

10주차

리눅스 시스템

2024.동계계절학기

CONTENTS

1. 데이터베이스

데이터베이스

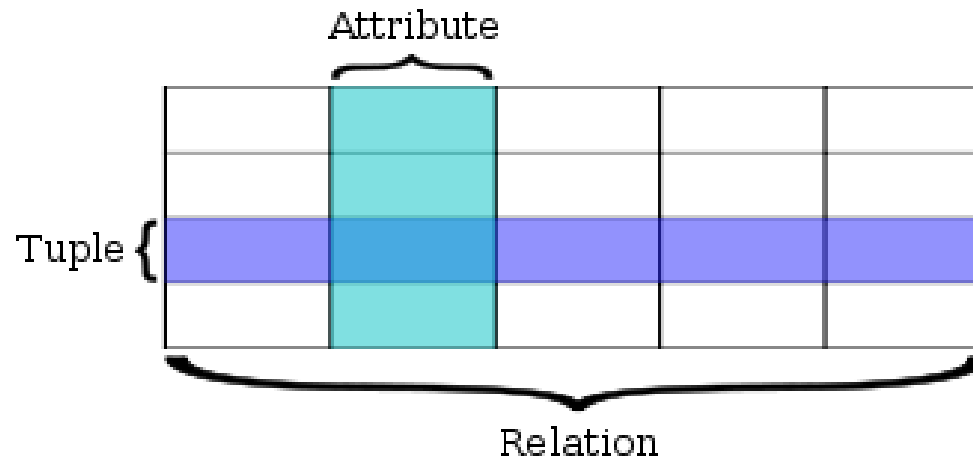
- **데이터베이스**

- 여러 사람이 데이터를 공유해서 사용하기 위해 체계적으로 통합해서 묶어놓은 데이터 집합
- 컴퓨터 시스템에서 데이터베이스는 데이터 집합을 관리하고 접근할 수 있게 하는 프로그램을 의미함

- **데이터베이스의 장점**

- 데이터 중복 최소화 및 저장공간 절약
- 효율적인 데이터 접근 및 빠른 접근 속도
- 일관성, 무결성, 보안성 제공
- 데이터 표준화

- **관계형 데이터모델(Relational Data Model)**
 - 데이터를 테이블에 대입하여 관계를 묘사하는 이론적 모델
 - 관계형 데이터모델을 다루는 데이터베이스 시스템을 관계형 데이터베이스 관리시스템(Relational DataBase Management System, RDBMS)이라고 함



- **SQL(Structured Query Language)**
 - SQL은 관계형 데이터베이스를 관리하기 위해 사용하는 질의 언어
 - 데이터베이스 스키마(Schema) 생성
 - 데이터 생성(Create)
 - 데이터 읽기(Retrieve, Read)
 - 데이터 수정(Update)
 - 데이터 삭제>Delete)
 - CRUD는 데이터를 관리하기 위해 기본적으로 필요한 기능

- **NoSQL**

- SQL이 아니라는 의미에서 NoSQL이라고 하기도 하고 “Not Only SQL”이라는 의미에서 전통적인 관계형 데이터베이스의 한계를 극복하기 위한 다양한 저장 방식을 지원함

- **NoSQL 주요 개념**

- **스키마 유연성**

- RDBMS와 달리 사전에 정의된 스키마가 필요하지 않아 데이터를 자유롭게 저장할 수 있음
- 구조가 자주 변경 되는 경우에 유리함

- **수평적 확장**

- 데이터를 여러 서버에 분산하여 저장할 수 있음
- 대규모 서비스에 유리함

- **다양한 데이터 모델**

- 키-값: Redis, DynamoDB
- 문서: MongoDB, CouchDB
- 그래프: Neo4j

CAP 이론

- **CAP Theory**

- CAP 이론은 Consistency(일관성), Availability(가용성), Partition Tolerance(분할허용)의 세가지 속성을 분산 시스템에서 동시에 완벽히 만족할 수 없다는 이론

- **CAP 이론 세가지 요소**

- **일관성(Consistency)**

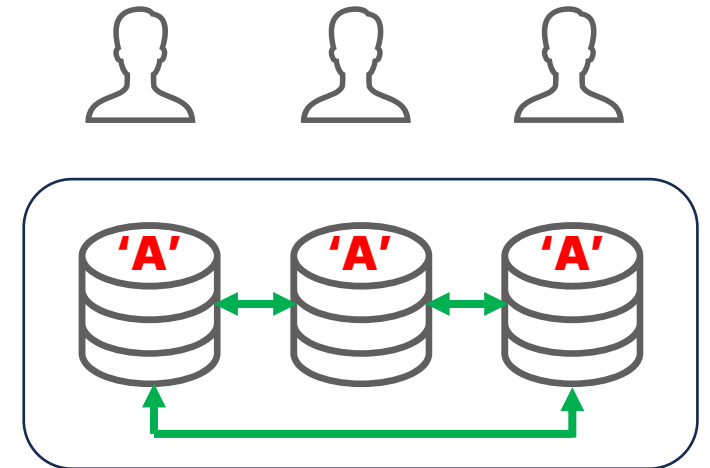
- 누가 데이터를 조회하더라도 같은 데이터를 얻을 수 있어야 함
- 즉, 한 노드에서 데이터가 업데이트 되면 모든 노드가 동일한 값을 가져야함

