Not Only SQL 맛보고..

옛날부터 웹 어플리케이션을 구축했다면

관계형 DB를 사용했습니다.

관계형 DB

매우 높은 완성도를 자랑하는 기술
안 쓸 이유는 없다.
쓸만한 성능이 나온다.
공부를 많이 해야 한다.
복잡하다.
튜닝이 필요하다.
~~DBA가 밥 먹게 해 준다.~~

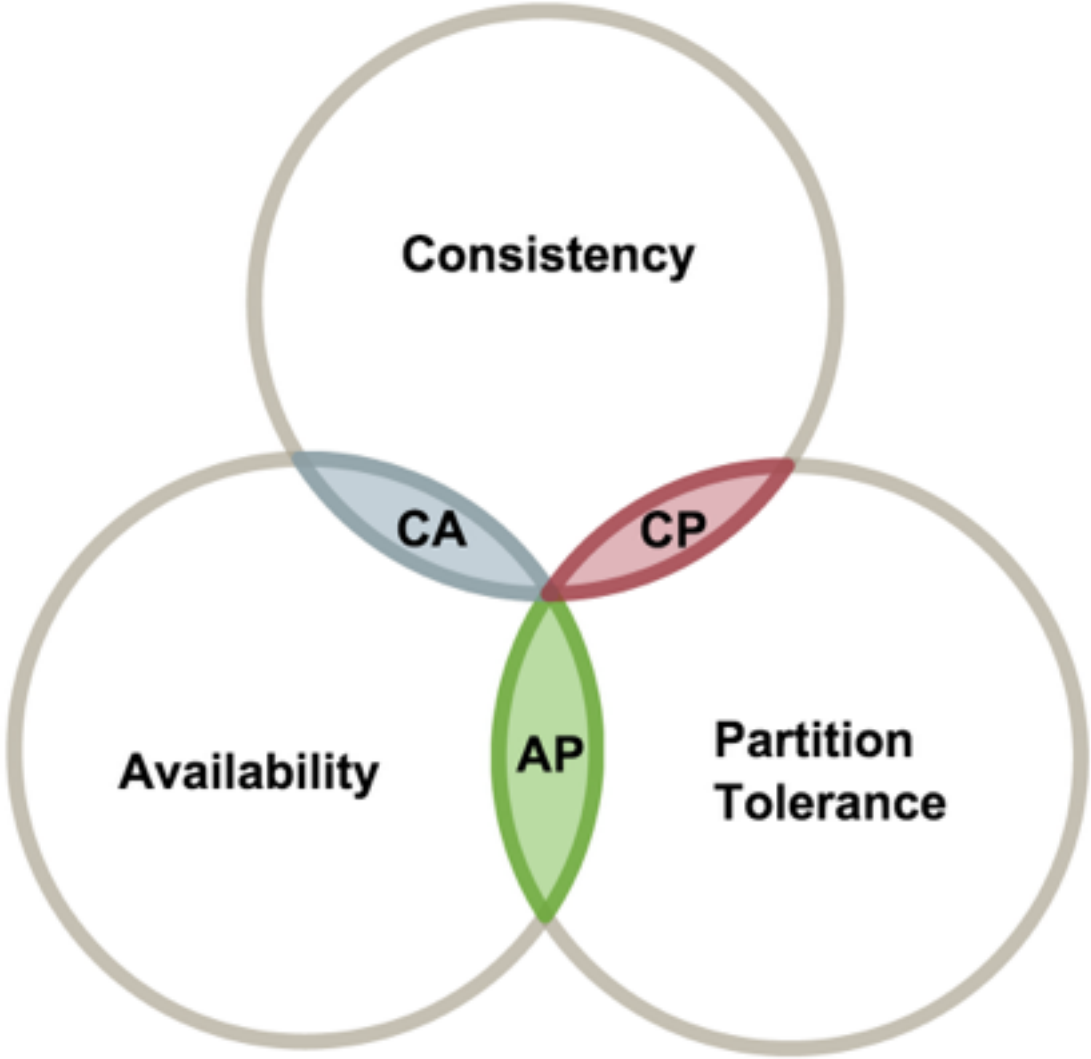
그럼 왜 NoSQL을 왜 사용하나?

왜 mongoDB를 사용하나? 와 유사한 질문

- 간단해서 ??

- 확장성이 좋아서 ??

CAP 이론



분산 시스템이 가지는 세가지 특성
(이중 두가지만 취할수 있다. 셋을 동시에 충족시키는 것은 불가능)

C (Consistency)
각각의 사용자가 같은 데이터를 볼수 있다
= 모든 노드가 같은 시간에 같은 데이터를 보여줘야 한다.

A (Availability)
모든 사용자가 항상 쓸 수있다.
= 몇몇 노드가 다운 되어도 다른 노드에게 영향을 주지 않는다.

P (Partition Tolerance)
물리적으로 분리된 분산 환경에서 작동한다
= 일부 메세지를 손실하더라도 시스템은 정상동작해야 한다

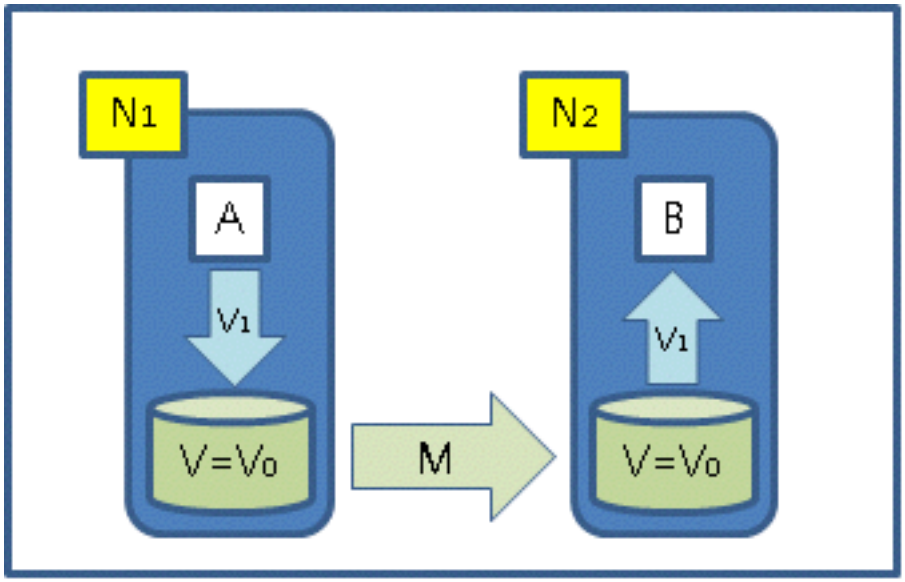
재정리해보자면 C= 동일성, A=독립성 , P = 생존성

CA (동일성 + 독립성)
시스템이 죽을지언정 메시지 손실은 방지하는 강한 신뢰형
전통적인 RDBMS가 여기에 해당한다. 트랜잭션이 필요한 경우 필수적.

CP (동일성 + 생존성)
모든 노드가 함께 퍼포먼스를 내야하는 성능형
구글의 BigTable, HBase, MongoDB, Redis

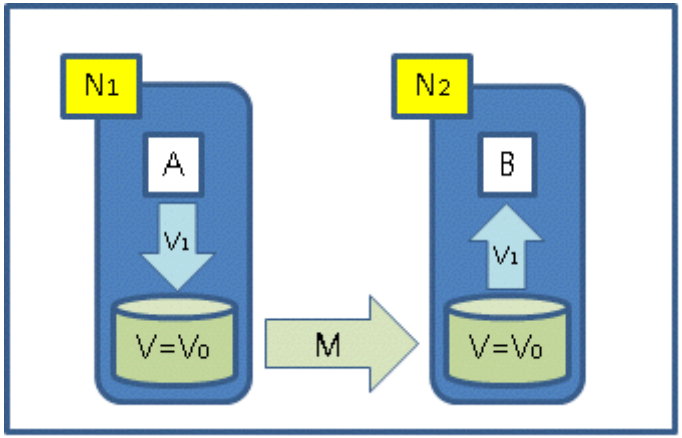
AP (독립성 + 생존성)
비동기화된 서비스 스토어에 적합
Dynamo, Tokyo Cabinet, Apache Cassandra, CouchDB, Oracle Coherence 등

CAP 이론 이해



- 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.

C가 필요한 경우



* C 모든 노드가 같은 시간에 같은 데이터를 보여줘야 한다.
* A 몇몇 노드가 다운 되어도 다른 노드에게 영향을 주지 않는다.
* P 일부 메세지를 손실하더라도 시스템은 정상동작해야 한다

- A가 V1이라고 썼기 때문에 B는 V1이라고 읽어야만 한다.
- A의 쓰기 동작은 M이 복구되기 전까지는 성공할 수 없다

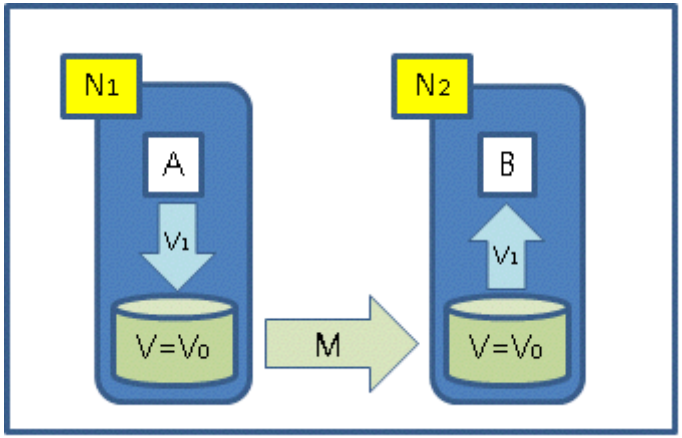
- CP (mongo)

M이 복구되기 전까지는 A의 Write는 block되거나 실패해야 한다. = Availability가 없음

- CA (rdbms - oracle / mysql)

M이 문제가 생길 수 없도록 구성 = Partition-Tolerance가 필요 없다 = 같은 곳을 바라보자!

