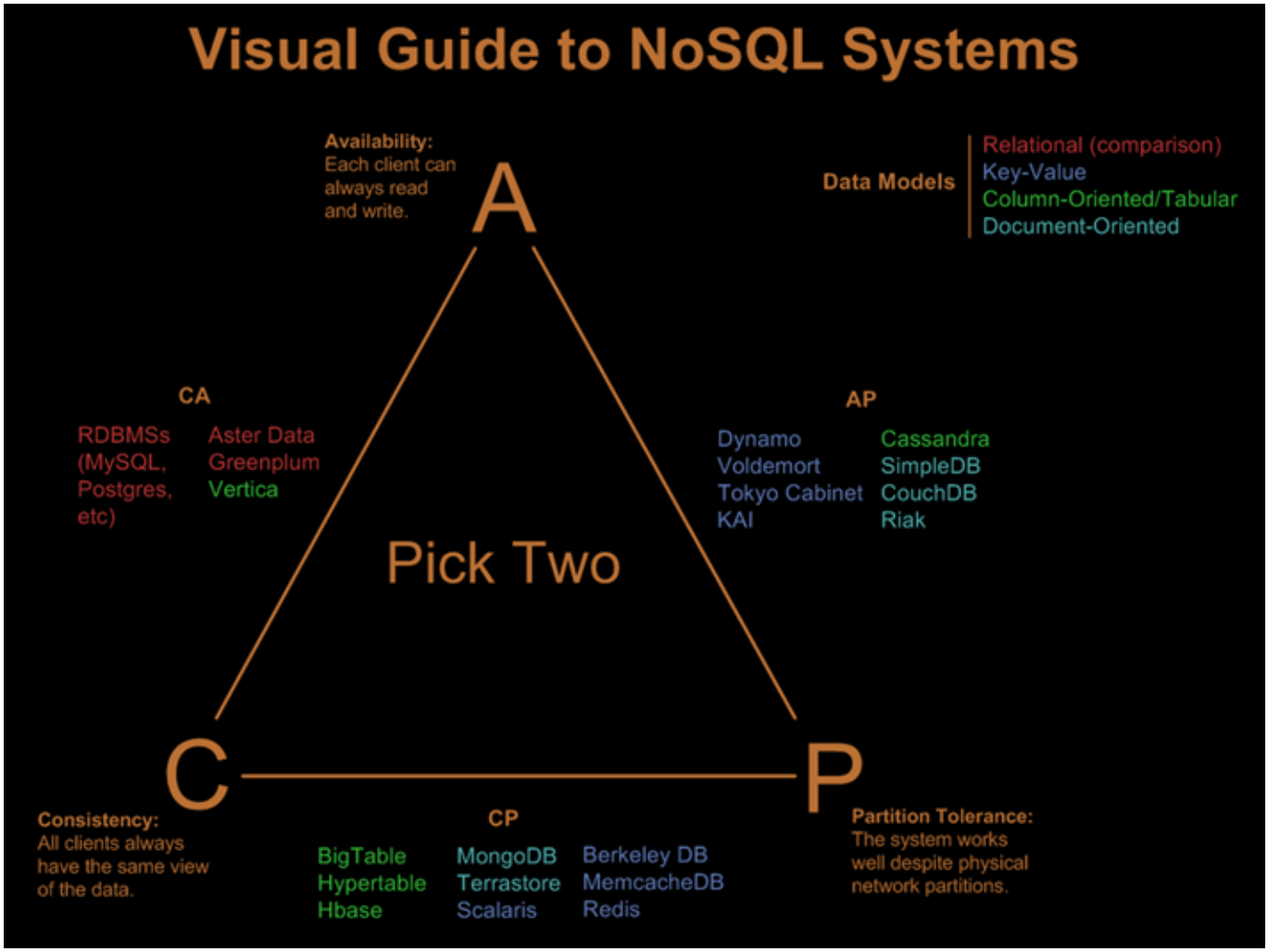
목차

Last update: 2017/02/02 11:02	nosql에_대해서_간단히_알아보자	https://atl.kr/dokuwiki/doku.php/nosql%EC%97%90_%EB%8C%80%ED%95%B4%EC%84%9C_%EA%B0%84%EB%8B%A8%ED%9E%88_%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90

NoSQL에 대해서 간단히 알아보자

출처 : https://embian.wordpress.com/2013/06/27/nosql-2/

CAP이론?



NoSQL 소개에는 빠지지 않고 등장하는 그림이다. 이 그림을 약 3~5초정도 보면 다음과 같은 의문이 생기게 된다.

" 그럼ACID는? "

> ACID(원자성, 일관성, 고립성, 지속성)는 데이터베이스 트랜젝션이 안전하게 수행된다는 것을 보장하기 위한 성질을 가리키는 약어이다. – 위키백과, (http://ko.wikipedia.org/wiki/ACID)

ACID는 RDBMS에서 매우 중요하게 생각하는 성질이라 할 수 있다. 그런데 NoSQL에서는 찾아보기 힘든 단어이다.

2000년 Eric Brewer가 “Towards Robust Distributed Systems”라는 제목으로 CAP 이론을 소개할 때 BASE에 대해서도 소개했었다.

성능과 가용성 등을 위해서 ACID의 C와 I의 속성을 포기하고 분산 시스템에 더 적합하다고 생각되는 성질을 정리한 것이 BASE이다.

BASE가 나타내는 성질은 다음과 같다.

Basically Available : 기본적으로 Available하고 Soft-state : 사용자가 관리(refresh, modify)하지 않으면 Data가 expire 될 수도 있으며 Eventually consistency : 지금 당장은 아니지만 언젠가는 Data가 일관성을 가진다

전달하고자 하는 의미는 알겠는데 찝찝한 부분이 있다!

1. Basically Available은 Basically라는 부사를 사용하지 않고 Availablility라고만 표현해도 되고 2. Eventually consistency도 마찬가지로 Eventually라는 부사를 사용하고 있다 3. 게다가 ACID처럼 명사들만의 약자가 아닌 Basically Available = 부사 + 형용사, Soft-state = 명사, Eventually consistency = 형용사 + 명사 로 구성되어있다.

이렇게 억지로 짜맞춘 단어처럼 보이는 BASE는 ACID와의 관계를 고려하면 좀 더 간결하고 명확한 의미 전달을 할 수 있는 명사들의 약자를 쓰지 않은 이유를 알 수 있다.

