BIG Data

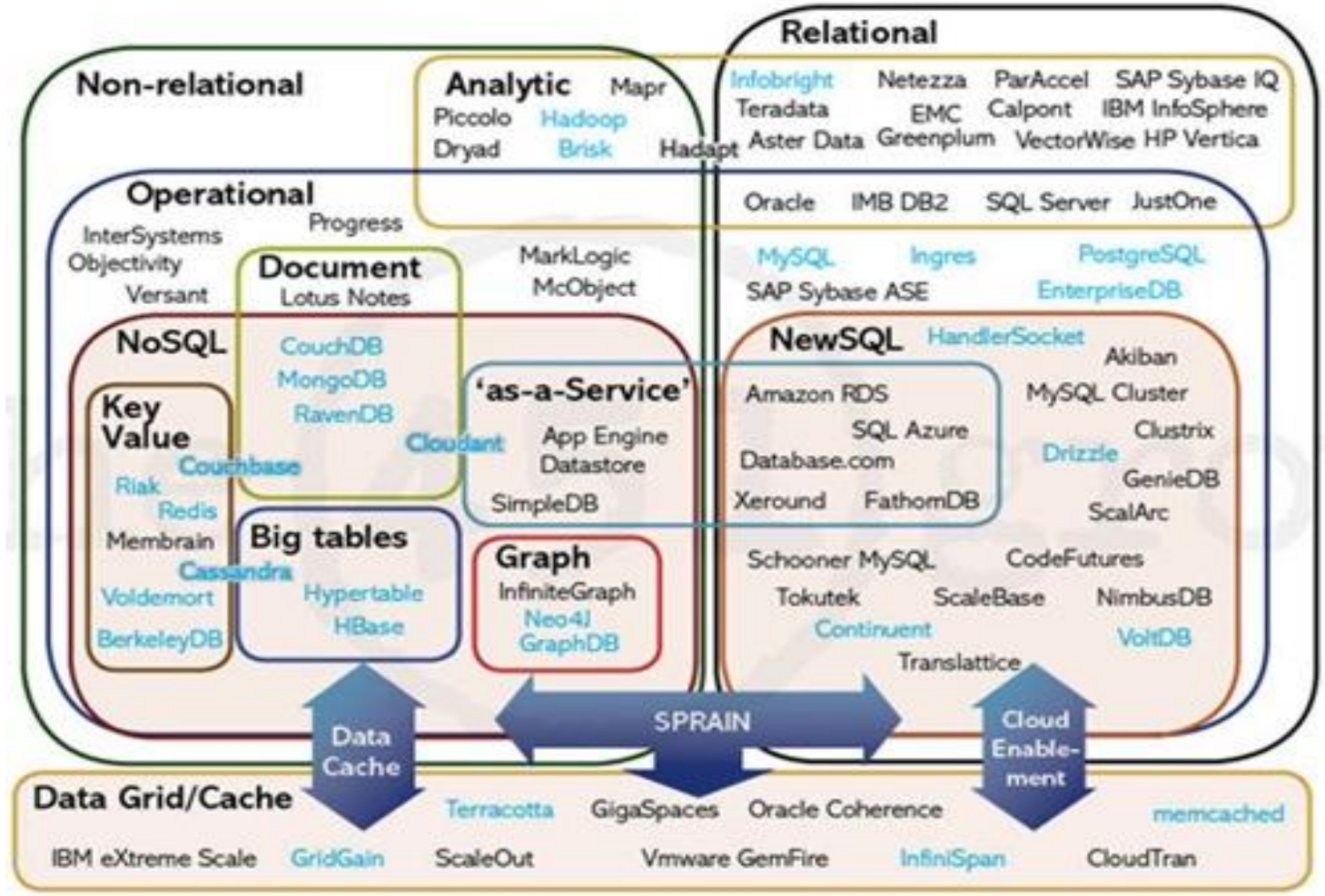
Sirojul Munir | rojulman@nurulfikri.ac.id | @rojulman

NoSQL

Sirojul Munir | rojulman@nurulfikri.ac.id | @rojulman

Solusi Tata Kelola Sistem Penyimpanan Data

- Data Terstruktur : Relational Database
- Data Tidak Terstruktur : NoSQL Database



Database Relational

- ***EF Codd (1970):***

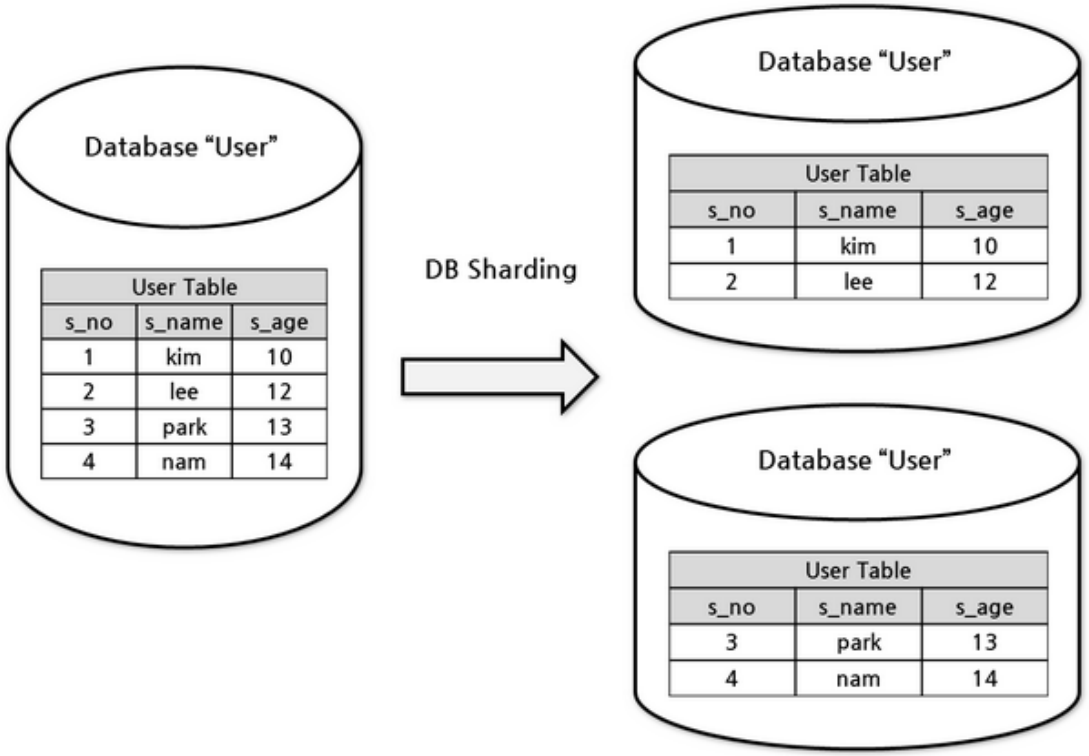
Memperkenalkan model relasional data, pada saat itu sebagian besar sistem database berdasarkan dua model data: Model hirarki (hierarchical model) dan model jaringan (network model). Prototipe sistem database model relasional dikembangkan di IBM dan di UC-Berkeley pada pertengahan tahun 1974

Relational Model

- Simple & Elegan
  - Database adalah kumpulan dari satu atau lebih dari relasi, dimana setiap relasi adalah berupa tabel, kolom dan baris
- Keuntungan
  - Tampilan data berbentuk tabular mudah dimengerti
  - Kemudahan tampilan data walaupun dengan query yang rumit
- Kekurangan
  - Tidak bisa menghandle untuk data yang tidak berstruktur seperti big data (volume, velocity,varieaty)
  - RDBMS tidak dirancang untuk system yang terdistribusi (walaupun dapat dilakukan dengan dengan konsep multi-node database → scaling-out/horiziontal scaling, model replikasi master-slave , sharding)

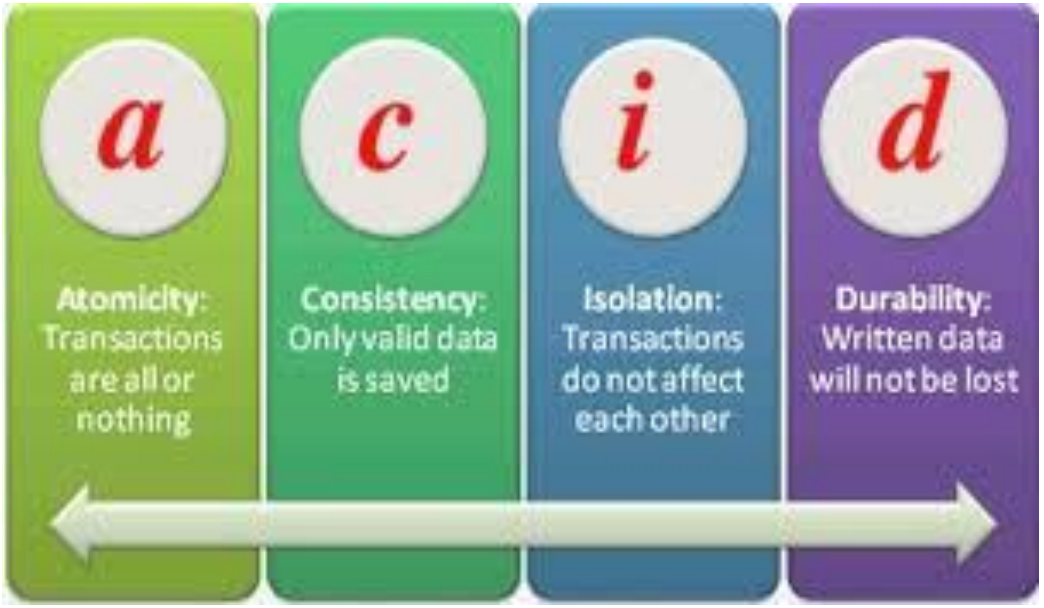
Sharding of data

- Distributes a single logical database system across a cluster of machines
- Uses range based partitioning to distribute documents based on a specific shard key
- Automatically balances the data associated with each shard
- Can be turned on and off per collection (table)



RDBMS : ACID

- Atomicity
  - Setiap transaksi sql bersifat atomic “semua diekskusi atau tidak sama sekali”
- Consistency
  - Memastikan semua transaksi bersifat konsisten dari satu state ke state lainnya
- Isolation
  - Sebuah transaksi bersifat independensi dan terisolasi, sebuah transaksi dipastikan tidak mempengaruhi transaksi lainnya
- Durability
  - Memastikan transaksi yang telah dilakukan (commited) tidak mengakibatkan hilanganya data



Relational Model : Tabel

column / field

row / record

No	NIM	Nama	Prodi	Thn Angkatan	IPK	Predikat
1	02011	Faiz Fikri	2012	TI	3.8	Cum Laude
2	02012	Alissa Khairunnisa	2012	TI	3.9	Cum Laude
3	01011	Rosalie Naurah	2010	SI	3.46	Memuaskan
4	01012	Defghi Muhammad	2010	SI	3.2	Memuaskan

- **Field/Column** – Satu jenis informasi/data yang Mempunyai Tipe Data Sama
- ***Record/Row*** – Satu kesatuan informasi yang terdiri atas satu Field atau lebih
- ***Character*** – Satuan terkecil dari data

