DB01

`Data` : **내제된 의미**가 있고, **저장이 가능**한 **알려진 사실**

`Database` : **논리적으로 관련**된 데이터의 집합

`Mini-world` : **실세계의 일부**를 표현한 것

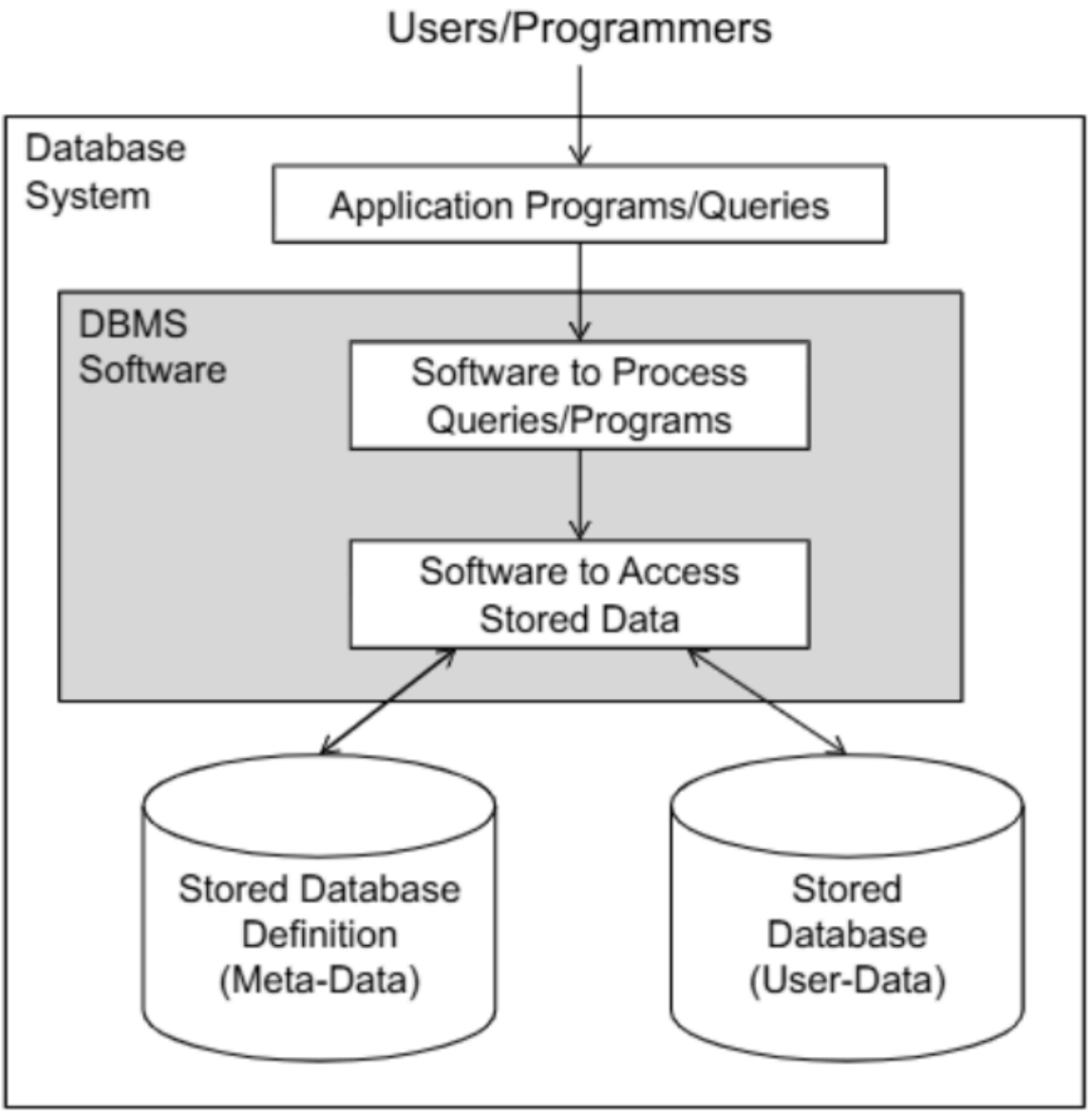
- Database를 설계하는 목적

`Database Management System` (DBMS)

- Database를 **쉽게 생성, 관리할 수 있도록 해주는 Softwares**
- User가 쉽게 Database를 생성 및 관리할 수 있도록 해준다.

`Database System` : DBMS + Database

- Appilication까지 포함하기도 하는데 이는 견해마다 차이가 있다.



- Application은 **자신의 Code**와 **DBMS를 호출하는 코드로 구성된다.**
- **Query Engines:** User에게 간단한 명령어로 DBMS를 다룰 수 있도록 해주는 부분 / SQL
- **Storage Engine**: 실제로 디스크에 있는 데이터 파일을 읽거나 쓰는 부분
- **Secondary storage: Meta-data와 user-data는 분리되어 저장된다.**
    - Meta-data란 사용자 데이터의 개수, 길이 등 사용자 데이터를 관리하기 위해 DBMS가 필요로 하는 데이터이다.

Application: 데이터가 필요할 때 DBMS를 호출하는 코드를 가진다. → **Query Engine**: 사용자가 입력한 SQL을 인식한다. → **Storage Engine**: Query engine에서 전달한 실행

계획에 맞게 Second storage에 접근한다.

Data Records

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

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

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

- Student ~ Grade_report 각각이 Data record이다.
- Prerequisite의 경우 Entity가 아니라 Relation이다.
  - **Data records는 Entity와 Relation을 모두 저장**한다.
- Section의 경우 Entity면서 Course와의 Relation도 포함한다.
  - **Data records는 Entity이면서 동시에 다른 Entity와의 Relation까지 저장할 수 있다.**

Database의 성질

1. **Self-describing nature**

   Meta data: Database의 구조를 설명한다.

