

nosql

-

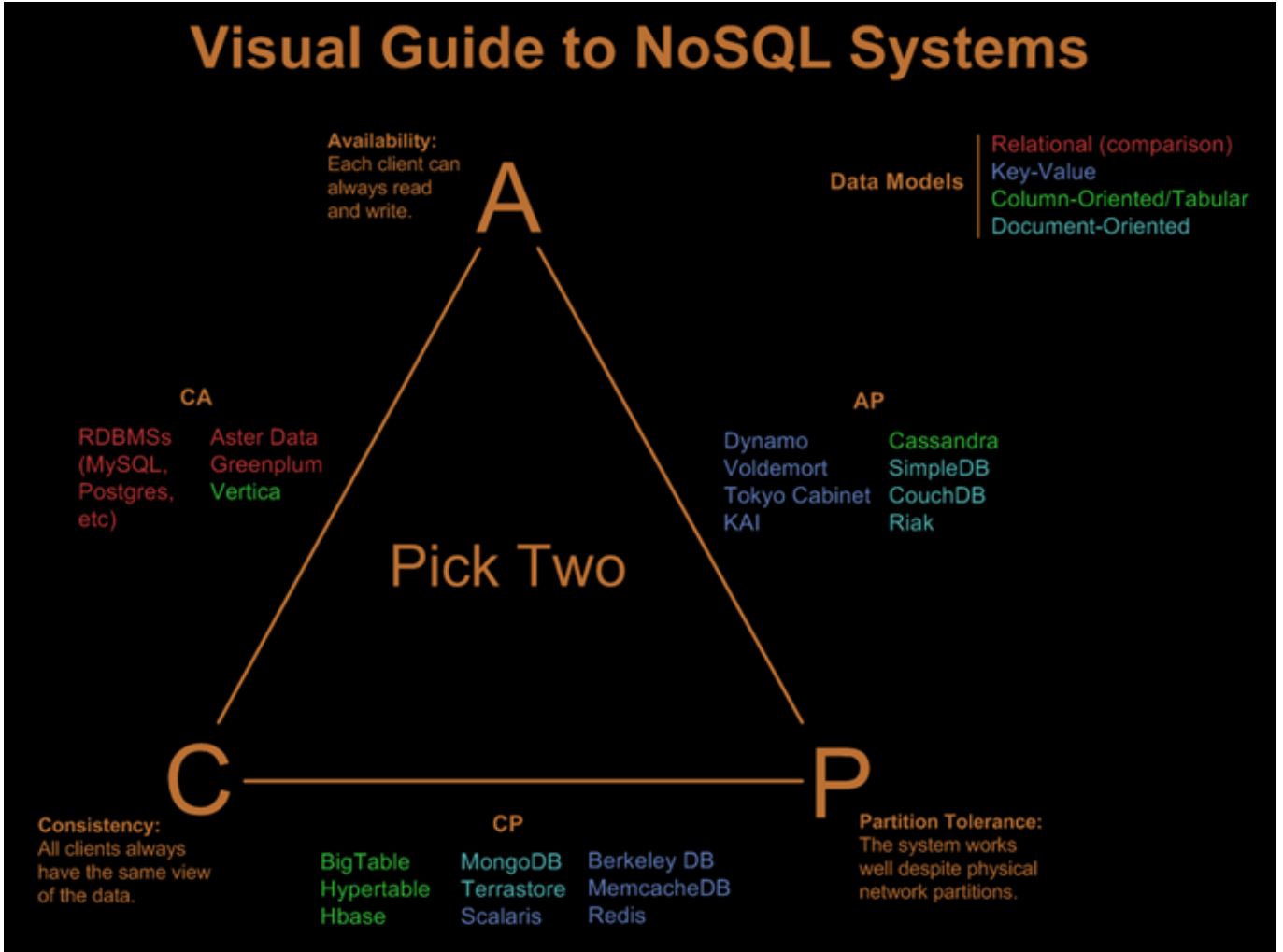
Last update: 2017/02/02 11:02 - https://atl.kr/dokuwiki/doku.php/nosql%EC%97%90_%EB%8C%80%ED%95%B4%EC%84%9C_%EA%B0%84%EB%8B%A8%ED%9E%88_%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90

-

NoSQL

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

CAP ?



