2021 Winter wheel seminar #6: Database

2021.1.11. ivy

Contents

Introduction

- DB?
- DB schema
- DB types
- Query
- Transaction

Relational DB

- RDB structure
- Consistency
- Key and Normalization
- RDB schema representation
- SQL
- MySQL DB

NoSQL DB

- Limits of SQL, RDB
- NoSQL DB model
- Scaling
- mongoDB
- redis
- RDB vs. NoSQL

Practice

- MySQL
- mongoDB

DB?

- DB (Database)
  - 특정 기능을 수행하는 데 필요한 상호 관련 데이터의 모임
    - 데이터의 특성
      - 통합된, 저장된, 운영에 필요한, 공용인 데이터
      - 실시간 접근성, 계속적인 변화, 동시 공유, 내용에 의한 참조, 데이터 독립성
  - DBMS
    - DataBase Management System. 데이터베이스를 관리하는 미들웨어
    - 정의, 조작, 제어 기능을 가짐
  - DB system
    - DB, DBMS, 기계, 관리자(DBA or YOU), schema, language, user 등
    - DB를 둘러싼 모든 것

DB schema

- Schema
  - DB system의 전반적인 구조를 정의한 것 (넓은 의미)
  - 개념 스키마 (좁은 의미)
    - DB의 논리적 구조를 정의한 것
    - DB에 접근하는 방식과는 무관한, 데이터 집합 자체의 구조
    - DB가 가져야 할 개체, 관계, 제약 조건 등을 포함
    - 무결성 규칙에 대한 정의 포함
  - 넓은 의미의 스키마
    - 외부 스키마 (외부 응용 프로그램에서 DBMS를 통해서 보는 구조)
    - 내부 스키마 (물리적 구조)

DB types

- DB types
  - Relational DB (관계형 데이터베이스, RDB)
    - 각 개체는 모두 "릴레이션Relation"이라는 이름의 테이블에 모임
    - 테이블들은 "관계Relationship"를 맺을 수 있음
    - 일관성Consistency이 잘 지켜짐
  - NoSQL DB
    - RDB의 단점을 극복하기 위해 도입
    - K-V store, Document model, Graph model 등 다양한 구조를 가짐
  - etc
    - Hierarchial DB, Network DB, Object Oriented DB…

Query

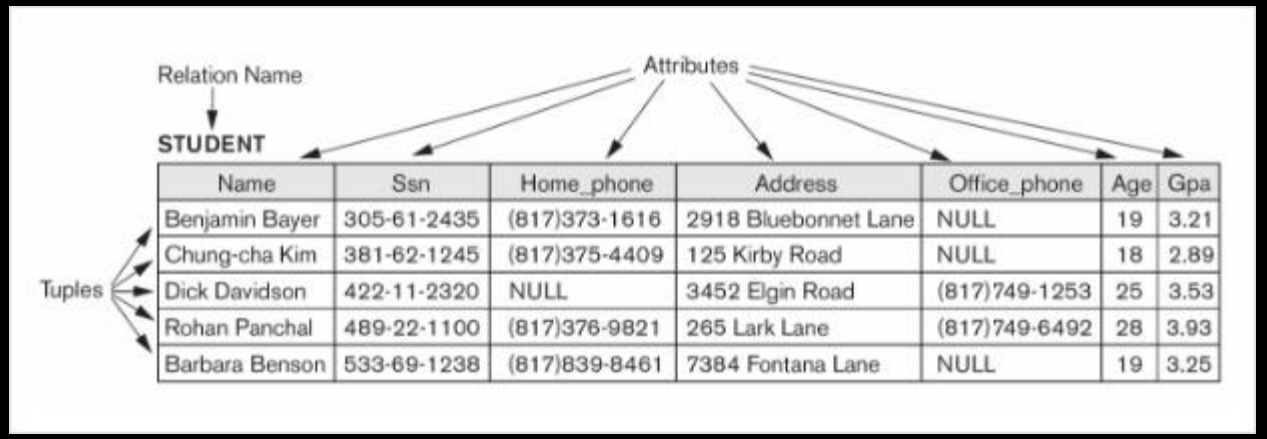
- Query
  - 질의質疑. 데이터베이스에 내리는 작업 명령
  - DB 종류에 따라 사용할 수 있는 쿼리의 종류도 변함
    - SQL을 쓰는 RDB의 경우
      - SELECT, INSERT, DELETE, UPDATE, DROP…
    - NoSQL 예시: mongoDB
      - find(), insertOne(), deleteOne(), updateOne(), …
  - SQL: RDB들이 공통적으로 지원하는 쿼리 작성 언어
    - Standard Query Language

Transaction

- Transaction
  - DB에 수행하는 논리적 작업의 단위
    - 쿼리 여러 개를 묶은 것
    - ACID 성질 (Atomic, Consistent, Isolated, Durable) 만족
      - 작업 하나가 완벽히 수행되도록 함
      - 서로 다른 작업 사이에 충돌이 일어나지 않도록 함
  - Transaction의 구조
    - Transaction 시작
    - 몇 개의 쿼리 실행 (Transaction에 쿼리들이 감싸져 있음)
    - Transaction 끝 → 에러 없을 시 Commit, 에러 발생 시 Rollback