A가 필요한 경우



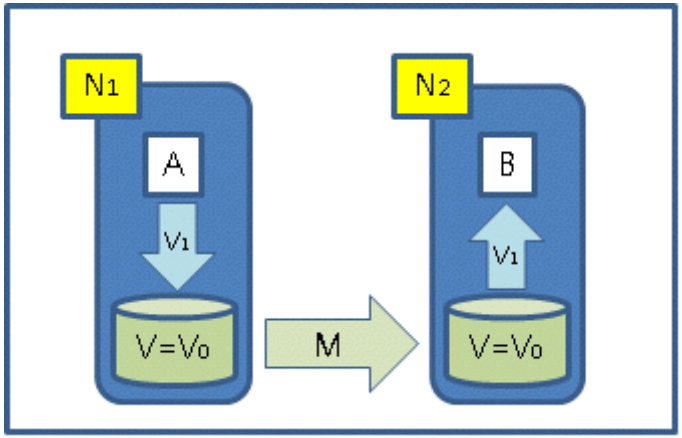
* C 모든 노드가 같은 시간에 같은 데이터를 보여줘야 한다.
* A 몇몇 노드가 다운 되어도 다른 노드에게 영향을 주지 않는다.
* P 일부 메세지를 손실하더라도 시스템은 정상동작해야 한다

- 어떤 경우에도 서비스가 Unavailable하면 안된다.

- AP (casandra , dynamo, riak)
A와 B가 꼭 동일한 데이터를 읽을 필요는 없음

- CA (rdbms - oracle / mysql)
M이 문제가 생길 수 없도록 구성 = Partition-Tolerance가 필요 없다 = 같은 곳을 바라보자!

P가 필요한 경우



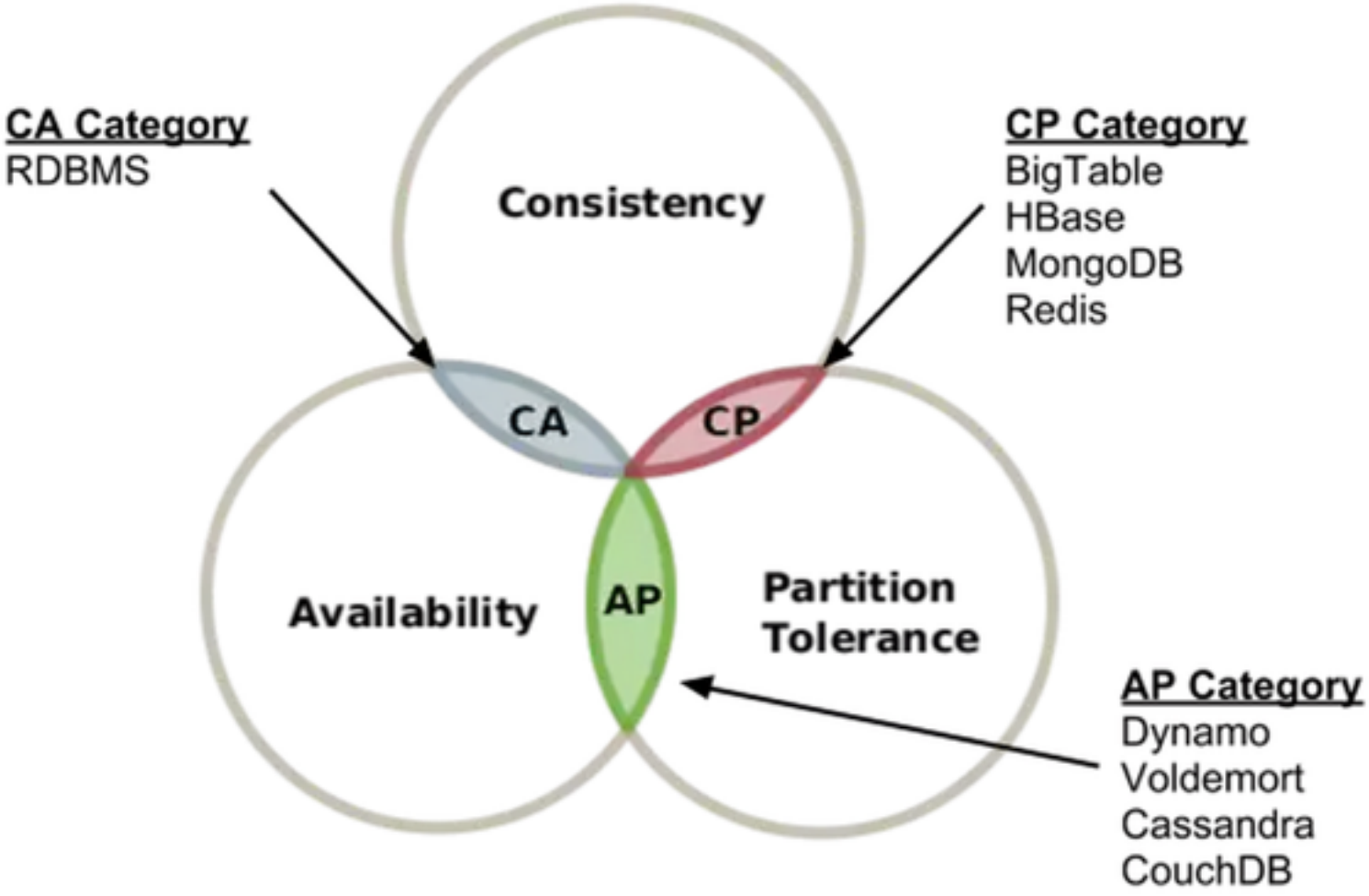
* C 모든 노드가 같은 시간에 같은 데이터를 보여줘야 한다.
* A 몇몇 노드가 다운 되어도 다른 노드에게 영향을 주지 않는다.
* P 일부 메세지를 손실하더라도 시스템은 정상동작해야 한다

- 메시지 전달 과정(M)에서 문제가 생기더라도 시스템에 영향이 가서는 안된다.

- AP (casandra , dynamo, riak)
A와 B가 꼭 동일한 데이터를 읽을 필요는 없음

- CP (mongo)
A의 쓰기 동작은 M이 복구되기를 기다린다. = 그동안 쓰기 서비스 불가능 = Availability가 없음 = CP

CAP 이론


CA Category
RDBMS
Consistency
CP Category
BigTable
HBase
MongoDB
Redis
CA
CP
Availability
AP
Partition
Tolerance
AP Category
Dynamo
Voldemort
Cassandra
CouchDB

Data Model의 종류

Key / Value

memcached, Dynamo

Relational

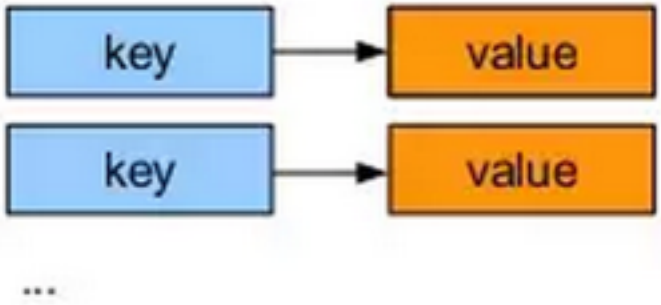
MySQL, Oracel

Document Oriented

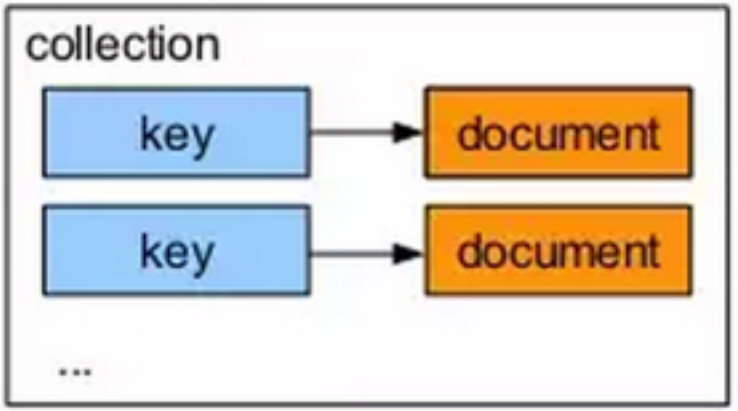
MongoDB, CouchDB, JSON stores

데이터 저장 방식 타입

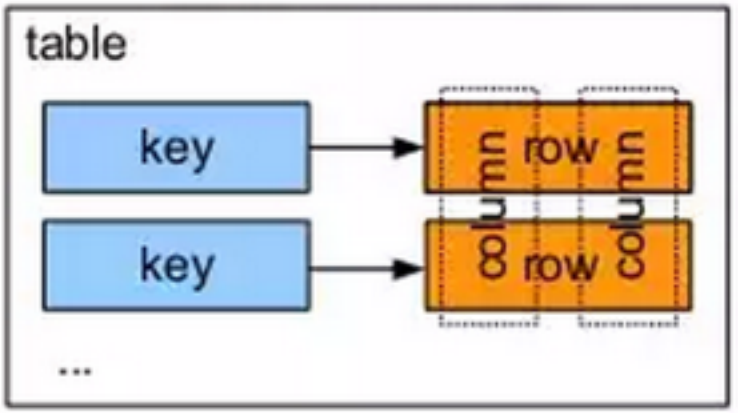
Key / value stores (opaque / typed)



Document stores (non-shaped / shaped)



Relational databases



데이터 베이스 분할..

수직 분할

- 팩트 테이블의 기본키가 중복
- 전체적인 공간은 더 많이 차지
- 특성에 따른 컬럼 또는 자주 사용하지 않는 컬럼을 분할 함으로써 크기를 줄이고 검색 비용이 줄어 든다

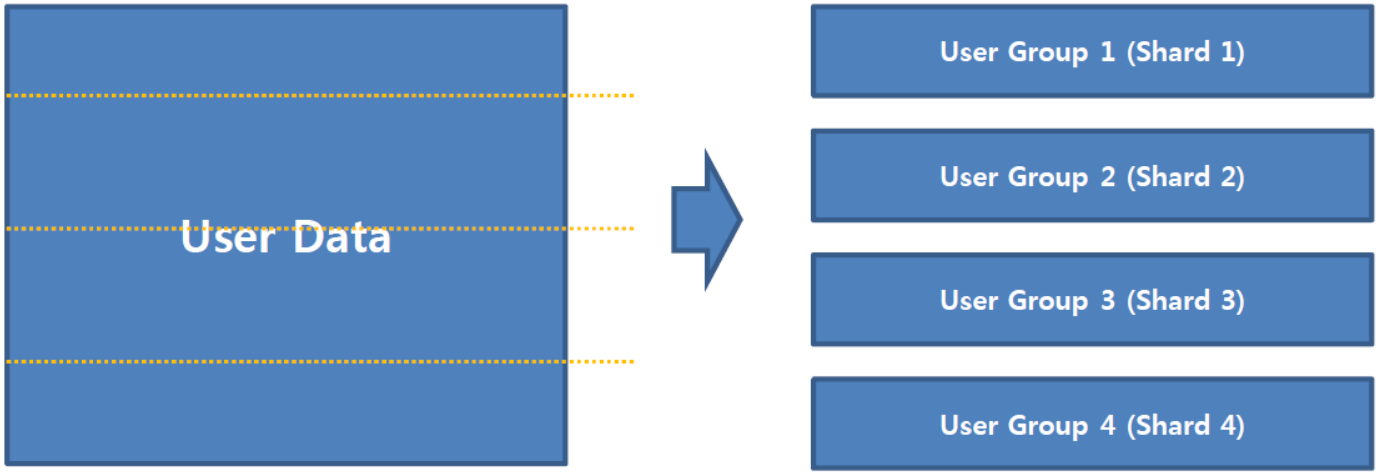
기간키	매장키	제품키	매출액	매출원가	운송비
20120101	110	1101	27,800	12,100	780
20120301	120	1200	39,000	15,500	810
20120401	210	100	27,800	13,000	810
20120405	120	1200	39,000	14,500	930
20120508	210	1301	30,300	14,300	840

수평 분할

- 하나 이상의 비즈니스 차원에 따라 행(ROW) 수준에서 팩트 테이블을 분할.
- 분할시 안정적인 애트리뷰트(기간, 원, 분기)로 분할.