NoSQL

3~5

“ ACID ? ”

ACID(, , ,) 가
 . - , (<http://ko.wikipedia.org/wiki/ACID>)

ACID RDBMS

NoSQL

2000 Eric Brewer가 "Towards Robust Distributed Systems" CAP
 BASE

가 ACID C I
 BASE

BASE가

Basically Available : Available Soft-state : 가 (refresh, modify)
 Data가 expire Eventually consistency : 가 Data가
 가

!

1. Basically Available Basically Availability 2.
 Eventually consistency 가 Eventually 가 ACID
 가 Basically Available = + , Soft-state = , Eventually consistency =
 +

BASE ACID

Acid Base # 1

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

ACID BASE # 2

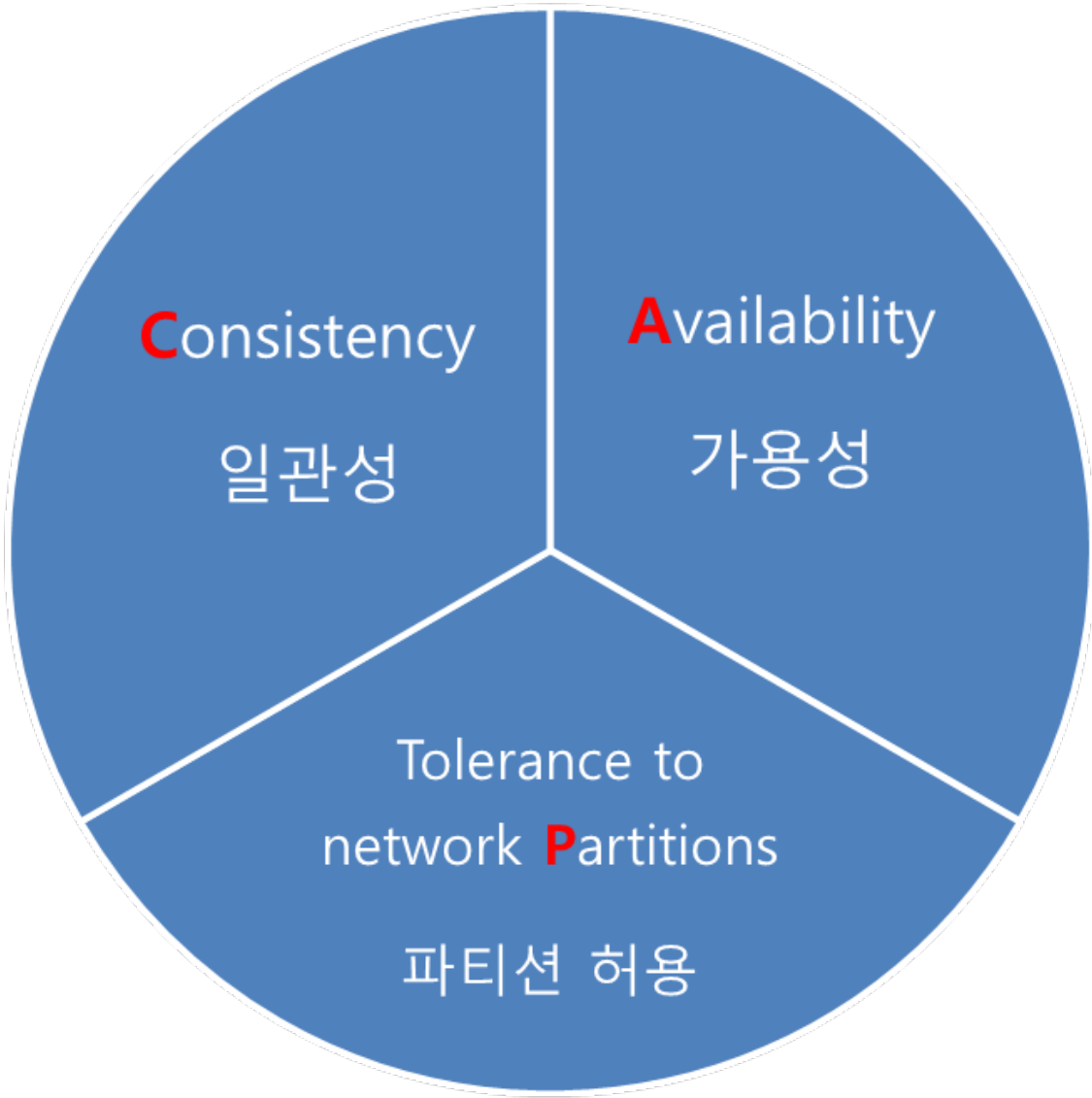
ACID	BASE
강한 Consistency	약한 Consistency
중요한 commit!	Availability가 우선!
엄격한 데이터	열심히 노력합니다.