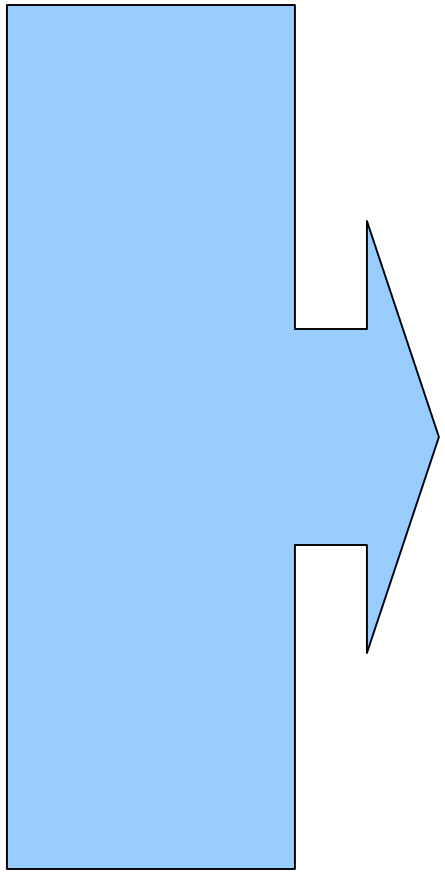
Vendor Relational DBMS

Proprietary

- MSQL Server
- MS Access
- Oracle
- IBM DB2
- SyBase

Open Source

- MySQL
- PostgreSQL
- MariaDB
- SQLite



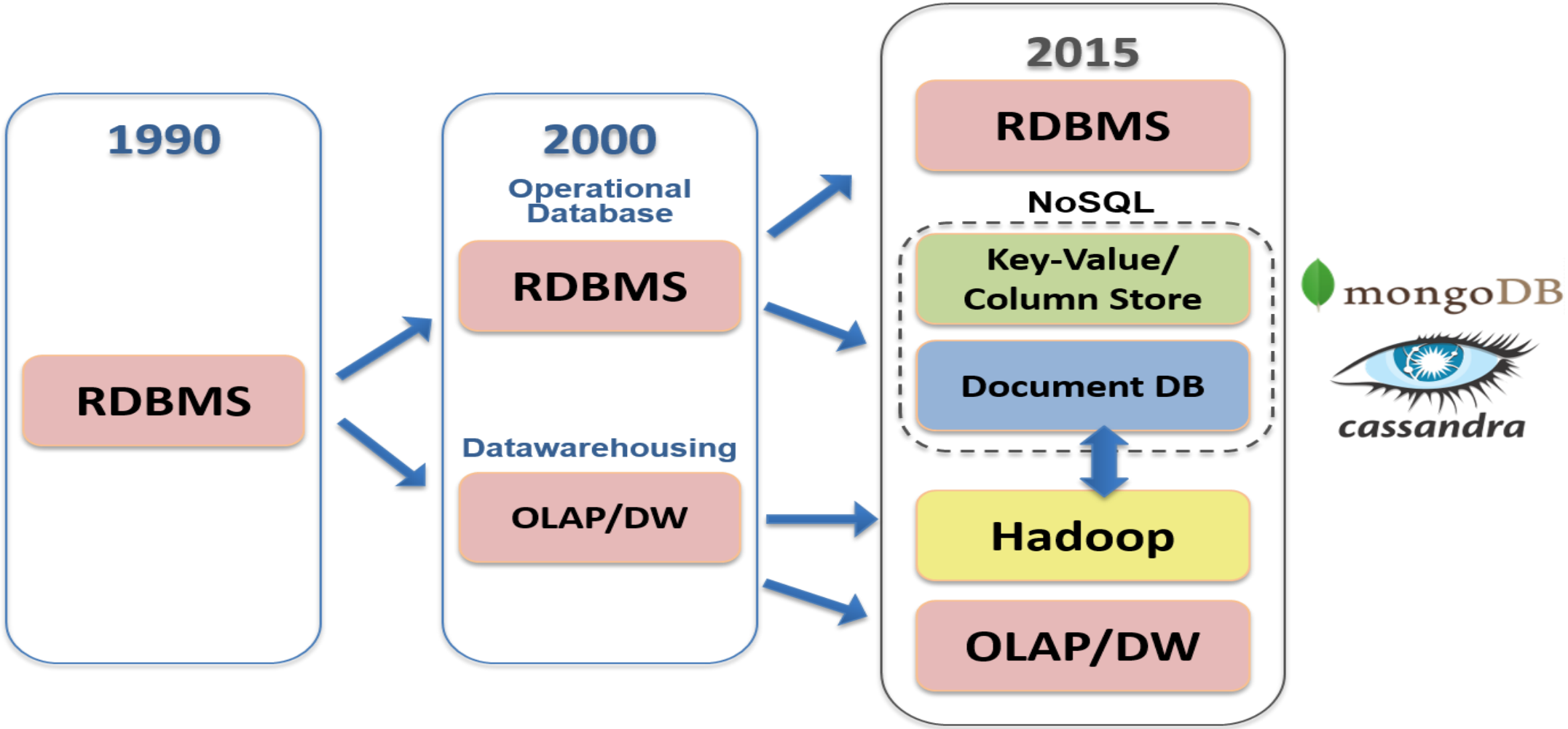
SQL:

Structured

Query

Language

Evolusi Database : RDBMS - BigData

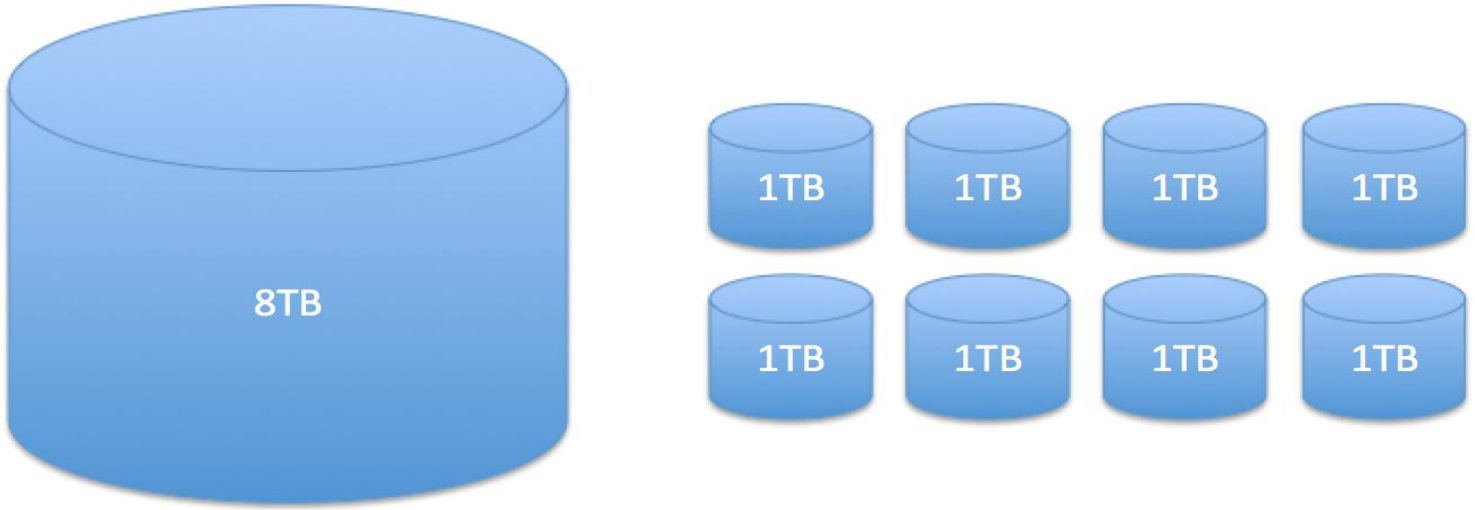


NoSQL

- Bagian dari DBMS (Database Management System)
- Not **O**nly **SQL**
- Sistem penyimpanan data non-relational database (tidak menggunakan query language SQL)
- Tidak memerlukan skema table (tidak memiliki skema yang fix)
- Tidak memiliki joins table seperti pada relational database
- NoSQL bukan untuk mengganti RDBMS, tetap melengkapi !!

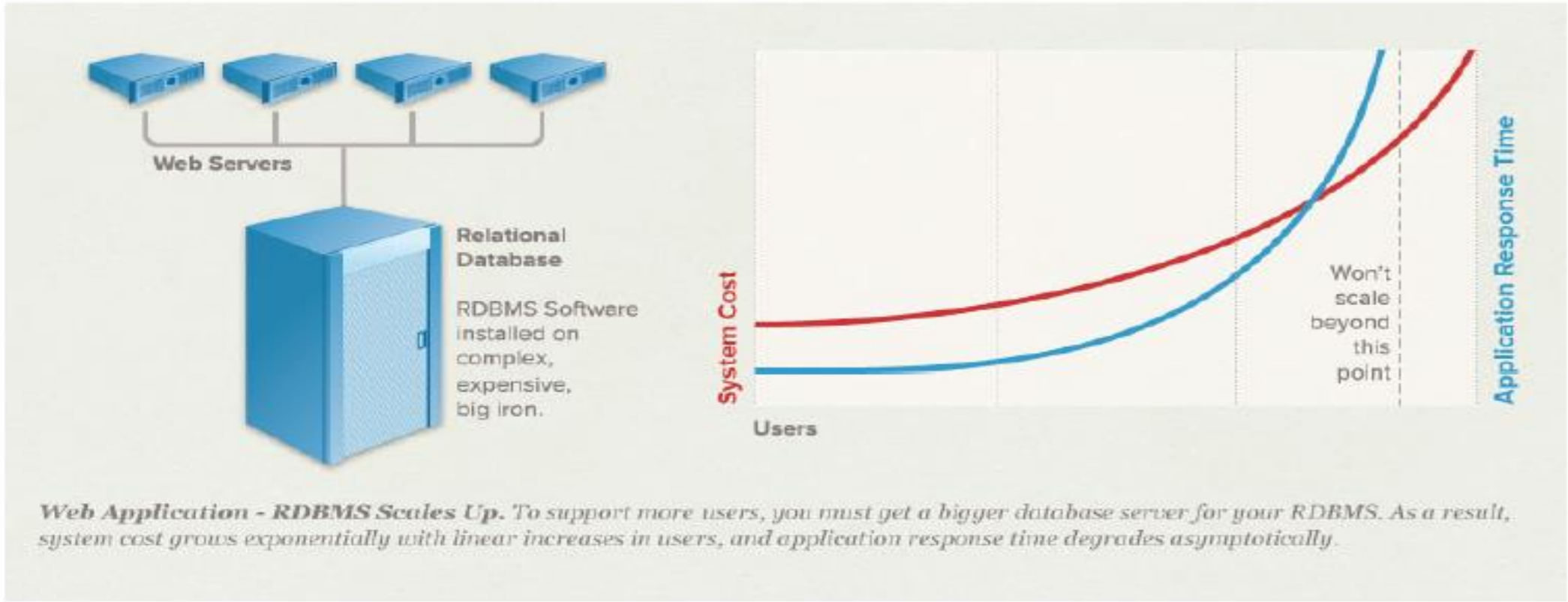
NoSQL vs RDBMS

- RDBMS : scaled up (peningkatan) dengan menambah hardware – processing power (RAM & Disk)
- NoSQL : scaled out (perluasan) dengan membagi beban load (partisi, sharing) menggunakan multinode