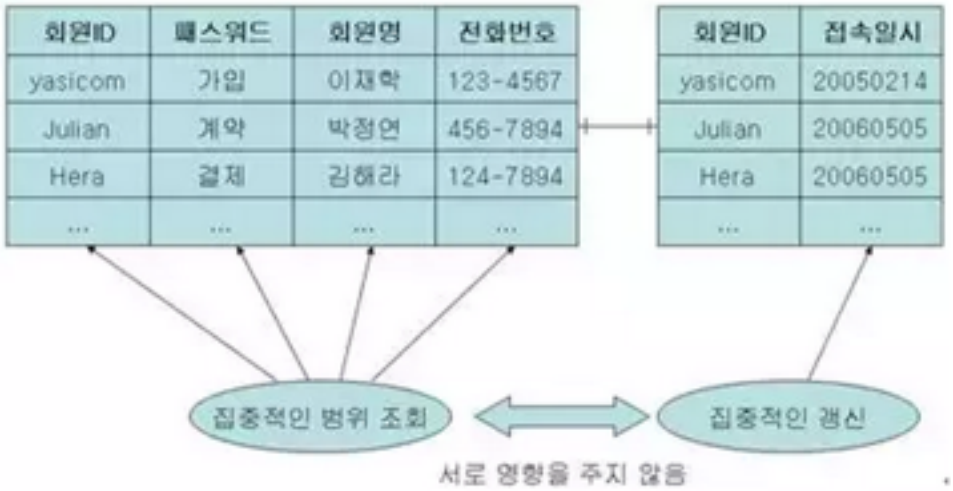
데이터 베이스 분할..

- 수평 분할(horizontal partitioning = Sharding) 은 하나의 테이블의 각 행을 다른 테이블에 분산시키는 것이다. 예를 들어 방대한 고객 데이터 테이블을 성별에 따라 '남녀'로 나누어 CustomerMen과 CustomerWomen 두 개의 테이블로 분할한다. 테이블은 2개로 분할되지만, 모든 고객을 나타내기 위해 양자를 결합한 뷰를 생성하면 된다.

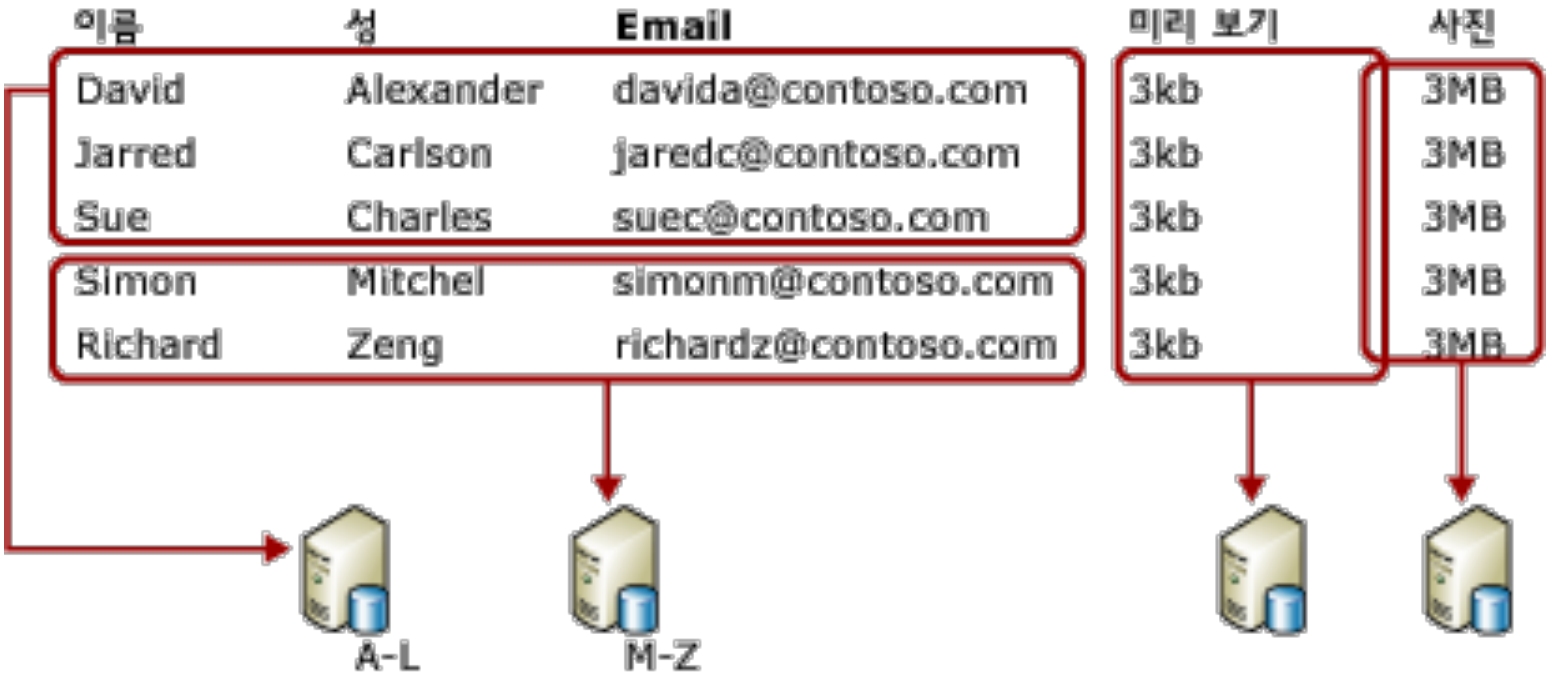


데이터 베이스 분할..

- 수직 분할(vertical partitioning)은 테이블의 일부 열을 빼내는 형태로 분할한다. 관계의 정규화는 본질적으로 수직 분할에 관련된 과정이다.
- 사용 빈도가 높은 데이터에만 액세스할 경우 성능이 향상된다. 예를 들어, 뉴스를 서비스할 때, 고객들은 최근의 데이터를 가장 많이 조회할 것이다. 이 경우 1개월 전의 데이터를 다른 테이블에 두면, 훨씬 효율적으로 검색할 수 있을 것이다.

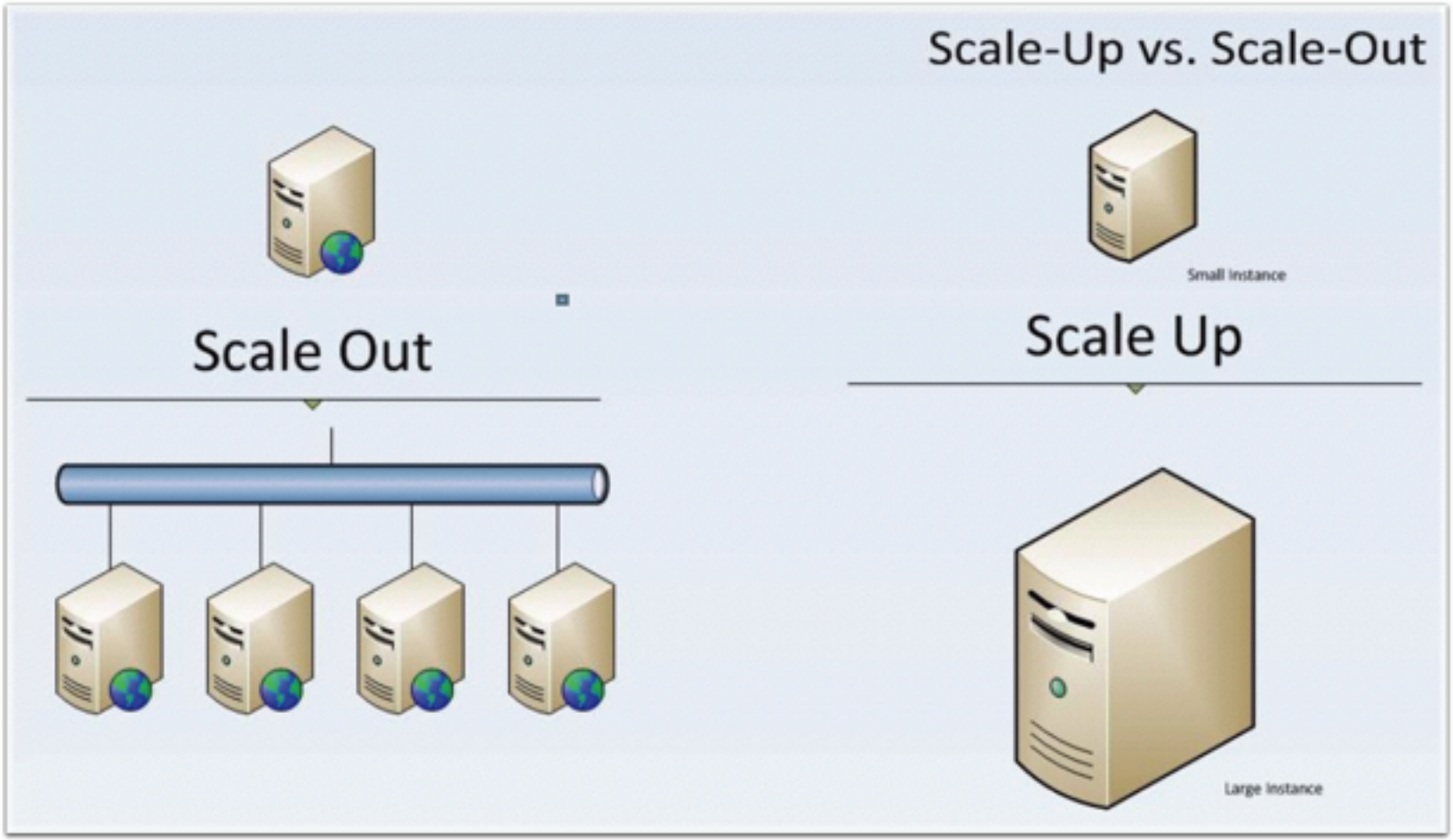


실제 어떻게 쓰나?

