1. NOSQL 개요

- NOSQL 이란?
- NOSQL 종류
- NOSQL 등장 배경
- Big Data와 NOSQL
- BASE 속성
- CAP 이론
- PACELC 이론
- NOSQL 전망
- NOSQL과 RDBMS 비교

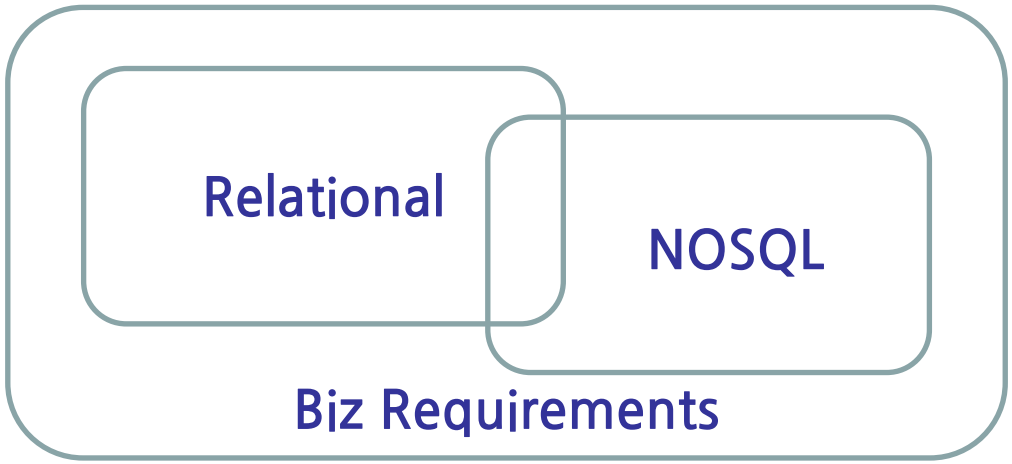
1.1 NOSQL 이란?

- **Not Only SQL**
  - 비관계형 데이터 스토리지 시스템, 비정형 데이터베이스들을 통칭
  - 일반적으로 고정된 테이블 스키마와 조인 개념을 필요로 하지 않음.
- **ACID 속성은 유연하게 적용.**
  - 하나 또는 그 이상의 속성 적용 하지 않음.
  - CAP 정리(CAP theorem)
- **대부분 Open Source**
- **수평적 확장(Scale out)**
- **Replication, Gossip 기능을 이용해 가용성 확보**

1.1. NOSQL 이란?

왜 NO SQL을 사용하는가?

- 관계형 데이터베이스로는 해결이 되지 않는 부분 존재
  - 관계형 데이터베이스는 비정형 데이터, 반정형 데이터 처리에 적합하지 않음.
- Schema Free → Agile!
- 클라이언트 개발에 더 적합 → Javascript, python, Ruby
- 자동화된 장애 극복, 복구
- 대규모 데이터 처리에 있어서 관계형 데이터베이스의 낮은 성능



1.2 NOSQL 종류

- **Key-Value**
  - Dynamo, Redis, Voldemort, Riak
- **Column Oriented**
  - Cassandra, HBase, Big Table
- **Document**
  - MongoDB, Couchbase
- **Graph**
  - Neo4J

1.3 NOSQL의 등장 배경

- **대량의 데이터를 Read/Write 할 필요성 증가**
  - SNS의 발전, 웹 데이터의 폭발적 증가
- **지속적으로 증가하는 사용자에 대한 신속한 증가**
  - Scale Up vs Scale Out
- **빠르게 변화하는 비즈니스에 대한 신속한 대응**
  - 대용량 데이터에 대한 실시간 & 배치 분석
- **비정형 데이터의 폭발적 증가**
  - 관계형 데이터베이스와 같은 정규화된 형태로 관리가 힘듬

