제 1 장 데이터베이스 시스템

1장. 데이터베이스 시스템

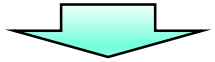
- 컴퓨터를 사용하여 정보를 수집하고 분석하는데 데이터베이스 기술이 활용되고 있음
- 정보와 데이터는 서로 다름
- 데이터베이스(database)의 정의

> 데이터베이스는 조직체의 응용 시스템들이 공유해서 사용하는 운영 데이터(operational data)들이 구조적으로 통합된 모임이다. 데이터베이스의 구조는 사용되는 데이터 모델에 의해 결정된다.

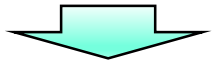
데이터 vs. 정보

데이터

Name	Address	Course	Grade
홍길동	123 Kensigton	Chemistry 102	C+
홍길동	123 Kensigton	Chinese 3	A
홍길동	122 Kensigton	Data Structures	B
홍길동	123 Kensigton	English 101	A
김철수	88 West 1st St.	Psychology 101	A
박영희	100 Capitol Ln.	Psychology 102	A
김철수	88 West 1st St.	Human Cultures	A
김철수	88 West 1st St.	Database	A



질의: Database 과목을 수강한 학생은?



정보: Database 과목을 수강한 학생은 김철수이다.

데이터는
프로그램과 질의에 의해
정보로 변환

1장. 데이터베이스 시스템(계속)

❑ 데이터베이스의 예

대학에서는 데이터베이스에 학생들에 관하여 신상 정보, 수강 과목, 성적 등을 기록하고, 각 학과에 개설되어 있는 과목들에 관한 정보를 유지하고, 교수에 관해서 신상 정보, 담당 과목, 급여 정보를 유지한다.

항공기 예약 시스템에서는 여행사를 통해 항공기 좌석을 예약하면 모든 예약 정보가 데이터베이스에 기록된다.

1장. 데이터베이스 시스템(계속)

- 데이터베이스의 특징
  - ✓ 데이터베이스는 데이터의 대규모 저장소로서, 여러 부서에 속하는 여러 사용자에 의해 동시에 사용됨
  - ✓ 모든 데이터가 중복을 최소화하면서 통합됨
  - ✓ 데이터베이스는 한 조직체의 운영 데이터뿐만 아니라 그 데이터에 관한 설명(데이터베이스 스키마 또는 메타데이터**(metadata)**)까지 포함.
  - ✓ 프로그램과 데이터 간의 독립성이 제공됨
  - ✓ 효율적으로 접근이 가능하고 질의를 할 수 있음
- 데이터베이스 관리 시스템(DBMS: Database Management System)
  - ✓ 데이터베이스를 정의하고, 질의어를 지원하고, 리포트를 생성하는 등의 작업을 수행하는 소프트웨어

1.1 데이터베이스 시스템 개요

❑ 데이터베이스 스키마

- ✓ 전체적인 데이터베이스 구조를 뜻하며 자주 변경되지는 않음
- ✓ 데이터베이스의 모든 가능한 상태를 미리 정의
- ✓ 내포(intension)라고 부름

❑ 데이터베이스 상태

- ✓ 특정 시점의 데이터베이스의 내용을 의미하며, 시간이 지남에 따라 계속해서 바뀜
- ✓ 외연(extension)이라고 부름

1.1 데이터베이스 시스템 개요(계속)

데이터베이스 스키마

```
DEPARTMENT(DEPTNO, DEPTNAME, FLOOR)
EMPLOYEE(EMPNO, EMPNAME, TITLE, DNO, SALARY)
```

데이터베이스 상태

DEPARTMENT

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9

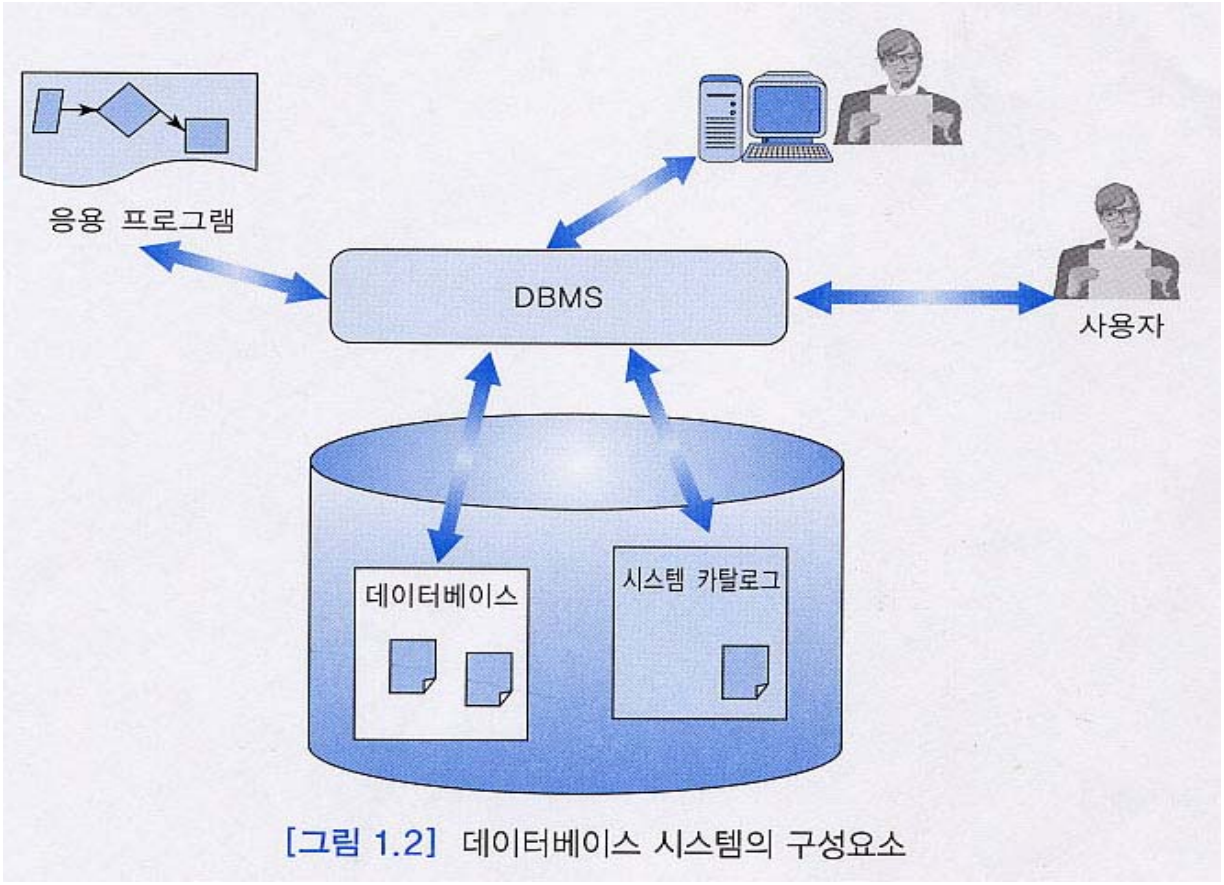
EMPLOYEE

EMPNO	EMPNAME	TITLE	DNO	SALARY
2106	김창섭	대리	2	2000000
3426	박영권	과장	3	2500000
3011	이수민	부장	1	3000000
1003	조민희	대리	1	2000000
3427	최종철	사원	3	1500000

[그림 1.1] 데이터베이스 스키마와 데이터베이스 상태

1.1 데이터베이스 시스템 개요(계속)

- 데이터베이스 시스템(DBS: Database System)의 구성 요소



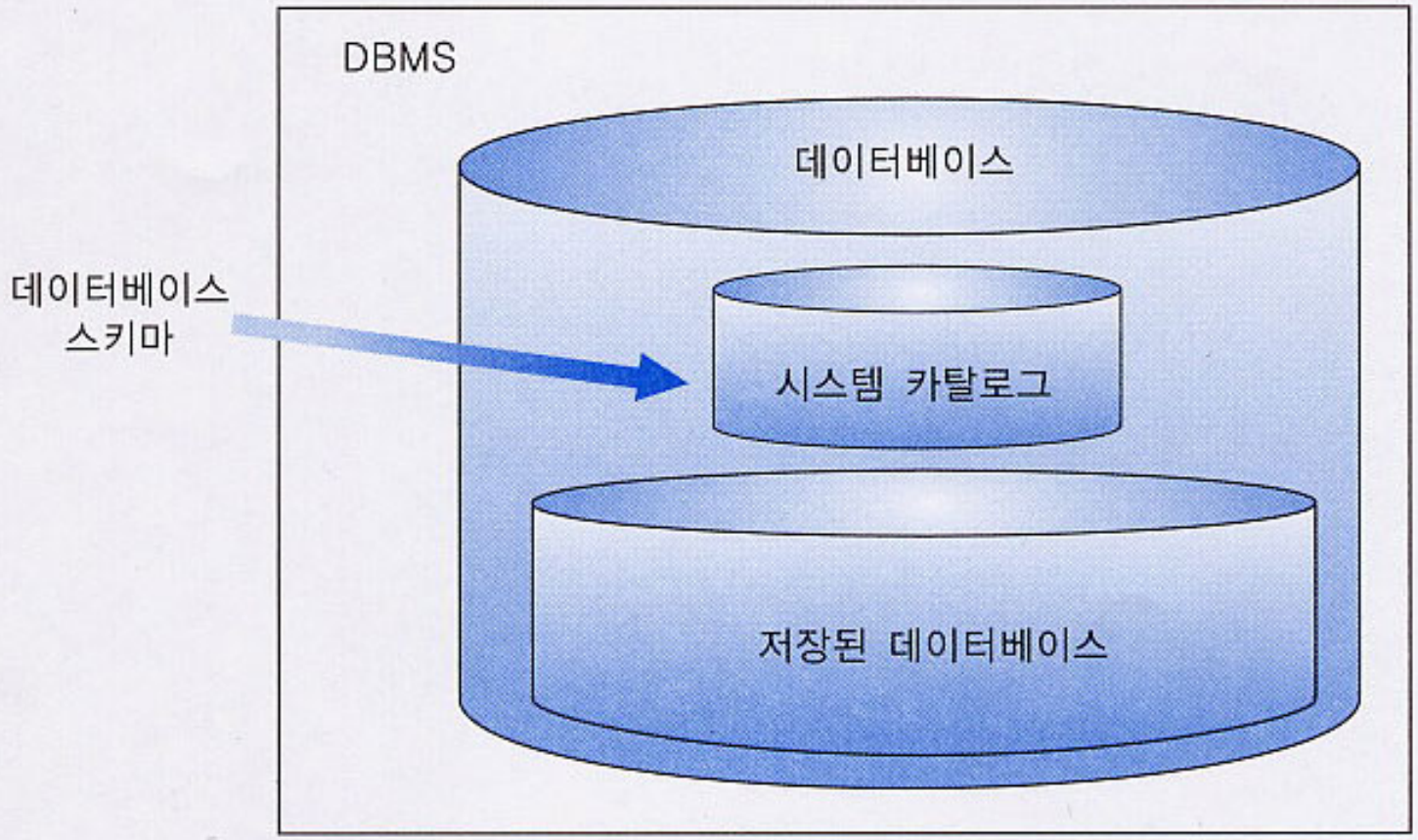
[그림 1.2] 데이터베이스 시스템의 구성요소

1.1 데이터베이스 시스템 개요(계속)

❑ 데이터베이스

- ✓ 조직체의 응용 시스템들이 공유해서 사용하는 운영 데이터들이 구조적으로 통합된 모임
- ✓ 시스템 카탈로그(또는 데이터 사전)와 저장된 데이터베이스로 구분할 수 있음
- ✓ 시스템 카탈로그(system catalog)는 저장된 데이터베이스의 스키마 정보를 유지

1.1 데이터베이스 시스템 개요(계속)



[그림 1.3] 시스템 카탈로그와 저장된 데이터베이스

1.1 데이터베이스 시스템 개요(계속)

❑ DBMS

- ✓ 사용자가 새로운 데이터베이스를 생성하고, 데이터베이스의 구조를 명시할 수 있게 하고, 사용자가 데이터를 효율적으로 질의하고 수정할 수 있도록 하며, 시스템의 고장이나 권한이 없는 사용자로부터 데이터를 안전하게 보호하며, 동시에 여러 사용자가 데이터베이스를 접근하는 것을 제어하는 소프트웨어 패키지
- ✓ 데이터베이스 언어라고 부르는 특별한 프로그래밍 언어를 한 개 이상 제공
- ✓ SQL은 여러 DBMS에서 제공되는 사실상의 표준 데이터베이스 언어

1.1 데이터베이스 시스템 개요(계속)

[그림 1.4] 컴퓨터 시스템에서 DBMS의 위치

1.1 데이터베이스 시스템 개요(계속)

❑ 사용자

- ✓ 데이터베이스 사용자는 여러 부류로 나눌 수 있음

❑ 하드웨어

- ✓ 데이터베이스는 디스크와 같은 보조 기억 장치에 저장되며, DBMS에서 원하는 정보를 찾기 위해서는 디스크의 블록들을 주기억 장치로 읽어들여야 하며, 계산이나 비교 연산들을 수행하기 위해 중앙 처리 장치가 사용됨
- ✓ DBMS 자체도 주기억 장치에 적재되어 실행되어야 함