BASE

(Acid) (Base) ;

CAP !

BASE ~ 가 CAP
CAP .

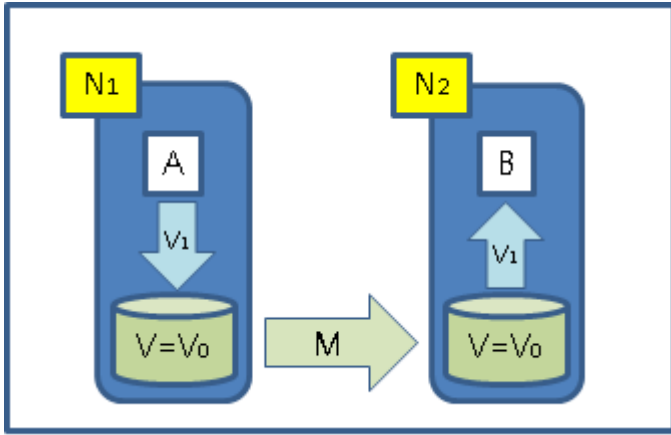


CAP “ 3 가 가 !” .

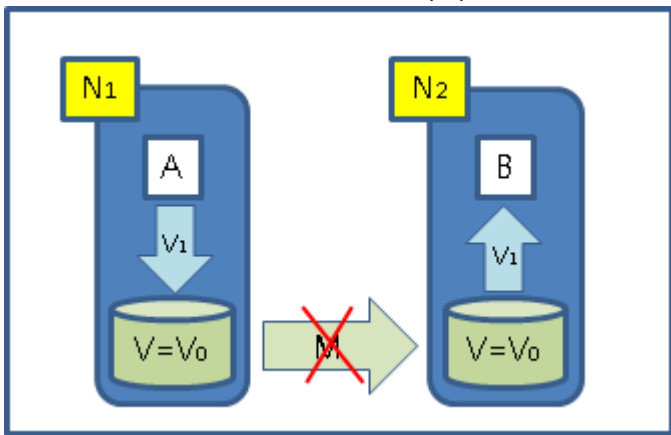
CAP 3 .

Consistency

- .. CAP Consistency ACID ‘C’가 ! ACID ‘C’ “ ”
- CAP ‘C’ “Single request/response operation sequence” .



- 가 N1, N2 DB V=V0 가 . -
 A, B 가 . - A V=V1 B가 . -
 (M) 가 ..



1. "C"가 - A가 V1 B V1 . - A
 M = Availability가 = CP - M 가 A Write block = Partition-
 Tolerance가 = CA
2. "A"가 - 가 Unavailable . - A B가
 = AP - M 가 = Partition-Tolerance가
 = CA
3. "P"가 - (M) 가 가
 . - A B가 = AP - A M
 . = 가 = Availability가 = CP

NoSQL??

NoSQL “ ”

1. No SQL : SQL .
2. Not Only SQL : SQL .. SQL
3. Non-relational operation database SQL : DB SQL

NoSQL “SQL SQL DB ”

가 “ ! ”
 “BASE” “NoSQL” “N
 O “..” NoSQL “ Database NoSQL
 Data Store”

NoSQL

NoSQL

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

NoSQL

NoSQL Key-Value Store

Key	Value
K ₁	V ₁
K ₂	V ₂
...	...

Key-Value

Key-Value Oriented

- Value가 String, Integer, List, Hash

Key	Value
K ₁	“VALUE”
K ₂	1
...	...