RDBMS – Scaling Up

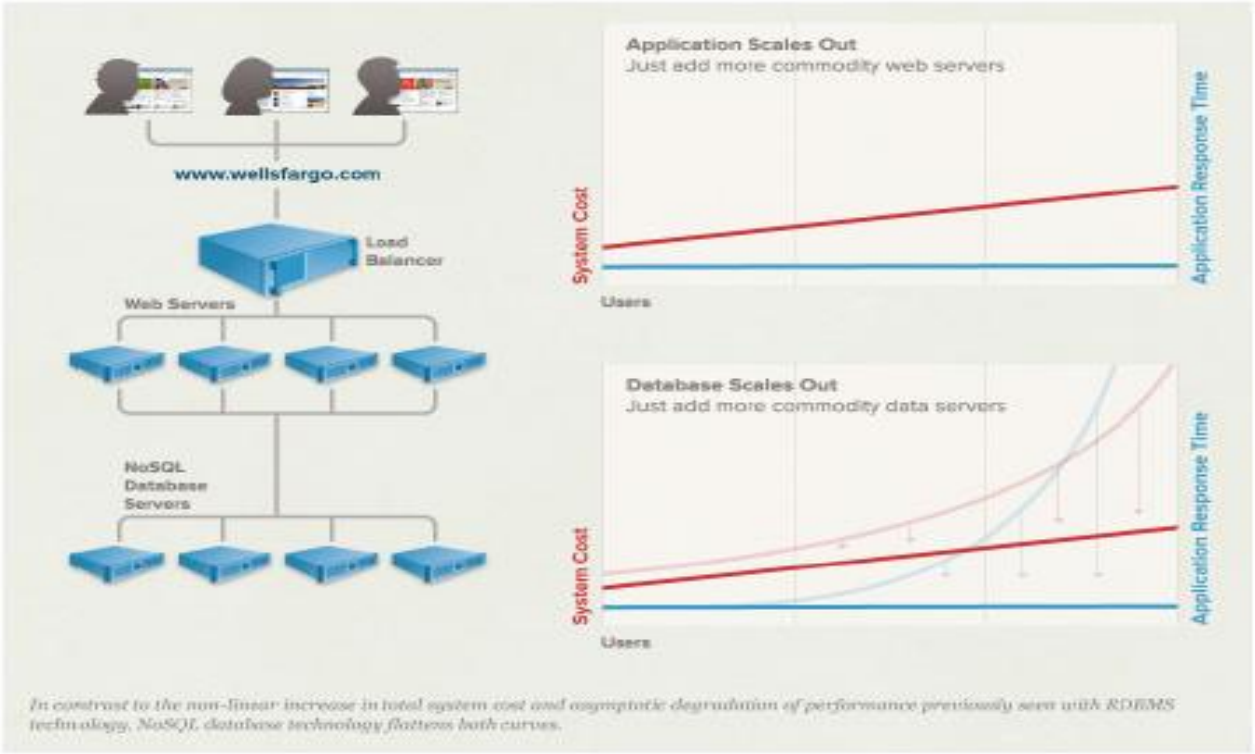
- At certain point relational database won't scale



Web Application - RDBMS Scales Up. *To support more users, you must get a bigger database server for your RDBMS. As a result, system cost grows exponentially with linear increases in users, and application response time degrades asymptotically.*

NoSQL – Scaling

- Scaling horizontally is possible with NoSQL
- Scaling up / down is easy
  - Supports rapid production-ready prototyping
- Better handling of traffic spikes



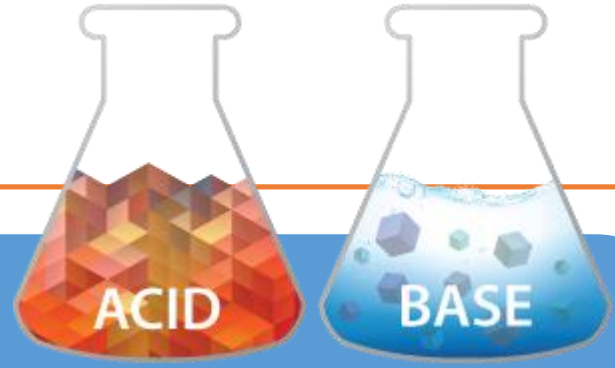
NoSQL vs RDBMS

- DBA Specialist
  - RDBMS require highly trained expert to monitor Database
  - NoSQL require less management, automatic repair and simple data models
- Big Data
  - Huge increase data, RDBMS: capacity and constraint of data volumes at its limits
  - NoSQL designed for Big Data
- Flexible data models
  - Change management to schema for RDBMS have to be carefully managed
  - NoSQL database more relaxed in structure of data
    - Database schema changed do not have to managed as one complicated change unit
    - Application already written to address an amorphous schema

NoSQL vs RDBMS

- Economics
  - RDBMS rely on expensive proprietary servers to manage data
  - NoSQL : clusters of cheap commodity servers to manage the data and transaction volume Big Data
  - Cost per gigabyte or transaction/second for NoSQL can be lower than the cost for RDBMS
- Support
  - RDBMS vendors provide a high level of support to client
  - NoSQL are open source projects with startups supporting them (reputation not yet established)
- Maturity
  - RDBMS Mature product: means stable and dependable
  - NoSQL are still implementing their basic feature set

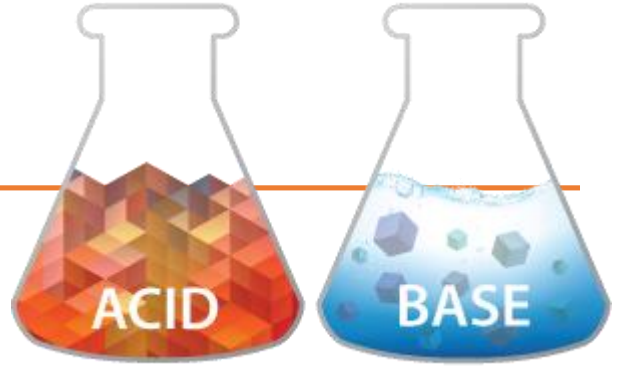
RDBMS ACID -> NoSQL BASE



Atomicity
Consistency
Isolation
Durability

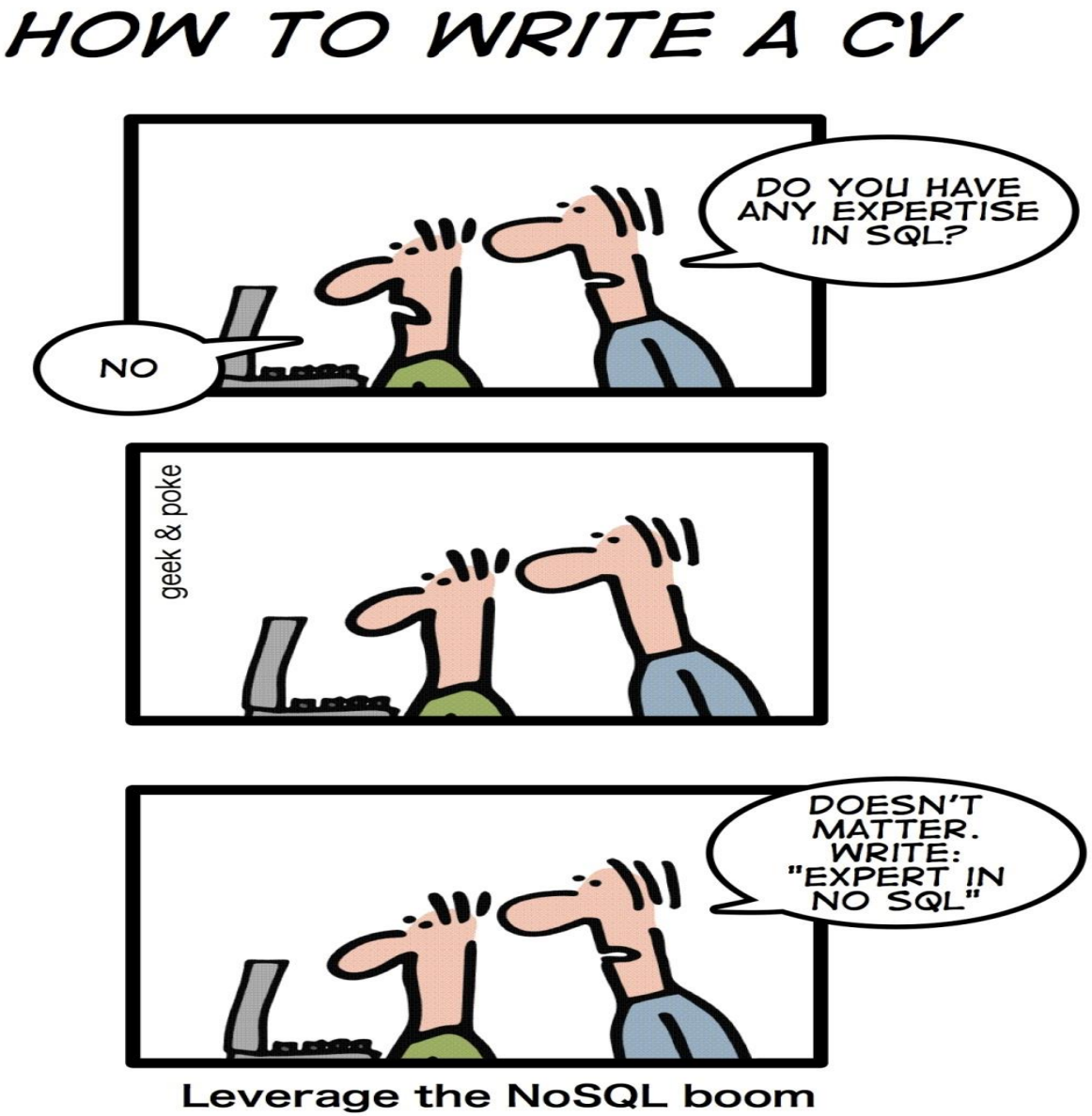
Basically Available (CP)
Soft-State
Eventually consistent

