

DEVKOR

Backend Study

컴퓨터의 데이터 저장 방식

- 주 기억 장치
 - RAM 등 메모리
 - 빠르지만 ,,
 - 휘발성
 - 용량도 많지 않음
- 보조 기억 장치
 - HDD, SSD 등
 - > 파일 시스템 사용?
 - 대용량에 부적합..
 - 파일로 데이터를 저장하고, 그 데이터를 사용하는 응용프로그램으로 파일을 읽음
 - 데이터 구조, 접근 방식 등을 변경하기가 어렵다
 - 데이터 관리가 쉽지 않다
 - 종속성,, 무결성

데이터베이스

- 그런 단점을 해결하기 위해 데이터베이스를 쓴다 !
- Data Base
 - 여러 사람이 접근하고, 공유하며 사용할 목적으로 데이터를 통합하여 관리하는 데이터의 집합
 - 탐색 기능은 물론이고 정렬 기능 등 데이터 이용에 좋은 여러 기능 제공
 - 여러 방식으로 데이터를 관리, 우리가 이번에 공부할 것은 관계형 데이터베이스!
 - 데이터를 여러 사람이 접근하여도 일관성-무결성 integrity에 문제가 생기지 않도록 하는 여러 제약 사항-특징 들이 존재
 - 데이터의 집합인 데이터베이스에 더해 그것을 관리하는 시스템과 통합되어 제공
 - > DataBase Management System, DBMS

DBMS 의 목적


- data isolation
 - 고립, 격리..
 - 파일 시스템의 단점 해결
 - 데이터의 삽입, 삭제를 구조적인 어떤 종속 없이 해결
- data integrity
 - 일관성, 무결성
 - 데이터가 필요한 제약 사항을 지키도록 함
 - ex.. 계좌잔고 > 0, password 길이.. 학년, 학번 ...

DBMS 의 목적

- Atomicity
 - 원자성.
 - git 처럼, DB 삽입 등의 연산이 원자성을 가짐
- Concurrent Access by multiple users
 - 여러 유저가 동일한 데이터에, 데이터의 일관성을 해치지 않으면서 동시에 접근할 수 있도록 함
- Security
 - 3요소
 - Integrity : 자료 오류 X
 - availability: 권한이 있으면, 쉽게 접근 가능
 - Confidentiality: 외부로부터 자료를 보호

Relational Data Base

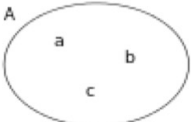
- Relational model을 기반으로 한 데이터베이스.
- Relational model
 - relation 형식으로 데이터를 저장




What is Relation ?

- In **mathematics**, an **n-ary relation** on **n sets**, is any subset of *Cartesian product* of the **n sets**

A

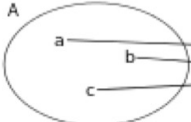


B




$A \times B = \{ \{a,x\}, \{b,x\}, \{c,x\}, \{a,y\}, \{b,y\}, \{c,y\}, \{a,z\}, \{b,z\}, \{c,z\} \}$

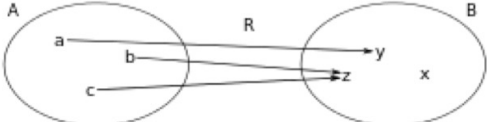
A



B



R



$R = \{ \{a,y\}, \{b,z\}, \{c,z\} \}$

A (binary) relation R among sets A and B is a subset of the Cartesian product of the sets A and B , $R \subseteq A \times B$.

Database System Concepts - 7th Edition

2.2

©Silberschatz, Korth and Sudarshan

Relational Data Base

- relation
- column & row

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

attributes
(or columns)

tuples
(or rows)

lec ord

- A_1, A_2, \dots, A_n are *attributes*
- $R = (A_1, A_2, \dots, A_n)$ is a *relation schema*

Example:

instructor = (*ID*, *name*, *dept_name*, *salary*)

