로그 레코드
1	[T1, start]
2	[T1, B, 300, 400]
3	[T1, C, 5, 10]
4	[T1, A, 100, 540]
5	[T1, commit]
6	[T2, start]
7	[T2, A, 540, 550]
8	[T2, E, 80, 480]
9	[T2, D, 60, 530]
10	[T2, commit]

로그는 스타트, 커밋, b, 300 ,400 과거 데이터만 있으면 됨

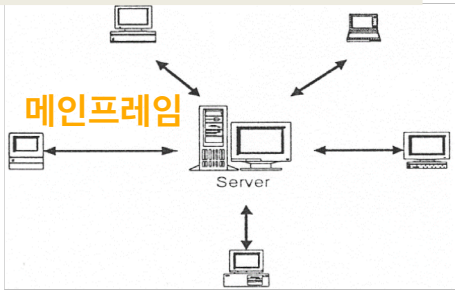
트렌젝션이 완료되어도 바로 반영하지 않고
버퍼에 저장. 따라서 redo를 수행함 - 체크포인트 (완전옛날x)

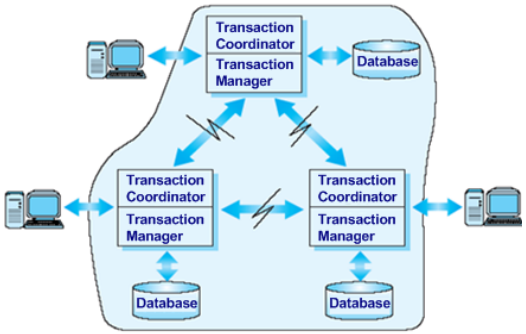
2.4 The Database System Environment

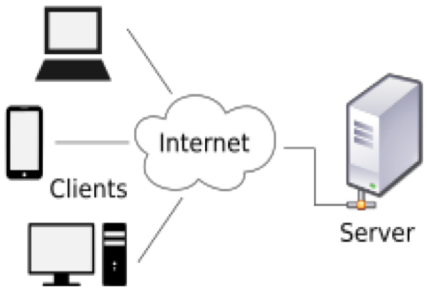
- **Database System Utilities**
  - Loading 데이터 웨어하우스로 가져올 때 (ETL)
    - Used to load existing data files such as text files or sequential files into the DB without writing programs
  - Backup
    - Creates a backup copy of the DB onto tape or other mass storage to restore the DB in case of catastrophic disk failure
  - Database storage reorganization
    - Allows restructuring the data from one type to another 해시 파일, …등등 파일 구조를 바꿀 때
  - Performance monitoring
    - Monitors DB usage and provides statistics to the DBA
- **Tools, Application Environments, and Communications Facilities**
  - CASE Tools are used in the design phase of DB systems
  - Data dictionary (data repository) system 다른 책들은 System catalog와 구분x
    - Stores design decisions, usage standards, application program descriptions, and user information accessed mainly by users 사용자에 의해 필요한 정보들을 저장함
  - Application development environments such as PowerBuilder
  - Communications software
    - Allows users at locations remote from the DBS site to access the DB through computer terminals, workstations, or personal computers

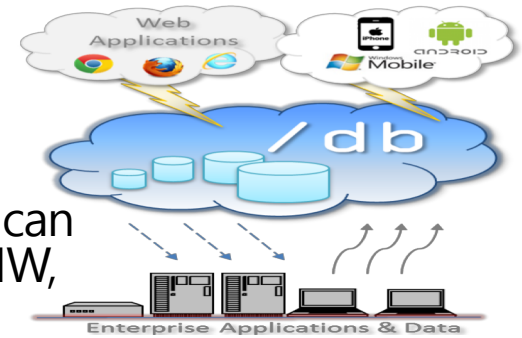
2.5 Centralized and Client/Server Architectures for DBMSs