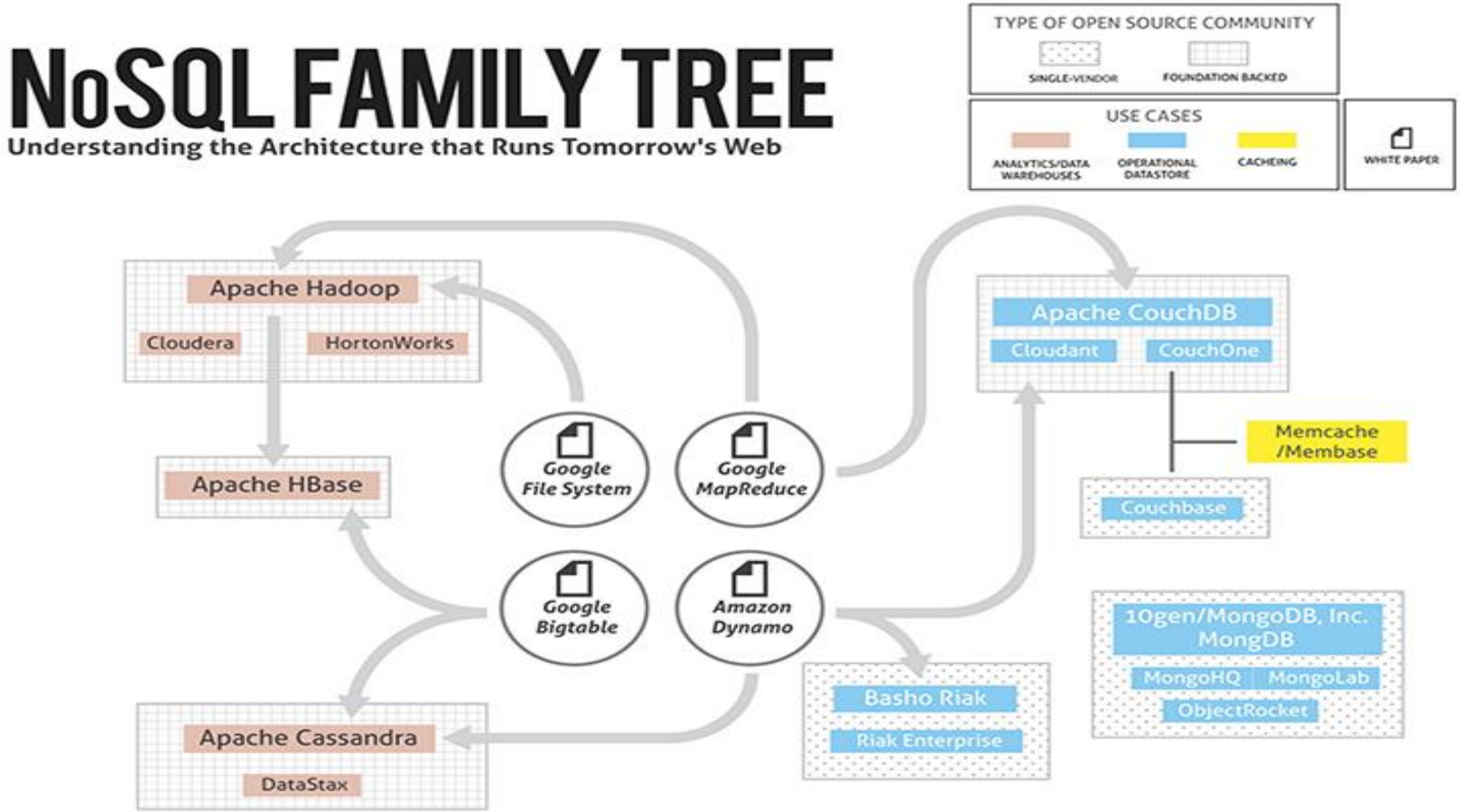
NoSQL : BASE

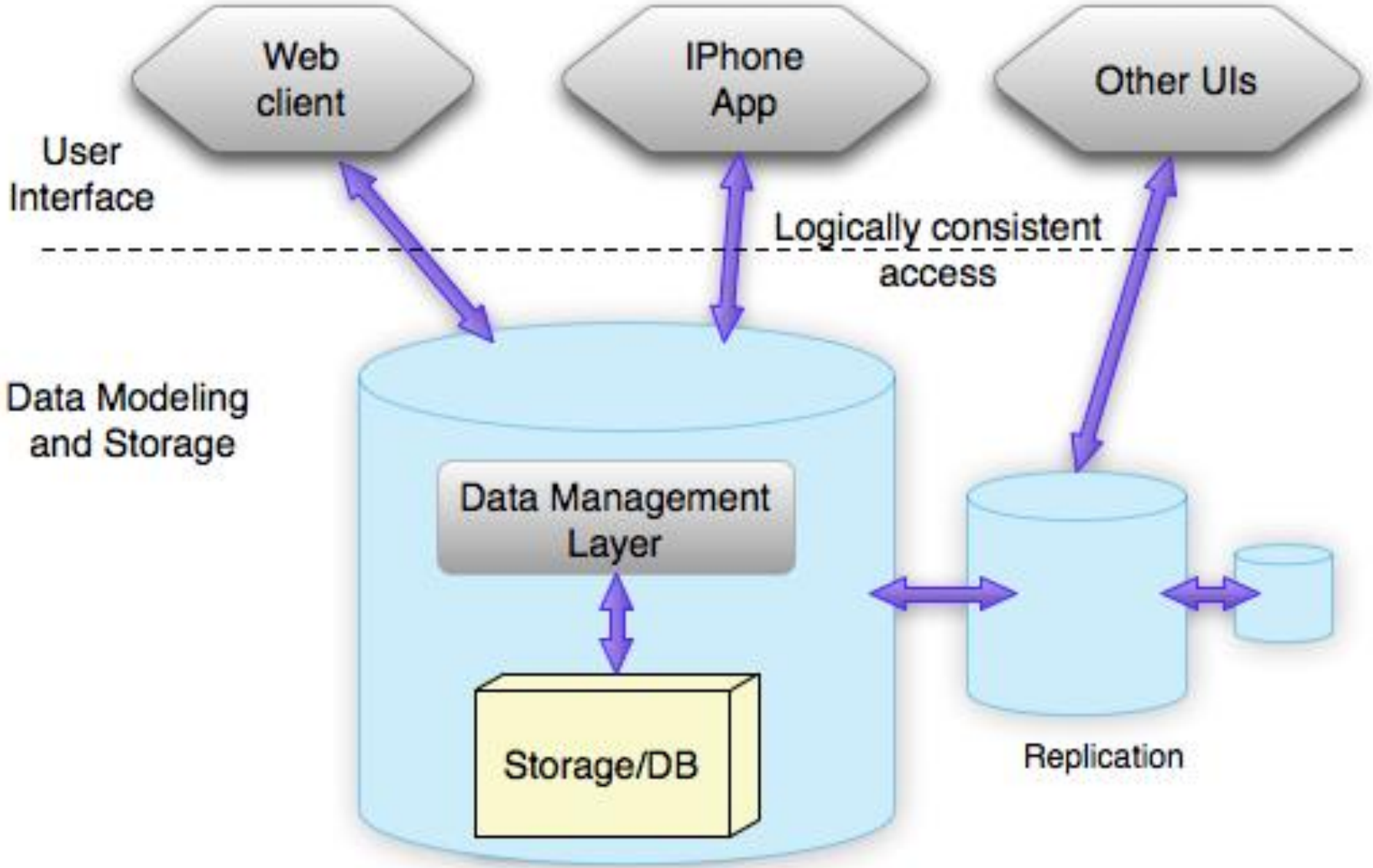


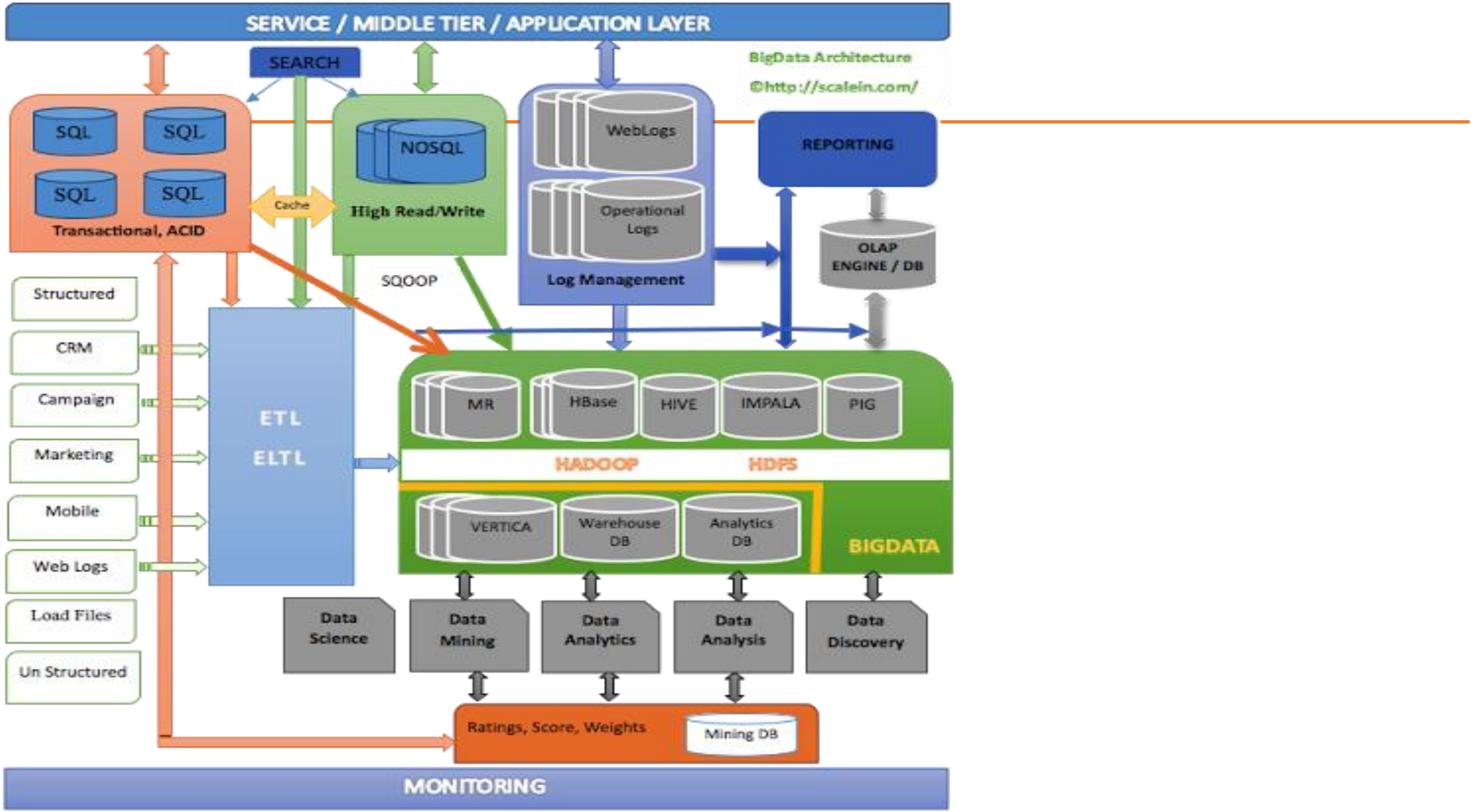
- Basically Available
  - fulfill request, even in partial consistency
  - the database appears to work most of the time
- Soft State
  - abandon the consistency requirements of the ACID model pretty much completely
  - Stores don't have to be write-consistent, nor do different replicas have to be mutually consistent all the time
- Eventually Consistency : ilustrasi
  - at some point in the future, data will converge to a consistent state; delayed consistency, as opposed to immediate consistency of the ACID properties
    - purely a liveness guarantee (reads eventually return the requested value); but
    - does not make safety guarantees, i.e.,
    - an eventually consistent system can return any value before it converges
    - https://hackernoon.com/eventual-vs-strong-consistency-in-distributed-databases-282fdad37cf7

intermezzo


HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom


NoSQL FAMILY TREE
Understanding the Architecture that Runs Tomorrow's Web
TYPE OF OPEN SOURCE COMMUNITY
SINGLE-VENDOR
FOUNDATION BACKED
USE CASES
ANALYTICS/DATA WAREHOUSES
OPERATIONAL DATASTORE
CACHEING
WHITE PAPER
Apache Hadoop
Cloudera
HortonWorks
Apache HBase
Google File System
Google MapReduce
Google Bigtable
Amazon Dynamo
Apache CouchDB
Cloudant
CouchOne
Memcache /Membase
Couchbase
10gen/MongoDB, Inc. MongDB
MongoHQ
MongoLab
ObjectRocket
Basho Riak
Riak Enterprise
Apache Cassandra
DataStax


Web client
IPhone App
Other UIs
User Interface
Logically consistent access
Data Modeling and Storage
Data Management Layer
Storage/DB
Replication


