전북대 소중해유(You)

MongoDB 소개:
기본 데이터 조작 및 활용

전북대학교 JEONBUK NATIONAL UNIVERSITY
전북대학교 SOFTWARE중심대학사업단
SW중심대학
과학기술정보통신부
IITP 정보통신기획평가원

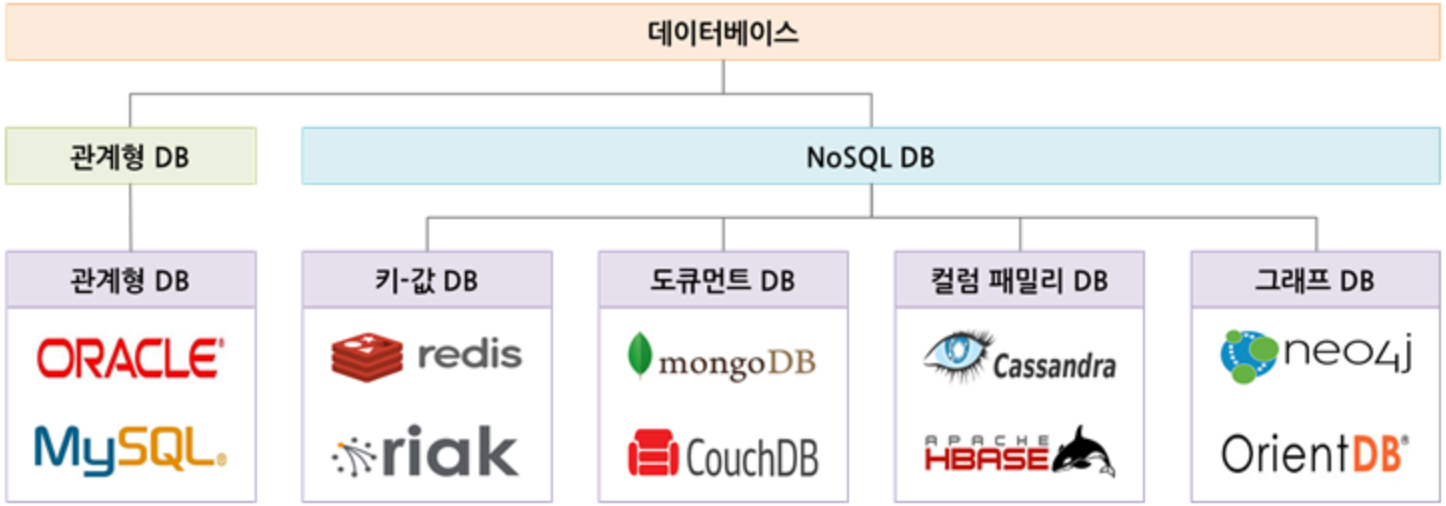
목차

MongoDB

NoSQL

NoSQL

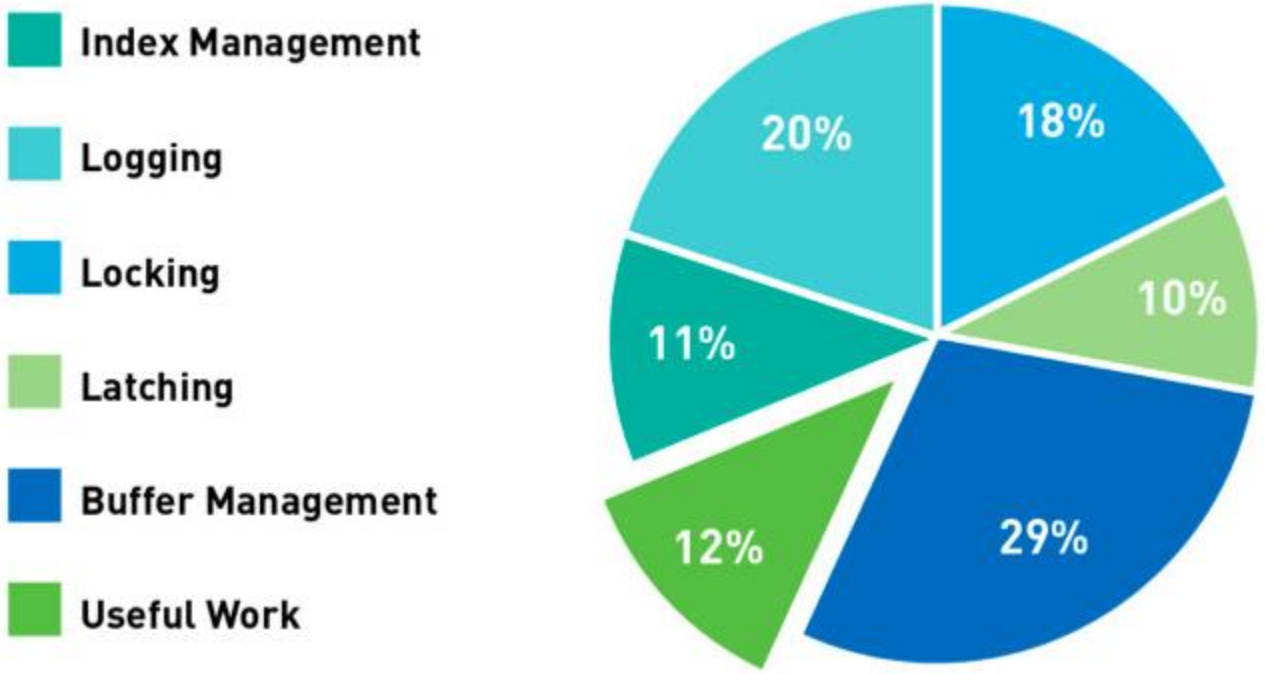
- Not Only SQL의 약자
- 기존 RDBMS 형태의 관계형 데이터베이스가 아닌 다른 형태의 데이터 저장 기술
- 특징: 비 관계형 DBMS로, 대규모의 데이터를 유연하게 처리할 수 있음



