KWEB Study Week3: Introduction to DB

KWEB 2학기 준회원 스터디

Today's Contents

1. Before Study
2. DB Basics
3. Relational DB
4. NoSQL 및 실습
5. 과제

M.G. BAE

J.H. BAEK

2017 Fall Semester

Before Study

- 지난 2주차에 우리는 Object & Function, Asynchronous 개념, Clojure 에 대하여 배웠습니다.
- 3주차에는 웹 서비스에서 중요한 부분을 차지하는 DB, DBMS에 대해 간단히 알아보고, NodeJS를 이용하여 MongoDB에 데이터를 저장해 볼 것 입니다.
- 실제로 DB는 컴과 3학년 전공과목이기도 합니다. 전공만큼 깊이는 아니더라도! 서비스에 필요한 기초만큼 익히는게 목표입니다~
- 뭔가 지금까지와는 다른 느낌일거에요! 지난 주차 복습과 함께 시작하겠습니다!

Previous Study: Object & Function

- Object
  - Javascript는 객체(Object) 기반의 스크립트 프로그래밍 언어이다.
  - 객체지향 언어에서 객체는 데이터(주체)와 그 데이터에 관련되는 동작(절차,방법,기능) 을 모두 포함하고 있는 개념이다.
  - 모든 객체는 자신의 속성값(Property)과 행동을 정의 하는 메소드(Method)를 가진다.

- Function
  - Javascript에서 함수는 1급 객체이며 일반 객체와 동일하게 다루어 진다. (변수! 인자! 반환값!)
  - 함수는 모든 객체에서 사용 가능하므로 함수를 "전역 메서드"라고도 합니다.
  - 뭐 함수는 많이 보셨을테니! 웬만큼은 아시죠? 넘어갈게요~

Previous Study: Asynchronous & Clojure

- Asynchronous
  - 응용프로그램이나 웹 환경에서는 비동기 처리가 필수적이다. (커피숍 설명 기억하시죠?)
  - **Javascript는 대부분의 처리를 비동기로 진행한다.**
  - 비동기 처리 이후 동작을 정의 하는 것을 콜백이라 한다.

- Clojure
  - **Javascript에서 함수는 자신이 선언된 시점의 변수 선언 정보를 기억한다.**
  - 사실 어려운 개념입니다. 당장은 이해가 안 가도 괜찮습니다.
  - Javascript를 통한 개발에 익숙해지면 자동적으로 알게 될겁니다.
  - 다른 프로그래밍 언어들의 Closure과도 비슷합니다.

DB (Database)

- DB: 여러 사람에 의해 공유되어 사용될 목적으로 통합하여 관리되는 데이터의 집합

- 체계화된 집합인 DB는 아래와 같은 특성들을 지닙니다.
  - 실시간 접근성 → 실시간으로 동작가능해야 함
  - 지속적인 변화 → CRUD(Create, Read, Update, Delete) 를 통한 데이터 관리가 가능해야 한다.
  - 동시 공유 → 여러 사용자가 동시에 같은 내용의 데이터를 이용가능해야 한다.
  - 내용에 대한 참조 → 사용자가 자료의 내용을 통해 자료를 찾을 수 있어야 한다.

- 오늘날 같은 데이터 중심 경제에서 DB는 이미 널리 사용되고 있고! 여러분들도 사실 많이 들어 보셨을 것이고 사실상 사용했습니다! 웹 상의 모든 데이터는 대체로 DB에 저장되기 때문이죠! 게시판만 생각해봐도 위의 특성들을 느낄 수 있지않나요..?

DB (Database)

- 개념상으로는 텍스트 파일도 DB라고 할 수 있습니다.
  - Node.js에서 사용하는 package.json 도 일종의 DB 입니다.

- 오른쪽은 DB 중 하나인 MariaDB의 Table 입니다.