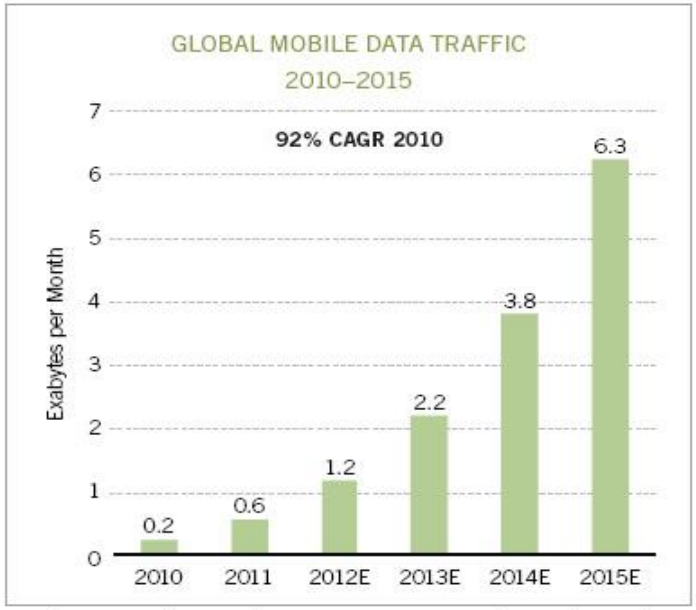
Not Only SQL

1.3 NOSQL의 등장 배경

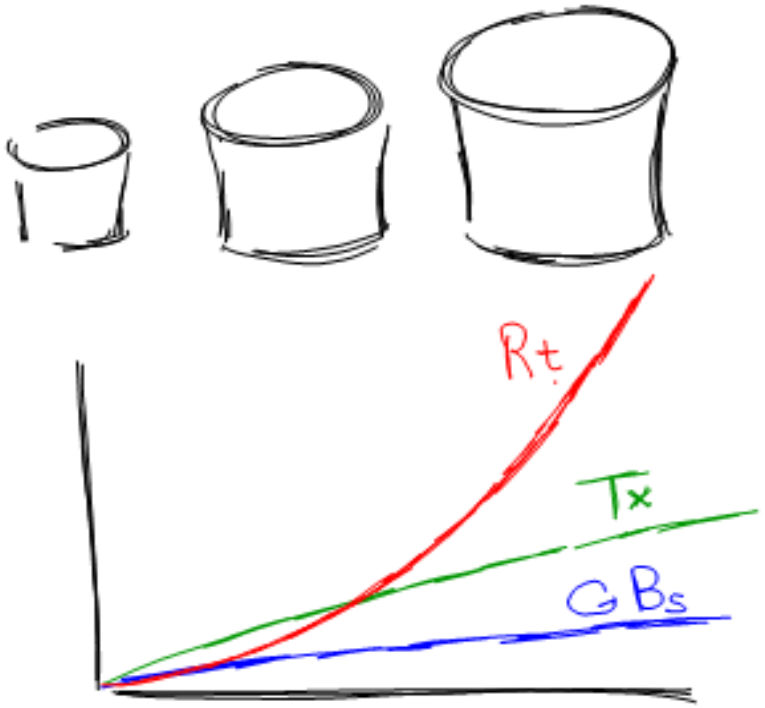
관계형 데이터베이스의 한계

- 관계형 데이터베이스로는 저장/관리하기 힘들어짐.
- 가능하다 하더라도 **많은 비용을 지불**해야 함.

Exhibit 1:
Globally, mobile data has been expanding exponentially.



Exabyte = 1 million terabytes. CAGR: Compound Annual Growth Rate. Source: Cisco VNI Mobile, 2011.