NoSQL을 사용하면 좋은 경우

- 정확한 데이터 구조를 할 수 없거나 변경/확장 될 수 있는 경우
- 읽기(read)처리를 자주하지만, 데이터를 자주 변경(update)하지 않는 경우 (즉, 한번의 변경으로 여러 문서를 변경할 일이 없는 경우)
- 데이터베이스를 수평으로 확장해야 하는 경우 (막대한 양의 데이터를 다뤄야 하는 경우)

관계형 데이터베이스
ACID



실질적으로 데이터를 넣고 빼고 하는 부분

원자성 (Atomicity)

일관성 (Consistency)

고립성(Isolation)

지속성 (Durability)

NoSQL을 Base 속성

- **Basically Available**
  - 기본적으로 이용 가능할 수 있어야 한다.
- **Soft state**
  - Eventually consistent 의 문제가 발견되면 일관성을 유지하기 위해 데이터 자동입력하는 성질
- **Eventually consistent**
  - 일관성을 유지되지 않는 상태가 될때 일관성 유지를 위해 이벤트를 잡아주는 성질

Document DB

- NoSQL의 일종
- Key-Value 형식으로 문서 저장
- Join을 사용하지 않음
- 스키마가 자유로움 (free)
- 대량 데이터 및 분산처리에 특화
- MongoDB, CouchDB 등

```
{
"id" :20221234,
"name" : "홍길동",
"height" : "183",
"age" : "22"
}
```

MongoDB

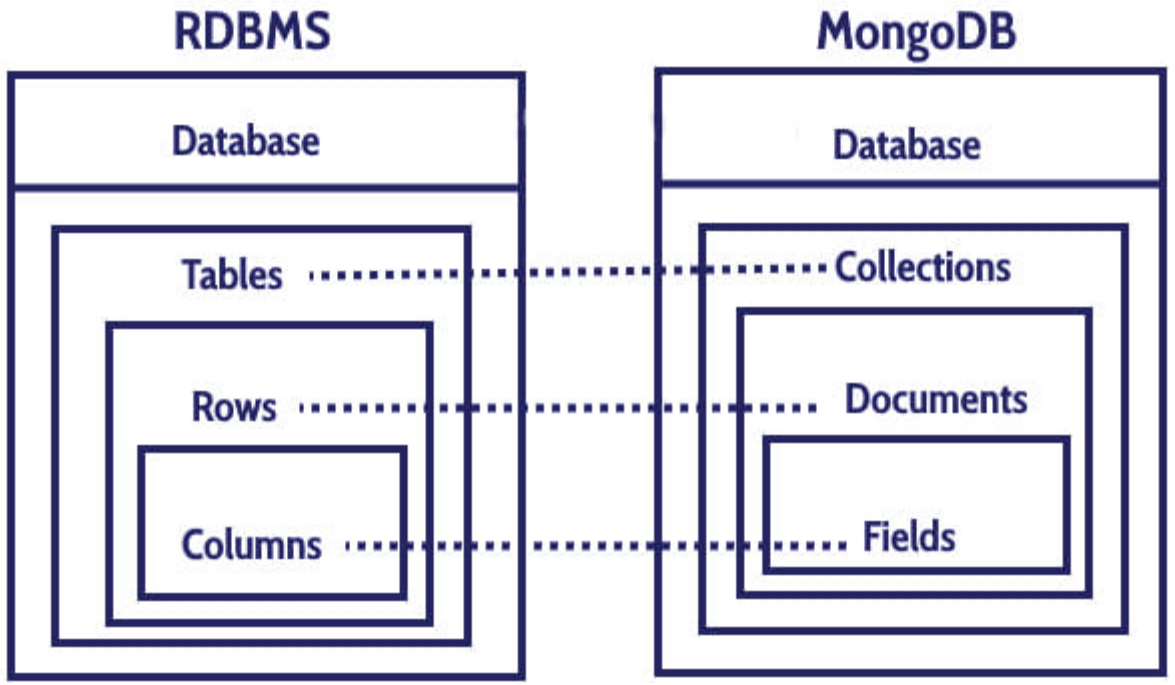
- DocumentDB(문서지향 데이터베이스)의 일종으로, 스키마가 따로 존재하지 않음
- 도큐먼트는 BSON이라는 데이터 포맷을 사용
- BSON은 Binary JSON을 의미하는 것으로 JSON을 바이너리 형식으로 저장
- JSON에서는 지원하지 않는 자료형인 Date와 BinData타입을 지원
- scale-out이 가능하며, Join과 Transaction이 없음.