- 하지만 DB의 특성을 만족하도록 이용하기는 상당히 어렵습니다. 그를 가능케 하는 무언가가 필요하겠죠?

```
MariaDB [minglecon2]> select * from wp_terms LIMIT 10;
+---------+-------------------------+-------------------------+------------+
| term_id | name                    | slug                    | term_group |
+---------+-------------------------+-------------------------+------------+
|       1 | 미분류                  | uncategorized           |          0 |
|       3 | Previously used menu 3  | previously-used-menu-3  |          0 |
|       4 | Previously used menu 1  | previously-used-menu-1  |          0 |
|       5 | Previously used menu 2  | previously-used-menu-2  |          0 |
|       6 | simple                  | simple                  |          0 |
|       7 | grouped                 | grouped                 |          0 |
|       8 | variable                | variable                |          0 |
|       9 | downloadable            | downloadable            |          0 |
|      10 | virtual                 | virtual                 |          0 |
|      11 | pending                 | pending                 |          0 |
+---------+-------------------------+-------------------------+------------+
```

DBMS

- DBMS (Database Management System): 데이터베이스를 관리하며 응용 프로그램들이 데이터베이스를 공유하며 사용할 수 있는 환경을 제공하는 소프트웨어

- 데이터베이스 자체만으로는 거의 아무 것도 못합니다! 따라서 그걸 관리하는 시스템이 필요하겠죠? 그래서 보통 소프트웨어와 통합된 DBMS가 제공됩니다.

- DBMS는 데이터베이스를 구축하는 틀을 제공하고, 효율적으로 데이터를 검색하고 저장하는 기능을 제공합니다.

- 전공과목이 아니니 간단히만 짚고 넘어갑시다!

DB Type

- 사실 상세한 분류는 많지만 우리는 크게 2가지로 알아봅시다.

- Relational DB
  - 관계형 데이터베이스로 대부분의 상용 DBMS가 이 분류에 포함됩니다.
  - 형식이 정해진 테이블 단위로 이루어짐!

- NoSQL (Not only SQL)
  - SQL만을 사용하지 않는 DBMS, Not only SQL
  - 정통적인 RDBMS 데이터의 형식에 제한이 많았고, NoSQL에선 이러한 제한들을 완화시켰음!
  - 데이터의 일관성을 약간 포기한 대신 여러 대의 컴퓨터에 데이터를 분산하여 저장하는 것이 목표!