1.1 데이터베이스 시스템 개요(계속)

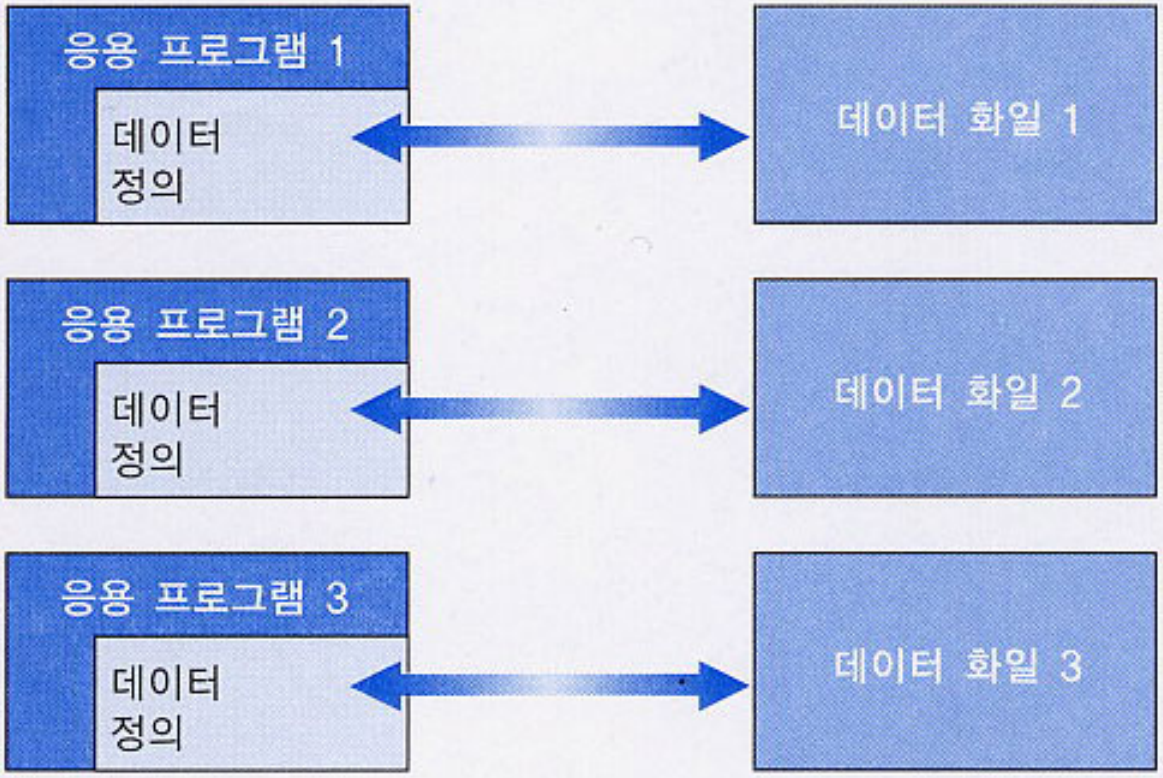
- 데이터베이스 시스템의 요구사항
  - ✓ 데이터 독립성
  - ✓ 효율적인 데이터 접근
  - ✓ 데이터에 대한 동시 접근
  - ✓ 백업과 회복
  - ✓ 중복을 줄이거나 제어하며 일관성 유지
  - ✓ 데이터 무결성
  - ✓ 데이터 보안
  - ✓ 쉬운 질의어
  - ✓ 다양한 사용자 인터페이스

1.2 화일 시스템 vs. DBMS

❑ 화일 시스템을 사용한 기존의 데이터 관리

- ✓ 화일 시스템은 DBMS가 등장하지 않았을 때인 1960년대부터 사용되어 왔음
- ✓ 화일의 기본적인 구성요소는 순차적인 레코드들
- ✓ 한 레코드는 연관된 필드들의 모임
- ✓ 화일을 접근하는 방식이 응용 프로그램 내에 상세하게 표현되므로 데이터에 대한 응용 프로그램의 의존도가 높음

1.2 화일 시스템 vs. DBMS(계속)

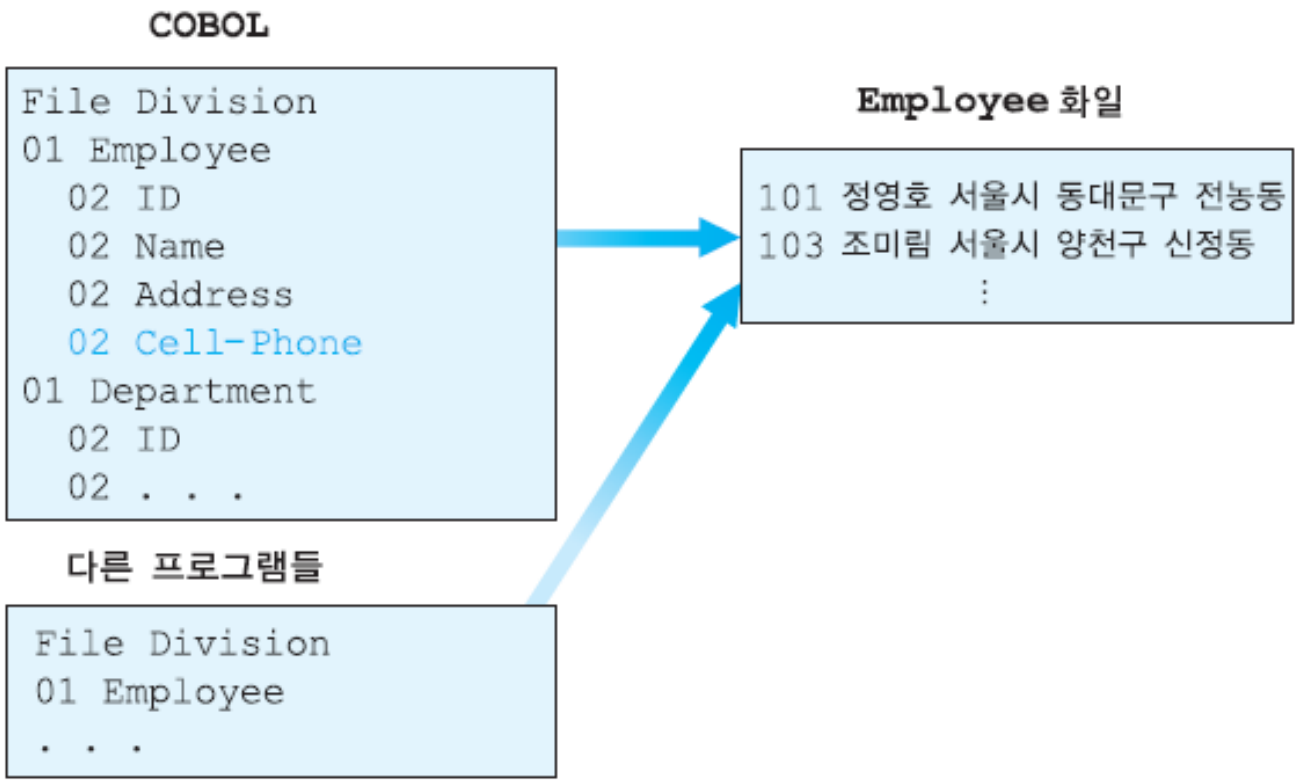


[그림 1.5] 화일 시스템에서 응용 프로그램과 화일 간의 대응 관계

1.2 화일 시스템 vs. DBMS(계속)

그림 1.6에서 두 개의 코볼 프로그램에는 Employee 화일의 필드들이 열거되어 있다. 만일 Employee 화일에 사원의 휴대폰 번호를 추가로 나타내려면 Employee 화일의 레코드를 하나씩 읽어서, 휴대폰 번호 필드를 추가한 레코드를 새로운 Employee 화일에 기록하는 프로그램을 작성해야 한다. 그 다음에 기존의 Employee 화일을 사용하던 모든 응용 프로그램들을 찾아서 휴대폰 번호 필드를 추가해야 한다.

1.2 화일 시스템 vs. DBMS(계속)



[그림 1.6] 응용 프로그램과 데이터 화일의 대응 예

1.2 화일 시스템 vs. DBMS(계속)

❑ 화일 시스템의 단점

✓ 데이터가 많은 화일에 중복해서 저장됨

예 :

그림 1.7에서 보는 것처럼 기업의 인사 관리 응용 프로그램에 사용되는 EMPLOYEE 화일과 사원 교육 관리 응용 프로그램에서 사용되는 ENROLLMENT 화일에 DEPARTMENT가 중복되어 나타날 수 있다. 어떤 사원의 DEPARTMENT 필드 값이 바뀔 때 두 화일에서 모두 수정하지 않으면 동일한 사원의 소속 부서가 화일마다 다르게 되어 데이터의 불일치가 발생한다.

EMPLOYEE 화일(인사 관리 프로그램용)

NAME	JUMIN-NO	DEPARTMENT	…	ADDRESS

ENROLLMENT 화일(교육 관리 프로그램용)

NAME	JUMIN-NO	DEPARTMENT	…	COURSE

[그림 1.7] 두 화일에서 DEPARTMENT가 중복됨

1.2 화일 시스템 vs. DBMS(계속)

- 화일 시스템의 단점(계속)
  - ✓ 다수 사용자들을 위한 동시성 제어가 제공되지 않음
  - ✓ 검색하려는 데이터를 쉽게 명시하는 질의어가 제공되지 않음
  - ✓ 보안 조치가 미흡
  - ✓ 회복 기능이 없음
  - ✓ 프로그램-데이터 독립성이 없으므로 유지보수 비용이 많이 소요됨
  - ✓ 화일을 검색하거나 갱신하는 절차가 상대적으로 복잡하기 때문에 프로그래머의 생산성이 낮음
  - ✓ 데이터의 공유와 융통성이 부족

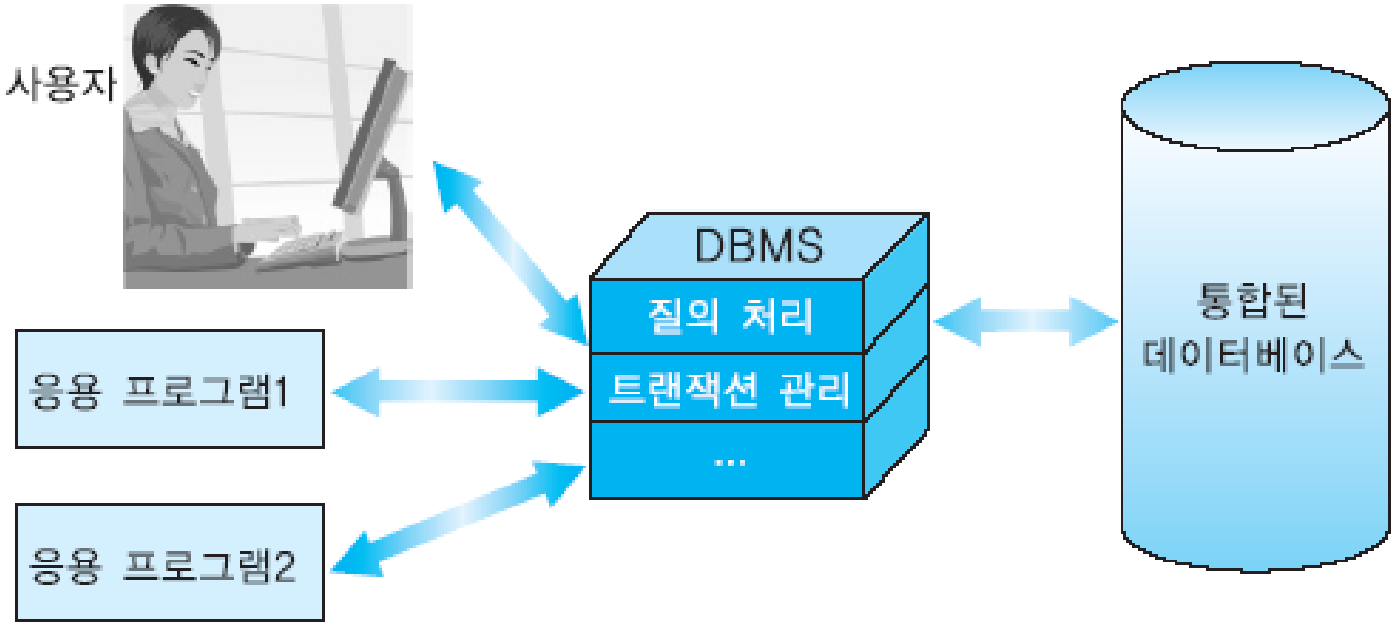
1.2 화일 시스템 vs. DBMS(계속)