MongoDB

- 가장 인기 있는 문서 지향 데이터베이스 중 하나
- 문서 저장소는 BSON 파일에 저장되므로 모든 JS 지원 가능
- 더 큰 스토리지 용량과 속도 요구 충족 가능
- 고정 된 구조를 정의하는 전제 조건을 제거 (스키마 없는 구현)
- 문서 데이터 모델은 응용 프로그램 코드의 개체에 자연스럽게 매핑
- MongoDB 4.0 부터 다중 문서 트랜잭션에 대한 지원 추가
- 지원되는 언어
- Node.js, C, C++, C#, Go, Java, Perl, PHP, Python, Ruby, Rust, Scala, and Swift, …

MongoDB

- 데이터 계층 구조



MongoDB의 장단점

▼ 몽고디비 장단점

장점	단점
• 스키마를 지정하지 않아도 되므로 데이터 저장의 유연성이 있음. 즉 모델에 필드를 추가할 때 DB에서는 추가로 할 일이 없음 • 단일 문서 검색 시 여러 테이블을 조인하는 것보다 빠른 경우가 많음[4] • 클러스터를 지원해주기 때문에 스케일아웃이 쉬움 • 다른 NoSQL 대비 인덱스 지원이 잘되어 있음	• 메모리를 많이 사용함 • 디스크 저장 공간을 RDB에 비해 많이 씀 • 복잡한 조인은 사용하기 힘듬 • 트랜잭션 지원이 RDB에 비해 약함

MongoDB의 장단점

▼ 몽고디비 장단점

장점	단점
• 스키마를 지정하지 않아도 되므로 데이터 저장의 유연성이 있음. 즉 모델에 필드를 추가할 때 DB에서는 추가로 할 일이 없음 • 단일 문서 검색 시 여러 테이블을 조인하는 것보다 빠른 경우가 많음[4] • 클러스터를 지원해주기 때문에 스케일아웃이 쉬움 • 다른 NoSQL 대비 인덱스 지원이 잘되어 있음	• 메모리를 많이 사용함 • 디스크 저장 공간을 RDB에 비해 많이 씀 • 복잡한 조인은 사용하기 힘듬 • 트랜잭션 지원이 RDB에 비해 약함

MongoDB의 연산자

▼ 몽고디비의 연산자

연산자	설명
$set	도큐먼트의 속성값을 변경할 때 사용합니다.
$unset	도큐먼트의 속성을 삭제할 때 사용합니다.
$rename	도큐먼트의 속성의 이름을 변경할 때 사용합니다.
$inc	필드의 값을 증가시킬 때 사용합니다.
$mul	필드의 값에 곱하기를 할 때 사용합니다.
$min	지정한 값과 현잿값 중 작은 값을 선택합니다.
$max	지정한 값과 현잿값 중 큰값을 선택합니다.
$currentDate	현재 날짜와 시간을 필드에 업데이트합니다.
$addToSet	배열 필드가 아직 없는 경우 해당 필드에 값을 추가합니다.
$pop	배열 필드에서 첫 번째 혹은 마지막 값을 삭제합니다.
$pull	배열 필드에서 모든 값을 삭제합니다.
$push	배열 필드의 끝에 값을 추가합니다.
$each	여러 개의 값을 추가해 배열 필드를 수정합니다.