SERVICE / MIDDLE TIER / APPLICATION LAYER
SEARCH
BigData Architecture
©http://scalein.com/
SQL
SQL
SQL
SQL
Transactional, ACID
Cache
NOSQL
High Read/Write
WebLogs
Operational Logs
Log Management
REPORTING
OLAP ENGINE / DB
SQOOP
Structured
CRM
Campaign
Marketing
Mobile
Web Logs
Load Files
Un Structured
ETL
ELTL
MR
HBase
HIVE
IMPALA
PIG
HADOOP
HDFS
VERTICA
Warehouse DB
Analytics DB
BIGDATA
Data Science
Data Mining
Data Analytics
Data Analysis
Data Discovery
Ratings, Score, Weights
Mining DB
MONITORING

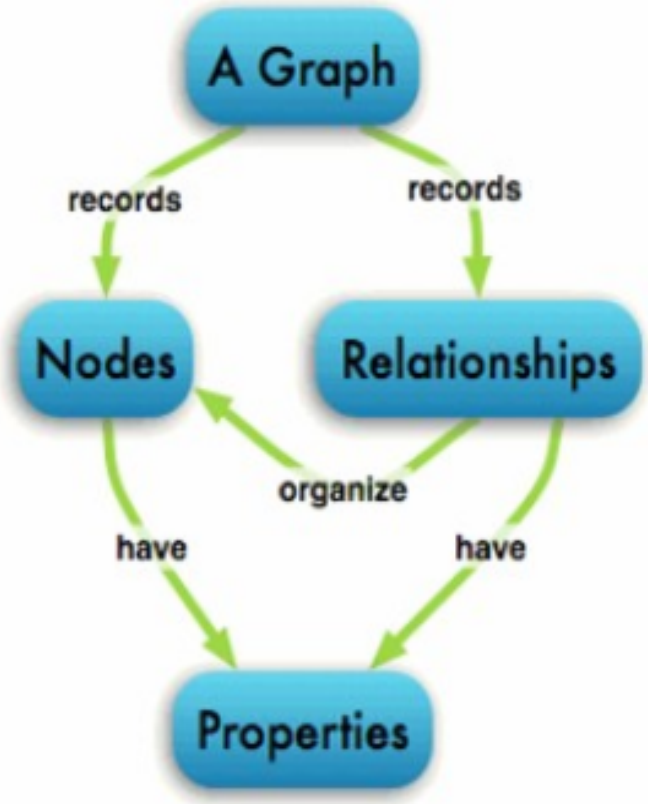
Model Data NoSQL

- Graph Database
- Key-Value Store
- Column Store
- Document Database

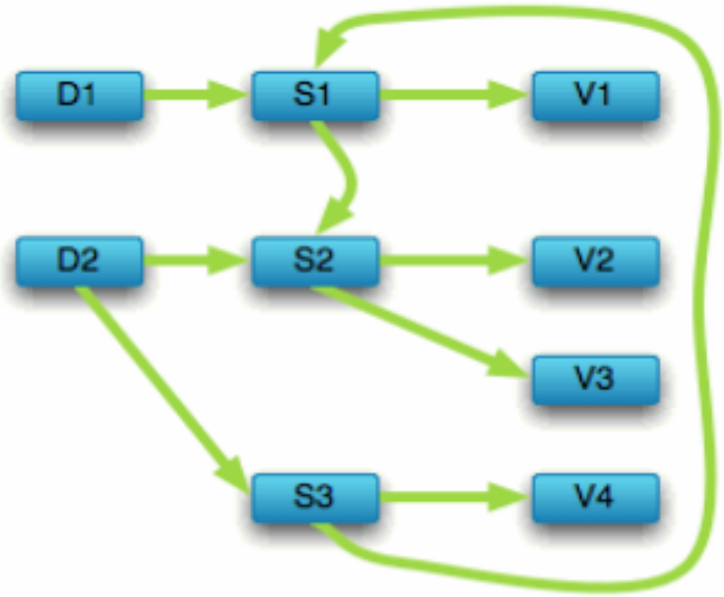
Graph Database

- Berdasarkan Teori Graph
- Database di desain berdasarkan relasi yang mewakili sebuah graph dan elemen interkoneksi graph
- Data tersimpan didalam node (simpul) dan edge (garis/sisi)
- Setiap node diorganisasikan berdasarkan relasi antar node (edge)
- Setiap node dan relasi yang terjadi terdapat pendefinisian properti
- Contohnya: Neo4j dan Titan

NoSQL Graph

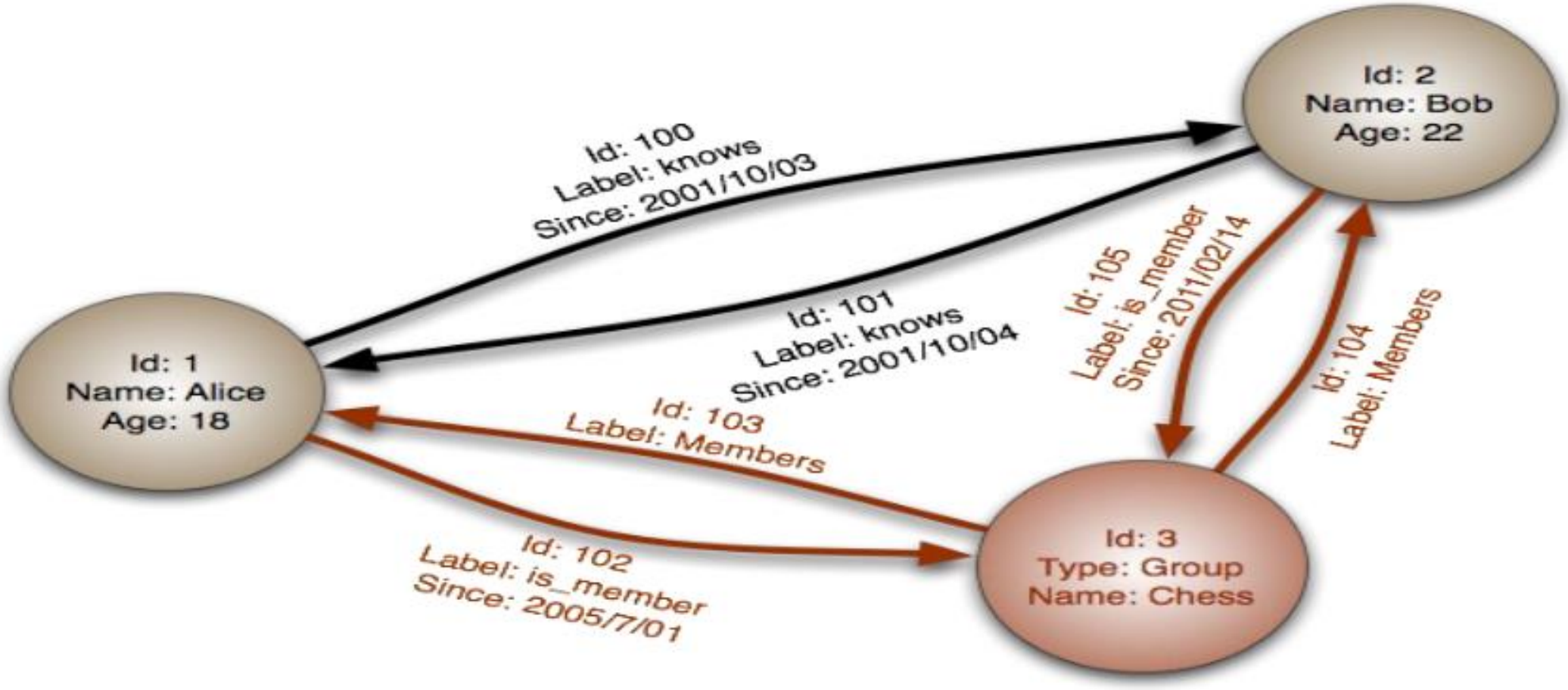


"A Graph Database -navigates a-> Document Store"



The container hierarchy of a document database accommodates nice, schema-free data that can easily be represented as a tree. Which is of course a graph. Refer to other documents (or document elements) within that tree and you have a more expressive representation of the same data that you can easily navigate with Neo4j.

Contoh Data : NoSQL - Graph



NoSQL : Key – Value Store

- Database di desain untuk menyimpan data dalam bentuk

format minimum skema (schema-less way)

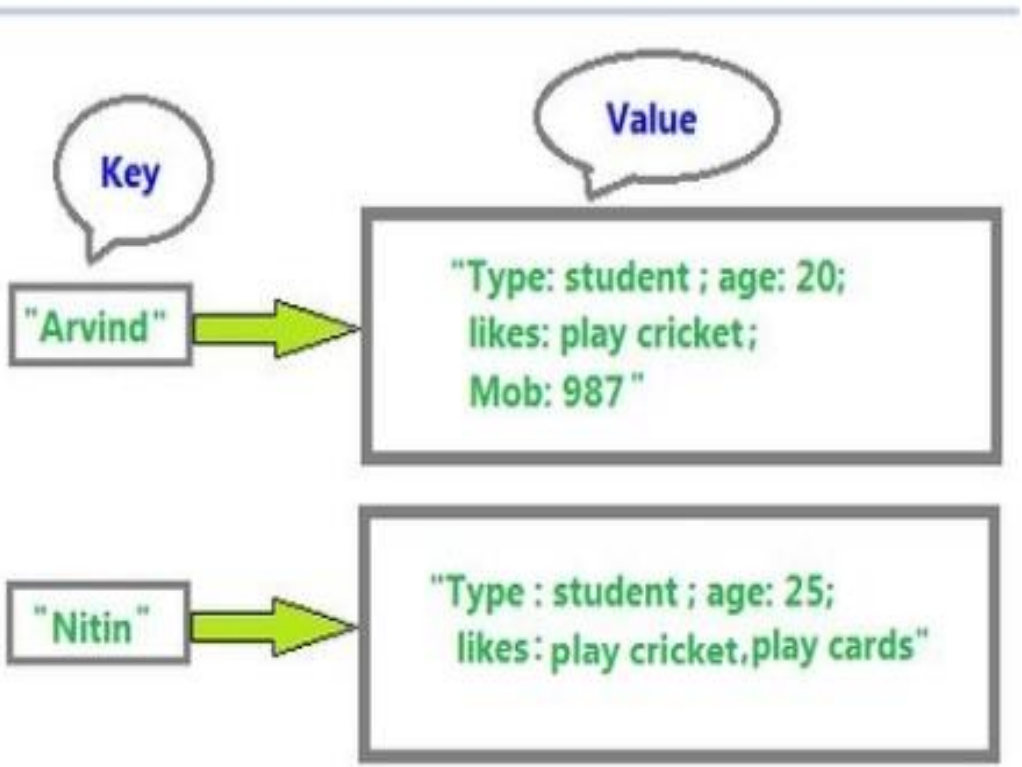
- Semua data tersimpan dalam format berpasangan key – value, yang didalamnya terdapat proses index key dan value
- Contohnya: Cassandra, DyanmoDB, Azure Table Storage ,(ATS), Riak, BerkeleyDB, Redis