Single Table

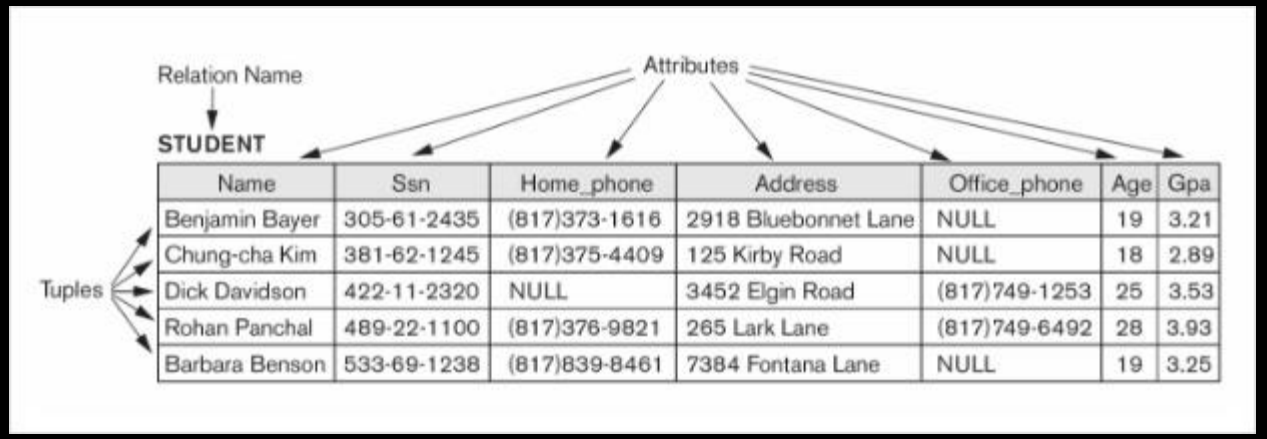
- 테이블의 구조
  - “릴레이션” = 테이블
    - 데이터를 표의 형태로 표현한 것
  - “튜플” = 레코드
    - 릴레이션을 구성하는 각각의 행
  - “속성” = 필드
    - 각 레코드가 갖는 데이터 값



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Table characteristic

- 테이블의 특징
  - 레코드
    - 서로 다른 각각의 개체를 표현하며, 순서는 없음
  - 필드
    - 필드 간의 순서는 중요하지 않음
    - 필드의 명칭은 유일하지만 레코드 사이에서 값은 동일할 수 있음
    - 원자값 저장



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Relationship

- 관계
  - 테이블 사이의 관계
    - ex) “학생” 테이블과 “수강” 테이블
      - 학생 1명이 여러 개의 수강신청 데이터를 갖고 있을 수 있음
      - 1:N 관계 (일대다 관계)
    - 관계의 종류
      - 1:1 관계, 1:N 관계, N:M 관계 (ex: “학생” 테이블과 “과목” 테이블)
      - 1:1 관계는 잘 쓰지 않음. N:M 관계는 2개의 1:N 관계로 분해할 수 있음 (“학생“-”수강”-”과목“)
    - 관계의 표현
      - 외래 키Foreign key를 통해 표현함
      - “수강” 테이블의 각 레코드가 “학생“ 테이블의 학생 ID를 필드로 가지고 있음

Consistency

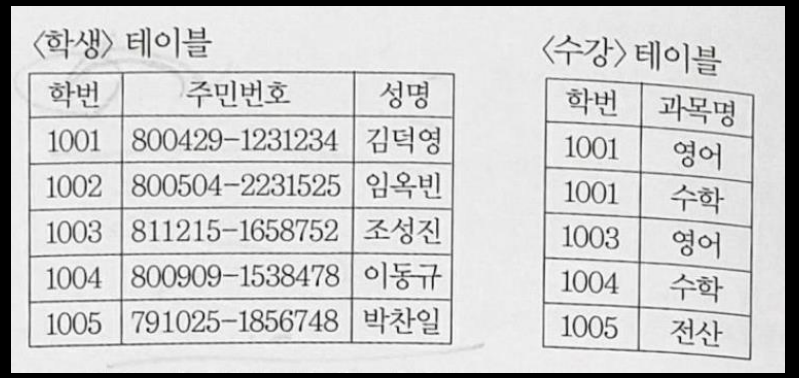
- Consistency
  - “무결성”
  - 정보에 모순이 생기지 않게 하는 것
    - ex) 학번 20200999인 학생은 존재하지 않는데, “수강“ 테이블에
      {‘id’: 234, ‘학번’: 20200999, ‘수강과목’: 32498}인 레코드 존재 → 모순
      - “학번” 테이블에 추가가 일어나지 않았을 가능성?
  - RDB의 중요한 기능
    - 관계가 있는 테이블들을 살펴보고, 모순이 생기는 쿼리에 대해서는
      오류를 발생시켜 잘못된 정보가 들어오는 것(무결성 손상)을 차단

ACID

- ACID 원칙
  - Transaction이 가져야 할 성질. 대부분의 RDB가 지원
  - Atomicity (원자성)
    - 명령이 완전히 실행되거나, 완전히 실행되지 않아야 한다.
  - Consistency (일관성, 무결성)
    - 명령 이후 모순이 없어야 한다.
  - Isolation (독립성)
    - 하나의 Transaction 안에서는 다른 작업의 영향을 받지 않는다.
  - Durability (지속성)
    - DB system은 데이터를 정확히 저장하고 보호한다.

Key

- Key
  - DB에서 탐색 혹은 정렬을 수행할 때 기준이 될 수 있는 필드
  - 후보 키
    - 유일성이 성립하는 필드. 기본 키로 사용할 수 있음. ex) “학생.학번“, “학생.주민번호”
  - 기본 키
    - 한 테이블에서 레코드를 유일하게 구분하기 위한 필드
    - 동일한 값 존재 불가. ex) “학생.학번“
  - 외래 키 (외부 키)
    - 다른 테이블의 기본 키를 참조하기 위해 가져온 것
    - 관계를 설명. ex) “수강.학번”