MongoDB의 사용 이유

- **신뢰성 (Reliability)**
  - 고가용성 지원: MongoDB는 primary와 secondary로 구성된 ReplicaSet 구조를 통해 고가용성을 지원하여 서버 장애에도 서비스가 계속 동작할 수 있음.
- **확장성 (Scalability)**
  - 수평확장 가능: MongoDB는 데이터를 샤딩하여 데이터와 트래픽이 증가해도 수평확장(scale-out)이 가능함.
- **유연성 (Flexibility)**
  - 다양한 데이터 형태 지원: 서비스 요구사항에 따라 다양한 종류의 데이터를 추가할 수 있으며, 스키마 변경 과정 없이 필요한 데이터를 손쉽게 저장하고 읽을 수 있음.

MongoDB의 사용 이유

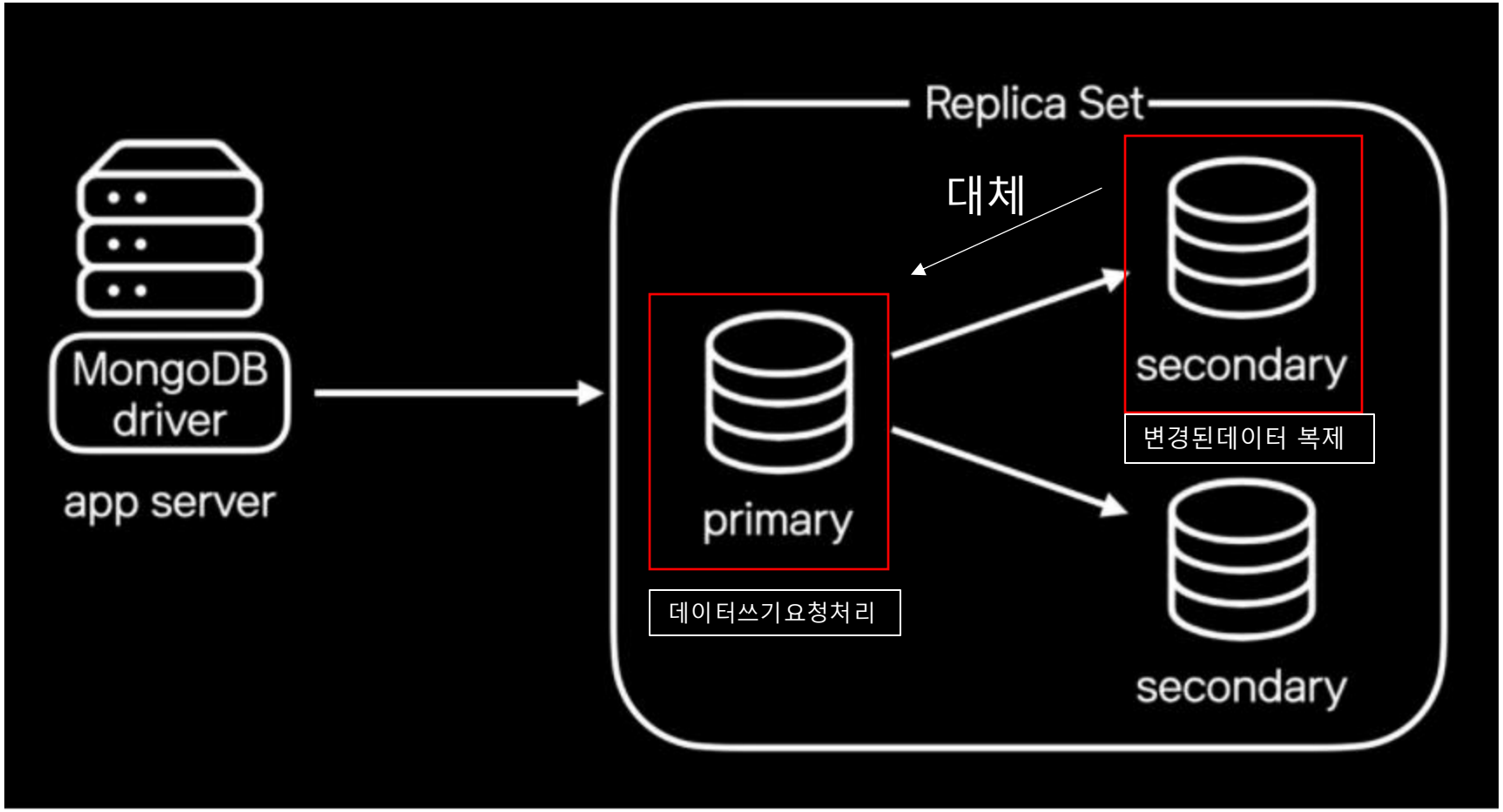
- **인덱스 지원 (Index Support)**
  - 다양한 인덱스 지원: MongoDB는 다양한 인덱스를 지원하여 여러 조건으로 빠른 데이터 검색이 가능함. 단, 컬렉션당 최대 64개의 인덱스만 생성할 수 있으며, 너무 많은 인덱스를 추가하면 부작용(side effects)이 발생할 수 있음.
- **성능 (Performance)**
  - 고성능 데이터 처리: MongoDB는 메모리 내에서 데이터를 처리하는 방식과 인덱싱, 복합 인덱스를 통해 빠른 읽기 및 쓰기 성능을 제공함. 특히, 대량의 데이터를 빠르게 검색하고 처리하는 데 강점을 가짐.
- **보안 (Security)**
  - 강력한 보안 기능 제공: MongoDB는 SSL/TLS를 통한 데이터 암호화, 역할 기반 접근 제어(Role-Based Access Control, RBAC)와 같은 보안 기능을 제공하여 데이터의 무결성을 보호하고, 불법적인 접근을 방지할 수 있음.