- **The architecture of a DBSs is greatly influenced by the underlying computer system on which the DB is running:**
  - Centralized architecture
    - All DBMS functionality, application program execution, and user interface processing carried out on one main frame computer
    - Users connect DB through the remote terminals/PCs which only provide display capabilities
  - Parallel architecture
    - Allows DBS activities to be speeded up, allowing fast response to transactions
  - Distributed architecture (Homogeneous, Heterogeneous)
    - Handles geographically or administratively distributed data spread across multiple DBSs 빅데이터 처리(no sql) 분산적으로 처리
  - Client-server architecture
    - A distributed application structure that partitions tasks or workloads between the service requester, called client, and the provider of a resource or service, called server
  - Cloud architecture
    - A DB accessible to clients from the cloud and delivered to users on demand via the Internet from a cloud DB provider's servers
    - Also referred to as DB as a Service(DBaaS) because users can access to a DB without the need for setting up physical HW, installing SW or configuring for performance









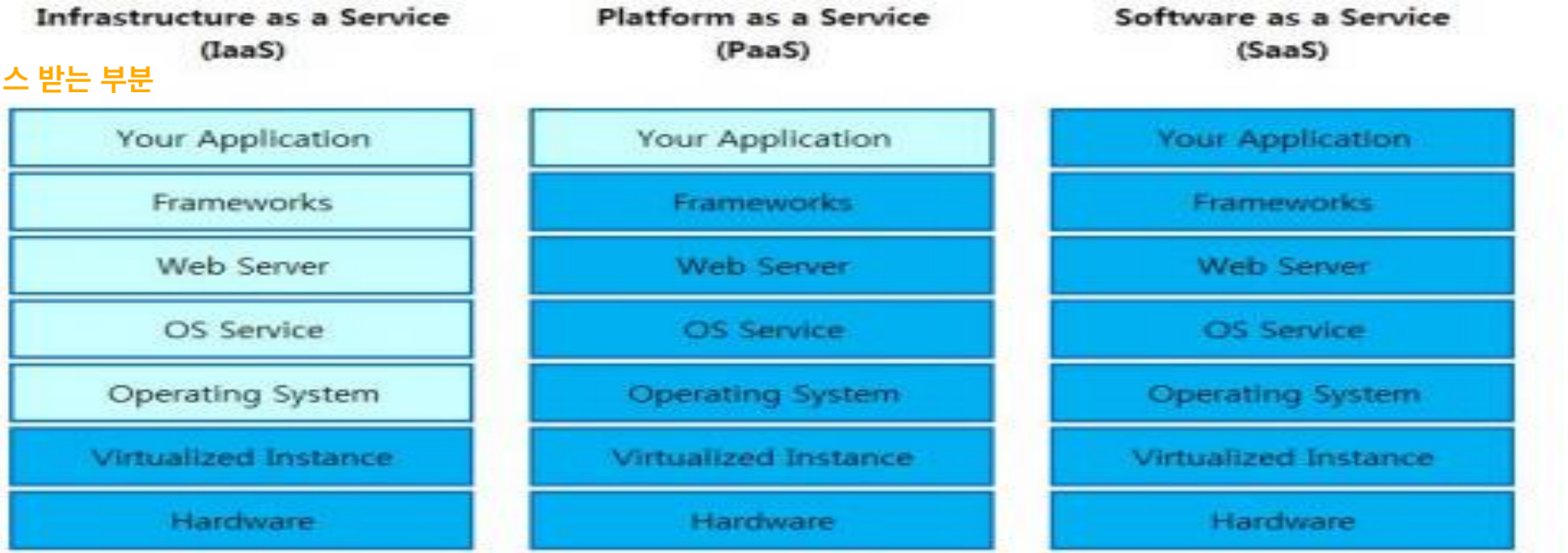
2.5 Centralized and Client/Server Architectures for DBMSs