   - User data를 유지하기 위해서 필수적이다.
   - **DBMS의 System Catalog**에 저장된다.

Database system은 User data뿐만이 아니라 Meta-data까지 저장해야 한다.

2. **Insulation betweem programs and data**

   Program은 Data의 변경의 큰 영향을 받지 않는다.

   - Database의 **Self-describing 성질** 때문이다.
   - Data의 구조가 바뀌어도 Program이 바뀔 필요 없이 Meta-data만 수정되면 된다.
   - **Higher program-data indepence**를 제공해 Program 관리가 쉬워진다.

3. **Data abstruction**

   Data의 구체적인 구조는 몰라도 된다.

   - DBMS를 통해 Database를 관리하면 실제로 저장되어져 있는 구조는 몰라도 된다.

   Data model을 통해 **Data가 실제로 어떻게 저장이 되어져 있는지는(Physical view) 숨기고 사용자가 이해하기 쉬운 구조로(Conceptual view) 확인**할 수 있다.

4. **Multiple views of data**

   View: Database에 data가 저장되어 있는 형식 그대로 보여주는게 아니라, **그것을 조합하여 사용자에게 더 나은 형태**로 보여주는 것

   각기 **다른 View를 이용하여 같은 Database라 하더라도 다른 관점**을 얻을 수 있다.

5. **Sharing of data**

   하나의 Database에 여러 사람이 접근하고자 할 때, **동시에 접근 및 사용을 가능**하도록 한다.

   Concurrency control을 통해 여러 사람이 같은 Data에 동시 접근해 문제가 발생하는 상황을 Control한다.

Users of Database systems

1. **System analyst**: End user(System을 사용할 사람들)의 **요구 사항을 분석하고 어떤 기능이 수행**되어야 하는지 분석한다.

2. **Database designer**: 저장될 data를 확인하고 **Database의 구조를 디자인**

3. **Application programmer**: **기능을 구현**하는 사람

4. **Database administrator (DBA):** Application을 포함한 **Database system을 monitoring**하는 사람

5. **End users: Database application program을 통해 database에 access**하고자 하는 사람들

Workers behind the scene

1. **DBMS system designer and implementer:** DBMS의 module과 interface를 구현하는 사람들

2. **Tool developer**: Database에 쉽게 접근할 수 있는 Software tool을 만드는 사람들

3. **Operator and maintenance personnel:** 데이터베이스 시스템을 위한 하드웨어 및 소프트웨어 환경의 실행(running)과 유지보수(maintenance)를 책임지는 사람들

DBMS Features

1. **Redundancy를 Control**
    - **Redundancy**: 여러 저장 공간에 같은 데이터가 저장되는 것
    - Disk 공간 낭비 + Data consistency 문제

2. **Unauthorized access**를 방지
    - 보안 관점
    - Privileged software: 다른 software에 비해 권한이 높음

3. **Persistent storage for program objects**
    - Program의 **종료와 관계없이 데이터는 유지**되어야 한다.
    - Data는 main memory에 저장되는 것이 아니라 Disk에 저장되어야 한다.

4. **Multiple user interface**
    - User가 사용하는 Interface, software language, API 등에 따른 여러 가지의 DBMS 접근 방법을 모두 제공해야 한다.

5. **Data 간의 복잡한 Relation을 표현**

6. **Integrity constraints**
    - **Constraint가 제약하는 성질을 반드시 만족**해야 한다.
    - DBMS는 이를 의해 **Integrity constraint를 정의하는 함수**를 제공해야 한다.
    - Constraint를 만족해야 Database에 결함이 없음이 보장이 된다.
    - DBMS가 **Integrity constraints를 만족하는지 자동으로 체크하는 기능**을 제공해야 한다.

- DBMS는 (1) 규칙을 **만들게** 해주고, (2) 그 규칙을 **어기지 못하도록** 스스로 감시하고 막아준다.

7. **Backup and recovery**
   - **Backup**: 나중에 문제 발생 시, 복구를 위해 Normal processing 중에 데이터의 Copy본을 남겨두는 것이다.
   - **Recovery**: 실제로 Hardware/software 문제 발생 시, Backup에 맞게 복구하는 것.
     - 단순히 Backup한 것을 가져오는 것이 아니라, **마지막 Backup 이후의 Update도 가져올 수 있어야 한다.**

DBMS의 장점

1. Data 관리를 위해 **직접 Programming할 필요가 없어 효율적**이다.
   - Database에 직접 접근해서 다루는 것은 Data가 많아질 경우 비효율적이다.

2. **up-to-date information**: 한 사용자의 Update가 다른 사용자에게 즉시 반영된다.

3. **Economy of scale**: DBMS를 사용하는 사용자가 많아질수록 DBMS의 비용은 낮아진다.

DBMS의 단점

1. Initial cost가 크다.
   - Hardware, software, education ...

2. Secure, Integriry constraint, Concurrency control 등으로 인한 **많은 연산량**

DBMS를 사용하지 않는 것이 나은 경우

1. **Application이 쉽고 앞으로 바뀔 가능성이 적은 경우**
    - DBMS의 Self-describing으로 인한 program-data indepedence를 보장하는 것이 Application이 간단하면 필요가 없다.

2. **마감시간이 정해진 경우**
    - 초기 비용이 비싸기 때문이다.

3. **Multiple user가 동시에 Database에 접근하지 않는 경우**
    - 사용자가 Data를 update하지 않는 경우도 포함한다.
    - 검색 엔진이 예시이다.