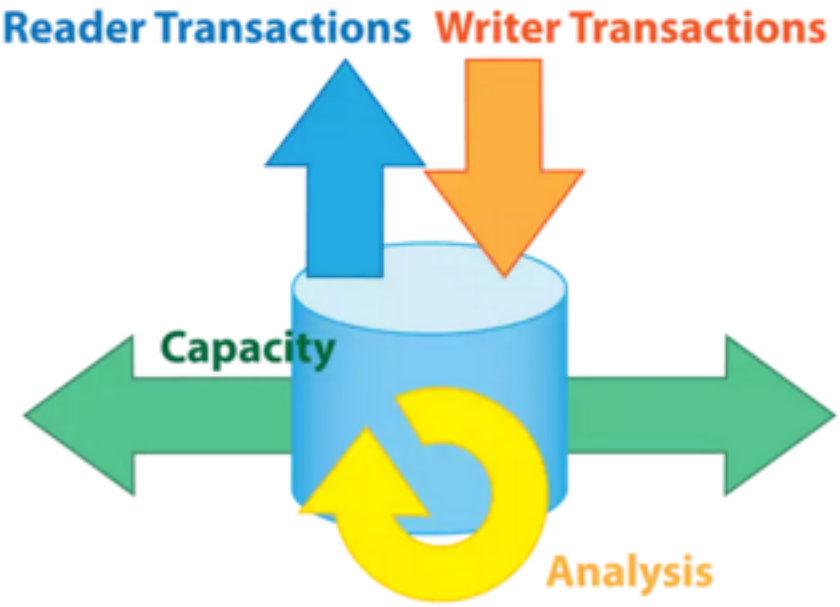

이름
성
Email
미리 보기
사진
David
Alexander
davida@contoso.com
3kb
3MB
Jarred
Carlson
jaredc@contoso.com
3kb
3MB
Sue
Charles
suec@contoso.com
3kb
3MB
Simon
Mitchel
simonm@contoso.com
3kb
3MB
Richard
Zeng
richardz@contoso.com
3kb
3MB
A-L
M-Z

NoSQL을 다루기 전 알아야 할 개념들

Scale Out vs Scale Up

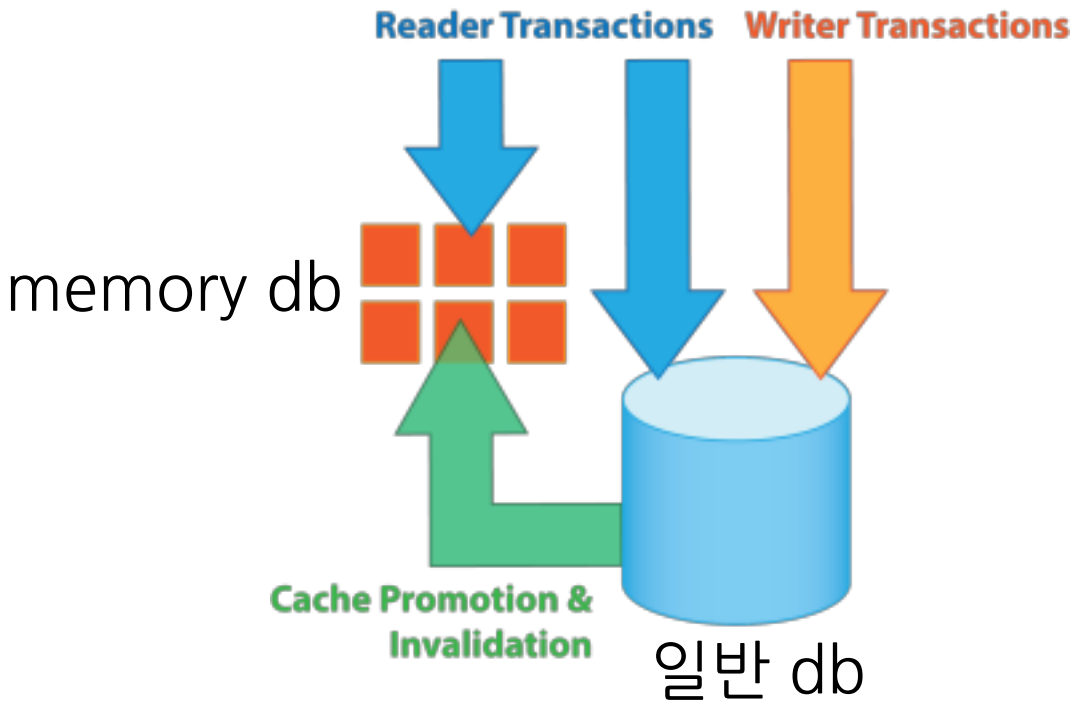


Transaction, Capacity, Analysis 란?

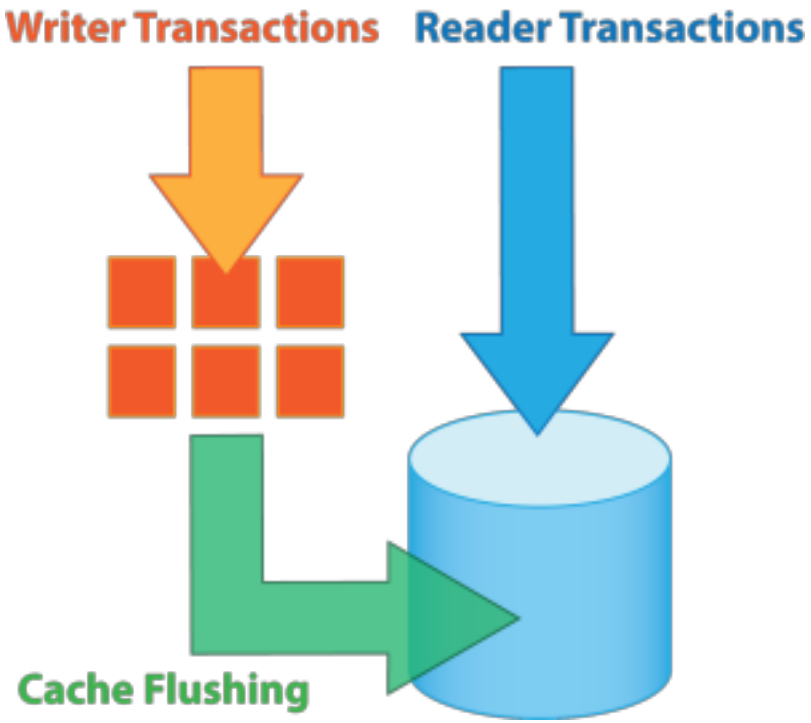


Capacity : 저렴한 가격으로 많은 데이터를 저장할수 있나?
(비용과 결부된 문제)