Acid와 Base의 비교 #1

항목	Acid	Base
리트머스	파란색 -> 빨간색	빨간색 -> 파란색
상호반응	BASE와 반응하여 중성화	Acid와 반응하여 중성화
특징	신맛이 남	미끌거림

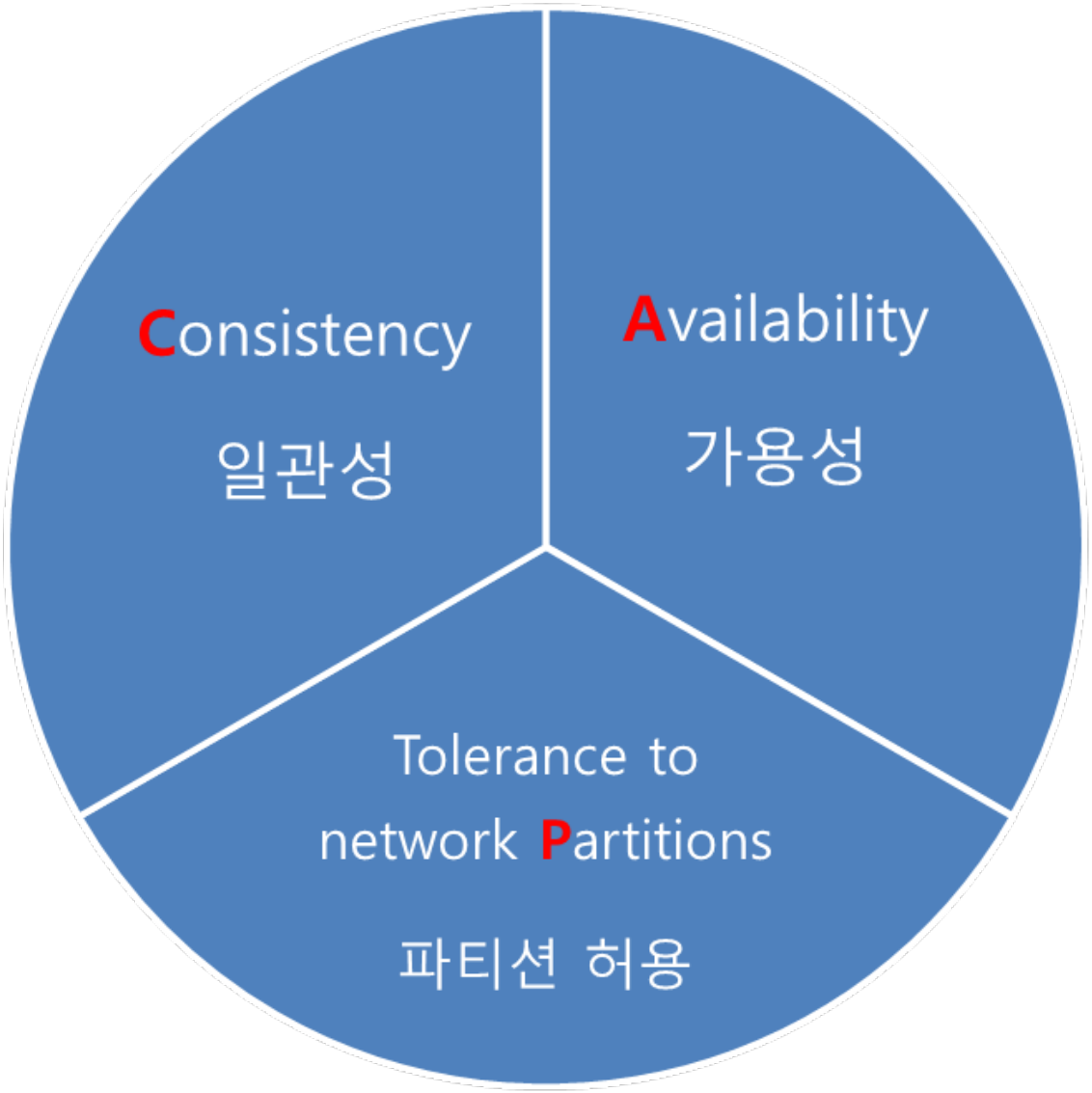
ACID와 BASE의 비교 #2

ACID	BASE
강한 Consistency	약한 Consistency
중요한 commit!	Availability가 우선!
엄격한 데이터	열심히 노력합니다.

분산 시스템에서 관리되는 데이터의 특징을 나타내는 BASE라는 단어는 의미의 전달보다는 중학교때 배운 산(Acid)과 염기(Base)와 같아보이려는 노력에 의해서 탄생했다는 것을 알 수 있다;;

CAP이론!

BASE의 탄생에 대해서는 그냥 그러려니~ 하고 넘어가고 CAP이론에 대해서 알아보자. 제일 처음에 있던 CAP 세모는 영어로 써져있고 글자도 많으니 좀 더 간단한 그림을 그려보면 다음과 같다.



CAP이론은 " 분산 시스템에서는 위 그림의 3개 속성을 모두 가지는 것이 불가능하다! " 이다.

CAP이론에서 보여주는 3개의 속성에 대해서 각각 알아보면 다음과 같다.

Consistency

- 우선.. CAP이론에서 말하는 Consistency는 ACID의 'C'가 아니다! ACID의 'C'는 " 데이터는 항상 일관성 있는 상태를 유지해야 하고 데이터의 조작 후에도 무결성을 해치지 말아야 한다 " 는 속성이다.
- CAP의 'C'는 "Single request/response operation sequence"의 속성을 나타낸다. 그 말은 쓰기 동

작이 완료된 후 발생하는 읽기 동작은 마지막으로 쓰여진 데이터를 리턴해야 한다는 것을 의미한다.
- 모든 노드가 같은 시간에 같은 데이터를 보여줘야 한다. (저장된 데이터까지 모두 같을 필요는 없음)
- 정리해보면 Consistency라는 단어보다는 Atomic이라는 단어가 더 정확하게 특징을 나타낸다고 할 수 있다. (Atomic Data Object, Atomic Consistency)
- 그럼 "CAP가 아니라 AAP가 되어야 할텐데.." 라는 생각이 들 수도 있다. 어차피ACID와 비교하기 위해서 BASE라는 단어도 억지로 만들어내는데 AAP보다는 CAP가 더 이뻐보이니 그렇게 했다고 생각하고 넘어가면 된다.

Availability

- 가용성
- 흔히 보는 단어이고 의미도 크게 혼동될 이유가 없어보인다. " 특정 노드가 장애가 나도 서비스가 가능해야 한다 " 라는 의미를 가진다.
- 데이터 저장소에 대한 모든 동작(read, write 등)은 항상 성공적으로 리턴되어야 한다.
- 명확해 보이는 단어이기는 하지만 분산 시스템에서의 특징을 말하는 것이기 때문에 " 서비스가 가능하다 " 와 " 성공적으로 리턴 " 이라는 표현이 애매하다. 얼마동안 기다리는 것 까지를 성공적이라고 할 수 있느냐에 대한 문제가 남아있다. " 20시간정도 기다렸더니 리턴이 왔에 Availability가 있는 시스템이야! " 라고 할 수 없기 때문이다.
- 다시한번 " 성공적으로 리턴 " 에 대해서 보면 모든 동작에 대해서 시스템이 "Fail!!"이라는 리턴을 성공적으로 보내준다면 그것을 Availability가 있다고 해야 하느냐에 대해서도 애매하다. CAP를 설명하는 문서들 중 "Fail!!"이라고 리턴을 하는 경우도 " 성공적인 리턴 " 이라고 설명하는 것을 보았다.

