데이터베이스 시스템

Saerom Park

secure.psr@gmail.com

데이터베이스 개요 [1/2]

❖ 데이터베이스 (Database)

- 서로 연관이 있는 데이터들의 모임
- 데이터베이스는 어떤 **특정한 의미**를 지니는 데이터의 모임
  - 임의로 모은 데이터는 데이터베이스라고 부르지 않음
- 데이터베이스는 **특정한 목적**을 위하여 설계, 구축, 운용
  - 사용자 그룹, 원하는 응용 들이 미리 정해져 있음

❖ 데이터베이스 관리 시스템 (DBMS: Database Management System)

- 사용자가 데이터베이스를 **생성**하고 **관리**할 수 있도록 하는 컴퓨터화된 시스템
  - 데이터베이스를 정의, 생성, 조작, 공유할 수 있도록 편리한 기능을 제공하는 범용 소프트웨어 시스템
- 데이터베이스 정의
  - 데이터 타입, 구조, 제약조건(constraint)들을 명세하는 과정
  - **메타데이터(meta-data)**: 데이터베이스를 설명하는 정보를 DBMS에 의해 카탈로그나 사전의 형태로 저장

데이터베이스 개요 [2/2]

❖ DBMS 의 주요기능

- 데이터베이스 조작 (manipulation)
  - 특정한 데이터를 검색하기 위한 질의, 데이터베이스의 갱신, 데이터로부터 리포트를 생성
- 데이터베이스 공유
  - 여러사용자와 프로그램이 데이터베이스에 **동시에 접근**하도록 하는 기능
  - 질의(query): 요청에 따라 일부 데이터를 결과로 반환
  - 트랜잭션(transaction): 하나의 논리적 함수를 수행하는 연산들의 모음
  - 응용프로그램 (application program) 은 DBMS에 질의 또는 데이터 요청을 보내서 데이터베이스에 접근
- 데이터베이스 보호 및 유지보수
  - 하드웨어 또는 소프트웨어 오동작 또는 붕괴로부터 시스템을 보호
  - 권한이 없는 또는 악의적인 접근을 하려는 **보안 위협으로부터 보호**하는 기능
  - 시간이 지남에 따라 **변화하는 요구사항을 반영**할 수 있도록 데이터베이스 시스템을 유지보수

데이터베이스 특징 [1/4]

❖ 파일 처리 시스템의 한계

- 소프트웨어 응용을 프로그래밍하는 작업의 일환으로 사용자가 특정한 소프트웨어 응용을 위하여 필요한 파일들을 **별도로 정의하고 구현**함
- 데이터의 중복과 비일관성과 같은 문제 발생 가능

❖ 데이터베이스 방식이 갖는 특징

- 데이터베이스 방식에서는 **단일 저장소**에 데이터가 저장되고 여러 사용자가 **질의, 트랜잭션, 응용프로그램**을 통하여 데이터를 공유
  - 데이터베이스 시스템의 자기 기술성
  - 프로그램과 데이터의 격리 및 데이터 추상화
  - 데이터에 대한 다중 뷰(view)의 제공
  - 데이터의 공유와 다수 사용자 트랜잭션 처리

데이터베이스 특징 [2/4]

❖ 데이터베이스 시스템의 자기 기술성

- 데이터베이스 시스템이 **데이터베이스 자체** 뿐만 아니라 **데이터 베이스의 구조와 제약 조건**에 대한 완전한 정의를 포함
- 데이터베이스 정의: 각 파일들의 구조, 각 데이터 항목의 타입과 저장형식, 다양한 제약 조건을 포함
  - DBMS의 카탈로그(catalog)에 저장
  - 메타데이터(meta-data): 카탈로그에 저장된 정보
- 기존의 파일처리 방식에서는 데이터의 정의가 대개 **응용프로그램의 일부**로 포함되기 때문에 프로그램 내에서 그 구조가 정의된 하나의 **특정한 데이터베이스**에 대해서만 동작할 수 있음

데이터베이스 특징 [3/4]

❖ 프로그램과 데이터의 격리 및 데이터 추상화

- 프로그램 데이터 독립성: 데이터 파일이 변경되어도 응용프로그램은 거의 변하지 않게 됨
- 프로그램 연산 독립성: 연산의 구현과 연산의 인터페이스를 분리
- 데이터 추상화

▶ 물리적 단계 (physical level)

- 데이터가 실제로 어떻게 저장되는지 기술

▶ 논리적 단계 (logical level)

- 어떤 데이터가 저장되었는지, 데이터들 사이에 어떤 관계가 있는지를 기술
- 물리적 데이터 독립성 (physical data independence)
- 데이터 베이스 관리자 (database administrator: DBA)

