2023 봄학기
숙명여대 DBMS

- **Lecture Name:** **Database Systems**
- **Semester:** **Spring, 2023**
- **Professor:** **Park Young-Ho**

Lecture 2장-2nd
DBMS 개념과 아키텍쳐

- **Lecture Name:** **Database Systems**
- **Semester:** **Spring, 2023**
- **Professor:** **Park Young-Ho**

Index

지난 시간 강의 리뷰 요약

- 모델
  - 대상을 잘 설명하기 위한 방법론
  - 구조, 연산, 제약조건으로 설명함
- 데이터모델
  - 데이터를 잘 설명하기 위한 방법론
  - 데이터 구조 및 타입, 그에 적용되는 연산, 제약조건으로 설명함
- 데이터 모델의 구분
  - 개념적 데이터 모델 (Conceptual Data Model)
  - 물리적 데이터 모델 (Physical Data Model)
  - 구현 데이터 모델 (Implementation Data Model)
  - 자가설명형 데이터 모델 (Self-describing Data Model)
- 스키마와 인스턴스
  - Schema: Catalog, Structure, Type, Rule, Intention, a seq. of Attr.
  - Instance: Record, Tuple, State, Snapshot, Extension, a seq. of Attr. Val.

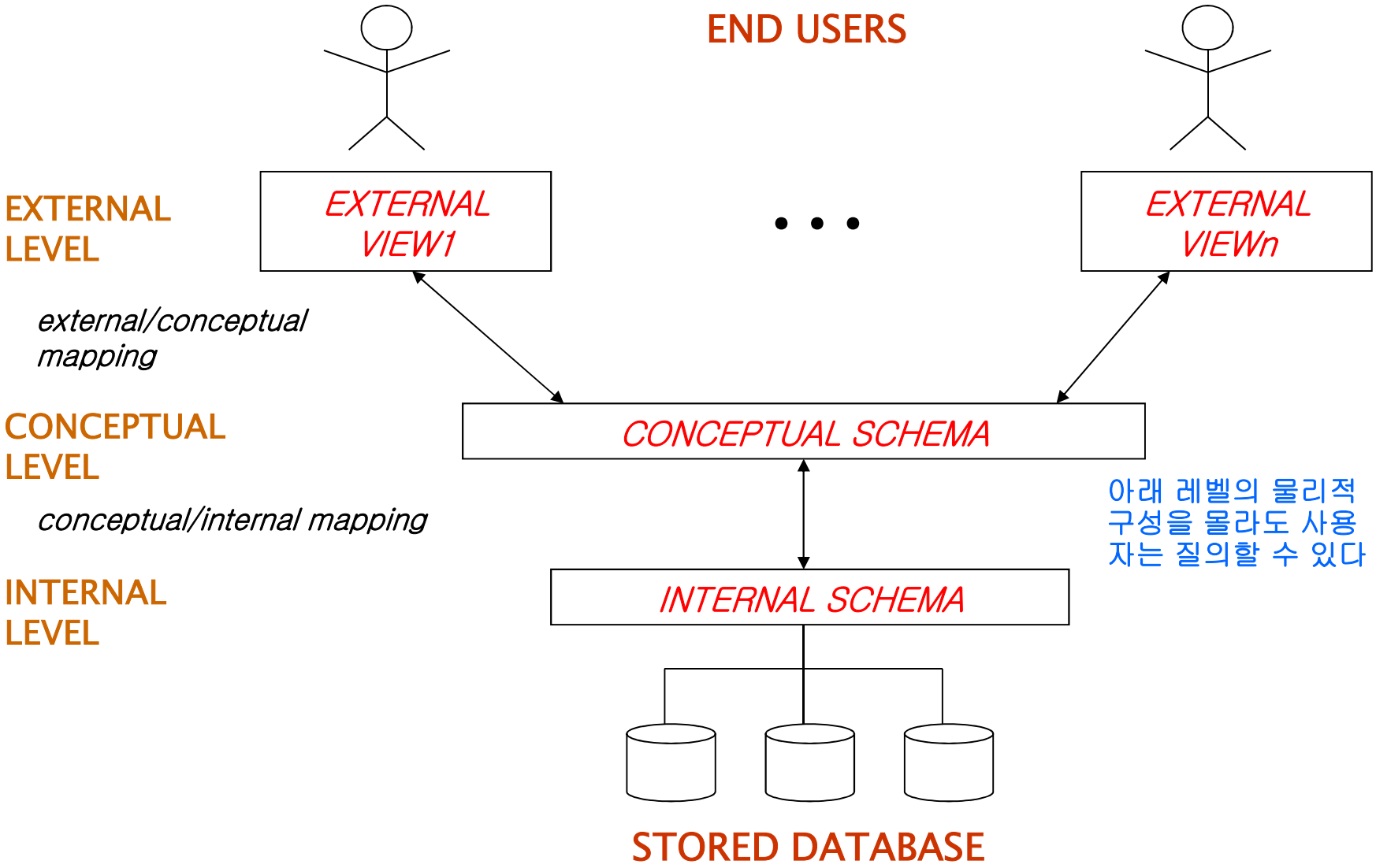
Index

Three-Schema Architecture (1/3)

- **DBMS**가 다음의 특성들을 지원하기 위한 구조임.
  - **Program-data independence** (상호간의 독립성)를 제공하기 위함
  - 데이터의 복수개의 뷰**(multiple view)**를 제공하기 위하여 만들어짐

- **DBMS schema**를 **3 level**로 구분하여 정의함.
  - ***External schemas:*** 다양한 사용자 뷰**(view)**들을 기술하는 레벨 **(external level)** 이다.
    - 예: **SQL** 언어로 각 사용자가 필요로 하는 정보인, **view**를 기술함.
