NoSQL 기초 개념과 MongoDB 시작하기

소프트웨어공학과 / 201332001 강인성 / hogu9401@gmail.com

1. NoSQL 개념과 특성

A. Not Only SQL

SQL를 굳이 이용하지 않고 데이터베이스를 관리하는 시스템으로 지칭을 한다. 데이터베이스는 이미 관계형 데이터베이스가 대부분을 차지를 하고 있지만, NoSQL에서는 관계형 데이터베이스 뿐만 아니라 다양한 유형의 데이터베이스를 사용하기 위하여 사용을 하는 의미로 받아들일 수 있다.

NoSQL에 대하여 알려진 정의는 정확하게 설명할 수 있는 방법은 공식적으로 알려지지 않았지만, NoSQL 데이터베이스의 짧고 강한 특성으로 설명을 할 수 있다면, **대부분 오픈 소스 제공과 SQL 문장을 사용하지 않는 Schema-less 데이터베이스로** 요약이 가능하다.

B. NoSQL의 특징

(1) 일관성에 대한 느슨함

관계형 데이터베이스(Relational Database)에서 주요 특성인 일관성은 다중 클라이언트에서 같은 시간에 조회하는 데이터에 대하여 동일한 데이터임을 보증을 하는 것이다. 그렇지만 NoSQL에서는 관계형 데이터베이스에 비해 데이터의 일관성이 느슨하게 처리되어 동일한 데이터가 나오지 않을 수도 있다. 일관성이 느슨한 이유는 크게 2가지로 나뉘어서 본다면 다음과 같이 나뉘어 볼 수 있다.

- 다수가 동시에 읽고 쓰는 상황에서 성능을 향상을 하기 위한 목적이 있다.

- 분산 환경에서 노드들이 잘 작동하고 있어도 시스템의 일부가 에러가 발생하면 데이터베이스를 사용할 수 없는 문제를 해결하기 위한 목적이 있다.

하지만 NoSQL에서도 일관성을 제공을 하지 않는다는 의미로 생각을 하면 안 된다. 느슨하게 처리를 한다고 생각을 해도 **데이터의 변경에 대한 시간의 흐름으로 여러 노드에 전파**하는 의미로 받아들여야 한다. 이러한 방안이 바로 **최종 일관성**을 지원하는 것으로 이해를 할 수 있다. 분산 노드 간의 데이터를 처리하는 2가지 방안으로는 **동기식 방법, 비동기식 방법**으로 나뉠 수 있다.

- 동기식(Synchronized) 방법 : 데이터의 저장 결과를 클라이언트로 응답하기 전에 모든 노드에 데이터를 저장을 하는 방안으로 느린 응답시간이 있는 단점도 있지만 데이터의 정합성을 보장할 수 있는 수단으로 볼 수 있다.

- 비동기식(Asynchronized) 방법 : 메모리나 임시 파일을 기록을 완료하고 클라이언트에 응답을 완료한 다음 특정 이벤트나 프로세스를 이용해 노드로 데이터를 동기화하는 방안으로 볼 수 있다. 빠른 응답시간을 보이는 장점도 존재하지만 쓰기 노드에 장애가 발생하면 데이터가 손실될 수 있다.

(2) 분산 저장 비용에 대한 효율성 제공

데이터와 트래픽의 증가하면서 기존 장비로는 원활한 데이터의 처리가 어려워진 추세로서 이를 해결하는 방안으로서는 처리하는 장비의 수를 늘리는 수평적 확장(Scale-Out)을 고려해야 하지만 이는 관계형 데이터베이스의 클러스터에서는 효율적으로 접근을 할 수는 없다. (물론 관계형 데이터베이스에서도 분산 저장에 대해서는 샤딩 기법(Sharding)을 통해서 진행이 가능하지만, 어플리케이션 레벨에서 모든 샤딩을 제어를 하는 면에서는 복잡하다.) 하지만 NoSQL에서는 집합 지향 모델(Aggregate-Oriented)을 사용해서 연관된 데이터들이 함께 분산하는 점에서는 복잡한 제어가 덜 필요하게 된다.