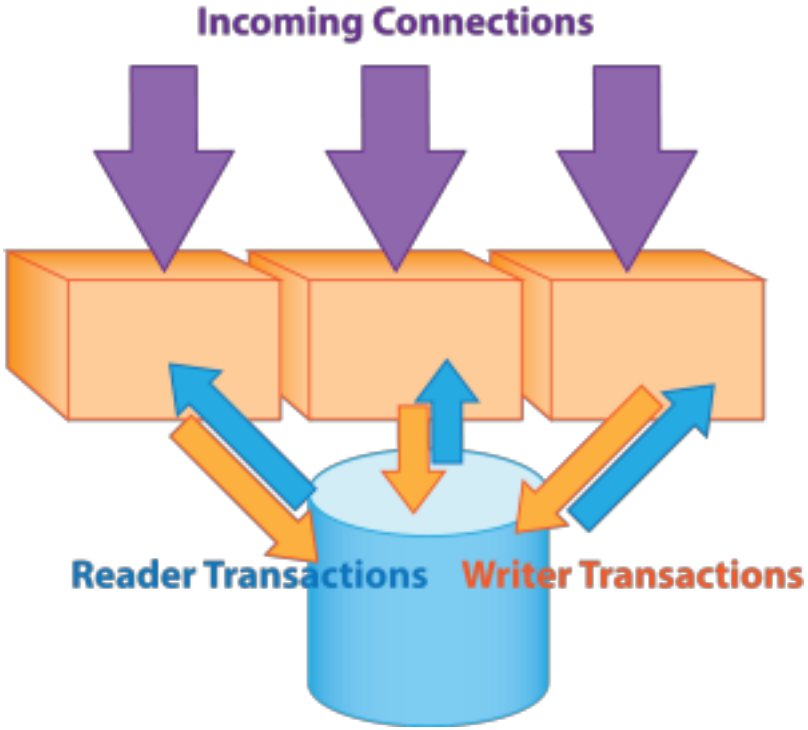
Read Caching



Write Coalescing

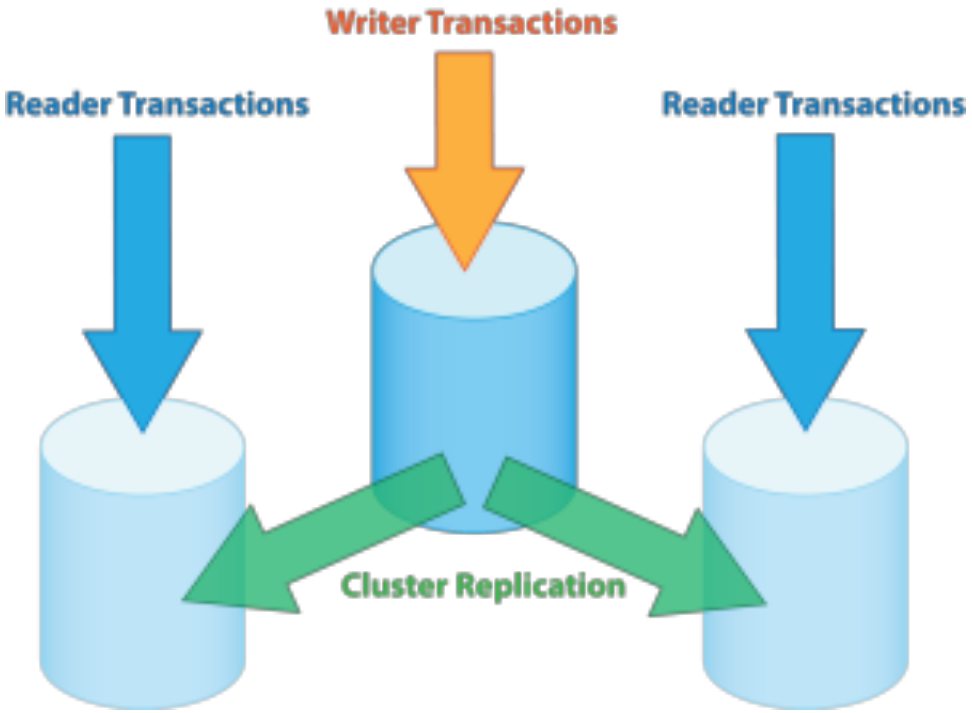


Connection Scaling

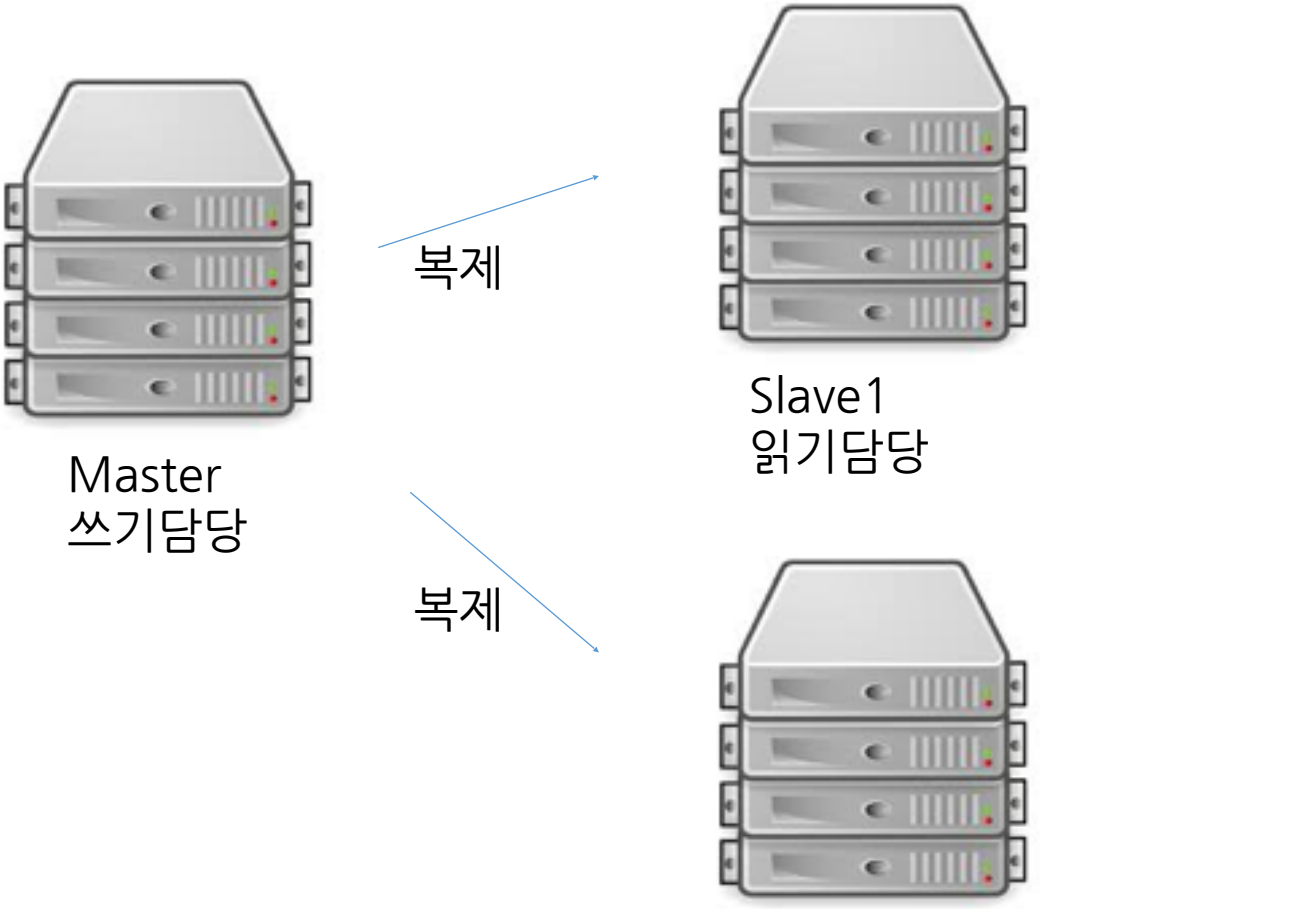


sharding이 필요하다

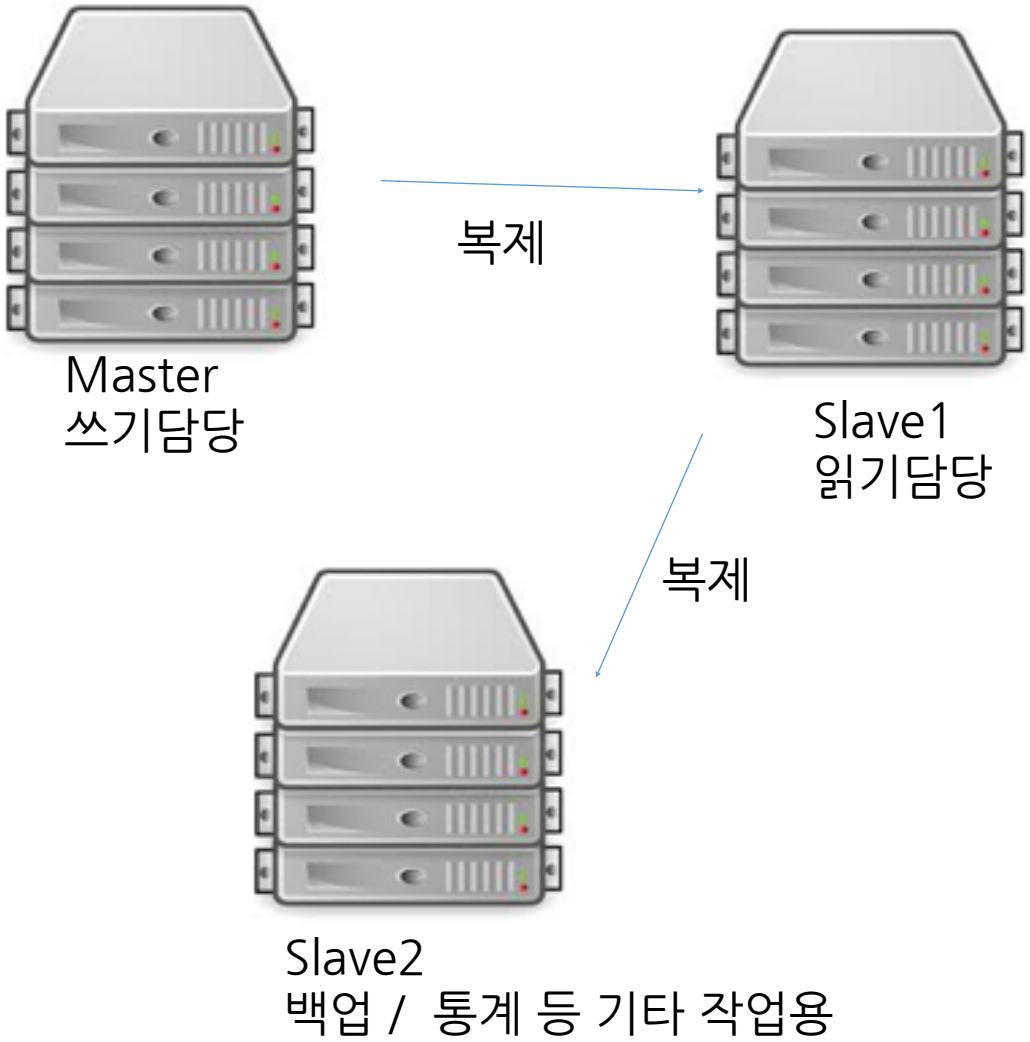
Master-Slave Replication



일반적인 형태의 복제 구성



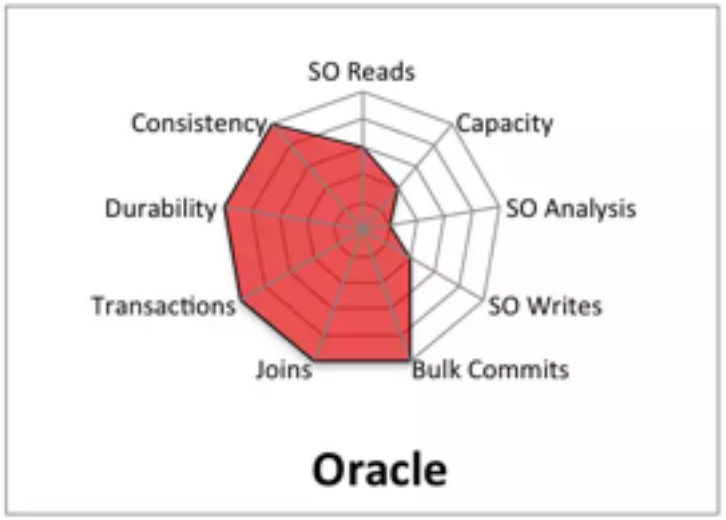
또 다른 구성 방법



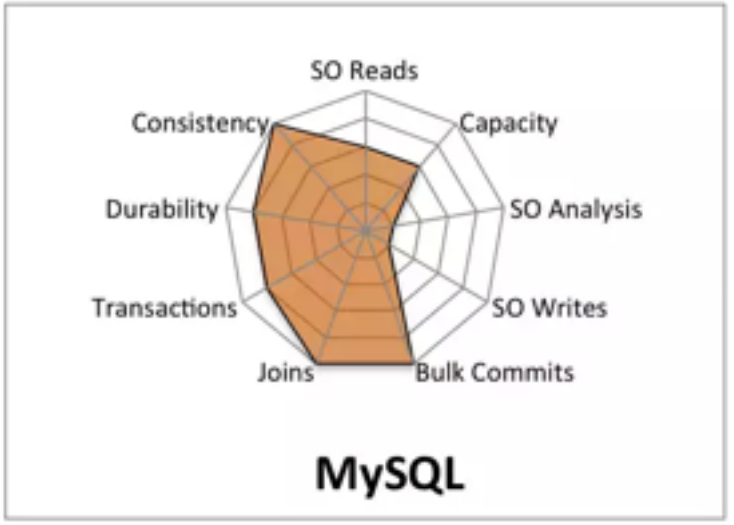
NoSQL 장단점 분석

용어 정리

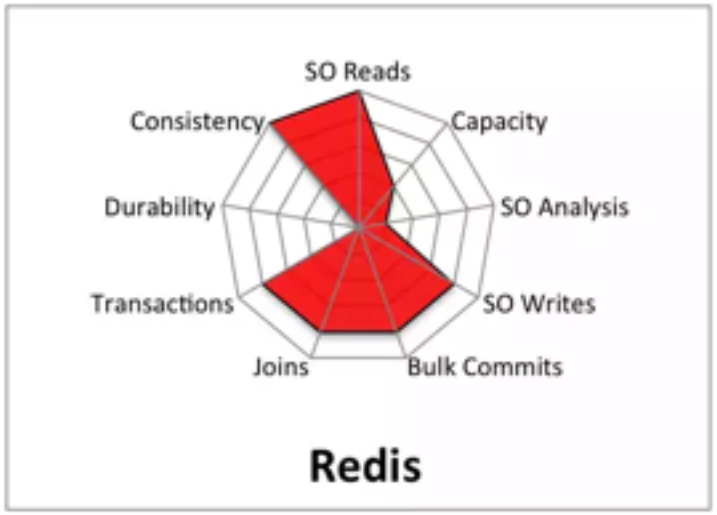
Scale Out Reads : 개발자가 별 공수없이 가능하냐?
(직접 개발자가 partitioning ,sharding을 구현해야 하면 없는 걸로)
Capacity : 많은 데이터 저장 가능하냐? 비용이 많이들면 없는 걸로
Scale Out Analysis : 개발자가 별 공수없이 가능하냐?
Scale Out Writes : 개발자가 별 공수없이 가능하냐?
Bulk Commits
Joins
Transactions
Durability
Consistency



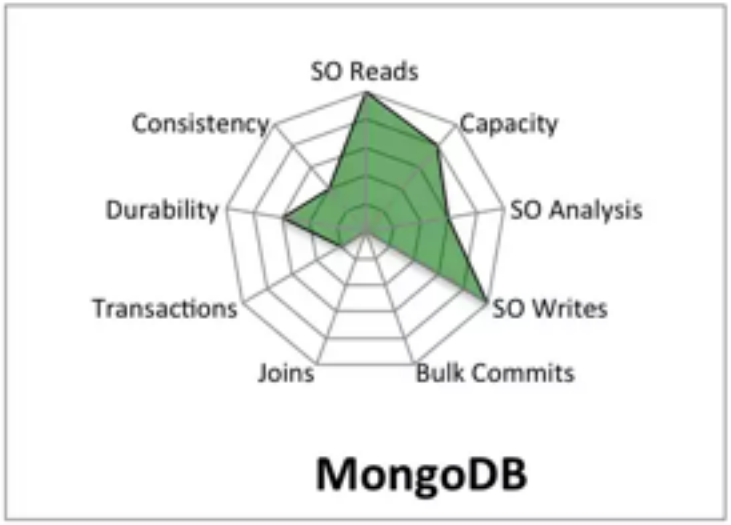
License: Commercial
Scaling Techniques: Master-Slave Replication, Connection Scaling
Database Features: Joins, Bulk Commits, strong ACID



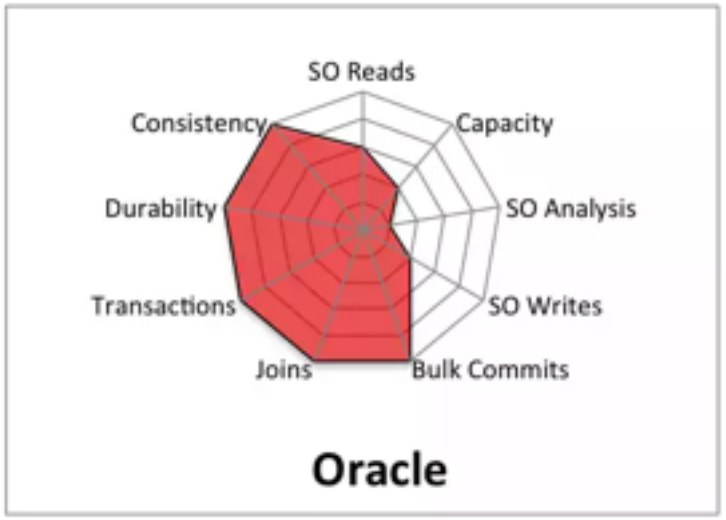
Licence: Open Source
Scaling Techniques: Master-Slave Replication
Database Features: Joins, Bulk Commits, strong ACID