Relational DB

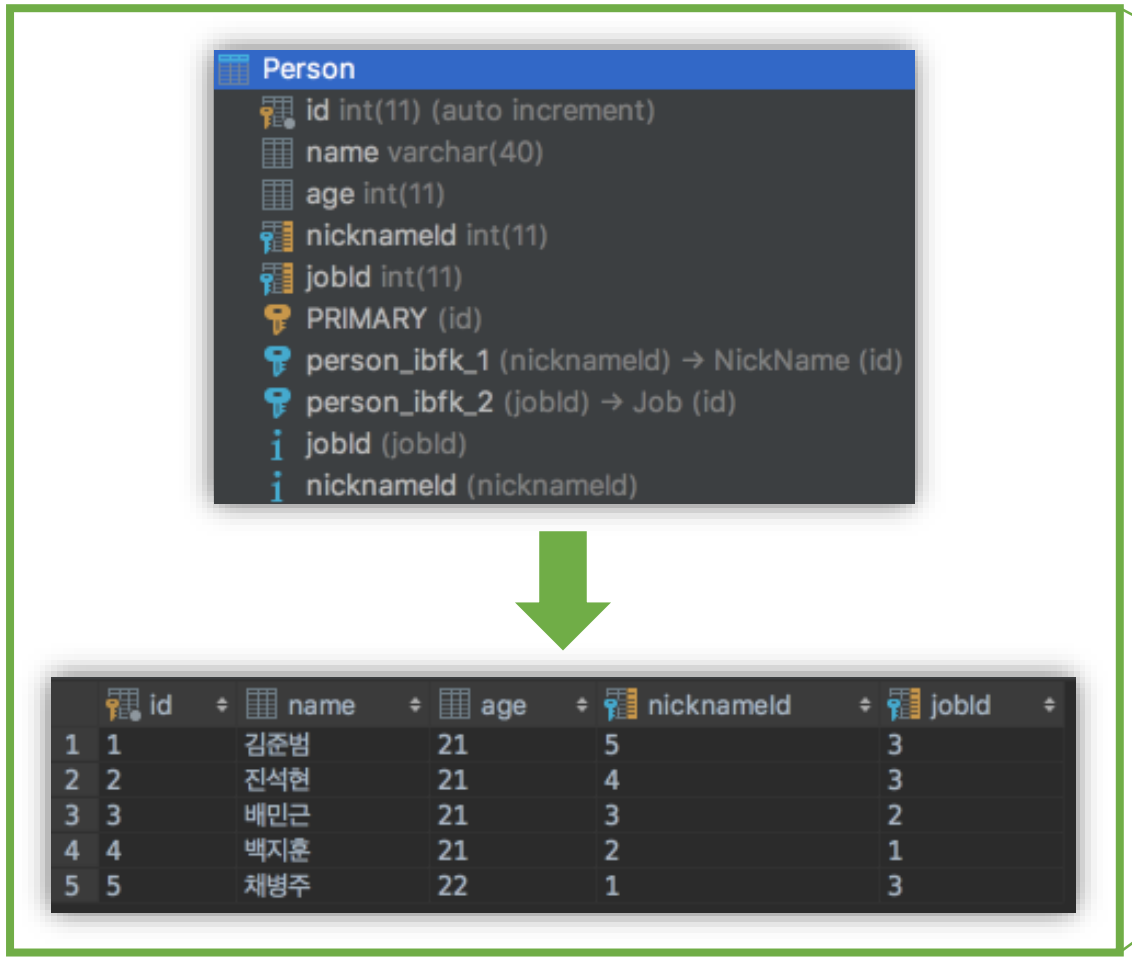
- 관계형 모델을 기반으로 하는 데이터베이스 관리 시스템이다.
  - 관계형 모델이란? → 데이터를 column과 row를 이루는 하나 이상의 테이블(또는 관계)로 정리하며, 고유 키(Primary key)가 각 row를 식별한다.

- DB계의 주류이며, 관계형 모델 기반이니 데이터를 column과 row라는 일종의 표 형태로 저장함! (한 Table의 row는 같은 길이의 column을 가져요!)

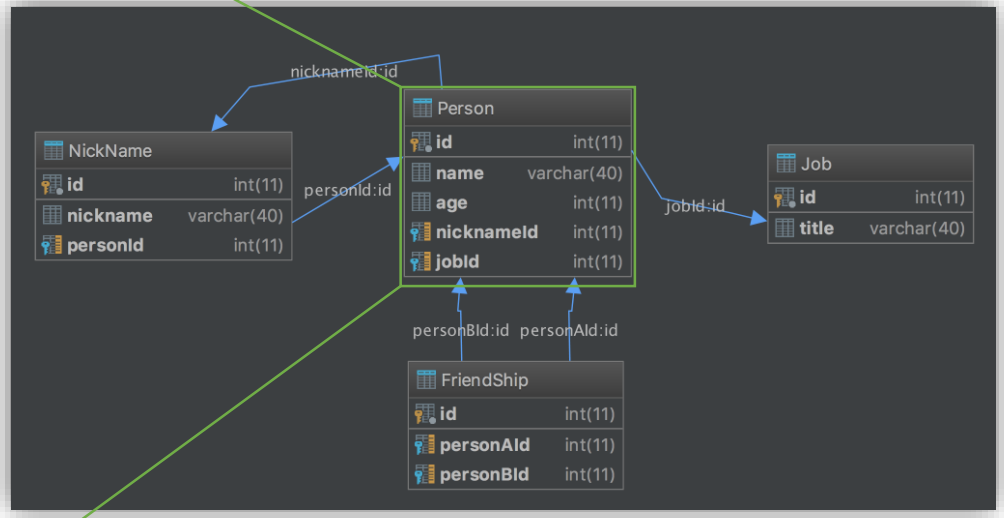
- column은 field (형식은 사전 지정함), row는 레코드 (실질적인 데이터)를 담고 있음!