- 클라우드 서비스 종류

추가 앞의 Daas

  - IaaS / PaaS / SaaS : 서비스 모델에 따른 구분으로, 서비스 제공자와 이용자가 관리·운용하는 범위에 따른 구분이다.
  - IaaS (Infrastructure as a Service): 네트워크, 스토리지, 서버와 같은 컴퓨팅 자원을 제공한다. 이용자 입장에서는 사용량이 급격히 변동하더라도 데이터센터를 추가 구축할 필요 없이 비용 지불만을 통해 탄력적으로 운용할 수 있다.
  - PaaS (Platform as a Service): 클라우드 상에서 소프트웨어 개발을 위한 플랫폼을 제공한다. 어플리케이션 개발자를 위한 것으로 다양한 API(Application Programming Interface) 제공을 통한 용이한 개발 환경을 의미한다.
  - SaaS (Software as a Service): 온라인 형태로 제공되는 소프트웨어·어플리케이션.

파란색 - 서비스 받는 부분



2.5 Centralized and Client/Server Architectures for DBMSs

- **Basic Client/Server Architectures** 두 번째 과제 - 서버/클라이언트 db 구현
  - Client
    - User machine that provides user interface capabilities and local processing
  - Server
    - System containing both hardware and software which provides services to the client machines
      - file access, printing, archiving, or database access
    - Servers with specific functionalities
      - File server , Printer server, Web server or E-mail server
  - Often client and server communicate over a computer network on separate hardware, but both client and server may reside in the same system

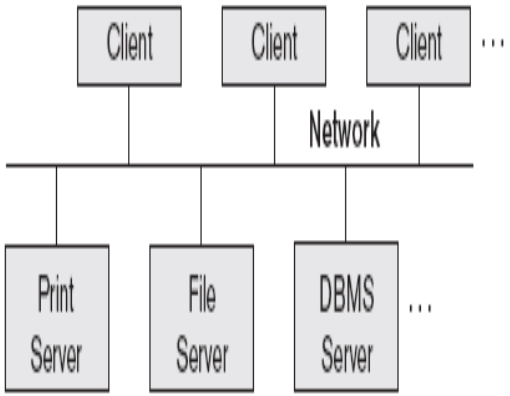


Figure 2.5
Logical two-tier client/server architecture.

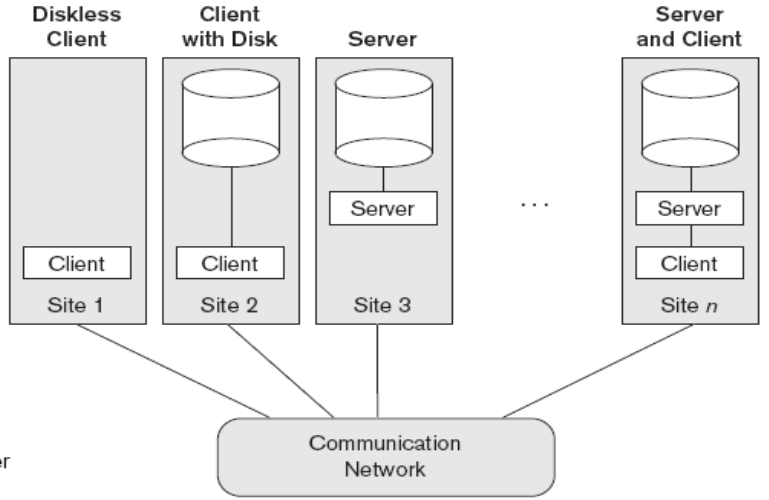
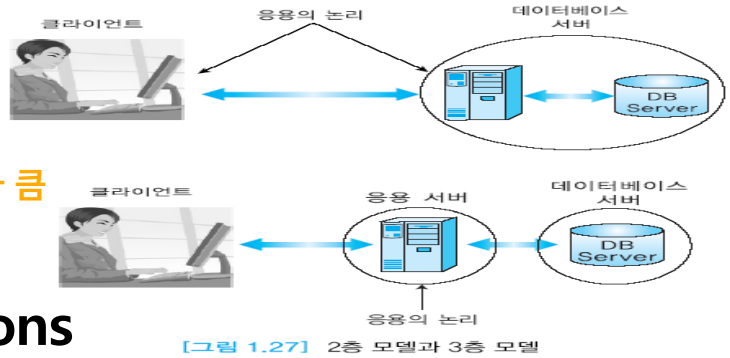