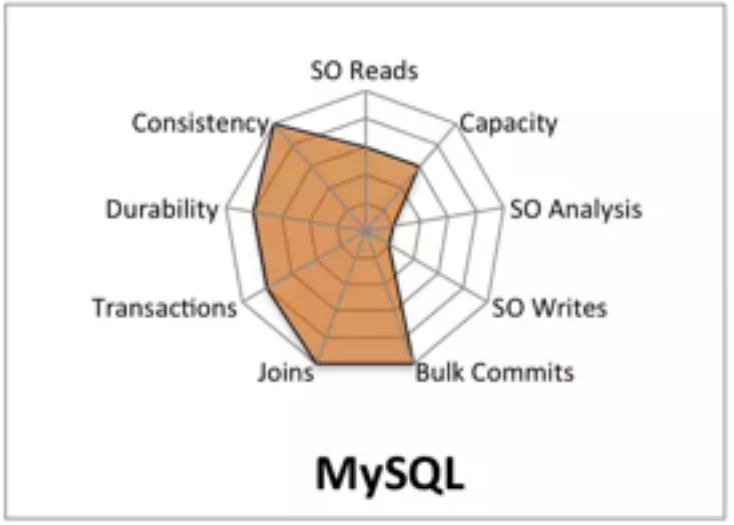
Licence: Open Source
Scaling Techniques: Read-Caching, Write Coalescing, Master-Slave Replication
Database Features: In-memory store, Joins, Bulk Commits



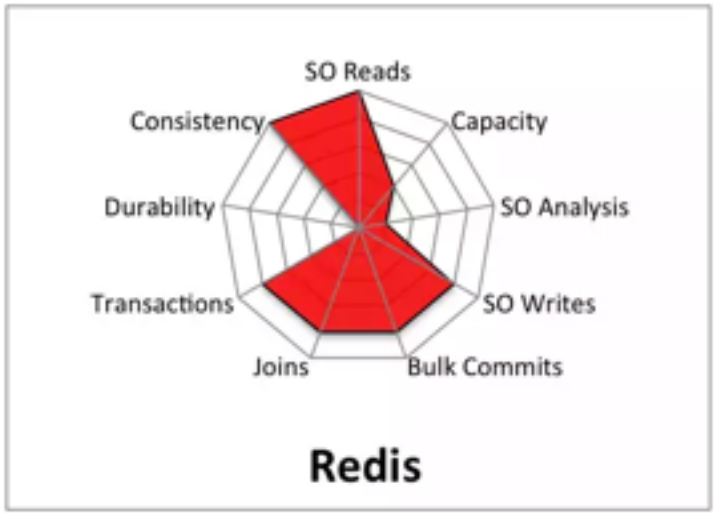
Licence: Open Source
Scaling Techniques: Distributed Data Store, Write Coalescing
Database Features: Weak support for Joins & ACID



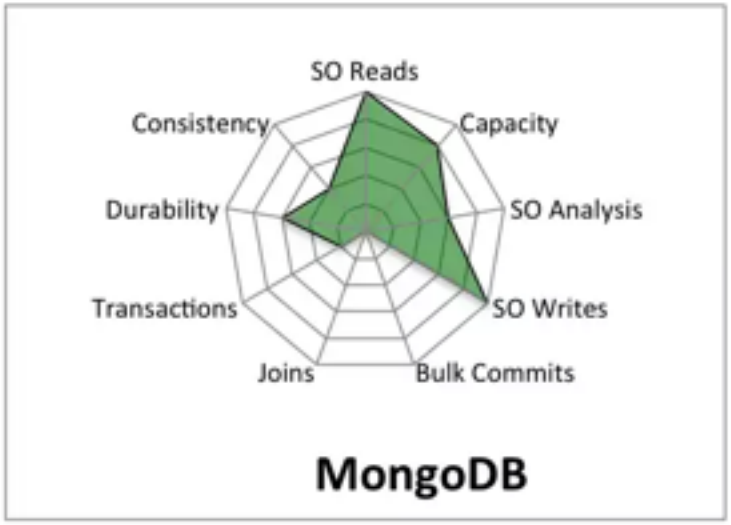
License: Commercial
Scaling Techniques: Master-Slave Replication, Connection Scaling
Database Features: Joins, Bulk Commits, strong ACID



Licence: Open Source
Scaling Techniques: Master-Slave Replication
Database Features: Joins, Bulk Commits, strong ACID



Licence: Open Source
Scaling Techniques: Read-Caching, Write Coalescing, Master-Slave Replication
Database Features: In-memory store, Joins, Bulk Commits



Licence: Open Source
Scaling Techniques: Distributed Data Store, Write Coalescing
Database Features: Weak support for Joins & ACID

MongoDB

10gen이라는 회사에서 만들었습니다.
클라우드 서비스를 SaaS로 제공하고 싶었는데 망했고
사람들은 그 중에서 DB만 쓰고 싶어했습니다.

그래서 남은 DB가 mongoDB가 되었습니다.