1.4 Big Data와 NOSQL

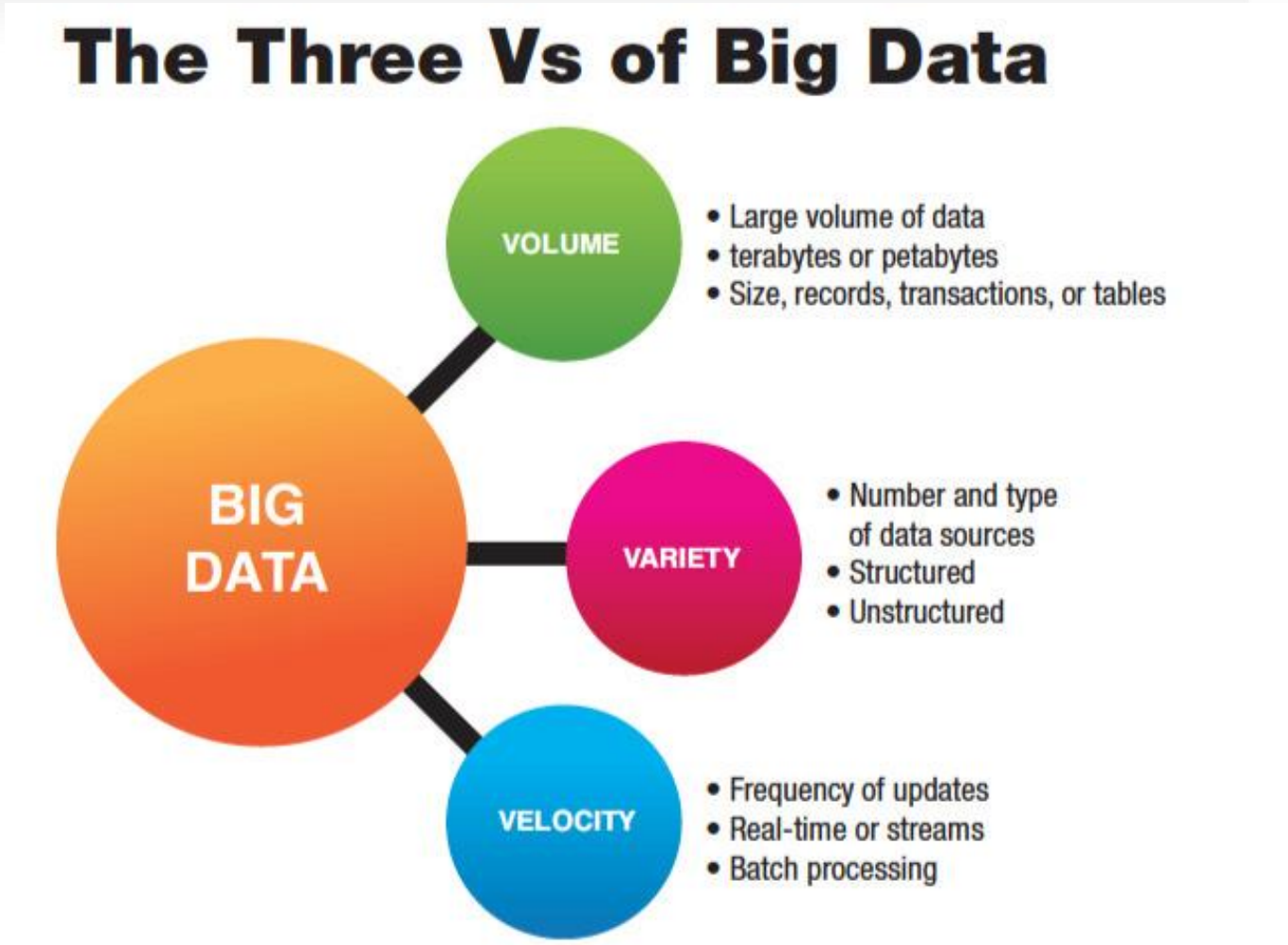


Figure 1

Source: Umesh Jain

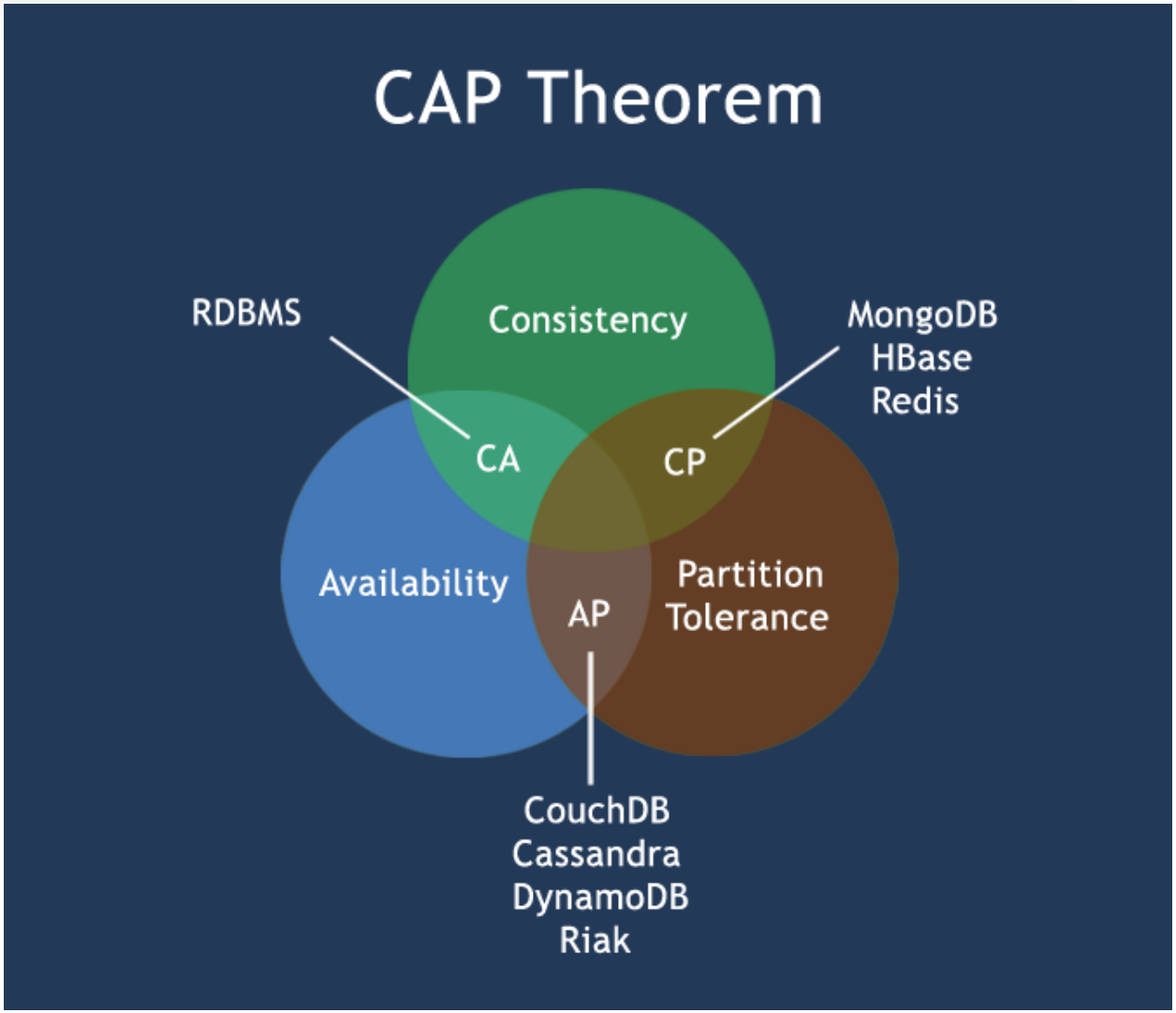
1.4 Big Data와 NOSQL

- **대량의 데이터 쓰기에 최적화**
- **대량의 데이터를 효과적으로 저장하기 위해 샤딩(Sharding)하거나 Ring 형태의 노드에 멀티 복제하는 방법을 사용**
  - MongoDB : 자동 샤딩
  - 대부분의 RDB : 수동 샤딩
  - Cassandra : Ring 형태의 Gossip 프로토콜을 이용한 다중 복제