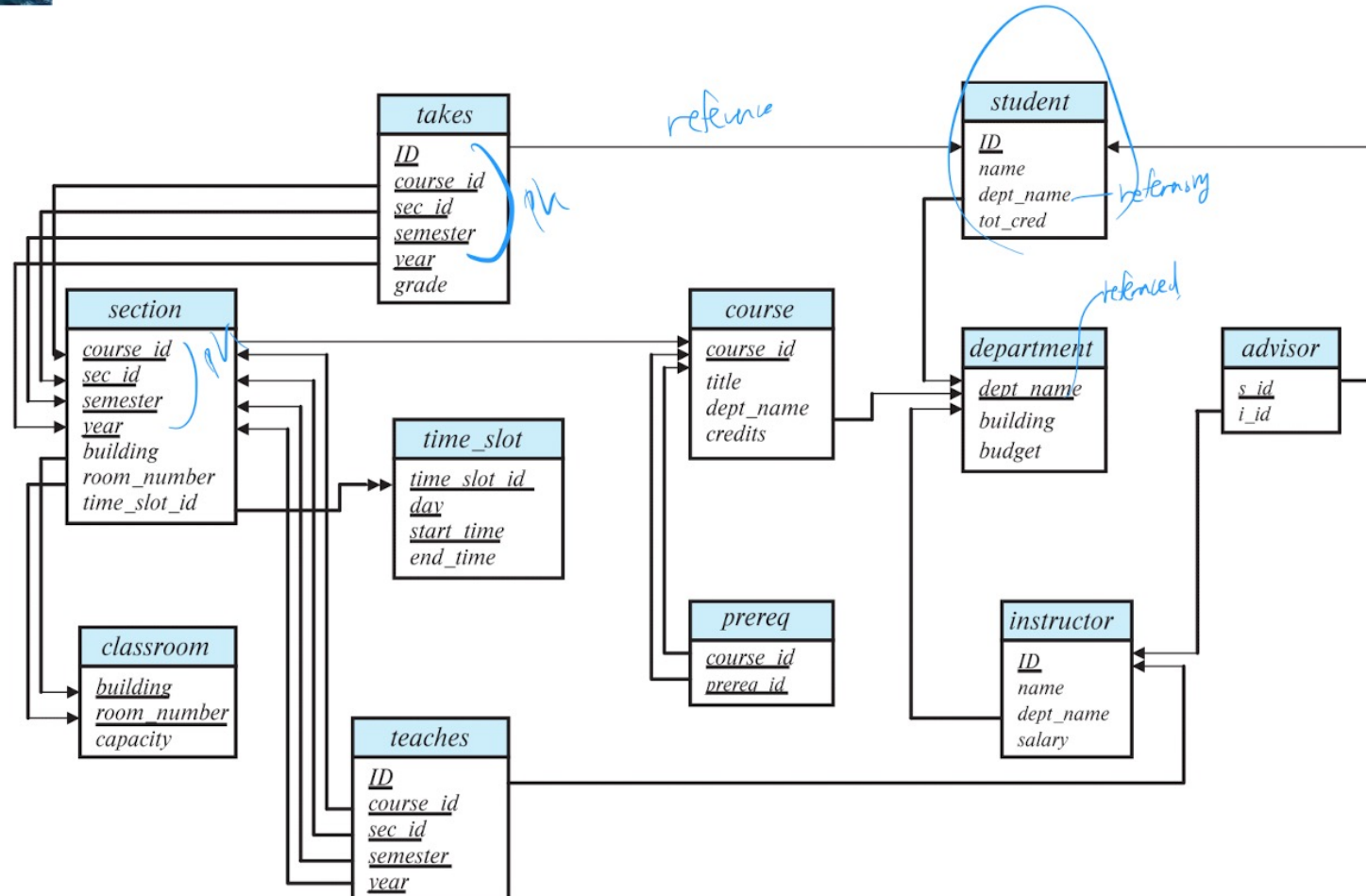
Relational Data Base

- Key ← 고유하게 하는 것
- super key
 - 값이 고유해질 수 있는 column의 집합
- candidate key
 - superkey 집합 중, column 수가 가장 적은 것
- primary key
 - candidate 중, 실제로 unique 식별자로 쓰는 key
- foreign key
 - 다른 테이블에서 가져온 값 .
 - PK만 참조 가능 !
 - data 무결성을 위함

DB schema



Schema Diagram for University Database



maria DB

<https://downloads.mariadb.org>

<https://kitty-geno.tistory.com/55>

This is a preview release with all potential upcoming features bundled into one package. Not all these features may make it into the final stable release of 10.11.

MariaDB Server Version

MariaDB Server 10.11.0 Alpha

Display older releases: ☐

Operating System

Windows

Architecture

x86_64

Package Type

MSI Package

Download

Mirror

Blendbyte - Taipei

Release date: 2022-09-26

File name: mariadb-10.11.0-winx64.msi

File size: 64.4 MB

•

Display signature and checksums

maria DB

<https://downloads.mariadb.org>

<https://kitty-geno.tistory.com/55>

This is a preview release with all potential upcoming features bundled into one package. Not all these features may make it into the final stable release of 10.11.

MariaDB Server Version

MariaDB Server 10.11.0 Alpha

Display older releases: ☐

Operating System

Windows

Architecture

x86_64

Package Type

MSI Package

Download

Mirror

Blendbyte - Taipei

Release date: 2022-09-26

File name: mariadb-10.11.0-winx64.msi

File size: 64.4 MB

•

Display signature and checksums

maria DB

```
$ /bin/bash c \  
    “$(curl -fsSl https://raw.githubusercontent.com/install/HEAD/install.sh)”  
$ brew update  
  
brew install mariadb  
brew services start mariadb
```

maria DB

```
$ mysql -u<username> -p<password>
```

```
$ CREATE DATABASE devkor;
```

```
$ USE devkor;
```

maria DB

Category	Data Types
문자형	CHAR ($n < 2^8$) / VARCHAR ($n < 2^{16}$) / TINYTEXT ($n < 2^8$) TEXT ($n < 2^{16}$) / MEDIUMTEXT ($n < 2^{24}$) / LONGTEXT ($n < 2^{32}$)
숫자형	TINYINT ($n < 2^8$) / SMALLINT ($n < 2^{16}$) / MEDIUMINT ($n < 2^{24}$) INT ($n < 2^{32}$) / BIGINT (∞) / FLOAT / DECIMAL / DOUBLE
날짜형	DATE / TIME / DATETIME / TIMESTAMP / YEAR
이진 데이터	BINARY / BYTE / TINYBLOB / BLOB / MEDIUMBLOB / LONGBLOB

maria DB

```
CREATE TABLE `departments` (  
    `id` INT NOT NULL AUTO_INCREMENT,  
    `kor_name` VARCHAR(8) NOT NULL,  
    `eng_name` VARCHAR(32) NOT NULL,  
    `college` VARCHAR(16) NOT NULL,  
    PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

maria DB