Contoh Data : Key – Value Store

Key Space : /user/userId

Value : {

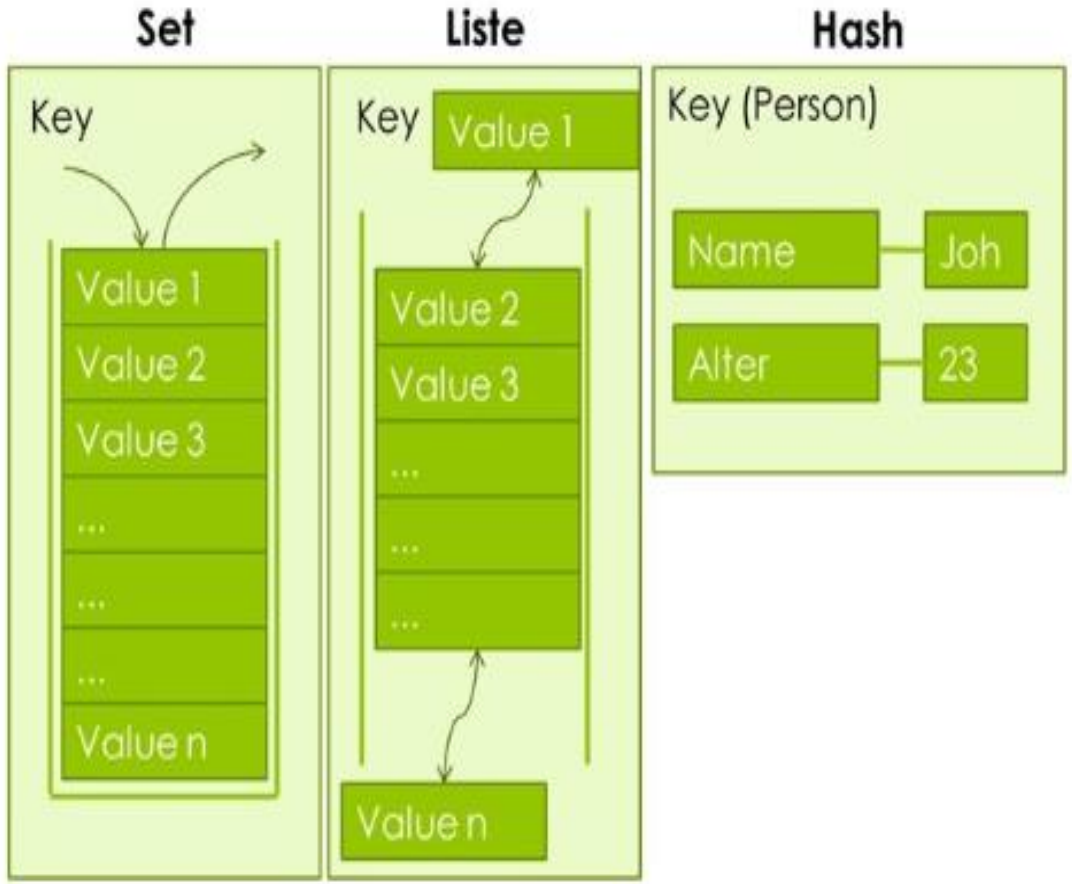
```
"name" : "User",
"namespace" : "com.company. avro",
"type" : "record",
    "fields": [
        {"name": "userId",      "type": "Integer", "default": 0},
        {"name": "firstName",  "type": "String", "default": ""},
        {"name": "lastName",   "type": "String", "default": ""}
    ]
}
```

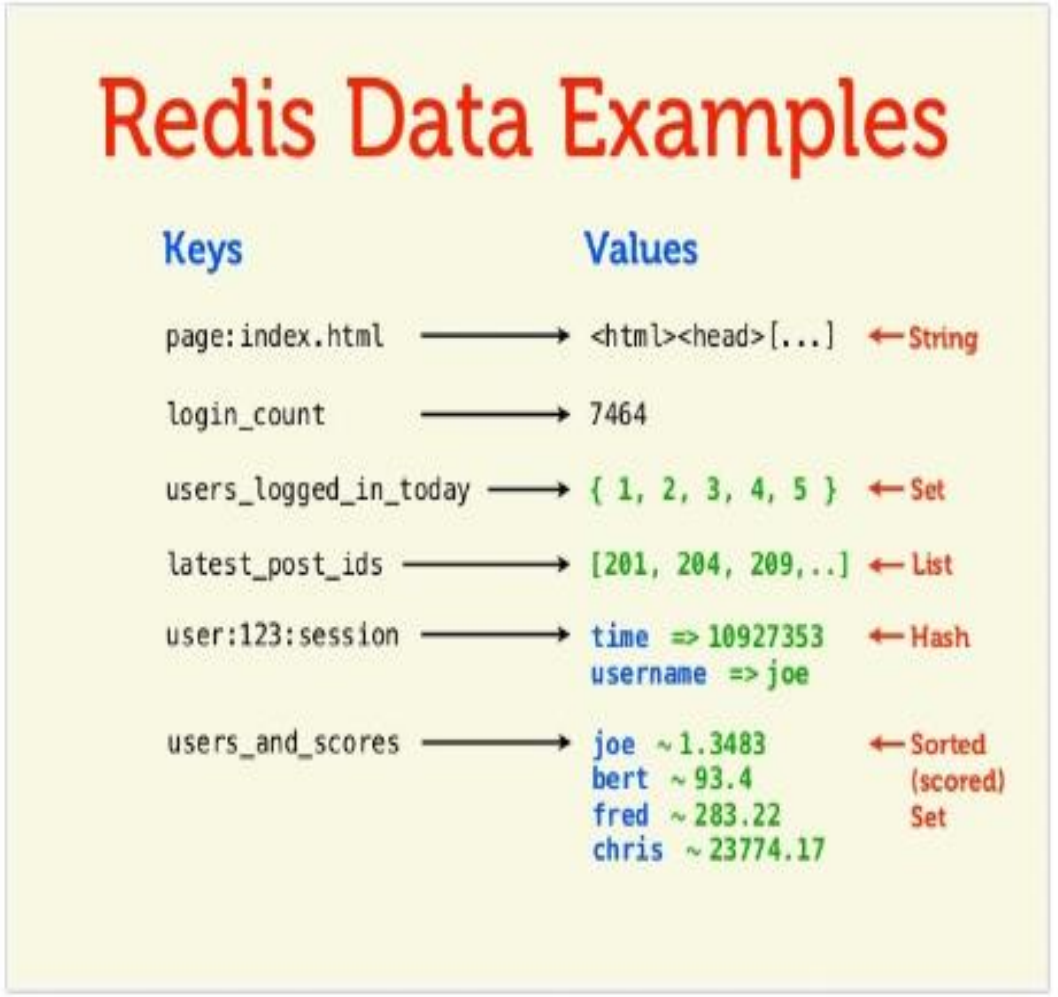


Key-Value Method

- Get(key), mengembalikan value dari key yang diberikan.
- Put(key, value), associates the value with the key.
- Multi-get(key1, key2, .., keyN), mengembalikan list dari value-value yang dihubungkan oleh multiple key.
- Delete(key), membuang entri data untuk sebuah key dari sebuah data store.

Key Value Store

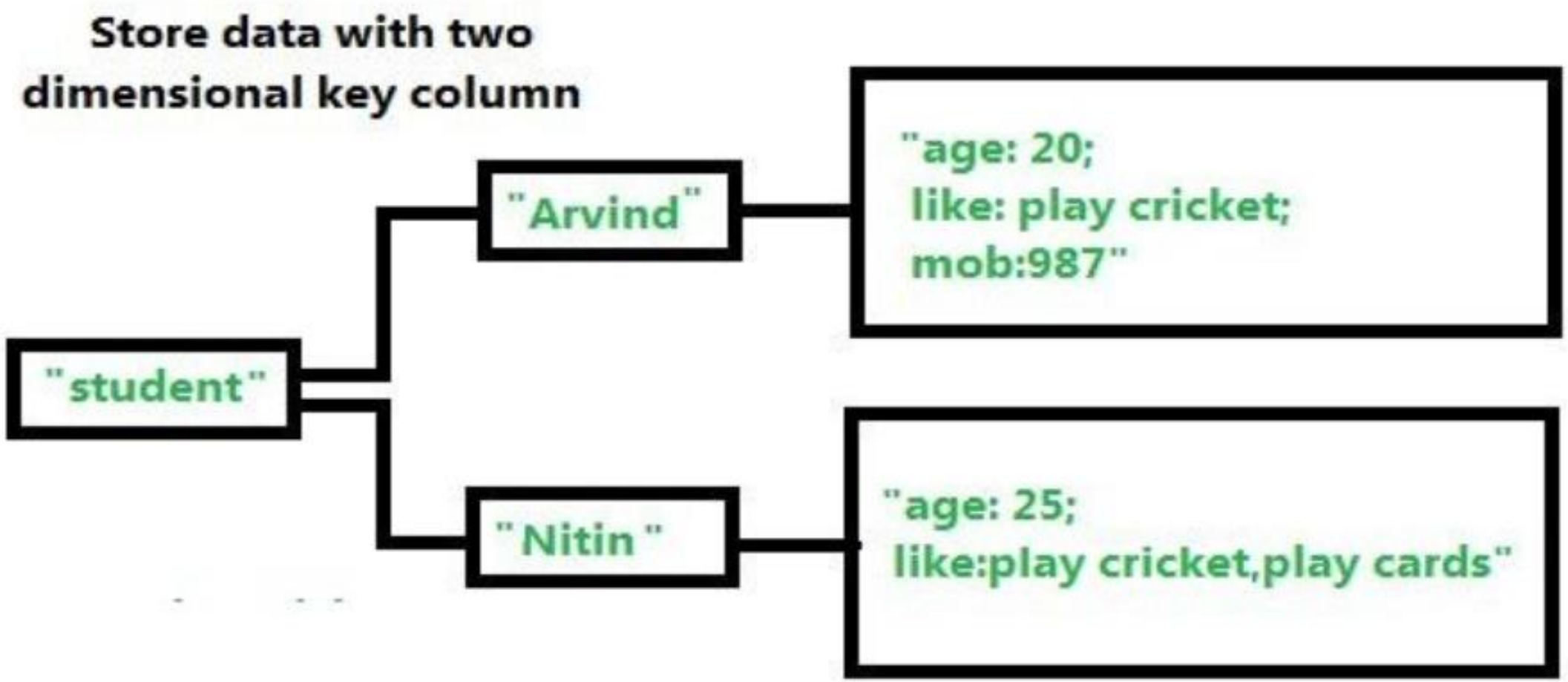




No-SQL :: Column Store

- Dikenal juga sebagai : wide column stores (sebuah key dapat di perluas menyimpan data bentuk kolum dengan banyak dimensi)
- Pendekatan penyimpanan data table sebagai bagian dari column data ketimbang sebagai row data
- Dikatakan sebagai inverse dari bentuk standard database (RDBMS)
- Contoh : Hbase, Big Table , HyperTable