MongoDB의 사용 이유

- **인덱스 지원 (Index Support)**
  - 다양한 인덱스 지원: MongoDB는 다양한 인덱스를 지원하여 여러 조건으로 빠른 데이터 검색이 가능함. 단, 컬렉션당 최대 64개의 인덱스만 생성할 수 있으며, 너무 많은 인덱스를 추가하면 부작용(side effects)이 발생할 수 있음.
- **성능 (Performance)**
  - 고성능 데이터 처리: MongoDB는 메모리 내에서 데이터를 처리하는 방식과 인덱싱, 복합 인덱스를 통해 빠른 읽기 및 쓰기 성능을 제공함. 특히, 대량의 데이터를 빠르게 검색하고 처리하는 데 강점을 가짐.
- **보안 (Security)**
  - 강력한 보안 기능 제공: MongoDB는 SSL/TLS를 통한 데이터 암호화, 역할 기반 접근 제어(Role-Based Access Control, RBAC)와 같은 보안 기능을 제공하여 데이터의 무결성을 보호하고, 불법적인 접근을 방지할 수 있음.

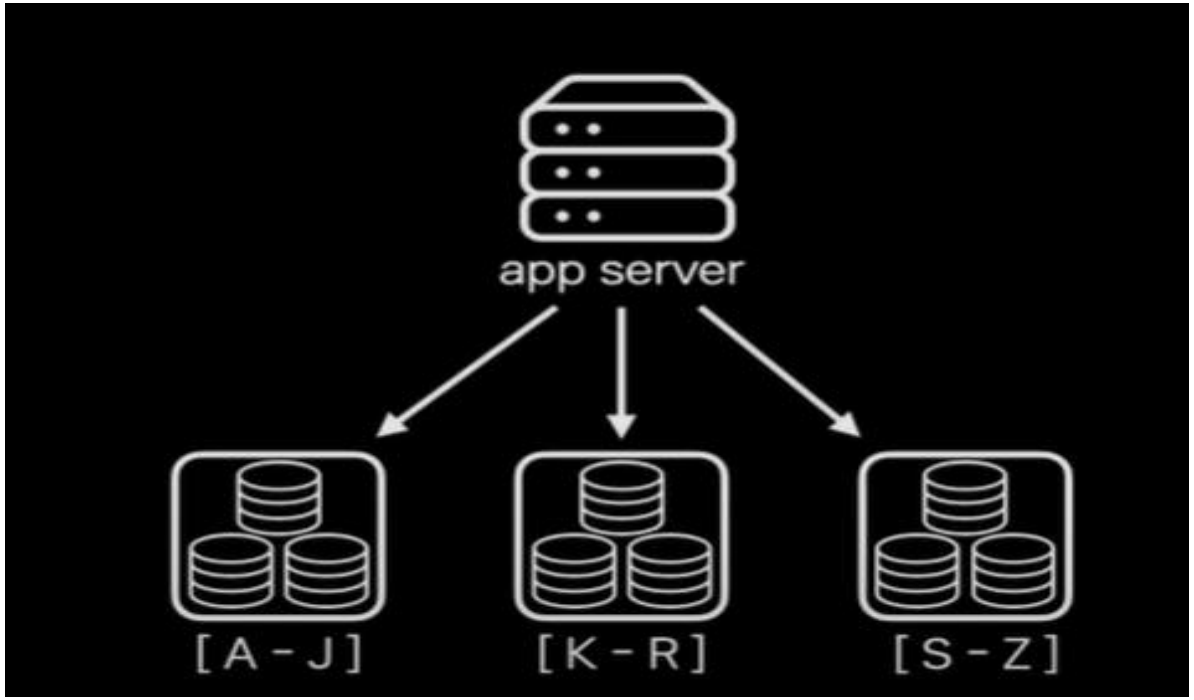
MongoDB의 사용 이유

- 신뢰성



MongoDB의 사용 이유

- **확장성**
  - 몽고DB는 데이터를 샤딩하여 분산시켜 줄 수 있음
  - 이러한 샤딩 과정은 서비스 중단없이 온라인으로 진행



MongoDB의 사용 이유

- **확장성**
  - 다양한 형태의 데이터를 한번에 담을 수 있음
  - 어플리케이션에서 다루는 오브젝트와 1:1 MATCH