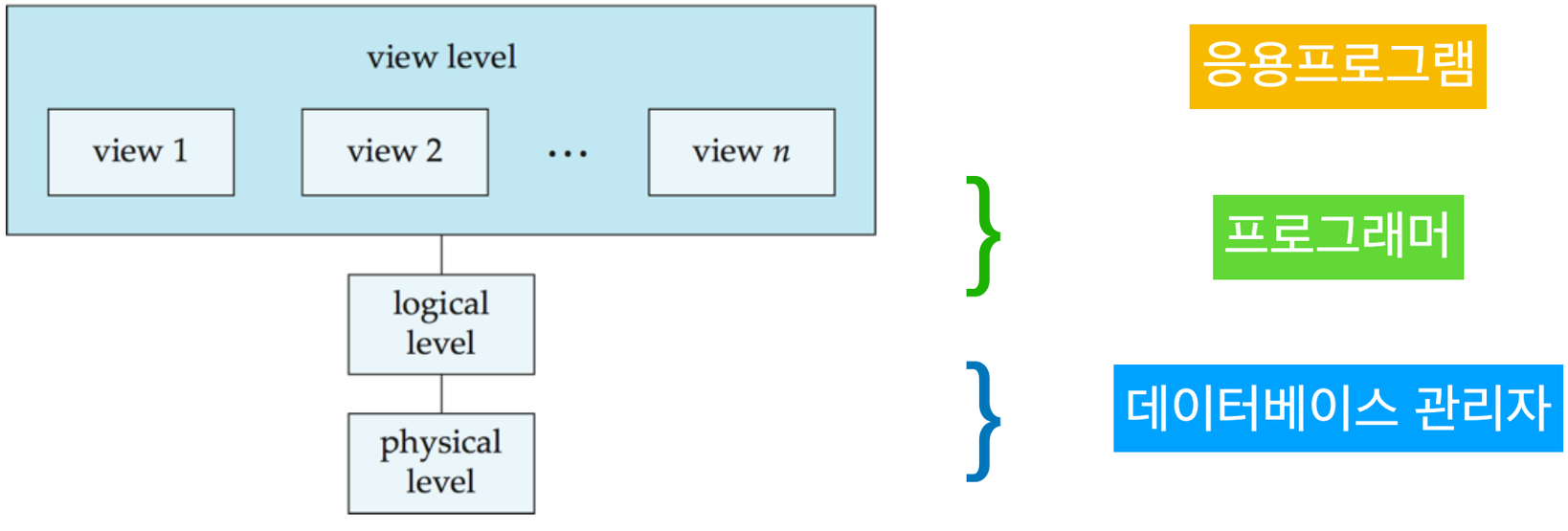
▶ 뷰 단계 (view level)

- 추상화의 최상위 단계. 요청에 따라 데이터 베이스의 일부분만을 기술
- 응용 프로그램 사용

데이터베이스 특징 [3/4]

❖ 프로그램과 데이터의 격리 및 데이터 추상화

- 프로그램 데이터 독립성: 데이터 파일이 변경되어도 응용프로그램은 거의 변하지 않게 됨
- 프로그램 연산 독립성: 연산의 구현과 연산의 인터페이스를 분리
- 데이터 추상화
  - 물리적 단계 (physical level)
    - 데이터가 실제로 어떻게 저장되는지 기술
  - 논리적 단계 (logical level)
    - 어떤 데이터가 저장되었는지, 데이터들 사이에 어떤 관계가 있는지를 기술
    - 물리적 데이터 독립성 (physical data independence)
    - 데이터 베이스 관리자 (database administrator: DBA)
  - 뷰 단계 (view level)
    - 추상화의 최상위 단계. 요청에 따라 데이터 베이스의 일부분만을 기술
    - 응용 프로그램 사용



데이터베이스 특징 [4/4]

❖ 데이터에 대한 다중 뷰의 제공

- 데이터베이스 사용자는 **서로 다른 관점(뷰)**을 가지고 데이터를보게 됨
- 뷰(view): 데이터베이스의 일부이거나 데이터 베이스로부터 유도되는 가상데이터(virtual data)
  - 데이터베이스에 실제로 저장되지는 않음
- 다수 사용자용 DBMS는 여러사용자들이 자신의 뷰를 정의할 수 있도록 하는 기능을 제공해야 함

❖ 데이터의 공유와 다수 사용자 트랜잭션 처리

- 다수 사용자용 DBMS: **여러 사용자**가 **동시에** 데이터베이스를 접근할 수 있는 DBMS
- 동시성 제어(concurrency control): 다수 사용자가 **동일한 데이터**를 **동시에** 변경하는 경우에도 데이터의 **일관성(consistency)**을 보장
  - 온라인 트랜잭션 처리(OLTP: On-Line Transaction Processing): 좌석 예약과같은 응용
- 동시에 수행되는 트랜잭션들이 상호 방해를 받지 않고 **정확하고 효율적으로** 수행되도록 보장 필요
- 트랜잭션(transaction): 데이터베이스 레코드를 읽거나 갱신하는것과 같이 **한 번 이상의 데이터베이스 접근**을 포함하는 프로그램 혹은 프로세스를 수행하는 것
  - **고립성(isolation)**: 다른 트랜잭션들과 고립되어 수행되는 것처럼 보이도록 보장
  - **원자성(atomicity)**: 트랜잭션 내의 모든 데이터베이스 연산들이 수행 완료되거나 아무 연산도 수행 안됨