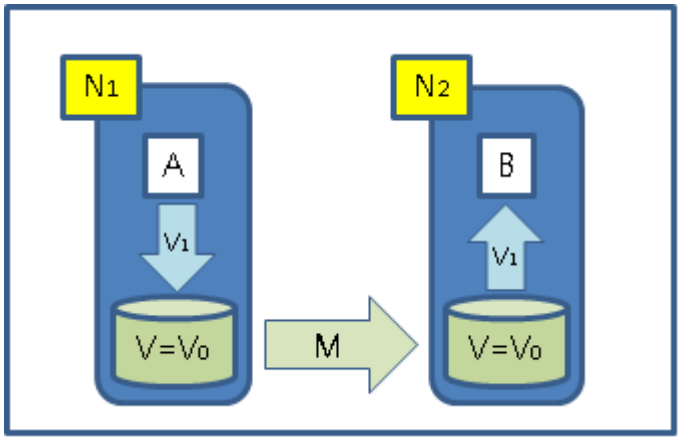
Tolerance to network Partitions

- 원래는 Tolerance to network Partitions인데 보통은 Partition-tolerance라고도 한다.
- 노드간에 통신 문제가 생겨서 메시지를 주고받지 못하는 상황이라도 동작해야 한다.
- Availablity와의 차이점은 Availability는 특정 노드가 " 장애 " 가 발생한 상황에 대한 것이고 Tolerance to network Partitions는 노드의 상태는 정상이지만 네트워크 등의 문제로 서로간의 연결이 끊어진 상황에 대한 것이다.

CAP 이론의 이해

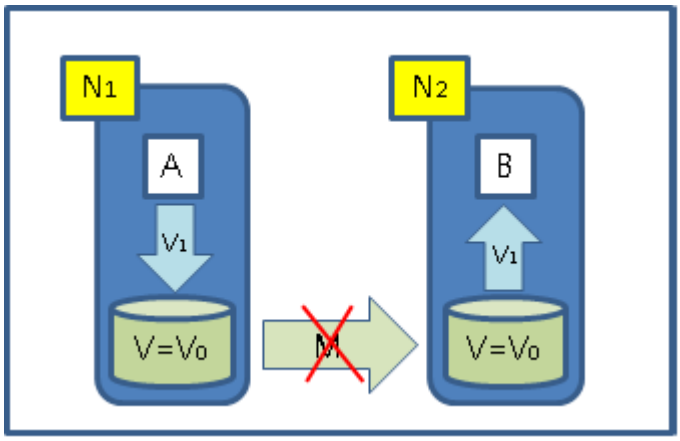
CAP이론은 2002년 MIT의 Seth Gilbert와 Nancy Lynch에 의해서 증명되었다. CAP이론을 증명하는 논문에는 그림이 하나도 없어서 너무너무 재미 없다. 재미없는 논문 보다는 간단한 그림으로 CAP이론을 이해해볼 수 있다.

CAP이론을 이해해보기에 앞서.. 다음과 같은 상황을 가정해본다.



– 네트워크가 N1, N2로 구분된 분산환경이다. – 각 DB 노드는 V=V0이라는 값을 가지고 있다. – 각 네트워크에는 A, B라는 클라이언트가 존재한다. – A는 V=V1이라고 쓰고 B가 그것을 읽는다.

이런 환경에서 메시지 전달 과정(M)에서 문제가 생겼을 때..



1. “C”가 꼭 필요한 상황인 경우 – A가 V1이라고 썼기 때문에 B는 V1이라고 읽을 수 있어야만 한다. – A의 쓰기 동작은 M이 복구되기 전까지는 성공할 수 없다. – M이 복구되기 전까지는 A의 Write는 block되거나 실패해야 한다. = Availability가 없음 = CP – M이 문제가 생길 수 없도록 구성 = Partition-Tolerance가 필요 없음 = CA

2. “A”가 꼭 필요한 상황인 경우 – 어떤 경우에도 서비스가 Unavailable하면 안된다. – A와 B가 꼭 동일한 데이터를 읽을 필요는 없음 = AP – M이 문제가 생길 수 없도록 구성 = Partition-Tolerance가 필요 없음 = CA

3. “P”가 꼭 필요한 상황인 경우 – 메시지 전달 과정(M)에서 문제가 생기더라도 시스템에 영향이 가서는 안된다. – A와 B가 꼭 동일한 데이터를 읽을 필요는 없음 = AP – A의 쓰기 동작은 M이 복구되기를 기다린다. = 그동안 쓰기 서비스 불가능 = Availability가 없음 = CP

NoSQL??

NoSQL이라는 단어에 대해서 “ 적극적으로 ” 인터넷을 뒤져보았더니 다음과 같이 설명하고 있었다.

1. No SQL : SQL이 없다는 의미이다.
2. Not Only SQL : SQL뿐이 아니다.. SQL말고도 더 있다
3. NOn-relational operation database SQL : 비관계형 DB SQL

종합해보면 NoSQL은 **“SQL이 없으면서도 SQL뿐만 아니라 다른거도 있는 비관계형 DB이다 ”**라고 할 수

있다. 이건 누가 들어도 “ 대단해! ” 라고 말할만 하다.

하지만 이미 “BASE”라는 단어에서 느낀것이 있기 때문에 “NoSQL”이라는 단어에서 “N은 이런 의미이고 O는 이런 의미이고.. ” 와 같이 단어 자체에 의미를 부여하려는 쓸데없는 시도는 무시하고 다시NoSQL에 대해서 정의를 내리면 NoSQL은 “ 관계형 데이터 모델을 사용하지 않는 모든Database 또는 Data Store” 라고 할 수 있다.

NoSQL의 종류

NoSQL에는 어떤 것들이 있는지 웹서핑을 해봤다. 결과는 다음과 같다.

- Key-Value Oriented Databases
- Column Oriented Databases
- Document Oriented Databases
- 등등

NoSQL의 종류에 대해서는 기본부터 알아보도록 하자.

NoSQL의 데이터 모델은 기본적으로 Key-Value Store 모델을 사용한다.

Key	Value
K_1	V_1
K_2	V_2
…	…

기본 데이터 모델을 기준으로 각각의 Key-Value 관리 방법을 알아보면 다음과 같다.