❑ DBMS를 사용한 데이터베이스 관리

- ✓ 여러 사용자와 응용 프로그램들이 데이터베이스를 공유
- ✓ 사용자의 질의를 빠르게 수행할 수 있는 인덱스 등의 접근 경로를 DBMS가 자동적으로 선택하여 수행
- ✓ 권한이 없는 사용자로부터 데이터베이스를 보호
- ✓ 여러 사용자에 적합한 다양한 인터페이스를 제공
- ✓ 데이터 간의 복잡한 관계를 표현하며, 무결성 제약조건을 DBMS가 자동적으로 유지
- ✓ 시스템이 고장 나면 데이터베이스를 고장 전의 일관된 상태로 회복시킴
- ✓ 프로그램에 영향을 주지 않으면서 데이터베이스 구조를 변경할 수 있음

프로그램-데이터 독립성(program-data independence)

1.2 화일 시스템 vs. DBMS(계속)



[그림 1.8] DBMS를 사용한 데이터베이스 관리

1.2 화일 시스템 vs. DBMS(계속)

- DBMS의 장점
  - ✓ 중복성과 불일치가 감소됨
  - ✓ 시스템을 개발하고 유지하는 비용이 감소됨
  - ✓ 표준화를 시행하기가 용이
  - ✓ 보안이 향상됨
  - ✓ 무결성이 향상됨
  - ✓ 조직체의 요구사항을 식별할 수 있음
  - ✓ 다양한 유형의 고장으로부터 데이터베이스를 회복할 수 있음
  - ✓ 데이터베이스의 공유와 동시 접근이 가능함

1.2 화일 시스템 vs. DBMS(계속)

〈표 1.1〉 화일 시스템 방식과 DBMS 방식의 비교

화일 시스템 방식	DBMS 방식
데이터에 대한 물리적 접근만 조정한다.	데이터에 대한 물리적 접근과 논리적인 접근을 모두 조정한다.
동일한 화일을 두 개 이상의 프로그램이 동시에 접근할 수 없다.	동일한 데이터를 다수 사용자가 동시에 접근할 수 있다.
데이터가 비구조적이며, 중복성과 유지보수 비용이 높다.	데이터가 구조화되어 있으며, 중복성과 유지보수 비용이 낮다.
어떤 프로그램이 기록한 데이터는 다른 프로그램에서 읽을 수 없는 경우가 많다.	접근 권한이 있는 모든 프로그램이 데이터를 공유한다.
데이터에 대한 접근은 미리 작성된 프로그램을 통해서만 가능하다.	질의어를 사용하여 데이터에 대한 융통성 있는 접근이 가능하다.
각 응용 프로그램마다 화일이 따로 있으므로 데이터가 통합되어 있지 않다.	데이터가 중복을 배제하면서 통합되어 있다.

1.2 화일 시스템 vs. DBMS(계속)

❑ DBMS 선정시 고려 사항

✓ 기술적 요인

- DBMS에 사용되고 있는 데이터 모델, DBMS가 지원하는 사용자 인터페이스, 프로그래밍 언어, 응용 개발 도구, 저장 구조, 성능, 접근 방법 등

✓ 경제적 요인

- 소프트웨어와 하드웨어 구입 비용, 유지 보수 비용, 직원들의 교육 지원 등

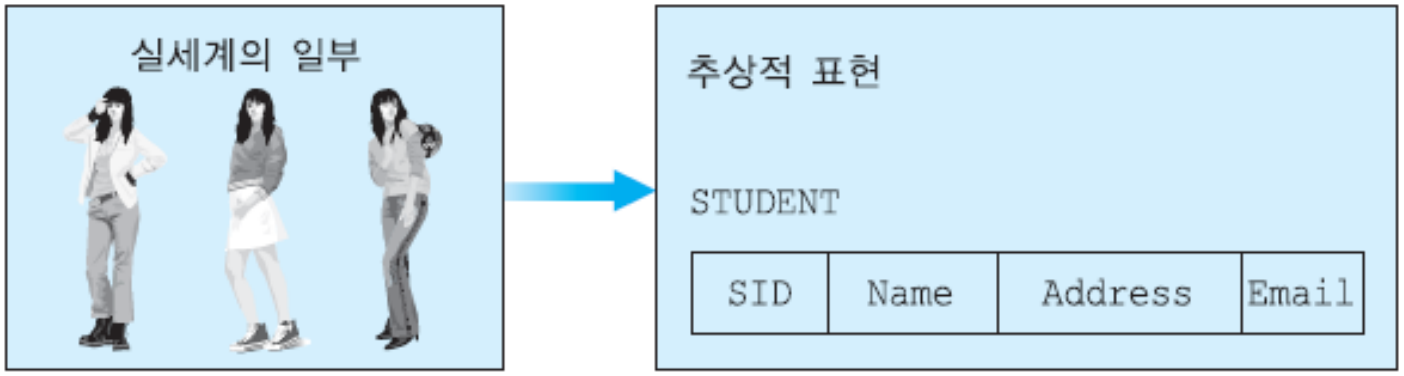
1.2 화일 시스템 vs. DBMS(계속)

- DBMS의 단점
  - ✓ 추가적인 하드웨어 구입 비용이 들고, DBMS 자체의 구입 비용도 상당히 비쌈
  - ✓ 직원들의 교육 비용도 많이 소요됨
  - ✓ 비밀과 프라이버시 노출 등의 단점이 존재할 수 있음
  - ✓ 초기의 투자 비용이 너무 클 때, 오버헤드가 너무 클 때, 응용이 단순하고 잘 정의되었으며 변경되지 않을 것으로 예상될 때, 엄격한 실시간 처리 요구사항이 있을 때, 데이터에 대한 다수 사용자의 접근이 필요하지 않을 때는 DBMS를 사용하지 않는 것이 바람직할 수 있음

1.3 DBMS 발전 과정

❑ 데이터 모델

- ✓ 데이터베이스의 구조를 기술하는데 사용되는 개념들의 집합인 구조(데이터 타입과 관계), 이 구조 위에서 동작하는 연산자들, 무결성 제약조건들
- ✓ 사용자에게 내부 저장 방식의 세세한 사항은 숨기면서 데이터에 대한 직관적인 뷰를 제공하는 동시에 이들 간의 사상을 제공



[그림 1.9] 관계 데이터 모델에서의 실세계 표현

1.3 DBMS 발전 과정(계속)

❑ 데이터 모델의 분류

✓ 고수준 또는 개념적 데이터 모델(conceptual data model)

- 사람이 인식하는 것과 유사하게 데이터베이스의 전체적인 논리적 구조를 명시
- 예: 엔티티-관계(ER: Entity-Relationship) 데이터 모델과 객체 지향 데이터 모델

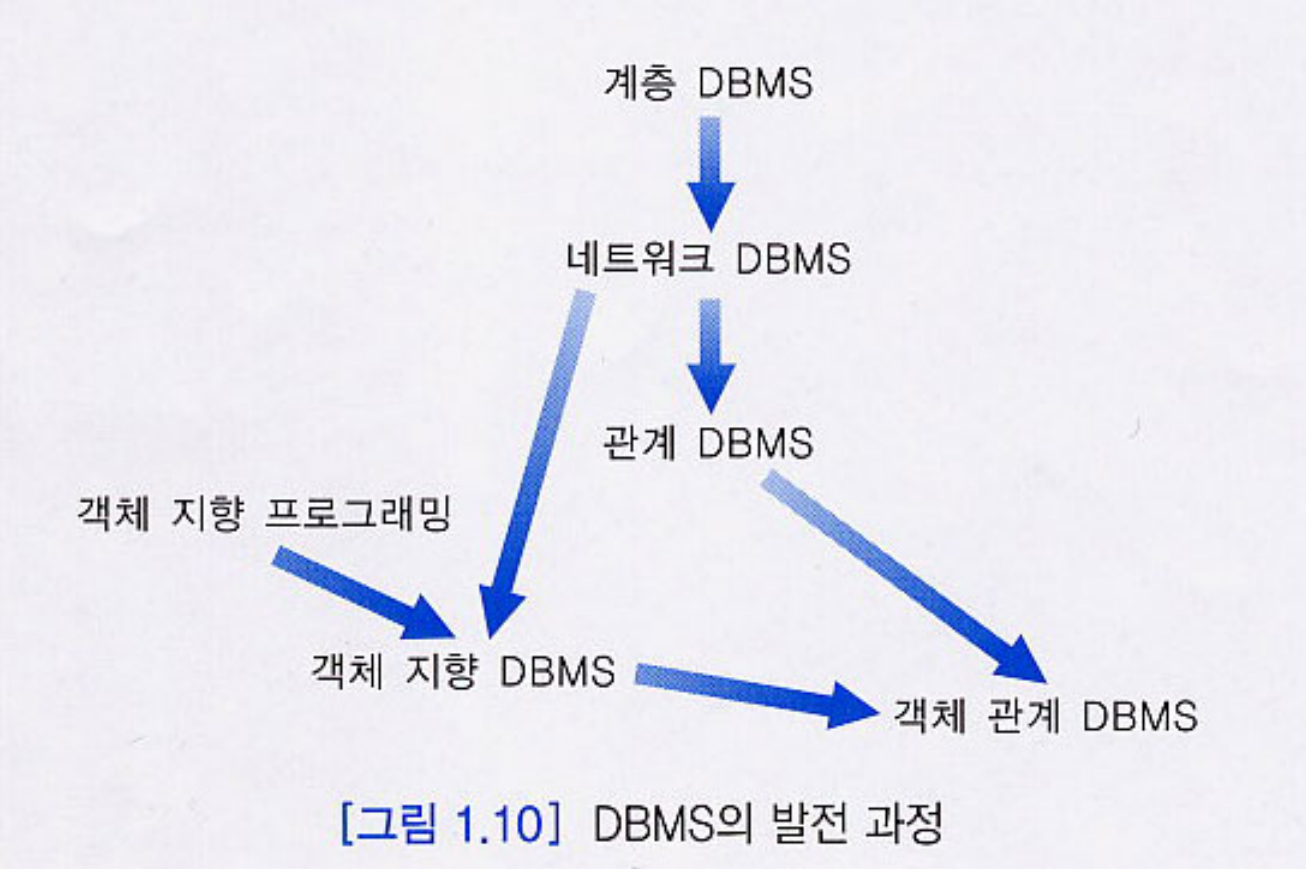
✓ 표현(구현) 데이터 모델(representation(implementation) data model)

- 최종 사용자가 이해하는 개념이면서 컴퓨터 내에서 데이터가 조직되는 방식과 멀리 떨어져 있지는 않음
- 예: 계층 데이터 모델(hierarchical data model), 네트워크 데이터 모델(network data model), 관계 데이터 모델(relational data model)

✓ 저수준 또는 물리적인 데이터 모델(physical data model)

- 데이터베이스에 데이터가 어떻게 저장되는가를 기술
- 예: Unifying, ISAM, VSAM 등

1.3 DBMS 발전 과정(계속)

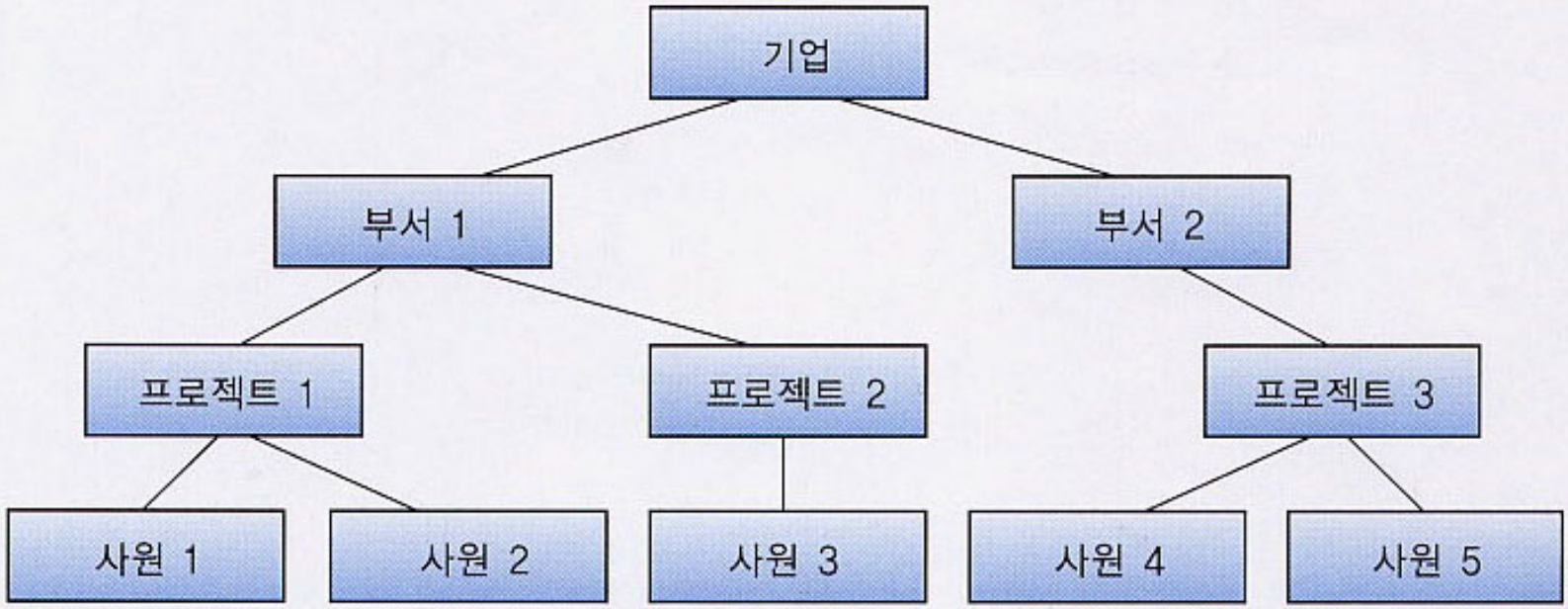


[그림 1.10] DBMS의 발전 과정

1.3 DBMS 발전 과정(계속)

- 계층 DBMS
  - ✓ 1960년대 후반에 최초의 계층 DBMS가 등장(예: IBM사의 IMS)
  - ✓ 트리 구조를 기반으로 하는 계층 데이터 모델을 사용한 DBMS
  - ✓ 계층 데이터 모델은 네트워크 데이터 모델의 특별한 사례
  - ✓ 장점
    - 어떤 유형의 응용에 대해서는 빠른 속도와 높은 효율성을 제공
  - ✓ 단점
    - 어떻게 데이터를 접근하는가를 미리 응용 프로그램에 정의해야 함
    - 데이터베이스가 생성될 때 각각의 관계를 명시적으로 정의해야 함
    - 레코드들이 링크로 연결되어 있으므로 레코드 구조를 변경하기 어려움