- **가용성(Availability)**

- 시스템이 항상 응답할 수 있는 상태를 유지

- **분할 허용성(Partition Tolerance)**

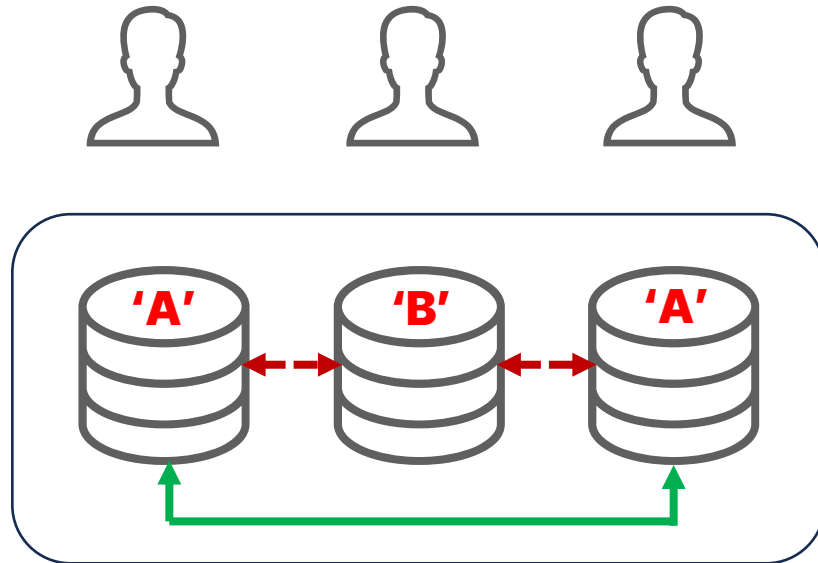
- 노드간 연결이 끊어져서 네트워크 분할이 발생해도 시스템은 계속 동작해야함



CAP 이론

- CAP가 동시에 만족하지 못하는 이유

- CAP 이론은 분산 시스템을 가정하고 있으므로, 네트워크 장애로 인해 분할이 발생한 것을 가정함
- 일관성과 가용성의 충돌
 - 네트워크 분할이 발생하면 노드 간에 데이터를 동기화 하지 못함
 - 일관성을 유지하려면 네트워크 분할 문제가 해결되어 동기화 될 때까지 가용성을 제공 못함
 - 가용성을 제공하려면 해당 데이터가 최신 데이터가 아닐지라도 응답을 해야하므로 일관성이 희생됨