Key-Value Oriented

– Value가 String, Integer, List, Hash 등의 기본 자료형인 경우이다

Key	Value
K_1	“VALUE”
K_2	1
…	…

– 이런 형식의 데이터 모델을 사용하는 경우를Key-Value Oriented Database라고 한다. – 키에 대한 접근은 빠르지만 범위 검색에 대해서는 취약하다. – 이런 데이터 모델을 사용하는DB로는 Redis, Memcached, Tokyo Cabinet 등이 있다.

Ordered Key-Value Oriented

– Key-Value Oriented 모델의 경우 범위 검색에 대해서 취약하다는 단점이 있다. – 만약Key, Value를 정렬해놓는다면 범위 검색에 대한 단점을 어느정도 극복할 수 있다.

Column Family Oriented

– 하나의Key에 하나의 Value만을 저장하는 단점을 극복했다. – 하나의Key에 여러개의 Column을 저장한다. – Column-Value의 묶음을 Column Family라고 한다.

Key	Value
K_1	C_1 | C_2 | C_3 | ... V_1 | V_2 | V_3 | ...
K_2	C_2 | C_5 | C_6 | ... V_4 | V_5 | V_6 | ...
...	...

– 흔히Column Oriented Database라고 하지만 Column Family Oriented Database라고 해야 한다. Column Oriented는 RDBMS에서 사용하는 데이터 모델 중 하나이다. – 보통은Ordered Key-Value Oriented 모델을 사용한다. 다시 말하면 Key는 정렬되어 저장되고 Column도 Column Name별로 정렬되어 저장하는 모델을 사용한다. – 이런 데이터 모델을 사용하는DB로는 HBase, Cassandra 등이 있다.

Document Oriented

– Value에 저장되는 값이 Document(JSON, XML 등)이다.

Key	Value
K_1	JSON/XML Document
K_2	JSON/XML Document
...	...

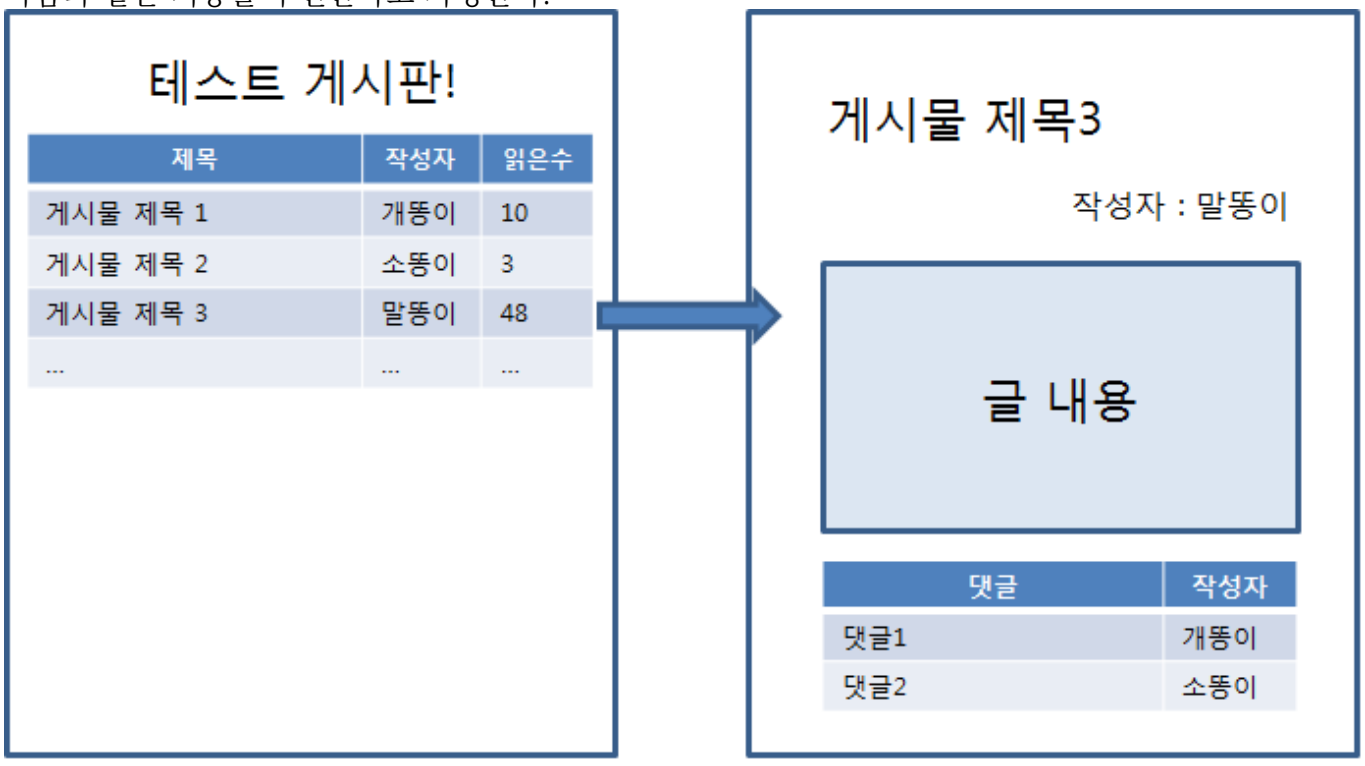
– 이런 데이터 모델을 사용하는DB로는 MongoDB, CouchDB 등이 있다.

NoSQL 데이터 모델링

RDBMS의 데이터 모델링과 NoSQL의 데이터 모델링을 비교하면 다음과 같다.

RDBMS	NoSQL
	
• RDBMS는 다양하고 복잡한 쿼리를 지원한다. • 테이블 디자인 후에 필요한 데이터를 가져올 수 있도록 쿼리를 작성한다. • 테이블간 데이터 중복을 최소한으로 하는 정규화된 테이블 설계를 한다.	• NoSQL은 매우 단순한 쿼리만을 지원한다. • 어떤 쿼리 결과가 필요한지 먼저 정의한 후 그 결과를 얻을 수 있도록 테이블을 디자인해야 한다. • 데이터가 두개 이상의 테이블에 중복되게 저장하는 비정규화된 테이블 설계를 한다.