Figure 2.6
Physical two-tier client/server architecture.

2.5 Centralized and Client/Server Architectures for DBMSs

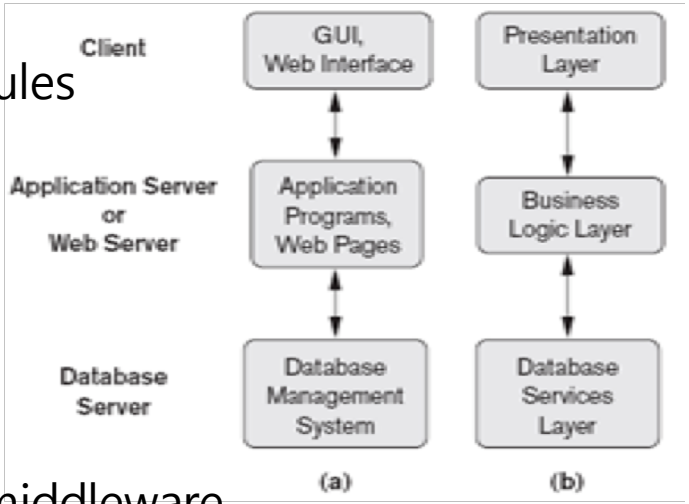
- **Two-Tier Client/Server Architectures for DBMSs**
  - Client handles user interface programs and application programs
  - Server handles query and transaction functionality related to SQL processing
  - Provides API(Application Programming Interface) such as ODBC and JDBC
    - Allows client-side programs to call the DBMS

응용은 종종 변경될 수 있는데,
클라이언트에 있으면 변경 오버헤드가 큼
3로 하면 ㄱㅊ



[그림 1.27] 2층 모델과 3층 모델

- **Three~n-Tier Architectures for Web Applications**
  - Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server
    - Web server (Application Server)
    - Runs application programs and stores business rules
  - N-tier
    - Divides the layers between the user and the stored data further into finer components
    - Typically, the business logic layer is divided into multiple layers
    - Vendors of ERP and CRM packages often use a middleware



2.6 Classification of Database Management Systems

- **Data model**
  - Hierarchical and network (legacy), Relational, Object, OR(object relational), XML DBMS

html, xml 차이
사용자 정의, 데이터 스스로 정의

xml - 반정형(semi structure)
구조는 있지만 완벽히 정형화 되지는 않음

- **Number of users**
  - Single-user, Multiuser

- **Number of sites**
  - Centralized, Distributed, Federated DBMS (or multidatabase system)

- **Cost**
  - Open source, Different types of licensing

- **General or special-purpose**
  - Special-purpose DBMS is built for a specific application such as airline reservations and telephone directory systems when performance is a primary consideration

실시간 트렌젝션 처리(oltp)

Summary

- **Data models, Schemas, and Instances**

독립성을 위해 로지컬, 피지컬 두가지 구현이 필요

- **Three-Schema Architecture and Data Independence**

- **Types of languages supported by the DBMSs**
- **Interfaces provided by the DBMSs**
- **The Database System Environment**
  - Compiler, Runtime DB processor, Concurrency control system, Backup and recovery system, Buffer management , Stored data manager, System catalog
- **Centralized and Client/Server Architectures for DBMSs**
- **DBMS classification criteria:**
  - Data model, number of users, number of sties, cost, general/special purpose