Contoh: Column Store 2 dimensi



Perbandingan: RDBMS vs Column Store

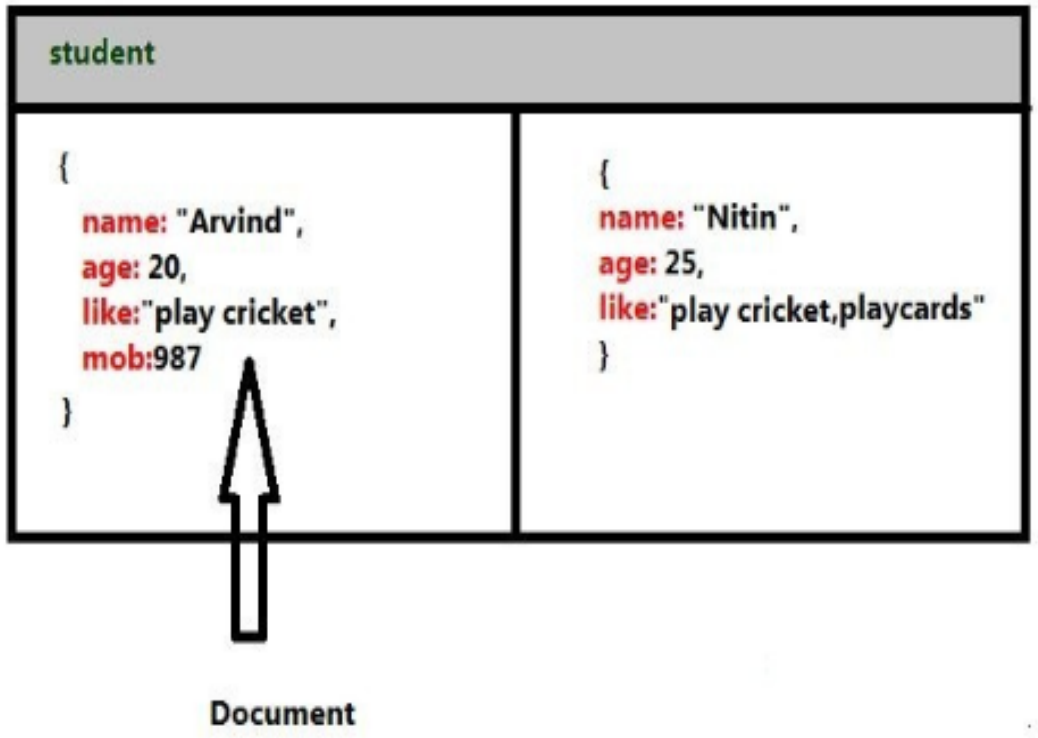
City	Pincode	Strength	Project
Noida	201301	250	20
Cluj	400606	200	15
Timisoara	300011	150	10
Fairfax	VA 22033	100	5

```
{
3PillarNoida: {
city: Noida
pincode: 201301
},
details: {
strength: 250
projects: 20
}
}
{
3PillarCluj: {
address: {
city: Cluj
pincode: 400606
},
details: {
strength: 200
projects: 15
}
},
{
3PillarTimisoara: {
address: {
city: Timisoara
pincode: 300011
},
details: {
strength: 150
projects: 10
}
}
{
3PillarFairfax : {
address: {
city: Fairfax
pincode: VA 22033
},
details: {
strength: 100
projects: 5
}
}
```

Document Database

- Pengembangan dari Key – Value Store yang disimpan dalam bentuk sebuah dokumen yang lebih komplek
- Setiap dokumen memiliki unik key (unique key) yang akan digunakan sebagai kunci untuk mengambil dokumen
- Database di desain : penyimpanan, pengambilan dan pengelolaan dengan format document-oriented information dikenal dengan semi structured data
- Memiliki struktur data berformat : XML , JSON (JavaScript On Notation), BSON (BinaryScript Object Notation)
- Contoh : MongoDB, CouchDB

Contoh : Data Document Database



{ officeName:"3Pillar Noida",

{ Street: "B-25, City:"Noida", State:"UP", Pincode:"201301"}

}

{officeName:"3Pillar Timisoara",

{Boulevard:"Coriolan Brediceanu No. 10", Block:"B, Ist Floor", City: "Timisoara", Pincode: 300011"}

}

{officeName:"3Pillar Cluj",

{Latitude:"40.748328", Longitude:"-73.985560"}

}

Pemilihan Database NoSQL

Datamodel	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value store	High	High	High	None	Variable (None)
Column Store	High	High	Moderate	Low	Minimal
Document Store	High	Variable (High)	High	Low	Variable (Low)
Graph Database	Variable	Variable	High	High	Graph Theory

NoSQL Database Vendor

- Key-value

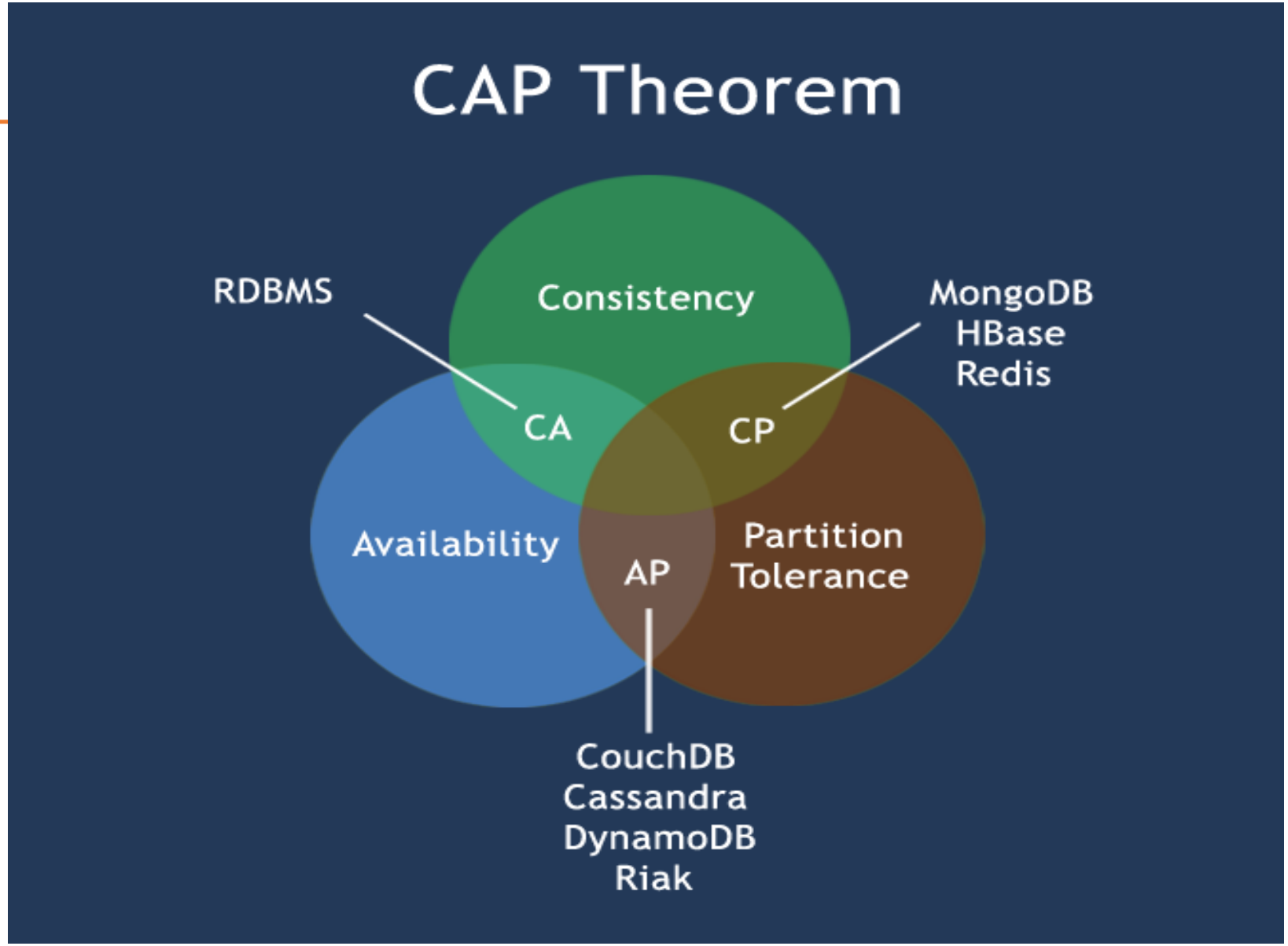
- Graph database

- Document-oriented

- Column Store

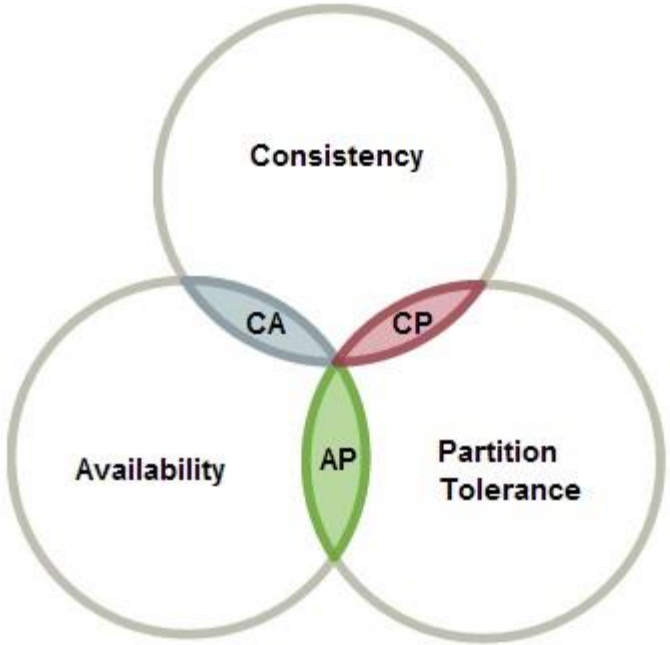
NoSQL Achitecture

CAP Theorem

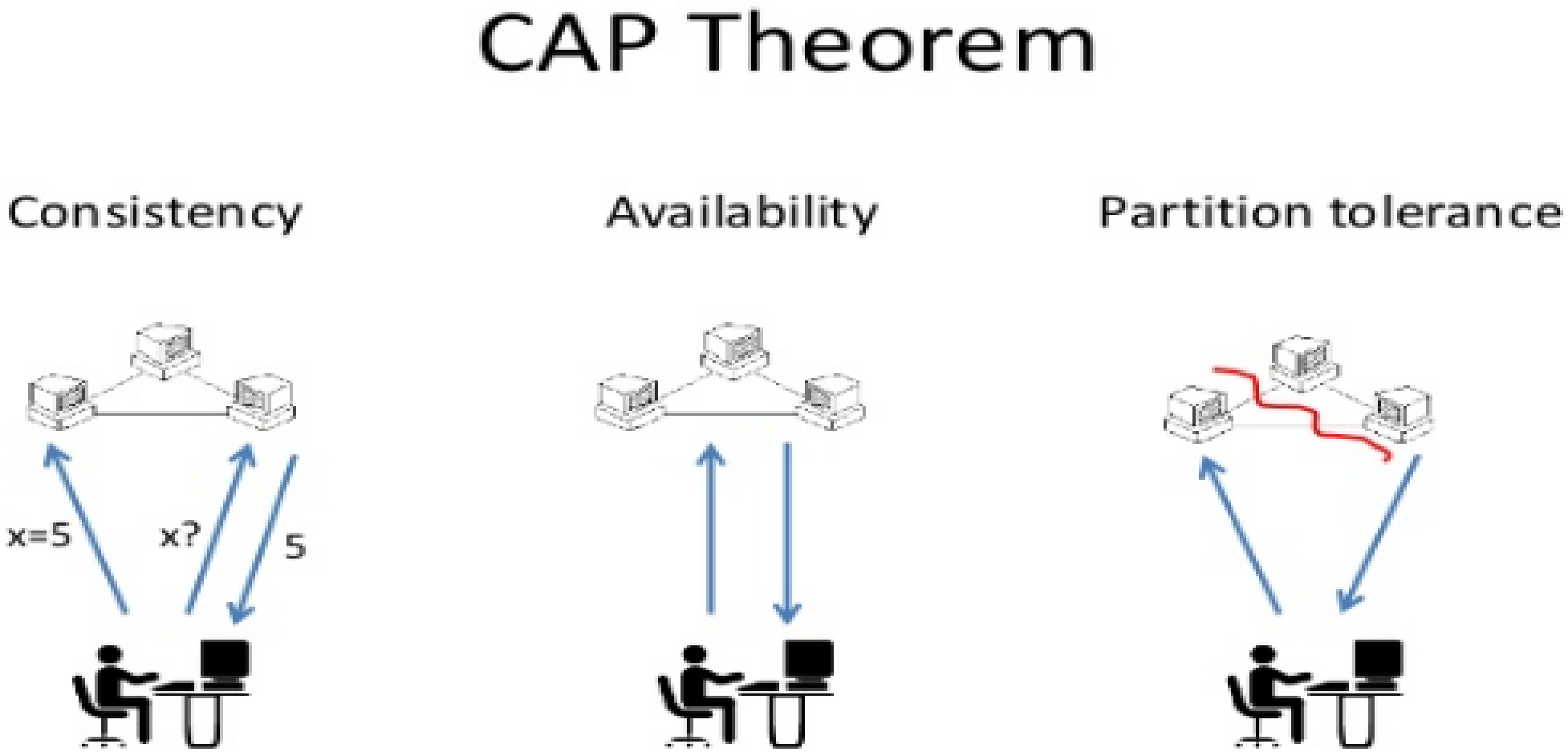

CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
Partition
Tolerance
AP
CouchDB
Cassandra
DynamoDB
Riak