  - 개발자들은 데이터를 쉽게 이해하고 빠르게 개발 가능

customers

customer_id	name	address	phone_number
1	MUZI	Busan	010-1111-1111
2	PRODO	Seoul	
3	RYAN	Seoul	010-2222-2222

ios / Android ?

customers

customer_id	name	address	phone_number	os
1	MUZI	Busan	010-1111-1111	iOS
2	PRODO	Seoul		
3	RYAN	Seoul	010-2222-2222	Android

MongoDB의 사용 이유

- **확장성**
  - 다양한 형태의 데이터를 한번에 담을 수 있음
  - 어플리케이션에서 다루는 오브젝트와 1:1 MATCH
  - 개발자들은 데이터를 쉽게 이해하고 빠르게 개발 가능

customers

customer_id	name	address	phone_number
1	MUZI	Busan	010-1111-1111
2	PRODO	Seoul	
3	RYAN	Seoul	010-2222-2222

ios / Android ?

customers

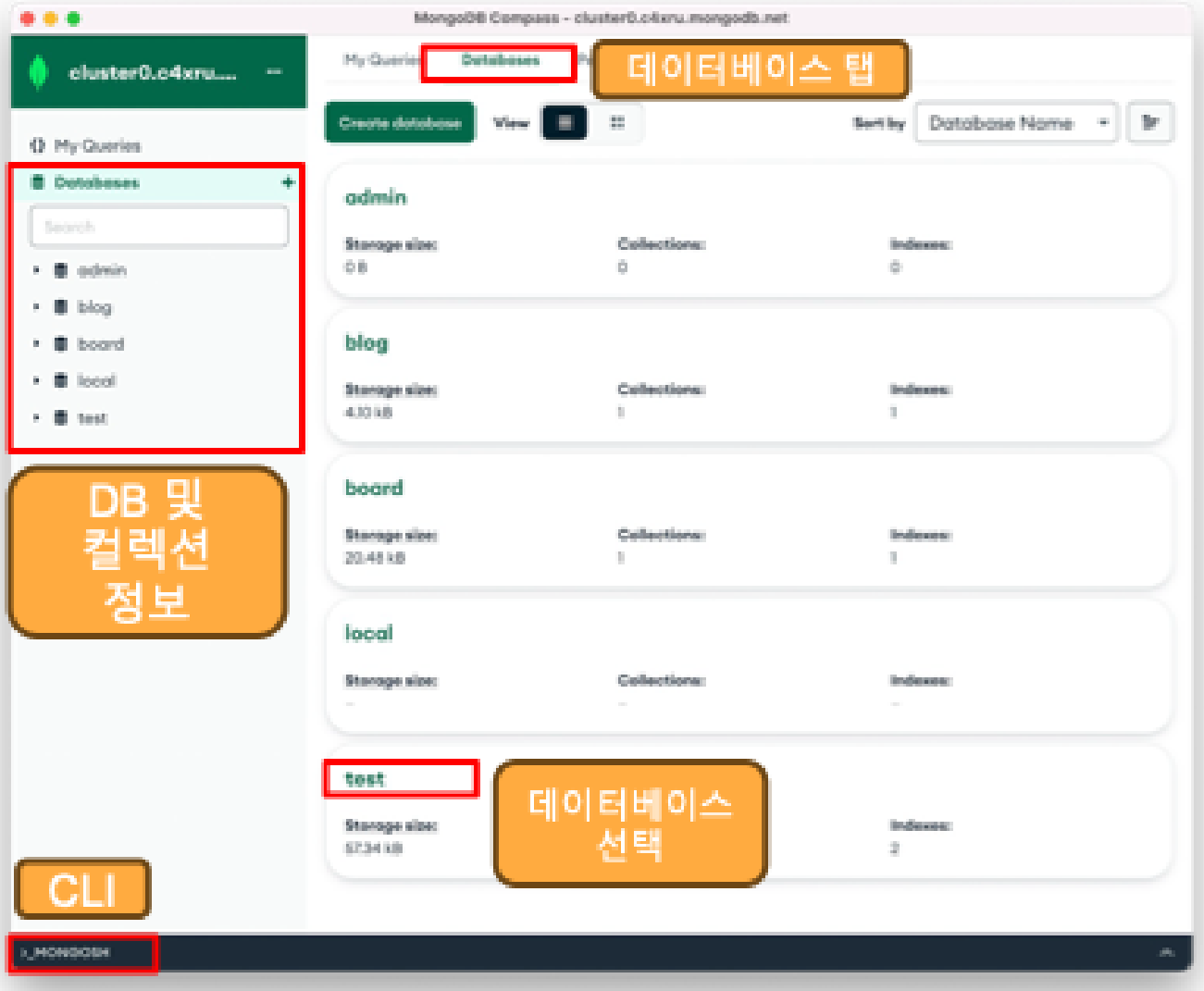
customer_id	name	address	phone_number	os
1	MUZI	Busan	010-1111-1111	iOS
2	PRODO	Seoul		
3	RYAN	Seoul	010-2222-2222	Android