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document or BSON document
column	field
index	index
table joins	embedded documents and linking
primary key Specify any unique column or column combination as primary key.	primary key In MongoDB, the primary key is automatically set to the _id field.
aggregation (e.g. group by)	aggregation pipeline See the SQL to Aggregation Mapping Chart.

mongoDB의 특징

- document data model
- 스키마가 없다 (삽입시 생성된다..)
- json / bson
- 애드혹 쿼리 및 세컨더리 인덱스 지원
- 복제 / 샤딩 지원
- 하나의 마스터가 존재 cf. Casandra
- 일부 원자적 연산 지원 (트랜잭션 X)

Document Data Model

```
{ _id: ObjectID('5d41402abc4b2a76b9719d911017c592'),
        title: '실종 슈퍼마리오 4-1 리커버 발견, 몸안의 초록 버섯도 일치',
        url: 'http://mario.game/0401',
        author: 'kuppa',
        vote_count: 7777,
        tags: ['마리오','쿠파', '만우절', '사망'],
        image: {
                url: 'http://naver.com/logo1.png',
                caption: '로고',
                width: 480
        },
        comment: [
                { user: '루이지',
                text: '마리오 본인도 인정, 논란 그만했으면..' },
                {user: '모그레인',
                text: '일어나라, 나의 쿠파여~'
                }
        ]
}
```

Document
도큐먼트는 { 로 시작하고 } 로 끝난다.
속성의 이름 : 값의 집합을 저장한다.
속성은 , 로 분리한다.
값에는 배열(리스트)가 올 수 있다.
값에는 다른 도큐먼트가 올 수 있다.

JSON

모든 도큐먼트는 JSON 형식으로 표현됨

JSON: **J**ava**S**cript **O**bject **N**otation.

JSON is a **syntax** for storing and exchanging data.

JSON is an **easier to use** alternative to XML.

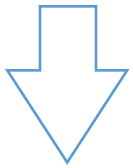
This JSON syntax defines an employees object, with an array of 3 employee records (objects):

JSON Example

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

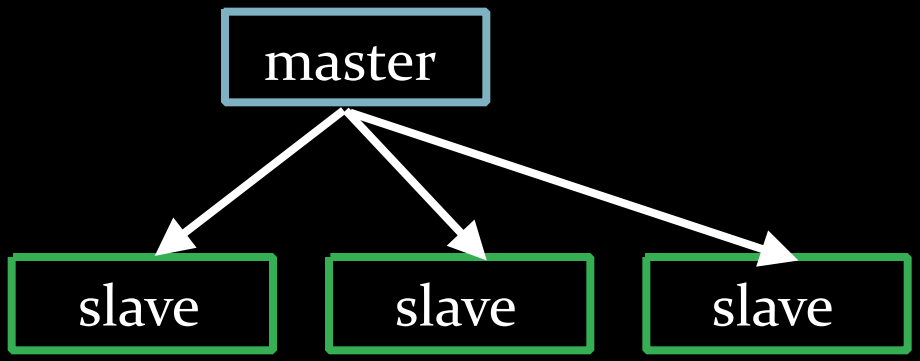
내부적으로 저장될때는 **BSON**으로 됩니다.

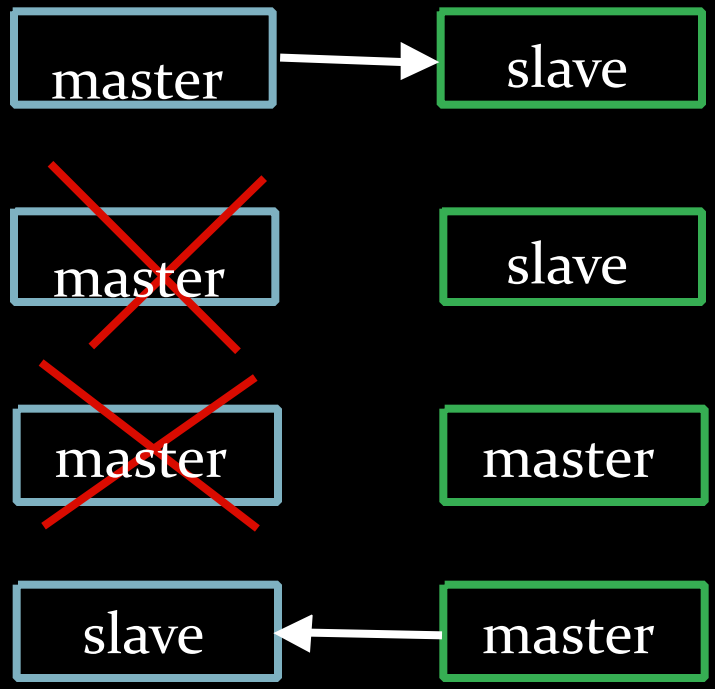
```
{“hello”:    “world”}
```



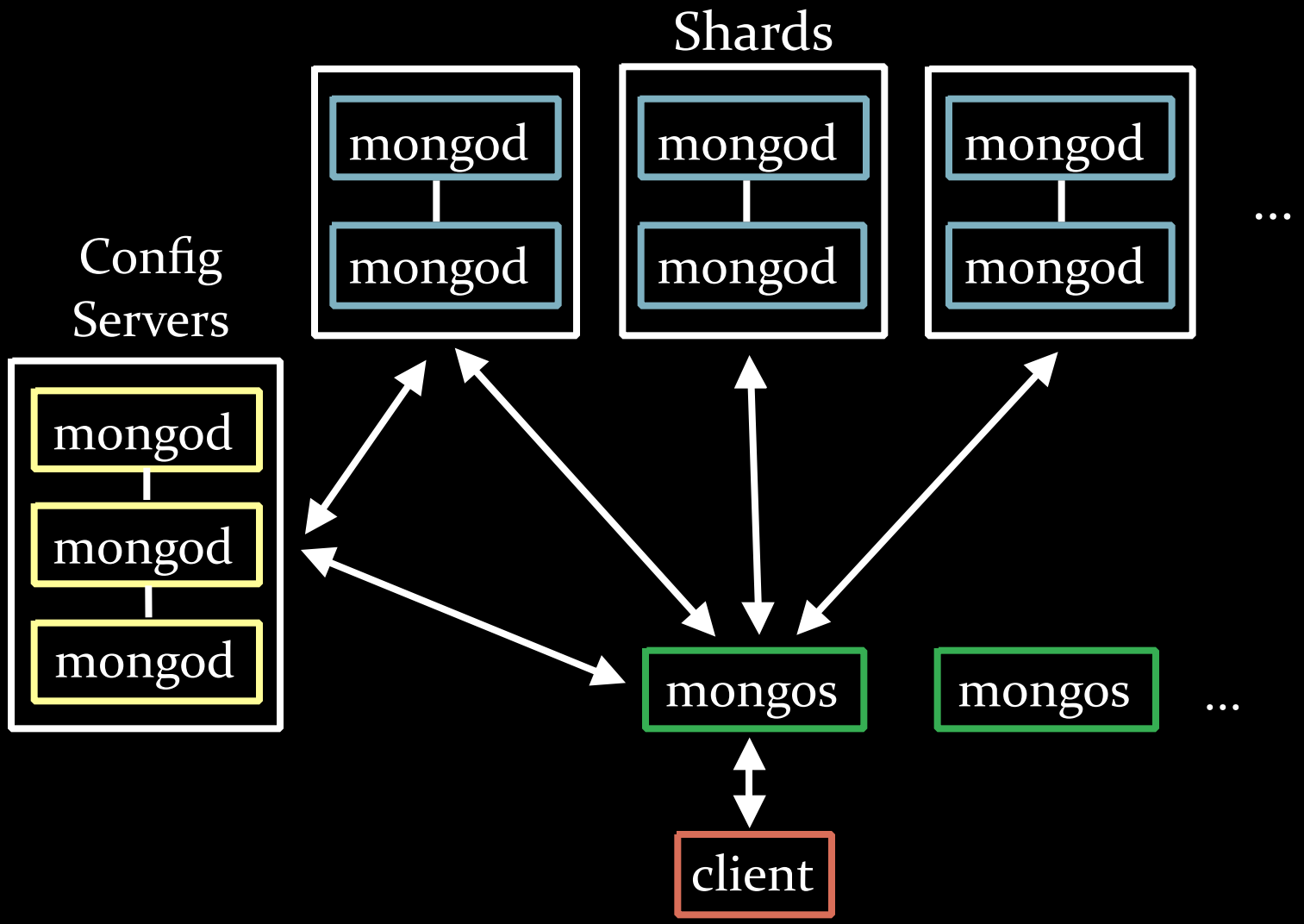
```
\x16\x00\x00\x00\x02hello
\x00\x06\x00\x00\x00world
\x00\x00
```

Replication





Auto-sharding



일단 설치부터 (ubuntu)

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install mongodb
```

참고:

http://blog.hibrainapps.net/145 (node.js 설치 우분투)

http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/

일단 설치부터 (mac)

```
$ brew update
$ brew install mongodb  (brew upgrade mongodb – 이미 설치한 분)


$ mongod –config /usr/local/etc/mongod.conf –fork   (백그라운드로 돌린다)