  - ***Conceptual schema:*** 전체 데이타베이스에 대한 데이터의 구조와 그 데이터에 부여되는 제약 조건들을 기술하는 레벨이다.
    - **Conceptual Schema**가 우리가 지금까지 배운 **Schema**를 말함.
  - ***Internal schema:*** **Physical data model**을 표현해 주는 것임.
    - **Data storage** 구조와 **Access path**들과 같은 디스크에 어떻게 저장할 것인가 등 구체적 방법을 기술하는 **low-level schema** 임.

Three-Schema Architecture (2/3)



Three-Schema Architecture (3/3)

- **schema level**들간의 "**mapping**"의 용도
  - **user**가 보는 **view**를 기반으로 질의가 주어졌을 때,
    computer가 보는 **view**의 관점으로 변환시켜 주기 위해서,
    각 **level**간에 "**mapping = query** 변환"이 필요함.
    - **(1)** 프로그램들이 **external schema**를 참조할 때 필요하게 된다.
    - **(2) DBMS**가 그 프로그램의 수행을 위해, **internal schema**에 **map**하기 위해 필요하다.

- 레벨간 **mapping**을 설명하기 위한 질의 처리**(query processing)** 수행 예제
  - "**DB**" 코스를 수강하는 "학생" 중 "나이"가 **30**세 이상인 학생**? (**고수준질문**)**
    - "**DB**" 릴레이션과, "학생" 릴레이션이 있는 **disk page**를 찾는다. (저수준 처리)
    - "나이"는 **tuple**의 시작으로 부터, **offset 39 bytes** 부터 시작한다.
    - (나이 순으로 **tuple** 이 정렬되어 있다면,) "나이"의 "**value**"가 **30**보다 작으면 왼쪽으로, 크면 오른쪽으로 간다.
    - 그 결과를 **projection**한다. 그리고, **Next record**는 **byte offset**이 **400 bytes**이므로 이를 더해가며, 결과를 릴레이션의 끝까지 찾는다.