〈학생〉 테이블

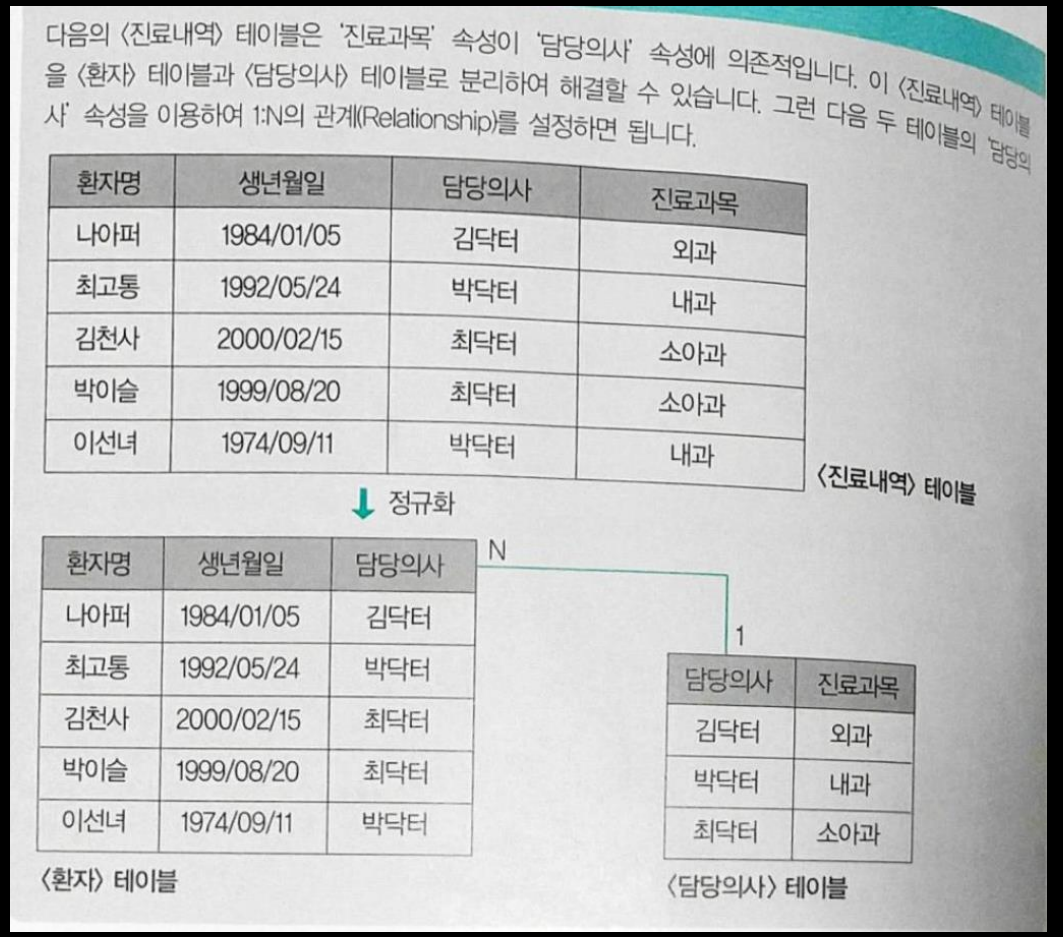
학번	주민번호	성명
1001	800429-1231234	김덕영
1002	800504-2231525	임옥빈
1003	811215-1658752	조성진
1004	800909-1538478	이동규
1005	791025-1856748	박찬일

〈수강〉 테이블

학번	과목명
1001	영어
1001	수학
1003	영어
1004	수학
1005	전산

Normalization

- 정규화Normalization
  - RDB의 관계형 스키마를 쪼개는 것
    - 중복 데이터가 존재하는 테이블 1개를 쪼개어 테이블 여러 개로 만듦
    - 쪼개진 테이블들은 일대다 관계를 갖게 됨
  - 정규화의 종류
    - 1NF, 2NF, 3NF, BCNF…
  - 정규화의 장점
    - 효과적인 검색 가능, 이상Anomaly 방지
    - 용량 축소, 불필요한 로직과 쿼리 제거



다음의 〈진료내역〉 테이블은 '진료과목' 속성이 '담당의사' 속성에 의존적입니다. 이 〈진료내역〉 테이블을 〈환자〉 테이블과 〈담당의사〉 테이블로 분리하여 해결할 수 있습니다. 그런 다음 두 테이블의 '담당의사' 속성을 이용하여 1:N의 관계(Relationship)를 설정하면 됩니다.

환자명	생년월일	담당의사	진료과목
나아퍼	1984/01/05	김닥터	외과
최고통	1992/05/24	박닥터	내과
김천사	2000/02/15	최닥터	소아과
박이슬	1999/08/20	최닥터	소아과
이선녀	1974/09/11	박닥터	내과

〈진료내역〉 테이블

↓ 정규화

환자명	생년월일	담당의사
나아퍼	1984/01/05	김닥터
최고통	1992/05/24	박닥터
김천사	2000/02/15	최닥터
박이슬	1999/08/20	최닥터
이선녀	1974/09/11	박닥터