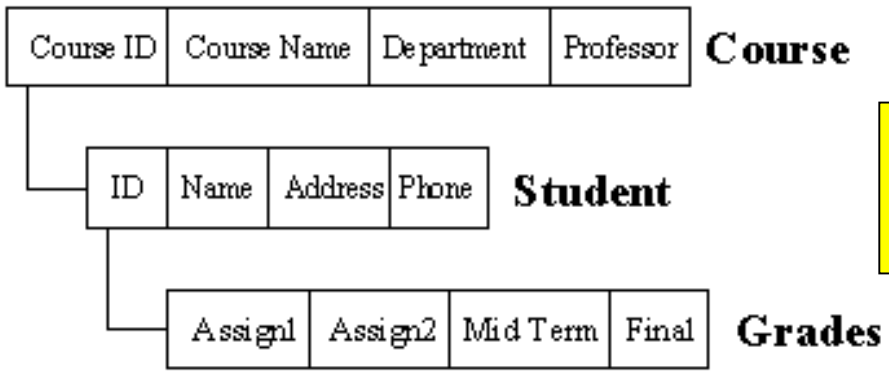
1.3 DBMS 발전 과정(계속)



[그림 1.11] 계층 데이터베이스의 예

예제: 계층 DBMS

Name	Address	Course	Grade
홍길동	123 Kensigton	Chemistry 102	C+
홍길동	123 Kensigton	Chinese 3	A
홍길동	122 Kensigton	Data Structures	B
홍길동	123 Kensigton	English 101	A
김철수	88 West 1st St.	Psychology 101	A
박영희	100 Capitol Ln.	Psychology 102	A
김철수	88 West 1st St.	Human Cultures	A
김철수	88 West 1st St.	Database	A

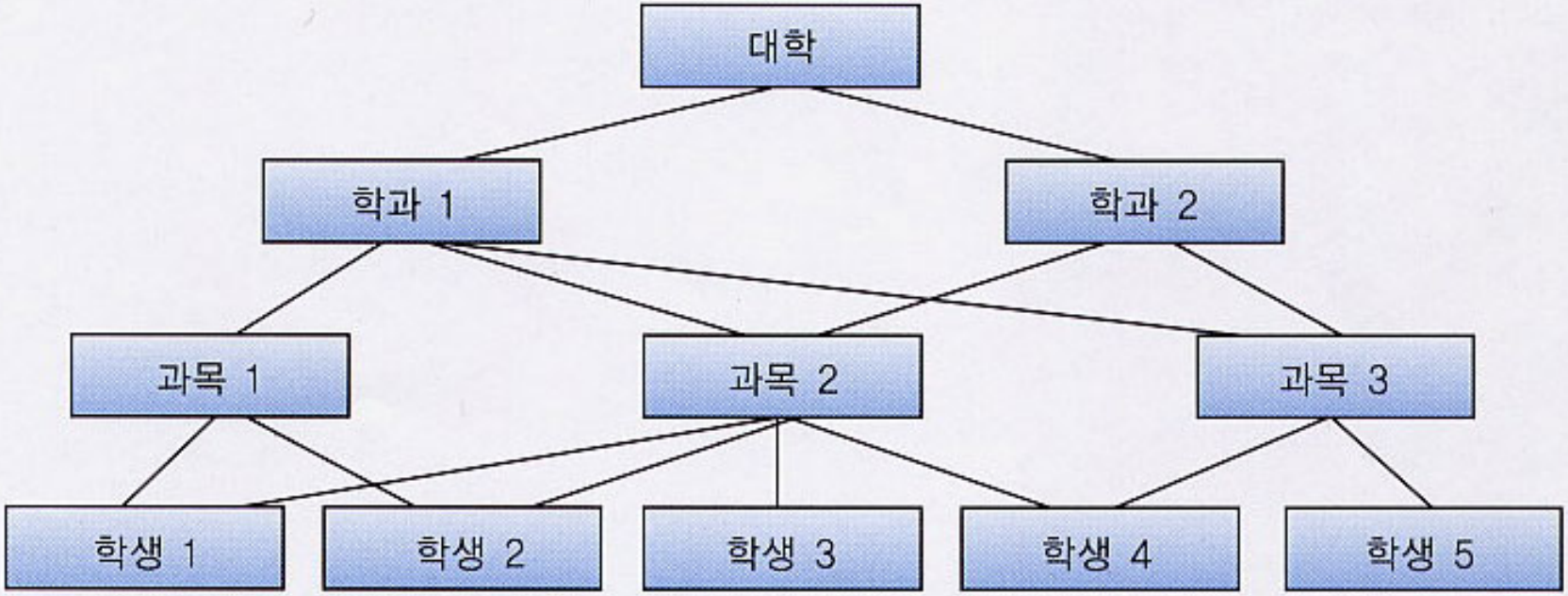


one-to-many 관계는 잘 처리하나
many-to-many 관계는 그렇지 못함

1.3 DBMS 발전 과정(계속)

- 네트워크 DBMS
  - ✓ 1960년대 초에 Charles Bachman이 하니웰(Honeywell) 사에서 최초의 네트워크 DBMS인 IDS를 개발
  - ✓ 레코드들이 노드로, 레코드들 사이의 관계가 간선으로 표현되는 그래프를 기반으로 하는 네트워크 데이터 모델을 사용
  - ✓ 네트워크 DBMS에서도 레코드들이 링크로 연결되어 있으므로 레코드 구조를 변경하기 어려움

1.3 DBMS 발전 과정(계속)



[그림 1.12] 네트워크 데이터베이스의 예

1.3 DBMS 발전 과정(계속)

- 관계 DBMS
  - ✓ 1970년에 E.F. Codd가 IBM 연구소에서 관계 데이터 모델을 제안
  - ✓ 미국 IBM 연구소에서 진행된 System R과 캘리포니아 버클리대에서 진행된 Ingres 프로젝트
  - ✓ 장점
    - 모델이 간단하여 이해하기 쉬움
    - 사용자는 자신이 원하는 것(what)만 명시하고, 데이터가 어디에 있는지, 어떻게 접근해야 하는지는 DBMS가 결정
  - ✓ 예: 오라클, MS SQL Server, Sybase, DB2, Informix 등

1.3 DBMS 발전 과정(계속)

STUDENT

NAME	ST_NO	ADDRESS	DEPT	GRADE
송치윤	52015	사당동	컴퓨터	3.3
김구완	53116	홍제동	정보통신	3.1
최재석	56034	양재동	정보관리	3.5
송혜영	52042	신정동	컴퓨터	2.9
조미림	53108	역삼동	정보통신	3.4

[그림 1.13] STUDENT 테이블

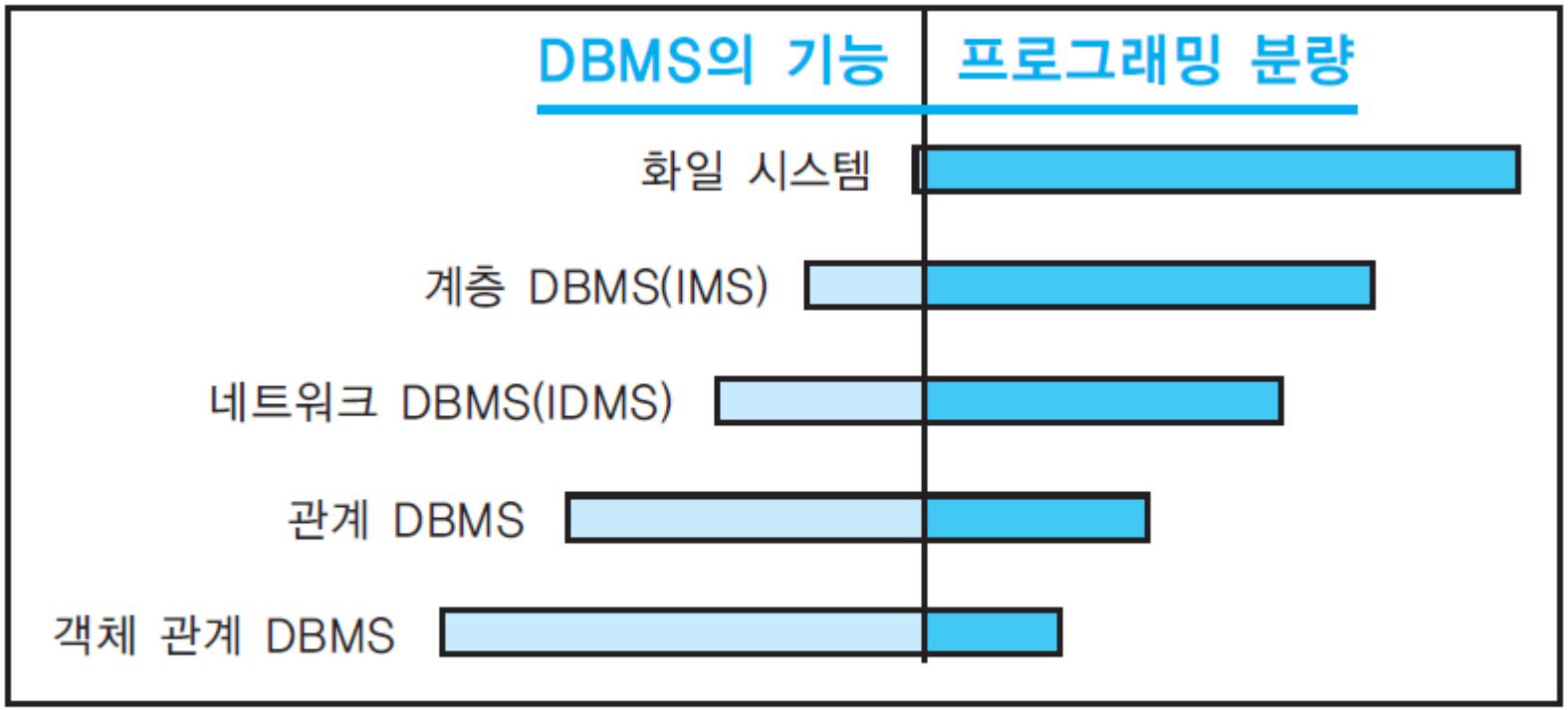
1.3 DBMS 발전 과정(계속)

- 객체 지향 DBMS
  - ✓ 1980년대 후반 들어 새로운 데이터 모델인 객체 지향 데이터 모델이 등장
  - ✓ 객체 지향 프로그래밍 패러다임을 기반으로 하는 데이터 모델
  - ✓ 장점
    - 데이터와 프로그램을 그룹화하고, 복잡한 객체들을 이해하기 쉬우며, 유지와 변경이 용이함
  - ✓ 예: ONTOS, OpenODB, GemStone, ObjectStore, Versant, O2 등
- 객체 관계 DBMS
  - ✓ 1990년대 후반에 관계 DBMS에 객체 지향 개념을 통합한 객체 관계 데이터 모델이 제안됨
  - ✓ 예: 오라클, Informix Universal Server 등