- Key-Value Oriented Database DB - Redis,
 Memcached, Tokyo Cabinet

Ordered Key-Value Oriented

- Key-Value Oriented . - Key, Value

Column Family Oriented

- Key Value . - Key Column
 . - Column-Value Column Family

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	...
...	...			

- Column Oriented Database Column Family Oriented Database
 Column Oriented RDBMS . - Ordered Key-Value
 Oriented Key Column Column Name
 . - DB HBase, Cassandra

Document Oriented

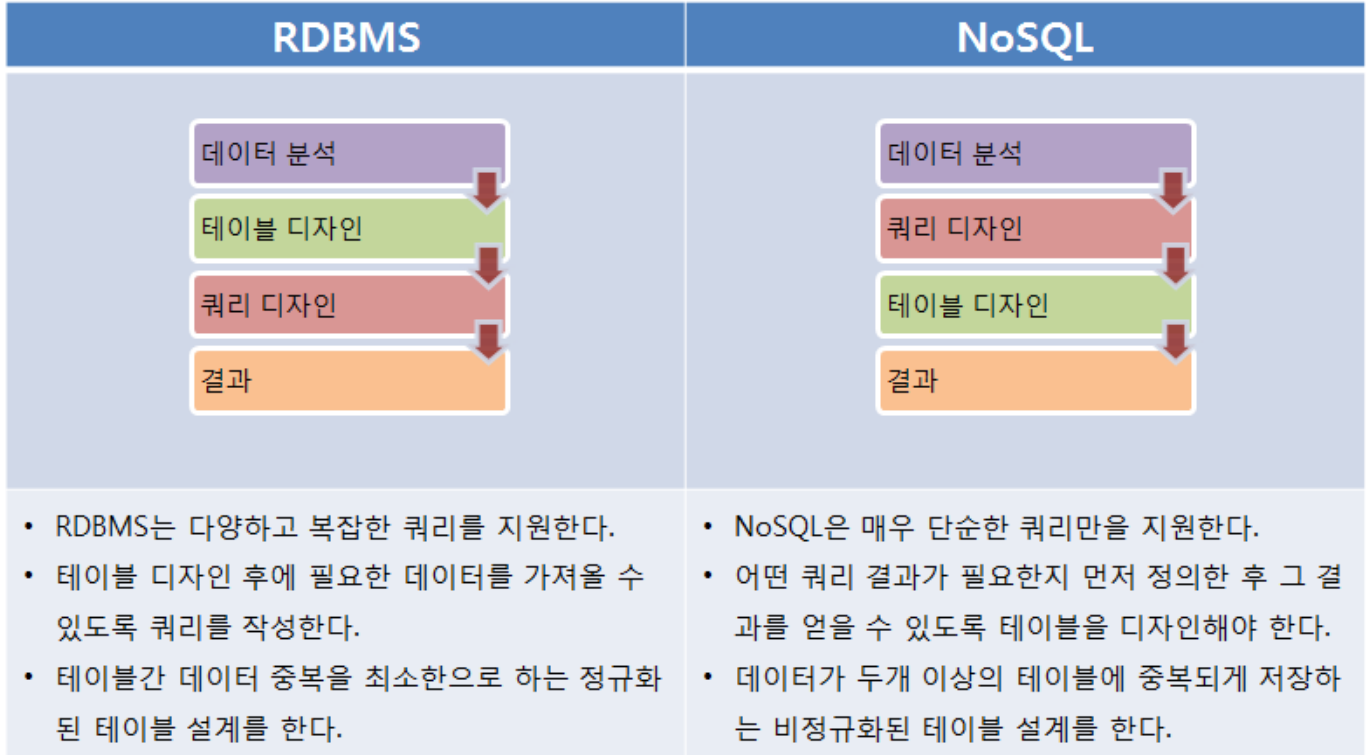
- Value Document(JSON, XML) .

Key	Value
K ₁	JSON/XML Document
K ₂	JSON/XML Document
...	...