데이터베이스 사용자 [1/2]

❖ 데이터 베이스 사용자 분류

- 대규모 데이터베이스에서는 데이터베이스를 정의, 구축, 관리, 조작하는데 많은 사람들이 관련됨

❖ 데이터베이스 관리자 (DBA: Database Administrator)

- 데이터베이스 자체, DBMS, 관련 소프트웨어를 관리하고 감독
- 데이터베이스에 대한 접근을 감독, 자원의 사용을 모니터링하고 조정
- 필요한 경우 소프트웨어나 하드웨어 자원을 구입
- 데이터베이스 보안이나 시스템의 성능 문제 책임짐

❖ 데이터베이스 설계자 (Database designer)

- 데이터베이스에 저장될 데이터를 선정하고, 데이터를 나타내고 저장하는 구조를 정의하는 역할 담당
- 사용자로부터 요구사항을 도출해 내고 그것이 만족되도록 데이터 베이스 설계
- 사용자 그룹들과 만나 요구사항을 도출하고 그것을 만족하는 데이터베이스 뷰를 개발
- 각각의 뷰는 다른 사용자 그룹들의 데이터 베이스 뷰와 통합되어 전체 데이터 베이스 구조가 완성

데이터베이스 사용자 [2/2]

❖ 최종 사용자 (end user)

- 데이터베이스에 대해 질의하고, 변경하고, 보고서를 작성하는 사람
- 캐주얼 사용자(casual end user)
  - 데이터베이스를 가끔 접근하지만 매번 다른 정보를 요구
  - 정교한 데이터베이스 질의어 인터페이스 사용
- 초보 사용자 (naive or parametric end user)
  - 미리 잘 프로그램되고 테스트된 기작성 트랜잭션 (canned transaction)들을 사용하여 데이터베이스를 검색하고 변경하는 사용자
- 전문 사용자 (sophisticated end user)
  - 복잡한 요구사항을 가진 엔지니어, 과학자, 비즈니스 분석가 등으로 복잡한 요구사항을 만족시키도록 DBMS 의 고급 기능을 이용하여 응용 프로그램을구현
- 독자적인 사용자 (stand-alone end user)
  - 메뉴나 그래픽 사용자 인터페이스를 제공하는 편리한 패캐지를 이용하여 개인 데이터 베이스를 유지하는 사용자

❖ 시스템 분석가 및 응용프로그래머 (소프트웨어 공학자)

- 사용자(특히 초보 사용자)의 요구 사항을 분석한 다음 이를 만족시키는 미리 작성된 표준 트랜잭션들의 명세를 설계
- 응용프로그래머: 트랜잭션들의 명세를 프로그램으로 구현, 테스트, 에러수정, 문서화 등의 작업 수행 및 유지관리

데이터베이스를 사용하지 않아도 좋은 경우

❖ 데이터베이스 사용이 부적합한 경우

- 데이터 베이스와 응용이 단순하고 잘 정의되어 있으며 전혀 변경이 일어나지 않는 경우
- 실시간 요구사항이 업격한 응용프로그램에서, DBMS 오버헤드로 인하여 실시간 요구가 만족되지 못하는 경우
- 제한된 저장 용량을 가진 내장형 시스템에 범용 DBMS가 탑재되지 못하는 경우
- 단일 사용자만이 데이터베이스를 접근하는 경우

데이터베이스 시스템 개념과 아키텍처

데이터 모델

❖ 데이터 추상화 (data abstraction)

- 데이터 저장 구조와 저장 장치의 세부 사항을 감추고 데이터를 잘 이해하는데 도움되는 필수적인 특징들은 강조하는 것
  - 데이터베이스 방식의 중요한 특징은 데이터 추상화를 제공함으로써 여러 사용자들이 각자 **선호하는 상세 수준**에서 데이터를 볼 수 있도록 하는 것에 있음

❖ 데이터 모델의 분류

- 저수준/물리적 데이터 모델
  - 데이터가 어떻게 컴퓨터의 저장 장치, 특히 하드디스크에 저장되는지에 대한 세부사항을 명시하는 개념을 제공
- 표현/구현 데이터 모델
  - 일반 사용자들이 쉽게 이해할 수 있는 개념을 제공하지만 컴퓨터 저장장치에서 데이터가 구성되는 방식과 완전히 무관한 것은 아님
- 고수준/개념적 데이터 모델
  - 많은 사용자들이 데이터를 인식하는 방식에 대한 개념을 제공
  - 엔터티(entity): 데이터베이스에 표현되는 실세계의 객체나 개념
  - 애트리뷰트(attribute): 엔티티를 기술하는 특성
  - 관계(relationship): 엔티티들 사이의 연관성