1.3 DBMS 발전 과정(계속)

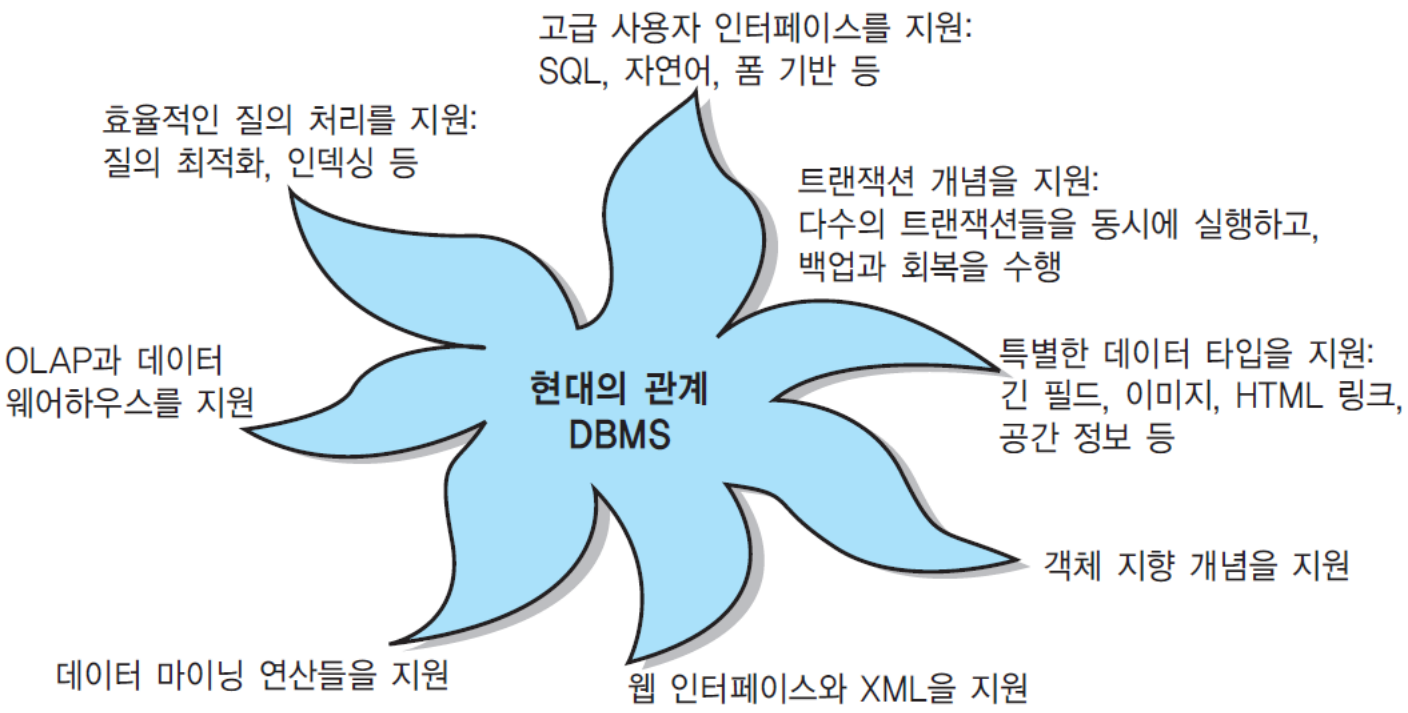
- 새로운 데이터베이스 응용
  - ✓ CAD 데이터베이스, 소프트웨어 공학 데이터베이스(재사용이 가능한 소프트웨어들의 라이브러리), 게놈 데이터베이스, 데이터 웨어하우스, 데이터 마이닝, OLAP, 멀티미디어 데이터베이스, 웹 데이터베이스 등

1.3 DBMS 발전 과정(계속)



[그림 1.14] DBMS의 데이터 모델에 따른 프로그래밍 분량

1.3 DBMS 발전 과정(계속)



[그림 1.15] 현대의 관계 DBMS의 기능

1.3 DBMS 발전 과정(계속)

〈표 1.2〉 DBMS들의 분류

기준	종류
데이터 모델에 따른 분류	• 계층 DBMS • 네트워크 DBMS • 관계 DBMS • 객체 지향 DBMS • 객체 관계 DBMS
사용자의 수에 따른 분류	• 단일 사용자 DBMS (주로 PC용) • 다수 사용자 DBMS
사이트의 수에 따른 분류	• 중앙 집중식 DBMS • 분산 DBMS
접근 방법에 따른 분류	• 범용 DBMS • 특별한 DBMS (예: 공간 DBMS)

1.4 DBMS 언어

- 데이터 정의어(DDL: Data Definition Language)
  - ✓ 사용자는 데이터 정의어를 사용하여 데이터베이스 스키마를 정의
  - ✓ 데이터 정의어로 명시된 문장이 입력되면 DBMS는 사용자가 정의한 스키마에 대한 명세를 시스템 카탈로그 또는 데이터 사전에 저장
  - ✓ 데이터 정의어의 기본적인 기능

    데이터 모델에서 지원하는 데이터 구조를 생성

    예, SQL에서 CREATE TABLE

    데이터 구조의 변경

    예, SQL에서 ALTER TABLE

    데이터 구조의 삭제

    예, SQL에서 DROP TABLE

    데이터 접근을 위해 특정 애트리뷰트 위에 인덱스를 정의

    예, SQL에서 CREATE INDEX

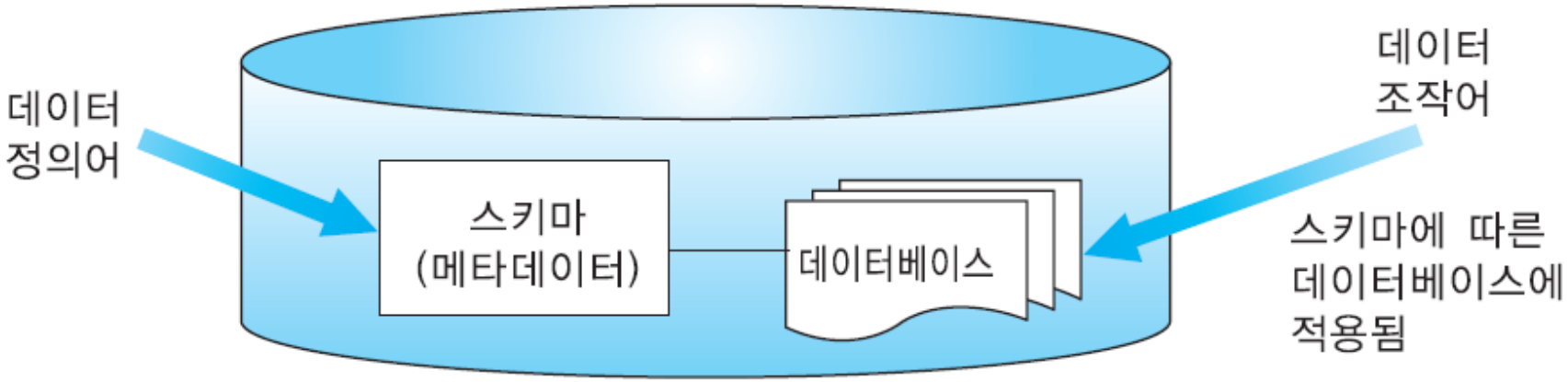
1.4 DBMS 언어(계속)

- 데이터 조작어(DML: Data Manipulation Language)
  - ✓ 사용자는 데이터 조작어를 사용하여 데이터베이스 내의 원하는 데이터를 검색하고, 수정하고, 삽입하고, 삭제
  - ✓ 절차적 언어(procedural language)와 비절차적 언어(non-procedural language)
  - ✓ 관계 DBMS에서 사용되는 SQL은 대표적인 비절차적 언어
  - ✓ 대부분의 데이터 조작어는 SUM, COUNT, AVG와 같은 내장 함수들을 갖고 있음
  - ✓ 데이터 조작어는 단말기에서 대화식으로 입력되어 수행되거나 C, 코볼 등의 고급 프로그래밍 언어로 작성된 프로그램에 내포되어 사용됨

1.4 DBMS 언어(계속)

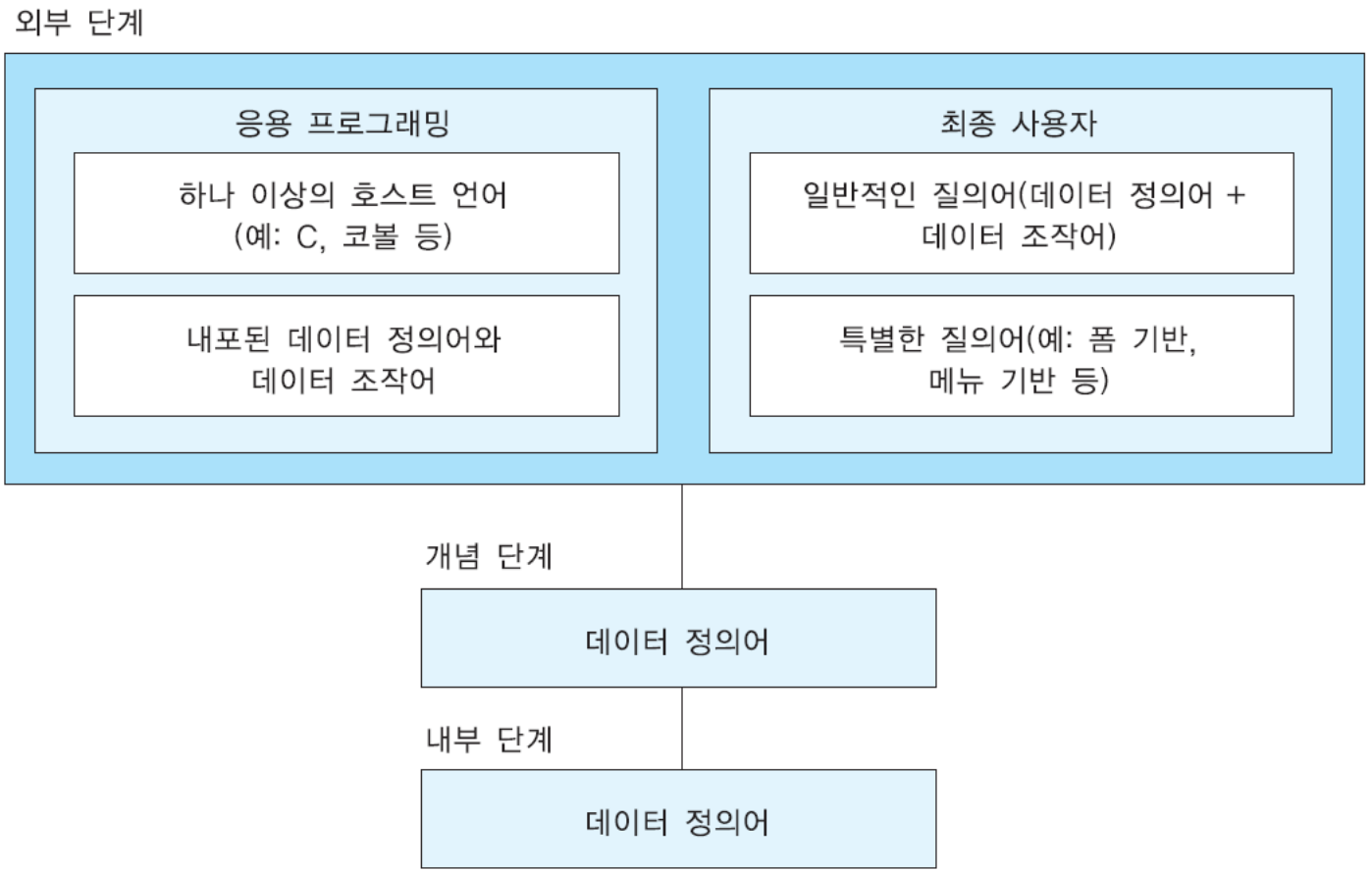
- 데이터 조작어의 기본적인 기능
  - 데이터의 검색
    - 예, SQL에서 SELECT
  - 데이터의 수정
    - 예, SQL에서 UPDATE
  - 데이터의 삭제
    - 예, SQL에서 DELETE
  - 데이터의 삽입
    - 예, SQL에서 INSERT