〈환자〉 테이블

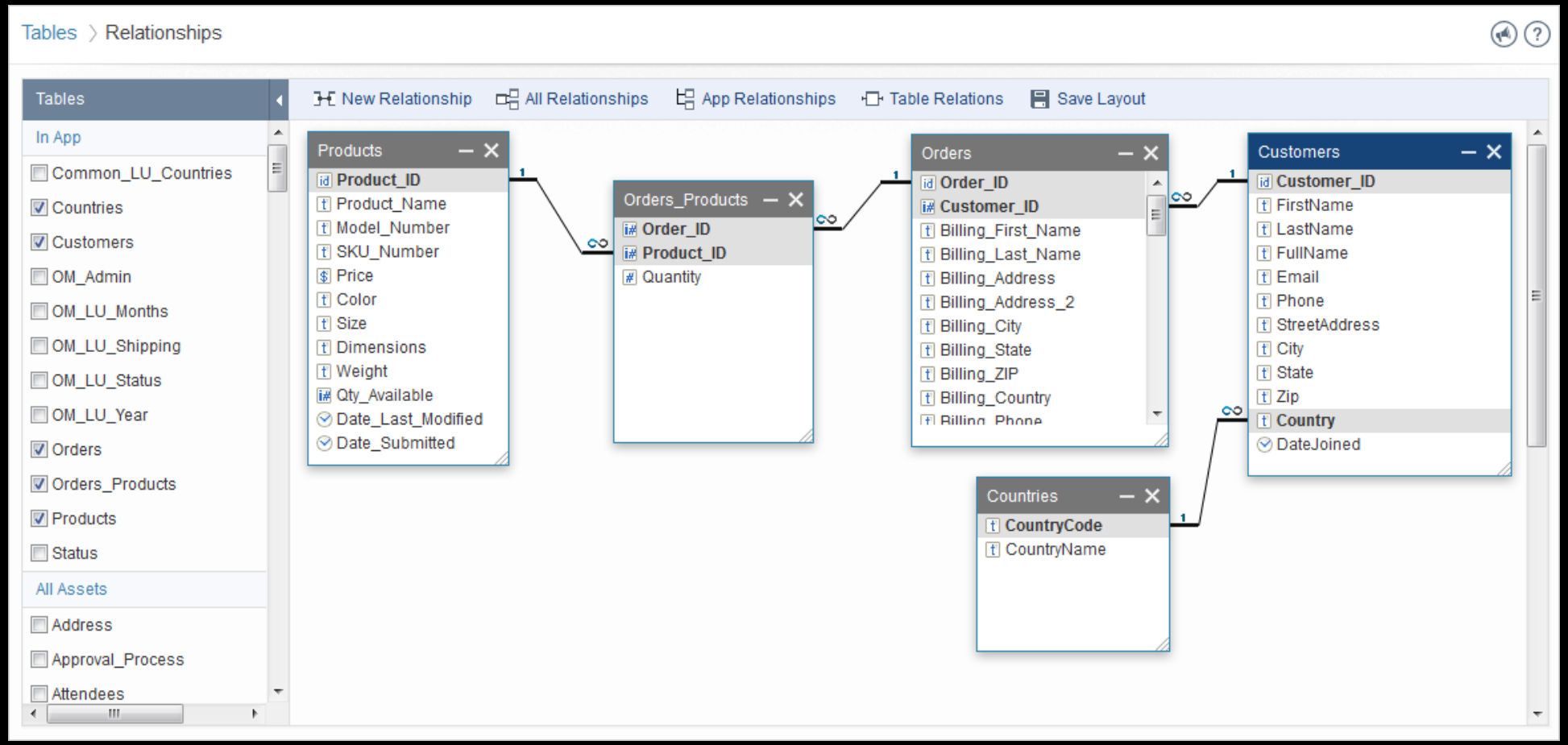
N — 1

담당의사	진료과목
김닥터	외과
박닥터	내과
최닥터	소아과

〈담당의사〉 테이블

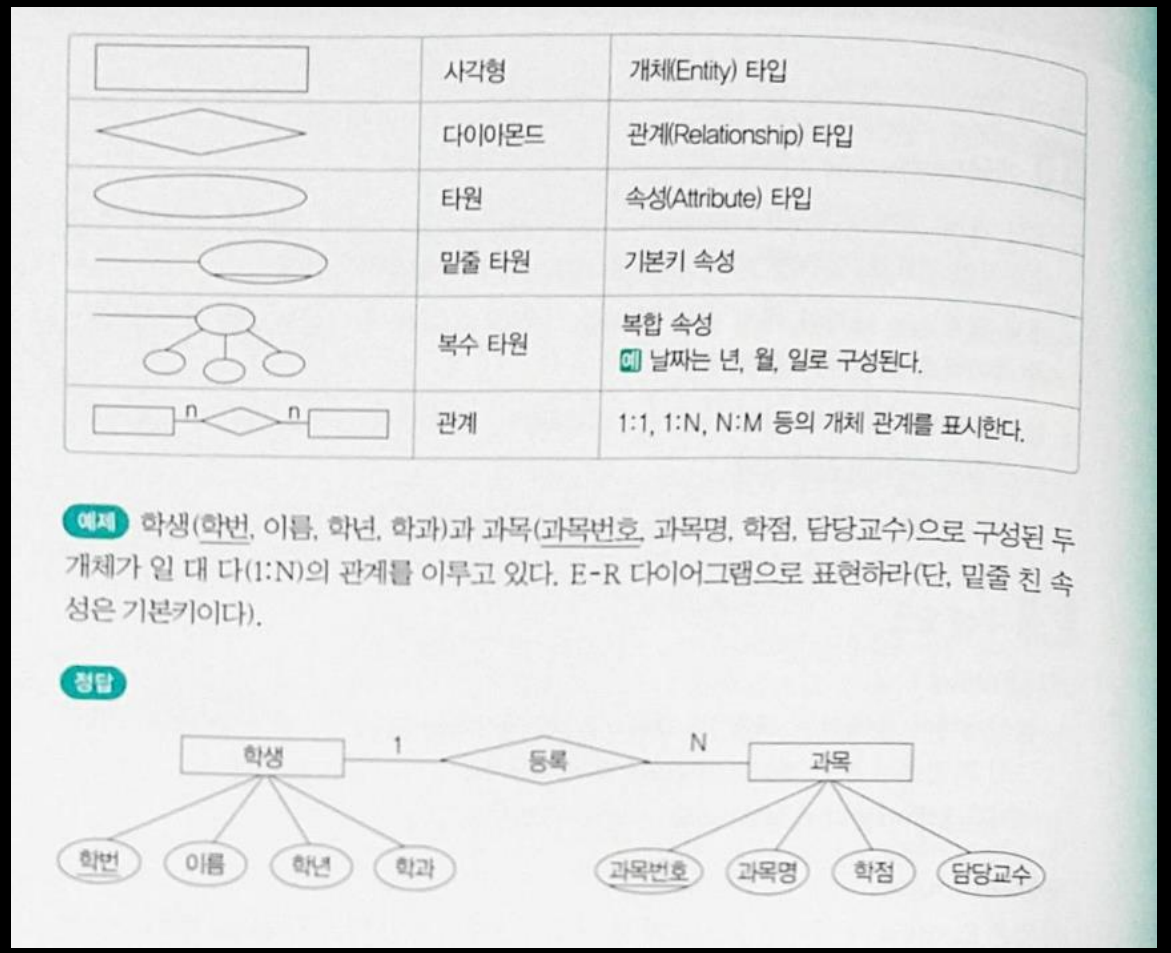
~/Relational_DB/RDB_schema_representation/ >

Tables



E-R diagram

- E-R diagram
  - 개체 관계도
  - Entity-Relationship Diagram
    - 개체, 속성, 관계 등으로 구성
    - 현실세계를 개념적인 논리 데이터로 표현하는 방법
    - 정보처리기사 빈출



	사각형	개체(Entity) 타입
	다이아몬드	관계(Relationship) 타입
	타원	속성(Attribute) 타입
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 날짜는 년, 월, 일로 구성된다.
n n	관계	1:1, 1:N, N:M 등의 개체 관계를 표시한다.

예제 학생(학번, 이름, 학년, 학과)과 과목(과목번호, 과목명, 학점, 담당교수)으로 구성된 두 개체가 일 대 다(1:N)의 관계를 이루고 있다. E-R 다이어그램으로 표현하라(단, 밑줄 친 속성은 기본키이다).