데이터베이스 시스템 개념 [1/2]

❖ 데이터베이스 스키마 (database schema)

- 데이터 베이스의 기술로 데이터 베이스 **설계 과정에서 명시**하며 자주 변경되지 않음
- 스키마 다이어그램 (schema diagram): 도식화된 스키마
  - 레코드들의 실제 인스턴스는 표시하지 않고 각 레코드 타입의 **구조를 나타냄**
  - 각 항목의 데이터 타입은 표현되지 않음
  - 많은 양의 제약 조건이 스키마 다이어그램으로 나타나기 어려움
- 스키마 구조물 (schema construct): 스키마의 각 객체

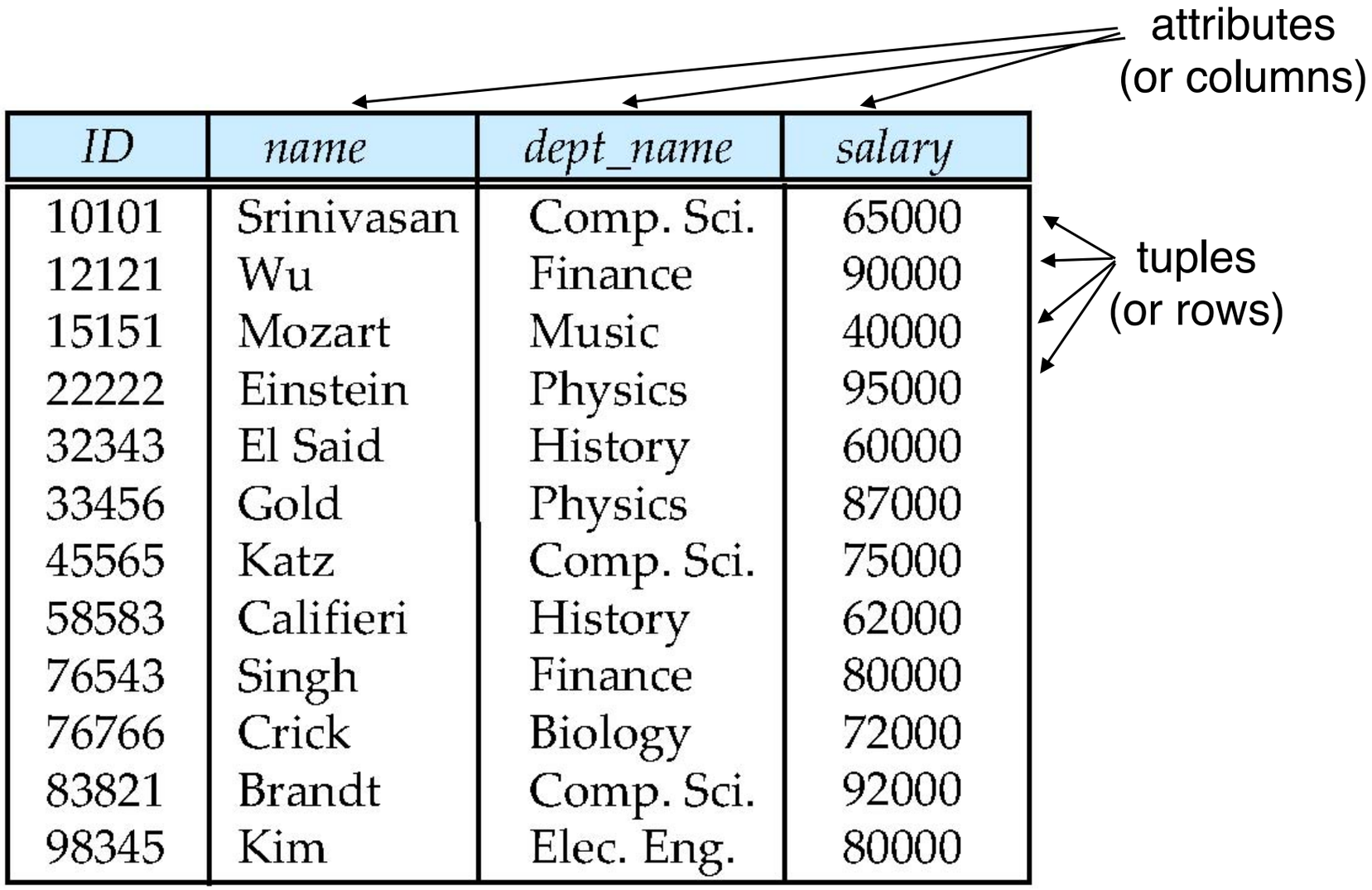
❖ 데이터베이스 상태 (스냅샷: snapshot)