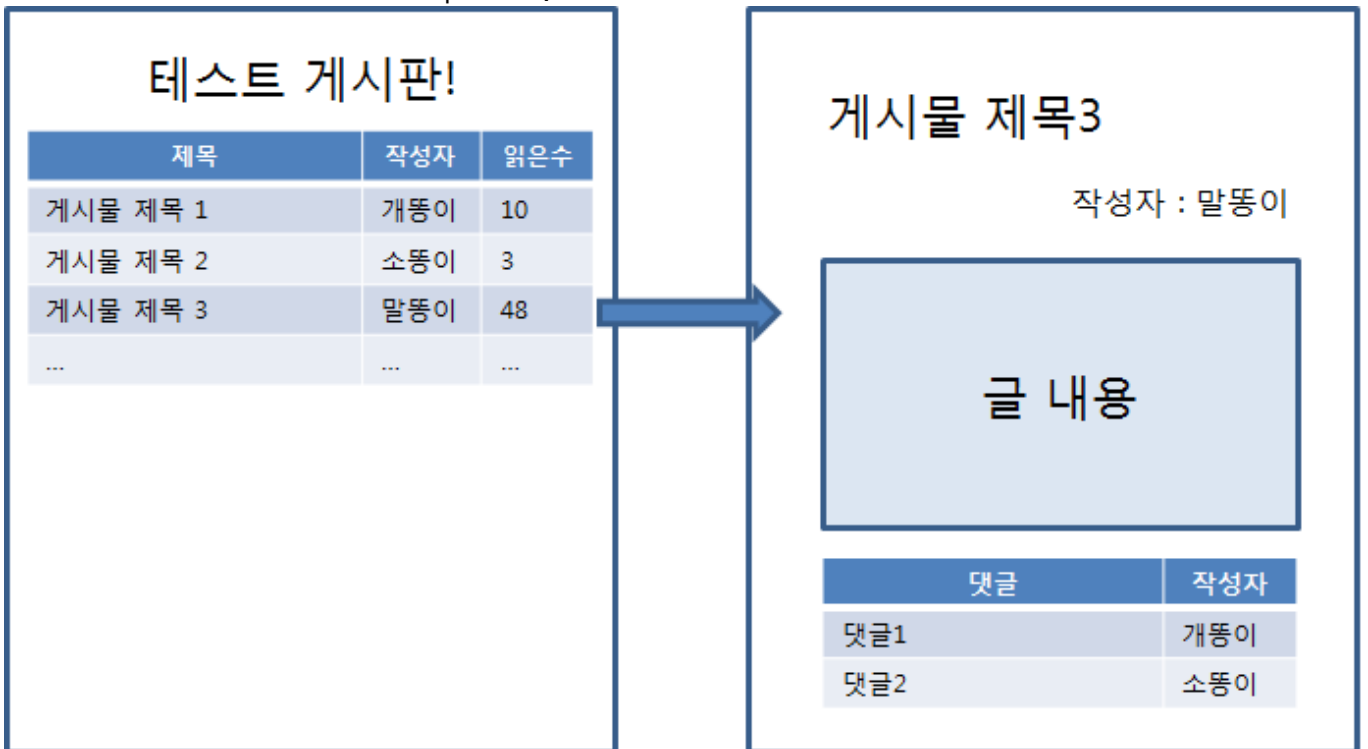
NoSQL

RDBMS

NoSQL



가

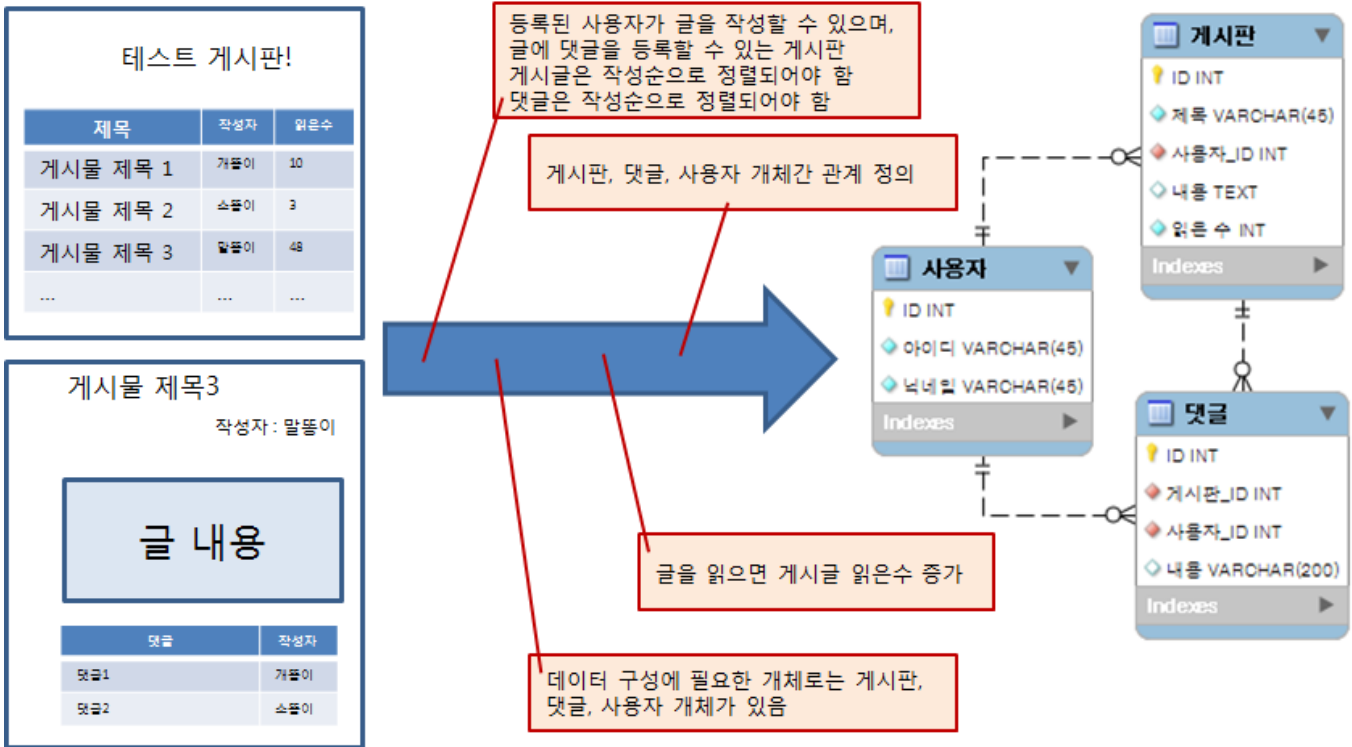


NoSQL

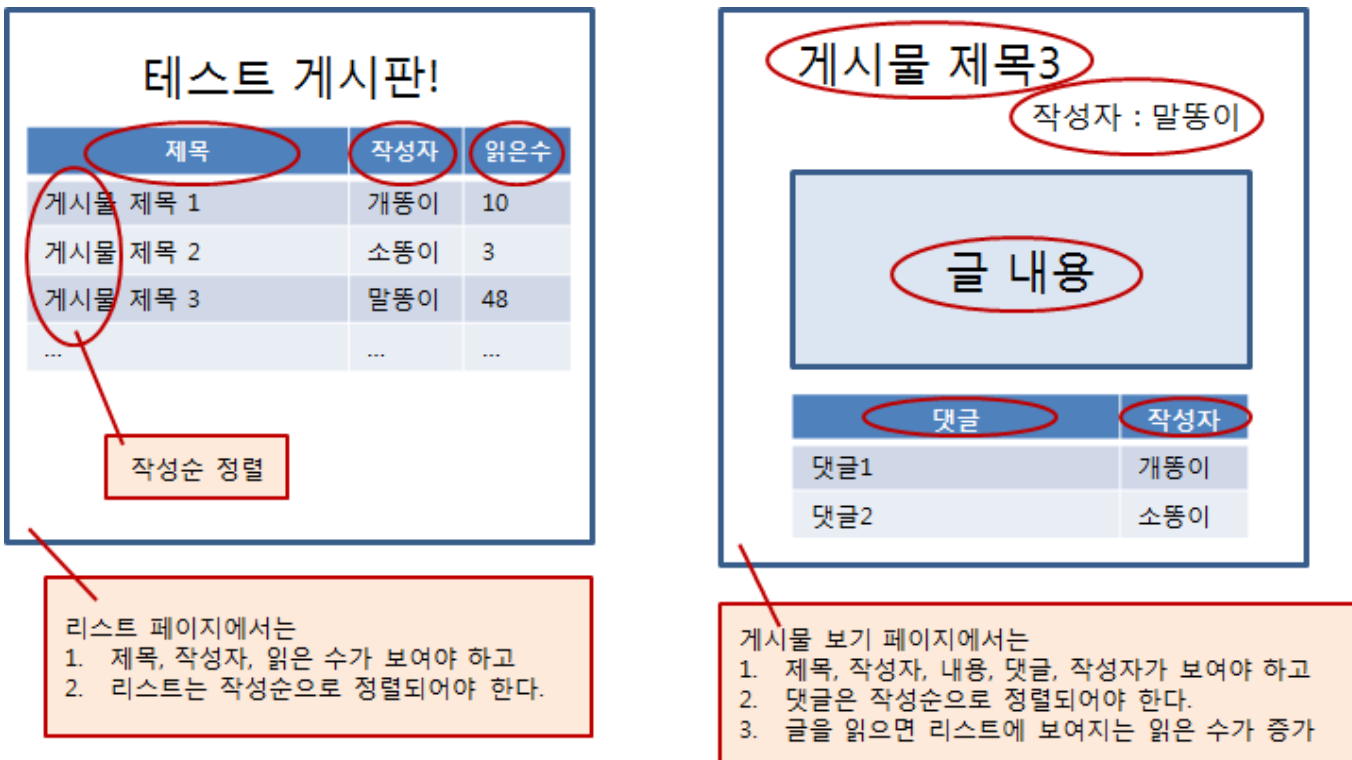