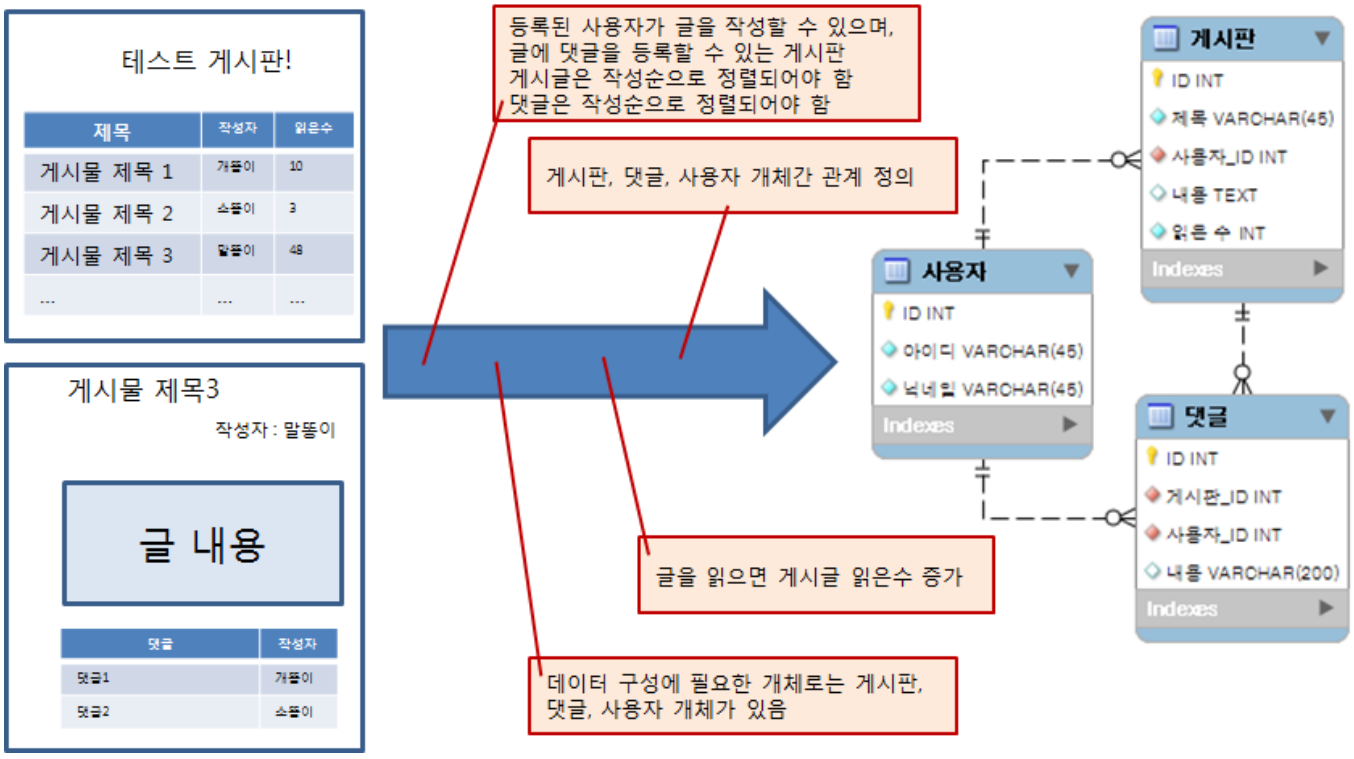
다음과 같은 기능을 구현한다고 가정한다.



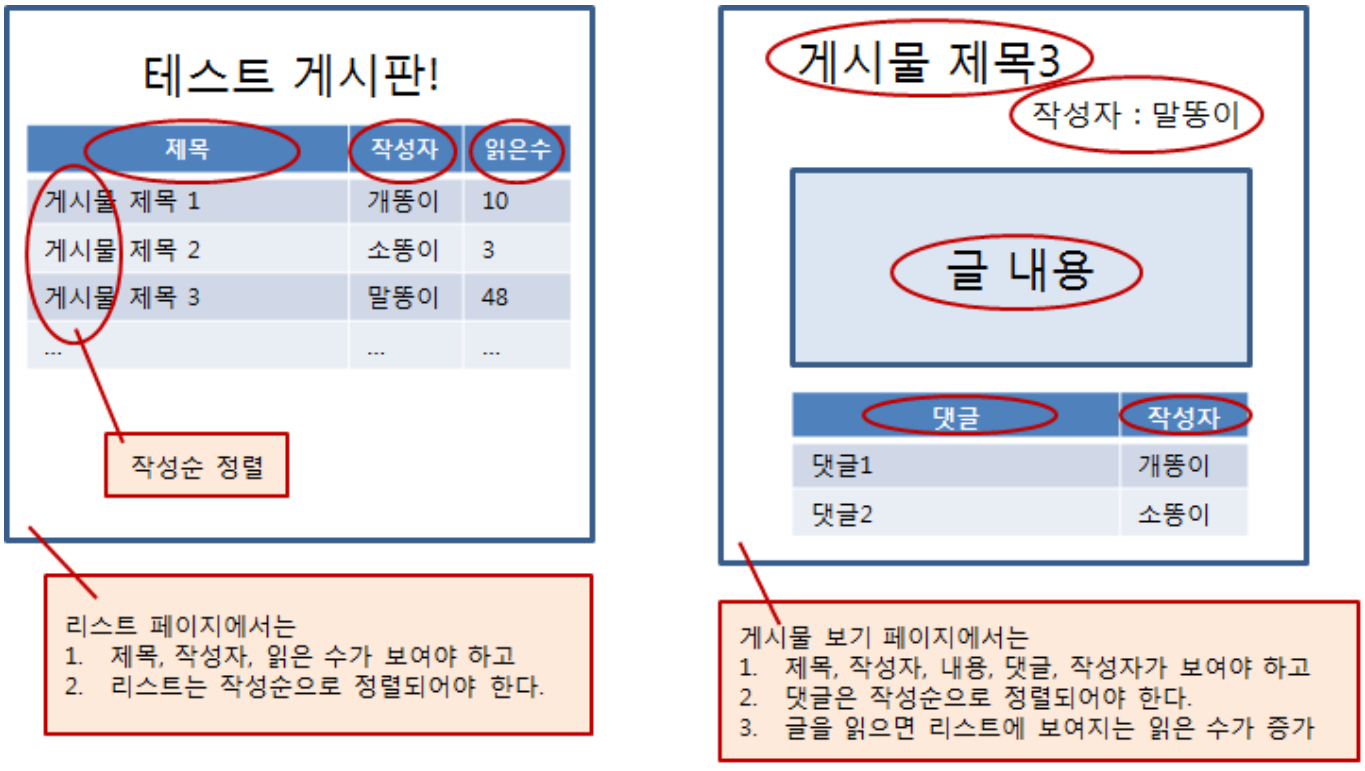
NoSQL로 구현할 때의 모델링 절차를 살펴보면 다음과 같다.