(3) 데이터 불일치 문제를 최소화

관계형 튜플(레코드) 내부 값은 단순하고 중첩된 레코드나 리스트 등의 다른 구조를 포함 할 수 없다. ORM(Objective-Relational Mapping) 프레임워크(JDBC, MyBatis, JPA etc.)를 통하여 해결을 할 수 있어도 관계형 모델과 메모리 내의 데이터 구조 간에는 데이터 불일치가 존재할 수도 있다. 그렇지만 NoSQL에서는 메모리 내의 데이터 구조가 어떻든 상관 없이 하나의 Aggregation(집합)으로 취급하여 저장을 하기 때문에 데이터 일치를 향상시키는데 도움을 준다.

(4) Schema-less(Schema 없이 동작하는 데이터베이스)

데이터베이스 스키마(Database Schema)는 데이터베이스에서 자료의 구조와 표현 방법, 자료 간의 관계를 형식 언어로 정의한 구조로 인지를 할 수 있는데 관계형 데이터베이스에서는 데이터베이스 관리 시스템(Database Management System)을 통해서 일일이 Table, View, Trigger 등을 Schema에서 미리 만들고 작업을 하는 것을 관례로 치지만, NoSQL에서는 Schema를 따로 관리하지 않고 이를 통해서 동작을 하지 않는다는 점을 인지를 할 수 있다. 즉 이 뜻은 데이터 구조를 미리 정의할 필요가 없고, 어느 데이터베이스 내부에 비형식적인 데이터를 저장을 할 때에 시간이 지나도 언제든지 바뀔 수 있다. 하지만 이러한 특징 때문에 데이터 타입에 따른 암묵적인 스키마는 여전히 존재할 수 있어서 단일한 값들에 대한 데이터 타입에서 불일치는 여전히 발생은 한다.

C. NoSQL 데이터 모델 종류

NoSQL 데이터베이스의 모델은 크게 4가지로 나뉠 수 있는데 대표적으로 쓰는 데이터베이스 모델은 Aggregate-Oriented Model(집합 지향 모델)을 기반으로 제공을 한다.

Aggregate-Oriented Model(表) : Key-Value Model / Document Model / Column-Family Model

Graph Model

이들을 토대로 데이터베이스 모델이 어떻게 나뉘는지에 대하여 인지를 하는 점도 중요하다. 종류 별로 어떻게 구조가 나뉘는지 인지를 해보도록 하자.

(1) Aggregate-Oriented Model(집합 지향 모델)

여기서 생각할 수 있는 집합(Aggregate)은 연산의 한 단위로 취급되는 연관된 객체의 집합이다. 관계형 데이터베이스는 오로지 객체와 관계를 다루어서 데이터베이스 스키마에 저장이 되어 있는 것과는 달리 집합 지향 데이터베이스는 집합 자료구조로 이루어져 있다. 관계형 데이터베이스에서 쓰였던 관계형 모델(Relational Model)에 현존하는 Entity에 대한 Transaction의 주 특성인 ACID(Atomicity, Consistency, Isolation, Durability)을 지원하지 않아도 하나의 집합에 대한 연산에서는 다행이 Transaction을 지원을 한다. 이는 여러 서버에서 구동되는 클러스터들을 통한 시스템에서 이용하기 적합하다. 관계형 데이터베이스와는 달리 오로지 **연관된 데이터들이 함께 움직이는 면에서** 수평적 확장(Scale-Out)이 용이하게 된다. 또한 메모리 내의 자료구조와 집합 간의 데이터의 일치성도 보장을 하여 객체 관계 매핑 프레임워크가 딱히 필요하지 않다. 데이터의 검색은 Key나 ID를 통해 검색을 하는데 용이하지만, 타 집합에 대한 Join 연산이 불가능하다. 이를 보완하기 위해 Map Reduce 기능을 통해 Join과 유사한 연산을 가능하도록 설계를 한 NoSQL 데이터 모델도 구비되어 있다. 이를 기반으로 구성을 하는 모델은 위에서도 언급을 했지만 Key-Value Model, Document Model, Column-Family Model 3가지로 볼 수 있다. 이 3가지에 대해서 간략하게 요약을 해보도록 하겠다.