MongoDB의 사용 이유

- **개발 효율 향상**
  - 모바일 게임은 단기간에 개발해야 함. 그러나 게임은 특성상 자주 변경이 발생함. 스키마리스인 MongoDB는 데이터 구조 변경에 유연하게 대응할 수 있음.
- **신기능, 변경 릴리스 대응**
  - 모바일 게임은 주에 3~4회 릴리스하는 경우가 많음. 그러나 컬럼 추가나 인덱스 추가를 온라인 상태에서 할 수 없으면 점검이 필요하게 됨. MongoDB를 사용하면 점검이 하지 않고 추가가 가능함.
- **유연한 쿼리, 인덱스**
  - 계층 구조화한 문서 내부에도 인덱스를 확장할 수 있음. KVS로도 이용할 수 있어서 유연한 쿼리 검색이 가능해 매력적임.
- **초기 비용이 작고, 확장성을 보장**
  - ReplicaSets(비동기 레플리케이션)과 Sharding(자동 데이터 분산)을 제공하고 있음

MongoDB 연결 실습

MongoDB Compass



MongoDB 연결 실습

MongoDB 예제

```javascript
const { MongoClient } = require('mongodb');

const uri = "mongodb://localhost:27017";
const client = new MongoClient(uri);

async function run() {
    try {
        await client.connect();
        console.log("Connected to MongoDB");

        const database = client.db('mydatabase');
        const collection = database.collection('mycollection');

        // CRUD 작업 수행
        await createDocument(collection);
        await readDocument(collection);
        await updateDocument(collection);
        await deleteDocument(collection);

    } finally {
        await client.close();
    }
}

async function createDocument(collection) {
    const doc = { name: "John Doe", age: 30, profession: "Engineer" };
    const result = await collection.insertOne(doc);
    console.log(`New document inserted with _id: ${result.insertedId}`);

    const docs = [
        { name: "Alice", age: 25, profession: "Designer" },
        { name: "Bob", age: 28, profession: "Manager" }
    ];
    const insertManyResult = await collection.insertMany(docs);
    console.log(`${insertManyResult.insertedCount} documents were inserte
}
```