CAP 이론

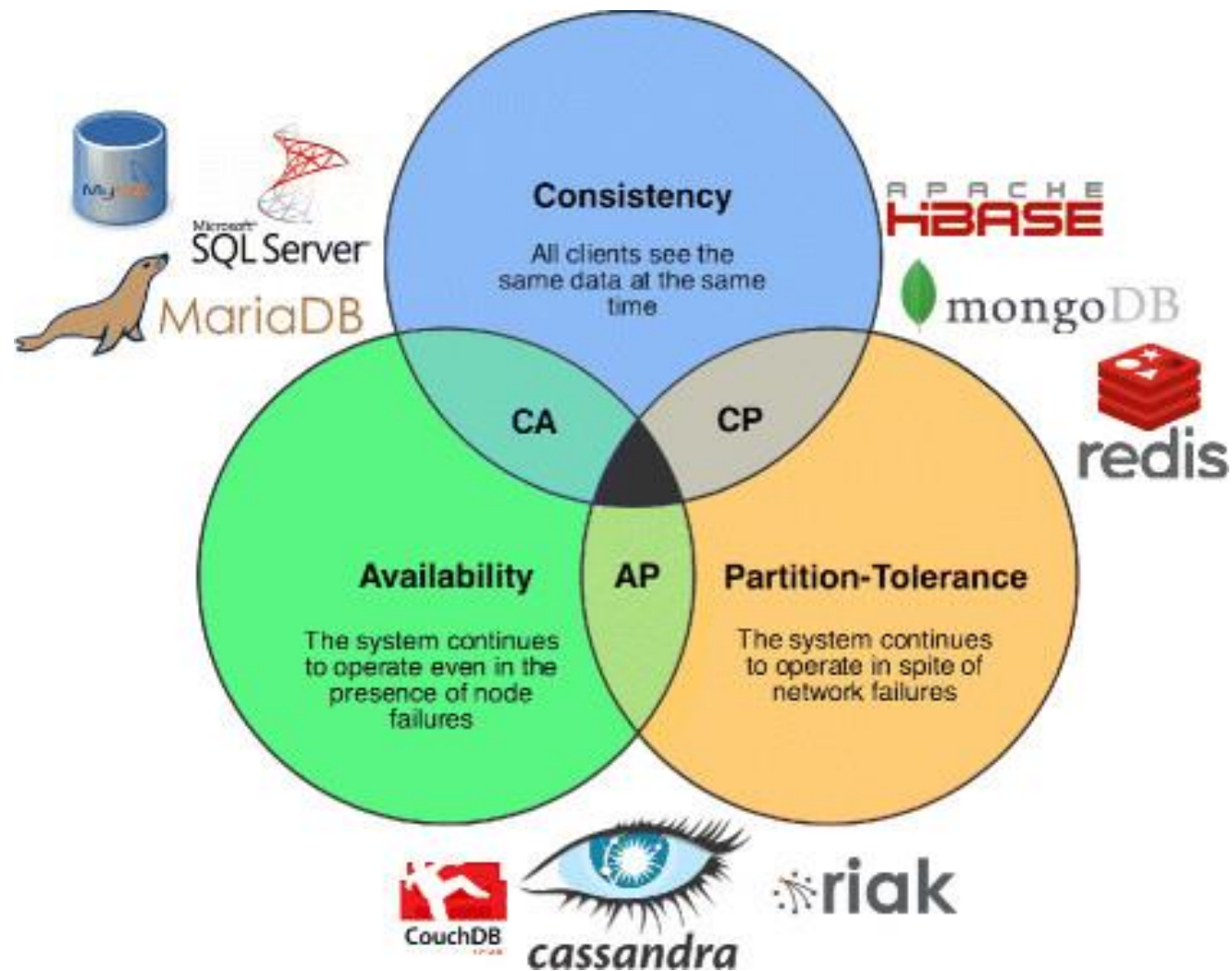
- **CAP-RDBMS**

- 전통적으로 단일 서버환경(소규모)을 기반으로 설계되었기 때문에 분산 처리보다는 트랜잭션의 일관성과 데이터 무결성을 더 중요시 여기는 경향
- 일관성과 가용성을 중시함 (CA)

- **CAP-NoSQL**

- 분할 허용성을 기본으로 하고 일관성과 가용성 중 하나를 선택함
- 예:
 - **Cassandra:** (PA) 우선, 일관성은 약함
 - **MongoDB:** 사용자가 (PC), (PA)를 조정 가능
 - **Redis:** (PC) 우선. Redis는 메모리 기반의 빠른 캐싱을 목적으로 하는 DB로 로그인 세션 등을 저장하는데 자주 사용됨. 이와 같이 세션 정보, 캐시 값 등 중요한 정보를 자주 다루기에 일관성이 중요함

CAP 이론



MySQL

- MySQL은 전 세계적으로 널리 사용되는 오픈 소스 데이터베이스 관리 시스템
- 빠르고 안정적이며 무료로 사용할 수 있어 널리 활용
- 주요 특징
 - **오픈 소스**: 오픈 소스로 누구나 무료로 사용할 수 있고 소스코드 수정이 가능
 - **SQL 언어 지원**: MySQL은 데이터베이스 관리를 위해 SQL(Structured Query Language)을 사용함. 이를 통해 데이터를 삽입, 조회, 수정, 삭제를 할 수 있음
 - **대규모 데이터 처리**: MySQL은 수백만개의 정보도 빠르게 처리할 수 있으며 여러 사용자가 동시에 접근해도 안정성 유지
 - **다양한 플랫폼 지원**: MySQL은 Windows, Linux, macOS 등 다양한 운영체제에서 사용할 수 있음

• MySQL 기본 명령어

- 셸에서 mysql에 접속하여 mysql 콘솔에서 명령어를 작업
- 모든 명령어는 세미콜론으로 끝남
- (참고) 관리자 계정으로 접속 가능한 DB는 일괄 제공함
 - mysql 계정 = mysql 비밀번호 = linux 계정 = DB_NAME

명령어	내용
\$ mysql -u ACCOUNT -p	shell에서 mysql에 ACCOUNT 계정으로 접속
show databases;	데이터베이스 목록 출력
create database DB_NAME;	데이터베이스 DB_NAME 생성
use DB_NAME;	사용할 데이터베이스로 DB_NAME을 선택