1.4 DBMS 언어(계속)



[그림 1.16] 데이터 정의어와 데이터 조작어

1.4 DBMS 언어(계속)

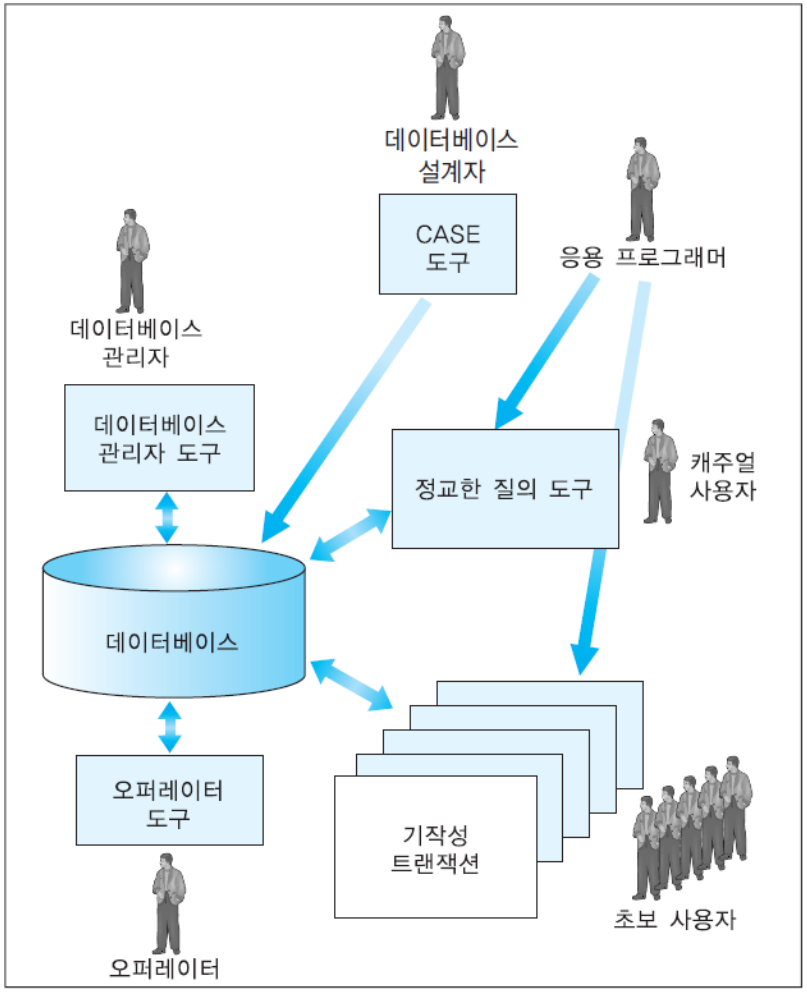


[그림 1.17] ANSI/SPARC 3단계 모델의 각 단계에서 사용되는 데이터 정의어와 데이터 조작어

1.4 DBMS 언어(계속)

- 데이터 제어어(DCL: Data Control Language)
  - ✓ 사용자는 데이터 제어어를 사용하여 데이터베이스 트랜잭션을 명시하고 권한을 부여하거나 취소

1.5 DBMS 사용자



[그림 1.18] DBMS의 사용자들

1.5 DBMS 사용자(계속)

- 데이터베이스 관리자(DBA: Database Administrator)
  - ✓ 데이터베이스 관리자는 조직의 여러 부분의 상이한 요구를 만족시키기 위해서 일관성 있는 데이터베이스 스키마를 생성하고 유지하는 사람(팀)
  - ✓ 데이터베이스 관리자의 역할
    - 데이터베이스 스키마의 생성과 변경
    - 무결성 제약조건을 명시
    - 사용자의 권한을 허용하거나 취소하고, 사용자의 역할을 관리
    - 저장 구조와 접근 방법(물리적 스키마) 정의
    - 백업과 회복
    - 표준화 시행

1.5 DBMS 사용자(계속)

- 응용 프로그래머
  - ✓ 데이터베이스 위에서 특정 응용(예, 고객 관리, 인사 관리, 재고 관리 등)이나 인터페이스를 구현하는 사람
  - ✓ 고급 프로그래밍 언어인 C, 코볼 등으로 응용 프로그램을 개발하면서 데이터베이스를 접근하는 부분은 내포된 데이터 조작어를 사용
  - ✓ 이들이 작성한 프로그램은 최종 사용자들이 반복해서 수행하므로 기작성 트랜잭션(canned transaction)이라 부름

1.5 DBMS 사용자(계속)

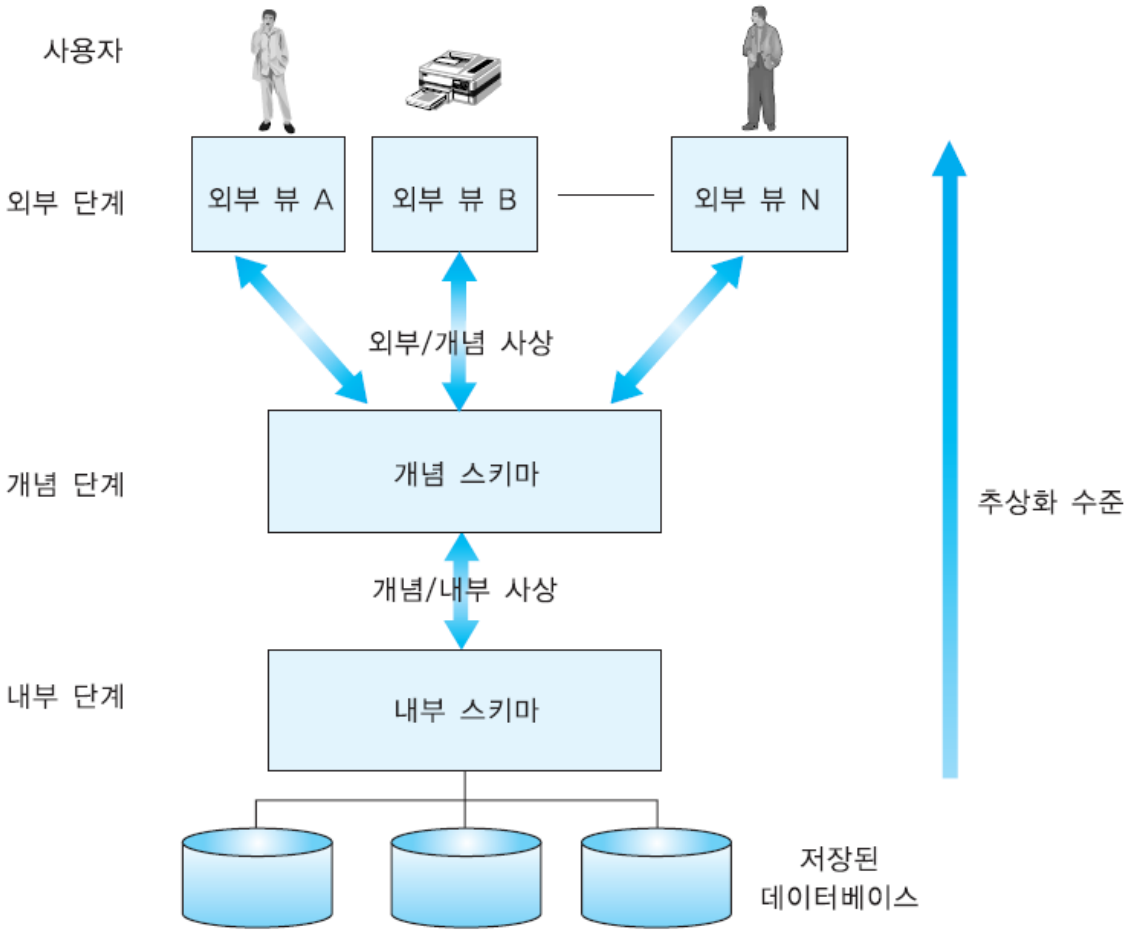
- 최종 사용자(end user)
  - ✓ 질의하거나 갱신하거나 보고서를 생성하기 위해서 데이터베이스를 사용하는 사람
  - ✓ 최종 사용자는 다시 데이터베이스 질의어를 사용하여 매번 다른 정보를 찾는 캐주얼 사용자와 기작성 트랜잭션을 주로 반복해서 수행하는 초보 사용자로 구분
- 데이터베이스 설계자(database designer)
  - ✓ ERWin 등의 CASE 도구들을 이용해서 데이터베이스 설계를 담당
  - ✓ 데이터베이스의 일관성을 유지하기 위해서 정규화를 수행
- 오퍼레이터
  - ✓ DBMS가 운영되고 있는 컴퓨터 시스템과 전산실을 관리하는 사람

1.6 ANSI/SPARC 아키텍처와 데이터 독립성

❑ ANSI/SPARC 아키텍처

- ✓ 현재의 대부분의 상용 DBMS 구현에서 사용되는 일반적인 아키텍처는 1978년에 제안된 ANSI/SPARC 아키텍처
- ✓ ANSI/SPARC 아키텍처의 3단계는 물리적, 개념적, 외부 단계로 이루어짐
  - 외부 단계(external level): 각 사용자의 뷰
  - 개념 단계(conceptual level): 사용자 공동체의 뷰
  - 내부 단계(internal level): 물리적 또는 저장 뷰

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)



[그림 1.19] ANSI/SPARC 3단계 아키텍처

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

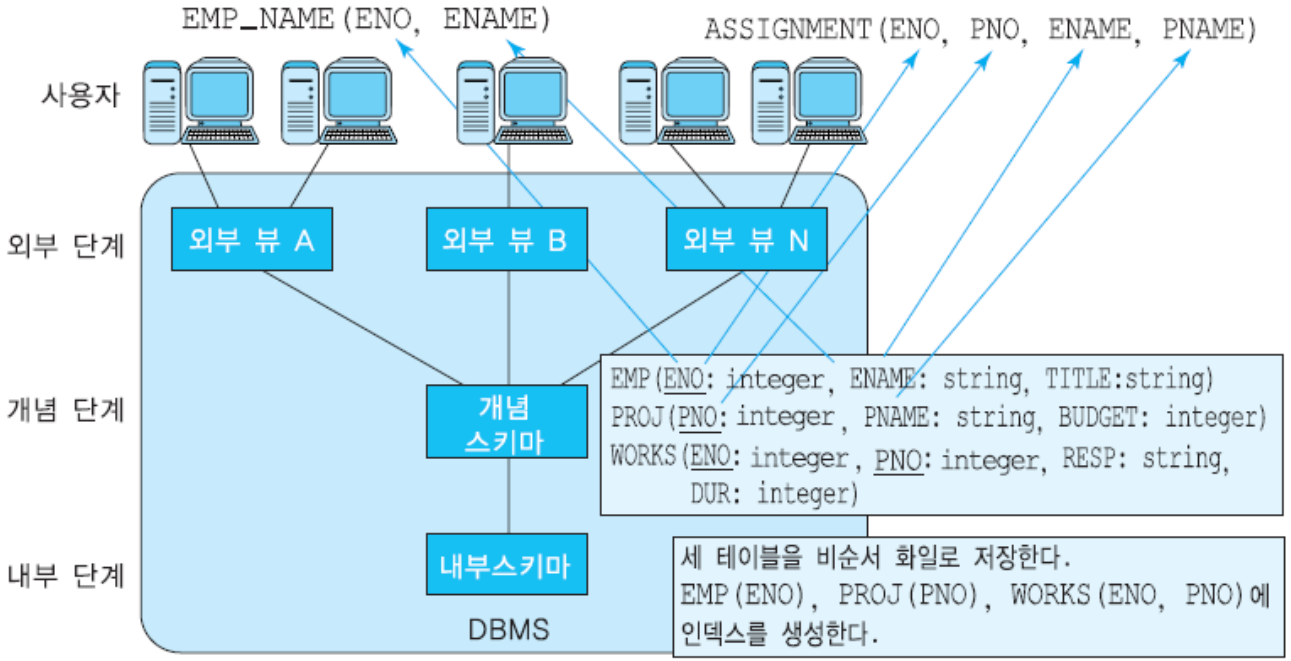
❑ 외부 단계

- ✓ 데이터베이스의 각 사용자가 갖는 뷰
- ✓ 여러 부류의 사용자를 위해 동일한 개념 단계로부터 다수의 서로 다른 뷰가 제공될 수 있음
- ✓ 일반적으로, 최종 사용자와 응용 프로그래머들은 데이터베이스의 일부분에만 관심을 가짐

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

예 :

그림 1.20은 관계 데이터 모델에서 개념 단계의 세 테이블 EMP, PROJ, WORKS로부터 일부 애트리뷰트들을 선택하여 외부 단계의 뷰 ASSIGNMENT와 EMP_NAME이 정의된 예를 보여준다.



[그림 1.20] 관계 데이터 모델에서 3단계 아키텍처의 예

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

❑ 개념 단계

- ✓ 조직체의 정보 모델로서, 물리적인 구현은 고려하지 않으면서 조직체 전체에 관한 스키마를 포함
- ✓ 데이터베이스에 어떤 데이터가 저장되어 있으며, 데이터 간에는 어떤 관계가 존재하고, 어떤 무결성 제약조건들이 명시되어 있는가를 기술함
- ✓ 데이터베이스에 대한 사용자 공동체의 뷰를 나타냄
- ✓ 데이터베이스마다 오직 한 개의 개념 스키마가 존재

예 :

그림 1.19에서는 개념 단계가 세 테이블 EMP, PROJ, WORKS로 구성되어 있다.