정답

SQL

- SQL
  - [에스큐엘] 혹은 [시퀄]. RDB에 대한 쿼리를 수행하는 표준적인 언어
  - Standard Query Language
    - (주로) 키워드는 대문자로 씀. 그러나 SQL 구문 자체는 대소문자 구분이 없음
    - 문자열은 따옴표로 감싸져 있어야 함
      - ‘Hello world’, “SPARCS”
    - 숫자는 따옴표를 쓰지 않음
      - 0x41, 1234
    - # 이후의 구문은 주석
    - 구문의 끝에는 세미콜론을 씀
    - DB나 테이블의 이름은 `dbname`처럼

Basic queries

- SQL Basics
  - DB 나열, 생성, 사용
    - SHOW DATABASES; CREATE DATABASE dbname; USE dbname;
  - 데이터 조회, 추가, 수정, 삭제
    - SELECT `col1`, `col2`, `col3` FROM `tablename` WHERE [condition];
    - INSERT INTO `tablename` (col1, col2, col3) VALUES (val1, val2, val3);
    - UPDATE `tablename` SET `fieldname` = [new value] WHERE [condition];
    - DELETE FROM `tablename` WHERE [condition];

Join

- Join
  - 관계를 맺고 있는 테이블 사이의 통합 쿼리
  - Join의 종류
    - Inner join: 양쪽 테이블 모두에 존재하는 경우만 가져오기
    - Left join: 왼쪽 테이블에서는 모두 가져오기
    - Right join: 오른쪽 테이블에서는 모두 가져오기
    - Cross join: 조건 없이 합쳐서 가져오기
      - 집합의 곱집합 개념 (Cartesian product)

So, a SQL query walks into a bar and sees two tables.
It walks up to them and says “Can I join you?”

Orders

OrderID	CustomerID	Order Date
10308	2	2019-06-27
10309	33	2019-04-05
10310	13	2018-12-25

Customers

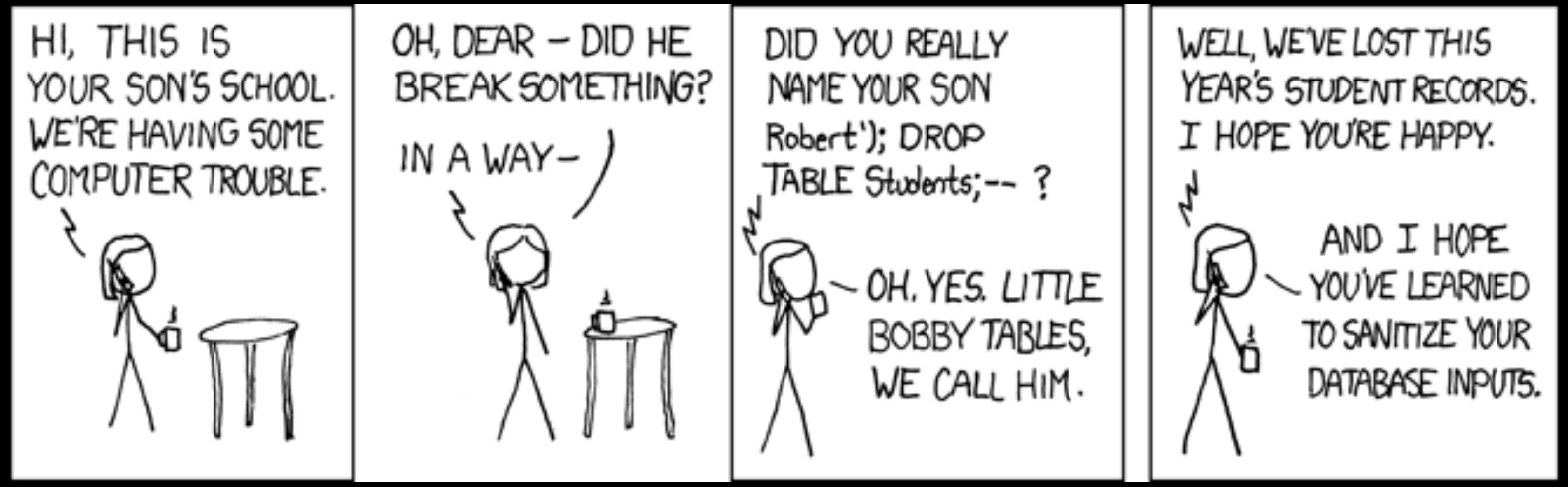
CustomerID	First	Last	Country
1	James	Bond	United States
2	Juan	Antonio	Mexico
3	JiSung	Park	South Korea

```
SELECT Orders.OrderID,
Customers.First,
Customers.Last
FROM Orders
INNER JOIN Customers ON
Orders.CustomerID=
Customers.CustomerID;
```

OrderID	First	Last
10308	Juan	Antonio

SQL Injection

- SQL Injection
  - SQL 언어의 취약점을 이용한 공격 방식
    - 외부에서 온 문자열이 SQL 쿼리문에 그대로 삽입되면 임의 코드 실행 가능
    - SQL prepared statement: 미리 쿼리를 컴파일하고 문자열만 넣는 방어 방법



MySQL DB

- MySQL
  - [마이에스큐엘]. 가장 많이 쓰이는 RDB
  - 오픈소스, 무료
  - Oracle 사의 인수
    - MariaDB, PostgreSQL 등의 대체재 등장
  - 제로보드, 그누보드 등에서 이용
  - phpMyAdmin

Limits of SQL & RDB

- RDB의 한계
  - Transaction, ACID
    - ACID 원칙을 지키기 위해, 쿼리 실행 시 DB 전체에 lock이 걸림
    - 초대규모 DB 운용 시 성능 하락
      - 빠르게 만들어지는 빅데이터를 모두 저장하는 DB의 필요성 등장
    - DB computer 하나의 성능을 올리는 데 한계가 발생
  - 일관성, 무결성을 (조금) 포기하는 대신, 성능과 확장성을 추구하는 시도
    - RDB가 아닌 다양한 데이터 모델이 만들어짐
      - K-V store, Document model, Graph model, …