- 데이터의 관계가 Table Schema로 사전 정의됩니다. (그냥 엑셀 생각해보세요)
  - Schema(스키마): 정보를 통합하고 조직화하는 인지적 개념 또는 틀 → 그냥 구조요! 구조!

Relational DB Example - MariaDB



	id	name	age	nicknameId	jobId
1	1	김준범	21	5	3
2	2	진석현	21	4	3
3	3	배민근	21	3	2
4	4	백지훈	21	2	1
5	5	채병주	22	1	3



Relational DB

- 역사가 오래된 만큼 가장 신뢰성이 높고 데이터의 분류, 정렬, 탐색 속도가 빠르다.
- 보통 SQL을 이용하여 고도로 정교한 검색 쿼리를 제공하며 상상하는 거의 모든 방식으로 데이터를 다룰 수 있게 해줍니다.
- But, schema 수정이 어렵고 데이터가 2차원 표 형태로 출력되어 객체와 합이 잘 맞지 않습니다. (보통 객체는 Tree 구조로 조직화!) 또한 부하분산이 잘되지 않음!
- 관계형 DB 종류: MySQL, MariaDB, MSSQL, Oracle, etc.

SQL

- SQL (Structured Query Language): 데이터베이스를 사용할 때, 데이터베이스에 접근할 수 있는 데이터베이스 하부 언어를 말한다.

- SQL로 질의 기능, 데이터 정의 및 조작 기능이 모두 가능합니다.
  - SQL문을 활용하여 데이터 베이스와 테이블의 생성 및 삭제 / 데이터의 생성, 수정, 검색, 삭제, 가공이 가능합니다.

- 요새는 다른 종류의 DB에도 널리 쓰이나 원래는 IBM의 Relational DB에서 사용되었습니다! (NoSQL 계열 DB를 제외하곤 대부분의 DB에서 사용됩니다.)

- SQL 구문은 DB 종류에 따라 다르지만 표준 SQL이 존재하며 대체로 이와 비슷한 형태를 띄고 있습니다!

SQL Example - MariaDB

Database 및 Table 생성

```sql
CREATE DATABASE kweb;
USE kweb;
SET FOREIGN_KEY_CHECKS = FALSE;
CREATE TABLE Person (
  id         INT AUTO_INCREMENT NOT NULL,
  name       VARCHAR(40),
  age        INT,
  nicknameId INT,
  jobId      INT,
  PRIMARY KEY (id),
  FOREIGN KEY (nicknameId) REFERENCES NickName (id),
  FOREIGN KEY (jobId) REFERENCES Job (id)
);

CREATE TABLE NickName (
  id       INT AUTO_INCREMENT NOT NULL,
  nickname VARCHAR(40),
  personId INT,
  PRIMARY KEY (id),
  FOREIGN KEY (personId) REFERENCES Person (id)
);
```

Record 삽입

```sql
INSERT INTO Person
(id, name, age, nicknameId, jobId)
VALUES
  (1, '김준범', 21, 5, 3),
  (2, '진석현', 21, 4, 3),
  (3, '배민근', 21, 3, 2),
  (4, '백지훈', 21, 2, 1),
  (5, '채병주', 22, 1, 3);

INSERT INTO NickName (id, nickname, personId) VALUES
  (1, '채시민', 5),
  (2, '백경관', 4),
  (3, '배피아', 3),
  (4, '진시민', 2),
  (5, '김시민', 1);
SET FOREIGN_KEY_CHECKS = TRUE;

INSERT INTO FriendShip (personAId, personBId)
VALUES (1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5);
```