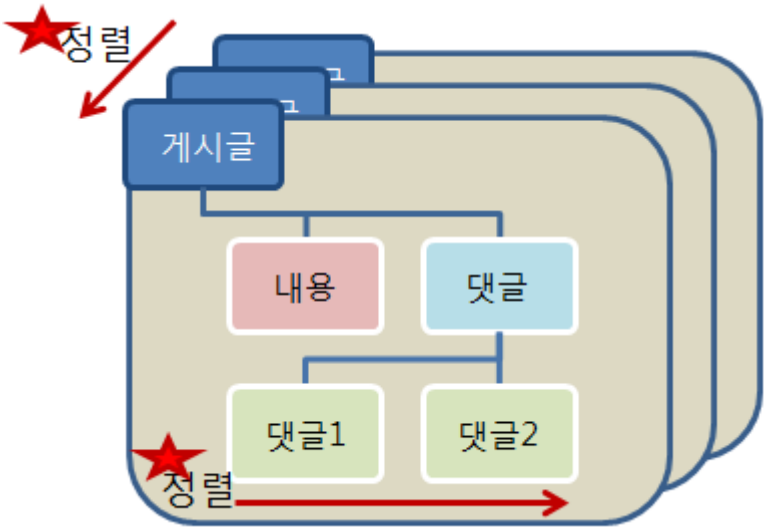
데이터 분석

– RDBMS도 모델링을 시작하기 위해서 데이터 분석을 해야 한다. 동일한 절차이지만 RDBMS와 NoSQL은 데이터에 대해서 서로 다른 시각으로 접근을 해야 한다. – RDBMS에서의 데이터 분석은 다음과 같다.



– 반면에NoSQL의 데이터 분석은 다음과 같다.

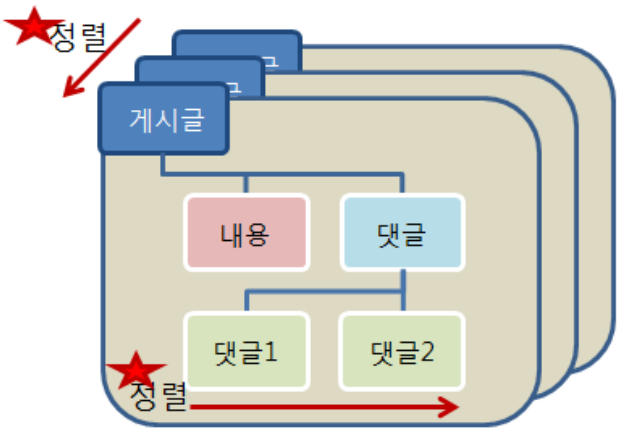




1) 게시판 내에서 게시글은 시간순으로 정렬되어야 함 2) 게시글:내용 = 1:1 3) 게시글:댓글 = 1:N 4) 댓글은 게시글 내에서 시간순으로 정렬되어야 함 5) “ 정렬 ” 에 별표RDBMS는 데이터의 관계에 집중해서 데이터 분석을 진행하는 반면에 NoSQL은 표현되는 데이터와 정렬되는 특징에 집중해서 분석을 진행한다.

쿼리 디자인

– 데이터 분석을 마치면NoSQL은 데이터의 특징을 고려해서 어떤 종류의 쿼리가 필요한지에 대해서 디자인한다.



쿼리 1

SCAN 게시판 order by 작성순
⇒ [{제목, 작성자, 읽은수, 내용, [{댓글, 작성자}]}, ...]

GET 게시글 order by 댓글.작성순
⇒{제목, 작성자, 읽은수, 내용, [{댓글, 작성자}]}

쿼리 2

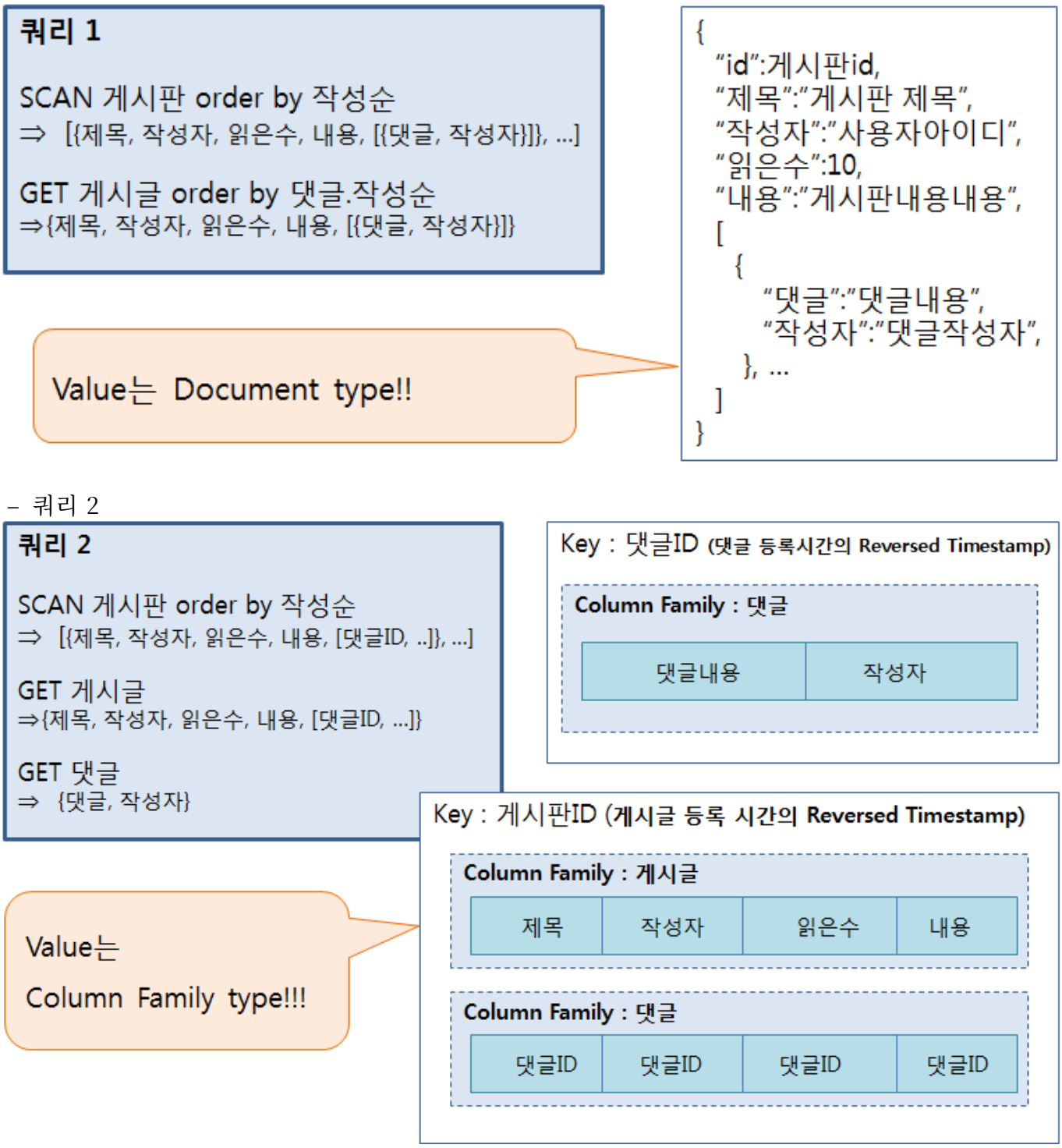
SCAN 게시판 order by 작성순
⇒ [{제목, 작성자, 읽은수, 내용, [댓글ID, ..]}, ...]