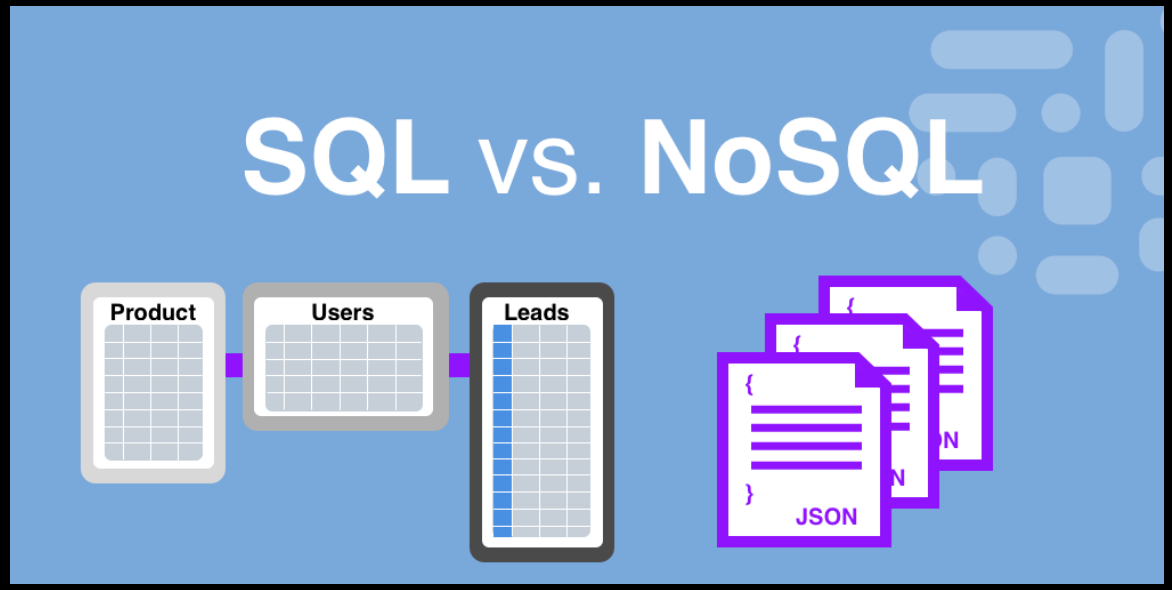
K-V store

- K-V store
  - Key-Value store.
    - 가장 간단한 DB 형태
    - 커다란 Hash table과 같음
    - Key로만 각 value를 검색할 수 있음

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Document model

- Document model
  - RDB와 비슷하지만 새로운 이름
    - 테이블 대신 “Collection”
    - 레코드 대신 “Document”
  - 스키마가 엄격하지 않음
    - 한 collection 안의 여러 Document들이 서로 다른 속성을 갖고 있을 수 있음
    - “스키마가 없다”라고도 표현함
  - 일관성을 어느 정도 포기

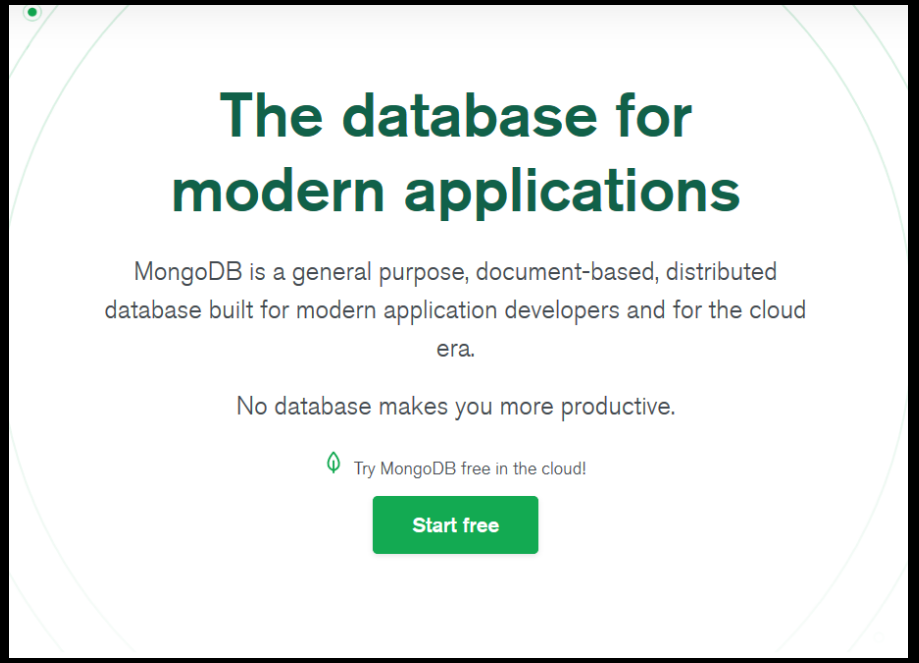


Scaling

- Scaling
  - 높은 성능이 요구될 때, 서버를 확장하는 것
  - 수직적 확장
    - 하나의 서버 성능을 높이는 것. RDB에서는 거의 수직적 확장만 가능
  - 수평적 확장
    - 서버를 여러 개로 늘리고, 데이터를 분산하여 저장하는 것
    - 사실상 NoSQL DB에서만 가능함 (분산 서버 전체에 lock을 걸기 어려움)
  - NoSQL의 장점
    - 빅데이터를 분산하여 저장. ex) F***book에 글을 쓸 때, DB가 RDB라면 글을 쓰는 Transaction을 위해 lock을 거느라 글을 쓰는 데 한참 걸릴 것

mongoDB

- mongoDB
  - 대표적인 Document model NoSQL DB
  - 각 document는 하나의 json
    - Object-oriented
    - Document가 가진 필드가 원자값이 아니라 json 배열, 객체 등일 수 있음
  - 유사-RDB 기능
    - 분산 처리가 가능한 ACID transaction 지원
      - Snapshot isolation
    - Join 지원
    - 느림