Record 검색

```sql
USE kweb;
SELECT * FROM Person
INNER JOIN NickName ON Person.id = NickName.personId;
```

@localhost - @localhost

Result 1-2

5 rows | Tab-se...d (TSV) | View Query

	Person.id	name	age	nicknameId	jobId	NickName.id	nickname	personId
1	5	채병주	22	1	3	1	채시민	5
2	4	백지훈	21	2	1	2	백경관	4
3	3	배민근	21	3	2	3	배피아	3
4	2	진석현	21	4	3	4	진시민	2
5	1	김준범	21	5	3	5	김시민	1

RDB & SQL

- 이 파트는 나중에! Node.js에 대해 좀 더 배우고 Relational DB와 SQL을 이용하여 코딩해볼겁니다!
- 고로 이 파트에 대한 실습은 나중으로 미루고! 오늘은 NoSQL을 이용하여 DB 맛보기를 해볼겁니다.
- Relational DB는 실무에 많이 사용되므로 나중에 잘 배워놓도록 합시다! (여러분이 3학년쯤 배울 DB 전공수업에서도 사용됩니다.)
- 지금은 이런 것들이 있는 줄만 알고 넘어갑시다!

NoSQL

- Not only SQL → 데이터를 저장하는데 SQL뿐만 아니라 다른 방법이 있다!
- 보통 관계형 모델을 사용하지 않고! 는 거짓말이고 사실 씁니다.
  - 흠흠.. 왜 쓰냐 NoSQL.. (쓰는 이유가 있겠죠?)
- NoSQL은 보통 빠르고 확장성이 뛰어납니다. 사실 확장성 하나만으로 쓸 이유는 충분합니다.
- Relational DB에 존재하는 여러 제한이 없어졌으나 어려우니 그렇쿤!하고 알아둡시다.
- 대표적인 NoSQL: MongoDB, OrientDB, Hadoop, Cassandra, etc.

실습

- NoSQL은 깊게 설명하면 어려울 것 같으니 이 정도 알고 한번 사용방법을 알아봅시다!
- 실습에는 NoSQL 중 널리 사용되는 MongoDB를 사용할 것입니다. MongoDB는 NoSQL 중에서 사용하기 쉽습니다.
- 상용으로 Relational DB가 많이 사용하긴 하나 가벼운 기능들은 NoSQL로 구현될 경우도 있습니다. (DB 선택은 목적대로 하시는 겁니다.)
- 일단 MongoDB에 대해 알아보고! 한번 DB 맛보기 정도만 해봅시다~

MongoDB

- 가장 널리 쓰이는 NoSQL 계열 DB로 앞서 말했듯이 접근성이 좋습니다!

- MongoDB는 RDBMS의 테이블, 행, 열에 대응하는 개념을 가지고 있으며(오른쪽 참조), 데이터를 JSON형식으로 보관합니다.

- 데이터의 형식에 대한 제약이 없기 때문에 어떠한 형식의 데이터도 저장 가능합니다. (말이 어렵죠? 그냥 저장하면 저장된다는 뜻입니다.)

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	`$lookup`, embedded documents
primary key	primary key
Specify any unique column or column combination as primary key.	In MongoDB, the primary key is automatically set to the _id field.

실습

추가해서 재배포해드리겠습니다.

과제 (~ 10/13 23:59)

- 3주차는 DB 기초에 대해 공부했습니다. 앞으로 DB를 쓸 것이므로 미리 그냥 배워본 시간이었어요!
- 이번주 과제는 mLab에 적절한 사용예시를 만들어 올리는 것입니다. (앞의 실습 결과 제출하셔도 됩니다.)
- 제출기한: 10월 13일 자정 - 제출은 각 스터디장에게! 캡쳐든 공유하든 원하시는대로!
- 과제 제출 E-mail
  - 월 7시: 15 배민근 (baemingun@naver.com)
  - 화 7시: 16 백지훈 (bjh970913@gmail.com)

It’s all today!