```
CREATE TABLE `courses` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `name` VARCHAR(20) NOT NULL,  
  `department` INT NOT NULL,  
  `code` VARCHAR(8) NOT NULL,  
  `is_major` TINYINT(1) NOT NULL,  
  `is_required` TINYINT(1) NOT NULL,  
  `credit` INT NOT NULL,  
  `period` INT NOT NULL,  
  PRIMARY KEY (`id`),  
  FOREIGN KEY (`department`)  
    REFERENCES `departments`(`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```


SQL

DB에서 데이터 관리를 위해 사용되는 표준 언어
아까 봤던 create table, database부터..
데이터를 삽입, 삭제, 조회 하는 언어

Structured Query Language

SQL - insert

```
INSERT INTO <TABLE> VALUES (a1, a2, a3, a4...);
```

SQL - 기본 구조

삭제 - DELETE

조회 - SELECT 명령어

+ FROM, WHERE

FROM - 어떤 테이블에서 삭제, 조회 할 것인지

WHERE - 해당 데이터의 제약 조건.

-> SELECT <columns> FROM <table> WHERE <const>;

-> DELETE FROM <table> WHERE <constaraint>;

SQL - WHERE

column의 attribute 값 제약

ex)

```
WHERE dept_name = 'Comp. Sci.';
```

```
WHERE salary > 500;
```

```
WHERE salary > 500 & dept_name = 'Comp. Sci.';
```

SQL - , -> cartesian product

```
SELECT column1, column2 FROM table;
```

```
SELECT column1, column2 FROM table1, table2;
```

SQL - join

Foreign key로 참조되는
column 값들을 넣어준다 !

이외에도 subquery, 등등등 여러 제약 조건이 존재

자세한 건

<https://mariadb.com/kb/en/sql-statements/>

<https://www.postgresqltutorial.com>

SQL - join

```
select name, title
from student natural join takes, course
where takes.course_id = course.course_id;
```

student natural join takes

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>tot_cred</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>	<i>grade</i>
00128	Zhang	Comp. Sci.	102	CS-101	1	Fall	2017	A
00128	Zhang	Comp. Sci.	102	CS-347	1	Fall	2017	A-
12345	Shankar	Comp. Sci.	32	CS-101	1	Fall	2017	C
12345	Shankar	Comp. Sci.	32	CS-190	2	Spring	2017	A
12345	Shankar	Comp. Sci.	32	CS-315	1	Spring	2018	A
12345	Shankar	Comp. Sci.	32	CS-347	1	Fall	2017	A
19991	Brandt	History	80	HIS-351	1	Spring	2018	B
23121	Chavez	Finance	110	FIN-201	1	Spring	2018	C+
44553	Peltier	Physics	56	PHY-101	1	Fall	2017	B-
45678	Levy	Physics	46	CS-101	1	Fall	2017	F
45678	Levy	Physics	46	CS-101	1	Spring	2018	B+
45678	Levy	Physics	46	CS-319	1	Spring	2018	B
54321	Williams	Comp. Sci.	54	CS-101	1	Fall	2017	A-
54321	Williams	Comp. Sci.	54	CS-190	2	Spring	2017	B+
55739	Sanchez	Music	38	MU-199	1	Spring	2018	A-
76543	Brown	Comp. Sci.	58	CS-101	1	Fall	2017	A
76543	Brown	Comp. Sci.	58	CS-319	2	Spring	2018	A
76653	Aoi	Elec. Eng.	60	EE-181	1	Spring	2017	C
98765	Bourikas	Elec. Eng.	98	CS-101	1	Fall	2017	C-
98765	Bourikas	Elec. Eng.	98	CS-315	1	Spring	2018	B
98988	Tanaka	Biology	120	BIO-101	1	Summer	2017	A
98988	Tanaka	Biology	120	BIO-301	1	Summer	2018	<i>null</i>

node + maria db

```
$ npm install mysql
```

```
import mysql from 'mysql';  
const connection = mysql.createConnection({  
  host: 'localhost',  
  port: 3306,  
  user: 'root',  
  password: 'passwd',  
  database: 'mydb'  
});  
connection.connect();
```

```
connection.query('SELECT * FROM students', (err, rows, fields) => {  
  if(err) {  
    next(err);  
  }  
  else {  
    const result = JSON.stringify(rows);  
    res.send(result);  
  }  
})
```


감사합니다!