- 어떤 **특정 시점**에 데이터베이스에 들이 있는 데이터
- 데이터베이스 내의 **인스턴스(instance)들의 현재 집합**
- 데이터베이스 상태들은 특정 데이터베이스 스키마와 대응됨
  - 레코드를 삽입하거나 삭제하고 레코드 내의 데이터 항목의 값을 변경할 때마다 데이터베이스 상태가 변화함

❖ 인스턴스 (instance)

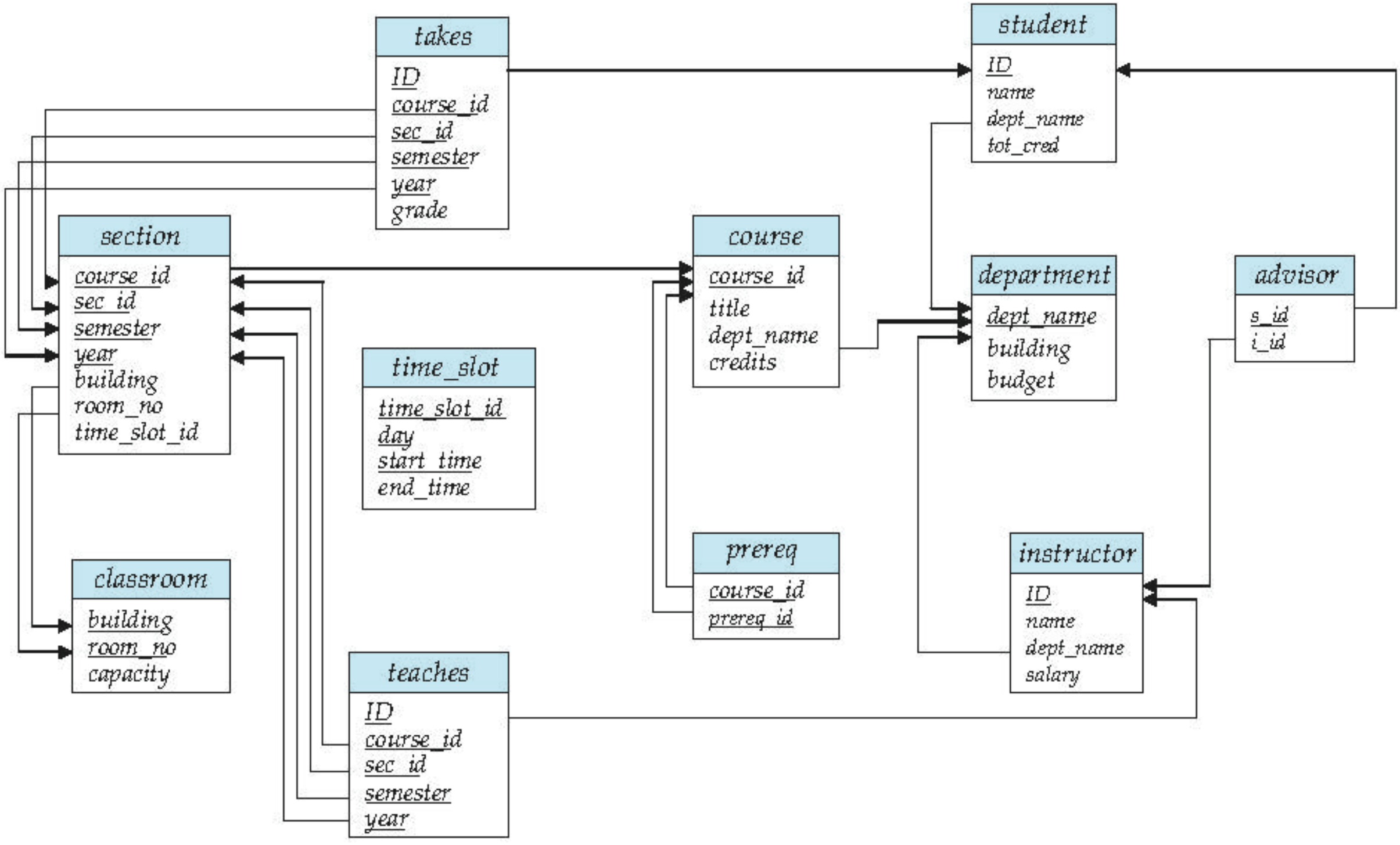
- 특정 스키마 구조물에 대한 실제 원소

데이터베이스 예제 [1/2]