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

❑ 내부 단계

- ✓ 실제의 물리적인 데이터 구조에 관한 스키마
- ✓ 데이터베이스에 어떤 데이터가 어떻게 저장되어 있는가를 기술함
- ✓ 인덱스, 해싱 등과 같은 접근 경로, 데이터 압축 등을 기술함
- ✓ 데이터베이스의 개념 스키마에는 영향을 미치지 않으면서 성능을 향상시키기 위해 내부 스키마를 변경하는 것이 바람직
- ✓ 내부 단계 아래는 물리적 단계
- ✓ 물리적 단계는 DBMS의 지시에 따라 운영 체제가 관리함

예 :

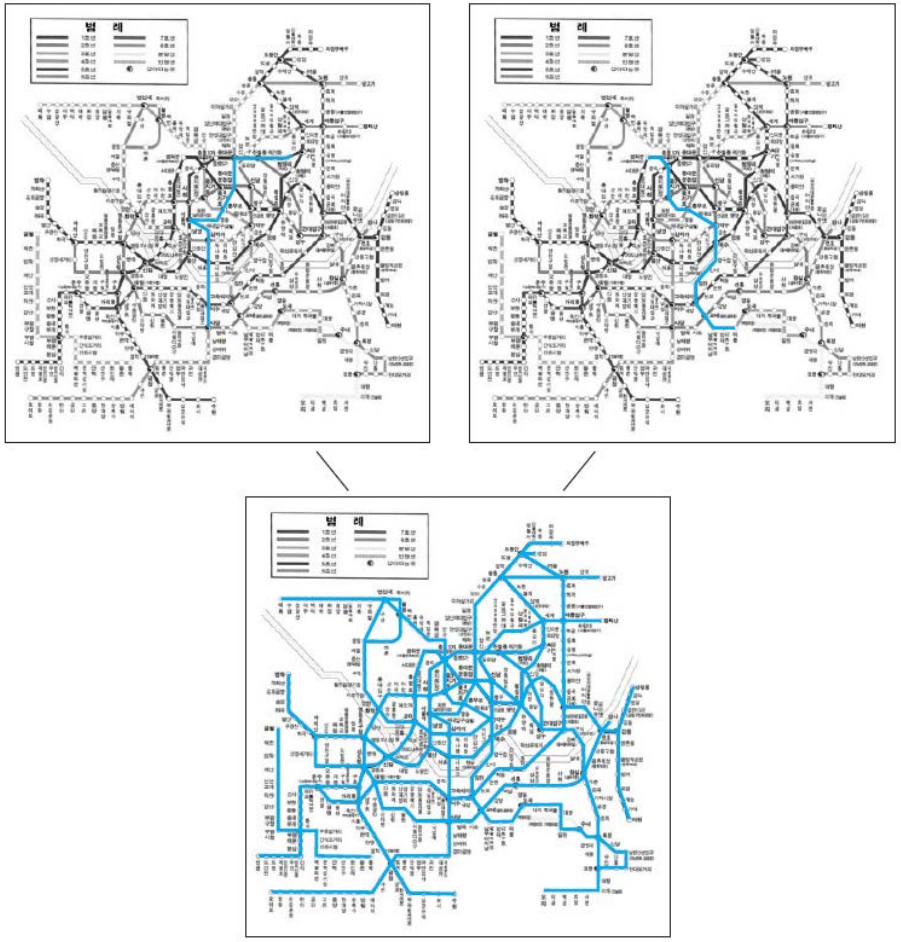
그림 1.19의 내부 단계에서는 EMP 테이블의 ENO 애트리뷰트, PROJ 테이블의 PNO 애트리뷰트, WORKS 테이블의 (ENO, PNO) 애트리뷰트에 인덱스가 정의되어 있으며 세 테이블이 모두 비순서 화일로 저장되어 있음을 알 수 있다.

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

전체 지하철 노선도에 대해 사당동에 사는 학생이 청량리에 있는 학교에 통학하기 위해서 사당역, 동대문역, 청량리역에만 관심을 갖는다.

전체 지하철 노선도에 대해 양재동에 사는 직장인이 광화문에 있는 사무실에 출퇴근하기 위해서 양재역, 종로3가역, 광화문역에만 관심을 갖는다.

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)



[그림 1.21] 전철 노선도

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

❑ 스키마 간의 사상

- ✓ DBMS는 세 가지 유형의 스키마 간의 사상을 책임짐
- ✓ 외부/개념 사상(external/conceptual mapping)
  - ▪ 외부 단계의 뷰를 사용해서 입력된 사용자의 질의를 개념 단계의 스키마를 사용한 질의로 변환
- ✓ 개념/내부 사상(conceptual/internal mapping)
  - ▪ 이를 다시 내부 단계의 스키마로 변환하여 디스크의 데이터베이스를 접근

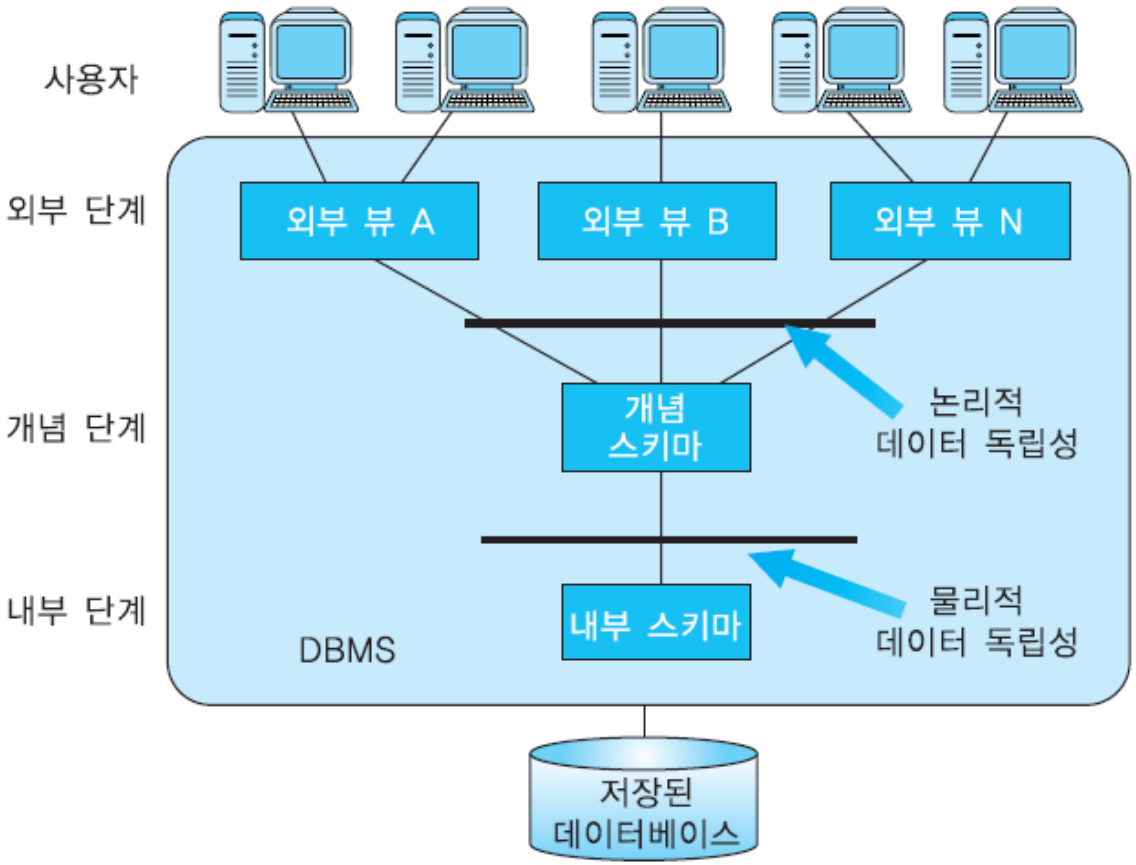
1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

❑ 데이터 독립성

✓ 상위 단계의 스키마 정의에 영향을 주지 않으면서 어떤 단계의 스키마 정의를 변경할 수 있음을 의미

- 논리적인 데이터 독립성(logical data independence)
- 물리적인 데이터 독립성(physical data independence)

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)



[그림 1.22] 논리적 데이터 독립성과 물리적 데이터 독립성

1.6 ANSI/SPARC 아키텍처와 데이터 독립성(계속)

❑ 논리적 데이터 독립성

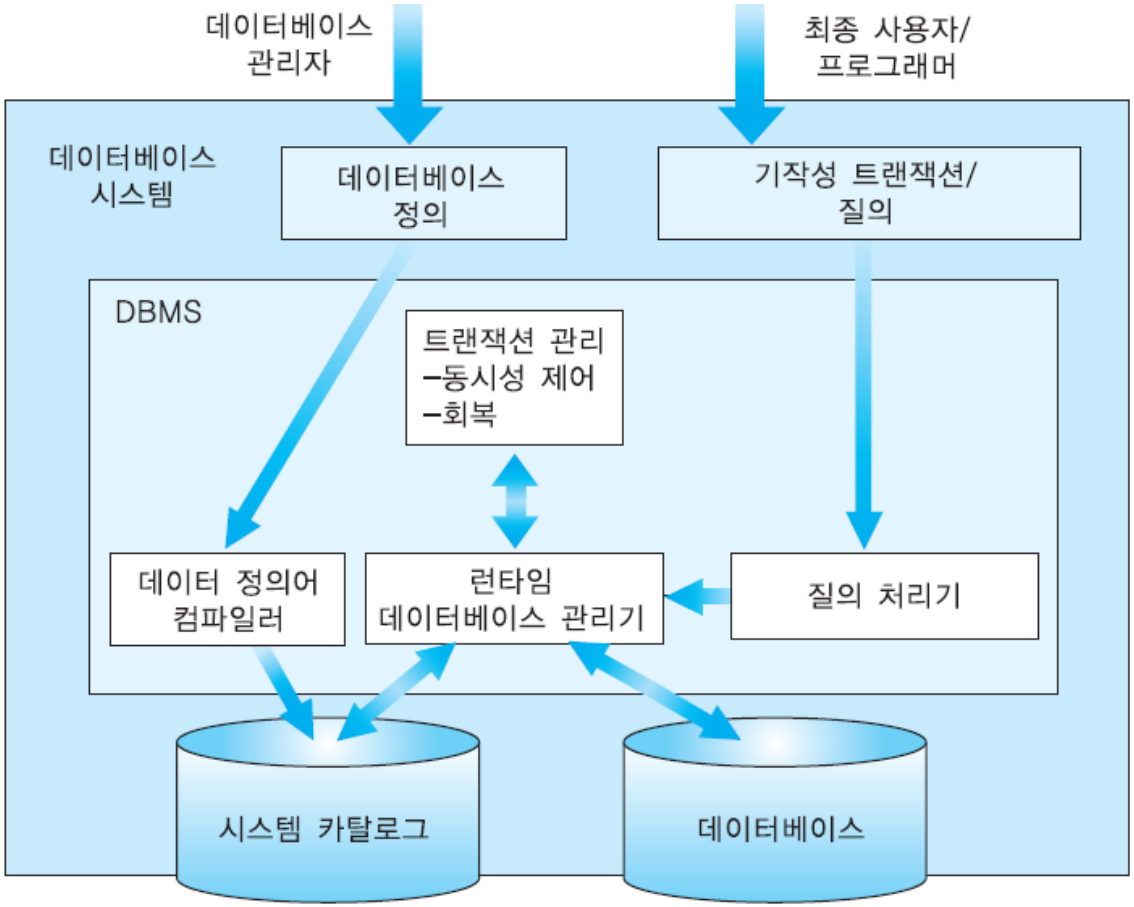
- ✓ 개념 스키마의 변화로부터 외부 스키마가 영향을 받지 않음을 의미
- ✓ 기존의 외부 스키마에 영향을 미치지 않고, 응용 프로그램을 다시 작성할 필요 없이 개념 스키마에 대한 변화가 가능해야 함

❑ 물리적 데이터 독립성

- ✓ 내부 스키마의 변화가 개념적 스키마에 영향을 미치지 않으며, 따라서 외부 스키마(또는 응용 프로그램)에도 영향을 미치지 않음을 의미
- ✓ 내부 스키마의 변화의 예:

  화일의 저장 구조를 바꾸거나 인덱스를 생성 및 삭제

1.7 데이터베이스 시스템 아키텍처



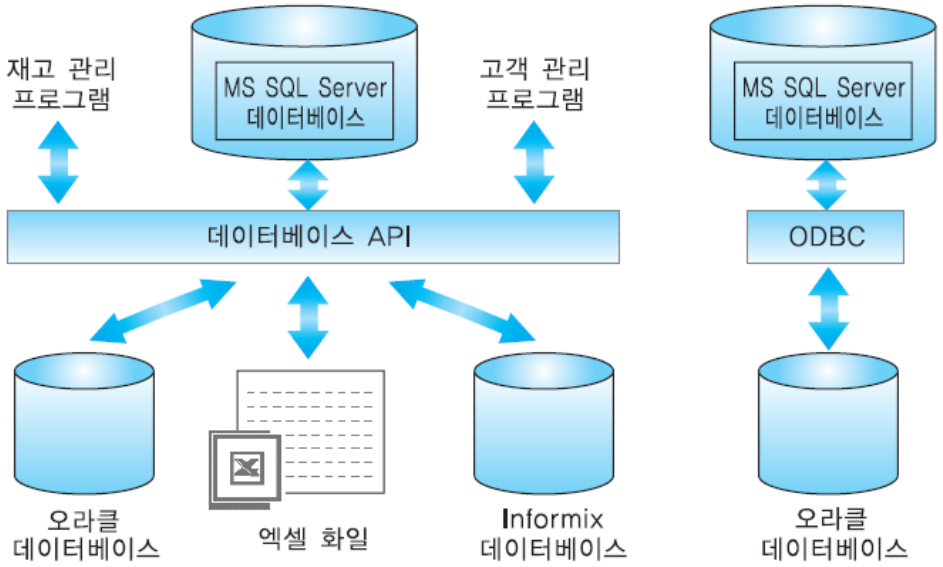
[그림 1.23] 데이터베이스 시스템의 간단한 아키텍처

1.7 데이터베이스 시스템 아키텍처(계속)