In-memory DB

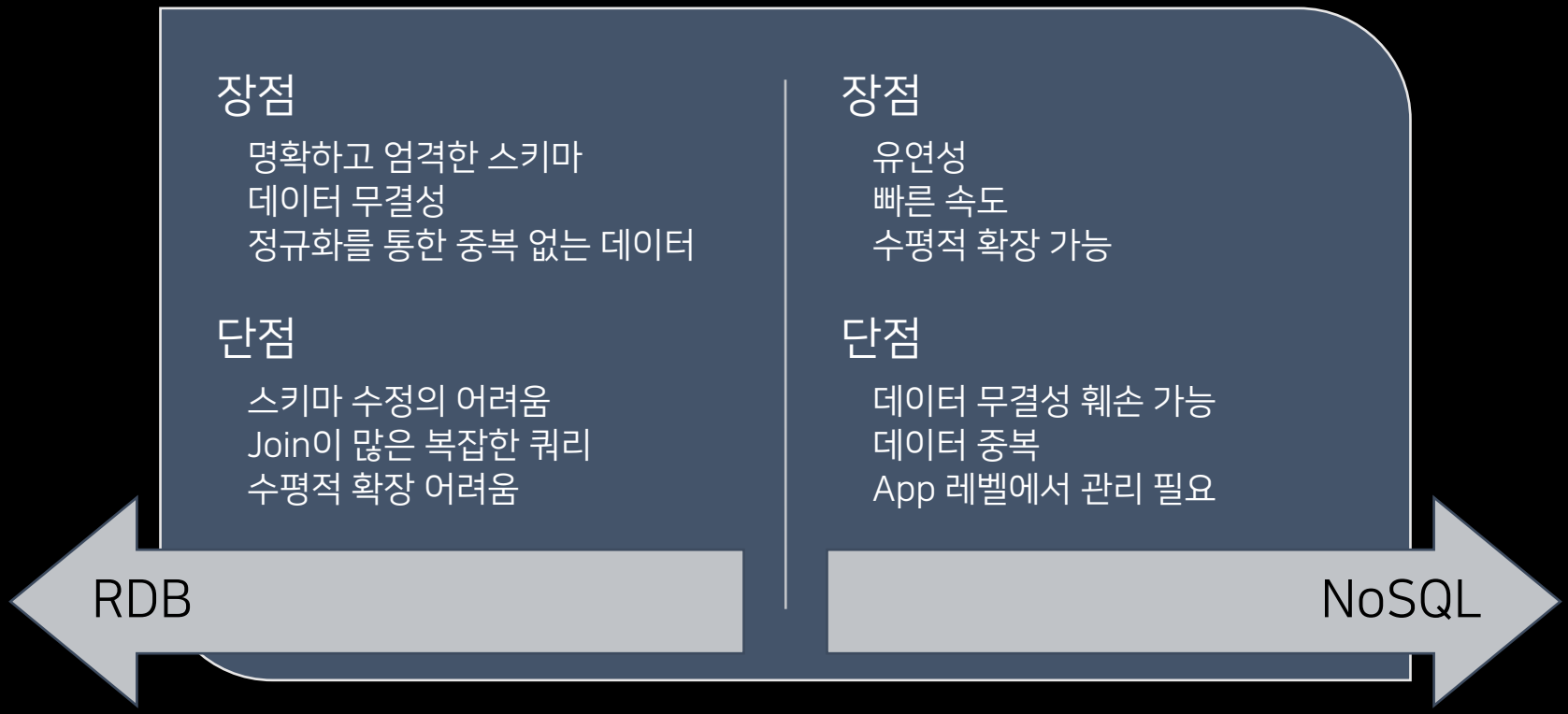
- In-memory DB
  - 메모리 위에 데이터를 저장하는 DB
  - 매우 빠른 처리를 필요로 하는 경우 사용
    - 자주 필요한 데이터의 캐싱
    - 예시: django caching
  - 프로세스 간 데이터 공유 방법
    - Message queue
    - 예시: django-channels

```
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
        }
    }
}
```

redis DB

- redis
  - “remote dictionary server”. 대표적인 in-memory DB
  - K-V store 구조
    - memcached 등 다른 in-memory K-V store와는 다르게 value에 list, set, hash table 등 다양한 값을 넣을 수 있음
    - 메모리에 올라간 DB가 정기적으로 보조기억장치에 백업되므로, 데이터가 완전히 손실될 일은 없음

RDB vs. NoSQL


장점
명확하고 엄격한 스키마
데이터 무결성
정규화를 통한 중복 없는 데이터
단점
스키마 수정의 어려움
Join이 많은 복잡한 쿼리
수평적 확장 어려움
RDB
장점
유연성
빠른 속도
수평적 확장 가능
단점
데이터 무결성 훼손 가능
데이터 중복
App 레벨에서 관리 필요
NoSQL

Installation (1)

- 설치 및 비밀번호 설정
  - sudo apt update
  - sudo apt install build-essential mysql-server #mysql 8.0 설치
  - sudo mysql
    - mysql > USE mysql;
    - mysql > ALTER USER 'root'@'localhost' IDENTIFIED WITH caching_sha2_password BY ‘new_password’;
    - mysql > FLUSH PRIVILEGES;
    - mysql > quit; # ‘root’라는 DB user로 MySQL에 접속할 수 있도록 해 주는 과정
  - sudo service mysql restart

Installation (2)

- 접속하기
  - mysql -u root -p
    - # -u: DB 접속 사용자 지정.
    - # -p: 비밀번호를 입력하겠다는 뜻.
    - # -h: 호스트 지정. (원격 접속 시)
    - # -P: 포트 지정. (원격 접속 시)
  - mysql > SHOW DATABASES;
    - 현재 존재하는 DB들 보기
    - MySQL 프로그램 안에는 여러 개의 DB가 존재. 기본적으로 MySQL 구동에 필요한 4개가 만들어져 있음

```
ubuntu@ip-172-31-36-156:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.22-0ubuntu0.20.04.3 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql>
```

Creation (1)

- DB와 테이블 만들기
  - mysql > CREATE DATABASE wheel;
    - wheel이라는 DB를 만듦
  - mysql > USE wheel;
    - 방금 만든 DB 안에서 작업하겠다고 선언하는 것
  - mysql > SHOW TABLES;
    - 현재 wheel에 존재하는 테이블 조회
  - mysql > CREATE TABLE users (name varchar(10) NOT NULL, num int(3) NOT NULL, uid varchar(20) NOT NULL, CONSTRAINT USERS_PK PRIMARY KEY (uid));

```
mysql> CREATE DATABASE wheel;
Query OK, 1 row affected (0.01 sec)

mysql> USE wheel;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

~/Practice/MySQL/ >

Creation (2)

```
mysql> create table users (name varchar(10) NOT NULL, num int(3) NOT NULL, uid varchar(20) NOT NULL, CONSTRAINT USERS_PK PRIMARY KEY (uid));
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> show tables;
+----------------+
| Tables_in_wheel |
+----------------+
| users          |
+----------------+
1 row in set (0.01 sec)

mysql> select * from users;
Empty set (0.01 sec)

mysql>
```

wheel이라는 DB에 테이블 `users`를 만들되, 필드는
Null일 수 없는 최대 10자 문자열인 `name`,
Null일 수 없는 3byte 정수인 `num`,
Null일 수 없는 최대 20자 문자열인 `uid`
를 두고, 제약 조건은
`uid`를 기본 키로 만드는 `USERS_PK`라는 이름의 제약 조건
을 두도록 하라.