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

데이터베이스 예제 [2/2]


takes
ID
course_id
sec_id
semester
year
grade
student
ID
name
dept_name
tot_cred
section
course_id
sec_id
semester
year
building
room_no
time_slot_id
course
course_id
title
dept_name
credits
department
dept_name
building
budget
advisor
s_id
i_id
time_slot
time_slot_id
day
start_time
end_time
prereq
course_id
prereq_id
classroom
building
room_no
capacity
instructor
ID
name
dept_name
salary
teaches
ID
course_id
sec_id
semester
year

데이터베이스 시스템 개념 [2/2]

❖ 스키마와 데이터베이스 상태

- 새로운 데이터베이스를 정의할 때 **새로운 스키마**만 DBMS에 명시하면 됨
- 초기 데이터가 처음으로 적재될 때 **데이터베이스 초기 상태**를 얻게 됨
- 데이터베이스에 갱신 연산이 수행될 때마다 다른 데이터베이스 상태를 얻음

❖ DBMS의 역할

- 데이터베이스의 모든 상태가 **유효한 상태**임을 보장하는 책임을 일부 지님
  - 유효한 상태: **스키마에 명시된 구조**와 **제약 조건**들을 만족하는 상태
  - **카탈로그**에 **메타데이터**라고 부르는 스키마와 제약 조건들에 관한 설명을 저장함으로써 필요할 때마다 DBMS는 스키마 참조 가능
    - 스키마: 내포 (intension)
    - 데이터베이스 상태: 스키마의 외연 (extension)

❖ 스키마 진화 (schema evolution)

- 응용의 요구사항에 따라 스키마는 가끔 **변경할 필요**가 생기기도 함
- 대부분의 DBMS는 스키마 진화에 필요한 연산을 데이터베이스 운영 중에도 실행할 수 있게 지원함

3단계-스키마 아키텍처 [1/3]

❖ 스키마 아키텍처

- 3단계-스키마 아키텍처의 목적은 **물리적 데이터 베이스**로부터 사용자 **응용들을 분리** 시키는 것
- 내부 단계 [내부 스키마]
  - 데이터베이스의 **물리적 저장 구조**를 기술
  - 물리적 데이터모델 사용
- 개념단계 [개념 스키마]
  - 모든 사용자들을 위한 전체 데이터베이스 구조를 기술
  - 물리적 저장 구조의 **세부 사항을 은폐**시키고 엔티티, 데이터 타입, 관계, 사용자 연산, 제약조건들을 나타내는데 중점을 둠
  - 표현 데이터 모델 사용
- 외부단계/ 뷰단계 [외부 스키마]
  - 특정 사용자 그룹이 관심을 갖는 데이터베이스 부분을 기술하고 그 사용자 그룹에게 데이터베이스의 나머지 부분을 **은폐시킴**
- 3단계 스키마는 데이터에 대한 기술이며 실제 데이터는 **물리적 단계**에만 위치함
  - 각 사용자 그룹은 외부 스키마를 참조하며 DBMS는 외부 스키마상에서 표현된 요구를 개념 스키마에 대한 요구로 변환시키고 다시 내부 스키마에 대한 요구로 변환시켜 저장된 데이터베이스에 접근함

데이터베이스 특징 [1/4]

❖ 파일 처리 시스템의 한계

- 소프트웨어 응용을 프로그래밍하는 작업의 일환으로 사용자가 특정한 소프트웨어 응용을 위하여 필요한 파일들을 **별도로 정의하고 구현**함
- 데이터의 중복과 비일관성과 같은 문제 발생 가능