- RDBMS

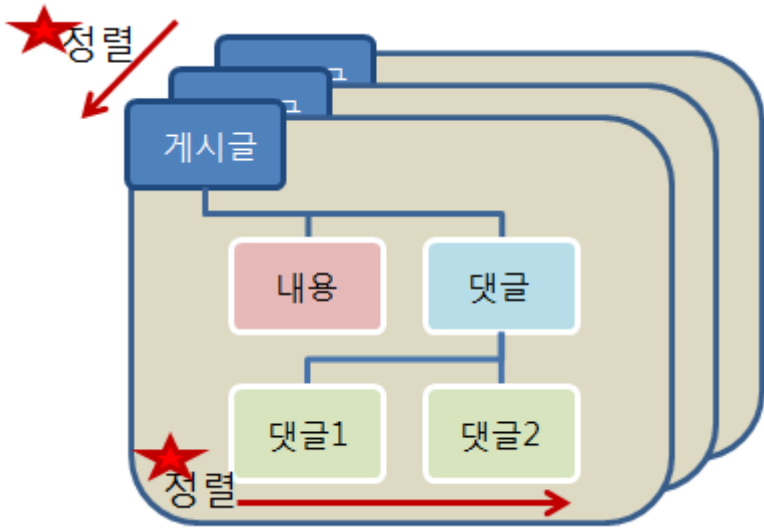
RDBMS NoSQL

- RDBMS



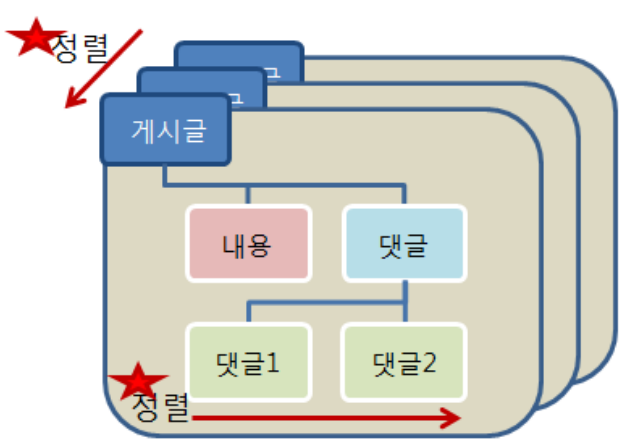
- NoSQL





- 1) NoSQL
 2) : = 1:1
 3) : = 1:N
 4)
 5) " " - RDBMS

NoSQL 가



쿼리 1
 SCAN 게시판 order by 작성순
 => [{제목, 작성자, 읽은수, 내용, [[댓글, 작성자]], ...}]
 GET 게시물 order by 댓글.작성순
 => {제목, 작성자, 읽은수, 내용, [[댓글, 작성자]]}