CAP Theorem : theory computer science

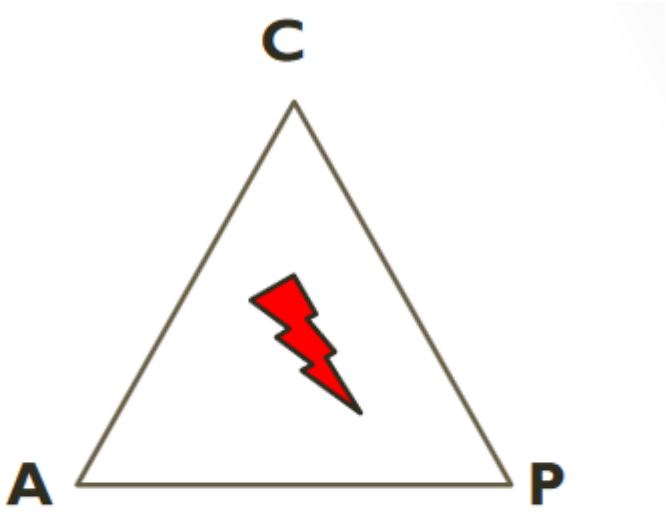
- Juga dikenal dengan nama : Brewer's Theorem (Eric Brewer 1998)
- **C**onsistency | **A**vailability | **P**artition Tolerance
- Teori digunakan sebagai tool untuk membuat perancangan sistem agar "aware" akan trade-off saat merancang networked shared-data system.
- Keterpenuhan atas 3 property CAP adalah tidak mungkin, hanya 2 yang mungkin terpenuhi : CA , CP atau AP



CAP Theorem Ilustration



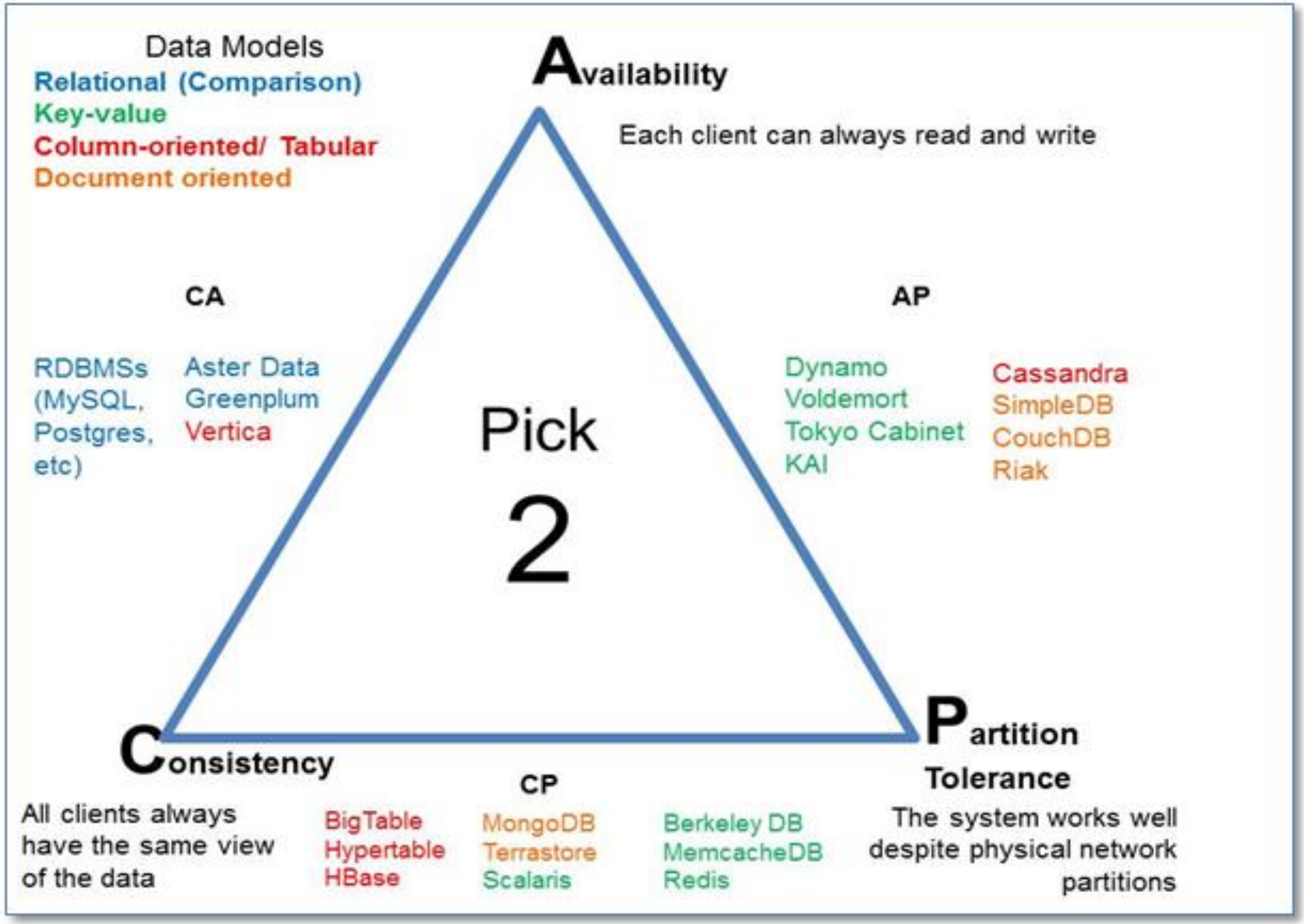
Theory of NoSQL : CAP - MongoDB

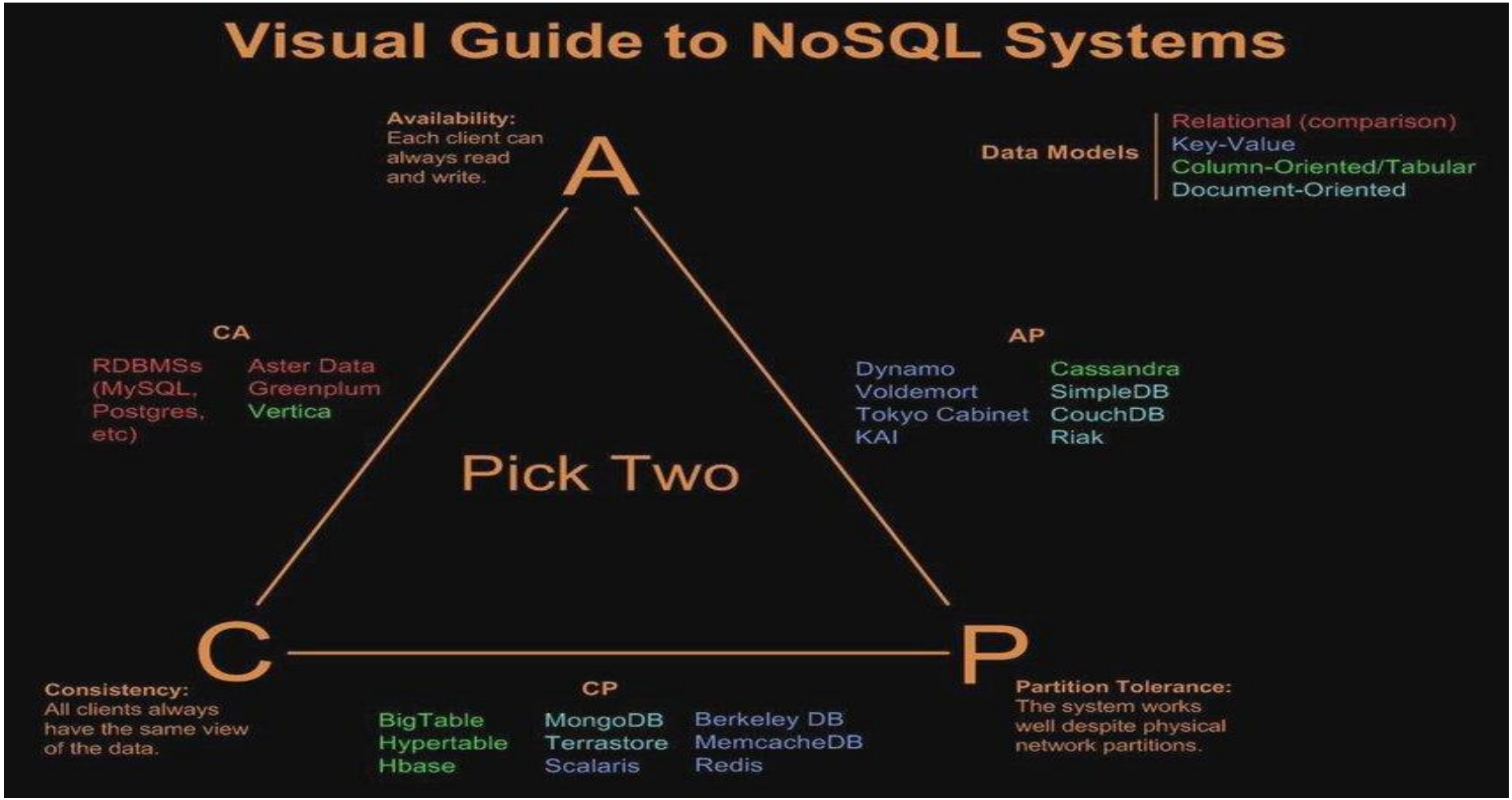


CAP Theorem:
satisfying all three at the same time is impossible

- Given:
  - Many Nodes
  - Nodes contains replicas of partitions of the data
- **Consistency**
  - **All replicas contain the same version of data**
  - **Client Always has the same view of the data (no matter what node)**
- **Availability**
  - **System remains operations on falling nodes**
  - **All client can always read and write**
- **Partition Tolerance**
  - **Multiple entry points**
  - **System remain operational on system split (communication malfunction)**
  - **System works well across physical network partitions**

CAP Theorem & Database




Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.