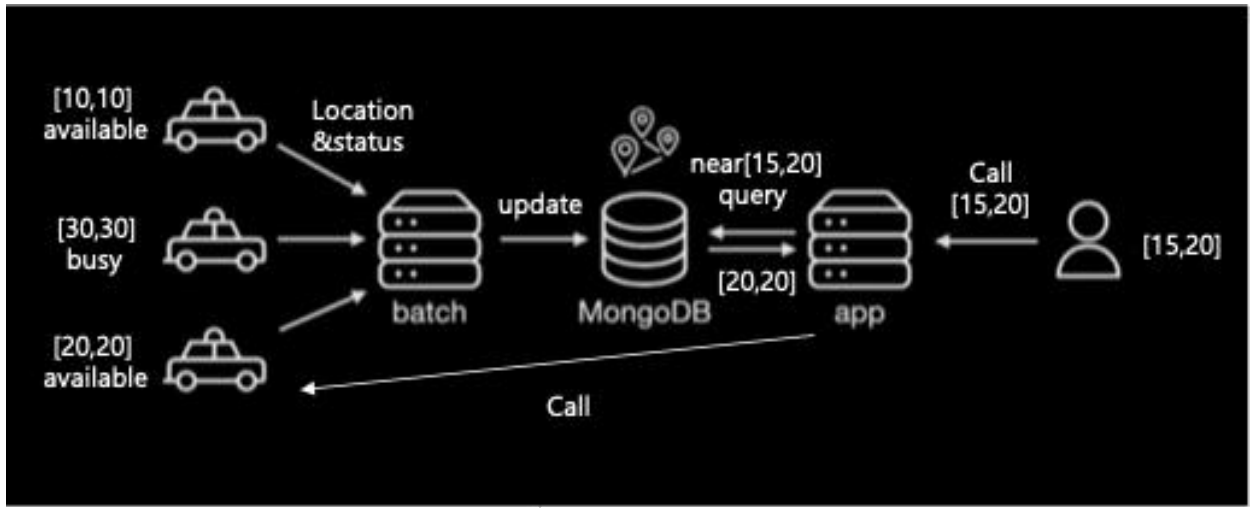
```javascript

async function readDocument(collection) {
    const query = { name: "John Doe" };
    const foundDoc = await collection.findOne(query);
    console.log("Found document:", foundDoc);

    const ageQuery = { age: { $gte: 25 } };
    const foundDocs = await collection.find(ageQuery).toArray();
    console.log("Found documents:", foundDocs);
}

async function updateDocument(collection) {
    const query = { name: "John Doe" };
    const updateDoc = { $set: { age: 31 } };
    const updateResult = await collection.updateOne(query, updateDoc);
    console.log(`${updateResult.matchedCount} document(s) matched the query, ${updateResult.modifiedCount} was/were updated`);

    const ageQuery = { age: { $lt: 30 } };
    const updateManyResult = await collection.updateMany(ageQuery, { $set: { profession: "Updated Profession" } });
    console.log(`${updateManyResult.matchedCount} document(s) matched the query, ${updateManyResult.modifiedCount} was/were updated`);
}

async function deleteDocument(collection) {
    const query = { name: "John Doe" };
    const deleteResult = await collection.deleteOne(query);
    console.log(`${deleteResult.deletedCount} document(s) was/were deleted`);

    const ageQuery = { age: { $gte: 30 } };
    const deleteManyResult = await collection.deleteMany(ageQuery);
    console.log(`${deleteManyResult.deletedCount} document(s) was/were deleted`);
}

run().catch(console.dir);

```

MongoDB 실습은 다음 시간에...

공간 인덱스

- **카카오택시 예시**

1. 전국의 카카오택시들은 자신의 위치정보와 승차가능 여부 등 여러가지 정보를 주기적으로 배치서버로 발송.
2. 배치서버는 택시정보를 몽고디비에 지속적으로 업데이트.
3. 택시의 위치와 상태정보는 실시간으로 몽고디비에 유지.
4. 사용자가 택시를호출하면 사용자의 위치정보가 함께 전달.
5. 요청을 받은 앱서버는 사용자의 위치를기준으로 일정 범위에 탑승가능한 택시를 몽고디비에 조회.
6. 몽고디비는 공간인덱스를 활용하여 요청받은 쿼리를 빠르게 처리하고 호출가능한 택시정보를 리턴



감사합니다.

- 본 온라인 콘텐츠는 2024년도 과학기술 정보통신부 및 정보통신기획평가원의 'SW중심대학사업' 지원을 받아 제작되었습니다.
- 본 결과물의 내용을 전재할 수 없으며, 인용(재사용)할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원이 지원한 'SW중심대학'의 결과물이라는 출처를 밝혀야 합니다.

전북대학교 JEONBUK NATIONAL UNIVERSITY
전북대학교 SOFTWARE중심대학사업단
SW중심대학
과학기술정보통신부
IITP 정보통신기획평가원