```
mysql> desc users;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(10) | NO   |     | NULL    |       |
| num   | int         | NO   |     | NULL    |       |
| uid   | varchar(20) | NO   | PRI | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Insertion

- 레코드 추가하기
  - mysql > INSERT INTO users (name, num, uid) VALUES (“Name", 20, “uid");
    - 한 명 추가
  - mysql > INSERT INTO users (name, num, uid) VALUES (“Name”, 20, “uid”), (“Name”, 20, “uid”);
    - 여러 명 추가
  - mysql > SELECT * FROM users;
    - 모든 유저의 모든 필드 조회
  - mysql > SELECT name, num FROM users WHERE num = 20;
    - 20학번인 유저의 이름, 학번만 조회

Update & Delete

- 레코드 수정하기
  - UPDATE users SET num=21 WHERE uid='ivy';
    - 학번 값 수정
- 레코드 삭제하기
  - DELETE FROM users WHERE uid='ivy';
    - 레코드 삭제

WHERE

- WHERE 조건식
  - “=“ : 같다
  - “>”, “<“, “>=“, “<=“ : 일반적인 부등호
  - “<>” : 같지 않다
  - “LIKE”: 어떤 표현식에 맞는 것만
    - ex) “%A%”: 중간에 A을 포함하는 문자열
    - ex) “A__”: A로 시작해서 두 개의 문자가 더 있는 문자열

Assignment

- 실습 제출
  - 1. `wheel` DB를 만들고 선택하기, `user` 테이블 만들기
  - 2. 자신의 이름, 학번, SPARCS uid가 나오도록 insert하기
  - 3. ivy 회원의 학번을 update하기
  - 4. uid에 'n' 문자가 포함된 레코드를 delete하기
  - 5. 모든 필드와 레코드를 select하기
  - 각 작업 과정을 스크린샷으로 찍어 제출!

Installation

- 설치 및 시동
  - 설치 과정 (https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/)
    - wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add –
    - echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
    - sudo apt-get update
    - sudo apt-get install -y mongodb-org
  - sudo service mongod start

Creation (1)

- 실행 및 생성
  - mongo
    - mongo shell을 불러오는 명령어. 옵션을 주면 localhost가 아닌 다른 서버의 mongod daemon에도 연결할 수 있다
  - > use wheel
    - wheel이라는 DB를 사용하겠다고 선언 (명시적으로 만들지 않아도 됨)
  - > show dbs
    - DB 목록 보기. 아직 wheel DB에는 collection이 없으므로 wheel DB가 나타나지 않음

Creation (2)

- 컬렉션 작업
  - > db.createCollection(“users”)
    - { “ok” : 1 } wheel DB 안에 users라는 Collection이 생성
  - > show collections
    - 방금 만든 collection이 나타남
  - > show dbs
    - wheel에 collection이 있으므로, 이제 나타남

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use wheel
switched to db wheel
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.createCollection("users")
{ "ok" : 1 }
> show collections
users
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
wheel   0.000GB
>
```

Insertion

- Document 추가하기
  - > db.users.insertMany([
    … {“name”: ”Name”, “num”: 20, “uid”: “uid”},
    …])
    - Collection에 여러 개의 document를 삽입
  - > db.users.find()
    - 전부 조회
    - > db.users.find.pretty() 하면 깔끔하게 보여줌

Giving criteria

- Criteria
  - > db.users.find({ “num”: { $gte: 20, $lte: 22 } })
    - 20학번부터 22학번까지의 유저 조회
  - > db.users.find({ $and: [{"name":“Name"}, {"uid":"ivy"}] })
    - 이름이 “Name”이고 uid가 “ivy”인 유저 조회

operator	설명
$or	주어진 조건중 하나라도 true 일 때 true
$and	주어진 모든 조건이 true 일 때 true
$not	주어진 조건이 false 일 때 true
$nor	주어진 모든 조건이 false 일때 true

operator	설명
$eq	(equals) 주어진 값과 일치하는 값
$gt	(greater than) 주어진 값보다 큰 값
$gte	(greather than or equals) 주어진 값보다 크거나 같은 값
$lt	(less than) 주어진 값보다 작은 값
$lte	(less than or equals) 주어진 값보다 작거나 같은 값
$ne	(not equal) 주어진 값과 일치하지 않는 값
$in	주어진 배열 안에 속하는 값
$nin	주어빈 배열 안에 속하지 않는 값

~/Practice/mongoDB/ >

Update & Delete

- Update
  - > db.users.updateMany({"num": 20}, {$set: {"num": 21}})
    - updateMany([criteria], {$set: [operation]}) 꼴
    - 20학번을 모두 21학번으로 수정 (미개봉 중고 ㅠㅠ)
    - updateMany가 아니라 update하면 단 하나만 update됨
    - schema가 엄격하지 않으므로, 새로운 속성을 추가할 수도 있음
- Delete
  - > db.users.deleteMany({“uid”: “ivy”})
    - 조건에 맞는 유저 삭제

Assignment

- 실습 제출
  - 1. `wheel` DB를 만들고 선택하기, `users` 컬렉션 만들기
  - 2. 자신의 이름, 학번, SPARCS uid가 나오도록 insert하기
  - 3. 미개봉 중고들의 학번을 정정하기
  - 4. uid에 'n' 문자가 포함된 document에 {"has_n":1} 속성 추가하기
    - 힌트: 연산자 $regex
  - 각 과정과 사이사이에 find()한 결과를 스크린샷으로 제출!

Q&A

wheel seminar