- 데이터 정의어 컴파일러(DDL compiler) 모듈
  - ✓ 데이터 정의어를 사용하여 테이블 생성을 요청하면 테이블을 파일 형태로 데이터베이스에 만들고, 이 테이블에 대한 명세를 시스템 카탈로그에 저장
- 질의 처리기(query processor) 모듈
  - ✓ 데이터 조작어를 수행하는 최적의 방법을 찾는 모듈을 통해서 기계어 코드로 번역
- 런타임 데이터베이스 관리기(run-time database manager) 모듈
  - ✓ 디스크에 저장된 데이터베이스를 접근
- 트랜잭션 관리(transaction management) 모듈
  - ✓ 동시성 제어(concurrency control) 모듈
  - ✓ 회복(recovery) 모듈

1.7 데이터베이스 시스템 아키텍처(계속)

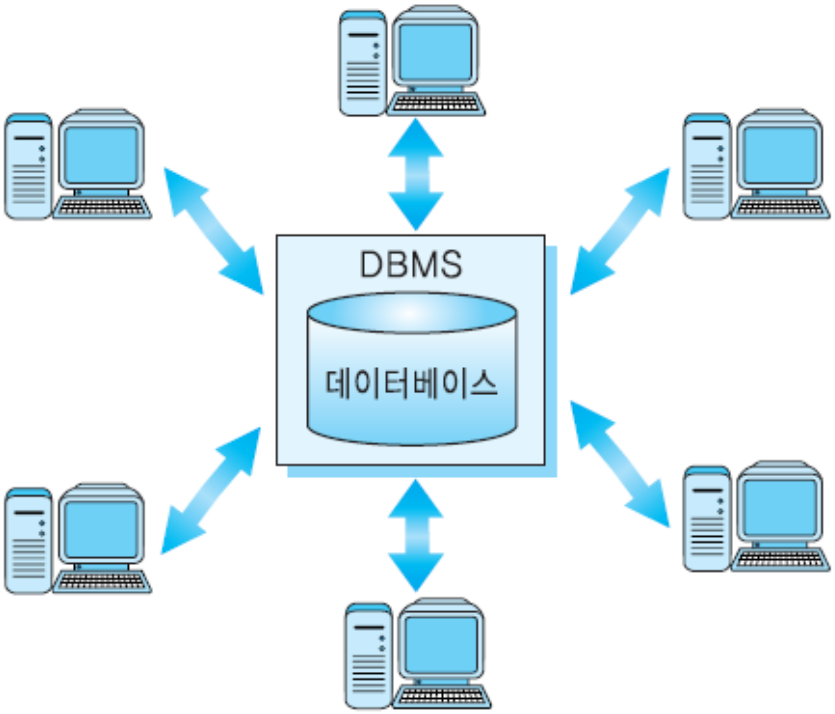
- 데이터베이스 API(Application Program Interface)
  - ✓ ODBC(Open Database Connectivity)는 마이크로소프트 사가 주도적으로 개발한 데이터베이스 API
  - ✓ ODBC를 지원하는 DBMS 간에는 서로 상대방의 데이터베이스를 접근할 수 있음



[그림 1.24] 데이터베이스 API(ODBC)의 역할

1.7 데이터베이스 시스템 아키텍처(계속)

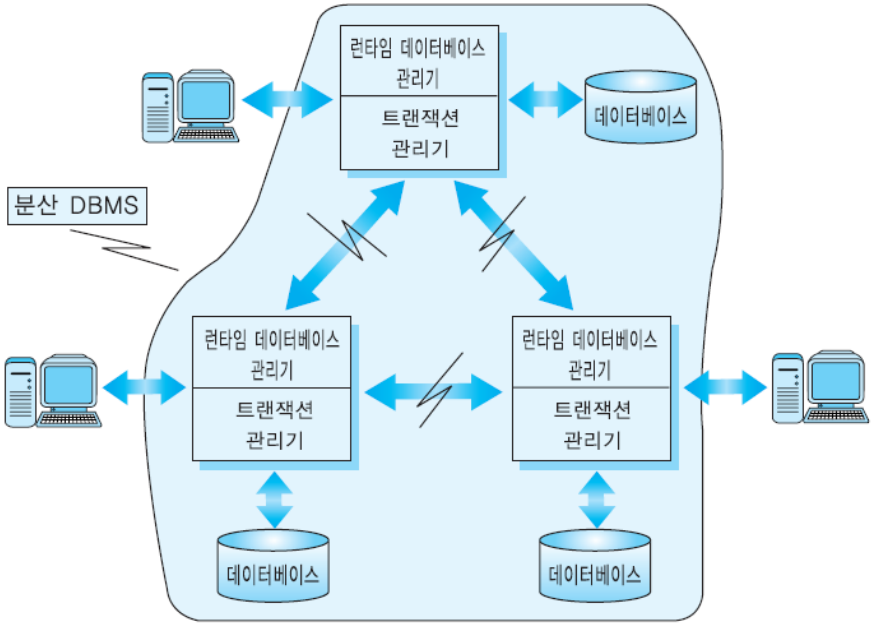
- 중앙 집중식 데이터베이스 시스템(centralized database system)
  - ✓ 데이터베이스 시스템이 하나의 컴퓨터 시스템에서 운영됨



[그림 1.25] 중앙 집중식 데이터베이스 시스템

1.7 데이터베이스 시스템 아키텍처(계속)

- 분산 데이터베이스 시스템(distributed database system)
  - ✓ 네트워크로 연결된 여러 사이트에 데이터베이스 자체가 분산되어 있으며, 데이터베이스 시스템도 여러 컴퓨터 시스템에서 운영됨
  - ✓ 사용자는 다른 사이트에 저장된 데이터베이스도 접근할 수 있음

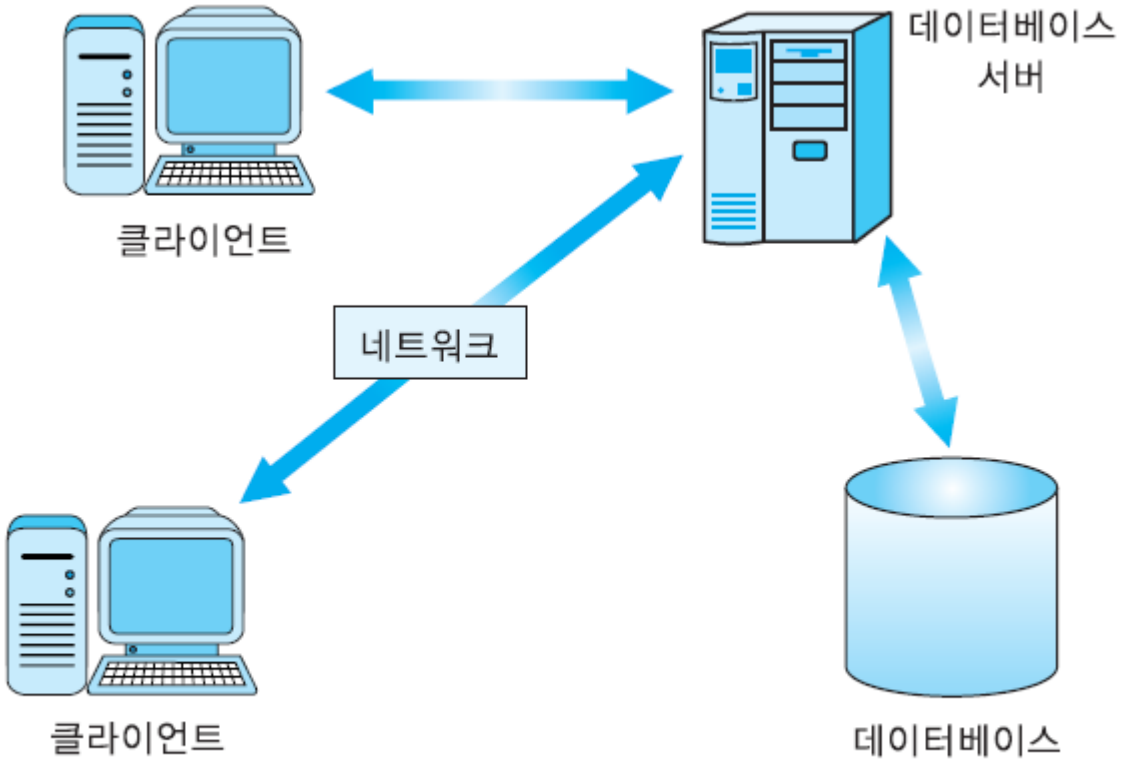


[그림 1.26] 분산 데이터베이스 시스템

1.7 데이터베이스 시스템 아키텍처(계속)

- 클라이언트-서버 데이터베이스 시스템(client-server database system)
  - ✓ PC 또는 워크스테이션처럼 자체 컴퓨팅 능력을 가진 클라이언트를 통해 데이터베이스 서버를 접근
  - ✓ 데이터베이스가 하나의 데이터베이스 서버에 저장되어 있음
  - ✓ 데이터베이스 시스템의 기능이 서버와 클라이언트에 분산됨
  - ✓ 서버는 데이터베이스를 저장하고 DBMS를 운영하면서 여러 클라이언트에서 온 질의를 최적화하고, 권한 검사를 수행하고, 동시성 제어와 회복 기능을 수행하고, 데이터베이스의 무결성을 유지하고, 데이터베이스 접근을 관리
  - ✓ 클라이언트는 사용자 인터페이스를 관리하고 응용들을 수행

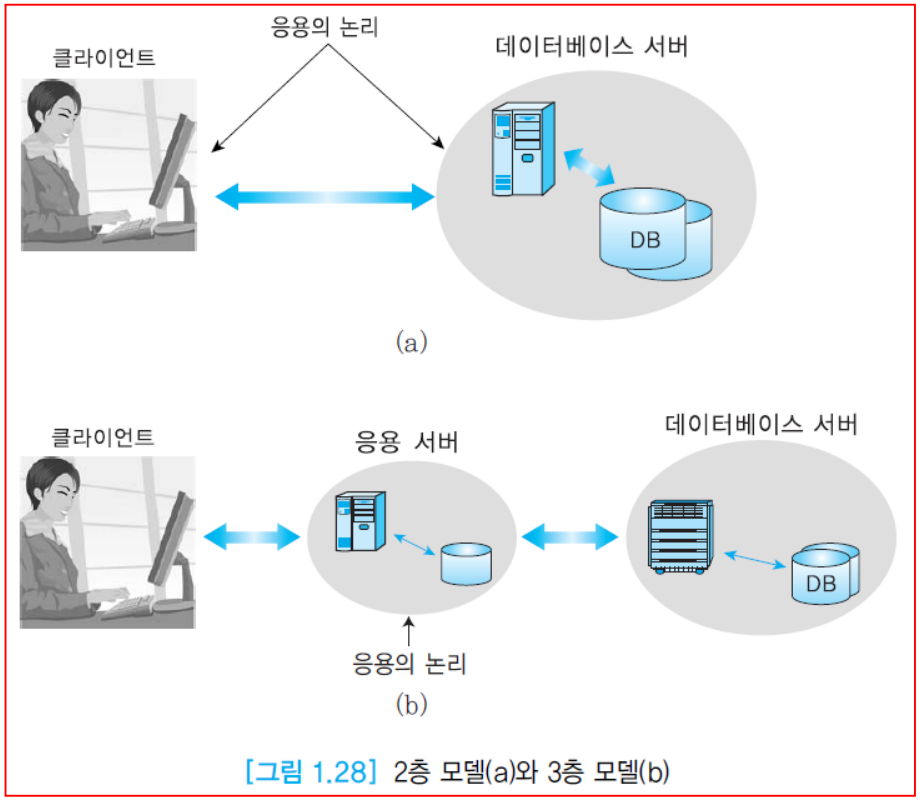
1.7 데이터베이스 시스템 아키텍처(계속)



[그림 1.27] 클라이언트-서버 데이터베이스 시스템

1.7 데이터베이스 시스템 아키텍처(계속)

- 2층 모델(2-tier model)
  - ✓ 클라이언트와 데이터베이스 서버가 직접 연결됨
- 3층 모델(3-tier model)
  - ✓ 클라이언트와 데이터베이스 서버 사이에 응용 서버가 추가됨



[그림 1.28] 2층 모델(a)와 3층 모델(b)

1.7 데이터베이스 시스템 아키텍처(계속)

- 클라이언트-서버 데이터베이스 시스템의 장점
  - ✓ 데이터베이스를 보다 넓은 지역에서 접근할 수 있음
  - ✓ 다양한 컴퓨터 시스템을 사용할 수 있음

- 클라이언트-서버 데이터베이스 시스템의 단점
  - ✓ 보안이 다소 취약할 수 있음