(1)-1 Key-Value Model

이는 NoSQL 데이터 모델 중에서 가장 단순한 형태로 수평적 확장이 또한 용이하다. 그리고 NoSQL 데이터 모델 중에서 제일 단순한 API를 제공을 하여 질의가 put, get, delete 3가지로 나뉘기 때문에 NoSQL를 공부하는데 있어서 크게 어려운 점을 못 느낀다. 그렇지만 값의 내용을 사용한 질의를(각 데이터들에 대한 조건을 통한 탐색을 사례로 생각할 수 있다.) 이용할 수 없는 단점이 존재한다. Key-Value 모델에서 쓰인 Aggregate는 Key를 통하여 값에 접근을 하기 위한 용도로 쓰고, 값은 데이터 형태에 대하여 큰 상관 없이(사진, 비디오 etc.) 사용이 가능하고, 질의의 속도도 굉장히 빠른 편이다. 그래서 고속 읽기와 쓰기에 최적화가 되어 있기 때문에 간략한 정보들을 쓰고 가져오는 경우에 활용을 하면 좋은 Model로서 대표적으로 쓰는 사례는 **사용자 프로필 정보, 웹 서버 클러스터를 위한 세션 정보, 장바구니 정보, URL 단축 정보 저장** 등에 사용을 한다. 이 모델을 사용하는 NoSQL는 Redis, LevelDB, Amazon Dynamo DB 등이 있다.

(1)-2 Document Model

Key-Value Model에서 값의 내용을 사용한 질의에 대한 단점과 각 Key를 통해 저장된 데이터들에 대하여 계층적인 구조를 형성하는 점을 해결하기 위한 Model로 생각할 수 있다. 여기서 쓰이는 Aggregate는 Key와 Document의 형태로 저장이 된다. 방금 전에 언급했던 Value와는 달리 계층적인 형태로 구성된 Document로 저장이 되어 객체지향에서의 객체와 유사하여 하나의 단위로 취급되어 저장을 하기 때문에 여러 개의 테이블에 나뉘어 저장을 할 필요가 없어지게 된다. 또한 Document 내의 item을 사용한 질의를 이용할 수 있게 되지만, XQuery나 다른 Document 질의 언어가 필요하다. 마찬가지로 Relational Mapping(객체 관계 매핑)이 필요하지 않고 객체를 Document의 형태로 바로 저장을 할 수 있다. 그리고 Key-Value Model와 마찬가지로 Key를 통한 검색은 유사하지만, 본래 관계형 데이터베이스에서 썼던 SQL 쿼리 문법과는 다르기 때문에 이에 대한 적응이 필요로 하다. 대부분의 문서 모델 NoSQL은 인덱스를 B-Tree를 사용해서 인덱스를 생성을 하게 되는데 데이터베이스에서 읽기와

쓰기에 대한 비율을 생각을 해서 써야 좋은 성능을 보일 수가 있다. 대표적으로 쓰는 사례는 **중앙 집중식 로그 저장, 타임라인 저장, 통계 정보 저장** 등에 사용을 한다. Document Model을 사용하는 NoSQL 데이터베이스는 MongoDB, CouchDB, MarkLogic 등이 존재한다.