❖ 데이터베이스 방식이 갖는 특징

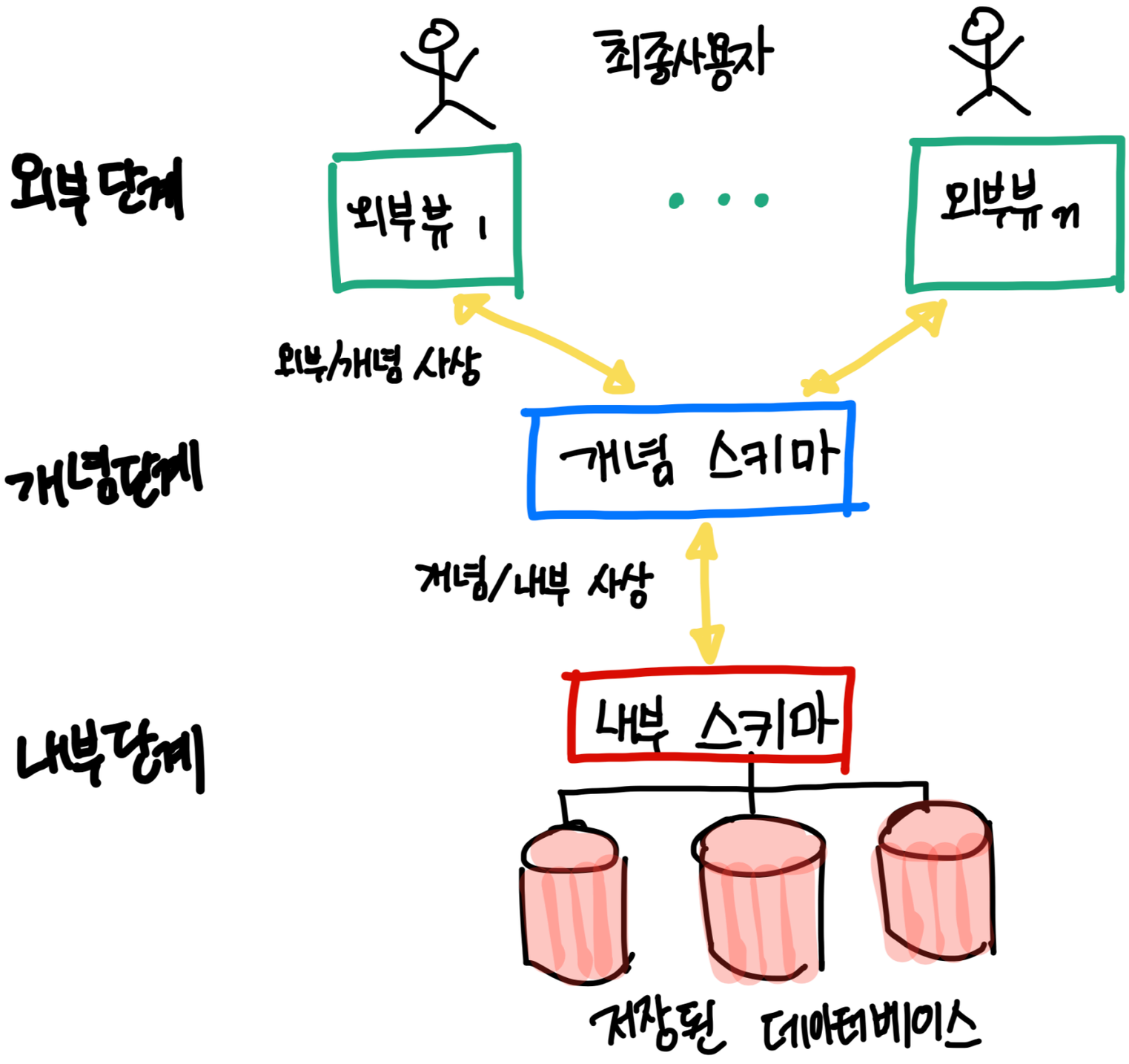
- 데이터베이스 방식에서는 **단일 저장소**에 데이터가 저장되고 여러 사용자가 **질의, 트랜잭션, 응용프로그램**을 통하여 데이터를 공유
  - 데이터베이스 시스템의 자기 기술성
  - 프로그램과 데이터의 격리 및 데이터 추상화
  - 데이터에 대한 다중 뷰(view)의 제공
  - 데이터의 공유와 다수 사용자 트랜잭션 처리

3단계-스키마 아키텍처 [2/3]

❖ 데이터 독립성

- 데이터 독립성은 **고수준의 스키마를 변경할 필요 없이** 데이터베이스 시스템의 어떤 단계에서 스키마를 변경할 수 있는 능력
- 논리적 데이터 독립성 (logical data independence)
  - 외부 스키마나 응용프로그램들을 변경하지 않으면서 **개념 스키마를 변경**하는 능력
    - 개념스키마 변경 예: 데이터베이스 확장, 제약조건 갱신, 데이터베이스 축소
  - 뷰정의나 사상만 변경함으로 외부 스키마 구조물들을 참조하는 응용프로그램들이 이전처럼 동작해야 함
- 물리적 데이터 독립성 (physical data independence)
  - 개념 스키마를 변경하지 않으면서 **내부 스키마를 변경**할 수 있음
    - 내부 스키마 변경 예: 특정 파일 검색이나 갱신 성능 향상을위해 접근 구조 추가
- 물리적 데이터 독립성은 대부분의 데이터베이스와 파일 환경에 존재하지만 논리적 데이터 독립성은 응용프로그램에 영향을 주지 않으며 구조적인 변경과 제약 조건을 허용해야 하기 때문에 **더 어려움**

3단계-스키마 아키텍처 [3/3]



데이터베이스 언어 [1/2]

❖ 데이터 베이스 언어 종류

- 데이터 조작 언어 (data manipulation language: DML)
  - 데이터베이스의 질의 및 갱신을 표현
- 데이터 정의 언어 (data definition language: DDL)
  - 데이터베이스 스키마를 기술
- 경계가 명확히 구분되어 있지는 않음. 예) SQL

❖ 데이터 조작 언어 (DML)