GET 게시글
⇒{제목, 작성자, 읽은수, 내용, [댓글ID, ...]}

GET 댓글
⇒ {댓글, 작성자}

테이블 디자인

– 쿼리 디자인에서 2가지 방법을 디자인했다. 각각의 쿼리에 맞는 테이블을 디자인하면 다음과 같다. – 쿼리 1

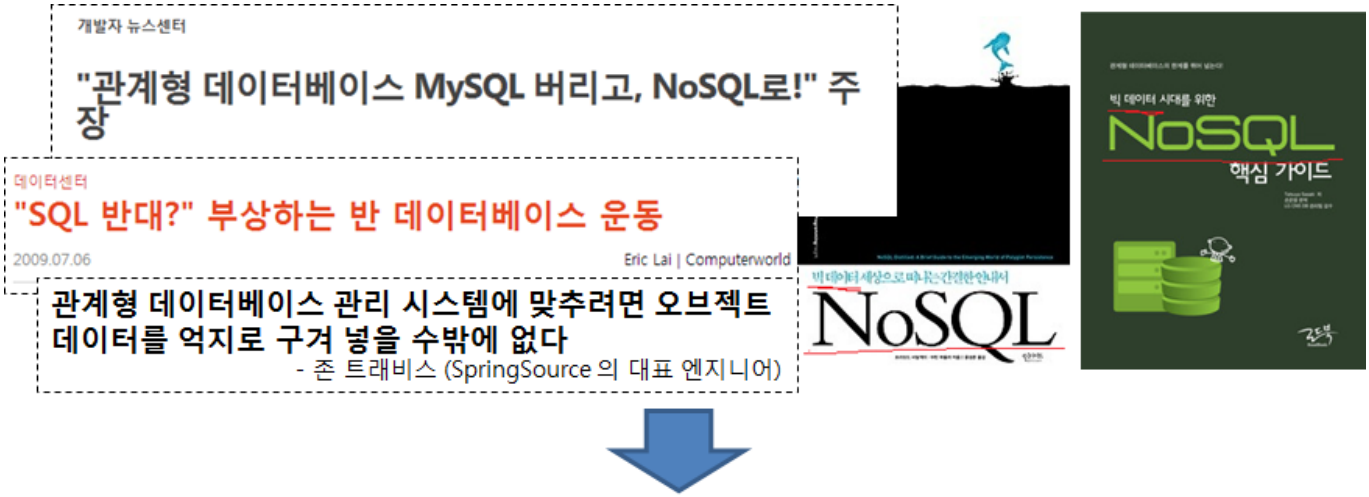


- 쿼리 2

- 각 쿼리에 맞는 테이블을 디자인했을 때 적합한NoSQL 모델은 Document Oriented, Column Family Oriented 라고 할 수 있다.

NoSQL 언제 사용하면 좋을까?

NoSQL에 대해서 조사해봤더니 다음과 같은 정보를 얻을 수 있었다.



빅데이터를 처리 = NoSQL?

역시나 다들 " 빅데이터 처리는NoSQL이야! " 라고 하는 것 같다. 그러면 다른 유명한 업체의 사용 경험을 들어보도록 하자!

1. Twitter

우리는 데이터도 짱 많고 데이터 증가 속도도 겁나게 빨라!
지금은 RDBMS인 MySQL을 쓰고있는데 불만이 많아!

1. NoSQL(Cassandra)은 가용성도 좋고
2. 쓰기 속도도 빠르고
3. 오픈소스 커뮤니티도 활발해!

앞으로는 트윗 저장소로 MySQL 대신 Cassandra를 쓸꺼야!

트윗 저장소로는 그냥 MySQL 쓰기로 했어요~
많은 데이터를 새로운 기술로 마이그레이션 하기 부담되기도 하고..
하지만 Data Mining결과를 저장하기도 하고 대용량 실시간 분석에도 쓰고있기는 해요~

2. Tumblr

우리는 데이터도 짱 많고 데이터 증가 속도도 겁나게 빨라!
데이터 저장소로는 RDBMS인 MySQL을 사용하고 있고 NoSQL(Hbase)로 바꿀 생각은 없어!

MySQL에 데이터를 잘게 쪼개서 저장하고 있어!
Hbase는 Shorten URL처리에 사용하고 있어!
Shorten URL cache에는 Redis도 쓰고있어!
Dashboard의 Secondary Index는 Redis를 중심으로 만들었어!