쿼리 2
 SCAN 게시판 order by 작성순
 => [{제목, 작성자, 읽은수, 내용, [댓글ID, ..], ...}]
 GET 게시물
 => {제목, 작성자, 읽은수, 내용, [댓글ID, ...]}
 GET 댓글
 => {댓글, 작성자}

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}]}

```

{
  "id": "게시판id",
  "제목": "게시판 제목",
  "작성자": "사용자아이디",
  "읽은수": 10,
  "내용": "게시판내용내용",
  [
    {
      "댓글": "댓글내용",
      "작성자": "댓글작성자",
    }, ...
  ]
}

```

Value는 Document type!!

2

쿼리 2

SCAN 게시판 order by 작성순

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

GET 게시글

⇒ {제목, 작성자, 읽은수, 내용, [댓글ID, ...]}

GET 댓글

⇒ {댓글, 작성자}

Key : 댓글ID (댓글 등록시간의 Reversed Timestamp)

Column Family : 댓글

댓글내용	작성자
------	-----

Key : 게시판ID (게시글 등록 시간의 Reversed Timestamp)

Column Family : 게시글

제목	작성자	읽은수	내용
----	-----	-----	----

Column Family : 댓글

댓글ID	댓글ID	댓글ID	댓글ID
------	------	------	------

Value는 Column Family type!!!

- Oriented

NoSQL

Document Oriented, Column Family

NoSQL

?

NoSQL

개발자 뉴스센터

"관계형 데이터베이스 MySQL 버리고, NoSQL로!" 주장

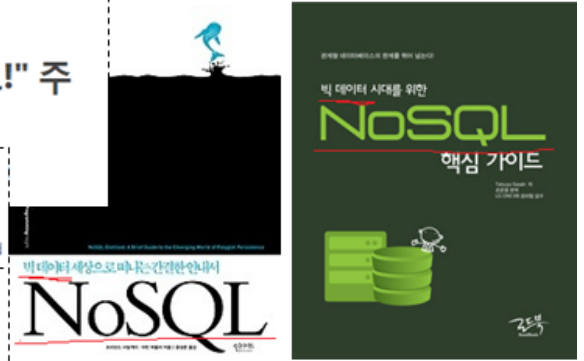
데이터센터

"SQL 반대?" 부상하는 반 데이터베이스 운동

2009.07.06

Eric Lai | Computerworld

관계형 데이터베이스 관리 시스템에 맞추려면 오브젝트 데이터를 억지로 구겨 넣을 수밖에 없다
- 존 트래비스 (SpringSource 의 대표 엔지니어)



빅데이터를 처리 = NoSQL?

“ NoSQL ! ”

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.

트윗 저장소로는 그냥 MySQL 쓰기로 했어요~
많은 데이터를 새로운 기술로 마이그레이션 하기 부담되기도 하고..

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

MySQL에 만족해요~



ㅅ ㅈ
@_@;;

가

NoSQL

?

“SQL이 어렵다고 느껴지면 NoSQL을 사용하지 말아야 한다!”
- Monty Widenius, the creator of MySQL

데이터 속성에 대해서 이해하지 못했다면
BigData라는 이유로 NoSQL을 함부로 사용하지 말아야 한다!
(적어도 다루려는 데이터가 필요로 하는 것이 ACID인지 BASE는 파악해야 함)



SQL, NoSQL, Data에 대해서 충분히 이해한 후에 사용한다!

()

- 1. RDBMS가 ACID NoSQL BASE
 - 2. Consistency, Availability, Partition Tolerance 가 가
 - 3. NoSQL Database Data Store
 - 4. NoSQL Key-Value . Value Key-Value, Column Family
Oriented, Document Oriented .
 - 5. RDBMS NoSQL .
 - 6. RDBMS NoSQL .
 - 7. NoSQL .
 - 8. SQL, NoSQL 가 Data
- !

From: <https://atl.kr/dokuwiki/> - AllThatLinux!

Permanent link: 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

Last update: 2017/02/02 11:02