(서비스 등록)
$ launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mongodb.plist
```

참고:

http://www.mongodbspain.com/en/2014/11/06/install-mongodb-on-mac-os-x-yosemite/

http://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/

이미 설치된거 내릴때.

http://stackoverflow.com/questions/8495293/whats-a-clean-way-to-stop-mongod-on-mac-os-x

mongoDB가 사용되는 곳

- 애자일 개발
  - MySQL + ORM + 튜닝 → mongoDB
  - MySQL + memcahced → mongoDB
- 분석과 로깅
- 웹 어플리케이션

mongoDB 배우기

- http://www.mongodb.org/
- https://university.mongodb.com/

- 이 두 사이트 만으로 충분하지는.. 않지만 좋은 시작점 :)

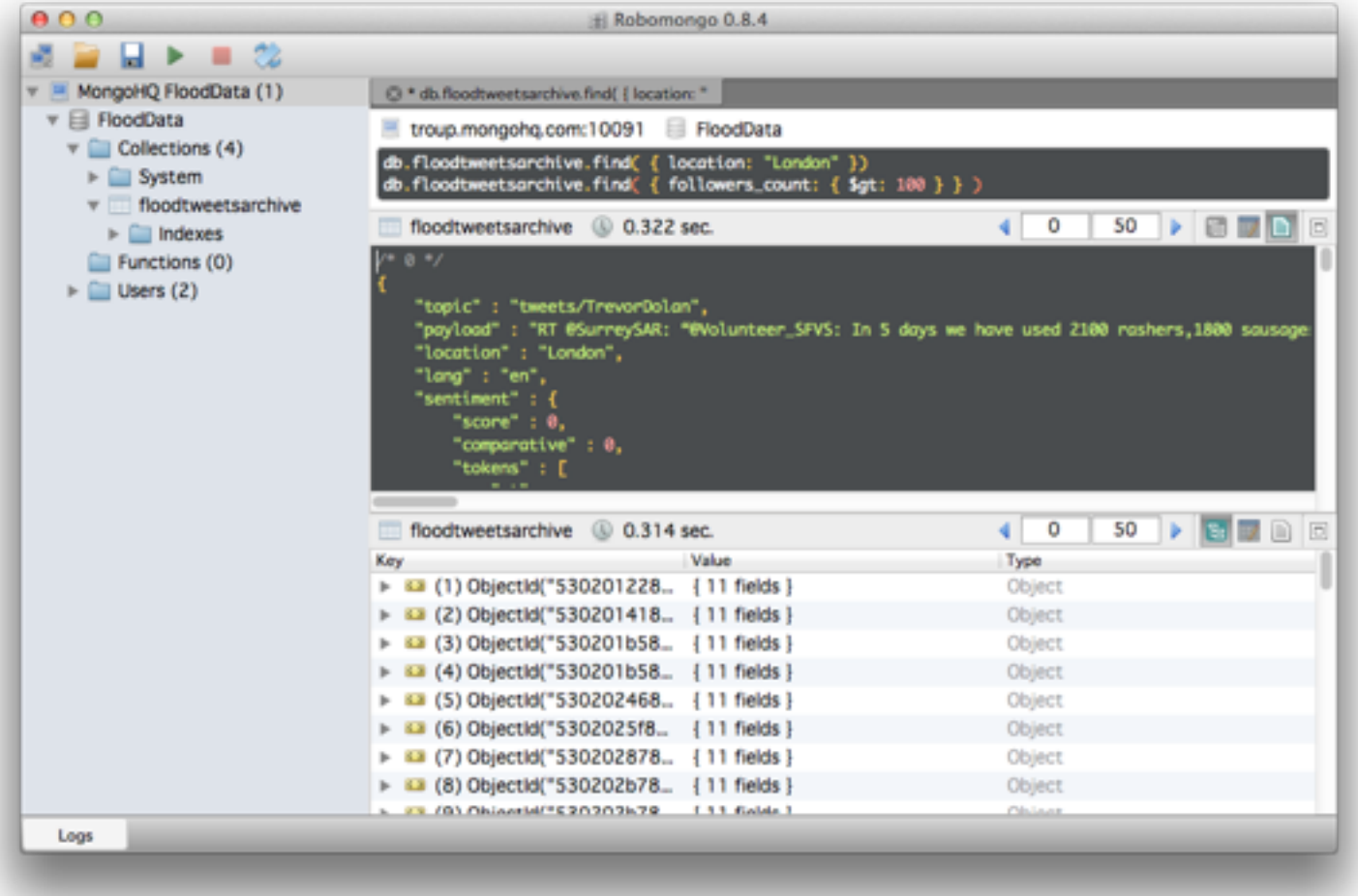
mongoDB shell

자바스크립트 기반 셸
mongoDB 기본기능 + 자바스크립트 사용가능

```
$ mongo
>
```

Robomongo

- 그외 여러 Admin UI는 http://docs.mongodb.org/ecosystem/tools/administration-interfaces/



CRUD 튜토리얼

쿼리문

- http://bit.ly/1ljbTxx (이 링크를 따라가며 실습을 해보아야)
- http://docs.mongodb.org/manual/reference/sql-comparison/

mongodb for node.js (CRUD)

- find, find cursors, find lt & gt, find projection, find And Modify, findOne
- import JSON, insert Array, insert Doc
- Regex selector
- remove
- save
- skip Limit Sort
- update, update Single, update Multi
- upsert

find

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'grade' : 100 };

    db.collection('grades').find(query).toArray(function(err, docs) {
        if(err) throw err;

        console.dir(docs);

        db.close();
    });
});
```

find cursor

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'grade' : 100 };

    var cursor = db.collection('grades').find(query);

    cursor.each(function(err, doc) {
        if(err) throw err;

        if(doc == null) {
            return db.close();
        }

        console.dir(doc.student + " got a good grade!");
    });
});
```

find lt and gt

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'student' : 'Joe', 'grade' : { '$gt' : 80, '$lt' : 95 } };

    db.collection('grades').find(query).each(function(err, doc) {
        if(err) throw err;

        if(doc == null) {
            return db.close();
        }

        console.dir(doc);
    });
});
```

find projection

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'grade' : 100 };

    var projection = { 'student' : 1, '_id' : 0 };

    db.collection('grades').find(query, projection).toArray(function(err, docs) {
        if(err) throw err;

        docs.forEach(function (doc) {
            console.dir(doc);
            console.dir(doc.student + " got a good grade!");
        });

        db.close();
    });
});
```

find and modify

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'name' : 'comments' };
    var sort = [];
    var operator = { '$inc' : { 'counter' : 1 } };
    var options = { 'new' : true };

    db.collection('counters').findAndModify(query, sort, operator, options, function(err, doc) {
        if(err) throw err;

        if (!doc) {
            console.log("No counter found for comments.");
        }
        else {
            console.log("Number of comments: " + doc.counter);
        }

        return db.close();
    });
});
```

findone

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'grade' : 100 };

    function callback(err, doc) {
        if(err) throw err;

        console.dir(doc);

        db.close();
    }

    /* START STUDENT CODE */
    db.collection('grades').findOne(query, callback);
    /* END STUDENT CODE */
});
```

import json

```
var MongoClient = require('mongodb').MongoClient
  , request = require('request');

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    request('http://www.reddit.com/r/technology/.json', function (error, response, body) {
        if (!error && response.statusCode == 200) {
            var obj = JSON.parse(body);

            var stories = obj.data.children.map(function (story) { return story.data; });

            db.collection('reddit').insert(stories, function (err, data) {
                    if(err) throw err;

                    console.dir(data);

                    db.close();
            });
        }
    });
});
```

insert array

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var docs = [ { 'student' : 'Calvin', 'age' : 6 },
                 { 'student' : 'Susie', 'age' : 7 } ];

    db.collection('students').insert(docs, function(err, inserted) {
        if(err) throw err;

        console.dir("Successfully inserted: " + JSON.stringify(inserted));

        return db.close();
    });
});
```

insert doc

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var doc = { '_id' : 'calvin', 'age' : 6 };

    db.collection('students').insert(doc, function(err, inserted) {
        if(err) throw err;

        console.dir("Successfully inserted: " + JSON.stringify(inserted));

        return db.close();
    });
});
```

regx (정규표현식) selector

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'title' : { '$regex' : 'Microsoft' } };

    var projection = { 'title' : 1, '_id' : 0 };

    db.collection('reddit').find(query, projection).each(function(err, doc) {
        if(err) throw err;

        if(doc == null) {
            return db.close();
        }

        console.dir(doc.title);
    });
});
```

remove

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'assignment' : 'hw3' };

    db.collection('grades').remove(query, function(err, removed) {
        if(err) throw err;

        console.dir("Successfully updated " + removed + " documents!");

        return db.close();
    });
});
```

save

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'assignment' : 'hw2' };

    db.collection('grades').findOne(query, function(err, doc) {
        if(err) throw err;

        doc['date_returned'] = new Date();

        db.collection('grades').save(doc, function(err, saved) {
            if(err) throw err;

            console.dir("Successfully saved " + saved + " document!");

            return db.close();
        });
    });
});
```

skip limit sort

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var grades = db.collection('grades');

    var cursor = grades.find({});
    cursor.skip(1);
    cursor.limit(4);
    cursor.sort('grade', 1);
    //cursor.sort([['grade', 1], ['student', -1]]);

    //var options = { 'skip' : 1,
    //                'limit' : 4,
    //                'sort' : [['grade', 1], ['student', -1]] };
    //var cursor = grades.find({}, {}, options);

    cursor.each(function(err, doc) {
        if(err) throw err;
        if(doc == null) {
            return db.close();
        }
        console.dir(doc);
    });
});
```

update

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'assignment' : 'hw1' };
    var operator = { 'assignment' : 'hw2', '$set' : { 'date_graded' : new Date() } };

    db.collection('grades').update(query, operator, function(err, updated) {
        if(err) throw err;

        console.dir("Successfully updated " + updated + " document!");

        return db.close();
    });
});
```

update single

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'assignment' : 'hw1' };

    db.collection('grades').findOne(query, function(err, doc) {
        if(err) throw err;
        if(!doc) {
            console.log('No documents for assignment ' + query.assignment + ' found!');
            return db.close();
        }

        query['_id'] = doc['_id'];
        doc['date_returned'] = new Date();

        db.collection('grades').update(query, doc, function(err, updated) {
            if(err) throw err;

            console.dir("Successfully updated " + updated + " document!");

            return db.close();
        });
    });
});
```

update multi

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { };
    var operator = { '$unset' : { 'date_returned' : '' } };
    var options = { 'multi' : true };

    db.collection('grades').update(query, operator, options, function(err, updated) {
        if(err) throw err;

        console.dir("Successfully updated " + updated + " documents!");

        return db.close();
    });
});
```

upset (update and insert..)

```
var MongoClient = require('mongodb').MongoClient;

MongoClient.connect('mongodb://localhost:27017/course', function(err, db) {
    if(err) throw err;

    var query = { 'student' : 'Frank', 'assignment' : 'hw1' };
    //var operator = { 'student' : 'Frank', 'assignment' : 'hw1', 'grade' : 100 };
    var operator = { '$set' : { 'date_returned' : new Date(), 'grade' : 100 } };
    var options = { 'upsert' : true };

    db.collection('grades').update(query, operator, options, function(err, upserted) {
        if(err) throw err;

        console.dir("Successfully upserted " + upserted + " document!");

        return db.close();
    });
});
```

확인질문
(1) >= 에 해당하는 구문은?
(2) insert() 와 save() 와 update() 의 차이는?
(3) and 조건을 사용하는 방법은?

숙제 (25% 배점)

1) nedb 기반의 memo restful 서비스를 mongo로 바꾸시오!

2) formidable npm 패키지를 이용해 글과 이미지를 같이 업로드 하는 restful 서비스를 위 memo 서비스에 추가하여 만드시오. (postman과 연동)

https://www.dropbox.com/s/ahrr4psuu2f65wg/imageupload.js?dl=0

3) multiple file (image) upload가 가능한 restful 서비스를 만드시오. 단 확인을 위해 read가 가능한 서비스도 만드시오. 즉 파일 이미지가 아닌 이미지 url 경로들의 묶음을 전달하며 된다.

mongodb primary secondary..

- http://mongodb.github.io/node-mongodb-native/driver-articles/anintroductionto1_1and2_2.html

참고 자료

- NOSQL 간단히 알아보자 https://embian.wordpress.com/2013/06/27/nosql-2/
- But I need a database that scales http://spiegela.com/2014/04/18/but-i-need-a-database-that-scales/

THANK YOU!!!