- 사용자가 적절한 데이터 모델로 구성된 **데이터를 접근하거나 조작**할 수 있도록 하는 언어
  - 저장된 정보 검색 | 새로운 정보 삽입 | 정보 삭제 | 저장된 데이터 수정
- 절차식 DML: 어떤 데이터가 필요하며 그 데이터를 어떻게 구할지 지정
- 선언적 DML: 필요한 데이터를 어떻게 구할지 명시 없이 어떠한 데이터가 필요한지만 지정
- 질의 (query): 정보의 **검색을 요청**하는 문장
- 질의어 (query language): 데이터 조작 언어에서 **정보의 검색을 담당**하는 부분

데이터베이스 언어 [2/2]

❖ 데이터 정의 언어 (DDL)

- 데이터 베이스의 **스키마의 정의를 표현**하는데에 사용 되는 언어이며, 데이터의 추가적 특성을 표현하는 데에도 사용.
  - 데이터 저장 및 정의 언어 (data storage and definition language): 저장 구조 storage structure)와 엑세스 방법을 지정함. 스키마 구현상의 세부 사항 정의
  - 명령문 (statement) → 메타데이터 (metadata: 데이터를 위한 데이터)
  - 데이터 사전 (data dictionary): 메타 데이터를 수록, 데이터베이스 시스템에 의해서만 접근되고 갱신될 수 있음.

데이터베이스 구성 모듈

❖ DBMS 의 구성 요소

- 카탈로그: DBMS 모듈들이 필요로 하는 다양한 유형의 정보를 포함
  - DBMS 모듈들이 필요로 하는 파일의 이름과 크기
  - 데이터 항목의 이름과 데이터 크기
  - 각 파일의 자세한 저장 형태
  - 스키마 사이의 사상 정보
  - 제약 조건들에 대한 정보
- 질의 컴파일러
  - 질의들을 파싱하고 질의 구문과 파일 및 데이터 원소들이 정확한지 입증하고 내부 형태로 컴파일 함
- 질의 최적화기
  - 연산들을 재배치, 연산들의 순서 바꿈, 중복 제거, 효율적인 검색 알고리즘 선택

DBMS 아키텍처 [1/2]

❖ 중앙집중식 DBMS 아키텍처

- DBMS의 모든 기능, 사용자의 응용 프로그램, 사용자 인터페이스 프로그램을 포함하는 **모든 시스템 기능**을 처리하기 위해 **메인프레임 컴퓨터**를 사용
- 사용자는 **자체 처리 능력이 없고** 화면의 결과만 보여주는 컴퓨터 터미널 사용
  - 모든 처리는 DBMS가 운영되는 컴퓨터 시스템 상에서 원격으로 수행되고 디스플레이 정보와 제어만 컴퓨터에서 터미널로 전송
- 현재는 하드웨어 가격 하락으로 대부분의 사용자는 터미널을 개인용 컴퓨터, 워크스테이션, 모바일 기기로 교체
  - 충분한 자체 처리 능력을 지님

➡ **클라이언트- 서버 DBMS 아키텍처 등장**

DBMS 아키텍처 [2/2]

❖ 일반적인 클라이언트-서버 아키텍처

- 많은 수의 PC, 워크스테이션 파일서버, 프린터, 데이터베이스 서버, 웹서버, 이메일서버, 기타소프트웨어 장치들이 네트워크를 통해서 서로 연결되어 있는 컴퓨터 환경을 다루기 위해 개발
  - 클라이언트: 사용자 인터페이스와 자체처리 능력을 제공하는 사용자 컴퓨터
  - 서버: 특정 기능을 갖는 특별한 서버들을 정의, 클라이언트 컴퓨터에게 서비스들을 제공할 수 있는 하드웨어와 소프트웨어를 포함하는 시스템

❖ DBMS를 위한 2층 클라이언트-서버 아키텍처

- 서버: 질의 서버, 트랜잭션 서버
- 클라이언트: 사용자 인터페이스와 응용프로그램
  - 질의의 결과는 필요에 따라 좀 더 처리하고 디스플레이 할 수 있음

❖ 웹 응용을 위한 3층 클라이언트-서버 아키텍처

- 클라이언트와 데이터베이스 서버 사이에 중간 단계 하나를 더 추가
- 중간단계 (중간 층) → 응용서버/ 웹 서버
  - 응용 프로그램들을 수행하고 데이터베이스 서버에 저장된 데이터에 접근하는데 사용되는 비즈니스 규칙 (프로시저 또는 제약 조건)을 저장함으로써 중간 역할을 수행
  - 클라이언트의 신임 여부를 검사 → 데이터 베이스 보안 향상
  - 데이터베이스 서버에서 처리된 데이터를 클라이언트로 보내는 통로 역할

웹 응용을 위한
3단계 클라이언트-서버 아키텍처