1.5 BASE 속성

- **Basically Available, Soft state, Eventually consistency**
  - Soft state : 데이터의 사본은 inconsistent 할 수도 있음. 노드의 상태는 내부에 포함된 정보가 아닌 외부에서 전달(전송)된 데이터에 의해 결정됨. 예) replication
  - Eventually Consistent : 데이터의 복사본은 더 이상의 업데이트가 없다면, 약간의 지연시간 후에 consistent 하게 됨. 예) DNS
  - Basically Available : Fault의 가능성이 있지만, 전체 시스템의 fault가 되지는 않음.
- **BASE는 ACID와 대치되는 개념**
- **BASE는 분산 데이터 시스템의 특징**

1.6 CAP 이론


CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
Partition
Tolerance
AP
CouchDB
Cassandra
DynamoDB
Riak

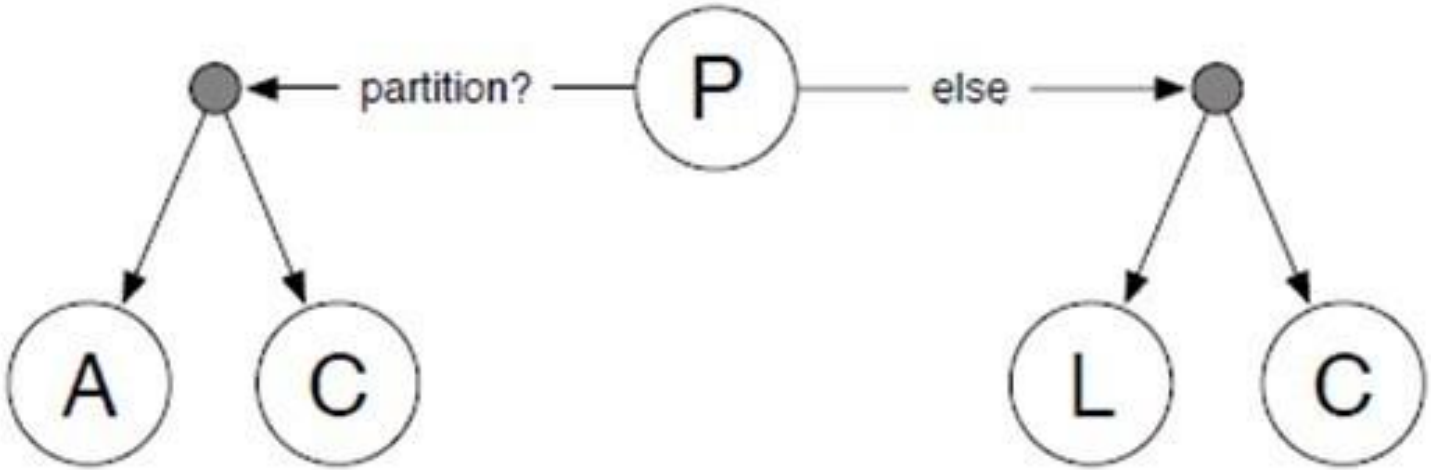
1.6 CAP 이론

- **분산시스템과 CAP**
  - 분산시스템은 기본적으로 Partition Tolerance를 지원해야 하기 때문에 C와 A중 하나는 포기하거나 약하게 적용해야 함.
  - C, A 중심
    - 관계형 데이터베이스 : 분산시스템을 고려해 설계된 것이 아님
  - A, P 중심
    - Amazon Dynamo, Cassandra, Riak 등
  - C, P 중심
    - BigTable, HBase, MongoDB, Redis 등
- **다만 중심일 뿐이고 나머지는 유연하게 지원**

1.7 PACELC 이론

PACELC

- 분산시스템에서 정상상황일때와 장애상황일 때를 나누어 설명하는 개념
- Partition, Availability, Consistency, else Latency, Consistency