MySQL을 제대로 쓰려고 노력 많이 했어요~
Redis 도입은 참 잘한거 같아요

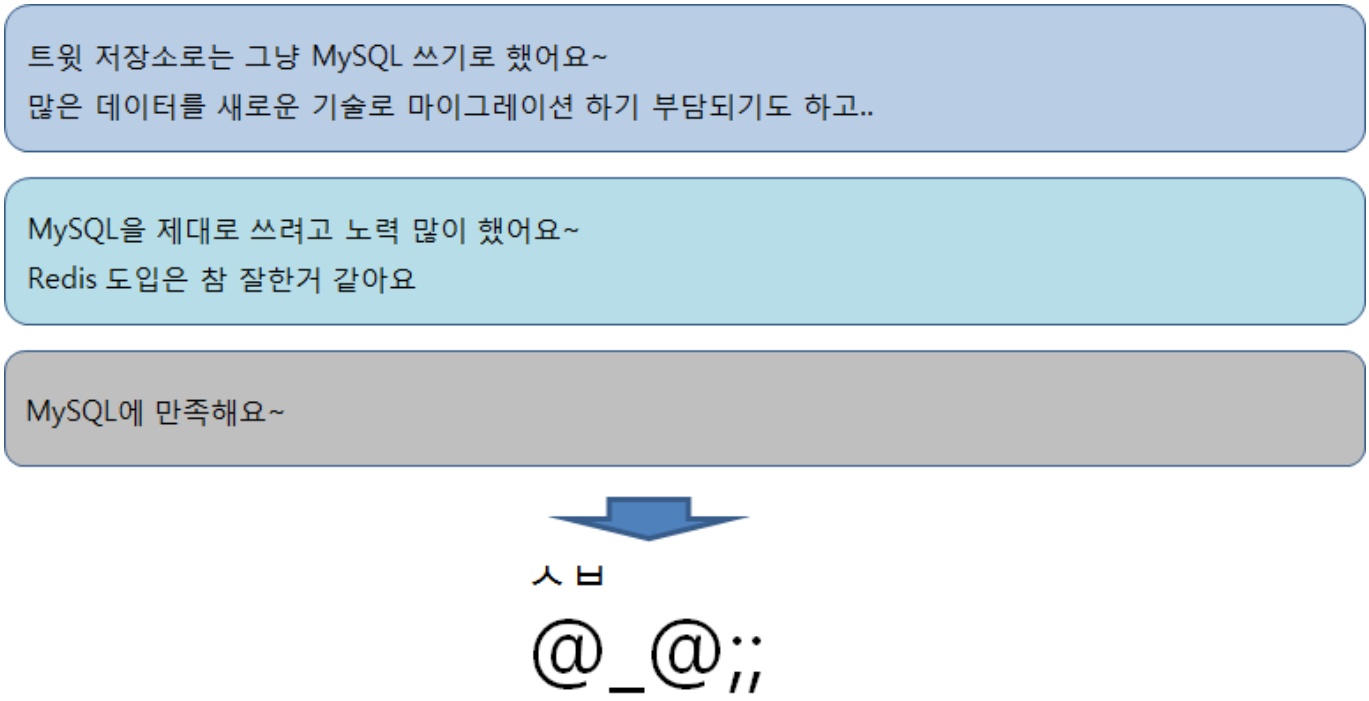
3. EVERNOTE

NoSQL은 짱빠르고 가용성도 높아!
그래도 우리는 MySQL을 쓸꺼야!

우리한테는 ACID가 매우 중요해!
다들 Big Data를 MySQL로 가능하냐고 물어보는데 사용자별로 데이터셋을 쪼개놨기 때문에 우리 입장에서는 "Big Data"가 아닌 "여러 개의 Medium Data"야!

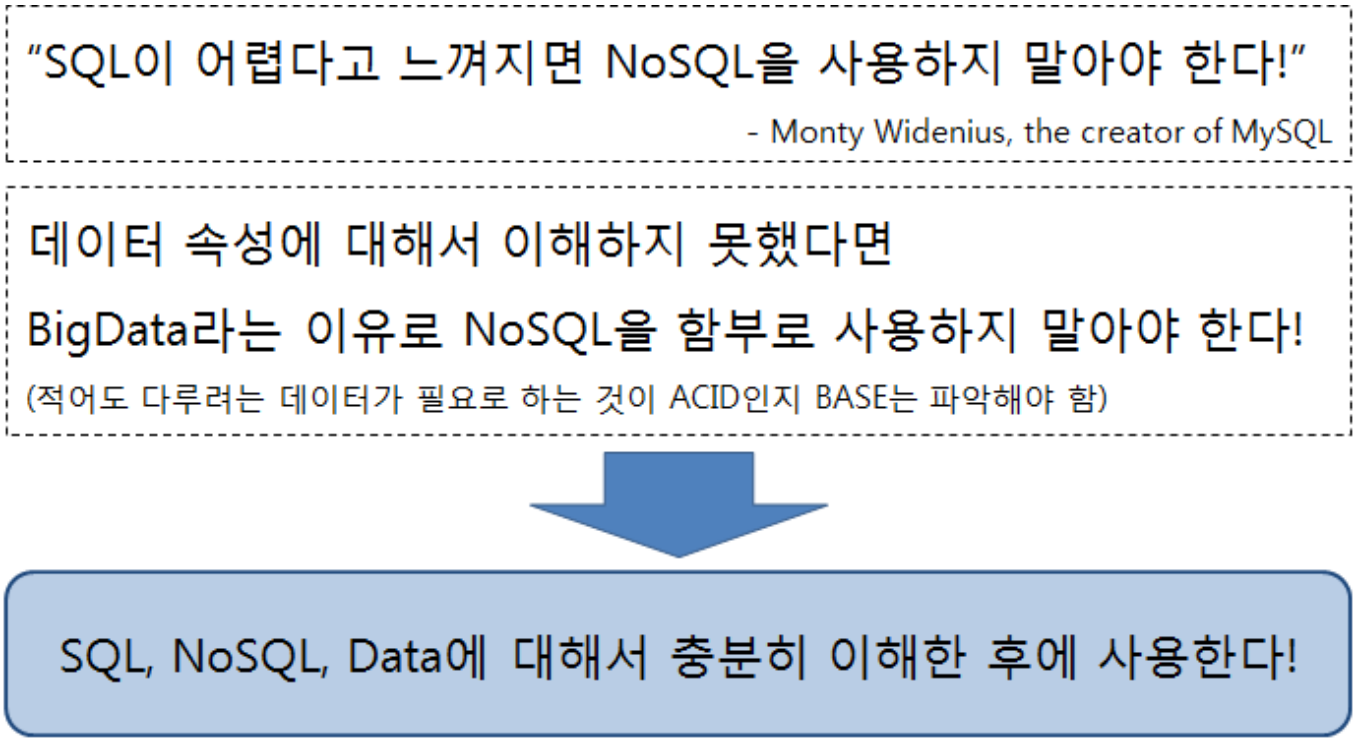
MySQL에 만족해요~

4. 결론



기대하는 결과가 나오지 않을 경우 질문을 바꿔보는 것도 좋은 방법이라 생각한다.

언제 NoSQL을 사용하면 안되나?



(바쁜 사람들을 위한 친절한)요약

1. RDBMS가 ACID라면 NoSQL는 BASE이다
2. 분산 시스템에서는 Consistency, Availability, Partition Tolerance의 특징을 모두 가지는 것이 불가능하다
3. NoSQL은 관계형 데이터 모델을 사용하지 않는 모든 Database 또는 Data Store를 말한다.
4. NoSQL의 기본 데이터 모델은 Key-Value이다. Value의 형식에 따라서 Key-Value, Column Family Oriented, Document Oriented로 구분되기도 한다.
5. 데이터 모델링에서 RDBMS는 테이블 디자인이 중요하지만 NoSQL은 쿼리 디자인이 중요하다.
6. RDBMS는 테이블 디자인에 정규화를 하지만 NoSQL은 데이터 비정규화를 한다.
7. 빅데이터 처리에 NoSQL이 꼭 필요한 것은 아니다.
8. 유행을 무조건 따르지 말고 SQL, NoSQL이 무엇인지 알아보고 내가 다루려는 Data에 대해서 파악을 한 후에 사용해야 한다!