(1)-3 Column-Family Model

위의 두 모델과는 다소 차이점이 존재한다. 단순하게 Key-Value, Key-Document를 이용해서 필드를 결정하는 방안과는 달리 여기서는 Key 필드 내부에는 Row Key가 존재하고, Value의 값은 Column-Family, Column-Name를 가지게 된다. 연관된 데이터들은 같은 Column-Family 안에 속하면서 각자의 Column-Name을 가지게 된다. 이는 구글의 빅 테이블의 대표적인 사례로 Column-Family Model은 이를 영향을 받아서 저장된 데이터는 하나의 커다란 테이블로 표현이 되어 질의는 Row, Column-Family, Column-Name을 통해 수행할 수 있어서 각 데이터에 대한 Time Stamp가 존재해 수정된 History 또한 인지를 할 수 있다. 데이터 형태는 제한이 없지만 BLOB 단위의 쿼리가 불가능하고, Row, Column의 초기 디자인도 신경을 써야 하고, 새로운 필드를 만드는데 비용이 커서 스키마를 변경하는 과정도 생각보다 어렵다. Column-Family Model SQL은 쓰기에 더욱 특화 되어 있어서 데이터를 메모리에 먼저 저장을 하고 응답을 하여 빠른 응답 속도를 제공을 하기 때문에 연산이 많은 서비스나 빠른 시간 안에 대량의 데이터를 입력하고 조회하는 서비스를 구현할 때 가장 좋은 성능을 보이게 된다. 대표적으로 **채팅 내용 저장, 실시간 분석을 위한 데이터 저장소 서비스** 등을 구현하는 방향이 좋다. Column-Family Model을 사용하는 NoSQL 데이터베이스는 Google Big Table, Cassandra, HBase 등이 존재한다.

(2) Graph Model

정점(Vertex)와 간선(Edge)들로 각 정점들이 간선을 통해 연결을 하여 그들을 구성을 하는 개념이 그래프(Graph)란 사실은 이미 알고 있을 것이다. 집합 지향 모델(Aggregate-Oriented Model)보다는 관계형 모델(Relational Model)에 더욱 가깝게 느낄 수 있다. 실제 세계의 데이터를 관계와 함께 표현하기 위한 디자인의 모델로, 데이터는 연속적인 노드, 관계, 특성의 형태로 저장이 되어 이들을 Graph 형태로 저장을 하게 되어 질의는 그래프 순회를 통해 이루어진다. 짧게 요약하면 **개체와의 관계를 그래프 형태로 표현하는 것**으로 볼 수 있어서 우리가 흔히 볼 수 있는 페이스북, 트위터, 카카오톡 등에서 **내 친구에서 친구를 찾는 질의**를 생각해 볼 수 있는데 연관된 데이터를 추천해주는 추천 엔진이나 패턴 인식 등의 데이터베이스로도 적합하고, 집합 지향 모델과는 다르게 개체의 Transaction의 속성인 ACID를 지원한다. 허나 Graph Model은 클러스터링에 부적합할 뿐더러, 질의 언어도 생각보다 습득하기 힘들기 때문에 NoSQL를 공부하는 입장에서는 많은 비용이 소요가 될 수도 있다.

D. Relational Database VS NoSQL

관계형 데이터베이스와 NoSQL의 선택의 여부에 대하여 고민을 하는 개발자들이 비일비재할 것이다. 실제로 어떤 Model로 구성이 되는 데이터베이스를 사용할지에 대한 정답은 확실하게 밝힐 수는 없다. 그렇지만 이미 많은 개발자들에게 Relational Database는 사용 방법도 이미 익숙하면서 사용하는 방법이 정해져 있어서 손쉽게 접근이 가능하다. 그렇지만 NoSQL를 주로 사용하는 사례들을 살펴본다면 소셜 네트워크 서비스에서 타임라인, 게시글, 메신저 메시지 등을 올리거나 게임 로그, 구매 내역 등 엄청난 양이 생성되는 데이터들이 존재 하지만 수정되는 일이 거의 없기 때문에 ACID Transaction을 지원을 해야 할까 란 의문이 남겨져 있고, 장비 성능에 대한 영향도 생각을 해서 데이터의 양이 누적되어도 여러 대의 장비에 빠른 속도로 저장을 하도록 해야 하며 수평적 확장(Scale-Out)이 가능하도록 해야 하는 것을 고려 한다면 NoSQL를 사용하는 것을 권장을 하고 있다. 하지만 데이터의 일관성을 확실하게 보장을 하거나 각기 다른 데이터들을 참조하는 Join 연산을 주로 하는 데이터베이스를 사용을 하는 것을 권장한다면 Relational Database를 사용하는 방안을 구상하는 것이 좋다. 즉 Relational Database와 NoSQL의 선택은 **일관성과 데이터 참조를 위한 보장으로** 볼 수 있다.