→ 즉, **External Schema**(고수준)를 참조하는 프로그램들은 **Internal schema**(저수준)를 참조하는 프로그램으로 변경되어야 한다.

→ 이것이 "**mapping**의 개념"이다.

Index

데이터 독립성(Data Independence) (1/2)

❖ **(중요) 왜 mapping을 하는가?**
(하위 레벨로부터 독립하자!
즉, 상위 레벨은 안 바꾸고, mapping만 바꾸자!)
다음의 중요한 **2가지 특성**을 부여하기 위함이다.

- 논리적 데이터(로 부터) 독립성**(Logical Data Independence)** (독립하자)
  - **external schema**나 그들을 이용하는 응용 프로그램들을 전혀 바꾸지 않고도 **conceptual schema**가 변경될 수 있도록 **DBMS**가 지원하는 특성
  - (예, 스키마에서 "이름"과 "주소"의 위치를 변경해도, 상위 프로그램은 불변)
  - 예) 즉, **GRADE_REPORT**를 바꿔도, 프로그램의 변경이 없어, **TRANSCRIPT**(학생 성적표)결과에 영향이 없어야 함

- 물리적 데이터 (로 부터) 독립성**(Physical Data Independence)** (독립하자)
  - **conceptual schema** (우리가 **DDL**로 작성한 **schema**)를 전혀 바꾸지 않고도 **internal schema** (**disk**내 **record** 위치 정보 등) 만 변경해도 되도록 지원하는 특성
  - 자료에 대한 접근 기능을 강화하여, 검색이나, 갱신 성능을 높이고자 할때, 물리적 부분(디스크) 접근 구조를 변경함 → 상위에 영향 안 주어야 함
  - 예) **disk** 내의 레코드 위치를 **track 15**에서 **track 24**로 이동해도 **schema**는 그대로 놔두면 됨

데이터 독립성(Data Independence) (2/2)

- 하위 레벨에 있는 어떤 **schema**에 변경이 생겼을 때,
  변경된 **schema**와 그것의 상위 레벨 **schema** 사이에 차이가 생긴다.
  이때, 단순히 "**mapping**"만을 변경해서,
  "데이터의 독립성"을 완전히 지원할 수 있도록 할 필요성이 있다.

- 그 상위 레벨 **schema**들은 안 바뀐다.
  그러므로, 응용 프로그램들은 **external schema**들을 참조**(refer)**하기 때문에 변경하지 않아도 되는 것이다.

Index

DBMS Languages (1/2)

- **Data Definition Language (DDL)**

  → **schema** 정의 언어 **= DDL**

  (DBMS와 대화하기 위한 준비 언어**)**

  - **DBA**와 **database designer**들이 데이타베이스의 **conceptual schema**를 표현하기 위해서 이 언어 **(DDL)**를 사용한다.

  - 대부분의 **DBMS**들에서, **DDL**은
    - **internal schemas (tables, index** 등 **conceptual schema),**
    - **external schemas (**사용자 **views)**

    를 모두 정의하기 위해 사용된다.

  - **(**중요하지 않음**)** 일부 어떤 **DBMS**들에서는,
    - **internal schemas** 표현을 위해서 **storage definition language (SDL)** 언어를 사용하고,
    - **external schemas** 표현을 위해서 **view definition language (VDL)** 언어를 분리하여 사용한다.

DBMS Languages (2/2)

- **Data Manipulation Language (DML)**

  → **queries = SQL**

  - 데이타베이스의 내용을 추출하고 갱신하기 위한 언어로, 다음과 같이, **(1)** 고급 언어 내재 방식과 **(2)** 문답식 방식이 있다.