1.8 NOSQL 전망

DB Engine Ranking

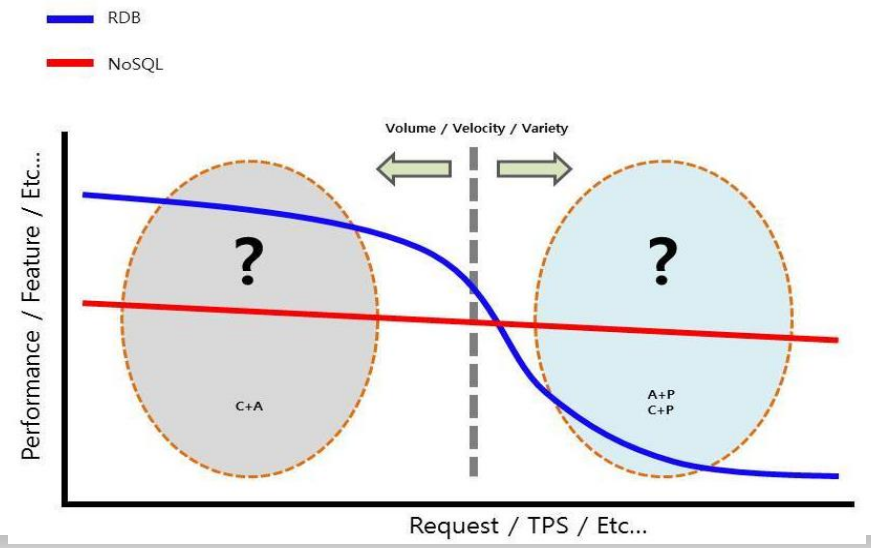
- http://db-engines.com/en/ranking

Rank Jan 2017	Rank Dec 2016	Rank Jan 2016	DBMS	Database Model	Score Jan 2017	Score Dec 2016	Score Jan 2016
1.	1.	1.	Oracle	Relational DBMS	1416.72	+12.32	-79.36
2.	2.	2.	MySQL	Relational DBMS	1366.29	-8.12	+67.03
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1220.95	-5.70	+76.89
4.	↑ 5.	4.	MongoDB	Document store	331.90	+3.22	+25.88
5.	↓ 4.	5.	PostgreSQL	Relational DBMS	330.37	+0.35	+47.97
6.	6.	6.	DB2	Relational DBMS	182.49	-1.85	-13.88
7.	7.	↑ 8.	Cassandra	Wide column store	136.44	+2.16	+5.49
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	127.45	+2.75	-6.59
9.	9.	↑ 10.	Redis	Key-value store	118.70	-1.20	+17.54
10.	10.	↓ 9.	SQLite	Relational DBMS	112.38	+1.54	+8.64
11.	11.	↑ 12.	Elasticsearch	Search engine	106.17	+2.90	+28.96
12.	12.	↑ 14.	Teradata	Relational DBMS	74.17	+0.79	-0.78
13.	13.	↓ 11.	SAP Adaptive Server	Relational DBMS	69.10	-1.32	-14.08
14.	14.	↓ 13.	Solr	Search engine	68.08	-0.92	-7.32
15.	15.	↑ 16.	HBase	Wide column store	59.14	+0.51	+5.77
16.	16.	↑ 18.	Splunk	Search engine	55.49	+0.57	+12.37
17.	17.	17.	FileMaker	Relational DBMS	53.49	-0.63	+4.66
18.	18.	↑ 19.	SAP HANA	Relational DBMS	51.93	+0.16	+13.32
19.	19.	↓ 15.	Hive	Relational DBMS	51.14	+1.74	-2.45
20.	20.	↑ 23.	MariaDB	Relational DBMS	45.04	+0.95	+17.28

1.9 NOSQL과 RDBMS 비교

비교

구분	RDBMS	NOSQL
장단점	• 데이터 무결성 보장(CA) • 정규화된(정형) 데이터 처리 • 확장성 문제. 분산환경에 적합(X)	• 데이터 무결성, 정합성을 보장하지 않을 수 있음. • 비정형, 반정형 데이터 처리
특징	• JOIN • ACID	• 강한 Consistency(X) • Schema가 없거나 변경이 유연함.
Use Cases	• 중요한 트랜잭션 처리(ex:금융)가 요구되는 경우	• 대량의 데이터 처리가 필요한 경우 • 빠른 성능을 요구하는 경우



* 출처 : http://call518.tistory.com/80