2. MongoDB 시작하기

여기서부턴 MongoDB에서 쓰는 DBMS의 이해를 가져보면서 더욱 나아가서 Spring Data와 JPA를 통한 REST API 구축과 Serverless를 활용하여 MongoDB에 있는 Document를 접목하는 과정으로 공부 일지를 작성할 예정이다. 이번 일지에서는 MongoDB에 대한 간략한 소개와 설치, 마지막으로 Relational Database의 대가인 MySQL와의 비교를 한 결과를 작성을 하겠다.

A. 소개 & 용어 정리

MongoDB는 위에서도 언급을 했지만, Aggregate-Oriented Model에 속하는 Document Model을 이용하여 구성한 NoSQL DBMS 중에 하나이다. MongoDB는 C++ 언어로 작성되었으며 NoSQL에서 Document Oriented Database(Document 지향 데이터베이스)를 기반으로 하면서 다른 환경 내부에서도 구현을 해볼 수 있는 Cross-Platform도 제공하고 있다. MongoDB에서 쓰는 대표적인 용어로는 Database, Collection, Document 3가지로 나뉠 수 있는데 각 용어의 뜻이 무엇을 뜻하는지 이해를 할 필요가 있다.

(1) Database

데이터베이스는 Collection들로 구성된 하나의 컨테이너로 볼 수 있는데 각 데이터베이스에서는 파일 시스템 내부에 현존하여 여러 파일들로 저장이 된다. 전형적으로 하나의 MongoDB 서버는 여러 개의 데이터베이스들을 가질 수 있다. 즉, Relational Database에서 다뤘던 Database와 같은 맥락으로 받아들일 수 있다.

(2) Collection

MongoDB에서 가지고 있는 Document들의 모음으로 이해를 할 수 있다. Relational Database에서 썼던 Table와 같은 맥락으로 이해를 할 수 있는데 각 Database에는 하나의 Collection이 무조건 포함이 되어 있다. 그렇지만 Relational Database에서는 Schema를 구축하고 난 후에 Table을 만드는 것이 관례이지만 Collection에는 Schema라는 개념을 포함하고 있지 않다. 대신에 마지막으로 작은 개념인 Document에는 동적 스키마(Dynamic Schema)를 가지고 있기 때문에 Collection 내부에 있는 Document들은 유사하거나 같은 목적이 존재할 수 있다.

(3) Document

Document는 Key-Value의 쌍으로 구성이 되어 있다. 이는 Relational Database에서는 Record로 생각할 수 있다. 또한 동적 스키마를 가지고 있는데 이는 같은 Collection 안에 있는 Document끼리 다른 Schema를 가질 수 있다는 의미로 생각할 수 있는데 이는 같은 Collection 내부에서 같은 필드 집합들이나 구조를 가질 필요가 없으며 각 Document 내부에 있는 공용 필드는 다른 타입의 데이터를 보유할 수 있다는 뜻으로 생각할 수 있다. 간략하게 Document가 어떤 구조로 되어 있는지에 대하여 아래 소스 코드를 통해 알아보자.

```
{
    _id : ObjectId(7df78ad8902c) // *
    userId : 'test123',
    phone : '01012345678',
    signDate : new Date(2018, 3, 3, 10, 20)
}
```

각 Document 내부에는 Field와 Value를 통하여 각각 상기가 되어 있는데 이러한 구조가 MongoDB 서버를 통해 전송을 하면 JSON 형 데이터가 Binary-Encoded Serialization(이진 변환 직렬화)를 통하여 보내지게 된다. 우리가 중요시 봐야 하는 데이터의 값은 *이다. _id 필드는 각 Document에 대하여 Unique을 보장하기 위해서 **MongoDB에서 자동적으로 생성을 해주는 객체 번호로** 이해를 할 수 있는데 Relational Database에서는 Primary Key로 이해를 하면 된다. 여기서 쓰인 _id의 ObjectId는 12Byte의 16진수로 이뤄지게 되는데 이 ObjectId에 대해서는 다음과 같이 구성이 되어 있다.