• MySQL 기본 명령어 (1/2)

내용	예시 코드
데이터베이스 생성	<code>CREATE DATABASE example_db;</code>
테이블 생성	<code>CREATE TABLE users (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100));</code>
데이터 삽입	<code>INSERT INTO users (name, email) VALUES ('Jonggyu', 'pjk5401@dau.ac.kr');</code>
데이터 조회	<code>SELECT * FROM users;</code>
데이터 조회(조건)	<code>SELECT * FROM users WHERE name = 'Jonggyu';</code>

• MySQL 기본 명령어 (2/2)

내용	예시 코드
데이터 업데이트	UPDATE users SET email = 'pjk5401@gmail.com' WHERE name = 'jonggyu';
데이터 삭제	DELETE FROM users WHERE name = 'jonggyu';
테이블 수정	ALTER TABLE users ADD COLUMN age INT;
테이블 삭제	DROP TABLE users;
데이터베이스 삭제	DROP DATABASE example_db;

예제 - MySQL

1. Users DB 생성

- **CREATE DATABASE user_db;**

2. Users Table 생성

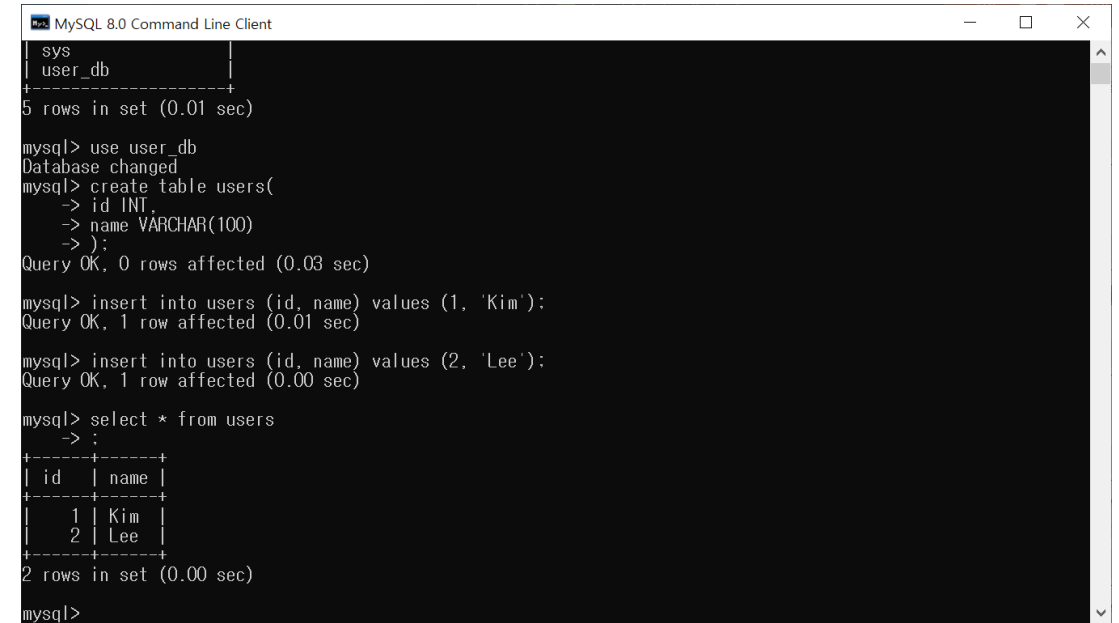
- **CREATE TABLE users (**
 id INT,
 name VARCHAR(100)
);

3. User Data 입력

- **INSERT INTO users (id, name) VALUES (1, 'Kim');**
- **INSERT INTO users (id, name) VALUES (2, 'Lee');**

4. User Data 조회

- **SELECT * FROM users;**



```
MySQL 8.0 Command Line Client
| sys |
| user_db |
+-----+
5 rows in set (0.01 sec)

mysql> use user_db
Database changed
mysql> create table users(
-> id INT,
-> name VARCHAR(100)
-> );
Query OK, 0 rows affected (0.03 sec)

mysql> insert into users (id, name) values (1, 'Kim');
Query OK, 1 row affected (0.01 sec)

mysql> insert into users (id, name) values (2, 'Lee');
Query OK, 1 row affected (0.00 sec)

mysql> select * from users
-> :
+-----+-----+
| id | name |
+-----+-----+
| 1 | Kim |
| 2 | Lee |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

MySQL<->Python

1. 가상환경에서 라이브러리 설치

- `$ pip install mysql-connector-python`

2. 기본예제

- 첨부파일 참고