1. 고급 언어 내재 방식**(embedded language)**

   DML commands (data sub language)는
   C, C++, C#, Java, Python or PASCAL와 같은
   General-purpose Programming Language (범용 프로그래밍 언어)에
   내재되는 언어**(embedded language)**를 말한다. **(ODBC**나 **JDBC** 드라이버 필요)

   또한, **DML**은
   JSP, ASP, PHP, HTML5과 같은
   Web Script Language (웹 언어)에
   내재되는 언어**(embedded language)**를 말한다. **(**웹 서버 설치 필요**)**

2. 문답식 방식 **(Interactive Queries,** 또는 **Stand-alone DML)**

   선택적으로, **stand-alone DML** 명령어들은
   (예를 들어, **SELECT, INSERT, UPDATE, DELETE** 와 같은 **DML**은**)**
   DBMS가 제공하는 **질의 창 (Query Interface)**를 통해 **query**를 **직접 작성**하여 질의 할 수 있다.

DML의 타입

- **High Level or Non-procedural Language (= 목적만 명시함)**
  - **For example, the SQL relational language**
  - ***"SET-at-a-time"*** **형태로 데이터를 검색한다.**
  - ***"Declarative Language"*** **라고도 부른다.**
  - **Are "set"-oriented and specify what data to retrieve rather than how to retrieve it.**

- **Low Level or Procedural Language (= job의 모든 과정을 구체적 명시함)**
  - ***"RECORD-at-a-time"*** **형태로 데이터를 검색한다.**
  - 호스트 프로그램 언어**(C, C++, Java, python** 등**)** 안에 **DML** 문장들**(SQL)**이 **embedded**된 후, 컴파일 되어, 하나의 실행파일이 된다.
  - 개별 **database RECORD**들에 대해서 찾고, 추출한다. 그리고, **Multi-Record**들을 각기 찾고, 추출하기 위해서는 **Host Programming Language**의 **looping**과 **construct**들을 사용한다.
    (한번에 레코드 하나, 그리고 **next, next, …** 해서 찾음, 수동화**)**

Index

DBMS Interfaces

- [0] Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- [1] Programmer interfaces for embedding DML in programming languages
- [2] User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.
- [3] Other DBMS Interfaces
  - E.g. Mobile interfaces allowing users to perform transactions using mobile apps

[1] DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - Embedded Approach
    - e.g. embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - Procedure Call Approach
    - e.g. JDBC for Java, ODBC (Open Database Connectivity)
    - For other programming languages as API's (application programming interfaces)
  - 전용 Database Programming Language Approach
    - e.g. ORACLE has PL/SQL, a programming language based on SQL
    - Language incorporates SQL and its data types *as integral components*
  - Scripting Languages: PHP (client-side scripting) and Python (server-side scripting) are used to write database programs.

[2] User-Friendly DBMS Interfaces

- Menu-based (Web-based), popular for browsing on the web
- Forms-based, designed for naïve users used to filling in entries on a form
- Graphics-based
  - Point and Click, Drag and Drop, etc.
  - Specifying a query on a schema diagram
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces

[3] Other DBMS Interfaces

- Natural language: free text as a query
- Speech : Input query and Output response
- Web Browser with keyword search
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

Database System Utilities

- To perform *certain functions* such as:
  - Loading data stored in files into a database. Includes data conversion tools.
    - 예: Google 10억건 로딩 (1000만건에 100기가) → 10 tera bytes 필요
  - Backing up the database periodically on tape.
  - Reorganizing database file structures.
  - Performance monitoring utilities.
  - Report generation utilities.
  - Other functions, such as sorting, user monitoring, data compression, etc.

Other Tools

- Data dictionary / Repository:
  - Used to store (1) schema descriptions and other information such as (2) design decisions, (3) application program descriptions, (4) user information, (5) usage standards, etc.
  - *Active data dictionary* is accessed by DBMS software and users / DBA all together.
    - Catalog의 Schema 등
  - *Passive data dictionary* is accessed by users / DBA only.

Other Tools

- Application Development Environments
  - CASE (computer-aided software engineering) tools.
- Examples:
  - PowerBuilder (Sybase)
  - JBuilder (Borland)
  - JDeveloper 10G (Oracle)

Index

수고하셨습니다.