4Byte 현재 날짜의 Timestamp 값	3Byte MongoDB가 작동되고 있는 Server의 Machine ID	2Byte MongoDB Server의 Process Id	3Byte Collection 내부의 순차 번호

(4) Relational Database와 MongoDB와의 비교

Relational Database와 MongoDB에서 쓰는 용어를 다음과 같이 비교할 수 있다. 관계형 데이터베이스에서 쓰는 용어와 여기서 쓰이는 용어와는 이름이 다를 뿐, 개념은 크게 달라지지 않는다. 다음 쪽에서 어떻게 차이가 나는지 비교를 해보도록 하자.

Relational Database	MongoDB
Database	
Table	Collection
Tuple / Row / Record	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (기본적으론 MongoDB에서 제공하는 _id를 이용한다.)
데이터베이스 서버 및 클라이언트	
Mysqld / Oracle	Mongod
mysql/sqlplus	Mongo

B. 장점

MongoDB를 통하여 얻을 수 있는 장점은 NoSQL과 거의 다를 바가 없다.

- Schema-less. 같은 Collection 안에 존재하여도 다른 Schema를 가질 수 있다.(Dynamic Schema)
- 각 객체의 구조가 뚜렷하다.
- 복잡한 Join 연산이 필요 없다.
- Deep Query-Ability, 즉 복잡한 쿼리에 대한 대응이 MongoDB에서 Document 기반의 Query Language를 사용하여 Relational Database에서 사용했던 SQL 문장의 성능을 제공한다.
- 어플리케이션에서 사용하는 객체를 데이터베이스에 추가할 때 데이터에 대한 변환과 매핑 작업이 딱히 필요 없다.

C. 환경 구축

MongoDB의 환경을 구축하는 방법은 여기서 작성을 하게 되면 문서가 길어져서 아래의 링크를 통해서 설치하도록 한다. 다만 MongoDB를 다운 받을 때 주의를 해야 하는 Community Server를 통해서 msi 파일을 설치를 해야 한다. 또한 MongoDB에 있는 데이터베이스 목록들을 GUI를 통하여 데이터 목록을 확인을 하고 싶은 경우에는 MongoDB Compass를 활용해서 참고를 하도록 하자.

http://openeg.co.kr/742 [MongoDB 설치 및 환경 구축 과정]

환경 변수를 등록 완료 하고 난 후에 C 드라이브에 data 빈 파일과 그 속에 db 빈 파일을 생성을 해 주도록 하자. 파일 경로는 아래와 같이 정리 되어 있으니 참고하면서 빈 폴더들만 추가하면 된다.

C:₩data₩db

그러면 cmd 창에서 mongod를 입력을 하면 서버가 작동이 된다. 갑자기 다른 문장이 안 나온다고 이거 작동 안 되는 거 아닌가라는 의심을 가질 수도 있는데 방금 mongod를 통해 실행 한 cmd 창을 닫지 않은 이상 서버는 계속 연결 되어 있으니 안심하고 이용하면 되겠다. 물론 MongoDB Compass를 이용해서 각각 Collection과 Document들을 삽입, 조회, 삭제, 수정도 가능하고 MySQL Workbench처럼 사용하되 JSON 데이터 형태를 유지하도록 유의만 하면 된다.

[출처]

1. NoSQL 정의와 특성

https://namu.wiki/w/NoSQL - 나무위키 NoSQL 문서.

http://asfirstalways.tistory.com/352 - NoSQL 특성에 대한 소개

https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4_%EC%8A%A4%ED%82%A4%EB%A7%88 – 데이터베이스 스키마의 개념

2. MongoDB 시작하기

https://velopert.com/436 - MongoDB 시작을 위한 간단한 튜토리얼

https://www.mongodb.com/compare/mongodb-mysql - MongoDB와 MySQL 비교를 설명한 원서.